PyConChina 2019

从模块化到全球分发
Python 在 Serverless 领域
你不能错过的最新功能

Pahud Hsieh(谢洪恩)

Specialist Solutions Architect, Serverless
Amazon Web Services

PyConChina 2019

A Little bit about Myself

hunhsieh

Pahud Hsieh

- Serverless Specialist SA, GCR
- Based in Taipei
- Focus on Serverless, Container and AWS CDK
- Father of a daughter
- Speaker in AWS Global Summits, KubeCon and CNCF Webinar
- Active in AWS User Groups from Shanghai, Taipei, Korea and JAWSUG
- Fan of road trip. Been to Siberia, Balkans, India, Cuba and Svalbard

@pahudnet

aws

PyConChina 2019

目录 CONTENTS

- Why Serverless
- How Serverless in Python
- Packaging and Distribution
- Serverless to the Next Level

aws

PyConChina 2019

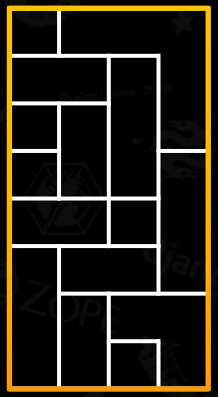
Why Serverless

aws

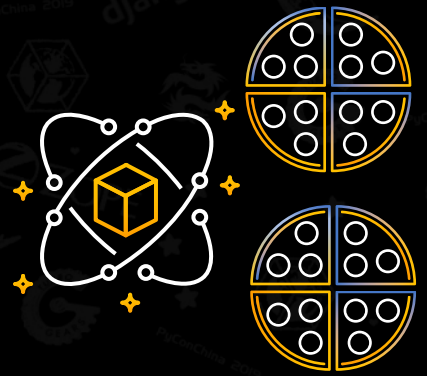
Development transformation at Amazon: 2001–2002

