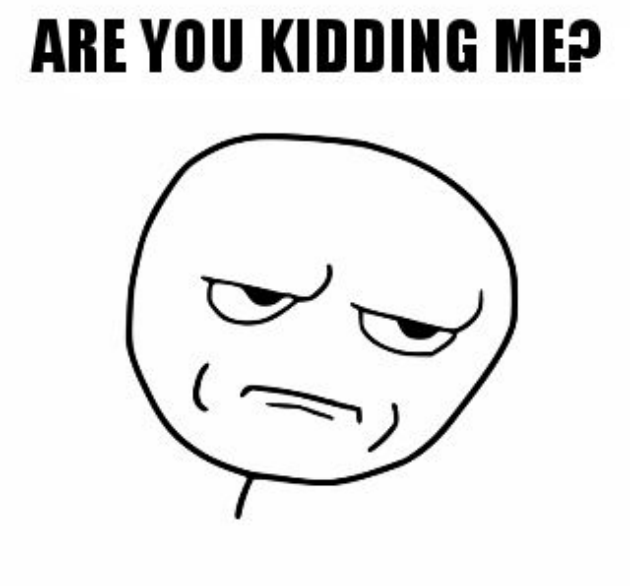
Knative: Building serverless platforms on top of Kubernetes

Ahmet Alp Balkan
@ahmetb

Google Cloud

Thanks to Mark Chmarny, Ryan Gregg, DeWitt Clinton and Bret McGowen for some of the slides used in this presentation.

Talking about servers at a serverless conference



Kubernetes

Serverless

I work on **developer tools** and experiences for **Kubernetes**.

I'm passionate about microservices, infrastructure-as-code, and RPC frameworks.

Past: worked at Microsoft Azure on Linux and Docker.

Follow me at @ahmetb.

Google Cloud

Kubernetes

Kubernetes is the de facto
platform for **running containers**.

Google Cloud

The incredible **Kubernetes** ecosystem

400+ Years of effort*

5,000+ Contributors

40k+ GitHub stars

*Sources: COCOMO Model, CNCF Certified Providers , k8s.devstats.cncf.io

Google Cloud

How did this happen?

Kubernetes encourages **cattle; not pets.**

- Individual machines don't matter.
  - Container isolates the app from the host.

- Containers are ephemeral: they come and go.

How did this happen?

Kubernetes has a declarative API.

- Apply the **desired configuration** to your cluster.
- Kubernetes will drive "current state" to the "desired state" eventually.

How did this happen?

Kubernetes keeps your applications running while you're asleep.

- Container died?
  - Restart it.
- Container unhealthy?
  - Reschedule to another node.
- Container overloaded?
  - Add more replicas automatically.

How did this happen?

Kubernetes API is extensible.

- You can create custom API types.
- You can write custom controllers to actuate the custom objects.

```
apiVersion: my.api/v1
kind: MysqlCluster
metadata:
  name: orders-db
spec:
  masters: 3
  replicas: 12
  storage:
    innodb: {}
```

Kubernetes is not easy

1. It was never meant to be used by developers directly.

2. Creating and operating Kubernetes clusters in production is pretty much a full time job.

Google Kubernetes Engine (GKE)

The **zero ops** cluster experience:

- update your cluster to new versions of Kubernetes
- scale the cluster up/down automatically
- detect and replace broken nodes of the cluster

Kubernetes isn't actually for developers

It's not the correct end-developer experience.

(This doesn't stop developers from using Kubernetes directly!)

But it's a great platform for building a PaaS on top of.

Developers using Kubernetes

Have to do	Want to do
Write code	**Write code**
Build docker image	
Upload image to registry	
Deploy service	
Expose to the internet	
Set up monitoring	
Set up autoscaling	

