Serverless? Not so FaaS!

Matt Stine (@mstine)
Global Field CTO / Chief Architect
mstine@pivotal.io
http://mattstine.com
Who is this guy?

- 18 year Enterprise IT veteran developer / architect
- 6 years focused on Cloud Native Platforms and Architecture
- Frequent conference circuit speaker
- Host of Software Architecture Radio http://softwarearchitecturerad.io
- I wrote a little cloud book...
Agenda

- The Four Abstractions
- FaaS vs. PaaS
- Use Cases
- Tradeoffs
- Provider Updates
- Rise of Kubernetes
- Java / Spring
- Smattering of Demos
Choose the right tool for the job

Container Orchestrator
- Container Scheduling
- Primitives for Network, Routing, Logs & Metrics

Application Platform
- Container Orchestrator
- Container Image & build
- L7 Network & Routing
- Logs, Metrics, Monitoring
- Services Marketplace
- Team, Quotas & Usage

Serverless Functions
- Application Platform
- Function scheduling
- Function exec services

Developer Provides

IaaS
Choose the right tool for the job

Developer Provides

CONTAINER

- Container Scheduling
- Primitives for Network

APPLICATION

- Container Orchestrator
- L7 Network & Routing
- Logs, Metrics, Monitoring
- Services Marketplace
- Team, Quotas & Usage

FUNCTION

- Application Platform
- Function scheduling
- Function exec services

Tool Provides

Lower complexity, higher efficiency

Higher flexibility, lower standardization

IaaS
**Strategic goal:** Push as many workloads as technically feasible to the top of the platform hierarchy.

Higher flexibility and less enforcement of standards

Lower development complexity and higher operational efficiency
I’m starting prototypes using a pure serverless approach then migrating “functions” to containers when I need lower latency or constant throughputs for long periods of time.
# What is FaaS?

**Developer / Operator experience**

<table>
<thead>
<tr>
<th>PaaS</th>
<th>FaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>runs <strong>apps</strong></td>
<td>runs <strong>functions</strong></td>
</tr>
<tr>
<td>push app to deploy server</td>
<td>register function and bind it to a trigger</td>
</tr>
<tr>
<td>server runs and waits for requests</td>
<td>function doesn't run until triggered</td>
</tr>
<tr>
<td>server listens to network</td>
<td>platform deploys and invokes functions</td>
</tr>
<tr>
<td>server handles lots of requests</td>
<td>functions handle <strong>events</strong> and then go away</td>
</tr>
<tr>
<td>scale out manually or by policy</td>
<td><strong>auto-scale</strong> based on concurrent event load</td>
</tr>
<tr>
<td>pay per instance</td>
<td>pay per use – time &amp; memory</td>
</tr>
</tbody>
</table>
Serverless is telling us “there’s even more you don’t care about”.

Dave Syer
Serverless Spring
S1P 2017
Three types of efficiencies enabled by Serverless

- Narrowly-scoped units of code, and built-in event integration, contribute to software development efficiencies.
- Functions which don't consume resources when idle can provide significant resource efficiencies.
- Applying serverless to distributed computing brings operational efficiencies based on automated event-based scheduling and self-scaling.

Pivotal
How do we even begin to learn how to scale the complexity of our distributed systems?

What if we could make systems that connect, and deploy, and monitor, and scale themselves?

But serverless is not just about efficiencies
### Three Categories of Use Cases

#### Web Events
- Website back-end services like form post handlers
- Back-end data services for mobile and web apps e.g. GraphQL
- Webhook handlers
- CI/CD automation
- Chat integrations
- Digital assistant services e.g. Alexa skills

#### Event-based Integration
- Scheduled tasks, ETL
- File processing e.g. images and videos
- Complex Event Processing and Change Data Capture
- Monitoring, notifications and alerting
- Custom auth e.g. via API Gateway

#### Large Scale Data Processing
- e.g. pyWren map/reduce
- IoT streams
- Log ingestion
- Machine Learning
- Stateful Stream Processing
# Tradeoffs

