The Fn Project

The open source, cloud-agnostic serverless platform

Serverless Everywhere and Anywhere

Travis Reeder
Architect at Oracle
@treeder

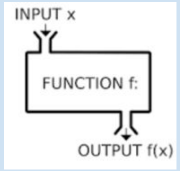
What is Serverless?

- **Serverless** is an abstraction of infrastructure and its operations including provisioning, scaling, patching, etc.
- **Serverless architecture** is when an app is built entirely on serverless components (compute, storage, networking)
- **FaaS** is the compute component in a serverless architecture

Functions-as-a-Service

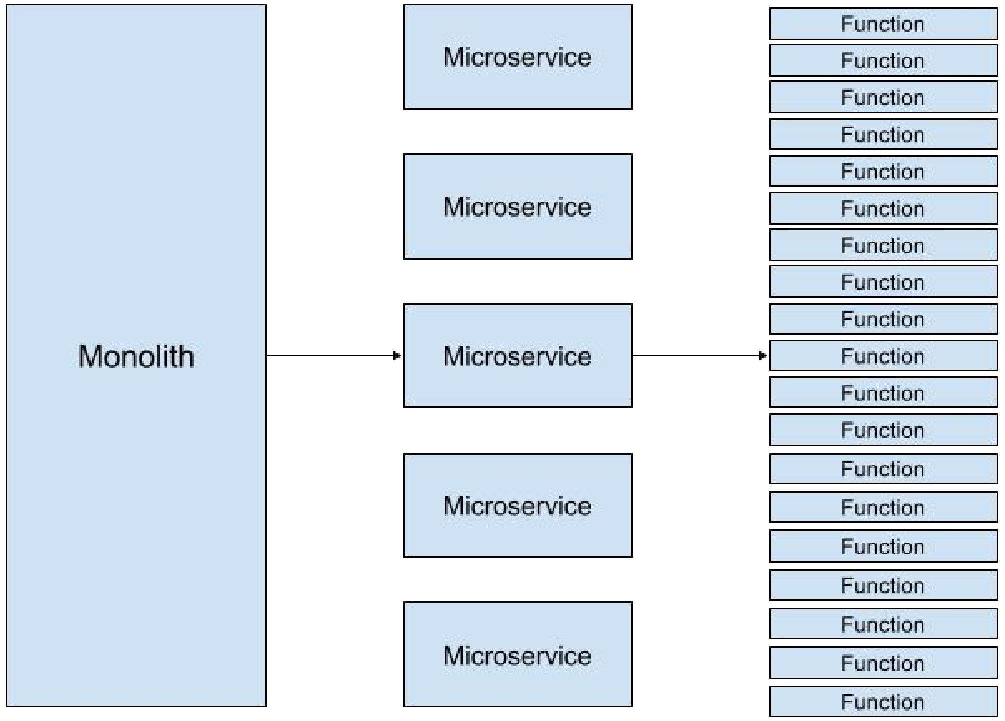
In mathematics, a **function** is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output.

Function (mathematics) - Wikipedia
https://en.wikipedia.org/wiki/Function_(mathematics)



- **Functions** are small bits of code that take some input, do one simple thing, then produce output
- **As a service** means these functions run on a system that takes care of provisioning, scaling, patching, maintaining, etc. Each function scales independently.

What about Microservices?



Why Serverless for developers?

- **Easier:** Just think about your code, not infrastructure
- **Faster:** Deploy faster, iterate faster, innovate faster
- **Cheaper:** Only pay for what you use to the 100ms (never idle)
- **Powerful:** Auto scaling and management

