基于AWS平台的DevOps实践

张孝峰

AWS中国解决方案架构师

议题

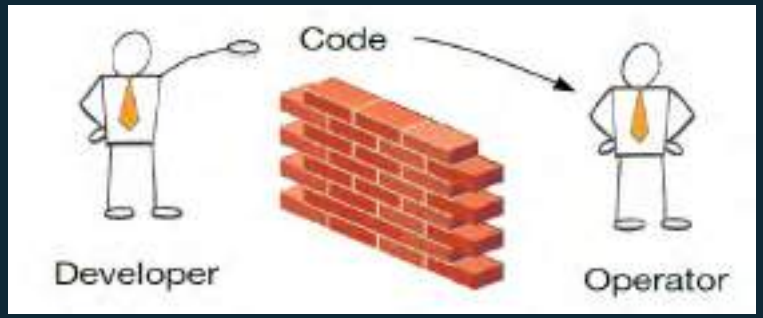
关于DevOps

用AWS实现DevOps的框架和工具

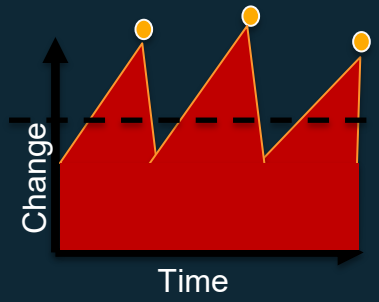
- CodeCommit
- CodePipeline
- Codebuild
- CodeDeploy
- CloudFormation
- Beanstalk
- OpsWorks
- Cloudwatch
- CloudTrail
- API Gateway
- Lambda
- ECS

关于 Devops.....

因为我们不希望
事情是这样的...

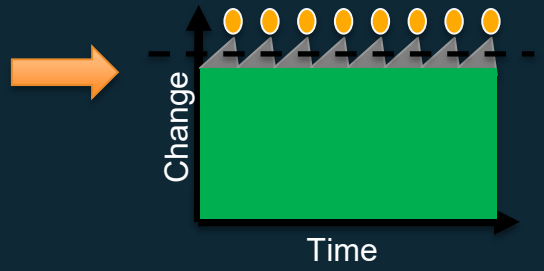


瀑布式开发，版本发布少



我们希望...

快速迭代, 敏捷开发



DevOps 重要内容

基础设施即代码

IT自动化和配置管理

版本控制的集成

持续集成和持续交付

持续部署

应用和基础设施的版本管理

监控和日志管理

.....

亚马逊的Devops故事

你眼中的亚马逊集团

2003: 50亿美元

2004: 70亿美元

2016:1360 亿美元

AWS中国（北京）区域由光环新网运营

aws


亚马逊开发流程的演进

传统应用发布的四个阶段： 冗长的周期和复杂的协作

源码 → 编译 → 测试 → 生产

- 写入源代码，例如.java文件。
- 新代码的结对检查。

- 编译代码
- 单元测试
- 风格检查
- 代码检测
- 创建容器镜像

- 和其他的系统做兼容性测试
- 压力测试
- 用户界面测试
- 攻击测试

- 把应用发布到生产环境




亚马逊开发形式的转变: 2001-2009

2001

Merchants.com

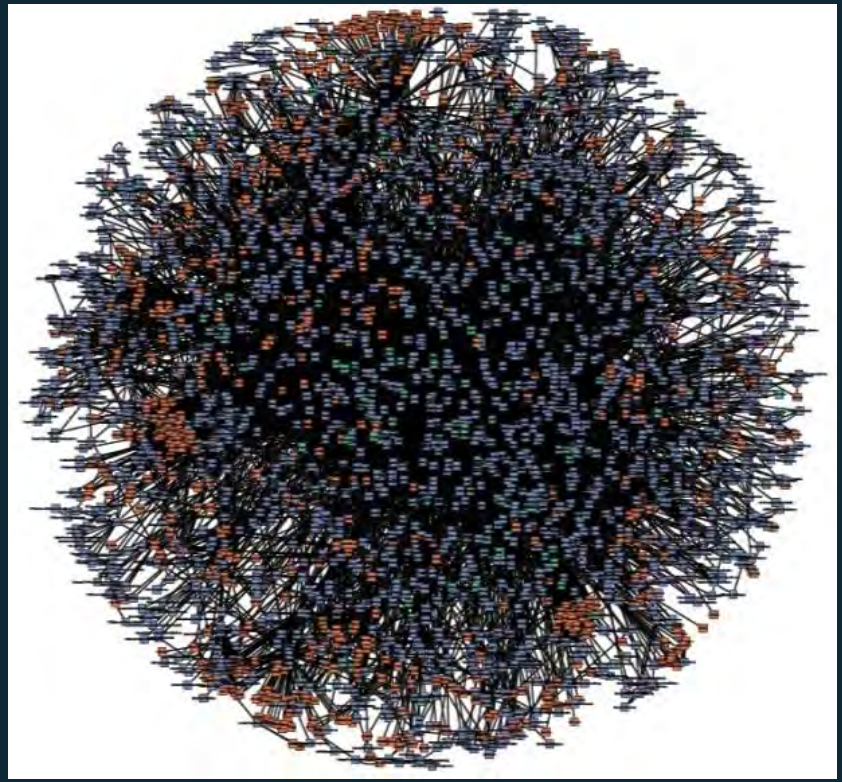
一个巨大的应用程序
+团队

2009

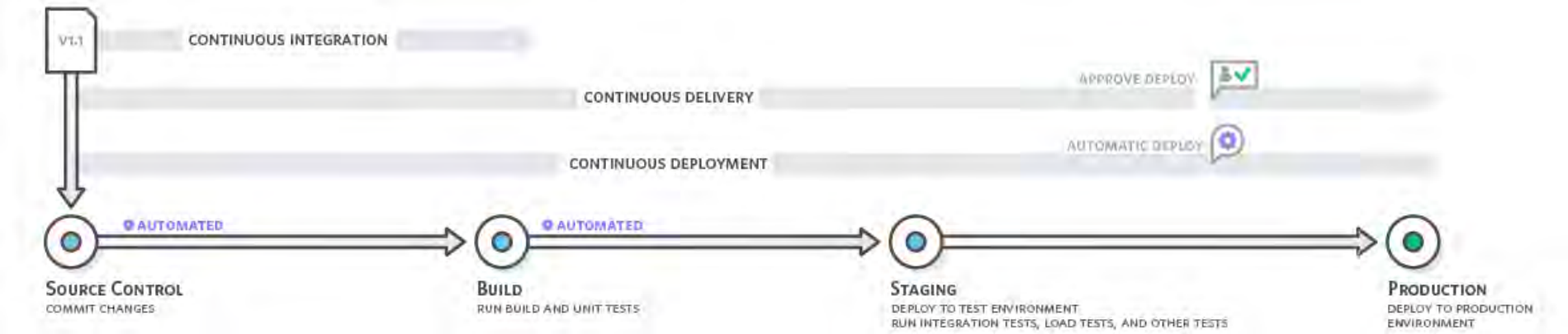
众多的微服务 + 小团队（2 pizza teams）

微服务MicroServices

- 面向服务的架构
- 单一目的
- 通过APIs 连接
- 高度解耦



软件生命期新的管道



- 持续集成
- 持续交付
- 持续发布

Amazon 开发团队最佳实践

在2014年：

数千个团队

× 微服务架构

× 持续交付

× 多环境部署

= 5千万次部署/年

11.6s	1,079	10,000	30,000
部署之间的平均间隔时间(工作日)	一个小时中最多部署次数	平均同时接受部署的EC2服务器数量	最多同时部署的服务器的数量

变革移动终端带来的变革：软件的交付方式被彻底改变

旧的软件交付方式

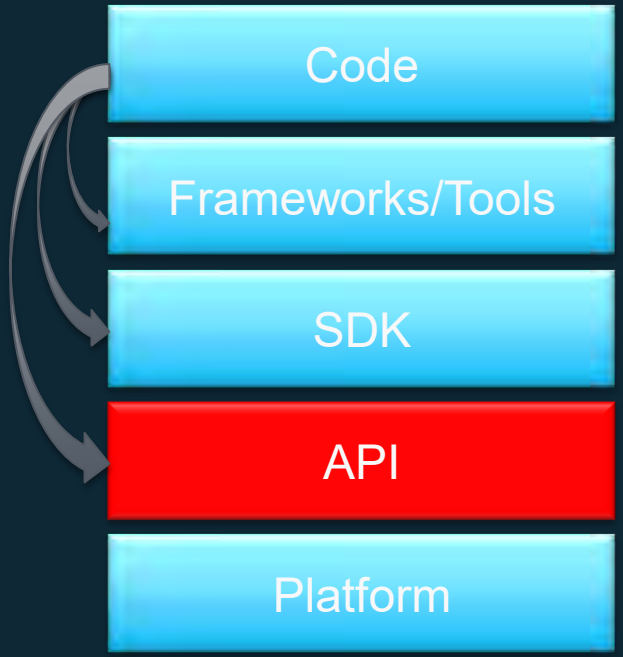
新的软件交付方式

面对变化：

是粗暴拓展基础设施？
还是改变企业组织协作模式？

基于AWS服务实现DevOps的框架和工具

AWS对DevOps的全面支持



Programs, CLI, Scripts, DML, …

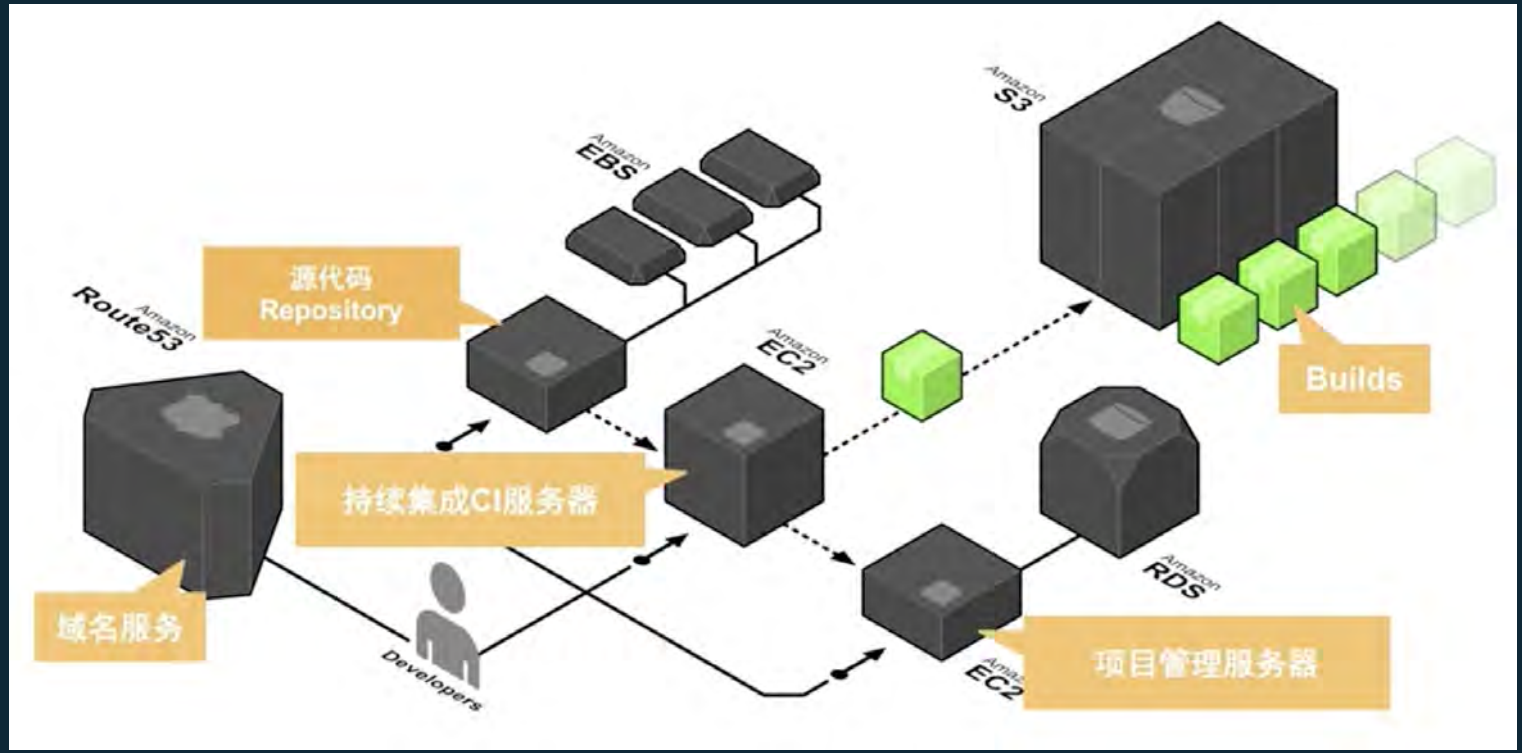
Python, Java, PHP, Node.js,
PHP, Ruby, GO, …

