redhat.

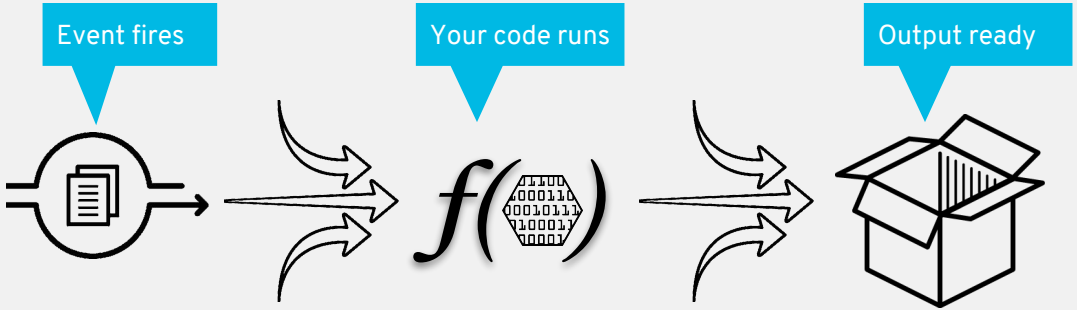
Knative 101: What it is, and what it will be

Giuseppe Bonocore
Solution Architect

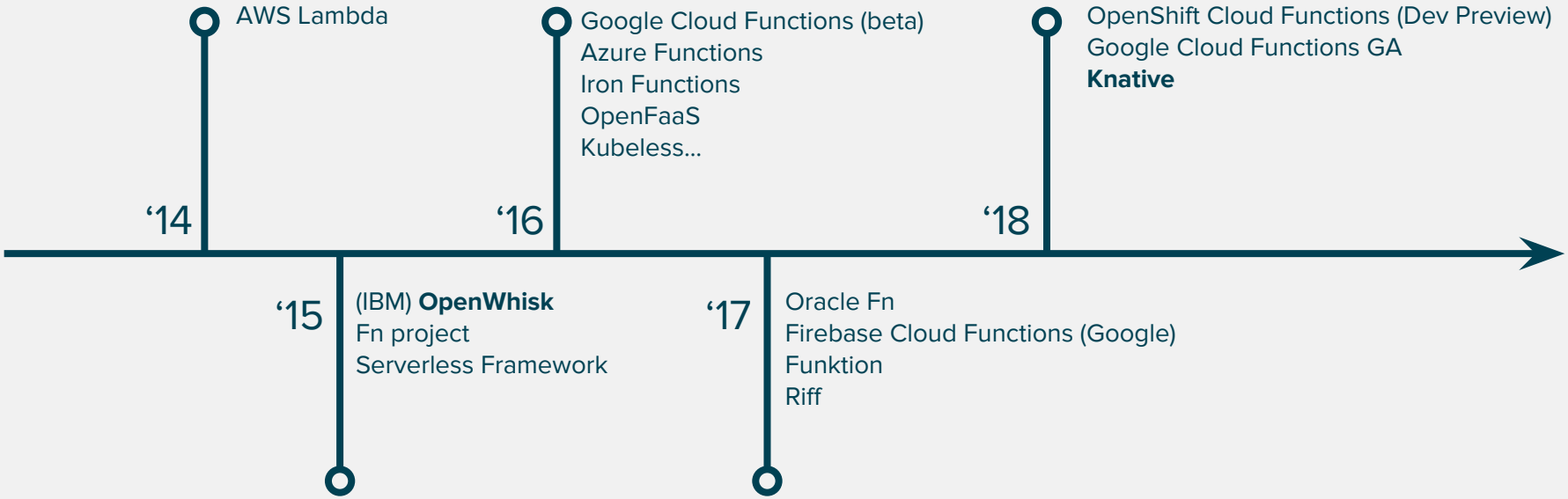

93
91
89
91


A Serverless Datacenter !