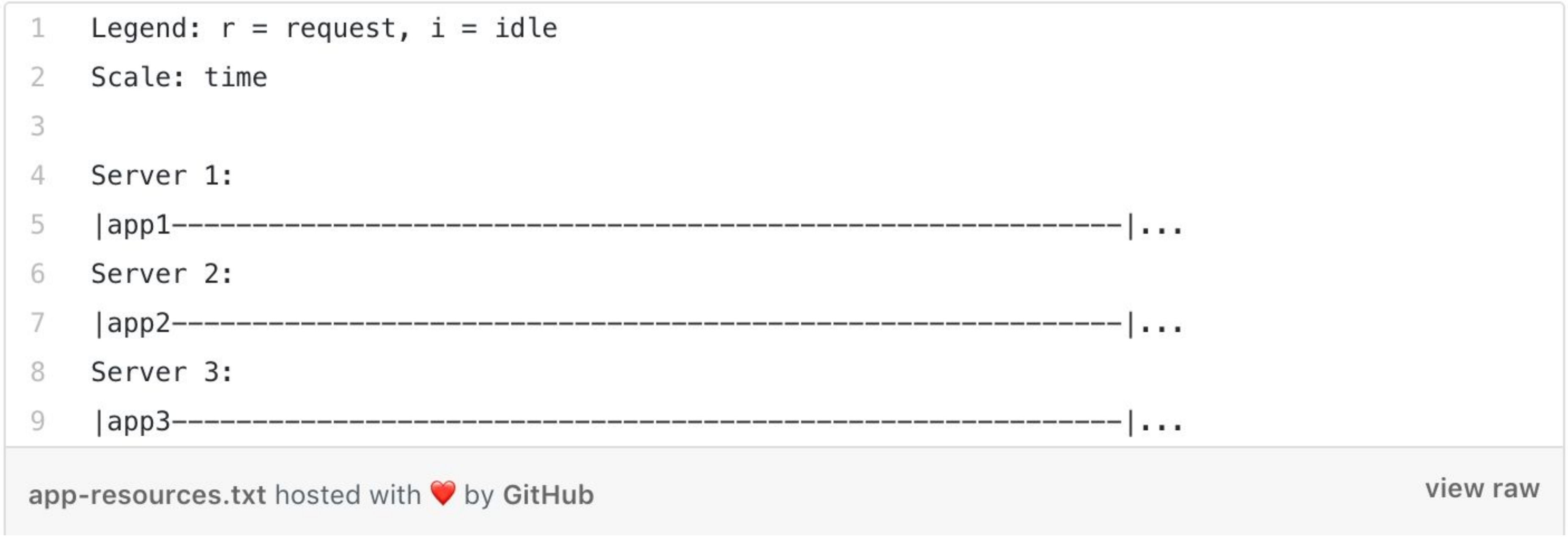
Why Serverless for a business?

- **Agility:** Devs move faster with less dependencies
- **Innovation:** Devs can quickly iterate on new ideas
- **Cost Reduction:** Pay only for execution, not for idle, and reduce ops costs.

Even on Private Cloud

- Same advantages for developers
- One system to manage and operate for all of your applications
- Optimizes hardware utilization

Typical Applications

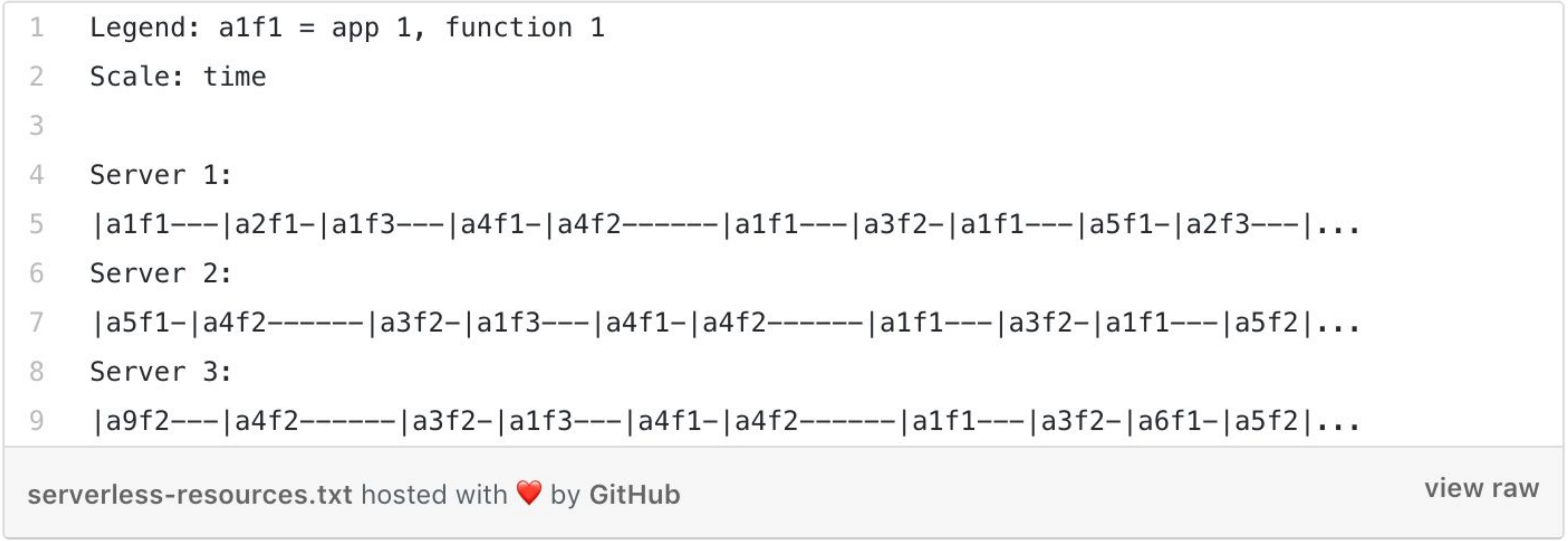


```
Legend: r = request, i = idle
Scale: time

Server 1:
|app1------------------------------------------------------------|...
Server 2:
|app2------------------------------------------------------------|...
Server 3:
|app3------------------------------------------------------------|...
```

