



Building Cloud Native Applications on Cloud Foundry

An in depth look at the microservices architecture pattern,
containers and Cloud Foundry

7: Orchestration

Objectives

- Define Orchestration
- Examine the CNCF cloud native platform reference model working draft
- Describe Mesos, Kubernetes and Docker Swarm
- Explain the role of the service
- Explore load balancing and service discovery

After Containers, what's left?

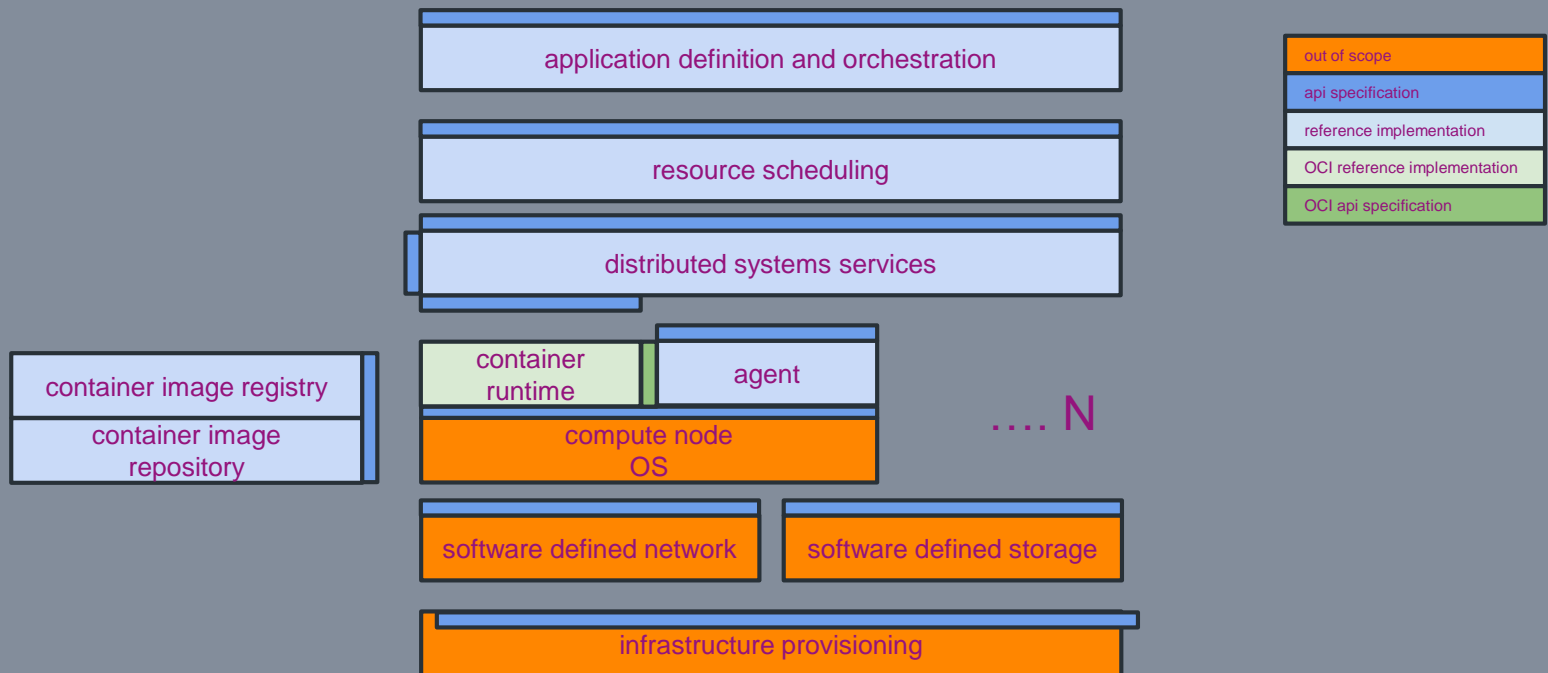
Copyright 2013-2018, RX-M LLC

■ Orchestration!

- Real cloud native applications are delivered in 10s of images and require 100s - 1000s of running containers
- **Registries** manage image distribution
- Orchestration manages:
 - **Container Scheduling**
 - Distributing containers to appropriate hosts
 - Host resource leveling
 - Availability zone diversity
 - Related container packaging and co-deployment
 - **Container Management**
 - Monitoring and recovery
 - Image upgrade rollout
 - Scaling
 - Logging
 - **Service Endpoints**
 - Discovery
 - HA
 - Load balancers
 - Auto scaling
 - **External Services**
 - Network configuration and management
 - Durable volume management



Cloud Native Parts

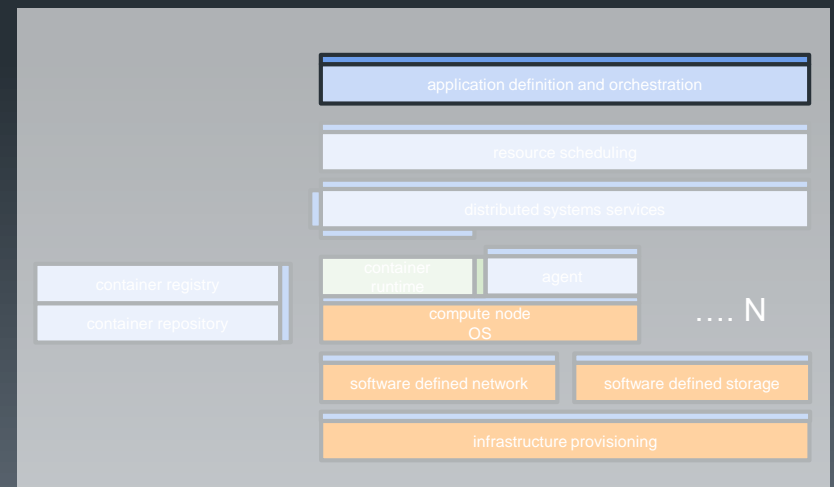


-- CNCF

Application definition and orchestration

Copyright 2013-2018, RX-M LLC

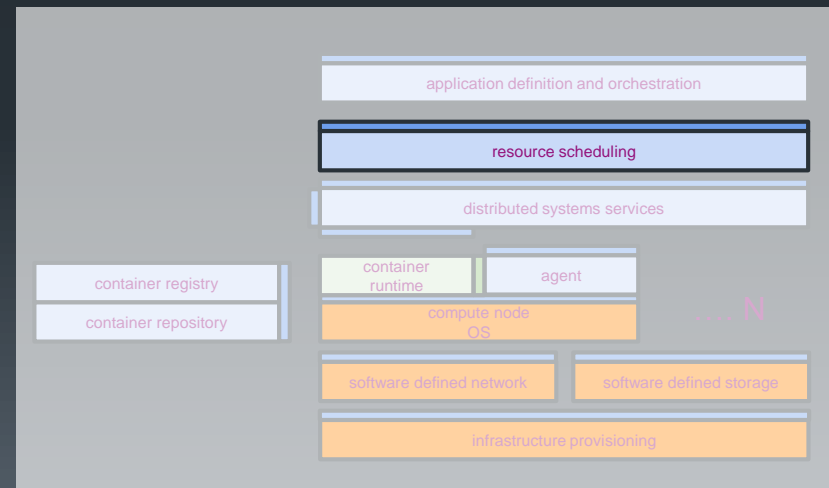
- standard service definition
 - define a standard distributed system service that can be deployed
- composite application definition
 - a standard model for packaging and shipping an app of many parts
- orchestrator
 - a standard framework to deploy and manage an app of many parts
 - providing workload/job specific orchestration and control



Resource scheduling

Copyright 2013-2018, RX-M LLC

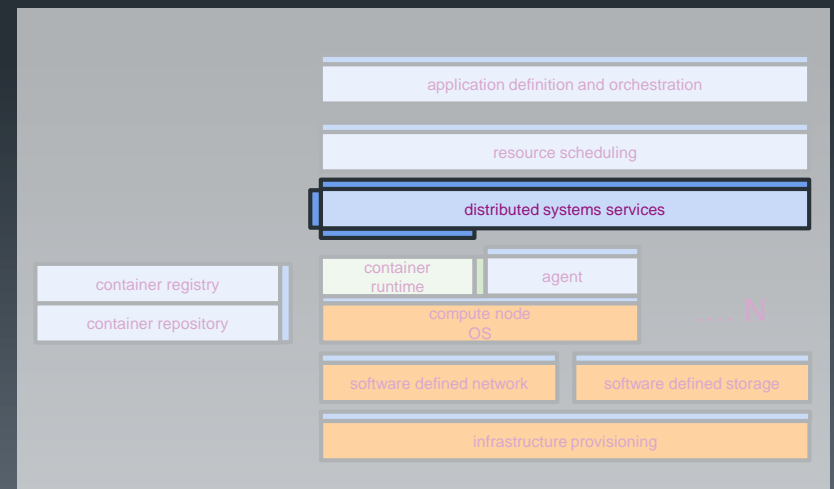
- map containers to nodes
- actively manage the health of a specific container
- handle active scaling of containers
- interface with infrastructure provisioning to request more resources
- interface with SDN and SDS to map to storage/network



Distributed systems services

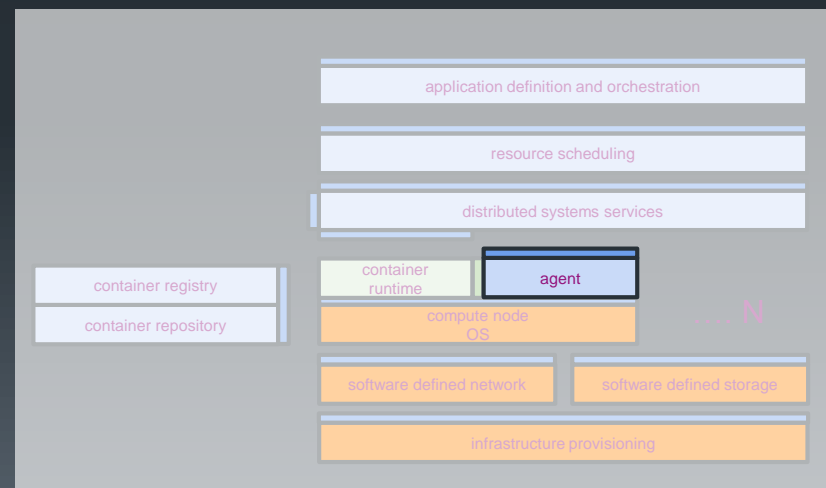
Copyright 2013-2018, RX-M LLC

- a standard set of services that are not bound to a single node
 - supporting application use cases
 - naming/discovery
 - locking/quorum
 - state management/sharding
 - logging/monitoring
 - supporting cluster use cases
 - distributed state management
 - distributed control plane
 - logging/auditing
- a minimum atom of consumption for software
 - within the cluster
 - between clusters
 - from outside the cluster



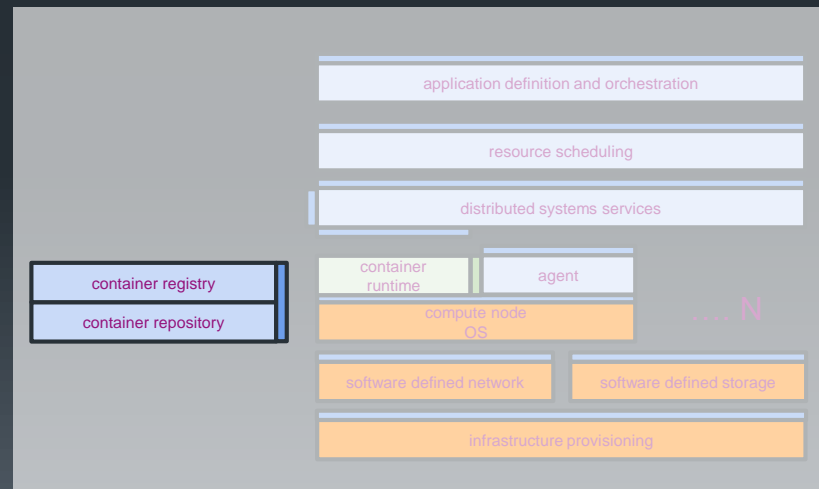
Local agent

- maintains the lifecycle of containers on a node
- monitors the health of container on a node



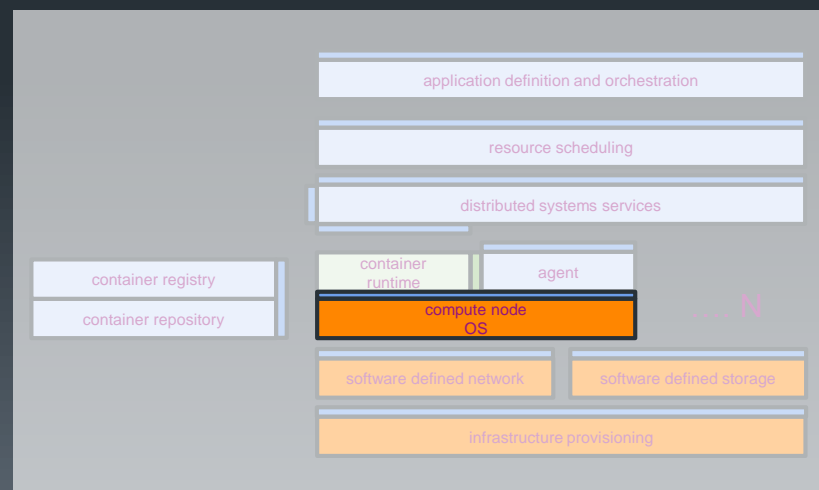
Container registry and repository

- a standard way to find a container
- a standard place to start and distribute containers



Compute node

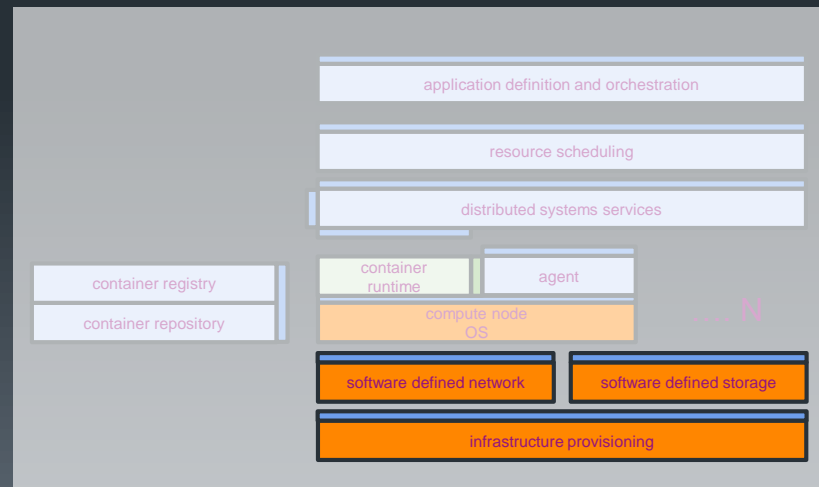
- a standard definition for what the minimum set of services on a compute node are
- the actual node distribution is out of scope



