DEV109-One Platform for Your Functions, Applications, and Containers

VILLE AIKAS
SENIOR STAFF SOFTWARE ENGINEER, GOOGLE CLOUD

MARK CHMARNY
TECHNICAL PROGRAM MANAGER, SERVERLESS, GOOGLE CLOUD

WEDNESDAY, JULY 25

Google Cloud

NEXT

About us

Ville Aikas
Sr. Staff Engineer
Google

Mark Chmarny
Tech Program Manager
Google

Kubernetes de facto platform

Why Kubernetes?

- Abstracts infra management pain
- Wide CSP support enables portability
- Rich ecosystem of point-solutions
- Good operator experience

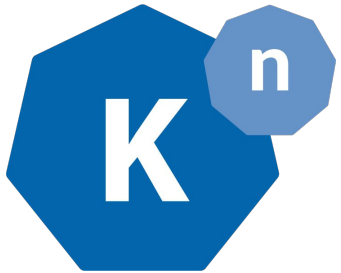
What about developer's experience?

Developers just wanna write code

Have to do	Want to do
Write code	Write code
Build docker image	
Upload image to registry	
Deploy service	
Expose to the internet	
Setup logging & monitoring	
Scale workload	

Introducing Knative (again)

Kubernetes-based building blocks for serverless workloads

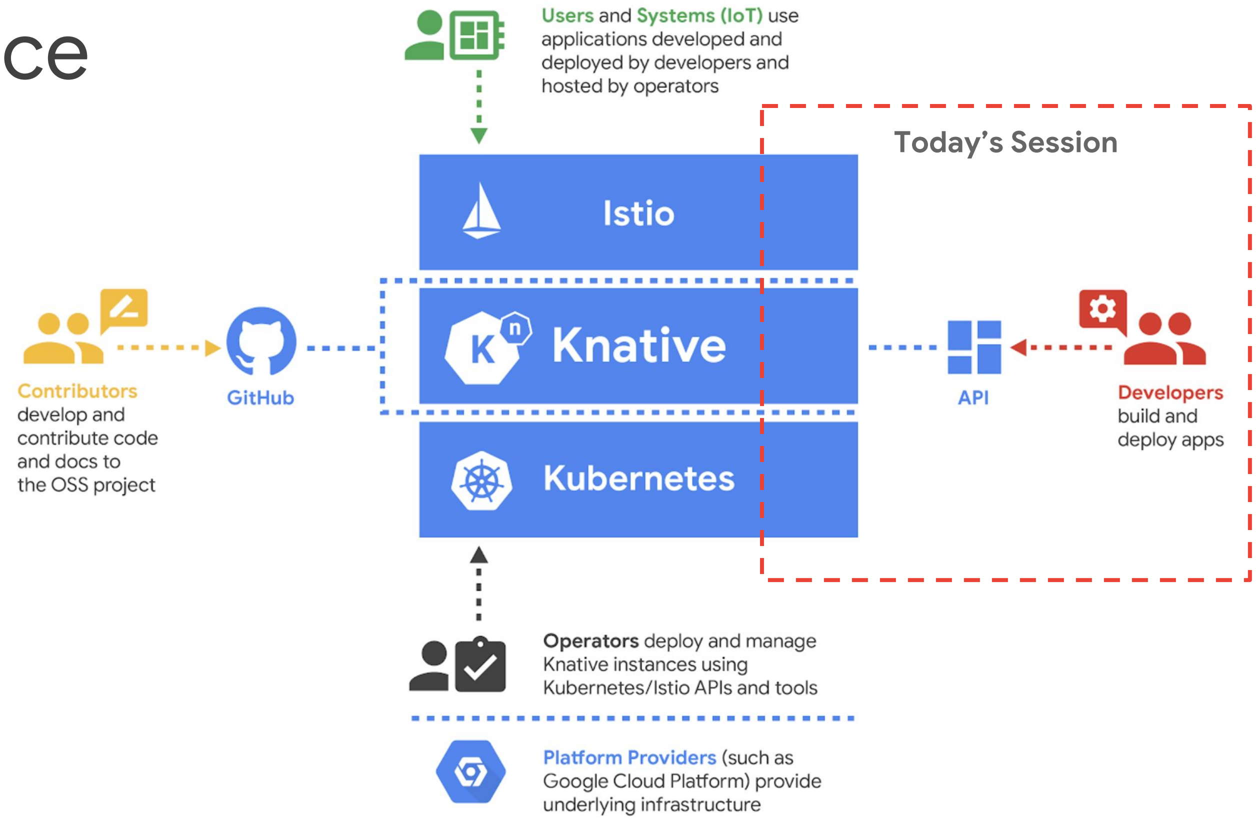


- Set of primitives (Build, Events, Serving)
- Solves for modern development patterns
- Implements learnings from Google, partners
