Infrastructure as Code In Aliyun

田涛涛(祝犁)

目　录

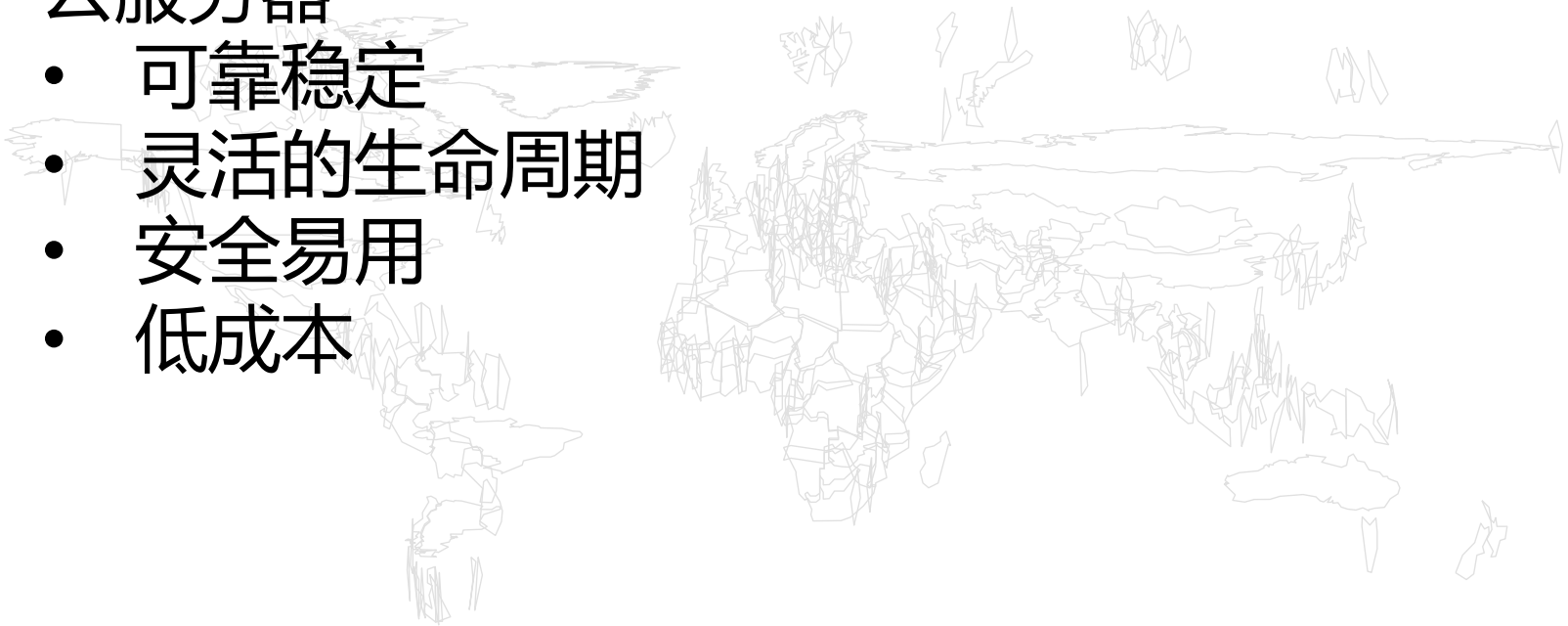
- About me
- 云资源
- Cloud Native Application Infrastructure
- Infrastructure as Code
- 资源编排(Resource Orchestration Service)
- ROS with Cloud-init
- ROS with Ansible
- WaitCondition