Infrastructure provisioning and integration

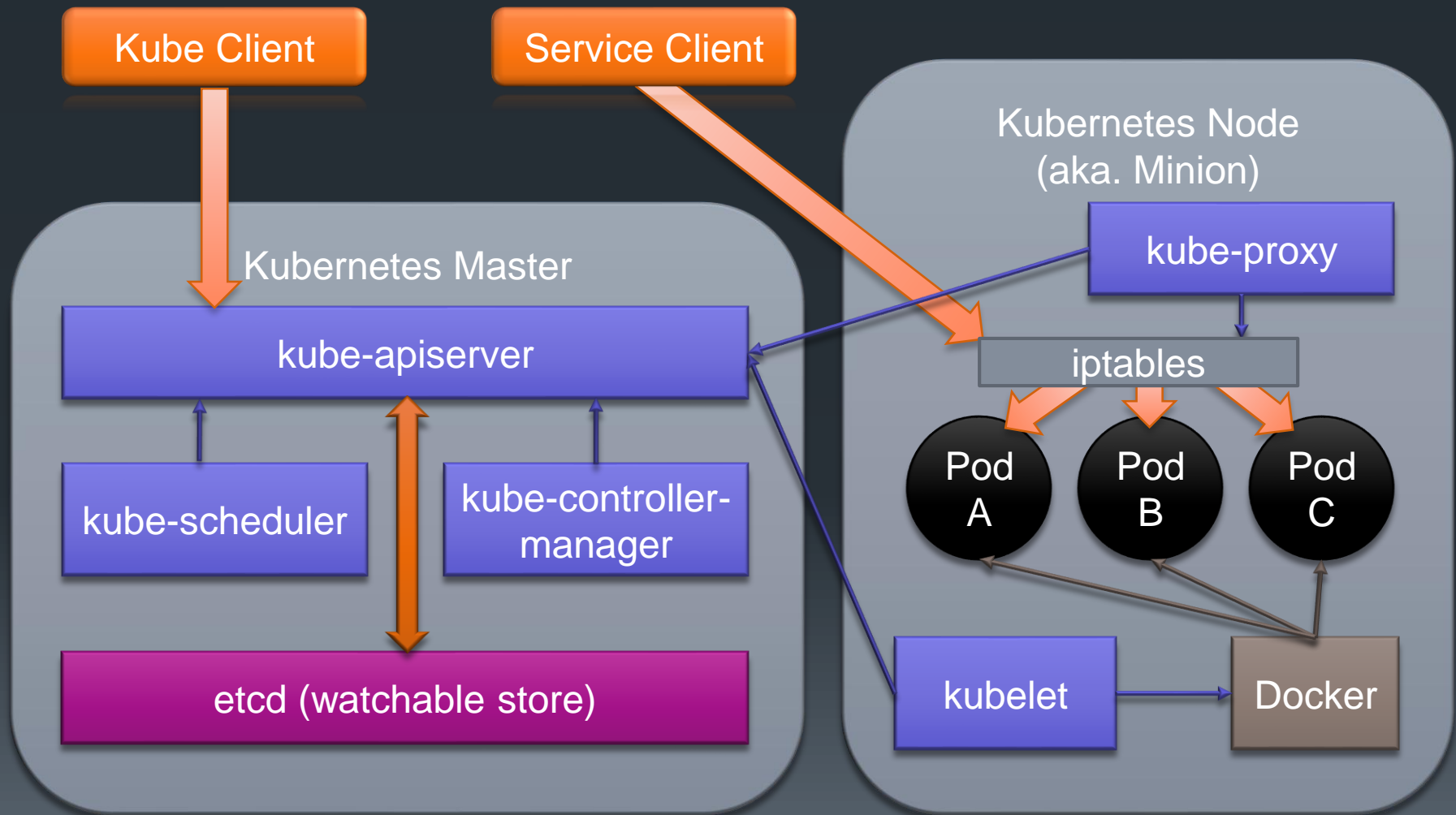
Copyright 2013-2018, RX-M LLC

- infrastructure provisioning
 - provide a standard interface and plugin model to request additional infrastructure
- software defined datacenter integration
 - provide a standard interface and plugin model for network and storage integration with clusters



Kubernetes Architecture

Copyright 2013-2018, RX-M LLC



Apache Mesos

Copyright 2013-2018, RX-M LLC

- Apache Mesos lets you program against your datacenter like it's a single pool of resources
- Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual)
- Enables fault-tolerant and elastic distributed systems to easily be built and run effectively
- Allows for the sharing a single cluster of commodity computer hardware between many different applications
 - Initially MPI and Hadoop
 - Later Spark, Cassandra, Jenkins and others
- Solves the reprovisioning/static partitioning problem with clusters
- Developed at the UC Berkeley AMP (Algorithms Machine People) Lab
 - Twitter hires Ben Hindman in 2010, the lead author and architect Mesos
 - Over several years, Mesos grows from a research project into the core infrastructure powering tens of thousands of servers at Twitter and many other companies
 - Produces Apache Aurora
- Ben Hindman and Florian Leibert from Twitter start Mesosphere in 2013
 - Producer of Marathon

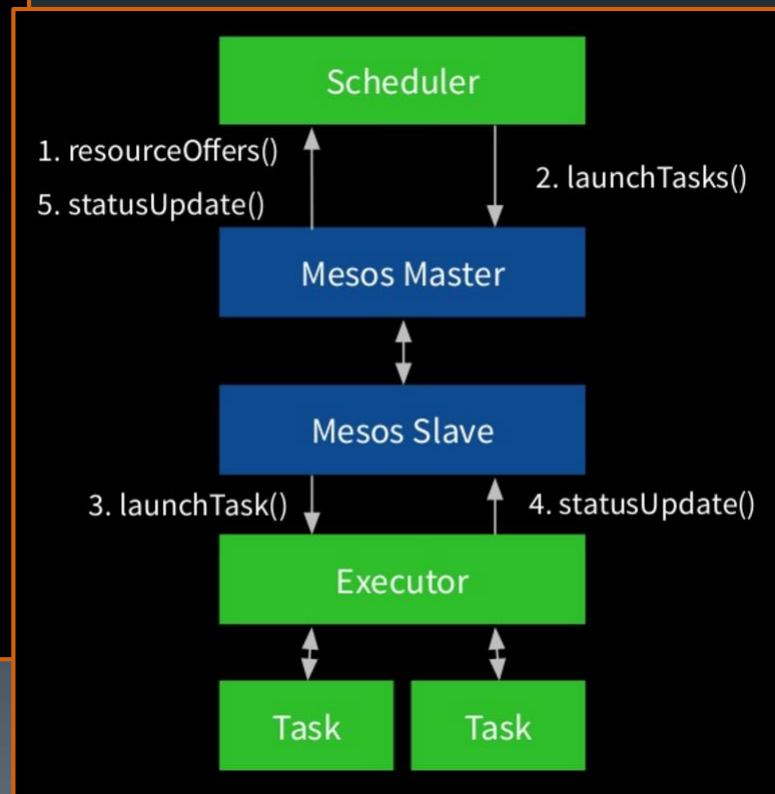
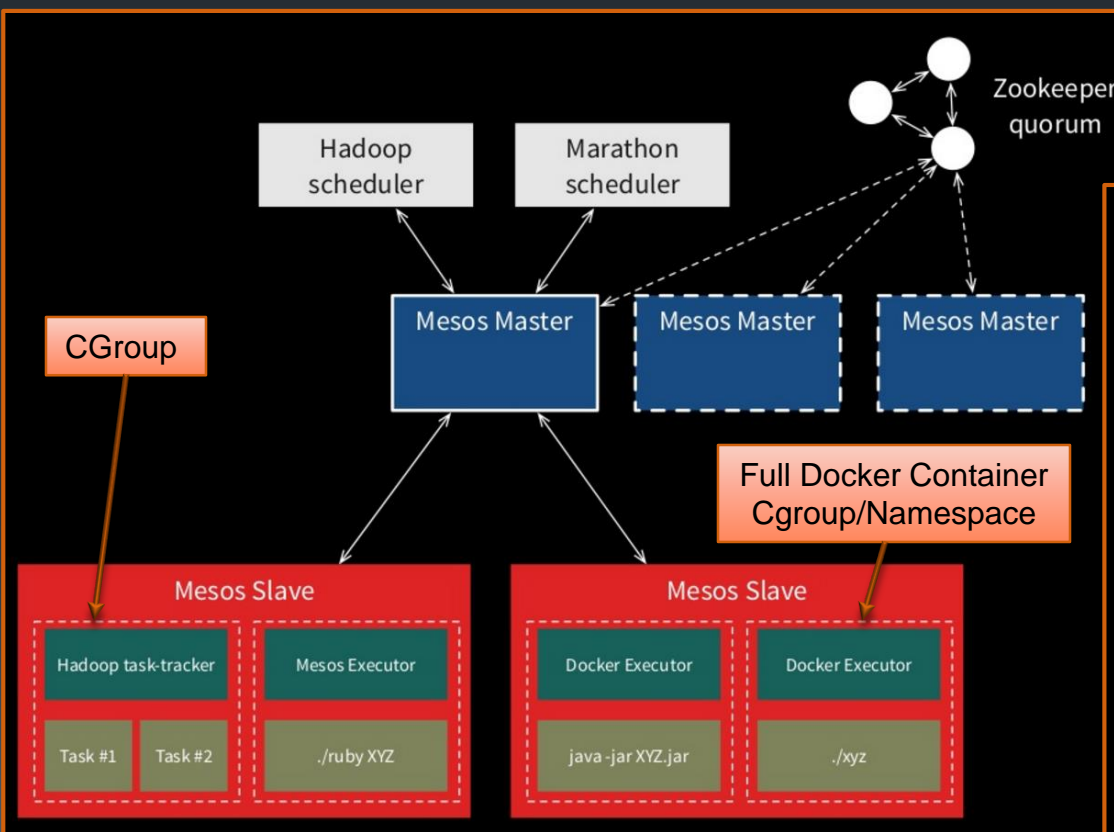


-- Florian Leibert

Mesos Architecture

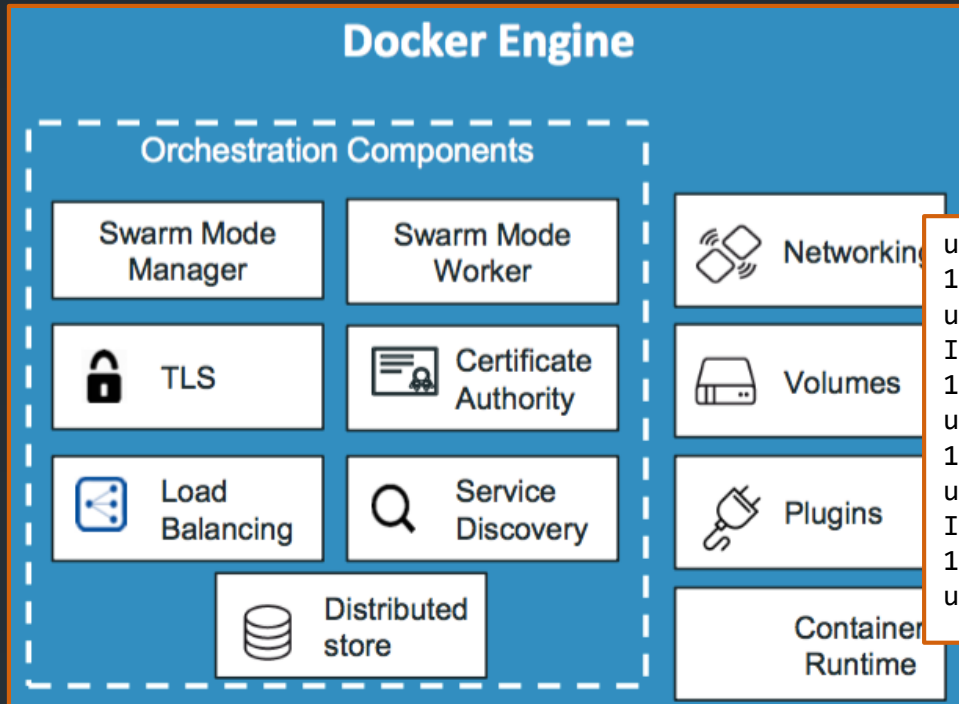
- A Mesos cluster has two components:
 - Masters - coordinates the cluster
 - Slaves - execute code in containers
- Mirrors the organization of frameworks
 - master/slave pattern is simple, flexible, and effective
 - The cluster software is however separate to ensure frameworks remain isolated and secure from one another

Copyright 2013-2018, RX-M LLC



Docker Swarm

Copyright 2013-2018, RX-M LLC



```
user@ubuntu:~$ docker service create nginx
141z2ky40mq8zxv7d7soynfxp
user@ubuntu:~$ docker service ls
ID                NAME                REPLICAS  IMAGE  COMMAND
141z2ky40mq8      awesome_babbage     1/1       nginx
user@ubuntu:~$ docker service scale 141z=2
141z scaled to 2
user@ubuntu:~$ docker service ls
ID                NAME                REPLICAS  IMAGE  COMMAND
141z2ky40mq8      awesome_babbage     2/2       nginx
user@ubuntu:~$
```

```
user@ubuntu:~$ docker swarm init --advertise-addr=10.0.0.220
Swarm initialized: current node (e7n1dpk4axdiyoo2mx4r2tb1r) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join \
--token SWMTKN-1-0wpinu43yhdm5zkwbtoagqvhgpcckuy14s2bmzeevksf0tx3fi-9k6q6w4osrj4s6000p50l8q1n \
10.0.0.220:2377
```

To add a manager to this swarm, run the following command:

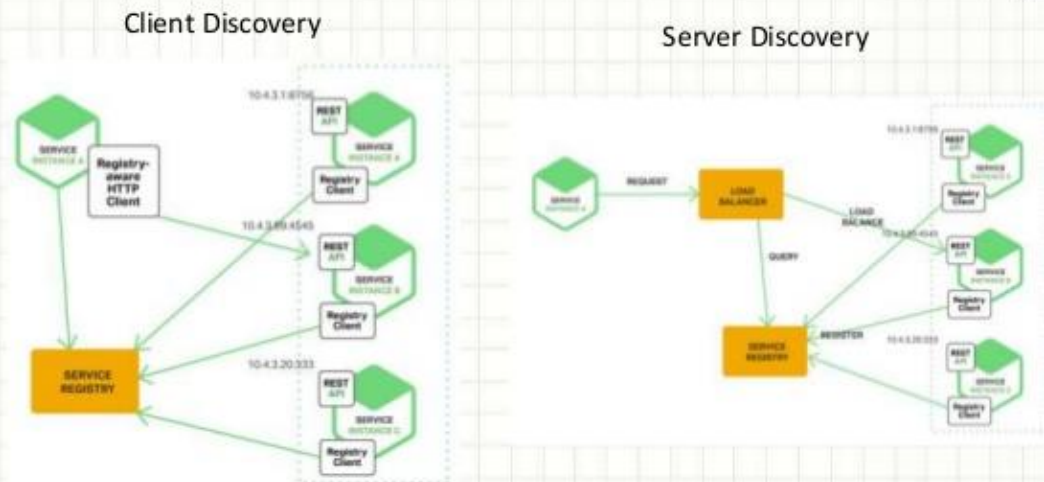
```
docker swarm join \
--token SWMTKN-1-0wpinu43yhdm5zkwbtoagqvhgpcckuy14s2bmzeevksf0tx3fi-0nf4ddcnixz0rbv4ebj125p3k \
10.0.0.220:2377
```


Service discovery schemes

Copyright 2013-2018, RX-M LLC

- Service discovery is the process by which a client locates a desired service
- Most common approach involves a client with a hardcoded or operationally supplied hostname using the resolver to acquire an IP address
 - /etc/hosts
 - DNS
- Resolver host records only supply an IP
 - SRV records can be used to acquire port numbers as well
- Dedicated services can also be accessed directly or wired into DNS/resolver systems
 - HashiCorp Consul
 - Apache Zookeeper
 - CoreOS Etcd
 - Netflix Eureka
 - Kubernetes SkyDNS
 - Docker Container Name/Alias/Link
 - Spring Cloud Config

Client vs Server side Service discovery

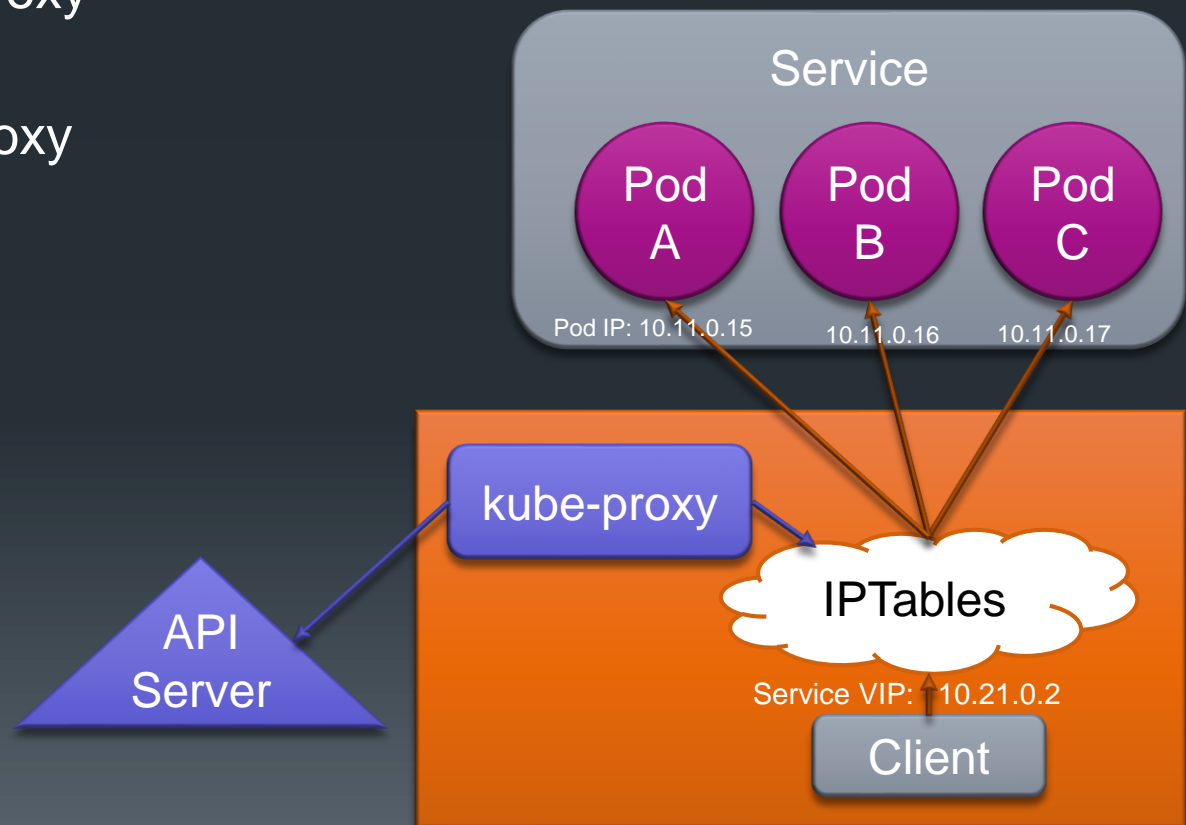


Pictures from <https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/>

Routing Meshes

Copyright 2013-2018, RX-M LLC

- Docker Swarm Mode – integrated with service functionality
 - Uses iptables
- Kubernetes – kube-proxy
 - Uses iptables
- Linkerd – software proxy
 - Uses Finagle dtabs
- Others...



Summary

- Orchestration in cloud native systems is the process of managing the deployment and ongoing operation of containerized microservices
- The CNCF cloud native platform reference model is working toward the creation of a common language which can be used to discussion the facets of a cloud native platform and the integrations points they expose
- There are several popular cloud native orchestration platforms available in the open source world
 - Mesos/Marathon
 - Mesos/Aurora
 - Kubernetes
 - Docker Swarm
- Services define and endpoint for a defined interface
- Load balancing schemes are used to distribute connections to service implementations
- Service discovery is the process used by service client to discover the service end point



Lab 7

- Kubernetes

8: FaaS

Objectives

- Understand the nature of functions as a service
- Describe the role of Gateways for functions
- Explain the means of triggering functions in the cloud
- Consider the various cloud function offerings and client side tools

Serverless

Copyright 2013-2018, RX-M LLC

- The Cloud has made possible fully “serverless” models of computing
 - Logic can be spun up on-demand in response to events originating from anywhere
 - Applications can be built from these bite-sized bits of business logic
 - Billing occurs only when code is running
 - Can server users counts from zero to planet-scale, all without managing any infrastructure
- “As we move forward, it’s becoming increasingly clear (to me at least) that the future will be containerized and those containers will run on serverless infrastructure.”
 - Brendan Burns, Microsoft Distinguished Engineer and Kubernetes founder



Serverless computing

Copyright 2013-2018, RX-M LLC

- **Serverless computing**

- A cloud service model in which the provider fully manages the invocation and hosting of a function
- Users provide the function code which must execute under one of the offered runtime environments
- Users are billed each time the function is invoked, rather than per VM/hour
- Also known as **Function as a Service (FaaS)**

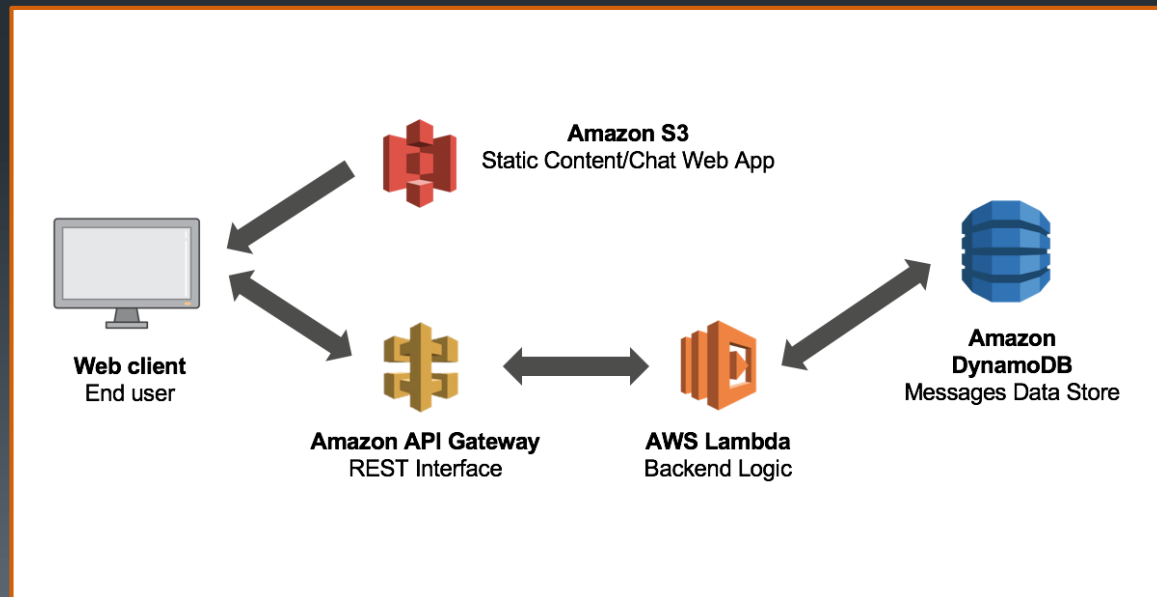
- Behind the scenes the cloud provider supplies a PaaS environment for the function to run on (which of course involves servers)

- "serverless computing" because user does not interact with, purchase, rent or provision servers

- Serverless code can be used in conjunction with normal services, such as microservices

- Serverless code is triggered in various ways

- By specific events (such as user registration with Amazon Cognito)
- By API calls to an API management platform exposing functions as REST API endpoints



History of FaaS

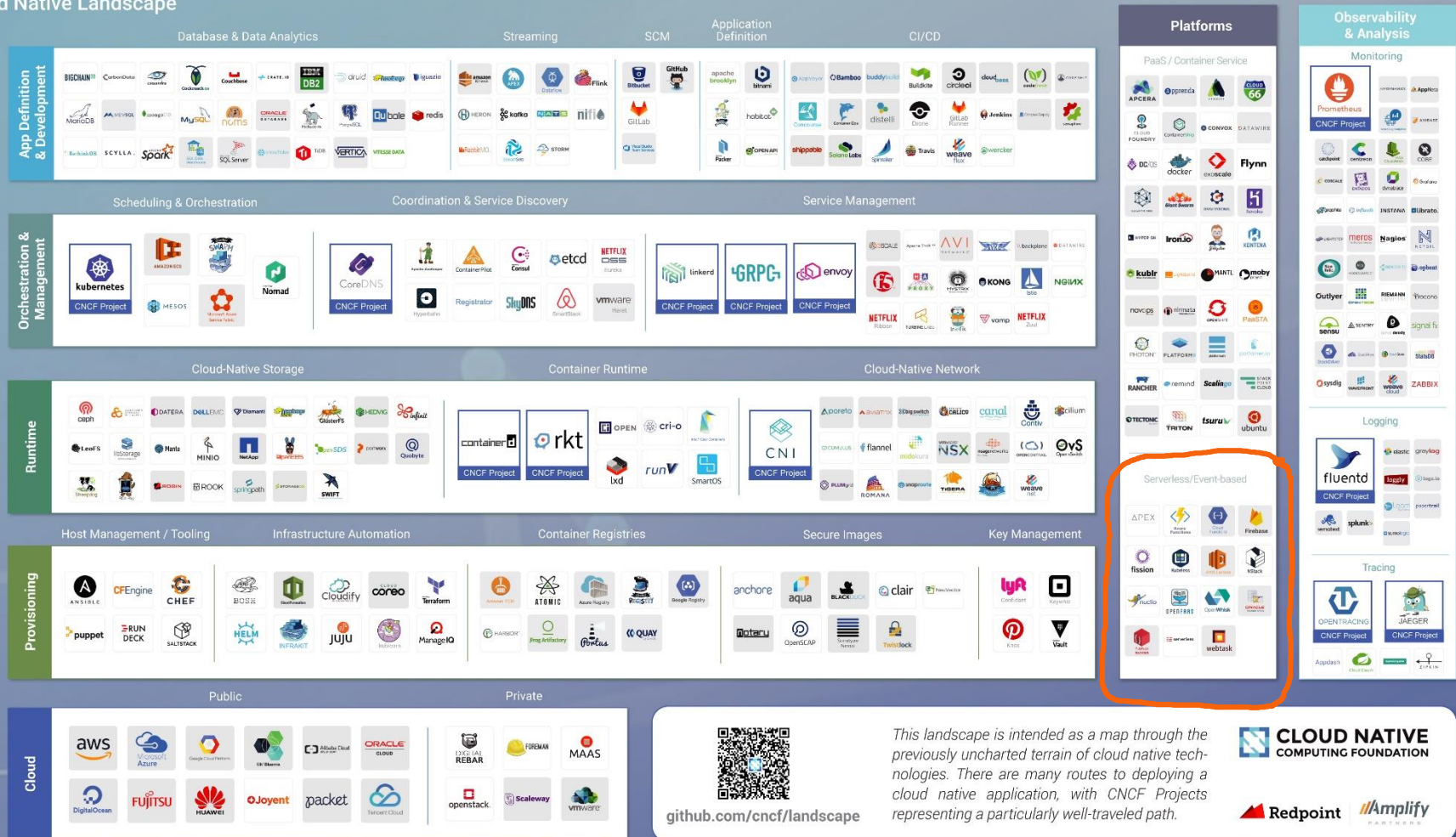
Copyright 2013-2018, RX-M LLC

- 2006 – First serverless offering provided by **Zimki**
- 2007 – First serverless offering shutdown by Zimki
 - CTO quits on stage at OSCON, ouch
- 2014 – **AWS Lambda** introduced, first major provider offering
 - Launched for Node.js later added Python, Java, C#
 - Others languages supported using Node.js as an invoker
- 2016 January – **Google Cloud Functions** alpha
 - Node.js support (now in beta)
- 2016 February – **IBM OpenWhisk** announced
 - Open source FaaS project for OpenStack
 - Now **Apache OpenWhisk**
 - Runs functions in Docker containers (any language)
 - Also PaaS support for Node and Swift (later added Python and Java)
- 2016 June – **Serverless Framework** appears on GitHub as the first application framework for FaaS
- 2016 November – **Microsoft Azure Functions**
 - Open source and usable in Azure and on any other cloud environment, public or private
- 2017 January – **OpenFaas** arrives on GitHub as the first **K8s FaaS framework**
- 2018 April – **Microsoft ACI** (Azure Container Instances) goes GA
- 2018 June – **AWS Fargate** for ECS and EKS goes GA