app-resources.txt hosted with ❤️ by GitHub view raw

Serverless Applications



```
Legend: a1f1 = app 1, function 1
Scale: time

Server 1:
|a1f1---|a2f1-|a1f3---|a4f1-|a4f2------|a1f1---|a3f2-|a1f1---|a5f1-|a2f3---|...
Server 2:
|a5f1-|a4f2------|a3f2-|a1f3---|a4f1-|a4f2------|a1f1---|a3f2-|a1f1---|a5f2|...
Server 3:
|a9f2---|a4f2------|a3f2-|a1f3---|a4f1-|a4f2------|a1f1---|a3f2-|a6f1-|a5f2|...
```

serverless-resources.txt hosted with ❤️ by **GitHub** **view raw**

Introducing the Fn Project

- Open-source serverless compute platform
- Can be deployed to any cloud and on-premise
- Simple and extensible by design
- Containers are primitives
- Hot containers provide fast response times
- Active w/ ~2500 commits across 50+ contributors
- Independently governed with plans for foundation
- Apache 2.0 license

For Developers

An Fn Function

- Small chunk of code wrapped into a container image
- Input via STDIN and environment
- Output to STDOUT
- Logs to STDERR

The Fn server handles everything else.

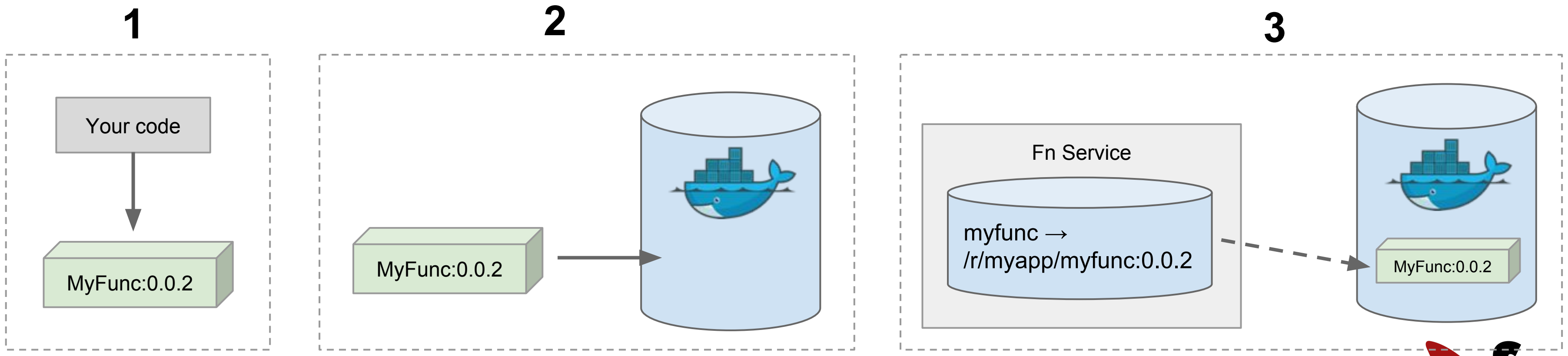
Fn CLI

- **fn init** --runtime java hello
- **fn run**
- **fn test**
- **fn deploy** --app myapp
- **fn call** myapp myfunc

  → http://localhost:8080/r/myapp/myfunc

fn deploy details

1. Builds container (multi-stage) + bumps version
2. Pushes container to registry
3. Creates/updates function route (servers lazy load images)



