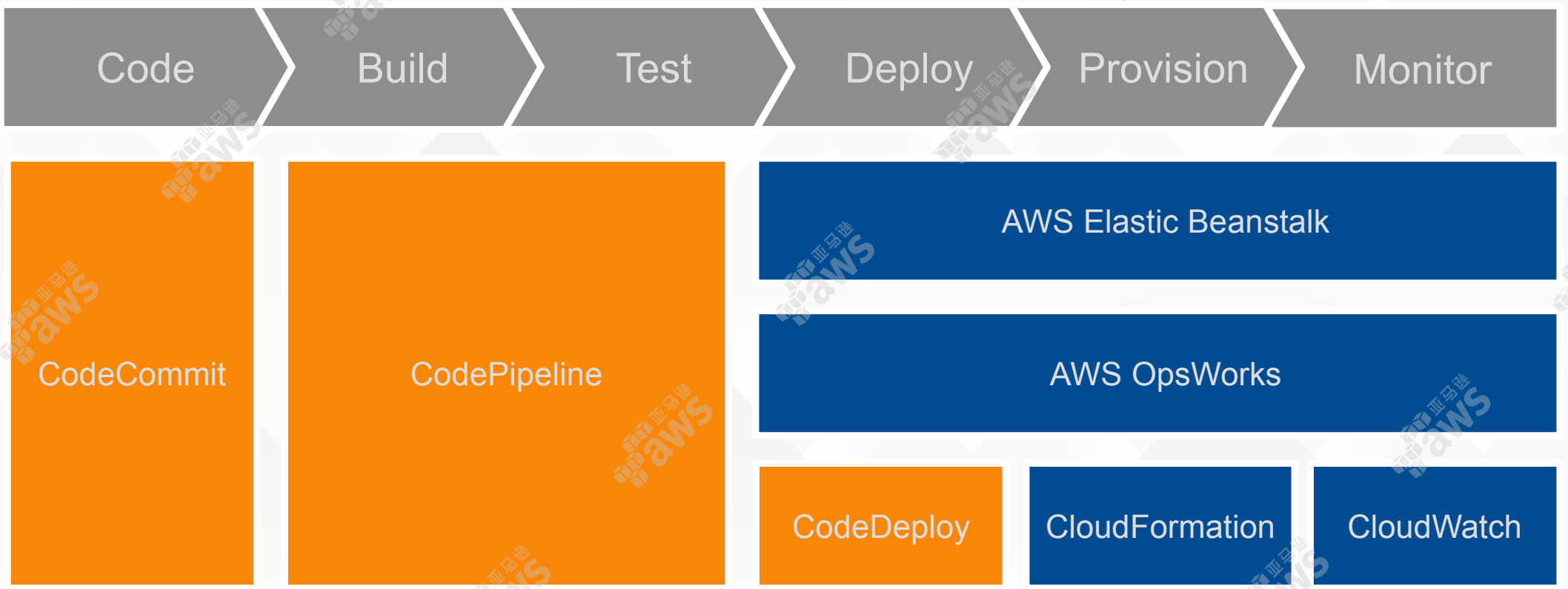

AWS Summit
AWS技术峰会2015 · 上海
亚马逊
aws

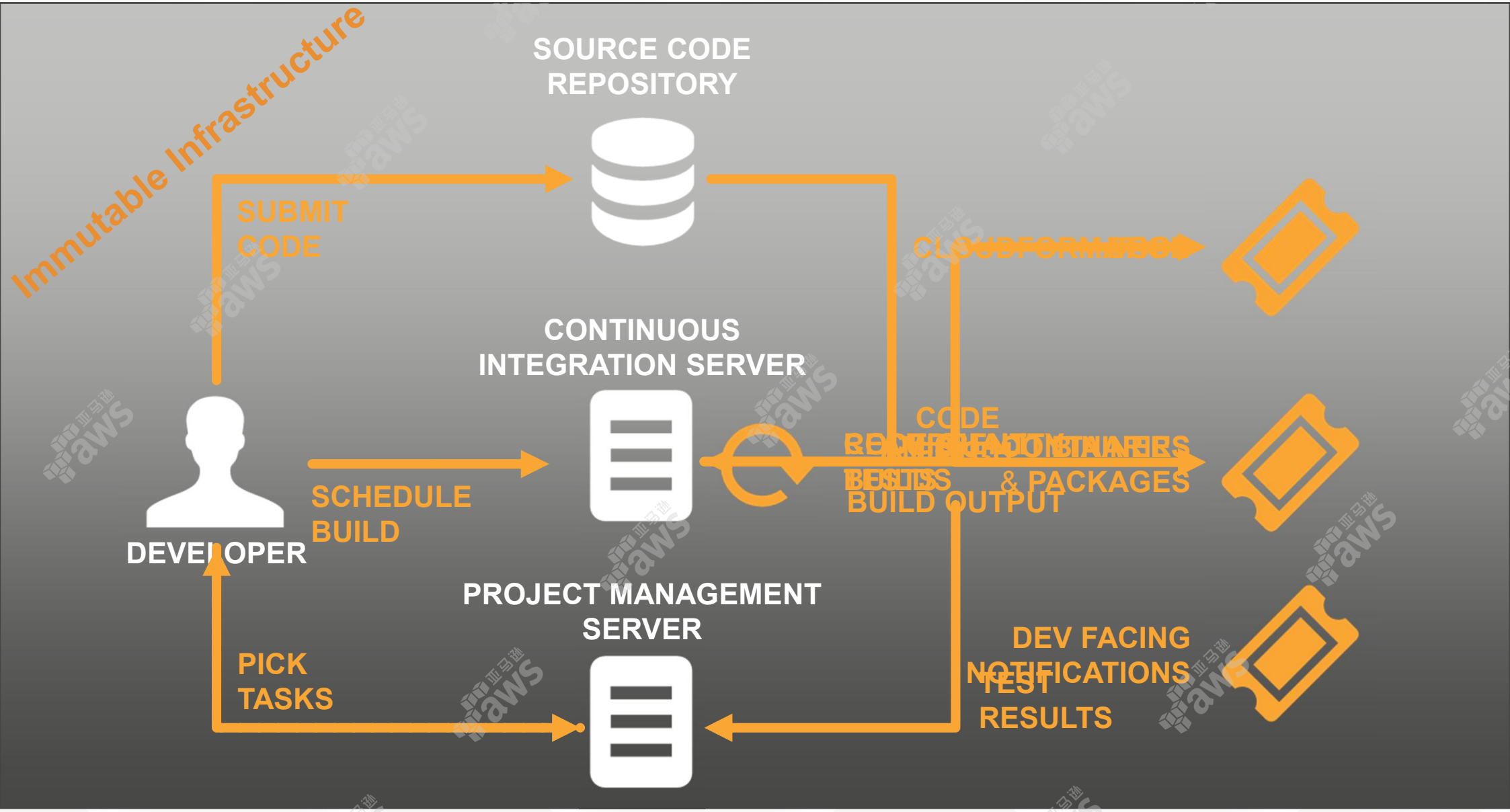
基于AWS的持续集成、交付和部署

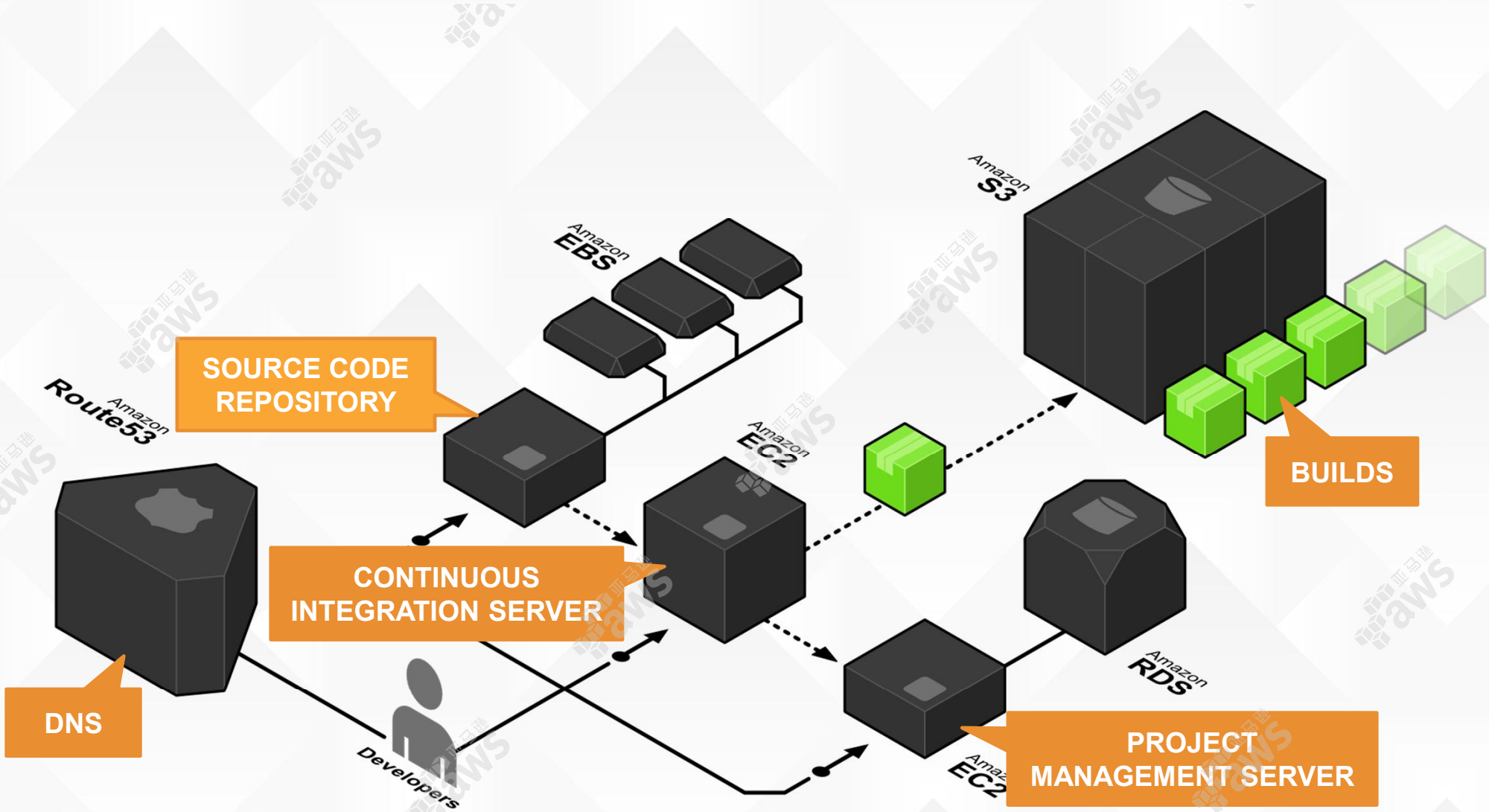
代闻

AWS解决方案架构师

软件开发的生命周期与AWS服务支持

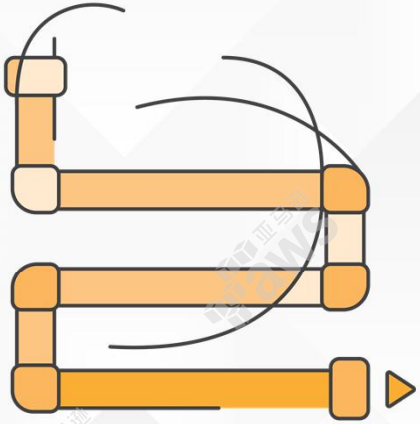



Immutable Infrastructure
SOURCE CODE REPOSITORY
SUBMIT CODE
CONTINUOUS INTEGRATION SERVER
SCHEDULE BUILD
DEVELOPER
PROJECT MANAGEMENT SERVER
PICK TASKS
& PACKAGES
BUILD OUTPUT
DEV FACING NOTIFICATIONS
TEST RESULTS


Amazon S3
Amazon EBS
SOURCE CODE REPOSITORY
Amazon Route53