How does it work ?




```
function main() {
    return {payload: 'Hello world'};
}
```


AWS Lambda
‘14
‘15
(IBM) **OpenWhisk**
Fn project
Serverless Framework
‘16
Google Cloud Functions (beta)
Azure Functions
Iron Functions
OpenFaaS
Kubeless...
‘17
Oracle Fn
Firebase Cloud Functions (Google)
Funktion
Riff
‘18
OpenShift Cloud Functions (Dev Preview)
Google Cloud Functions GA
Knative

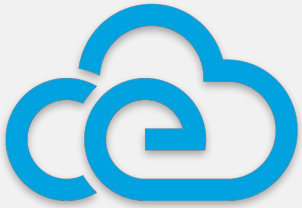
Is Serverless Open source?

CLOUD NATIVE
COMPUTING FOUNDATION

Standards!

Standards!

Standards!



cloudevents

redhat

THE LINUX FOUNDATION

Serverless Cloud Native Landscape

v20180525

See the serverless interactive landscape at s.cncf.io

Greyed logos are not open source

Tools

dashbird, Epsagon, event gateway, Gloo, IO|pipe, Iron.io, Node Lambda, OVERCLOCK, python-λ, SIGMA, STACKERY, THUNDRA

Security

intrinsic, Protego, PURESEC, snyk

Framework

APEX, Architect, aws Chalice, AWS SAM, Claudia.js, FLOGO, gun.io, serverless, Shep, SPARTA, Spring Cloud Function

Platform

Hosted

ALGORITHMIA, AWS Lambda, Azure Functions, BINARIS, Clay, cloudboost, CLOUDFLARE, Google Cloud Functions, HUAWEI, HYPER.SH, IBM Cloud, NANO-LAMBDA, PubNub FUNCTIONS, spotinst, «stdlib», syncano, twilio, WEBLAB, ZEIT

Installable

OpenWhisk, AppScale, DISPATCH, fission, fn, GALACTIC FOG, Kubeless, LunchBadger, nuclio, OPENFAAS, OpenLambda, rff

s.cncf.io

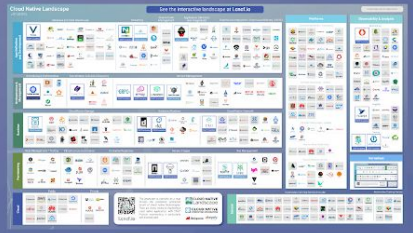
Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. This landscape illustrates a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

CLOUD NATIVE Landscape

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint

Cloud Native Landscape



README.md

Welcome to Knative

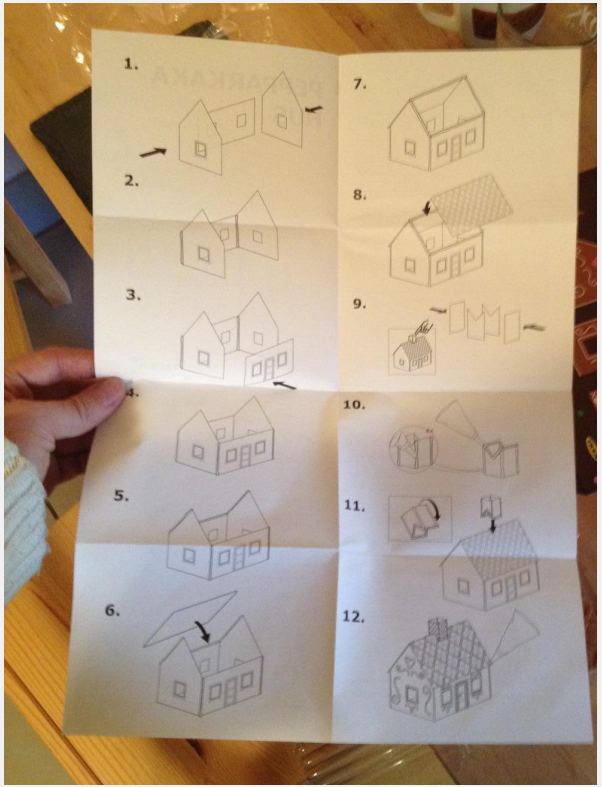
Knative (pronounced kay-nay-tiv) **extends Kubernetes** to provide a set of **middleware components** that are essential to build **modern**, **source-centric**, and **container-based** applications that can **run anywhere**: on premises, in the cloud, or even in a third-party data center.

redhat.

