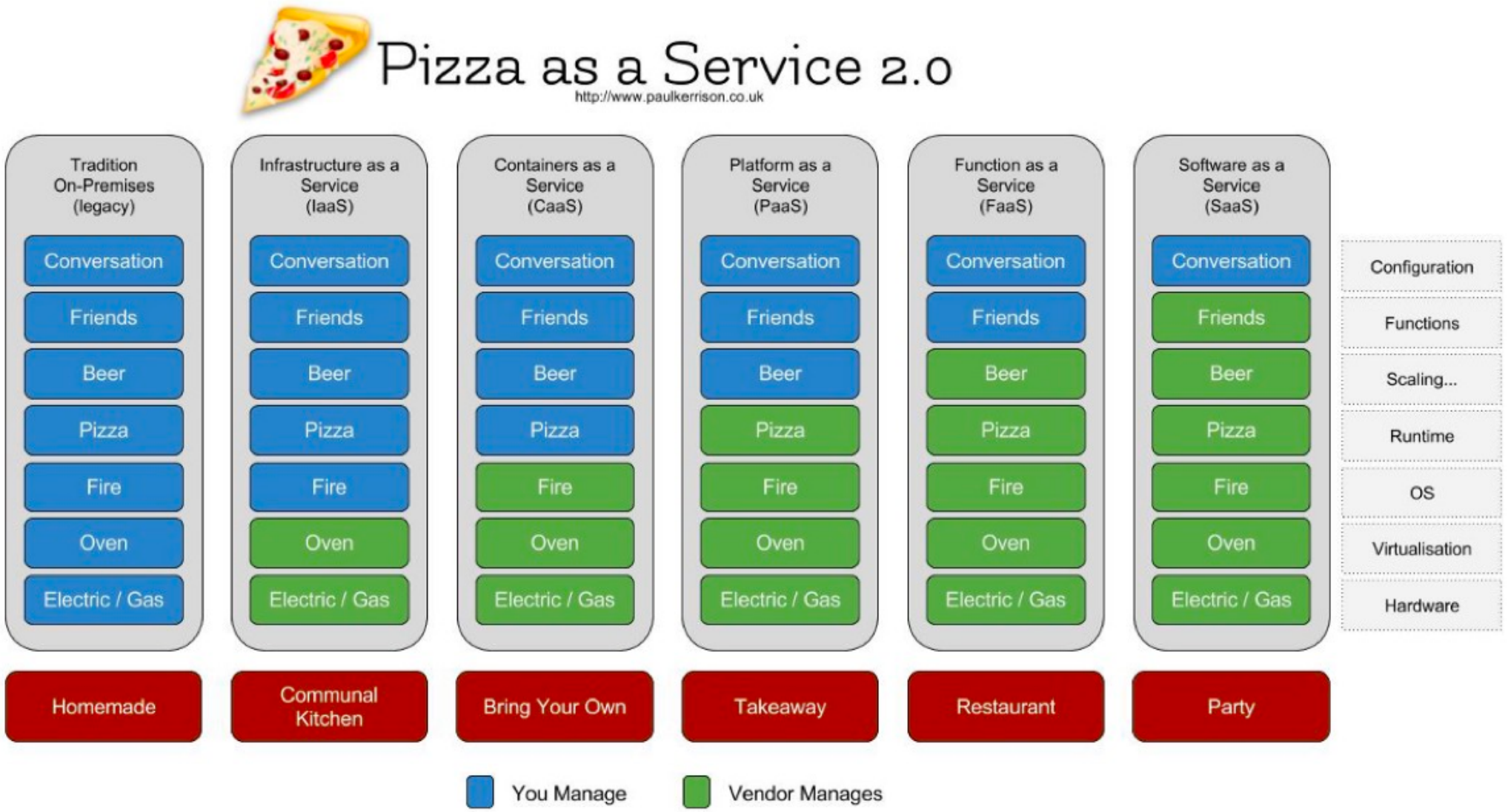
Knative – new hero in serverless world

Błażej Kwaśniak

Wrocław DevOps Meetup

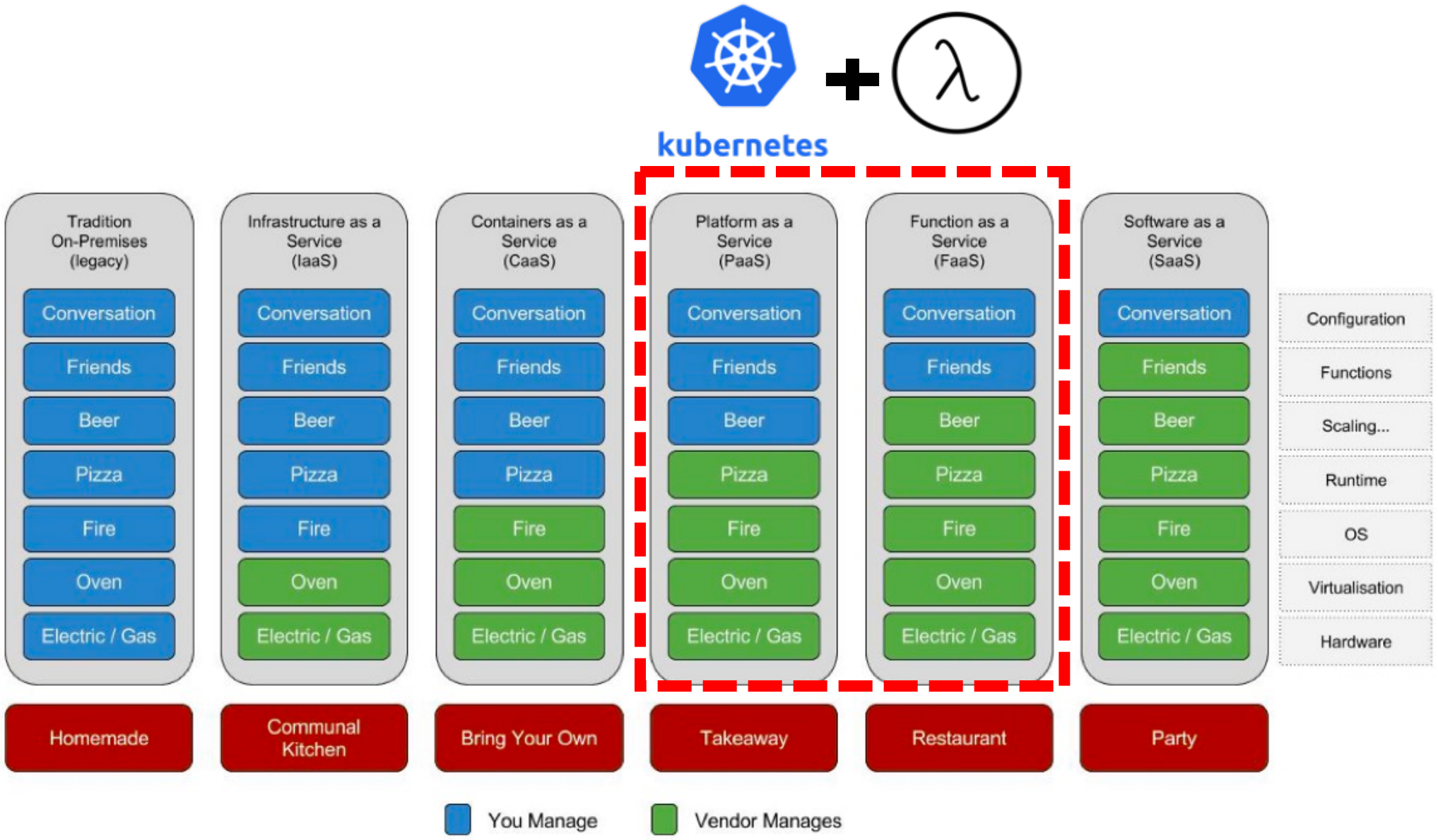
09.04.2019

softserve



Source: *https://www.securitybrigade.com/blog/weekly-cyber-security-news-4/pizza-as-a-service-it-2-0-cloud-deployment/*

softserve



Source: *https://www.securitybrigade.com/blog/weekly-cyber-security-news-4/pizza-as-a-service-it-2-0-cloud-deployment/*

