Serverless & Google Cloud

Julien Landuré

- Directeur technique Groupe
- GDE Cloud + Official Trainer
- 🙌 Orga GDG Cloud Nantes / DevFest

@jlandure

@jlandure

Éric Briand

- Directeur technique Nantes
- GDE Cloud + Official Trainer
- Orga CNCF Meetup Nantes

@eric_briand

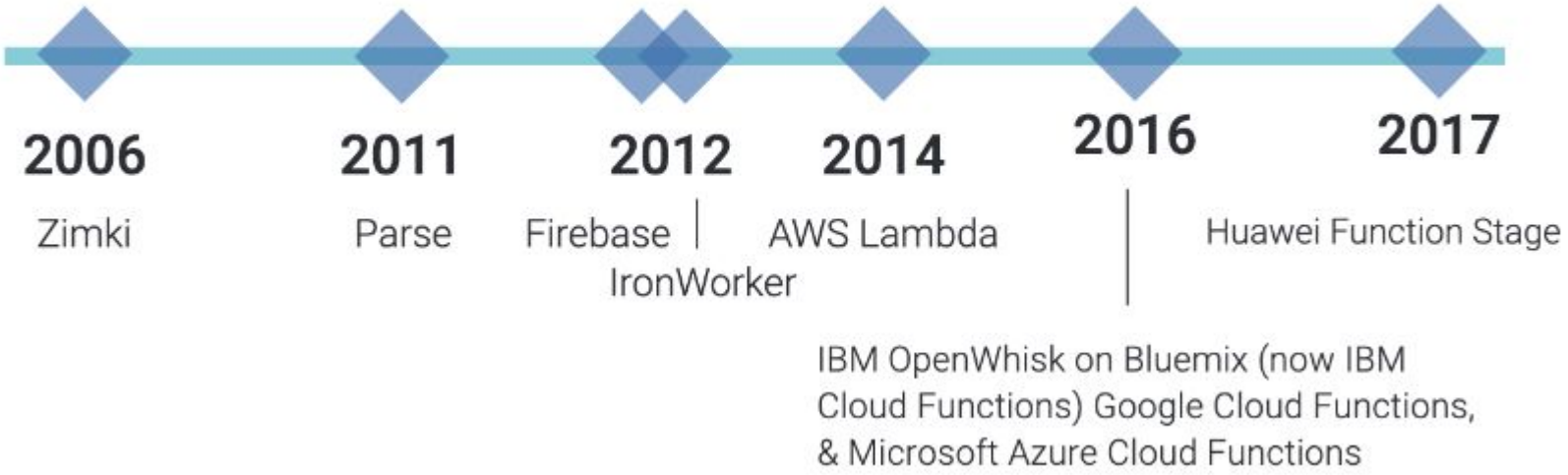
@ebriand

Back to the future...

Serverless, just a buzzword? 🤔

https://youtu.be/P8pCWhPI9B4

A short history of serverless technology



https://github.com/cncf/wg-serverless/blob/master/whitepapers/serverless-overview/cncf_serverless_whitepaper_v1.0.pdf

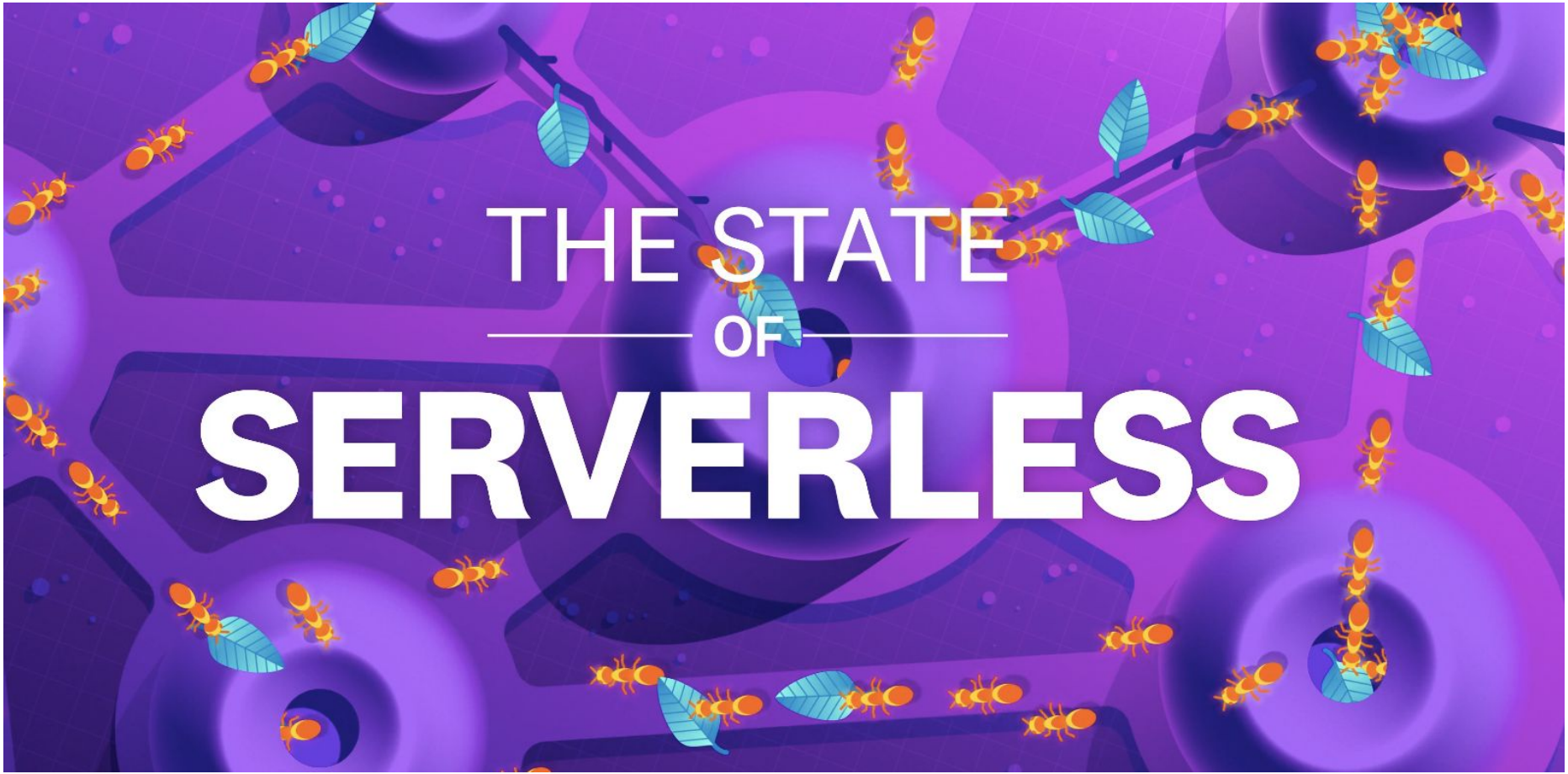
Serverless definition

Serverless computing refers to the concept of building and running applications that do not require server management. It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

