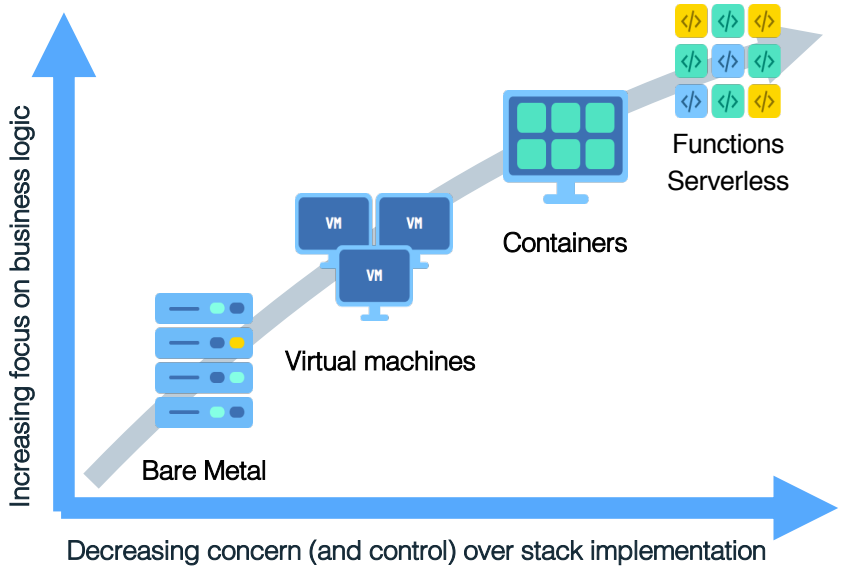
Stop calling Knative Serverless!

Doug Davis
IBM - STSM - OM Knative
dug@us.ibm.com | @duginabox

Why Cloud Native?

- Break-up the monolith
  - Better resource utilization
  - Reduced costs
- Abstraction of infrastructure
  - Devs focus on code not infrastructure
  - Faster time to market = $



Choices of infrastructure ...

- Platform as a Service — e.g. CloudFoundry
- Containers as a Service — e.g. Kubernetes
- Functions as a Service/Serverless — e.g. OpenWhisk

Which do you use?

Characteristics of Platform as a Service

	PaaS
Simplified UX	✓
Containers	✓
Micro-services / small-ish tasks	✓
Stateless	✓
Endpoint + Load Balancing	✓
Build	✓
Pay per usage (public cloud)	✓
On-demand infrastructure - auto-scaling	✓
Access to advanced features (eg. net/vol)	

Characteristics of C/F as a Service

	PaaS	CaaS	Fn/Srvr
Simplified UX	✓		✓
Containers	✓	✓	✓
Micro-services / small-ish tasks	✓	✓	✓
Stateless	✓	✓	✓
Endpoint + Load Balancing	✓	DIY	✓
Build	✓		✓
Pay per usage (public cloud)	✓	✓	✓
On-demand infrastructure - auto-scaling	✓	DIY	✓
Access to advanced features (eg. net/vol)		✓	
Event driven/tooling			✓
Scale to zero			✓
Async invocations			✓
Non-restrictive task execution times	✓	✓	
Non-restrictive resource usage (eg. mem)	✓	✓	

Why be forced to choose?

The need for something different...

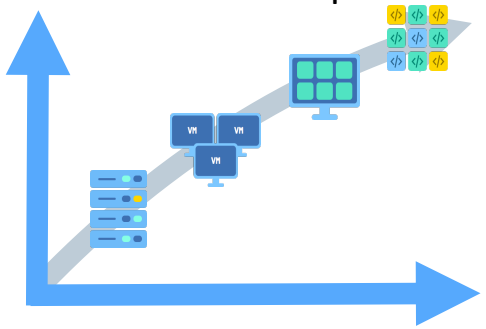
Kubernetes: Most popular Container Management project today.
Deploy and manage your containerized applications.

