Cloud Agnostic Serverless with Knative

Going Serverless anywhere on Kubernetes

TRIGGERMESH
SERVERLESS MANAGEMENT PLATFORM

GitLab

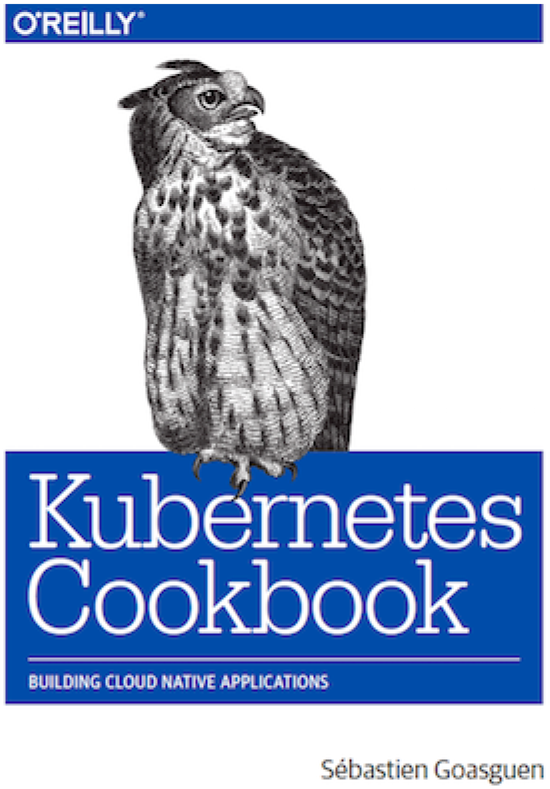
By

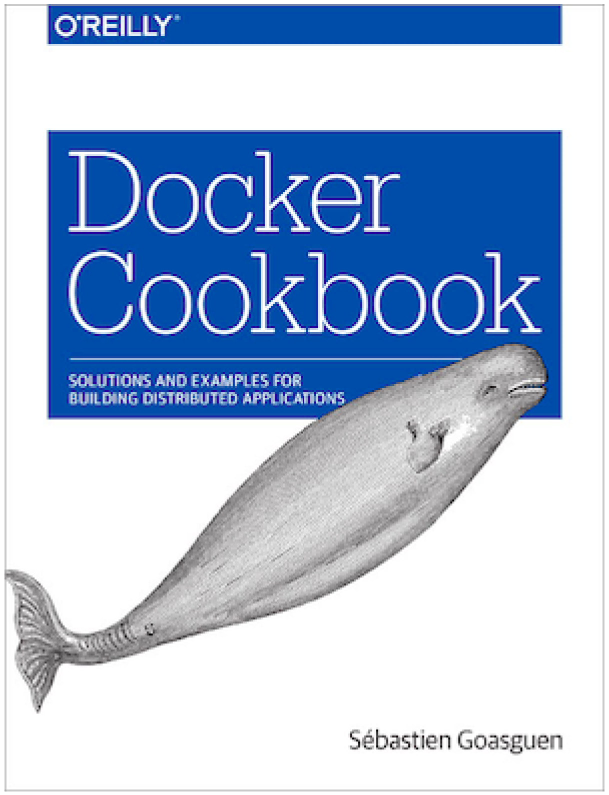
- **@sebgoa** (Sebastien Goasguen)
- **@eggshellcullen** (Cullen Taylor)
- **@cab105** (Chris Baumbauer)
- **@pritianka** (Priyanka Sharma)
- And the nice helpers from GitLab ...

By who ?

Kompose, Kmachine, kubeless, Cabin, TriggerMesh ...

