中国科学技术大学
University of Science and Technology of China

Software Architecture

SSE USTC Qing Ding
dingqing@ustc.edu.cn
http://staff.ustc.edu.cn/~dingqing

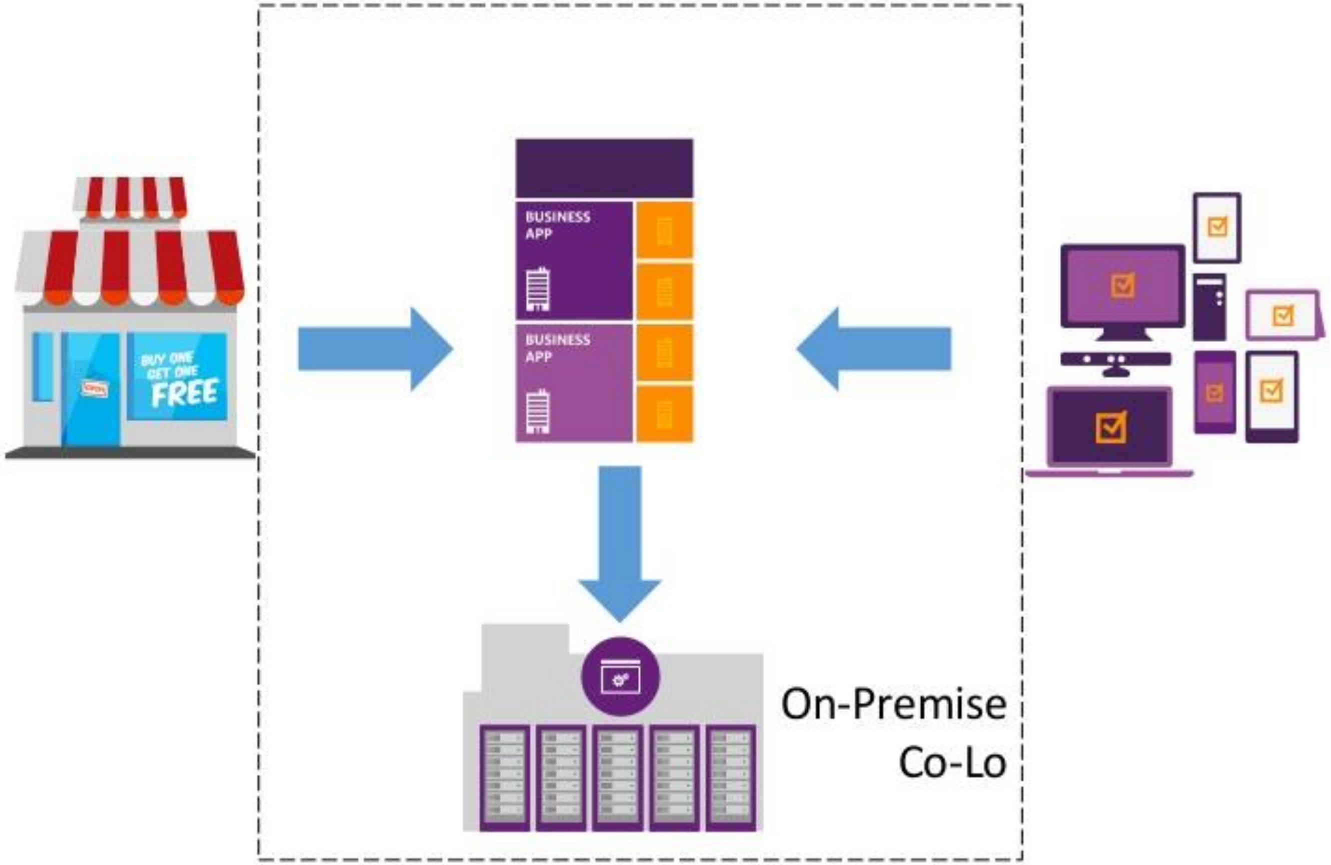
Serverless Architecture

Function as a Service

了解特点，不用知道具体细节

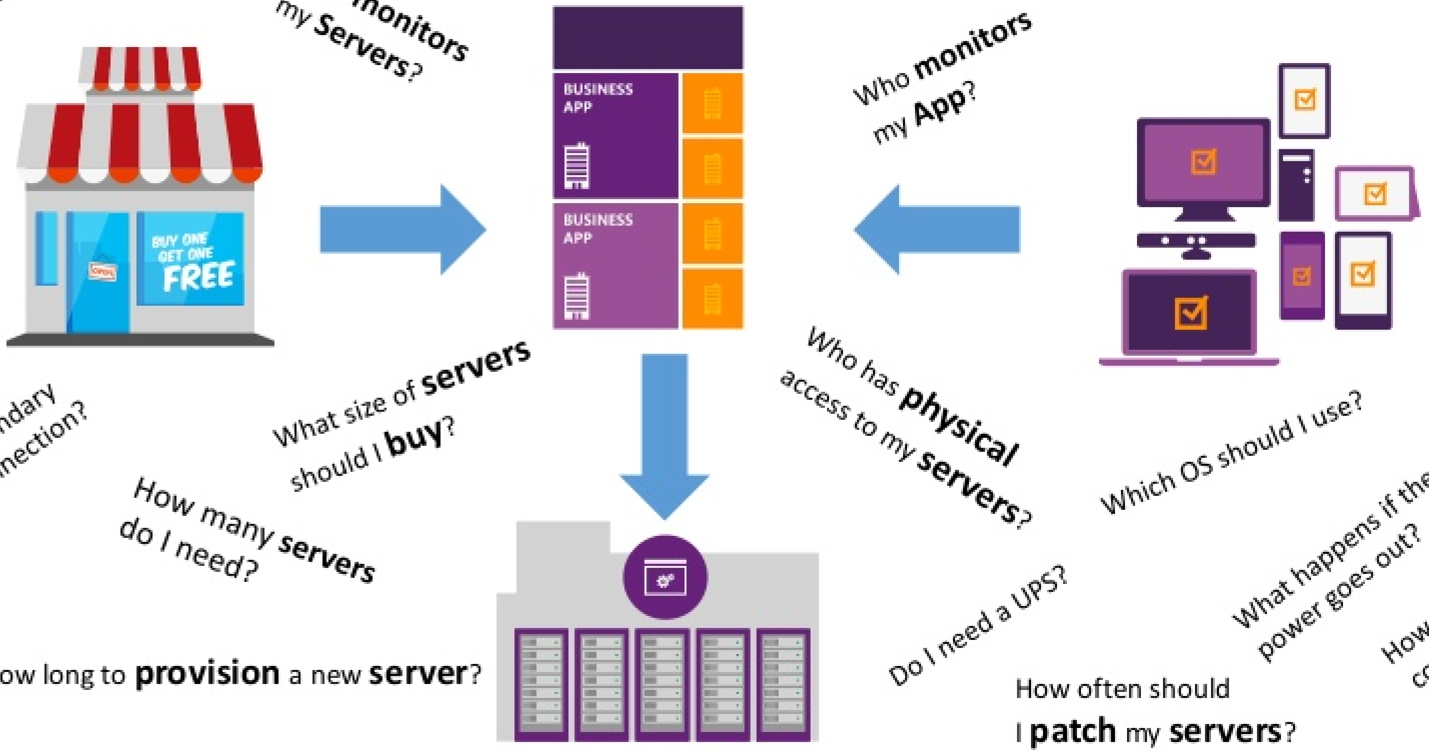
- Evolution of Serverless Computing
- *What is Serverless*
- Why is Serverless attractive
- AWS Architecture
- Apache OpenWhisk
- Azure Functions
- Future of Serverless: Research Challenges and Questions

Before cloud



On-Premise
Co-Lo

Before cloud



Which packages should be on my **server**?

What is the right **size** of **servers** for my business needs?

How I **scale** my app?

What happens in case of **server hardware** failure?

How can I increase **server** utilization?

What media should I use to keep backup?

Who **monitors** my **Servers**?

How I **deploy** new **code** to my **server**?

Who **monitors** my **App**?

How often should I backup my **server**?

What storage I need to use?

Do I need secondary network connection?

What size of **servers** should I **buy**?

How many **servers** do I need?

Who has **physical** access to my **servers**?

Which OS should I use?

Are my **server** in a secure location?

It takes how long to **provision** a new **server**?

Do I need a UPS?

What happens if the power goes out?

How can I dynamically configure my app?

How often should I **patch** my **servers**?

Is it PaaS time?

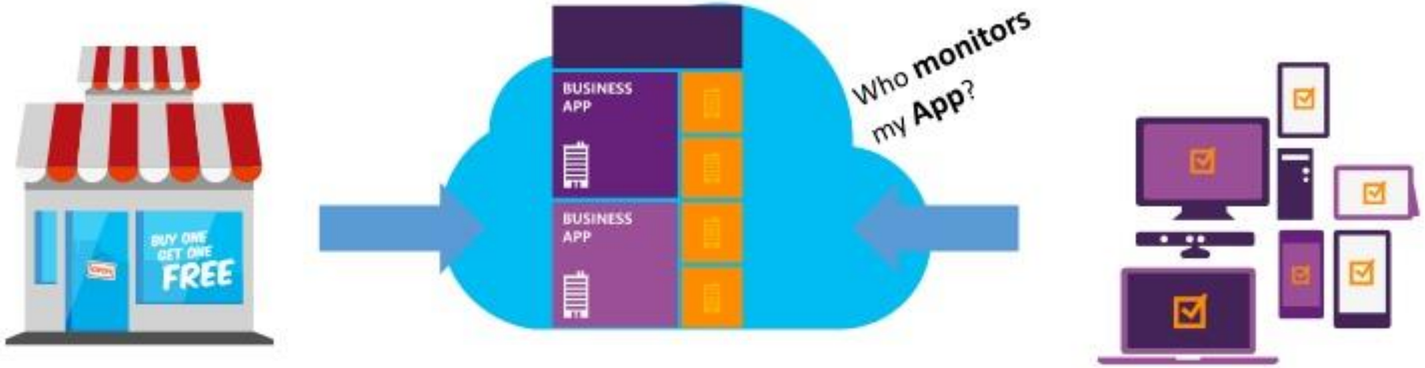
Which packages should be on my **server**?

What is the right **size** of **servers** for my business needs?

How I **scale** my app?

How can I increase **server** utilization?

How I **deploy** new **code** to my **server**?



Who **monitors** my **App**?

How often should I backup my **server**?

How many **servers** do I need?

Which OS should I use?

How can I dynamically configure my app?

How often should I **patch** my **servers**?

Serverless . . .

What is the right **size** of **servers** for my business needs?

How I **scale** my app?

How can I increase **server** utilization?

How many **servers** do I need?

What is Serverless?

