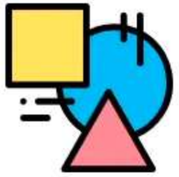
What is Serverless?

What is Serverless?

Serverless architecture generally describes fully managed cloud services.
The classification of a cloud service being serverless is not a Boolean answer (yes or no), but an answer on a scale where a cloud service has a degree of serverless.

A serverless service could have all or most of the following characteristics:

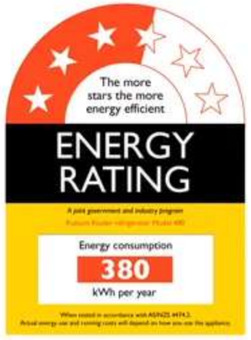
- Highly elastic and scalable
- highly available
- Highly durable
- Secure by default



Abstracts away the underlying infrastructure and are billed based on the execution of your business task.

Serverless can **Scale-to-Zero** meaning when not in use the serverless resources cost nothing.

Pay-for-Value (you don't pay for idle servers).

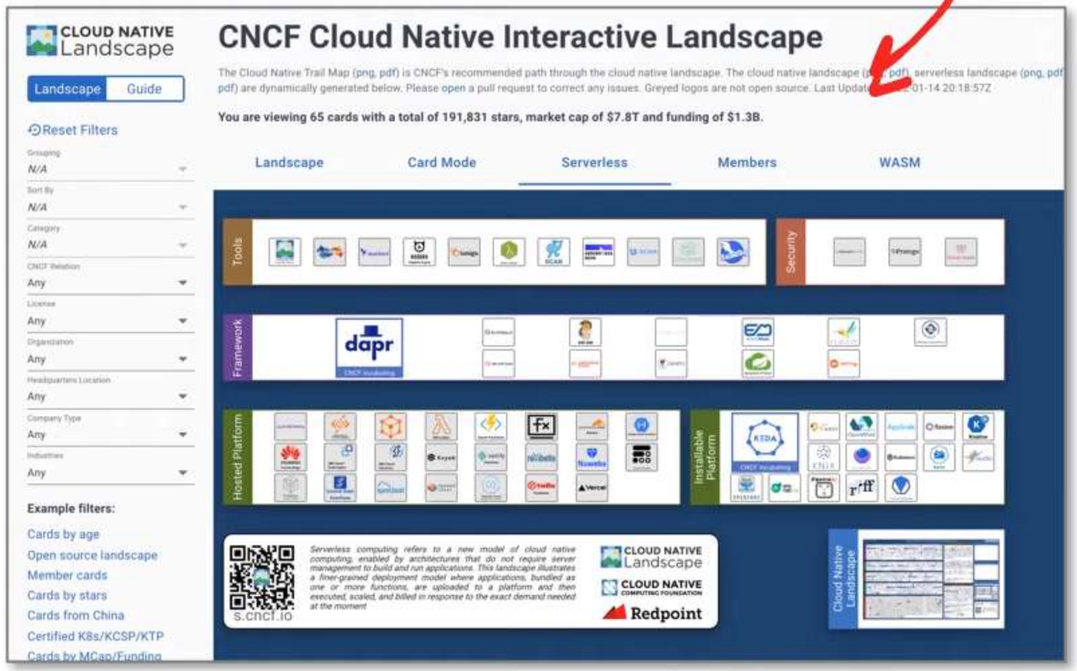


An analogy of serverless could be similar to an energy rating labels which allows consumers to compare the energy efficiency of a product. Some services are more serverless than others.

Cloud Native and K8s Serverless

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

CNCF has a landscape just for **serverless**:



What CNCF classifies as *"serverless"* are:

Function as a Service: eg.

- AWS Lambda
- Azure Serverless Functions
- Google Cloud Functions

Serverless Frameworks: eg.

- Dapr
- AWS SAM
- Chalice

Installable Platforms: eg

- Kubernetes-based Event Driven Autoscaling (KEDA)
- Apache OpenWisk
- OpenFaaS
- Knative
- Fission
- Kubeless

Tools: eg

- Lumigo
- Dashbird

Knative

Knative

Kubernetes-based platform to **deploy and manage modern serverless workloads**

Knative is a project to create a standard set of building blocks for Kubernetes to enable serverless development patterns.

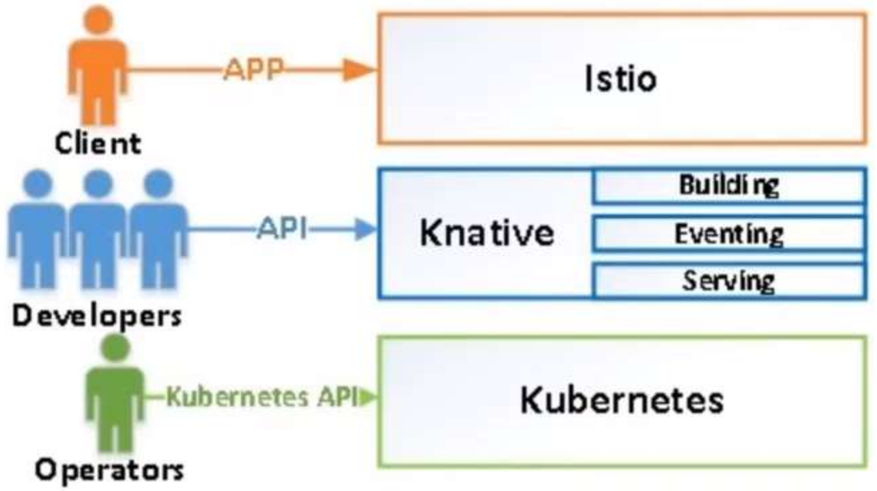
Knative generally is composed of two parts:

Knative Serving

- take containerized code and deploy it with relative ease.
- Scale to zero costs

Knative Eventing

- Trigger serverless functions based on Kubernetes API events
- Loop in other event sources to trigger serverless functions



"Considerations"

- It's not a complete serverless framework
- It does not offer a Function as Service (FaaS) offering

Knative

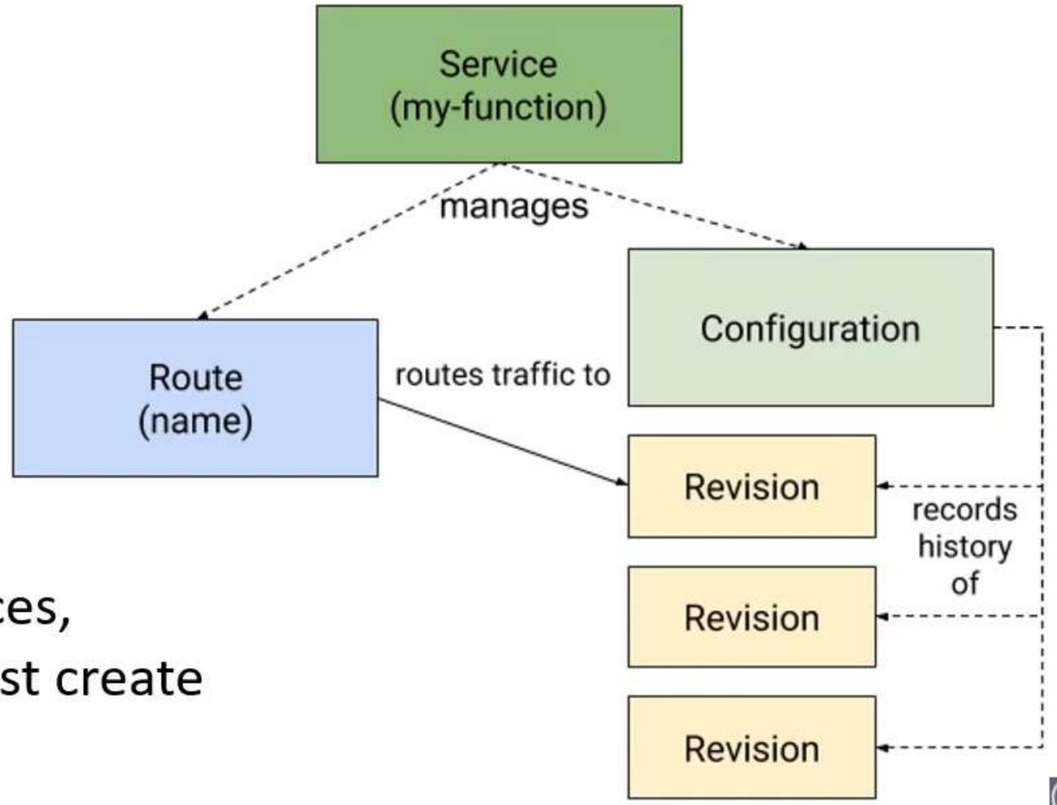
Knative defines its own set of Kubernetes Objects as **Kubernetes Custom Resource Definitions (CRDs).**

```
kn service create hello \
--image gcr.io/knative-samples/helloworld-go \
--port 8080 \
--env TARGET=World \
--revision-name=world
```

Knative has its own CLI called **kn**, used alongside Kubectl.

Knative components:

- **Service** — Manage lifecycle of a workload
- **Route** — Mapping network endpoints
- **Configuration** — Maintains desired state
- **Revision** — Point-in-time snapshots of code



Think of Knative of abstracting away deployments, services, autoscaling and many more K8s components, and you just create and deploy Kn Services.

Knative vs OpenFaaS

Unlike OpenFaas, Knative is not a full-fledged serverless platform, but it is better positioned as a platform for creating, deploying, and managing serverless workloads.

However, from the point of view of configuration and maintenance, OpenFaas is simpler. With OpenFaas, there is no need to install all components separately as with Knative, and you don't have to clear previous settings and resources for new developments if the required components have already been installed.

Still, as mentioned above, a significant drawback of OpenFaaS is that the container launch time depends on the provider, while Knative is not tied to any single cloud solution provider.

Based on the pros and cons of both, organizations may also choose to use Knative and OpenFaaS together to effectively achieve different goals.