Lesson learned: decompose for agility

2001 ----------> **2002**

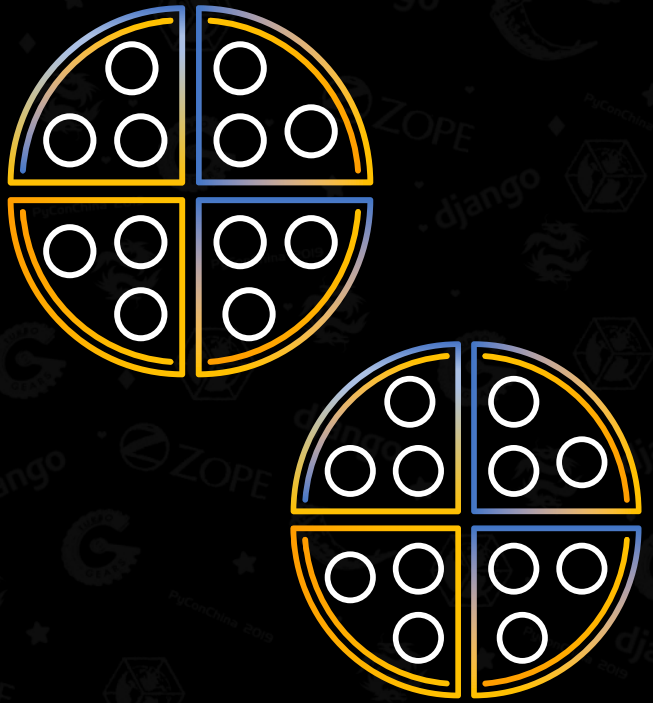


monolithic application
+ teams



microservices
+ 2 pizza teams

Two-pizza teams



- Full ownership
- Full accountability
- “DevOps”
- Focused innovation

When the impact of change is small, release velocity can increase

Monolith

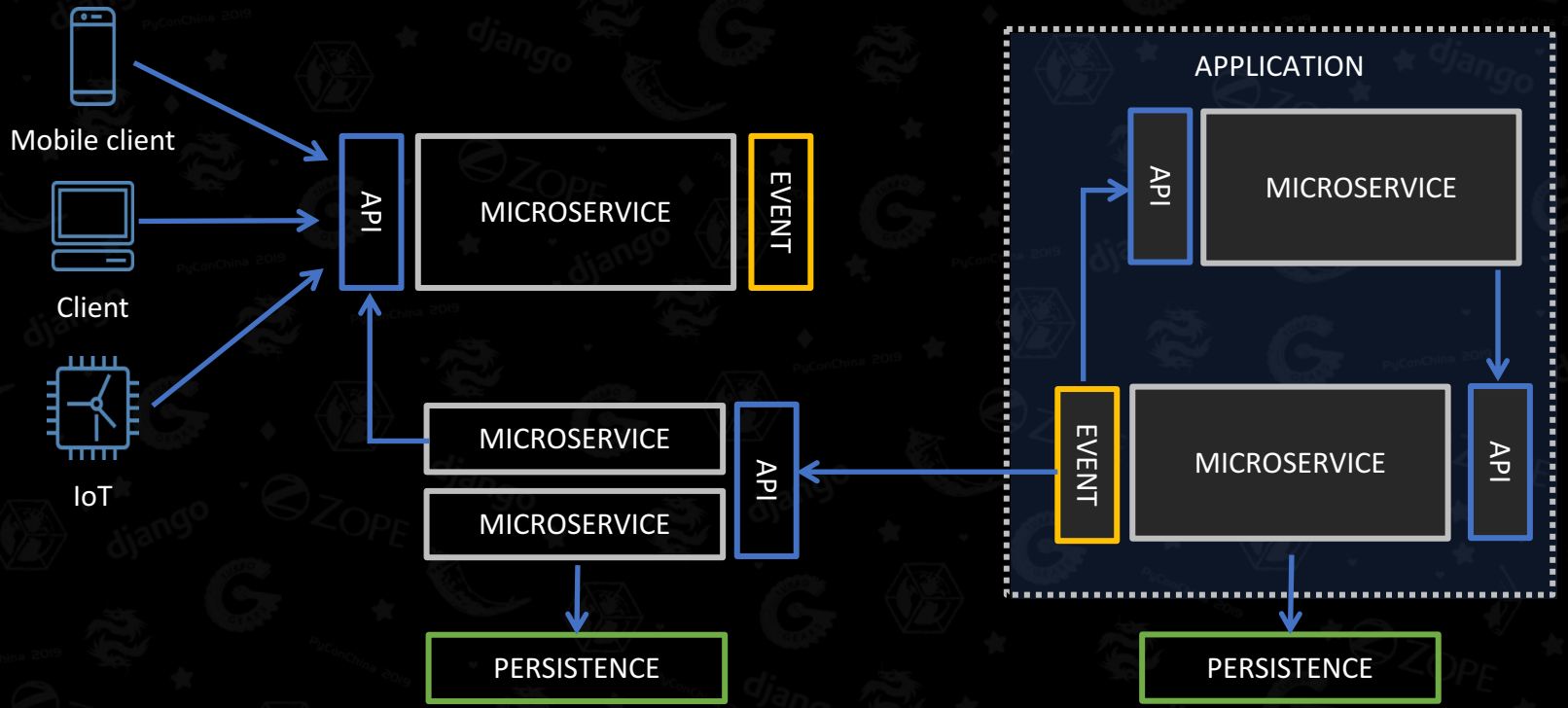
Does everything

Microservices

Does one thing

PyConChina 2019

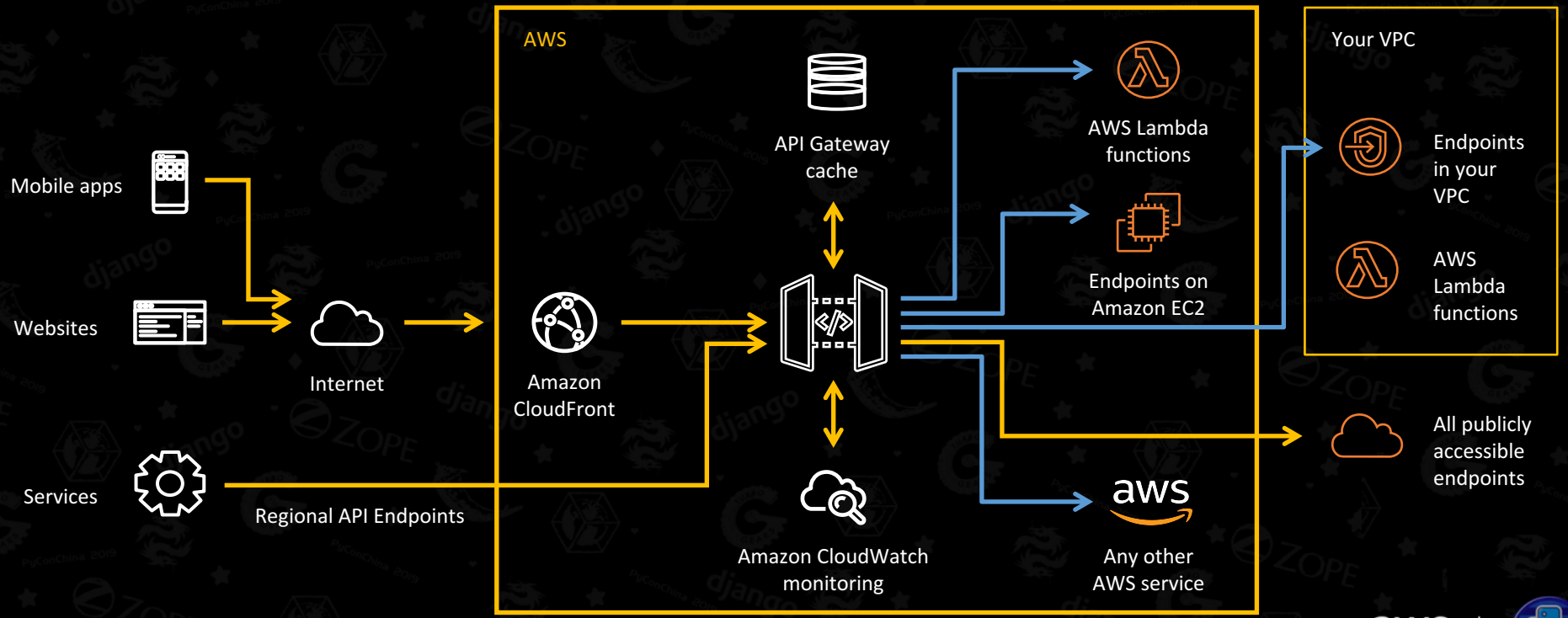
Microservices Architecture



APIs are the front door of microservices

PyConChina 2019

Manage APIs with API Gateway



Decouple state from code using messaging

Messaging

Amazon Simple Queue Service

Queues

Simple

Fully-managed

Any volume

Amazon Simple Notification Service

Pub/sub

Simple

Fully-managed

Flexible

Amazon CloudWatch Events

Synchronization

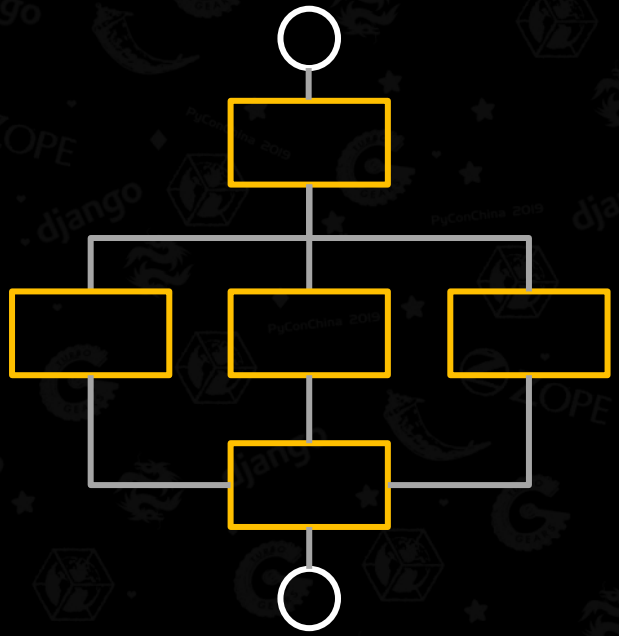
Rapid

Fully-managed

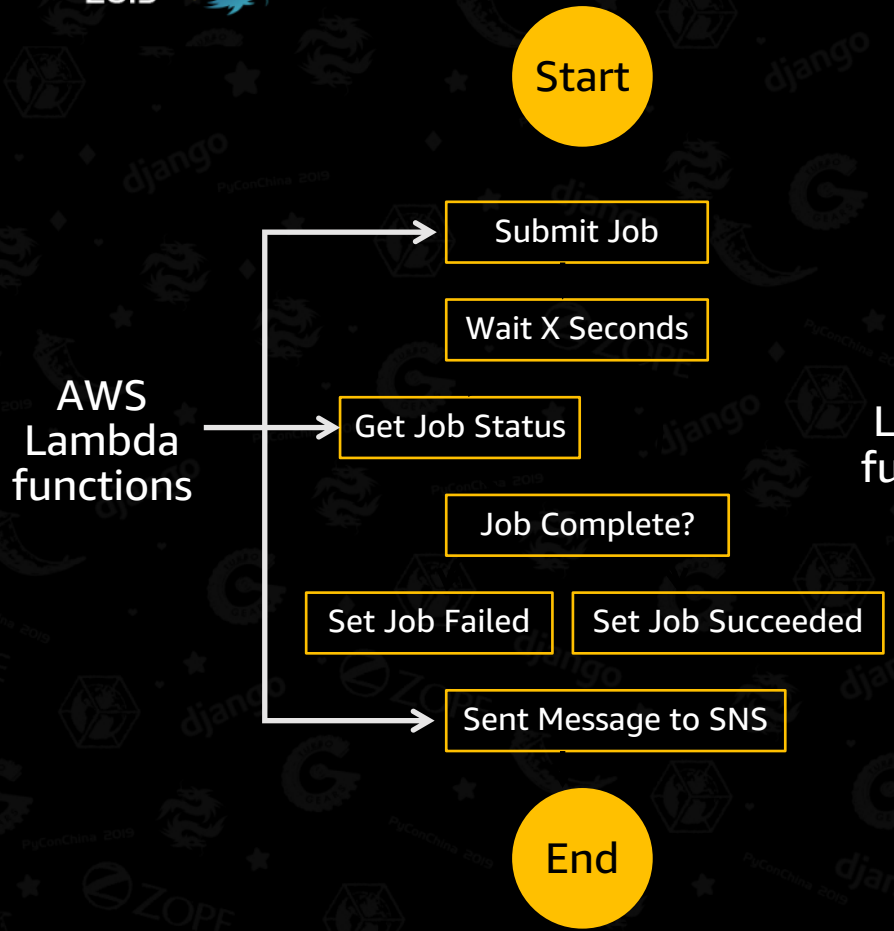
Real-time

Build workflows to orchestrate everything

Track status of data and execution



Remove redundant code



No Lambda functions

Serverless for Functions and Containers

AWS Lambda

Serverless event-driven code execution

Short-lived

All language runtimes

Data source integrations

AWS Fargate

Serverless compute engine for containers

Long-running

Bring existing code

Fully-managed orchestration

Lambda layers

Let functions easily share code; upload layer once, reference within any function

Layer can be anything: Dependencies, training data, configuration files, etc.