About me

- 田涛涛(祝犁)
- 阿里云弹性计算开发
  - 云服务器管控(ECS)
  - 资源编排架构和开发(ROS)
- IBM DB2管控和开发

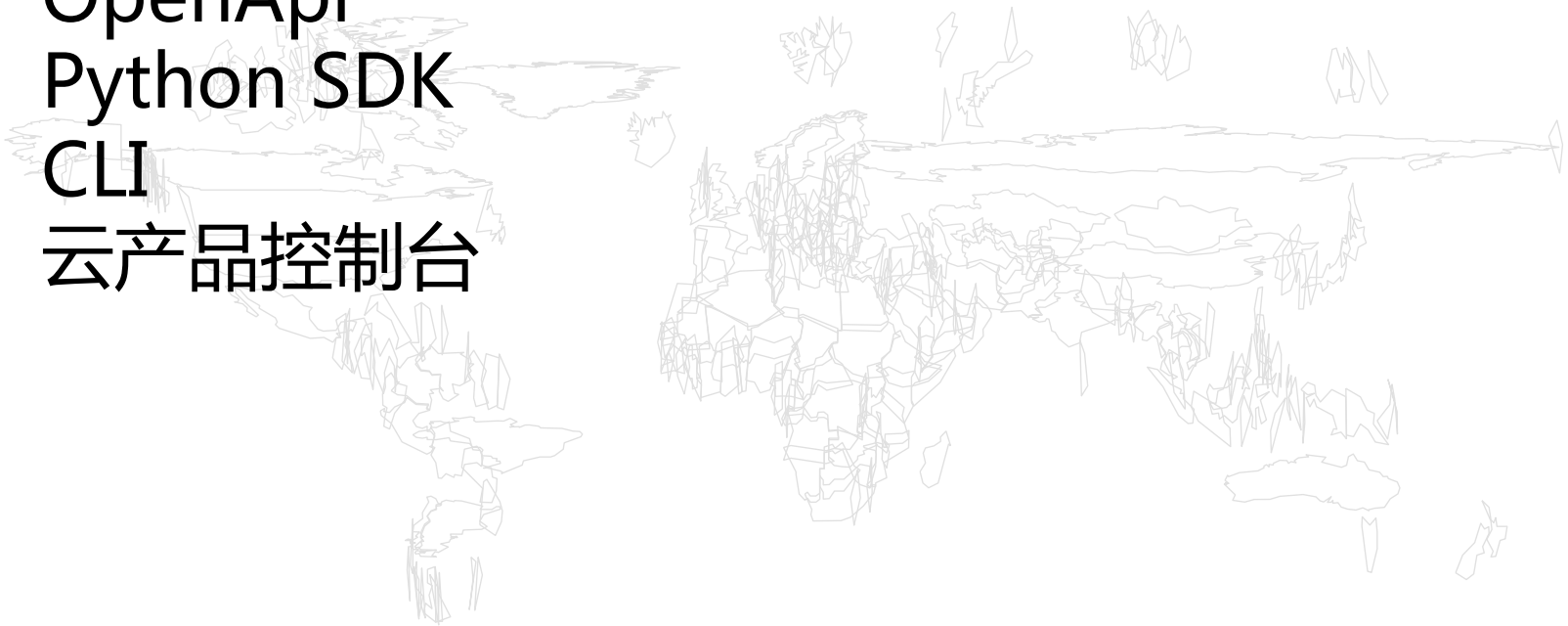
云资源基础实施

- 云服务器
  - 可靠稳定
  - 灵活的生命周期
  - 安全易用
  - 低成本



云服务器的管控渠道

- OpenApi
- Python SDK
- CLI
- 云产品控制台

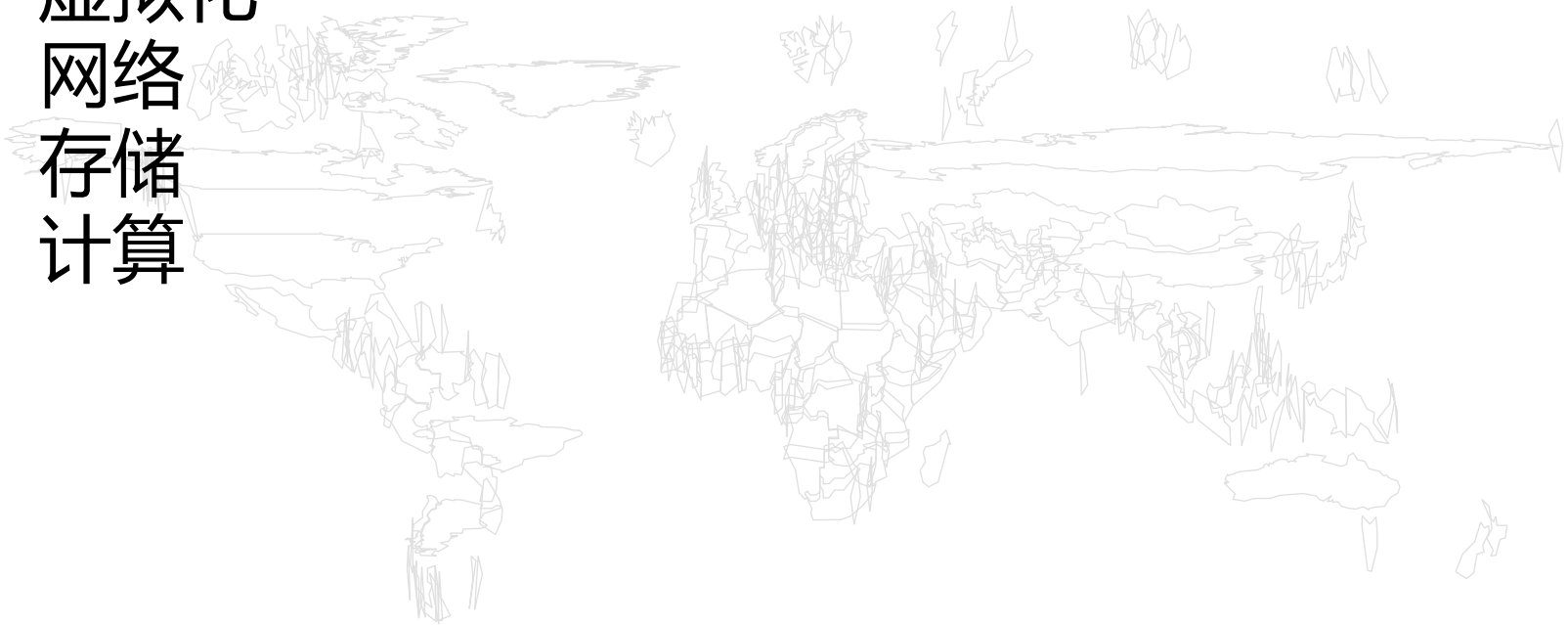


不仅仅是云服务器

应用为王的时代

云资源Infrastructure主要包括

- 虚拟化
- 网络
- 存储
- 计算



Cloud Native Application

- DevOps
- 微服务架构
- 持续构建和交付
- 容器
- Infrastructure as Code

Cloud Application Infrastructure-计算

- 云服务器(ECS)
- 容器服务(Docker)
- 高性能计算(HPC)
- E-MapReduce(Hadoop/Spark/Hbase)
- …

Cloud Application Infrastructure-存储

- 数据库(RDS/MongoDB/Redis/...)
- CDN
- OSS
- NAS
- ...

	自动化服务项	云数据库 RDS	ECS自建数据库
可靠性	数据高可靠性保障	✓	✓
	数据自动备份	✓	
	支持两年内数据恢复	✓	
安全性	DDOS防护	✓	✓
	IP白名单	✓	
	拦截SQL注入、暴力破解	✓	
可用性	主备架构	✓	
	同城容灾	✓	
	异地容灾	✓	
可扩展性	弹性扩容	✓	✓
	读写分离	✓	
成本	数据库运维成本	低	高