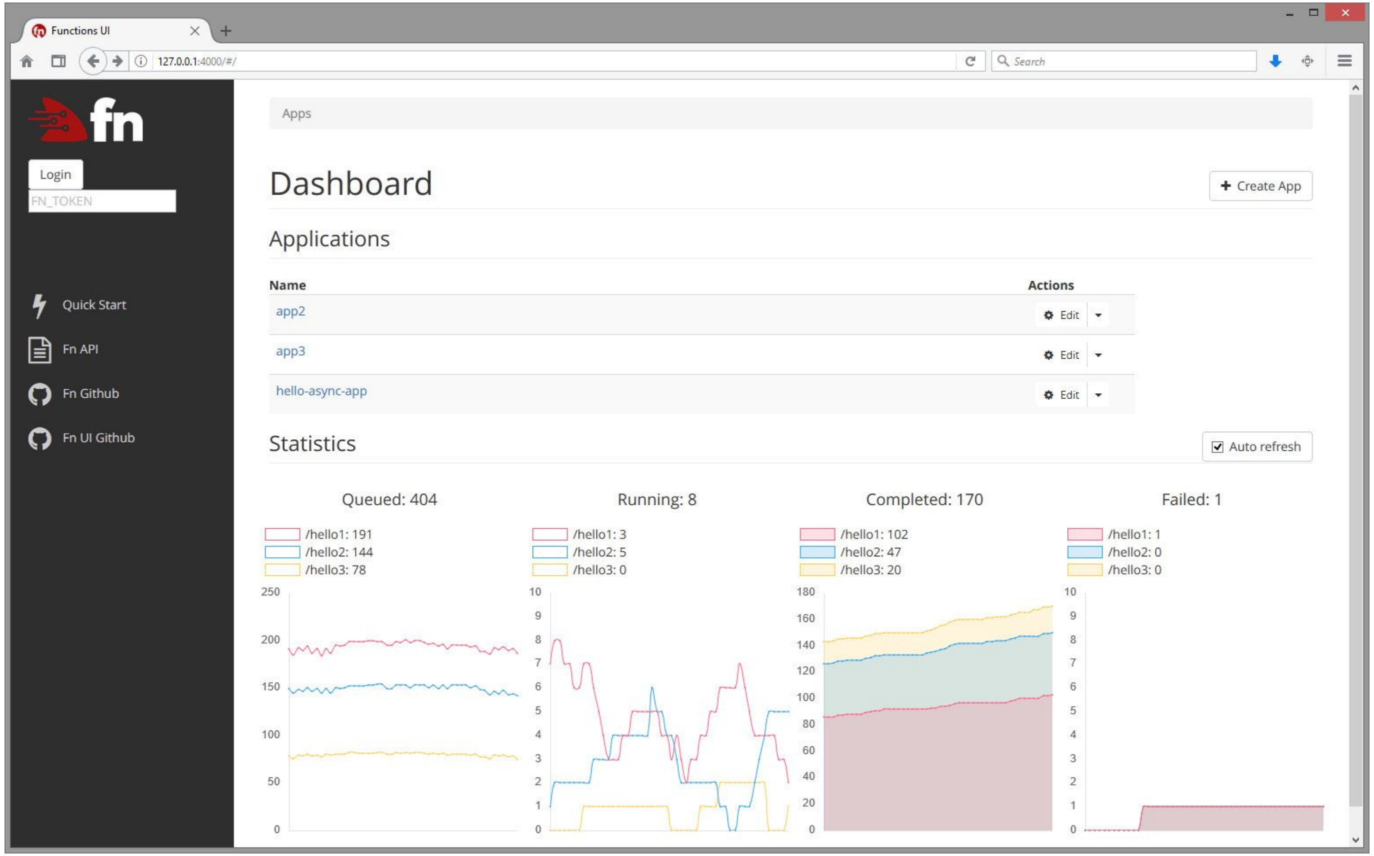
Function Development Kits (FDKs)

- Used to help with parsing input and writing output
- Familiar syntax for Lambda developers
- Simply write a `handler` function that adheres to the FDK's interface and it will parse STDIN and provide the input data to your function and deal with writing the proper output format.
- Makes it a lot easier to write hot functions

Debugging

- **fn calls list** myapp
- **fn calls get** myapp <call-id>
- **fn logs get** myapp <call-id>
- Metrics created using OpenTracing w/ initial collectors and extensions for Prometheus, ZipKin, and soon Jaeger