softserve

Serverless

softserve

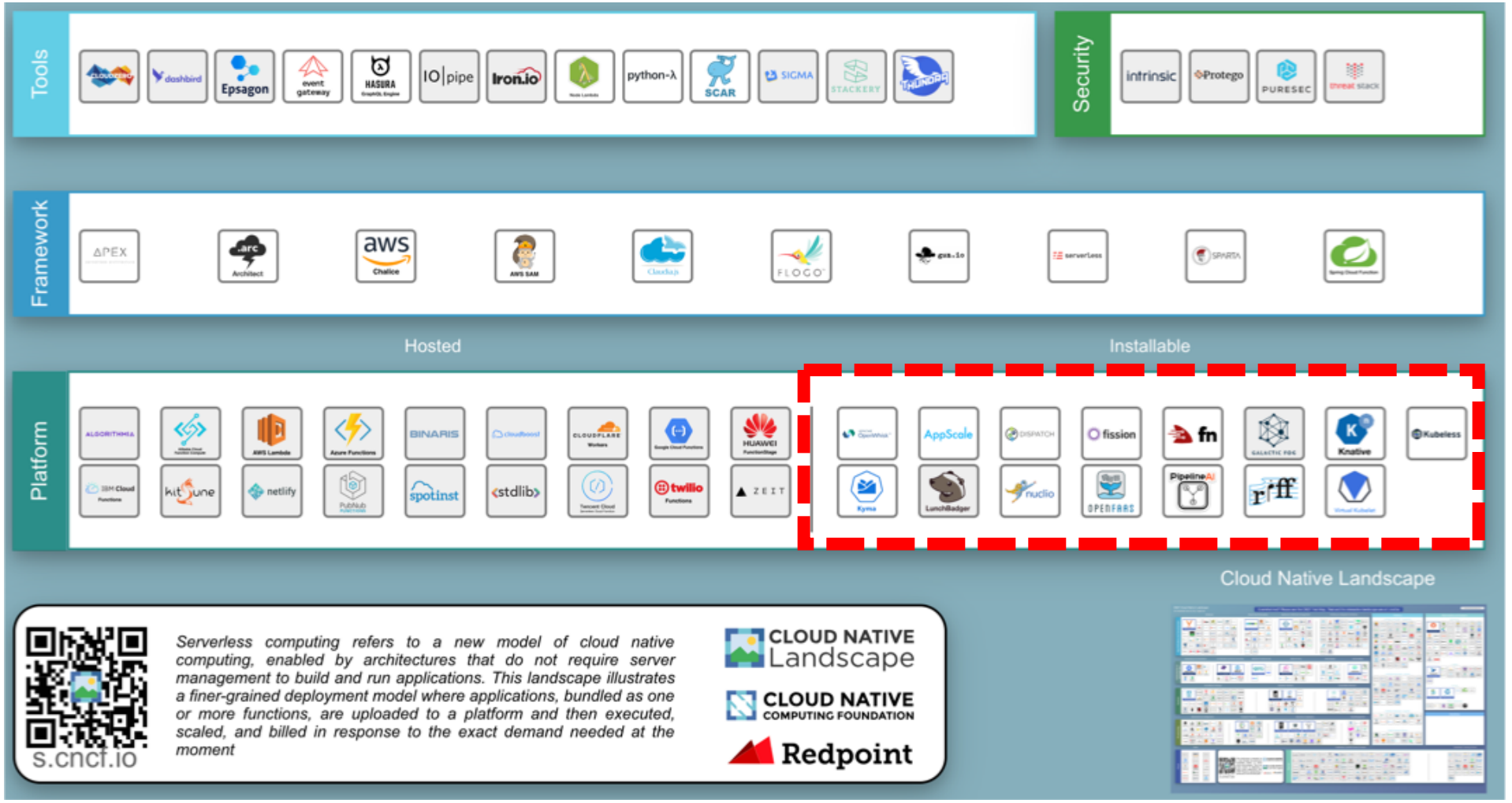

93
91

*"**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."*

Source: https://www.cncf.io/blog/2018/02/14/cncf-takes-first-step-towards-serverless-computing/

softserve

CNCF landscape



Source: https://landscape.cncf.io/format=serverless

softserve

Hands on

Kubeless

softserve

Installation

```
[~/DevOps/kubeless] brew install kubeless
...
[~/DevOps/kubeless] kubectl create ns kubeless
[~/DevOps/kubeless] kubectl create -f https://github.com/kubeless/kubeless/releases/download/
v1.0.3/kubeless-v1.0.3.yaml
...
[~/DevOps/kubeless] kubectl get pods -n kubeless
NAME                                           READY   STATUS    RESTARTS   AGE
kubeless-controller-manager-86b4b7b997-lpq5w   3/3     Running   0          43s
[~/DevOps/kubeless]
```

softserve

Deploy

```
[~/DevOps/kubeless] cat func.py
def hello(event, context):
  print(event)
  return event['data']
[~/DevOps/kubeless]
[~/DevOps/kubeless] kubeless function deploy hello --runtime python3.6 --from-file func.py --handler fu
nc.hello
INFO[0000] Deploying function...
INFO[0000] Function hello submitted for deployment
INFO[0000] Check the deployment status executing 'kubeless function ls hello'
[~/DevOps/kubeless]
[~/DevOps/kubeless]
[~/DevOps/kubeless] kubeless function ls hello
NAME    NAMESPACE       HANDLER         RUNTIME         DEPENDENCIES    STATUS
hello   default         func.hello      python3.6                       1/1 READY
[~/DevOps/kubeless] kubeless function call hello --data 'Hello DevOps!'
Hello DevOps!
[~/DevOps/kubeless]
[~/DevOps/kubeless] curl hello:8080 --data '{"Hello": "DevOps from curl"}' --header "Content-Type:appli
cation/json"
{"Hello": "DevOps from curl"}%
[~/DevOps/kubeless]
```

softserve

There is no magic here...

```
[~/DevOps/kubeless] kubectl get svc
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
hello         ClusterIP   10.23.252.32   <none>        8080/TCP   10m
...
[~/DevOps/kubeless]
[~/DevOps/kubeless] kubectl get deployments
NAME    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
hello   1         1         1            1           10m
[~/DevOps/kubeless]
[~/DevOps/kubeless] kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
hello-6995d65ccd-pfz5f   1/1     Running   0          10m
[~/DevOps/kubeless]
[~/DevOps/kubeless] kubectl exec -it hello-6995d65ccd-pfz5f cat /kubeless/func.py
def hello(event, context):
  print(event)
  return event['data']
[~/DevOps/kubeless]
[~/DevOps/kubeless] kubectl exec -it hello-6995d65ccd-pfz5f md5sum /kubeless/func.py
4c393c8efbc00a78dd68b82f4100224b  /kubeless/func.py
[~/DevOps/kubeless]
[~/DevOps/kubeless] md5sum func.py
4c393c8efbc00a78dd68b82f4100224b  func.py
[~/DevOps/kubeless]
```

softserve

softserve

softserve

A bit of marketing...

Essential base primitives for all

Knative provides 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 components are built on Kubernetes and codify the best practices shared by successful real-world Kubernetes-based frameworks. It enables developers to focus just on writing interesting code, without worrying about the “boring but difficult” parts of building, deploying, and managing an application.

Source: https://cloud.google.com/knative/

softserve

More human-readable definition

What is Knative anyway?

