KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Knative Contribfest!

What is Knative?

Knative aims to provide a consistent, portable serverless experience across different environments

- Goal is to be “Kubernetes-native”
- A gentle introduction to Kubernetes for developers
- Managed by operators the same way they manage Kubernetes today

Knative is modular. Consume what you want, don’t need to adopt the rest.

- The modules work well together. (“Voltron effect”)
- And you may get synergy by adopting several together.

High Level Knative Components

Independent components that work well together

Serving	Eventing	Functions
Automatic HTTP Services	CloudEvents over HTTP	Simple Code to Container
Container → URL	Declarative Event-Driven Architecture	Make HTTP, CloudEvents services simple to write

Pluggable Implementations

Knative builds on underlying technologies, which are replaceable
Integrate with your existing stack

Serving

Istio

CONTOUR

gateway-api.sigs.k8s.io

Eventing

kafka

RabbitMQ

NATS

Functions

Buildpacks.io

Java

node JS

python

GO

How the Repos are Structured

`knative` (core)	`knative-sandbox` (extensions)
Base interfaces	Serving routing plugins
Core released modules	Eventing transport plugins
CLI	Event source plugins
Test infrastructure	CLI plugins
Documentation	Experiments!
Community organization	

Introduction to the Tools

Tools For Your Machine

We use a fairly consistent set of tools:

- `golang` (at least 1.18)
- IDE of your choice (VSCode?)
- `kubectl`
- `ko`: https://github.com/ko-build/ko
- `bash` – Macs need bash 5, not bash 2 (get it from brew)

You shouldn't need Docker on your local machine!

Windows – you may need WSL; some unit tests may not pass (but PRs to fix welcomed!)

Development Clusters

Two options:

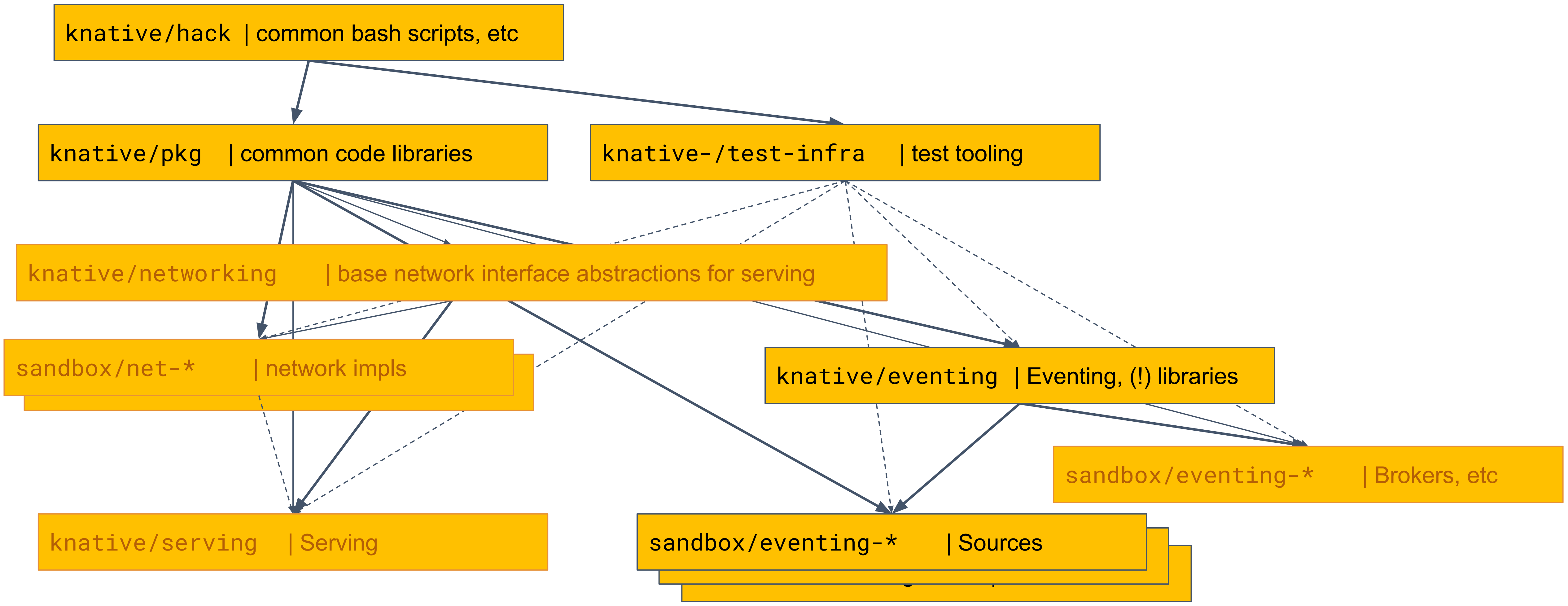
Use a cloud cluster, with cloud provider Kubernetes