Amazon EC2
BUILDS
CONTINUOUS INTEGRATION SERVER
DNS
Developers
Amazon RDS
PROJECT MANAGEMENT SERVER

AWS code services

AWS CodeCommit



AWS CodePipeline

AWS CodeDeploy

持续集成

- 代码变更通过Unit Test和Mock Test后，自动部署到主线分支（mainline branch）
- 对代码和部署进行变更时，采用迭代的方式
- 快速发现Bug
- 加速自动化部署
- 支持快速开发和部署

源码控制 - CodeCommit

Private Git repositories hosted on Amazon S3

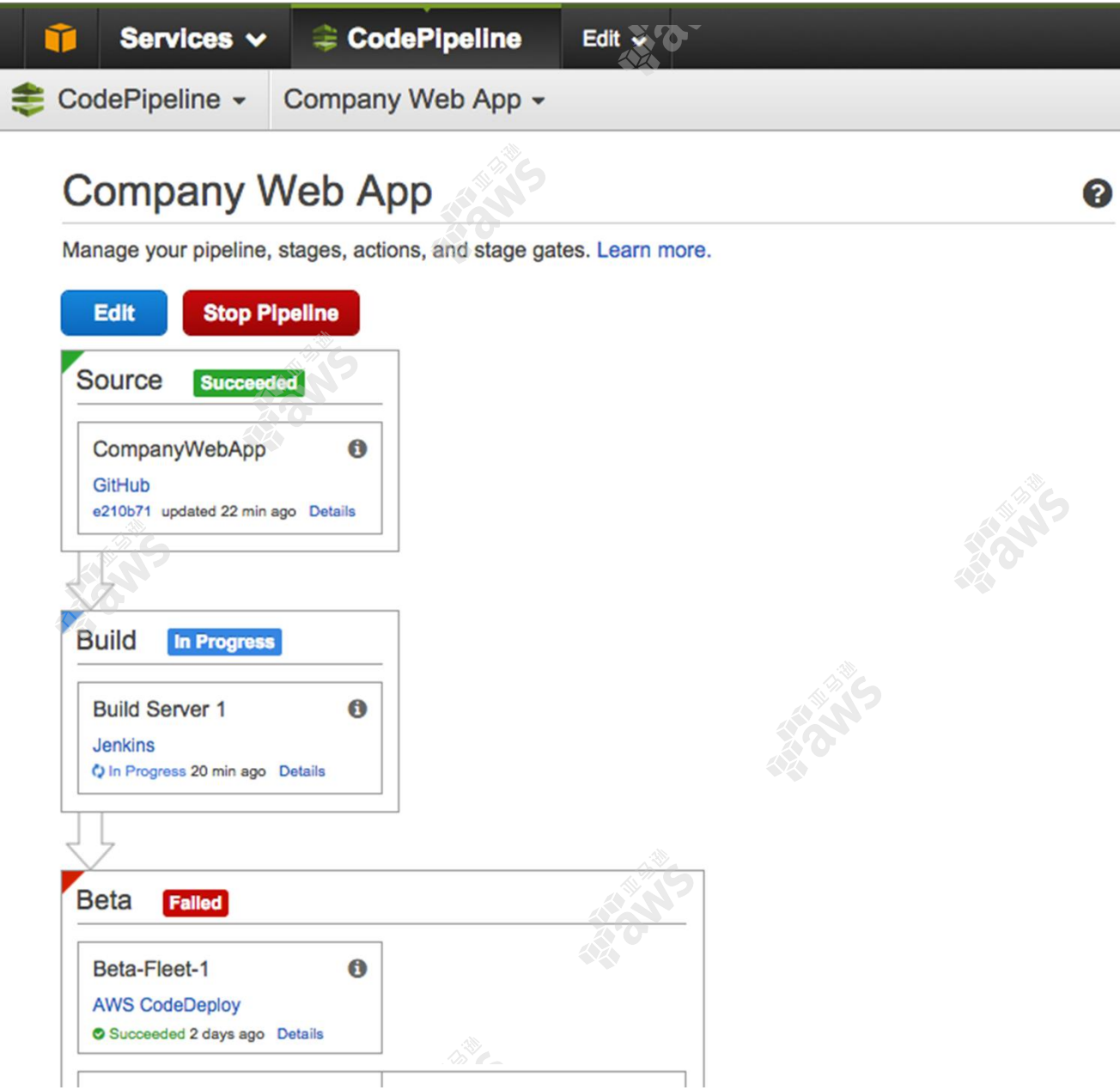
- 基于Amazon的私有Git Repositories
- 完全兼容Git
- 充分利用云的优势 (扩展性,持久性, 可靠性, 按需付费以降低成本)
- Repositories没有大小限制
- 在线代码工具，支持browse, edit, diff

