用基础设施即代码
自动化架构迁移

ThoughtWorks 高级咨询师
顾宇

关于我

- ThoughtWorks 高级咨询师。
- 在 ThoughtWorks 接受了良好的软件开发教育之后基本放弃写代码。
- 开源爱好者，经济学票友，咖啡师，非著名电影导演。
- 专注于DevOps，SRE，持续交付，微服务以及全功能产品团队的设计、实践、落地以及经验推广。

案例背景 - 跨国互联网公司并购

多国，多语言，多IT团队

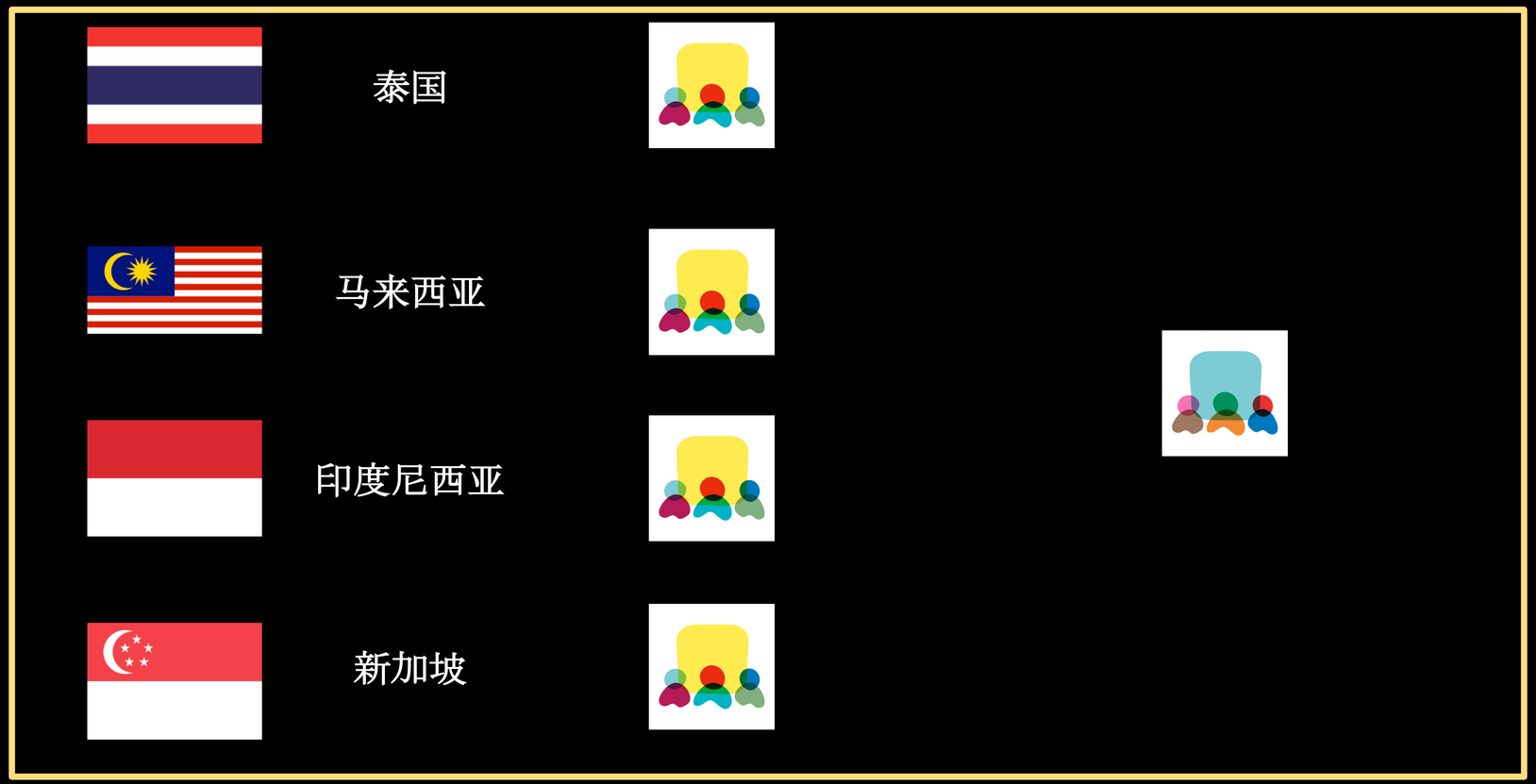
泰国

马来西亚

印度尼西亚

新加坡

多国，多语言，单IT团队



还记得康威定律吗？

Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

- Melvin Conway(1967)

