



AWS Summit

AWS技术峰会 2015 · 上海





AWS Lambda: 事件驱动的云中代码

黄德滨, 合作伙伴解决方案架构师



简单问题

- 你想把S3存储桶的图片自动创建缩略图
- 你想检查一下存在DynamoDB中的地址格式是不是正确的

...原来的解决方案

- 创建一批机器获取这些数据或者文件。
- 对于每批数据或者文件提交一个任务到队列当中。
- 创建另外一批机器读取这个队列并执行任务。
- 需要为部署这些服务提供相应的解决方案。
- 计划整个架构的承载能力，考虑容错机制，长期的平均使用率、突发的高峰使用率等等。
- 24x7x365监控整个架构利用率、健康状况、安全情况等等
- 当资源不够时横向或者纵向扩展
- 操作系统以及运行环境补丁的更新
-

有没有更好的方式？

如果每个AWS的服务都产生事件将怎样？



AWS Lambda – 优势

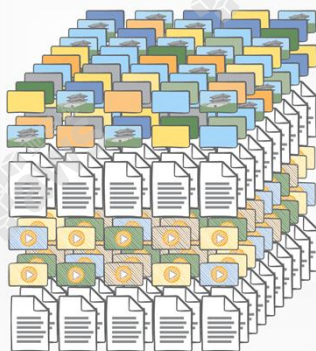
1

无需管理服务器



2

持续扩展



3

亚秒级收费



AWS Lambda – 能力

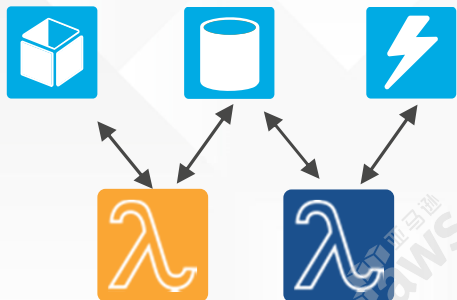
自带你的代码



计算“能力级别”自选



灵活的调用路径



细粒度的权限控制



AWS Lambda – 如何工作

编程



无状态



部署



监控&日志



实现的需求

- 不用操心扩展能力/无需配置
- 内置的部署功能
- 默认的高可用
- 自带你的代码（BYOC）
- 无需为闲置付费

*Bring Your Own Code 使用自己的代码

AWS Lambda五个应用场景



可扩展的移动
应用和IoT后台



无服务器模式的
微服务



第三方应用平
台的扩展



Amazon服务的
功能扩展



实时的流数据
分析



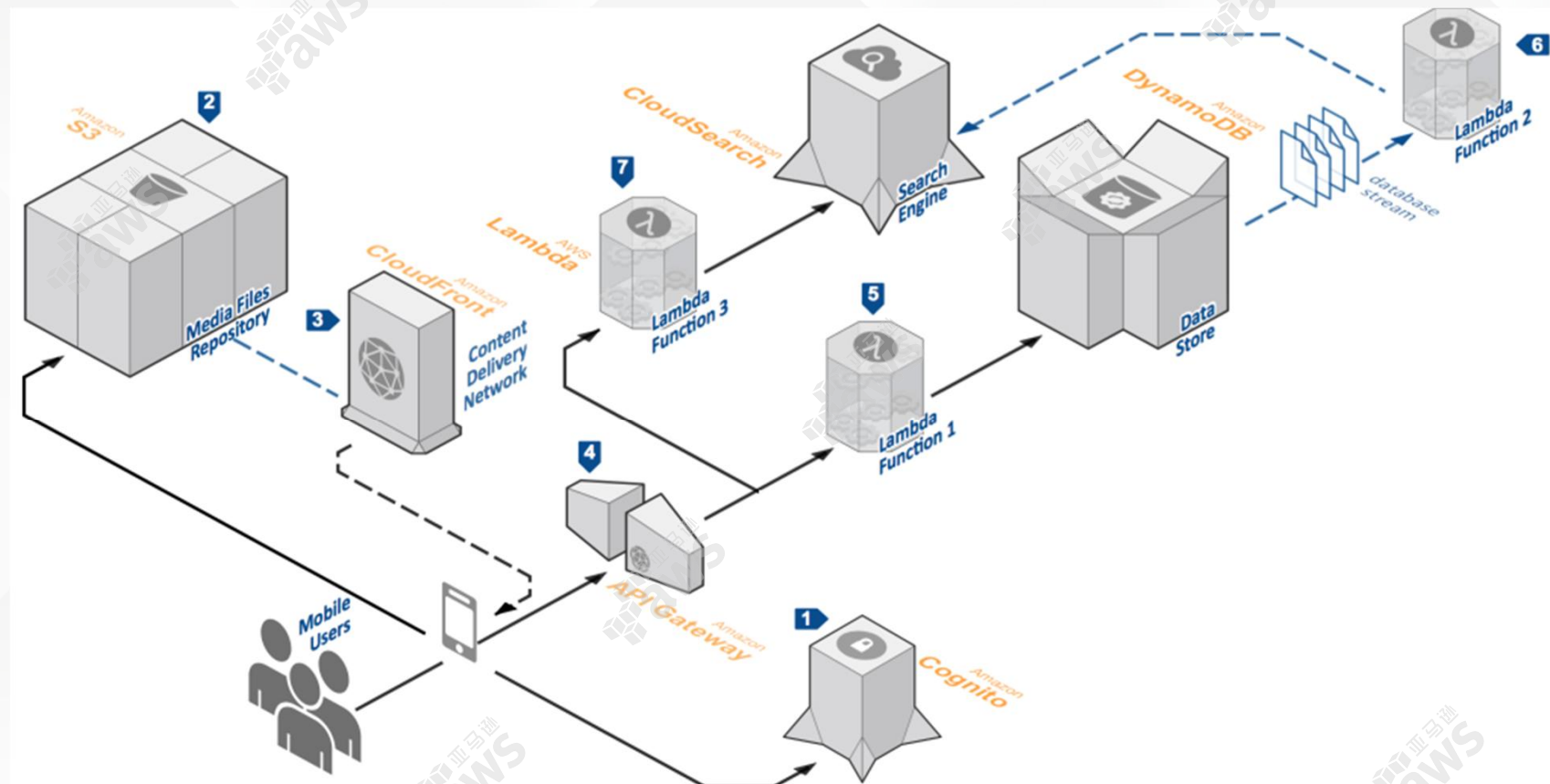
移动与IoT应用后台



一个可扩展的移动应用后台。。。
不需要敲代码的移动应用后台



演示：AWS Lambda作为移动应用后台



让我们看一下演示。。

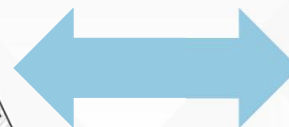
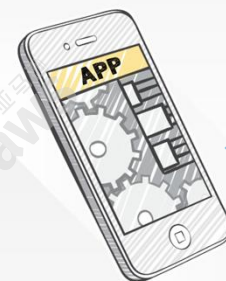
创建移动后台应用：各种“插件”

