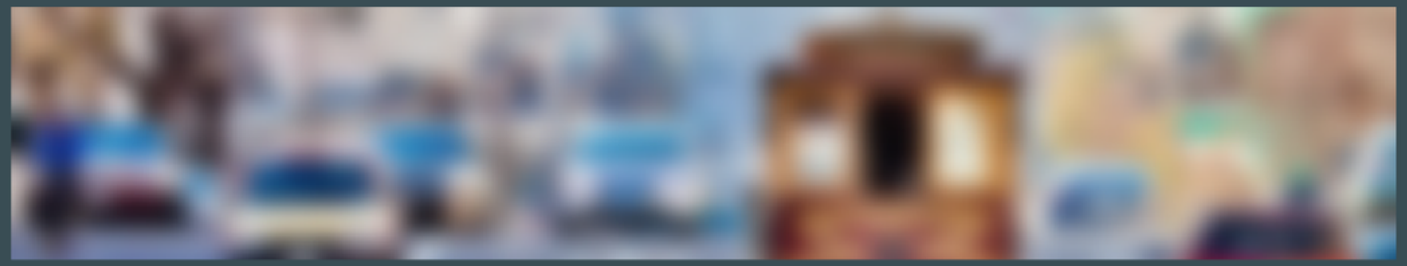


A New Model for Image Distribution

dockercon 15

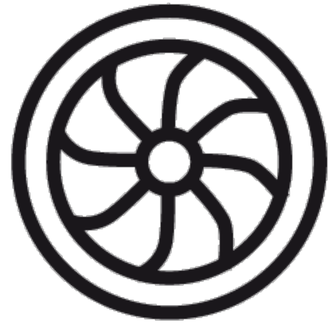
SF JUNE 22-23

Stephen Day
Distribution, Tech Lead
Docker, Inc.
stephen@docker.com
@stevvooe
github.com/stevvooe

Overview

- Why does this matter?
- History
- Docker Registry API V2
- Implementation
- The Future

What is Docker?



dockercon 15

SF JUNE 22-23

What is an Image?

dockercon 15

SF JUNE 22-23

What is an Image?

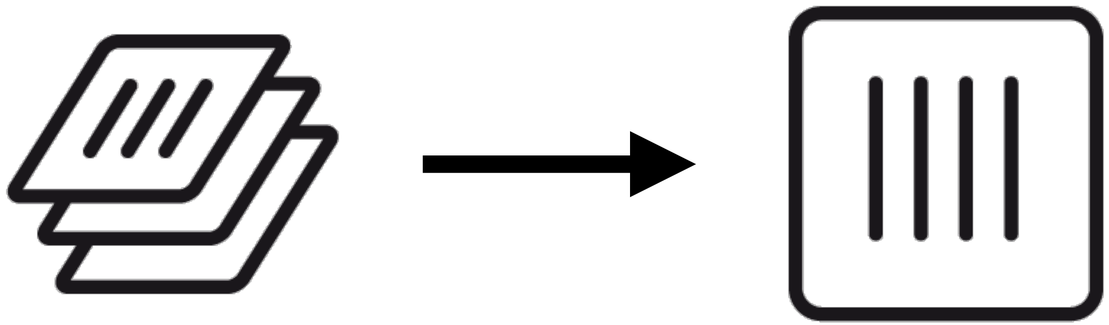
- Identified by a name
  - `ubuntu`
  - `redis`
  - `stevvooe/myapp`

- `docker run ubuntu`
  - Runs a container, created from image `ubuntu`

What is an Image?

A runnable component with a filesystem

- Containers, the runtime of docker, are created from images
- Filesystem made up with “layers”
  - Just tar files
  - Layers can be shared between images
- Includes a description organizing layers into an image



What is the Docker Registry?

dockercon 15

SF JUNE 22-23

What is the Docker Registry?

- A central place to store and distribute docker images
- Stores the layers and the description of how they make up an image
- Implements a common API agreed upon by Docker clients

What is the Docker Registry?

A central place to store and distribute docker images

- Several Implementations
  - A simple web server to make images available
  - A complete web application
  - Services
    - Docker Hub
    - Docker Trusted Registry
- Documentation: https://docs.docker.com/registry/