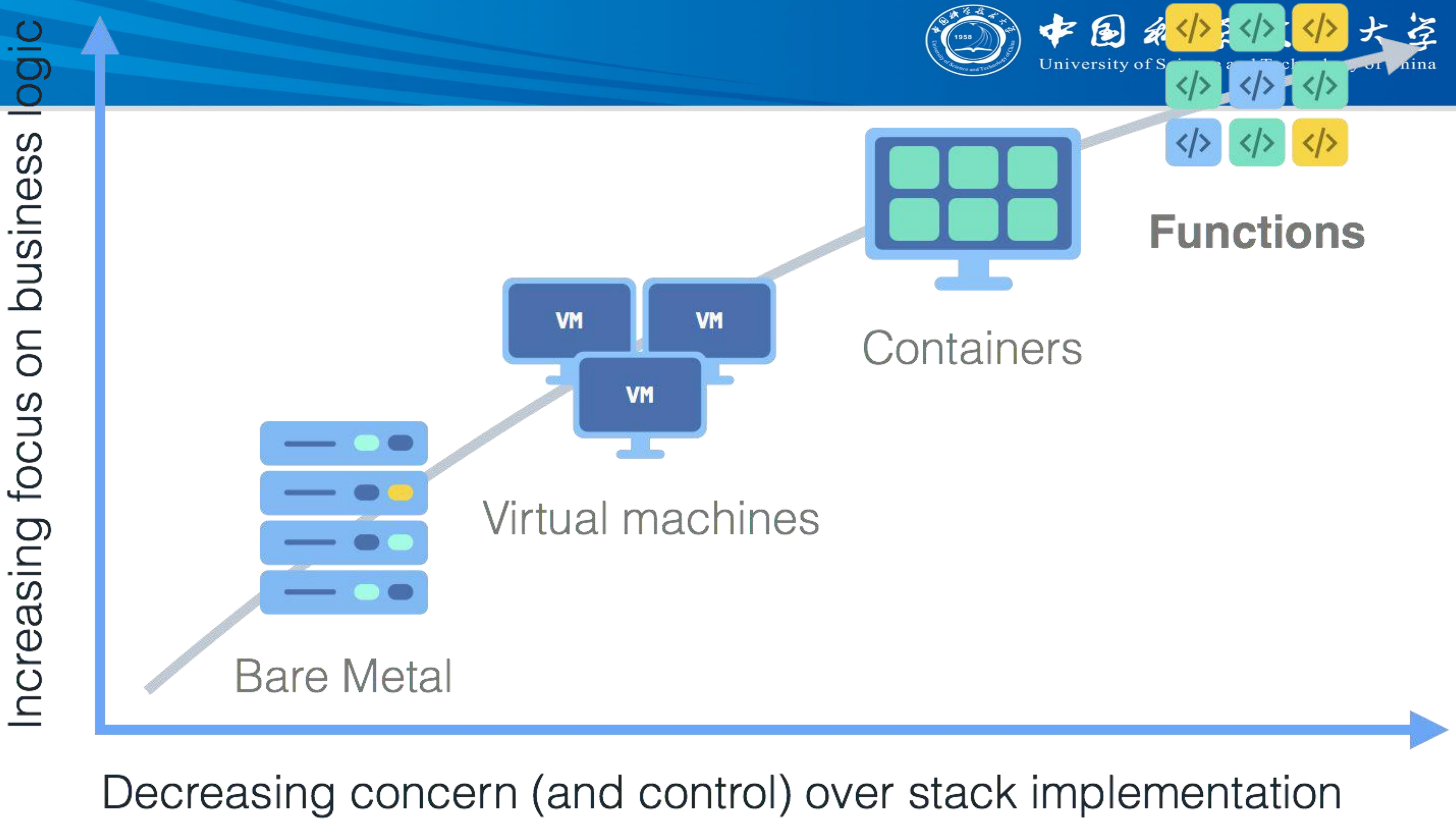
Abstraction of servers

服务抽象化

Event-driven/ instant scale

事件驱动

Sub-second billing



Increasing focus on business logic

Bare Metal

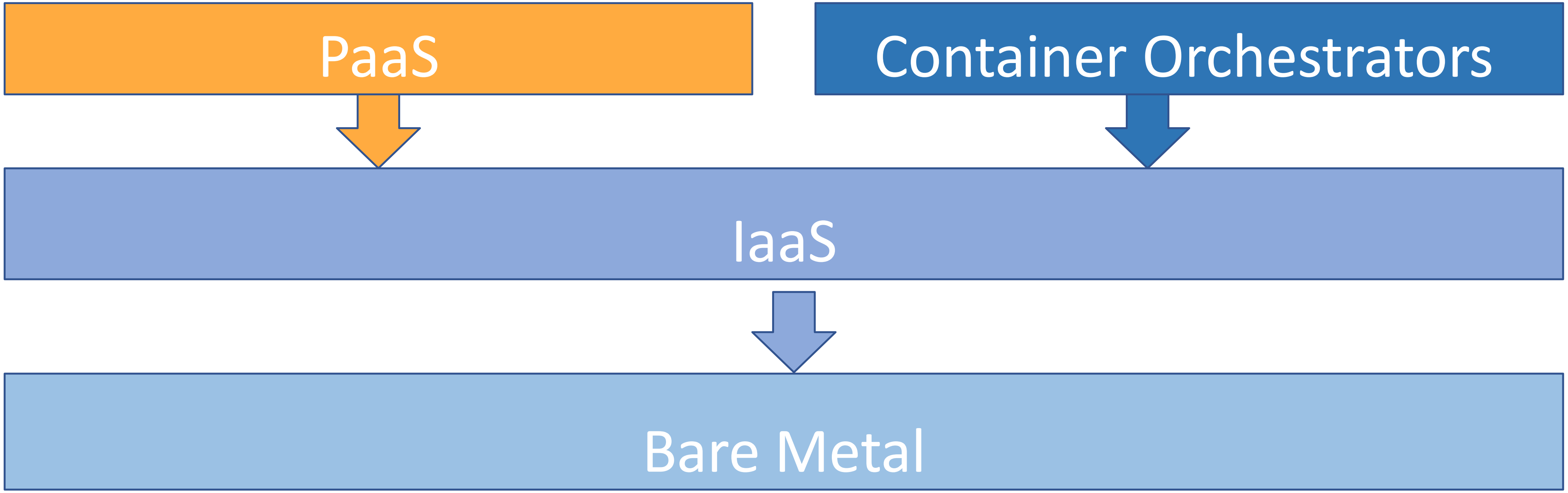
Virtual machines

Containers

Functions

Decreasing concern (and control) over stack implementation

Evolution Of Serverless

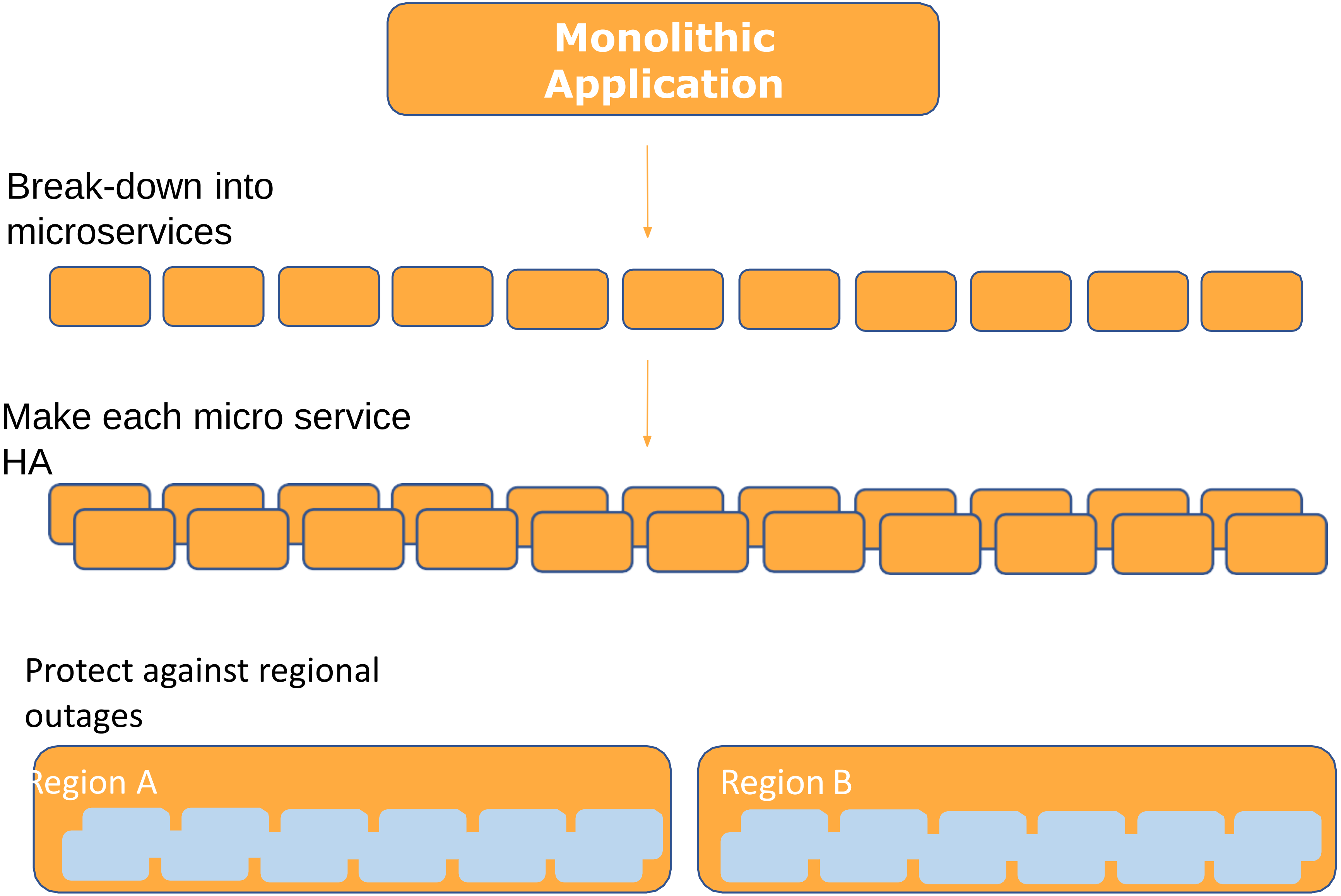


PaaS

Container Orchestrators

IaaS

Bare Metal



Break-down into microservices

Make each micro service HA

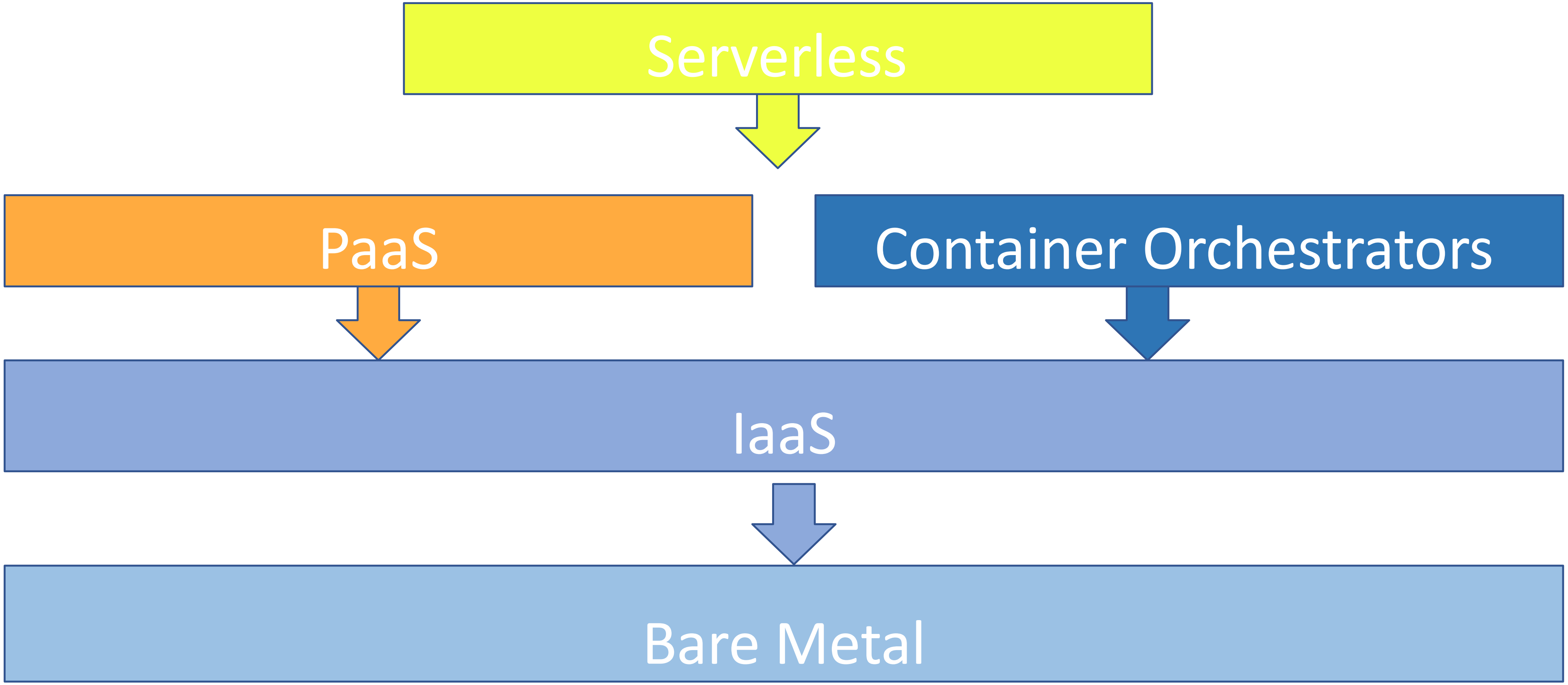
Protect against regional outages

Explosion in number of containers / processes:

Increase of infrastructure cost footprint

Increase of operational management cost and complexity

Enter Serverless



IaaS – PaaS, FaaS, SaaS

Private Cloud	Infrastructure (as a service)	Platform (as a service)	Function (as a service) (serverless arch)	Software (as a service)
🟧 Functions	🟧 Functions	🟧 Function	🟧 Functions	🟦 Functions
🟧 Data	🟧 Data	🟧 Data	🟧 Data	🟦 Data
🟧 Application	🟧 Data	🟧 Application	🟦 Application	🟦 Application
🟧 Runtime	🟧 Runtime	🟦 Runtime	🟦 Runtime	🟦 Runtime
🟧 Backend Code	🟧 Backend Code	🟦 Backend Code	🟦 Backend Code	🟦 Backend Code
🟧 OS	🟧 OS	🟦 OS	🟦 OS	🟦 OS
🟧 Virtualization	🟦 Virtualization	🟦 Virtualization	🟦 Virtualization	🟦 Virtualization
🟧 Server Machines	🟦 Server Machines	🟦 Server Machines	🟦 Server Machines	🟦 Server Machines
🟧 Storage	🟦 Storage	🟦 Storage	🟦 Storage	🟦 Storage
🟧 Networking	🟦 Networking	🟦 Networking	🟦 Networking	🟦 Networking

🟦 Public Cloud Provider - responsibility

🟧 Application Writer - responsibility

Awesome Vizualisation picked from : Ref : http://www.slideshare.net/manuel_silveyra/austin-cf-meetup-20150224/3

PS: We expect Container as a Service term in 2017-18 too, there is a separate section on it later

What is Serverless?

a cloud-native platform

for

short-running, stateless computation

and

event-driven applications

which

scales up and down instantly and automatically

and

charges for actual usage at a millisecond granularity

Server-less means no servers?
Or worry-less about servers?

Runs code **only** on-demand on a per-request basis

Serverless deployment & operations model

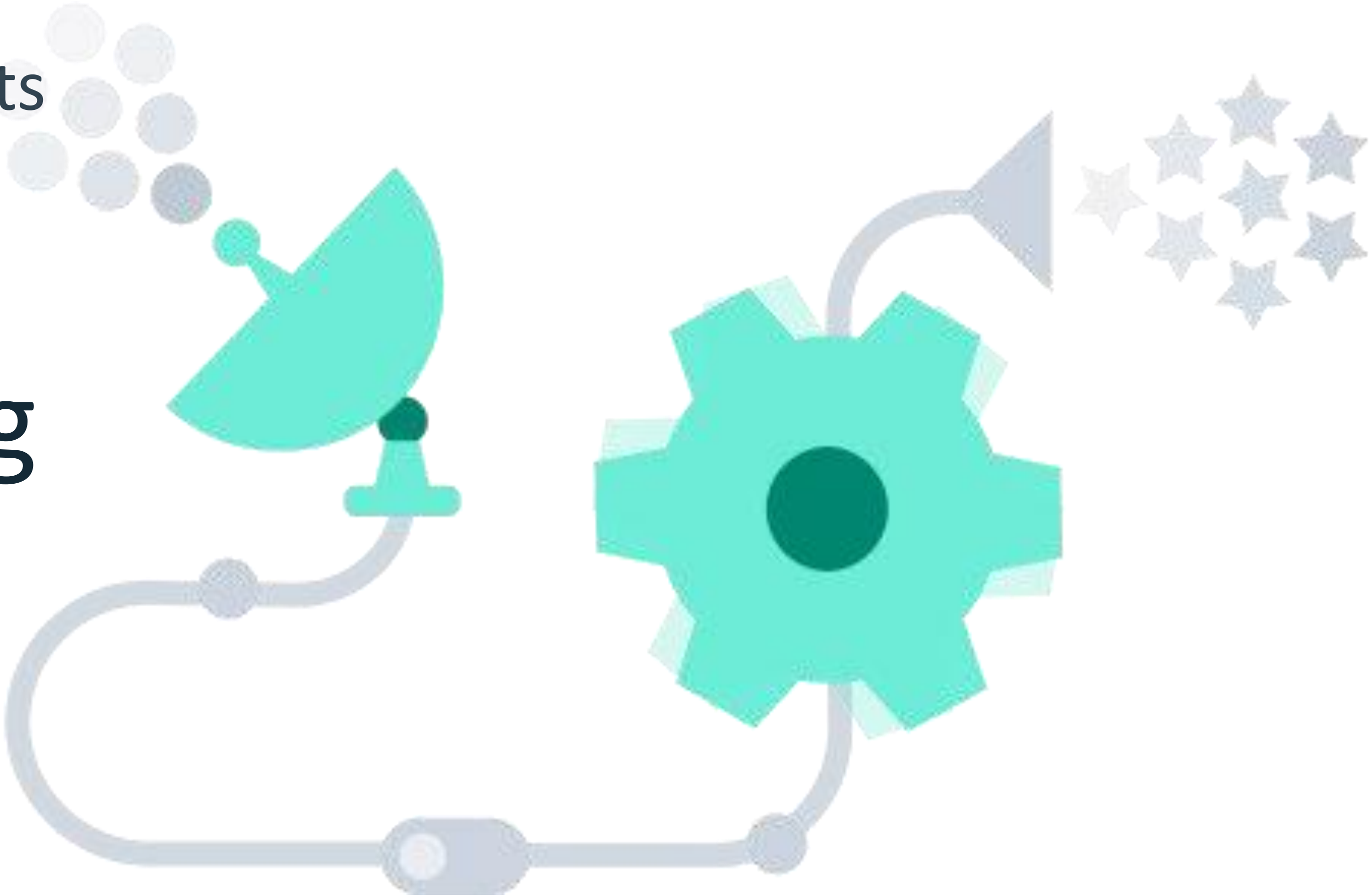
No servers

Just code

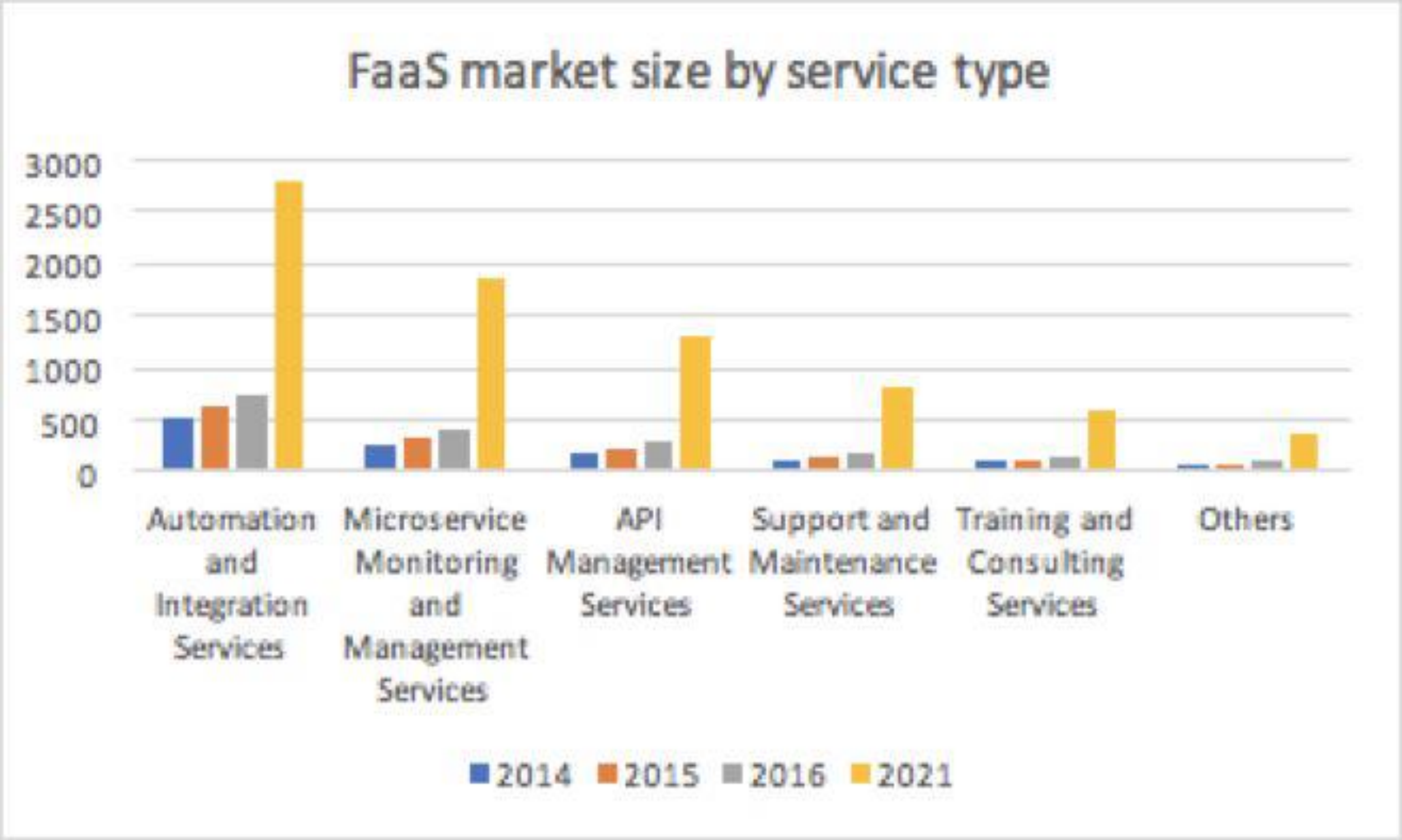
What triggers code execution?

Runs code **in response** to events

Event-programming model



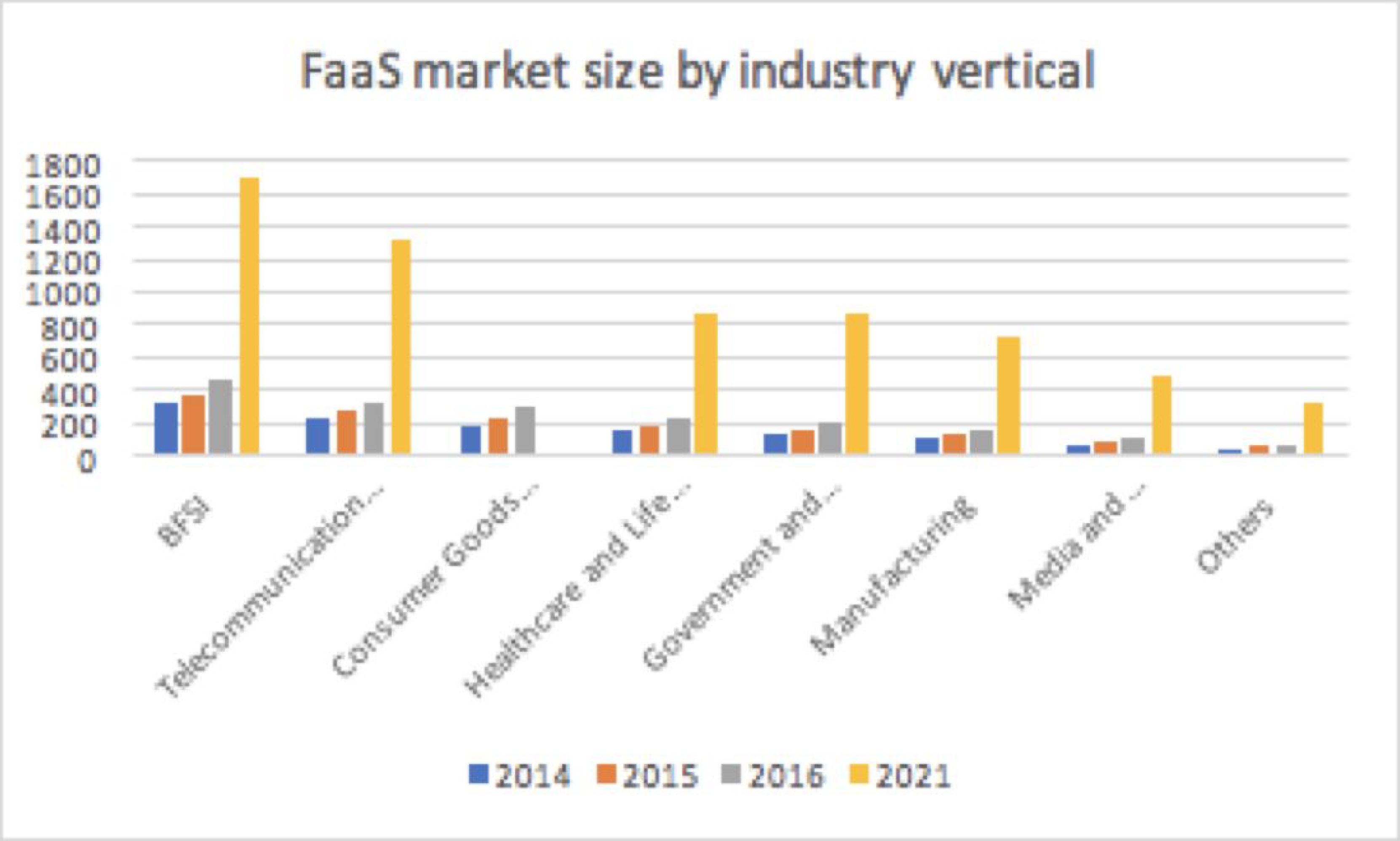
FaaS market is growing quickly



Source: FaaS Market - Global Forecast to 2021 - study by MarketsAndMarkets (http://www.marketsandmarkets.com/)

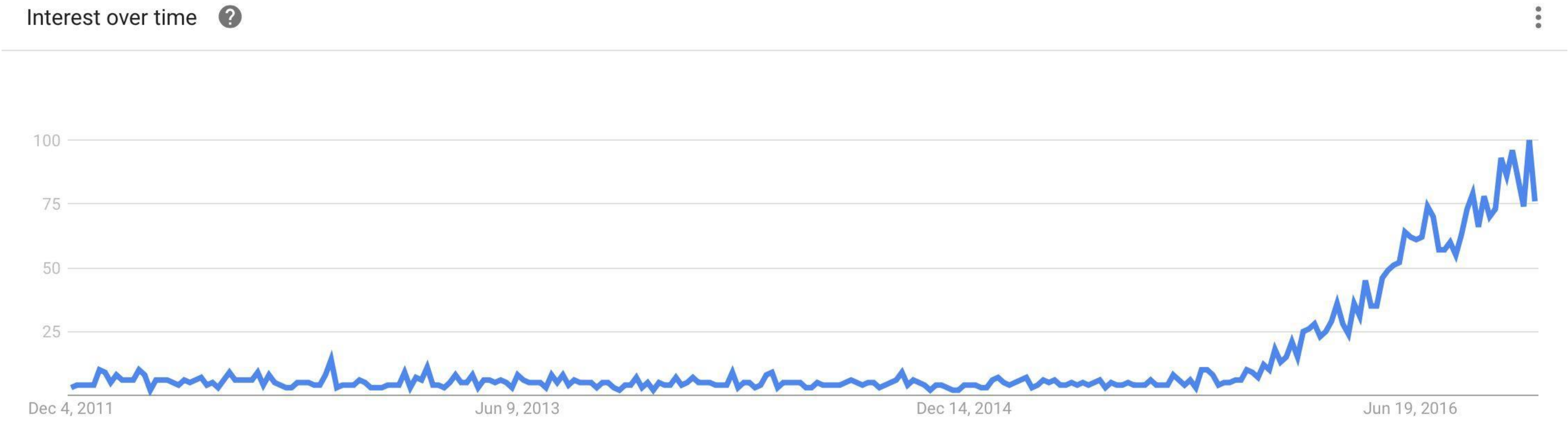
FaaS market is growing quickly

中国科学技术大学
University of Science and Technology of China



Source: *FaaS Market - Global Forecast to 2021 - study by MarketsAndMarkets (http://www.marketsandmarkets.com/)*