Need a real repo, etc to push containers

Use a local Kind or Minikube cluster

For kind, use `KO_DOCKER_REPO=kind.local`

Repo dependencies – NOT A MONOREPO!


knative/hack | common bash scripts, etc
knative/pkg | common code libraries
knative-/test-infra | test tooling
knative/networking | base network interface abstractions for serving
sandbox/net-* | network impls
knative/eventing | Eventing, (!) libraries
sandbox/eventing-* | Brokers, etc
knative/serving | Serving
sandbox/eventing-* | Sources

What's in a repo?

Well-known bash scripts:

- `/hack`
  - `update-deps.sh` → codegen, go mod/vendor checks, checksums, schema gen, etc
- `/test`:
  - e2e-tests.sh → integration tests

Automation:

- Robots to update deps, cleanup behind you (`knative-sandbox/knobots`)
- Presubmits to run tests, check lint / fmt / codegen steps
- Prow / Tide for merge requirements
- GitHub Actions for additional checks that don't require a multi-node cluster

…

whew

that’s a lot.

So, when do we get started?

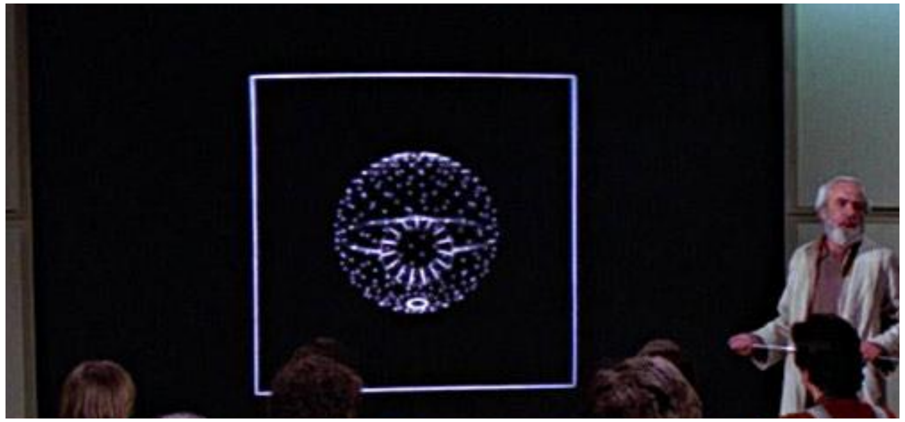
Eventing Sources

Our target – Eventing Sources

Sources are pre-built ways of collecting events

Rather than writing hundreds of lines of code

… write a dozen lines of yaml!



Assume that the eventing source is as good as the code you'd write.

```java
package com.example.messagingrabbitmq;

import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.core.TopicExchange;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer;
import org.springframework.amqp.rabbit.listener.adapter.MessageListenerAdapter;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class MessagingRabbitmqApplication {

  static final String topicExchangeName = "sample-exchange";

  static final String queueName = "spring-boot";

  @Bean
  Queue queue() {
    return new Queue(queueName, false);
  }

  @Bean
  TopicExchange exchange() {
    return new TopicExchange(topicExchangeName);
  }

  @Bean
  Binding binding(Queue queue, TopicExchange exchange) {
    return BindingBuilder.bind(queue).to(exchange).with("foo.bar.#");
  }

  @Bean
  SimpleMessageListenerContainer container(ConnectionFactory connectionFactory,
      MessageListenerAdapter listenerAdapter) {
    SimpleMessageListenerContainer container = new SimpleMessageListenerContainer();
    container.setConnectionFactory(connectionFactory);
    container.setQueueNames(queueName);
    container.setMessageListener(listenerAdapter);
    return container;
  }

  @Bean
  MessageListenerAdapter listenerAdapter(Receiver receiver) {
    return new MessageListenerAdapter(receiver, "receiveMessage");
  }

  public static void main(String[] args) throws InterruptedException {
    SpringApplication.run(MessagingRabbitmqApplication.class, args).close();
  }

}
```

```yaml
apiVersion: sources.knative.dev/v1alpha1
kind: RabbitmqSource
metadata:
  name: rabbitmq-source
spec:
  rabbitmqClusterReference:
    name: rabbitmq-default-user
    namespace: default
  rabbitmqResourcesConfig:
    exchangeName: "sample-exchange"
    queueName: "spring-boot"
  sink:
    ref:
      apiVersion: eventing.knative.dev/v1
      kind: Broker
      name: default
```

