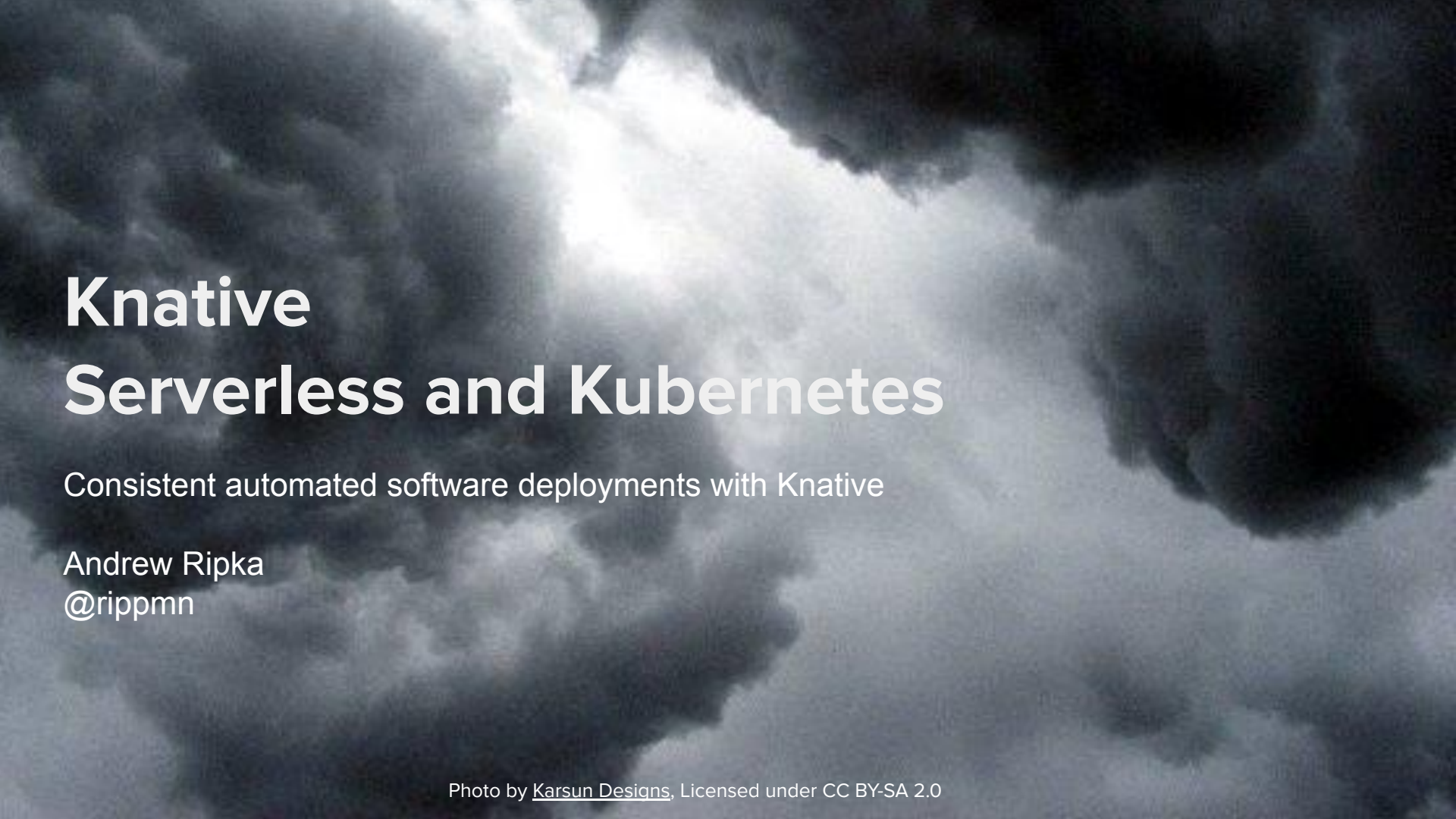
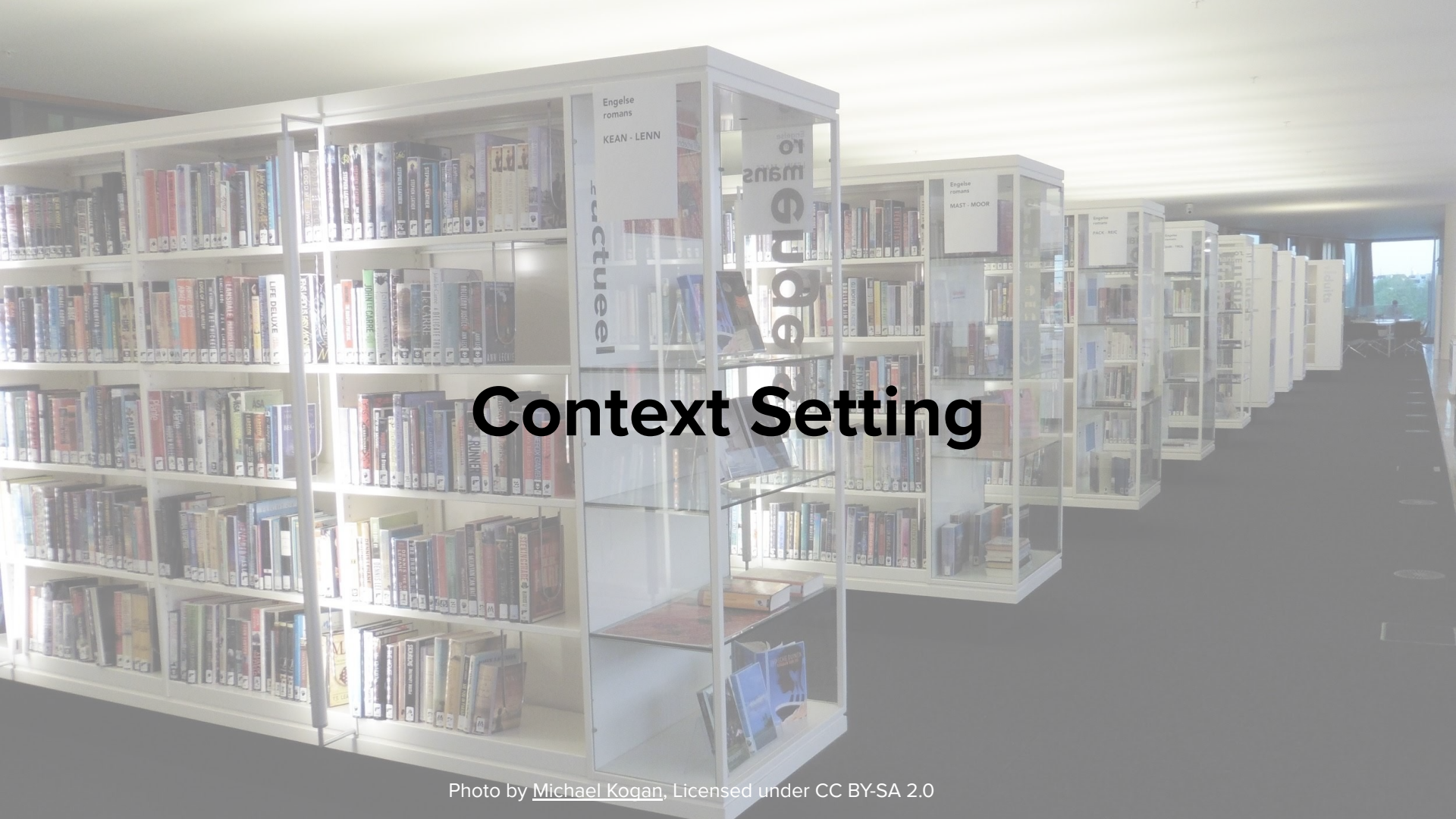