- Ingredient or platform for OSS FaaS frameworks

github.com/knative

Knative Audience

Knative Audience



Knative objects
...and demos

Knative is easy to start with

- Specify only what's necessary
- API familiar to existing Kubernetes users
- Easy to start, single command — it just works

DEMO: deploying pre-built image

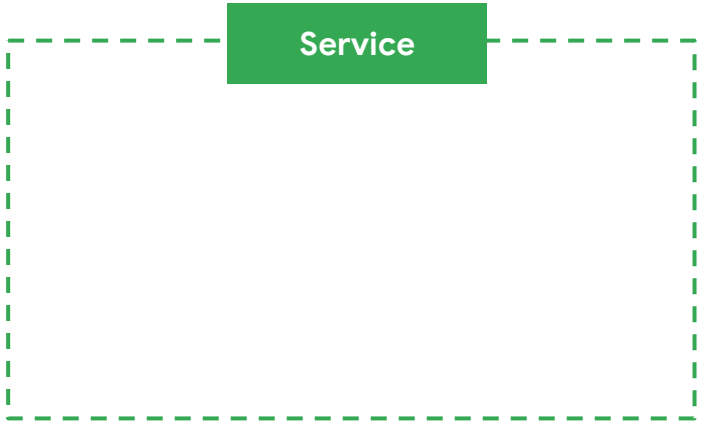
Knative also grows with You to address more complex use-cases

Knative Serving defines principled objects

Knative defines primitives with clear separation of concerns

So far, we used Service, a **lite version of Knative objects**

```
spec:
   container:
      image: gcr.io/knative-samples/simple-app:latest
      env:
         - name: SIMPLE_MSG
           value: "Hello GCP Next 2018!"
```

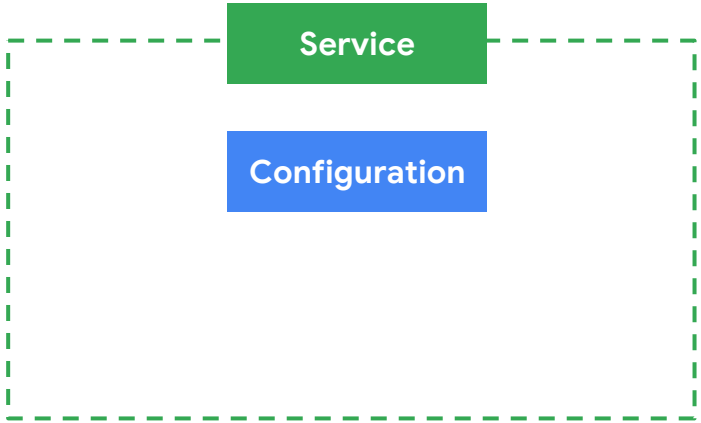


NEXT'18

Knative Serving defines principled objects

Knative defines primitives with clear separation of concerns

Configuration
Current/desired state for your application
Code & configuration (separated, ala 12 factor)