Resources
Containers
Pods
Replica Sets
Deployments
Services
Endpoints
Secrets
Networks
Volumes/PV/PVC
Ingress/LBs

Tooling & Other "stuff"
yaml
Spec vs Status
helm
kubectl
Istio
...

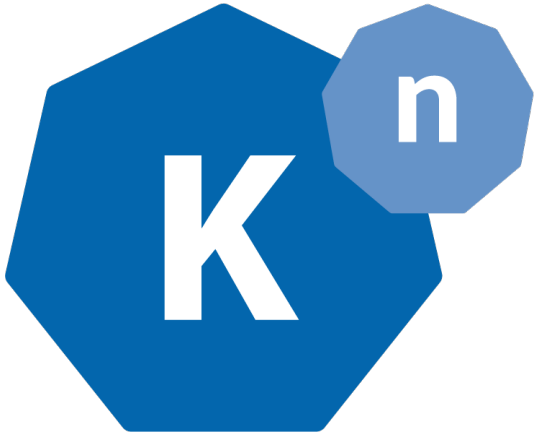
Blue/green deployments

The promise of Cloud Computing:
- Higher level abstractions
- Management of infrastructure for me
  - "Decreasing concern (and control) over infrastructure implementation"



I just want to deploy my code!

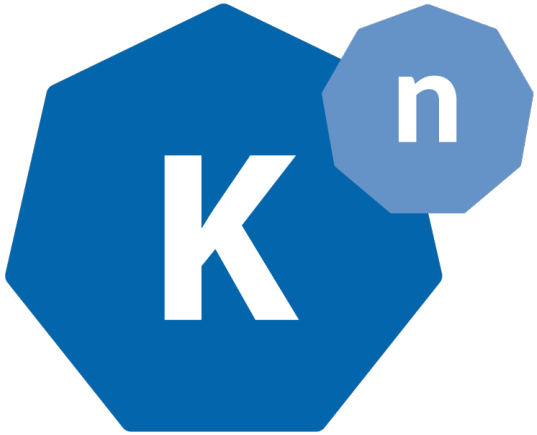
Introducing Knative



- An opinionated and simplified view of application/container management
- Allowing developers to focus on coding
  - Leverage abstractions & features from PaaS, Serverless
- Built as an extension to Kubernetes
  - Still Kubernetes under the covers
  - Access to full Kubernetes if needed
  - Integrates with the non-Knative workloads
- Building blocks on which Cloud Providers can build a platform