Knative
Serverless and Kubernetes

Consistent automated software deployments with Knative

Andrew Ripka
@rippmn

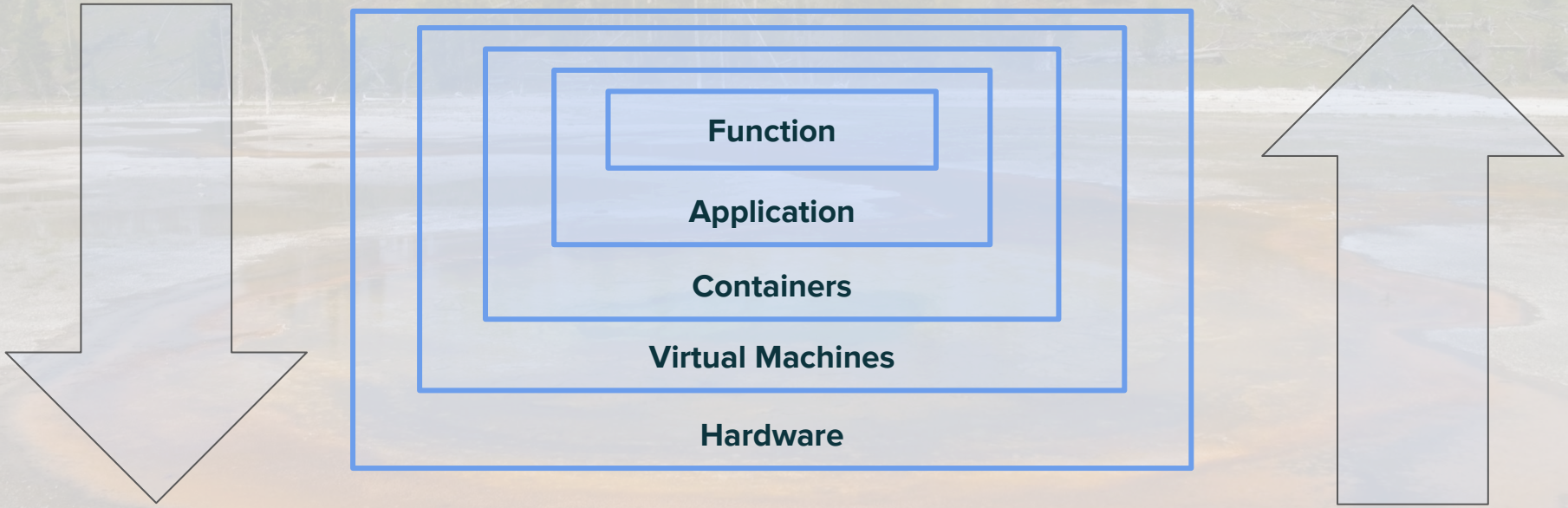


Context Setting

Platform Developer Abstractions

More Standards

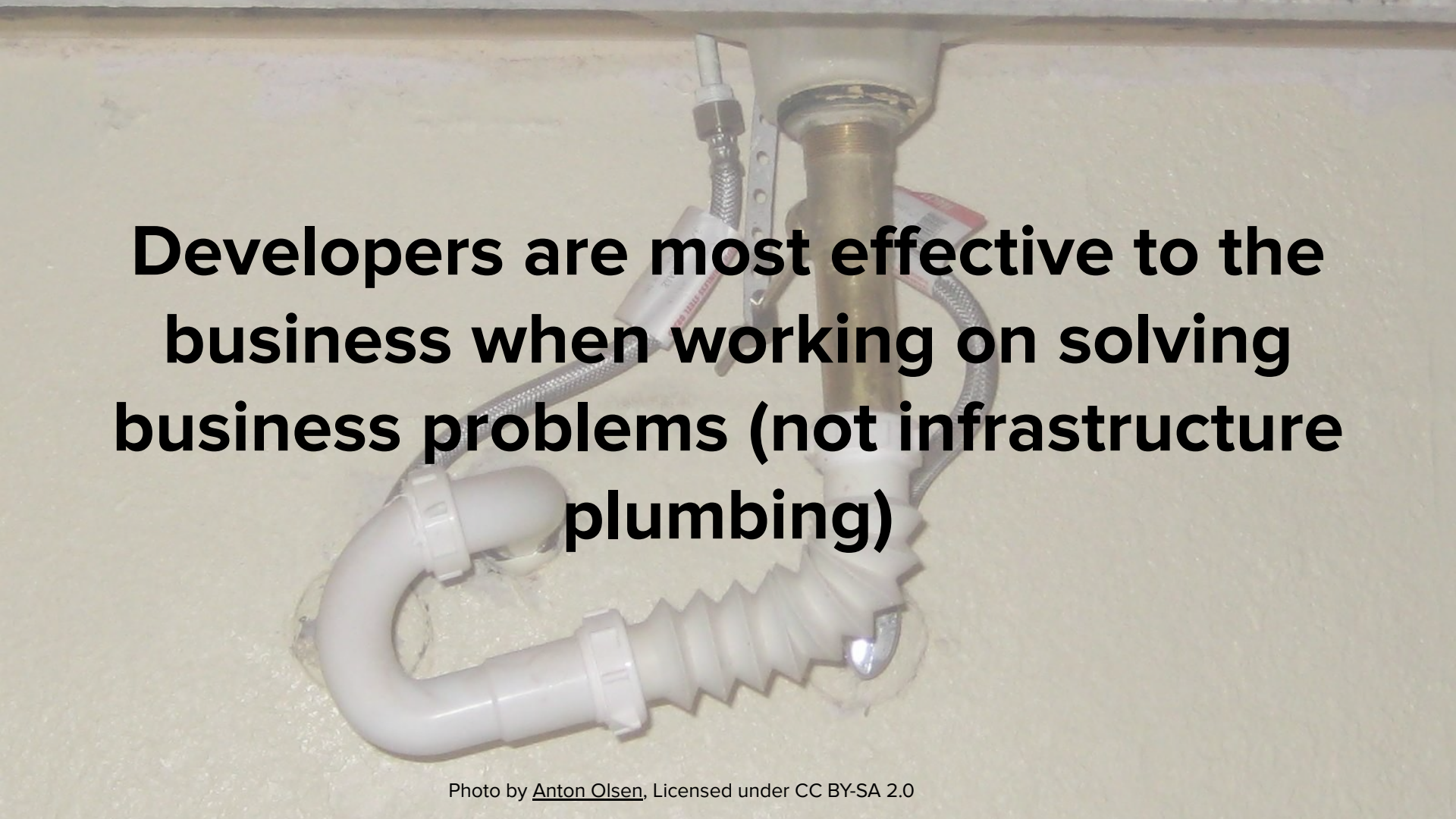
Higher Efficiency



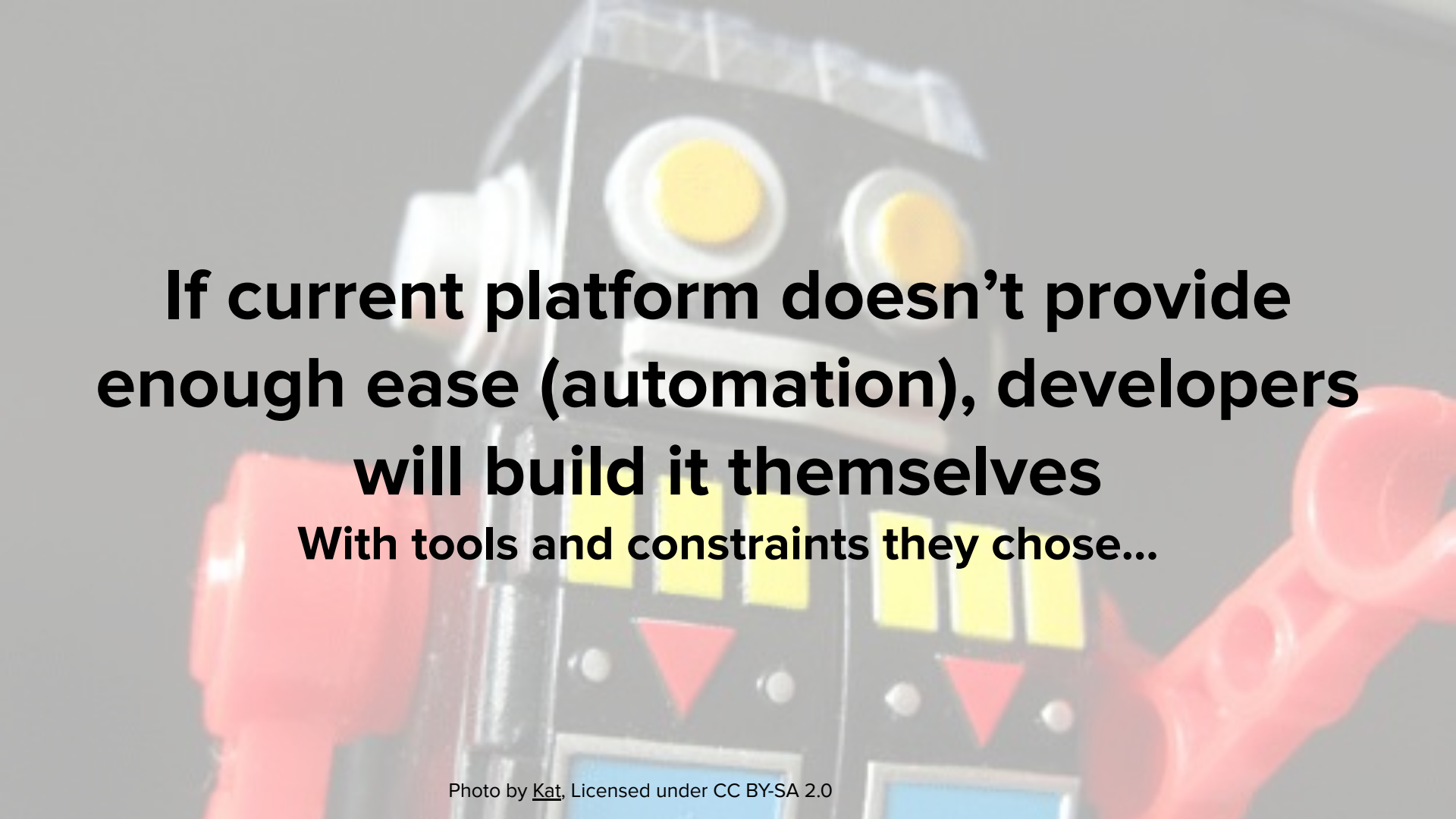
More Flexibility

What is Serverless?

- Frees devs from details that aren't part of the core business logic of their code. (e.g. Infrastructure)
- Small single purpose functions that are invoked in response to events
- Scale to 0 when no events occurring



Developers are most effective to the business when working on solving business problems (not infrastructure plumbing)



If current platform doesn’t provide enough ease (automation), developers will build it themselves

With tools and constraints they chose...

Ad hoc automation is awesome, until someone who wasn't involved gets paged.