History

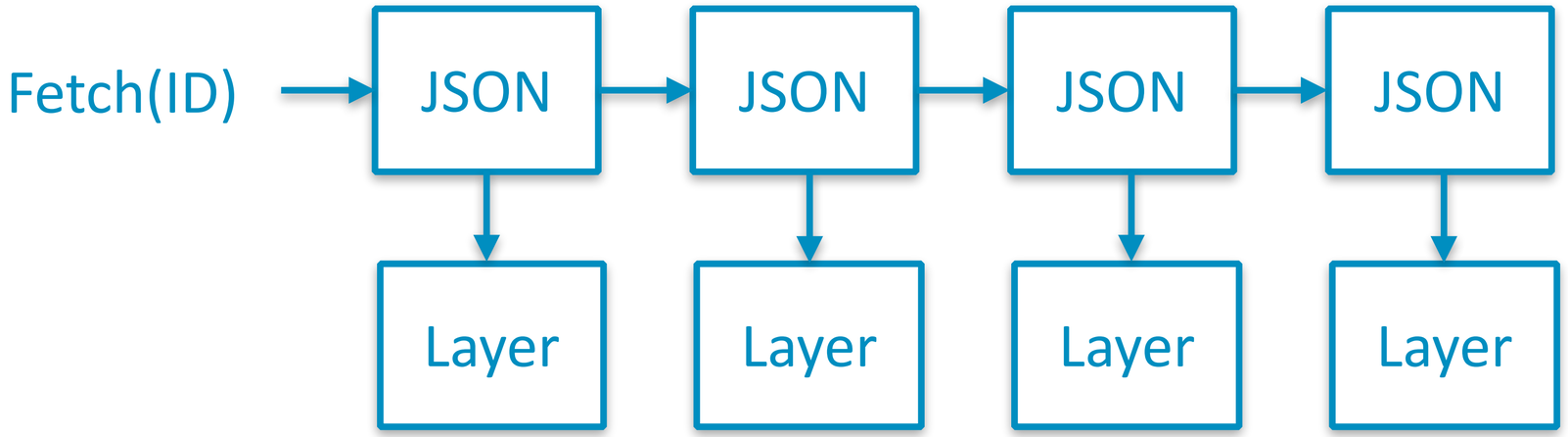
dockercon 15

SF JUNE 22-23

Docker Registry API V1: History

- Layer Oriented
- Layer IDs are randomly assigned
- JSON object corresponding to each layer referencing a parent
- Naming accomplished through tags

{



Registry API V1 URL Layout

Methods	URL
GET	/v1/_ping
GET, PUT	/v1/images/(image_id)/layer
GET, PUT	/v1/images/(image_id)/json
GET	/v1/images/(image_id)/ancestry
GET	/v1/repositories/(namespace)/(repository)/tags
GET, PUT, DELETE	/v1/repositories/(namespace)/(repository)/tags/(tag*)
DELETE	/v1/repositories/(namespace)/(repository)/
GET	/v1/search

https://docs.docker.com/reference/api/hub_registry_spec/

Docker Registry API V1: Problems

- Abstraction
  - Exposes Internals of Image to distribution mechanism
- Security
  - Image IDs must be kept secret
  - Who assigns the layer IDs?
  - Hard to audit, verify
- Performance
  - Fetch a layer, fetch the parent, fetch the parent, …

Docker Registry API V1: Problems

- Implementation in Python
  - Affected ease of deployment
  - Reduced sharing with main Docker Project
- More information:
  - https://github.com/docker/docker/issues/8093

Docker Registry API V2

dockercon 15

SF JUNE 22-23

Docker Registry API V2: Goals

- Simplicity
  - Easy to implement
  - Works with static host
- Security
  - Verifiable Images
  - Straightforward access control

Docker Registry API V2: Goals

- Distribution
  - Separate location of content from naming
- Performance
  - Remove the single track
- Implementation
  - Use Go to increase code sharing with Docker Engine

Docker Registry API V2: Content Addressable

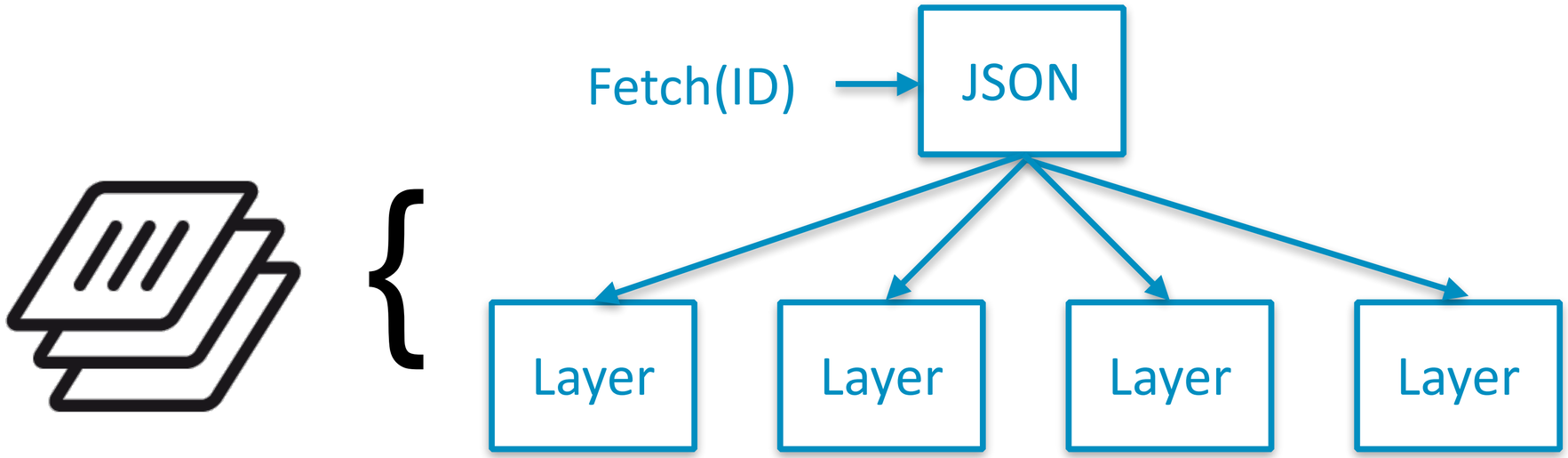
- Layers are treated as content-addressable blobs
  - Much better for security
  - Permits safe-distribution through untrusted channels
    - All data can be verified
  - De-duplication
- Improved cache-ability
- Content address is known as the “digest”