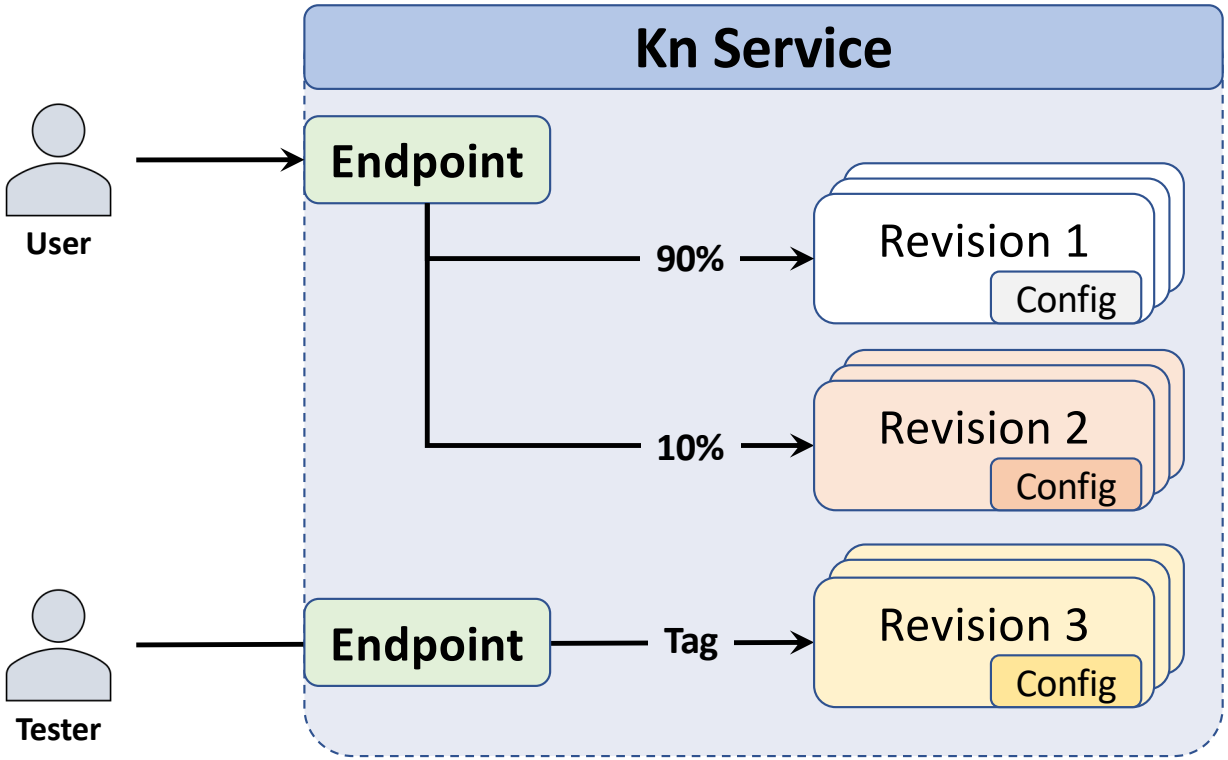
Knative

2 Main Components



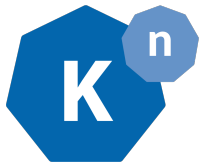
- **Serving** is the runtime component
  - Host your application as K8s pods
- **Eventing** contains tools for managing events
  - Between loosely coupled services

Knative Serving - The Model

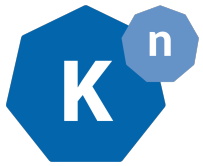


And Knative manages all of these resources for you!

- Deploy app as pod/revision
  - Revision specific config
  - E.g. image, env vars, scale
- Networking auto-setup
- Revisions are scaled up/down
  - Based on load
  - Even down to zero
- Updates create Revisions
  - Auto-migration to new
- Traffic splitting based on %
- Dedicated URLs to Revisions



Knative - Demo - The Core



Knative - Demo - The Core - Summary

- Deploy Service with just "name" and "image"
- Access it via HTTP and HTTPS
- Auto-scaled up and down (to zero) based on incoming load
- Rolling upgrade to new version of Service
- Traffic split (e.g. 50/50) between versions
- Dedicated URL to specific version
- Exclude specific version from traffic router

- What would it take to do all of this via vanilla Kubernetes?

Knative Serving - Things to know

- Container Images MUST run HTTP servers
- Multi-threaded model - but configurable
- Configuration options
  - Container Concurrency
  - Min / Max / Target Scale
  - Scaling: requests vs cpu
- Simplified resource model

```
apiVersion: serving.knative.dev/v1beta1
kind: Service
metadata:
  name: echo
spec:
 template:
    spec:
      containers:
        - image: duglin/echo
```