REST API 

基础服务 

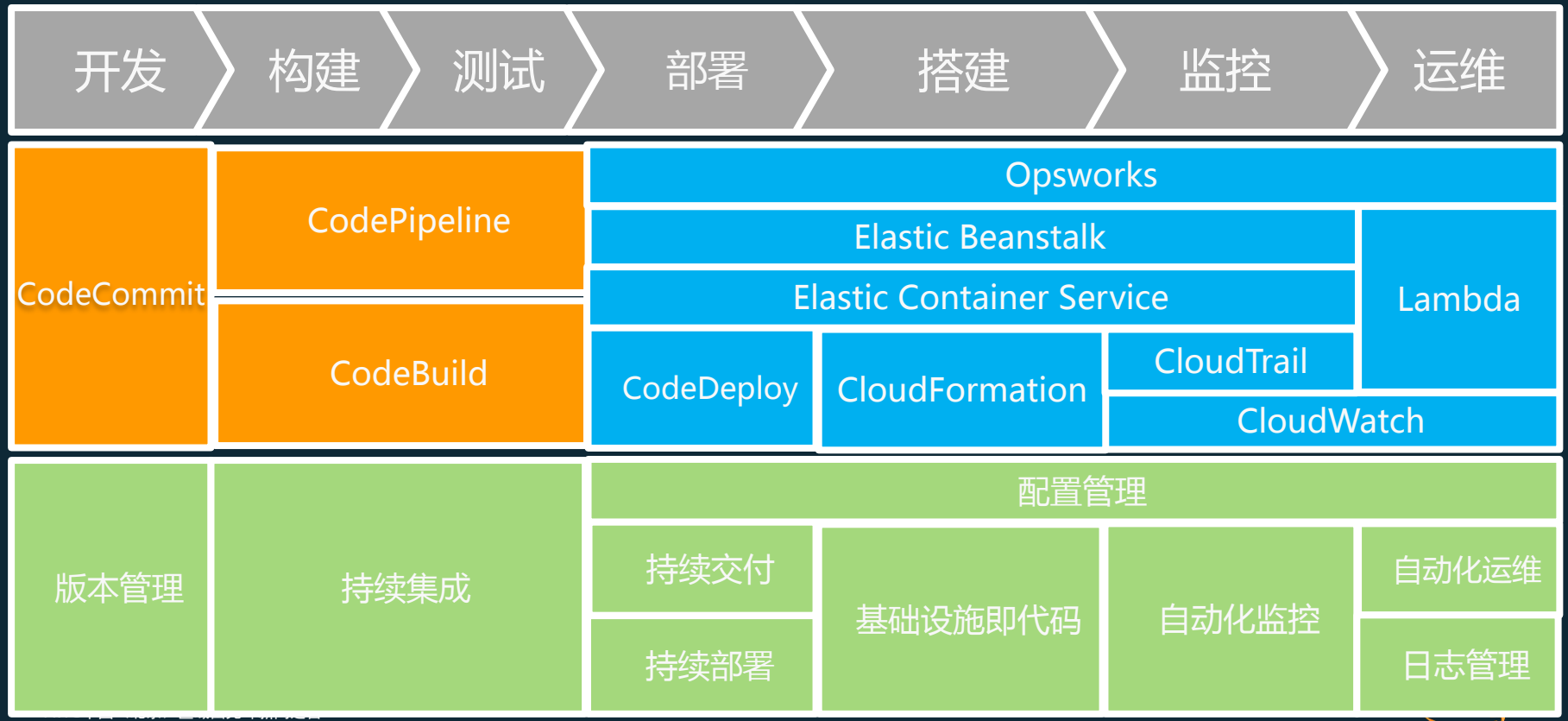
建立自动交付渠道

持续集成模型


Amazon S3
Amazon EBS
源代码 Repository
Amazon Route53
Amazon EC2
Builds
持续集成CI服务器
域名服务
Developers
Amazon RDS
项目管理服务器
Amazon EC2

AWS CodeCommit

AWS DevOps 服务

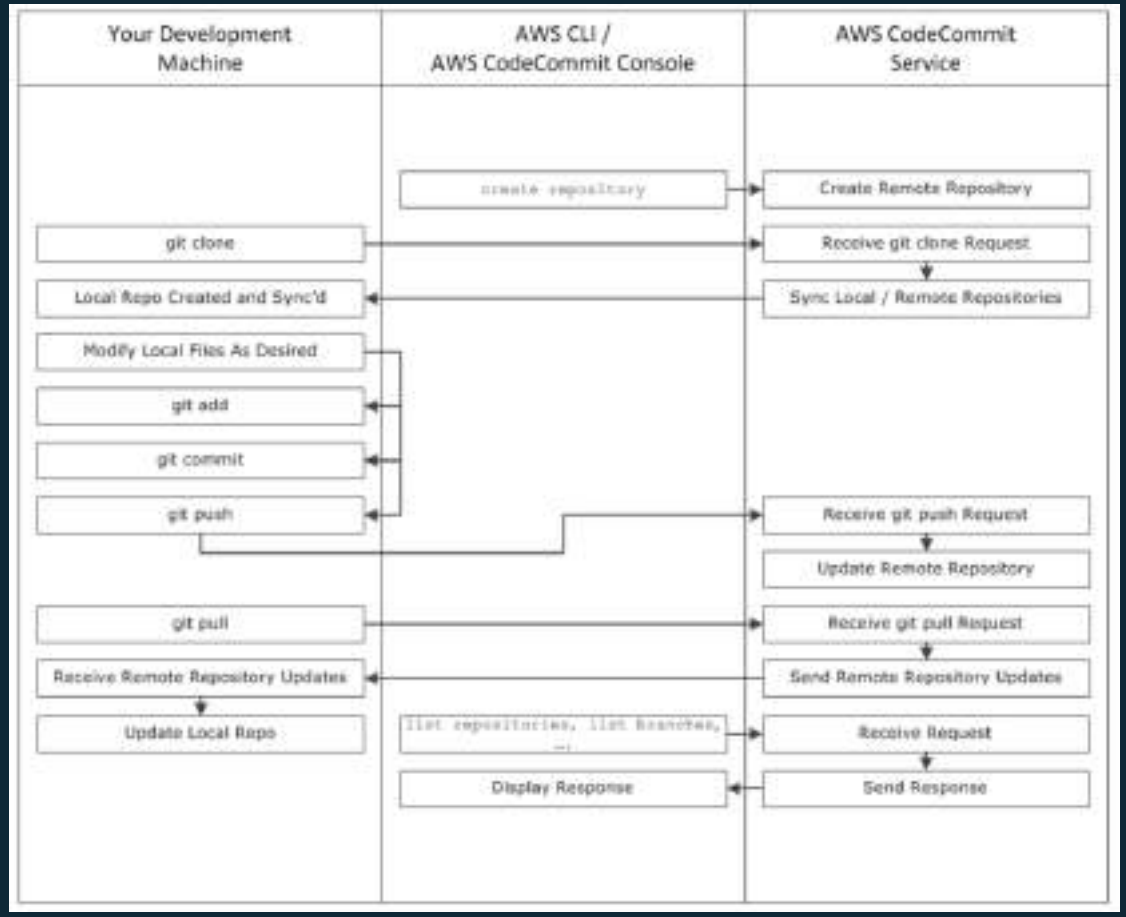


AWS CodeCommit

可扩展的托管型源代码控制服务，托管私有的 Git 存储库

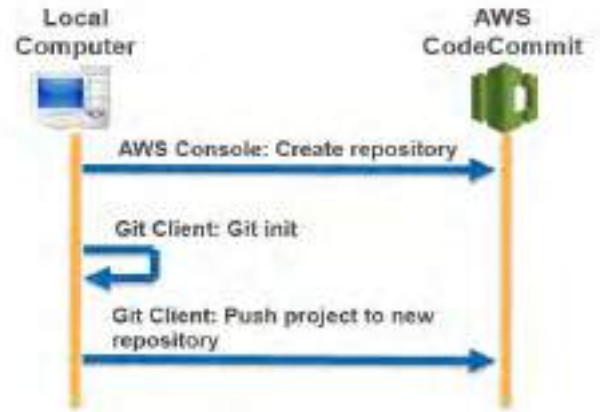
- 用于托管私有 Git 存储库的安全、高度可扩展的托管源代码控制服务。
- 版本控制项目, 能够处理具有大量文件或分支、大尺寸文件及冗长版本历史记录的存储库.
- 可以存储从代码到二进制文件的一切内容。
- 支持 Git 的标准功能，可与现有的基于 Git 的工具无缝协作。
- 具有Amazon S3可扩展，高可用，高持久
- 数据静止时可用客户专用秘钥加密

AWS CodeCommit 使用方式

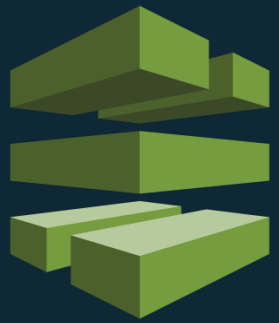


迁移到codecommit

- 将本地或非版本控制内容迁移到 AWS CodeCommit

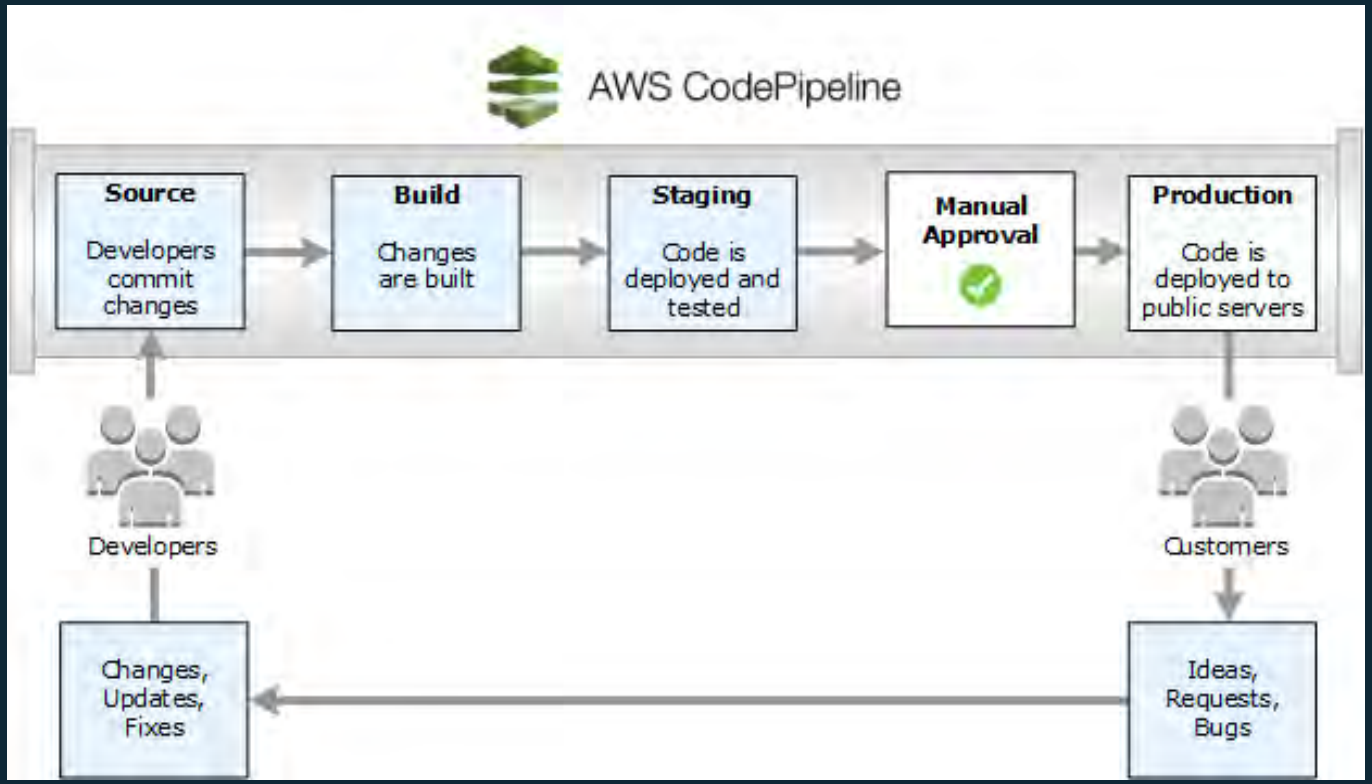


- 将Git 存储库迁移到 AWS CodeCommit



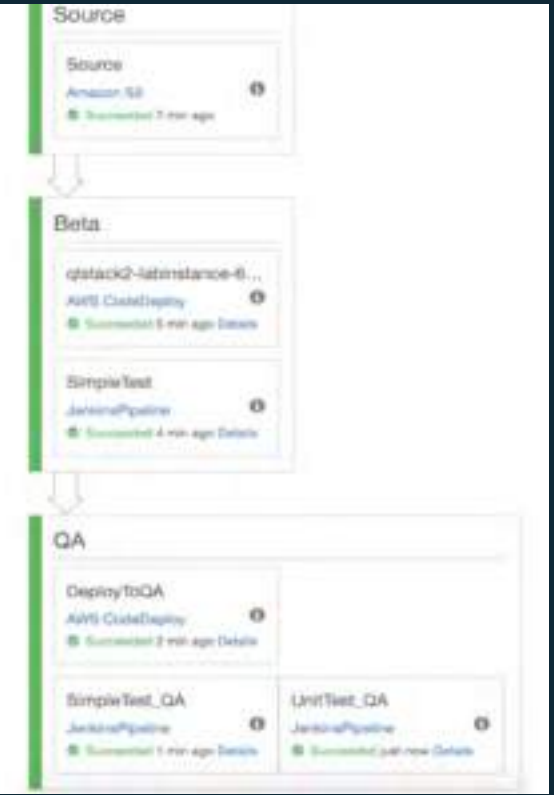
AWS CodePipeline

AWS CodePipeline管道流程



持续交付 - CodePipeline

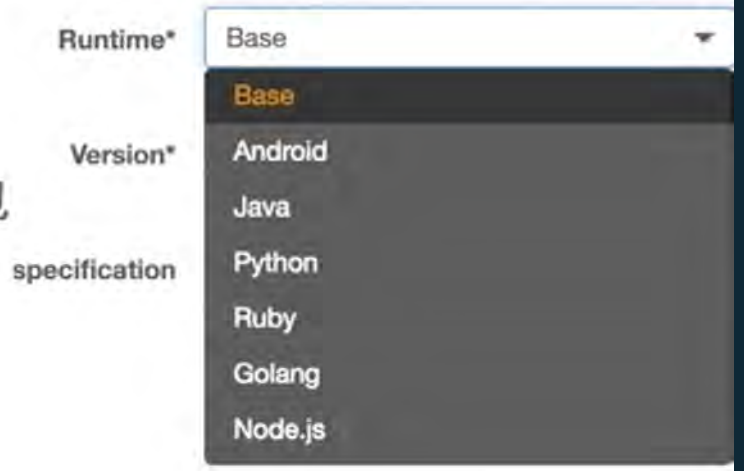
- 可自定义的自动化版本发布，并且集成了编译和测试
- 对自定义的版本发布工作流建模、可视化
  (源代码 ➔ 编译 ➔ beta ➔ gamma ➔ 线上生产)
- 自动化编译、测试和部署
- 执行自定义规则
- 与第三方工具集成