CodeCommit 示例

```
$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/aws-cli
Cloning into 'aws-cli'...
Receiving objects: 100% (16032/16032), 5.55 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (9900/9900), done.
Checking connectivity... done.
$ nano README.rst
$ git commit -am 'updated README'
[master 4fa0318] updated README
 1 file changed, 1 insertion(+)
$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 297 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/aws-cli
   4dacd6d..4fa0318  master -> master
```

持续交付 - CodePipeline

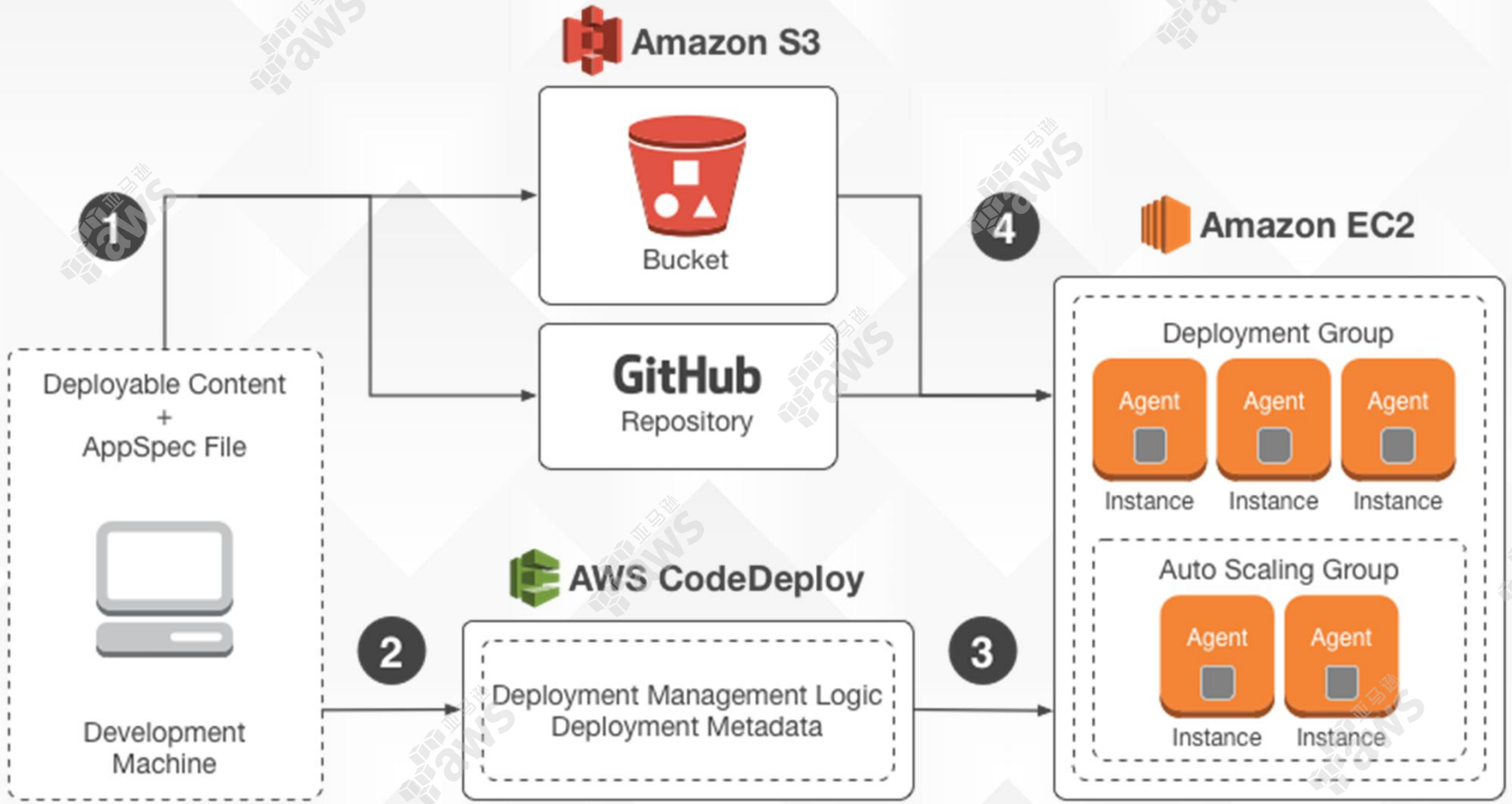
- 可自定义的自动化版本发布，并且集成了编译和测试
- 对自定义的版本发布工作流建模、可视化
  (源代码 ➔ 编译 ➔ beta ➔ gamma ➔ 线上生产)
- 自动化编译、测试和部署
- 执行自定义规则
- 与第三方工具集成


Services
CodePipeline
Edit
N.Virginia
Help
CodePipeline
Company Web App
Company Web App
Manage your pipeline, stages, actions, and stage gates. Learn more.
Edit
Stop Pipeline
Source
Succeeded
CompanyWebApp
GitHub
e210b71 updated 22 min ago Details
Build
In Progress
Build Server 1
Jenkins
In Progress 20 min ago Details
Beta
Failed
Beta-Fleet-1
AWS CodeDeploy
Succeeded 2 days ago Details