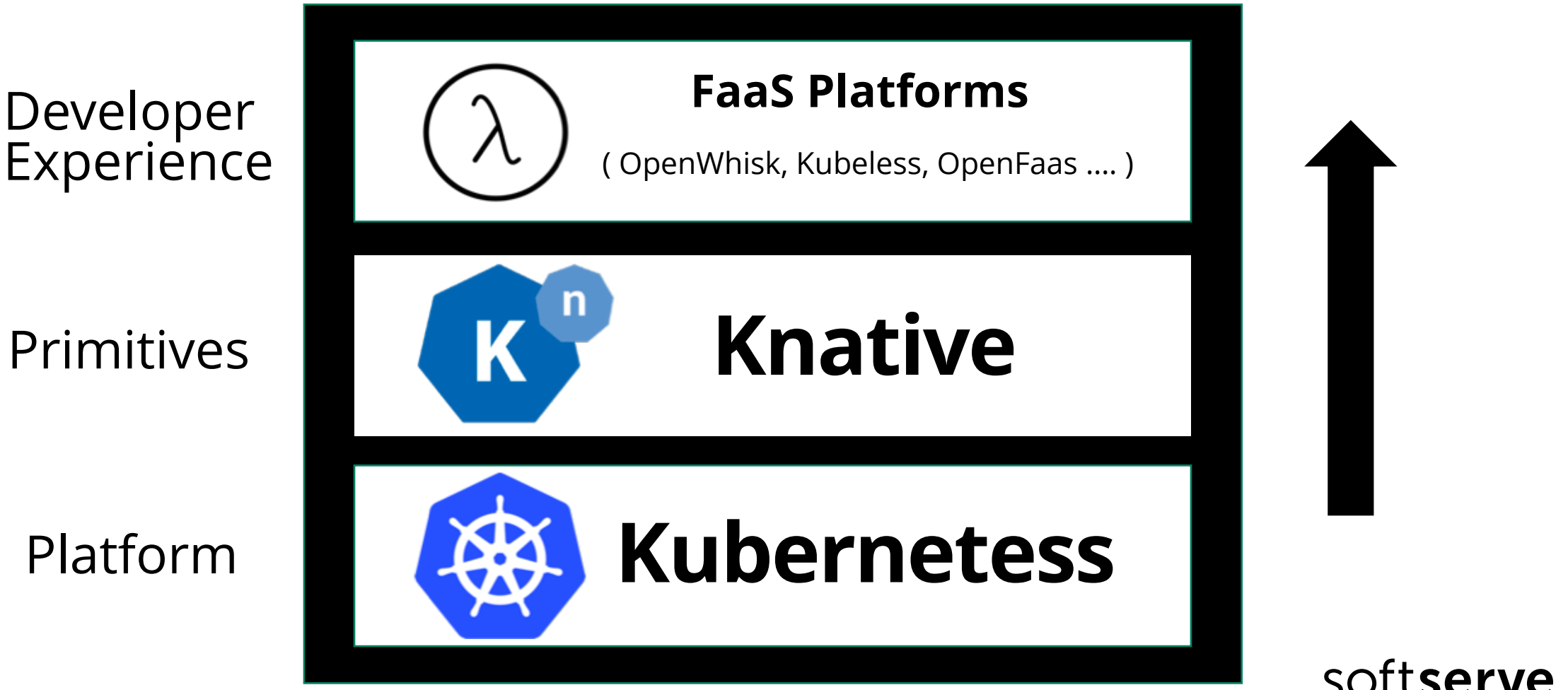
Knative is a collection of open source building blocks for serverless containers running on Kubernetes.

At this point, you might be wondering: “Kubernetes, serverless, what’s going on?” But, when you think about it, it makes sense. Kubernetes is hugely popular container management platform. Serverless is how application developers want to run their code. Knative brings the two worlds together with a set of building blocks.

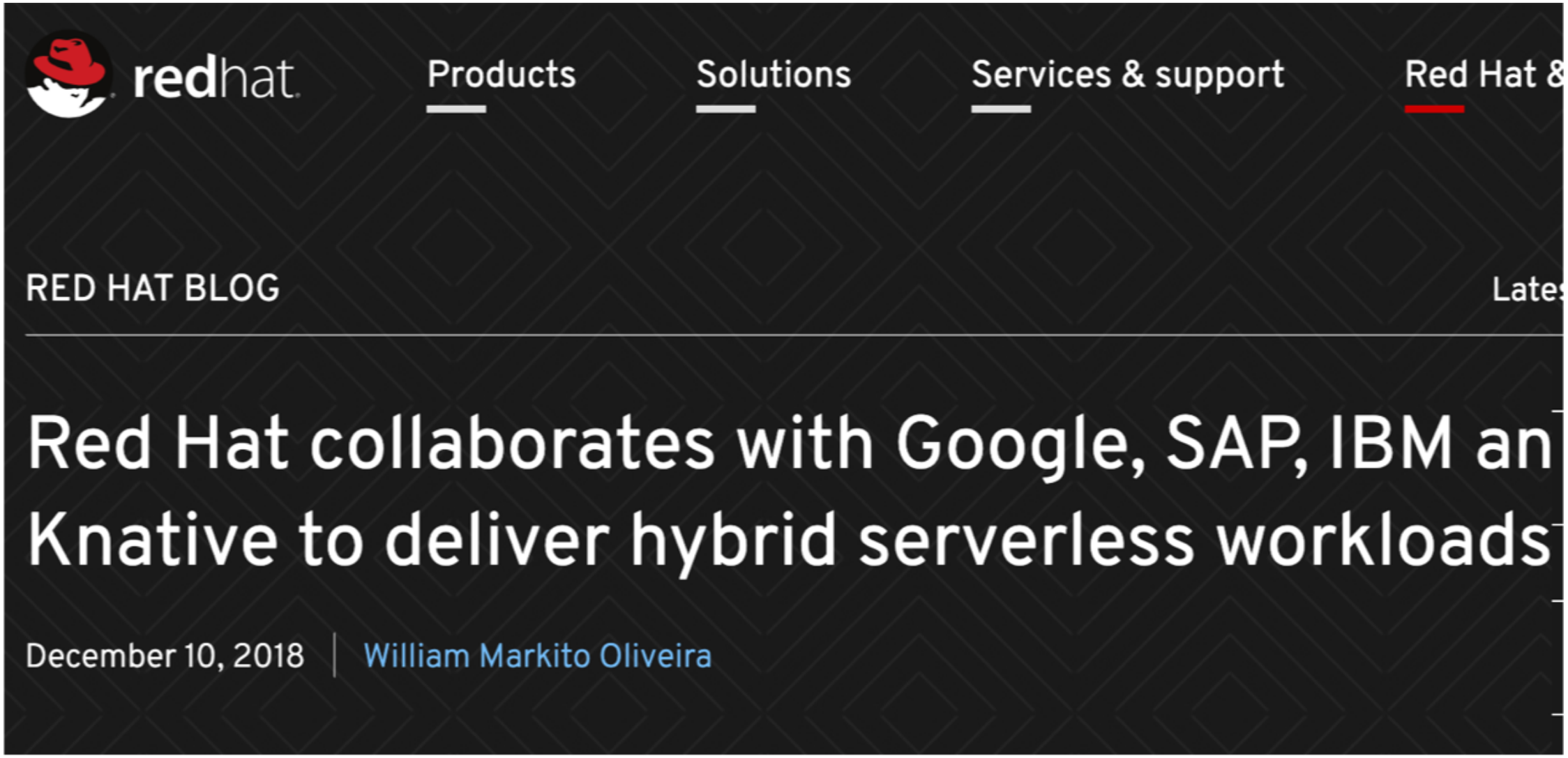
Source: https://medium.com/google-cloud/hands-on-knative-part-1-f2d5ce89944e