Cloud Native Landscape

Copyright 2013-2018, RX-M LLC

Cloud Native Landscape v0.9.8



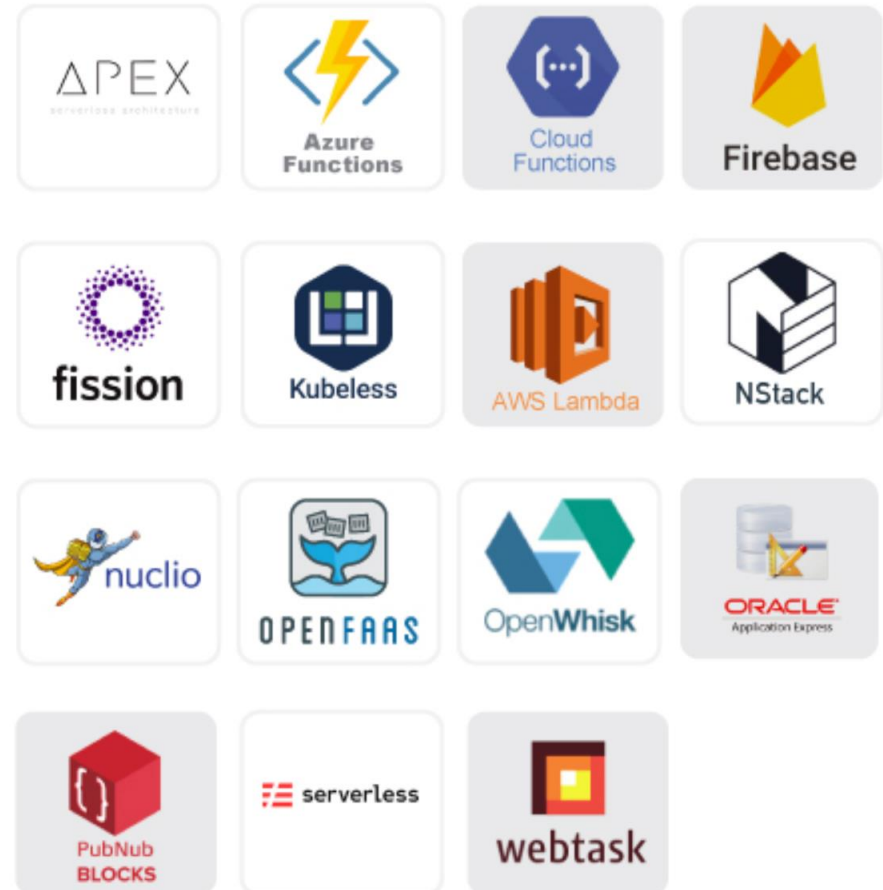
Greyed logos are not open source

Approaches to FaaS

Copyright 2013-2018, RX-M LLC

- **Public Cloud**
 - AWS Lambda
 - Azure Functions
 - Google Cloud Functions
 - IBM Cloud Functions
 - PubNub Blocks
- **Database centric**
 - Oracle APEX (Application Express)
 - API/Event integration with Oracle DBs
 - Firebase
 - API/Event integration with GCF
- **Frameworks/IDEs**
 - APEX
 - Dev front end for AWS Lambda
 - Serverless
 - Multicloud Framework for FaaS applications
 - Webtask
 - Cloud IDE for building task based applications
- **Open Source Platforms**
 - Fission [targets k8s]
 - Kubeless [targets k8s]
 - Nstack
 - Cross cloud serverless analytics platform
 - Nuclio [targets k8s]
 - Serverless scale-out event processing platform
 - OpenFaaS [targets k8s/swarm]
 - OpenWhisk

Serverless/Event-based



AWS Lambda

Copyright 2013-2018, RX-M LLC

- AWS Lambda lets you run code without provisioning or managing servers
- Pay for the compute time you consume
- Lambda can run code for virtually any type of application or backend service with zero administration
- Upload code and Lambda takes care of everything required to run and scale the code with high availability
- You can set up code to automatically trigger from other AWS services or call it directly from any web or mobile app
- AWS Lambda supports Java, Node.js, and Python
 - More expected



AWS Blueprints

Copyright 2013-2018, RX-M LLC

- AWS offers over 100 lambda blue prints to help you quickly wire functions to other AWS services

The screenshot shows the AWS Lambda console interface for creating a new function. The top navigation bar includes 'Services' and 'Resource Groups'. The breadcrumb trail indicates 'Lambda > New function'. On the left sidebar, there are links for 'Select blueprint' (highlighted), 'Configure triggers', 'Configure function', and 'Review'. The main content area is titled 'Select blueprint' and includes a help icon. Below the title, a paragraph explains that blueprints are sample configurations of event sources and Lambda functions, and that they are licensed under CC0. A filter bar at the top of the blueprint list includes a 'Select runtime' dropdown and a 'Filter' input field. The page shows 'Viewing 1-9 of 83' blueprints. The visible blueprints are arranged in a grid:

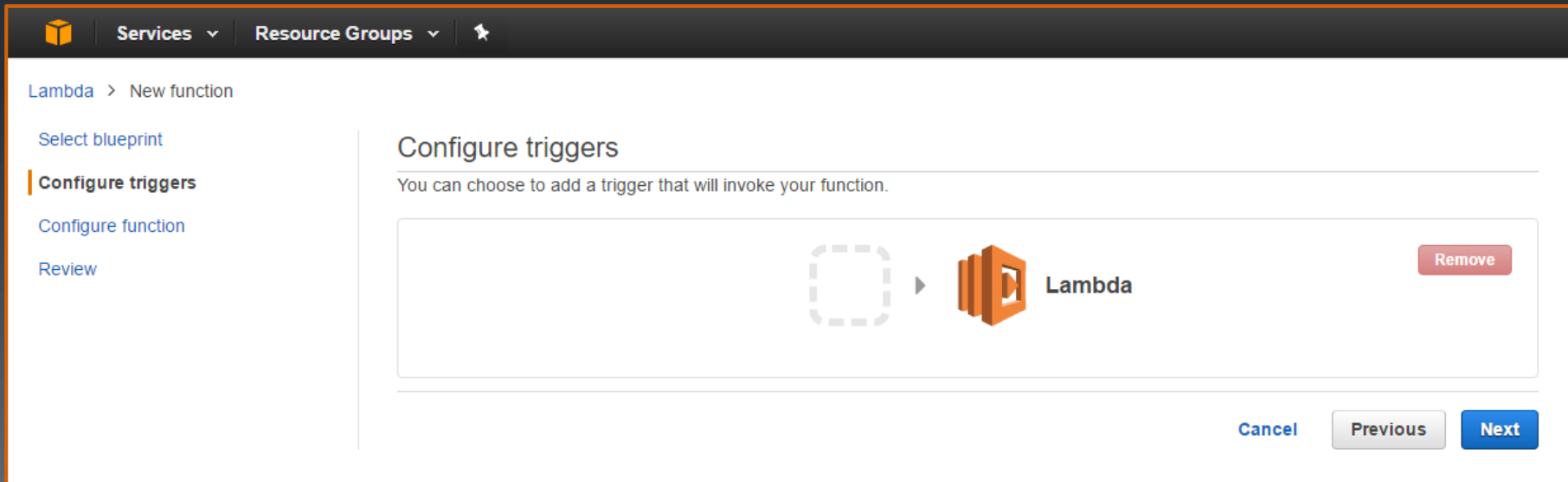
Blank Function	kinesis-firehose-syslog-to-json	alexa-skill-kit-sdk-factskill
Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard. custom	An Amazon Kinesis Firehose stream processor that converts input records from RFC3164 Syslog format to JSON. nodejs · kinesis-firehose	Demonstrate a basic fact skill built with the ASK NodeJS SDK nodejs · alexa
batch-get-job-python27	kinesis-firehose-apachelog-to-j...	cloudfront-modify-response-he...
Returns the current status of an AWS Batch Job. python2.7 · batch	An Amazon Kinesis Firehose stream processor that converts input records from Apache Common Log format to python2.7 · kinesis-firehose	Blueprint for modifying CloudFront response header implemented in NodeJS. nodejs · cloudfront · response hea...
s3-get-object-python	config-rule-change-triggered	lex-book-trip-python
An Amazon S3 trigger that retrieves metadata for the object that has been updated. python2.7 · s3	An AWS Config rule that is triggered by configuration changes to EC2 instances. Checks instance types. nodejs4.3 · config	Book details of a visit, using Amazon Lex to perform natural language understanding python2.7 · lex

A 'Cancel' button is located at the bottom right of the console window.

AWS Triggers

- Lambdas can be invoked by one or many events within AWS
 - Amazon S3
 - Amazon DynamoDB
 - Amazon Kinesis Streams
 - Amazon Simple Notification Service
 - Amazon Simple Email Service
 - Amazon Cognito
 - AWS CloudFormation
 - Amazon CloudWatch Logs & Events
 - AWS CodeCommit
 - Scheduled Events (powered by Amazon CloudWatch Events)
 - AWS Config
 - Amazon Echo
 - Amazon Lex
 - Amazon API Gateway
 - Invoked On Demand

Copyright 2013-2018, RX-M LLC



The screenshot shows the AWS Lambda console interface for configuring a new function. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a star icon. The breadcrumb trail indicates 'Lambda > New function'. On the left sidebar, there are links for 'Select blueprint', 'Configure triggers' (which is highlighted with an orange bar), 'Configure function', and 'Review'. The main content area is titled 'Configure triggers' and contains the instruction 'You can choose to add a trigger that will invoke your function.' Below this, there is a visual representation of a trigger: a dashed square icon followed by a right-pointing arrow, then the Lambda icon and the text 'Lambda'. A red 'Remove' button is located to the right of this trigger entry. At the bottom right of the console, there are three buttons: 'Cancel', 'Previous', and 'Next'.

Lambda Language Support

Copyright 2013-2018, RX-M LLC

- Lambda functions can be uploaded various ways:
 - raw code
 - Zip (if dependencies other than aws-sdk are required)
 - S3
- Lambdas can be coded in C#, Node.js, Java or Python
 - You can invoke binaries from Node.js to use other languages

The screenshot shows the AWS Lambda console interface for configuring a new function. The left sidebar contains navigation links: 'Select blueprint', 'Configure triggers', 'Configure function' (highlighted), and 'Review'. The main content area is titled 'Configure function' and includes a description: 'A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.'

The configuration fields are as follows:

- Name***: myFunctionName
- Description**: (empty)
- Runtime***: Node.js 6.10 (selected from a dropdown menu that also lists C#, Edge Node.js 4.3, Java 8, Node.js 4.3, and Python 2.7)
- Lambda function code**: Provide the code for your function. Use the editor to write code, or upload a zip file containing your code and libraries (other than the aws-sdk). If you need custom libraries, you can upload your code and libraries.
- Code entry type**: Raw code

The code editor shows the following code:

```
1 exports.handler = (event, context, callback) => {  
2   // TODO implement  
3   callback(null, 'Hello from Lambda');  
4 };
```


Accessing state from Lambda functions

Copyright 2013-2018, RX-M LLC

- Lambdas have built in access to the aws-sdk
- They can use Dynamo, S3, RDS and any other stateful AWS services
- Lambdas are configured with a IAM roles which control their platform permissions
- Lambdas are built to process events
 - Platform events (such as API gateway calls) can then be mapped to lambda events

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name* trashlevel

Description Mobile backend (read/write to DynamoDB)

Runtime* Node.js 4.3

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk). If you need custom libraries, you can upload your code and libraries as a .ZIP file. [Learn more](#) about deploying Lambda functions.

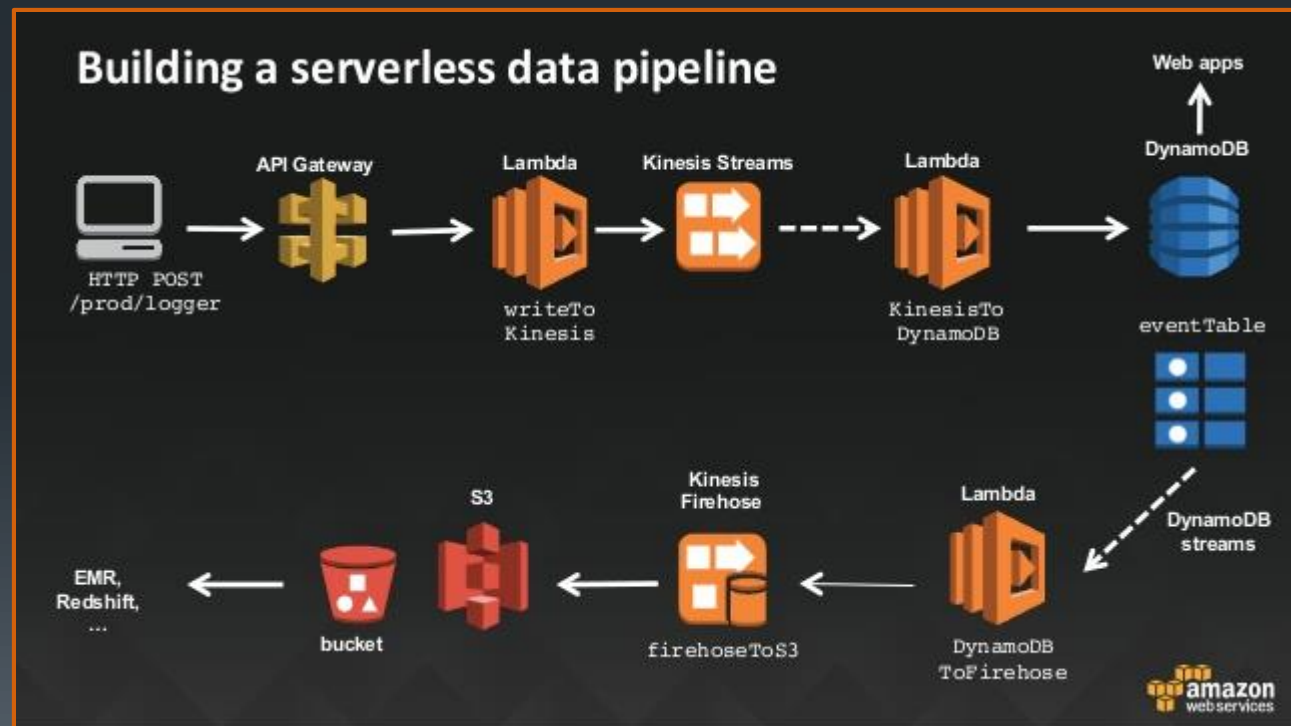
Code entry type Edit code inline

```
10 /**
11  * Provide an event that contains the following keys:
12  *
13  *   - operation: one of the operations in the switch statement below
14  *   - tableName: required for operations that interact with DynamoDB
15  *   - payload: a parameter to pass to the operation being performed
16  */
17 exports.handler = (event, context, callback) => {
18   //console.log('Received event:', JSON.stringify(event, null, 2));
19
20   const operation = event.operation;
21   const payload = event.payload;
22
23   if (event.tableName) {
24     payload.TableName = event.tableName;
25   }
26
27   switch (operation) {
28     case 'create':
29       dynamo.putItem(payload, callback);
30       break;
31     case 'read':
32       dynamo.getItem(payload, callback);
33       break;
34     case 'update':
35       dynamo.updateItem(payload, callback);
36       break;
37     case 'delete':
38       dynamo.deleteItem(payload, callback);
```