	基础运维（机器、网络等）成本	零成本	零成本

Cloud Application Infrastructure-安全

- 访问控制
- 安全组
- 服务器安全
- DDoS
- Web防火墙
- 数据加密
- …

Cloud Application Infrastructure-网络

- VPC(Virtual Private Cloud)
- EIP
- NAT网关
- 高速通道
- …

	专有网络	经典网络
安全隔离	使用隧道技术达到与传统 VLAN 相同隔离效果 广播域隔离在实例网卡级别	用户共享阿里云网络
软件定义网络	按需配置网络设置、软件定义网络 管理操作实时生效	网络由阿里云统一管理，用户不能独立规划
访问控制	灵活的访问控制规则，满足政务、金融用户的安全隔离规范 可通过 RAM 实现对网络的权限管理	支持实例维度的访问控制与权限管理
丰富的网络连接方式	支持使用高速通道实现跨地域/跨用户的内网互通和物理专线接入 支持使用NAT网关进行DNAT/SNAT转发；	不同地域间内网隔离 ，不支持高速通道 不支持NAT网关，无官方SNAT/DNAT功能
功能强大的公网网关	通过NAT网关支持灵活的DNAT/SNAT转发，支持多IP共享带宽	不支持NAT网关，无官方SNAT/DNAT功能，不支持多个IP共享带宽

Cloud Application Infrastructure-中间件和应用

- 云监控Cloud Monitor)
- 消息服务(Message Service)
- 权限控制(RAM)
- 日志服务(Log Service)
- 弹性伸缩(Auto Scaling)
- 视频服务
- …

Cloud Application Infrastructure-应用

云服务器Cluster
避免单点
　　负载均衡(SLB)
同一个Region
　　多可用区(Multi-Zone)