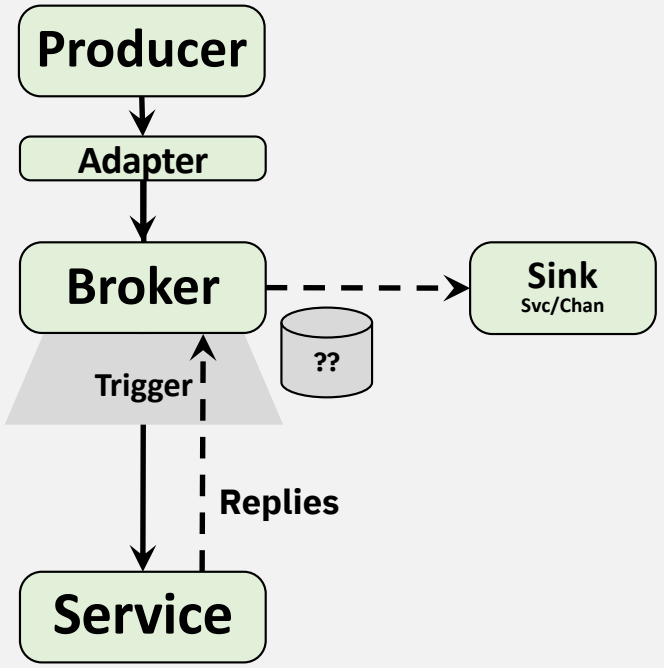
Knative Eventing - Core

- **Eventing Primitives**
  Manage the coordination/delivery of events

  - **Event Source** - connects Event Producer to "sink"
    - Create the subscription for you
    - Often, creates an Adapter (KnService) to receive the events and convert them into CloudEvents

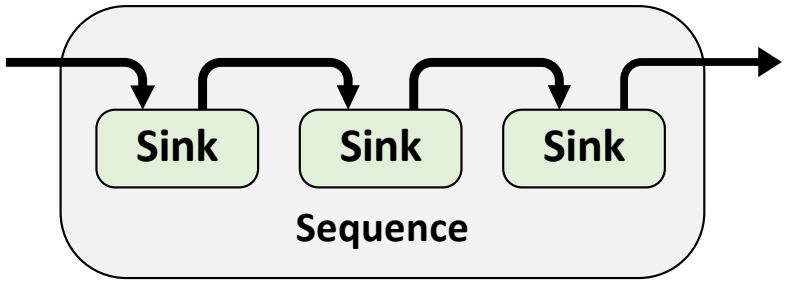
  - **Broker** - a receiver of events
    - E.g. a queue

  - **Trigger** - (subscription) ask for events from a Broker
    - Filters - to subset the stream of events
      Often based on CloudEvent's metadata

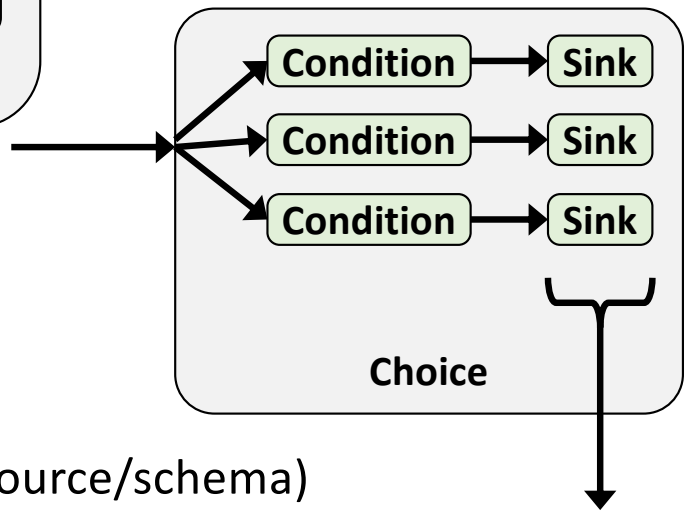


Knative Eventing - Other Features

- Sequence
  - Ordered set of "Sinks"
  - Final "Reply"