Serverless Big Data

Copyright 2013-2018, RX-M LLC

- Extensive pipelines can be configured through event chains invoking one or more lambdas in the progression of processing



Google Cloud Functions

- Google Cloud Functions is a lightweight compute solution for developers to create single-purpose, stand-alone functions that respond to Cloud events without the need to manage a server or runtime environment
- You can invoke Cloud Functions with an HTTP request using the POST, PUT, GET, DELETE, and OPTIONS HTTP methods
 - To create an HTTP endpoint for your function, you specify `--trigger-http` as the trigger type when deploying your function
 - From the caller's perspective, HTTP invocations are synchronous, meaning that the result of the function execution will be returned in the response to the HTTP request
- `gcloud beta functions deploy helloHttp --stage-bucket cloud-functions --trigger-http`
- `curl -X POST https://<YOUR_REGION>-<YOUR_PROJECT_ID>.cloudfunctions.net/helloHttp -H "Content-Type:application/json" --data '{"name":"Keyboard Cat"}'`

The screenshot shows the Google Cloud Platform console interface for creating a new Cloud Function. The top navigation bar includes the Google Cloud Platform logo and 'My First Project'. The main header shows 'Cloud Functions' and a 'Create function' button. The form fields are as follows:

- Name:** A text input field containing 'excessive-cancel-rate-alert'.
- Region:** A dropdown menu set to 'us-central1'.
- Memory allocated:** A dropdown menu set to '256 MB'.
- Timeout:** A text input field set to '60' with a unit of 'seconds'.
- Trigger:** Radio buttons for 'Cloud Pub/Sub topic' (selected), 'Cloud Storage bucket', and 'HTTP trigger'.
- Topic:** A dropdown menu set to 'order-cancel'.
- Source code:** Radio buttons for 'Inline editor' (selected), 'ZIP upload', 'ZIP from Cloud Storage', and 'Cloud Source repository'.

Below the form, there are tabs for 'index.js' and 'package.json'. The 'index.js' tab is active, showing the following code:

```
1 /**
2  * Triggered from a message on a Cloud Pub/Sub topic.
3  *
4  * @param {!Object} event The Cloud Functions event.
5  * @param {!Function} callback The callback function.
6  */
7 exports.subscribe = function subscribe(event, callback) {
8   // The Cloud Pub/Sub Message object.
9   const pubsubMessage = event.data;
10
11   // We're just going to log the message to prove that
12   // it worked.
13   console.log(Buffer.from(pubsubMessage.data, 'base64').toString());
14
15   // Don't forget to call the callback.
16   callback();
17 };
18
```

Economics of FaaS

Copyright 2013-2018, RX-M LLC

- Pay for use not while idle!
 - 1 GB-second is 1 second of wallclock time with 1GB of memory provisioned
 - 1 GHz-second is 1 second of wallclock time with a 1GHz CPU provisioned

CLOUD FUNCTIONS PRICING

Google Cloud Functions charges for invocations, compute time, and outbound data. Inbound data, and outbound data to other Google APIs in the same region is free. For detailed pricing information, please view the [pricing guide](#).

	FREE LIMIT PER MONTH	PRICE ABOVE FREE LIMIT (PER UNIT)	PRICE UNIT
Invocations *	2 million invocations	\$0.40	per million invocations
Compute Time	400,000 GB-seconds	\$0.0000025	per GB-Second
	200,000 GHz seconds	\$0.0000100	per GHz-Second
Outbound Data (Egress)	5GB	\$0.12	per GB
Inbound Data (Ingress)	Unlimited	Free	per GB
Outbound Data to Google APIs in same region	Unlimited	Free	per GB

Azure Functions

Copyright 2013-2018, RX-M LLC

- Triggers:
 - Timers (cron)
 - Platform Events (object upload, new message in queue, new vm created, etc.)
 - HTTP End points
- Supported Languages:
 - JavaScript – NPM support
 - C# - NuGet support
 - F#
 - Python
 - PHP
 - Bash
 - Batch
 - PowerShell
 - Any binary executable
- Integrates with:
 - Visual Studio Team Services
 - GitHub
 - BitBucket

Open sourced here:

<https://github.com/azure/azure-webjobs-sdk-script>

As an example, here's a simple Node.js function that receives a queue message and writes that message to Azure Blob storage:

```
module.exports = function (context, workItem) {
    context.log('Node.js queue trigger function processed work item ', workItem.id);
    context.bindings.receipt = workItem;
    context.done();
}
```

And here's the corresponding `function.json` file which includes a trigger **input binding** that instructs the runtime to invoke this function whenever a new queue message is added to the `samples-workitems` queue:

```
{
  "bindings": [
    {
      "type": "queueTrigger",
      "direction": "in",
      "queueName": "samples-workitems"
    },
    {
      "type": "blob",
      "name": "receipt",
      "direction": "out",
      "path": "samples-workitems/{id}"
    }
  ]
}
```

And here's a Python function for the same function definition doing the same thing:

```
import os

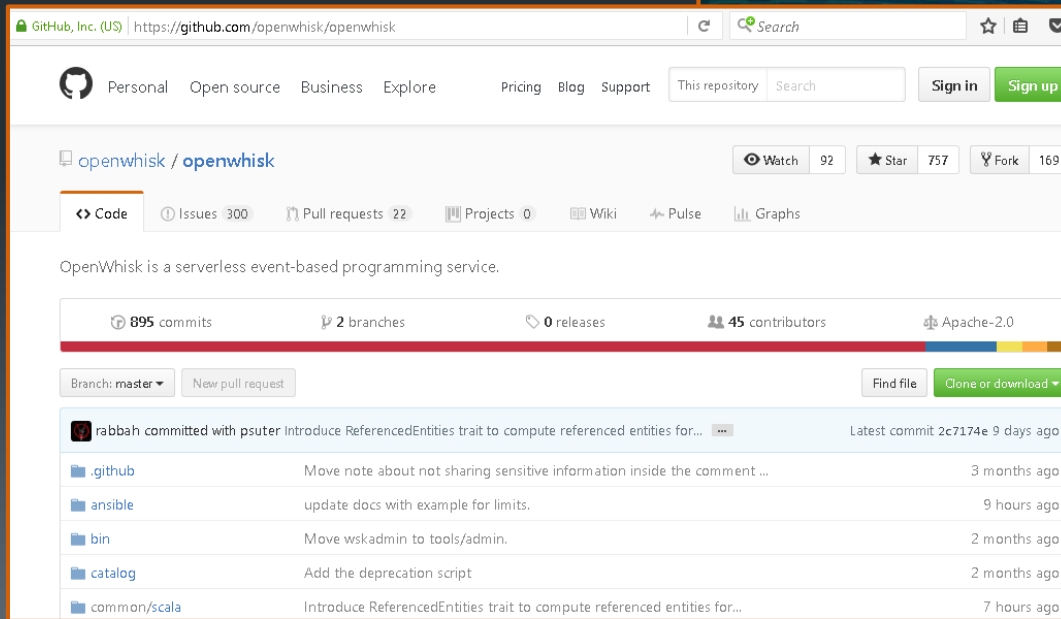
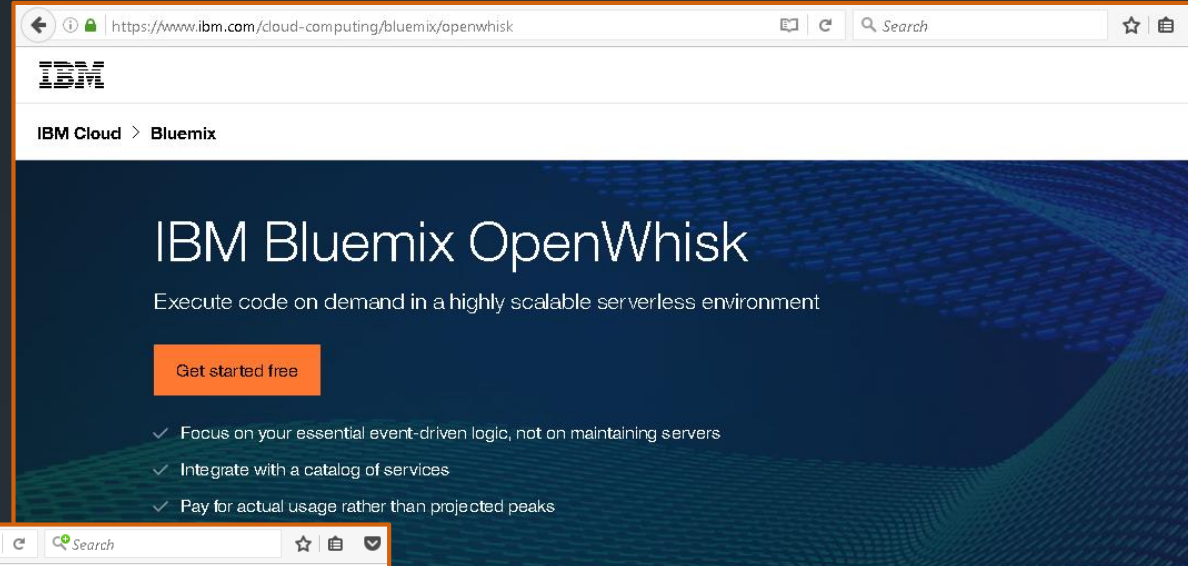
# read the queue message and write to stdout
workItem = open(os.environ['input']).read()
message = "Python script processed work item '{0}'".format(workItem)
print(message)

# write to the output binding
f = open(os.environ['receipt'], 'w')
f.write(workItem)
```

IBM OpenWhisk

Copyright 2013-2018, RX-M LLC

- An open source lambda platform
 - Now Apache OpenWhisk
- Runs on BlueMix, OpenStack, AWS, Vagrant etc.
- Can execute containers
 - No Language restrictions



event-
scale

The OpenWhisk serverless architecture accelerates development as a set of small, distinct, and independent actions. By abstracting away infrastructure, OpenWhisk frees members of small teams to rapidly work on different pieces of code simultaneously, keeping the overall focus on creating user experiences customers want.

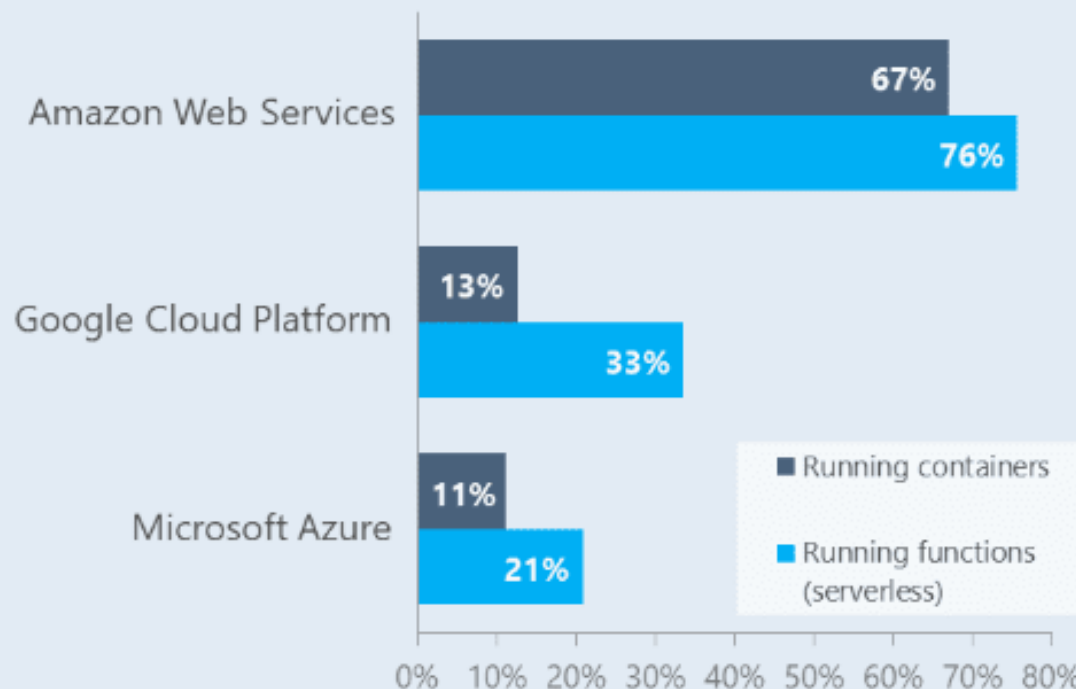
Connect actions into flexible, scalable sequences

OpenWhisk is serverless, using business rules to bind events, triggers, and actions to each other. OpenWhisk actions run automatically only when needed. Its serverless architecture promotes quickly, scalably creating and modifying action sequences to meet the evolving demands of mobile-driven user

Cloud Container/FaaS Breakdown

Copyright 2013-2018, RX-M LLC

Containers and Serverless Deployment With Top Three Public Cloud Providers



Source: The New Stack Analysis of Cloud Native Computing Foundation survey conducted in Fall 2017. Q. Your company/organization deploys containers to which of the following environments? (check all that apply). n=761. Q. Is your organization using serverless technology? n=755, of which 221 (29%) said "yes". Q. If yes, which serverless platform does your organization use? (check all that apply). n=221.