Pivotal

Build

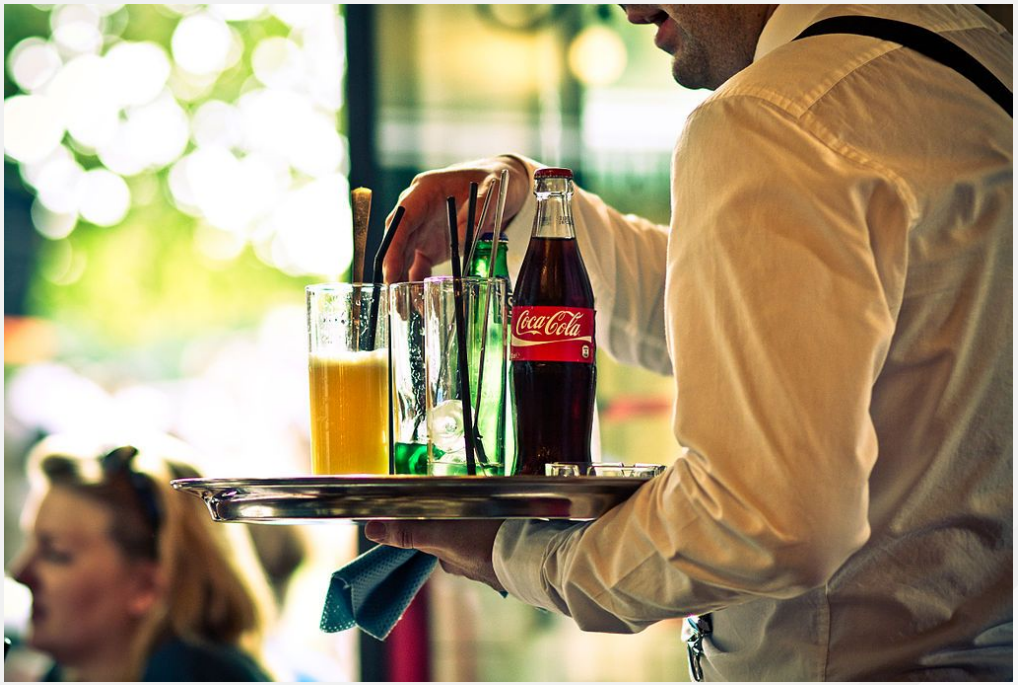
A pluggable model for building artifacts, like jar files, zips or containers from **source code**.



https://www.flickr.com/photos/harrywood/11517650335

Serving

An event-driven model that serves the container with your application and can "**scale to zero**".



https://commons.wikimedia.org/w/index.php?curid=19085820

Events

Common infrastructure for consuming and producing events that will **stimulate applications**.