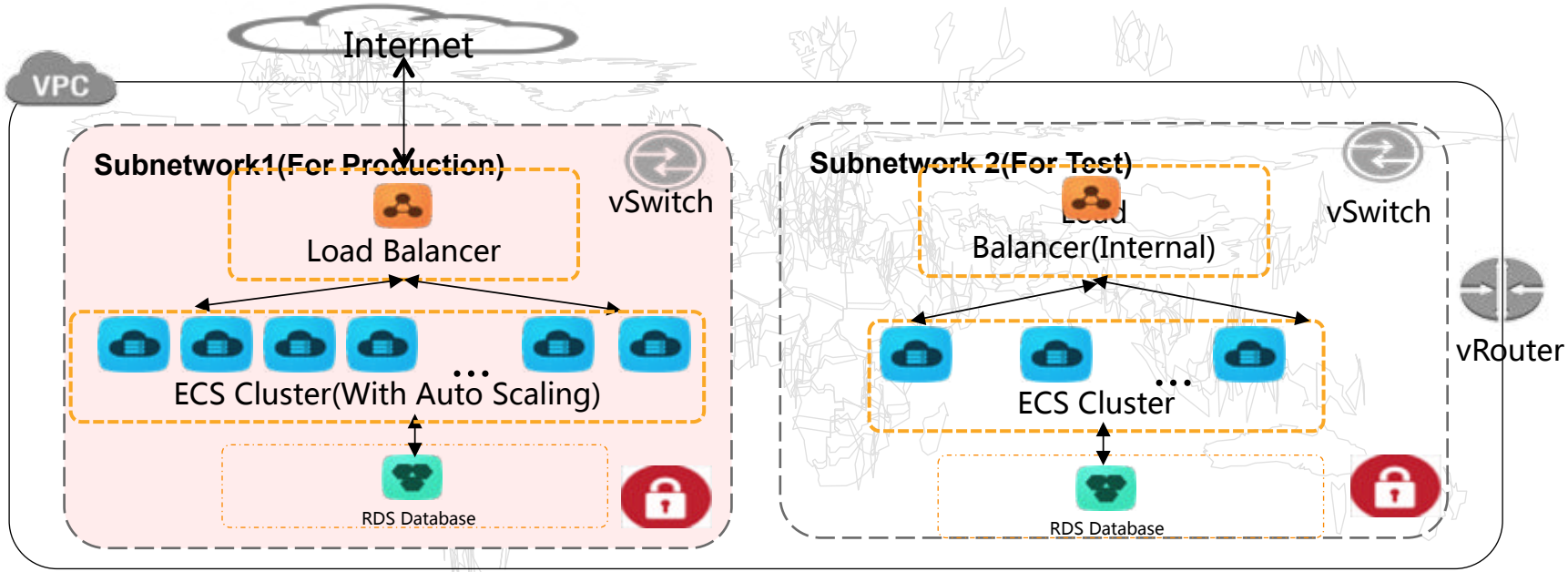
Cloud Application Infrastructure

- 云计算服务商提供众多的基础资源
- 弹性扩容、缩容应用
- Automatic

从开发到交付

- 生产环境
- 预发布环境
- 集成环境
- 测试开发环境

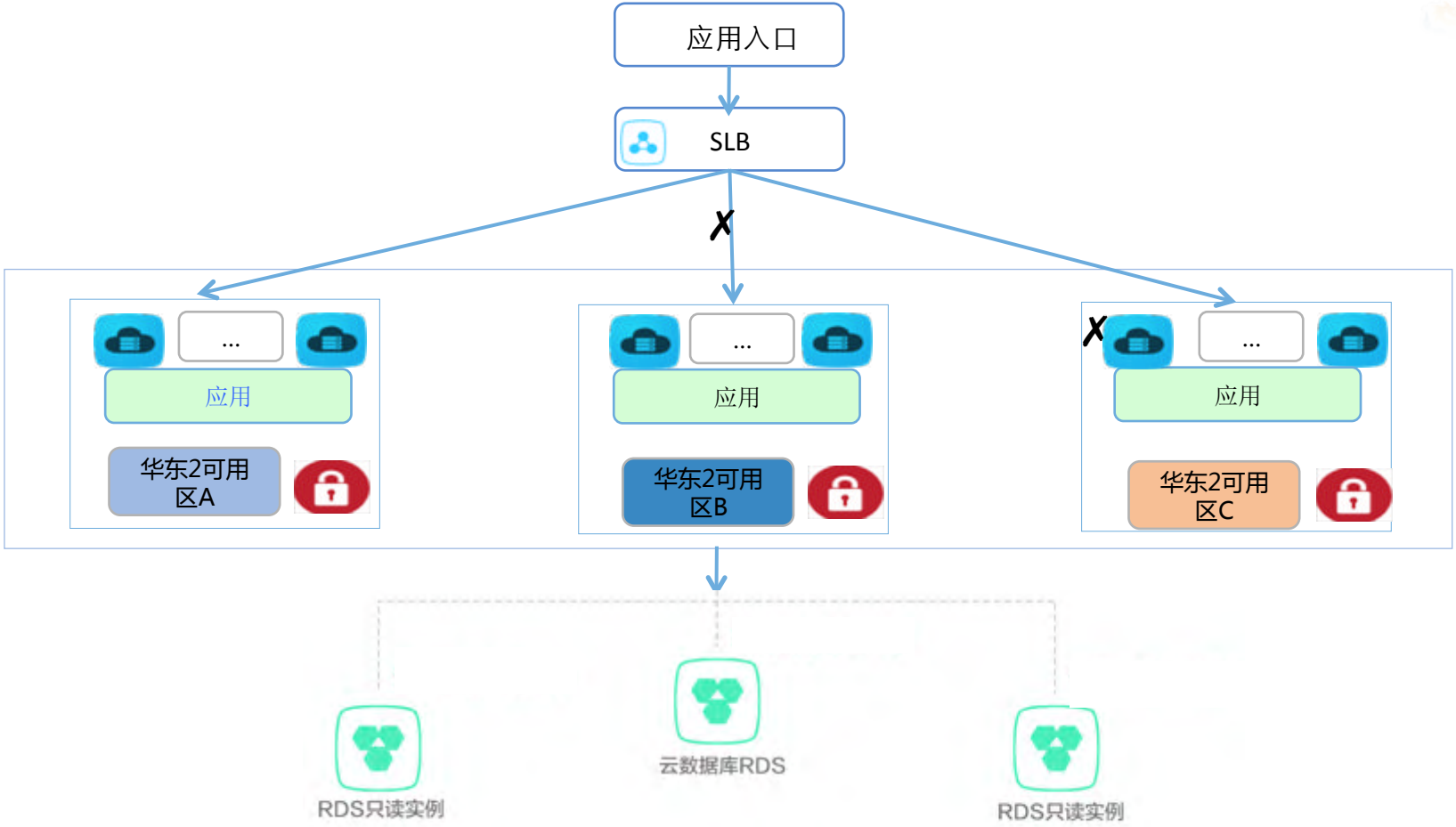
经典的网络应用架构-拓扑



Failover

- Everything fails
- All the time

HA Cloud Infrastructure



Infrastructure as Code

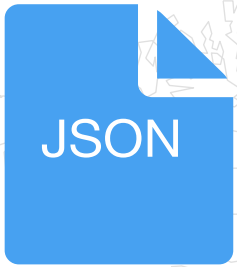
- Code based infrastructure management
- 将经验变成可复用、可交付的云计算架构
- 快速可靠部署
- 隐藏复杂的API细节
- Devops的基础

资源编排(Resource Orchestration Service)

- 通过模板描述多个云计算资源的依赖关系、配置等，并自动完成所有资源的创建和配置.
- 快速的交付一个资源组
- 交付应用
- AUTOMATE IT. SHARE IT.

Infrastructure as Code

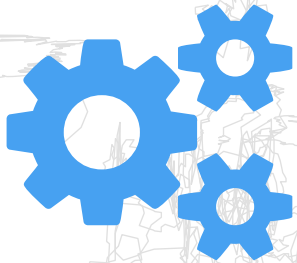
模版(Template)



JSON格式文本文件

申明模板参数
定义资源需求
说明资源依赖关系和配置细节

资源编排(ROS)



编排引擎

根据模板创建资源栈
更新维护资源栈