需要用户登录？

使用Amazon Cognito认证。

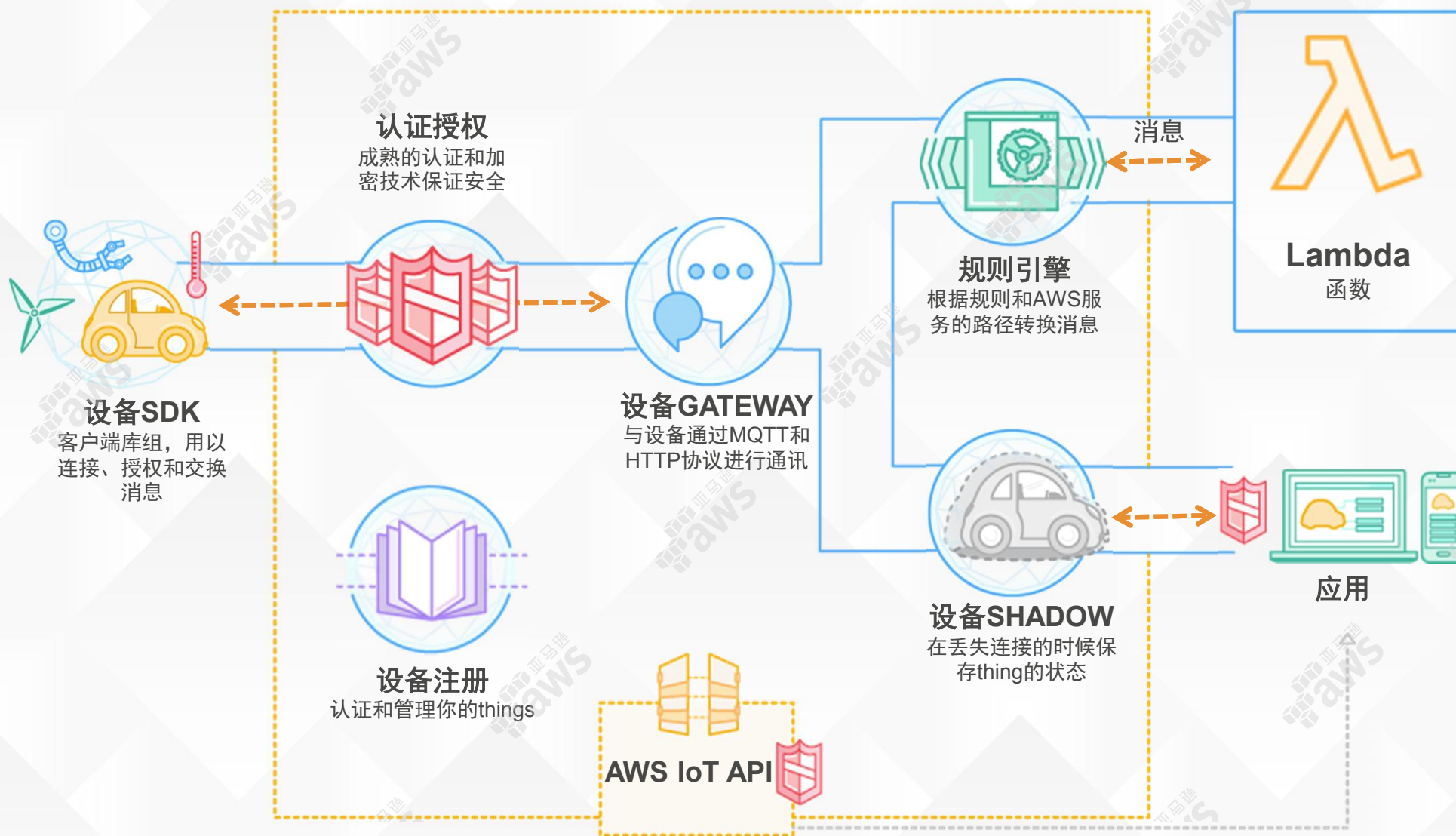
需要根据设备进行渲染？

设备信息就在函数上下文对象中。



AWS Lambda

AWS IoT + AWS Lambda



为什么这么容易？

消除“在你的环境里运行”和“在客户的环境里运行”之间的距离

自动扩展

自动补丁

内置监控

内置日志

闲置无需付
费

内置Web服
务器

HTTP端点

自动部署

内置安全

深入探讨：编程模式





- 三种主要语言
 - Node.js
 - Java
 - Python *New*
- 加上Scala、Clojure和其他“jvm”语言
- 后台运行进程
 - Node.js、Java和Python
 - 也可以使用csh或者你自己的可执行程序
- Cron计划执行模式 *New*
 - 标准的cron语法
 - 最小5分钟的粒度






无服务器的微服务



可以直接开放为HTTP端点

 **AWS**  **Services**  **Edit** 

lambda-demo-account  N. Virginia  Support 

Lambda >

Step 1: Select blueprint






Step 2: Configure function

Step 3: Configure endpoints

Step 4: Review

Configure endpoints

You can create an endpoint for your Lambda function using Amazon API Gateway. You can either create the endpoint using the default values below, or use the [API Gateway console](#) to create a custom endpoint.

API name	<input type="text" value="LambdaMicroservice"/>	
Resource name	<input type="text" value="myAPI"/>	
Method	<input type="text" value="POST"/>	
Deployment stage	<input type="text" value="prod"/>	
Security	<input type="text" value="Open with access key"/>	

Your API endpoint may be invoked by any user with the API key. Go to the [API Gateway console](#) to configure keys.