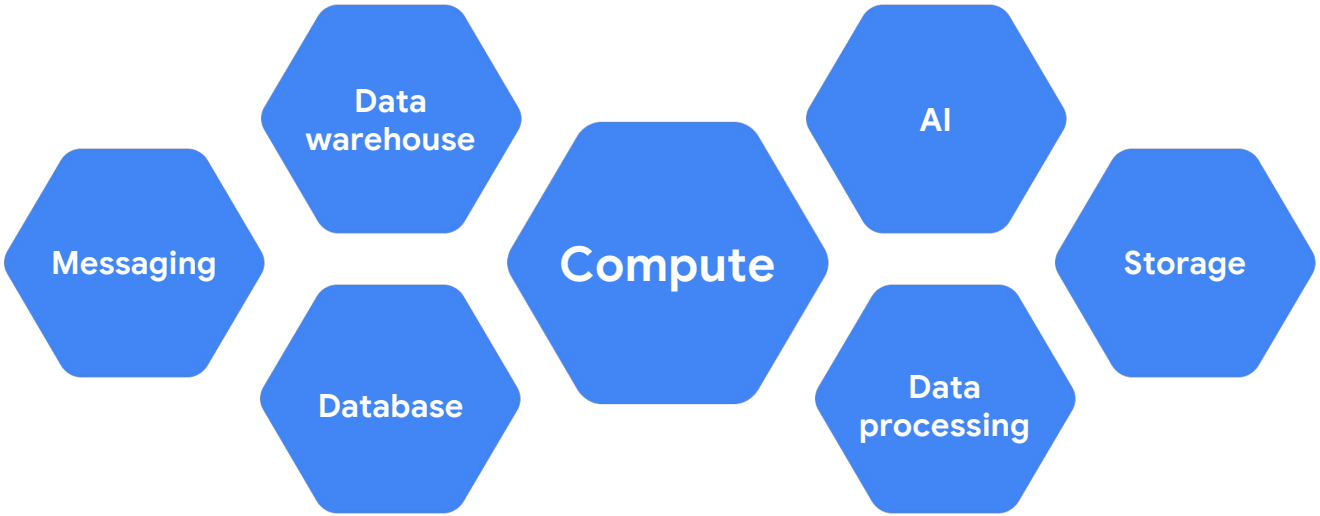
Why do developers still use Kubernetes?

- Some people want control over their infrastructure (VMs, machines, OS images, networking, security, ...)
- Not everyone is on cloud yet, or they want to avoid vendor lock-in.
- Kubernetes lets you effectively manage a large set of {machines, deployments}.

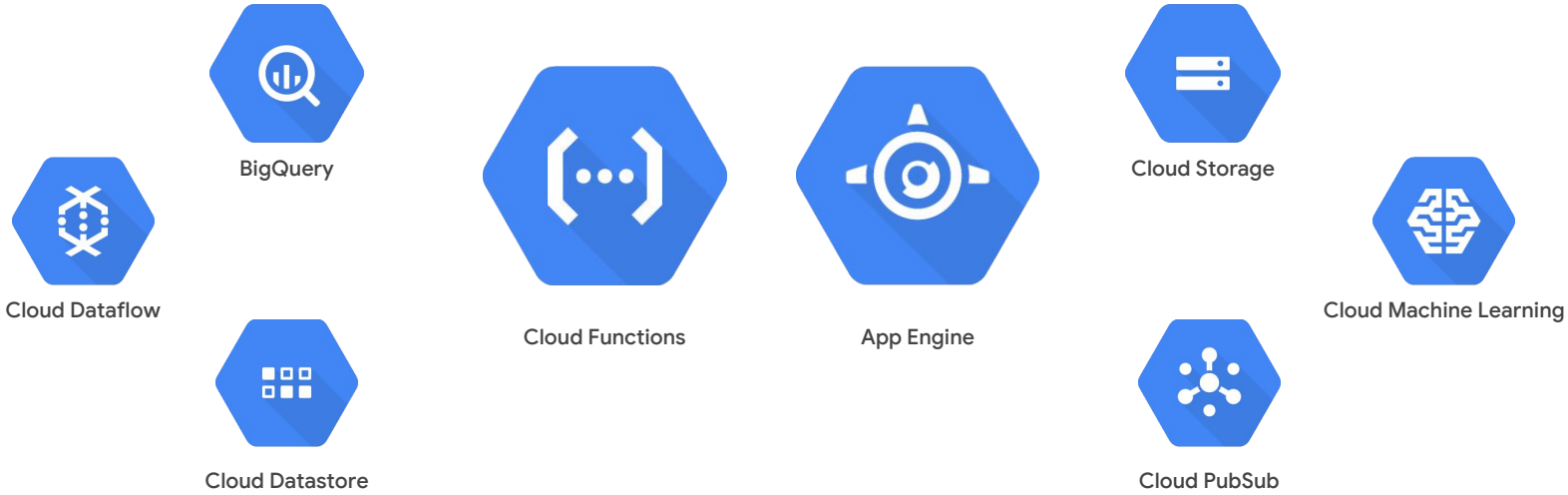
Serverless

Serverless > Functions

Serverless is more than snippets of code



Serverless is more than snippets of code



Promise of serverless

No servers

Idiomatic

Event-driven

Free of lock-in

Developers

... just want to run their code.
... want to use their **favorite languages** and dependencies.
... don't want to manage the infrastructure.