设计系统的组织，其产生的设计等同于组织之内、组织之间的沟通结构

多国，多语言，多站点

泰国

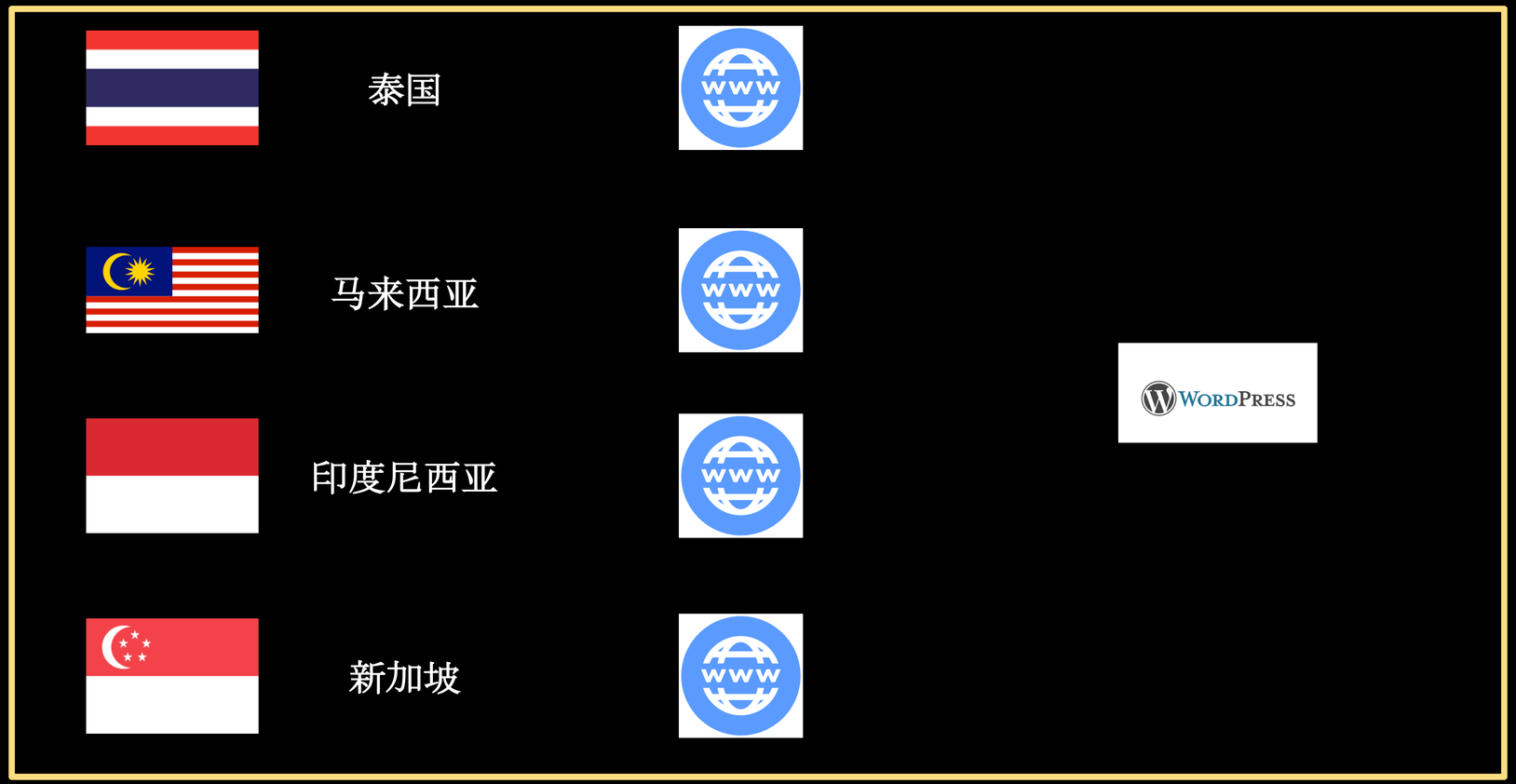
马来西亚

印度尼西亚

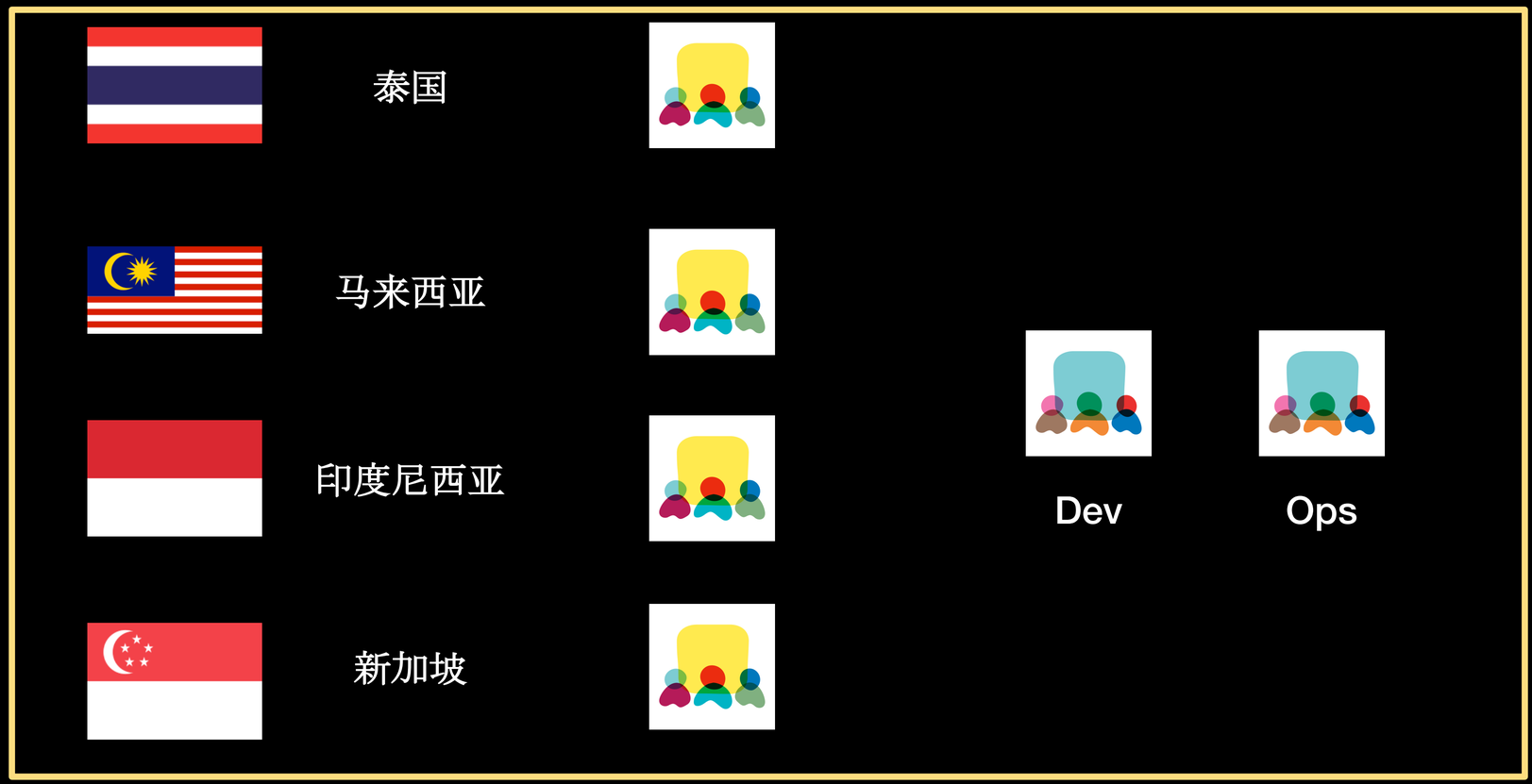
新加坡

并购不仅仅是把团队合并到一起

多国，多语言，单站点



多国，多语言，单站点



DevOps 的两种组织形式

- Team Own Ops
- Team Share Ops

新的组织目标

- 单一产品开发团队，各地区共享。
- 共享运维团队，多个业务/地区共享。
- 构建 DevOps （从 Team Own Ops 到 Team Share Ops 变化）
- 提升 Dev / Ops / DevOps 技能和水平

通过 DevOps 加速合并进程

- 语言不同，技术栈相同，实践相同
- TDD 在多样性开发团队间帮了大忙
- 用自动化作为制度提升效率

当组织结构和系统架构不一致，
就需要进行架构迁移

架构迁移 inception

- 了解当前的现状
- 明确系统架构的目标
- 设计迁移策略
- 为新组织设计系统架构

当前的现状

当前的组织现状

- 少量运维人员 维护大量技术栈/语言/业务各异的站点
- 风格各异的遗留系统
- 区域大体相似和定制化业务

当前架构的问题

- 一个架构师，很多的隐藏知识
- 基础架构即代码里有很多硬编码
- 整体、非部分更新

架构迁移的目标

目标

- 采用基础设施即代码管理基础设施
- 将现有基础设施架构迁移到云上
- 规范基础设施的管理
- 能够实现部分更新
- 能力和组织结构的迁移

把基础设施变成一种产品

- 基础设施架构经验的复用，得到更安全，更稳定，更灵活的架构。
- 高度自动化，可以快速建设和定制。
- 关注点分离，根据场景把变更缩小在一定的范围里。
- 开发和各环境抽象一致性。

迁移策略

迁移策略原则

- 最少的变更——一个变更点
- 最少的影响——灰度发布

迁移策略

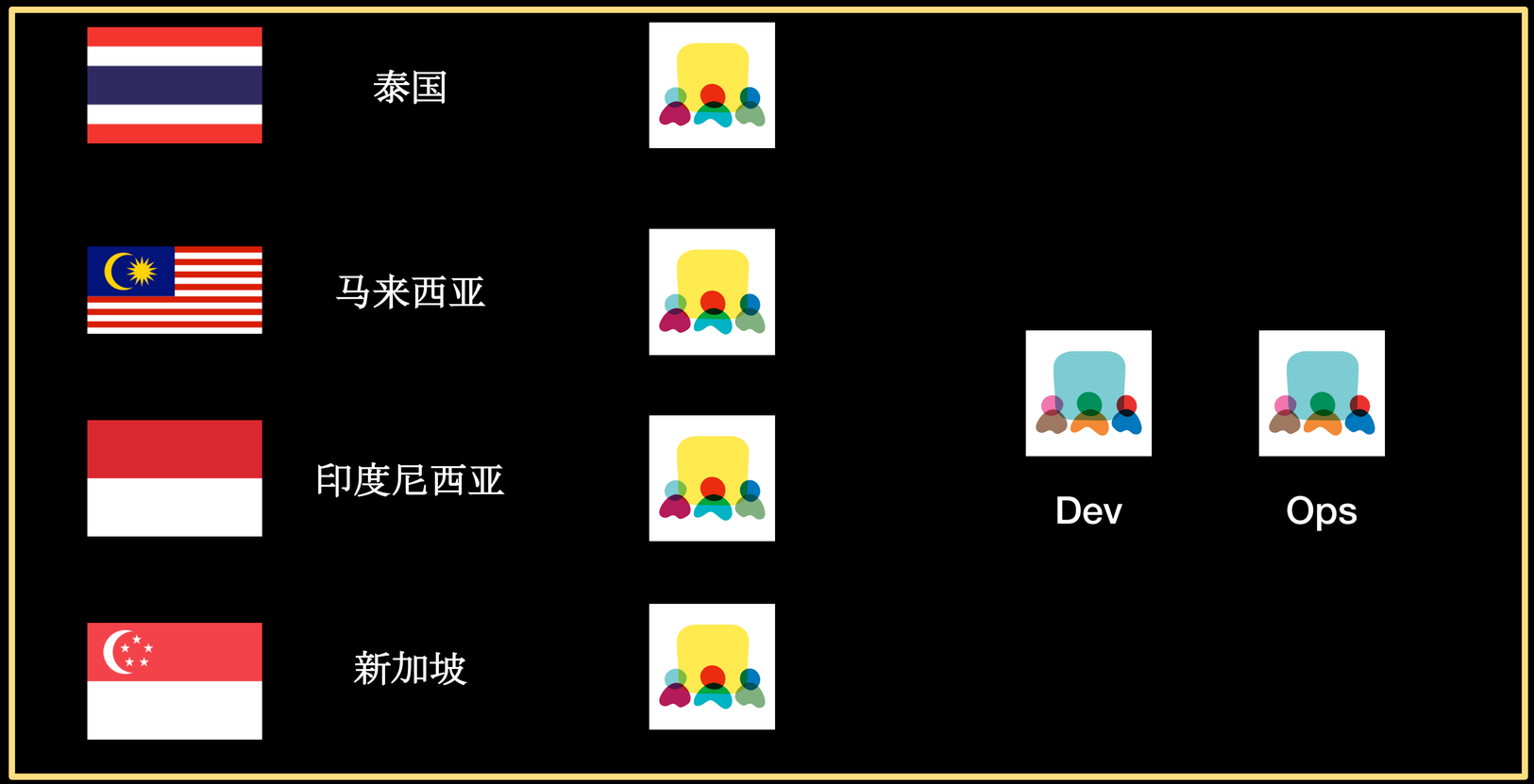
- 将遗留系统应用按照（架构/应用/数据）进行封装。
- 用基础设施即代码构建一套新的架构，构建可复用架构模型。
- 对三个部分分别用该脚本实施自动化迁移。
- 测试完成后切换总域名，切换域名。
- 两周过后，老系统停机。

