AWS 自动化运维

Davy
解决方案架构师

amazon web services

AWS 应用管理服务

高级服务

自己动手

Elastic Beanstalk

OpsWorks

CloudFormation

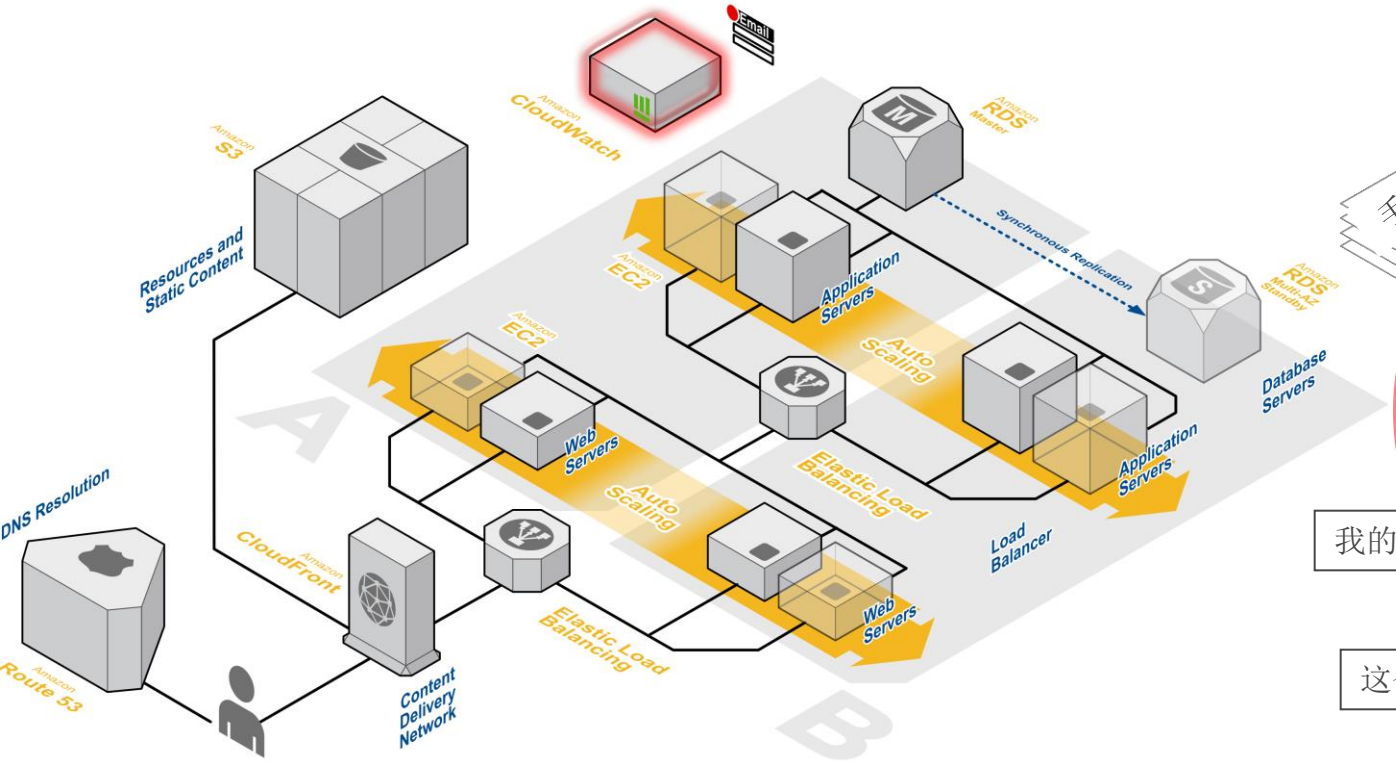
EC2

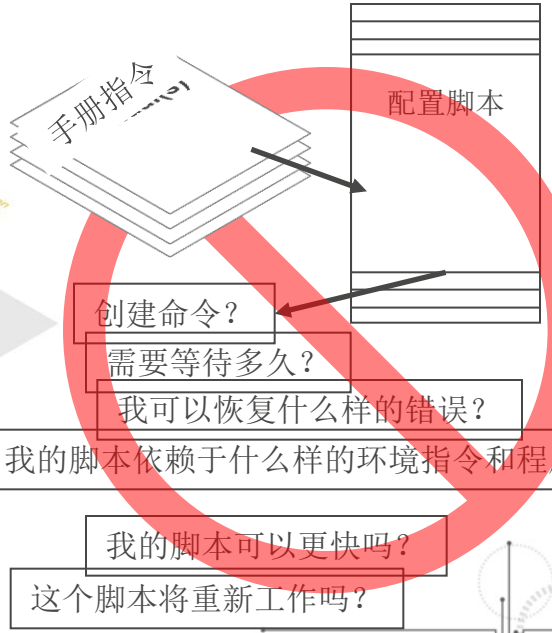
便利

控制

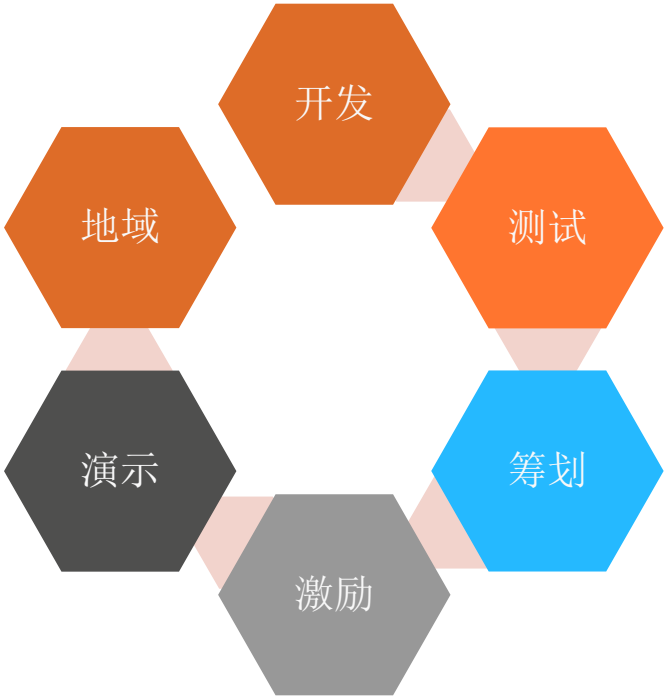
AWS CloudFormation

建模 → 单击 → 完成





AWS CloudFormation



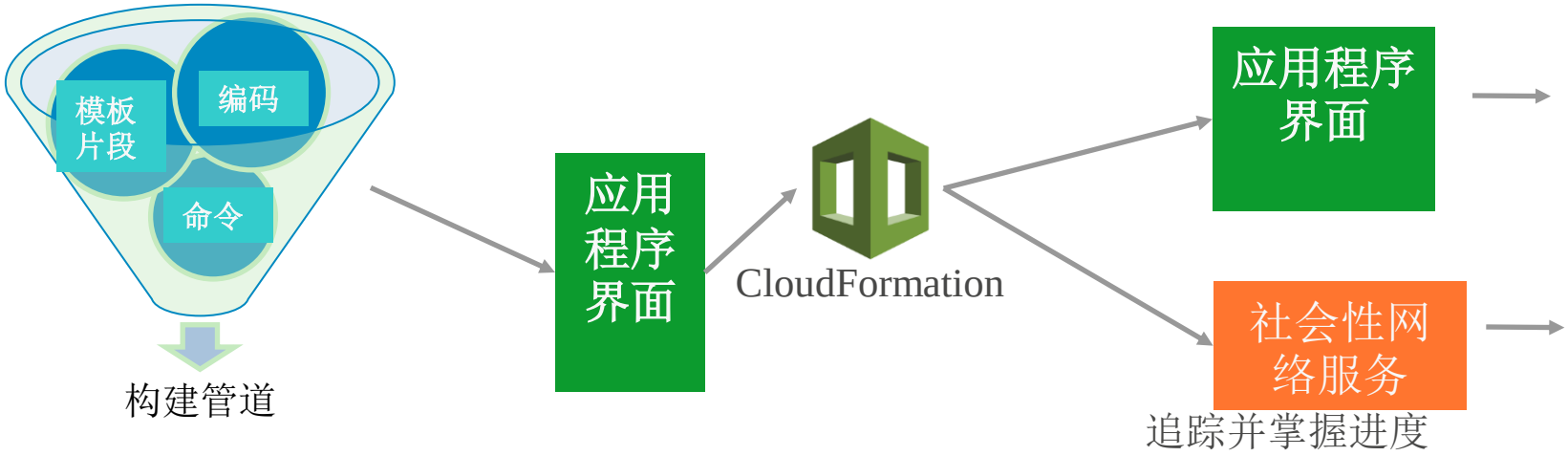
版本控制 → 复制

标准化

服务目录

AWS CloudFormation

自动化



基础结构设计

模板管理

堆栈管理

引导指令

基础设施即代码

admin_instance.json

app_and_cache_private_instance_group.json

app_private_instance_group.json

cache_private_instance_group.json

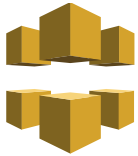
cloud_front_custom_distribution.json

```
},
"Properties": {
  "KeyName": { "Ref": "KeyPair" },
  "ImageId" : { "Ref" : "AmiId" },
  "InstanceType": { "Ref": "InstanceType" },
  "SecurityGroups": [
    { "Ref": "AppInstancesSecurityGroup" },
    { "Ref": "DBClientTrafficContainer" },
    { "Ref": "CacheClientTrafficContainer" },
    { "Ref": "OutboundAccessTrafficContainer" }
  ],
```