Open Source Serverless Platforms

- Open-source serverless Cons:
 - Not as integrated as cloud provider serverless
- Open-source serverless Pros:
 - Broader event/integration options
 - More parameter choices
 - Local debugging support
 - Generally faster
 - Ability to avoid cold starts

Serverless Framework

Copyright 2013-2018, RX-M LLC

- The Serverless Framework helps you develop and deploy your functions, along with the infrastructure resources they require
- A CLI that offers structure, automation and best practices
- The Serverless Framework is different than other application frameworks because it manages your code as well as your infrastructure, configuring and wiring events to functions

<https://serverless.com>

SERVERLESS
FRAMEWORK
VERSION 1.0

Build auto-scaling, pay-per-execution,
event-driven apps on AWS Lambda

▶ WATCH THE VIDEO

📖 READ THE DOCS

```
# Install serverless globally
$ npm install serverless -g

# Create an AWS Lambda function in Node.js
$ serverless create --template aws-nodejs

# Deploy to live AWS account
$ serverless deploy

# Function deployed!
$ http://api.amazon.com/users/update
```

-> Read the [docs](#) or connect with the [community](#)

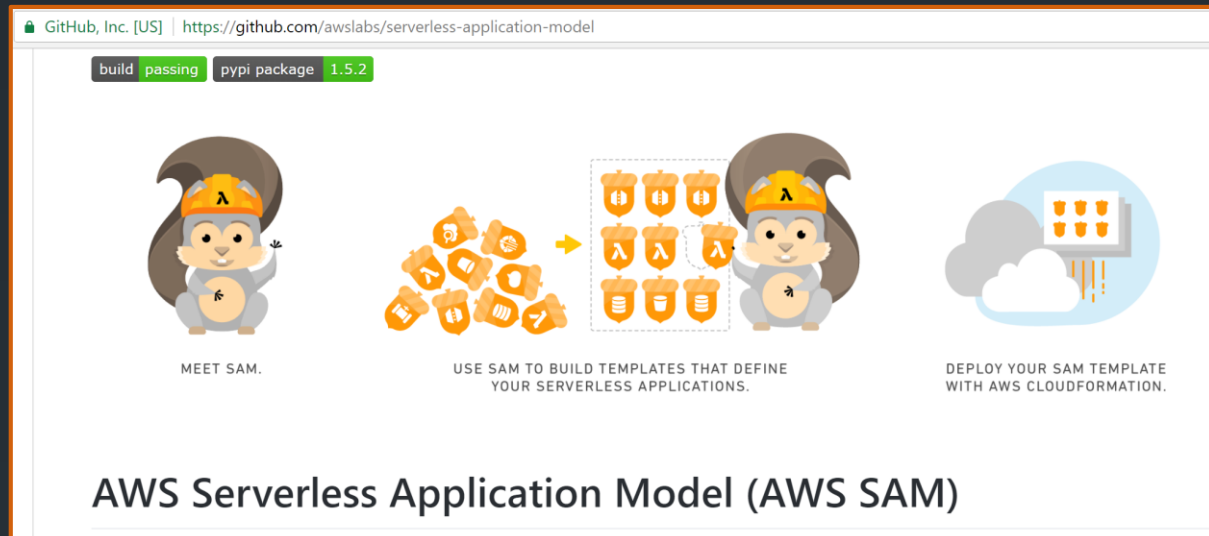
Powered by



AWS Serverless Application Model

Copyright 2013-2018, RX-M LLC

- AWS SAM
- Prescribes rules for expressing Serverless applications on AWS
- Apache 2.0 Open Source
 - <https://github.com/awslabs/serverless-application-model>



The banner for the AWS SAM GitHub repository. At the top, it shows the repository name 'aws/serverless-application-model' with a 'build passing' status and 'pypi package 1.5.2'. Below this is a cartoon squirrel character wearing a hard hat with a lambda symbol, labeled 'MEET SAM.'. To the right is a diagram showing a pile of serverless icons (Lambda, API Gateway, S3, etc.) being transformed into a structured template, labeled 'USE SAM TO BUILD TEMPLATES THAT DEFINE YOUR SERVERLESS APPLICATIONS.'. Further right is another squirrel character, labeled 'DEPLOY YOUR SAM TEMPLATE WITH AWS CLOUDFORMATION.', with a cloud icon showing a template being deployed.

AWS Serverless Application Model (AWS SAM)

You can use SAM to define serverless applications in simple and clean syntax.

This GitHub project is the starting point for AWS SAM. It contains the SAM specification, the code that translates SAM templates into AWS CloudFormation stacks, general information about the model, and examples of common applications.

The SAM specification and implementation are open sourced under the Apache 2.0 license. The current version of the SAM specification is available at [AWS SAM 2016-10-31](https://github.com/awslabs/serverless-application-model).

Creating a serverless application using SAM

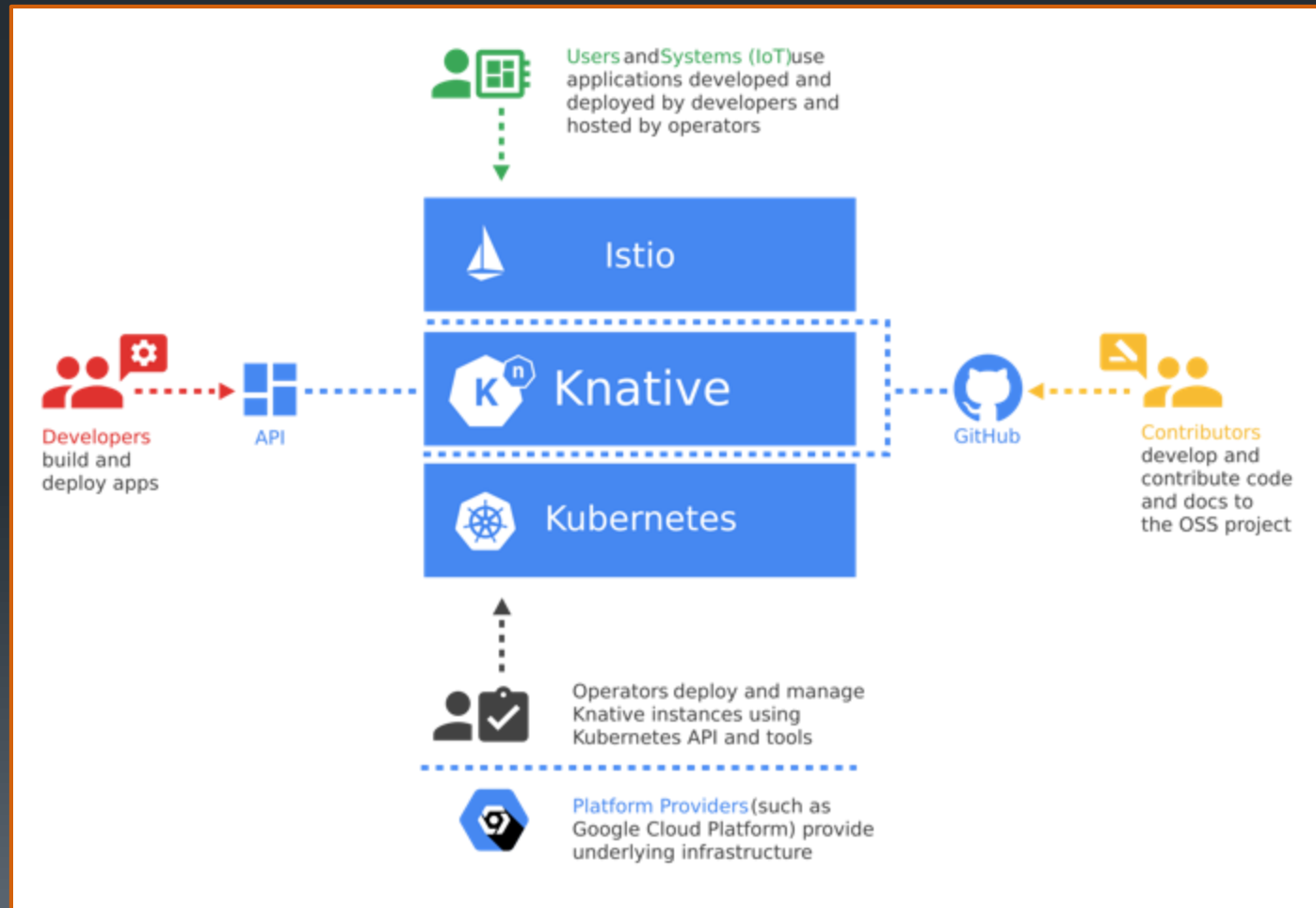
To create a serverless application using SAM, first, you create a SAM template: a JSON or YAML configuration file that describes your Lambda functions, API endpoints and the other resources in your application. Then, you test, upload, and deploy your application using the [SAM Local CLI](#). During deployment, SAM automatically translates your application's specification into CloudFormation syntax, filling in default values for any unspecified properties and determining the