@triggermesh https://github.com/triggermesh

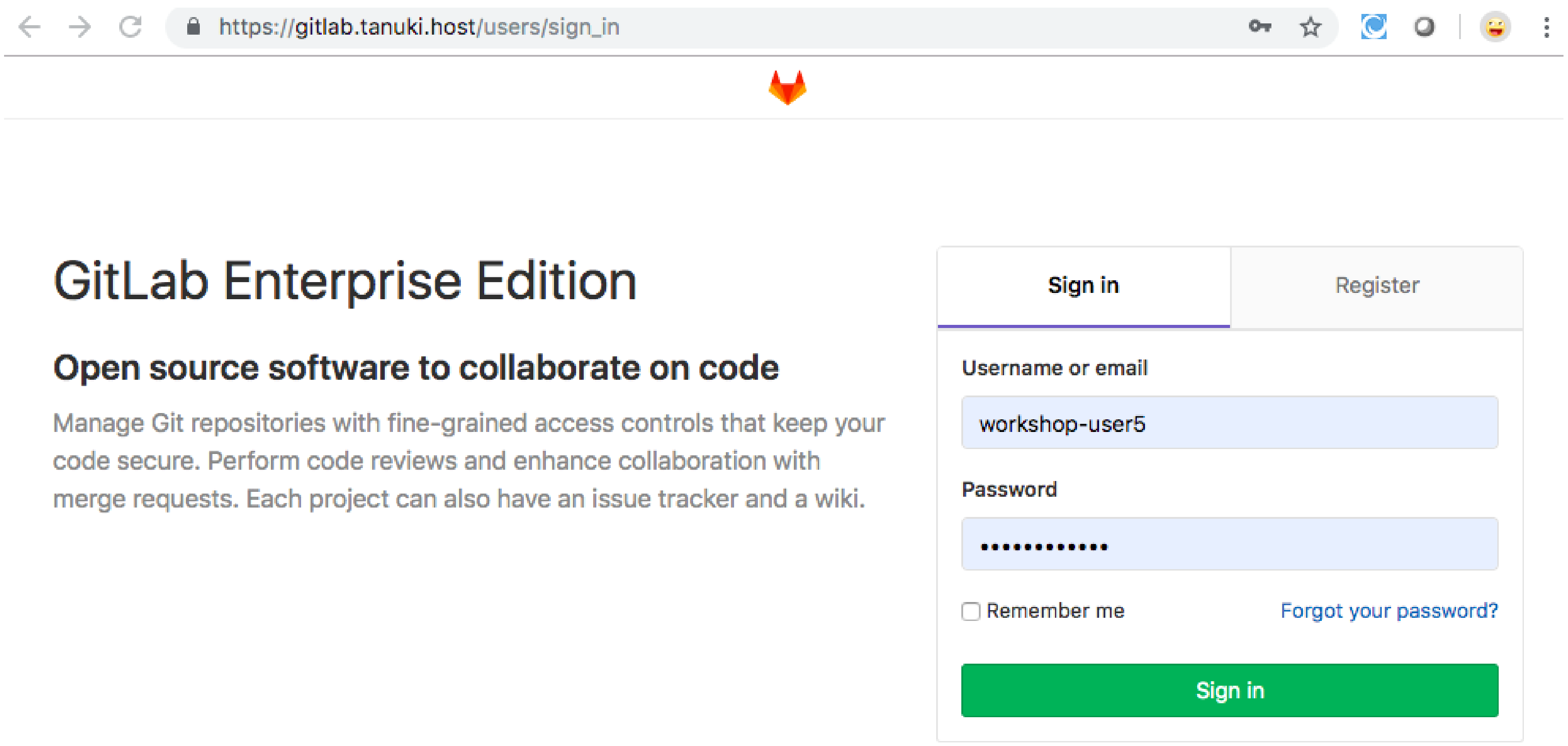



O'REILLY
Docker Cookbook
SOLUTIONS AND EXAMPLES FOR BUILDING DISTRIBUTED APPLICATIONS
Sébastien Goasguen

Pre-requisites

That little sign-in card

`workshop-userXYZ` on `https://gitlab.tanuki.host`



https://gitlab.tanuki.host/users/sign_in

GitLab Enterprise Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Username or email: workshop-user5

Password: •••••••••••••

Under the hood pre-requisites

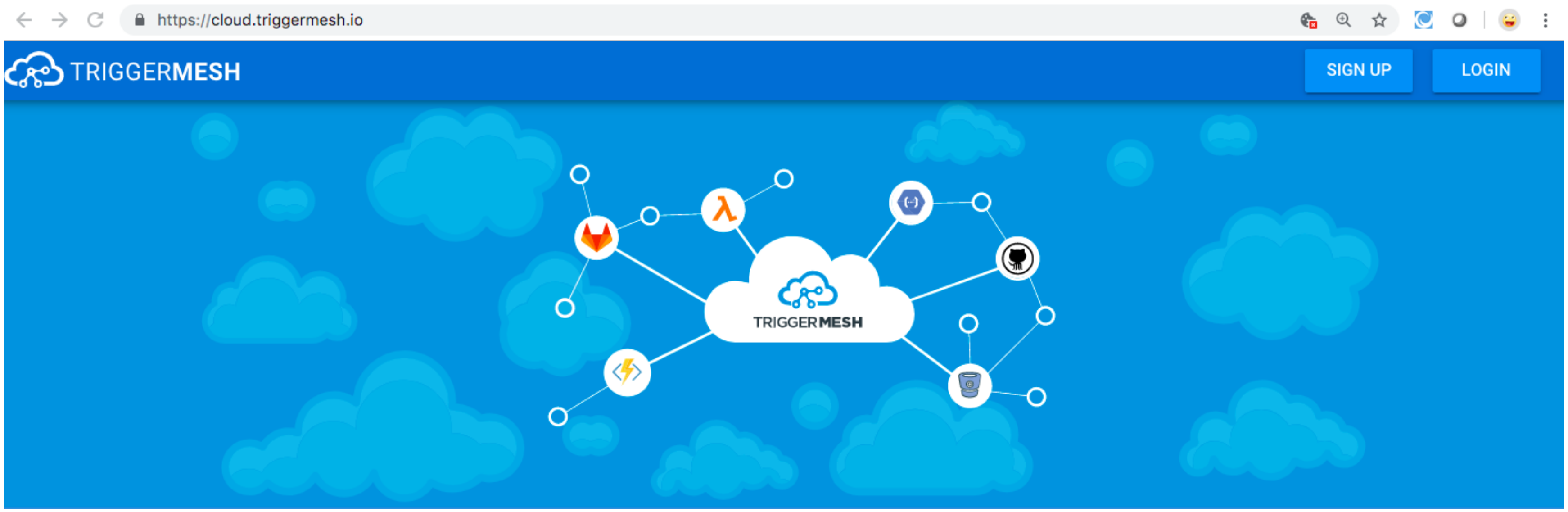
- `kubectl` , https://kubernetes.io/docs/user-guide/prereqs/
- `tm` https://github.com/triggermesh/tm
- Sign-in to https://cloud.triggermesh.io

handled for you, you don't need to do it :)