https://www.flickr.com/photos/brucematsunaga/35273827070

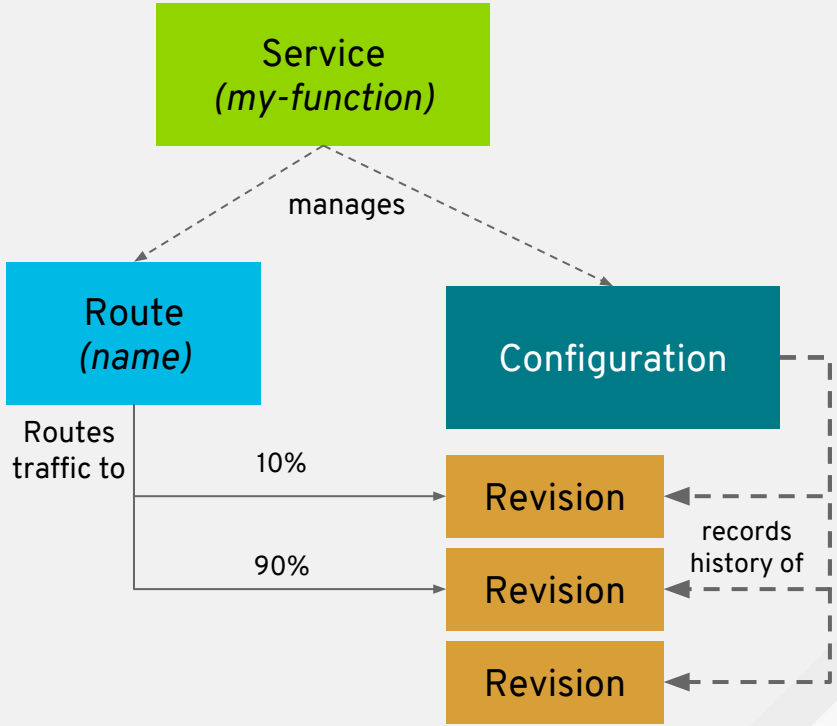
Build

- 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']
```

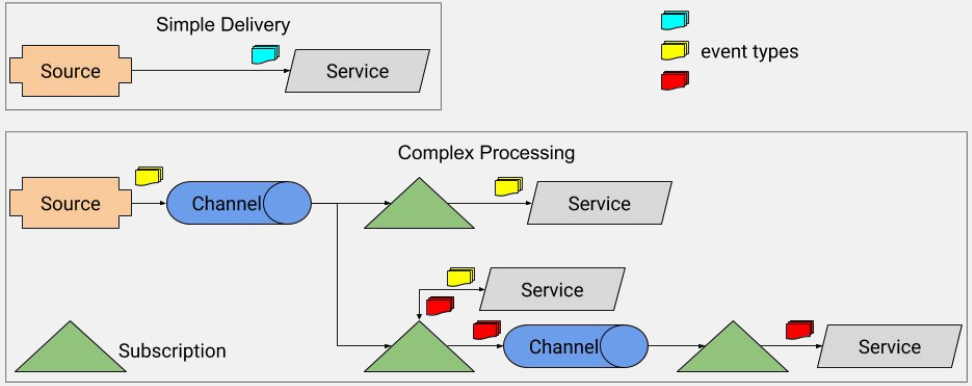
Serving

- 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

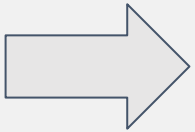
- Design goals consistent to the specification of **CNCF Cloudevents**
- **Source:** Produce the events
- **Event Consumers**