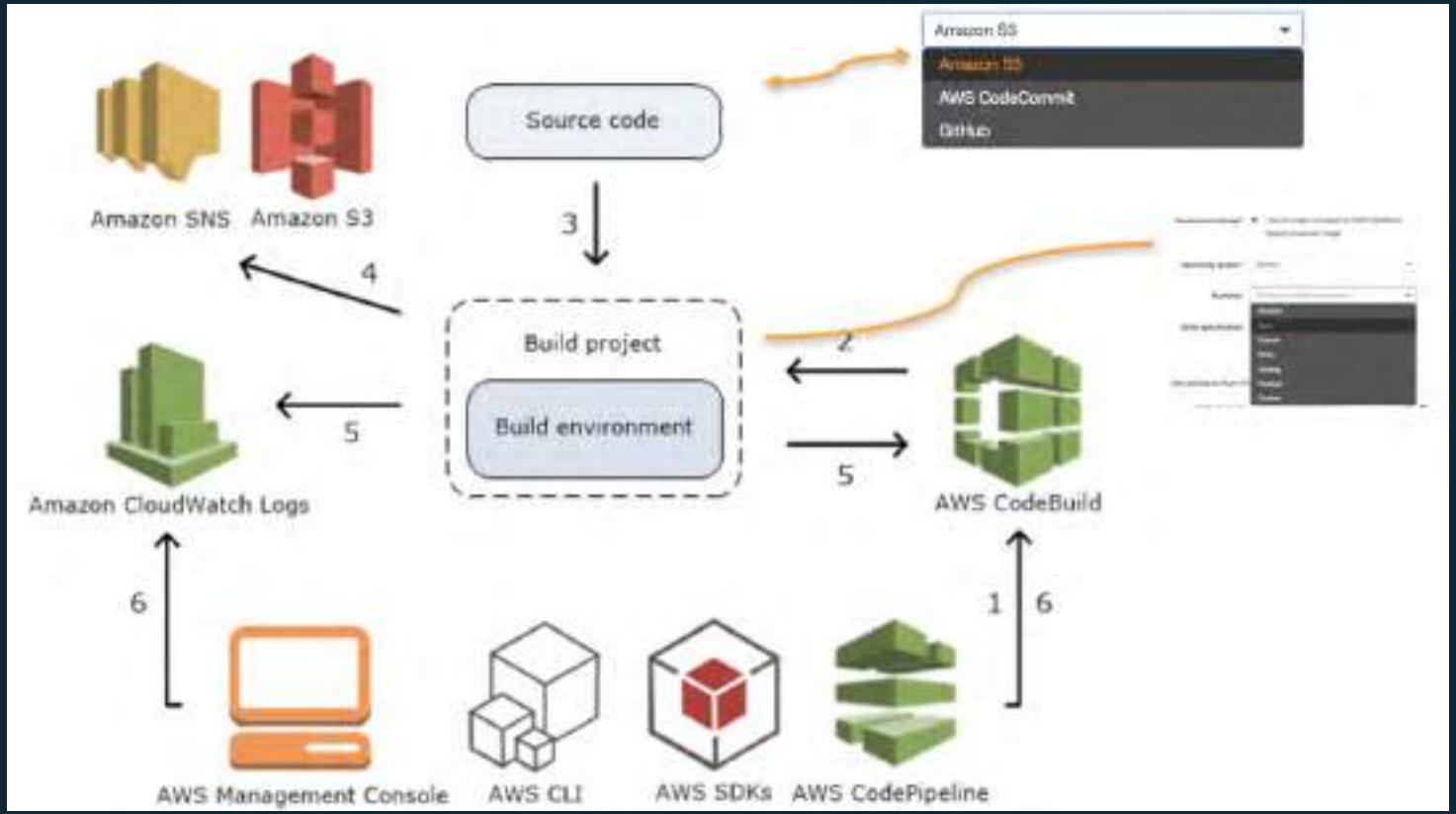
AWS CodeBuild

AWS CodeBuild

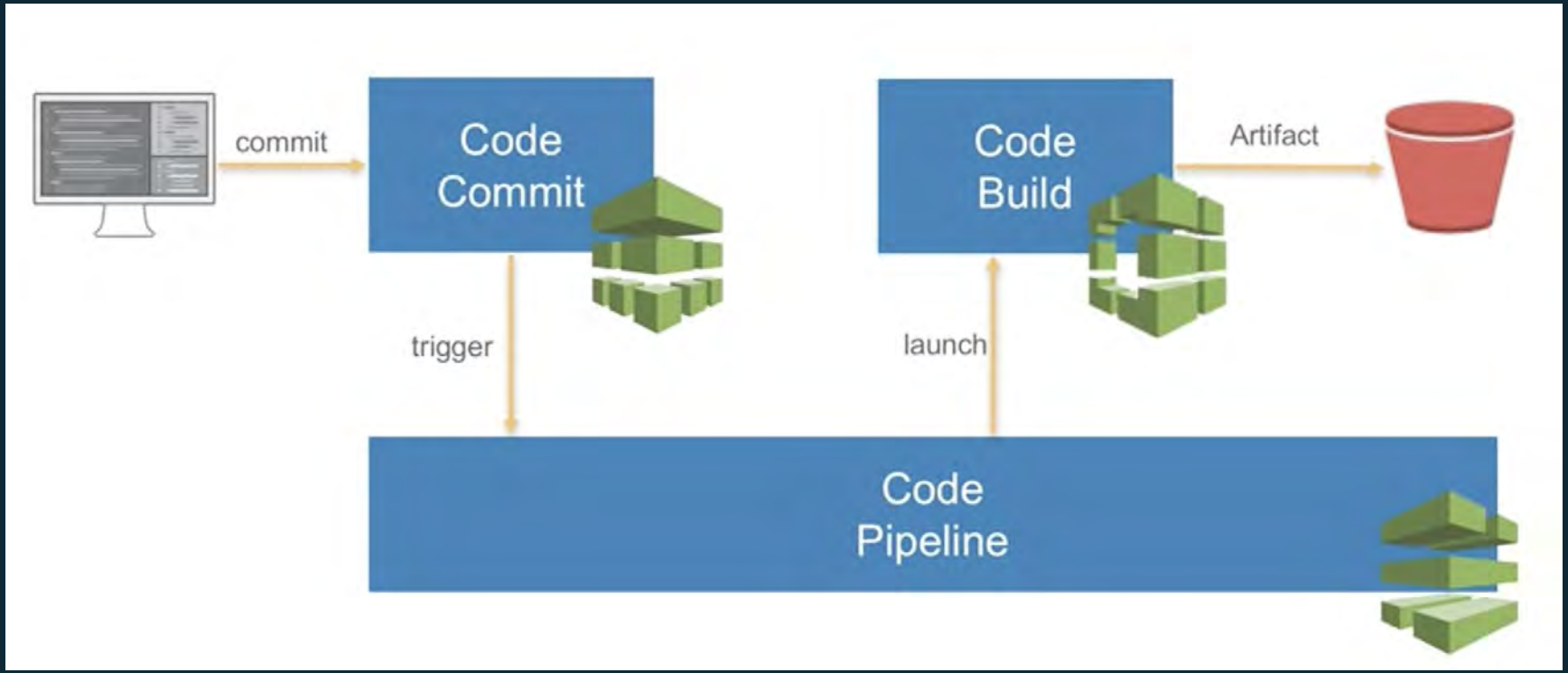
- 2016年Re:Invent推出的新服务
- 全托管的云中Build服务
  - 编译代码，进行测试，生成软件包
- 无需从头搭建Build服务器
- 使用容器技术，build结束即销毁
- 支持代码库
  - CodeCommit, GitHub, S3



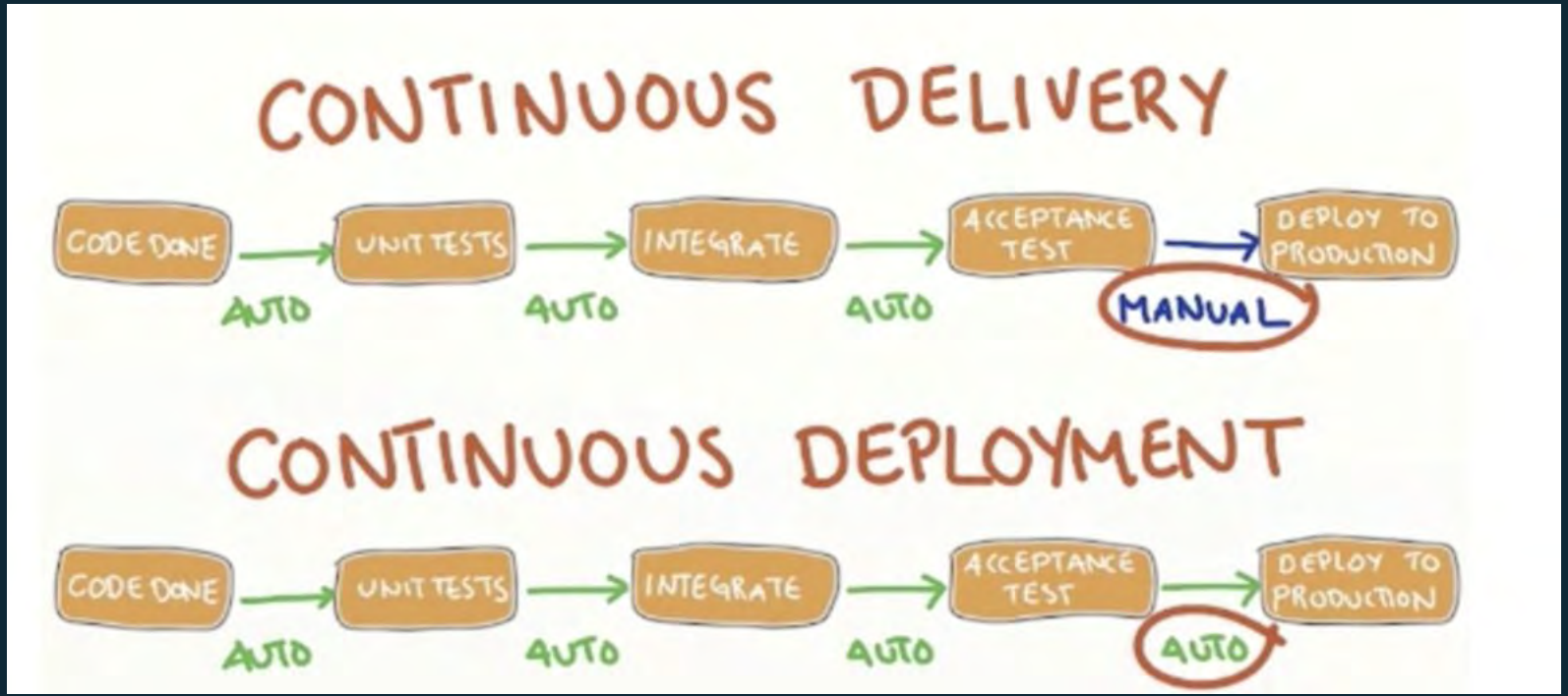
AWS CodeBuilde 工作流程



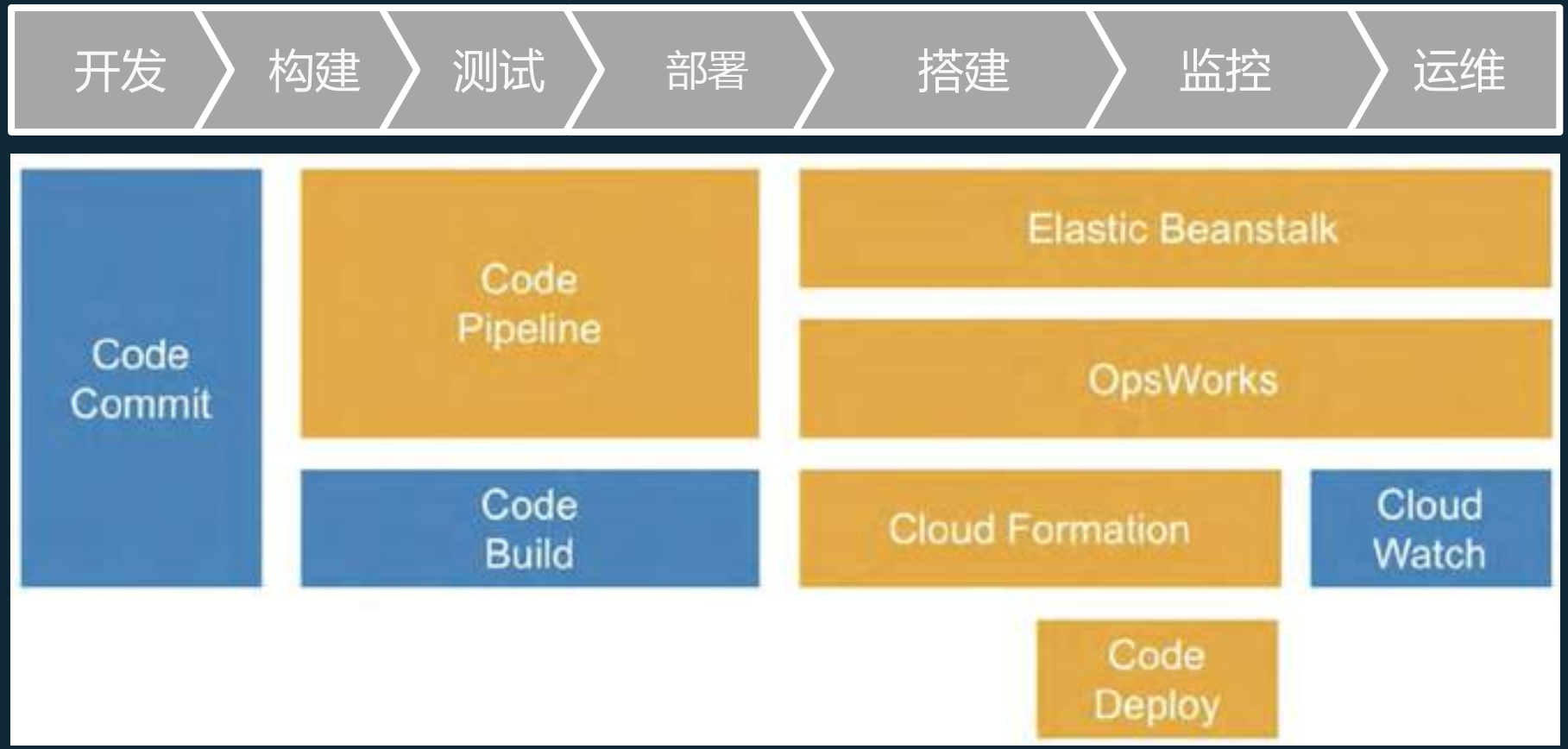
持续集成


commit
Code
Commit
Code
Build
Artifact
trigger
launch
Code
Pipeline

持续交付与持续部署的区别



AWS DevOps 服务



AWS CodeDeploy

AWS CodeDeploy

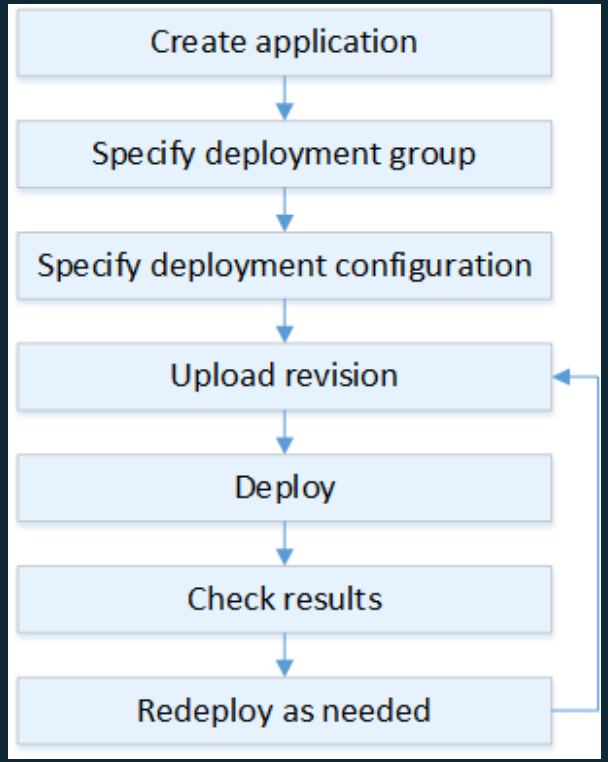
提供功能包括滚动更新、部署健康监测、回滚、集中化的管理、
历史检索。支持的语言包括JS、Python、Java、C++、Shell脚本，
可集成原有的工具链。