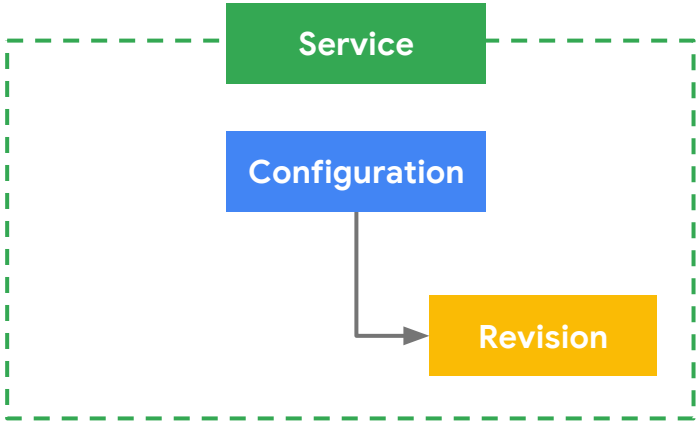
NEXT'18

Knative Serving defines principled objects

Knative defines primitives with clear separation of concerns

Configuration
Current/desired state for your application
Code & configuration (separated, ala 12 factor)

Revision
Point in time snapshots for your code and configuration



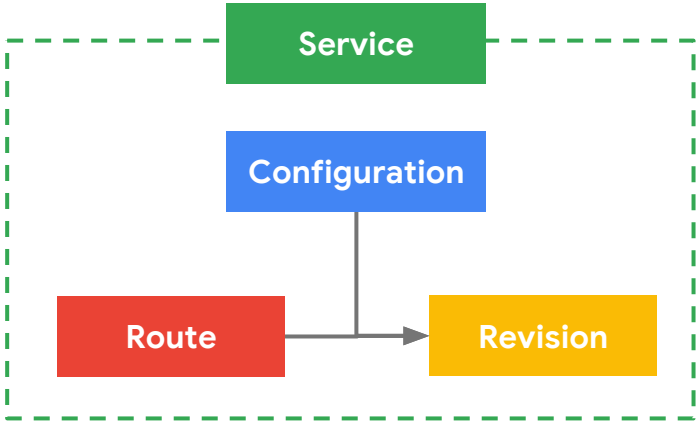
Knative Serving defines principled objects

Knative defines primitives with clear separation of concerns

Configuration
Current/desired state for your application
Code & configuration (separated, ala 12 factor)

Revision
Point in time snapshots for your code and configuration

Route
Maps traffic to a revisions
Supports fractional, named routing

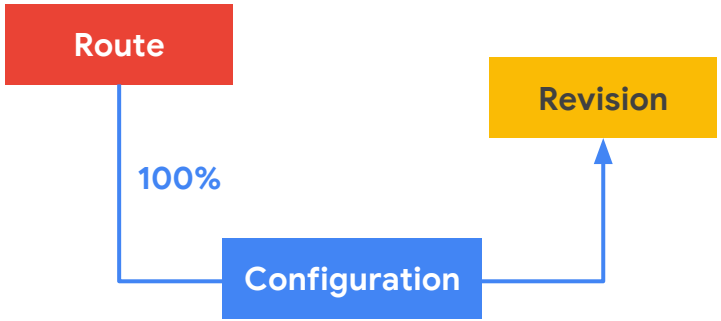


Knative is good day one,
even better in days after

Blue-green deployment model

Demo: Deploying and updating live service

```
kind: Route
...
spec:
  traffic:
  - revisionName: blue-green-00001
    percent: 100
```



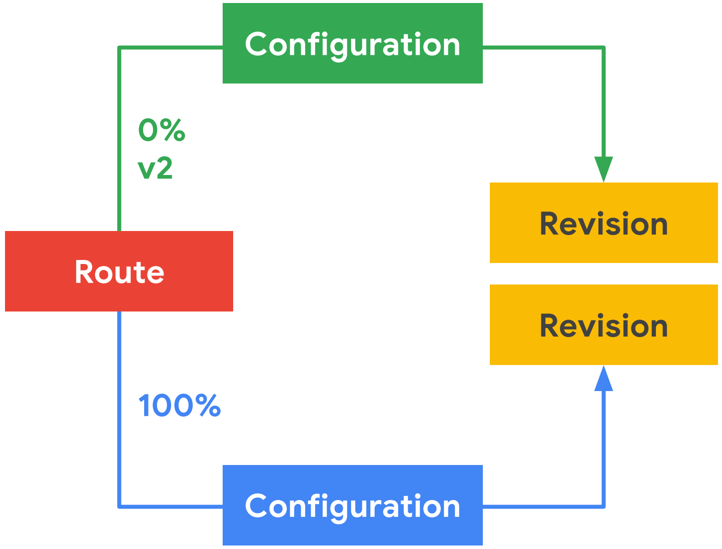
NEXT'18

Blue-green deployment model

Deploy updated version of the service

- Blue continues to take 100% of traffic
- Named route (v2) to green version

```
kind: Route
...
spec:
 traffic:
 - revisionName: blue-green-00001
   percent: 100
 - revisionName: blue-green-00002
   percent: 0
   name: v2
```