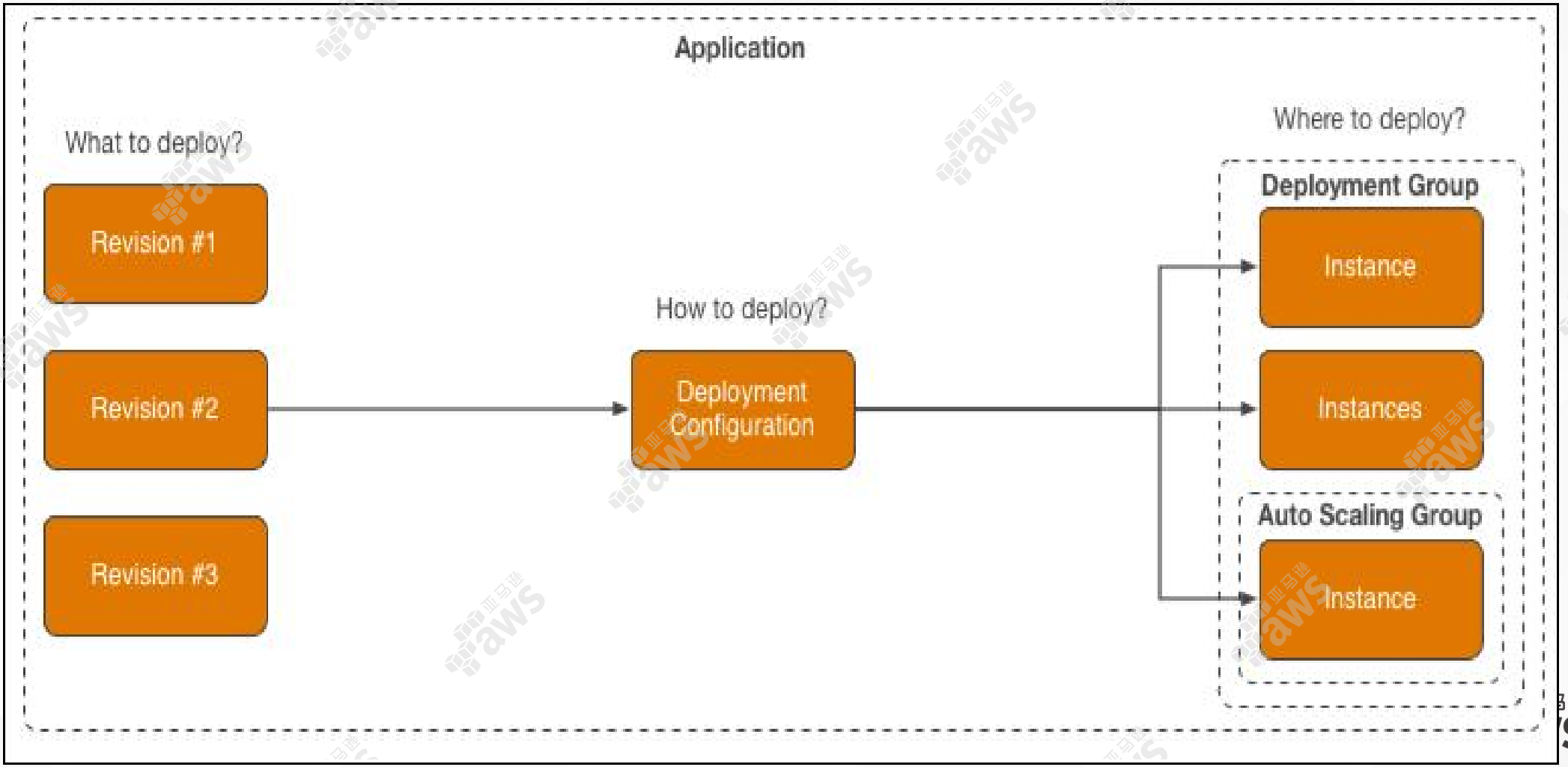
持续部署 - CodeDeploy

- 协调服务器的软件更新，滚动更新以避免宕机时间
- 针对部署的健康检测，以及回滚
- 集成自动扩展（Auto Scaling）
- 对所有应用程序适用
- 重用已有的部署工具
(Bash, Powershell, Chef, Puppet…)

工作流


Amazon S3
Bucket
GitHub
Repository
1
4
Amazon EC2
Deployment Group
Agent
Agent
Agent
Instance
Instance
Instance
Auto Scaling Group
Agent
Agent
Instance
Instance
Deployable Content
+
AppSpec File
Development
Machine
2
AWS CodeDeploy
Deployment Management Logic
Deployment Metadata
3

基本原理


Application
What to deploy?
Revision #1
Revision #2
Revision #3
How to deploy?
Deployment Configuration
Where to deploy?
Deployment Group
Instance
Instances
Auto Scaling Group
Instance

Deployments

View information about your deployments, deployment groups, and deployment events. Diagnose problems, create new deployments, and stop in-progress deployments.

Create New Deployment | CodeDeploy Resources

Filter: All Deployments | Search by Deployment ID | Deployments per page 10 | Viewing 6 Deployments

Deployment ID	Application	Deployment Group	Revision Location	Start Time	End Time	Status	Actions
d-QTUFJ6RR4	App_WordPress	WordPress_InstGroup_us-we ...	s3://SDSDeploymentBucket/WordPress ...	May 28, 2014 10:00:00 AM UTC	May 28, 2014 11:00:00 AM UTC	Failed	

Details

Health Constraint Error
Unable to continue deployment without violating minimum healthy host constraints, as a result 1 host was skipped.
Learn More

Deployment ID d-QTUFJ6RR4
Deployment Config Half at a Time
Minimum Healthy Hosts 50%
Revision Location s3://SDSDeploymentBucket/WordPressApp.zip/f0JPvV1ZjSu732H5CL.jGcyrOOpwBGpD
Revision Created May 27, 2014 5:00:00 PM UTC
Description Changes to global navigation. Updated headers, and security updates.

Instances

10 of 10 Instances Completed

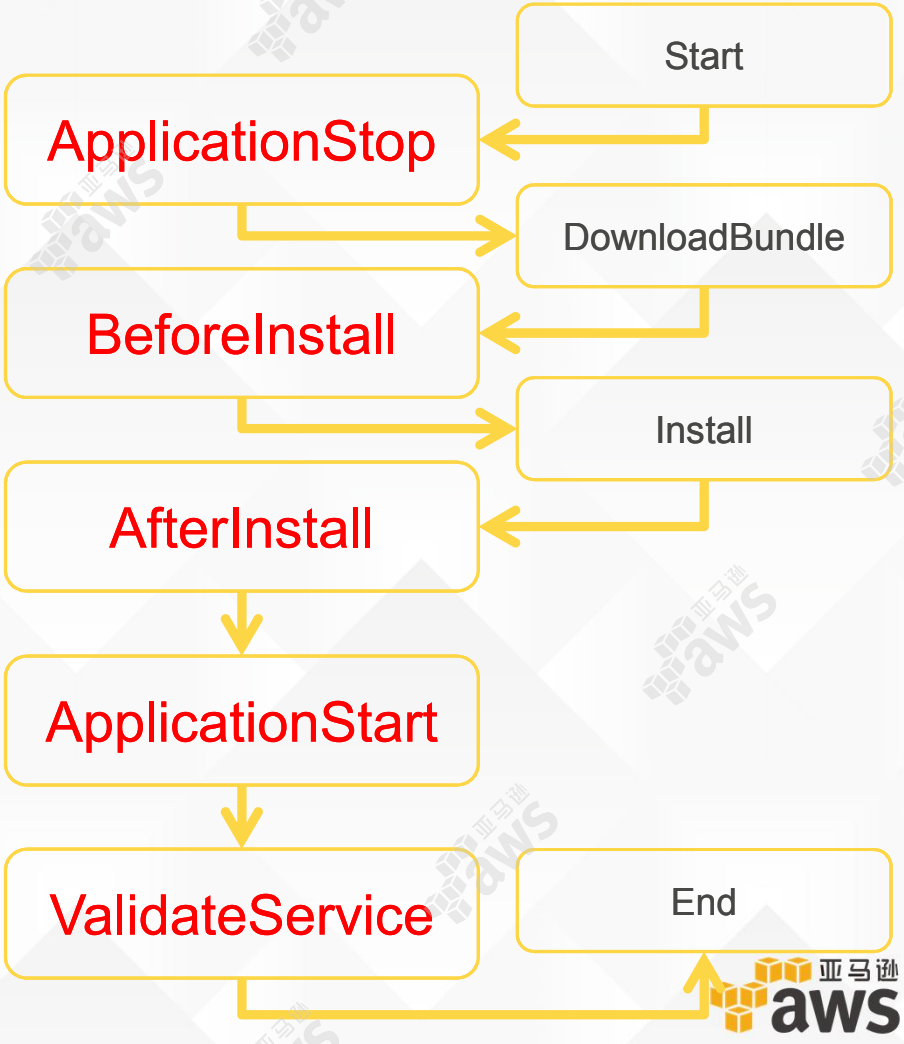
1 Succeeded | 5 Failed | 4 Skipped

View Deployment Details

d-XZM1E04W5	App_WordPress	WordPress_InstGroup_ea	s3://SDSDeploymentBucket/WordPress ...	May 13, 2014 1:23:01 PM UTC	May 13, 2014 2:46:01 PM UTC	Succeeded	