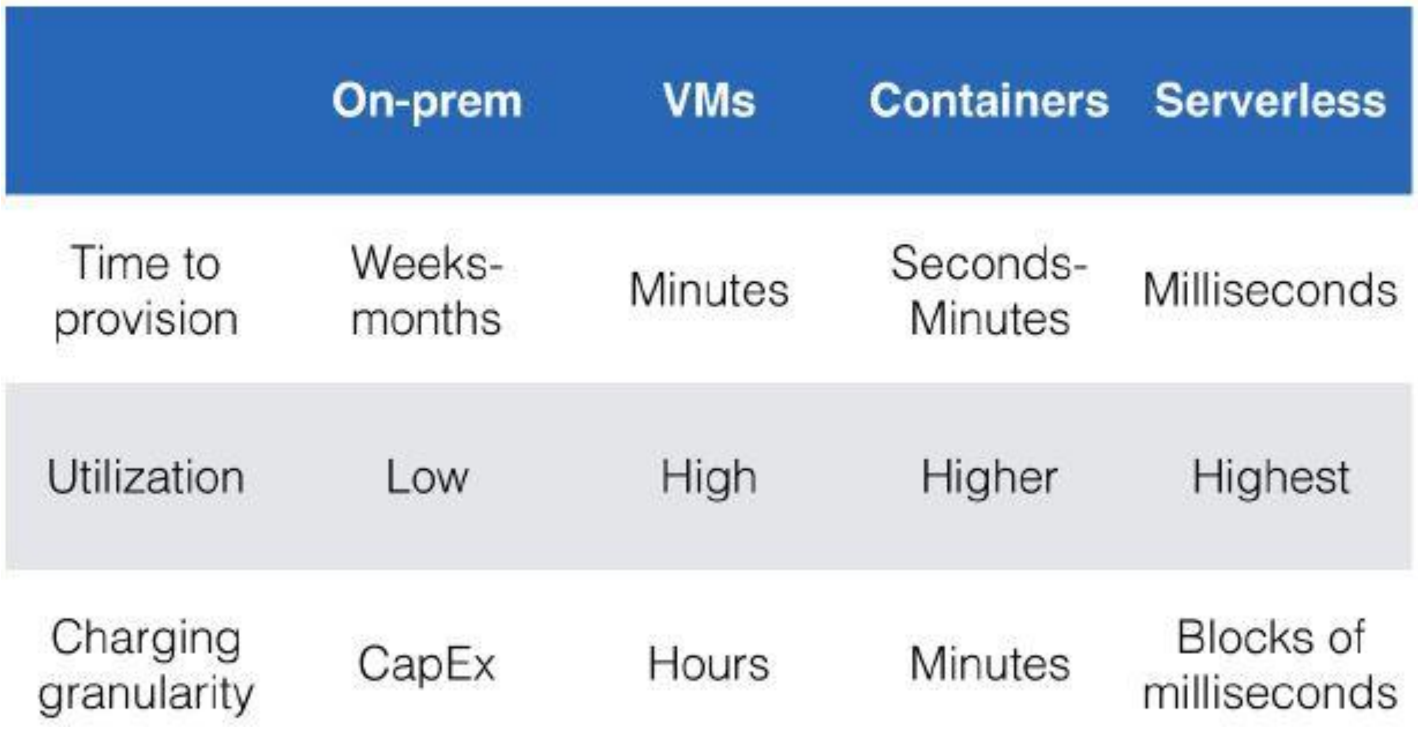
Google Search Trend over time



Interest over time

Why is Serverless attractive?

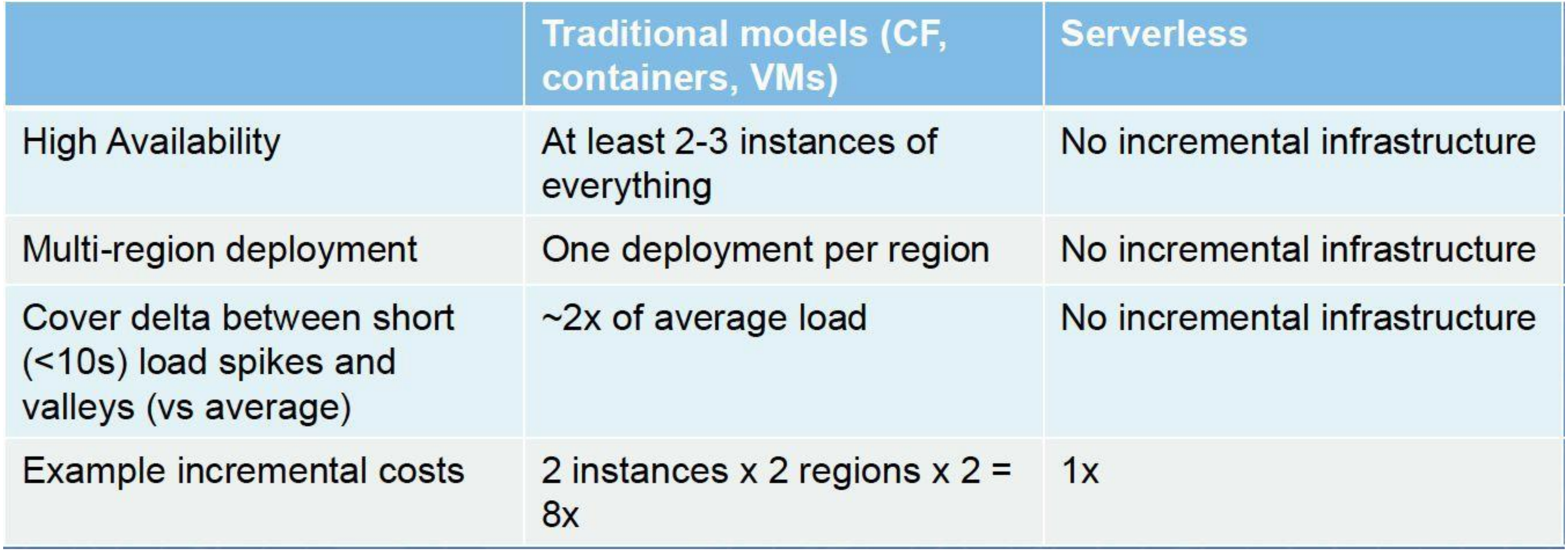
- Making app development & ops dramatically faster, cheaper, easier
- Drives infrastructure cost savings



	On-prem	VMs	Containers	Serverless
Time to provision	Weeks-months	Minutes	Seconds-Minutes	Milliseconds
Utilization	Low	High	Higher	Highest
Charging granularity	CapEx	Hours	Minutes	Blocks of milliseconds

Source: Jason McGee, IBM; Serverless Conference 2017.

Key factors for infrastructure cost savings



	Traditional models (CF, containers, VMs)	Serverless
High Availability	At least 2-3 instances of everything	No incremental infrastructure
Multi-region deployment	One deployment per region	No incremental infrastructure
Cover delta between short (<10s) load spikes and valleys (vs average)	~2x of average load	No incremental infrastructure
Example incremental costs	2 instances x 2 regions x 2 = 8x	1x

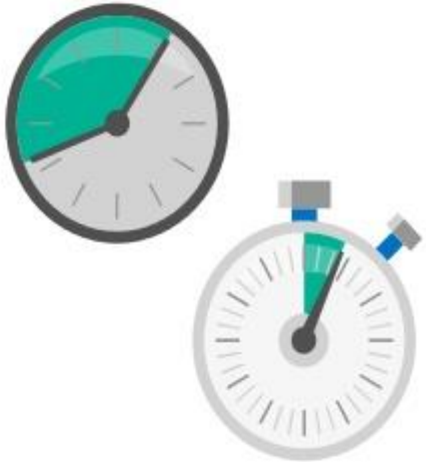
Benefits of Serverless Computing

- No Servers to Manage
- Continuous Scaling
- Dynamic allocation of resources
- Avoid overallocation of resources
- Never Pay for Idle: pay-per-usage

Benefits of Serverless?

Reduced
DevOps

Focus on
Business
Logic



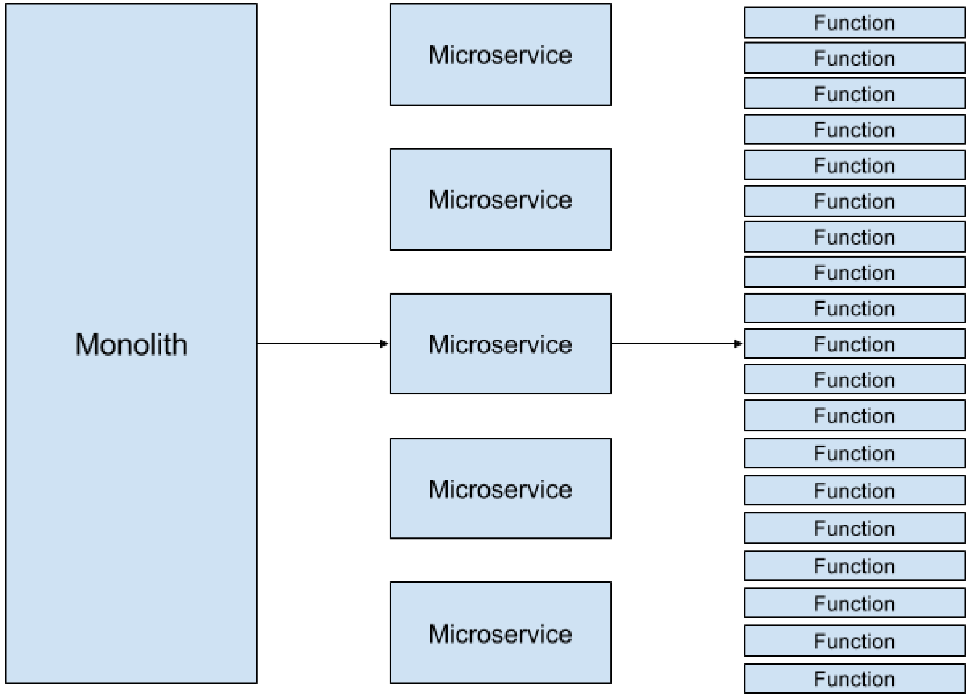
Reduced Time
To Market

Serverless Function

We’ve gone from monoliths to microservices to functions

Microservices

- Smaller-grained services
- Specified Functions
- Defined Capabilities



Monolith

Microservice
Microservice
Microservice
Microservice
Microservice

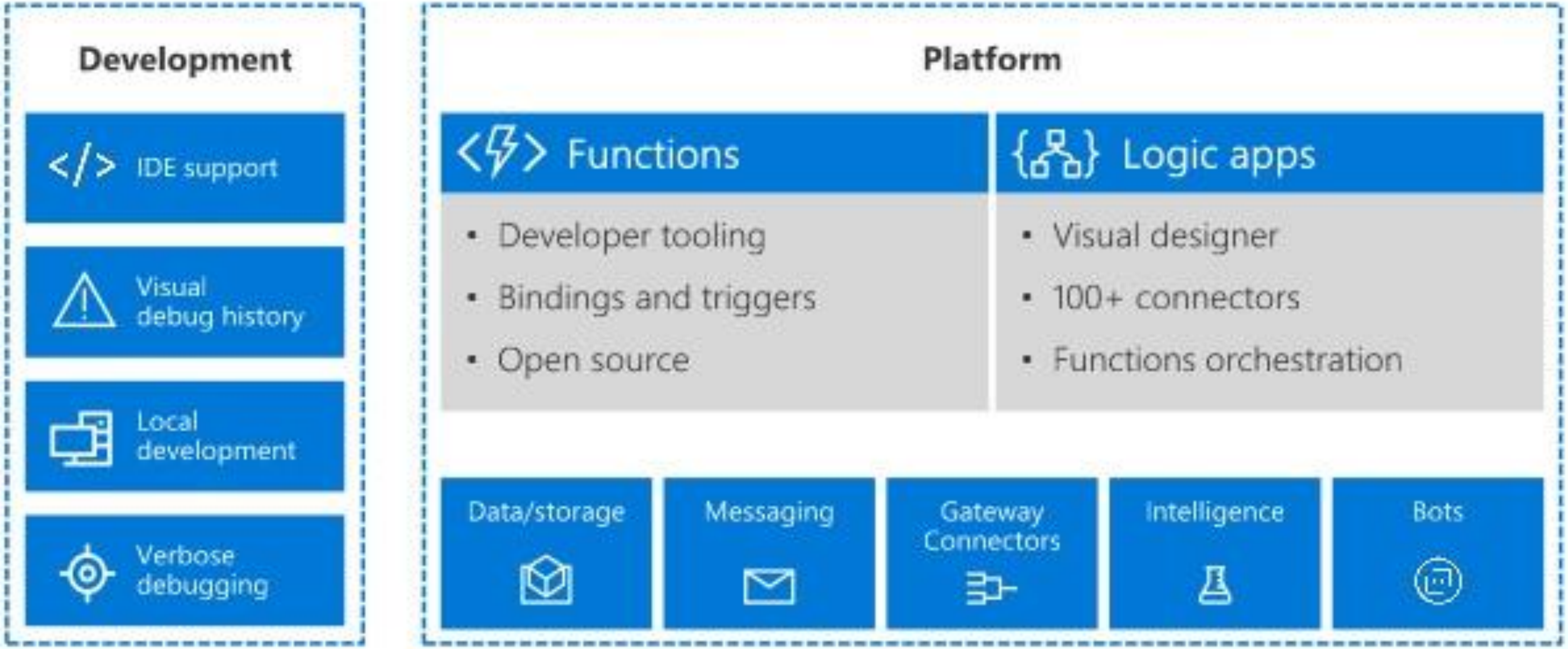
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function
Function

- Event handler
- Serverless back ends
- Data processing

“Serverless function” or more accurately “Functions as a Service”

- **Principles of FaaS:**
- Complete abstraction of servers away from the developer
- Billing based on consumption and executions, not server instance sizes
- Services that are event-driven and instantaneously scalable

Serverless application platform components



Development

- IDE support
- Visual debug history
- Local development
- Verbose debugging

Platform

Functions

- Developer tooling
- Bindings and triggers
- Open source

Logic apps

- Visual designer
- 100+ connectors
- Functions orchestration

Data/storage | Messaging | Gateway Connectors | Intelligence | Bots

What is Serverless good for?

Serverless is good for

short-running stateless event-driven

短时、无状态、事件驱动的

Microservices Mobile

Backends

Bots, ML Inferencing IoT

Modest Stream Processing

Service integration

Serverless is not good for

long-running stateful number crunching

长时、有状态、数值计算的

Databases

Deep Learning Training

eavy-Duty Stream Analytics

Spark/Hadoop Analytics Numerical

Simulation

deo Streaming

Current Platforms for Serverless

APACHE OpenWhisk

AWS Lambda

OpenLambda

fission

Kubernetes

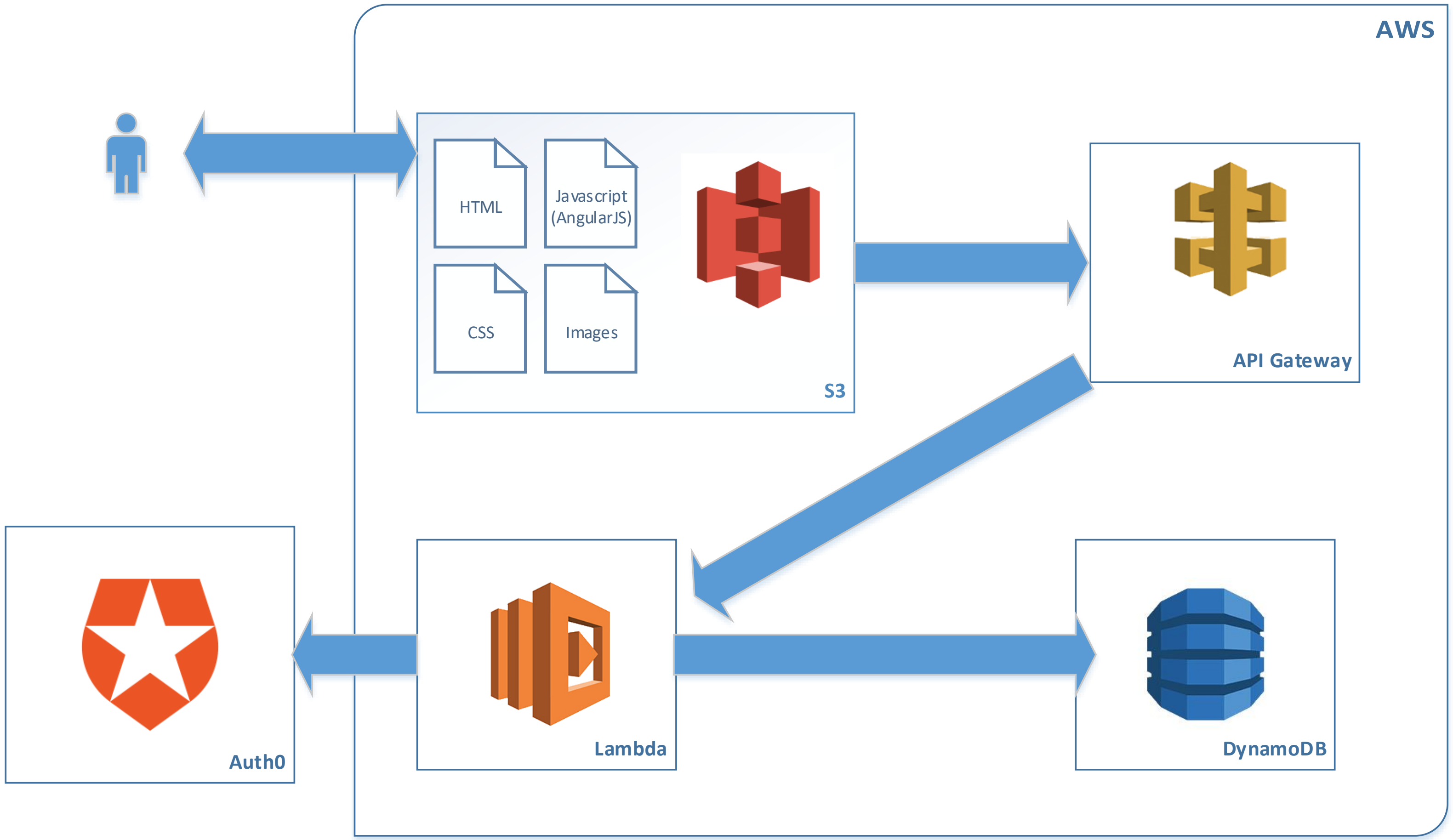
Iron.io

Azure Functions

Google Functions

Red-Hat

AWS Architecture



AWS Architecture – S3

- Cloud document storage
  - Hosts static web resources
  - Can be configured to host public websites
    - HTML, CSS, JavaScript, images
    - Supports CNAME aliases
    - Supports .htaccess style configuration for URL rewrite and redirect
    - Supports SSL for custom domain names
  - Billed for storage and for transfer

AWS Architecture – API Gateway

- Acts as the ‘front door’ to the application
- Handles authorization and access control
- Exposes Lambda functions to your front-end application code
- Billed for API calls