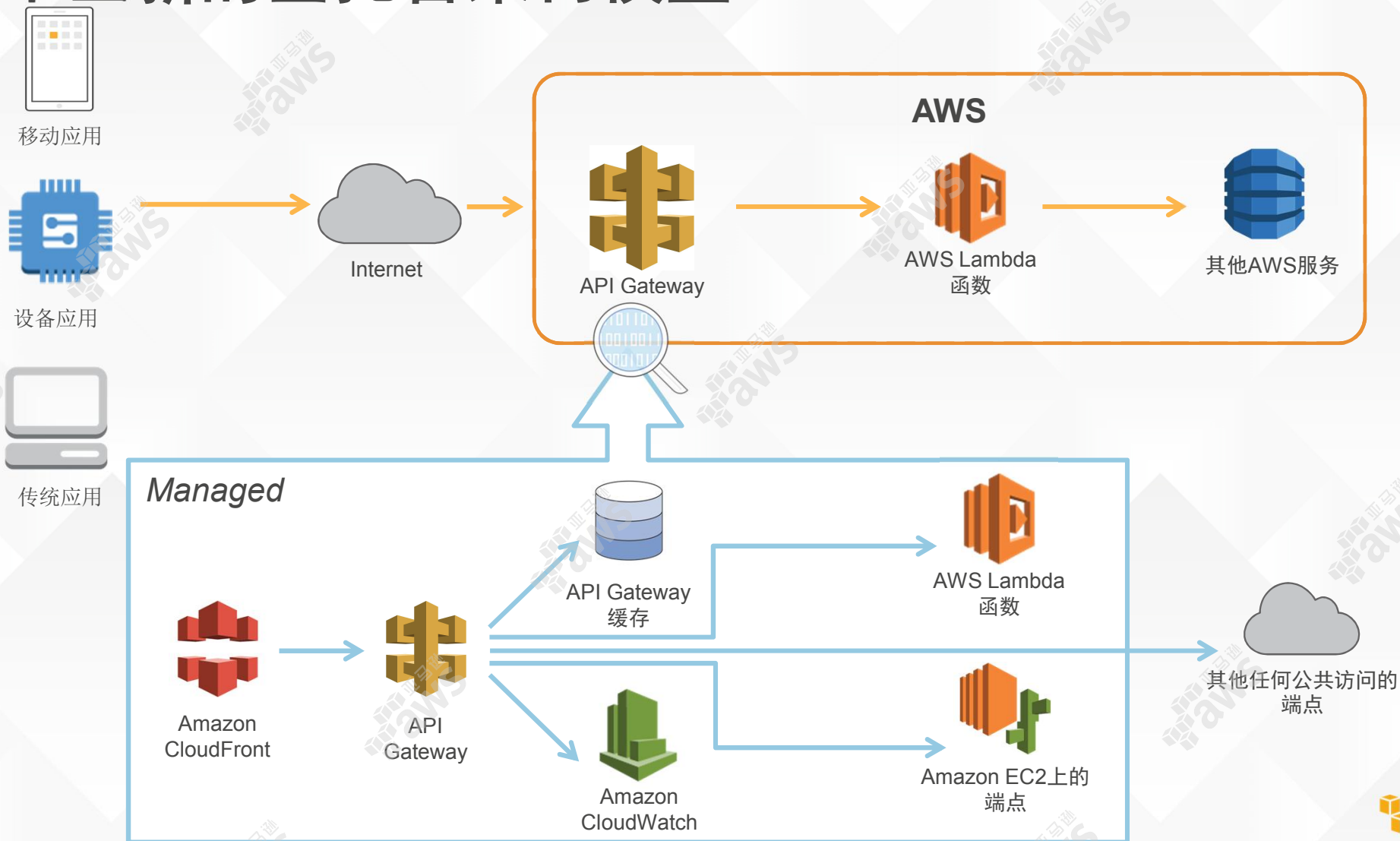
可以直接在SDK中调用

- 开发语言
 - JavaScript
 - Java
 - Object-C / Swift
 - .NET
 - PHP
 - Go
 - Python
 - Ruby
- 浏览器
- 移动端
 - Android
 - iOS
- 各种容器
 - JVM
 - Node.js

```
<script>
function updateScores() {
  AWS.config.update({accessKeyId: globalConfig.accessKeyId, secretAccessKey});
  AWS.config.region = globalConfig.region;

  //Post New Score
  var lambda = new AWS.Lambda();
  lambda.invoke({
    FunctionName: globalConfig.lambdaFunction,
    Payload: JSON.stringify({action: 'getHighScores'})
  }, function(err, data) {
    if (err) console.log(err.stack)
    else {
      var result = JSON.parse(data.Payload);
      $('#scores').empty();
      for (i = 0; i < result.length; i++) {
        var item = result[i];
        $("#scores").append('<div class="item"><div class="player">'+
          div class="score">'+item.score+'</div></div>');
      }
      setTimeout(updateScores, 8000);
    }
  });
}
updateScores();
</script>
```