面向服务的体系结构

亚马逊简单队列服务

亚马逊远程数据服务



亚马逊内容推送服务

亚马逊路径53服务

亚马逊S3服务

亚马逊高速云缓存服务

亚马逊云监控服务

AWS CloudFormation

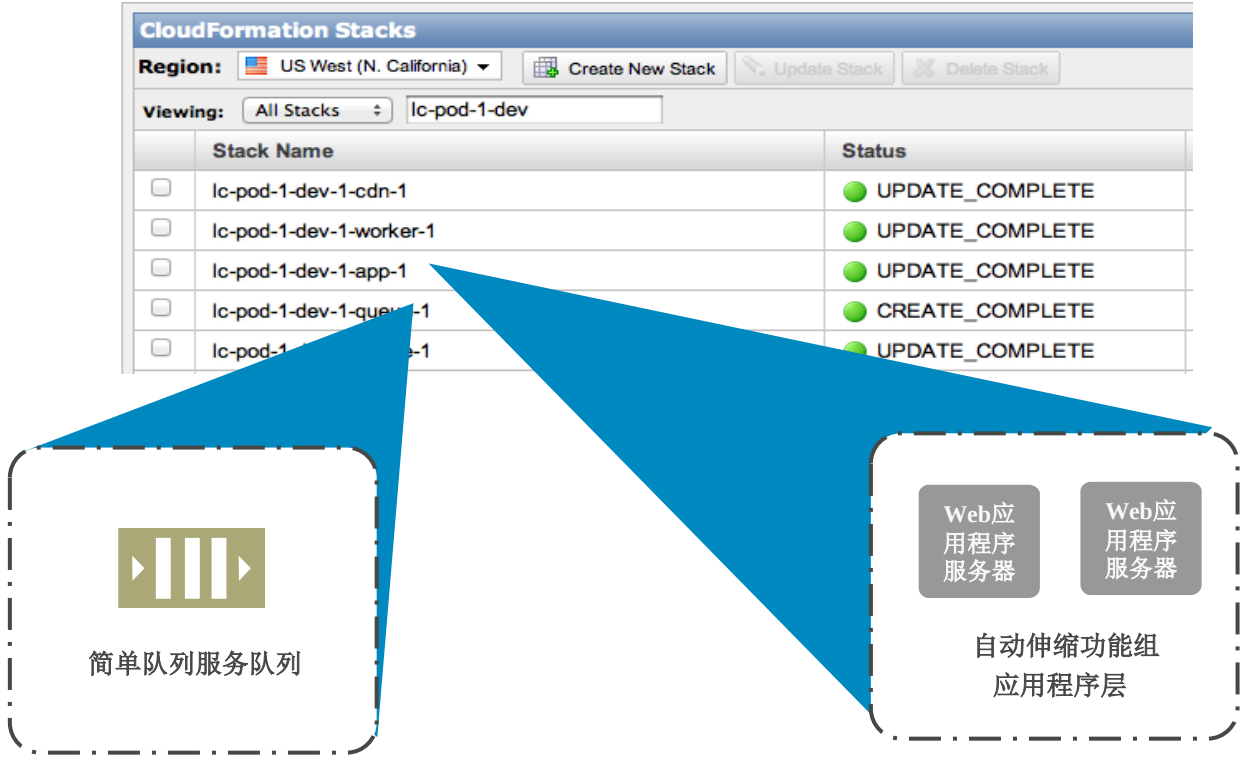
AWS身份及访问管理

亚马逊简单队列服务

亚马逊弹性计算机CloudFormation

亚马逊社会性网络服务

多个模板,松散耦合


CloudFormation Stacks
Region:
US West (N. California)
Create New Stack
Update Stack
Delete Stack
Viewing:
All Stacks
lc-pod-1-dev
Stack Name
Status
lc-pod-1-dev-1-cdn-1
UPDATE_COMPLETE
lc-pod-1-dev-1-worker-1
UPDATE_COMPLETE
lc-pod-1-dev-1-app-1
UPDATE_COMPLETE
CREATE_COMPLETE
UPDATE_COMPLETE
简单队列服务队列
Web应用程序服务器
Web应用程序服务器
自动伸缩功能组
应用程序层

多个模板，松散耦合

intuit

容易推断

可重复使用

堆栈管理

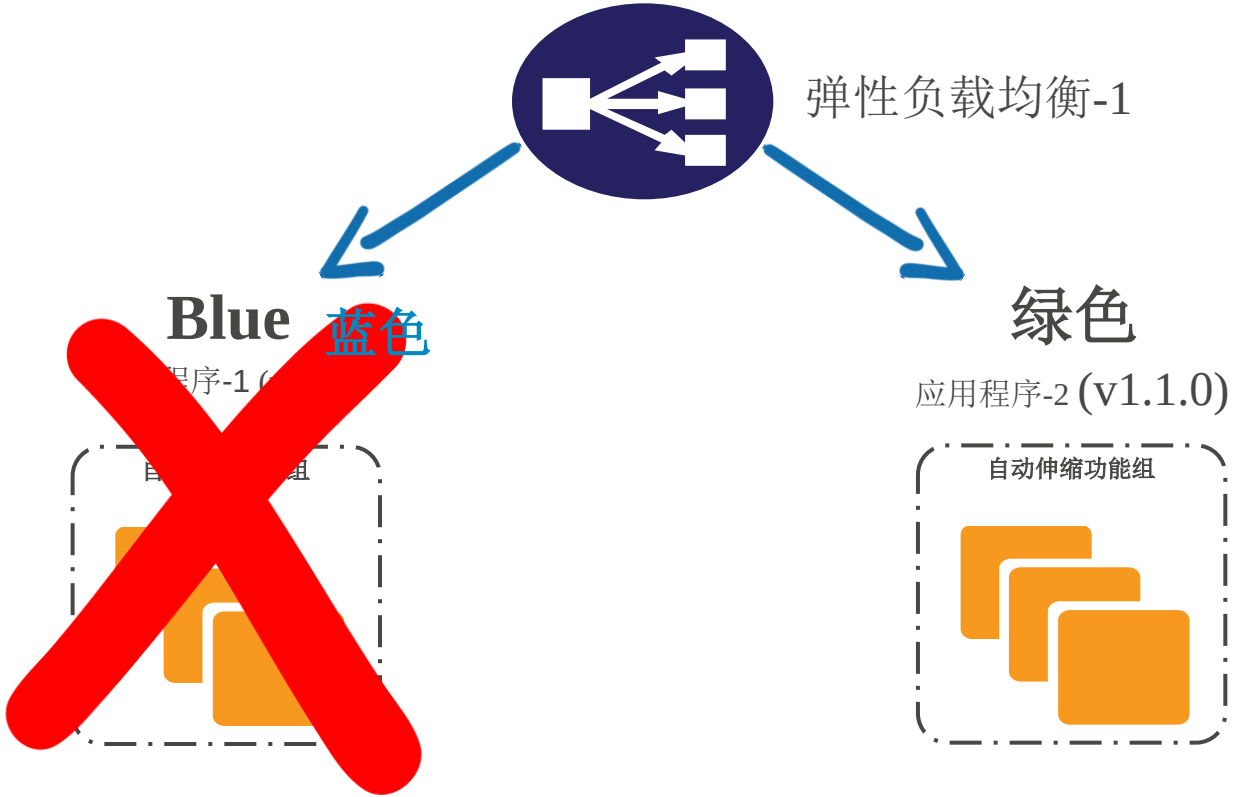
简单部署

https://github.com/intuit/simple_deploy

简单部署指令

属性
克隆
创建
部署
破坏
环境
事件
执行
实例

列表
输出
参数
保护
资源
状态
模板
更新


弹性负载均衡-1
Blue 蓝色
自动伸缩功能组
绿色
应用程序-2 (v1.1.0)
自动伸缩功能组

$ 简单的部署环境

Default

lc_preprod_us_west_1

lc_preprod_us_west_2

lc_preprod_us_east_1

PROD_lc_prod_us_west_1_PROD

PROD_lc_prod_us_west_2_PROD

PROD_lc_prod_us_east_1_PROD

```
$ simple_deploy list \
    –-environment lc_preprod_us_west_1
```

lc-dev-elb-1

lc-dev-app-1

lc-dev-db-master-1

lc-dev-db-parameter-group

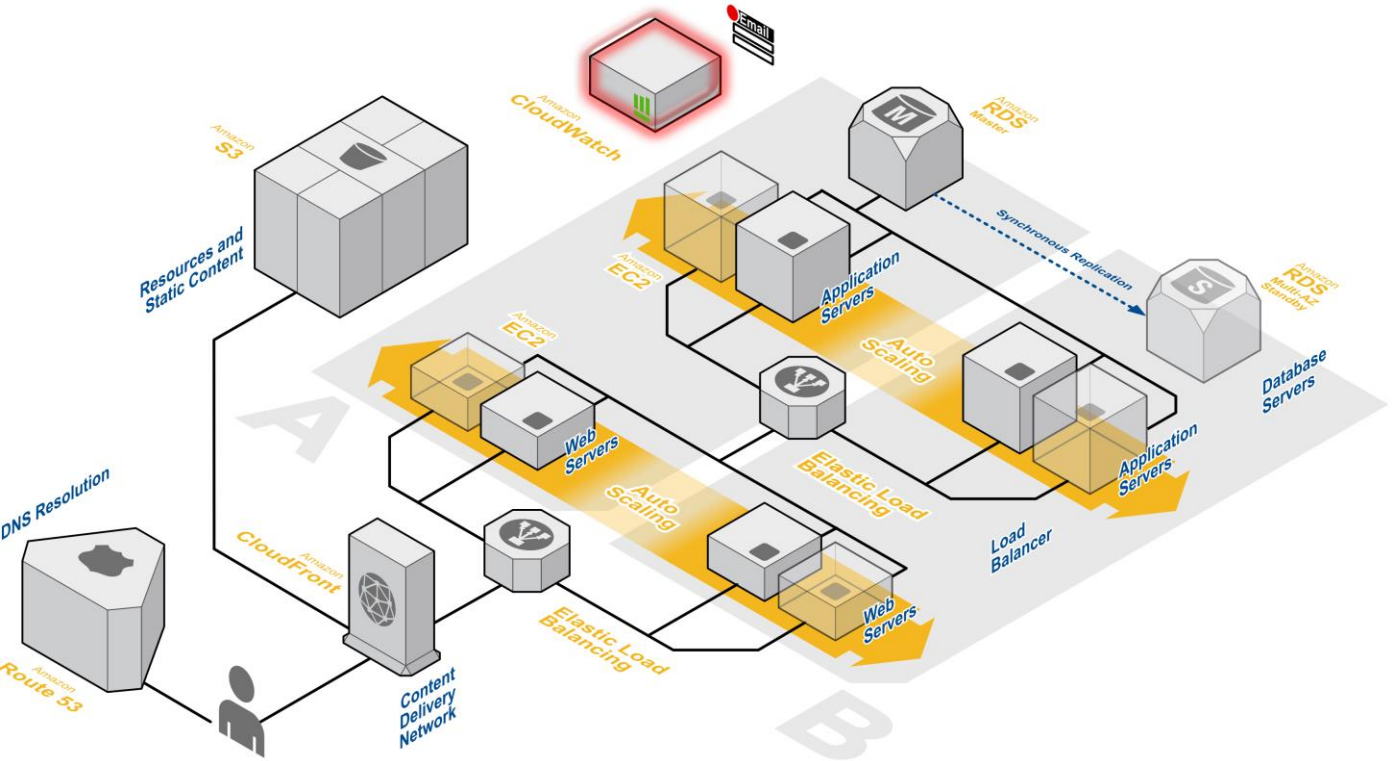
```
simple_deploy create \
  –-environment lc_preprod_us_west_1 \
  –-name lc-dev-app-2 \
  –-template app.json \
  –-input-stack lc-dev-elb-1 \
  –-input-stack lc-dev-db-master-1 \
  –-attribute chef_repo=3f57f9f \
  –-attribute app=bcb68de
```

```
simple_deploy clone
  --environment lc_preprod_us_west_1 \
  --source-stack lc-dev-1-app-1 \
  --name lc-dev-1-app-2 \
  --attribute app=afdac509b \
  --attribute chef_repo=a4531e5ff6
```

```
simple_deploy destroy
  --environment lc_preprod_us_west_1 \
  --name lc-dev-1-app-1
```