TriggerMesh Cloud

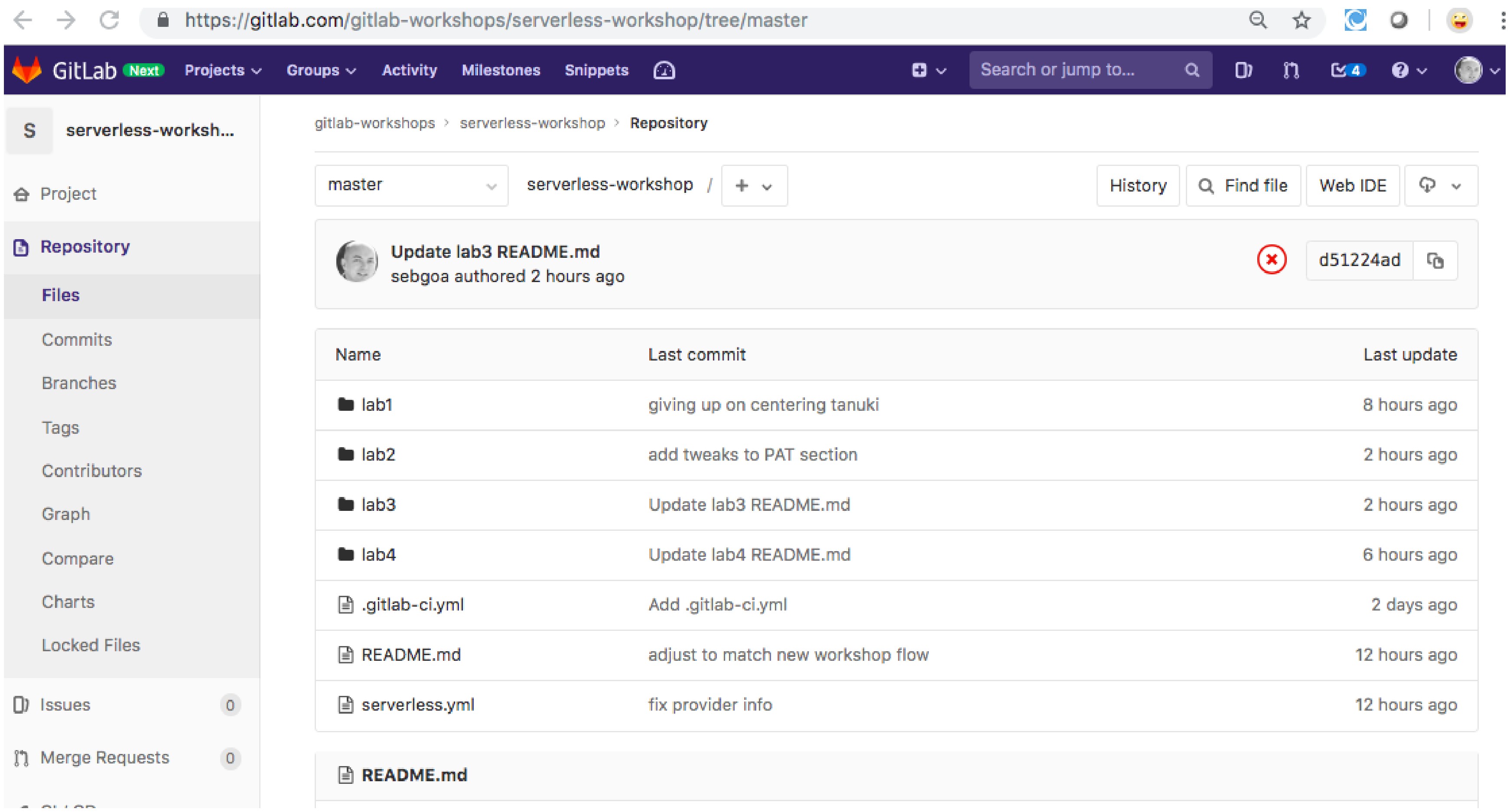
https://cloud.triggermesh.io

- Runs Knative so you don't have to
- Exposes some of the Kubernetes API
- Free + gain time



Lab Content

https://gitlab.com/gitlab-workshops/serverless-workshop



Update lab3 README.md
sebgoa authored 2 hours ago
d51224ad

Name	Last commit	Last update
lab1	giving up on centering tanuki	8 hours ago
lab2	add tweaks to PAT section	2 hours ago
lab3	Update lab3 README.md	2 hours ago
lab4	Update lab4 README.md	6 hours ago
.gitlab-ci.yml	Add .gitlab-ci.yml	2 days ago
README.md	adjust to match new workshop flow	12 hours ago
serverless.yml	fix provider info	12 hours ago

README.md

Agenda

A bit of introduction

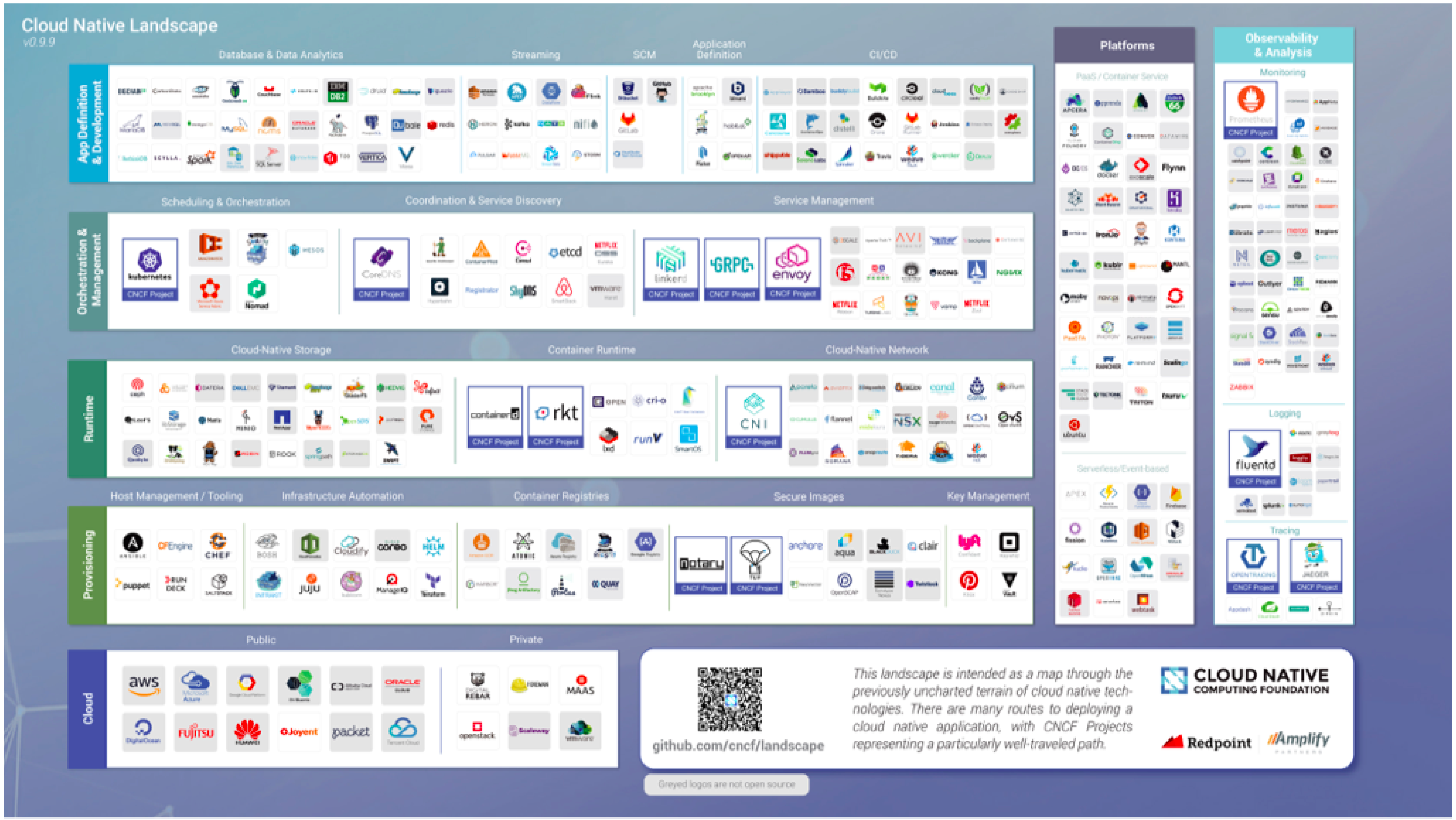
Four Labs

... it is a Workshop after all !