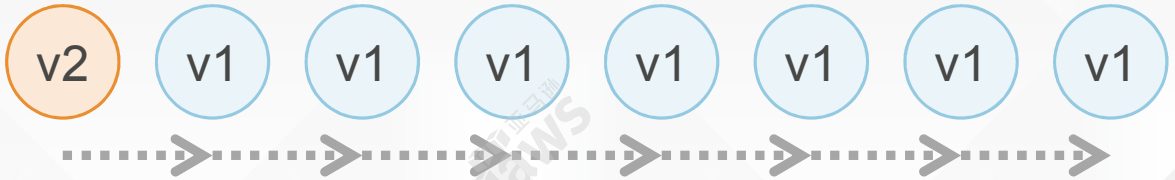
AppSpec file

```
os: linux
files:
  - source: Config/config.txt
    destination: webapps/Config
  - source: Service
    destination: /webapps/DogSuit
hooks:
  ApplicationStop:
    - location: Scripts/Deactivate_Service
  BeforeInstall:
    - location: Scripts/Flush_Logs.sh
  AfterInstall:
    - location: Scripts/Decrypt_Secrets.sh
  ApplicationStart:
    - location: Scripts/Start_Pooch_Service.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/Check_Dogs_Barking.sh
      runas: codedeployuser
```



Deployment Config

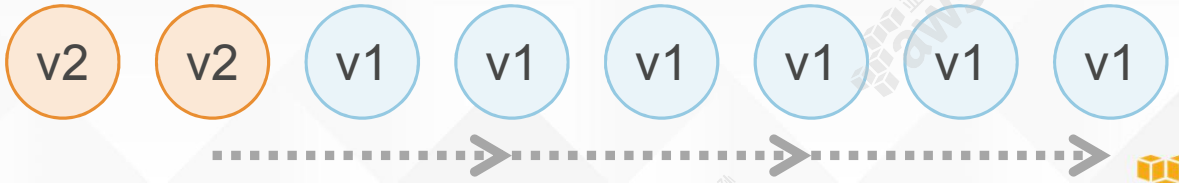
One-at-a-time
Min. healthy hosts = 99%



Half-at-a-time
Min. healthy hosts = 50%

All-at-once
Min. healthy hosts = 0

[Custom]
Min. healthy hosts = 75%



第三方工具集成

GitHub

Atlassian

CloudBees

Travis CI

CODESHIP

Solano Labs

circleci

Jenkins

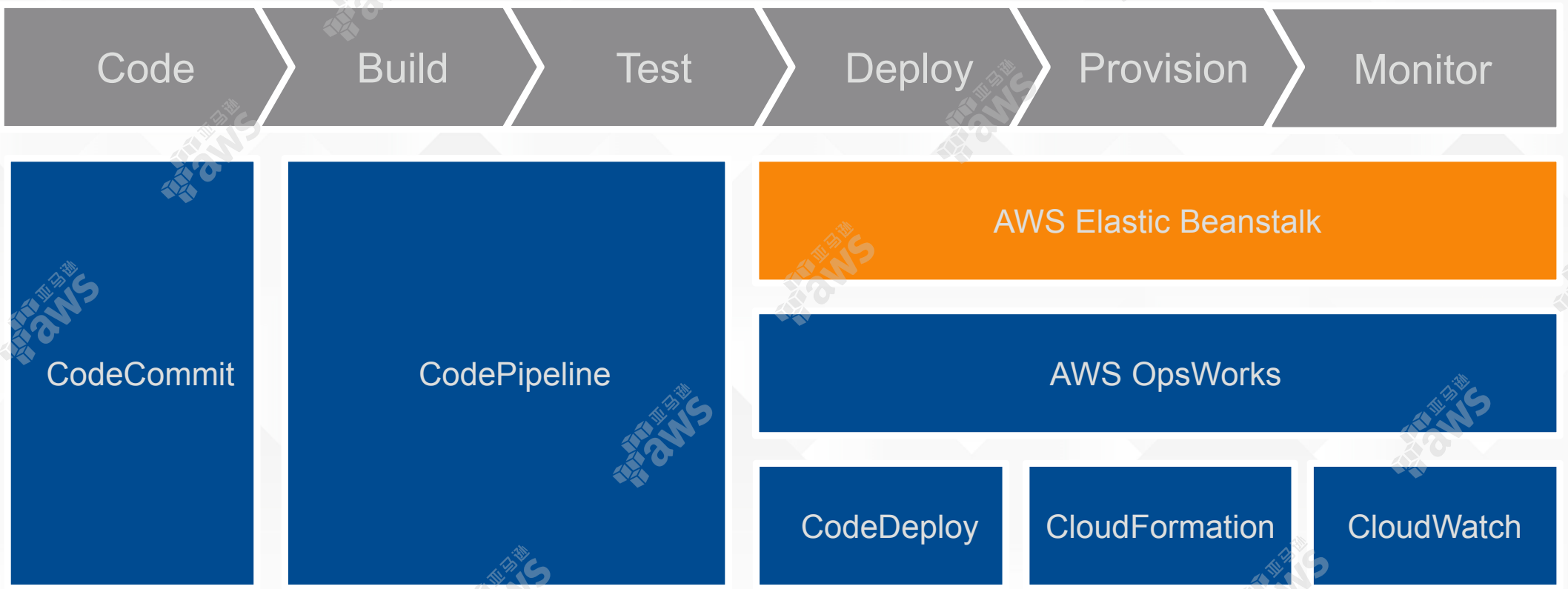
CHEF

puppet labs

ANSIBLE

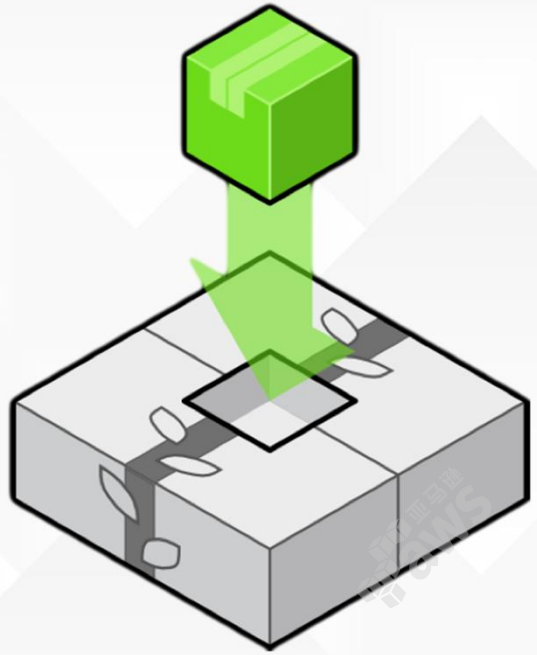
SALTSTACK

软件开发的生命周期与AWS服务支持



AWS Elastic Beanstalk (EB)

- Easily deploy, monitor, and scale three-tier web applications and services.
- Infrastructure provisioned and managed by EB – but you maintain complete control.
- Preconfigured application containers that are easily customizable.
- Support for these platforms:



Elastic Beanstalk 基本概念与模型

Application

Environments

- Infrastructure resources (such as EC2 instances, ELB load balancers, and Auto Scaling groups)
- Runs a single application version at a time for better scalability
- An application can have many environments (such as staging and production)