AWS CloudFormation的新功能

举例说明



可扩展的

可信的

高可用性

两种类型的任务

开发

NEW 并行堆栈处理

NEW 丰富的模板语言

操作

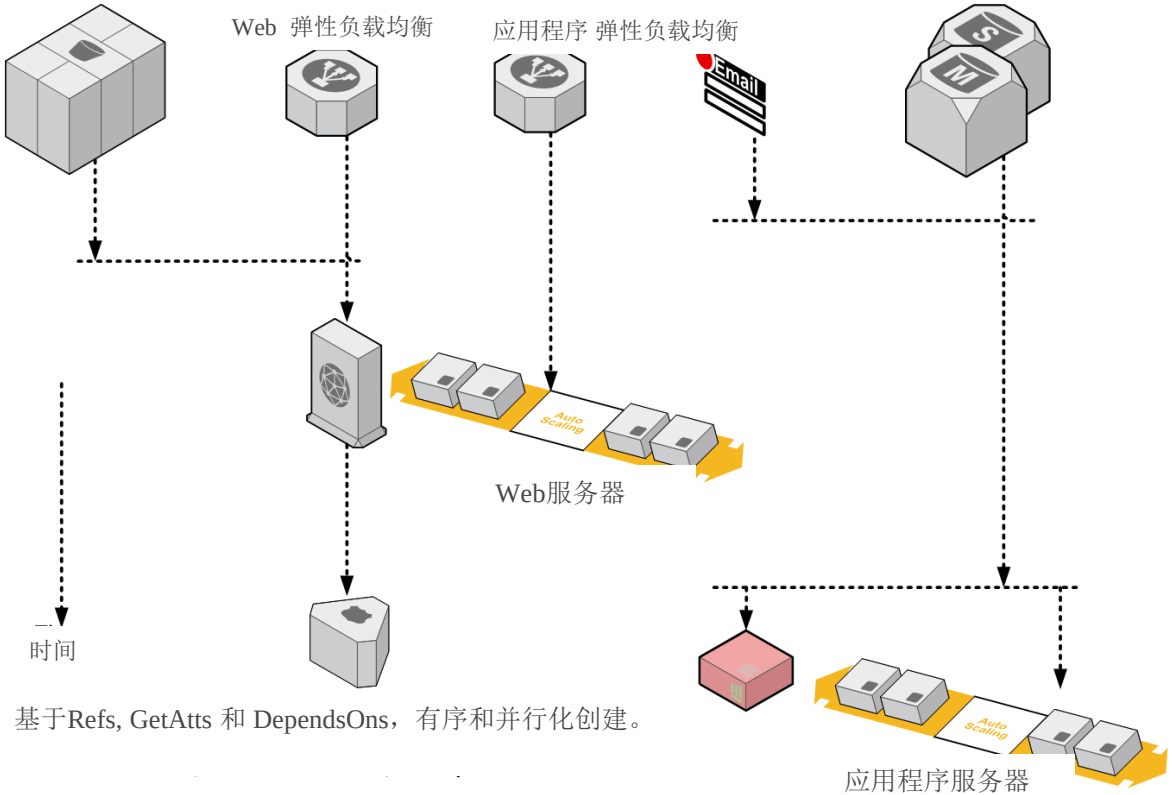
NEW 故障安全堆栈管理

NEW 无需停机进行更新

NEW 互联和身份及访问管理角色

并行堆栈处理

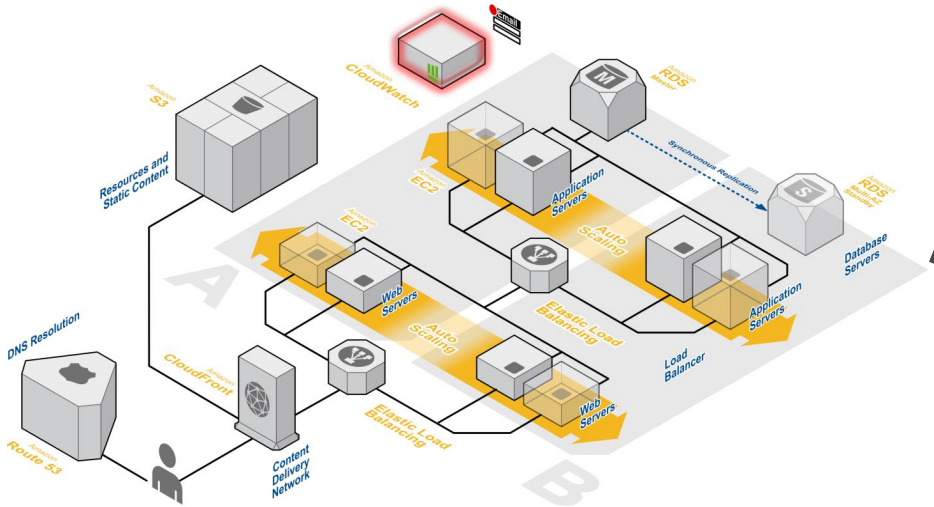
并行堆栈处理



基于Refs, GetAtts 和 DependsOns，有序和并行化创建。

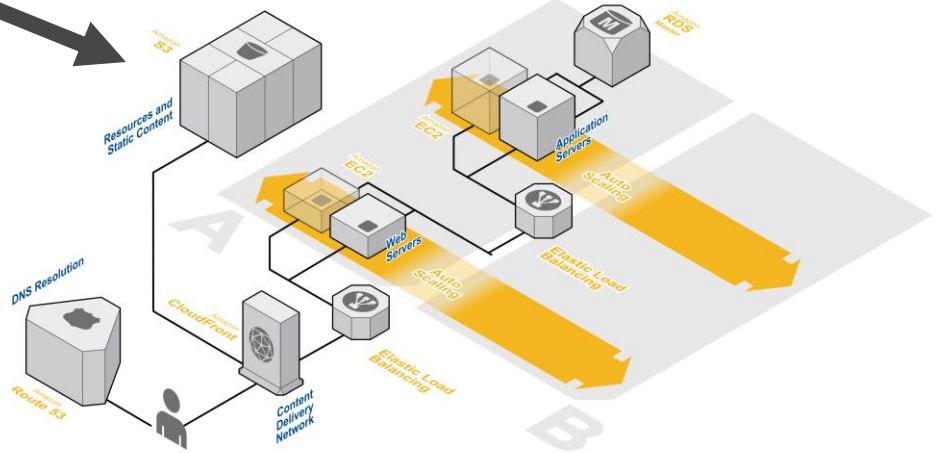
丰富的模板语言

条件


Amazon S3
Resources and Static Content
Amazon CloudWatch
Amazon RDS Master
Synchronous Replication
Amazon RDS Standby
Amazon EC2
Application Servers
Auto Scaling
Database Servers
Web Servers
Elastic Load Balancing
Load Balancer
DNS Resolution
Amazon CloudFront
Amazon Route 53
Content Delivery Network
A
B

生产

开发


Amazon S3
Resources and Static Content
Amazon RDS Master
Amazon EC2
Application Servers
Auto Scaling
Web Servers
Elastic Load Balancing
DNS Resolution
Amazon CloudFront
Amazon Route 53
Content Delivery Network
A
B

用户定义的资源名

默认，

- AWS CloudFormation生成独特的资源名
- “prodstack20131113-DBStorageAlarm-19BL0MOXL0TPI”“prodstack20131113-DB存储警报-19BL0MOXL0TPI”

此外，

- **灵活地**使用自定义名称,保持它们的唯一性
- “销售数据存储警报”

开发

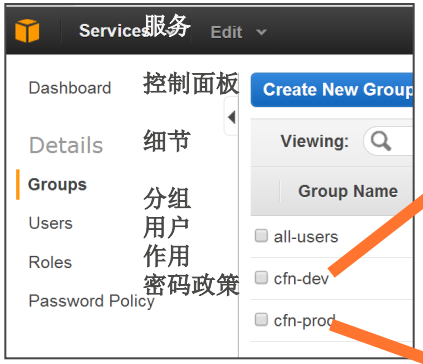
- 并行堆栈处理
- 丰富的模板语言

操作

- 故障安全堆栈管理
- 无需停机进行更新
- 互联和身份及访问管理角色

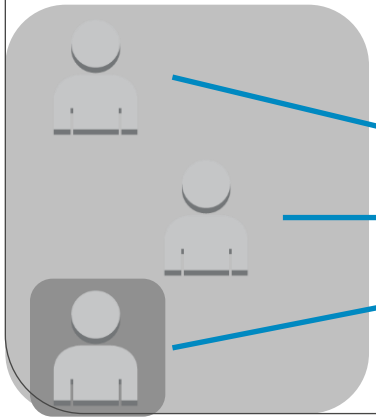
故障安全堆栈管理

堆栈保护




```
{
"Effect" : "Allow",
"Action" : [ "cloudformation:*" ],
"Resource" :
"arn:aws:cloudformation:us-west-
2:123456789012:stack/Dev*"
}
```

```
{
  "Effect" : "Allow",
  "Action" : [ "cloudformation:*" ],
  "Resource" : "*"
}
```