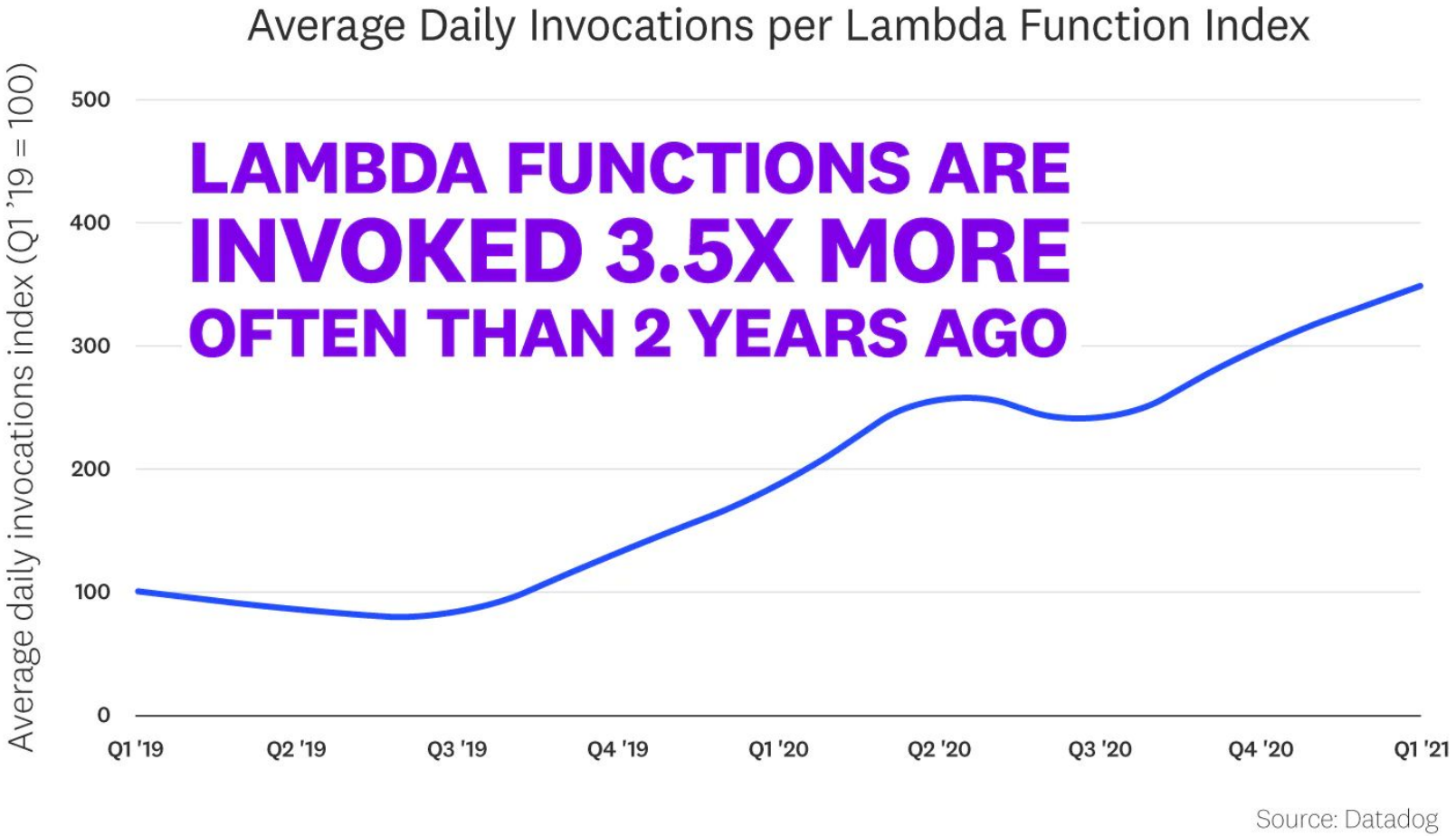
https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview

Serverless in 2021

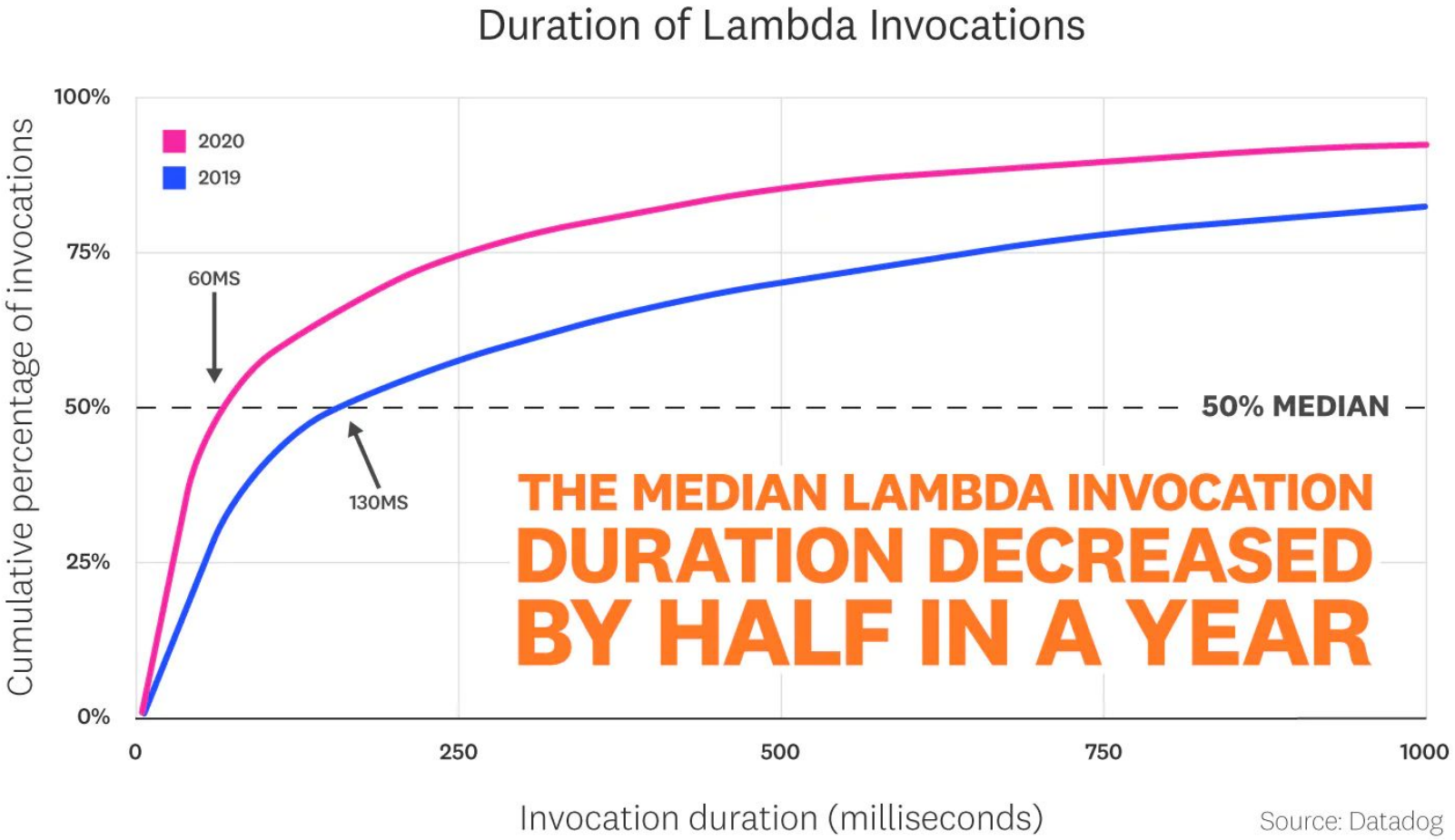
https://landscape.cncf.io/serverless

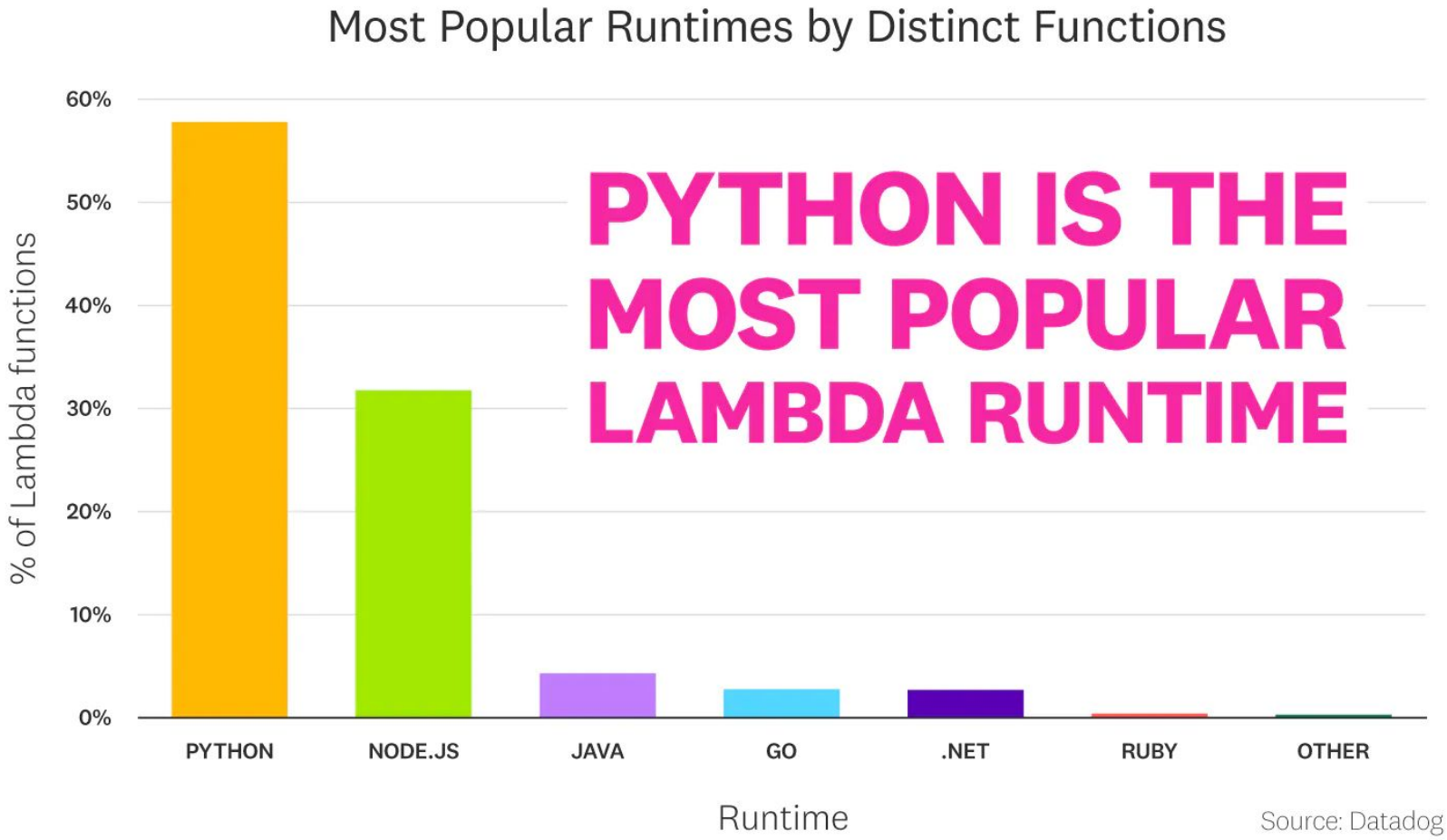


https://www.datadoghq.com/state-of-serverless/

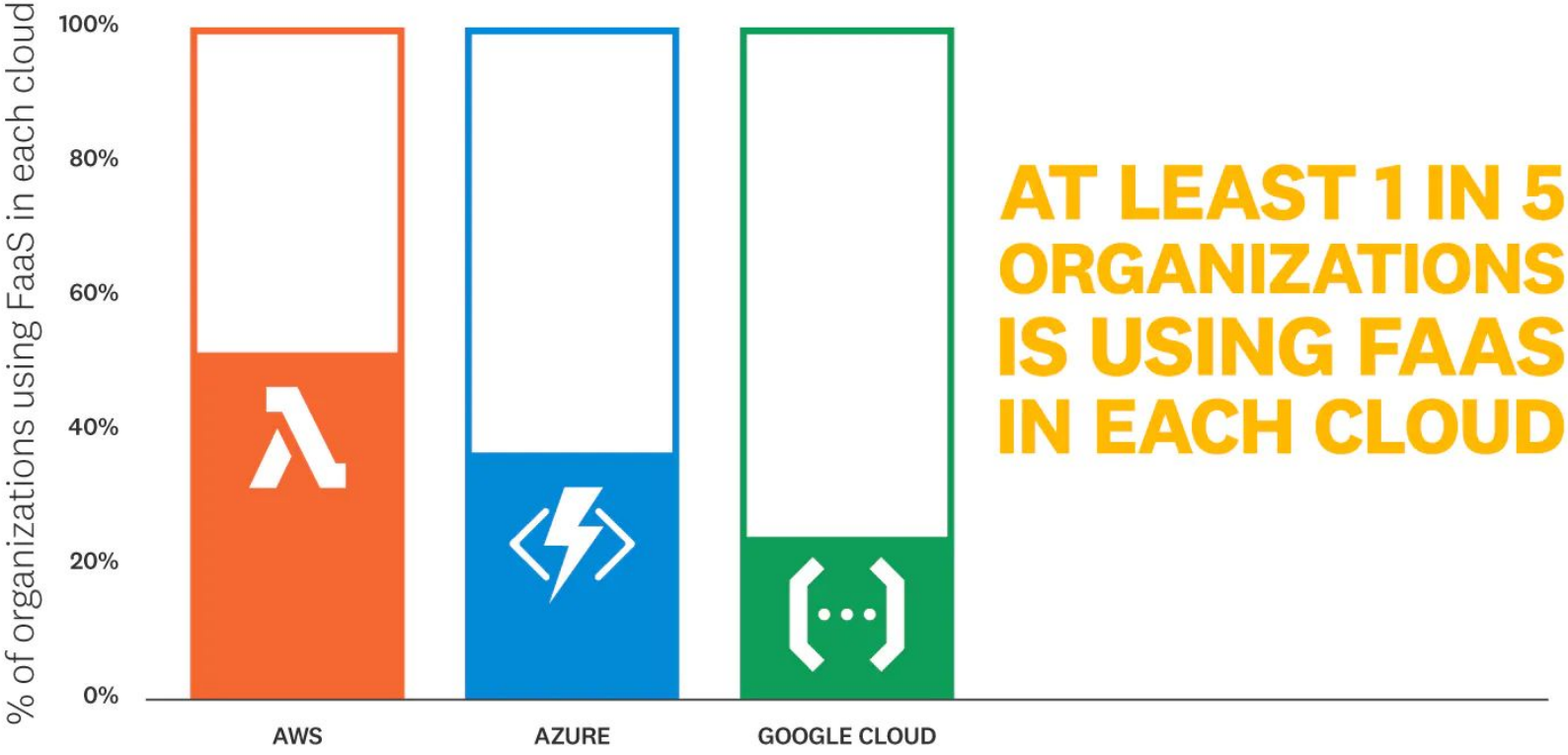


Source: Datadog


Duration of Lambda Invocations
100%
75%
50%
25%
0%
Cumulative percentage of invocations
2020
2019
60MS
130MS
50% MEDIAN
THE MEDIAN LAMBDA INVOCATION
DURATION DECREASED
BY HALF IN A YEAR
0
250
500
750
1000
Invocation duration (milliseconds)
Source: Datadog


Most Popular Runtimes by Distinct Functions
60%
50%
40%
30%
20%
10%
0%
% of Lambda functions
PYTHON IS THE MOST POPULAR LAMBDA RUNTIME
PYTHON
NODE.JS
JAVA
GO
.NET
RUBY
OTHER
Runtime
Source: Datadog