Application versions

- Application code
- Stored in Amazon S3
- An application can have many application versions (easy to rollback to previous versions)

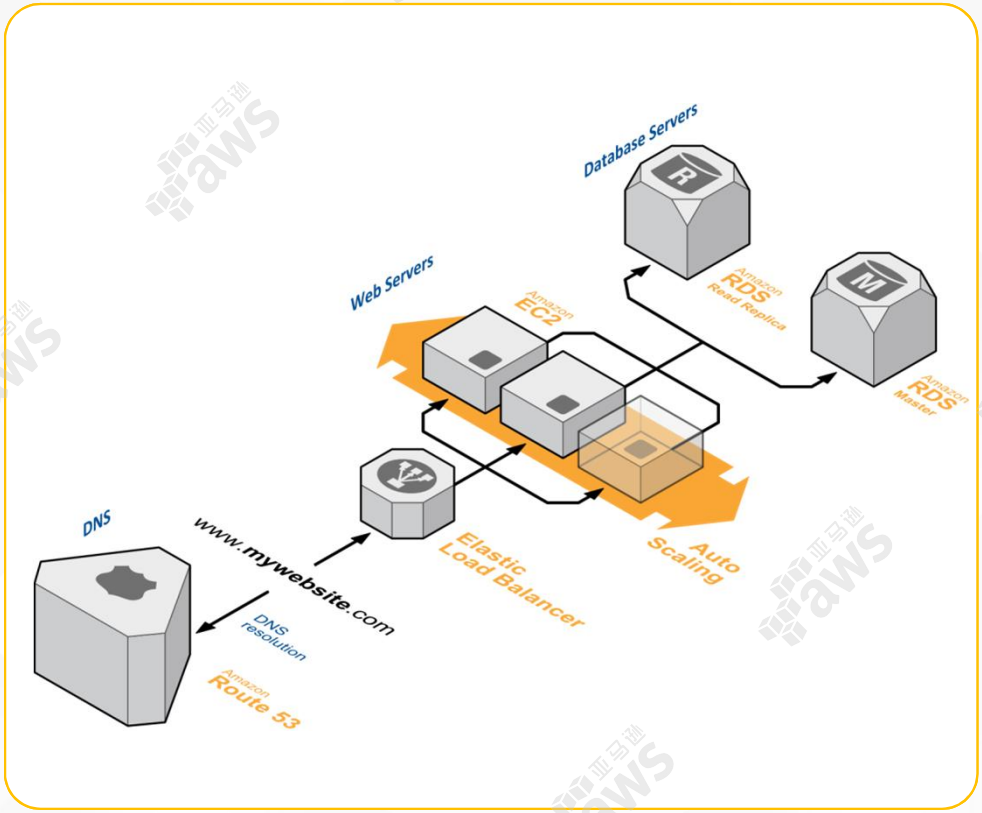
Saved configurations

- Configuration that defines how an environment and its resources behave
- Can be used to launch new environments quickly or roll-back configuration
- An application can have many saved configurations

Elastic Beanstalk environment

- Two types:
  - Single instance
  - Load balancing, auto scaling
- Two tiers (web server and worker)
- Elastic Beanstalk provisions necessary infrastructure resources such as load balancers, auto-scaling groups, security groups, and databases (optional)
- Configures Amazon Route 53 and gives you a unique domain name

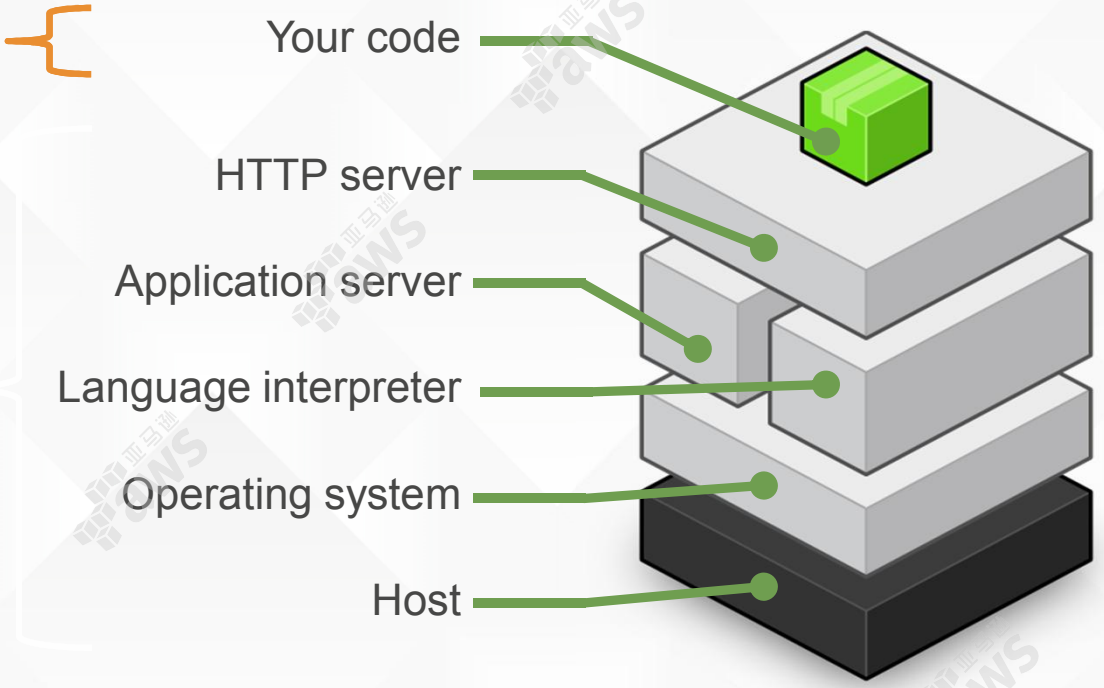
  (For example: yourapp.elasticbeanstalk.com)