The case for

Serverless on Kubernetes

Developers want serverless

... just want to run their code.

... want to use their favorite languages and dependencies.

... don't want to manage the infrastructure.

Operators want Kubernetes

Kubernetes is great orchestrating microservices

They love using GKE and not having to do operations for Kubernetes.

Kubernetes is not the right abstraction for their developers.

Why would you want serverless on Kubernetes?

Ask the developers of >**13** open source Kubernetes-based FaaS/serverless projects. :)

Why would you want serverless on Kubernetes?

1. Your company doesn't use cloud (or wants vendor lock-in)
2. Need control over the infrastructure, machines, host OS, ...
3. Kubernetes offers good abstractions to build upon.

How to get this on Kubernetes?

No servers

Idiomatic

Event-driven

Free of lock-in

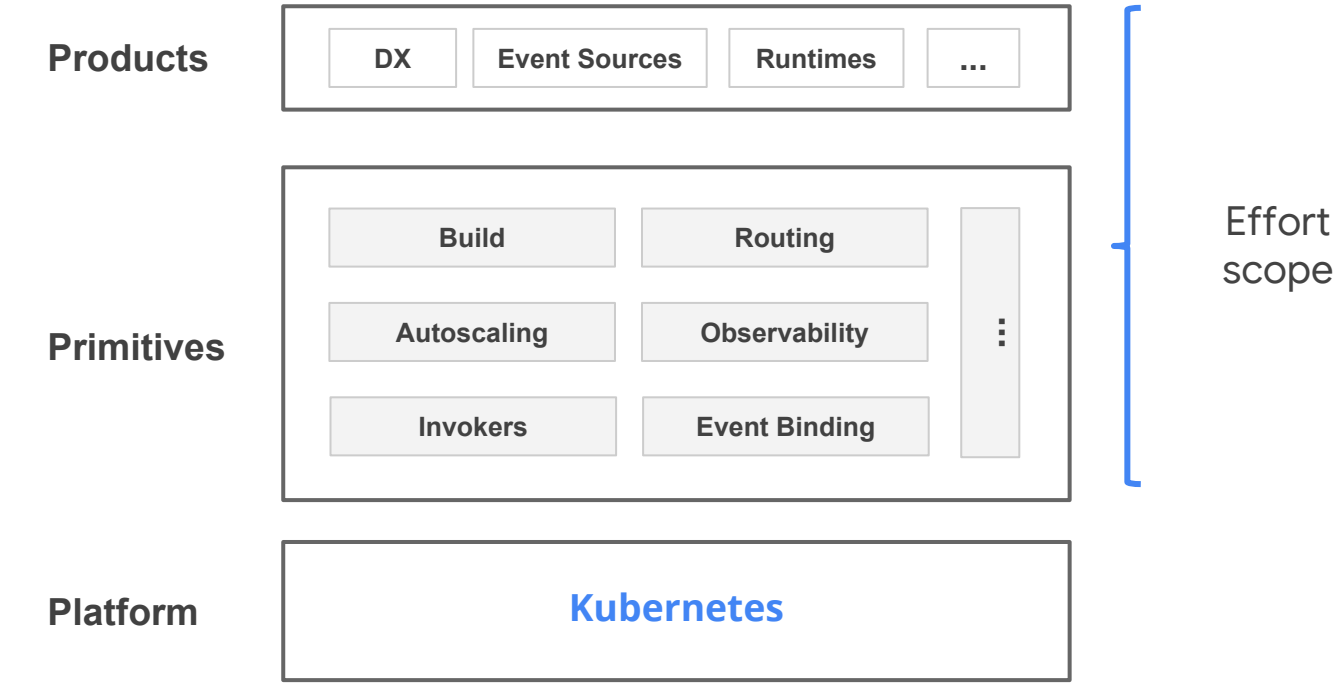
“The majority of people managing infrastructure just want PaaS. There's only one requirement:

“The majority of people managing infrastructure just want PaaS. There's only one requirement: It has to be built by them.”