Docker Registry API V2: Digests

- Uniquely identifies content
- A cryptographically strong hash
  - Chose a name, digest, that does not conflict with other concepts (map, dict, crc, etc.)
  - Simply using *sha256(bytes)*
- Independently Verifiable
  - By agreeing on common algorithm, IDs chosen for content without coordination
- Strongly-typed with tools to parse and verify
  - http://godoc.org/github.com/docker/distribution/digest

Docker Registry API V2: Manifests

- Describes the components of an image in a single object
  - Layers can be fetched immediately, in parallel



Docker Registry API V2: Manifests

```
{
   "name": <name>,
   "tag": <tag>,
   "fsLayers": [
      {
         "blobSum": <digest>
      },
      ...
    ]
   ],
   "history": [<v1 image json>, ... ]
}
```

Docker Registry API V2: Manifest

- Content-addressable:
  - `docker pull ubuntu@sha256:812699139434 2c2775a9ba4a843869112da8156037451fc424454db43c25d8b0`
- Leverages Merkle DAG
  - Because the digests of the layers are in the manifest, if any bit in the layer changes, the digest of the manifest changes
  - Similar to git, ipfs, camlistore and a host of other projects
- Tags are in the manifest
  - This will going away

Docker Registry API V2: Repositories

- All content is now part of a named repository
  - Image IDs are no longer a secret
  - Simplified authorization model
    - repository + operation (push, pull)
  - Clients must “prove” content is available to another repository by providing it
- Opened up namespace to allow more than two components
  - No reason to have registry enforce “<user>/<image>”
  - API “reversed” to make static layout easier

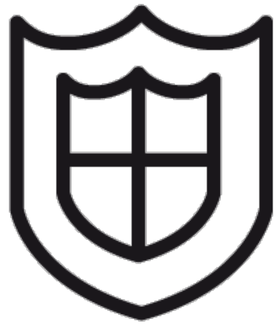
Registry API V2 URL Layout

Methods	URL
GET	/v2/
GET	/v2/<name>/tags/list
GET, PUT, DELETE	/v2/<name>/manifests/<reference>
GET	/v2/<name>/blobs/<digest>
POST	/v2/<name>/blobs/uploads/
GET, PUT, PATCH, DELETE	/v2/<name>/blobs/uploads/<uuid>

https://docs.docker.com/registry/spec/api/

Docker Registry API V2: Design

- Shared-nothing
  - “Backend” ties a cluster of registries together
  - Allows scaling by adding instances
  - Performance limited by backend
    - Make backend faster, registry gets faster
- Pull-optimized
  - Most important factor when distributing software
  - May hurt certain use cases
- Resumable Pull and Push (specified but not implemented)
  - Resumable pull already available with http Range requests
  - Two-step upload start for resumable push
  - Built into the protocol for future support
- A living specification
  - Meant to be used and modified
  - Always backwards compatible



Docker Registry API V2: Differences with V1

- Content addresses (digests) are primary identifier
- Unrolled image description model
- Multi-step upload
  - Provides flexibility in failure modes
  - Options for future alternative upload location (redirects)
- No Search API
  - In V1, this API does everything
  - Replacing with something better
- No explicit tagging API
  - This will change: https://github.com/docker/distribution/pull/173