Fn UI



Functions UI

127.0.0.1:4000/#/

Login

FN_TOKEN

Quick Start

Fn API

Fn Github

Fn UI Github

Apps

Dashboard

+ Create App

Applications

Name	Actions
app2	Edit
app3	Edit
hello-async-app	Edit

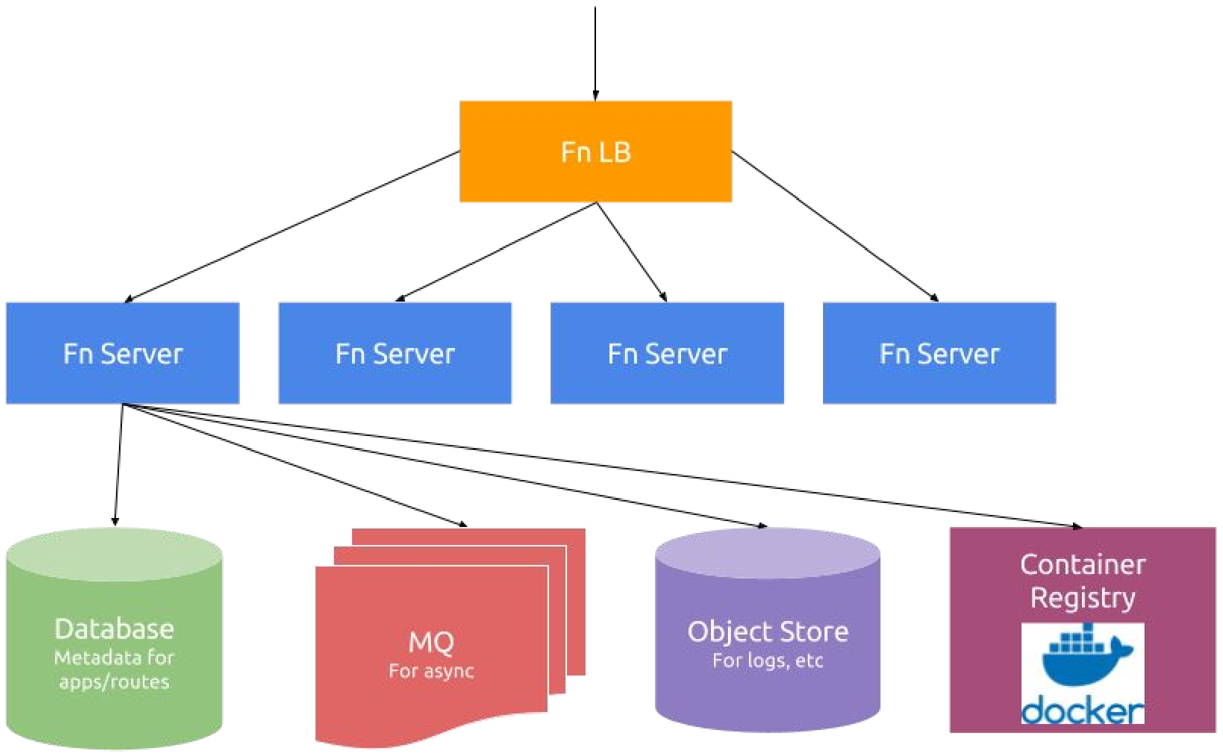
Statistics

☑ Auto refresh

Queued: 404	Running: 8	Completed: 170	Failed: 1
/hello1: 191	/hello1: 3	/hello1: 102	/hello1: 1
/hello2: 144	/hello2: 5	/hello2: 47	/hello2: 0
/hello3: 78	/hello3: 0	/hello3: 20	/hello3: 0

For Operators

Architecture



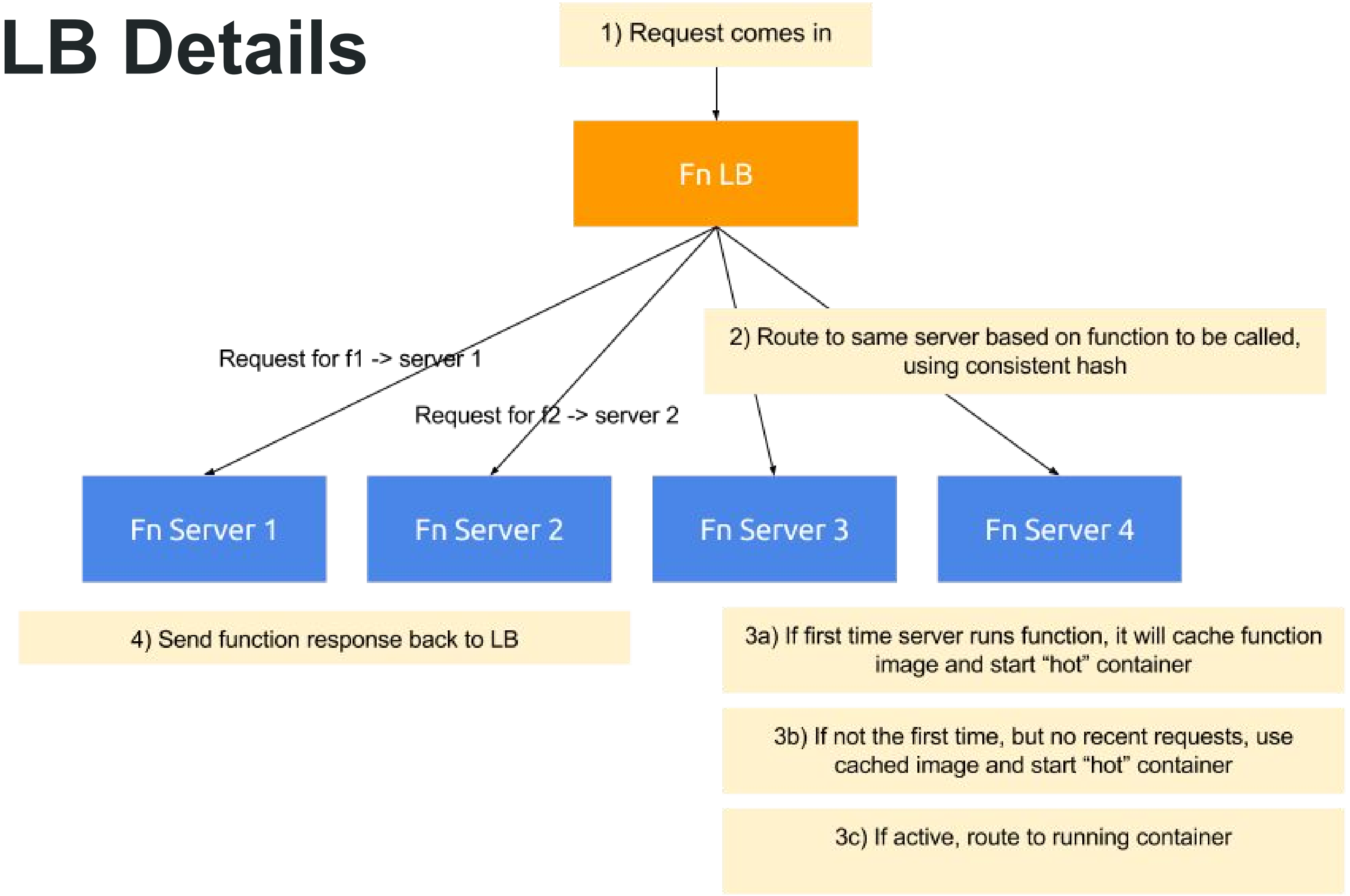
Fn Server

- Handles CRUD operations for setting up routes and functions
- API gateway -> Executes function, returns response to clients
- Queues async function calls
- Executes async function calls when capacity is available
- Written in Go, easy to extend via extension system