<table>
<thead>
<tr>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Control</td>
<td>Rise of Open Source</td>
</tr>
<tr>
<td>Vendor Lock-In</td>
<td>Rise of Abstraction Layers</td>
</tr>
<tr>
<td>Stateless!</td>
<td>Rumors of Statefulness</td>
</tr>
<tr>
<td>Maturity</td>
<td>Starting to Grow Up</td>
</tr>
<tr>
<td>Developer UX</td>
<td>Developer UX Improving!</td>
</tr>
</tbody>
</table>

*Pivotal.*
AWS Lambda

https://aws.amazon.com/lambda/

- First FaaS, introduced in Nov 2014.
- Entry tier pricing attractive for low-volume Web handlers eg bots.
- Built-in triggers facilitate integration with many other AWS services.
- Auto-scales with concurrent events.
- Runs code written for Node.js, Python, Java 8, .NET Core, (and recently) Go.
- SAM (for declarative deployments) and SAM Local (for local simulation).
DEMO
mstine at eolian.local in ~/work [22:35:27]
mvn archetype:generate -DarchetypeGroupId=com.mattstine.aws -DarchetypeArtifactId=aws-lambda-archetype -DarchetypeVersion=1.0-SNAPSHOT
Azure Functions

Available since Nov 2016 with support for .NET and JavaScript based on WebJobs C# OSS runtime.

New v2 (.NET core) runtime in preview also supports Java and Linux or Windows containers.

Developers can use Core-Tools to build and run functions locally.

Included with Azure Stack.

https://azure.microsoft.com/services/functions/
# mstine at eolian.local in ~/work [23:56:37]
→ mvn archetype:generate
   -DarchetypeGroupId=com.microsoft.azure
   -DarchetypeArtifactId=azure-functions-archetype
Google Cloud Functions
https://cloud.google.com/functions/

- Still in Beta
- Supports node.js runs in a docker container performs npm install build step can deploy from git repo
- Blocks per request and scales using concurrency (like Lambda)
- Integrates with events from HTTP, Pub/Sub, Storage, and Firebase.
The Rise of Kubernetes

Kubernetes is an open-source platform designed to automate deploying, scaling, and operating application containers.

With Kubernetes, you are able to quickly and efficiently respond to customer demand:

- Deploy your applications quickly and predictably.
- Scale your applications on the fly.
- Roll out new features seamlessly.
- Limit hardware usage to required resources only.

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
Kubernetes has become the de facto standard for container orchestration.
Kubernetes FTW

- Amazon Elastic Container Service for Kubernetes (EKS)
- Azure Container Service (AKS)
- Google Kubernetes Engine (GKE)
FaaS Kubernetes Landscape

**Public Cloud**
- AWS Lambda
- Azure Functions
- Google Cloud Functions
- AKS + FaaS
- GKS + FaaS

**Kubernetes Native FaaS**
- Fission
- Kubeless

**Other OSS FaaS (Will Run on K8S)**
- Apache OpenWhisk
- OpenFaaS
- fn Project
- nuclio

Pivotal.
Other OSS FaaS / Will Run on K8S

- **Apache OpenWhisk** ([https://openwhisk.apache.org](https://openwhisk.apache.org))
  (packs the container for you, but not K8S native)

- **OpenFaaS** ([https://www.openfaas.com](https://www.openfaas.com))
  (uses Docker Swarm or K8S as backend runtime - you pack the container)

- **fn Project** ([https://fnproject.io](https://fnproject.io))
  (orchestrator agnostic - you pack the container)

- **nuclio** ([https://nuclio.io](https://nuclio.io))
  (runs on k8s, packs container for you, manually configured ingress)
Kubernetes Native FaaS

- **Fission** ([https://openwhisk.apache.org](https://openwhisk.apache.org))
  (Abstracts Docker/K8S, provides runtimes and triggers)

- **Kubeless** ([http://kubeless.io](http://kubeless.io))
  (K8S CRDs, Deployments/Pods, Config Map, Services)

- **Riff** ([https://projectriff.io](https://projectriff.io))
  (Let’s go deeper...)
What is riff?