Docker Registry 2.0

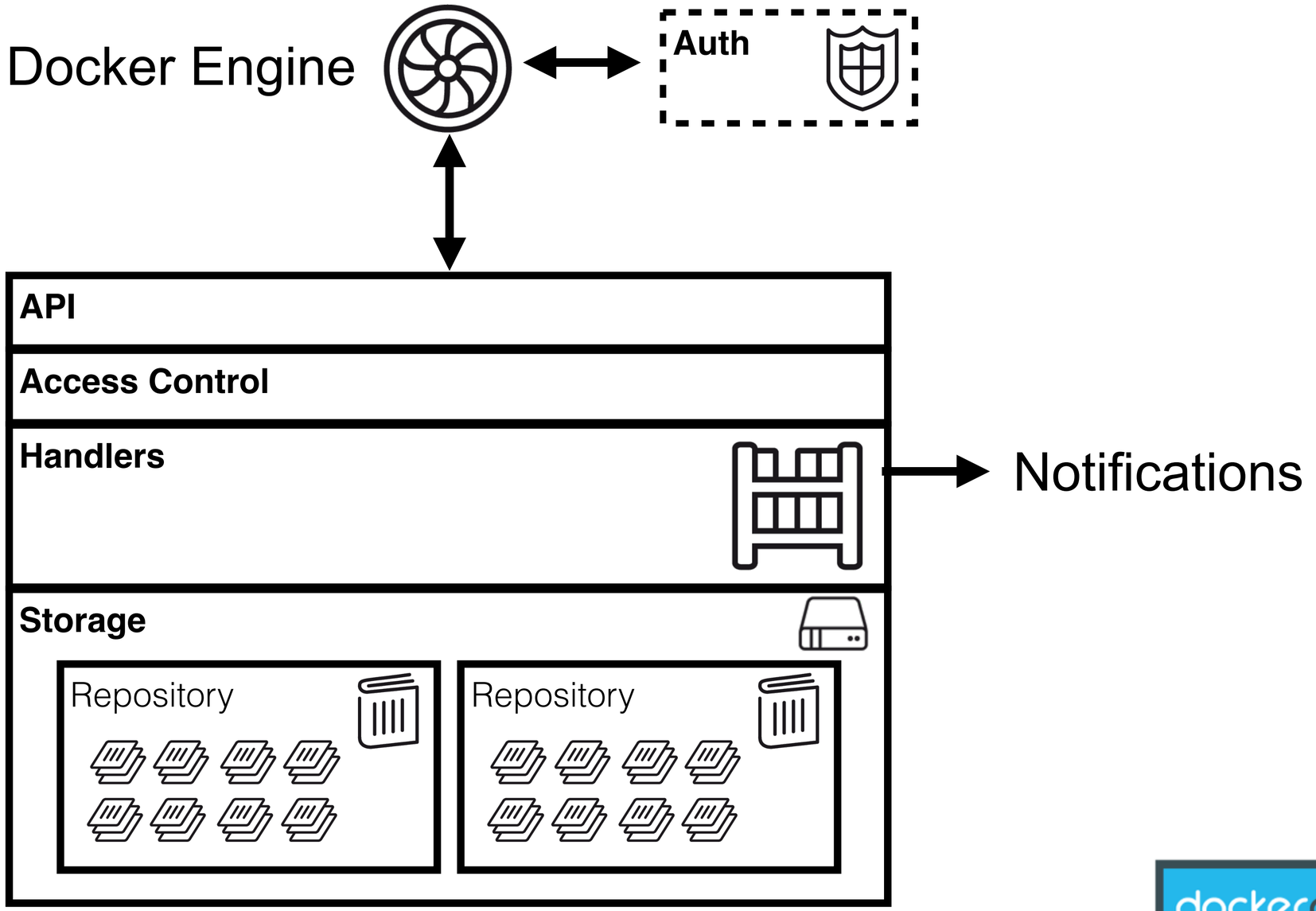
dockercon 15

SF JUNE 22-23

“[A registry] should be neither seen nor heard. ”

–Earl Milford

Docker Registry 2.0: Architecture



Docker Registry 2.0: An Ingredient

- Move away from monolithic architecture
- Narrower scope
  - Distribute content
- Extensible
  - Authentication
  - Index
  - Ponies
- Strong core
  - Docker Hub
  - Docker Trusted Registry

Docker Registry 2.0

- Full support released with Docker 1.6
  - Minimal bugs
  - Most problems are common to version upgrades
    - Header required to declare support for 2.0 API
- Validated most concepts in 1.3, 1.4 with V2 preview
  - Much faster pull performance
  - You've probably already used it with Docker Hub
- There are some edge cases
  - push-heavy workflows
  - disk IO when verifying large images
  - We are mitigating these

Docker Registry 2.0: Should you use it?

- Are you on Docker 1.6+?
  - Yes.
    - Evaluate it
    - Test it
    - Break it (and file bugs https://github.com/docker/distribution/issues)
    - Deploy it
- Are you on Docker <1.6?
  - Are you entrenched in v1?
    - Perhaps, hold off
  - Run dual stack v1, v2
    - Not recommended