(Like the operations team)

A platform's overall goal should be to make the right thing to do the easiest thing to do

(thus providing automation with desired constraints)

So you say you want a serverless platform

and you want to do it yourself...

Why not Kubernetes?

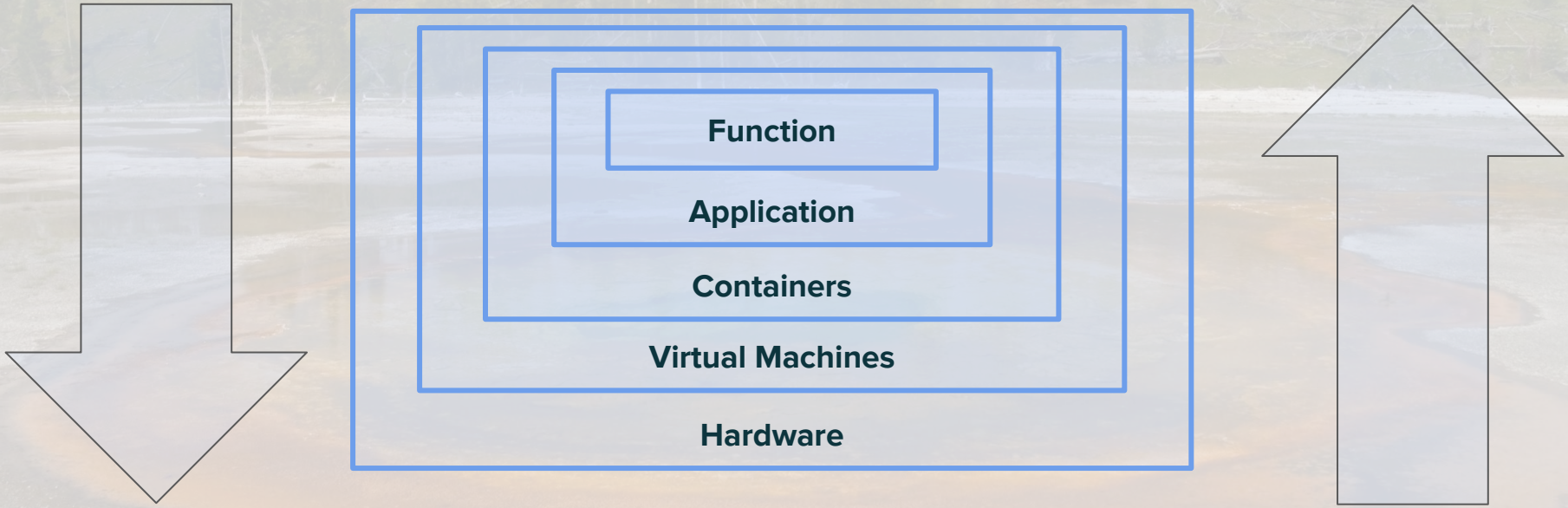
It’s building blocks

An infrastructure abstraction

Platform Developer Abstractions

More Standards

Higher Efficiency



More Flexibility

Serverless Platform Coarse Grained Requirements

- Package to deployable
- Deploy and Run
- Map traffic and events to running handlers

Serverless Platform

- Contains container creation and deployment process automation
- Allows scale to zero and back up when necessary
- Handling of revisions with rollback with traffic splitting
- An eventing system with configurable sources/flows/subscribers
- Something to collect metrics and export telemetry from the app.