- **riff is for functions!**
- riff provides developers with a FaaS for executing Functions triggered by Events.
- **riff is a new open source project**
- sponsored by Pivotal
- built by a team of engineers from Spring and PCF.

https://github.com/projectriff

Pivotal
Every 1.0s: kubectl get po, deploy --namespace riff-system

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>po/projectriff-kafka-855897b9b6-mgld6</td>
<td>1/1</td>
<td>Running</td>
<td>4</td>
<td>2d</td>
</tr>
<tr>
<td>po/projectriff-riff-function-controller-6cd454579b-lp5rp</td>
<td>1/1</td>
<td>Running</td>
<td>6</td>
<td>2d</td>
</tr>
<tr>
<td>po/projectriff-riff-http-gateway-bc46b8d8f-k5ssq</td>
<td>1/1</td>
<td>Running</td>
<td>6</td>
<td>2d</td>
</tr>
<tr>
<td>po/projectriff-riff-topic-controller-65f56c4ff-htlsg</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d</td>
</tr>
<tr>
<td>po/projectriff-zookeeper-5f898c6869-j2ghg</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESIRED</th>
<th>CURRENT</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploy/projectriff-kafka</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2d</td>
</tr>
<tr>
<td>deploy/projectriff-riff-function-controller</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2d</td>
</tr>
<tr>
<td>deploy/projectriff-riff-http-gateway</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2d</td>
</tr>
<tr>
<td>deploy/projectriff-riff-topic-controller</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2d</td>
</tr>
<tr>
<td>deploy/projectriff-zookeeper</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2d</td>
</tr>
</tbody>
</table>
Developers write functions

greeter.sh

```bash
echo "hello $1"
```

square.js

```javascript
module.exports = (x) => x ** 2
```
Functions are packaged as containers

(in riff 0.0.6, invokers become installable K8S resources)
Sidecars connect functions with event brokers

Event Broker

broker-specific API

Function Pod

Sidecar Container
- binder
- dispatcher

Function Container
- invoker
- function

HTTP

HTTP
gRPC

stdio
Functions and topics are Kubernetes resources

Function YAML
- name
- input / output topics
- artifact / params

Topic YAML
- name
- params

Kafka

Http Gateway
Functions scale with events

**riff function controller**
- Interacts with k8s API
- Monitors event-lag
- Scales functions 0-1 and 1-N
Spring Cloud Function

Spring Cloud Function is a project with the following high-level goals:

• Promote the implementation of business logic via functions.
• Decouple the development lifecycle of business logic from any specific runtime target so that the same code can run as a web endpoint, a stream processor, or a task.
• Support a uniform programming model across serverless providers, as well as the ability to run standalone (locally or in a PaaS).
• Enable Spring Boot features (auto-configuration, dependency injection, metrics) on serverless providers.

It abstracts away all of the transport details and infrastructure, allowing the developer to keep all the familiar tools and processes, and focus firmly on business logic.

https://cloud.spring.io/spring-cloud-function/
Spring Cloud Functions:

- Plain old java.util.function Functions, Consumers, Suppliers
- Registered as Beans via @FunctionScan
- Can leverage Reactor’s Flux (Reactive Streams Publisher)
- Adapters for Java FaaS providers (AWS, Azure, OpenWhisk)
Plain Old Java Functions

```java
@SpringBootApplication
public class Application {

    @Bean
    public Function<Flux<String>, Flux<String>> uppercase() {
        return flux -> flux.map(value -> value.toUpperCase());
    }

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```
# mstine at eolian.local in ~/workspace/spring-cloud-function/spring-cloud-function-samples/function-sample-pojo on git:72ef5d8  [1:55:43]

# mstine at eolian.local in ~/workspace/spring-cloud-function/spring-cloud-function-samples/function-sample-pojo on git:72ef5d8  [1:55:40]
Transforming How The World Builds Software

Matt Stine (@mstine)
Global Field CTO / Chief Architect
mstine@pivotal.io
http://mattstine.com
GREAT INDIAN DEVELOPER SUMMIT 2019

Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com  @greatindiandev  bit.ly/gidslinkedin