一个全新的全托管架构模型



这种模式的关键点

1

- AWS Lambda + Amazon API Gateway意味着没有需要管理的基础架构——我们替你扩展

2

- 安全很重要，也很复杂——由AWS的IAM来负责

3

- Swagger导入并生成客户端的SDK——我们可以自动做大部分事情

我们要用到的AWS服务



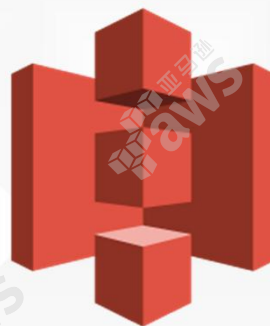
Amazon API Gateway

提供API服务端点并路由调用



AWS Lambda

执行应用的业务逻辑



Amazon S3

提供静态网页服务







Amazon DynamoDB

数据存储

让我们看一下演示。。。

深入探讨：编程模式

- 同步或者异步运行代码。。。 
 - AWS命令行接口
 - AWS Lambda自己调用自己（没错，支持Self-hosts模式）
 - 更长的运行时间：最长5分钟 *New* 
- 代码是预授权的 
 - 选择一个角色Lambda会使用这个角色
 - 跨帐号调用也是支持的
 - 与Cognito和API Gateway一起提供更为健壮的安全架构 