softserve

Developer Experience with Knative



softserve

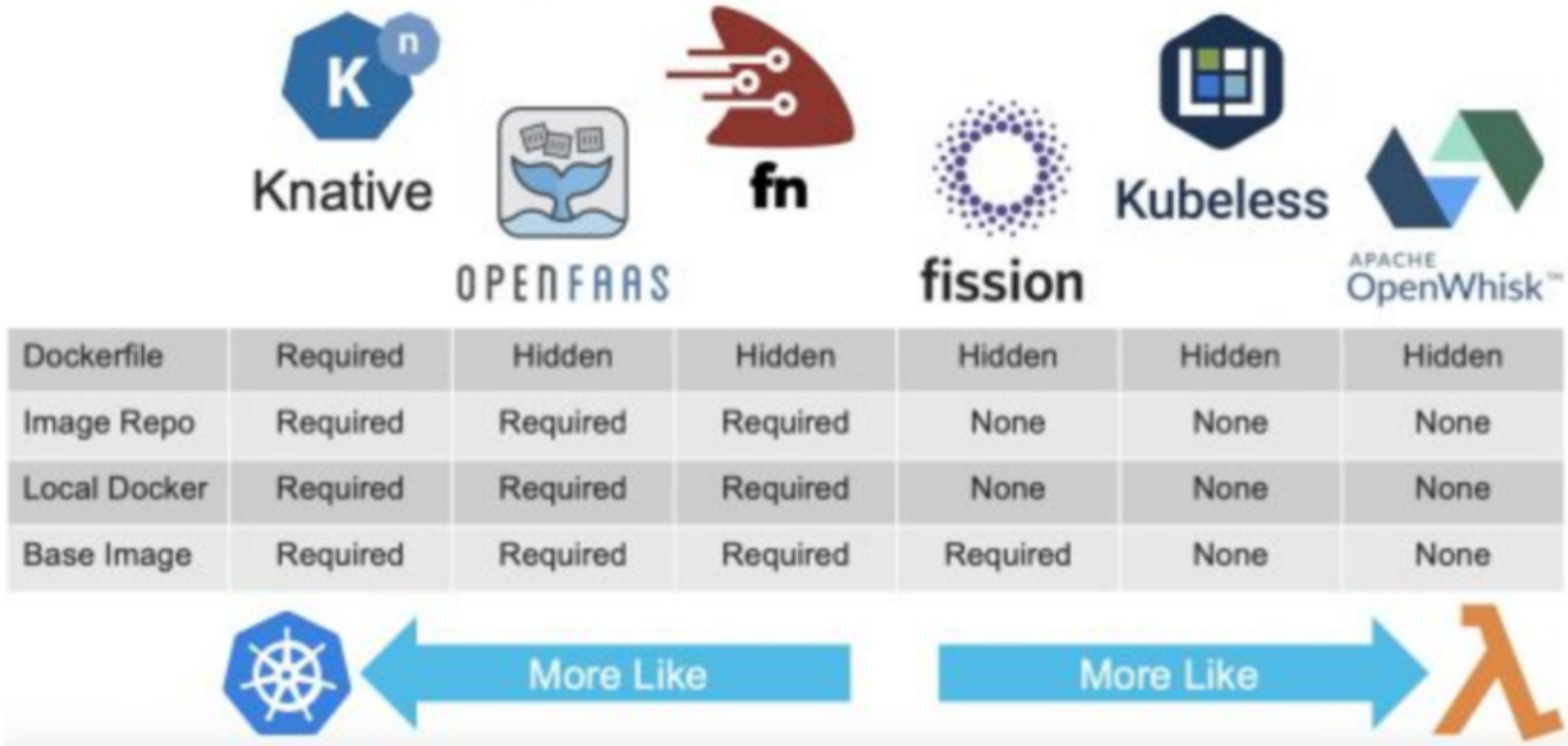


Source: https://www.redhat.com/en/blog/red-hat-collaborates-google-and-others-knative-deliver-hybrid-serverless-workloads-enterprise

softserve

Source: https://www.ibm.com/blogs/bluemix/2019/02/announcing-managed-knative-on-ibm-cloud-kubernetes-service-experimental/

softserve



Dockerfile	Required	Hidden	Hidden	Hidden	Hidden	Hidden
Image Repo	Required	Required	Required	None	None	None
Local Docker	Required	Required	Required	None	None	None
Base Image	Required	Required	Required	Required	None	None