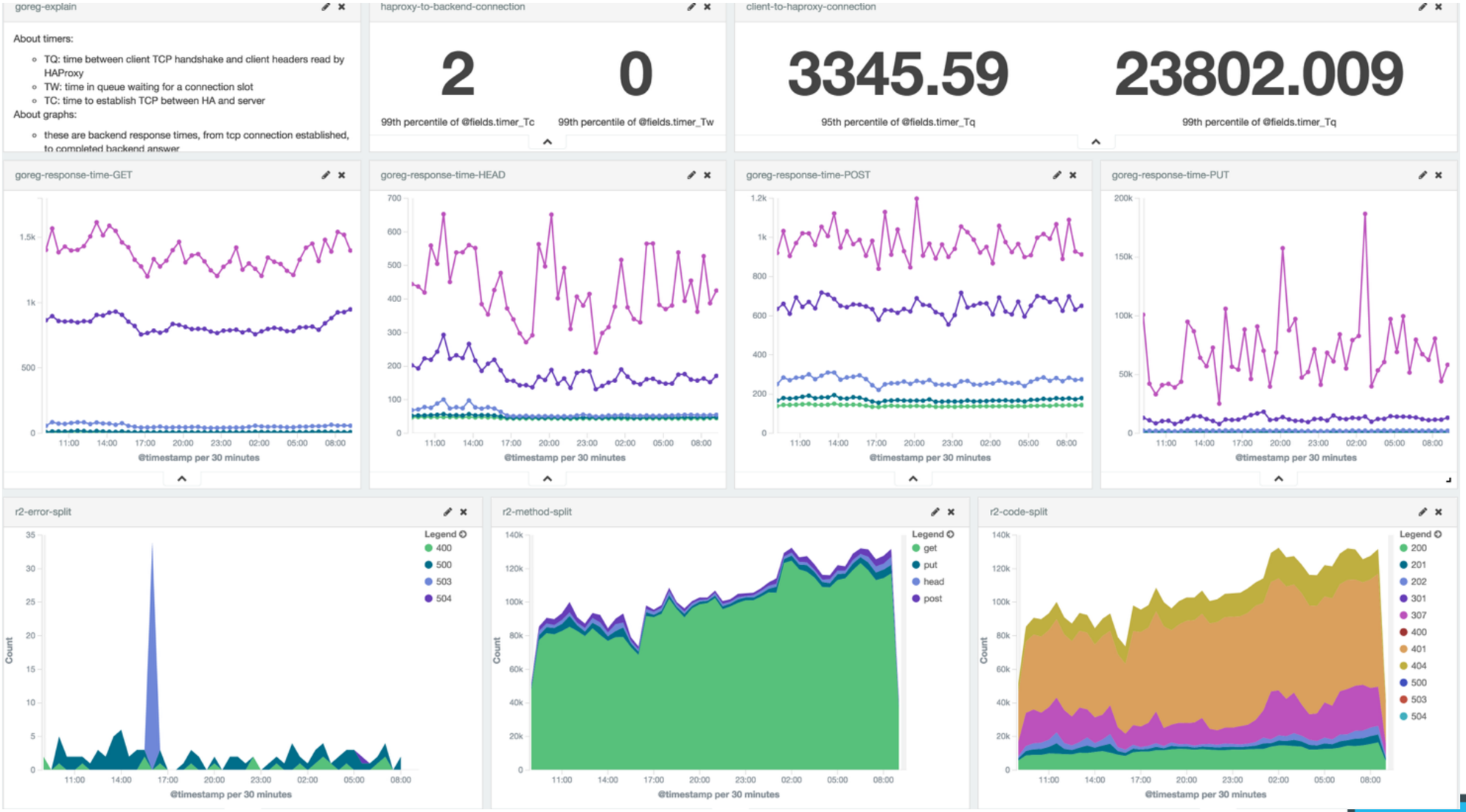
Docker Registry 2.0: Deploying

- Internal deployments
  - Use the filesystem driver — it is *really* fast
  - Backup with rsync
- Scale storage
  - Use S3 driver
    - Make sure you are “close” since round trip times can have an effect
- Scale Reads
  - Use round robin DNS
    - Do not use this for HA
  - Rsync to followers on read-only filesystem
  - Add machines to taste
- https://docs.docker.com/registry/deploying/