重试和出错处理

资源栈/应用(Stack)

阿里云资源集合

以资源栈为单位管理一组资源
监控与报警
升级、扩容

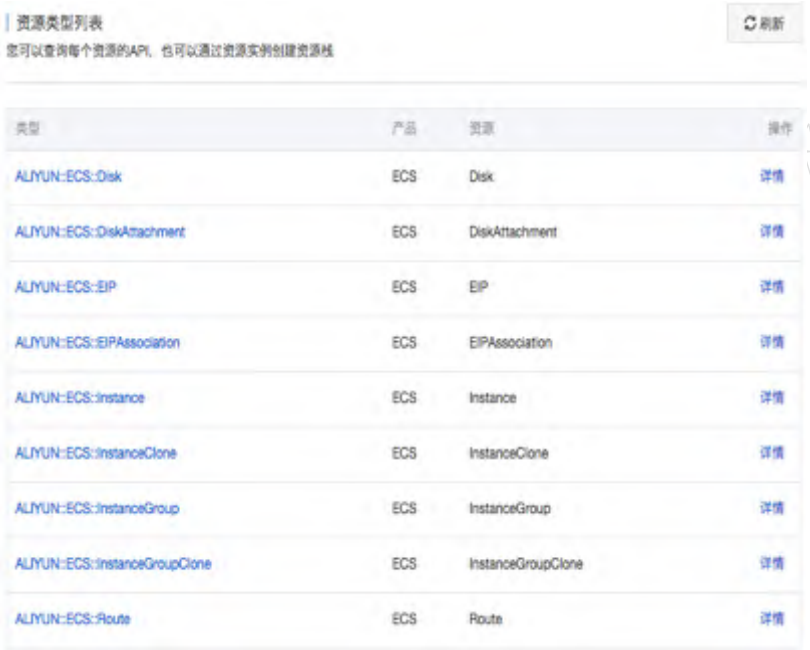
迭代

ROS Template Structure

```
{
        "ROSTemplateFormatVersion": "2015-09-01",

        "Description": "A text descriptions for template usage",

        "Parameters": {
                // A set of parameters used to customize the template per deployment
        },

        "Resources": {
                // A set of Aliyun resources and relationships between them
        },

        "Outputs": {
                // A set of values to the stack creator
        }
}
```

资源抽象和定义

- ALIYUN::ECS::Instance
- Properties
  - ImageId
  - SecurityGroupId
  - InstanceType
  - …
- Attributes
  - InstanceId
  - PublicIp
  - …