迁移策略

- 用 Ansible + Cloudformation 封装并构造新架构
- 全量数据 dump / import，自动化备份到 s3 上
- 用 Docker 封装 wordpress 应用
- 用 nginx + wordpress 共同做 301、302 迁移
- 用脚本做迁移，迁移脚本分类管理
- 切换域名指向（手动）

为新的组织结构设计架构

新的组织目标



场景设计

基础设施的变更场景

- 基础设施创建/配置
- 持续交付
- 维护/故障处理

基础设施创建/配置 的场景

- 自动化完成架构生成，通过版本化配置管理完成
- 不要修改现有架构/通过生成新架构迁移

持续交付 的场景

- 构建开发环境
- 代码提交
- 自动化测试
- 构建 Docker Image
- 发布 Docker Image
- 部署/发布 Docker 镜像

维护/故障处理的场景

- 问题检查
- 数据库维护

安全是设计的首位

架构安全和安全架构

- 架构安全：关键是人的操作，对架构的影响。
- 安全架构：架构对应用的保护。

架构的安全

- 减少 AWS 系统账户，缩小权限
- 基础设施即代码创建账户，分类管理账户
- IP 白名单
- 对环境的操作只能通过基础设施即代码完成
- 再也不要用用户名/密码了
- 所有的人工使用的 key 都是临时的，用完即删除

安全的架构

- 所有的应用更改只能通过 CI 完成，包括 CI 自身的搭建
- 应用账户由非登录账户和角色构成，根据场景设计角色
- 通过堡垒机维护，堡垒机一次性使用
- 内外网分离，多区域分离

基础设施即代码

利用Ansible + Cloudformation
管理基础设施

基础设施即代码的本质

- 是一套 ROC 管理 （ Resource Oriented Computing）的类库
- 代码是类，基础设施就是对象
- 本质上是一种依赖和配置（代码）管理
- 用面向对象的方法和原则去设计你的场景（单一责任、最小知识、依赖倒置、开闭原则，接口隔离）

从命名上看 Dev 和 Ops 的思维

- SubnetPublicA
- SubnetPublicB

- PublicSubnetA
- PublicSubnetB

Dev 和 Ops 的思维

- Dev 从使用场景的角度考虑
- Ops 从资源管理的角度考虑

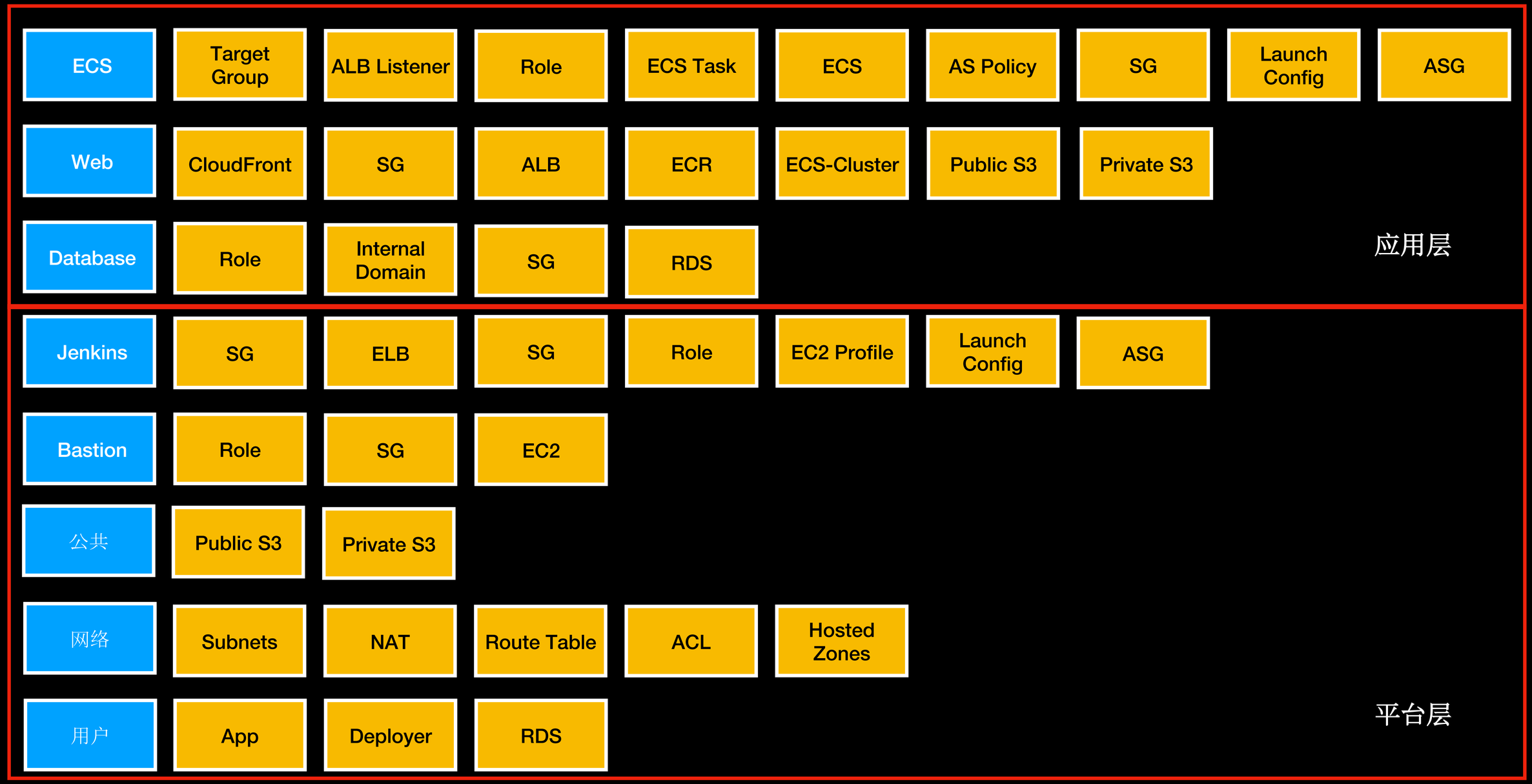
基础设施即代码

- 最少侵入性原则，避免提升复杂度
- 降低环境间的不一致
- 基础设施应当是对应用程序透明的
- 基础设施的持续交付
- 杜绝一切不规范的人为变更，减少变更步骤
- 给人为操作设置最小权限

使用 Ansible 和 Cloudformation 的原则

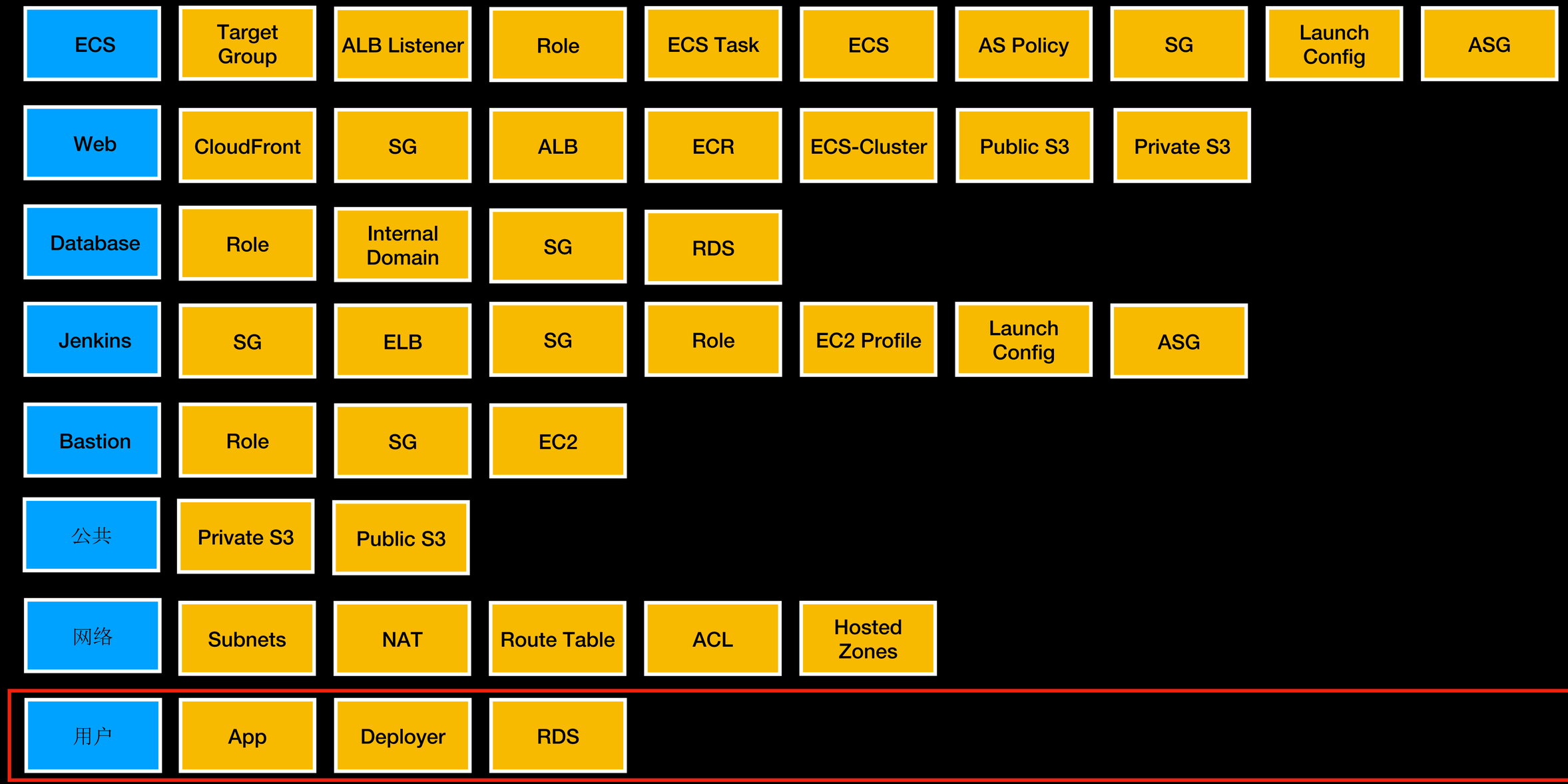
- 把 Cloudformation 当做模板
- 把 Ansible 当做模板引擎
- 利用好 role 的抽象，但不能太多
- 少用 playbook 里的变量，关键信息要用 KMS 加密
- 多用 CloudFormation 里的环境变量导出 （Export）