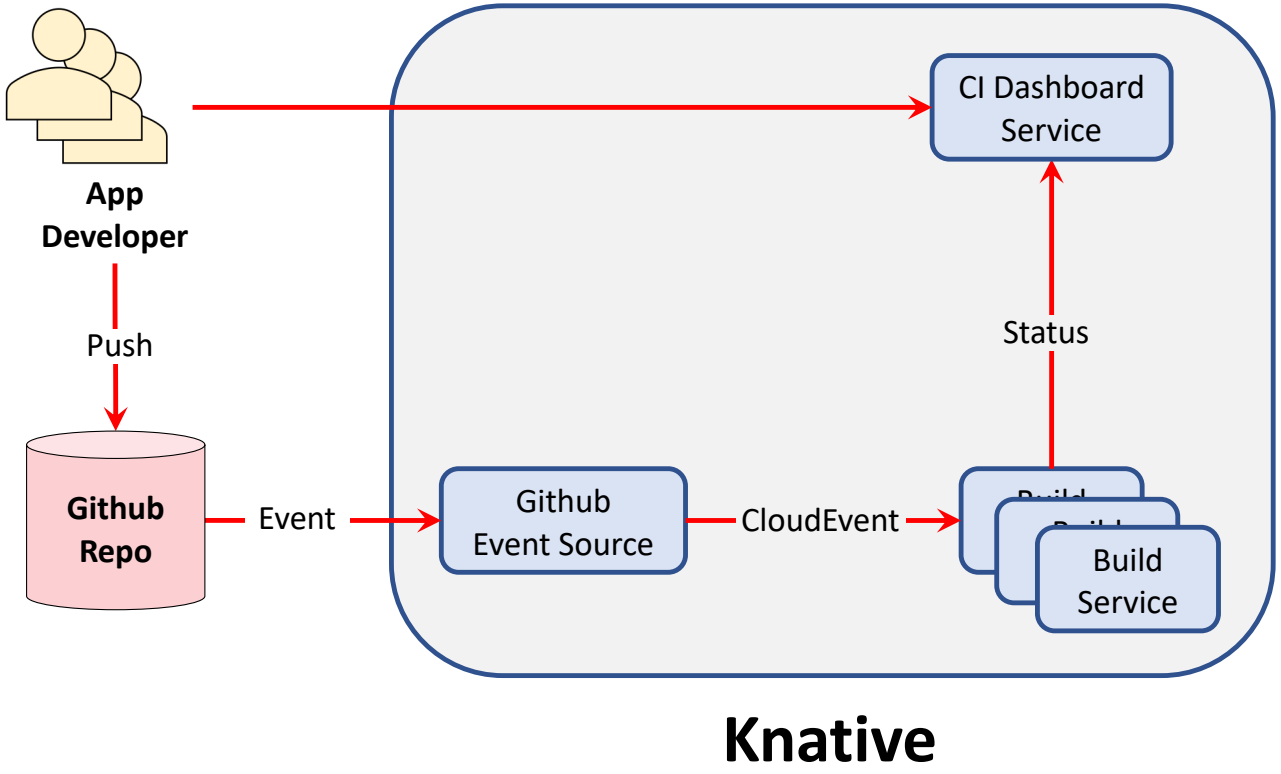
- Choice
  - Fan-out with filters/conditions



- Event Registry
  - Collection of EventTypes (Kn Broker, CloudEvent type/source/schema)

- CloudEvents - https://cloudevents.io
  - Regardless of Producer, Transport or Format - there's consistent metadata

Knative - Demo - Build Service



- Deploy CI Dashboard KnSvc
- Deploy a "Build" KnSvc
  - Concurrency = 1
- Create a Github Event Source
  - Notified of new Commits
  - Sink = Build KnSvc
- Dev commits & checks status
- Build Service scales
- Do it !