FaaS Usage by Cloud Platform



AT LEAST 1 IN 5 ORGANIZATIONS IS USING FAAS IN EACH CLOUD

Source: Datadog

Positionnement

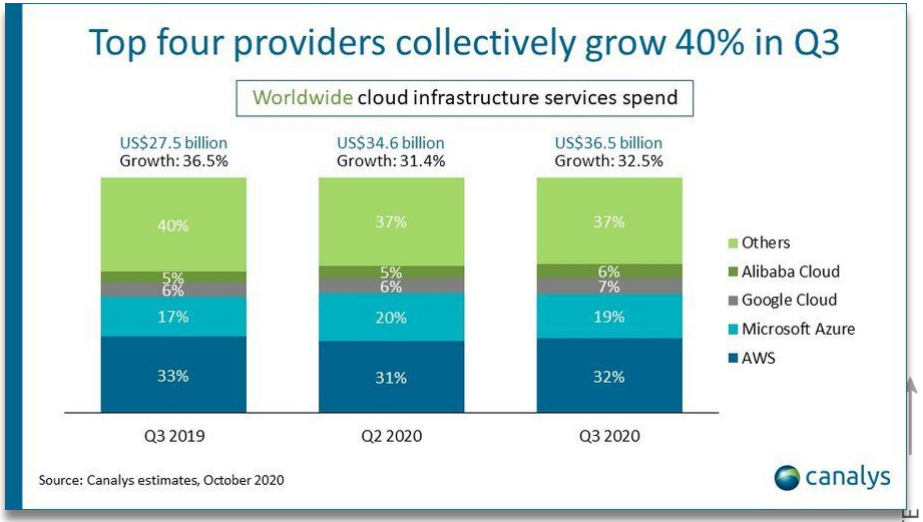
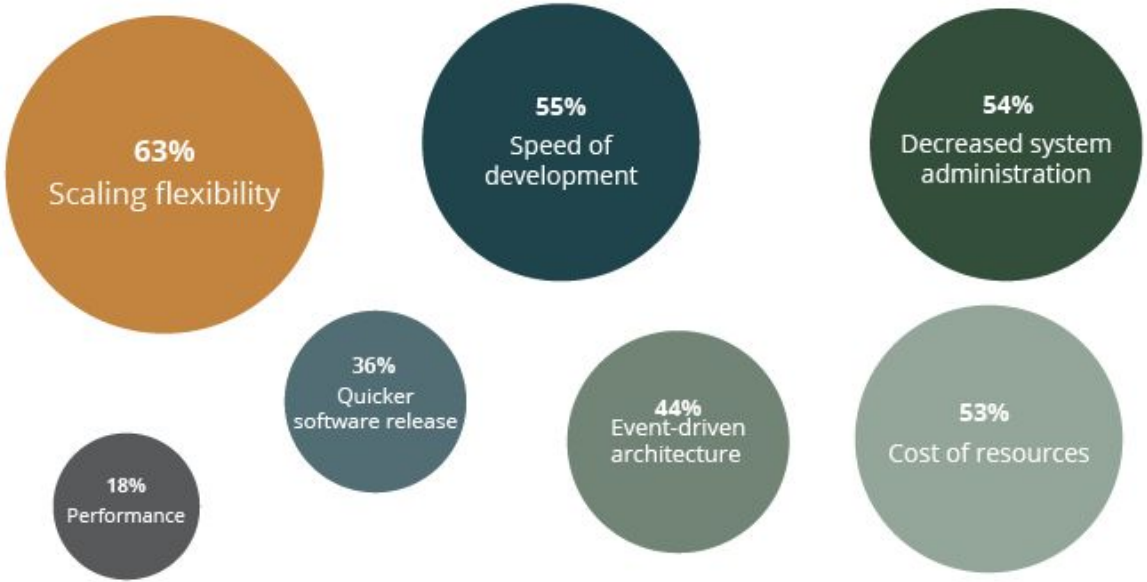


Figure 1. Magic Quadrant for Cloud Infrastructure and Platform Services

Serverless Trends

Why?