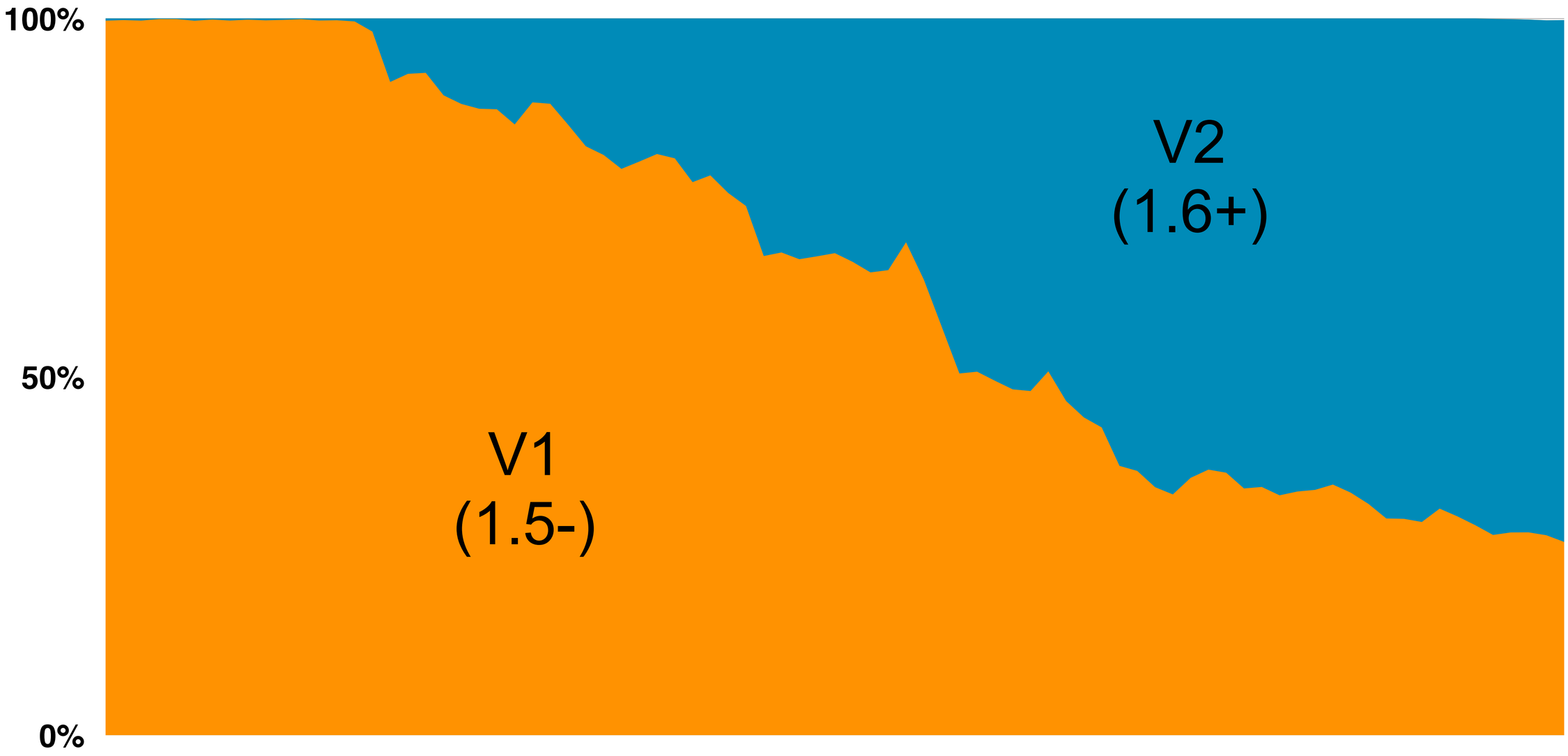
Docker Registry 2.0: Docker Hub

- Running the Hub
  - S3 backend
    - Having some trouble with round trips to s3 :(
  - Decent performance with very little caching
    - A lot of low hanging fruit left to tackle
- No longer intertwined with Docker Hub services
  - Independent Authentication Service
  - Heightened Availability