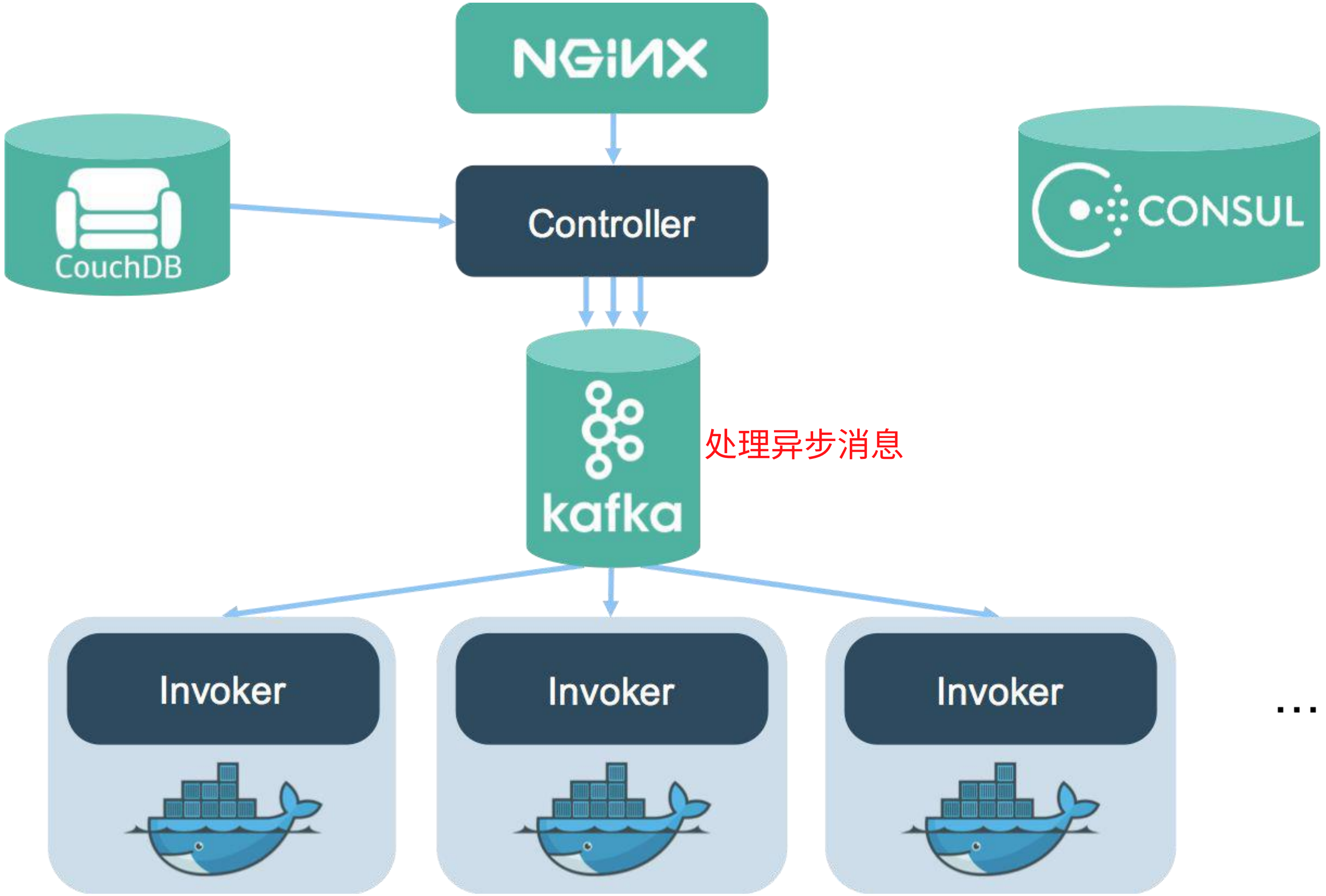
AWS Architecture – Lambda

- Run code without servers
  - Supports Node.js, Python, Java, C#, and Go.
  - Can use existing libraries
- Upload functions, configure triggers
- Works closely with API gateway via proxy integration
- Business logic and data access code goes here
- Billed for compute cycles (function run time)

AWS Architecture – DynamoDB

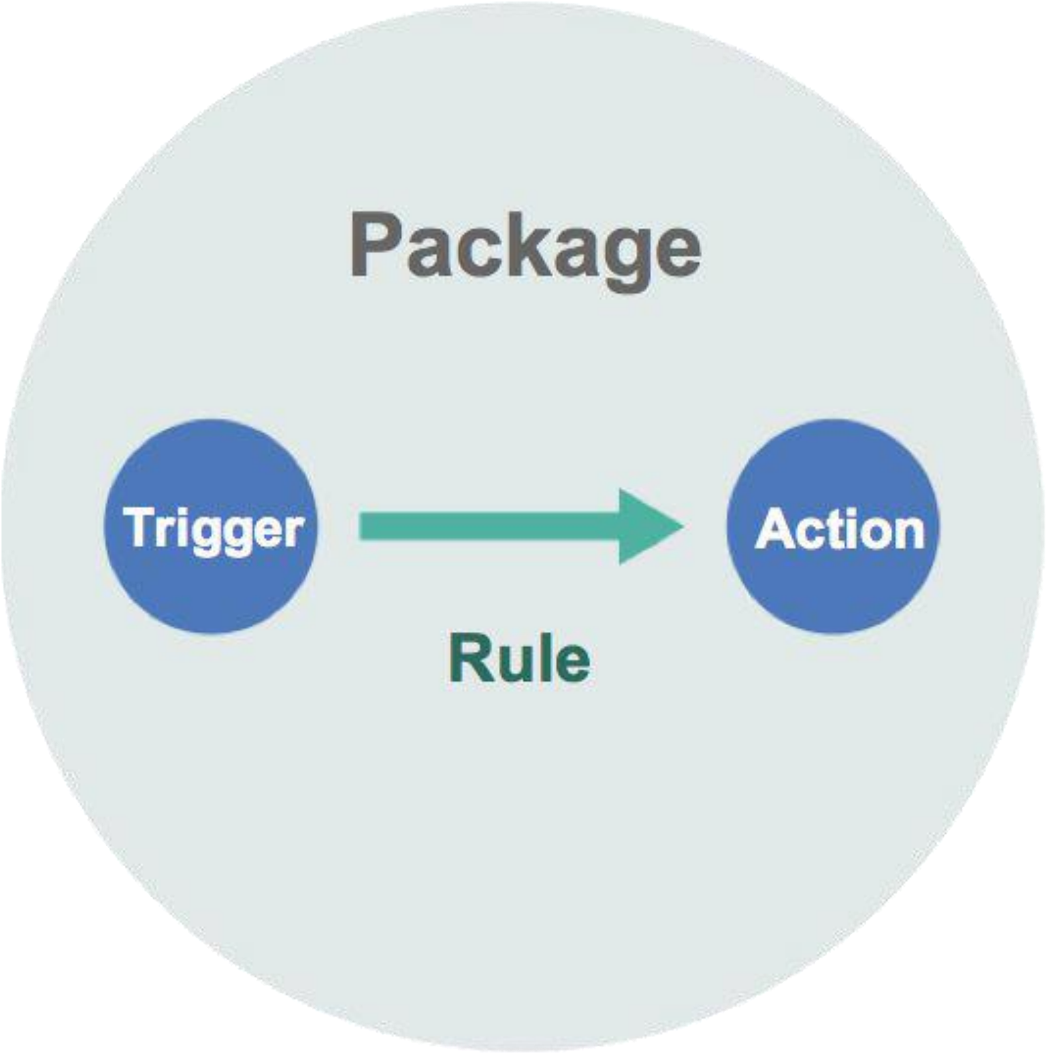
- Document database in the cloud
  - Non-relational (NoSQL)
  - Fast and scalable
- Built in security, backup, restore, caching
- Billed for storage and for transfer

Apache OpenWhisk Serverless Architecture



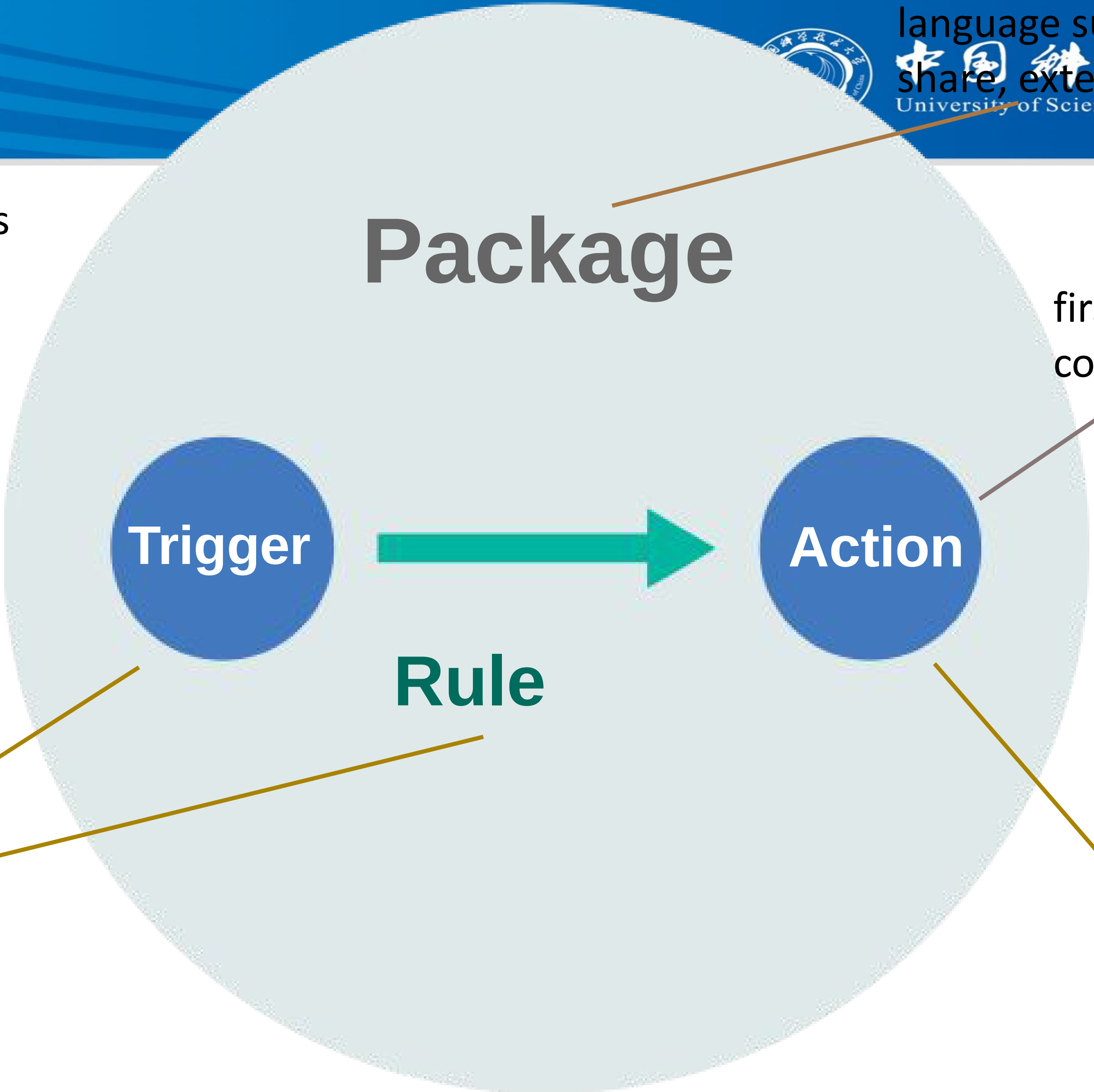
处理异步消息

Apache OpenWhisk: High-level serverless programming model



language support to encapsulate, share, extend code

all constructs first-class — powerful extensible language



first-class functions compose via sequences

first-class event-driven programming constructs

docker containers as actions

A

Action: a stateless function
(event handler)

A Action

```
function main(params) {   console.log( “Hello
    “ + params.name);
return { msg:  “Goodbye  “ + params.name) };
}
```

A

Action: Python

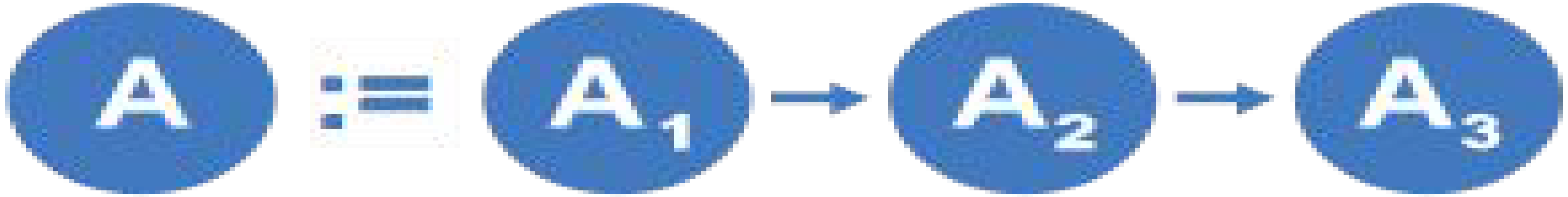
```
def lambda_handler(event, context):
    print("hello world")
```

A

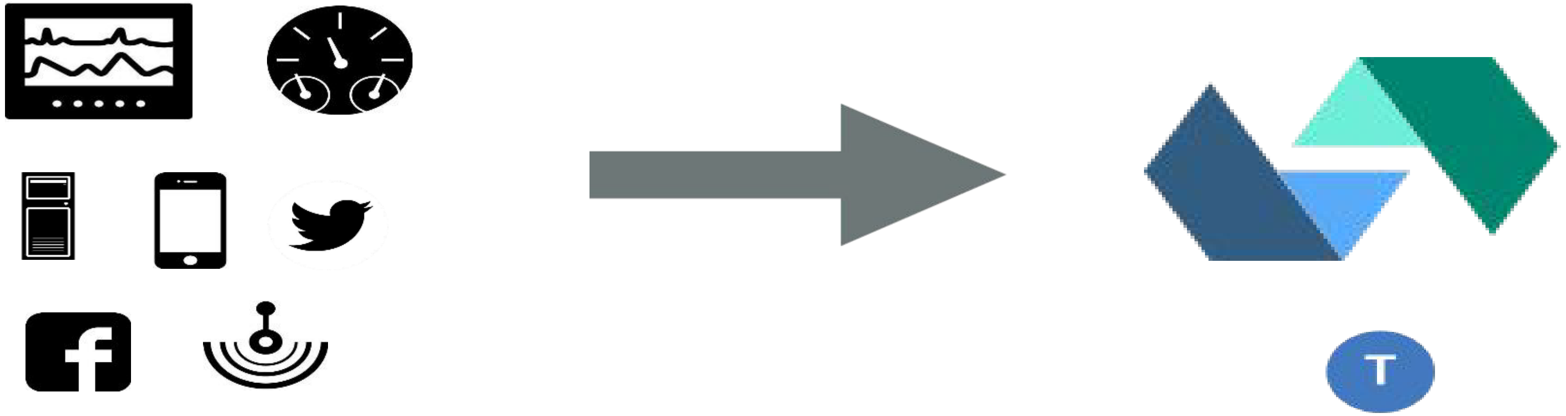
Action

```
func main(params:[String:Any]) -> [String:Any]
   {  var reply = [String:Any] ()
if let name = params[“name”] as? String
   {  print(“Hello \(name)”)
reply[“msg”] = “Goodbye \(name)”
}
return reply
}
```

A Action: sequence



Trigger: a class of events (feed)



AWS Lambda Trigger Sources

DATA STORES

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis
- Amazon Cognito

ENDPOINTS

- Amazon Alexa
- Amazon API Gateway
- AWS IoT

CONFIGURATION REPOSITORIES

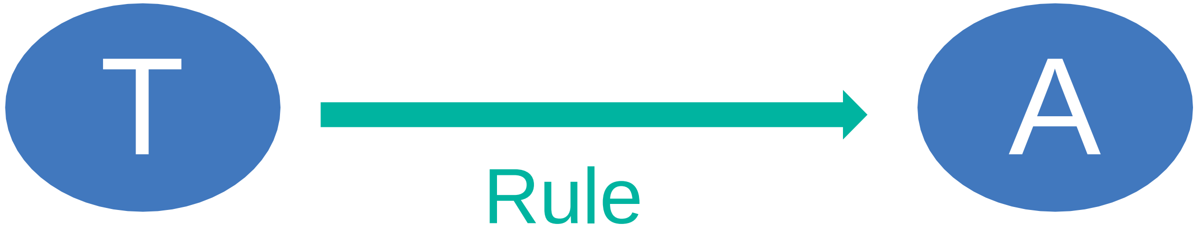
- AWS CloudFormation
- AWS CloudTrail
- AWS CodeCommit
- Amazon CloudWatch

EVENT/MESSAGE SERVICES

- Amazon SES
- Amazon SNS
- Cron events

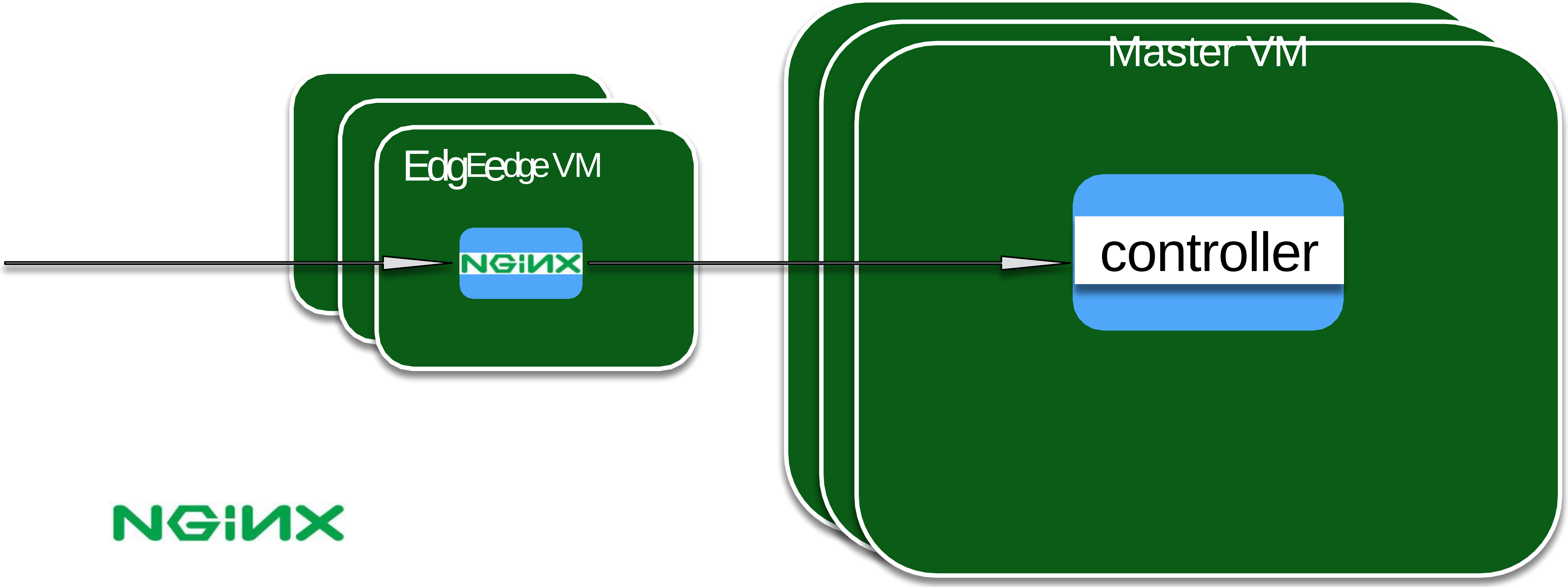
Source: Deep Dive into AWS Lambda, Vyom Nagrani , Manager Product Management, AWS Lambda, AWS Online Tech Talks, January 2017.