Promote separation of responsibilities and let developers iterate faster on writing business logic

Built-in support for secure sharing by ecosystem

Including library dependencies in a layer

Runtime	Folders
Python	python python/lib/python3.7/site-packages (site directories)
Node.js	nodejs/node_modules nodejs/node8/node_modules (NODE_PATH)
Java	java/lib (CLASSPATH)
Ruby	ruby/gems/2.5.0 (GEM_PATH) ruby/lib (RUBY_LIB)
All	bin (PATH) lib (LD_LIBRARY_PATH)

- AWS Lambda was released in 2014(almost 5 years)
- Support Python 3.7/3.6/2.7, Java 8, NodeJS 10/8.10 .NET Core 2.1(C# and PowerShell) , Go 1.x, Ruby 2.5
- Cloud9 IDE Integration
- Environment Variables with KMS Support
- Support Custom Runtime
- Support Lambda Layer
- 99.95% SLA

PyConChina 2019

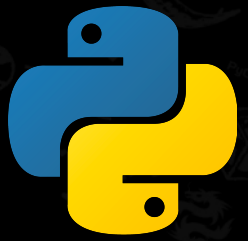
Decouple and Modularize your Serverless Application

aws

Lambda layers

- Let functions easily share code; upload layer once, reference within any function
- Layer can be anything: Dependencies, training data, configuration files, etc.
- Promote separation of responsibilities and let developers iterate faster on writing business logic
- Built-in support for secure sharing by ecosystem

Build your Python Lambda Layer



python or python/lib/python3.7/site-packages

```
pillow.zip
| python/PIL
└ python/Pillow-5.3.0.dist-info
```

Including library dependencies in a layer

Runtime	Folders
Python	python python/lib/python3.7/site-packages (site directories)
Node.js	nodejs/node_modules nodejs/node8/node_modules (NODE_PATH)
Java	java/lib (CLASSPATH)
Ruby	ruby/gems/2.5.0 (GEM_PATH) ruby/lib (RUBY_LIB)
All	bin (PATH) lib (LD_LIBRARY_PATH)