Knative

What is Knative is not

- It’s not a fully featured Function as a Service platform
- It’s intended to be used by others to build FaaS or PaaS products

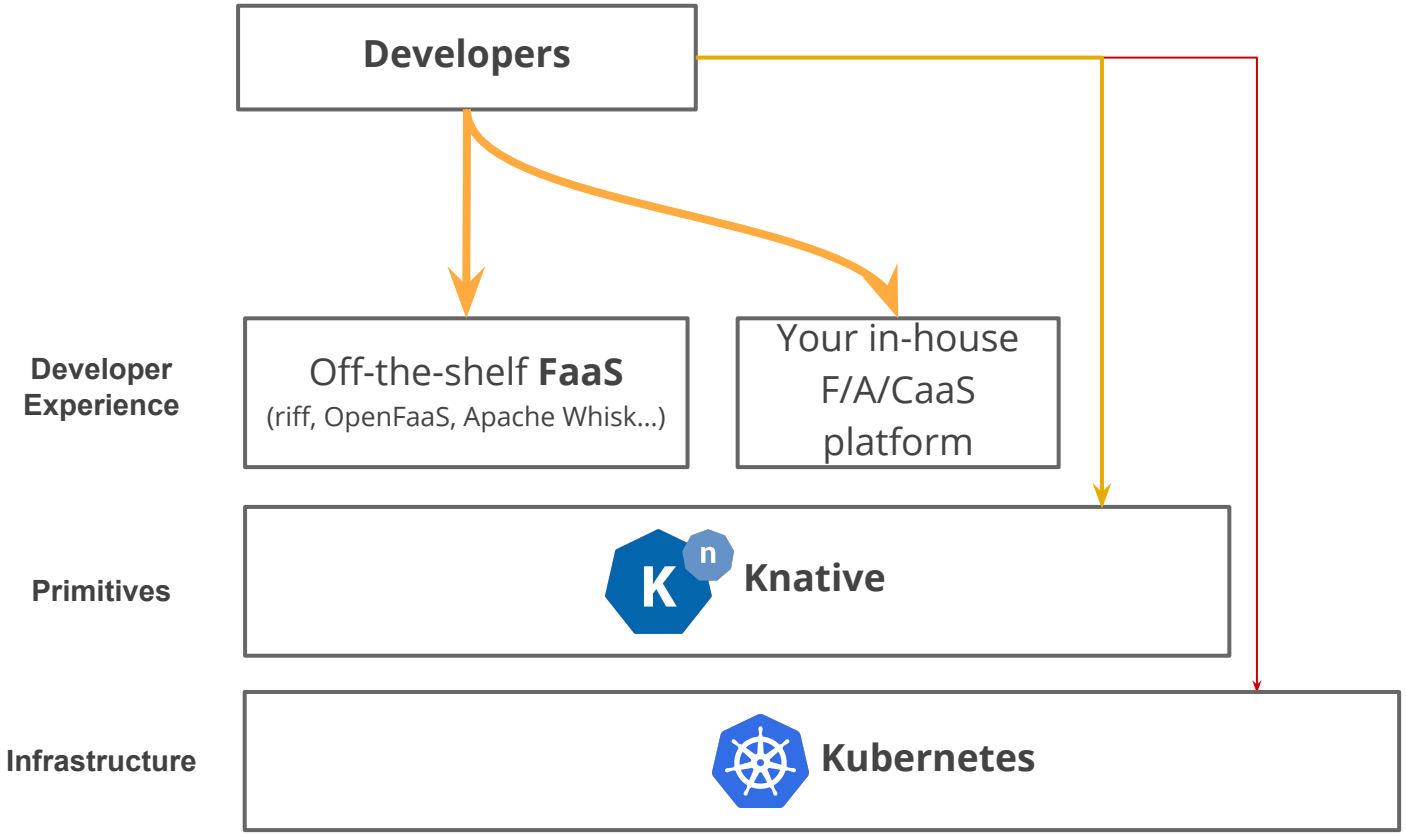
What is Knative

- An abstraction on top of Kubernetes
  - Deploys and runs containers
- Set of building blocks to construct your own FaaS/AaaS/CaaS
  - abstracts common tasks through custom Kubernetes API objects
- Set of middleware components essential to build modern, source-centric, and container-based applications that can run anywhere.

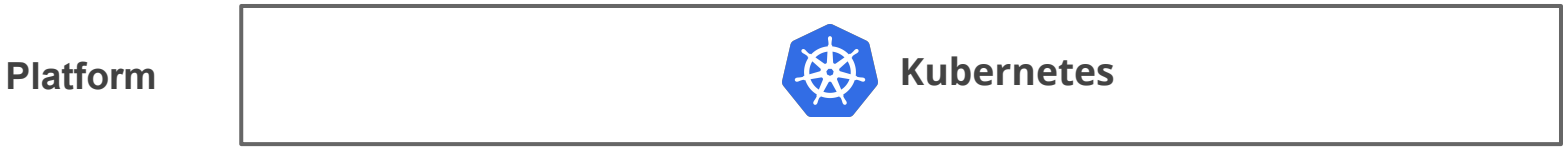
Knative Codified Patterns

- Deploying a container
- Orchestrating source-to-URL workflows on Kubernetes
- Routing and managing traffic with blue/green deployment
- Scaling automatically and sizing workloads based on demand
- Binding running services to eventing ecosystems