基础设施逻辑分层


ECS
Target Group
ALB Listener
Role
ECS Task
ECS
AS Policy
SG
Launch Config
ASG
Web
CloudFront
SG
ALB
ECR
ECS-Cluster
Public S3
Private S3
Database
Role
Internal Domain
SG
RDS
应用层
Jenkins
SG
ELB
SG
Role
EC2 Profile
Launch Config
ASG
Bastion
Role
SG
EC2
公共
Public S3
Private S3
网络
Subnets
NAT
Route Table
ACL
Hosted Zones
用户
App
Deployer
RDS
平台层

平台层

用户


ECS
Target Group
ALB Listener
Role
ECS Task
ECS
AS Policy
SG
Launch Config
ASG
Web
CloudFront
SG
ALB
ECR
ECS-Cluster
Public S3
Private S3
Database
Role
Internal Domain
SG
RDS
Jenkins
SG
ELB
SG
Role
EC2 Profile
Launch Config
ASG
Bastion
Role
SG
EC2
公共
Private S3
Public S3
网络
Subnets
NAT
Route Table
ACL
Hosted Zones
用户
App
Deployer
RDS

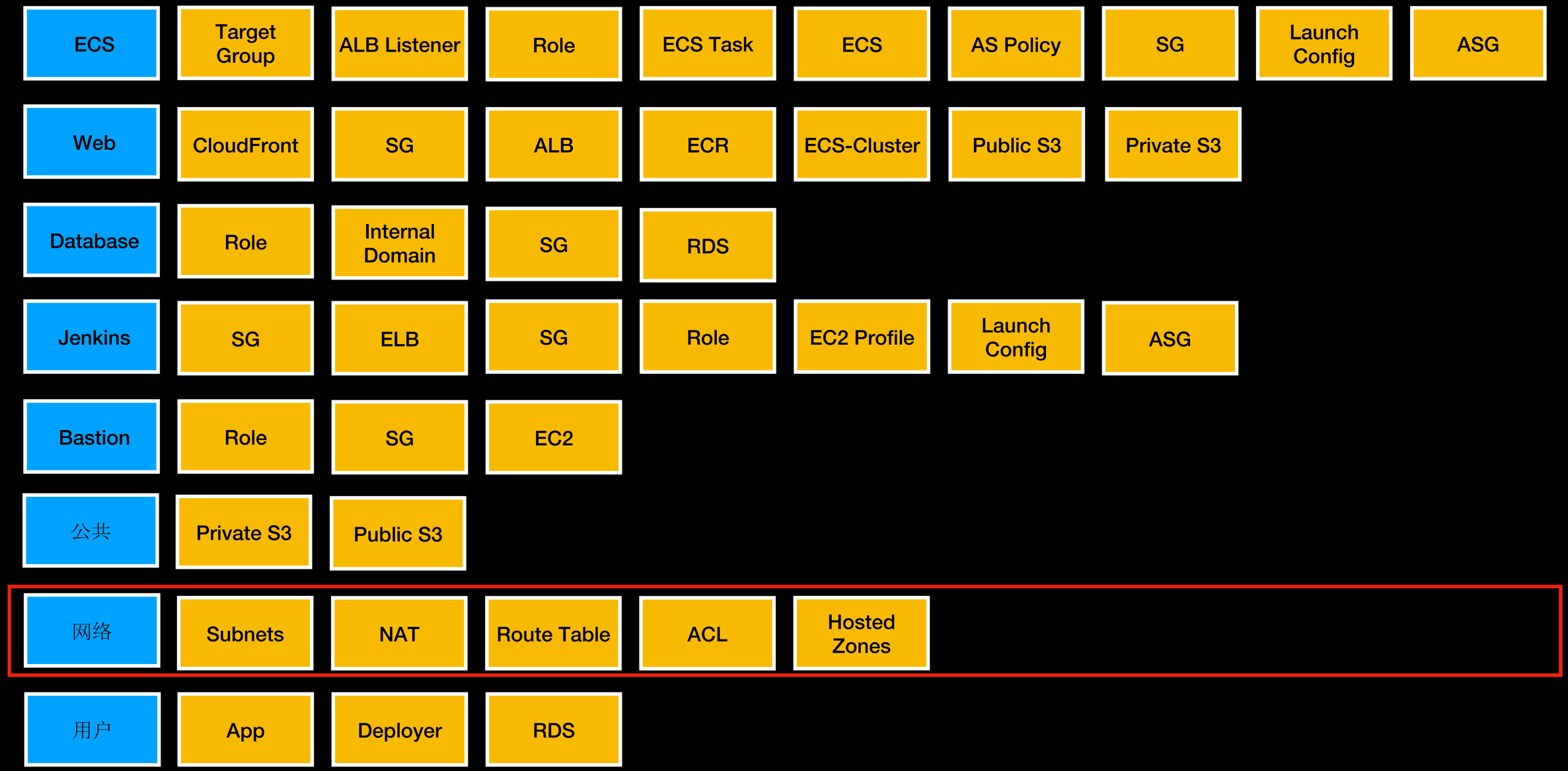
用户

- 根据场景设定专门的用户和角色
- 杜绝随意修改基础设施
- 通过基础设施即代码生成用户
- 两个可登录账户（Root-account, Administrator）