Source: https://blogs.cisco.com/cloud/examining-the-faas-on-k8s-market

softserve

Components

softserve

Components

Serving

How your code receives requests and scales with them

DETAILS

- Request-driven compute runtime
- Scale-to-zero / scale out per load
- Deploy from container registry
- Multiple revisions of same app
- Route traffic across revisions

Build

How your code is built and packaged as a container

DETAILS

- Pluggable model to build containers from source code
- Build in-cloud or on-cluster
- Push image to registry
- Templates available (e.g. Buildpacks)

Eventing

How your code is triggered by events

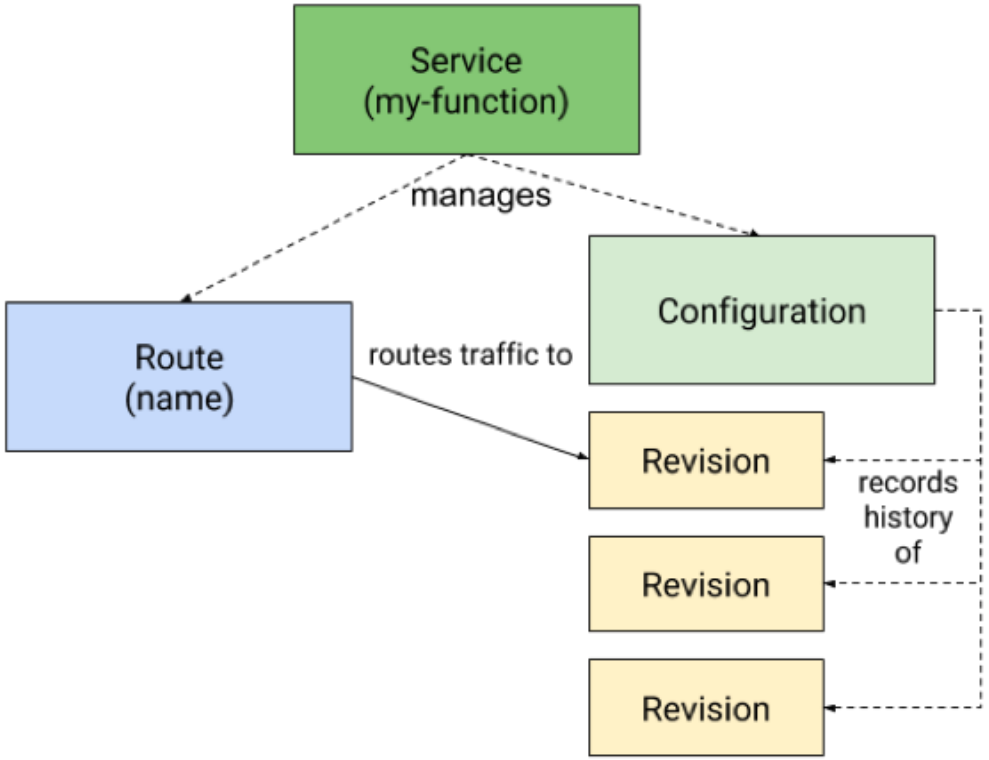
DETAILS

- Apps and functions consume and publish event streams
- Multiple event sources available
- Encourages asynchronous, loosely coupled architecture

softserve

Serving

- **Laverages Istio, Jaeger etc.**
- **Services** (not K8s services !) are top-level controllers that manage a set of Routes and Configurations to implement a network service.
- **Configurations** - desired state for your deployment. Configuration change creates new revision.
- **Revision -** snapshot of code and configuration
- **Routes –** configuration of ingress controller over a collection of Revisions and/or Configurations



Source: https://www.knative.dev/docs/serving/

softserve

Serving

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: helloworld-python
  namespace: default
spec:
  runLatest:
    configuration:
      revisionTemplate:
        spec:
          container:
            image: docker.io/{username}/helloworld-python
            env:
              - name: TARGET
                value: "Python Sample v1"
```

Source: https://www.knative.dev/docs/serving/samples/hello-world/helloworld-python/index.html

softserve

Serving - autoscaling

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: autoscale-go
  namespace: default
spec:
  runLatest:
    configuration:
      revisionTemplate:
        metadata:
          annotations:
            # Knative concurrency-based autoscaling (default).
            autoscaling.knative.dev/class:  kpa.autoscaling.knative.dev
            autoscaling.knative.dev/metric: concurrency
            # Target 10 requests in-flight per pod.
            autoscaling.knative.dev/target: "10"
            # Disable scale to zero with a minScale of 1.
            autoscaling.knative.dev/minScale: "1"
            # Limit scaling to 100 pods.
            autoscaling.knative.dev/maxScale: "100"
```