Characteristics of * as a Service

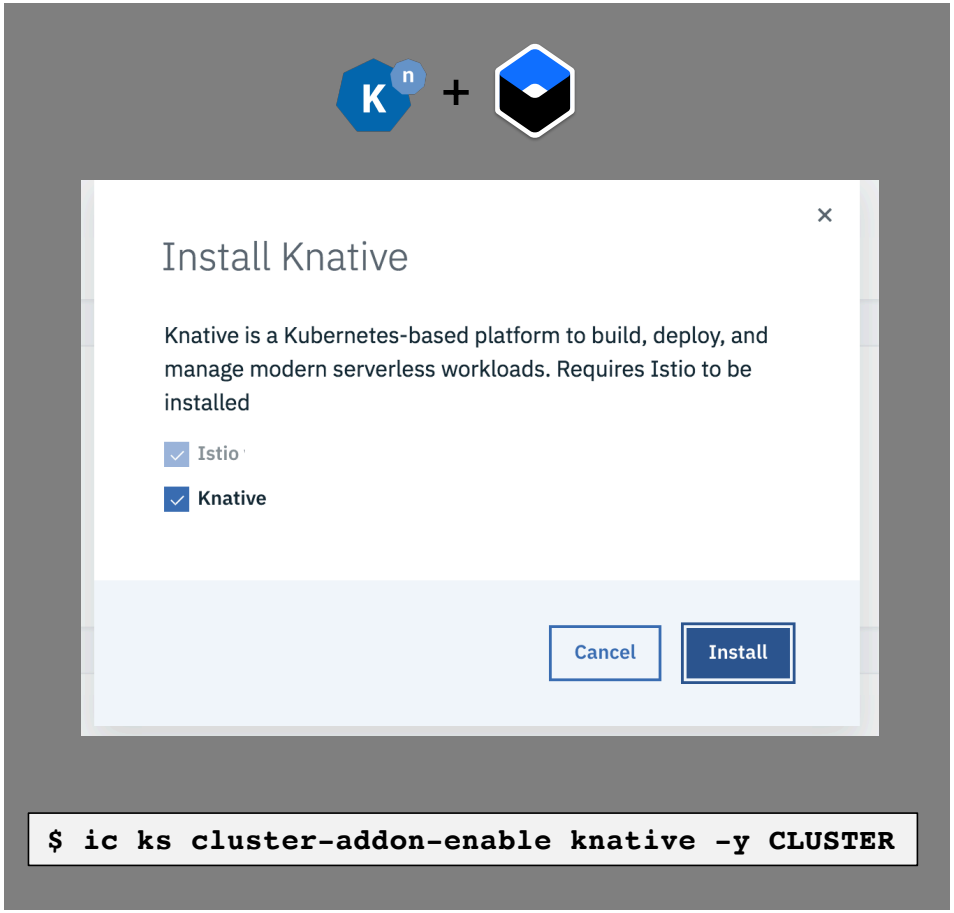
	PaaS	CaaS	Fn/Srvr	Knative
Simplified UX	✓		✓	✓
Containers	✓	✓	✓	✓
Micro-services / small-ish tasks	✓	✓	✓	✓
Stateless	✓	✓	✓	✓
Endpoint + Load Balancing	✓	DIY	✓	✓
Build	✓		✓	WIP
Pay per usage (public cloud)	✓	✓	✓	✓
On-demand infrastructure - auto-scaling	✓	DIY	✓	✓
Access to advanced features (eg. net/vol)		✓		WIP
Event driven/tooling			✓	✓
Scale to zero			✓	✓
Async invocations			✓	WIP
Non-restrictive task execution times	✓	✓		WIP
Non-restrictive resource usage (eg. mem)	✓	✓		✓

Knative - In Summary

- No longer have to choose which *aaS
- Get the best of
  - CaaS, PaaS, FaaS & Serverless
  - Simplified Kubernetes Experience
- Without losing the option of full Kubernetes when needed
- Your decisions: architecture of your application / containers
  - Size of your containers
  - Boundaries between them
  - Which bits to scale and when

IBM & Knative

- IBM involved in all parts of Knative
- Managed Knative add-on for IBM Cloud Kubernetes Service (IKS) - "experimental"
- One click install of Knative into your cluster
  - Includes Istio
- Updates to Knative are managed
- https://ibm.com/iks



Thank You!

Doug Davis
dug@us.ibm.com | @duginabox