Knative 101: What it is, and what it will be

Giuseppe Bonocore
Solution Architect

Giuseppe Brindisi
Solution Architect
A Serverless Datacenter!
“Serverless computing is a cloud-computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity. [...]”

Serverless computing can simplify the process of deploying code into production. Scaling, capacity planning and maintenance operations may be hidden from the developer or operator.”

Wikipedia
How does it work?

Event fires → Your code runs → Function → Output ready

```javascript
function main() {
    return {payload: 'Hello world'};
}
```
Is Serverless Open source?
Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. This landscape illustrates a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.
Welcome to Knative

Knative (pronounced kay-nay-tiv) 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.
A pluggable model for building artifacts, like jar files, zips or containers from **source code**.
Serving

An event-driven model that serves the container with your application and can "scale to zero".
Common infrastructure for consuming and producing events that will **stimulate** applications.

[https://www.flickr.com/photos/brucematsunaga/35273827070](https://www.flickr.com/photos/brucematsunaga/35273827070)
A **Build** is a list of containers run in-order, with source mounted in

- Implemented as a Kubernetes Custom Resource Definition (CRD).
- **BuildTemplates** provide reusable, parameterized recipes that can be used to create Builds
- Pipelines? Maybe
- Aka **S2I** (for the OpenShifters)

```yaml
apiVersion: build.knative.dev/v1alpha1
kind: Build
metadata:
  name: example-build
spec:
  serviceAccountName: build-auth-example
  source:
    git:
      url: https://github.com/example/build-example.git
      revision: master
  steps:
  - name: centos-example
    image: centos
    args: ["centos-build-example", "SECRETS-example.md"]
  steps:
  - image: quay.io/example-builders/build-example
    args: ['echo', 'hello-example', 'build']
```
Leverages Istio

**Configurations** maintains the desired state for your deployment. Modifying a configuration creates a new revision.

**Revisions** represent an immutable snapshot of code and configuration

**Routes** configure ingress over a collection of Revisions and/or Configurations

**Services** (nope, not K8s services) are top-level controllers that manage a set of Routes and Configurations to implement a network service.
Eventing

- Design goals consistent to the specification of CNCF CloudEvents
- **Source**: Produce the events
- **Event Consumers**
  - **Addressable**: Receives and ack (K8s services)
  - **Callable**: Receives and transform
- **Channel**: named endpoint for event forwarding and persistence layer. Implemented by Kafka, AMQP...
- **Subscription**: Registration between channels and services or other channels
CRDs

- Configuration
- Revision
- Route
- Service
- Build

- III. Config: strict separation of config from code
- I. Codebase: different versions may be active
- VIII. Concurrency: Scale out via the process model
- IX. Disposability: fast startup and graceful shutdown
- V. Build, release, run: Strictly separate build and run

https://12factor.net
Pizza as a Service 2.0

http://www.paulkerrison.co.uk

Traditional On-Premises (legacy)
- Conversation
- Friends
- Beer
- Pizza
- Fire
- Oven
- Electric / Gas
- Homemade

Infrastructure as a Service (IaaS)
- Conversation
- Friends
- Beer
- Pizza
- Fire
- Oven
- Electric / Gas
- Communal Kitchen

Containers as a Service (CaaS)
- Conversation
- Friends
- Beer
- Pizza
- Fire
- Oven
- Electric / Gas
- Bring Your Own

Platform as a Service (PaaS)
- Conversation
- Friends
- Beer
- Pizza
- Fire
- Oven
- Electric / Gas
- Takeaway

Function as a Service (FaaS)
- Conversation
- Friends
- Beer
- Pizza
- Fire
- Oven
- Electric / Gas
- Restaurant

Software as a Service (SaaS)
- Conversation
- Friends
- Beer
- Pizza
- Fire
- Oven
- Electric / Gas
- Party

You Manage
Vendor Manages

Configuration
Functions
Scaling...
Runtime
OS
Virtualisation
Hardware

https://medium.com/@pkerrison/pizza-as-a-service-2-0-5085cd4c365e
“Serverless computing refers to the concept of building and running applications that do not require server management. It describes a finer-grained deployment model where applications, **bundled as one or more functions**, are uploaded to a platform and then executed, scaled, and **billed in response to the exact demand needed at the moment.**”
FaaS

\{ f(x) \}

Serverless

- User experience
- Services
- No pods for services
- No pods for functions (on user projects)
- Debugging/IDE Integration
- API Gateway Integration
- Billing/Charging model
  - Per function call
  - Per execution time
  - Resource consumption

Services

- DB
- Storage
- Messaging
- Events

(API)
Common use cases

- Processing web hooks
- Scheduled tasks (a la cron)
- Data transformation
- Mobile image manipulation (compression, conversion, and so on)
- Voice packet to JSON transformation (Alexa, Cortana, and so on)
- Mobile video analysis (frame-grabbing)
- PDF generation
- Mobile/MBaaS /single-page apps
- Chat bots

Web
Mobile
IoT
DevOps Automation

Focus on convenience and business value, no distractions.

Asynchronous, concurrent, easy to parallelize into independent units of work
When not to use Serverless

- Real-time, ultra-low latency applications
- Long running tasks that can't be split into steps
- Advanced or complex observability and monitoring requirements
- Memory or CPU requirements are very demanding and specific
- Can't deal with cold-start
Q&A

- bonocore@redhat.com
- gbrindis@redhat.com

https://blog.openshift.com/knative-serving-your-serverless-services/
https://github.com/knative/