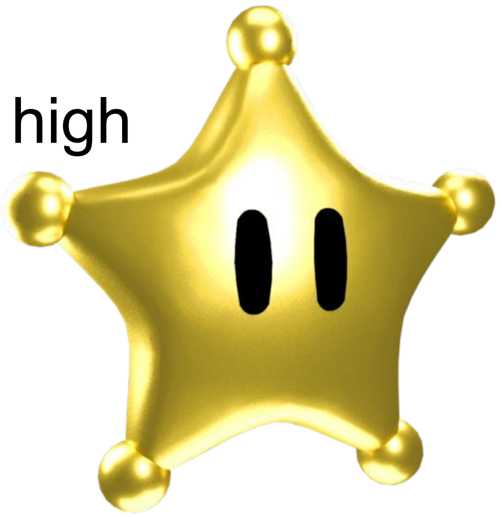
Also: polyglot, instrumented…

What does good look like?

Assume your destination is a Broker or Channel that implements high availability.

Check:

- Test coverage (from `go cover test/...`)
- DEVELOPMENT.md – does it exist? Is it useful?
- Is the Source multi-tenant? This can be nice, but isn't necessary



Observability	
Metrics	Events received and sizes Deliveries succeed / failed / latency
Tracing	Are traces supported? Can they pass through from the upstream?
Logs	Generally less of an issue Developers might not have access

Documentation	
Tutorial	Install the Source Create an instance, receive an event
How-To	Describe configuration parameters
Reference	What are all the CRD fields What do events look like (schema / example)

Your mission, should you choose to accept it…

Organize by tables, and tackle one of the following repos:

```
knative-sandbox/eventing-couchdb
knative-sandbox/eventing-github
knative-sandbox/eventing-gitlab
knative-sandbox/eventing-kogito
knative-sandbox/eventing-redis
```

Additional sources if we have time / tables:

ApiServer	Camel	Kafka	PingSource / Heartbeats	RabbitMQ	WebSocket

Whoof, that's a lot!

We know!

What does success look like?

- We updated the DEVELOPMENT.md and added 20% test coverage.
- We got an integration test to work!
- We wrote some documentation, and put it under /docs or README.md
  - (And maybe updated the https://knative.dev/docs/eventing/sources link!)
- We struggled for an hour, and don't think this works. We updated the README to warn folks, and recommended archiving the repo.

Let’s Get to Work!

(information carousel follows)

Tools For Your Machine

We use a fairly consistent set of tools:

- `golang` (at least 1.18)
- IDE of your choice (VSCode?)
- `kubectl`
- `ko`: https://github.com/ko-build/ko
- `bash` – Macs need bash 5, not bash 2 (get it from brew)

You shouldn't need Docker on your local machine!

Windows – you may need WSL; some unit tests may not pass (but PRs to fix welcomed!)

Your mission, should you choose to accept it...

KubeCon CloudNativeCon North America 2022

Organize by tables, and tackle one of the following repos:

```
knative-sandbox/eventing-couchdb
knative-sandbox/eventing-github
knative-sandbox/eventing-gitlab
knative-sandbox/eventing-kogito
knative-sandbox/eventing-redis
```

Need help?
https://slack.knative.dev/
#knativecon

Additional sources if we have time / tables:

ApiServer	Camel	Kafka	PingSource / Heartbeats	RabbitMQ	WebSocket

ContribFest Hosts

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

October 24-28, 2022

Mahamed Ali
Customer Reliability Engineer
Rackspace

Evan Anderson
Software Engineer
VMware

Paul Schweigert
Software Engineer
IBM

https://slack.knative.dev/

#knativecon