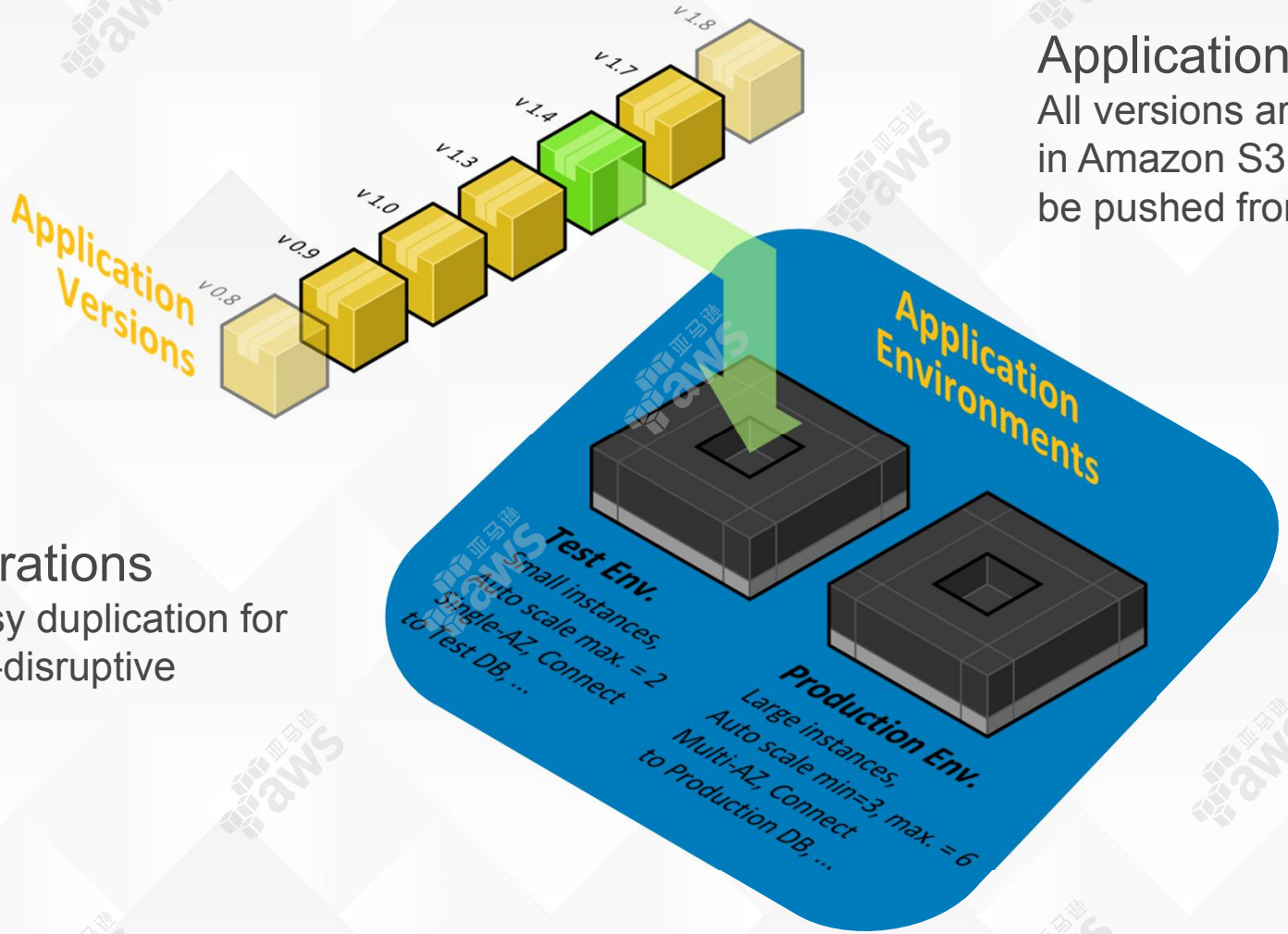
On-instance configuration

Focus on building your application

- Elastic Beanstalk configures each EC2 instance in your environment with the components necessary to run applications for the selected platform
- No more worrying about logging into instances to install and configure your application stack



应用版本与已存配置



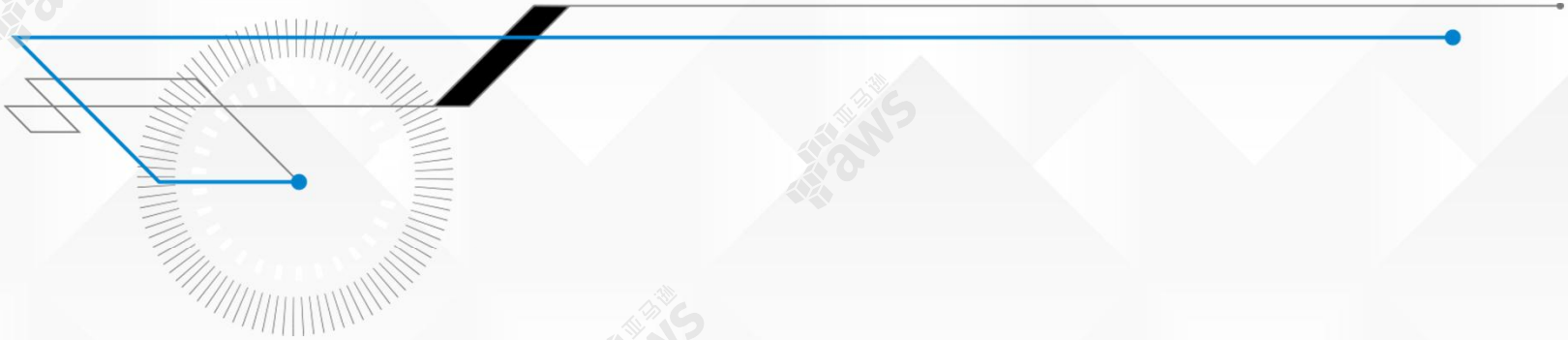
Application versions

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

Saved configurations

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

开发者工作流

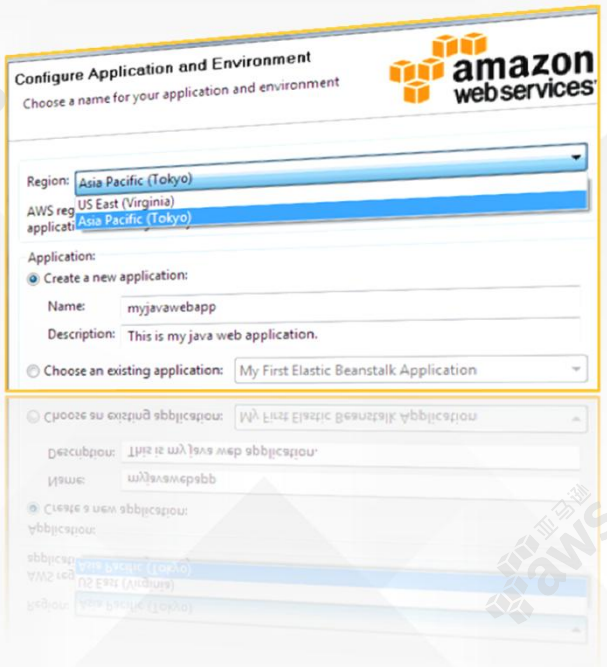


Deployment options

1. Via the AWS Management Console
2. Via Git / EB CLI

```
$ git aws.push
```

3. Via the AWS Toolkit for Eclipse and the Visual Studio IDE



Deployment configuration

Your code

01 Region

02 Stack (container) type

03 Single instance OR Load balanced with Auto Scaling

04 Database (RDS) Optional

Example: CLI workflow

Prerequisites:

MB12

- AWS account – your access and secret keys
- EB CLI
  - Linux / Unix / Mac: Python 2.7 or 3.0
  - Windows PowerShell 2.0
- A credential file containing info from 1
- Git 1.66 or later (optional)

MB12 These should be bullets because they are not sequential steps.
Mcguire, Barbara, 2014/10/27

Example: CLI workflow

Initial 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 create
```

Example: CLI workflow

Update your app:

01 Update your code

02 Push the new code

```
$ git add .
$ git commit -m “v2.0”
$ eb deploy
```

03 Monitor the deployment progress

```
$ eb status
```

Example: Deploy Docker container to EB

- Three ways:
  - Dockerfile *(image built on instance).*
  - Dockerrun.aws.json *(manifest file that describes how to run the Docker image).*
  - Application archive *(should include Dockerfile or Dockerrun.aws.json file).*
- Benefits:
  - Enables high-fidelity deployments.
  - You own the runtime. You can use any language or framework, even those not currently supported by Elastic Beanstalk (such as Go, Scala, and Clojure).

Dockerfile

```
FROM dockerfile/nginx

#Add custom index.html
ADD index.html /usr/share/nginx/html/
```

Dockerrun.aws.json

```
{
    "Image":{
        "Name" : "example/wordpress"
    },
    "Ports" : [
        {
            "ContainerPort" : "80"
        }
    ]
}
```

Example: Deploy Docker container to EB

Using the EB command line tool:

01 Initialize your Git repository

```
$ git init .
```

02 Create your Elastic Beanstalk app

```
$ eb init
```

03 *Follow the prompts to configure the environment and copy Dockerfile*

04 Add your code