将代码快速部署到任何EC2上的环境
自动, 可复用且提供完全管理的服务
减少部署或应用错误带来的停机时间

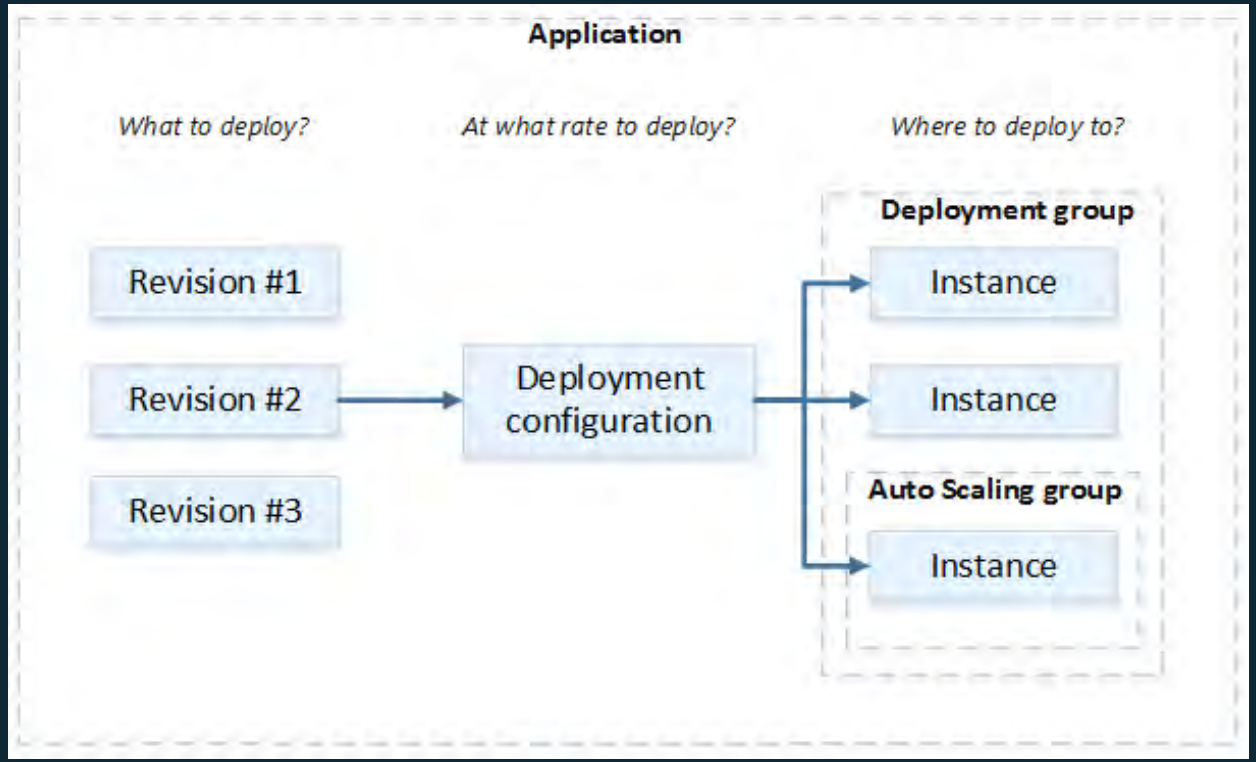
从单个实例扩展到上千个实例
部署到任何服务器上: AWS or on-premises
中央集中控制和监控
代码滚动升级

WS CodeDeploy 部署流程与组件关联

部署流程



组件关联



1) 打包 app　2) 指定部署目标　3) 部署!

```
version: 0.0
os: linux
files:
  - source: chef/
    destination: /etc/chef/codedeploy
  - source: target/hello.war
    destination: /var/lib/tomcat6/webapps
hooks:
  ApplicationStop:
    - location: deploy_hooks/stop-tomcat.sh
  BeforeInstall:
    - location: deploy_hooks/install-chef.sh
    - location: deploy_hooks/chef-solo.sh
  ApplicationStart:
    - location: deploy_hooks/start-tomcat.sh
  ValidateService:
    - location: deploy_hooks/verify_service.sh
```

1) 打包 app　2) 指定部署目标　3) 部署!

将实例按下述编组:

- 弹性扩展组(Auto-scaling group)
- Amazon EC2标签
- On-premises标签

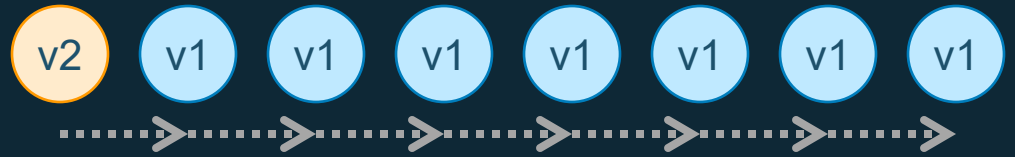
1) 打包 app　2) 指定部署目标　3) 部署!

AWS CLI & SDKs
AWS Console
CI / CD Partners
GitHub

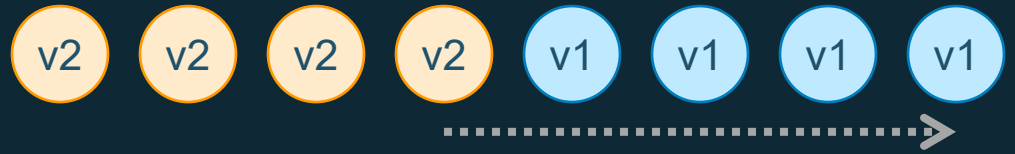
```
aws deploy create-deployment \
--application-name MyApp \
--deployment-group-name TargetGroup \
--s3-location
bucket=MyBucket,key=MyApp.zip
```