CloudFormation

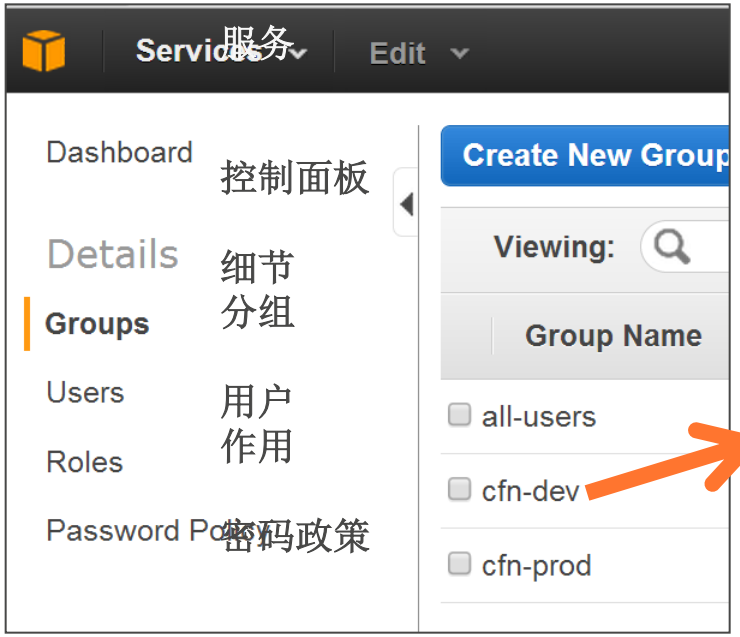
开发1

开发2

开发3

生产

堆栈保护

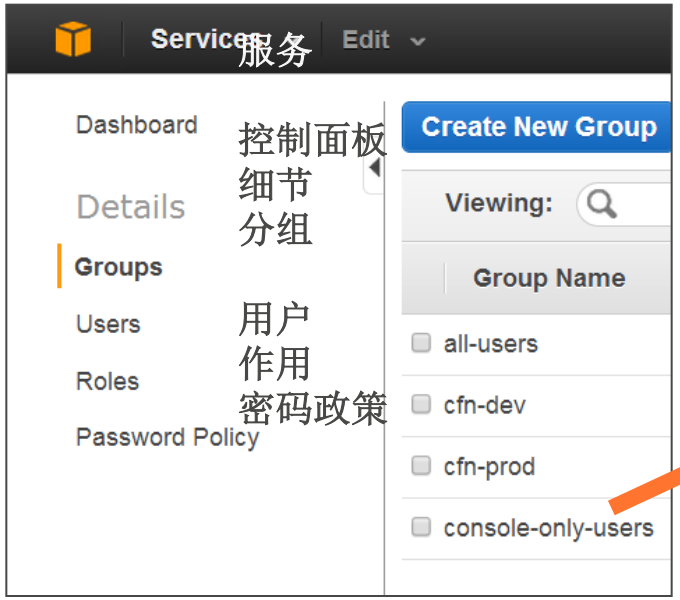



```
{
"Effect" : "Deny",

"Action" : [ "cloudformation:DeleteStack",
"cloudformation:UpdateStack" ],

"Resource" :
"arn:aws:cloudformation:us-west-
2:123456789012:stack/productionstack/*"
}
```

堆栈保护

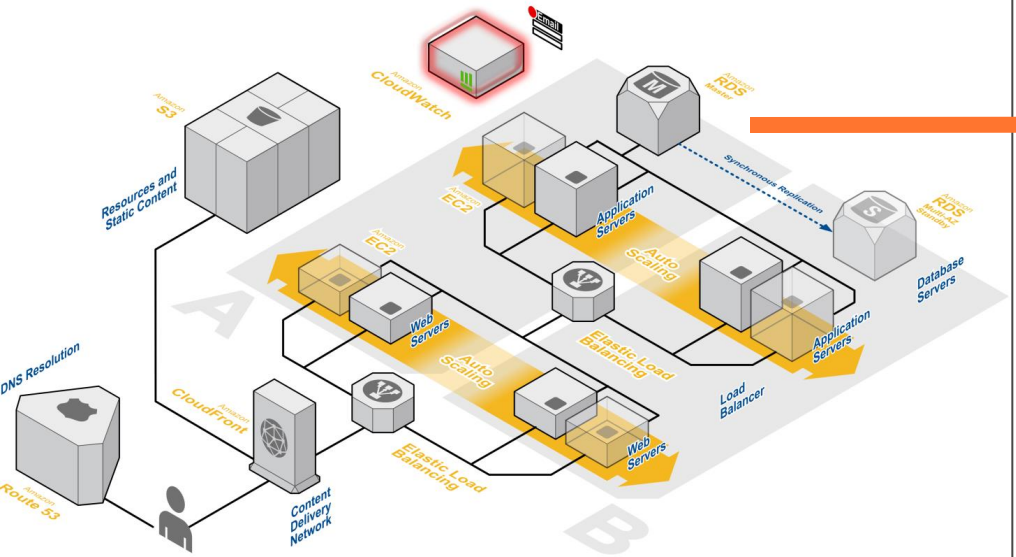
```
"Resources" : {
"StackProtectionPolicy" : {
"Type" : "AWS::IAM::Policy",
"Properties" : {
"PolicyName" : "StackProtectionPolicy",
"Groups" : [ { "Ref" : "DenyGrp" } ],
"PolicyDocument" : {
"Statement" : [
{
"Effect" : "Deny",
"Action" : [ "cloudformation:DeleteStack", "cloudformation:UpdateStack" ],
"Resource" : { "Ref" : "AWS::StackId" }
}
```

资源保护




```
{
    "Effect" : "Deny",
"Action" : [        "ec2:TerminateInstances"
    ],
    "Condition": {
"Null": { "ec2:ResourceTag/*cloudformation*" :
    "true" }
    },
    "Resource" : "*"
}
```

防止更新



堆栈政策文件

```
{
 "Statement" : [
   {
    "Effect" : "Deny",
    "Action" : "Update:*",
    "Principal" : "*",
    "Resource" :      "ResourceType/AWS::RDS::DBInstance"
   },
   {
    "Effect" : "Allow",
    "Action" : "Update:*",
    "Principal" : "*",
    "Resource" : "*"
   }
 ]
}
```

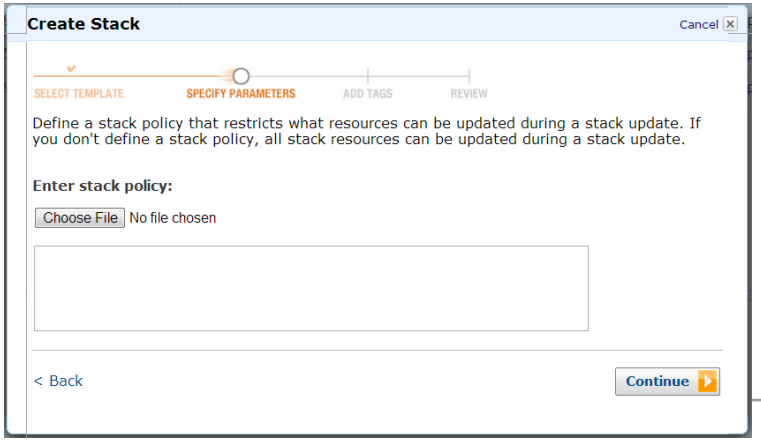
防止更新

细粒度访问控制堆栈政策

```
{
 "Statement" : [
   {
    "Effect" : "Deny",
    "Action" : "Update:Replace",
    "Principal" : "*",
    "Resource" :
    "LogicalResourceId/MyInstance"
   },
   {
    "Effect" : "Allow",
    "Action" : "Update:*",
    "Principal" : "*",
    "Resource" : "*"
   ...
```

设置堆栈政策

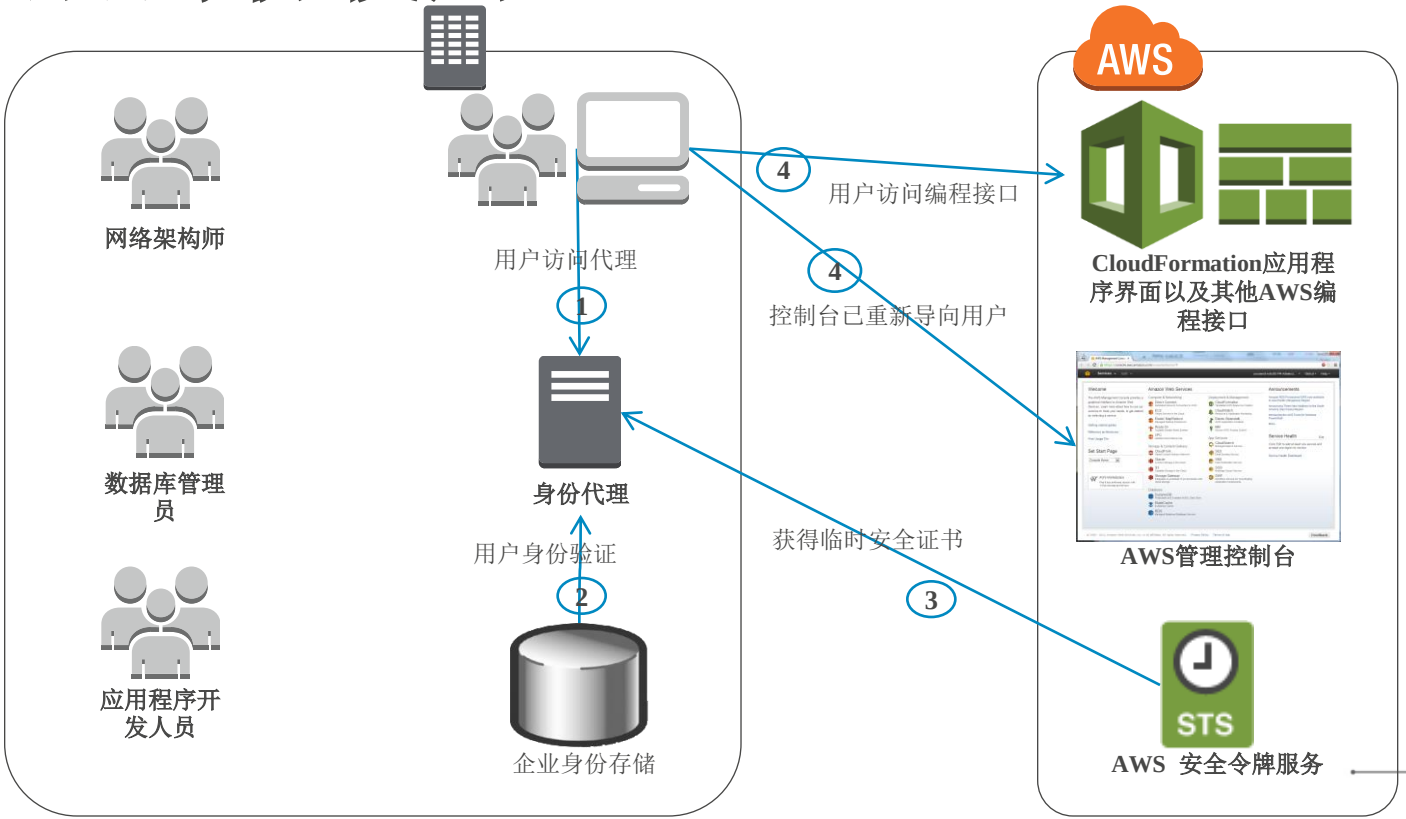
> aws cloudformation create-stack
--template-url ...
--stack-policy-url ...