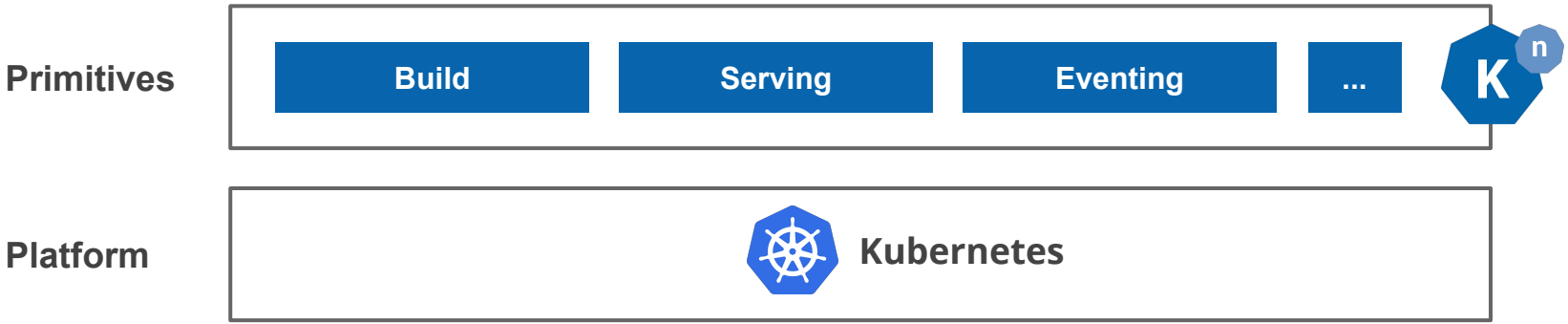
Expected usage



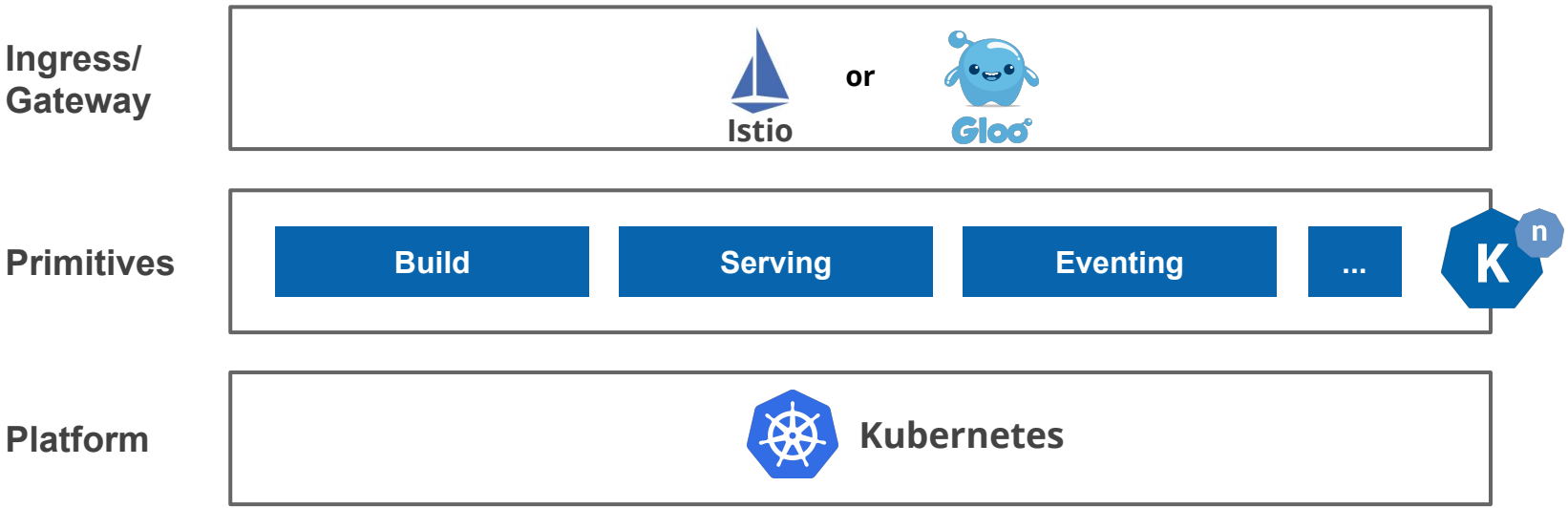
Knative architecture



Knative architecture



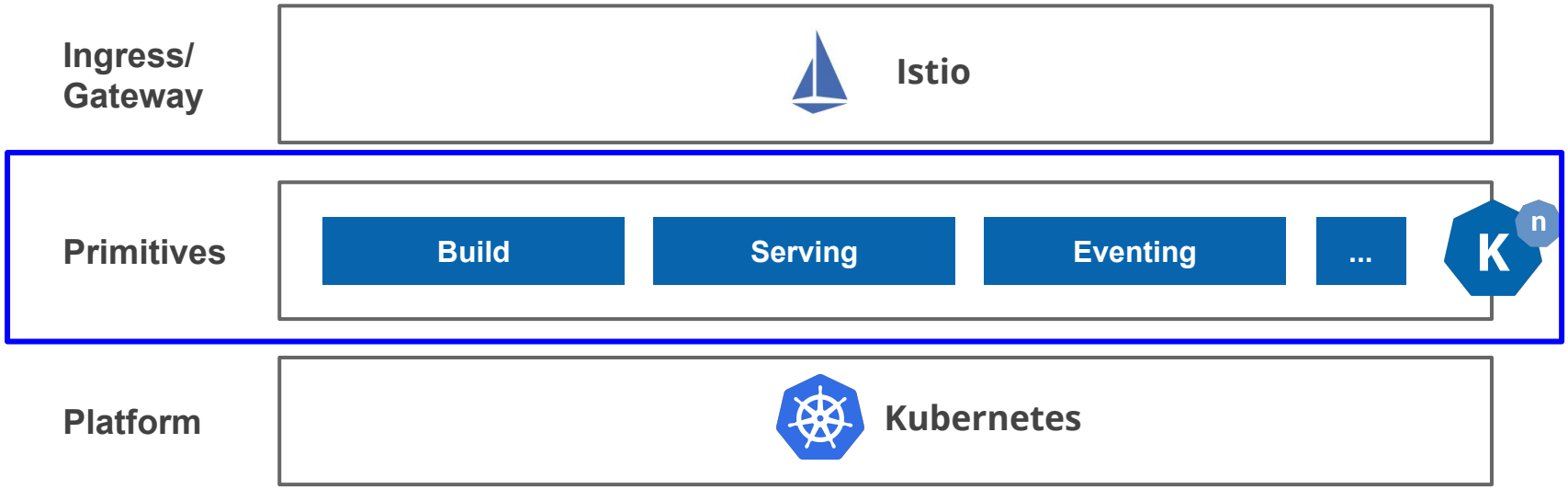
Knative architecture



Audiences


Users and Systems (IoT) use applications developed and deployed by developers and hosted by operators
Istio
Knative
Kubernetes
Developers build and deploy apps
API
GitHub
Contributors develop and contribute code and docs to the OSS project
Operators deploy and manage Knative instances using Kubernetes API and tools
Platform Providers (such as Google Cloud Platform) provide underlying infrastructure