Fn LB

- Simple, fast load balancer that routes functions to certain nodes consistently for hot function efficiency
- Scales each function independently based on traffic to any particular function
- Can be used to scale Fn servers and infrastructure as well as it has a view of global state of all fn servers

Fn LB Details


1) Request comes in
Fn LB
Request for f1 -> server 1
Request for f2 -> server 2
2) Route to same server based on function to be called, using consistent hash
Fn Server 1
Fn Server 2
Fn Server 3
Fn Server 4
4) Send function response back to LB
3a) If first time server runs function, it will cache function image and start "hot" container
3b) If not the first time, but no recent requests, use cached image and start "hot" container
3c) If active, route to running container

Supporting Services

- DB, MQ, blob store are all pluggable modules that are thin wrappers around their respective drivers.
  - DB: MySQL, sqlite3, Postgres
  - Queue: Redis, Kafka
  - Registry: Any Docker v2-compliant, even private
- Metrics/Monitoring
  - OpenTracing API for metrics
  - Prometheus support, pluggable backends
  - Logging via syslog

DEMO!

Thank you!

Travis Reeder
Architect at Oracle
@treeder

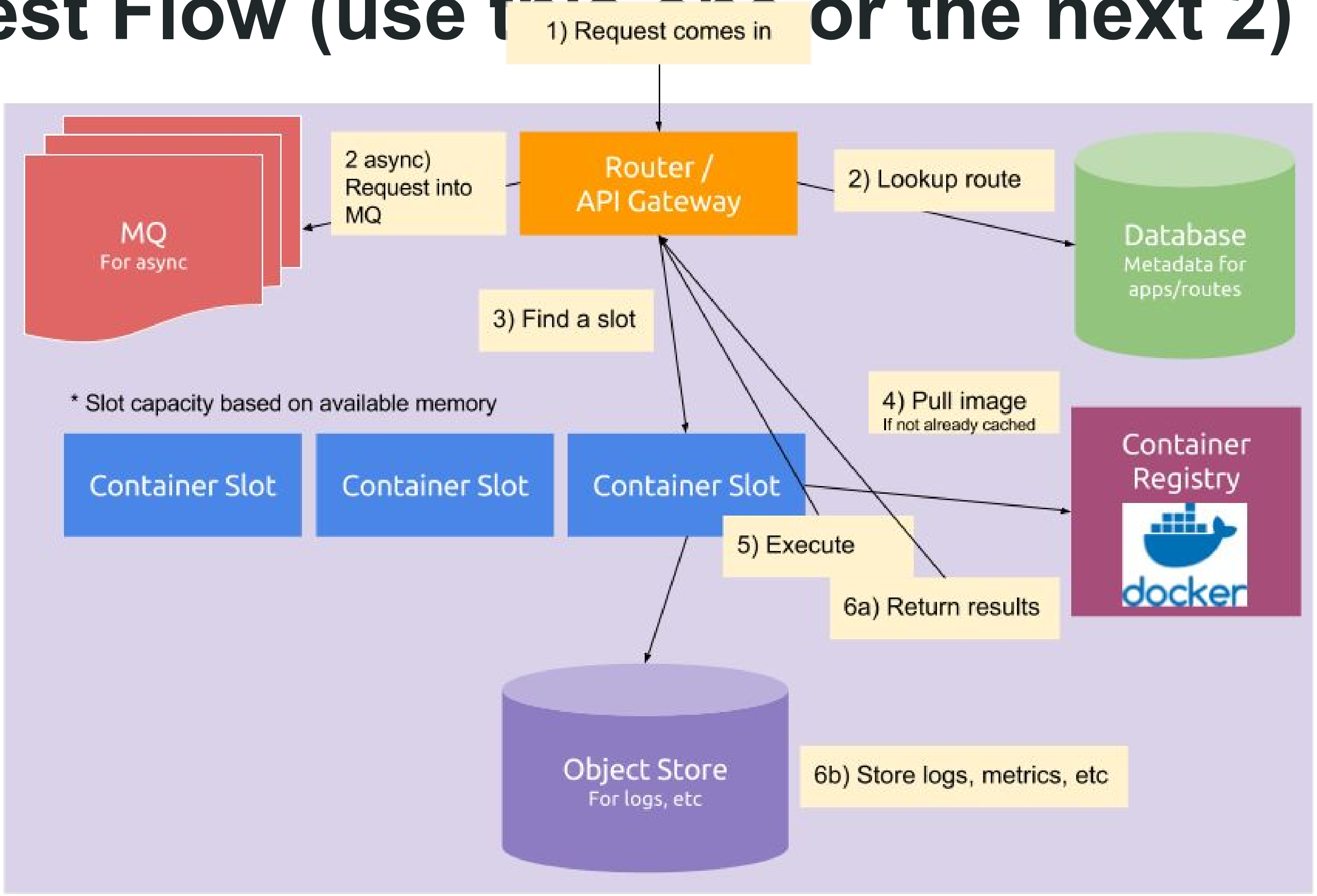
Get Involved

1. Star the project: github.com/fnproject/fn
2. Join the conversation: slack.fnproject.io
3. Learn more: fnproject.io
4. We're hiring engineers and evangelists: **travis.reeder@oracle.com**