自动化创建用户/角色

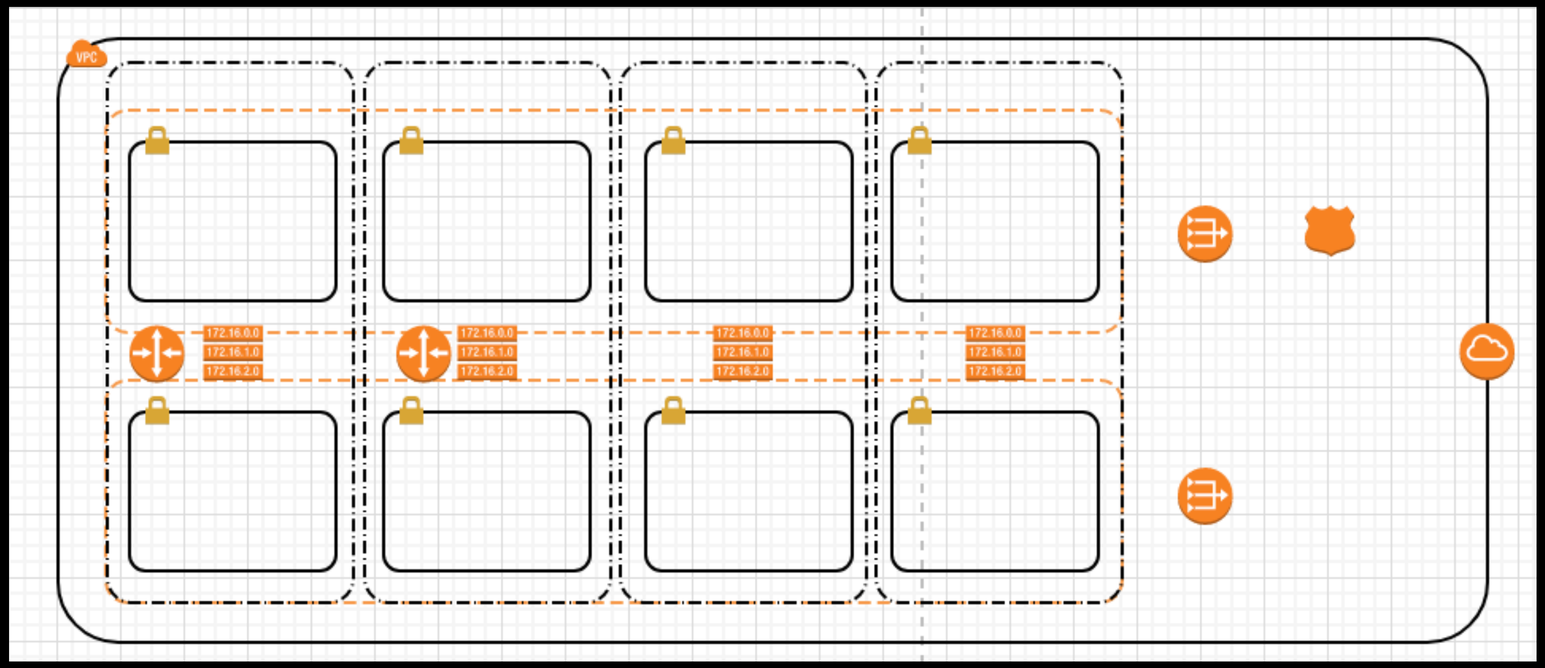
- 基础设施修改用户 *****
- 应用/数据库用户/角色 ***
- 流水线用户 ***
- 系统管理员用户 ***
- 维护用户 **
- 查询用户 *

网络

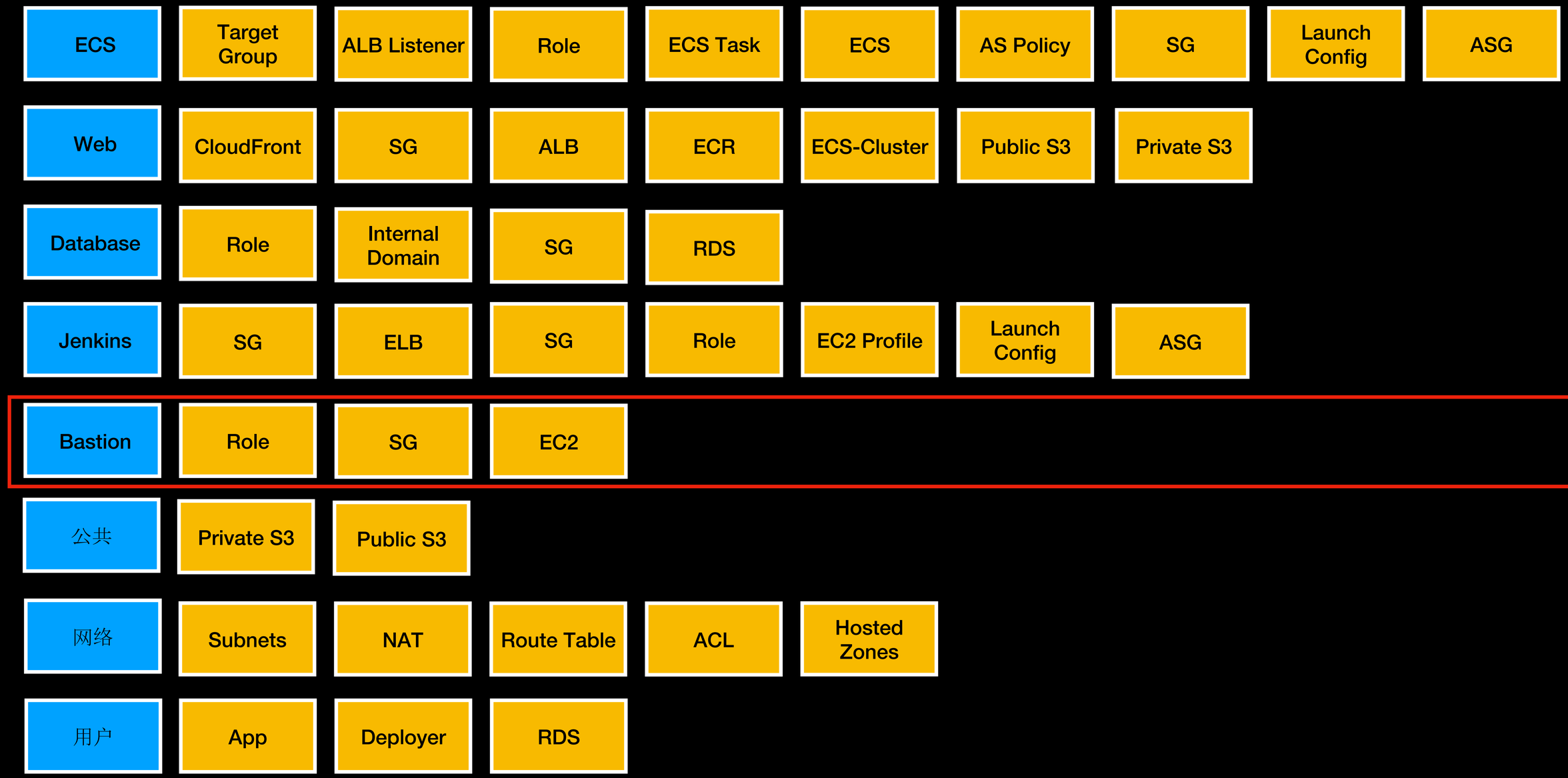

ECS
Target Group
ALB Listener
Role
ECS Task
ECS
AS Policy
SG
Launch Config
ASG
Web
CloudFront
SG
ALB
ECR
ECS-Cluster
Public S3
Private S3
Database
Role
Internal Domain
SG
RDS
Jenkins
SG
ELB
SG
Role
EC2 Profile
Launch Config
ASG
Bastion
Role
SG
EC2
公共
Private S3
Public S3
网络
Subnets
NAT
Route Table
ACL
Hosted Zones
用户
App
Deployer
RDS

网络

- 双可用区域（Multi-AZ）
- 公/私分离
- ACL



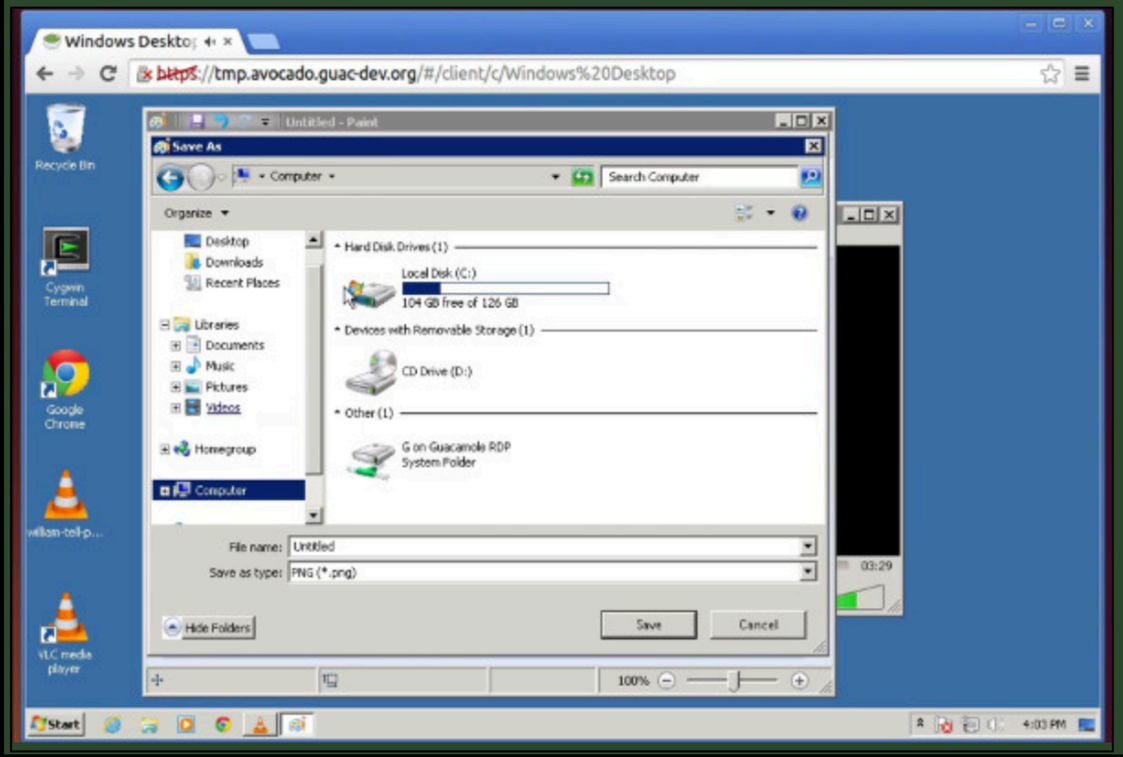
堡垒机


ECS
Target Group
ALB Listener
Role
ECS Task
ECS
AS Policy
SG
Launch Config
ASG
Web
CloudFront
SG
ALB
ECR
ECS-Cluster
Public S3
Private S3
Database
Role
Internal Domain
SG
RDS
Jenkins
SG
ELB
SG
Role
EC2 Profile
Launch Config
ASG
Bastion
Role
SG
EC2
公共
Private S3
Public S3
网络
Subnets
NAT
Route Table
ACL
Hosted Zones
用户
App
Deployer
RDS