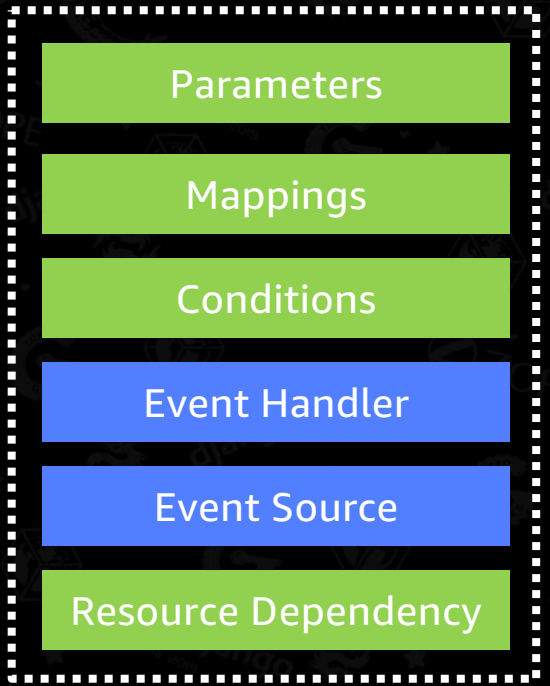
Meet SAM!

Package and Distribution

Serverless Application Model

Serverless Application Repository

What does a Serverless App look like



SAM Template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/todo_list.zip
      Handler: index.handler
      Runtime: python3.7
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetHtml:
          Type: Api
          Properties:
            Path: /{proxy+}
            Method: ANY

  ListTable:
    Type: AWS::DynamoDB::Table
```

Tells AWS CloudFormation this is a SAM template it needs to “transform”

Creates a AWS Lambda function with the referenced managed AWS IAM policy, runtime, code at the referenced zip location, and handler as defined. Also creates an Amazon API Gateway and takes care of all mapping/permissions necessary

Creates a Amazon DynamoDB table

Build, Package and Distribute

- $ sam build
- $ sam package
- $ sam deploy
- $ sam publish

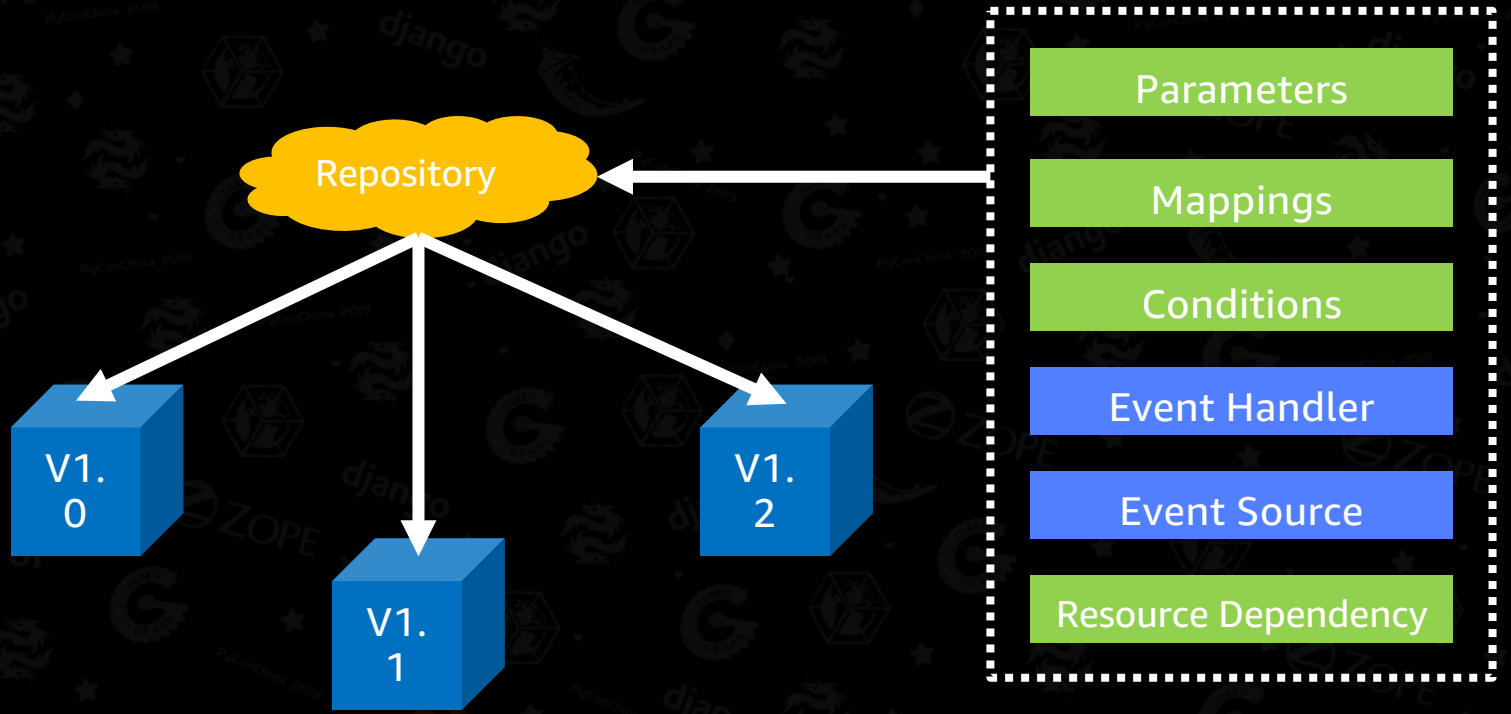
PyConChina 2019

Develop/package/deploy SAM application

```
$ ENV_VAR=VALUE sam local invoke FunctionName --event event.json

$ ENV_VAR=VALUE sam local start-api

$ sam package --template-file template.yaml \
        --s3-bucket my-s3-bucket \
        --output-template-file packaged.yaml

$ sam deploy --template-file packaged.yaml \
       --capabilities CAPABILITY_IAM CAPABILITY_AUTO_EXPAND \
       --stack-name my-stack-name \
       --parameter-overrides Foo=bar