IT Landscape(s)

We are being bombarded with new tech every day.

Our landscapes of tools and solutions is increasingly hard to understand



This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

It's the future !

CircleCI Blog

It's The Future

https://circleci.com/blog/its-the-future/

It Is Complicated

- Create a cluster in the cloud, install a container runtime, install an orchestrator
- Install an app packager
- Install those two or three other systems running on top of your orchestrator
- Now deal with this new networking paradigm
- Finally get your app up after having broken it down in nanoservices

And you will benefit from scale, resiliency and added automation, if you do things right.

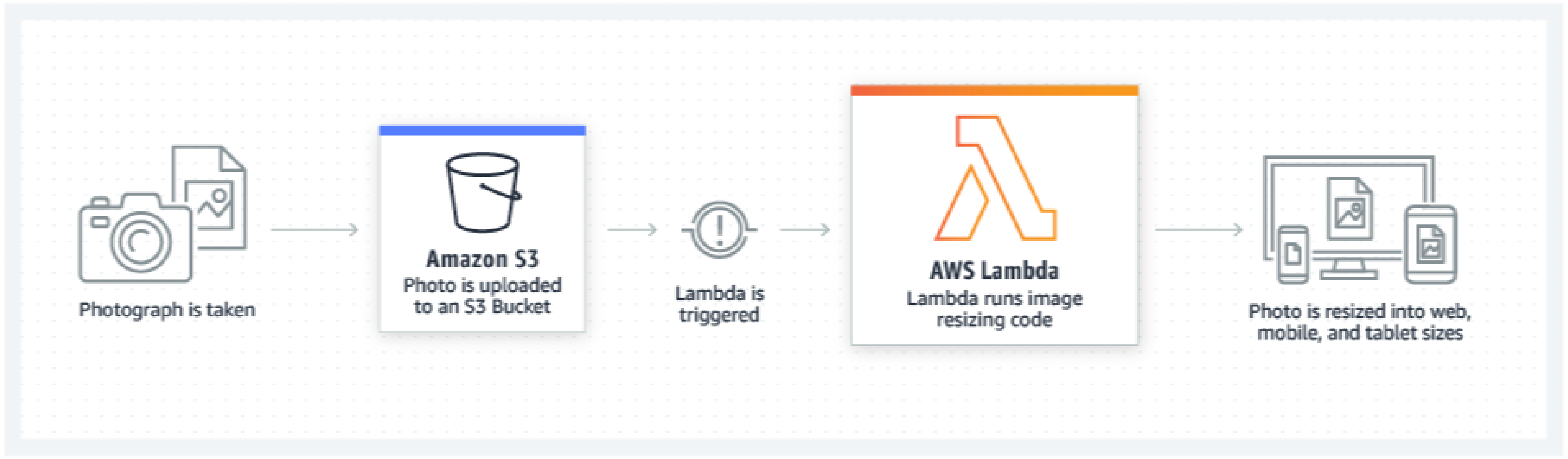
Solution

- New abstractions
- New paradigm
- Hopefully simplicity !!

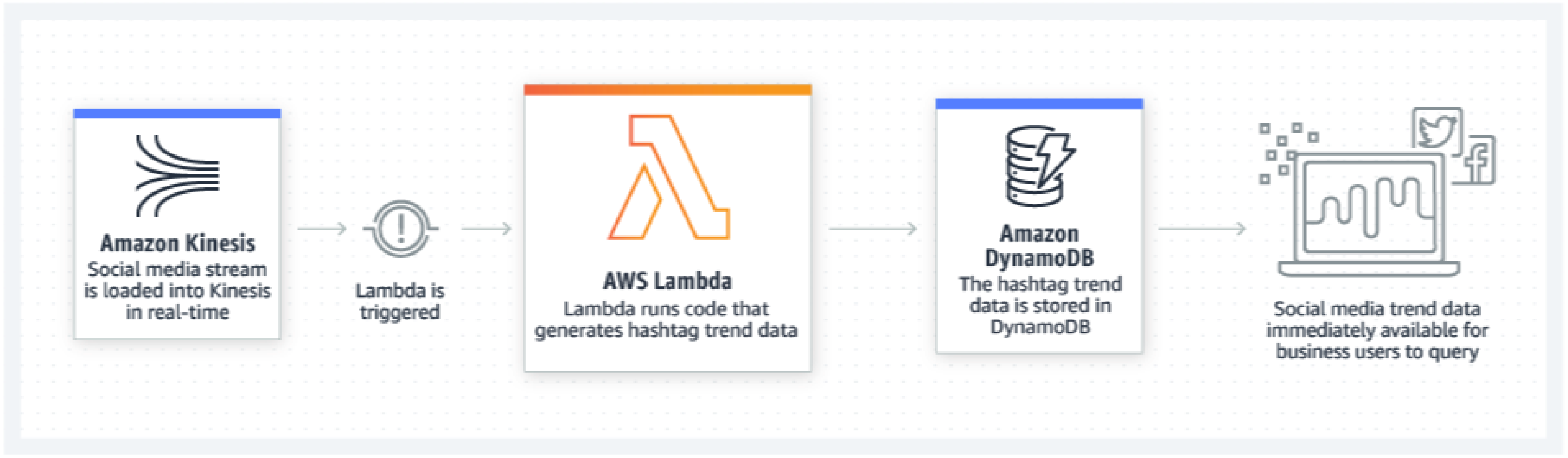
Serverless

- Event Driven Architecture (decoupled components)
- Servicefull
- Fine grain pay per-use
- FaaS as processing between cloud services linked by events