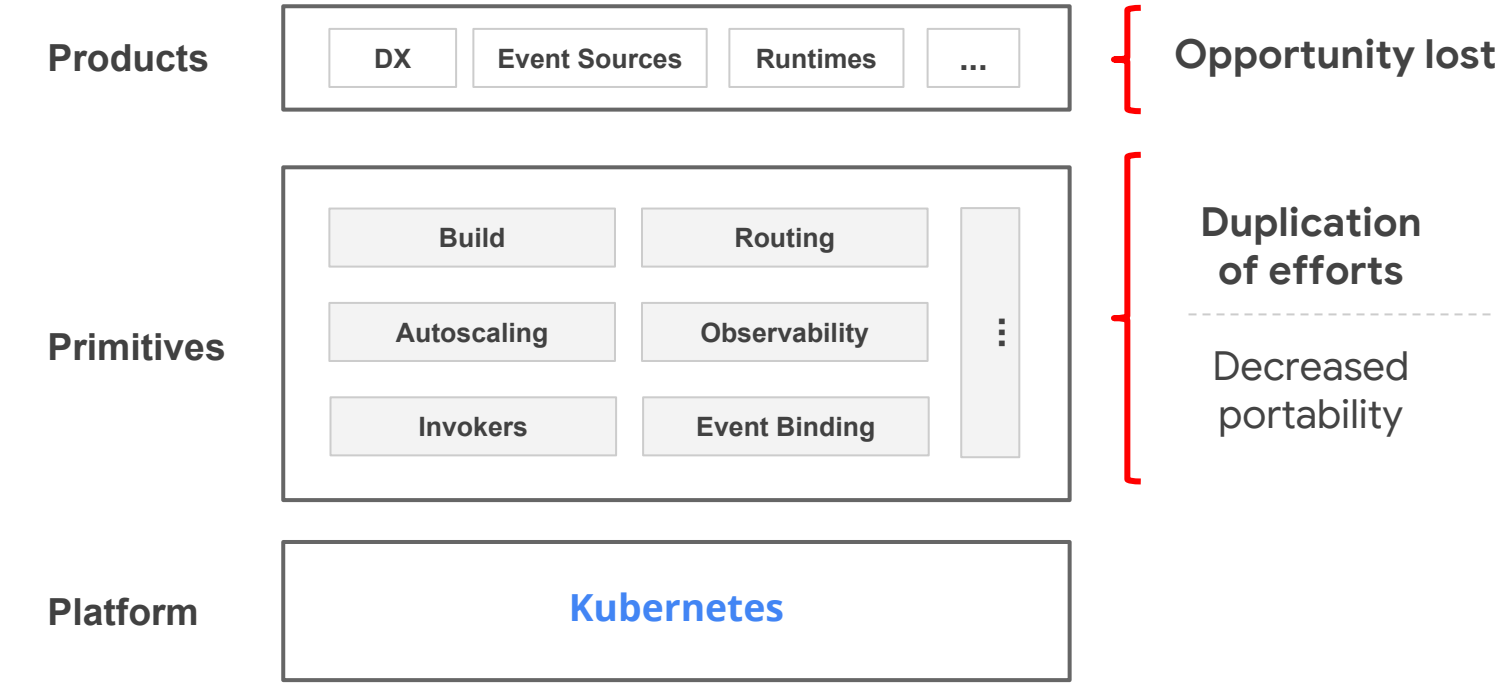
-Kelsey Hightower

Google Cloud

Serverless stack



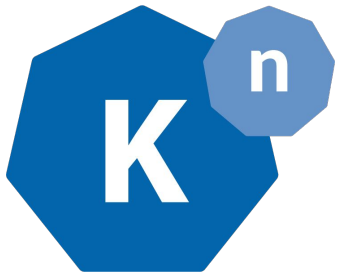
Serverless stack



What if I told you, we can still fulfill the **serverless promises** on servers (but you don't have to manage them)?

Knative

Hello Knative



Building blocks for creating
serverless experiences
on top of **Kubernetes**.