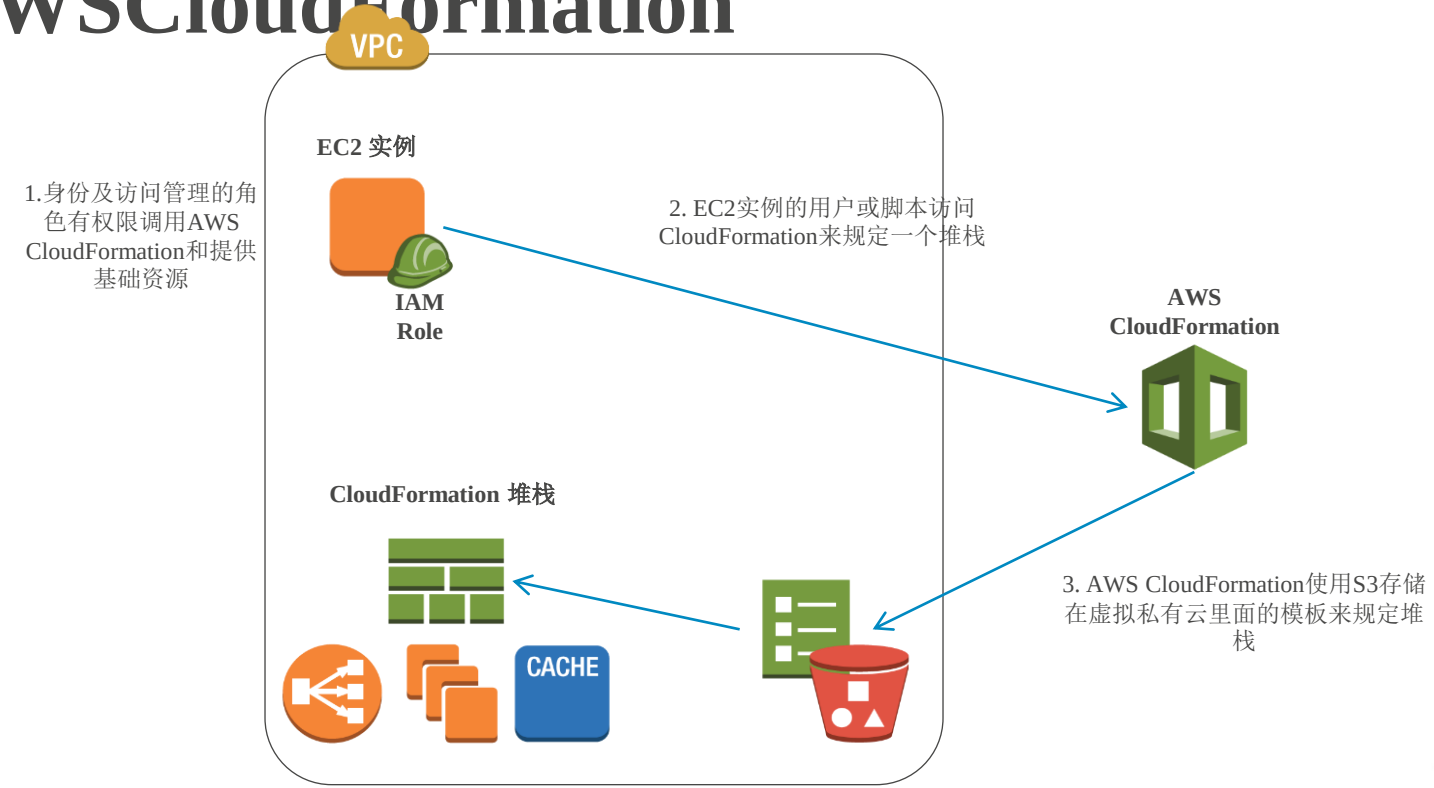
无需停机进行更新

```
"WebServerGroup" : {
"Type" : "AWS::AutoScaling::AutoScalingGroup",
"Properties" : {
"LaunchConfigurationName" : { "Ref" : "LaunchConfig" },
...
},
"UpdatePolicy" : {
"AutoScalingRollingUpdate" : {
"MinInstancesInService" : "2",
"MaxBatchSize" : "3",
"PauseTime" : "PT20M"
}
}
},
```

以联合身份使用AWS CloudFormation



使用身份及访问管理角色来调用 AWSCloudFormation



Opsworks

应用程序管理的挑战

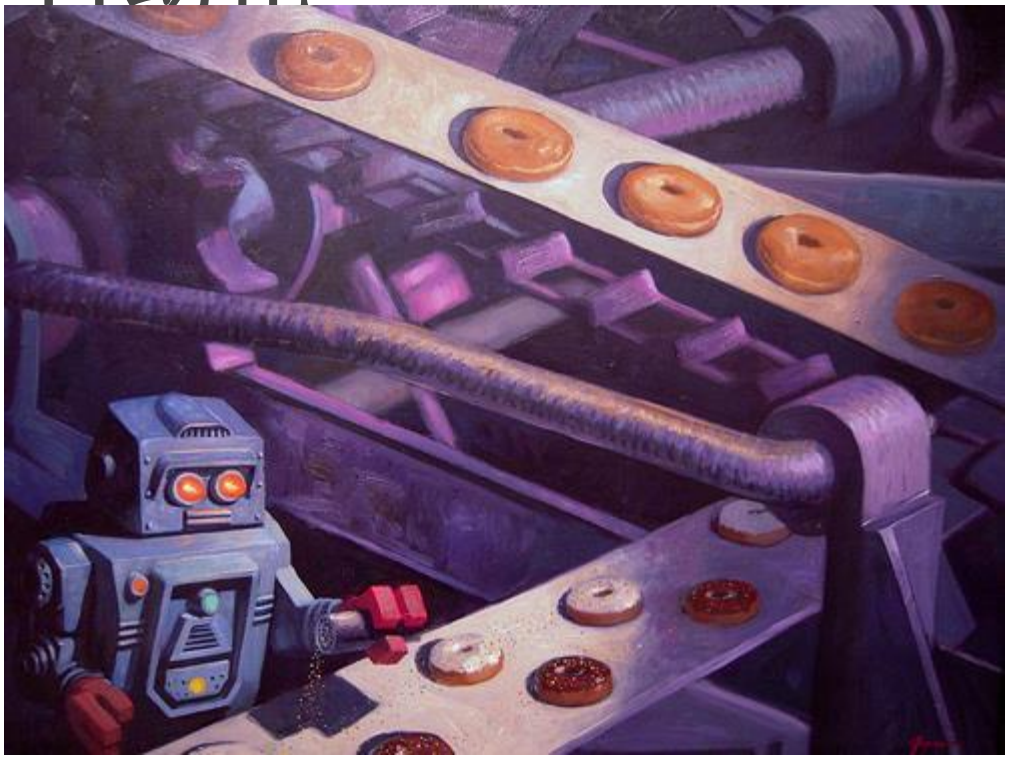
- 您所使用的应用程序的可靠性和可扩展性是非常重要的。
- 操作任务需要保持平稳运行
- 规定　　监视器
- 配置　　规模
- 安装　　安全
- 随着您的应用程序使用，例行的操作任务会耗费更多的时间，而且更容易出错
- 不能为了便于使用而去权衡控制性或者灵活性。

来源： http://www.mixph.com/2008/10/how-to-make-donuts-food-business.html

这是我们想要的.....

来源： http://thethriftythings.com/2013/01/doughnut-heaven-with-krispy-kreme-2013-bloggers-summit.html

菜单 + 自动化

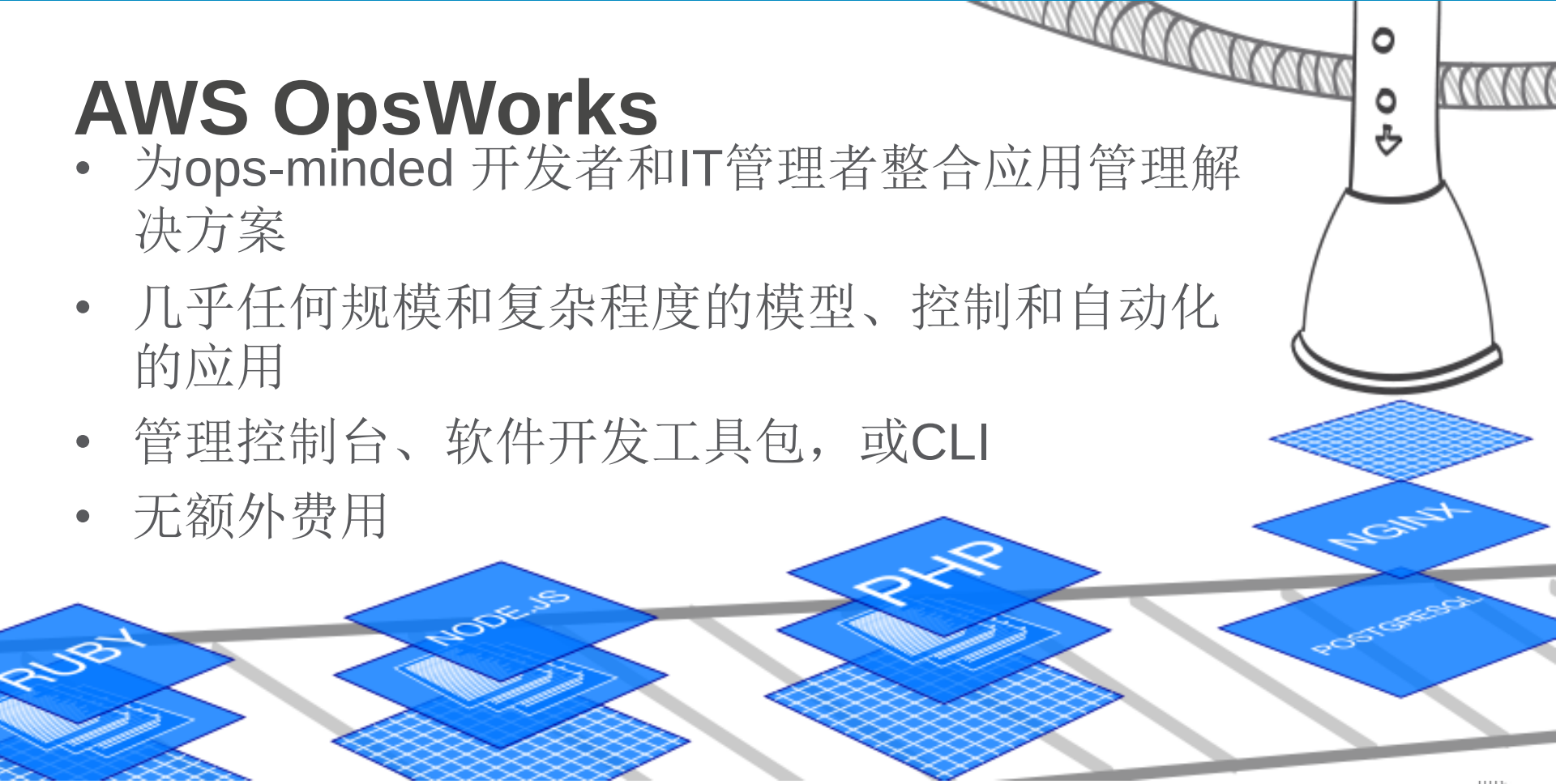


Source www.ericjoyner.com

当今的基础建设, 任何东西都是代码...
从手写开发应用程序到您的配置管理工具,
再到使用CloudFormation网盘来调配资源。

AWS OpsWorks

- 为ops-minded 开发者和IT管理者整合应用管理解决方案
- 几乎任何规模和复杂程度的模型、控制和自动化的应用
- 管理控制台、软件开发工具包，或CLI
- 无额外费用



为什么选择 AWS OpsWorks?