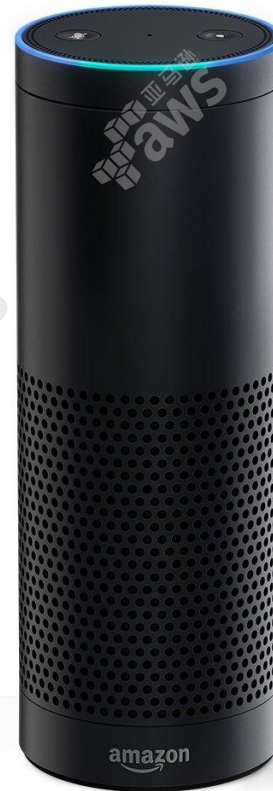


扩展第三方的平台

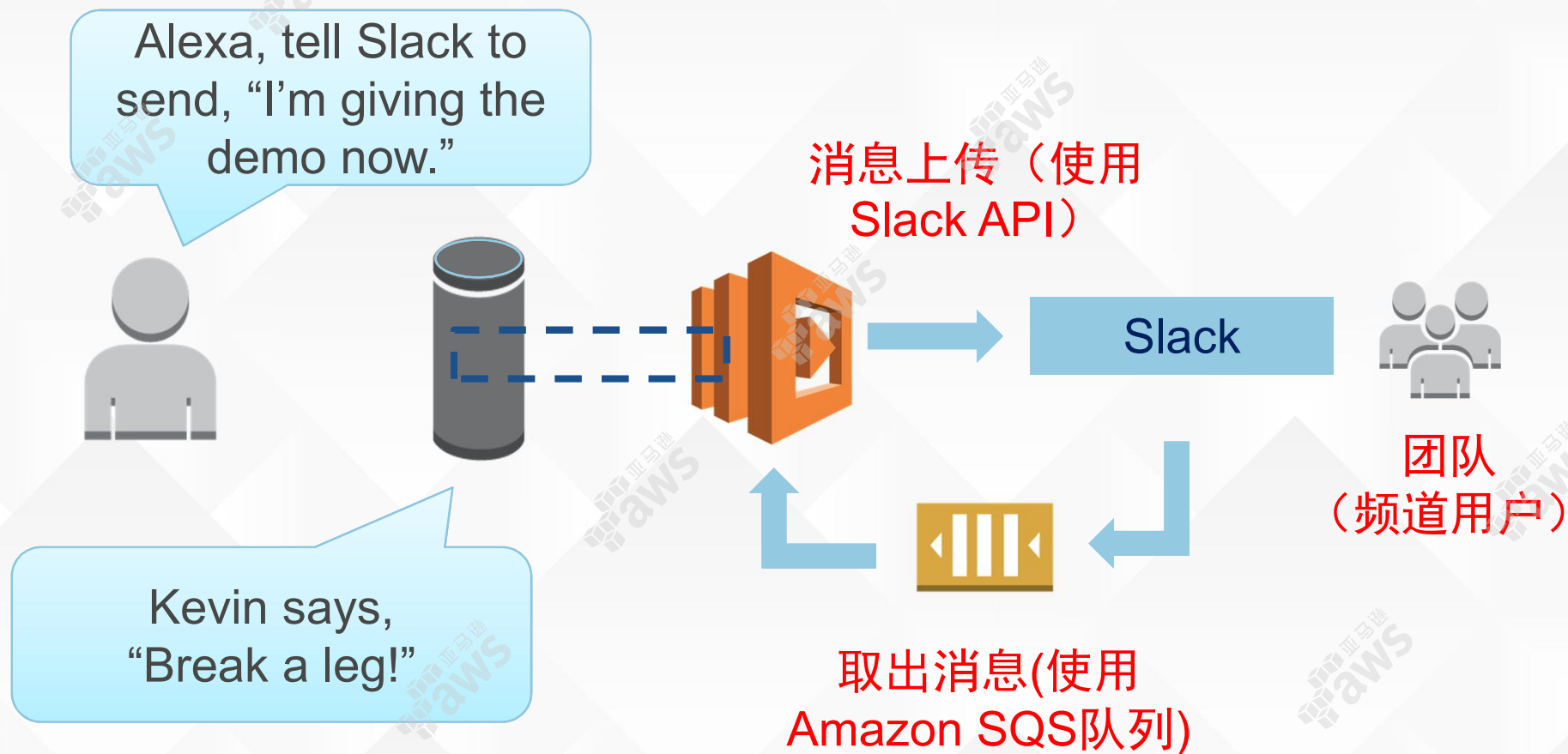


接入设备的计算资源

- Alexa Skills Kit——创建语音应用
- AWS Lambda作为接入设备或者IoT的平台



Slack演示架构

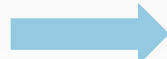


GitHub事件响应



团队
(资源库用户)

GitHub



GitHub事件



Amazon SNS消息

AWS Lambda – 合作伙伴平台

ALGORITHMIA

splunk

amazon echo

twilio

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint and customize as needed, or skip this step if you want to author a Lambda function and configure it otherwise noted, blueprints are licensed under [CC0](#).

Filter

splunk-logging