  - **Addressable:** Receives and ack (K8s services)
  - **Callable:** Receives and transform



- **Channel:** named endpoint for event forwarding and persistence layer. Implemented by Kafka, AMQP...
- **Subscription:** Registration between channels and services or other channels

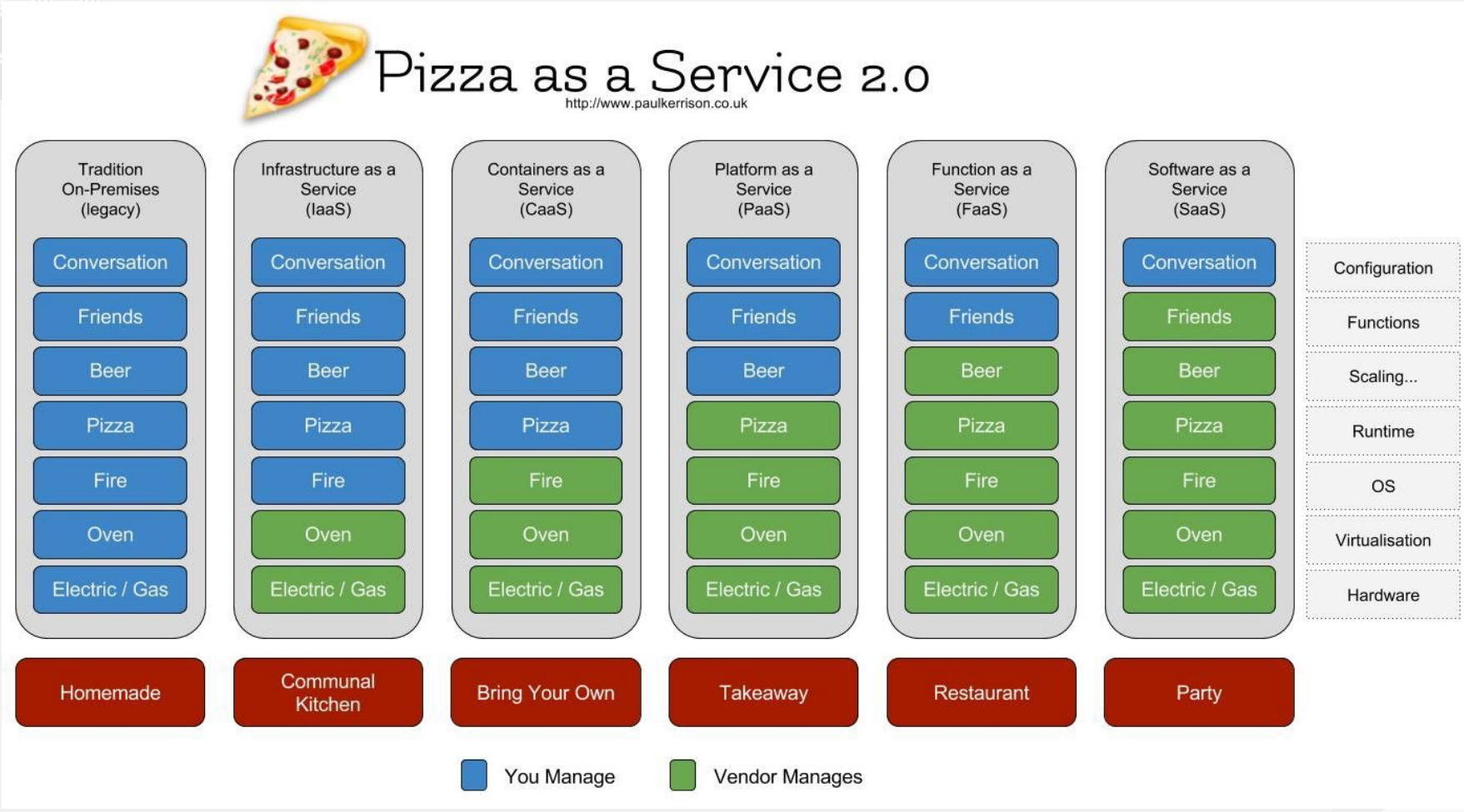
CRDs

- **Configuration**
- **Revision**
- **Route**
- **Service**
- **Build**

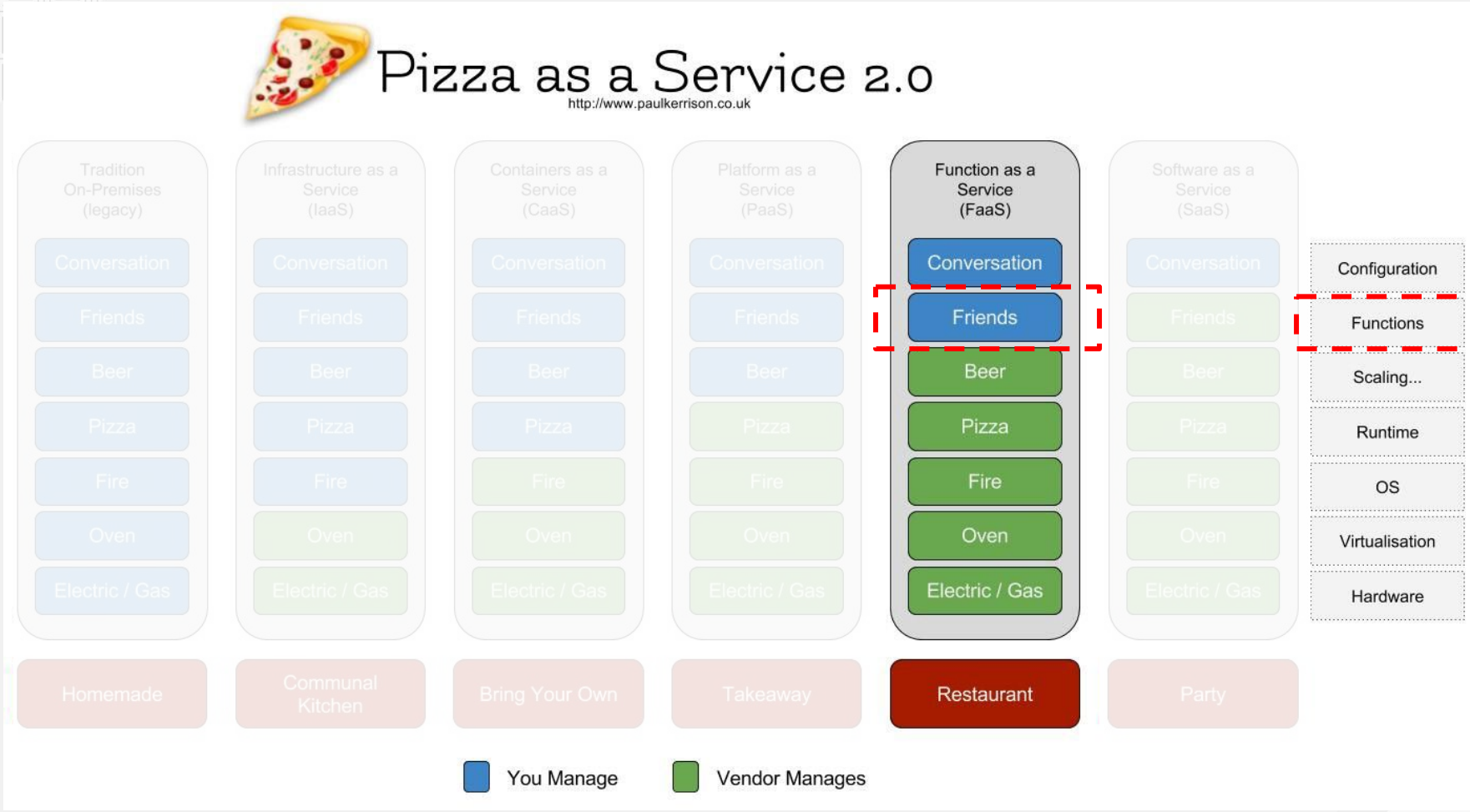


- III. Config: strict separation of config from code