```
$ git add Dockerfile
```

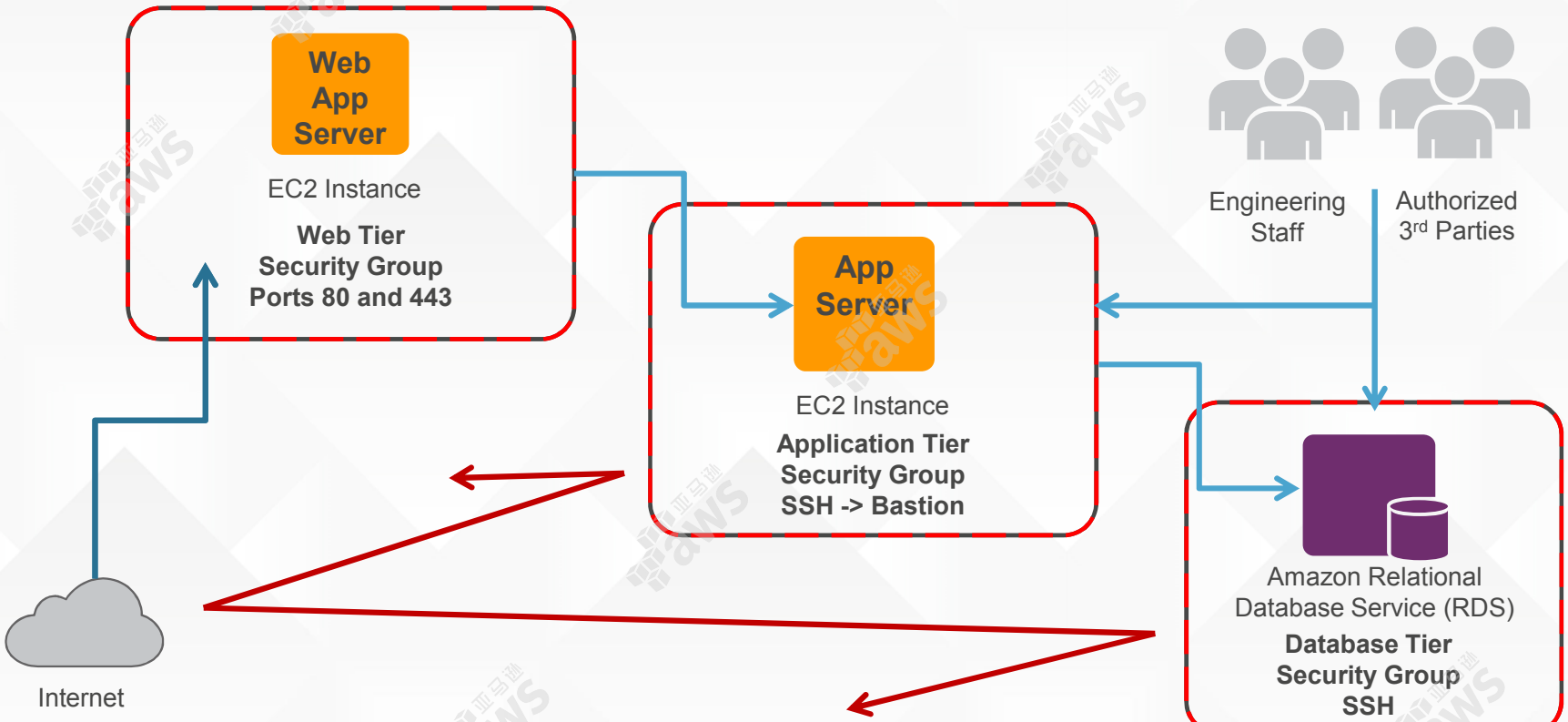
05 Commit

```
$ git commit –am “v1.0”
```

06 Create the resources and launch the application

```
$ eb create
```

架构设计

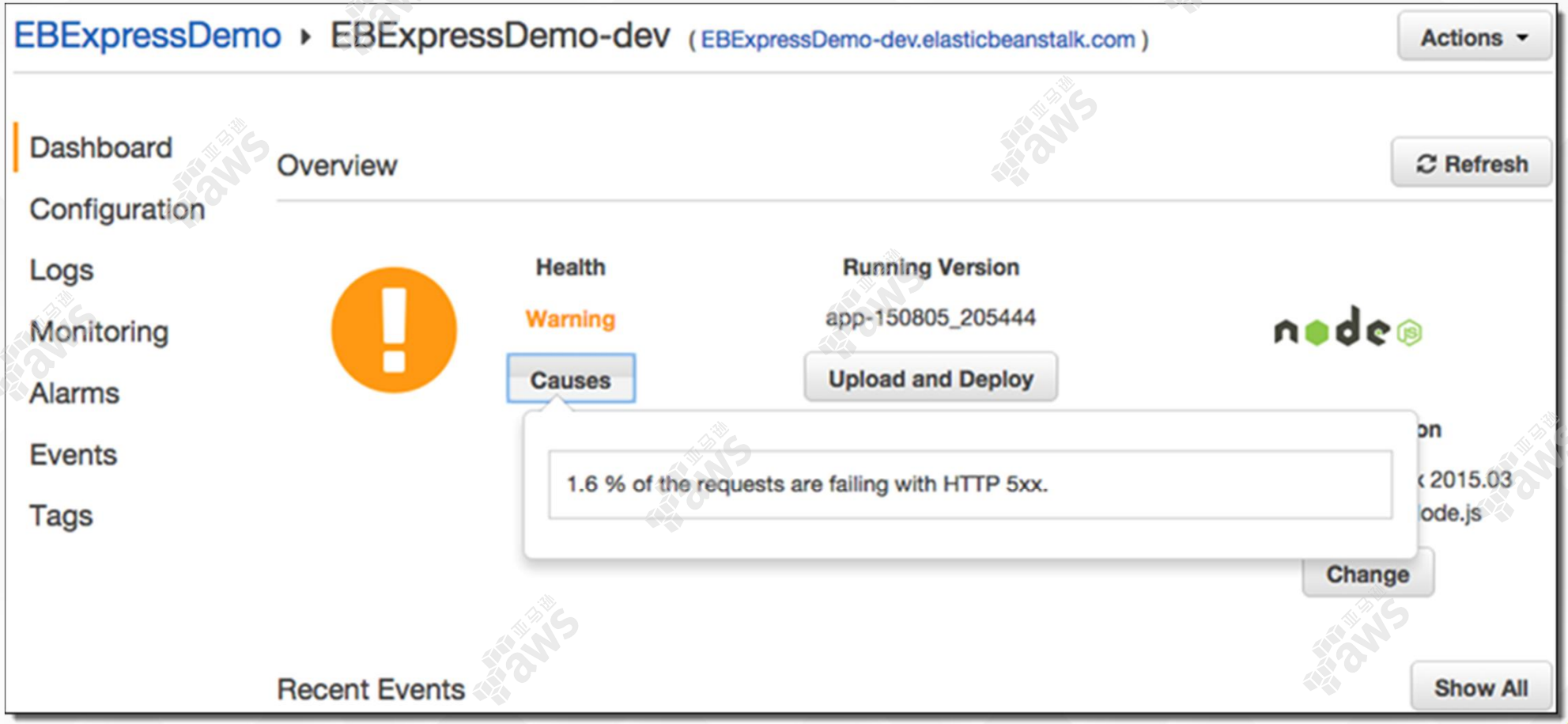

Web
App
Server
EC2 Instance
Web Tier
Security Group
Ports 80 and 443
App
Server
EC2 Instance
Application Tier
Security Group
SSH -> Bastion
Engineering
Staff
Authorized
3rd Parties
Amazon Relational
Database Service (RDS)
Database Tier
Security Group
SSH
Internet

ElasticBeanstalk 的特点

- 托管的基础设施
- 代码和应用版本化管理
- 基础环境版本化管理
- 自动化运维，平滑部署（如基于DNS的蓝绿部署）

Elastic Beanstalk 对基础环境的监控

Elastic Beanstalk对应用的监控



Thank You