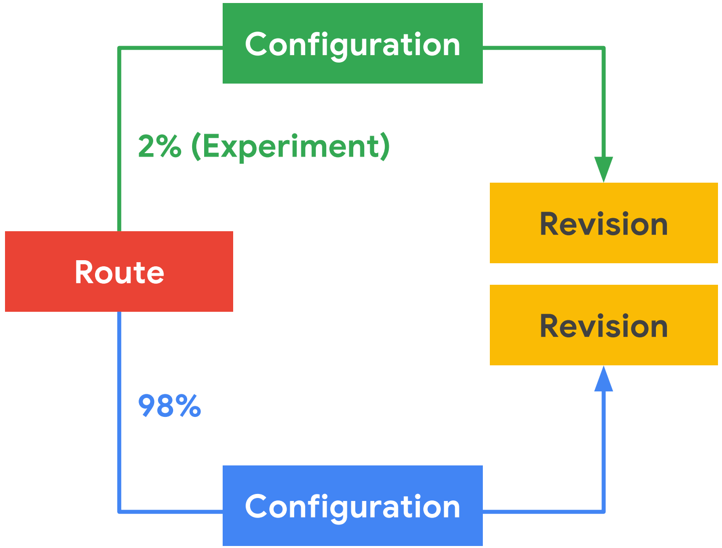


Blue-green deployment model

Update service configuration

- Send % of traffic to green
- Still have explicit v2 route

```
kind: Route
...
spec:
 traffic:
 - revisionName: blue-green-00001
   percent: 50
 - revisionName: blue-green-00002
   percent: 50
   name: v2
```

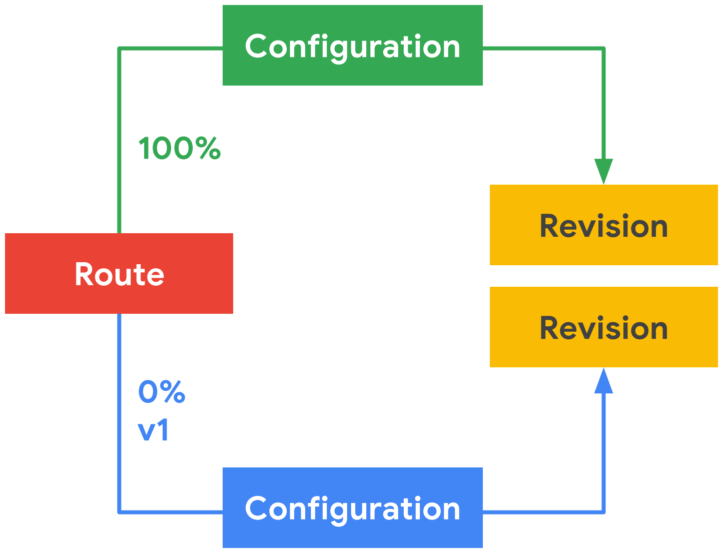


Blue-green deployment model

Update service configuration

- Incrementally add %, until all traffic is on green
- Keep explicit named route to blue
  Secured with RBAC-based ACL

```
kind: Route
...
spec:
 traffic:
 - revisionName: blue-green-00001
   percent: 0
   name: v1
 - revisionName: blue-green-00002
   percent: 100
```



NEXT 18

Knative automates many common tasks

Knative auto-scales

Knative scales down when you don’t need it

- Developers don’t have to think about underlying infrastructure

Knative auto-scales

Knative scales up linearly with your load

- Supports unpredictable usage pattern
- 1-n when you app starts taking traffic

DEMO: Scaling to 0, 0-1, 1-n based on RPS

Knative auto-builds

Supports GitOps or src-to-URL development patterns

Demo: Deploy from git repo to user-accessible URL

Why developers care?

- 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.