Monitoring culture



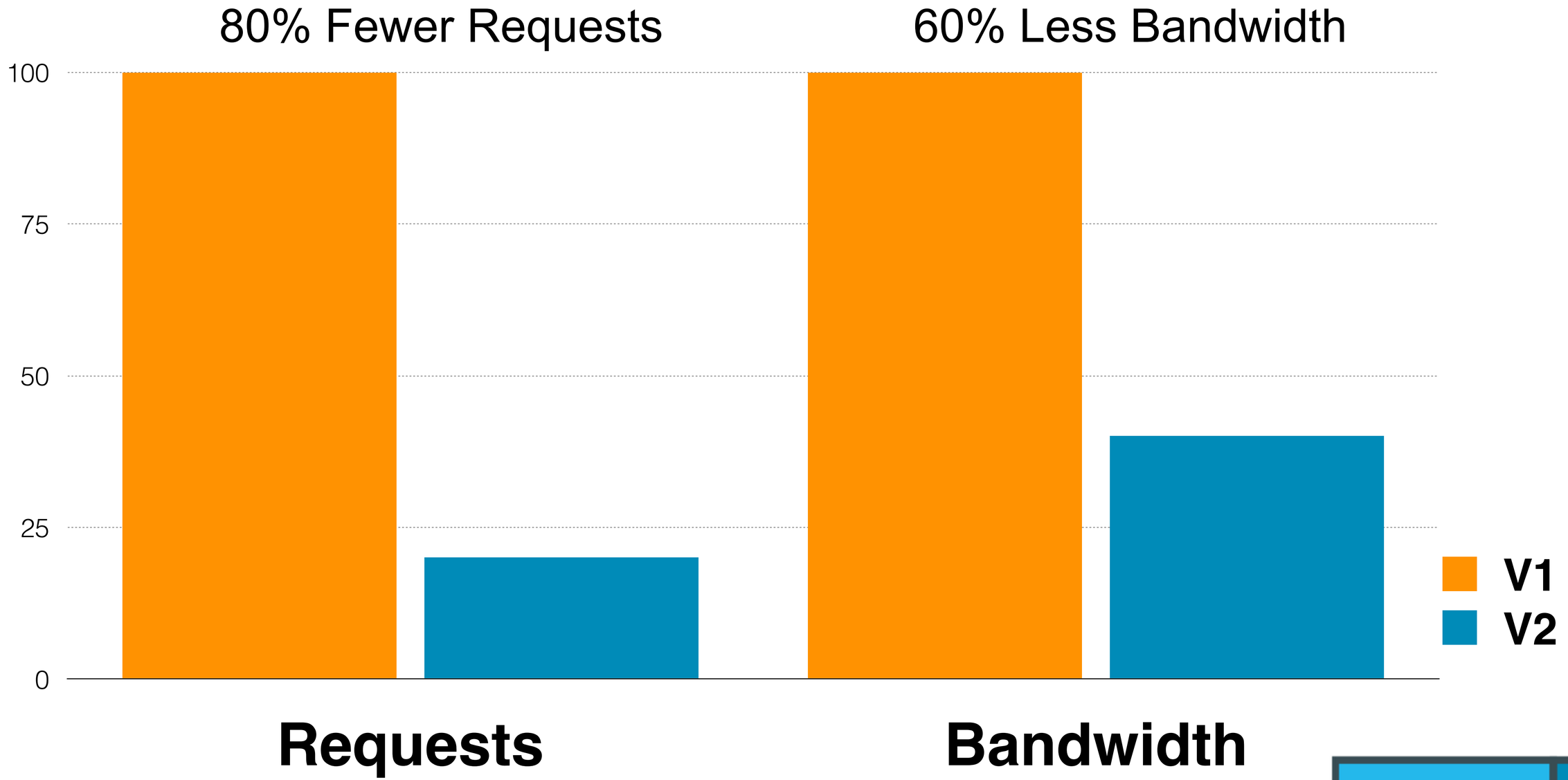
Docker Hub Adoption



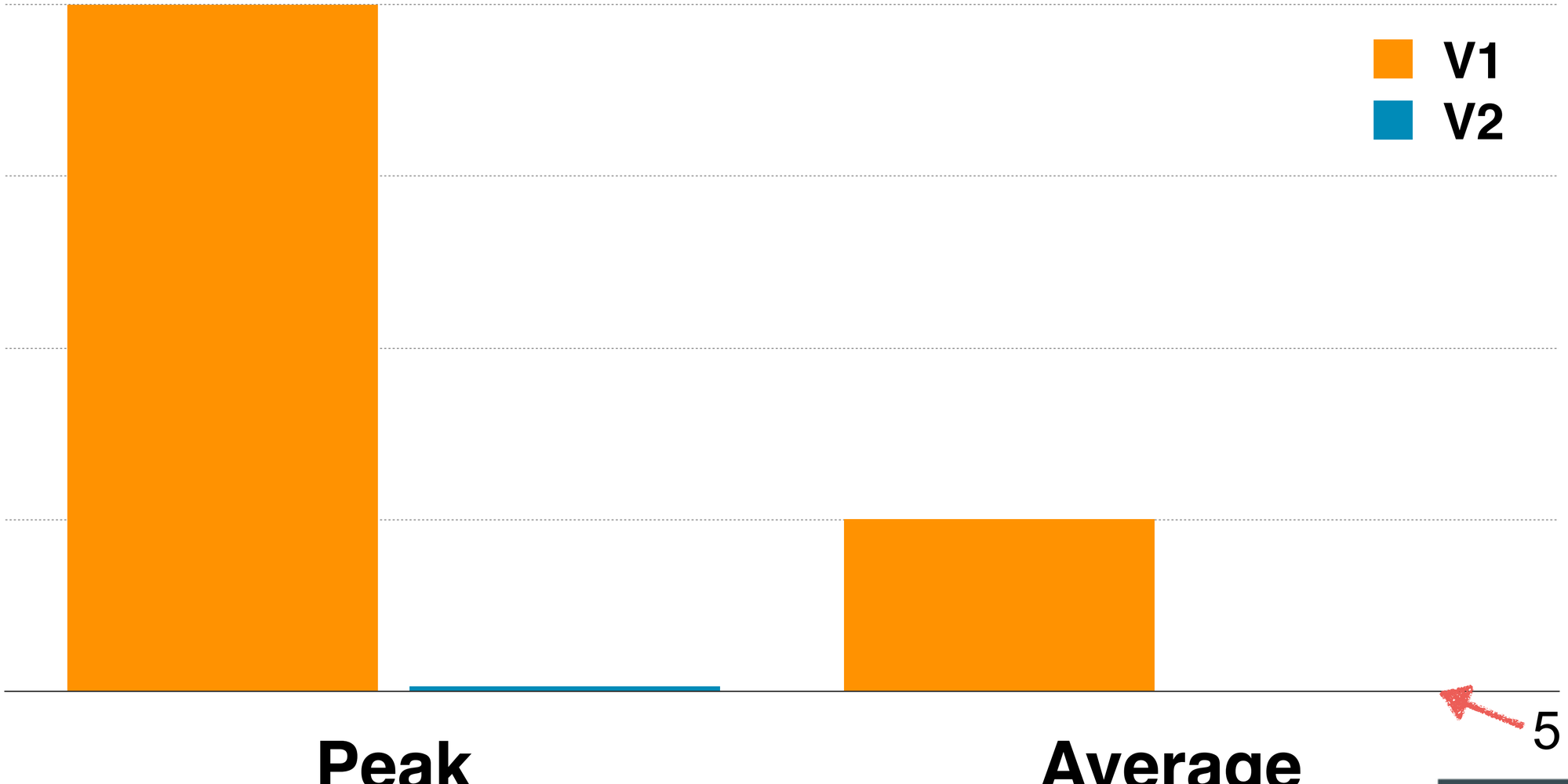
Docker Hub Adoption

- Overall usage increasing
- A V2 world and growing

V1/V2 Protocol Overall Comparison

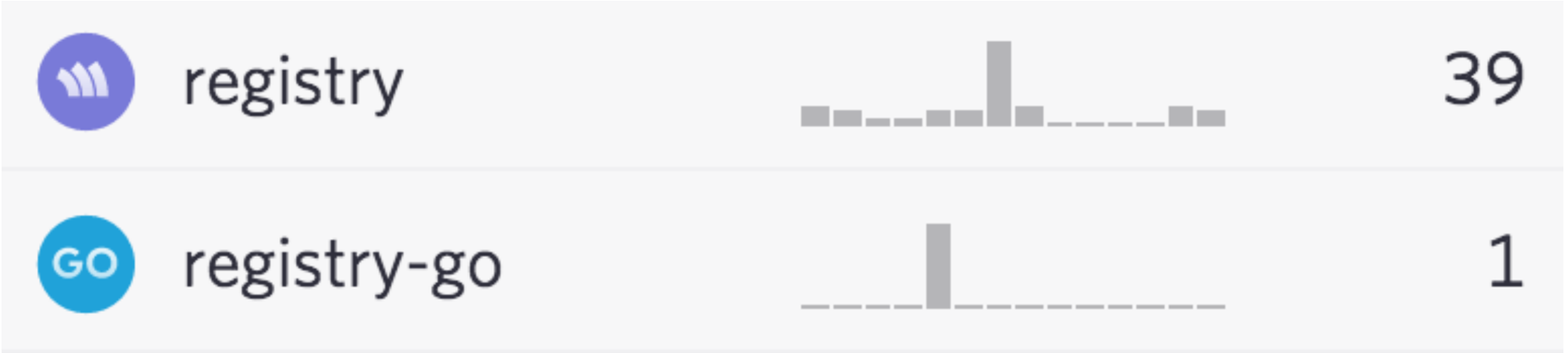


V1/V2 Protocol HTTP Errors



Exceptional Panicking

- **1** Panic in Three Months of Production
- **4000** protocol level errors per 30 minutes in V1
- **5** protocol level errors per 30 minutes in V2



Docker Registry 2.1

dockercon 15

SF JUNE 22-23

Docker Registry 2.1

- Key Changes
  - Documentation
  - Pull-through Caching
  - Soft-Deletion
  - Native Basic Auth Support
  - Stability
  - Catalog API
  - Storage Drivers
- Release coming by mid-July

Docker Distribution

dockercon 15

SF JUNE 22-23

Docker Distribution: Goals

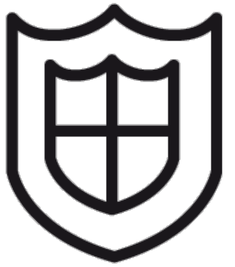
- Goals
  - Improve the state of image distribution in Docker
  - Build a solid and secure foundation
- Focus
  - Security
  - Reliability
  - Performance
- Unlock new distribution models
  - Integration with trust system (notary!)
  - Relax reliance on registries
  - Peer to Peer for large deployments

Docker Distribution: Future

- Ingredients
  - From the start, we have targeted solid packages
  - Provide Lego to build image distribution systems
- Clean up the docker daemon code base
  - Defined new APIs for working with docker content
  - Increase feature velocity
  - Generalize around strong base
- Current Manifest format is provisional
  - Still includes v1 layer JSON
  - Content-addressability + mediatypes make support new formats trivial
  - https://github.com/docker/distribution/pull/62
- Feature parity with V1 and maturity
  - Building collective operational knowledge
- Deletes and Garbage Collection
  - Diverse backend support makes this hard
  - https://github.com/docker/distribution/issues/461
  - https://github.com/docker/distribution/issues/462
- Search
  - See the goals of Distribution to see why this is interesting

- Road Map: https://github.com/docker/distribution/wiki



Thank you

Stephen Day

Google Group: distribution@dockerproject.org
GitHub: https://github.com/docker/distribution
IRC on Freenode: #docker-distribution