github.com/knative

Knative collaborators

Pivotal

SAP

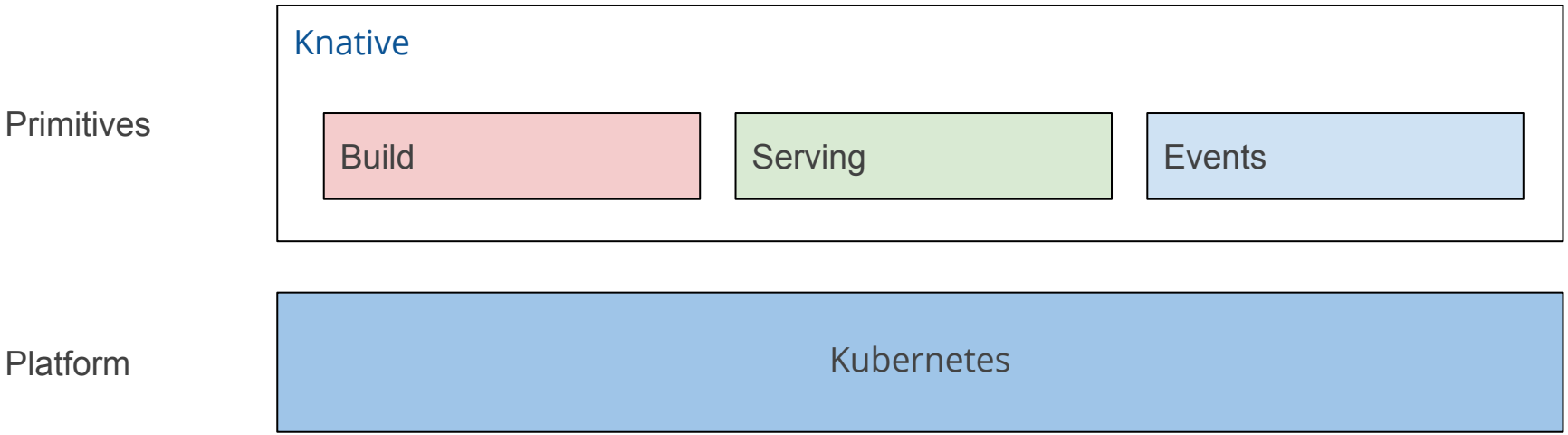
redhat

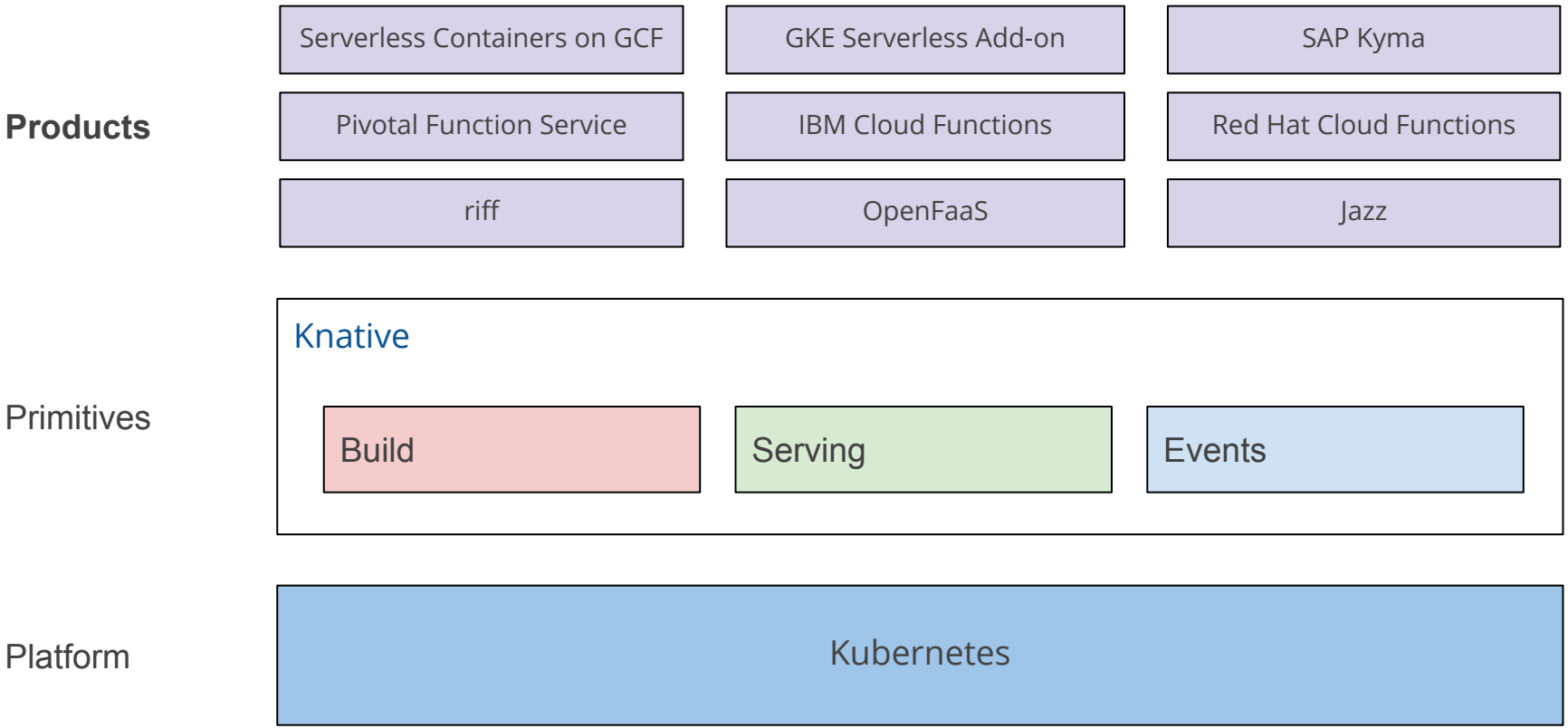
IBM

The Knative Stack

Platform

The Knative Stack





Google Cloud

What Knative is

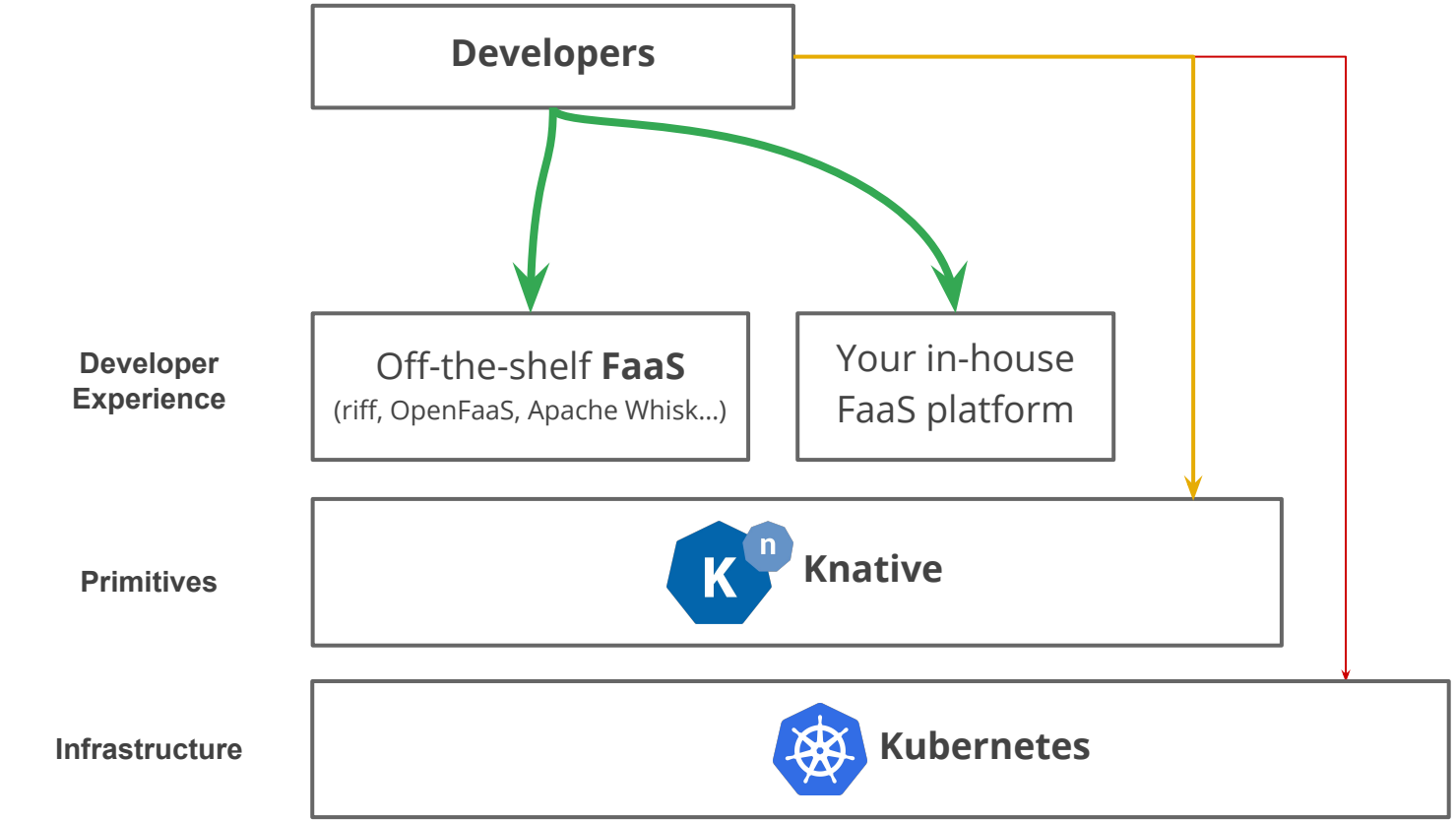
- An open source project
- Set of building blocks to construct your own FaaS/PaaS
  - abstracts common tasks through custom Kubernetes API objects
- An abstraction on top of Kubernetes.
  - **It's still Kubernetes:** Runs containers at the end of the day.

What Knative is not

- It's not a Google product.
- It's not a FaaS.