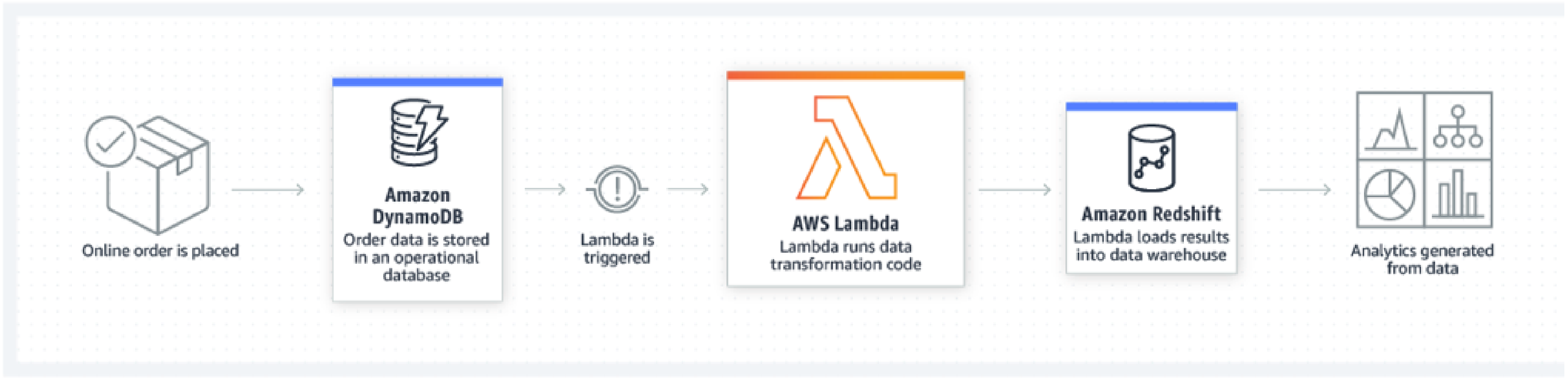
File-processing


Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

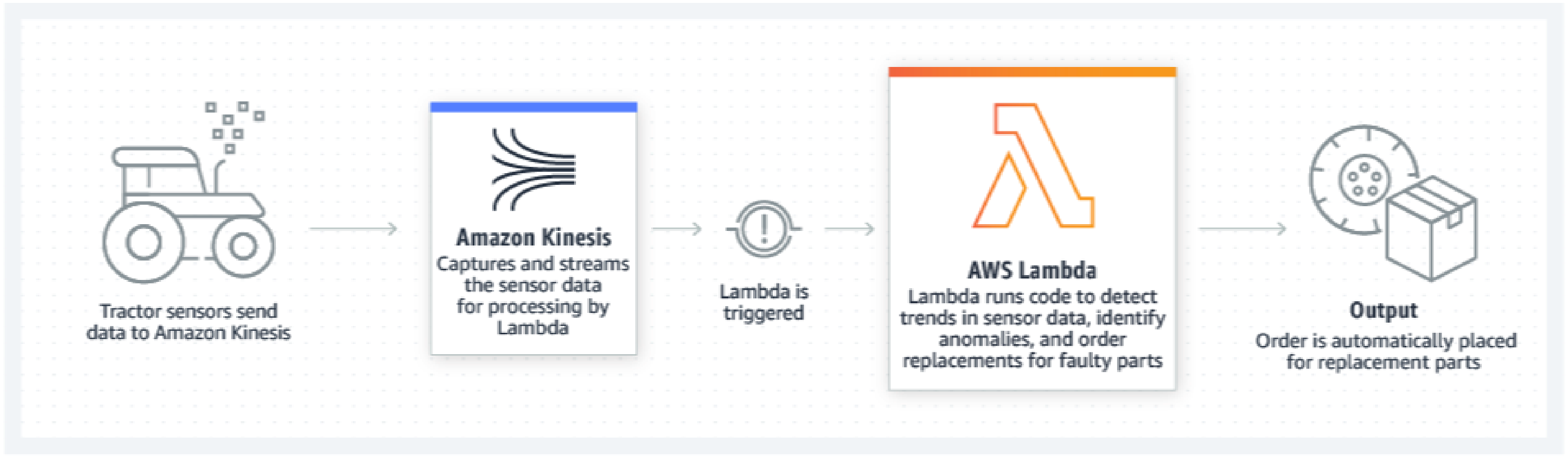
Stream Processing


Amazon Kinesis
Social media stream is loaded into Kinesis in real-time
Lambda is triggered
AWS Lambda
Lambda runs code that generates hashtag trend data
Amazon DynamoDB
The hashtag trend data is stored in DynamoDB
Social media trend data immediately available for business users to query

Extract, Transform, Load (ETL)


Online order is placed
Amazon DynamoDB
Order data is stored in an operational database
Lambda is triggered
AWS Lambda
Lambda runs data transformation code
Amazon Redshift
Lambda loads results into data warehouse
Analytics generated from data

IoT


Tractor sensors send data to Amazon Kinesis
Amazon Kinesis
Captures and streams the sensor data for processing by Lambda
Lambda is triggered
AWS Lambda
Lambda runs code to detect trends in sensor data, identify anomalies, and order replacements for faulty parts
Output
Order is automatically placed for replacement parts

Observations

- AWS is again the lader
- "Simple" pipeline but that can scale
- Serverless but also **ServiceFull**

Challenge

How can you build these applications:

- On your own or just using the services
- Without Lockin
- Using services that may only be available on-prem
- But with limited operational cost while having scale and resilience

Knative as a Solution

Extending Kubernetes

Builder more complete abstractions on top of k8s

- automatic scaling
- better deployment scenarios
- traffic splitting
- automated builds
- event driven flows
- ...

CRD Example *refresh*

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: databases.foo.bar
spec:
  group: foo.bar
  version: v1
  scope: Namespaced
  names:
    plural: databases
    singular: database
    kind: DataBase
    shortNames:
    - db
```

Let's create this new resource and check that it was indeed created.

```
$ kubectl create -f database.yml
$ kubectl get customresourcedefinition
NAME                        KIND
databases.foo.bar           CustomResourceDefinition.v1beta1.ap
```

Custom Resources

You are now free to create a *customresource*.

```
$ cat db.yml
apiVersion: foo.bar/v1
kind: DataBase
metadata:
  name: my-new-db
spec:
  type: mysql
$ kubectl create -f foobar.yml
```

And dynamically `kubectl` is now aware of the *customresource* you created.