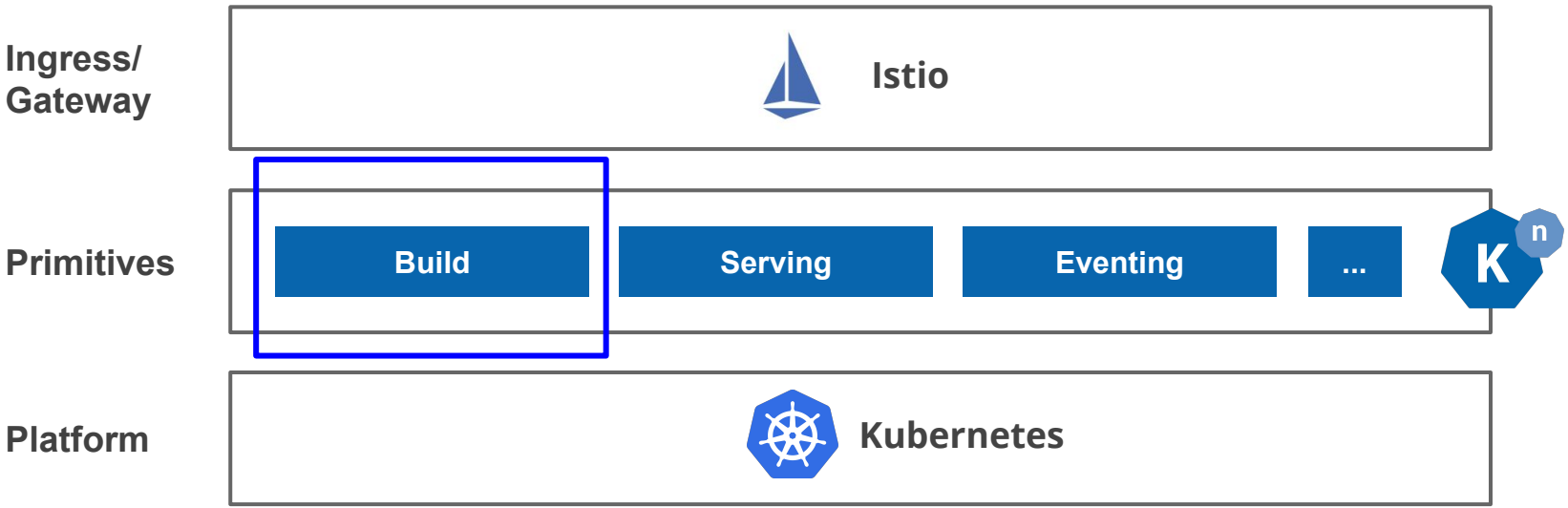
Knative Components



Knative Components

- **Build** - Source-to-container build orchestration
- **Serving** - Request-driven serverless compute using containers with revisions and scale to zero
- **Eventing** - Declarative management and delivery of events

Knative Components



Build

- Defines process that runs to completion and can provide status.
- Build container image from source that can be deployed to Knative Serving
- Uses Kubernetes Service Account to connect to secured resources

Build Components

- **Build** is made up of Steps containing builders
- **Builder** - container image that accomplishes a task (discrete or composite)
  - One of the steps typically push to container registry
- **Source** - data necessary for build in mounted volume
  - GCS
  - Git
  - Custom (container image)

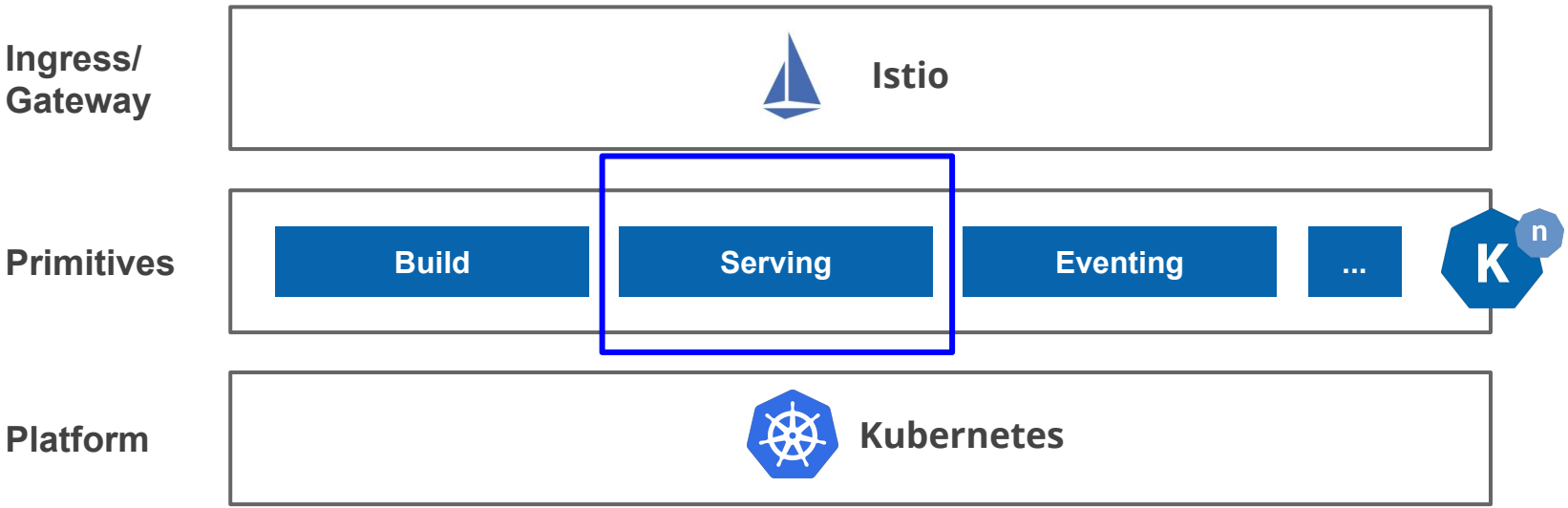
Build Components (Cont.)

- **Service Account** - privileges to run build with
- **Volumes** - non default volumes to use during build steps
- **Timeout** - max time for resource allocation and build steps

Build Templates

- Encapsulate build process for reuse and consistency
  - limited parameterization
- Similar to “compile” phase of Heroku and Cloud Foundry buildpacks
- Limited to namespace but can be shared across cluster as Cluster Build Template

Knative Components

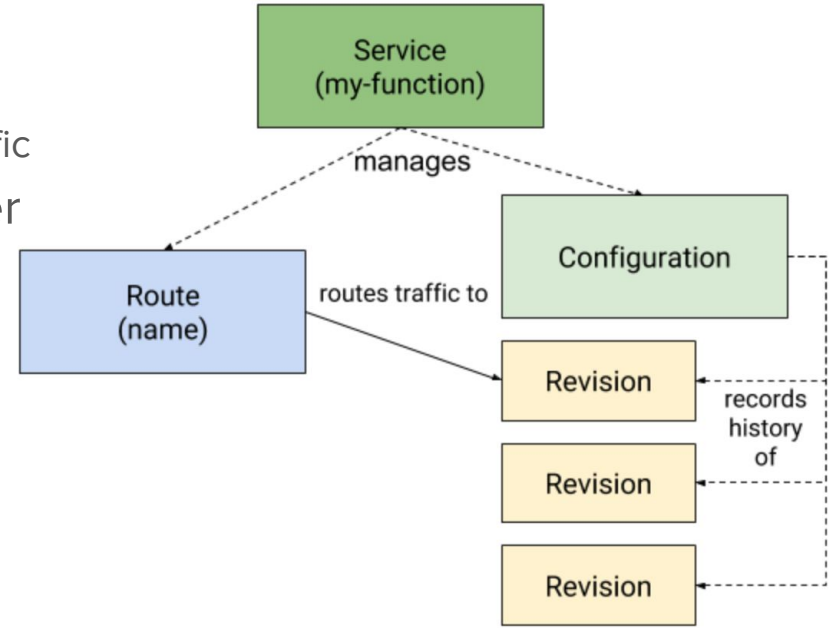


Serving

- Rapid deployment of serverless containers
- Automatic scaling up and down to zero
- Routing and network programming for ingress gateway components (Istio or Gloo)
- Point-in-time snapshots of deployed code and configurations (enabling rollback)
- Provide observability through logs and metrics