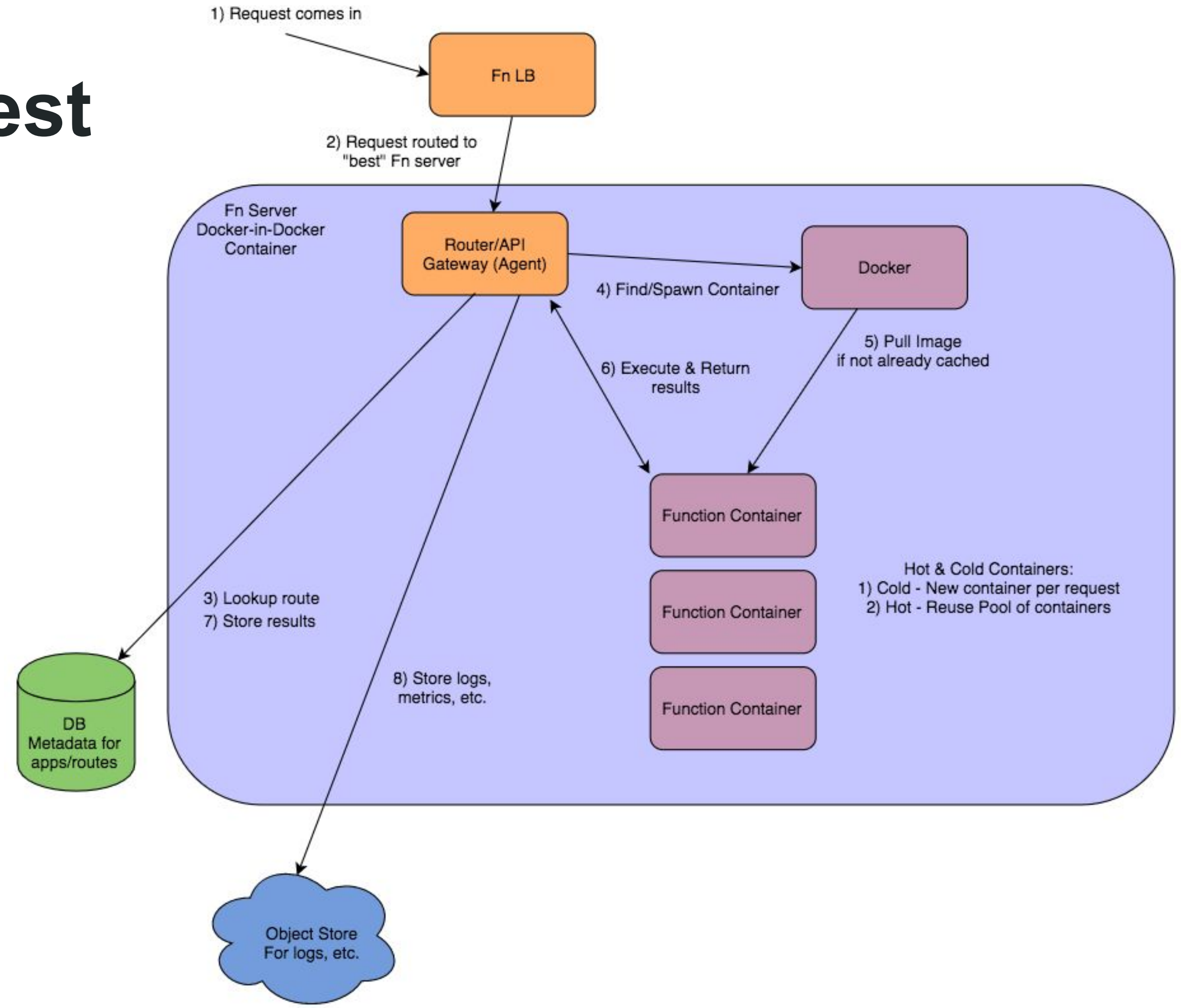
Appendix

Request Flow

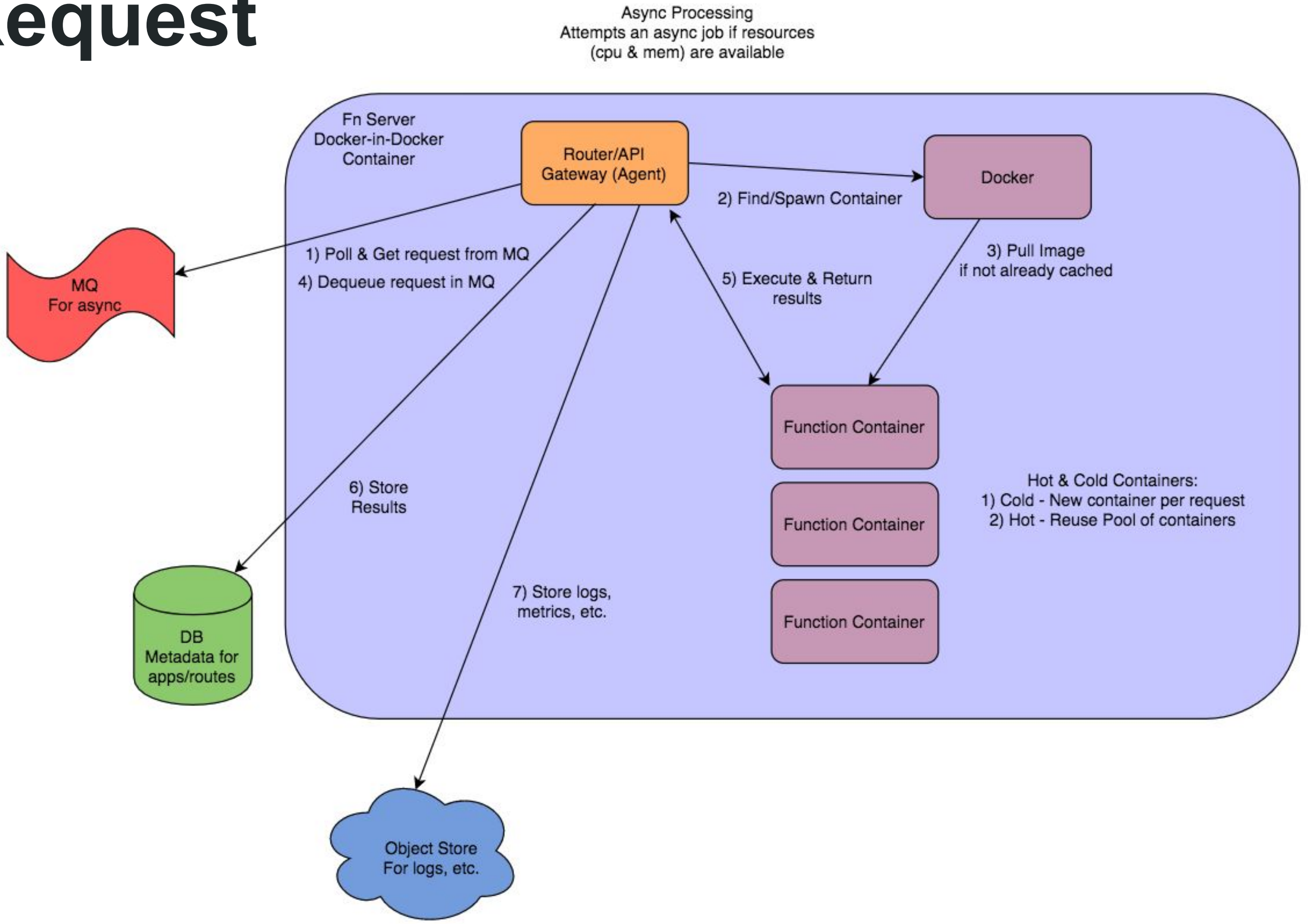
Request Flow (use this one or the next 2)



Sync Request


1) Request comes in
Fn LB
2) Request routed to "best" Fn server
Fn Server Docker-in-Docker Container
Router/API Gateway (Agent)
Docker
4) Find/Spawn Container
5) Pull Image if not already cached
6) Execute & Return results
Function Container
Function Container
Function Container
Hot & Cold Containers:
1) Cold - New container per request
2) Hot - Reuse Pool of containers
3) Lookup route
7) Store results
8) Store logs, metrics, etc.
DB Metadata for apps/routes
Object Store For logs, etc.

Async Request


Async Processing
Attempts an async job if resources
(cpu & mem) are available
Fn Server
Docker-in-Docker
Container
Router/API
Gateway (Agent)
Docker
2) Find/Spawn Container
MQ
For async
1) Poll & Get request from MQ
4) Dequeue request in MQ
5) Execute & Return
results
3) Pull Image
if not already cached
Function Container
Function Container
Function Container
Hot & Cold Containers:
1) Cold - New container per request
2) Hot - Reuse Pool of containers
6) Store
Results
DB
Metadata for
apps/routes
7) Store logs,
metrics, etc.
Object Store
For logs, etc.