```
$ kubectl get databases
NAME        KIND
my-new-db   DataBase.v1.foo.bar
```

Operator Framework(s)

- Kubebuilder: https://github.com/kubernetes-sigs/kubebuilder
- Operator Framework: https://github.com/operator-framework/operator-sdk
- Metaontroller: https://github.com/GoogleCloudPlatform/metacontroller

... Write your own

Knative CRDs

Knative components are a set of Kubernetes controllers. There are Knative CRDs and associated controllers

```
$ kubectl get crd | grep knative
brokers.eventing.knative.dev                              39d
builds.build.knative.dev                                  160d
buildtemplates.build.knative.dev                          160d
channels.eventing.knative.dev                             160d
clusterchannelprovisioners.eventing.knative.dev           160d
configurations.serving.knative.dev                        160d
containersources.sources.eventing.knative.dev             160d
revisions.serving.knative.dev                             160d
routes.serving.knative.dev                                160d
services.serving.knative.dev                              160d
subscriptions.eventing.knative.dev                        160d
triggers.eventing.knative.dev                             39d
...
```

Lab 1: Knative Serving

> Knative Serving builds on Kubernetes to support deploying and serving of serverless applications and functions.

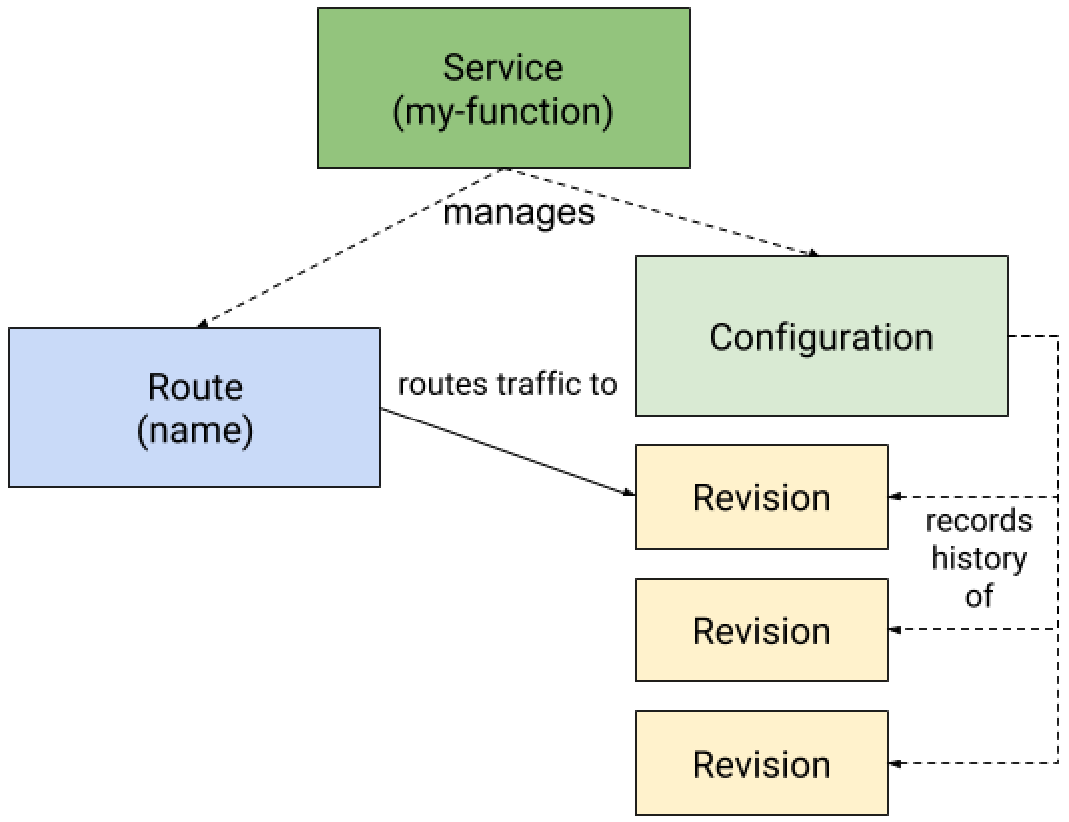
```
$ kubectl get pods -n knative-serving
NAME                              READY     STATUS
activator-6f55c97c6d-tsm5w        2/2       Running
autoscaler-84cc7b78c4-ng96p       2/2       Running
controller-db5bbf4b9-6vdq9        1/1       Running
webhook-85ddccf9c6-gfcjh          1/1       Running
```

Under the hood still a Deployment and a Pod ...

Knative Serving API Objects

- **Service**: The `service.serving.knative.dev` resource automatically manages the whole lifecycle of your workload.
- **Route**: The `route.serving.knative.dev` resource maps a network endpoint to a one or more revisions.
- **Configuration**: The `configuration.serving.knative.dev` resource maintains the desired state for your deployment.
- **Revision**: The `revision.serving.knative.dev` resource is a point-in-time snapshot of the code and configuration for each modification made to the workload.

Knative Serving Objects Diagram



Serving Specification

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    spec:
      containers:
        - image: gcr.io/knative-samples/helloworld-go
          env:
            - name: TARGET
              value: "Go Sample v1"
```

`kubectl apply -f hello.yaml` or paste it in the TriggerMesh UI or create it via GitLab CI.

`gitlab-ci.yml` explanation

Use `tm` to create a `Service` object in the TriggerMesh Cloud.

```
stage:
  - deploy-function

deploy-hello-function:
  stage: deploy-function
  environment: test
  image: gcr.io/triggermesh/tm:latest
  before_script:
    - echo $TMCONFIG > tmconfig
  script:
    - tm --config ./tmconfig deploy --wait; echo
```

`serverless.yml` explanation

Similar to the famous Serverless framework. Get all info needed to create a Knative service from the `serverless.yaml` more succint manifest.

```yaml
functions:
  hello:
    source: hello
    runtime: https://gitlab.com/gitlab-workshops/workshop-resou
    description: "python Hello function with KLR template"
    buildargs:
     - DIRECTORY=hello
     - HANDLER=hello.endpoint
```

Go !

Lab 2: Serverless Containers

But but...