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Transform: 'AWS::Serverless-2016-10-31'
3 Description: A starter AWS Lambda function.
4 Resources:
5   helloworld:
6     Type: 'AWS::Serverless::Function'
7     Properties:
8       Handler: index.handler
9       Runtime: nodejs6.10
10      CodeUri: .
11      Description: A starter AWS Lambda function.
12      MemorySize: 128
13      Timeout: 3
```

Knative

Copyright 2013-2018, RX-M LLC

- Knative is an open source FaaS project started by Google, Pivotal, and other industry leaders
- Extends Kubernetes and Istio to support functions as a service on any cloud



FaaS Use Cases

Copyright 2013-2018, RX-M LLC

- **Mobile Backend**

- Gateways can map REST routes to FaaS routines which can then use other cloud services such as Analytics, Database, Authentication, and Storage

- **APIs & Microservices**

- Functions can be API event-driven or invoked directly from other microservices

- **Data Processing / ETL**

- Storage events can trigger FaaS methods, such as when a file is created, changed, or removed, causing image processing, transcoding, validation or transformation

- **Webhooks**

- HTTP triggers can cause FaaS methods to respond to events originating from 3rd party systems like GitHub, Slack, Stripe, or from anywhere that can send HTTP/S requests

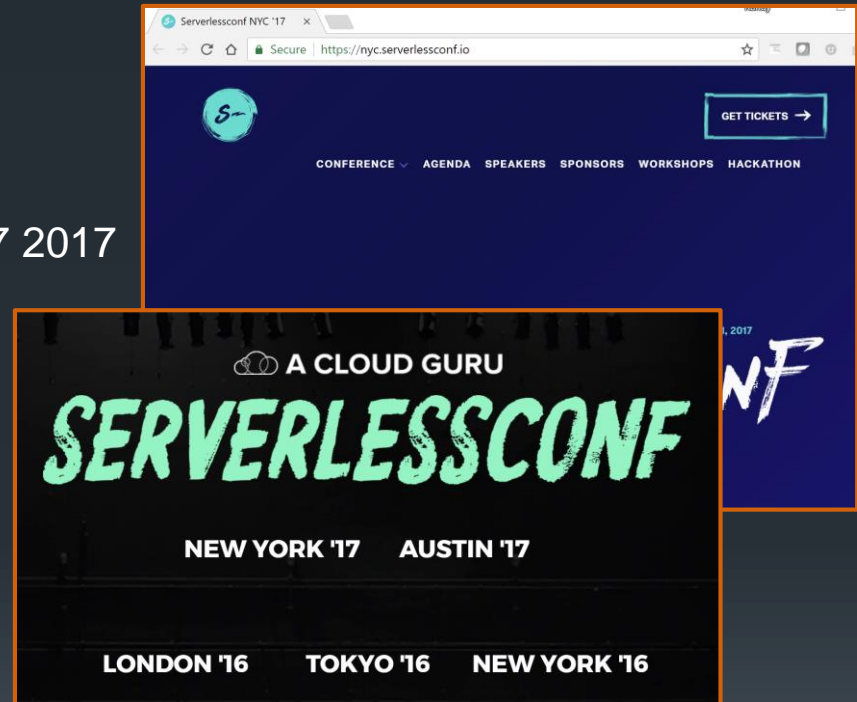
- **IoT**

- Hundreds of thousands of devices streaming data into Cloud Pub/Sub systems can automatically launching FaaS methods to process, transform and store data

Serverless Conferences

Copyright 2013-2018, RX-M LLC

- Serverless conferences are popping up everywhere
 - Serverlessconf.io
 - Hells Kitchen New York, Oct 2017
 - QCon: Learning from Expert Peers
 - New York June 26-30, Shanghai October 19-21 and San Francisco November 13-17 2017
 - All with Serverless tracks
 - AWS Re:Invent
 - Lots of coverage of AWS Lambda and serverless
 - Las Vegas Nov 2017
- Meetups
 - 127 meetups
 - > 30,000 members
 - <https://www.meetup.com/topics/serverless-architecture/>



CNCF Serverless WG

Copyright 2013-2018, RX-M LLC

- Exploring the intersection of cloud native and serverless
- Coms:
 - Google Group: [cncf-wg-serverless](https://groups.google.com/forum/#!forum/cncf-wg-serverless)
 - Slack #serverless channel: <https://slack.cncf.io/>
 - GitHub: <https://github.com/cncf/wg-serverless>
- Goals
 - Provide clarity for ourselves and consumers w.r.t. what this topic is all about
 - Define common terminology
 - Define the scope of the space as it exists today - over time this may change
 - Identify common use cases and patterns between existing implementations
 - Identify where serverless fits in relative to PaaS and container orchestration
 - Identify potential next steps for the community (may or may not happen within CNCF)
 - Identifying areas where we'd like to see some harmonization or interop work
 - Create a whitepaper on Serverless in relation to Cloud Native

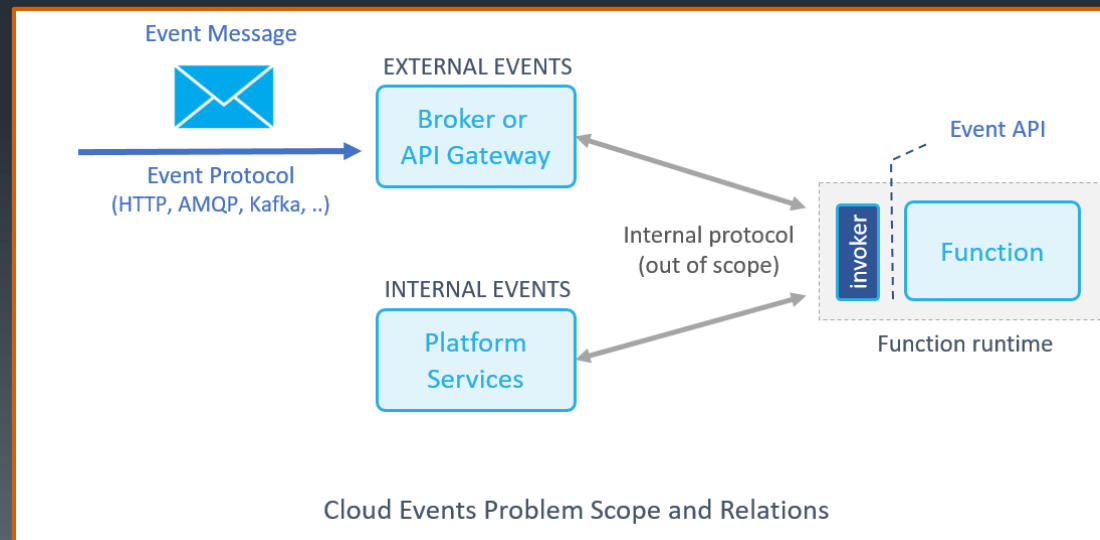
The screenshot shows the GitHub repository page for `cncf / wg-serverless`. At the top, there are buttons for Watch (47), Star (52), and Fork (8). Below this, the repository is described as "CNCF Serverless WG" with a link to <https://cncf.io>. It has 38 commits, 1 branch, 0 releases, 10 contributors, and is licensed under Apache-2.0. The main content area shows a list of commits. The latest commit by yaronha, titled "Add clarifications and few edits (#16)", was made 7 days ago. Other commits include "Delete older nuclio presentation (#13)" (10 days ago), "Initial commit" (6 months ago), and "Add link to the CNCF Slack (#15)" (9 days ago). The repository also has a README.md file and a LICENSE file.

Commit	Author	Message	Time Ago
presentations	yaronha	Delete older nuclio presentation (#13)	10 days ago
proposals	yaronha	Add clarifications and few edits (#16)	7 days ago
LICENSE	yaronha	Initial commit	6 months ago
README.md	yaronha	Add link to the CNCF Slack (#15)	9 days ago

Serverless Model

Copyright 2013-2018, RX-M LLC

- Serverless communication between services and functions is done through events/messages
- Describing and publishing events is a key to standardization
 - The event message and protocol
 - This enables a common way to consume events (an Event API)
- A common definition will allow function portability across platforms and interactions between services on different platforms



OpenEvents

- Proposals to the CNCF WG for a common event format and API
 - OpenEvents
 - Cloud Auditing Data Federation (CADF) event format as standardized at the Digital Mgmt. Task Force (DMTF). See <https://www.dmtf.org/standards/cadf>
- A range of event processing models should be considered
 - Most cloud vendors have their own proprietary implementations of eventing and event distribution

openevents

Introduction

OpenEvents is a specification for a common, vendor-neutral format for event data.

All cloud providers (e.g. Microsoft, IBM, Google, Amazon) have their own event payload format. Lack of a consistent format means that developers must constantly re-learn how to receive events within a single provider, or across providers. This specification aims to drive convergence on a common set of attributes (e.g., names and types) for event data.

Our end goal is to offer this specification to the [Cloud Native Computing Foundation](#).

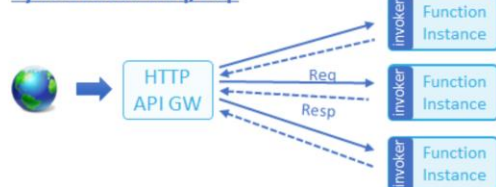
Version

Version 0.3 - 2017/09/20

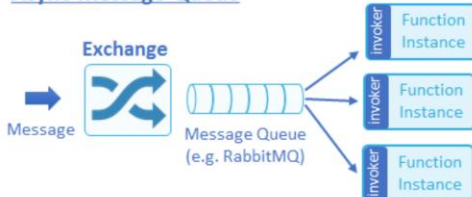
Schema

- eventId** - *string* - ID of the event. Can be specified by the producer. The semantics of this string are explicitly undefined to ease the implementation of producers (e.g. a database commit ID).
- eventType** - *string* - Type of the event (e.g. `customer.created`). Producers can specify the format of this, depending on their service. This specification does not (yet) enforce a type format (up for discussion). We are also discussing putting the type version in here as well.

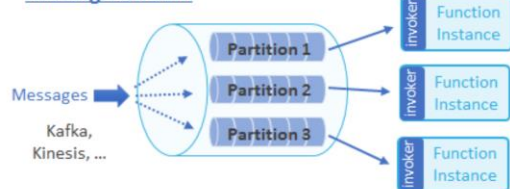
Synchronous Req/Rep



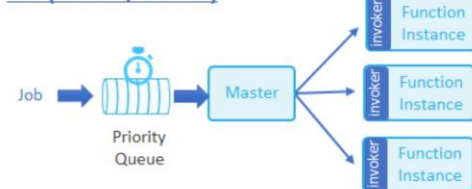
Async Message Queue



Message Stream



Job (Master/Worker)



timezone Z - Timestamp of event creation generated by the producer.

event payload. Payload depends on the event type. Producer can specify format/encoding of event.

content type of the data (e.g. `application/json`).

the resource that published the event. This object has the following fields:

the event source. Providers define list of event sources.

event source.

version of the OpenEvents specification (e.g. `1.0`).

metadata - This is for additional metadata and this does not have a required structure. The goals

Problems with Serverless

Copyright 2013-2018, RX-M LLC

- Attempting to observe Function-as-a-Service (FaaS) serverless applications can present challenges
 - Nowhere to install monitoring agents
 - No opportunity for background processing
 - Telemetry data this has to be sent during a function invocation (when the user is still waiting on a potentially business critical response)
- AWS Case
 - Deep integration between AWS Lambda and Kinesis has made event-driven architectures easier to implement within AWS
 - Tracing function invocations through asynchronous event sources like AWS Kinesis is not currently supported by existing tools like Amazon X-Ray
- The space is developing quickly but many critical production tools remain to be created/enabled

Summary

- All of the major public cloud providers now offer FaaS
 - Amazon AWS Lambda
 - Google Cloud Platform – Cloud Functions
 - Microsoft Azure Functions
 - IBM OpenWhisk
 - Now Apache OpenWhisk

Lab 8

- FaaS with OpenWhisk

9: API Gateways

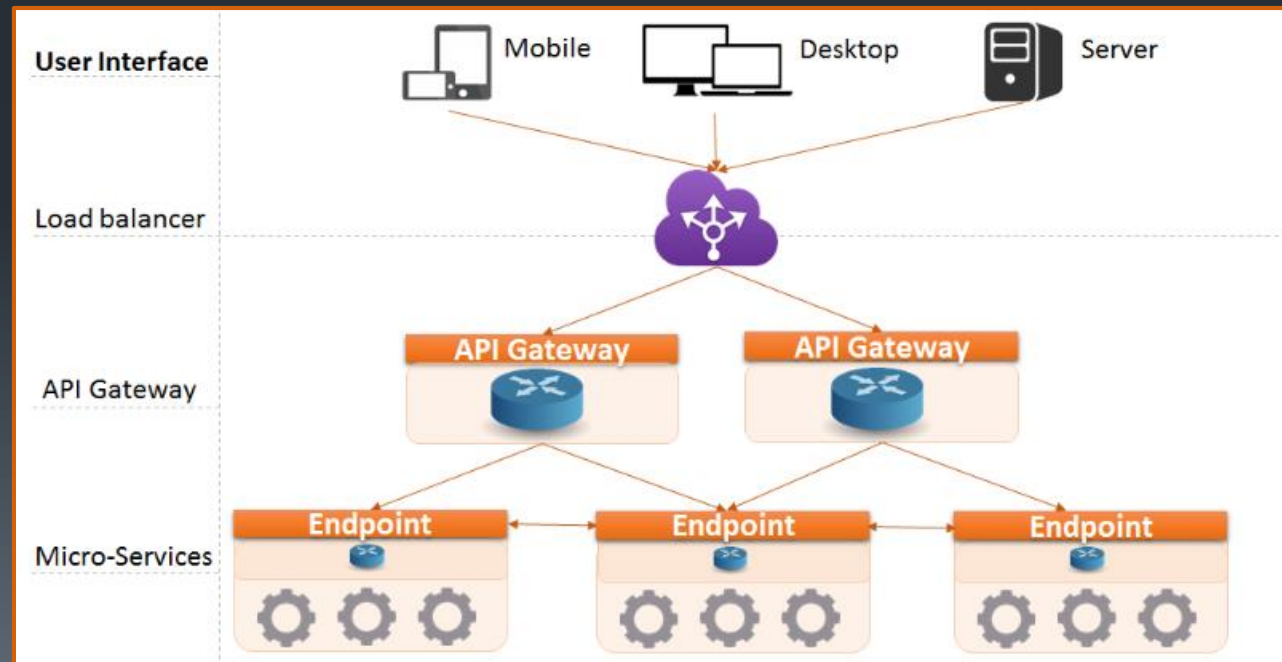
Objectives

- Understand the role of an API gateway
- Explain the importance of bounded context and how API Gateways can enforce isolation
- List key benefits and responsibilities of API Gateways

The role of gateways

Copyright 2013-2018, RX-M LLC

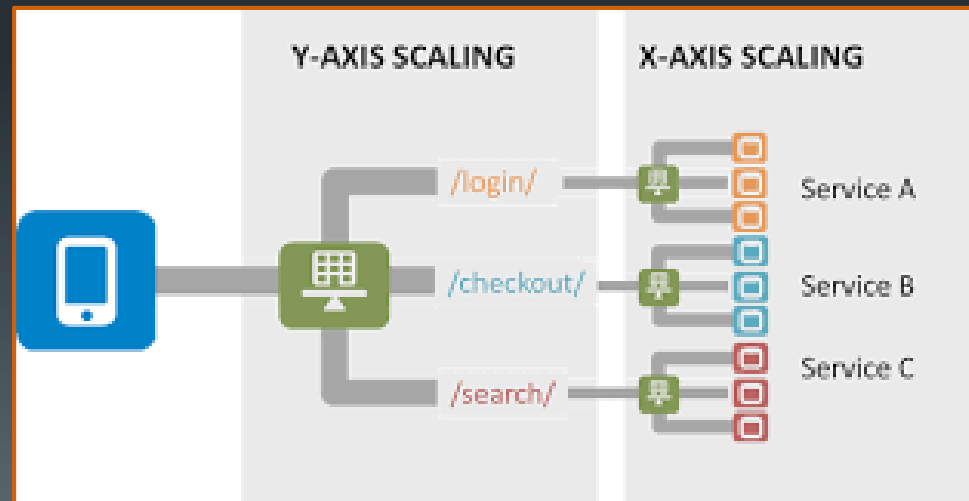
- Microservice based applications must define a way for clients to interact with the application
- With a monolithic application there is just one set of (typically replicated, load-balanced) endpoints
- In a microservices architecture each microservice exposes a fine-grained endpoint
- API Gateways expose a perimeter API isolating the microservices within the bounded context of the application
- Clients use the API Gateway perimeter API to consume services



Service constructs and end points

Copyright 2013-2018, RX-M LLC

- Services are conceptual in cloud native systems
 - Containers implement services
 - Orchestration systems start and stop additional containers for a service as load dictates (autoscaling)
 - Load balancing mechanisms distribute clients across the containers available
- Endpoints define the connection target for clients
 - End points are manifested by real or virtual Load Balancers