部署方式

一次一台
最少健康主机数目 = 99%

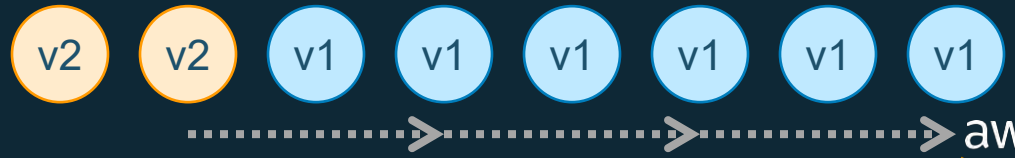


一次一半
最少健康主机数目 = 50%

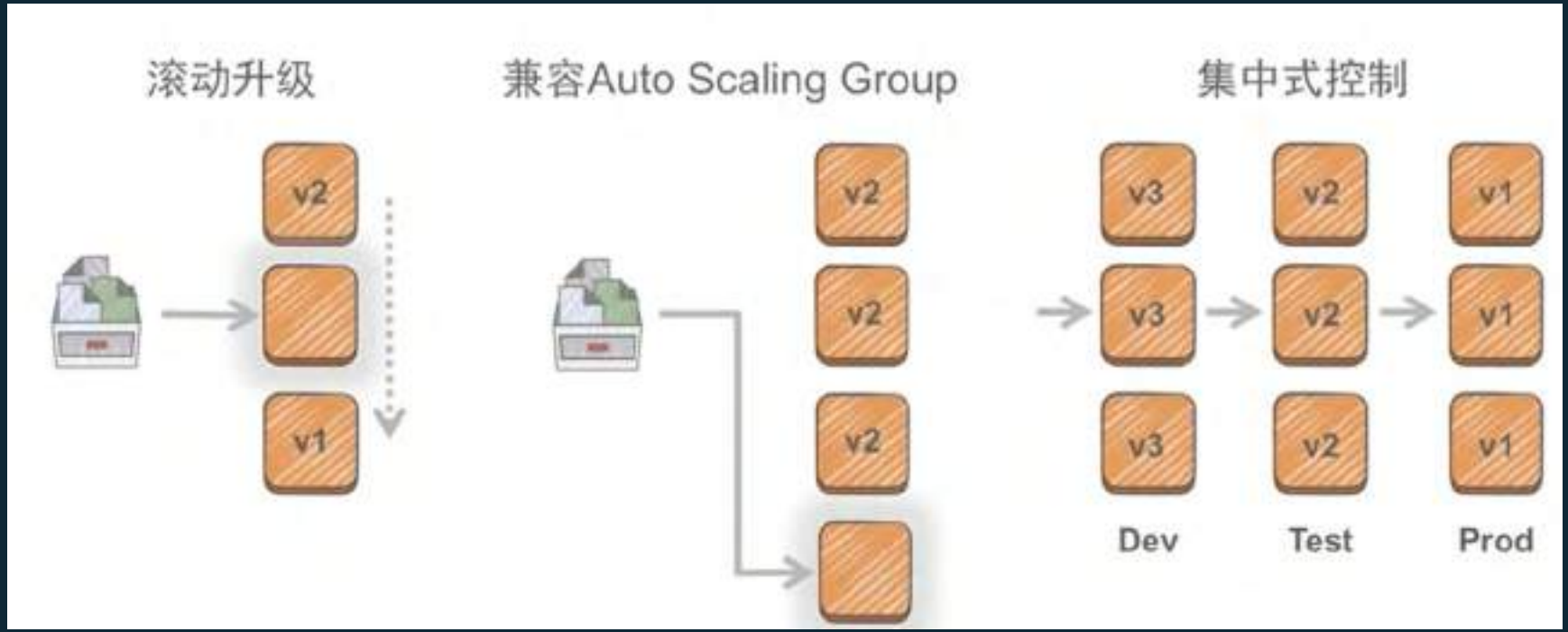


一次全部
最少健康主机数目 = 0

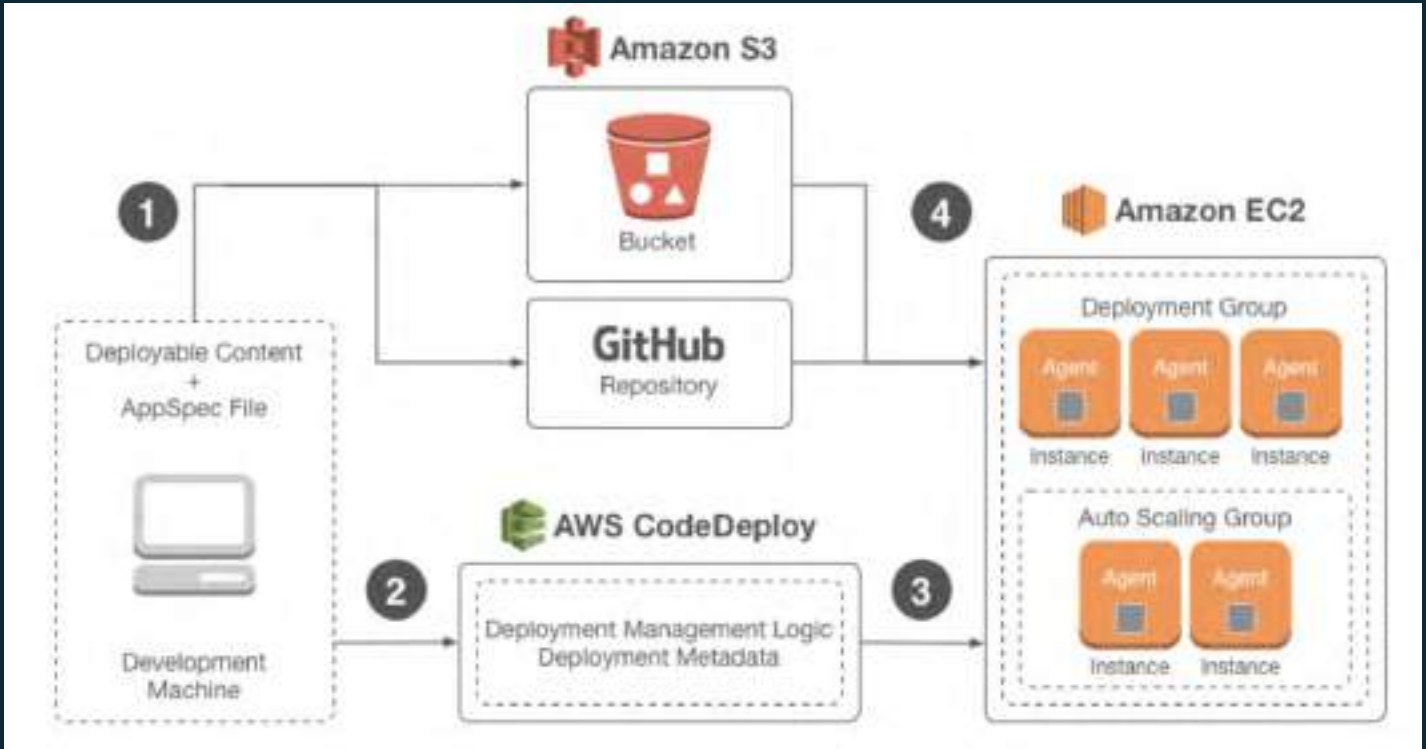
[其他定制方式]
最少健康主机数目 = 75%



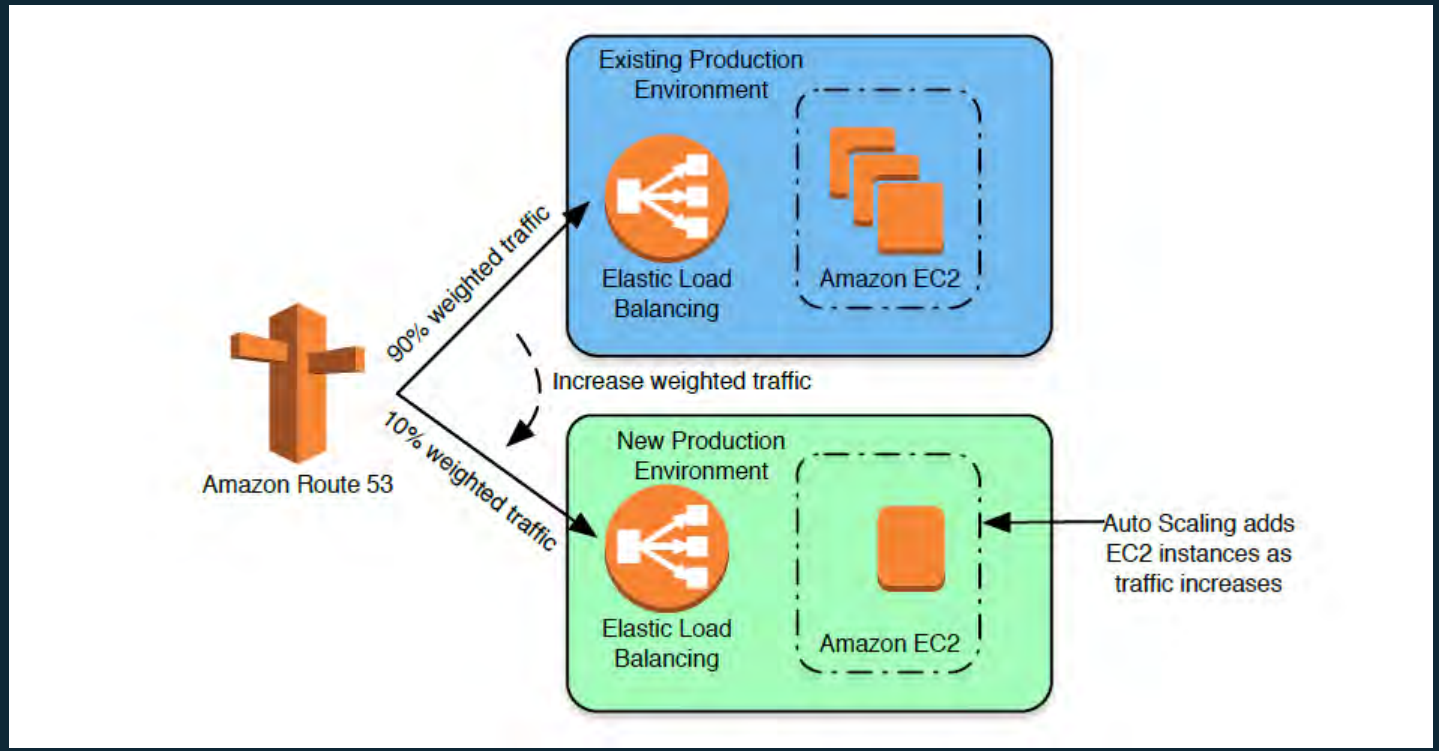
Codedeploy如何工作



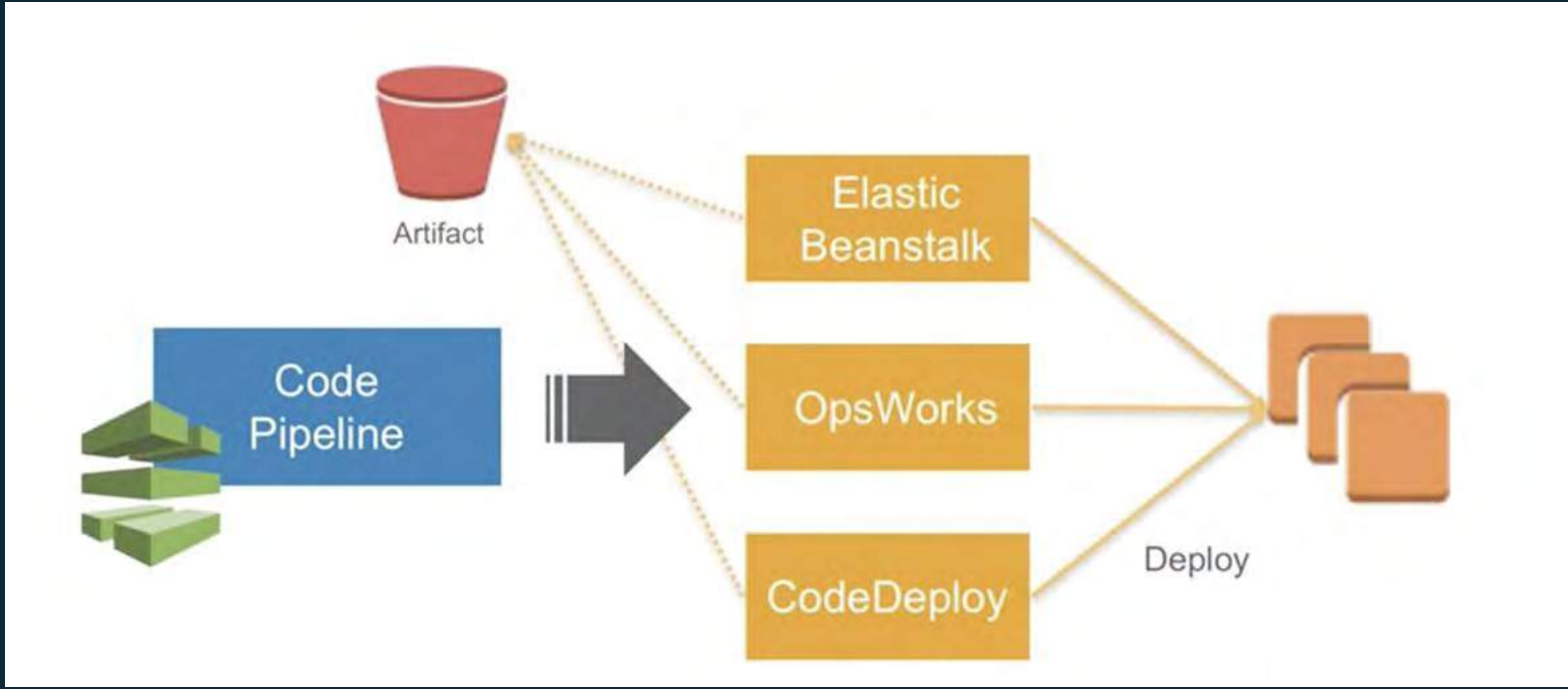
CodeDeploy就地部署



蓝绿部署

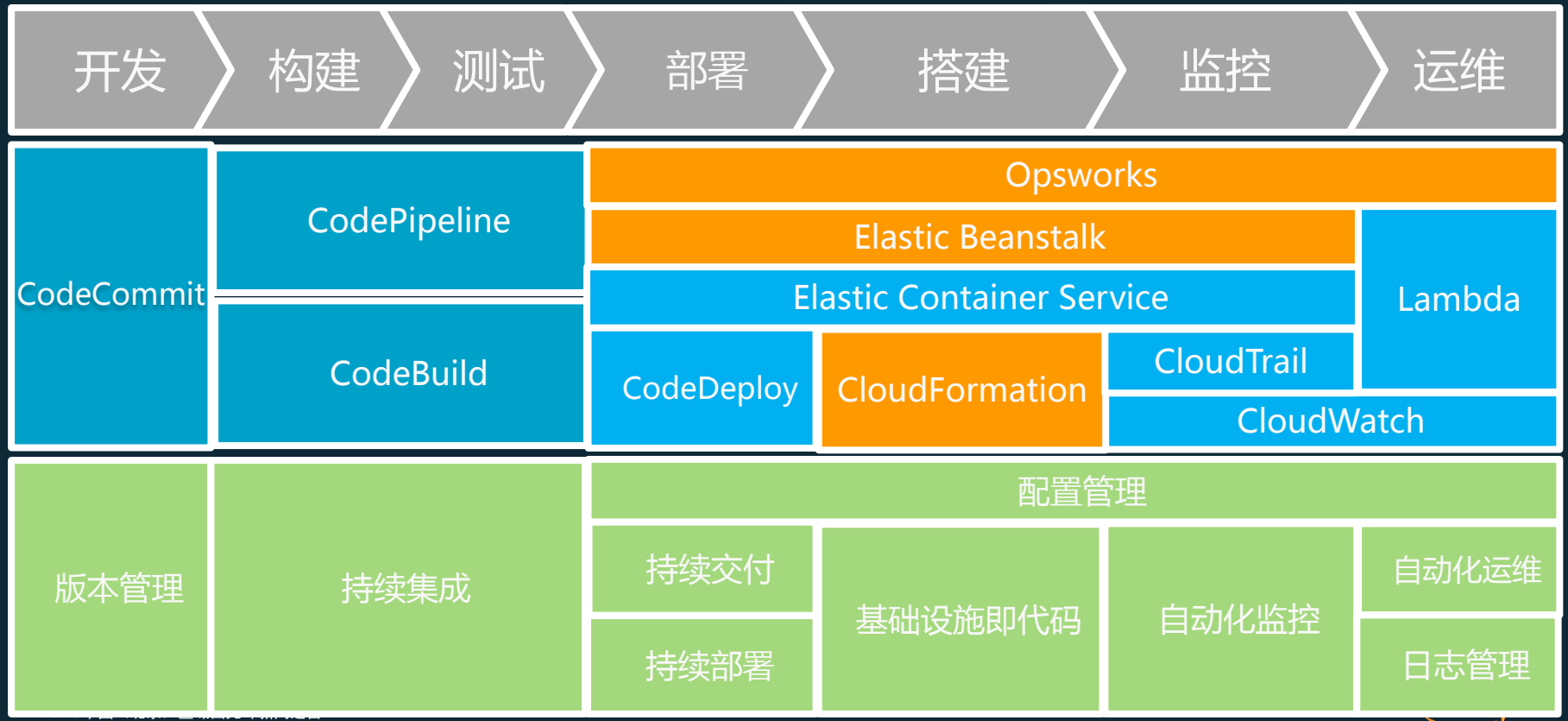

Existing Production Environment
Elastic Load Balancing
Amazon EC2
90% weighted traffic
Increase weighted traffic
10% weighted traffic
Amazon Route 53
New Production Environment
Elastic Load Balancing
Amazon EC2
Auto Scaling adds EC2 instances as traffic increases

持续部署


Artifact
Elastic Beanstalk
Code Pipeline
OpsWorks
CodeDeploy
Deploy

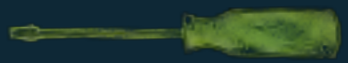
基础架构代码化

AWS DevOps 服务



操作AWS服务的三种方式

Web控制台



APIs and SDKs

命令行

DevOps基础

AWS Tools (SDK, CLI, IDE, etc.): http://aws.amazon.com/tools/

一切皆是API


```
{
    "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.",
    "Parameters" : {
        "KeyPair" : {
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",
            "Type" : "String"
        }
    },
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "KeyName" : { "Ref" : "KeyPair" },
                "ImageId" : "ami-75g0061f",
                "InstanceType" : "m1.medium"
            }
        }
    },
    "Outputs" : {
        "InstanceId" : {
            "Description" : "The InstanceId of the newly created EC2 instance",
            "Value" : { "Ref" : "Ec2Instance" }
        }
    }
}
```

AWS Tools (SDK, CLI, IDE, etc.): http://aws.amazon.com/tools/

Elastic Beanstalk

基于AWS的基础设施层面DevOps实践要素

高级服务

自己动手

Elastic Beanstalk

OpsWorks

CloudFormation

代码/命令行

便利

IAM

CloudWatch

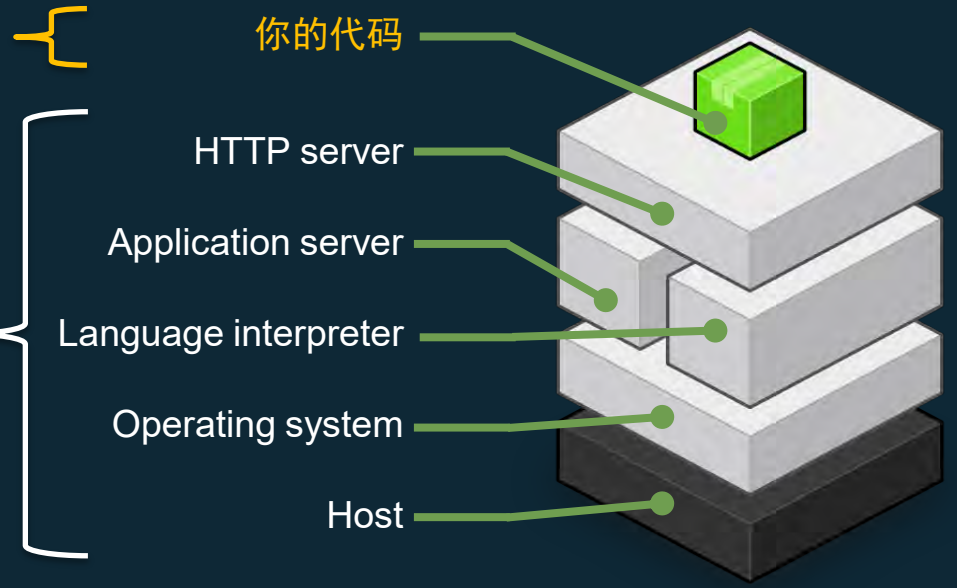
控制

aws

Elastic Beanstalk vs. DIY

聚焦于构建自己的应用

Elastic Beanstalk负责配置EC2 实例，包括应用运行的目标平台所需要的其他组件，用户不需要登录到实例里面安装和配置应用堆栈



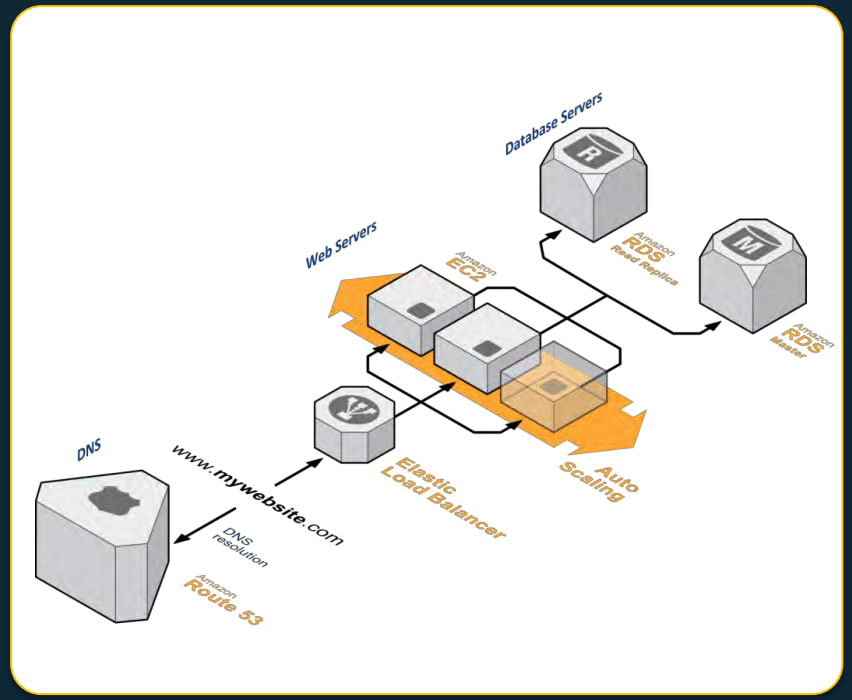
用户提供

Elastic Beanstalk提供

如何部署应用

1. 通过AWS Management Console
2. 通过AWS Toolkit for Eclipse and Visual Studio IDE
3. 使用EB命令行接口(EB CLI)

```
$ eb deploy
```