Demonstrates logging events to Splunk's HTTP Event Collector.

nodejs · splunk

alexa-skills-kit-color-expert

Demonstrates a basic skill built with the Amazon Alexa Skills Kit.

nodejs · alexa

cognito-sync-trigger

lambda-test-harness

为什么那么容易？

平台提供者

- 不用任何主机
- 不用任何License
- 不用管理能力
- 不用运行Web服务
- 低时延同步调用
- “触发并忘记”的事件

应用开发者

- 成本低廉
- 永远免费的节点
- 不用基础设施
- 无需“样板”代码
- 可以选择语言
- 没有库的限制

深入探讨：资源的选择

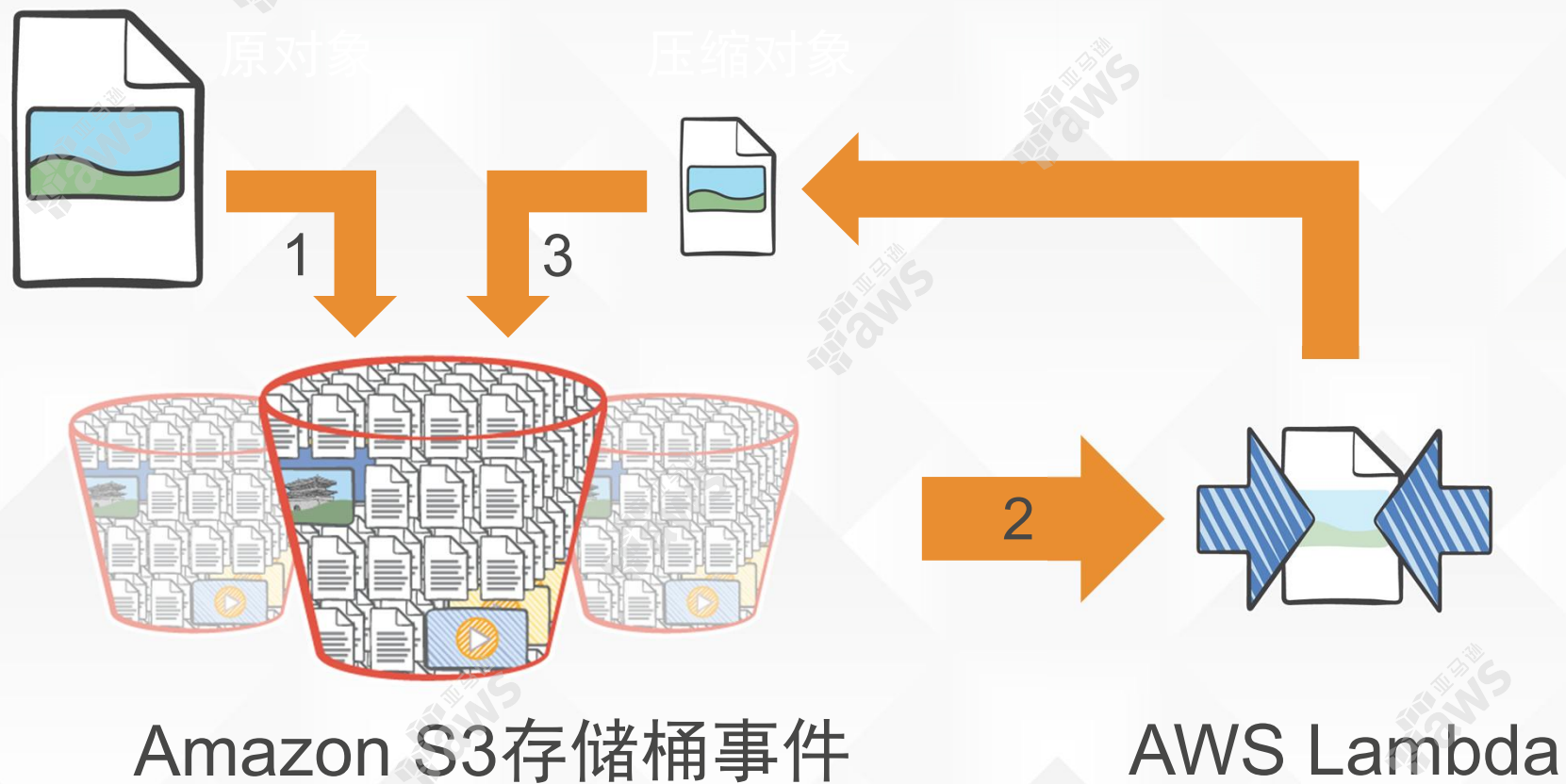
- AWS Lambda提供23种“能力级别”
- 更高的级别提供更多的内存和更强的CPU性能
 - 128 MB, 低CPU性能
 - 1.5 GB, 高CPU性能
- 更高的能力级别：执行计算密集型和爆发式的任务时延迟低
- 能力级别影响定价
- 函数运行时间从100毫秒到5分钟之间