简单	高效	灵活	强大	安全
操作简单，可快速启动 且高效	减少常规和脚本配置错误	任何规模和复杂度程序的简化开发	减少自动化成本和时间	能控制细化权限

提高效率

- 可扩展的基础设施
- 灵活的架构
- 经常调配
- 存储环境

AWS OpsWorks 为我们提供自动化操作所需的工具 .
我们可以扩展怪兽世界, 该游戏是facebook最大的游戏之一, 面对数百万计的使用者而不需要两个以上的后端开发人员

Jesper Richter-Reichhelm
工程领导者

Wooga

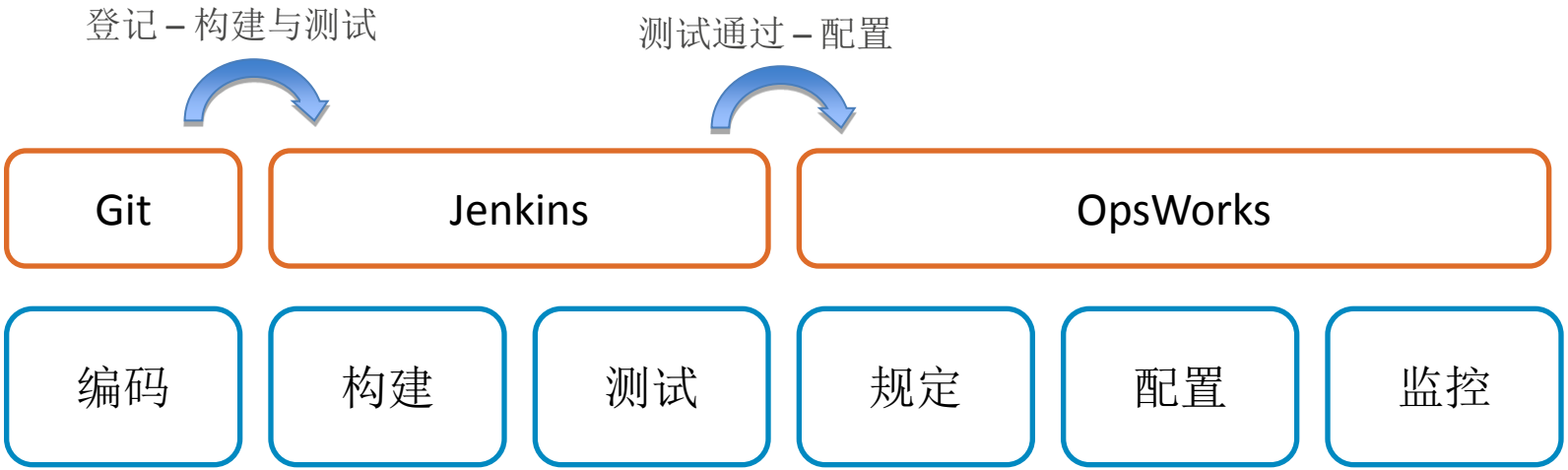
提高控制

- 版本配制
- 控制任何您可以编写的脚本
- 锁定控制台访问
- 更改和部署日志

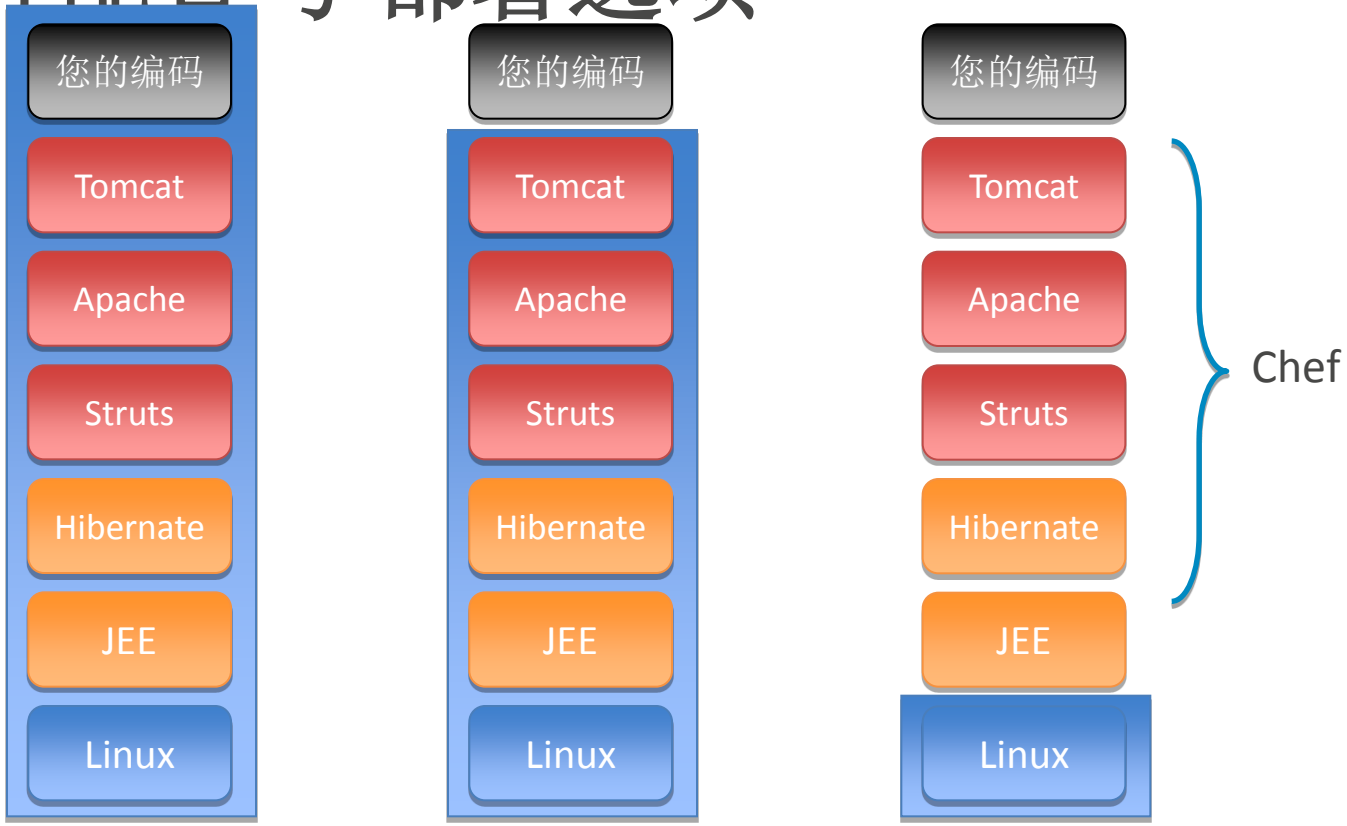
Crashlytics 使用 AWS OpsWorks支持快速增长的移动崩溃的报告解决方案。
通过使用 AWS OpsWorks, 我们可以专注于我们的业务的发展和成长并且不需要花费基础设施和运营任务的开发周期。

Jeff Seibert
Crashlytics公司CEO

提高可靠性



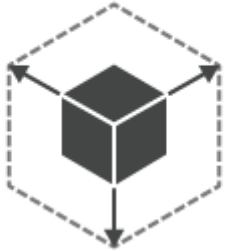
软件配置与 部署选项


您的编码
Tomcat
Apache
Struts
Hibernate
JEE
Linux
您的编码
Tomcat
Apache
Struts
Hibernate
JEE
Linux
您的编码
Tomcat
Apache
Struts
Hibernate
JEE
Linux
Chef

使用Opsworks的基本步骤

堆栈表示您要一起管理的云基础架构和应用程序。

单层定义了该如何安装和设定一组实例和相关资源。



决定如何扩展：手动、全天候的实例、自动、负载型或基于时间的实例。

然后部署您的应用程序于具体事例，并以Chef菜单自定义部署情况。

Chef是什么，OpsWorks该如何使用它？

生命周期事件

设置

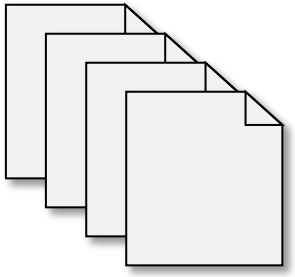
配置

部署

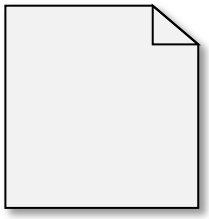
取消部署

关闭

菜单



元数据



- Chef是一个开源框架，可以自动进行软件部署和配置。
- 在您的堆栈中，每当一个变化发生的时候，或者根据要求，所有实例都被报告，菜单运行。

演示Chef 菜单

菜单

```
execute “mysql-connect” do
   command “/usr/bin/mysql
       -u#{node[:deploy][:myphpapp][:database][:username]}
       -p#{node[:deploy][:myphpapp][:database][:password]}
       #{node[:deploy][:myphpapp][:database][:database]}
…
```

+

元数据

```
“deploy”: {
    “myphpapp”: {
        “database”: {
            “username”: “root”,
            “password”:  “abcxyc”,
…
```