The **Serverless** spectrum on Google Cloud

Serverless

Operational Model

No Infra Management

Managed Security

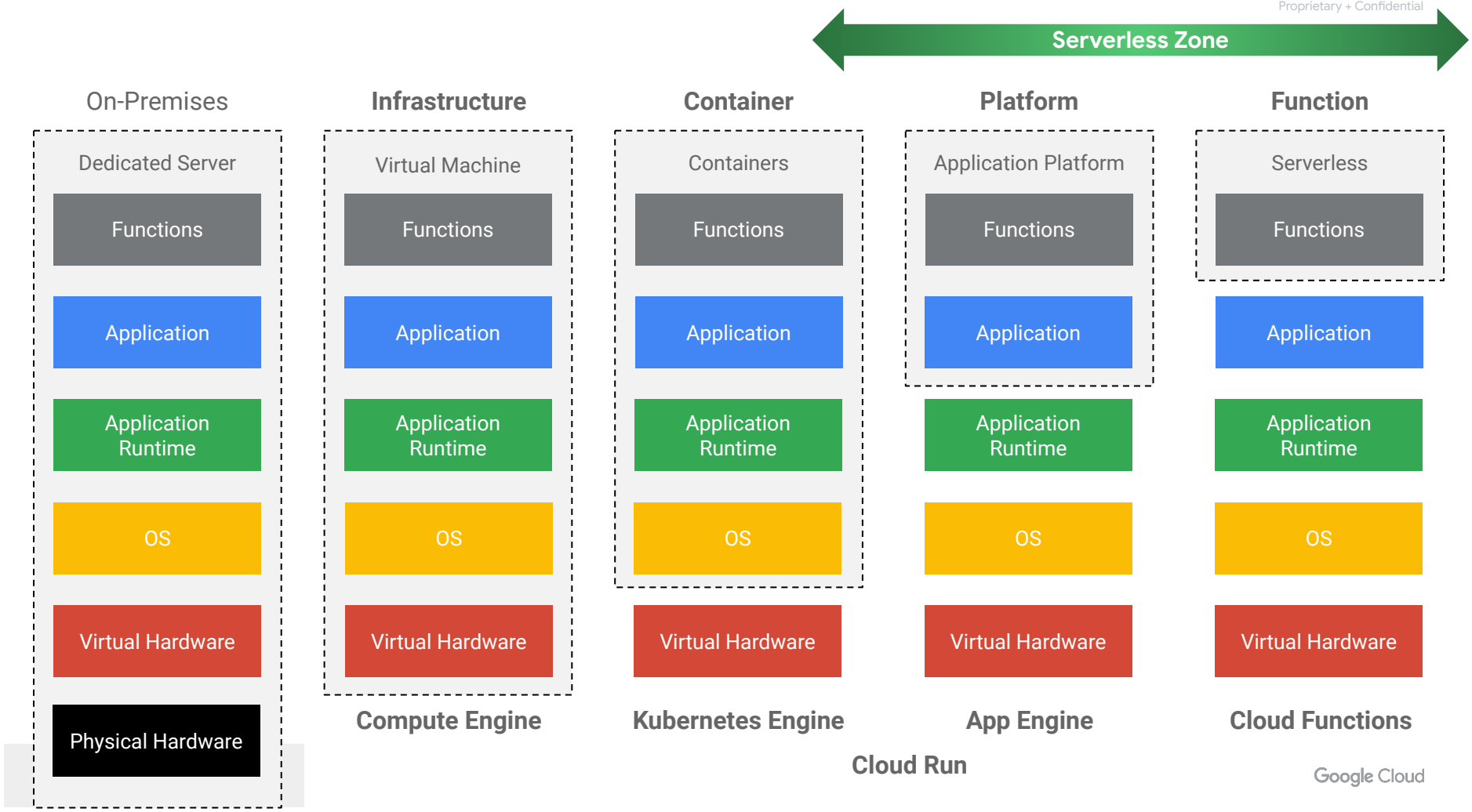
Pay only for usage

Programming Model

Service-based

Event-driven

Stateless



Serverless compute options

Serverless functions

Cloud Functions

Source code-based event driven functions

Serverless web applications

App Engine

Source code-based web apps and API backends

Serverless HTTP workloads

Cloud Run

HTTP containers fully managed

HTTP containers on GKE cluster

Cloud Functions

Cloud Functions

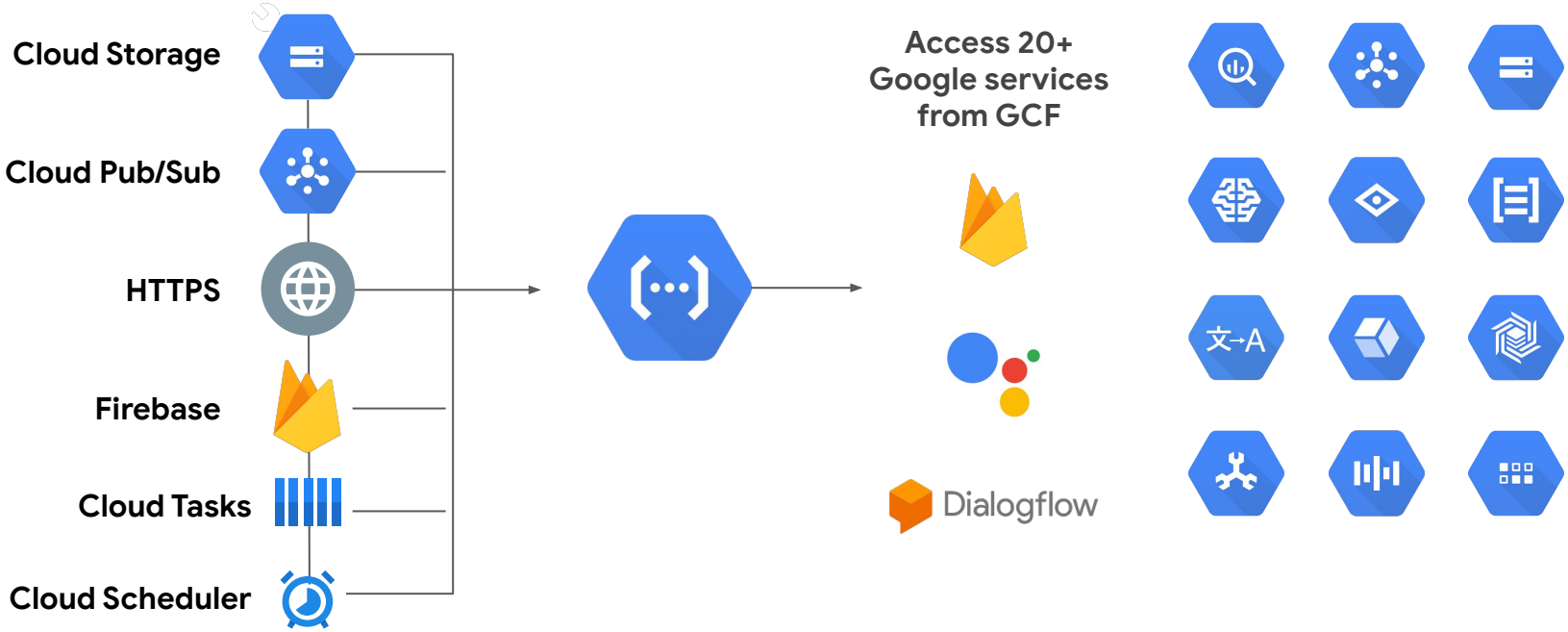
Back end code functions that automatically run in response to a **trigger event**

Microservices

Cloud "glue"

Node.js, Python, Go, Java

Cloud Functions as cloud glue



Good fit? Constraints?

Good fit

"Serverless"

Using Pub/Sub and/or Cloud Storage

Don't want to think about runtime env

Data transformations (ETL)

Cloud-based HTTP glue and webhooks

Constraints

Runtimes: Node.js, Python, Go, Java

Function level granularity

Must interact via events

No custom domain name

Concurrency level of 1 (more cold starts)

Cloud Run

04

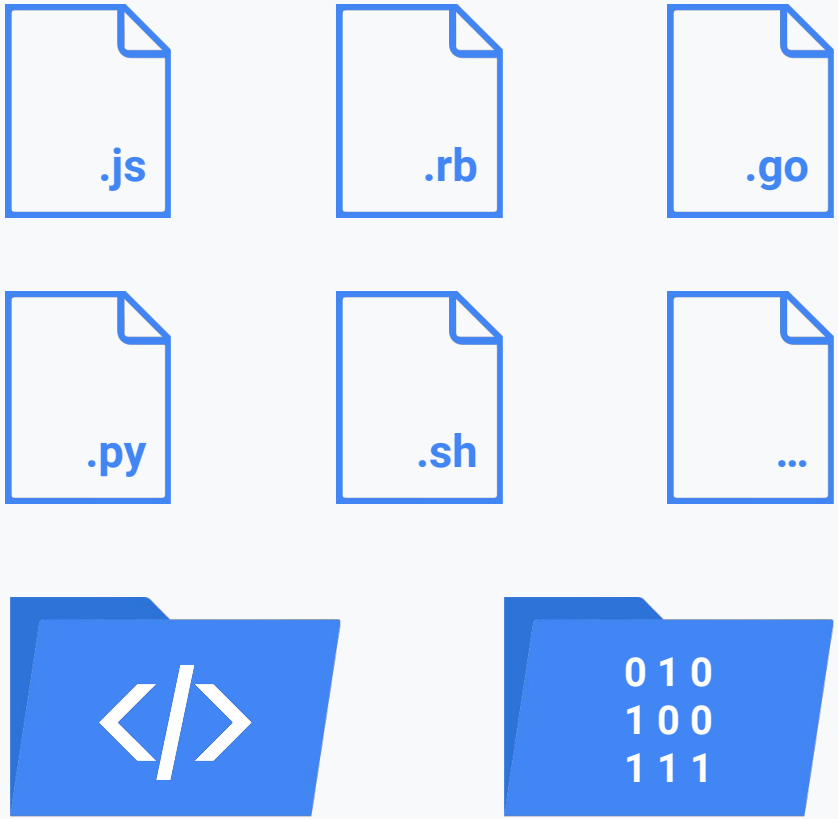
Containers

Any language

Any library

Any binary

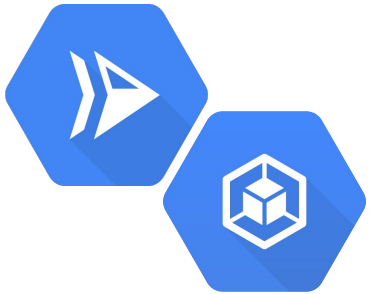
Ecosystem of base images



Serverless containers with Knative & Cloud Run

Cloud Run

Fully managed, deploy your workloads and don't see the cluster.