$ sam logs --name FunctionName --stack-name StackName --tail
```

aws

Package, Publish and Deploy

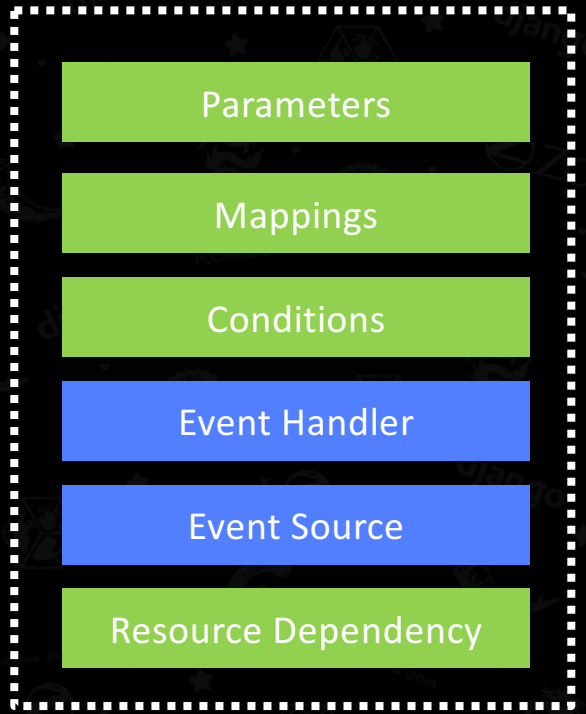


PyConChina 2019

Serverless Application Repository(SAR)

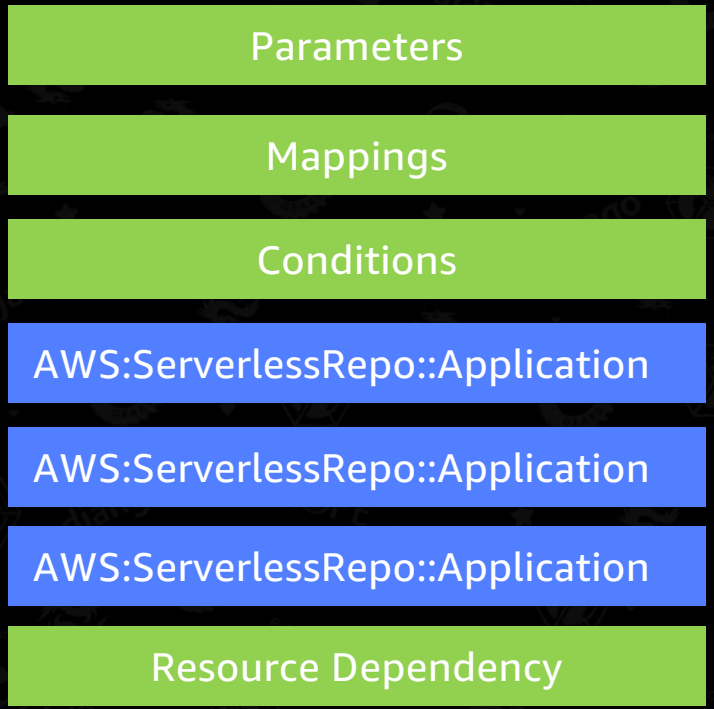
aws

What does a Serverless App look like



AWS:ServerlessRepo::Application

Now your Serverless Stack looks like this



Serverless Stack as Code

Re-used in Private or Public

Zero-code required

Parameter Overrides

Global Deployment

No servers and containers

Reusing Your Serverless App means:

No Server + (almost) **No Code** !

Distribute to the world

Metadata in SAM template

```
Metadata:
  AWS::ServerlessRepo::Application:
    Name: my-serverless-app
    Description: "My Demo Serverless App for SAR"
    Author: Pahud Hsieh
    SpdxLicenseId: Apache-2.0
    LicenseUrl: LICENSE
    ReadmeUrl: README.md
    Labels: [demo','lambda','kubectl','eks', 'aws', 'kubernetes', 'k8s']
    HomePageUrl: https://github.com/pahud/my-demo-sar-app
    SemanticVersion: 1.0.1
    SourceCodeUrl: https://github.com/pahud/my-demo-sar-app
```

$ sam publish

PyConChina 2019

Publish SAM application to SAR

```
$ sam package --template-file template.yaml \
        --s3-bucket my-s3-bucket \
        --output-template-file packaged.yaml