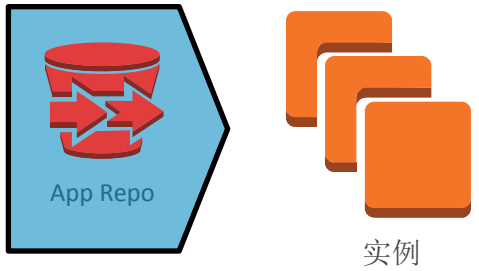
=

命令

```
“/usr/bin/mysql  -uroot –pabcxyz myphpapp …
```

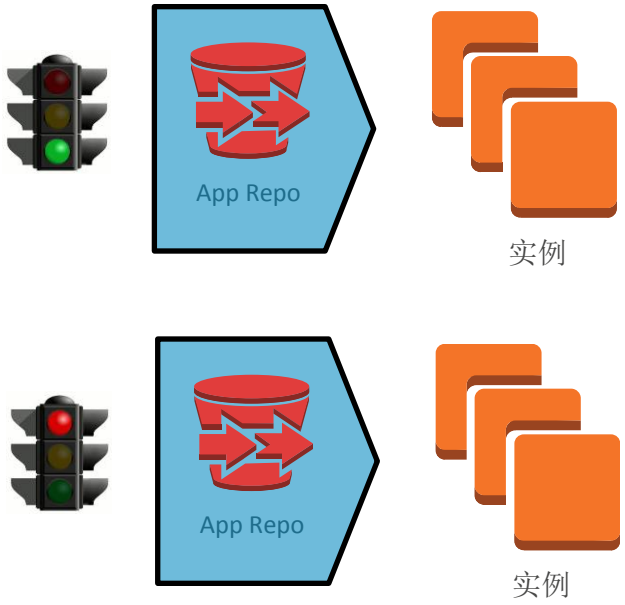
部署一个新的应用程序版本：选项1

- 在您的源回购中指向您的应用程序到新版本
- 部署一个实例和测试
- 部署所有的实例
- 如果需要，回滚



部署一个新的应用程序版本：选项2

- 复制您的堆栈
- 在复制的堆栈上部署新的应用程序版本
- 当准备好时切换DNS
- 成功运行时删除旧的堆栈



OpsWorks&IAM角色

- 使用情况：您想限制EC2实例用户，这个可以管理，例如：当他们打算重新启动一个测试实例的时候，防止有人意外重启您的结果实例。
- 答案：使用IAM用户和OpsWorks
  - 给予Jane OpsWorks政策
  - 从EC2上执行的操作上限制Jane
  - Jane能够通过OpsWorks创建EC2实例，但她无法用EC2描述/管理实例

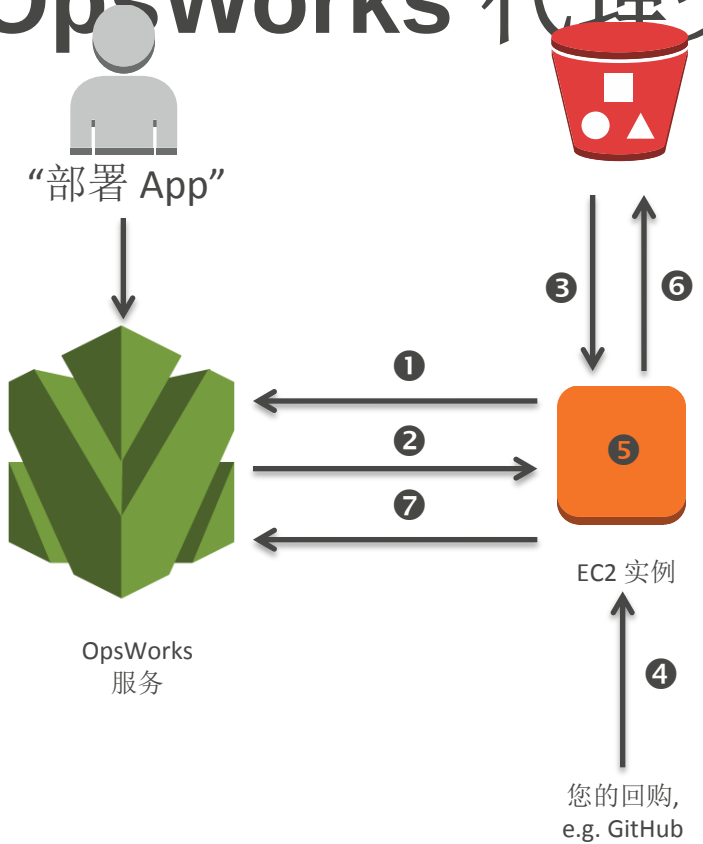
OpsWorks & EC2 角色

- 使用情况：您希望您的EC2实例能够安全地调用AWS服务而不以源代码的方式存储敏感的进入键。
- 答案：EC2角色。每一个EC2实例获取一个角色，他用于建立与OpsWorks服务的安全连接。可以拓展这个角色，以允许访问其他服务。请参阅文档初排的S3例子。

OpsWorks启动设定EC2是什么样的？

- 实例以IAM角色开始
  - 用户数据通过私有密钥，OpsWorks公钥
  - 例如下载和安装OpsWorks代理
- 代理连接到实例服务，获取运行信息
  - 使用实例的IAM角色进行实例验证
  - 从OpsWorks实例队列中提取设定JSON
  - 解密和验证消息，运行Chef菜单
  - 上传Chef日志，Chef返回运行状态
- 更多消息代理商投票服务实例

OpsWorks 代理交流



1. 实例与OpsWorks服务相连，发送保持活力的重要特征和接收生命周期事件
2. OpsWorks发送生命周期事件和指针配置的JSON（元数据，菜单），在S3存储器中
3. 下载配置的JSON
4. 从回购下拉菜单和其他构建资产
5. 执行菜单与元数据
6. 上传Chef日志
7. 报告Chef运行状态

OpsWorks 和 VPC

- 对于每一个人，OpsWorks完全支持新的VPC。
  一些注意事项:
  - 实例必须能够连接到OpsWorks服务端点，以获得配置更新
  - 在默认的VPC和默认子网掩码条件下只提供EC2实例-在这个时候，还不能针对特定的VPC或子网
  - 您的帐号必须支持默认VPC（EC2-VPC）
- 今年我们正在考虑什么？
  - 让实例瞄准特定的VPC/子网
  - *您还有什么需要？*

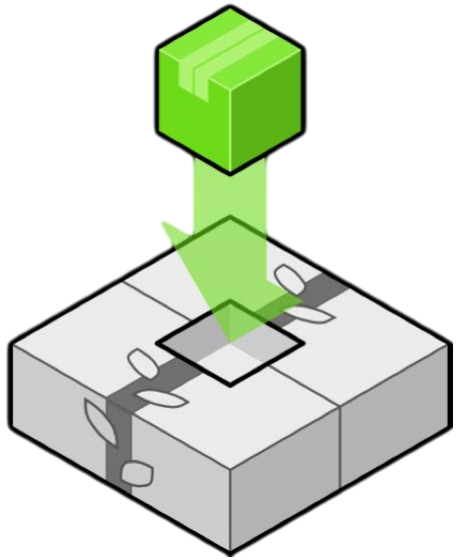
OpsWorks 和其他 AWS 资源

- 用户要求：支持ELB，RDS等。
- 按客户要求优化AWS层。
- 然而，许多资源并不需要充分全部的OpsWorks支持就可以创建并使用。例如，通过自定义的JSON添加连接参数，RDS就可以使用。

```
{ "deploy":
    { "myphpa": {
    "database": {
    "adapter": "mysql",
    "username": "awsuser",
    "database": "mydb",
    …
} } } }
```

Elastic Beanstalk (EB)

- Your code deployed into a container of your choosing
- Infrastructure deployed and managed by EB - but you still maintain complete control
- Supports these platforms:



EB Components

Application

Version(s)

- Application code
- Stored in Amazon S3
- One app can have many versions

Environment(s)

- Infrastructure resources (instances etc.)
- Can only run one version at a time
- One app can have many environments

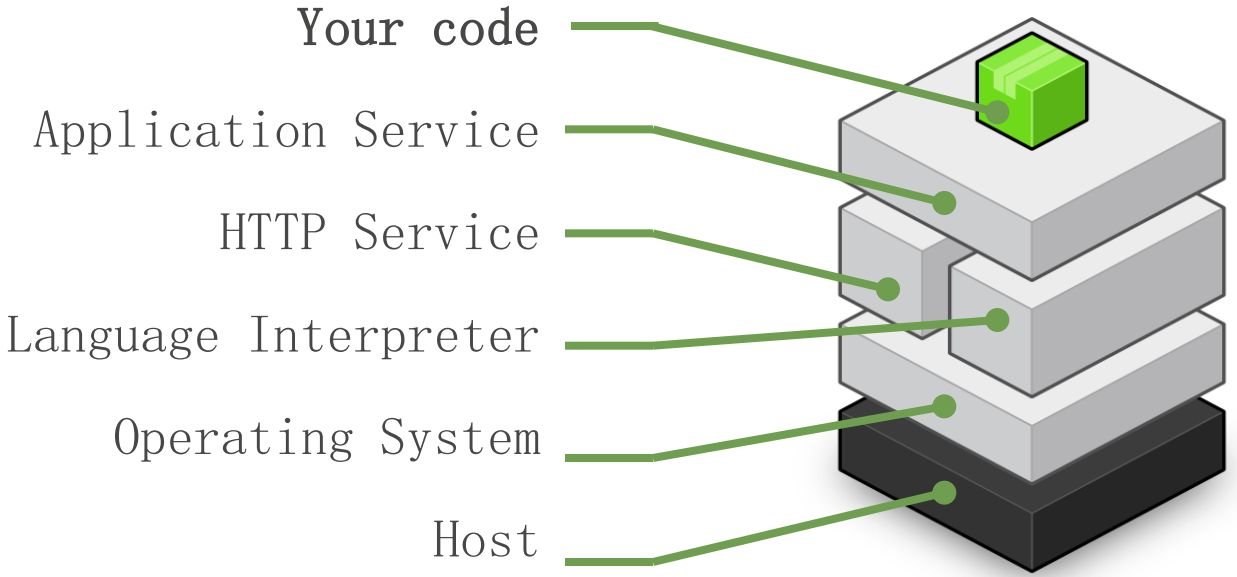
Configuration

- Settings define how an env. and resources behave
- E.g. auto-scaling, instance types, database configuration
- One per env. only