Cloud Run on Anthos

Deploy into Anthos, run serverless side-by-side with your existing workloads.

Knative Everywhere

Use the same APIs and tooling anywhere you run Kubernetes with Knative.

Container contract

Listen on 0.0.0.0 on port $PORT *(default **8080**)*

HTTP server must start < 4 min *(timeout → 504)*

Request time < 60 min *(default → 5 min)*

Stateless *(in-memory file system, doesn't persist)*

Computation only within request *(No background activity)*

Container resources

1 vCPU per container instance *(configurable to 4vCPU)*

512 MiB of memory up to a max of 8 GiB *(configurable)*

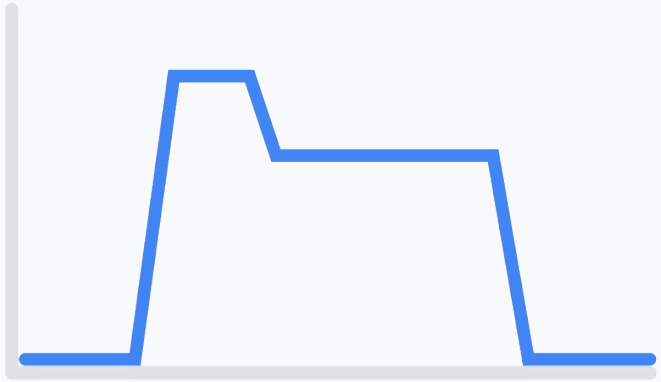
250 concurrent requests per container *(configurable 1-250)*

1000 max containers by default *(configurable 1-1000)*

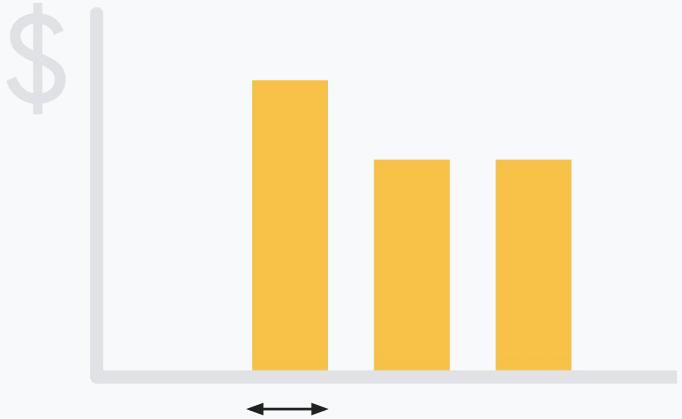
Access to a Metadata Server

Sandboxed by **gVisor**

Pay per use



CPU / Memory / Requests



100ms

Good fit? Constraints?

Good fit

Stateless

HTTP request-response workloads

Scale: way up, down to zero, bursty

Specific runtime requirements: language, dependencies, etc

Constraints

Must use containers

Decide on build process (Cloud Build, etc)

App Engine

What is App Engine?

Serverless, but for your whole web app and not just single functions.

App Engine goal: let **developers focus on code**, GCP handle the rest

Can scale very quickly

Open-source, idiomatic experience

Use any library, extension, binary, or framework

Python, Java, Node.js, PHP, Ruby, Go

Why choose...
App Engine

- The right abstraction: use a web framework that supports routes and HTTP methods
- Web traffic requires minimal latency
- Custom domains
- Java, Python, Node.js, Go, PHP, Ruby

Good fit? Constraints?

Good fit

HTTP/S request-response

Stateless serving applications

Scaling to high traffic

CDN static asset serving

Constraints

Standard (1st gen)

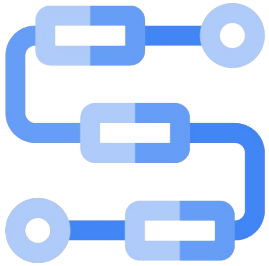
- Runtimes for Python, Java, PHP, Go, Node.js

Standard (2nd gen)

- Can use use binary extensions
- Python, Node.js, Java, Go, Ruby, PHP

Flexible runtimes

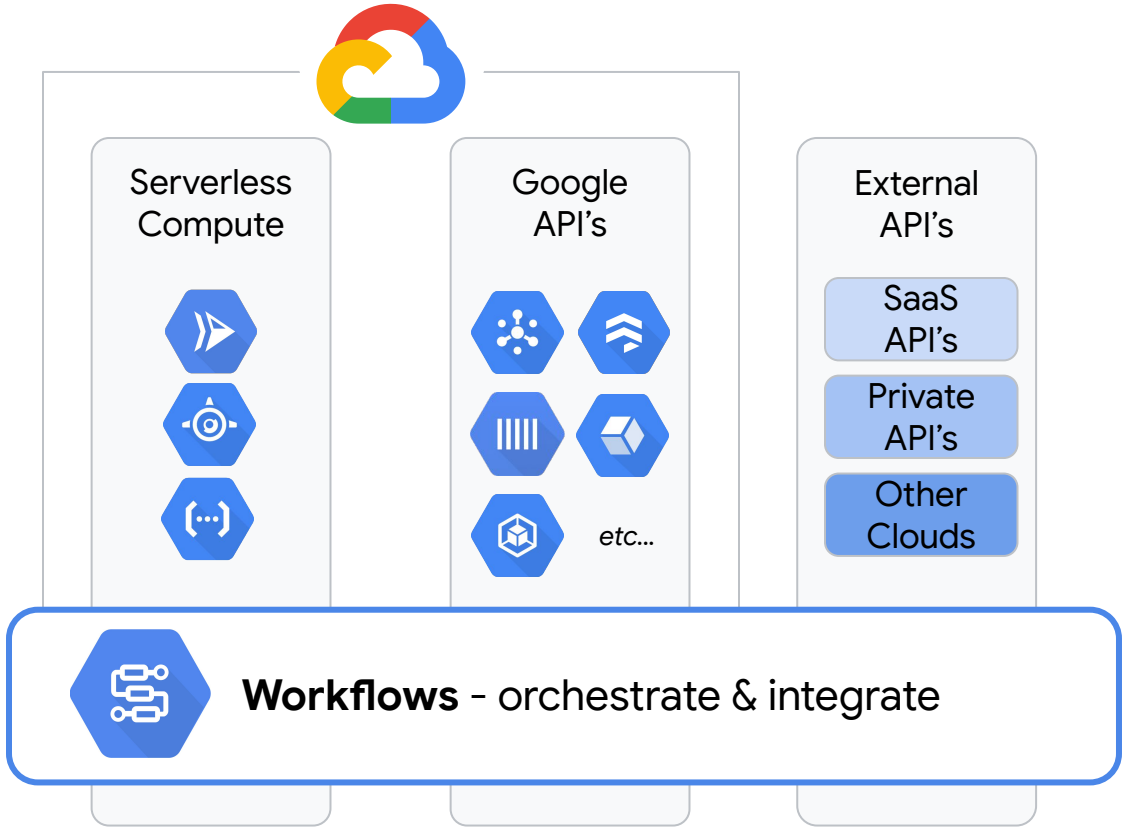
- Inherit Docker constraints
- Not best for very low traffic sites (no scale to 0)
- Slow deploy times