$ sam publish --template-file packaged.yaml
The following metadata of application "arn:aws:serverlessrepo:us-east-1:068896461592:applications/my-serverless-app" has been updated:
{
 "Description": "My Demo Serverless App for SAR",
 "Author": "Pahud Hsieh"
}
Click the link below to view your application in AWS console:
https://console.aws.amazon.com/serverlessrepo/home?region=us-east-1#/published-applications/arn:aws:serverlessrepo:us-east-1:0123456789:my-serverless-app
```

aws

One-click Deployment from SAR

Deploy from SAR console

Region	Click and Deploy
ap-northeast-1	SAR Deploy Now
ap-northeast-2	SAR Deploy Now
ap-northeast-3	SAR Deploy Now
ap-south-1	SAR Deploy Now
ap-southeast-1	SAR Deploy Now
ap-southeast-2	SAR Deploy Now
ca-central-1	SAR Deploy Now
eu-central-1	SAR Deploy Now
eu-north-1	SAR Deploy Now
eu-west-1	SAR Deploy Now
eu-west-2	SAR Deploy Now
eu-west-3	SAR Deploy Now
sa-east-1	SAR Deploy Now
us-east-1	SAR Deploy Now
us-east-2	SAR Deploy Now
us-west-1	SAR Deploy Now
us-west-2	SAR Deploy Now

Just a single click!

PyConChina 2019

Using Lambda Layers

- Put common components in a .zip file and upload it as a Lambda layer
- Layers are immutable and can be versioned to manage updates