What can you do with Knative?

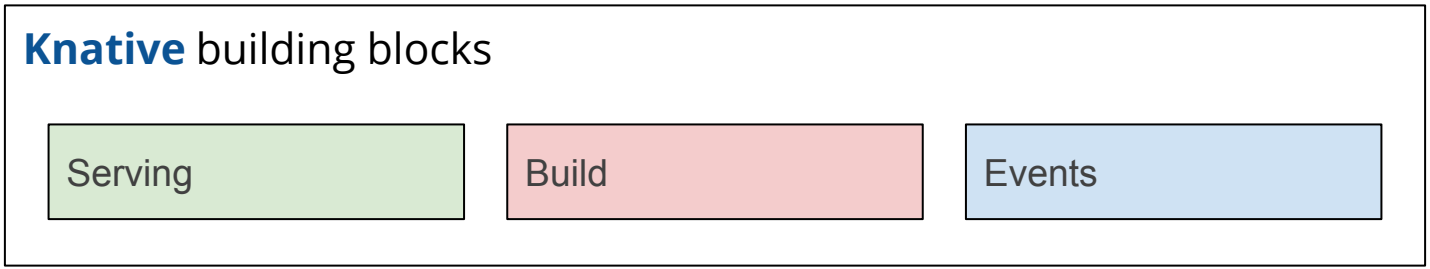
- [Developers] Use it directly to deploy stuff (not easy, but works fine)
- [Operators] Put a level of abstraction between your devs and Kubernetes.
- [Platform Architects] Use it to build your own serverless platform.
  - e.g. DIY Heroku or GCF/Lambda.



Google Cloud

DIY FaaS on Kubernetes (oversimplified)

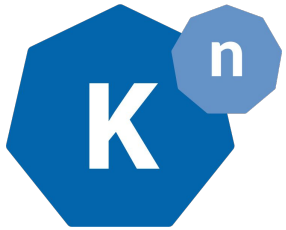
- Something to wake up your workload (activation) on request.
- Something to scale up, and back to zero.
- Something to turn your app/function into a container
- Something to collect metrics and export telemetry from the app.
- Handling of revisions of the code+config (+ability to rollback)
- A way to offer traffic splitting (gradual rollout)
- An eventing system with configurable sources/flows/subscribers


Knative building blocks
Serving
Build
Events

Knative components

- **Serving:** Revisions, Traffic Splitting, Autoscaling
- **Build:** On-cluster builds and transformations
- **Eventing:** Declarative way to bind event sources to services

Knative Serving



Benefits

- Seamlessly scale up and down
- Built-in traffic splitting between revisions
- Integrates networking and service mesh automatically
- Easy to reason about object model

Pluggable

- Connect to your own logging and monitoring platform, or use the built-in system
- Auto-scaler can be tuned or swapped out for custom code

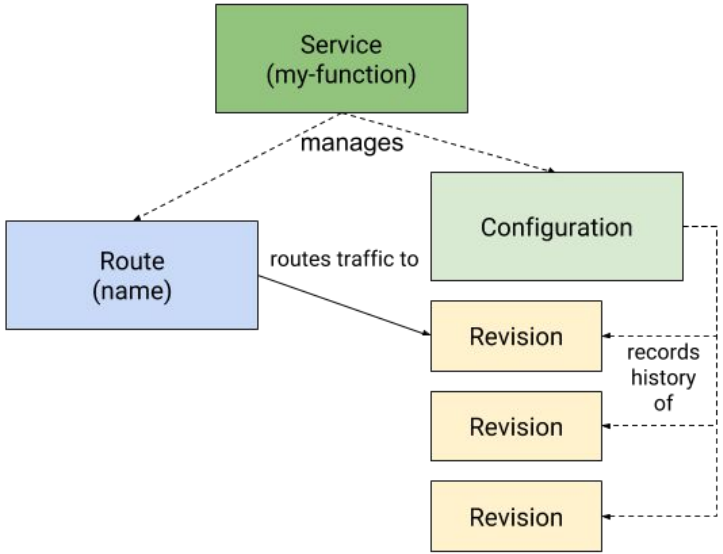
Knative Serving

Primitives with clear separation of concerns:

Configuration
Current/desired state of an application
Code & configuration separated (a la 12-factor)

Revision
Point in time snapshots for your code and configuration

Route
Maps traffic to a revisions
Supports fractional, named routing



Knative Build

Lets you go from source code to container images.

- Build pipelines can consist of **multiple steps**
- Each build step is a **container image**.
- Builds run inside the containers on the cluster.