 - VIP
 - Per Node Ports
 - Netflix Ribbon
 - AWS Elastic/Application Load Balancer
 - Google Cloud Load Balancer
 - Nginx
 - HA Proxy
 - Traefik
 - Etc.



Auto Scaling

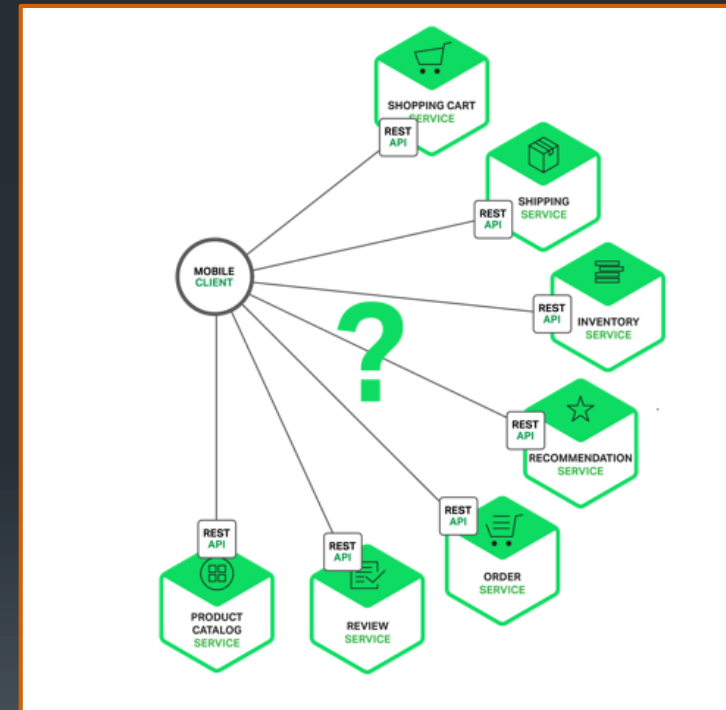
Copyright 2013-2018, RX-M LLC

- **Cluster scaling** – infrastructure scaling
 - Adding/removing worker nodes based on cluster utilization
- **Application scaling** – pod/container scaling
 - Based on CPU utilization, HTTP requests per second, etc.
 - **Horizontal Pod Autoscalers (HPAs)** – adding/removing pod replicas depending on certain metrics
 - **Vertical Pod Autoscalers (VPAs)** – increases/decrease resource requirements of containers/pods

Ingres integration anti-patterns

Copyright 2013-2018, RX-M LLC

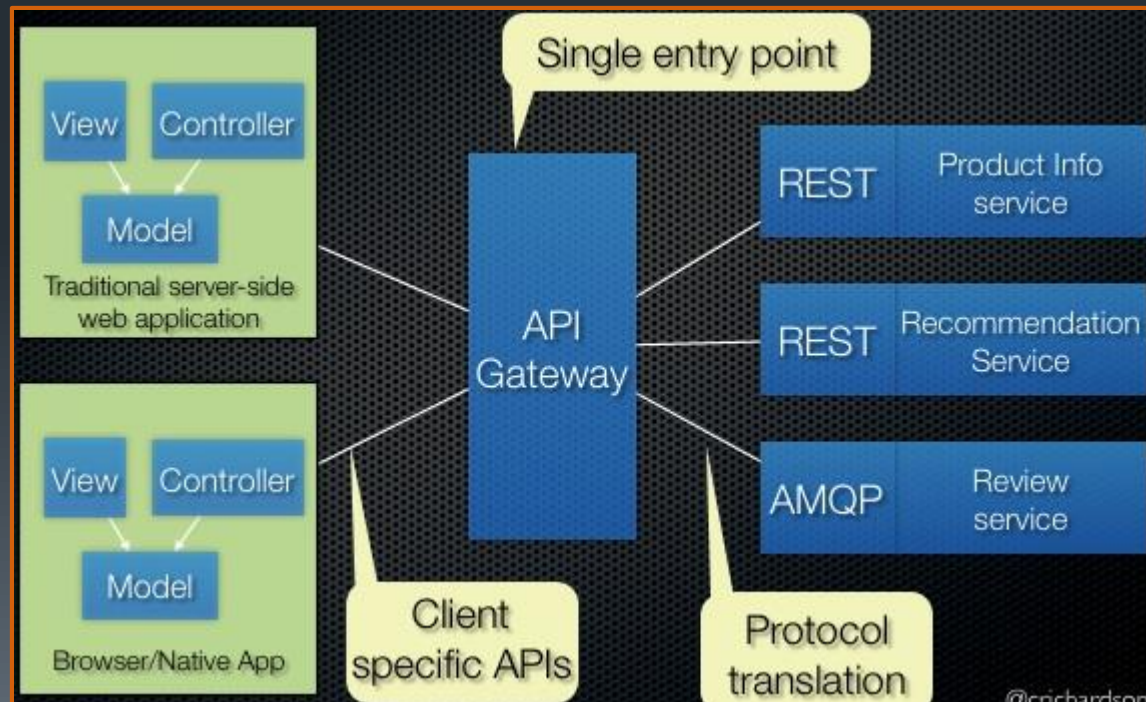
- **Direct Client-to-Microservice Communication**
 - A client could make requests to each microservice directly
 - Each microservice would have a public endpoint (`https://serviceName.api.company.name`) mapped to the microservice's load balancer
- There are challenges and limitations with this option
 - The mismatch between the needs of the client and the fine-grained APIs exposed by each of the microservices
 - The client has to make many separate requests to perform an operation
 - Chatty services in a 10Gb data center may work well, but they almost always produce problems over the slower, less reliable Internet
 - Microservices may use protocols that are not web-friendly (Thrift, AMQP messaging protocol, etc.)
- A huge drawback with this approach is that it makes it difficult to refactor the microservices
 - How the system is partitioned is now the client's concern (!)
 - Information about the implementation of services should never escape the enclosing bounded context
- It rarely makes sense for clients to talk directly to microservices



Gateway Features

Copyright 2013-2018, RX-M LLC

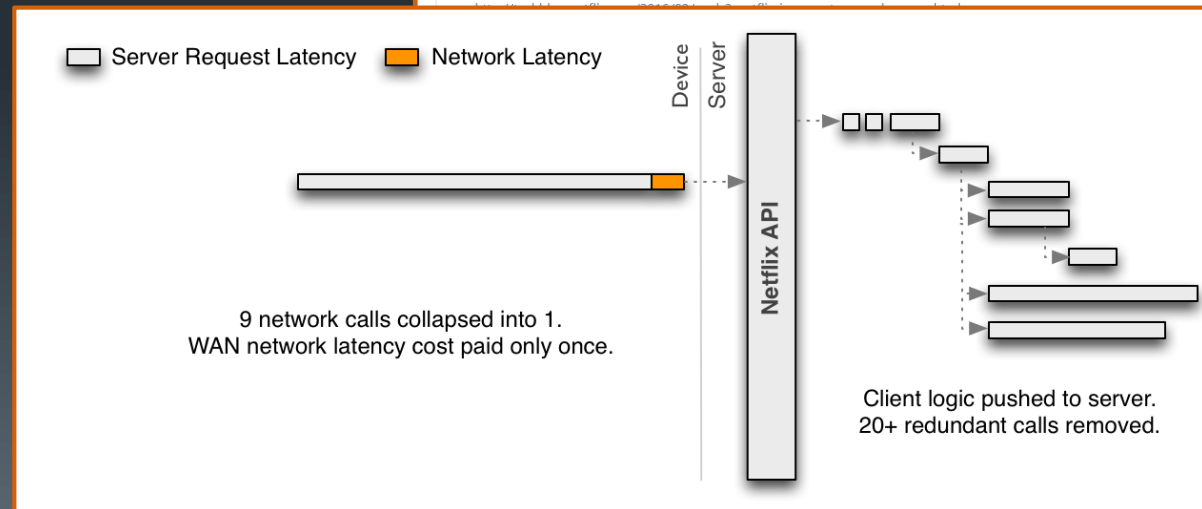
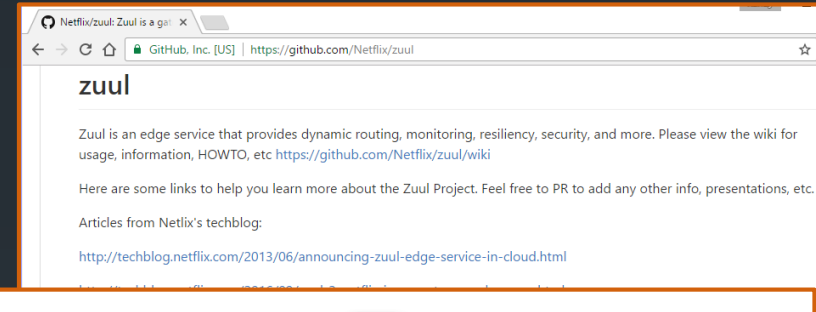
- API Gateways should allow systems to rapidly change behavior in order to react to changing circumstances
- Key functions:
 - **Authentication and Security** - identifying authentication requirements for each resource and rejecting requests that do not satisfy them
 - **Insights and Monitoring** - tracking meaningful data and statistics at the edge in order to give an accurate view of production
 - **Dynamic Routing** - dynamically routing requests to different backend clusters as needed
 - **Stress Testing** - gradually increasing the traffic to a cluster in order to gauge performance
 - **Circuit Breakers** - allocating capacity for each type of request and dropping requests that go over the limit
 - **Caching** - building some responses directly at the edge instead of forwarding them to an internal cluster
 - **Multiregion Resiliency** – routing requests across regions in order to diversify usage and move the edge closer to clients



Netflix Gateway

Copyright 2013-2018, RX-M LLC

- API Gateways are responsible for:
 - Request routing
 - Composition
 - Protocol translation
- Typically requests from external clients go through an API Gateway
 - Often request invoke multiple microservices requiring data aggregation
- An example API Gateway is the Netflix API Gateway **Zuul**
 - **Netflix streaming service** is available on hundreds of different kinds of devices including televisions, set-top boxes, tablets, phones, etc.
 - Handles billions of requests per day
 - Netflix attempted to provide a **one-size-fits-all API** for their streaming service, discovered it **didn't work** because device diversity and their unique needs
 - The **Zuul API Gateway** provides an **API tailored for each device** by running device-specific adapter code
 - An adapter typically handles each request by **invoking on average six to seven backend services**

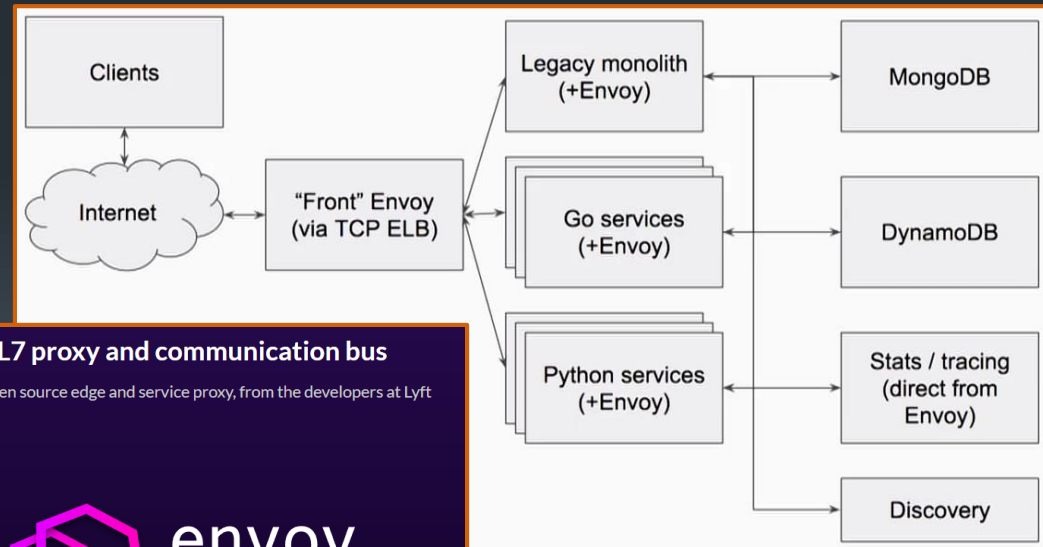


Envoy

Copyright 2013-2018, RX-M LLC

- A self contained OSS network service proxy
- **HTTP/2**
 - First class in/out support for HTTP/2 and transparent HTTP/1.1 to HTTP/2
- **Load balancing**
 - Automatic retries
 - Circuit breaking
 - Global rate limiting
 - Request shadowing (copy requests to QA)
 - Zone local LB (avoids cross zone fees)
- **Filtering**
 - Envoy allows HTTP/TCP/IP filtering
 - Filters can be chained
 - New filters can be written
- **Coded in C++11**
- **gRPC support**
- **HTTP routing**
 - Redirection, virtual hosts, virtual clusters, matching on different request parameters
- **TLS Support**
 - Termination and initiation, client certificate verification, and certificate pinning
- **Observability**
 - Envoy exposes rich statistics and distributed tracing via third party providers (e.g. light step, zipkin)

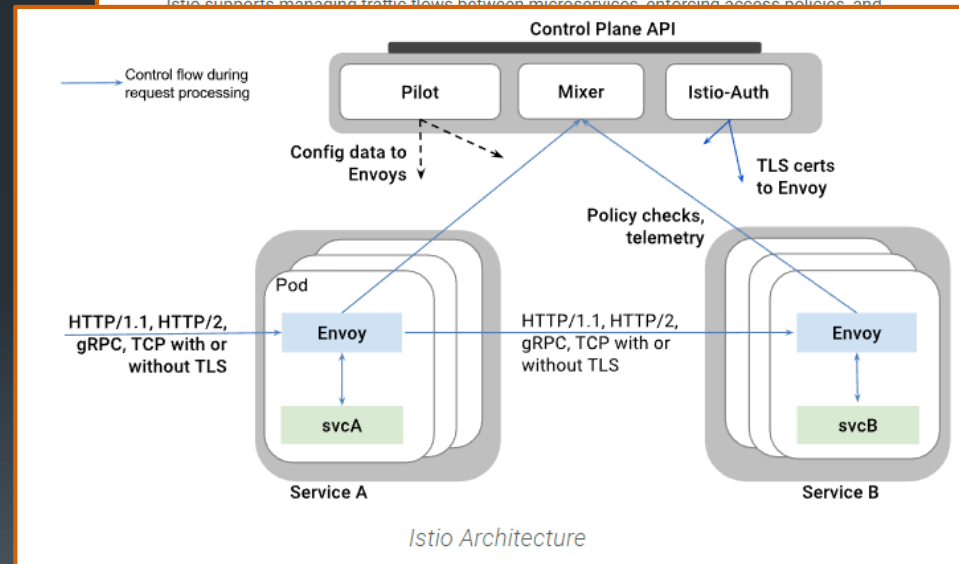
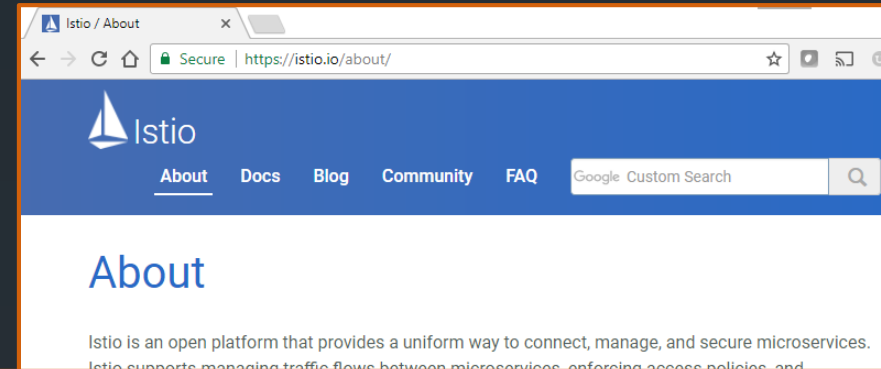
- **MongoDB**
 - Envoy contains a MongoDB wire format parser used to gather statistics about database connections
- **DynamoDB**
 - Envoy contains a DynamoDB API parser that is used to gather statistics about database requests and responses
- **Service discovery**
 - Envoy supports asynchronous DNS resolution as well as integration with an external service discovery service
- **Health checking**
 - Envoy is capable of active health checking of backend servers
 - Active health checking along with service discovery yields eventually consistent and extremely resilient load balancing



Istio

Copyright 2013-2018, RX-M LLC

- In a phrase: **Kubernetes integrated Envoy**
- Istio supports **policy based** traffic management between microservices
- Istio **generates telemetry** data
- Istio is **transparent to the actual microservices** it intermediates
- Features:
 - Automatic load balancing for HTTP, gRPC, and TCP traffic
 - Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection
 - A pluggable policy layer and configuration API supporting access controls, rate limits and quotas
 - Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress
 - Secure service-to-service authentication with strong identity assertions between services in a cluster
- Only supports Kubernetes today
 - Support is planned for additional Cloud Foundry, Mesos, and bare metal
- Components:
 - **Envoy** Service Proxy
 - **Mixer** collects telemetry and enforces service mesh policy
 - **Pilot** is an interface for users to configure Istio via
 - **Istio-Auth** provides service-to-service and end-user authentication using mutual TLS



Welcome
Concepts
Tasks
Samples
Reference

Frequently Asked Questions
Troubleshooting Guide
Report a Bug
Report a Doc Issue
Edit This Page on GitHub

User | Dev Mailing Lists
Twitter
GitHub

Copyright © 2017 Istio Authors
Istio Version 0.1
This site was built on 10-Jul-2017

Istio and CloudFoundry

Copyright 2013-2018, RX-M LLC

- Pivotal is making Istio core to
 - App platforms
 - CFAR/PCF
 - Container platforms
 - CFCR/PKS
 - Function platforms
 - Knative/PFS
 - Pivotal Function Service (PFS) is not yet released



The banner features a blue background with a white wavy pattern. On the left is the Pivotal Cloud Foundry logo, which consists of three interlocking loops. To the right of the logo is the text "Pivotal Cloud Foundry" in white. Further right is a white sailboat icon. Below the banner, the text "Istio 1.0 is Now GA!" is displayed in a large, bold, black font. Underneath this, a smaller line of text reads: "Istio 1.0 is now GA. Here's what it does, why it matters, and why we're adding it to Cloud Foundry and other projects to make your life better."

Istio 1.0 is Now GA!

Istio 1.0 is now GA. Here's what it does, why it matters, and why we're adding it to Cloud Foundry and other projects to make your life better.

AWS API Gateway

- A managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at scale
- In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints
- Support for:
 - Amazon Elastic Compute Cloud (Amazon EC2)
 - AWS Lambda
- Handles:
 - traffic management
 - authorization
 - access control
 - Monitoring
 - API version management
- Charges for the API calls received and data transferred out

```
{
  "swagger": "2.0",
  "info": {
    "title": "PetStore",
    "description": "Sample API"
  },
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "text/html"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "responses": {
          "default": {
            "statusCode": "200",
            "responseParameters": {
              "method.response.header.Content-Type": "'text/html'"
            },
            "responseTemplates": {
              "text/html": "<html>\n<head>\n...
              </body>\n</html>"
            }
          }
        },
        "requestTemplates": {
          "application/json": "{$\"statusCode\": 200}"
        },
        "type": "mock"
      }
    },
    ...
  },
  ...
}
```

Summary

- API gateway expose a client centric aggregate API, encapsulating the many finer grained services within a given bounded context

Lab 9

- Gateways