Go containers? Go serverless?
A cloud native journey

Michael Hausenblas, Developer Advocate, Red Hat
@mhausenblas

2019-02-02, FOSDEM Go dev room
$ whois mhausenblas

• Developer Advocate @ Red Hat (Go, Kubernetes), Cloud Native Ambassador @ CNCF
• Developer Advocate @ Mesosphere (Mesos, DC/OS, Kubernetes)
• Chief Data Engineer @ MapR (HDFS, HBase, Drill, etc.)
• Applied research—4y in Ireland, 7y in Austria (Linked Data, media semantics)
• Nowadays mainly developing tools in Go (before: Python, Node.js, Java, C++)
• Developer by trade, ops by passion
The Monolith
imgn!

Upload

Hit me up on Twitter: @mhausenblas
imgn!

Upload

Hit me up on Twitter: @mhausenblas
Act 1
Containerized Microservices
demo time!

github.com/mhausenblas/imgn/containers
Kubernetes

• Container lifecycle management
• Declarative API + control loops
• Extensible through plug-ins and custom resources/controllers
imgn as containerized microservices app

- Microservices:
  - serving static assets for UI, exposing HTTP API:
    - upload image
    - list images + metadata
  - batch metadata extraction
- Using Minio (S3 clone) as shared storage for images and metadata
- Leverages vanilla Kubernetes abstractions (deployments, services)
Alternative container-based architectures

• lift and shift (aka: put existing monolith into a pod)
• both microservices as containers in one pod, share data via local volume
• app containers in different pods, share data via persistent volume
Act 2

Serverless
demo time!

github.com/mhausenblas/imgn/functions
Serverless

Function-as-a-Service (FaaS)
AWS Lambda, knative, etc.

databases & datastores
Redshift, Azure Data Lake, etc.

object storage
S3, Google Cloud Storage, etc.

message queues
SQS, Azure Service Bus, etc.

query
BigQuery, Amazon Athena, etc.

boringis.cool/#lets-talk-about-serverless
Function-as-a-Service concept

- event-driven (i.e. needs trigger)
- short-running (practically minutes)
- stateless (externalize state/integrations)
imgn serverless

• Three Lambda functions:
  • upload image
  • list images + metadata
  • metadata extraction

• Using S3 as shared storage for images and metadata and static assets for UI
• Leverages SAM to deploy Lambda functions, API Gateway, CloudWatch
Alternative serverless architectures

• trigger metadata extraction on s3:ObjectCreated event rather than periodically via CloudWatch event
• replace two-phased pre-signed URL/S3 bucket put flow with a Amazon Cognito-based flow using AWS SDK for JavaScript
Comparison
## Containers and serverless: similar, yet different

<table>
<thead>
<tr>
<th></th>
<th>Containerized microservices</th>
<th>serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>unit of deployment</strong></td>
<td>pod</td>
<td>function</td>
</tr>
<tr>
<td><strong>build artifact</strong></td>
<td>container image</td>
<td>ZIP, JAR file</td>
</tr>
<tr>
<td><strong>artifact distribution</strong></td>
<td>container registry</td>
<td>S3 buckets</td>
</tr>
<tr>
<td><strong>event triggers</strong></td>
<td>no built-ins, requires framework</td>
<td>built-in (API Gateway, S3 buckets, etc.)</td>
</tr>
<tr>
<td><strong>state</strong></td>
<td>can be stateful, requires some effort</td>
<td>stateless, but lots of integrations</td>
</tr>
</tbody>
</table>
## Containers and serverless: similar, yet different

<table>
<thead>
<tr>
<th></th>
<th>Containerized microservices (Kubernetes)</th>
<th>Serverless (AWS Lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>generally good</td>
<td>can be challenging</td>
</tr>
<tr>
<td>Observability</td>
<td>not opinionated, integration points</td>
<td>AWS specific</td>
</tr>
<tr>
<td>Billing</td>
<td>pay for resources, no matter if used or not</td>
<td>pay what you consume</td>
</tr>
<tr>
<td>Lift and shift</td>
<td>possible</td>
<td>no</td>
</tr>
<tr>
<td>Local development</td>
<td>doable</td>
<td>limited</td>
</tr>
</tbody>
</table>
The good

- Containerized microservices: Kubernetes provides portability
- Serverless: developers can focus on business logic
- Both containers and serverless:
  - ratio of “code to config” (YAML manifests) comparable
  - increase dev/deploy velocity
The bad

• containerized microservices:
  • handling container images
  • DX is poor

• serverless:
  • language dependent latencies
  • state hydration
Resources
Articles and blog posts

• Understanding Cloud Native Infrastructure
  infoq.com/articles/cloud-native-infrastructure

• 5 reasons you should be doing container native development
  open.microsoft.com/2018/04/23/5-reasons-you-should-be-doing-container-native-development/

• Developing on Kubernetes
  kubernetes.io/blog/2018/05/01/developing-on-kubernetes/

• What is a Service Mesh, and Do I Need One When Developing Cloud Native Systems?
  skillsmatter.com/skillscasts/10668-looking-forward-to-daniel-bryant-talk
Articles and blog posts

• Best AWS Lambda Use Cases From Video Processing To Predictive Page Rendering
dashbird.io/blog/best-aws-lambda-serverless-use-cases/

• Learn about serverless with these books, videos, and tutorials
oreilly.com/ideas/learn-about-serverless-with-these-books-videos-and-tutorials

• AWS Lambda and the Monolith
medium.com/@ryannedolan/aws-lambda-and-the-monolith-a0eb2d1516ef

• From Monolith to Microservices — Part 1: AWS Lambda and API Gateway
articles.microservices.com/from-monolith-to-microservices-part-1-aws-lambda-and-api-gateway-8ce5cf3f0d99