R Rule Trigger Action



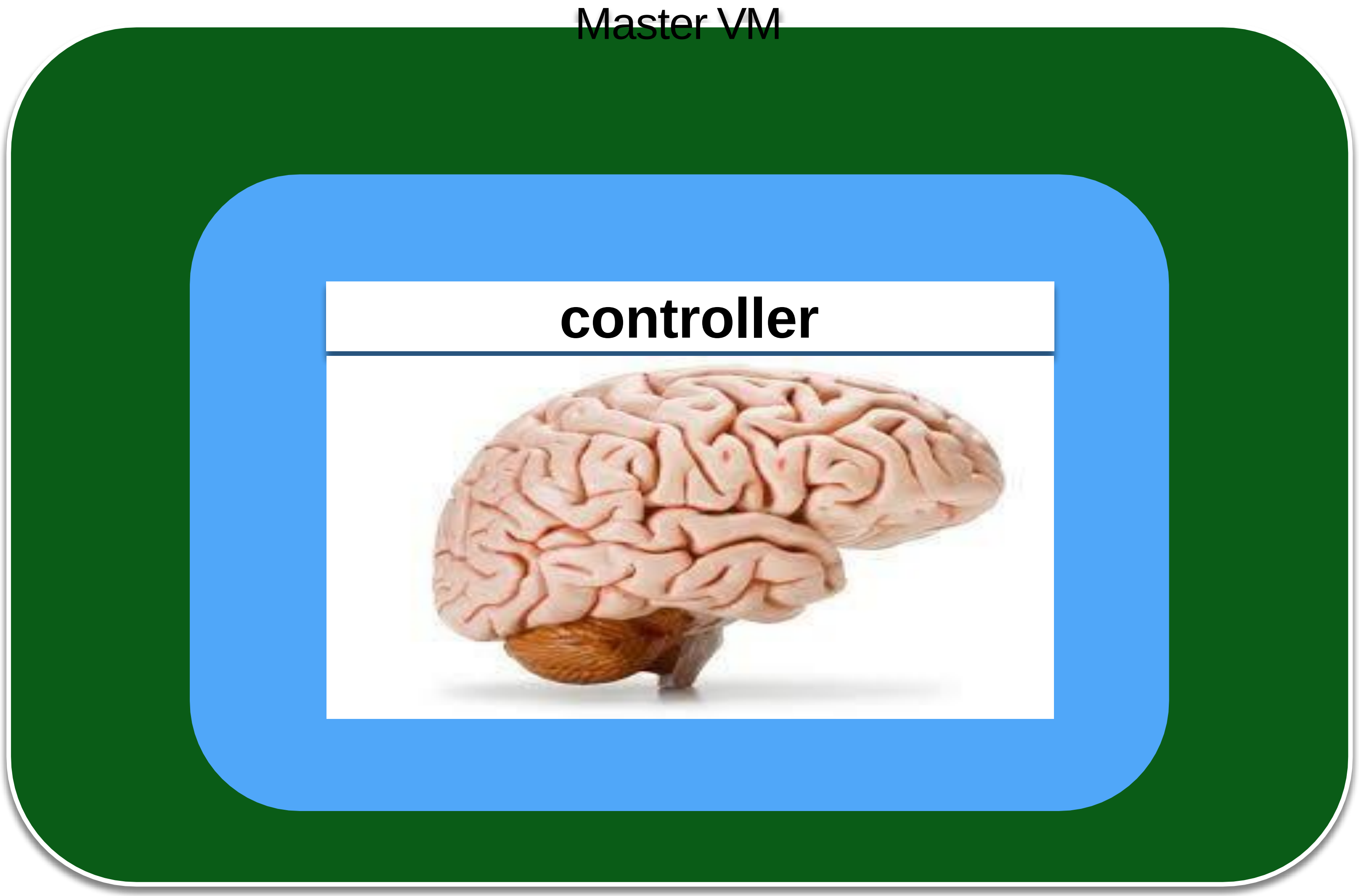
Apache OpenWhisk: Step 1. Entering the system

`POST /api/v1/namespaces/myNamespace/actions/myAction`



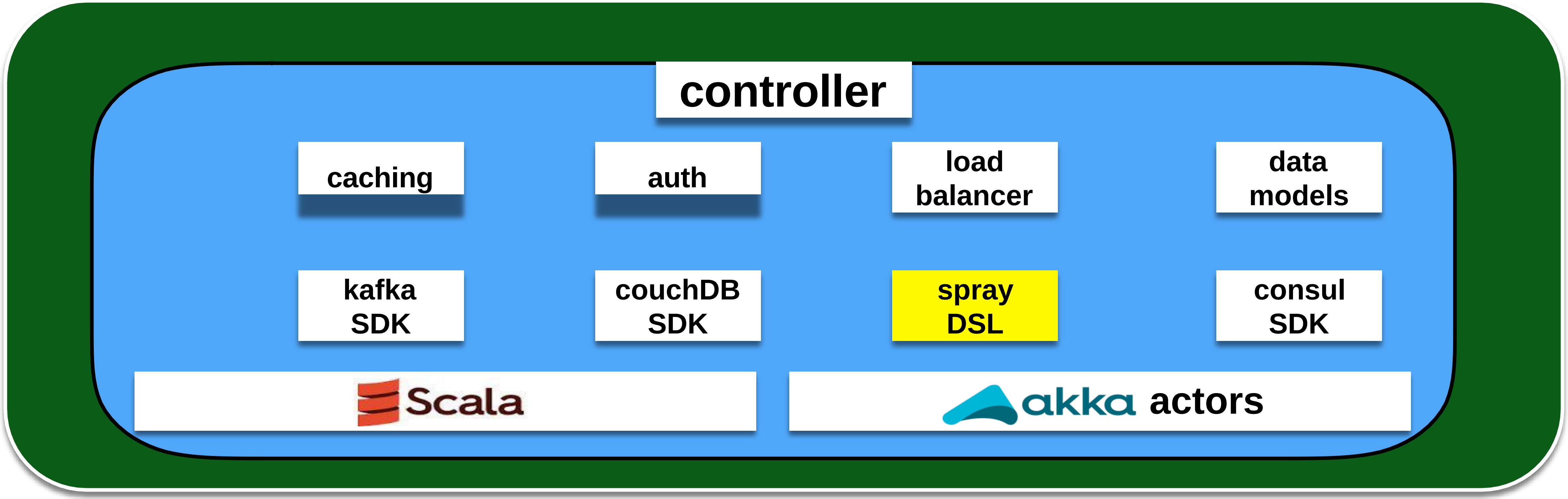
NGINX

Apache OpenWhisk: Step 2. Handle the request



Apache OpenWhisk: Step 2. Handle the request

Master VM



controller

caching

auth

load balancer

data models

kafka SDK

couchDB SDK

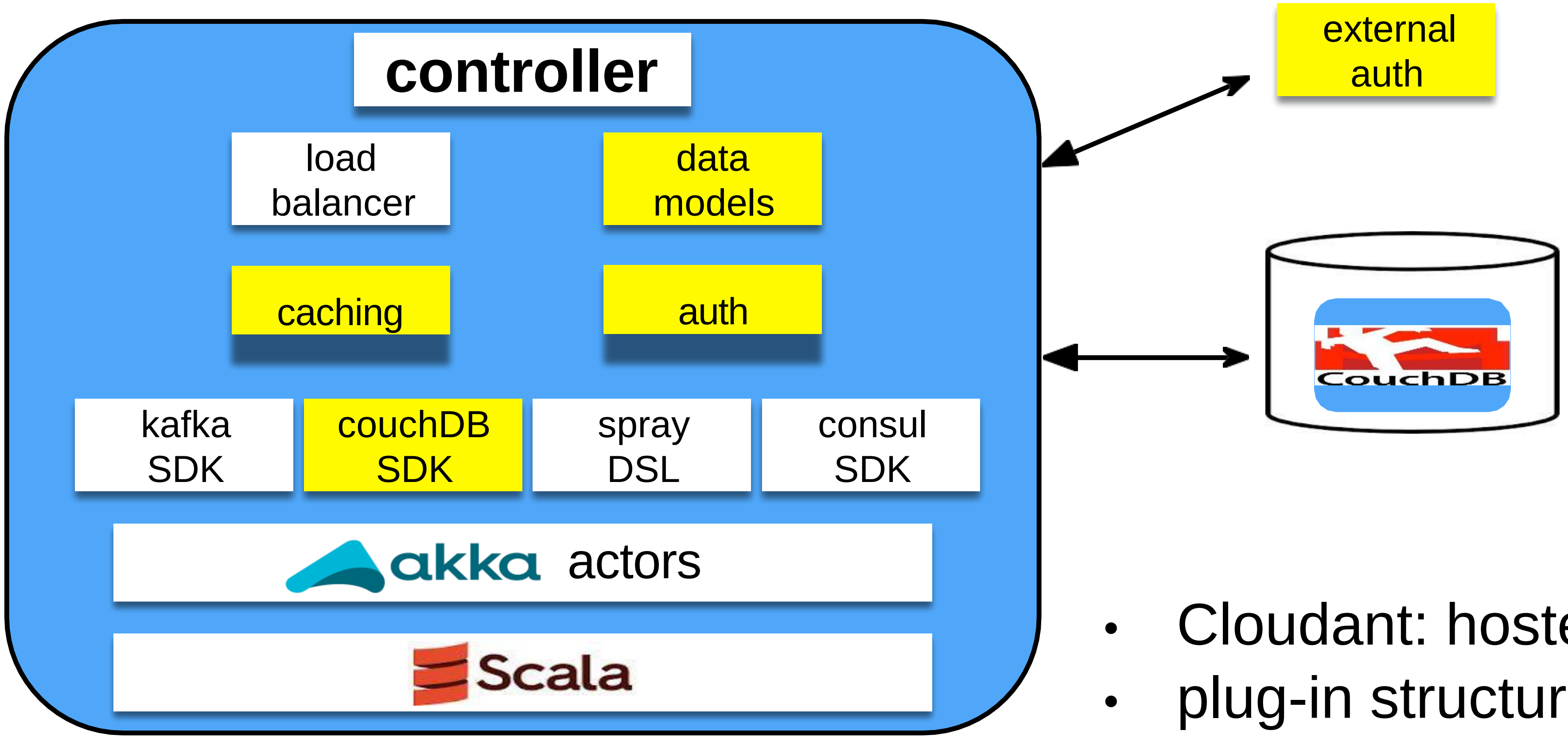
spray DSL

consul SDK

Scala

akka actors

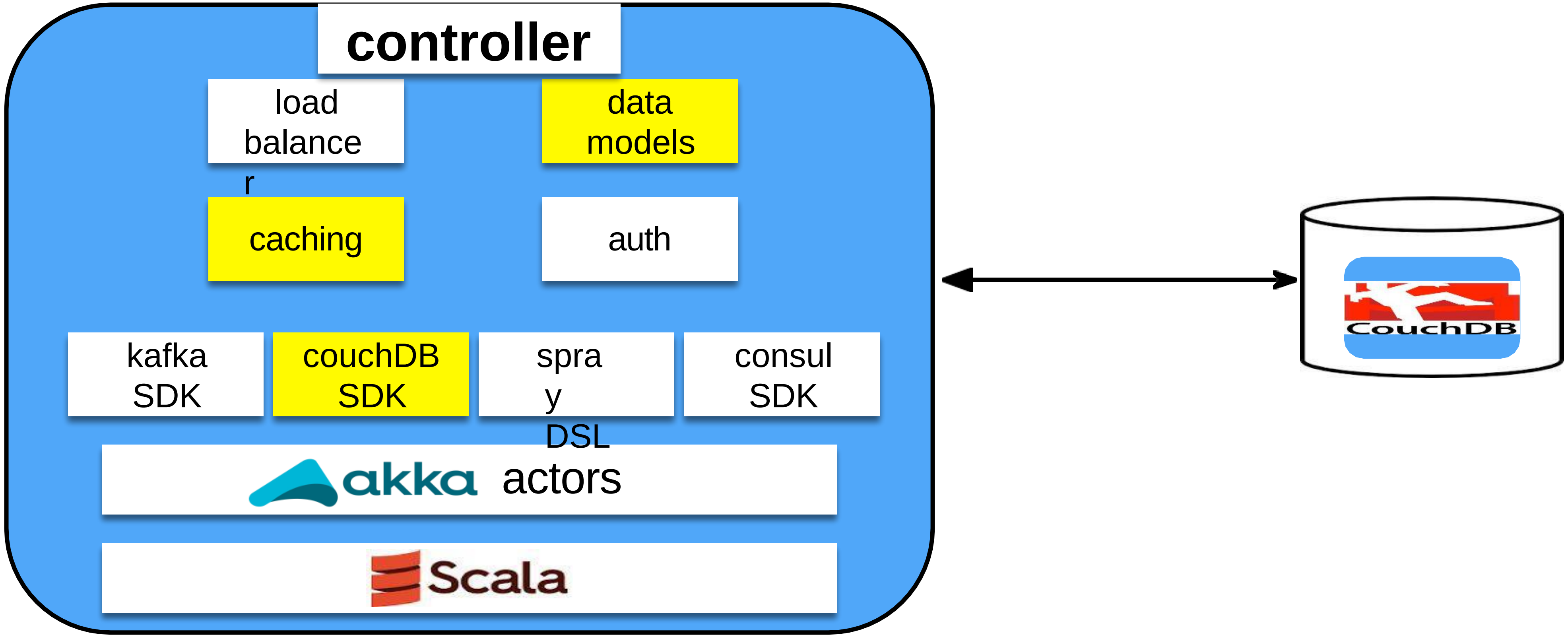
Apache OpenWhisk: Step 3. Authentication + Authorization



- Cloudant: hosted CouchDB
- plug-in structure for custom authentication module

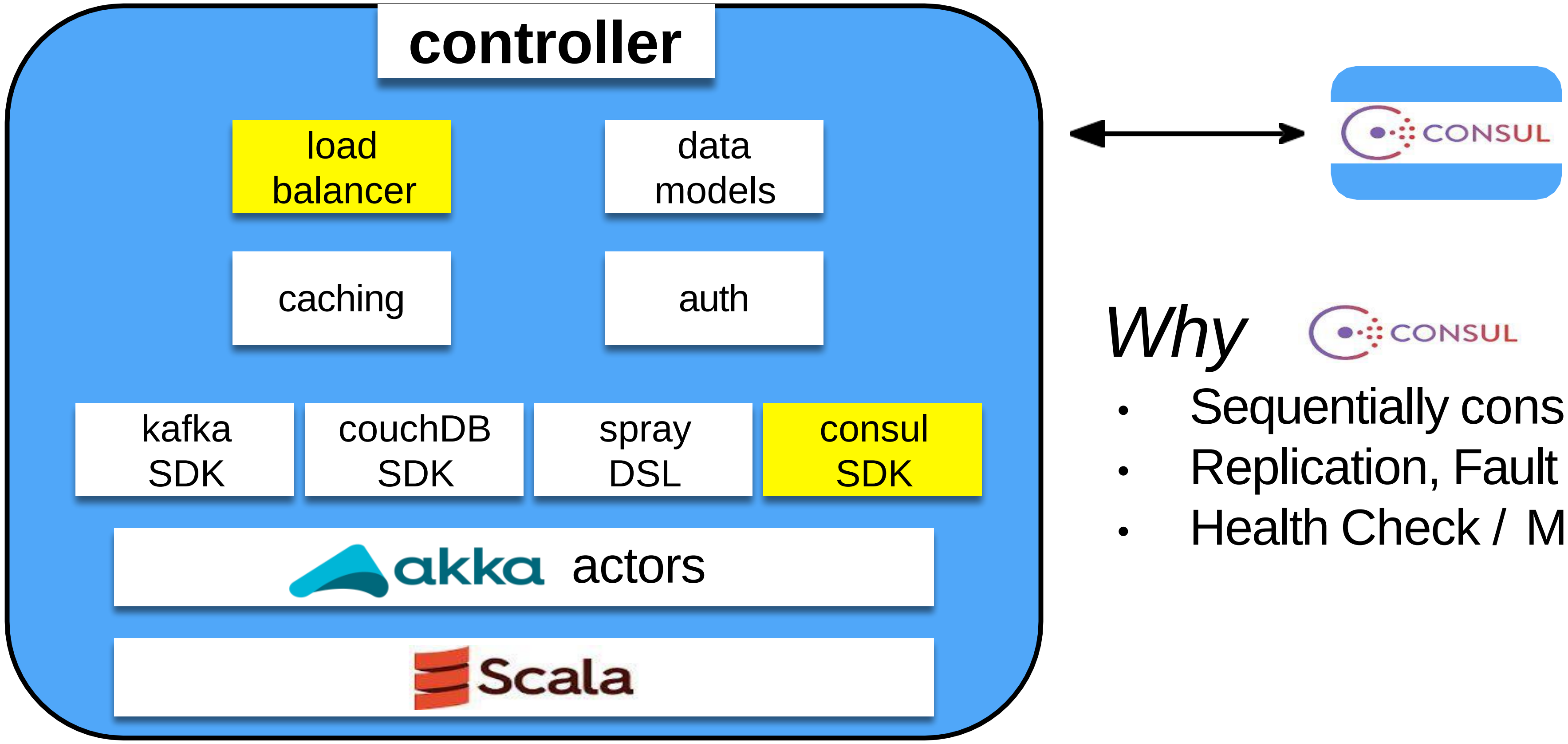
Apache OpenWhisk: Step 4. Get the action

- check resource limits
- actions stored as documents in CouchDB
  - binaries as objects (attachments)



Apache OpenWhisk: Step 5. Looking for a home

Load balancer: find a slave to execute Slave health, loac CONSUL in

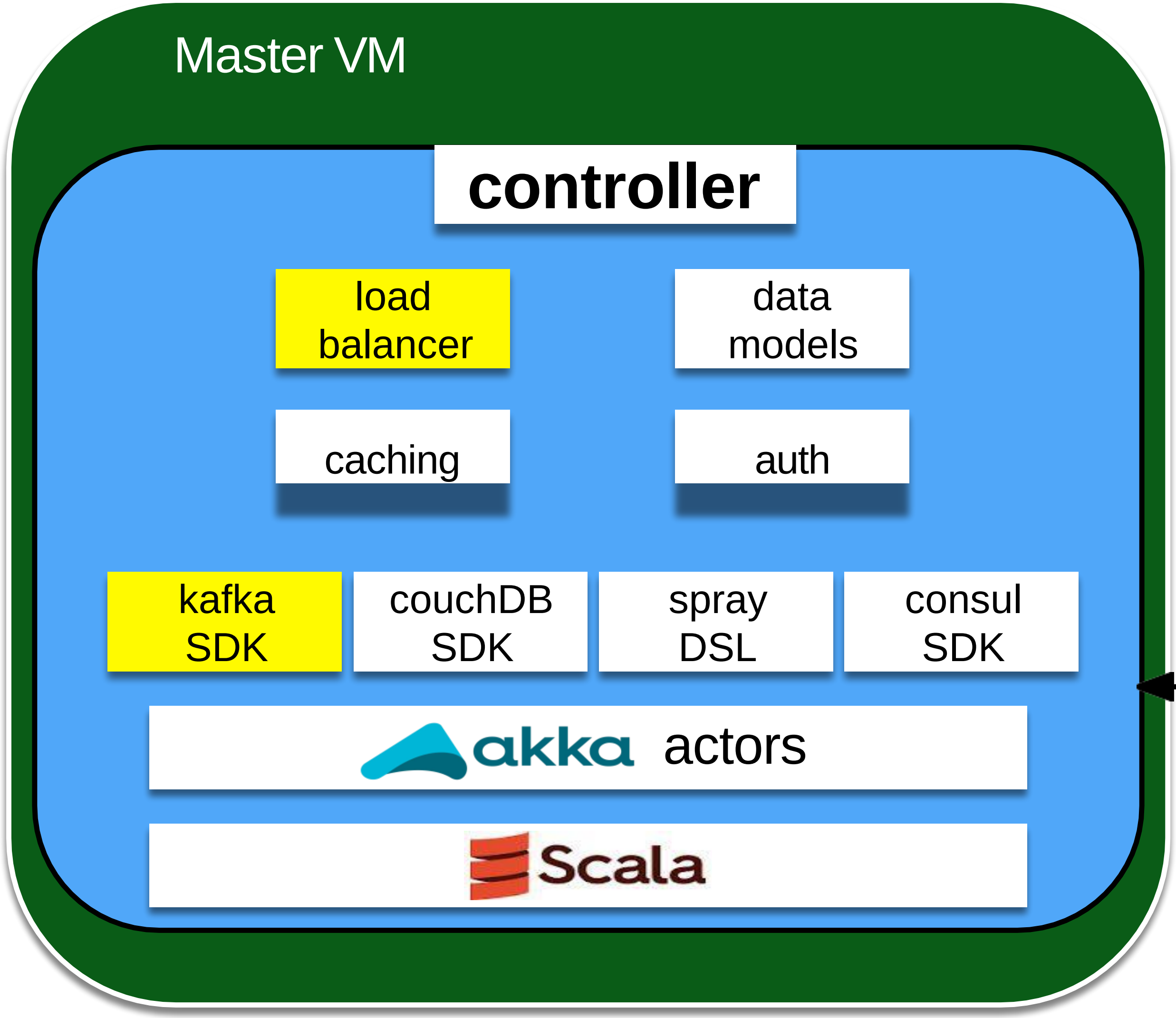


Why CONSUL *?*

- Sequentially consistent KV store
- Replication, Fault Tolerance
- Health Check / Monitoring utilities

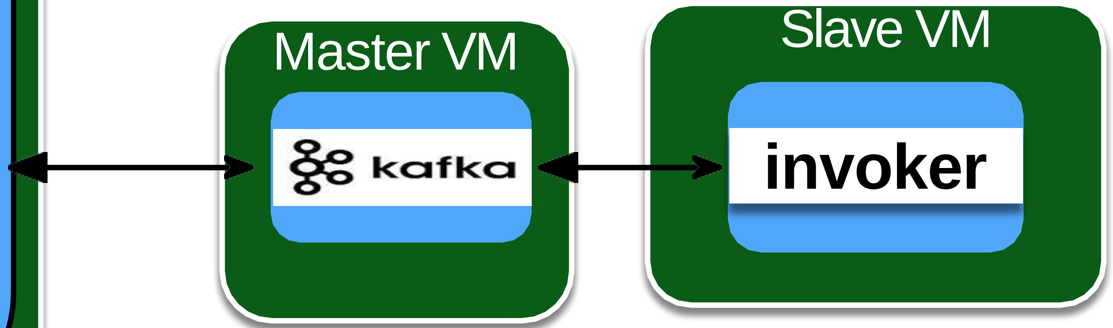
Apache OpenWhisk: Step 6. Get in line!