部署应用需要提供的信息

你的代码

01 Region

02 堆栈（容器）类型

03 单实例e 或 负载均衡+弹性扩展组

04 数据库(RDS) 可选

Elastic Beanstalk构建不同版本环境

Cloudformation

CloudFormation – 组件和技术实现

模板

JSON 格式的文件

参数定义

需要的资源

具体配置

CloudFormation

框架

创建堆栈

更新堆栈

错误检查和回滚

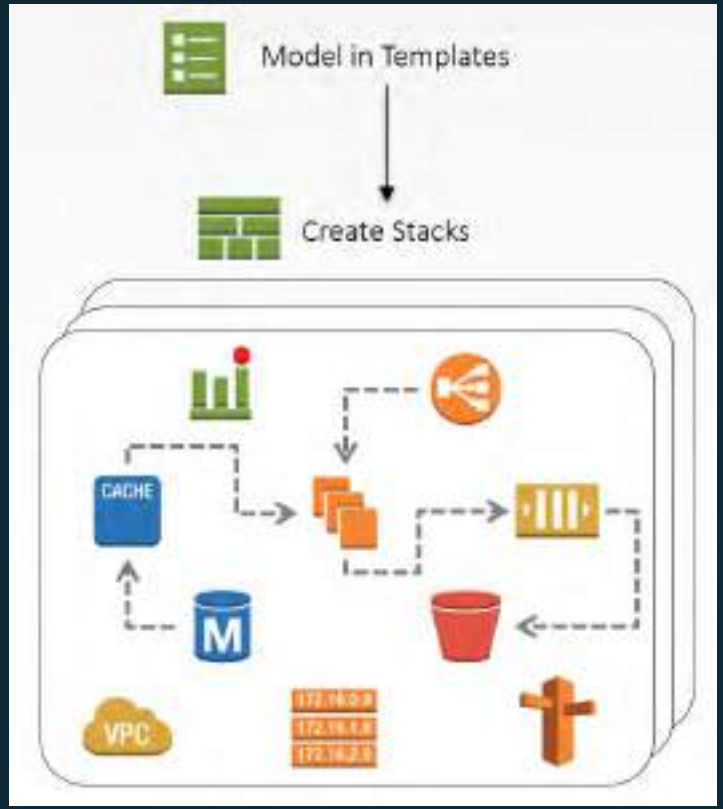
堆栈

配置好的AWS资源

完整的SAWS服务支持

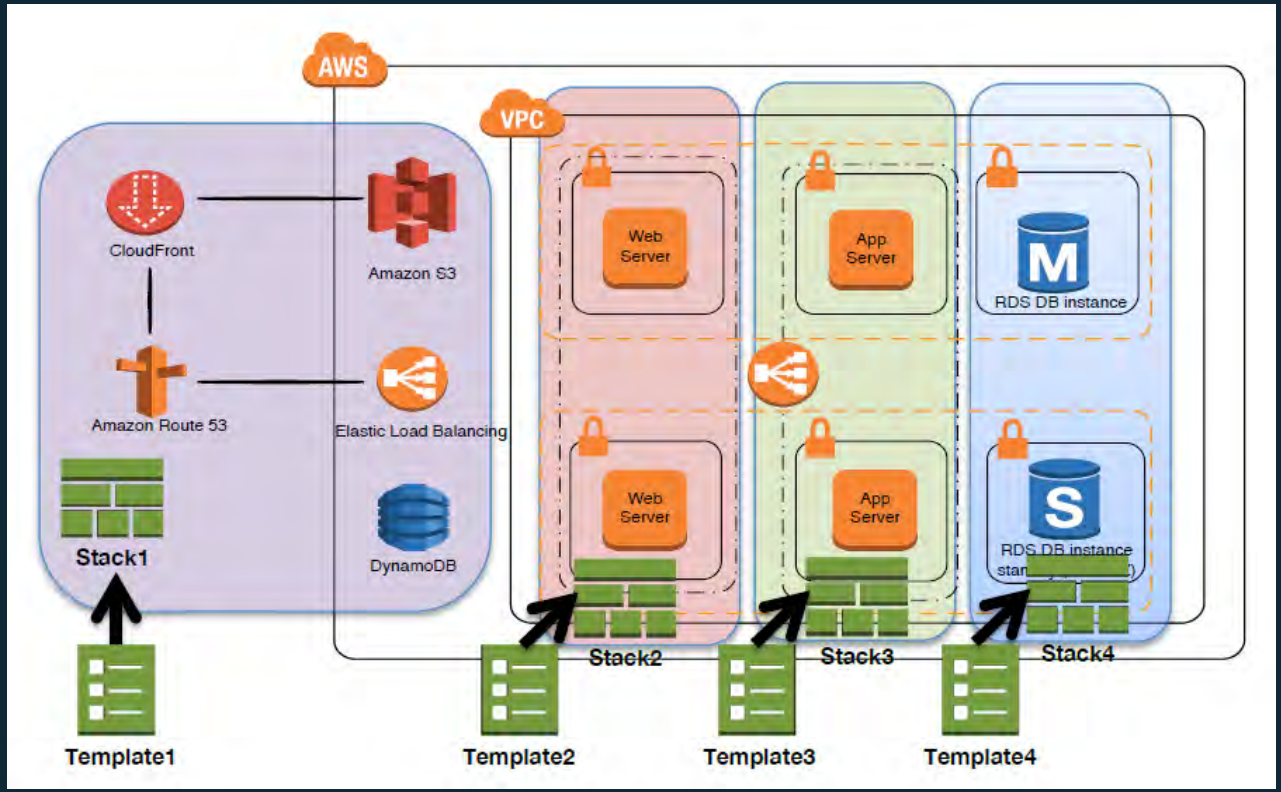
可定制化

基础平台模板化



- 简化AWS服务的部署，快速部署一个Stack
- 形成基础设施的模板
- CloudFormation自动解决资源部署的先后依赖关系
- 版本控制、拷贝、更新模板
- 第三方管理工具可以通过API集成CloudFormation

基于模板的快速部署

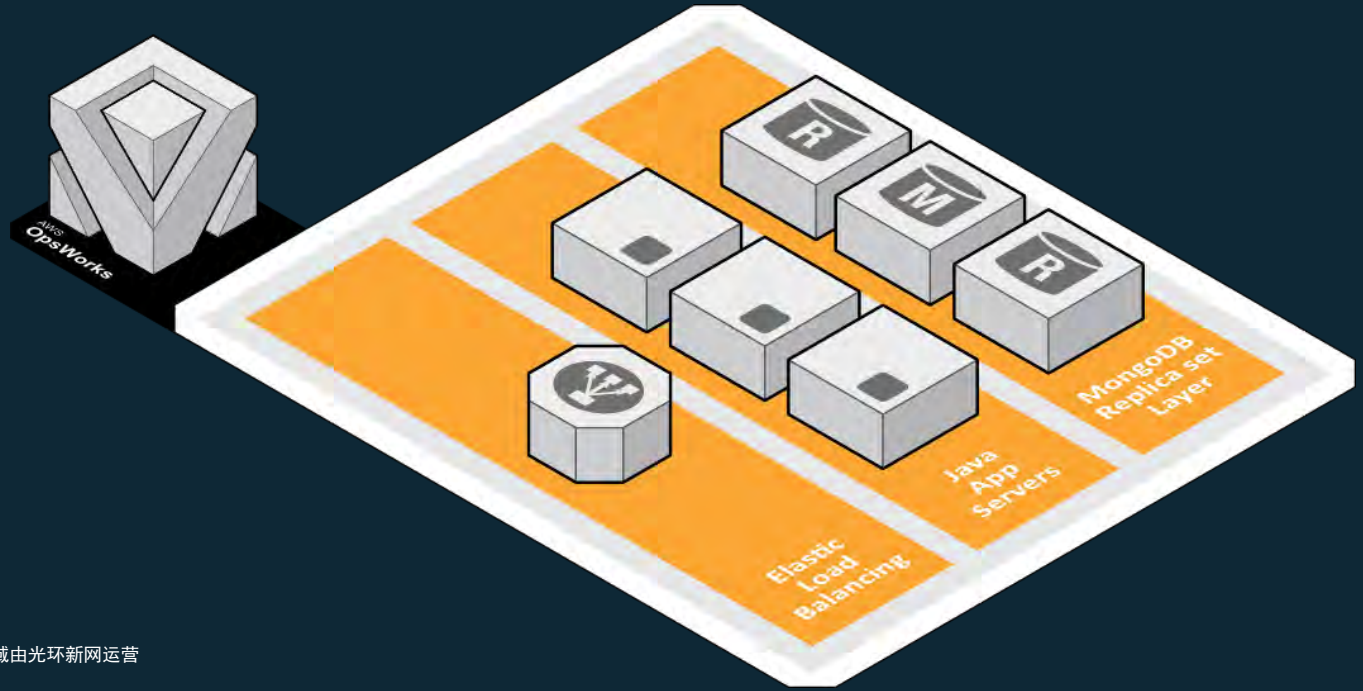

AWS
VPC
CloudFront
Amazon S3
Amazon Route 53
Elastic Load Balancing
DynamoDB
Stack1
Web Server
App Server
RDS DB instance
Web Server
App Server
RDS DB instance
Stack2
Stack3
Stack4
Template1
Template2
Template3
Template4