Source: https://www.knative.dev/docs/serving/samples/autoscale-go/index.html

softserve

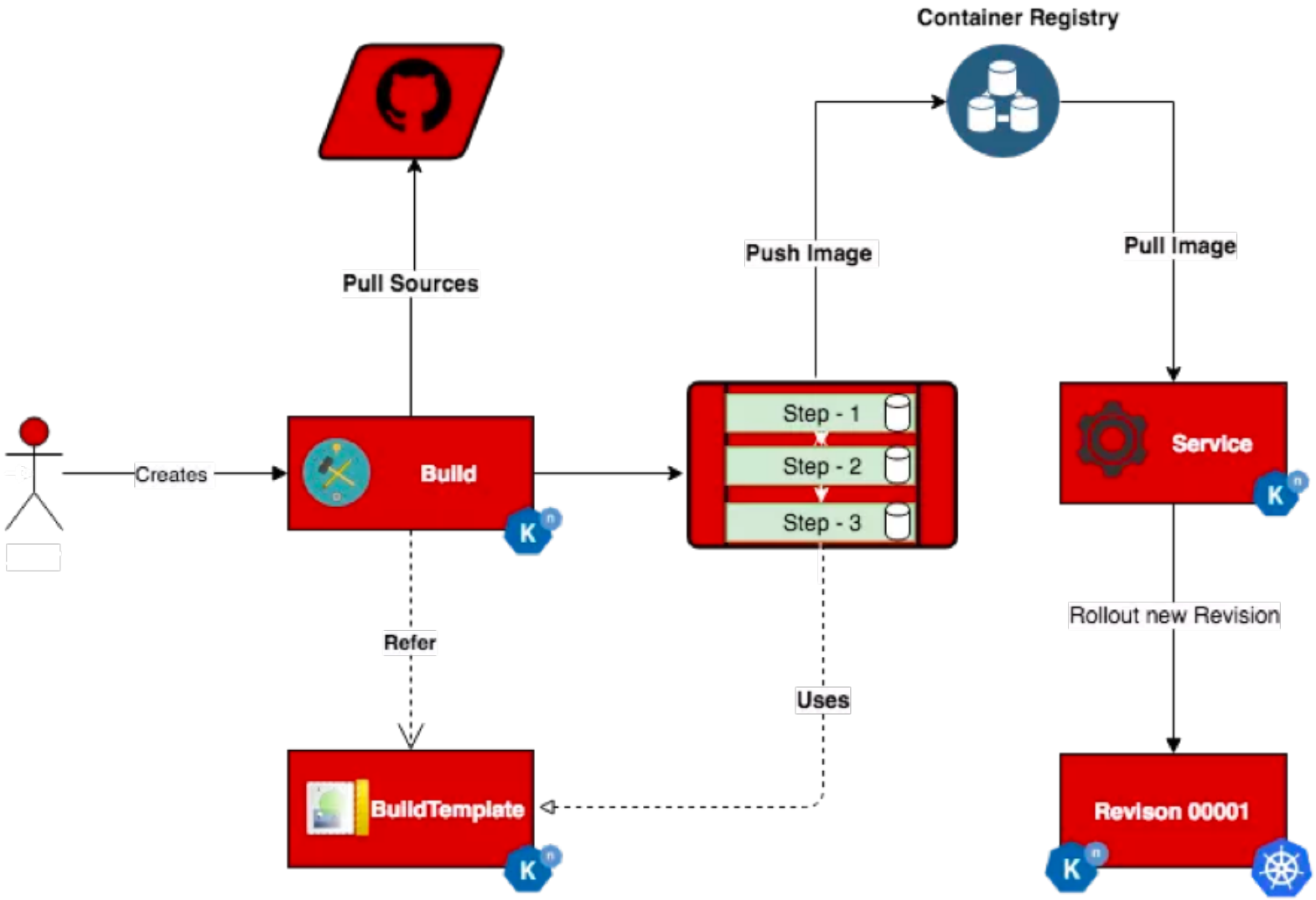
Serving - routes

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Route
metadata:
  name: blue-green-demo # Route name is unchanged, since we're updating an existing Route
  namespace: default
spec:
  traffic:
    - revisionName: blue-green-demo-00001
      percent: 100 # All traffic still going to the first revision
    - revisionName: blue-green-demo-00002
      percent: 0 # 0% of traffic routed to the second revision
      name: v2 # A named route
```

Source: https://www.knative.dev/v0.3-docs/serving/samples/blue-green-deployment/

softserve

Build



Source: https://blog.openshift.com/knative-building-your-serverless-service/

softserve

Build

- Builds are ran completely within Kubernetes
- Code is pulled from git at build time
- Packaged as container images and pushed to a registry of your choice
- **BuildTemplates** provide reusable,parameterized recipes that can be reused to create others
- Multiple **steps** suport (pipelines ?)

softserve

Build