通过Python SDK创建一台云服务器(1)

```
template = '''

{
    "ROSTemplateFormatVersion": "2015-09-01",
    "Resources": {
        "My_ECS_Instance": {
            "Type": "ALIYUN::ECS::Instance",
            "Description": "Create a ECS instance for demo.",
            "Properties": {
                "ImageId": "m-25qoptbjn",
                "InstanceType": "ecs.s2.large",
                "InternetChargeType": "PayByTraffic",
                "IoOptimized": "none",
                "SystemDisk_Category": "cloud",
                "SecurityGroupId": {
                    "Fn::GetAtt": [
                        "mySecurityGroup",
                        "SecurityGroupId"
                    ]
                }
            }
        },

        "mySecurityGroup": {
            "Type": "ALIYUN::ECS::SecurityGroup",
            "Properties": {
                "SecurityGroupName": "mySecurityGroup"
            }
        }
    }
}

'''
```

通过Python SDK创建一台云服务器(2)

```python
from aliyunsdkcore.client import AcsClient
from aliyunsdkros.request.v20150901 import DescribeRegionsRequest, CreateStacksRequest

import json

client = AcsClient('Your access id key', 'Your access secret', 'cn-beijing')

req = DescribeRegionsRequest.DescribeRegionsRequest()
status, headers, body = client.get_response(req)

if status == 200:
    regions = json.loads(body)
    print(regions)
else:
    print('Unexpected errors: status=%d, error=%s' % (status, body))

create_stack_body = '''
{
    "Name": "%s",
    "TimeoutMins": %d,
    "Template": %s
}
''' % ('my_demo_stack', 60, template)

print(create_stack_body)

req = CreateStacksRequest. CreateStacksRequest()
req.set_headers({'x-acs-region-id': 'cn-beijing'})
req.set_content(create_stack_body);

status, headers, body = client.get_response(req)
if status == 201:
    result = json.loads(body)
    print(result)
else:
    print('Unexpected errors: status=%d, error=%s' % (status, body))
```

基于资源编排交付应用

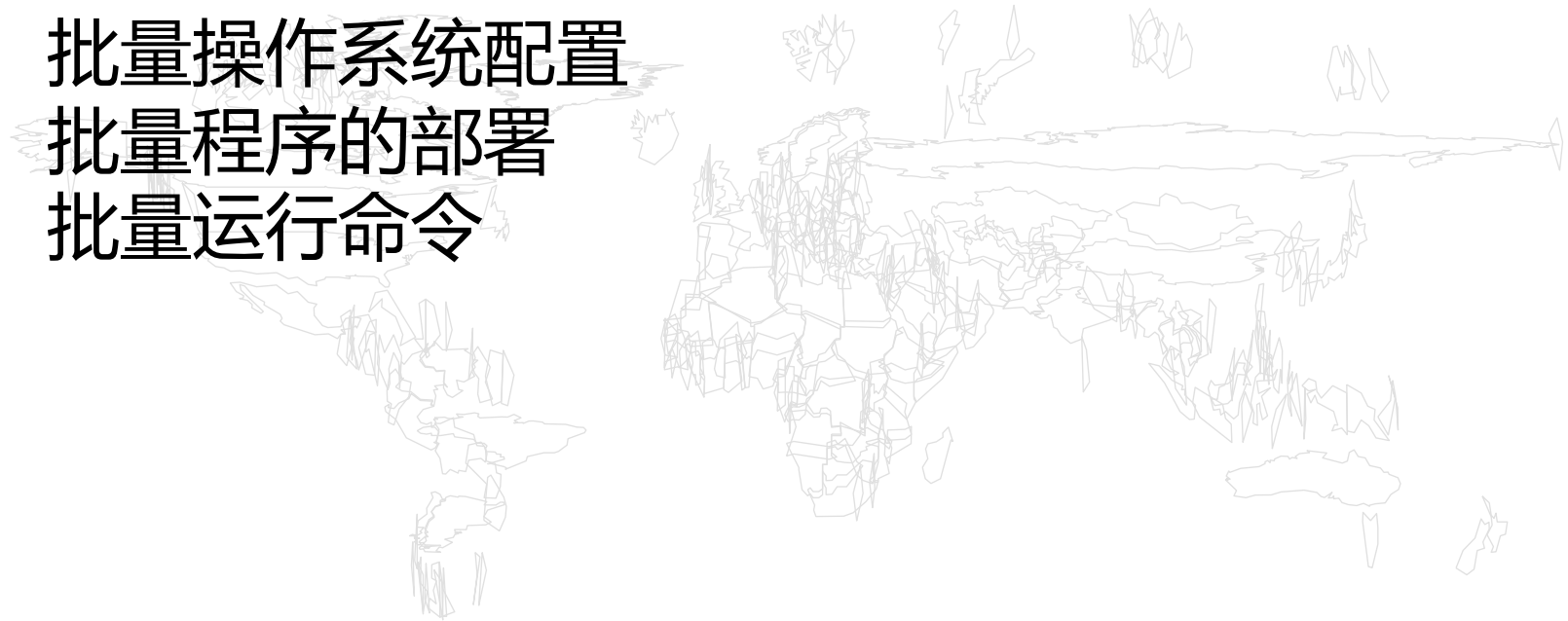
- Meta-Data
  - 主要包括虚拟机自身的一些常用属性，如 hostname、网络配置信息、资源 InstanceId 等，其主要的形式为键值对。
- Cloud-Init
  - Cloud-Init 是一个在云主机启动时操作和定制云主机环境的包。
- User-Data
  - User-Data 是实现云主机个性化定制的基础，用户通过 User-Data 设置一些定制化数据，如启动某项服务，下载一些包并设置这些包的安装脚本等，Cloud-Init 在云主机启动时加载并执行这些数据，从而完成对云主机的个性化定制。