堡垒机策略

- 有限用户使用
- 随用随删
- IP/白名单
- 只能访问 ssh2
- 独立的 ssh-key

Clientless - Apache Guacamole

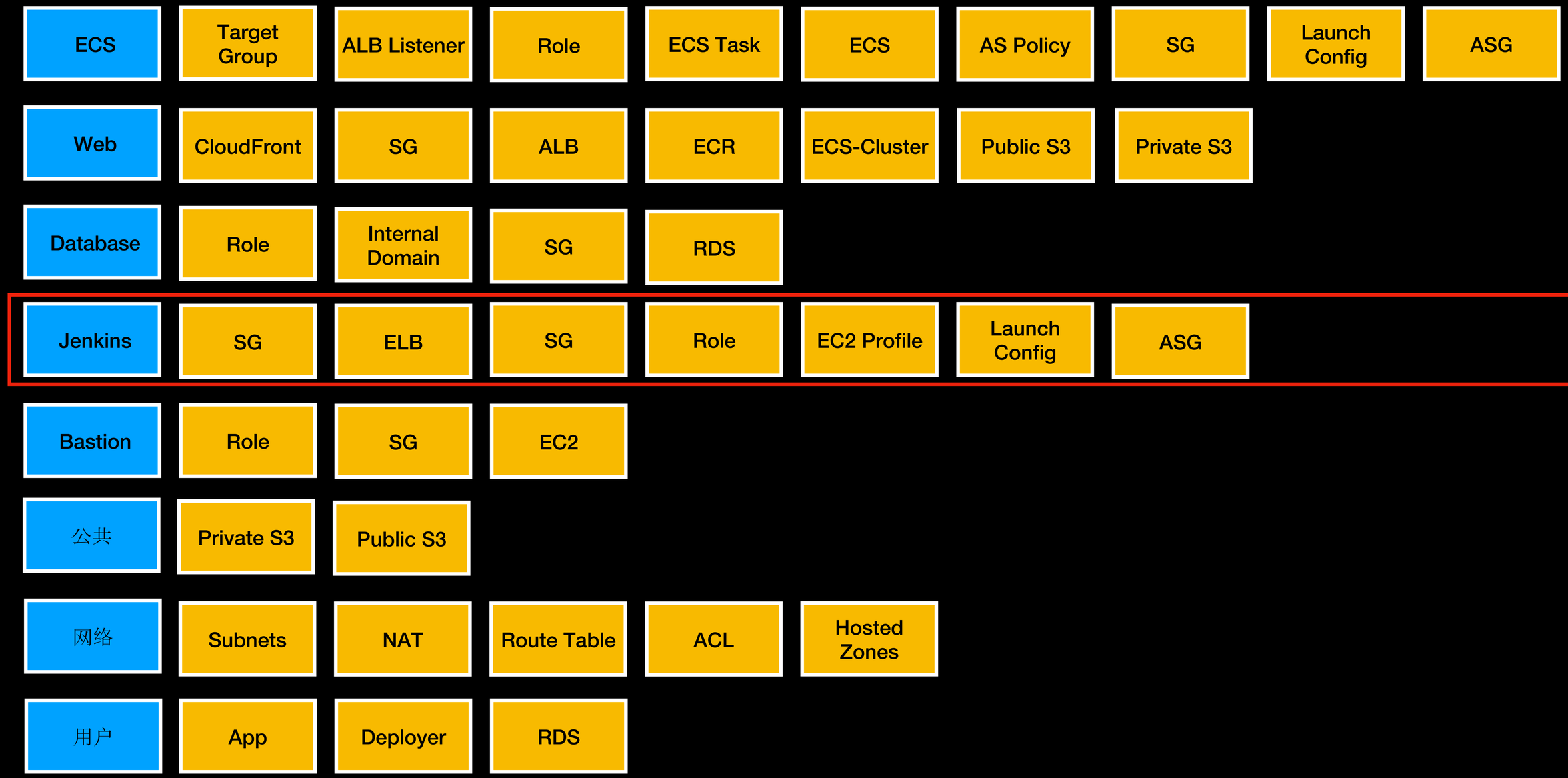


Apache Guacamole is a **clientless remote desktop gateway**. It supports standard protocols like VNC, RDP, and SSH.

We call it *clientless* because no plugins or client software are required.

Thanks to HTML5, once Guacamole is installed on a server, all you need to access your desktops is a web browser.

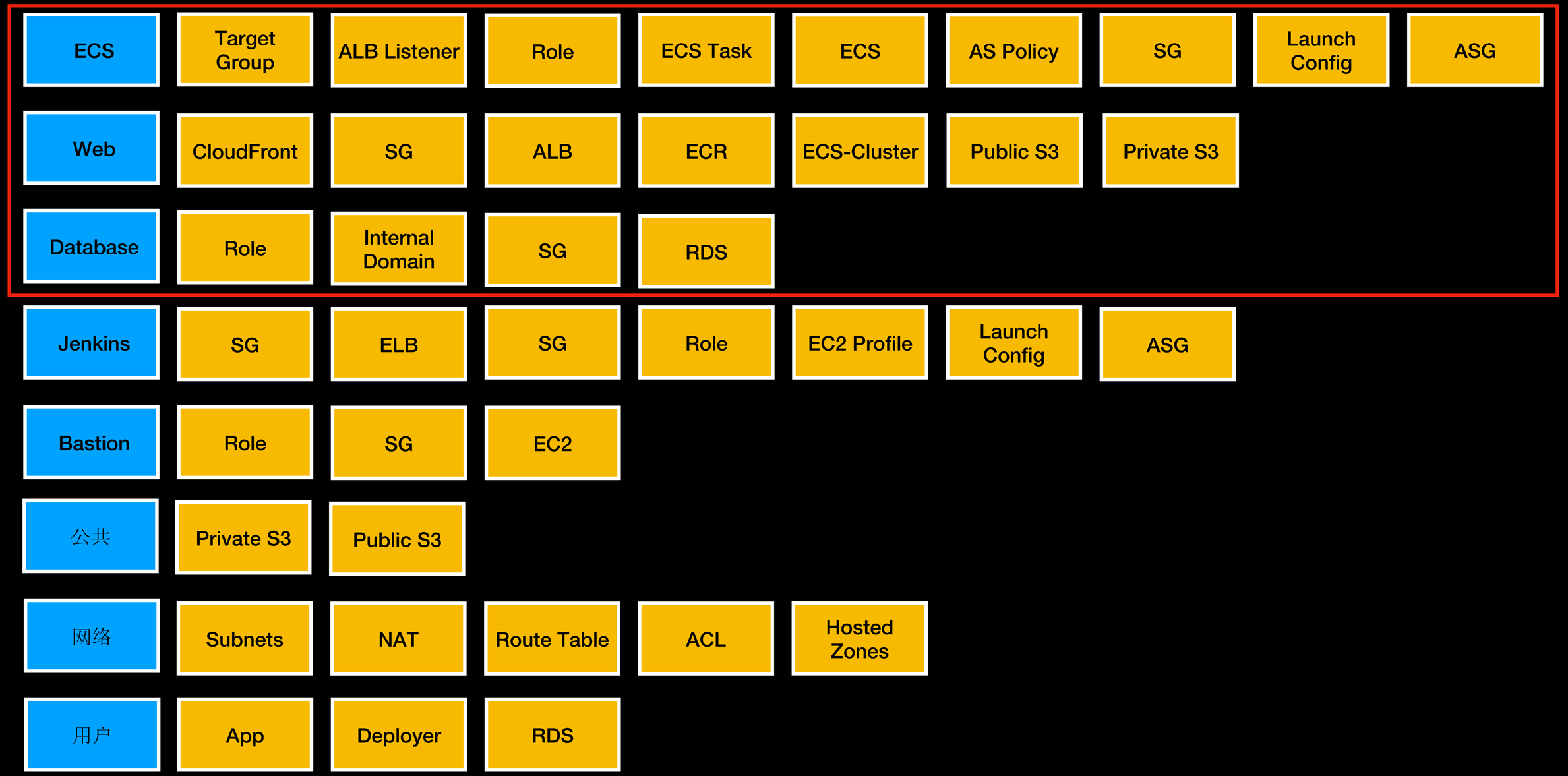
CI


ECS
Target Group
ALB Listener
Role
ECS Task
ECS
AS Policy
SG
Launch Config
ASG
Web
CloudFront
SG
ALB
ECR
ECS-Cluster
Public S3
Private S3
Database
Role
Internal Domain
SG
RDS
Jenkins
SG
ELB
SG
Role
EC2 Profile
Launch Config
ASG
Bastion
Role
SG
EC2
公共
Private S3
Public S3
网络
Subnets
NAT
Route Table
ACL
Hosted Zones
用户
App
Deployer
RDS

