Stop calling Knative Serverless!

Doug Davis
IBM - STSM - OM Knative
dug@us.ibm.com | @duginabox
Why Cloud Native?

- Break-up the monolith
  - Better resource utilization
  - Reduced costs
- Abstraction of infrastructure
  - Devs focus on code not infrastructure
  - Faster time to market = $

Choices of infrastructure ...

- Platform as a Service  e.g. CloudFoundry
- Containers as a Service e.g. Kubernetes
- Functions as a Service/Serverless e.g. OpenWhisk

Which do you use?
### Characteristics of Platform as a Service

<table>
<thead>
<tr>
<th>Feature</th>
<th>PaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified UX</td>
<td>✓</td>
</tr>
<tr>
<td>Containers</td>
<td>✓</td>
</tr>
<tr>
<td>Micro-services / small-ish tasks</td>
<td>✓</td>
</tr>
<tr>
<td>Stateless</td>
<td>✓</td>
</tr>
<tr>
<td>Endpoint + Load Balancing</td>
<td>✓</td>
</tr>
<tr>
<td>Build</td>
<td>✓</td>
</tr>
<tr>
<td>Pay per usage (public cloud)</td>
<td>✓</td>
</tr>
<tr>
<td>On-demand infrastructure - auto-scaling</td>
<td>✓</td>
</tr>
<tr>
<td>Access to advanced features (eg. net/vol)</td>
<td></td>
</tr>
</tbody>
</table>
# Characteristics of C/F as a Service

<table>
<thead>
<tr>
<th>Feature</th>
<th>PaaS</th>
<th>CaaS</th>
<th>Fn/Srvr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified UX</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Containers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Micro-services / small-ish tasks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Stateless</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Endpoint + Load Balancing</td>
<td>✓</td>
<td>DIY</td>
<td>✓</td>
</tr>
<tr>
<td>Build</td>
<td>✓</td>
<td>DIY</td>
<td>✓</td>
</tr>
<tr>
<td>Pay per usage (public cloud)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>On-demand infrastructure - auto-scaling</td>
<td>✓</td>
<td>DIY</td>
<td>✓</td>
</tr>
<tr>
<td>Access to advanced features (eg. net/vol)</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Event driven/tooling</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scale to zero</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Async invocations</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-restrictive task execution times</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Non-restrictive resource usage (eg. mem)</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

*Why be forced to choose?*
The need for something different...

Kubernetes: Most popular Container Management project today. Deploy and manage your containerized applications.

**Resources**
- Containers
- Pods
- Replica Sets
- Deployments
- Services
- Endpoints
- Secrets
- Networks
- Volumes/PV/PVC
- Ingress/LBs

**Tooling & Other "stuff"**
- yaml
- Spec vs Status
- helm
- kubectl
- Istio
- ...

- Blue/green deployments

**The promise of Cloud Computing:**
- Higher level abstractions
- Management of infrastructure for me
  - "Decreasing concern (and control) over infrastructure implementation"

I just want to deploy my code!
Introducing Knative

• An opinionated and simplified view of application/container management

• Allowing developers to focus on coding

• Built as an extension to Kubernetes
  • Still Kubernetes under the covers
  • Access to full Kubernetes if needed
  • Integrates with the non-Knative workloads

• Building blocks on which Cloud Providers can build a platform
Knative

2 Main Components

• **Serving** is the runtime component
  • Host your application as K8s pods

• **Eventing** contains tools for managing events
  • Between loosely coupled services
**Knative Serving - The Model**

- Deploy app as pod/revision
  - Revision specific config
  - E.g. image, env vars, scale
- Networking auto-setup
- Revisions are scaled up/down
  - Based on load
  - Even down to zero
- Updates create Revisions
  - Auto-migration to new
- Traffic splitting based on %
- Dedicated URLs to Revisions

And Knative manages all of these resources for you!
Knative - Demo - The Core
Knative - Demo - The Core - Summary

- Deploy Service with just "name" and "image"
- Access it via HTTP and HTTPS
- Auto-scaled up and down (to zero) based on incoming load
- Rolling upgrade to new version of Service
- Traffic split (e.g. 50/50) between versions
- Dedicated URL to specific version
- Exclude specific version from traffic router

- What would it take to do all of this via vanilla Kubernetes?
Knative Serving - Things to know

• Container Images MUST run HTTP servers

• Multi-threaded model - but configurable

• Configuration options
  • Container Concurrency
  • Min / Max / Target Scale
  • Scaling: requests vs cpu

• Simplified resource model

```yaml
apiVersion: serving.knative.dev/v1beta1
kind: Service
metadata:
  name: echo
spec:
template:
  spec:
    containers:
      - image: duglin/echo
```
Knative Eventing - Core

- Eventing Primitives
  Manage the coordination/delivery of events

  - **Event Source** - connects Event Producer to "sink"
    - Create the subscription for you
    - Often, creates an Adapter (KnService) to receive the events and convert them into CloudEvents

  - **Broker** - a receiver of events
    - E.g. a queue

  - **Trigger** - (subscription) ask for events from a Broker
    - Filters - to subset the stream of events
      Often based on CloudEvent's metadata
Knative Eventing - Other Features

• **Sequence**
  • Ordered set of "Sinks"
  • Final "Reply"

• **Choice**
  • "If" / "switch" statement

• **Event Registry**
  • Collection of EventTypes (Kn Broker, CloudEvent type/source/schema)

• **CloudEvents** - [https://cloudevents.io](https://cloudevents.io)
  • Regardless of Producer, Transport or Format - there's consistent metadata
Knative - Demo - Build Service

- Deploy CI Dashboard KnSvc
  - Concurrency = 1
- Deploy a "Build" KnSvc
- Create a Github Event Source
  - Notified of new Commits
  - Sink = Build KnSvc
- Dev commits & checks status
- Build Service scales
- Do it!
## Characteristics of * as a Service

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>PaaS</th>
<th>CaaS</th>
<th>Fn/Srvr</th>
<th>Knative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified UX</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Containers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Micro-services / small-ish tasks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Stateless</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Endpoint + Load Balancing</td>
<td>✓</td>
<td>DIY</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Build</td>
<td>✓</td>
<td>DIY</td>
<td>✓</td>
<td>WIP</td>
</tr>
<tr>
<td>Pay per usage (public cloud)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>On-demand infrastructure - auto-scaling</td>
<td>✓</td>
<td>DIY</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Access to advanced features (eg. net/vol)</td>
<td></td>
<td>✓</td>
<td>WIP</td>
<td></td>
</tr>
<tr>
<td>Event driven/tooling</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Scale to zero</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Async invocations</td>
<td></td>
<td>✓</td>
<td>WIP</td>
<td></td>
</tr>
<tr>
<td>Non-restrictive task execution times</td>
<td>✓</td>
<td>✓</td>
<td>WIP</td>
<td></td>
</tr>
<tr>
<td>Non-restrictive resource usage (eg. mem)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Knative - In Summary

• No longer have to choose which *aaS

• Get the best of
  • CaaS, PaaS, FaaS & Serverless
  • Simplified Kubernetes Experience

• Without losing the option of full Kubernetes when needed

• Your decision: architecture of your application / containers
  • Size of your containers
  • Boundaries between them
  • Which bits to scale and when
Thank You!

Share your experiences on adopting open source technology and get paid $185!

Come join the chat sessions hosted by IBM Design | Research. Share your thoughts, meet fellow developer attendees, and get paid $185. Sign up here:

https://bit.ly/2Mr0D6h