Cloudformation 一键部署全站

Opsworks

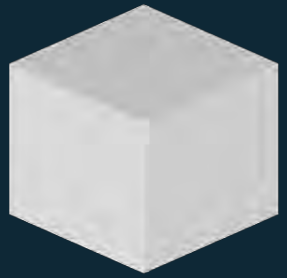
OpsWorks

灵活的应用管理解决方案，提供自动化工具来建模和管理应用，以及支撑应用运行所需要的可扩展的复杂的基础设施。



基于Chef的配置管理

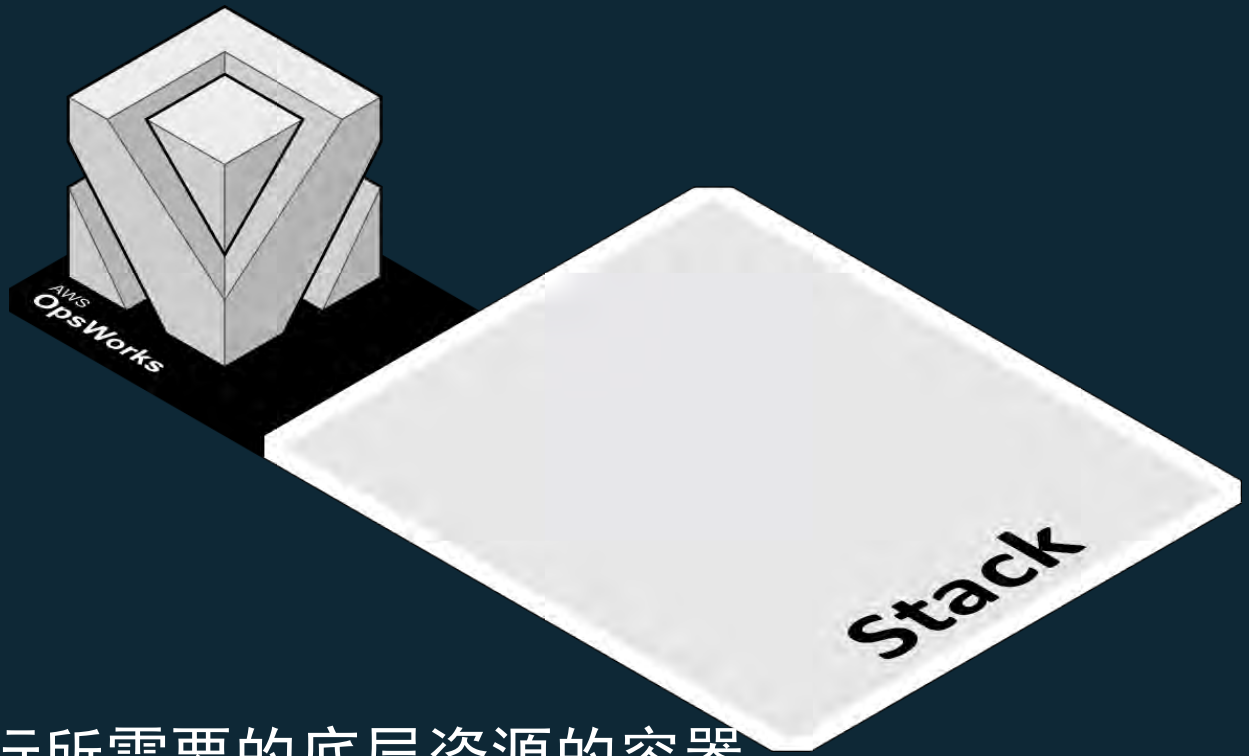
每个EC2上
运行agent



OpsWorks

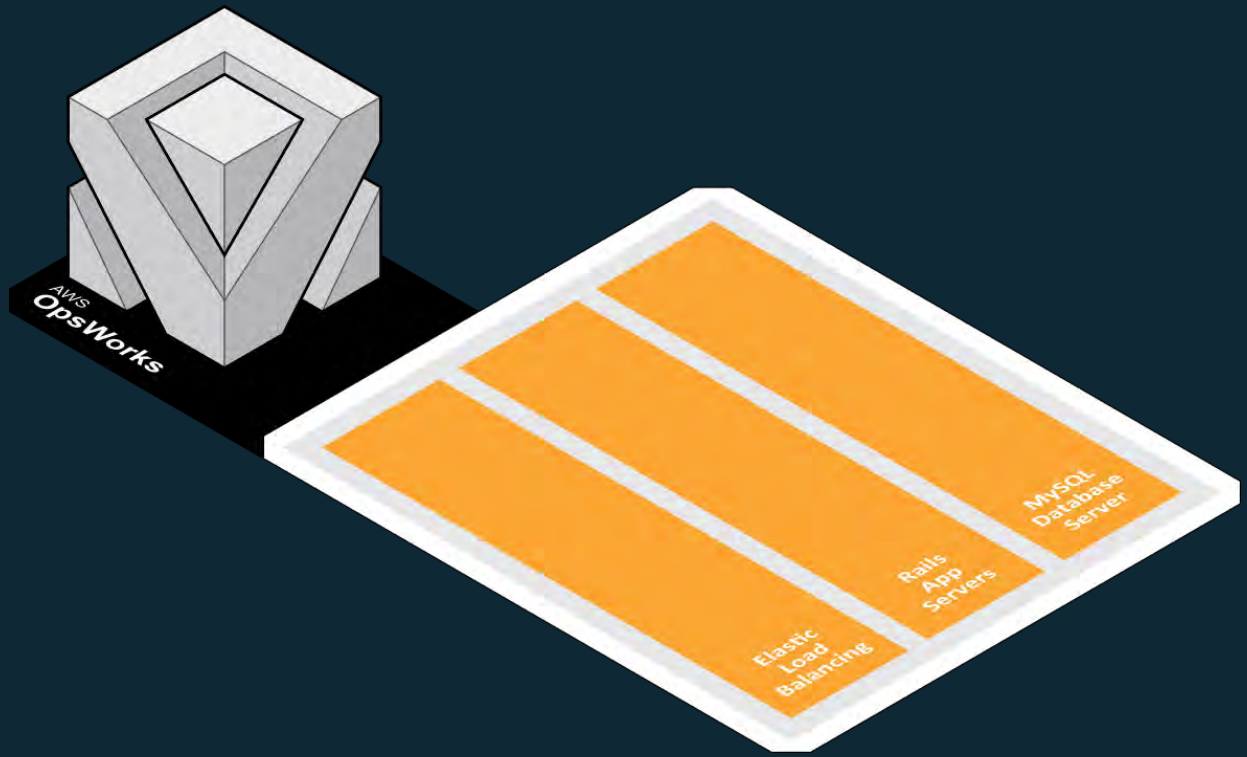
Agent与OpsWorks服务通信，理解其触发的命令，然后运行在Chef solo模式.

创建应用堆栈

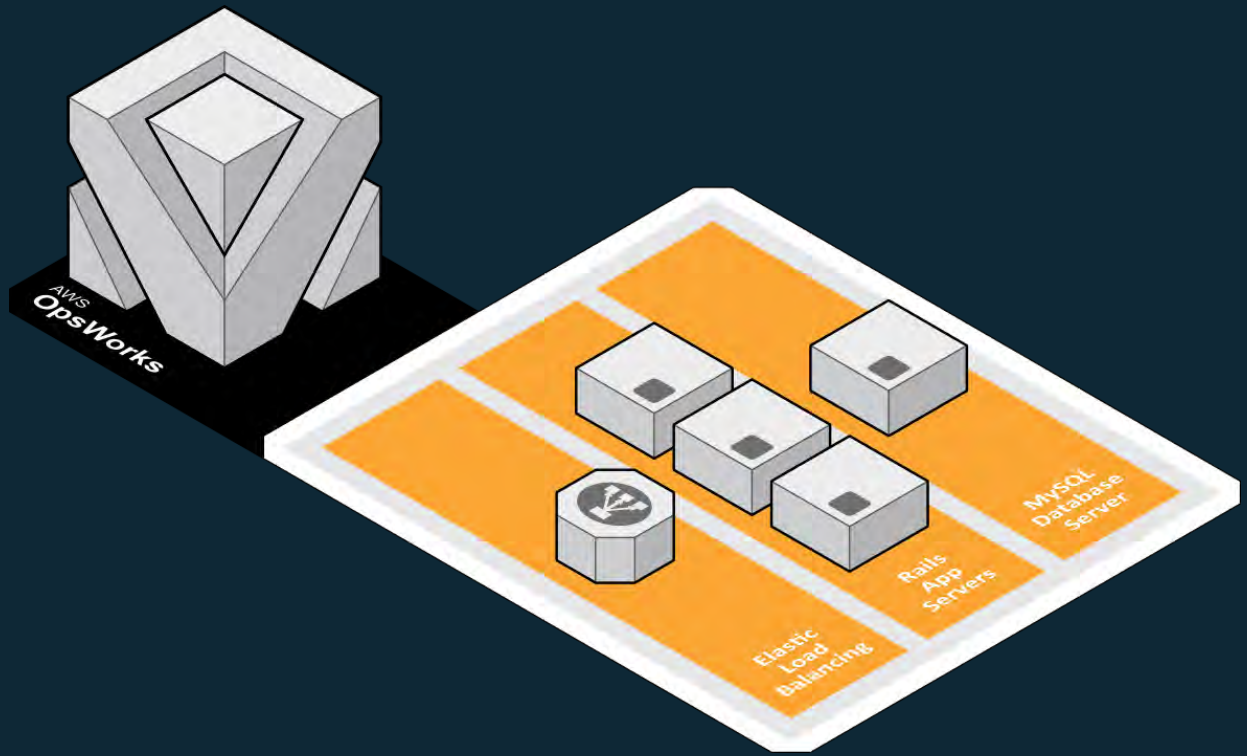


支撑应用运行所需要的底层资源的容器

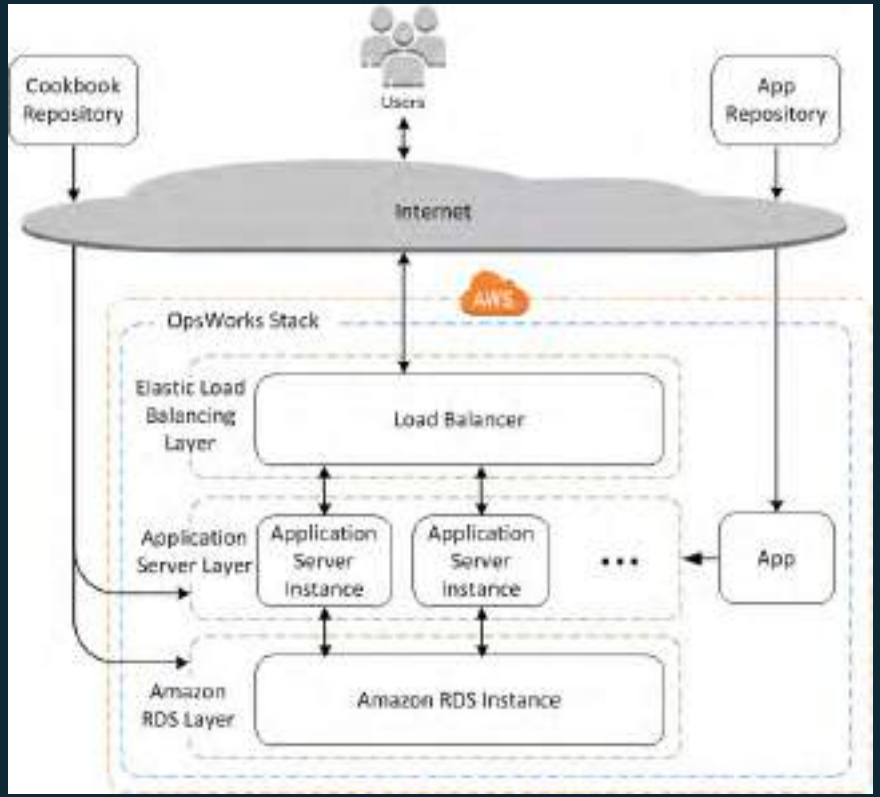
根据业务逻辑定义不同的层



为每一层添加实例


AWS
OpsWorks
MySQL
Database
Server
Rails
App
Servers
Elastic
Load
Balancing

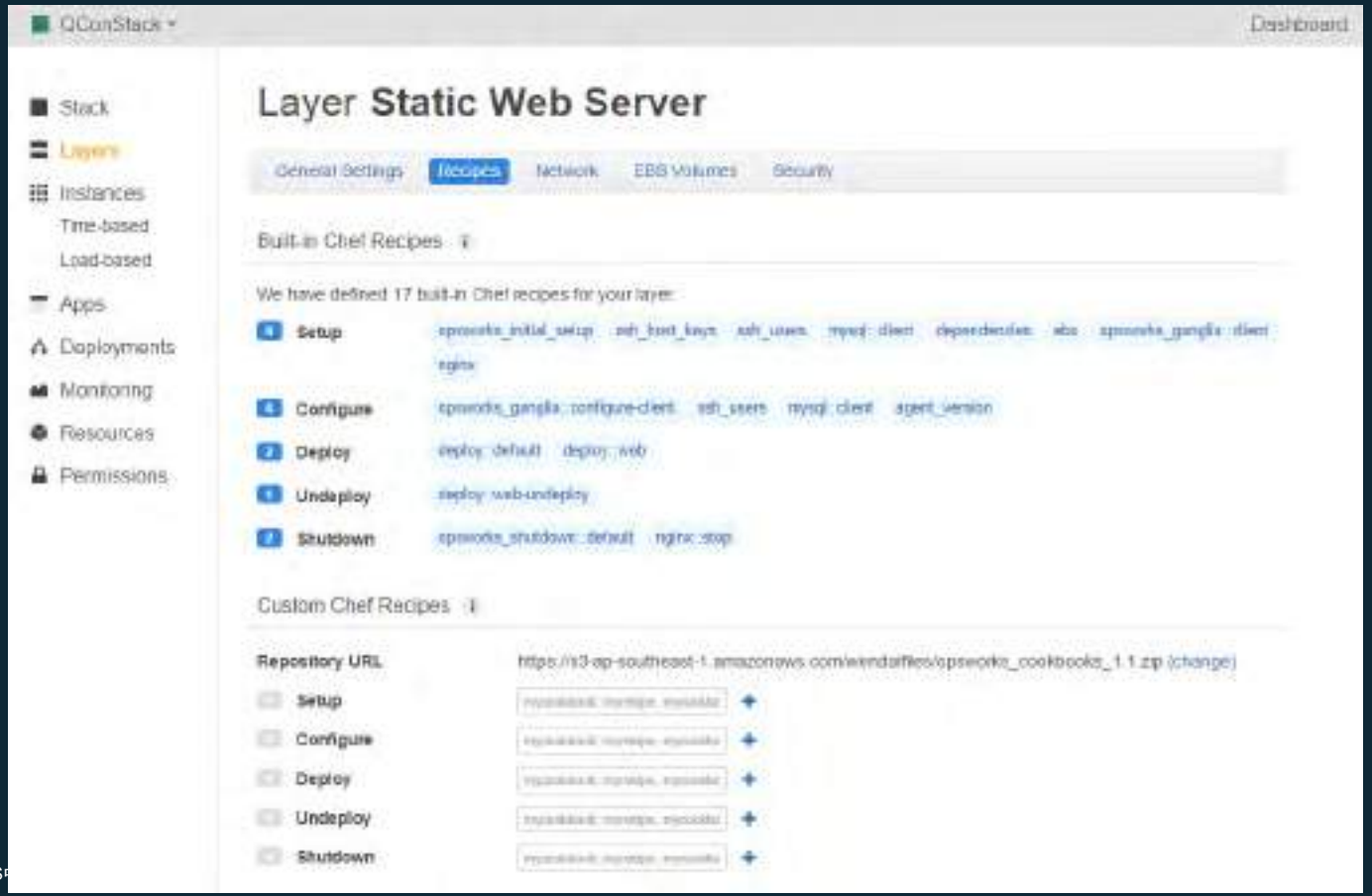
OpsWorks工作原理



AWS OpsWorks Agent支持的生命周期事件

setup configure deploy undeploy shutdown

使用内置Chef Recipe或定制Chef Recipes配置应用



第三方工具集成


GitHub
Atlassian
CloudBees
Travis CI
CODESHIP
Solano Labs
circleci
Jenkins
CHEF
puppet labs
ANSIBLE
SALTSTACK

贯穿始终的安全与监控

AWS IAM (Identity & Access Mgmt)