扩展Amazon服务的功能



扩展Amazon S3添加自动压缩功能



如何添加一个Amazon S3功能

1. 随便从网上找一段Node.js压缩代码示例。
2. 从Amazon S3事件示例开始：
 1. 从S3上GET原对象
 2. 压缩
 3. 将压缩的版本PUT回S3
3. 选择一些S3存储桶应用这个事件。

让我们看一下演示。。。。

可以与多种Amazon AWS服务集成



IoT
后台



Amazon S3
事件触发



Amazon SWF
任务



Amazon SNS
定制化消息



Amazon Cognito
事件触发



API Gateway
微服务



Amazon
Dynamo DB
事件触发



Amazon
Kinesis
处理器

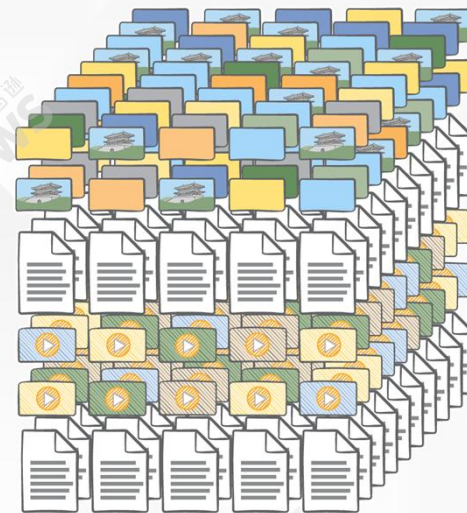


AWS
CloudFormation
定制化资源

... 还在继续增长

为什么这么容易？请求级别的扩展！

- 谁知道事件的频率？ S3和 Lambda!
- 你不可能低于或者高于配置（依据设计）
- 根据使用付费





实时的流数据分析



容易的实时数据流分析架构

智能设备

点击流数据

日志数据

使用Amazon Kinesis汲取数据 (PUT记录)