- When a version is deleted or permissions to use it are revoked, functions that used it previously will continue to work, but you won't be able to create new ones
- You can reference up to five layers, one of which can optionally be a custom runtime

aws

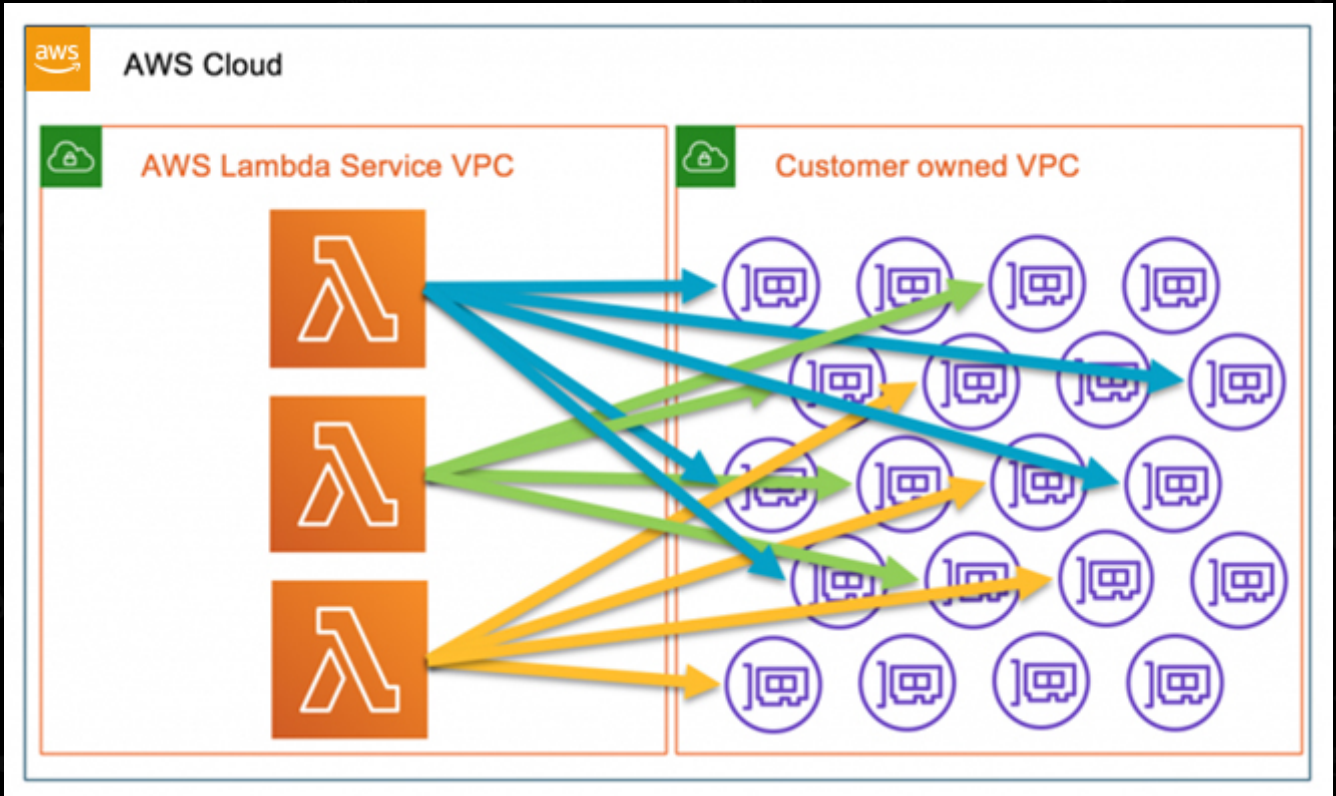
PyConChina 2019

Lambda in VPC Improvement

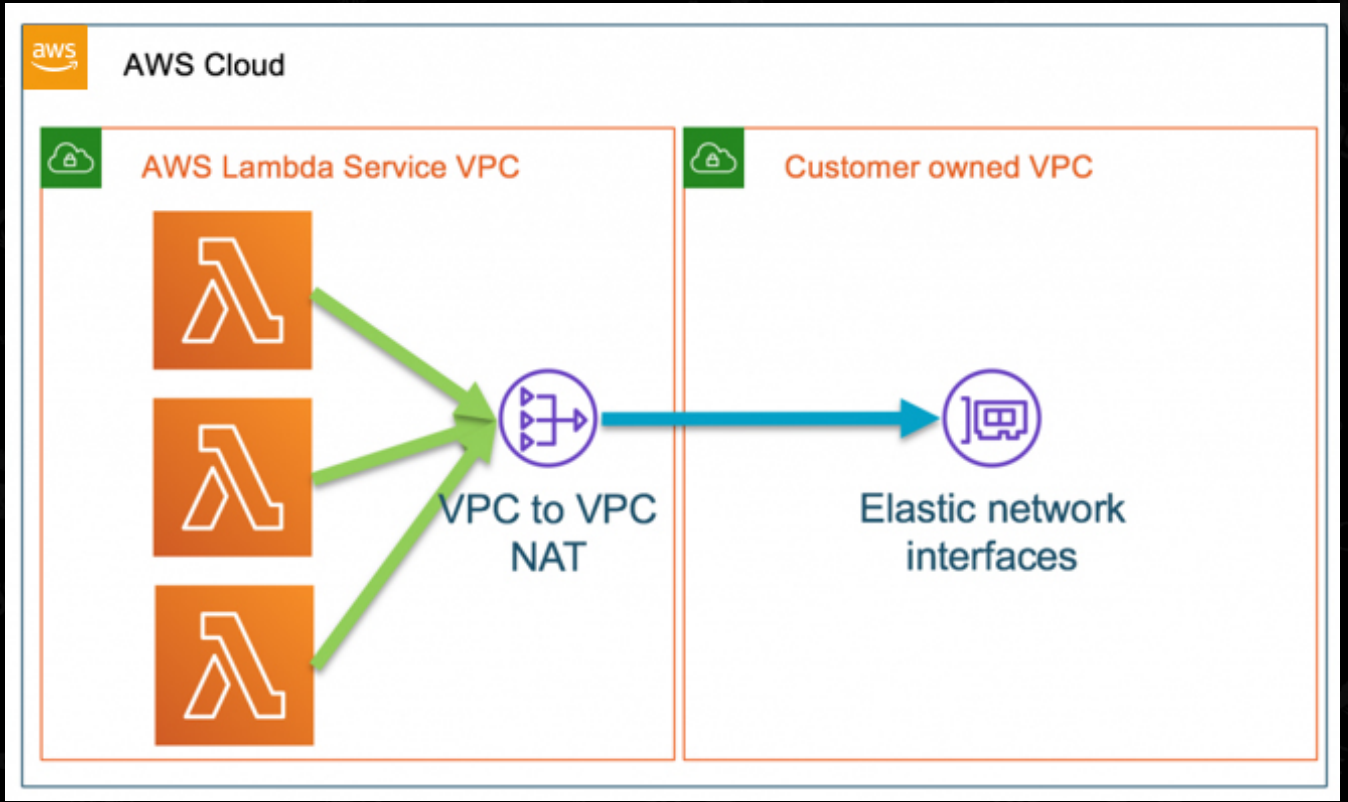
https://aws.amazon.com/cn/blogs/compute/announcing-improved-vpc-networking-for-aws-lambda-functions/

aws

Improved VPC Networking - Before

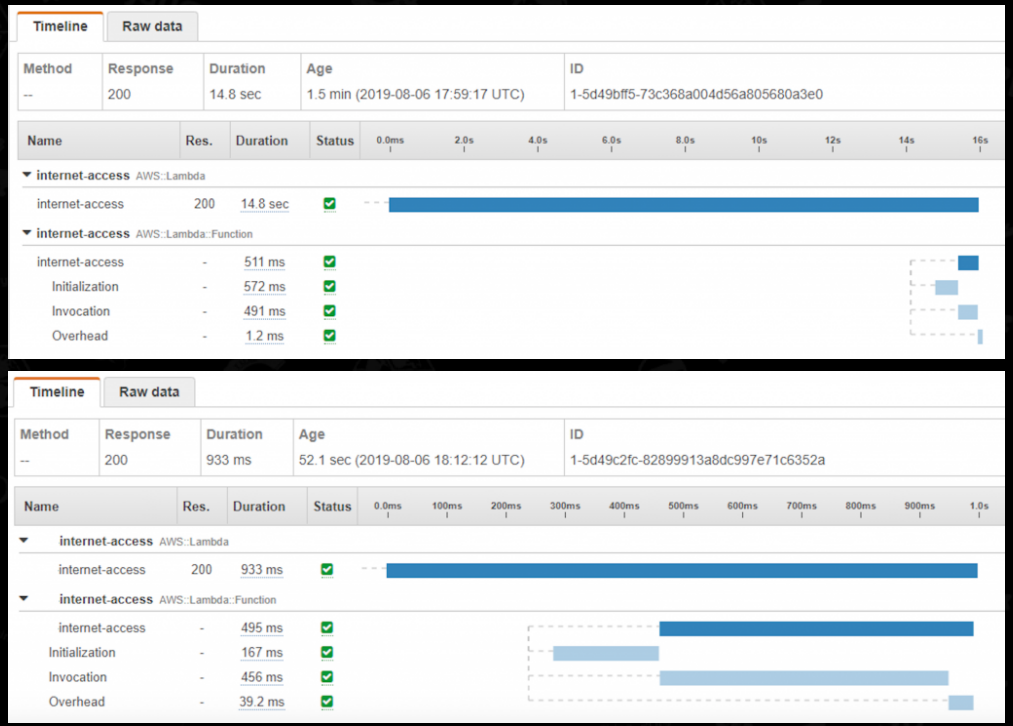


Improved VPC Networking - After



PyConChina 2019

Latency reduction from 14.8sec to less than 933ms



And we are rolling out to All AWS Regions

https://aws.amazon.com/cn/blogs/compute/announcing-improved-vpc-networking-for-aws-lambda-functions/

However, as a Python builder, What features should I really care about?

https://aws.amazon.com/cn/blogs/compute/announcing-improved-vpc-networking-for-aws-lambda-functions/

PyConChina 2019

AWS Cloud Development Kit

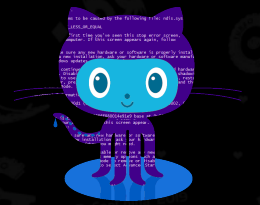
aws

PyConChina 2019

AWS Cloud Development Kit (AWS CDK)

A multi-language software development framework for modeling cloud infrastructure as reusable components