Manage **users, groups & permissions**

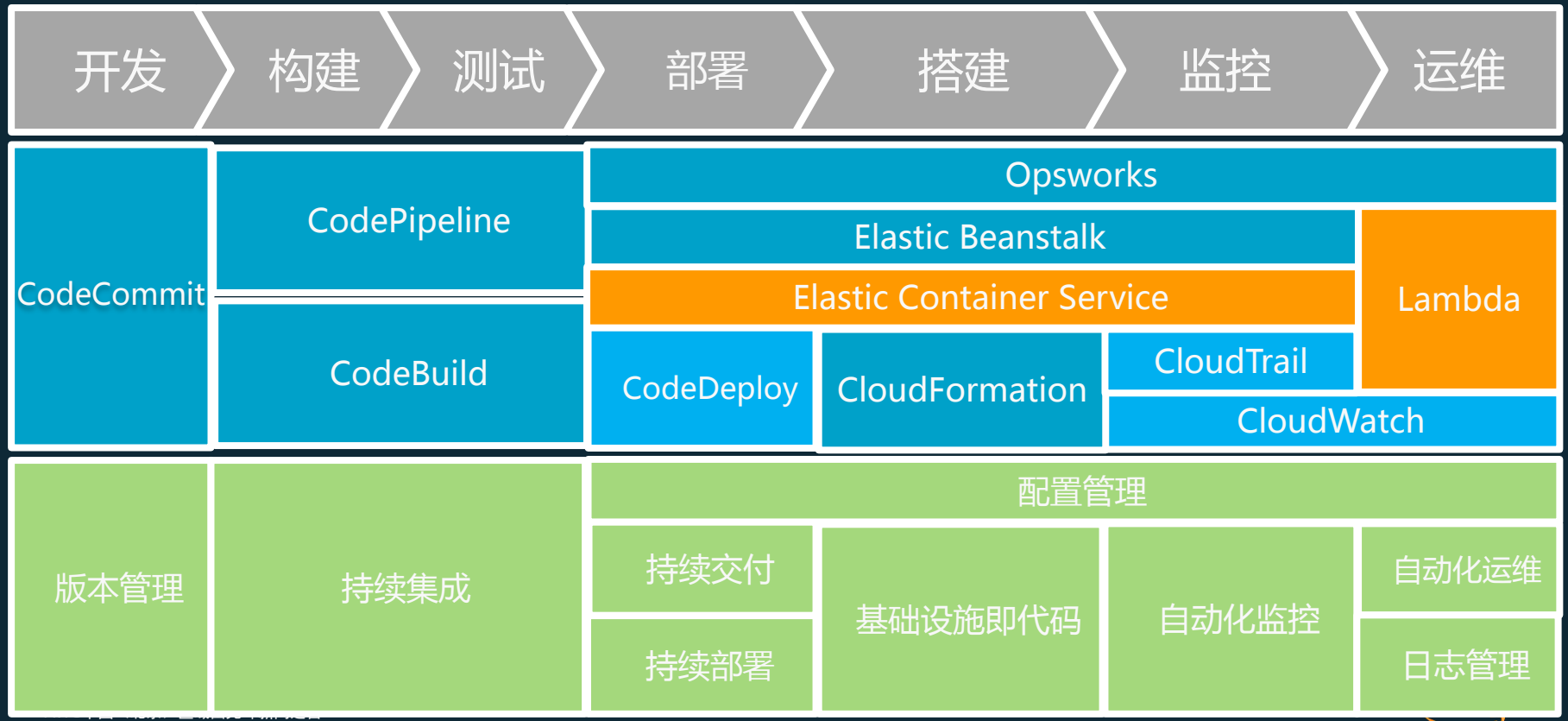
Amazon CloudWatch

Monitor resources

DevOps，不是一个目的地，而是一条路……

无服务器优化

AWS DevOps 服务



在微服务架构中使用API Gateway和Lambda

Amazon API Gateway

托管API及路由API调用

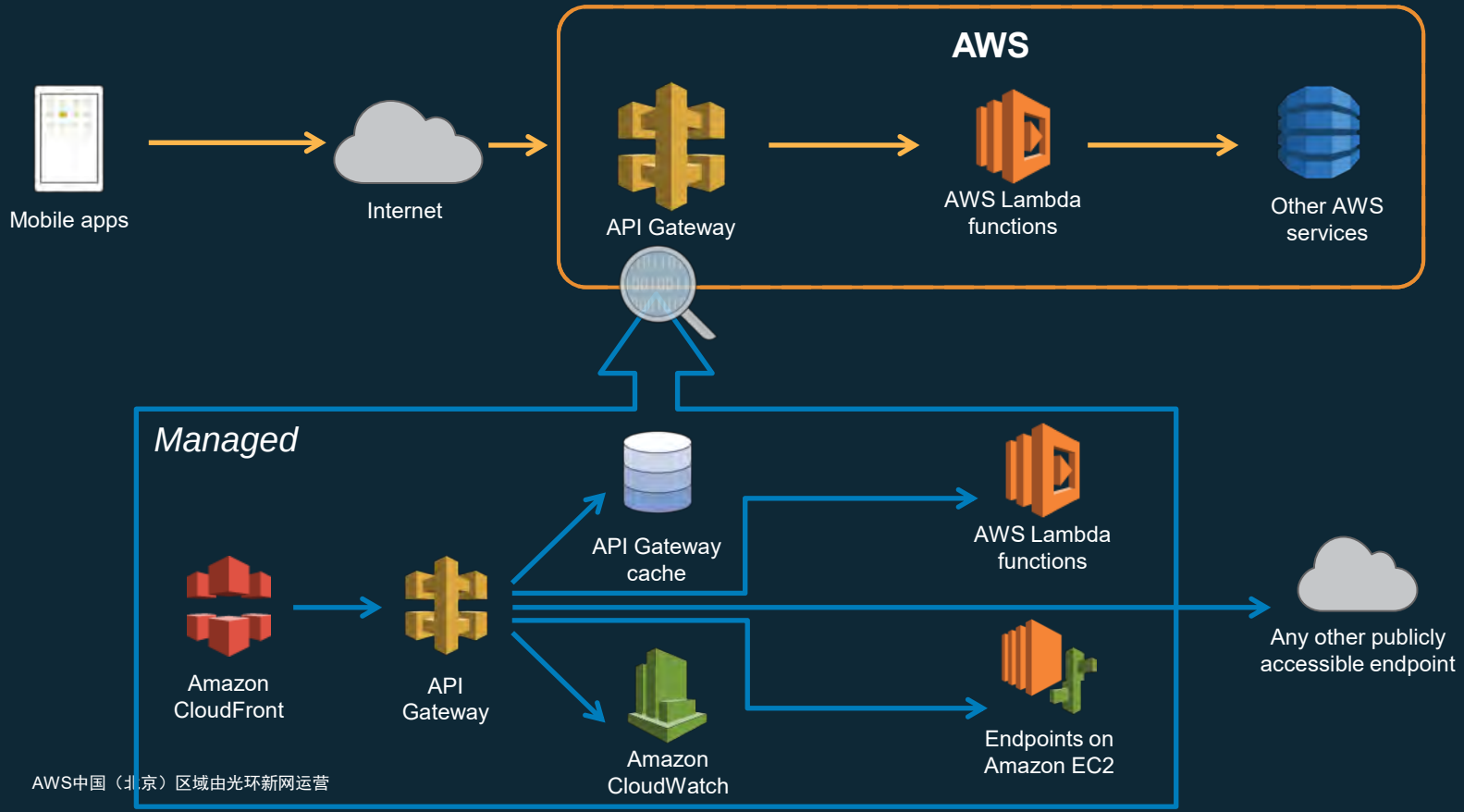
AWS Lambda

执行应用的业务逻辑

Lambda + API Gateway 意味着客户不需要管理基础设施

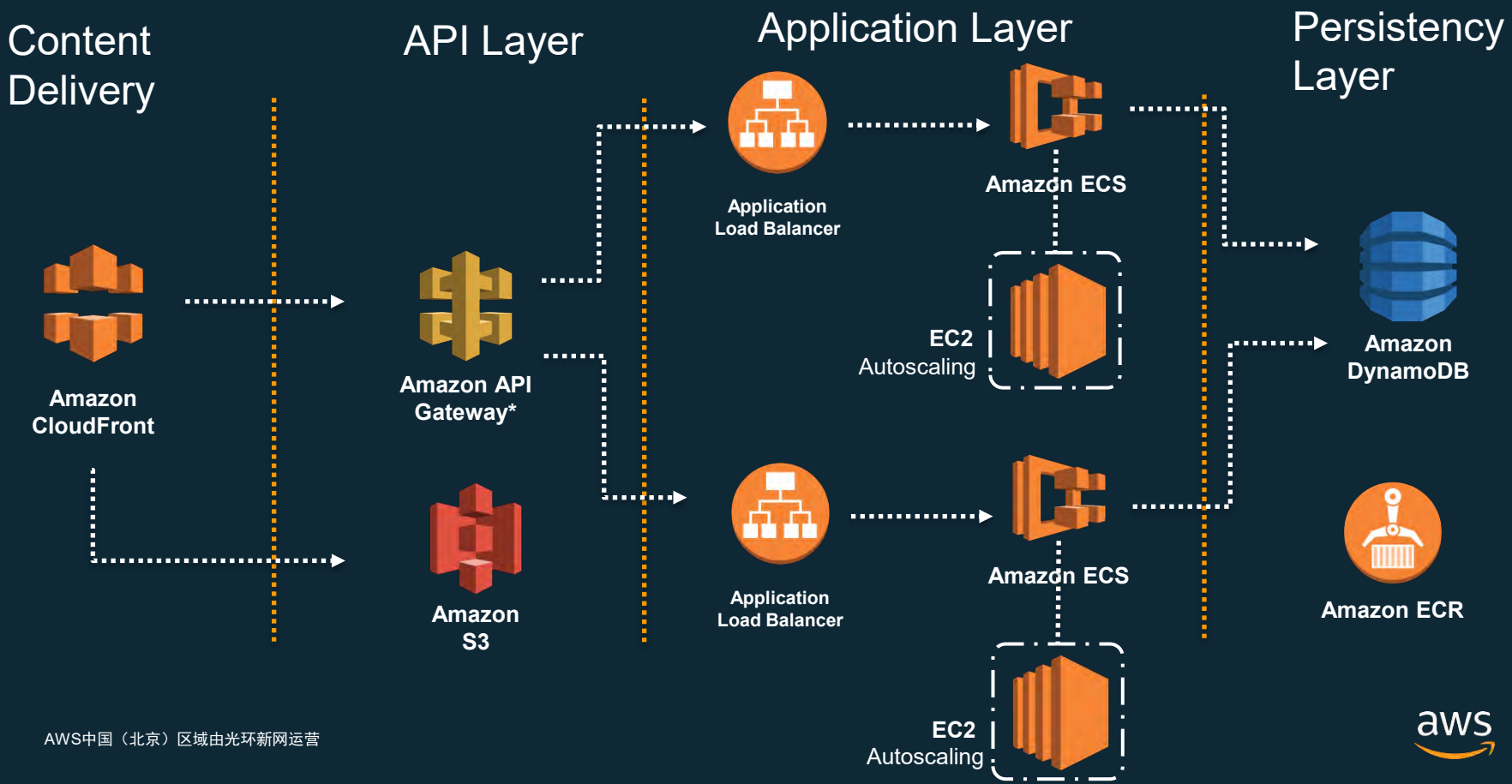
无服务器后台的架构模式

无服务器后台的架构模式

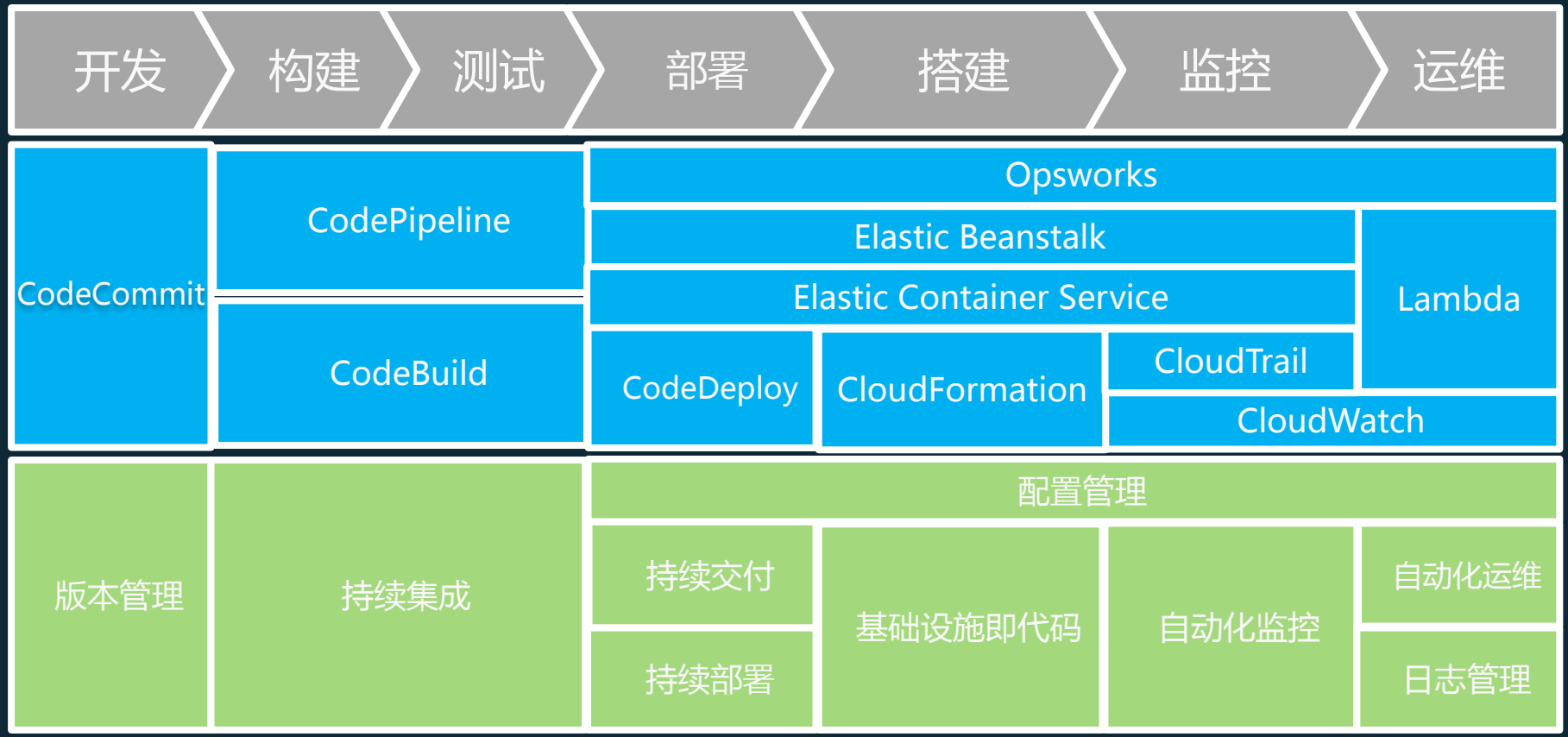


AWS中国（北京）区域由光环新网运营

ECS上的微服务架构


Content Delivery
API Layer
Application Layer
Persistency Layer
Amazon CloudFront
Amazon API Gateway*
Amazon S3
Application Load Balancer
Application Load Balancer
Amazon ECS
Amazon ECS
EC2 Autoscaling
EC2 Autoscaling
Amazon DynamoDB
Amazon ECR

AWS DevOps 服务



小结

开发运维DevOps概念

- 公司文化、Organization、流程的改变
- 系统架构向微服务Microservices模式改变

基于AWS的DevOps实践指南

- 实现基于AWS的基础设施层面自动化运维
- 基于AWS的Code services创建应用的自动交付渠道
- 使用API Gateway, Lambda优化微服务架构

一切皆软件，一切皆API

Thank you!