Jenkins

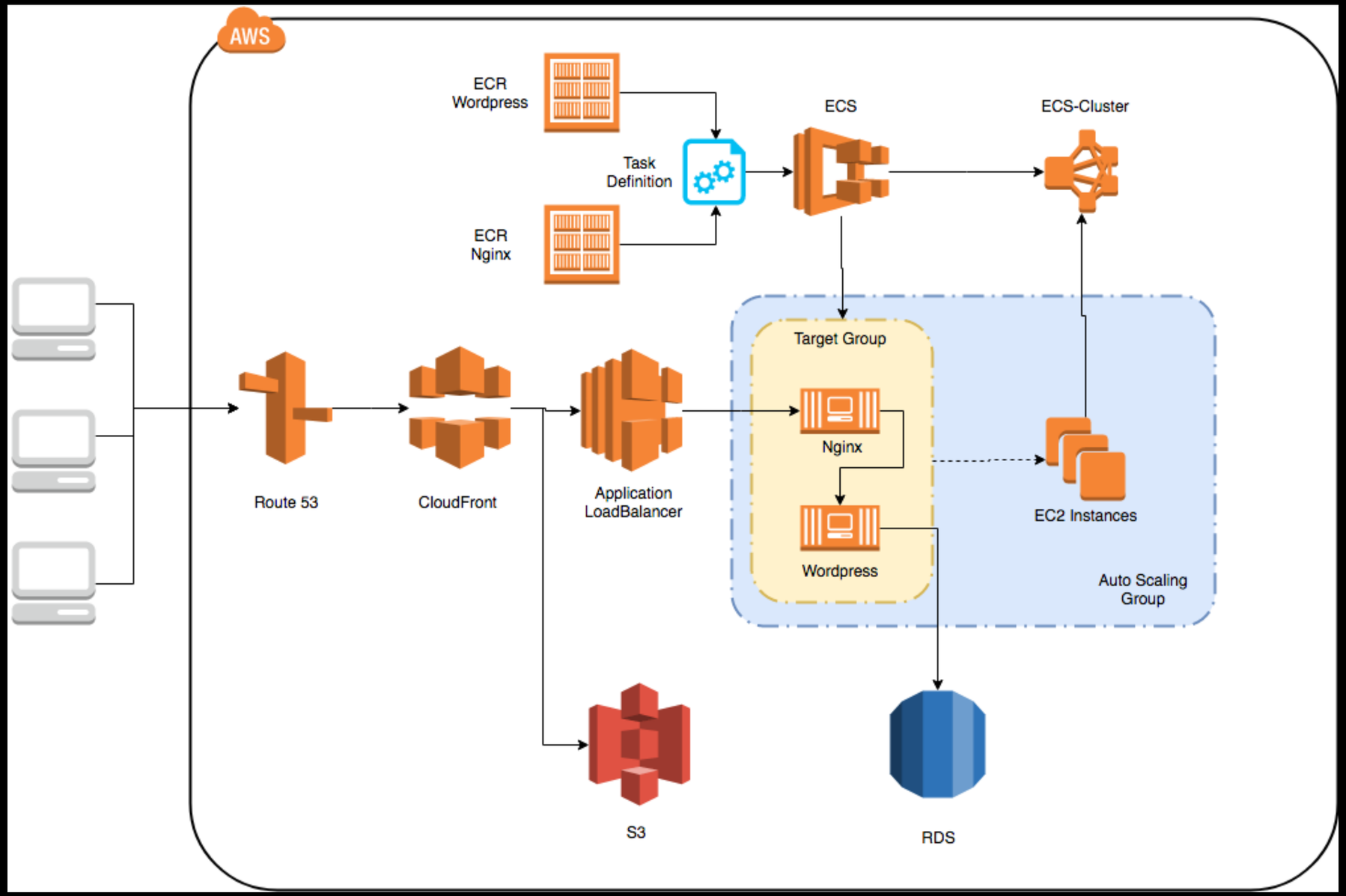
- 自动备份/自动恢复
- 有限访问
- 高可用 or 无状态

应用层

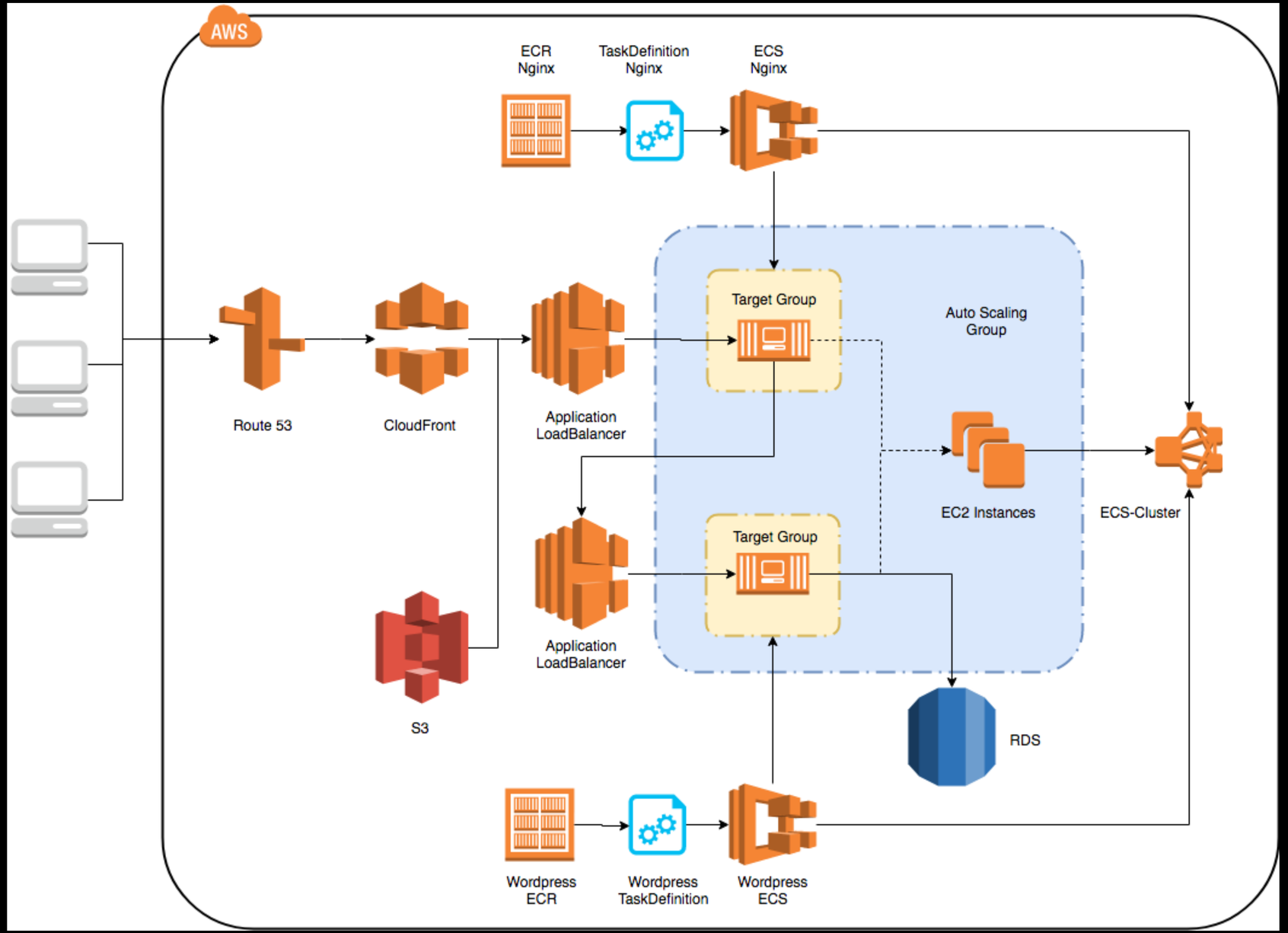

ECS
Target Group
ALB Listener
Role
ECS Task
ECS
AS Policy
SG
Launch Config
ASG
Web
CloudFront
SG
ALB
ECR
ECS-Cluster
Public S3
Private S3
Database
Role
Internal Domain
SG
RDS
Jenkins
SG
ELB
SG
Role
EC2 Profile
Launch Config
ASG
Bastion
Role
SG
EC2
公共
Private S3
Public S3
网络
Subnets
NAT
Route Table
ACL
Hosted Zones
用户
App
Deployer
RDS