Makes it possible to do **GitOps** and go from "git push" to a running URL.

Knative Build: Source-to-container image

```
source:
  git:
    url: git://github.com/example/foo.git
steps:
- name: compile
  image: gcc
  args: ["gcc", "-o", "a.out"]
- name: build
  image: docker-cli
  args: ["docker", "build", "--tag", "my-image", "."]
- [...]
```

Knative Build: Function-to-container

Step 1: function to app (add an invoker for your function)

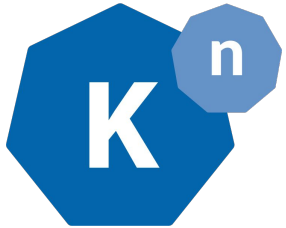
Step 2: app to a container (add a buildpack environment)

Step 3: push container to a registry

Knative Build: Benefits

- Flexible: Control over how your source is turned into artifact (container image).
- Builds happen on the cluster.
    - No need for Docker locally
    - Cached Docker builds
    - Faster image pushes
    - No cross-compiling toil

Knative on-cluster build



Benefits

- No cross-compiling toil
- No need for Docker locally
- Cloud caching, faster image push
- Tooling ecosystem for Enterprise Policy to audit Builds

Loosely coupled

- Use it to get started, and graduate to decoupled CI
- Keep your existing CI/CD to get started, and graduate to audited Builds

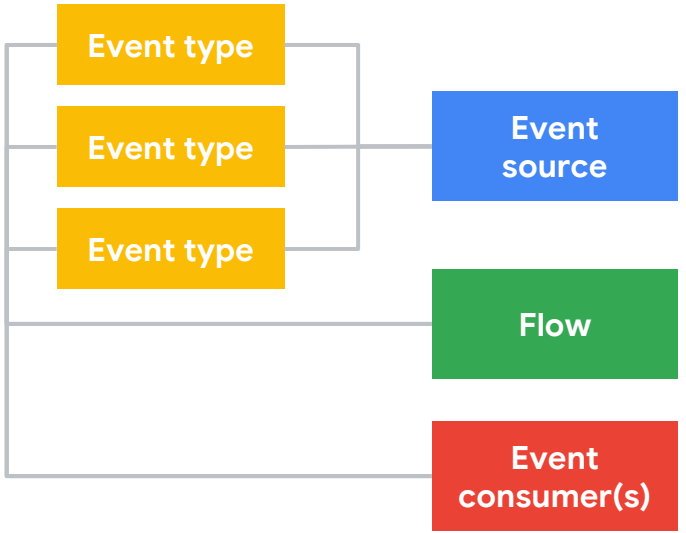
Google Cloud

Knative Eventing

(Work in progress, subject to change.)

Eventing constructs :

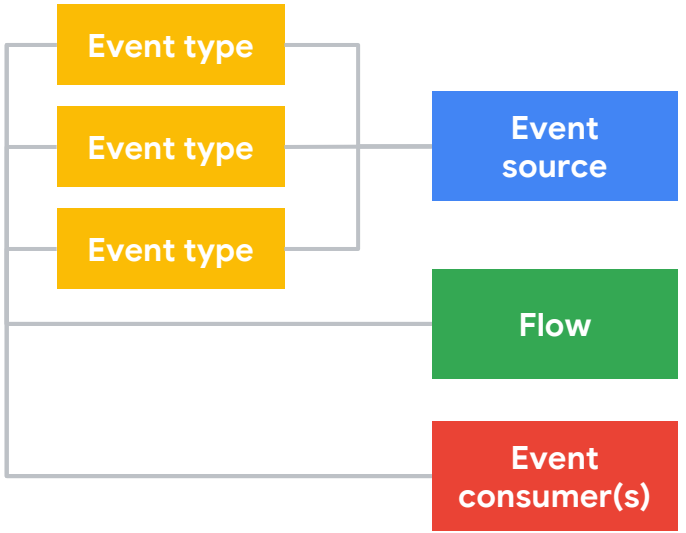
- **Event Sources** (producer)
- **Event Types** (different events)
- **Event Actions** (any route)
- **Event Feeds** (configuration)



Knative Eventing

Benefits

- Declaratively bind between event produces and your services
- Custom event pipelines to connect with your own existing systems



Knative development principles

01	Idiomatic	• Feel native on Kubernetes • Meet the developer
02	Extensible	• Loose coupling at the top • Pluggable at the bottom
03	Organic	• Codify the commonalities • Build a stable platform

Demo

Thanks!

github.com/knative

cloud.google.com/knative

g.co/serverlessaddon (alpha sign-up)