- I. Codebase: different versions may be active
- VIII. Concurrency: Scale out via the process model
- IX. Disposability: fast startup and graceful shutdown
- V. Build, release, run: Strictly separate build and run

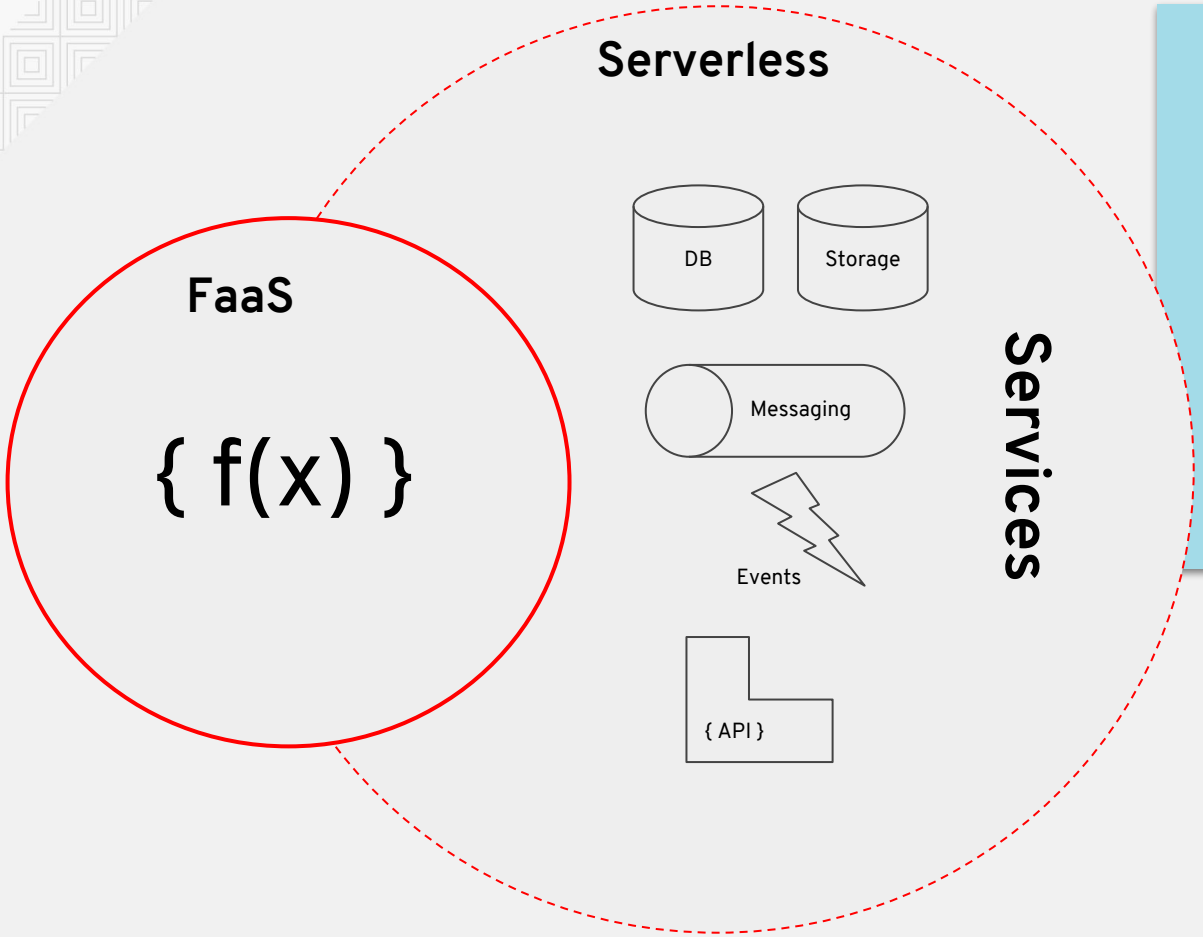
https://12factor.net



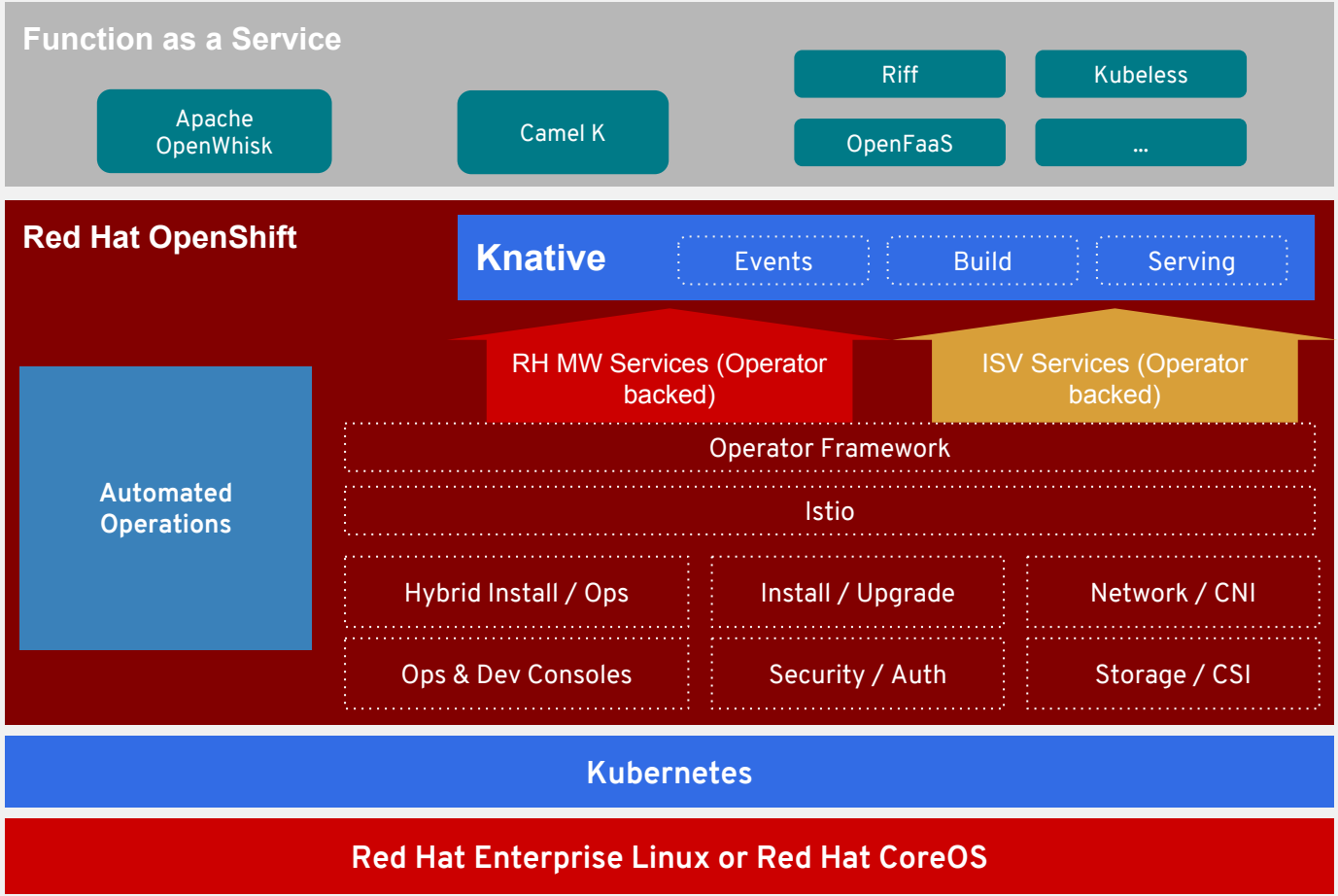
https://medium.com/@pkerrison/pizza-as-a-service-2-0-5085cd4c365e



https://medium.com/@pkerrison/pizza-as-a-service-2-0-5085cd4c365e



- 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


Function as a Service
Apache OpenWhisk
Camel K
Riff
Kubeless
OpenFaaS
...
Red Hat OpenShift
Knative
Events
Build
Serving
RH MW Services (Operator backed)
ISV Services (Operator backed)
Automated Operations
Operator Framework
Istio
Hybrid Install / Ops
Install / Upgrade
Network / CNI
Ops & Dev Consoles
Security / Auth
Storage / CSI
Kubernetes
Red Hat Enterprise Linux or Red Hat CoreOS

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

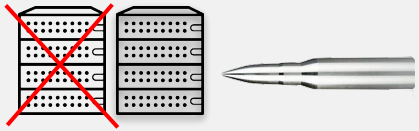
IoT

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

- bonocore@redhat.com

Q&A

https://blog.openshift.com/knative-serving-your-serverless-services/
https://github.com/knative
https://github.com/redhat-developer-demos/knative-tutorial

*“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

CNCF Serverless Whitepaper v1.0:

*"**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."*