Post request to execute to queue in kafka

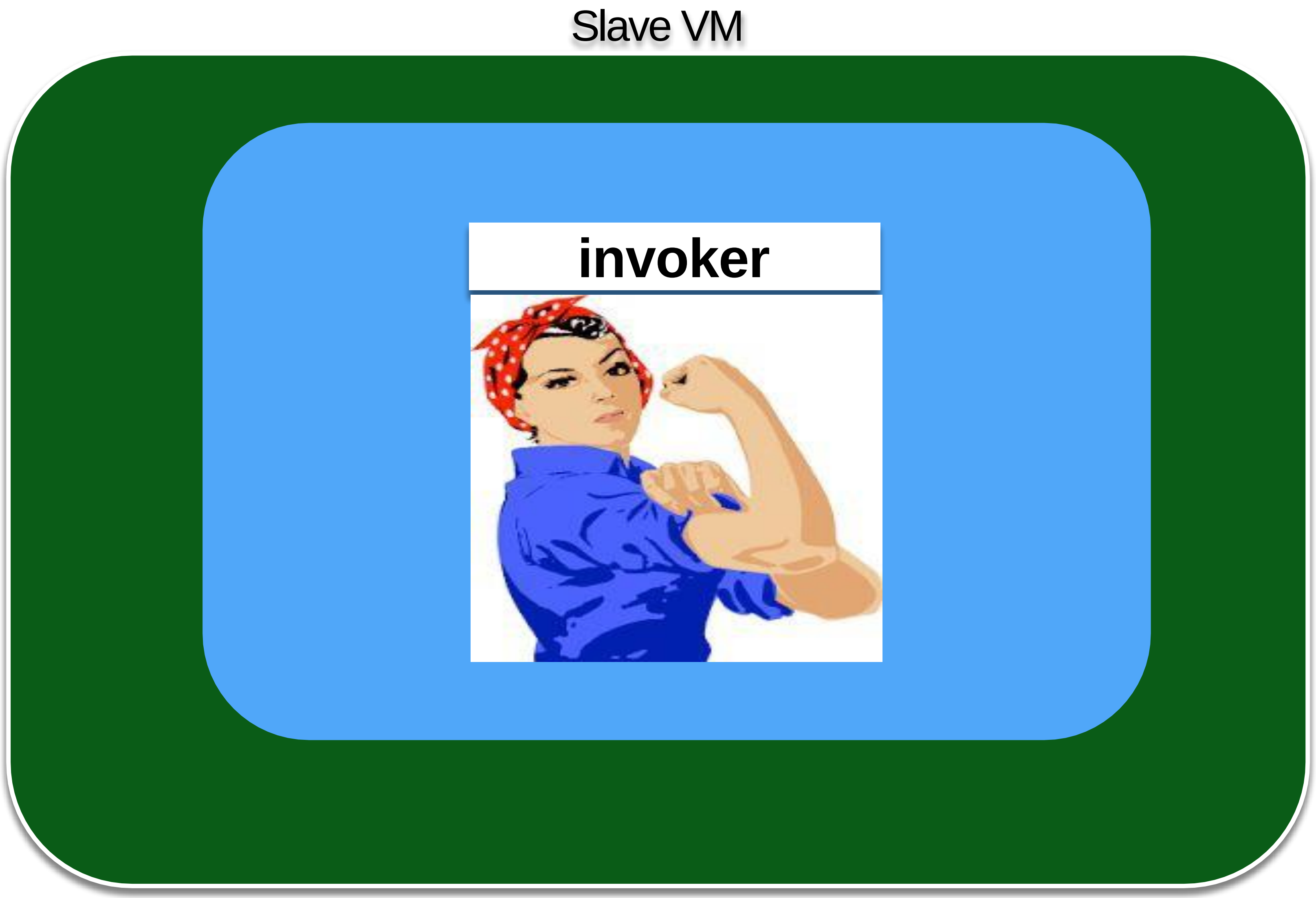


Why kafka ?

- High throughput fault-tolerant queues
- *Point-to-point* messages via topics
  - explicit load balancing



Apache OpenWhisk: Step7. Get to Work!

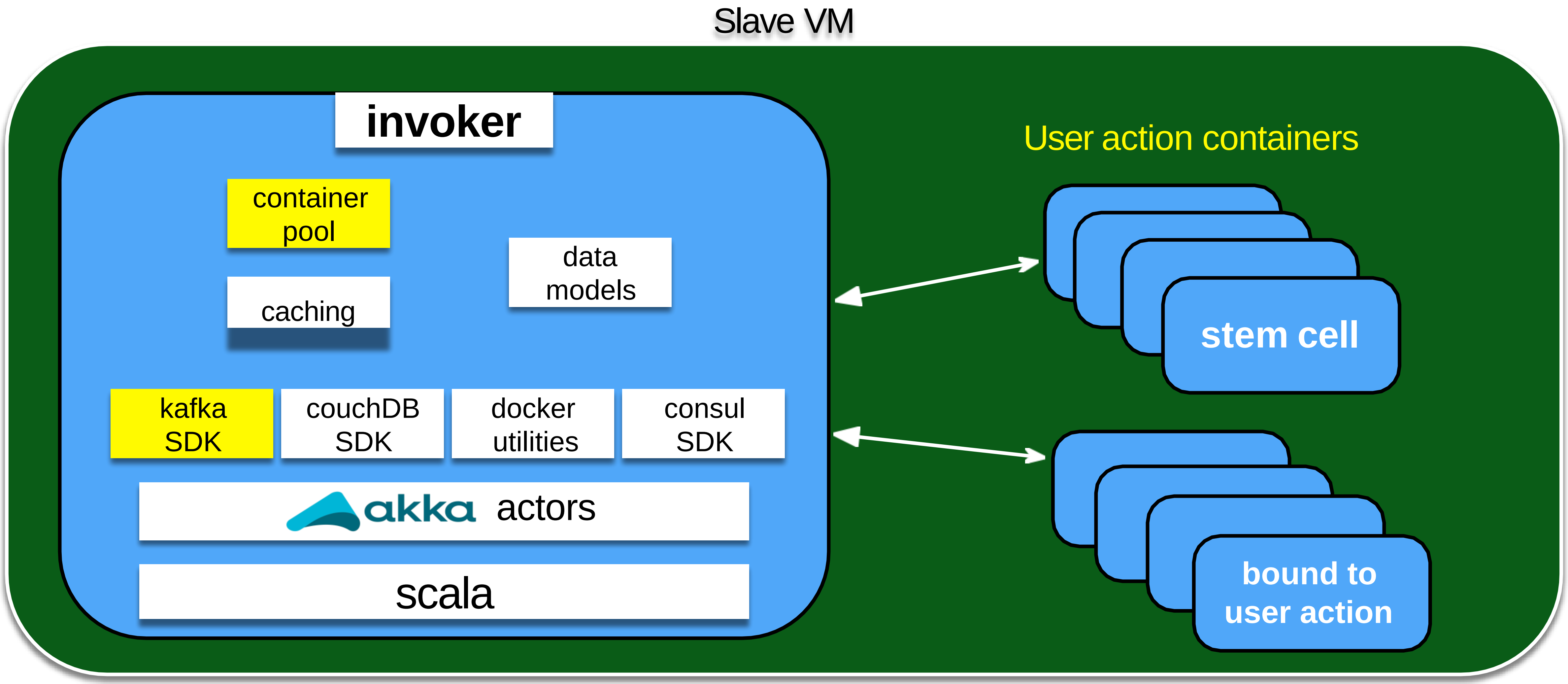


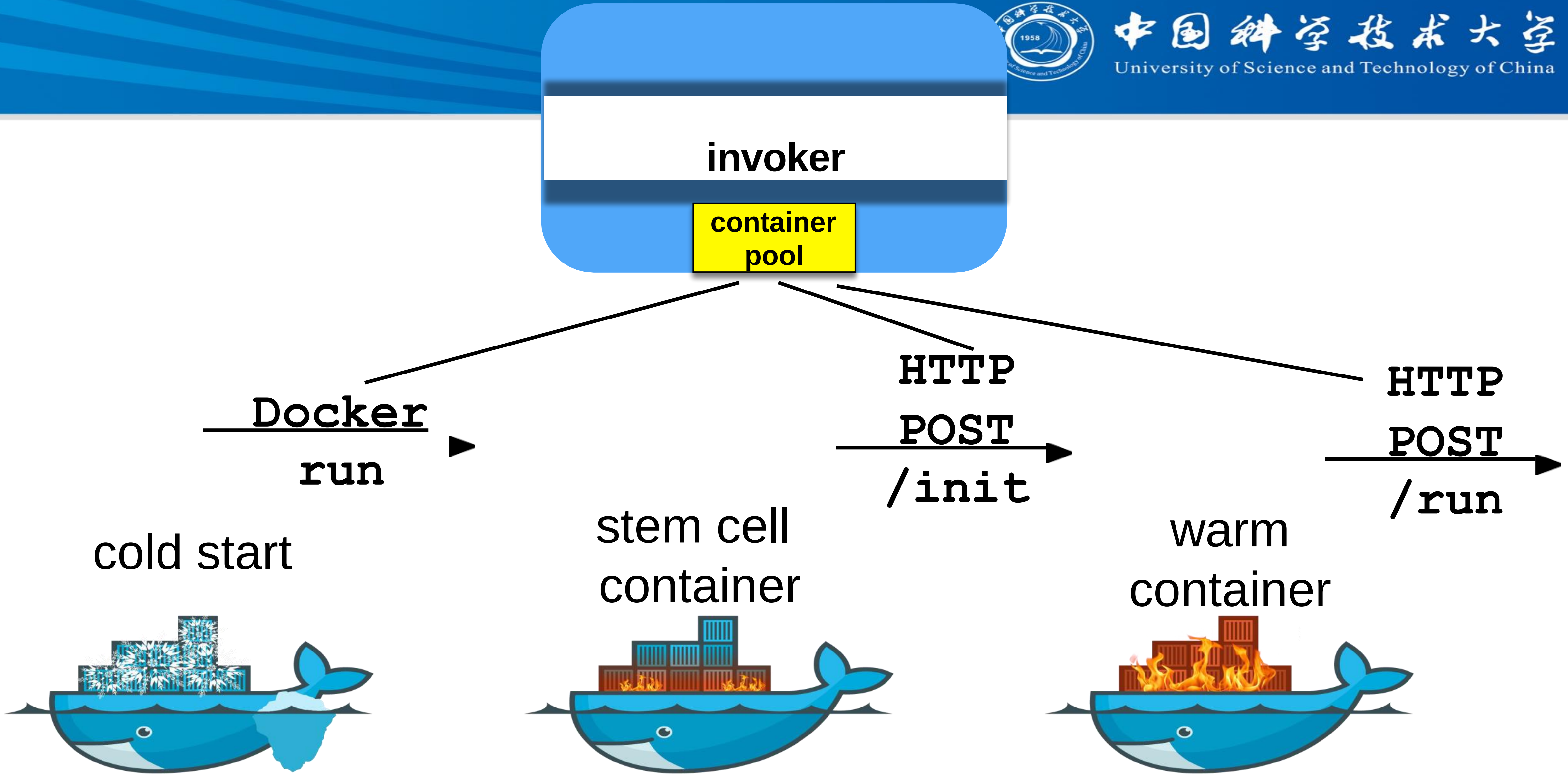
Slave VM

invoker

Apache OpenWhisk: Step 7. Get to work!

- each user action gets it own container (isolation)
- containers may be reused
- container pool allocates and garbage collects containers





invoker

container pool

Docker run

HTTP POST /init

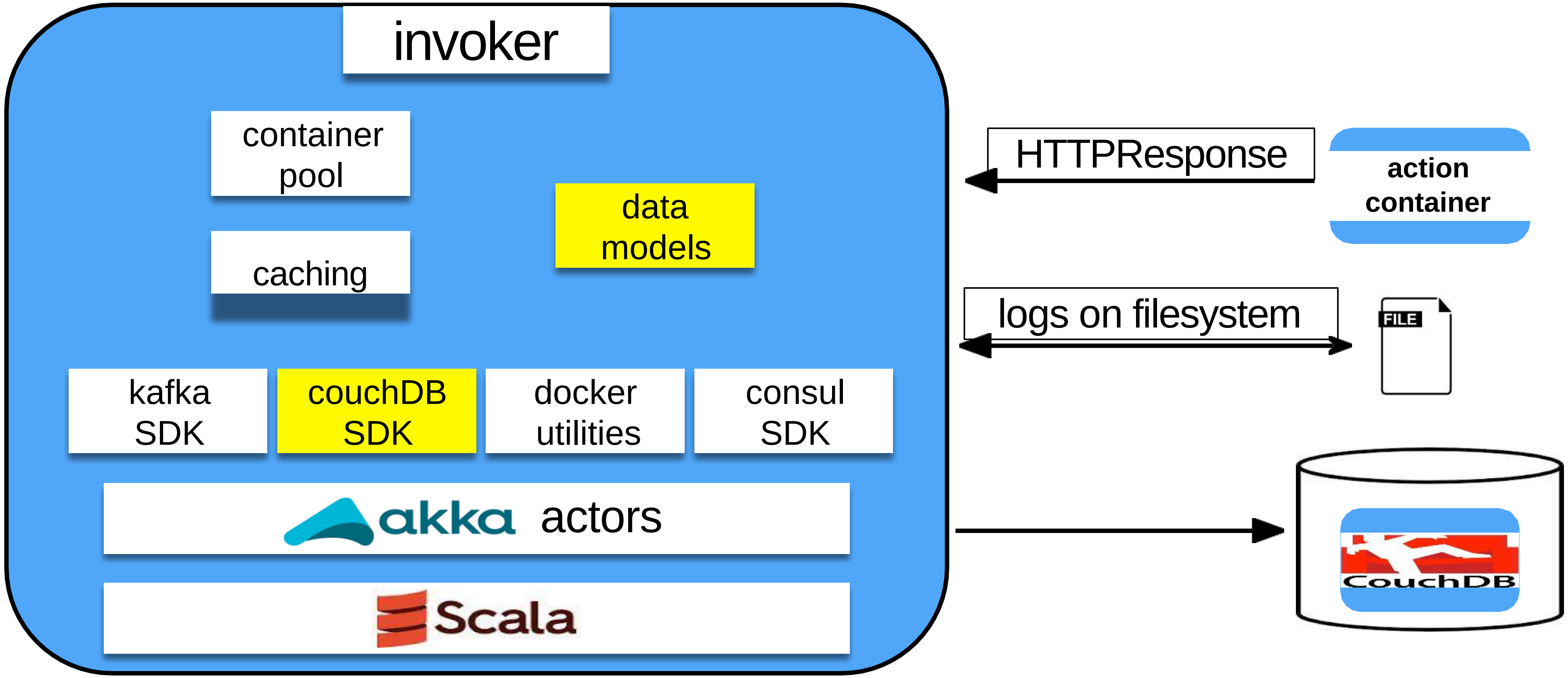
HTTP POST /run

cold start

stem cell container

warm container

Apache OpenWhisk: Step 8. Store the results.



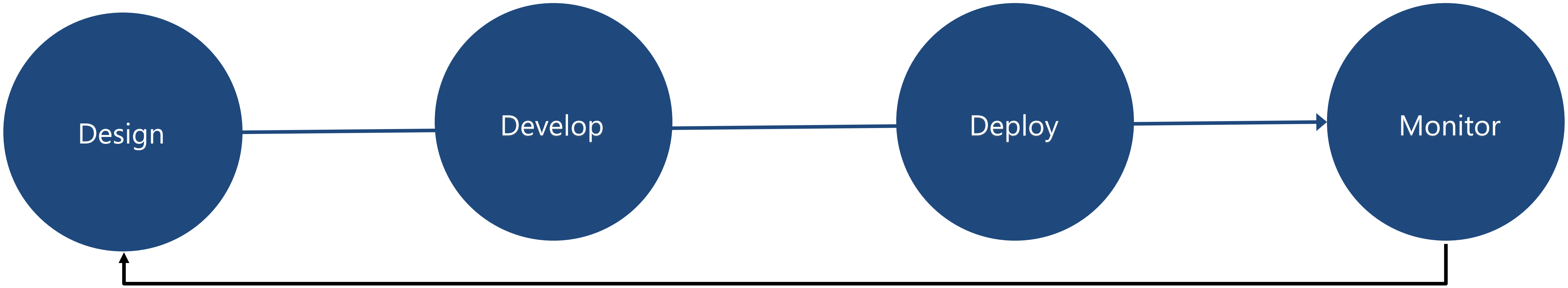
Additional architectural concerns for Serverless for service providers

- Cold start problem
  - Keep invokers ready (“stem cell”) or running (“warm”) after invocation
  - Tradeoff with latency and resource reservation
- Auto scale
  - Add to and remove from the invoker pool
  - Hibernate when idle
- Fine-grained billing
  - Overhead of metering
  - Choice of which resources to bill (CPU, memory, network, …)
  - Understandable billing policy (simple vs detailed)?

Related work

- Reactive programming
- Event-based applications
- Stream processing systems
- Dataflow programming
- Workflows and business processes
- Service composition
- Service oriented architectures
- many more ...