Wordpress Multisite

ECS - V1


AWS
ECR
Wordpress
ECR
Nginx
Task
Definition
ECS
ECS-Cluster
Target Group
Nginx
Wordpress
Auto Scaling
Group
EC2 Instances
Route 53
CloudFront
Application
LoadBalancer
S3
RDS

ECS - V2


AWS
ECR
Nginx
TaskDefinition
Nginx
ECS
Nginx
Route 53
CloudFront
Application
LoadBalancer
Target Group
Auto Scaling
Group
EC2 Instances
ECS-Cluster
Application
LoadBalancer
Target Group
S3
RDS
Wordpress
ECR
Wordpress
TaskDefinition
Wordpress
ECS

域名

域名

- 公共：分配给公共资源，总入口手动设置。每个资源的域名自动化生成。
- 私有：为了开发人员能有一致的开发环境，注意内外网隔离。

多国域名

- 根据访问区域转向统一的 .com （hosted zone）
- <国家>.<产品>.<运营团队>.主域名.com
- 通过 Wordpress 的 multi-site 功能的 subdomain 实现单实例，多内容。

多国，多语言，单站点

泰国 .com

马来西亚 .my

印度尼西亚 .id

新加坡 .sg

.com

CDN

CDN

- 采用 AWS Cloudfront，一方面是因为便宜，另一方面是可以用基础设施即代码初始化。
- 对于 CMS 系统来说，要尽可能的发挥 CDN 的作用
- CDN 插件

备份

尽量减少备份和还原

- 所有的应用都在 Docker 镜像里
- 内容在 RDS / S3 里
- 基础设施架构在 Ansible + Cloudformation 里

自扩展实例和压力测试

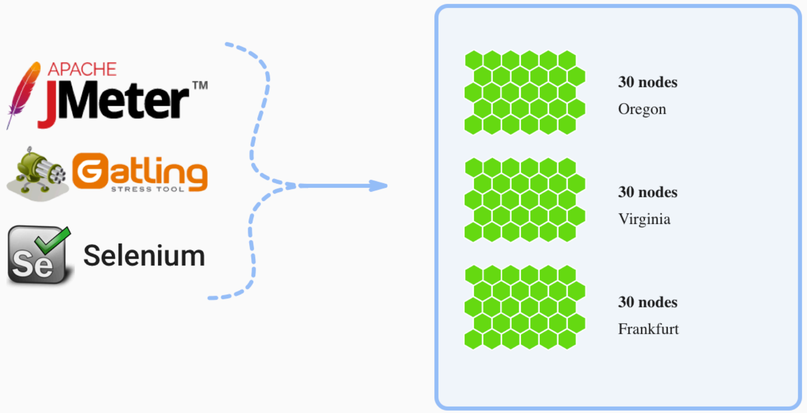
自扩展条件

- Docker Container 扩展
- EC2 Instance 扩展

flood.io

Flood Load Testing Built for Everyone

Unlike traditional load testing services, Flood is designed to maximize testing success. We do this by allowing you to run as many users and tests as you need, while we take care of the servers and collecting the results.



Design

Use the open source tools you love, including JMeter, Gatling, and Selenium.

Scale

Quickly distribute your load test plan across hundreds of servers in multiple AWS regions.

Analyse

Quick feedback loops are critical for successful load testing, so we show you results as soon as we get them, allowing you to stop a test and make changes as necessary.

自动化 E2E 测试

cypress.io

A complete end-to-end testing experience.

Install the Cypress Test Runner and write tests locally.

Set up tests

Installing Cypress is simple. No dependencies, extra downloads, or changes to your code required.

Write tests

Write tests easily and quickly, and watch them execute in real time as you build your web application.

Test Runner

Build up a suite of tests and record them in CI.

Run tests

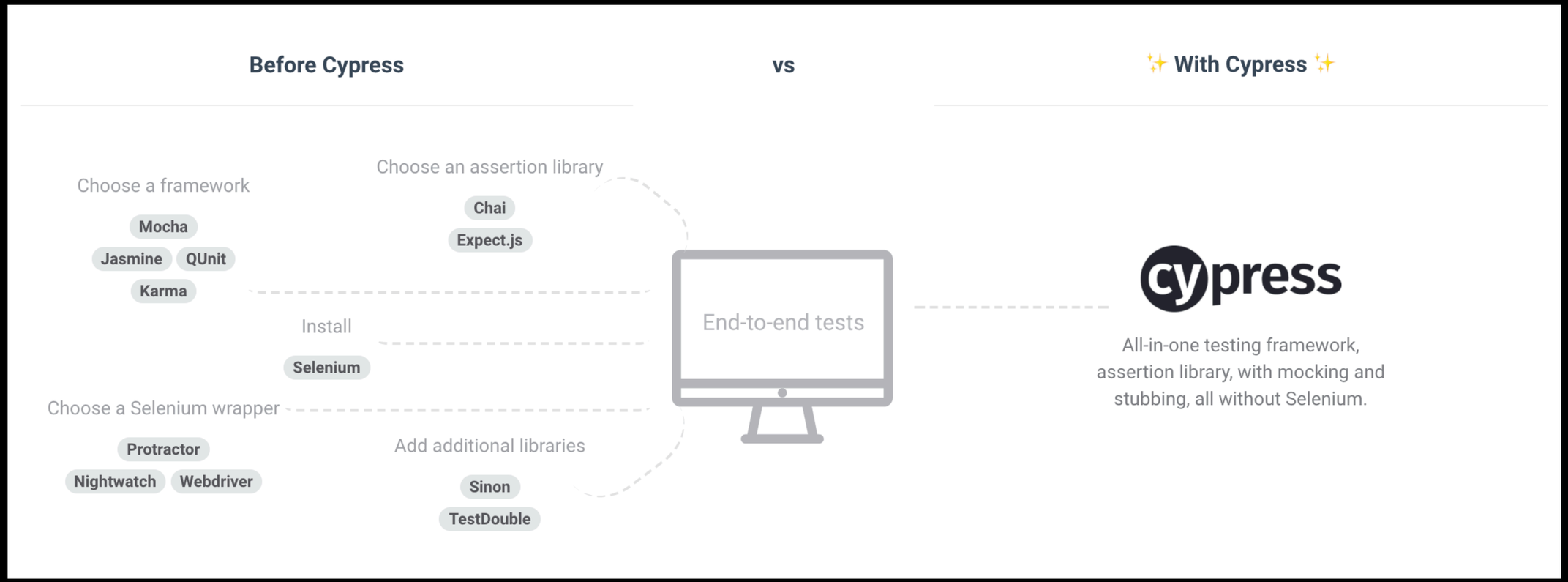
With our Dashboard Service, debugging your tests in CI is as easy as running tests locally.

Record tests

Cypress records CI test data, screenshots and video, which your team can view in your Dashboard.

Dashboard Service

cypress.io


Before Cypress
vs
✨ With Cypress ✨
Choose a framework
Mocha
Jasmine
QUnit
Karma
Choose an assertion library
Chai
Expect.js
Install
Selenium
Choose a Selenium wrapper
Protractor
Nightwatch
Webdriver
Add additional libraries
Sinon
TestDouble
End-to-end tests
cypress
All-in-one testing framework, assertion library, with mocking and stubbing, all without Selenium.

发布和部署分离

发布(release)和部署(deploy)分离

- 部署 (Deploy) 是一个技术行为，是让构建好的软件运行在环境上。
- 发布 (Release) 是一个业务行为，是让用户接触应用。

除了代码提交和发布，一切自动化

- 持续部署流水线（测试、构建镜像、构建 Task）
- 发布流水线（构建新的 ECS 重新绑定 Target Group）

两类流水线

基础设施流水线

- 每天都可以在不同的 VPC 上进行架构的重建和测试
- 不断通过自动化架构创建测试架构的稳定性和灵活性

每天进行一次架构迁移

- 今天的生产环境是昨天的开发环境
- 今天的开发环境是明天的生产环境
- 每日可用架构

应用程序流水线

- 自动持续的向环境部署（构建 Image, Task ）
- 通过新建 ECS 实现发布

总结

- 架构迁移要为组织结构迁移服务
- 把自动化和基础设施即代码当做制度使用（康威定理和逆定理）
- 安全的架构和架构的安全
- 基础设施逻辑分层
- 基础设施即代码本质上是一套类库，从面向对象的原则考虑。
- 每日可用架构。

日志

监控和告警

Thanks