Serving Components

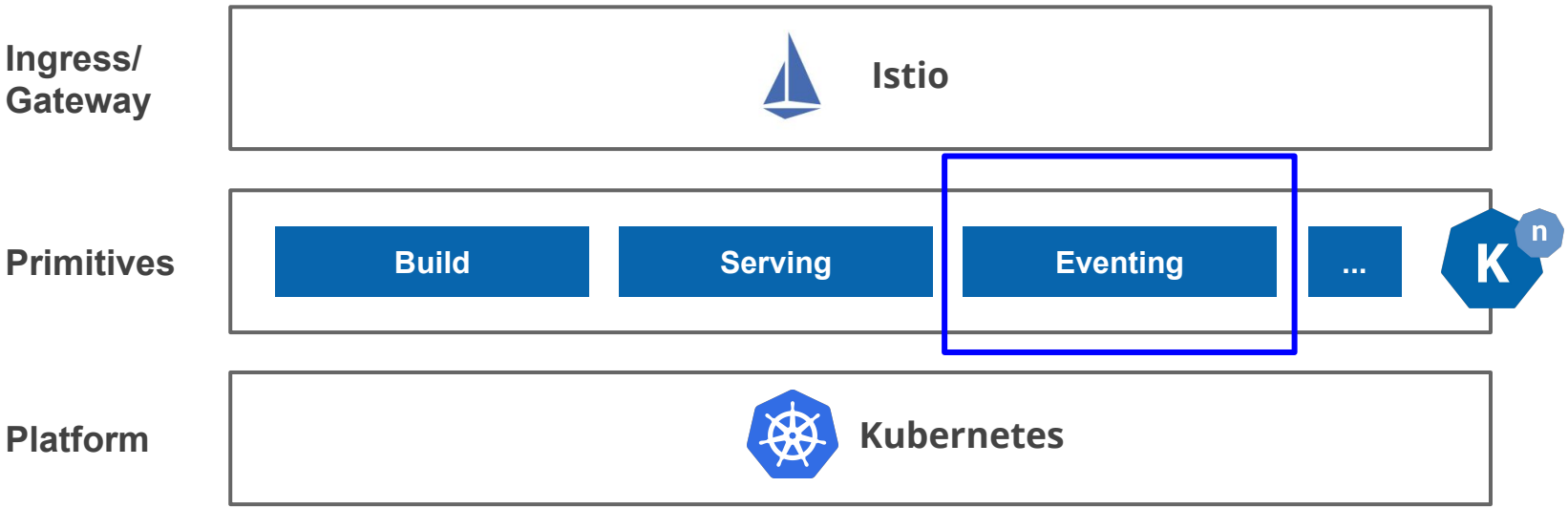
- **Route** map Http routable endpoint to revision(s)
  - Allows for percentage based splitting of traffic
- **Revision** - immutable code (as container or build) and config
  - Generated by configuration changes
  - Retried no route mapped to (resources deleted) but saved for rollback
- **Configuration** - Latest revision of code and config
- **Service** - Set of routes and configurations
  - Single abstraction of software component



Logging And Metrics

- Knative collects logs using Fluentd
- Plugable logging observability implementations
  - Elasticsearch and Kibana
  - Stackdriver
  - Custom logging plugin
- Provides Grafana metrics observability

Knative Components

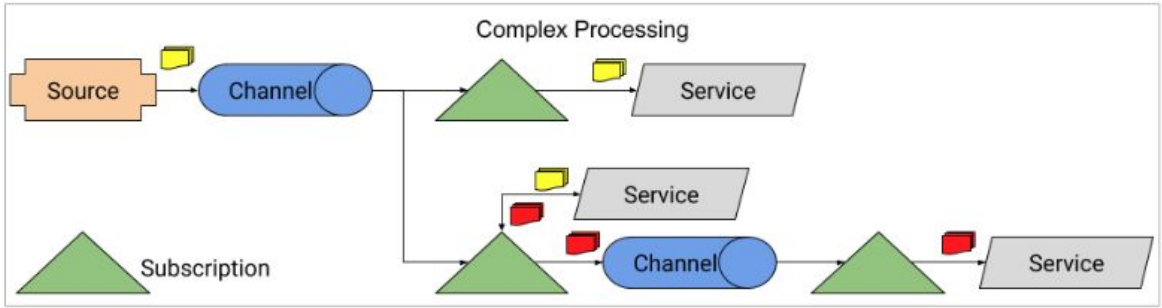


Eventing

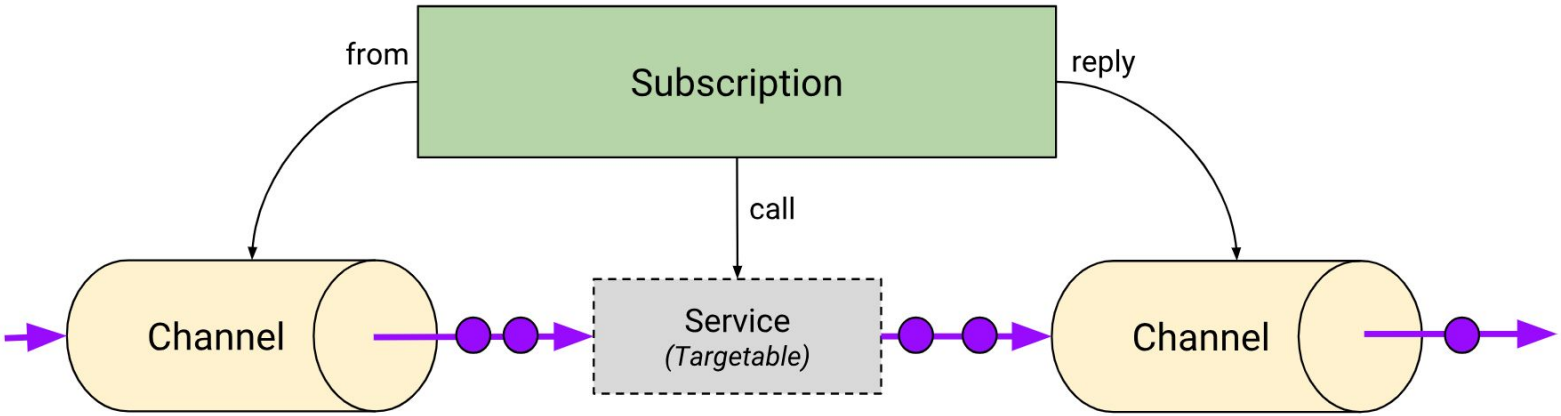
- Composable primitives enabling late-binding loosely coupled event producers and event consumers
- Consumers can be other Services allowed to:
  - Create new applications without modifying the event producer or event consumer.
  - Select and target specific subsets of the events from their producers.
- Consistent with CloudEvents specification developed by CNCF Serverless WG