Serverless Apps Lifecycle



Azure Serverless Apps Design

Distributed Architecture

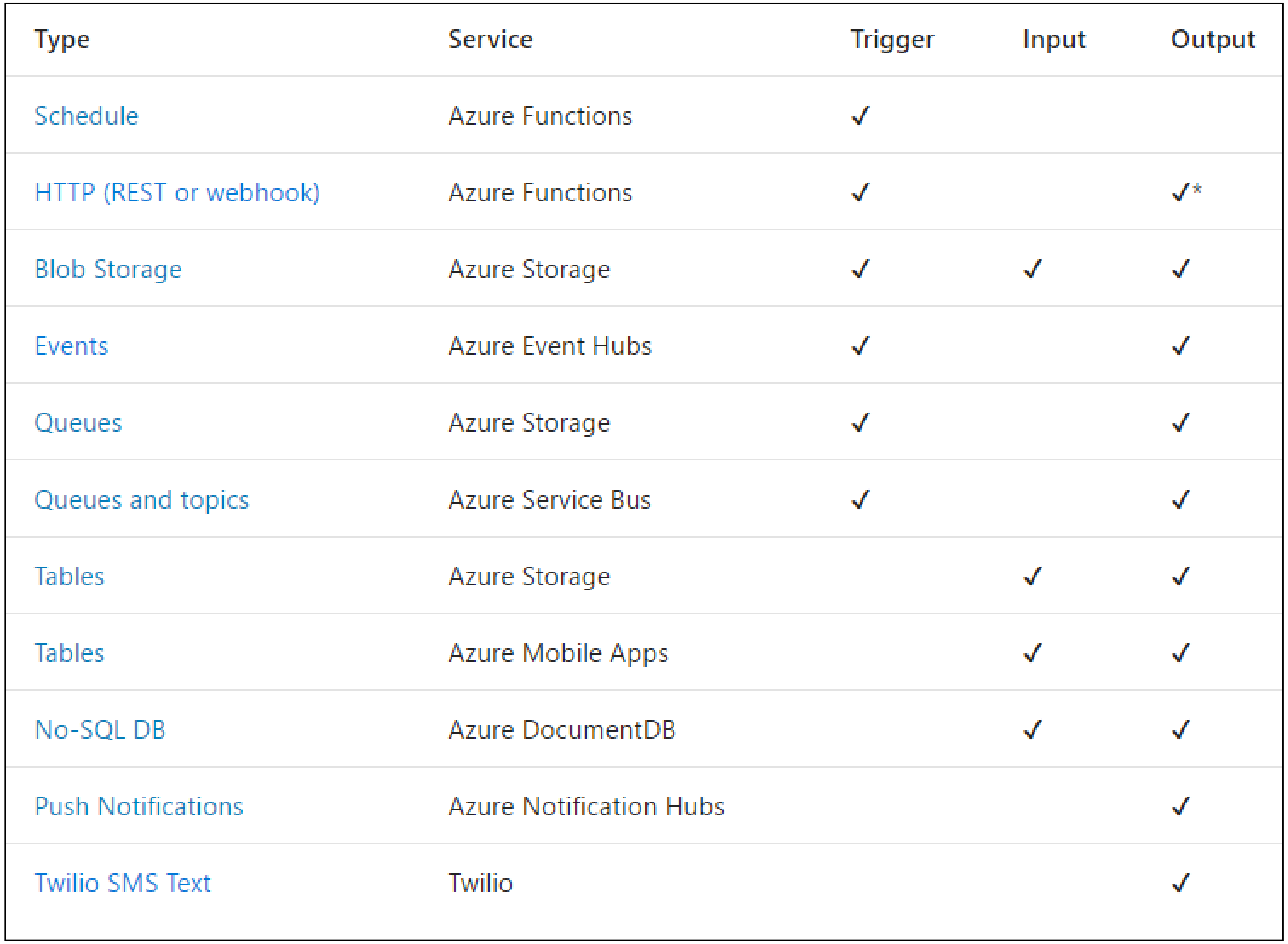
- Design stateless and ASync solutions to enable scaling.
- Connect with other Azure Services via triggers and bindings.
- Use Logic Apps to orchestrate workflows
- Use managed connectors to abstract calls to cloud and on-premises services.

Cloud DevOps

- Design for automation. Use ARM templates.
- Design DevOps for the cloud: safe deployment with test/development and production environment separation and test on the target platform.
- Monitor the running apps with App Insights and tune for best experience.

Triggers and Bindings

Triggers and Bindings



Type	Service	Trigger	Input	Output
Schedule	Azure Functions	✓		
HTTP (REST or webhook)	Azure Functions	✓		✓*
Blob Storage	Azure Storage	✓	✓	✓
Events	Azure Event Hubs	✓		✓
Queues	Azure Storage	✓		✓
Queues and topics	Azure Service Bus	✓		✓
Tables	Azure Storage		✓	✓
Tables	Azure Mobile Apps		✓	✓
No-SQL DB	Azure DocumentDB		✓	✓
Push Notifications	Azure Notification Hubs			✓
Twilio SMS Text	Twilio			✓

Use Bindings in Your Code

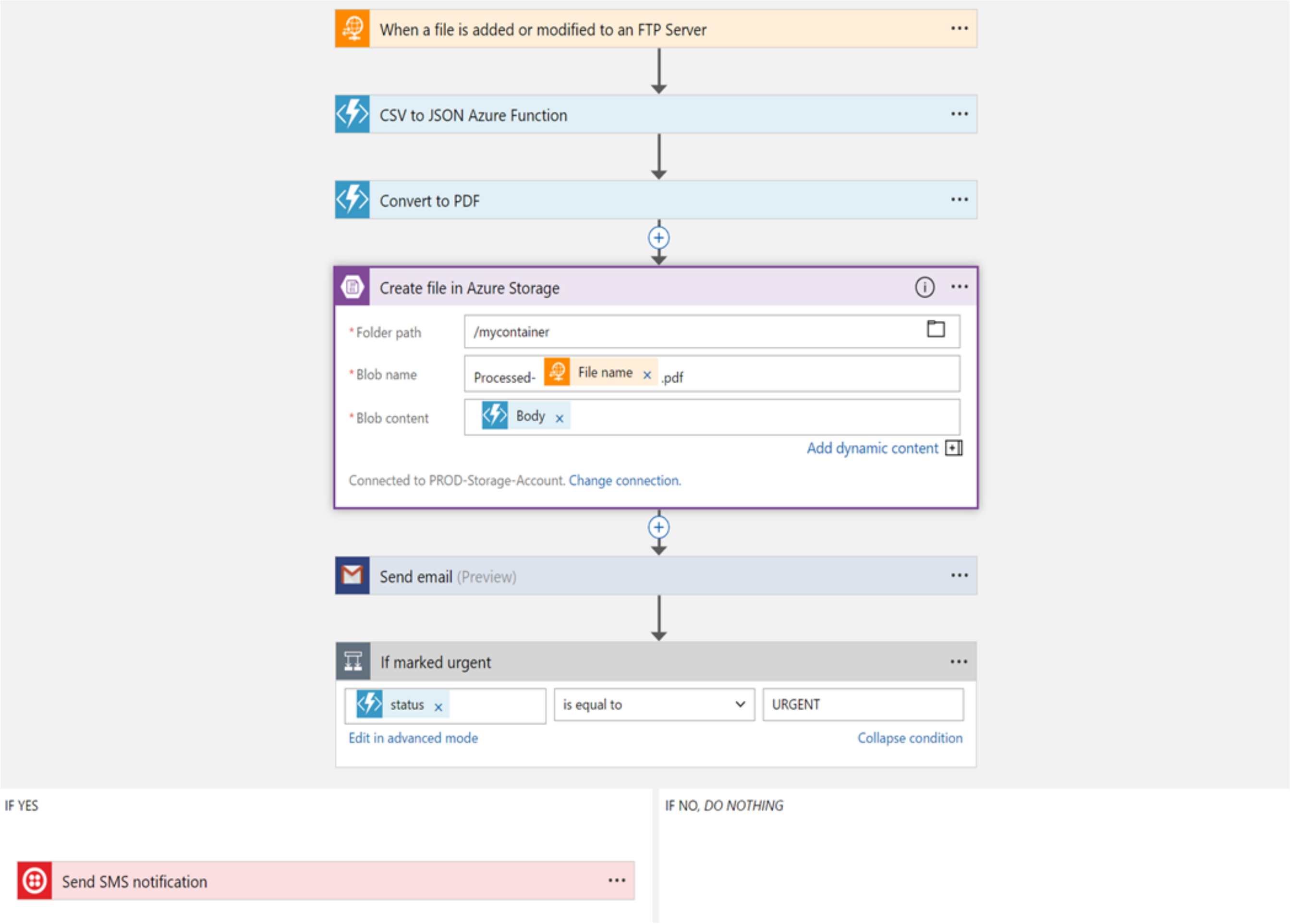
```
public static class OrderHandler
{
    [FunctionName("OrderWebhook")]
    public static async Task<HttpResponseMessage> Run(
        [HttpTrigger] HttpRequestMessage req,
        [Queue("aievents1", Connection = "AiStorageConnection")]
           IAsyncCollector<String> eventOutput,
           TraceWriter log)
    {
        log.Info($"Webhook was triggered!");

        string jsonContent = await req.Content.ReadAsStringAsync();
        dynamic data = JsonConvert.DeserializeObject(jsonContent);

        await eventOutput.AddAsync(
            JsonConvert.SerializeObject(GetLogData(data)));

        int orderId = PlaceOrder(data);

        return req.CreateResponse(HttpStatusCode.OK,
                                           new {orderNumber = orderId });
    }

 . . .
}
```

function.json

```
"bindings": [
    {
      "type": "httpTrigger",
      "direction": "in",
      "webHookType": "genericJson",
      "name": "req"
    },
    {
      "type": "http",
      "direction": "out",
      "name": "res"
    },
    {
      "type": "queue",
      "name": "eventOutput",
      "queueName": "aievents1",
      "connection":"AiStorageConnection",
      "direction": "out"
    }
  ]
```

Logic Apps Workflow Designer

- Workflow in the cloud
- Powerful control flow
- Connect functions and APIs
- Declarative definition to persist in source control and drive deployments



When a file is added or modified to an FTP Server

CSV to JSON Azure Function

Convert to PDF

Create file in Azure Storage

*Folder path: /mycontainer

*Blob name: Processed- File name × .pdf

*Blob content: Body ×

Add dynamic content

Connected to PROD-Storage-Account. Change connection.

Send email (Preview)

If marked urgent

status × | is equal to | URGENT

Edit in advanced mode

Collapse condition

IF YES

Send SMS notification

IF NO, DO NOTHING

Logic Apps

Cloud APIs and platform

- Supports over 125 built-in connectors
- Scales to meet your needs
- Enables rapid development
- Extends with custom APIs and Functions

API connections

- Authenticate once and reuse

SaaS

- appFigures
- Asana
- Azure API Management
- Azure App Services
- Azure Automation
- Azure Cognitive Face API
- Azure Cognitive LUIS
- Azure Cognitive Text Analytics
- Azure Cognitive Vision
- Azure Data Lake Store
- Azure Document DB
- Azure Event Hub
- Azure Functions
- Azure Machine Learning
- Azure Resource Manager
- Azure Service Bus
- Azure SQL
- Azure Storage Blob
- Azure Storage Queues
- Basecamp
- Bing Search
- BitBucket
- Bitly
- Blogger
- Box
- Buffer
- Campfire
- Chatter
- Common Data Service
- Disqus
- DocuSign
- Dropbox
- Dynamics AX Online
- Dynamics CRM Online
- Dynamics CRM Service Bus
- Dynamics Financials
- Dynamics Operations
- Easy Redmine
- Eventbrite
- Facebook
- FreshBooks
- Freshdesk
- GitHub
- Gmail
- Google Calendar
- Google Contacts
- Google Drive
- Google Sheets
- Google Tasks
- GoTo Meeting
- GoTo Training
- GoTo Webinar
- Harvest
- HelloSign
- Infusionsoft
- JIRA
- Insightly
- Instagram
- Instapaper
- MailChimp
- Mandrill
- Medium
- Microsoft Project Online
- Microsoft Translator
- MSN Weather
- Muhimbi PDF
- Office 365
- Office 365 Users
- Office 365 Video
- OneDrive
- OneDrive for Business
- OneNote
- Outlook.com
- Outlook Tasks
- PagerDuty
- Pinterest
- Pipedrive
- Pivotal Tracker
- Power BI
- Project Online
- Redmine
- Salesforce
- Salesforce Chatter
- SendGrid
- SharePoint Online
- Slack
- SmartSheet
- SparkPost
- Stripe
- Survey Monkey
- Todoist
- Toodledo
- Trello
- Twilio
- Twitter
- Typeform
- UserVoice
- VS Team Services
- Webmerge
- Wordpress
- Wunderlist
- Yammer
- YouTube
- Zendesk

Protocols/Native

- HTTP, HTTPS
- HTTP Webhook
- FTP, SFTP
- SMTP
- RSS
- Compose, Query, Parse JSON
- Wait
- Terminate
- Workflow

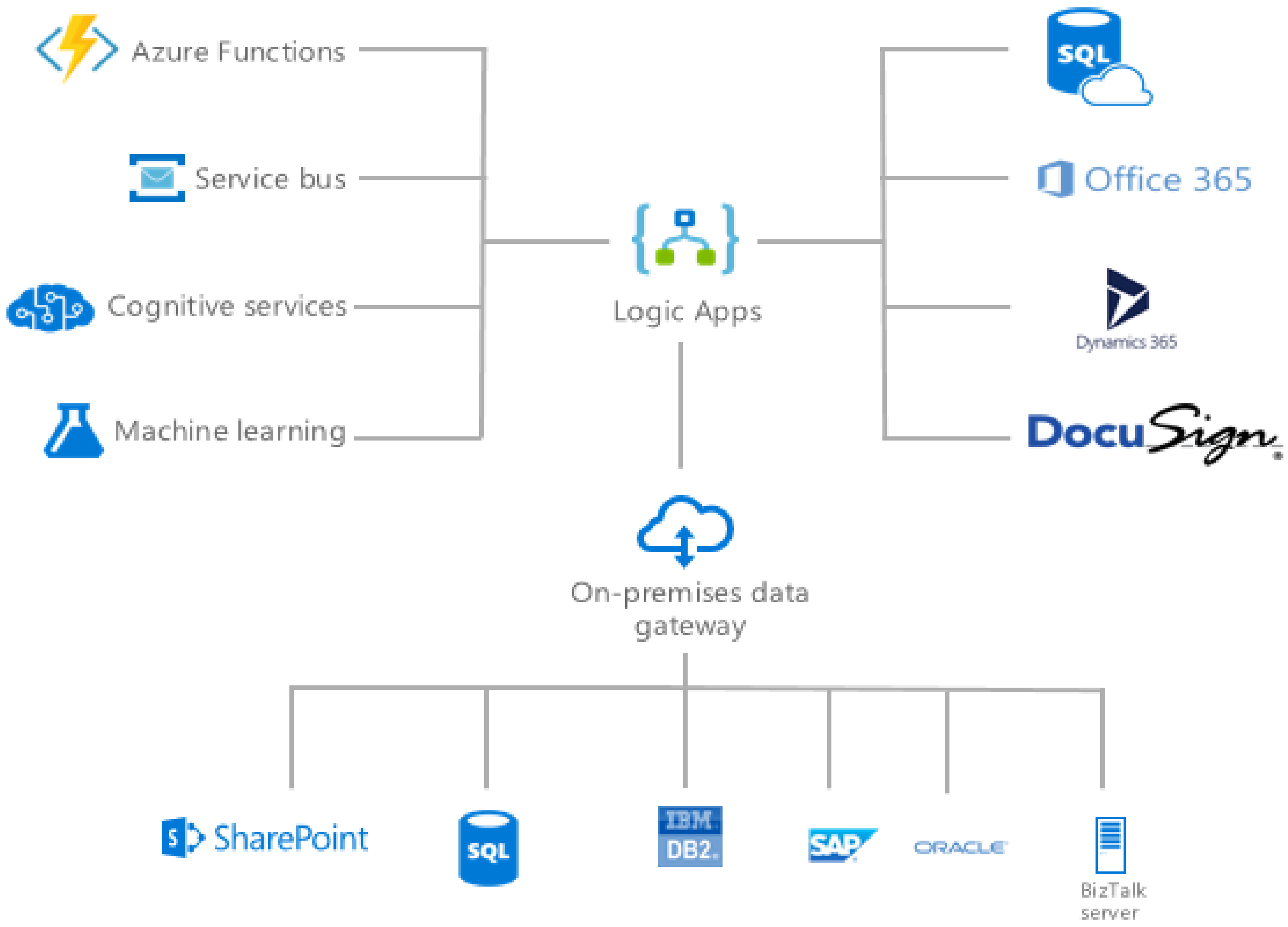
XML and EDI

- XML Validation
- Transform XML (+Mapper)
- Flat File Encode
- Flat File Decode
- X12
- EDIFACT
- AS2
- Integration Account Artifact Lookup

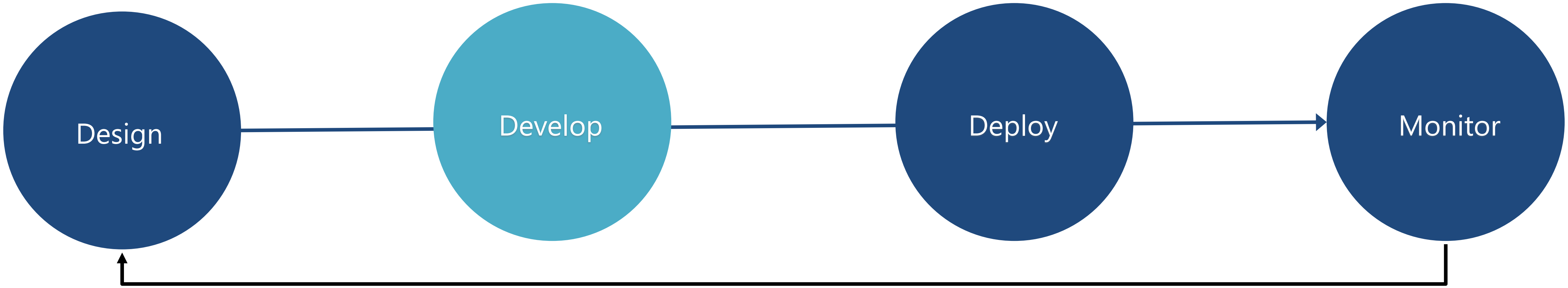
Hybrid

- BizTalk Server
- File System
- IBM DB2
- Informix
- Oracle DB
- SharePoint Server
- SQL Server
- SAP
- Websphere MQ

Logic Apps Connect Everything



Develop

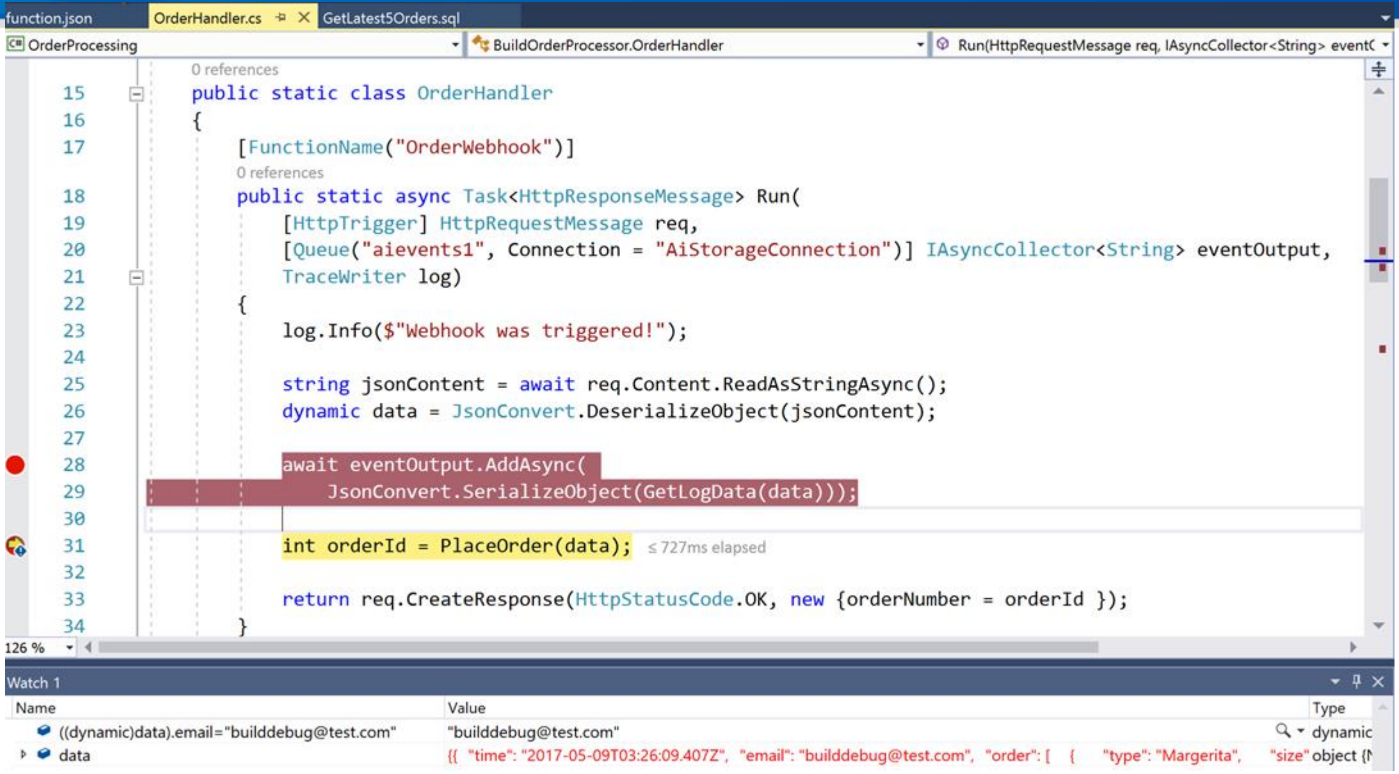


Local Development Tooling Options

- Azure Functions Core Tools
  - Provides the entire Functions runtime
  - Trigger off of Azure events and debug locally
- JavaScript
  - Use Visual Studio Code or any Node debugger
- C#
  - Use Visual Studio 2015 or 2017
  - Use class libraries with attributes in Visual Studio 2017