AWS Lambda提取记录



你的代码只在每批纪录处理时运行一次



Amazon S3



Amazon DynamoDB



Amazon Redshift



深入探讨：重试和事件排序

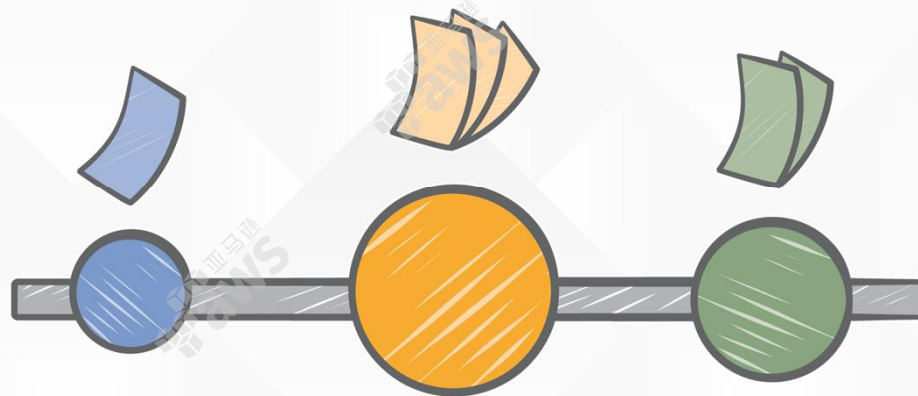
- 三种可能：
 - 同步调用你的AWS Lambda函数。
 - 使用AWS SDK？在这里添加重试逻辑。
 - 使用RESTful直接调用Lambda？你可以完全控制重试。
 - 根据调用决定顺序。
 - Amazon S3或者SNS触发Lambda函数，或者异步调用你的代码
 - 总共三次重试，然后事件将被忽略
 - 没有顺序
 - Lambda获取Amazon Kinesis或者Amazon DynamoDB更新流
 - 重试没有限制，有顺序保证



实用的提示

细粒度的价格

- 以100毫秒为单位购买
- 请求费用低
- 没有按小时、天或者月度计费
- 没有按设备收费



永远不用为闲置付费。

免费使用

100万请求和40万GB计算资源每个月每个帐户。

构建和部署集成



Jenkins



Grunt



AWS CloudFormation



Amazon S3

合作伙伴



CODESHIP

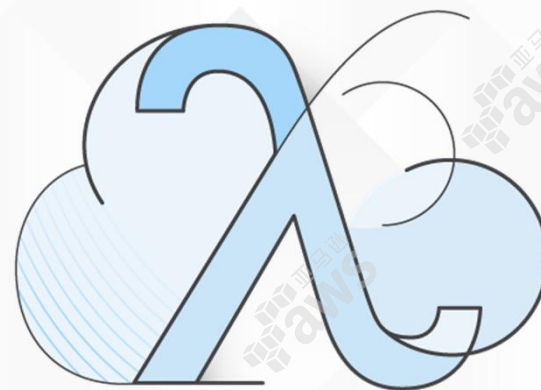
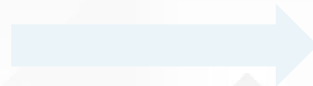
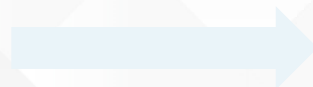
版本: 开发阶段

- 开发阶段保持AWS Lambda的简单:
 - 上传代码
 - 任何时候都可以修改
 - 最后的更新生效



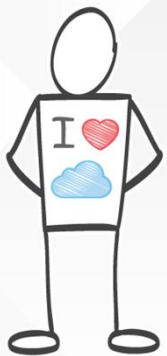
```
exports.handler =  
function(event, context)  
{ context.succeed("hi"); }
```

```
exports.handler =  
function(event, context)  
{ context.succeed("bye"); }
```



版本：发布

- 在任何时候都可以将开发发布为一个新的版本：
 - “复制” 一个开发版本到一个编号版本
 - 发布的版本都是只读的（包括配置）
 - 每个函数都是简单的整数计数



```
exports.handler =  
function(event, context)  
{context.succeed("hi");}
```

```
exports.handler =  
function(event, context)  
{context.succeed("bye");}
```

1

2

Versions

版本：调用Lambda函数

- 开发版本：

FunctionName (或)

FunctionName:\$LATEST

- 特定版本：

FunctionName:1

FunctionName:2

- 命名版本：

FunctionName:production

FunctionName:v1_2_3_4

版本：别名

- 可以为任何一个版本命名别名：
 - 可以在代码中使用ARN映射
 - 可以不用修改客户端代码



```
exports.handler =  
function(event, context)  
{context.succeed("hi");}
```

```
exports.handler =  
function(event, context)  
{context.succeed("bye");}
```

prod

dev

Aliases

API和Lambda代码的版本

Amazon API Gateway: API版本

- `/prod/my_url_endpoint`
→
- `MyFunction:prod_rel`

AWS Lambda: 代码版本

- `MyFunction:prod_rel`
→
- `Function:3` →
- `{your code}`



我们已经忙了很久。

现在该你了。





1



进入AWS Lambda控制台，创建一个函数，运行它。
(成为在中国的前100万个请求)



2.

恭喜，你已经成为一个Lambda函数专家了！ 添加一个事件源或者一个HTTP端点。



3.

创建世界上最简单的移动应用后端。
(提示：从内置的CRUD例子开始！)





变成AWS Lambda粉丝！

aws.amazon.com/blogs/compute

aws.amazon.com/lambda

AWS Lambda论坛





Thank You