Eventing Components (Legacy)

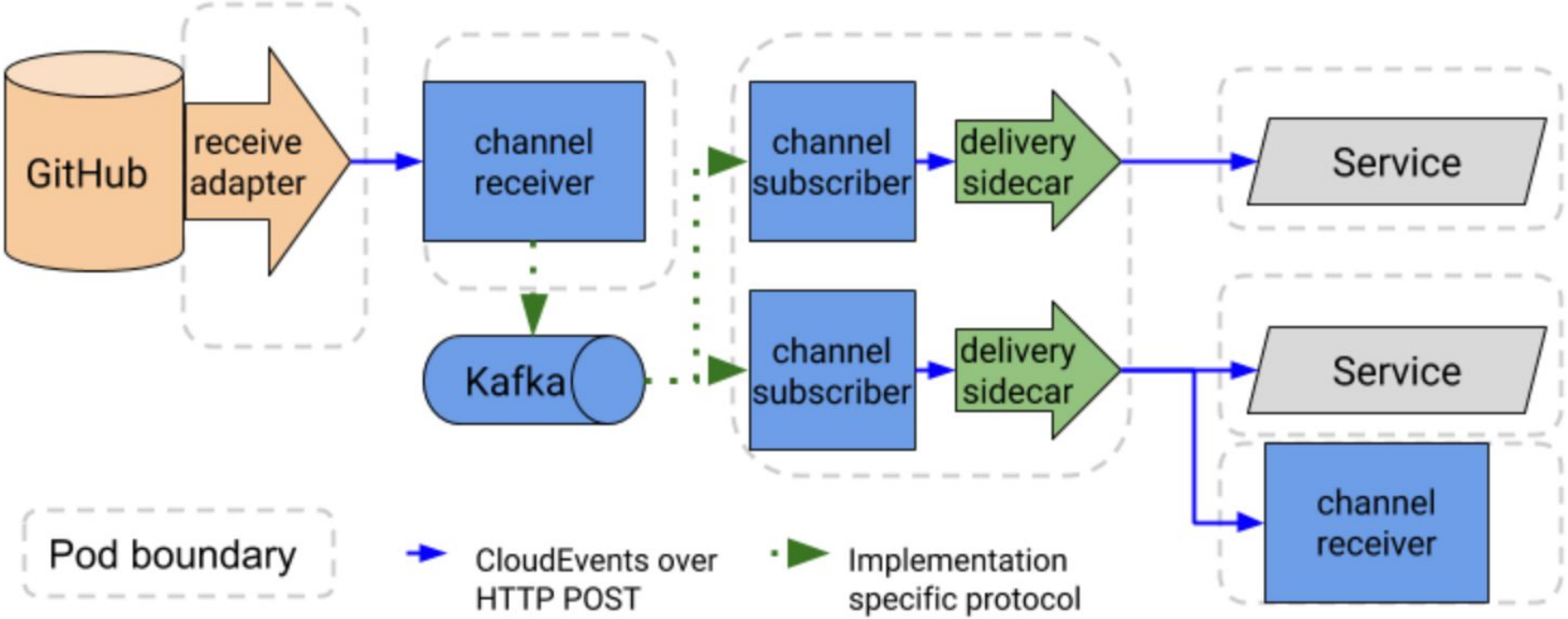
- **Source -** Register interest in events from other systems and publish to a sink
  - K8s events, Kafka, GitHub, GitLab, BitBucket, custom sources
- **Service** - Consumer of events
  - Addressable acknowledge delivery
  - Callable - return 0 or 1 events
- **Channel** - event forwarding persistence
  - Plugable ClusterChannelProvisioner implementation for message providers
- **Subscription -** means to declare interest in events from a channel



Channels & Subscriptions

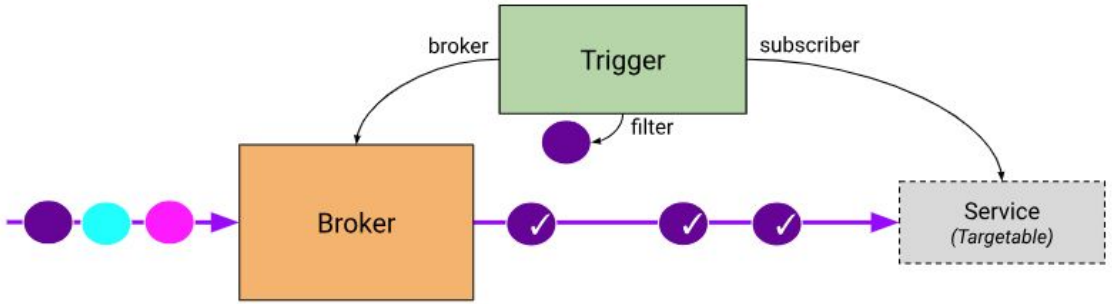


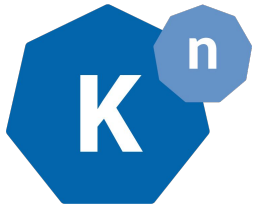
Data Plane


GitHub
receive adapter
channel receiver
Kafka
channel subscriber
channel subscriber
delivery sidecar
delivery sidecar
Service
Service
channel receiver
Pod boundary
CloudEvents over HTTP POST
Implementation specific protocol

Eventing Components (New as of Knative 0.5)

- **Broker** - mesh of events sent to interested subscribers
  - Use a channel template to create internal channels
- **Trigger** - declares subscription to events from a broker
  - Can be used to filter events based upon type





Config Samples

Config Sample Links

- Build
  - Build - https://github.com/GoogleCloudPlatform/knative-build-tutorials/blob/master/docker-build/README.md
  - Build Template - https://github.com/knative/build-templates/blob/master/kaniko/kaniko.yaml
  - Build with template - https://github.com/knative/build-templates/tree/master/kaniko
- Serving
  - Deploy and routing examples - https://github.com/knative/docs/blob/master/docs/serving/samples/blue-green-deployment.md
- Eventing
  - Channel - https://github.com/gswk/knative-eventing-demo/blob/master/channel.yaml
  - Subscription - https://github.com/gswk/knative-eventing-demo/blob/master/subscription.yaml
  - Broker - https://github.com/knative/eventing/tree/master/docs/broker
  - Trigger - https://github.com/knative/docs/blob/master/docs/eventing/samples/gcp-pubsub-source/trigger.yaml

Remember these are primitives to build PaaS/FaaS

Where to Go to learn more

- http://knative.dev
- https://github.com/knative/
- https://starkandwayne.com/blog/introduction-to-knative/
- https://medium.com/@pczarkowski/introduction-to-knative-b93a0b9aeeef
- https://www.ibm.com/cloud/learn/knative
- https://cloud.google.com/knative/