C# and Visual Studio

- Based on class libraries
- Get the full power of IntelliSense, unit testing, and local debugging
- Use Web Jobs attributes to define triggers and bindings



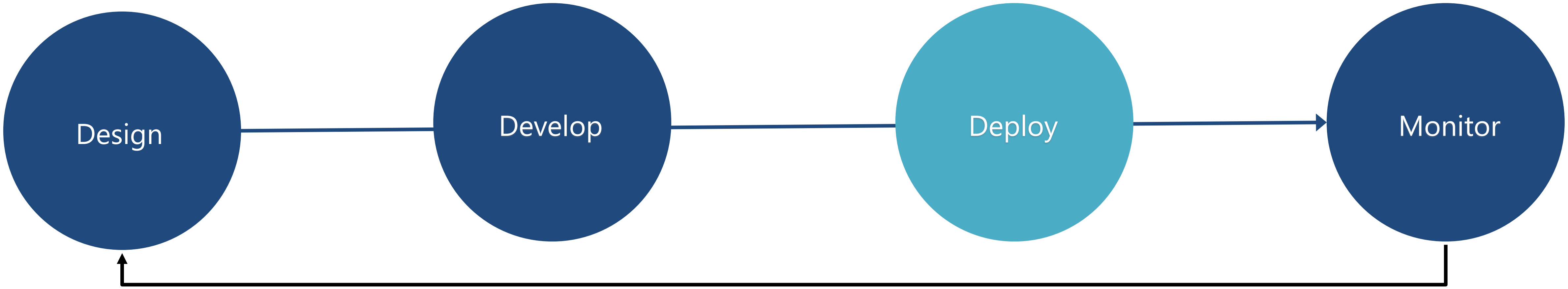
function.json
OrderHandler.cs
GetLatest5Orders.sql
OrderProcessing
BuildOrderProcessor.OrderHandler
Run(HttpRequestMessage req, IAsyncCollector<String> event(
126 %
Watch 1

```
0 references
public static class OrderHandler
{
    [FunctionName("OrderWebhook")]
    0 references
    public static async Task<HttpResponseMessage> Run(
        [HttpTrigger] HttpRequestMessage req,
        [Queue("aievents1", Connection = "AiStorageConnection")] IAsyncCollector<String> eventOutput,
        TraceWriter log)
    {
        log.Info($"Webhook was triggered!");

        string jsonContent = await req.Content.ReadAsStringAsync();
        dynamic data = JsonConvert.DeserializeObject(jsonContent);

        await eventOutput.AddAsync(
            JsonConvert.SerializeObject(GetLogData(data)));

        int orderId = PlaceOrder(data);  ≤ 727ms elapsed

        return req.CreateResponse(HttpStatusCode.OK, new {orderNumber = orderId });
    }
```

Name	Value	Type
((dynamic)data).email="builddebug@test.com"	"builddebug@test.com"	dynamic
data	{{ "time": "2017-05-09T03:26:09.407Z", "email": "builddebug@test.com", "order": [{ "type": "Margerita", "size"	object {N

Learn more at https://aka.ms/vs2017functiontools

Deploy



Deployment Options

Resource deployment

- Azure Resource Manager (i.e. ARM)

Content deployment

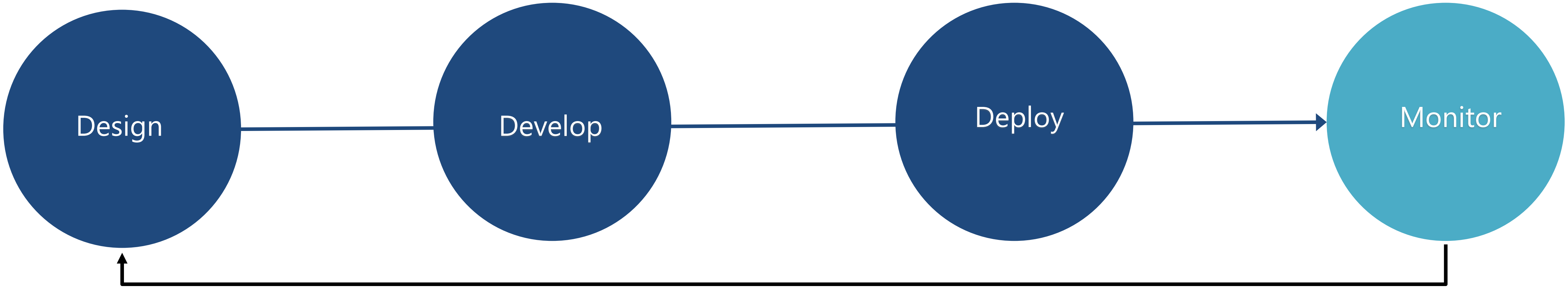
- Visual Studio
- Azure CLI (Logic App)
- Azure Functions Core Tools (Function App)
- CI/CD

https://www.visualstudio.com/en-us/docs/build/get-started/aspnet-4-ci-cd-azure-automatic

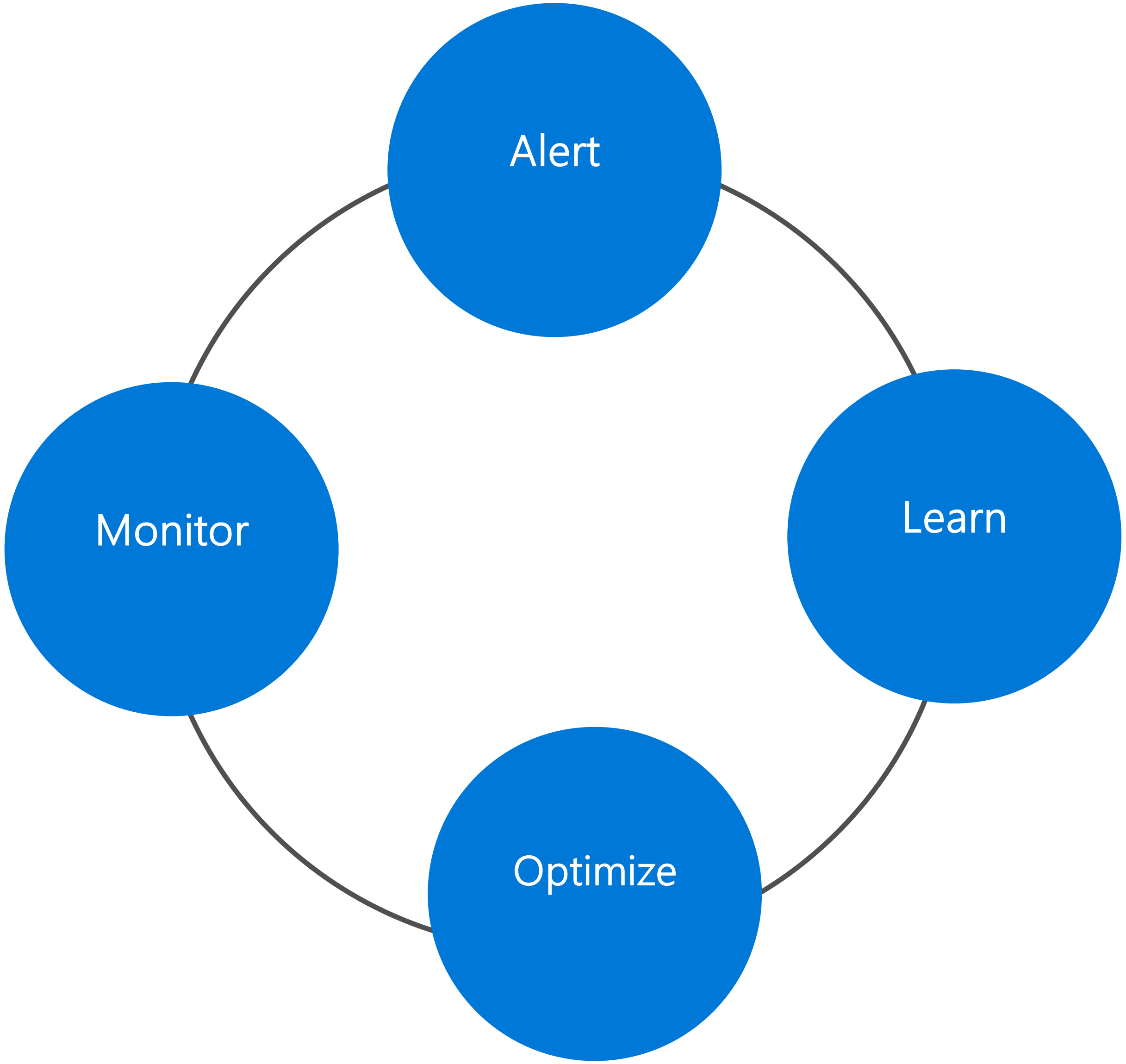
Safe deployment practices

- Use Azure Functions deployment slots for environment separation and swap deployments

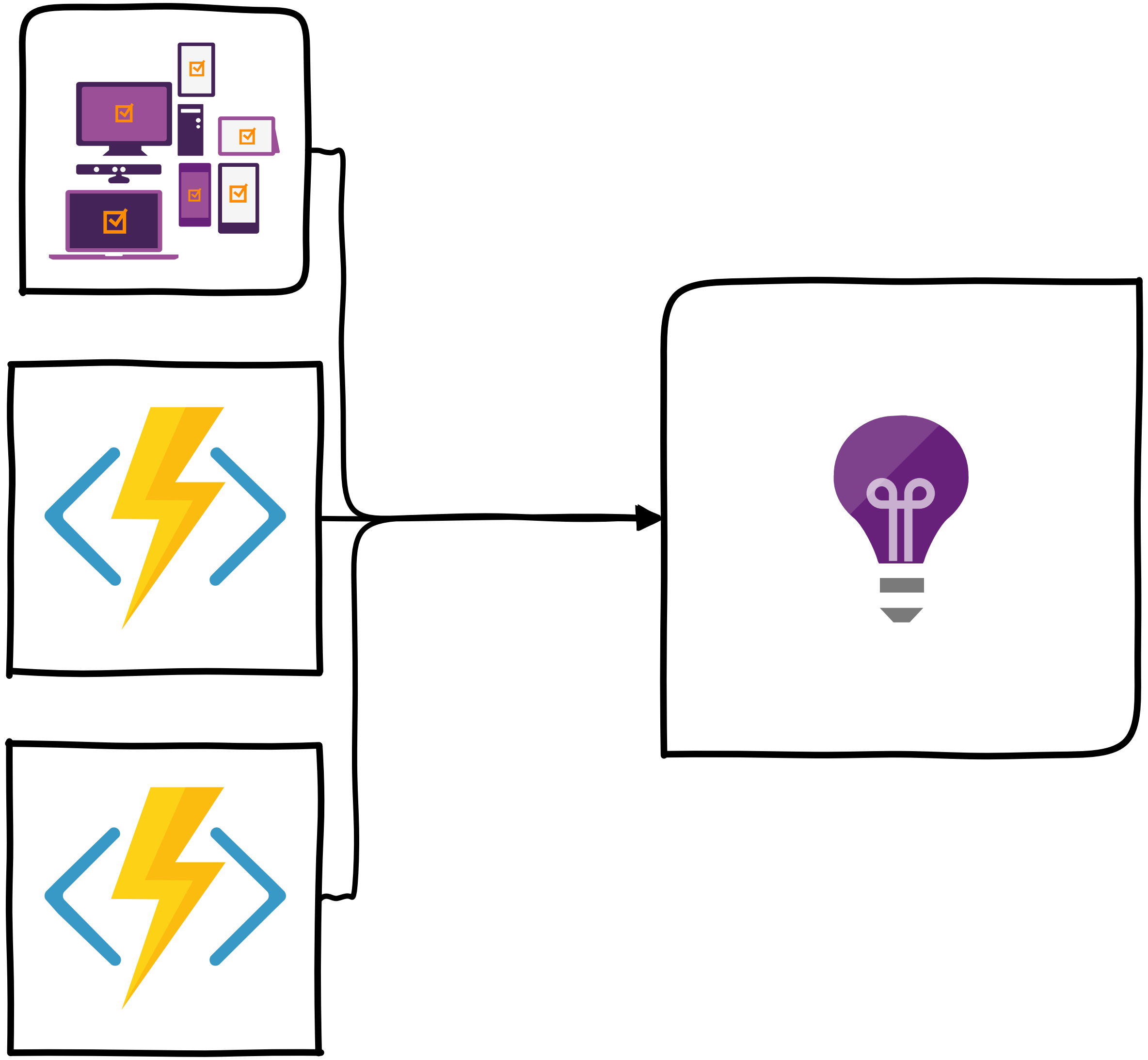
Monitor



Key Scenarios for Monitoring



- Extensible Application Performance Management (APM)
- Rich data: Metrics, Traces, Exception tracking, Dependencies, Page Views, User data, custom events
- Easy to use graph/alerts, powerful analytics portal, integration with PowerBI and other analytics services



Azure Functions Runtime

Developer experience

- Same consistent Programming Model
- Same Azure Functions portal
- Publish directly from Visual Studio tooling
- Leverage triggers: timer trigger and new SQL Service Broker trigger

Administrator features

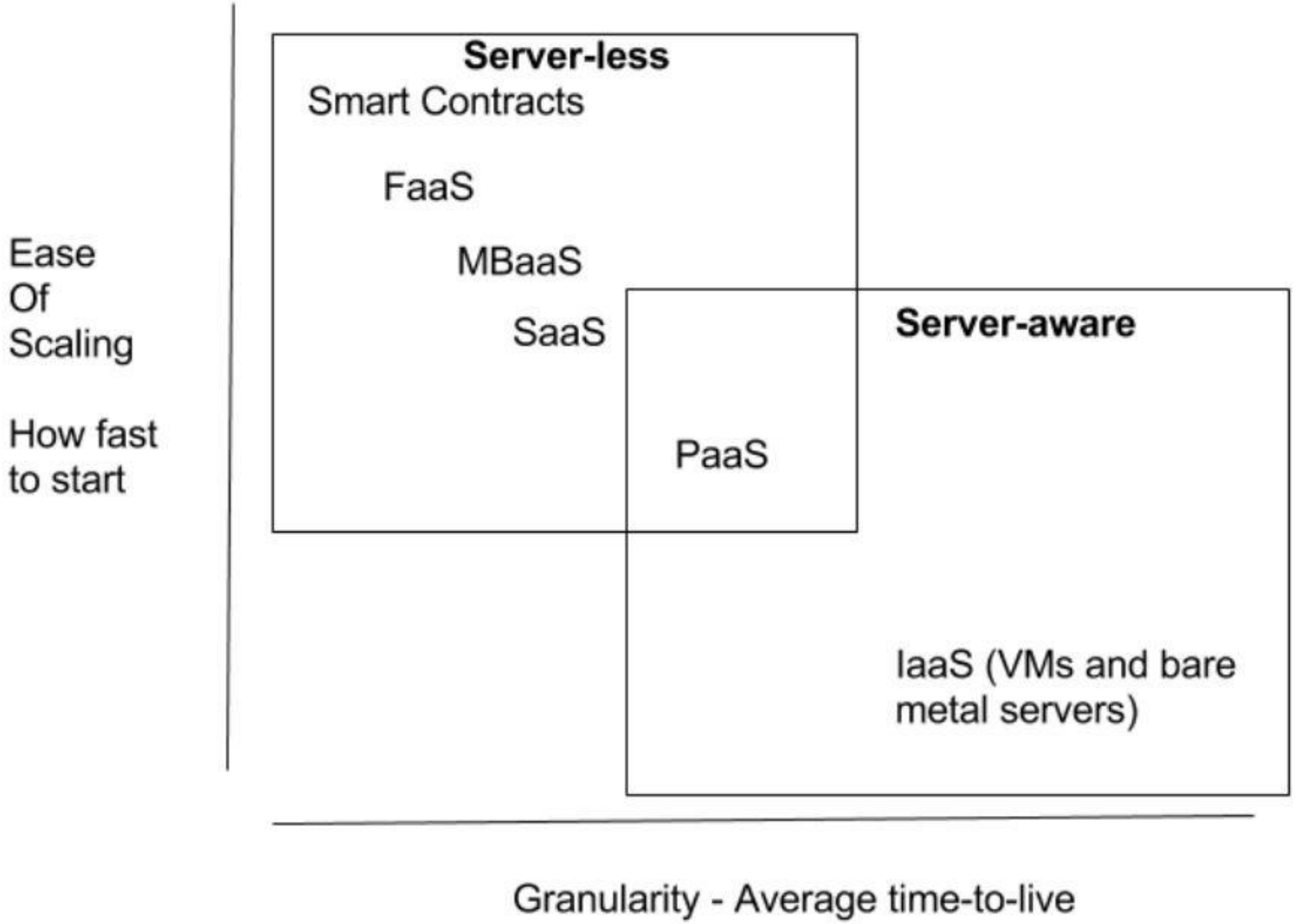
- Take advantage of Azure Functions on premises
  - Workers can run in spare compute – i.e. desktops left on overnight within orgs
- Only provision two types of roles
  - Management Role – Hosts Portal, Publishing Endpoint and
  - Worker Role – Runs Function in Windows Server Containers

Future of Serverless: Research Challenges and Questions

Serverless as next step in Cloud Computing?

- Cost - pay-as-you-go is enough?
- Server-less - can servers be really hidden?
- Problem of state: stateless, state in other place, or state-ful supported in FaaS?
- Security - no servers!
- Legacy systems and serverless?
  - Hybrid model?

Cloud computing: server-less vs server-aware?



Programming model(s) for Serverless?

- Tools
- Deployment
- Monitoring and debugging
  - Short-lived functions, scaling to large invocations,
  - Looking for problems is like finding needles in ever growing haystack?
- Serverless IDEs?
- Decompose micro-service into FaaS?
  - Code granularity is function?
- Managing state inside and outside FaaS
- Concurrency, recovery semantics, transactions?

Open Problems - how FaaS fits into cloud?

- Just another *aaS?
- Can different cloud computing service models be mixed?
- Can there be more choices for how much memory and CPU can be used by serverless functions?
- Does serverless need to have IaaS-like based pricing?
- What about spot and dynamic pricing with dynamically changing granularity?

Open Problems: new tooling needed?

- Granularity of serverless is much smaller than traditional server based tool
- Debugging is much different if instead of having one artifact (a micro-service or traditional monolithic app) developers need to deal with a myriad of smaller pieces of code …
  - That haystack can grow really big really fast ...

Open Problems: can “legacy” code be made to run serve

- Today the amount of existing (“legacy”) code that must continue running is much larger than the new code created specifically to run in serverless environments
- The economical value of existing code represents a huge investment of countless hours of developers coding and fixing software
- Therefore, one of the most important problems may be to what degree existing legacy code can be automatically or semi-automatically decomposed into smaller-granularity pieces to take advantage of these new economics?

Open Problems: is serverless fundamentally stateless?

- Is serverless fundamentally stateless?
- Current serverless platforms are stateless will there be stateful serverless services in future?
- Will there be simple ways to deal with state?
- Can there be serverless services that have stateful support built-in
  - And with different degrees of quality-of-service?

Open Problems: patterns for building serverless solutions?

- Combine low granularity basic building blocks of serverless (functions, actions, triggers, packages, ...) into bigger solutions?
- How to decompose apps into functions so that they user resources optimally?
- Are there lessons learned that can be applied from OOP design patterns, Enterprise Integration Patterns, etc.?

Open Problems: serverless beyond traditional cloud of servers?

- IF functions is running outside of data-center is it serverless?
  - Cost, scalability, ...
- Internet of Things (IoT) will have many small devices each capable of running small amount of code - like functions in serverless?
- New domains, new concerns?
  - For example for IoT energy usage may be more important than speed?
- Are Blockchain smart contracts server-less?
  - For example when Ethereum users are running smart contracts they get paid for the “gas” consumed by the code, similar to fuel cost for an automobile but applied to computing (no need for data-center!)