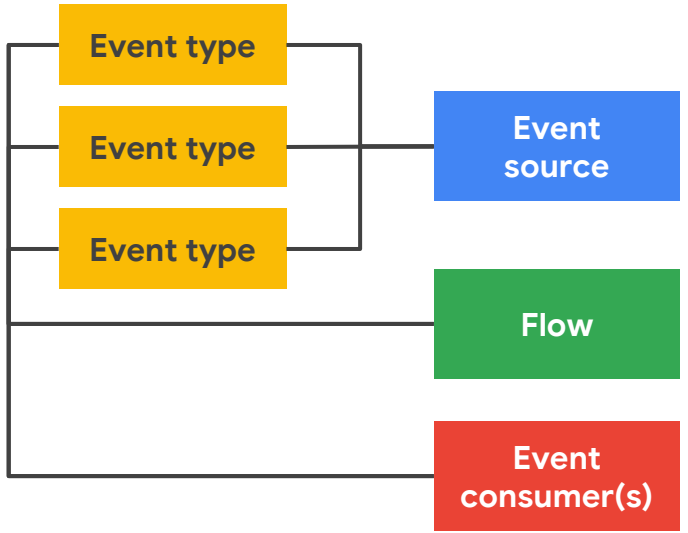
Knative is serverless, and serverless is more than just functions

Knative Eventing defines principled objects

Eventing constructs :

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

DEMO: Processing IoT Core events



NEXT'18

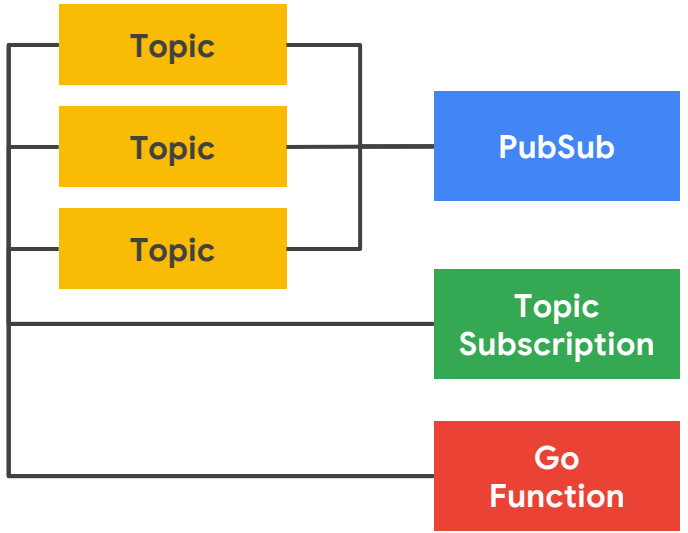
Knative Eventing defines principled objects

Same API, whether apps and functions

User-defined event sources

Decoupled event producers and consumers

CNCF CloudEvents support (0.1)



So how does Knative help developers?

Developers just wanna write code

Have to do	Want to do
Write code	Write code
~~Build docker image~~	
~~Upload image to registry~~	
~~Deploy service~~	
~~Expose to the internet~~	
~~Setup logging & monitoring~~	
~~Scale workload~~	

Knative is extensible

Knative implements opinions

Some “opinions” may not be ideal for your use-case

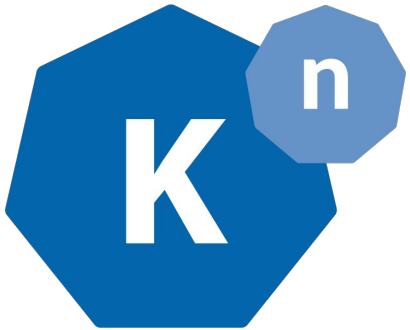
Knative API

- Event sources, event types
- Buildpack build templates
- Network configuration
- Logging targets

Kubernetes

- Auto-scaling strategy
- Function invokers
- Message bus

Knative is ready for you



Install, Samples, Docs
github.com/knative/docs

Serverless on GKE
g.co/serverlessaddon

Want to contribute?
knative/docs/community

Have questions?
knative.slack.com

Anything else?
@AikasVille
@mchmarny

NEXT'18

Thank you

Google Cloud