I thought Serverless had nothing to do with Containers, can't I just run my code ?

Sure but it will need to run somewhere and be packaged. Containers are a great packaging artefcats. If you give me your code, I still need to package it, aka. Build.

Hence we need a way to create Containers within a Kubernetes cluster

Knative Build

Could run standalone from other Knative components. You could use it out of the box to do basic CI/CD.

```
$ kubectl get pods -n knative-build
NAME                               READY     STATUS    RESTART
build-controller-694d8444f8-x6z2t  1/1       Running   0
build-webhook-7d9b46cdd7-9g6rf     1/1       Running   0
```

But we already have GitLab CI

And we can reliably build Container Images using GitLab CI

Plus...

Store those images in public or private container registries

Here Comes Kaniko

> kaniko is a tool to build container images from a Dockerfile, inside a container or Kubernetes cluster.

https://github.com/GoogleContainerTools/kaniko

```
docker run \
    -v $HOME/.docker/config.json:/kaniko/config.json \
    -v ${context}:/workspace \
    --env DOCKER_CONFIG=/kaniko
    gcr.io/kaniko-project/executor:latest \
    --destination runseb/foo
```

One function and one app linked

- Deploy our function
- Build an App with Kaniko
- Deploy that app

Deploy this application as a serverless container ala Google Cloud Run

```
...
sample-app-build:
 stage: build-app
 image:
   name: gcr.io/kaniko-project/executor:debug-v0.6.0
   entrypoint: [""]
 script:
   - /busybox/echo "{\"auths\":{\"$CI_REGISTRY\":{\"username\":
   - /kaniko/executor --context $CI_PROJECT_DIR --dockerfile $C
```

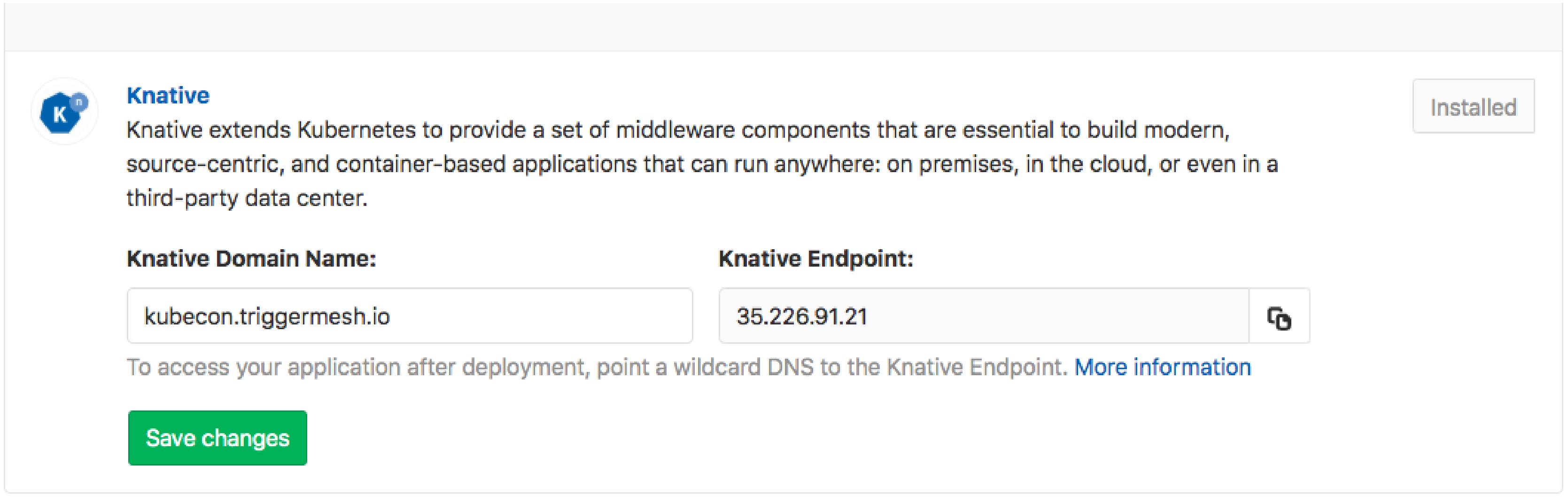
Go !

Lab 3: Knative on your own

Knative gets Installed on your Kubernetes cluster via the GitLab Knative integration.

Under the hood, Knative uses a Helm chart from:

`https://github.com/triggermesh/charts`



Knative

Knative extends Kubernetes to provide a set of middleware components that are essential to build modern, source-centric, and container-based applications that can run anywhere: on premises, in the cloud, or even in a third-party data center.

Knative Domain Name: kubecon.triggermesh.io

Knative Endpoint: 35.226.91.21

To access your application after deployment, point a wildcard DNS to the Knative Endpoint. More information

Knative Installation

At a high level we will:

- Create some CRDs
- Create some namespaces
- Launch controllers in those namespaces

Then we will be able to create the Knative API objects.

```
$ kubectl get ns | grep knative
knative-build             Active    160d
knative-eventing          Active    160d
knative-monitoring        Active    160d
knative-serving           Active    160d
knative-sources           Active    160d
```

Provider Agnostic Installation

https://knative.dev/docs/install/knative-with-any-k8s/

With the `0.5` release, let's still install Istio:

```
kubectl apply --filename https://raw.githubusercontent.com/knat
```

Then the Knative CRDs:

```
kubectl apply --selector knative.dev/crd-install=true \
   --filename https://github.com/knative/serving/releases/downl
  ...
```