通过UserData部署WordPress

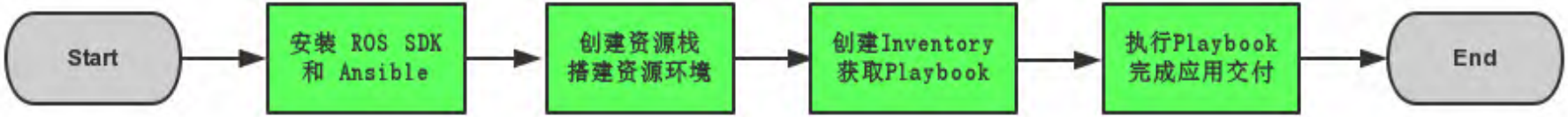
```
"SystemDiskCategory" : "cloud_ssd",
"UserData": {
  "Fn::Replace": [
    { "print": "echo"},
    {
      "Fn::Join": ["",
        [
          "#!/bin/sh",
          "\n", "DatabaseUser=",{ "Ref": "DBUser" }, "\n",
          "DatabasePwd=",{ "Ref": "DBPassword" },"\n",
          "DatabaseName=",{"Ref": "DBName" }, "\n",
          "WebRootPath='/var/www/html'\n",
          "ApacheIndex='Options Indexes FollowSymLinks'\n",
          "ApacheIndexReplace='Options -Indexes FollowSymLinks'\n",
          "yum install -y curl httpd mysql-server php php-common php-mysql\n",
          "yum install -y php-gd php-imap php-ldap php-odbc php-pear php-xml php-xmlrpc\n",
          "chkconfig httpd on\n",
          "chkconfig mysqld on \n",
          "wget http://wordpress.org/latest.tar.gz\n",
          "tar -xzvf latest.tar.gz\n",
          "sed -i \"s/database_name_here/$DatabaseName/\" wordpress/wp-config-sample.php\n",
          "sed -i \"s/username_here/$DatabaseUser/\" wordpress/wp-config-sample.php\n",
          "sed -i \"s/password_here/${DatabasePwd:-$DatabasePwdDef}/\" wordpress/
          wp-config-sample.php\n",
          "mv wordpress/wp-config-sample.php wordpress/wp-config.php\n",
          "cp -a wordpress/* $WebRootPath\n",
          "rm -rf wordpress*\n",
          "service httpd stop\n",
          "usermod -d $WebRootPath apache &>/dev/null\n",
          "chown apache:apache -R $WebRootPath\n",
          "sed -i \"s/$ApacheIndex/$ApacheIndexReplace/\" /etc/httpd/conf/httpd.conf\n",
          "service httpd start\n",
          "service mysqld start\n",
          "chmod 400 /etc/my.cnf\n",
          "mysql -u root << EOF \n",
          "CREATE DATABASE $DatabaseName default charset utf8 COLLATE utf8_general_ci;\n",
          "CREATE USER $DatabaseUser IDENTIFIED by '$DababasePwd';\n",
          "grant all on $DatabaseName.* to $DatabaseUser@localhost identified by
          '$DatabasePwd';\n",
          "EOF\n"
        ]
      ]
    }
```