```python
from aws_cdk import core, aws_ec2, aws_ecs, aws_ecs_patterns


class CdkPyFargateStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # import default VPC
        vpc = aws_ec2.Vpc.from_lookup(self, 'VPC', is_default=True)

        # ECS cluster
        cluster = aws_ecs.Cluster(self, 'Cluster', vpc=vpc)
        svc = aws_ecs_patterns.ApplicationLoadBalancedFargateService(
            self, 'FargateService',
            cluster=cluster,
            image=aws_ecs.ContainerImage.from_asset('flask-docker-app'),
            container_port=5000,
            environment={
                'PLATFORM': 'AWS Fargate :-)'
            }
        )

        core.CfnOutput(self, 'SericeURL', value="http://{}".format(
            svc.load_balancer.load_balancer_dns_name))
```



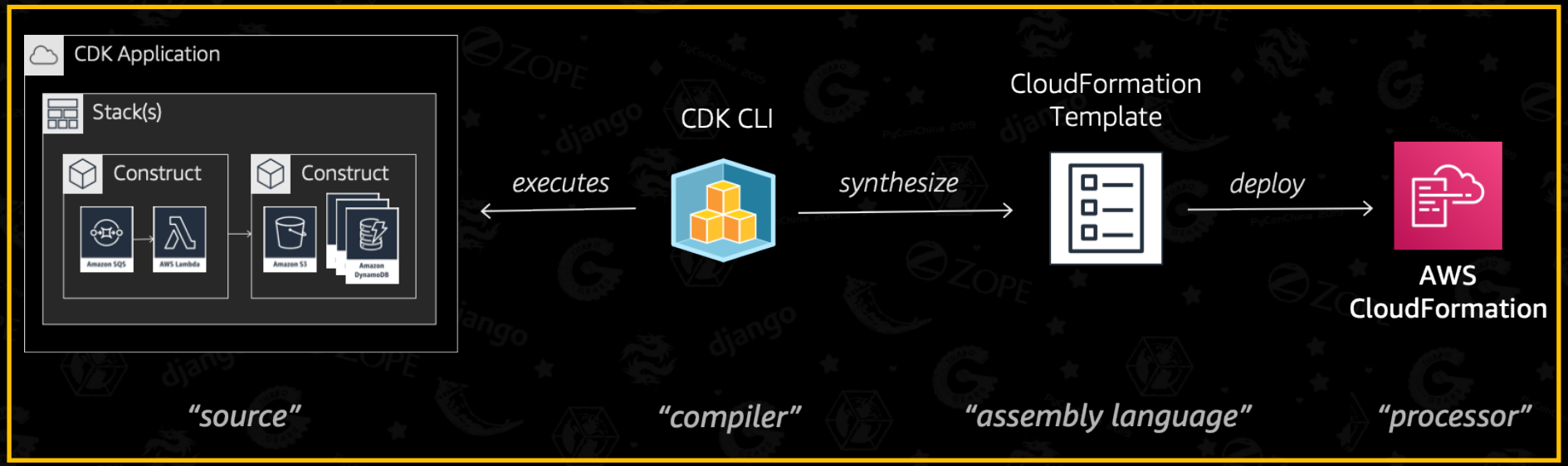
AWS CloudFormation

aws

PyConChina 2019

AWS Cloud Development Kit (AWS CDK)

The big picture—from AWS CDK app to provisioned infrastructure



Show me the Code

PyConChina 2019

Just two lines

And you got a serverless REST API in Python

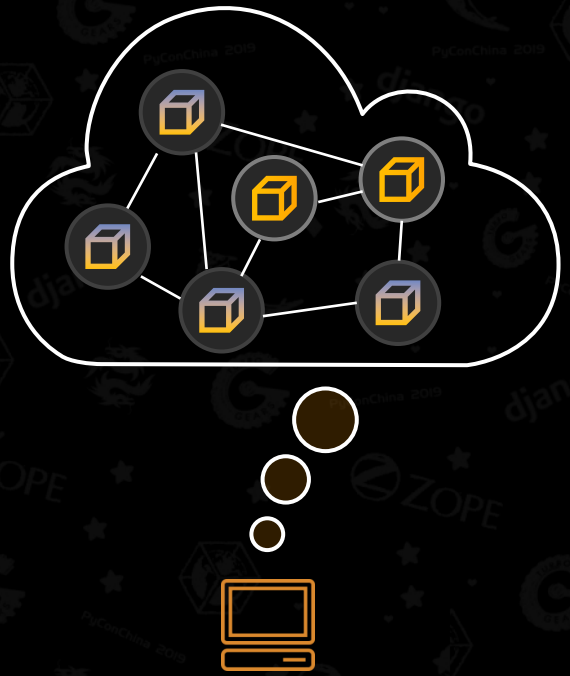
```
1   from aws_cdk import core, aws_apigateway, aws_lambda
2
3
4   class CdkPyServerlessRestApiStack(core.Stack):
5       def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
6           super().__init__(scope, id, **kwargs)
7
8           backend = aws_lambda.Function(self, 'Func',
9                                         code=aws_lambda.Code.from_asset('../function/hello-world'),
10                                        handler='lambda_function.handler',
11                                        runtime=aws_lambda.Runtime.PYTHON_3_7
12                                        )
13
14          api = aws_apigateway.LambdaRestApi(self, 'RestApi', handler=backend)
15
```

$ cdk deploy

RestApiEndpoint = https://ru2zb03qhb.execute-api.ap-northeast-1.amazonaws.com/prod/

aws

How do I edit and debug my serverless application code?



Author and debug using your favorite IDEs

		Developer Preview	Developer Preview
AWS Cloud9	AWS Toolkit for PyCharm	AWS Toolkit for IntelliJ	AWS Toolkit for Visual Studio Code
Python, Node	Python	Java, Python	.NET, Node

PyConChina 2019

Let’s Build together with Python today!

aws

PyConChina 2019

Workshop Today:
14:30-17:30
在 AWS 云上部署与发布你面向全球的 Python Serverless 应用

14:30-17:30

AWS Work Shop：在AWS部署与发布你面向全球的Python Serverless应用

学习与开发了一整天的Python，我该如何封装我的应用面向全球发布呢?

在这三小时的workshop里面，我们将会带你一步一步发布你的Python应用到无服务器环境成为全球开发者皆可使用的AWS Lambda Layer，并且介绍如何用最新的AWS CDK(Cloud Development Kit)来封装你的应用发布到无服务器容器环境(AWS Fargate)、无服务器函数环境(AWS Lambda)以及全受管的Kubernetes环境(Amazon EKS)。

1. AWS Serverless最新功能介绍，包括AWS Lambda Layer, AWS Lambda Custom Runtime, AWS Serverless App Repository等
2. 封装与发布你的Python Library成为AWS Lambda Layer并且面向全球发布
3. 进一步封装你的Python核心应用与Layer并且发布到AWS SAR(Serverless App Repository)
4. 生成与发布你的SAR Buttons提供全球用户一键部署
5. AWS CDK介绍
6. 在你自己熟悉的IDE运行AWS CDK in Python
7. 使用AWS CDK in Python来开发一个无服务器端网址应用
8. 使用AWS CDK 来开发与部署你的无服务器容器应用
9. 使用AWS CDK 来快速部署你的Amazon EKS(Elastic Kubernetes Service)应用
10. Q&A

aws

PyConChina 2019

hunhsieh

Pahud Hsieh

THANK YOU

pahudnet

@亚马逊 AWS-Pahud

github.com/pahud

@pahudnet

aws