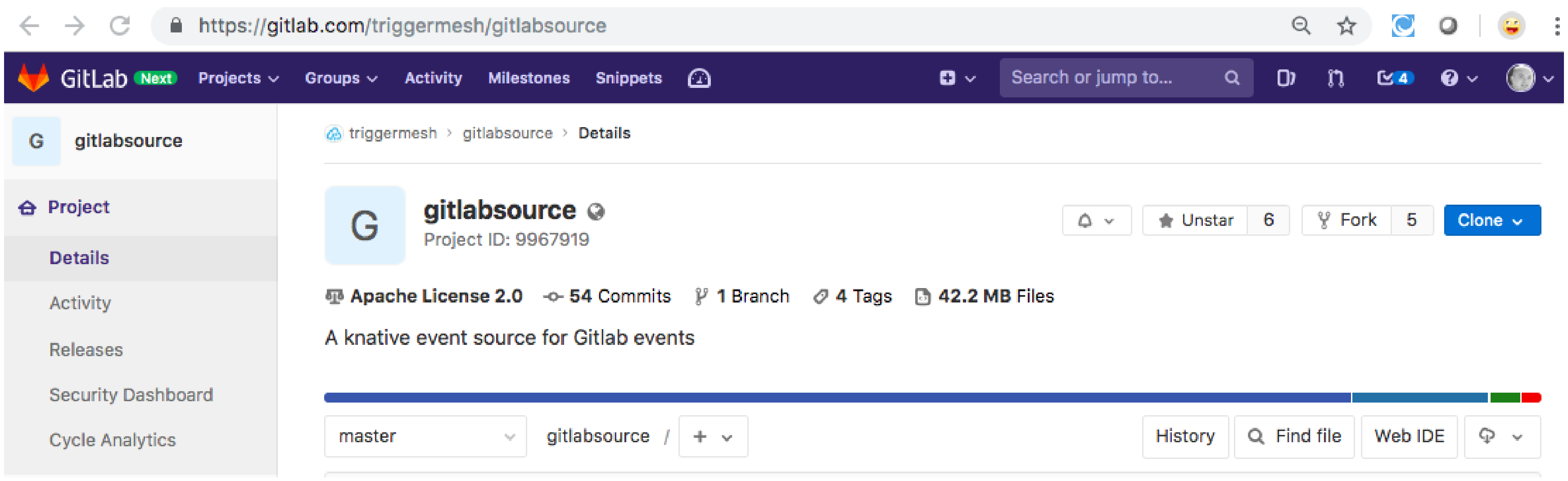
Then the Knative controllers:

```
kubectl apply --filename https://github.com/knative/serving/rel
   --filename https://github.com/knative/build/releases/downloa
   ...
```

Go !

Lab 4: Knative eventing

Triggering Function on Events with Knative Eventing



gitlabsource
Project ID: 9967919

Apache License 2.0 · 54 Commits · 1 Branch · 4 Tags · 42.2 MB Files

A knative event source for Gitlab events

Knative Eventing

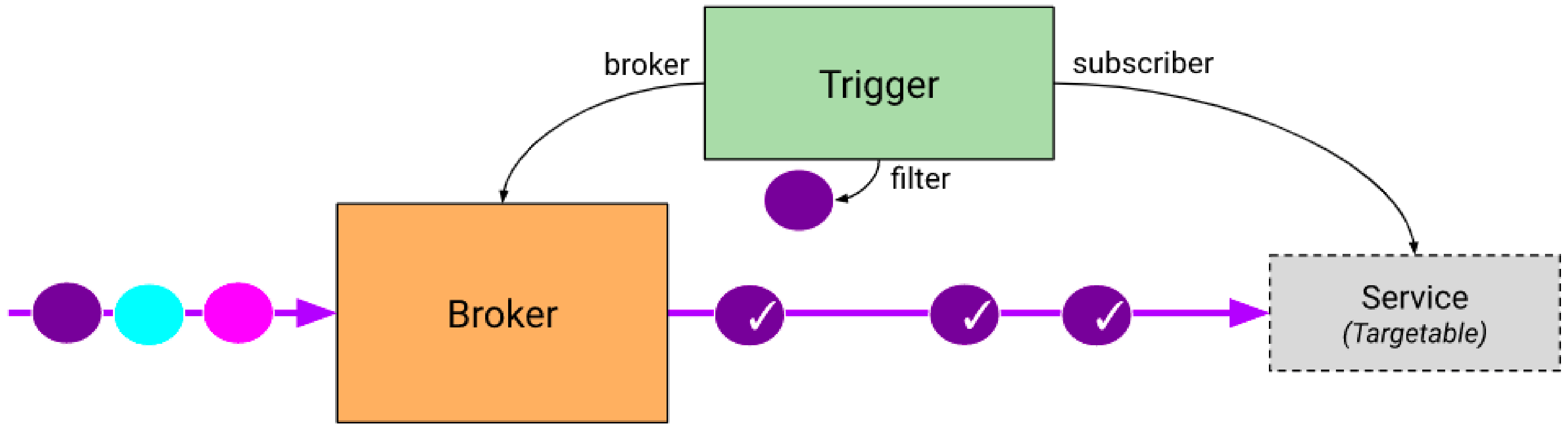
> Knative Eventing is a system that is designed to address a common need for cloud native development and provides composable primitives to enable late-binding event sources and event consumers.

Consume events from *Sources*, use those events to *Trigger* execution of *functions*.

Knative eventing Objects

Architecture still in flux (`v0.6`) trying to find the right abstractions to decouple eventing from messaging and provide easy to use objects.

- Channel
- Subscription
- Broker
- Trigger



Knative Eventing

When install, Knative will have a `knative-eventing` namespace

```
$ kubectl get pods -n knative-eventing
NAME                                            READY     STATU
eventing-controller-774f79f989-xp2kc            1/1       Runni
in-memory-channel-controller-5c686c86c7-5kvgr   1/1       Runni
in-memory-channel-dispatcher-7bcd7f556-q25qb    2/2       Runni
webhook-5b689bfcc4-78772                        1/1       Runni
```

You may see other channel controllers (e.g Kafka, NATS, GCP PubSub ...)

Knative Eventing Objects

Sources, Channels, Triggers, Brokers ...

```
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: CronJobSource
metadata:
  name: test-cronjob-source
spec:
  schedule: "*/2 * * * *"
  data: '{"message": "Hello world!"}'
  sink:
    apiVersion: serving.knative.dev/v1alpha1
    kind: Service
    name: event-display
```

Go !

Wrap-Up

- Knative is an extension of the Kubernetes API
- It provides APIs to build serverless workloads
- Serving gives you scale to zero
- Eventing allows you to trigger function when events happen

Knative gives you a portability/multi-cloud solution to serverless.

You can do this lab again at your own pace !!!

Serverless is more than FaaS, it blends Event Driven Architecture (EDA) with new containerized workloads.

Thank You

@sebgoa

@eggshellcullen

@cab105

GitLab

TRIGGERMESH

SERVERLESS MANAGEMENT PLATFORM