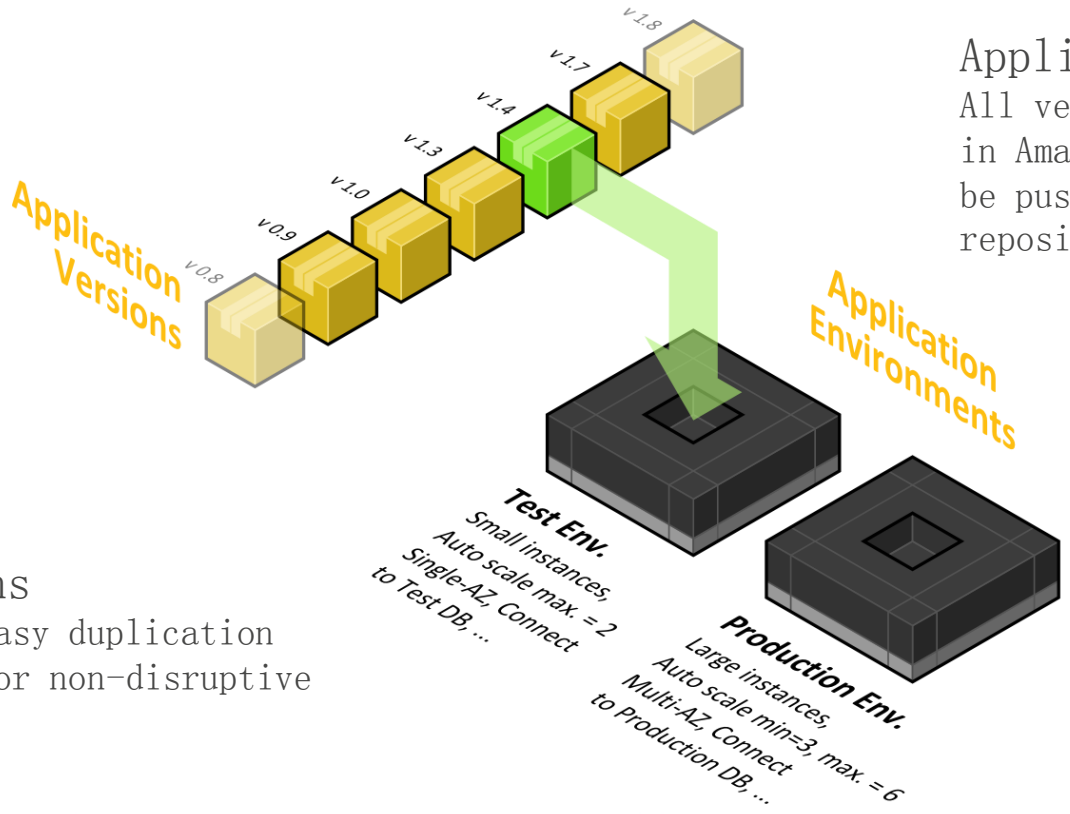
Template(s)

- Saved configurations that can be loaded to launch new environments quickly or roll-back settings

Under the hood



App Versions & Environments



Application Versions

All versions stored durably in Amazon S3. Code can also be pushed from a Git repository!

Environments Configurations

Save these for easy duplication for A/B testing or non-disruptive deployments

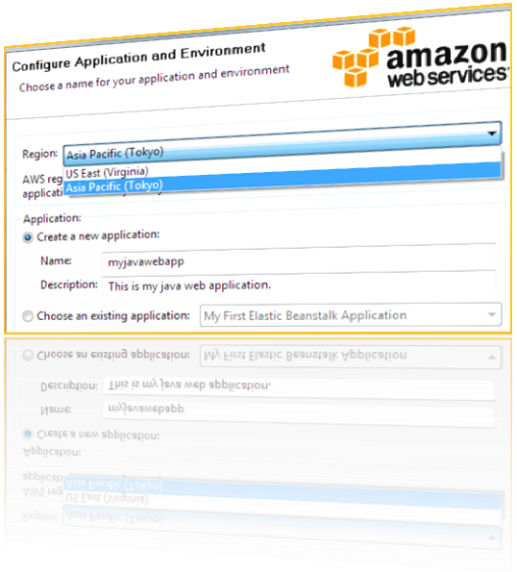
Deployment Options

1. Via AWS Console

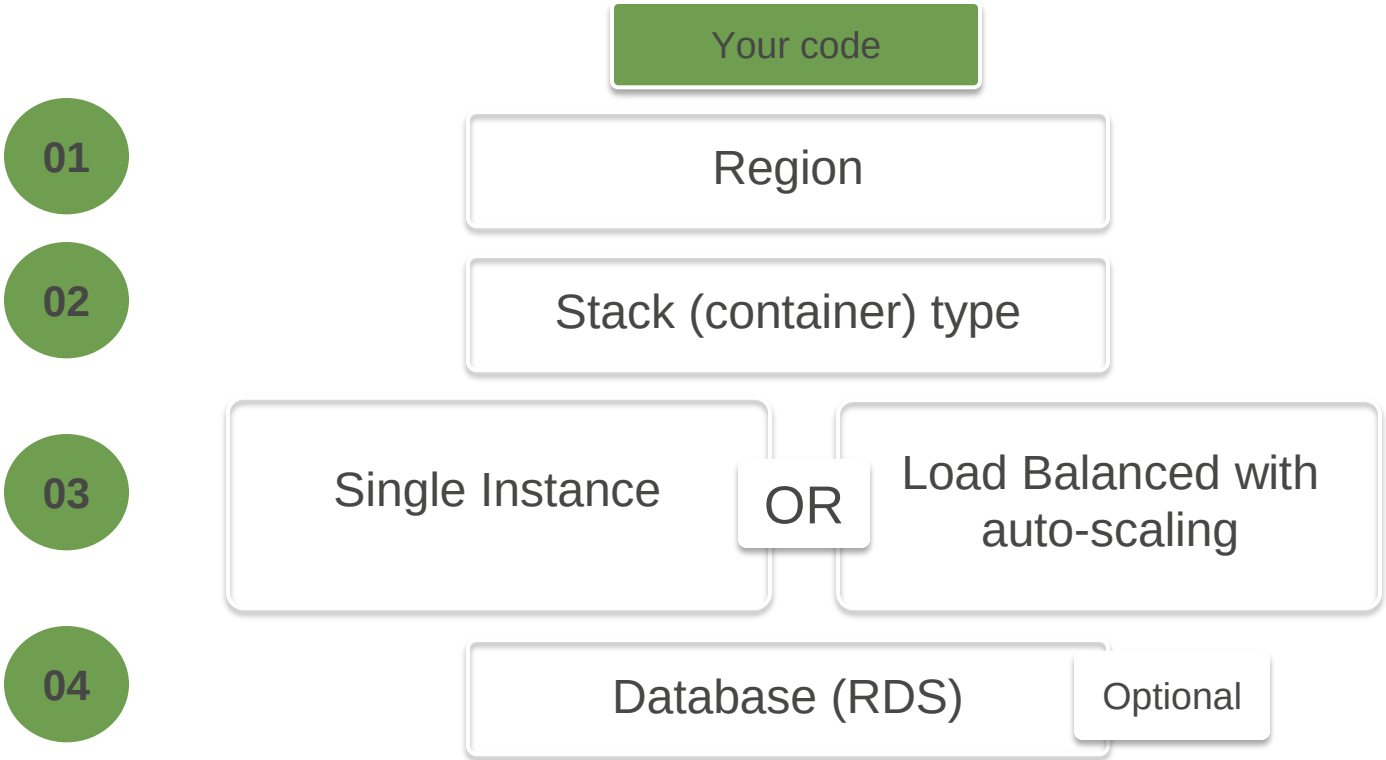
2. Via Git / EB CLI

```
$ git aws.push
```

1. Via AWS Toolkit for Eclipse and Visual Studio IDE



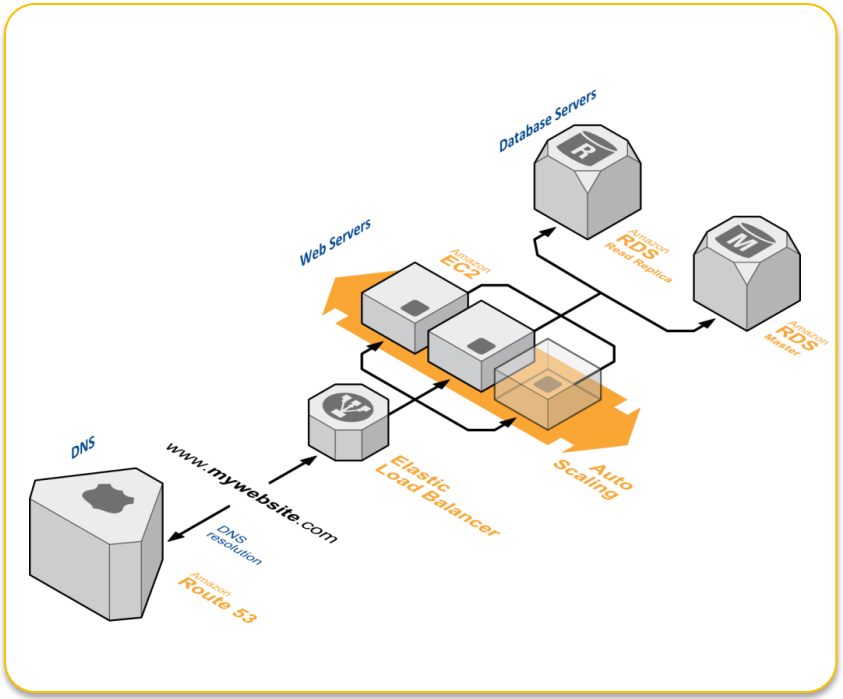
Deployment Configuration



Load Balanced/Auto-scaling Architecture

- EB deploys the load balancer, web/app servers + code, and backend database (optional)
- Creates S3 buckets for logs
- Configures Route53 and gives you a unique domain name

Yourapp.elasticbeanstalk.com



CLI Deployment Pre-requisites

1. AWS Account – your access and secret keys

2. EB CLI
   - Linux/Unix/Mac : Python 2.7 or 3.0
   - Windows : Powershell 2.0

3. A credential file containing info from 1.

4. Git 1.66 or greater (optional)

CLI App Deployment

01 Initialize your Git repository

```
$ git init .
```

02 Create your Elastic Beanstalk app

```
$ eb init
```

03 *Follow the prompts to configure the environment*

04 Add your code

```
$ git add .
```

05 Commit

```
$ git commit -m “v1.0”
```

06 Create the resources and launch the application

```
$ eb start
```

Update Your App

01 *Update your code*

02 Push the new code

```
$ git add .
$ git commit -m “v2.0”
$ git aws.push
```

03 *Monitor deployment progress*

```
$ eb status
```

Zero Downtime Deployments

1. Create a new environment for an existing application

1. Deploy your updated application code to the new environment

1. Then use the “Swap URLs” feature to transition users to the new production environment

Iterate on your Architecture

Customize Elastic Beanstalk environments using ebextensions

•Add custom software/frameworks etc. to instances

•Add other components like in memory caching (Amazon Elasticache Redis and Memcached), distributed queue - Amazon SQS and Amazon CloudFront to improve the scalability of your app

```
Resources:
  MyElastiCache:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      CacheNodeType:
         Fn::GetOptionSetting:
             OptionName : CacheNodeType
             DefaultValue: cache.m1.small
      NumCacheNodes:
           Fn::GetOptionSetting:
             OptionName : NumCacheNodes
             DefaultValue: 1
      Engine:
           Fn::GetOptionSetting:
             OptionName : Engine
             DefaultValue: memcached
```