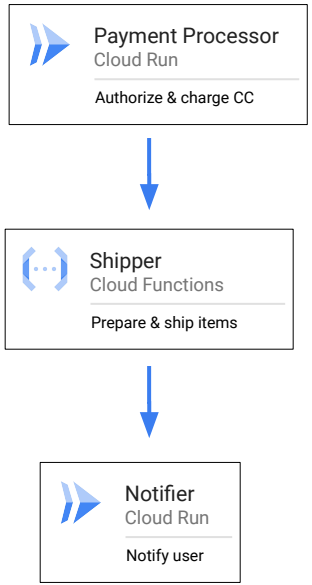


Workflows

From a loosely-coupled event-driven choreography, towards an orchestrated architecture

06


Serverless Compute
Google API's
etc...
External API's
SaaS API's
Private API's
Other Clouds
Workflows - orchestrate & integrate



YAML or JSON syntax

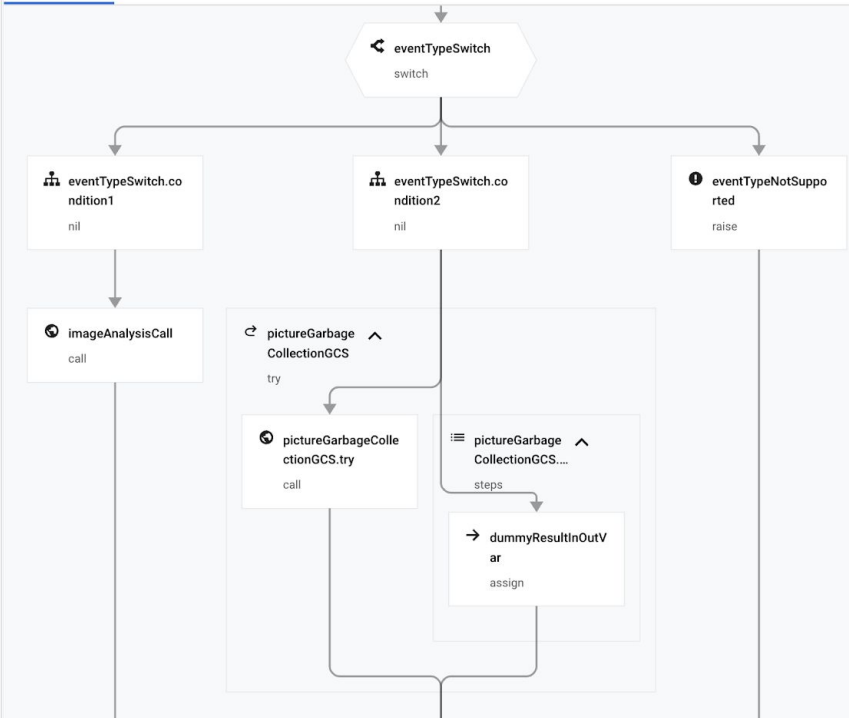
```
- processPayment:
    params: [paymentDetails]
    call: http.post
    args:
        url: https://payment-processor.run.app/...
        body:
          input: ${paymentDetails}
    result: processResult
- shipItems:
    call: http.post
    args:
        url: https://.../cloudfunctions.net/ship
        body:
            input: ${processResult.body}
    result: shipResult
- notifyUser:
    call: http.post
    ...
```

Code

```
main:
  params: [args]
  steps:
    - init:
        assign:
          - file: ${args.file}
          - bucket: ${args.bucket}
          - gsUri: ${"gs://" + bucket + "/" + file}
          - projectId: ${sys.get_env("GOOGLE_CLOUD_PROJECT_ID")}
          - urls: ${args.urls}
    - eventTypeSwitch:
        switch:
          - condition: ${args.eventType == "google.storage.object.finalize"}
            next: imageAnalysisCall
          - condition: ${args.eventType == "google.storage.object.delete"}
            next: pictureGarbageCollectionGCS
    - eventTypeNotSupported:
        raise: ${"eventType " + args.eventType + " is not supported"}
        next: end
    - imageAnalysisCall:
        call: http.post
        args:
          url: https://vision.googleapis.com/v1/images:annotate
          headers:
            Content-Type: application/json
          auth:
            type: OAuth2
          body:
            requests:
            - image:
                source:
```

Visualisation

MAIN



Demo 

Serverless Everywhere

- 👍 Un sujet tendance avec de vrais concepts
- 💪 Un usage “Serverless” large
  - PubSub & Messaging
  - Cloud SQL & Serverless exports
  - Google BigQuery & “serverless data warehouse”
  - AutoML & IA services
- 💸 Démarche FinOps : des tarifs avantageux !
- 🔐 Moins de choses à gérer 👉 meilleure sécurité

Merci ! Des questions ?

Formations Zenika à propos du sujet

- Google Cloud Fundamentals: Core Infrastructure (officielle)
- Architecting with Google Compute Engine (officielle)
- Serverless avec GCP
- Séminaire Cloud : l'état de l'art

@jlandure

@eric_briand