```yaml
apiVersion: build.knative.dev/v1alpha1
kind: Build
spec:
  serviceAccountName: build-bot
  source:
    git:
      url: https://github.com/mchmarny/simple-app.git
      revision: master
  template:
    name: kaniko
    arguments:
      - name: IMAGE
        value: docker.io/{DOCKER_USERNAME}/app-from-source:latest
  timeout: 10m
```

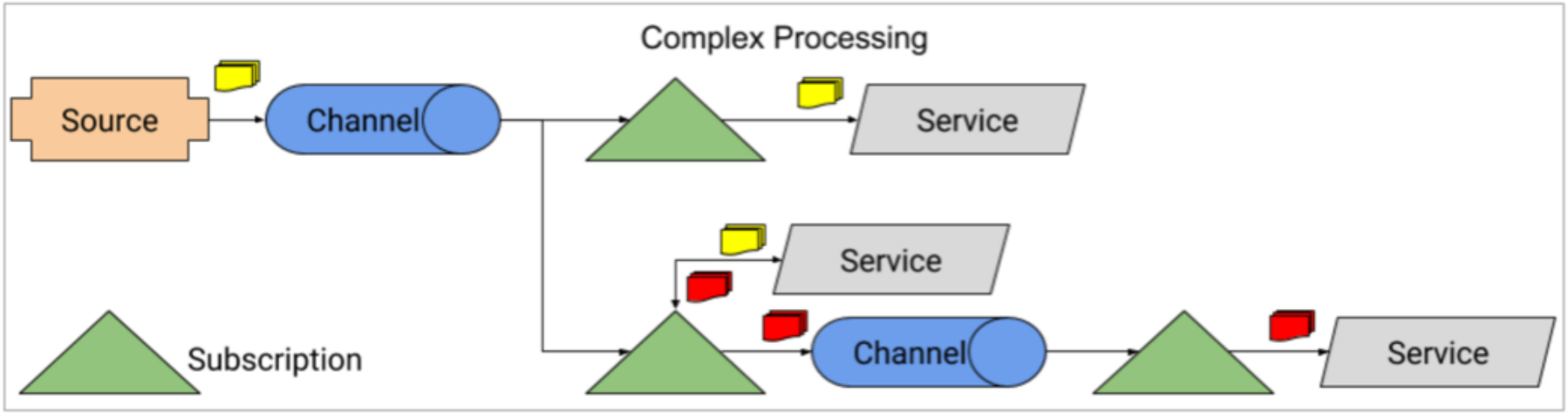
softserve

Eventing

- Implements CNCF Cloudevents spec (https://cloudevents.io/)
- Easy to consume events - no messaging-specific code
- Messaging layer is abstracted from the developer and pluggable for the operator
- Abstract persistent Channels allows events to be delivered to multiple functions
- Pluggable, Customizable Event Sources

Eventing

- **Source** – event producer
- **Consumers** - adressable (receive and ACK) or callable (receive and transform)
- **Channel -** named endpoint for event forwarding and persistence layer (Kafka, SQS etc.)
- **Subscription -** registration between channels and services or other channels



Source: https://www.knative.dev/docs/eventing/

softserve

Eventing

```yaml
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: GcpPubSubSource
metadata:
  name: testing-source
spec:
  gcpCredsSecret:  # A secret in the knative-sources namespace
    name: google-cloud-key
    key: key.json
  googleCloudProject: MY_GCP_PROJECT  # Replace this
  topic: testing
  sink:
    apiVersion: eventing.knative.dev/v1alpha1
    kind: Channel
    name: pubsub-test
```

```yaml
apiVersion: eventing.knative.dev/v1alpha1
kind: Channel
metadata:
  name: pubsub-test
spec:
  provisioner:
    apiVersion: eventing.knative.dev/v1alpha1
    kind: ClusterChannelProvisioner
    name: in-memory-channel
```

Source: https://www.knative.dev/docs/eventing/samples/gcp-pubsub-source/

softserve

Eventing

```yaml
apiVersion: eventing.knative.dev/v1alpha1
kind: Subscription
metadata:
  name: gcppubsub-source-sample
spec:
  channel:
    apiVersion: eventing.knative.dev/v1alpha1
    kind: Channel
    name: pubsub-test
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1alpha1
      kind: Service
      name: message-dumper
```

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: message-dumper
spec:
  runLatest:
    configuration:
      revisionTemplate:
        spec:
          container:
            # This corresponds to
            # https://github.com/knative/eventing-sources/blob/release-0.3/cmd/message_dumper/dumper.go
            image: gcr.io/knative-releases/github.com/knative/eventing-sources/cmd/message_dumper:latest
```

```
[~/Desktop] gcloud pubsub topics publish testing --message="Hello world"
```

Source: https://www.knative.dev/docs/eventing/samples/gcp-pubsub-source/f

softserve

Eventing

```
27  func (md *MessageDumper) ServeHTTP(w http.ResponseWriter, r *http.Request) {
28          w.WriteHeader(http.StatusOK)
29          if reqBytes, err := httputil.DumpRequest(r, true); err == nil {
30                  log.Printf("Message Dumper received a message: %+v", string(reqBytes))
31                  w.Write(reqBytes)
32          } else {
33                  log.Printf("Error dumping the request: %+v :: %+v", err, r)
34          }
35  }
36
37  func main() {
38          http.ListenAndServe(":8080", &MessageDumper{})
39  }
```

Source: https://github.com/knative/eventing-sources/blob/release-0.3/cmd/message_dumper/dumper.go

softserve

Market adoption

Pages / OpenWhisk Project Wiki / Proposals

[WIP] OpenWhisk on Knative

Created by Benjamin M. Browning, last modified on Aug 13, 2018

https://cwiki.apache.org/confluence/display/OPENWHISK/%5BWIP%5D+OpenWhisk+on+Knative

Knative Example (Apache Camel K)

This example shows how Camel K can be used to connect Knative building blocks to create awesome applications.

A video version of this demo is available on YouTube.

https://github.com/apache/camel-k/tree/master/examples/knative

softserve

Furher reading

- https://medium.com/google-cloud/hands-on-knative-part-1-f2d5ce89944e
- https://www.knative.dev/docs/
- https://winderresearch.com/a-comparison-of-serverless-frameworks-for-kubernetes-openfaas-openwhisk-fission-kubeless-and-more
- https://starkandwayne.com/blog/introduction-to-knative/
- https://medium.com/tag/knative
- https://redhat-developer-demos.github.io/knative-tutorial/knative-tutorial/v1.0.0/index.html
- https://itnext.io/how-to-use-knative-to-automate-an-application-build-on-kubernetes-904034341f9f

softserve

My (biased) opinion

It’s too early to say if Knative might change Serverless world...

but it’s definitely worthy to try and keep an eye on it.


Questions?
softserve
Robots Machines
Create your superposition
softserve

Thank You !

softserve