Ansible

- SOLVE IT. AUTOMATE IT. SHARE IT.
- 批量操作系统配置
- 批量程序的部署
- 批量运行命令



基于ROS 和Ansible 快速交付应用



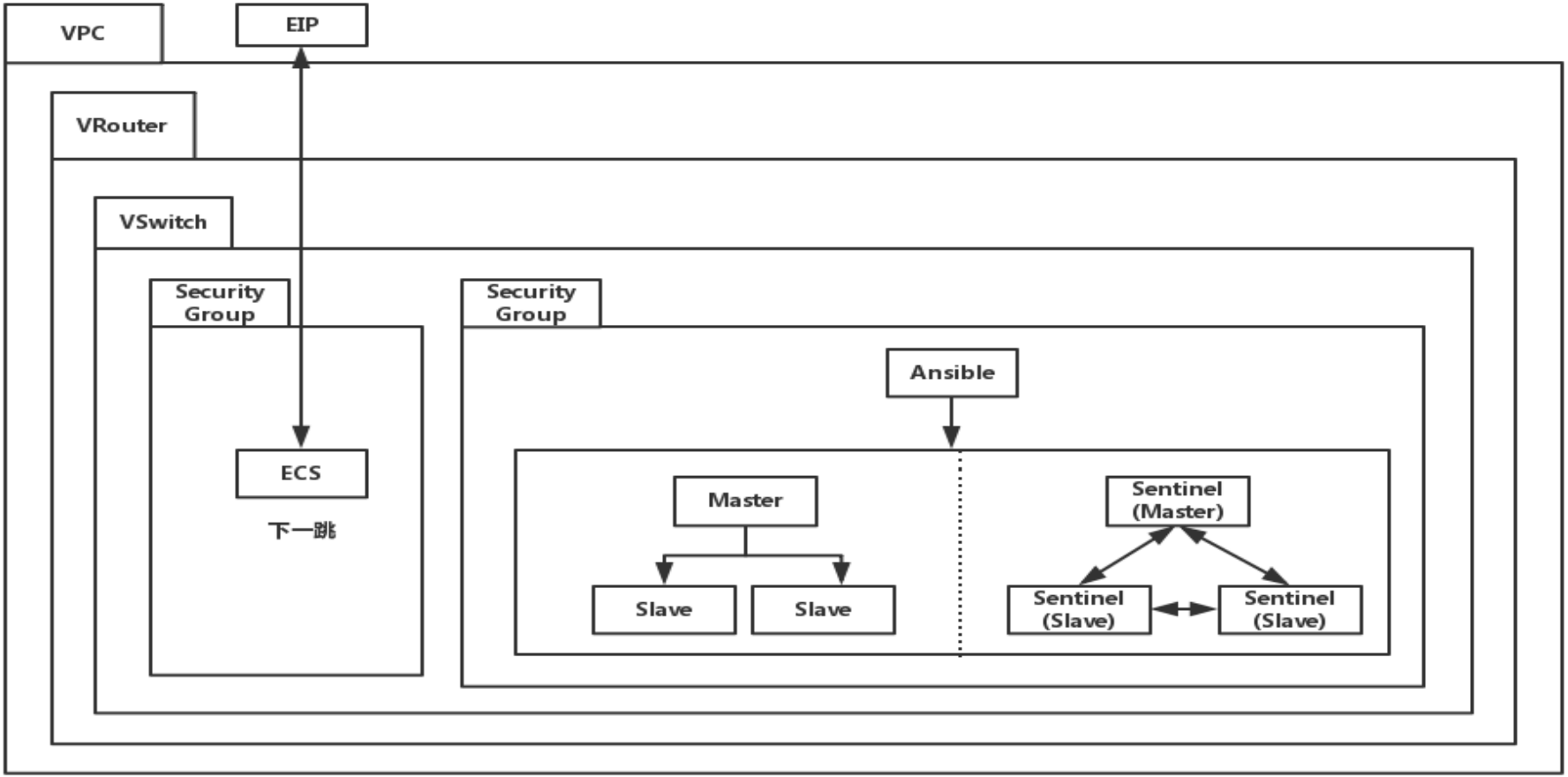
基于ROS 和Ansible 部署Redis 集群-Inventory

```
# define func to create inventory
def edit_hosts(remote_ips, remote_ports, remote_users, remote_pass, is_sentinel):
if remote_ips is None or 0>=len(remote_ips):
    print('Error! Remote ips is empty!')
    return
else:
    if remote_ports is None or len(remote_ips)!=len(remote_ports):
        print('Error! Remote ports is empty!')

    if remote_users is None or len(remote_ips)!=len(remote_users):
        print('Error! Remote users is empty!')

    if remote_pass is None or len(remote_ips)!=len(remote_pass):
        print('Error! Remote pass is empty!')

    host_str = ' ansible_ssh_port=%s ansible_ssh_user=%s ansible_ssh_pass=%s\n'
    file = open(hosts_dir + '/' + hosts_file, 'wb')
    file.write( '[%s]\n' % host_master )
    file.write( ('%s'+host_str) % (remote_ips[0], remote_ports[0], remote_users[0], remote_pass[0]) )
    if len(remote_ips)>1:
        file.write( '\n[%s]\n' % host_slave )
        for index in range(1,len(remote_ips)):
            file.write( ('%s'+host_str) % (remote_ips[index], remote_ports[index], remote_users[index], remot
e_pass[index]) )
    if is_sentinel:
        file.write( '\n[%s]\n' % host_sentinel )
        for index2 in range(0,len(remote_ips)):
            file.write( ('%s redis_sentinel=True'+host_str) % (remote_ips[index2], remote_ports[index2], remo
te_users[index2], remote_pass[index2]) )
    file.close()
print 'End'
```

基于ROS 和Ansible 部署Redis 集群-Execute

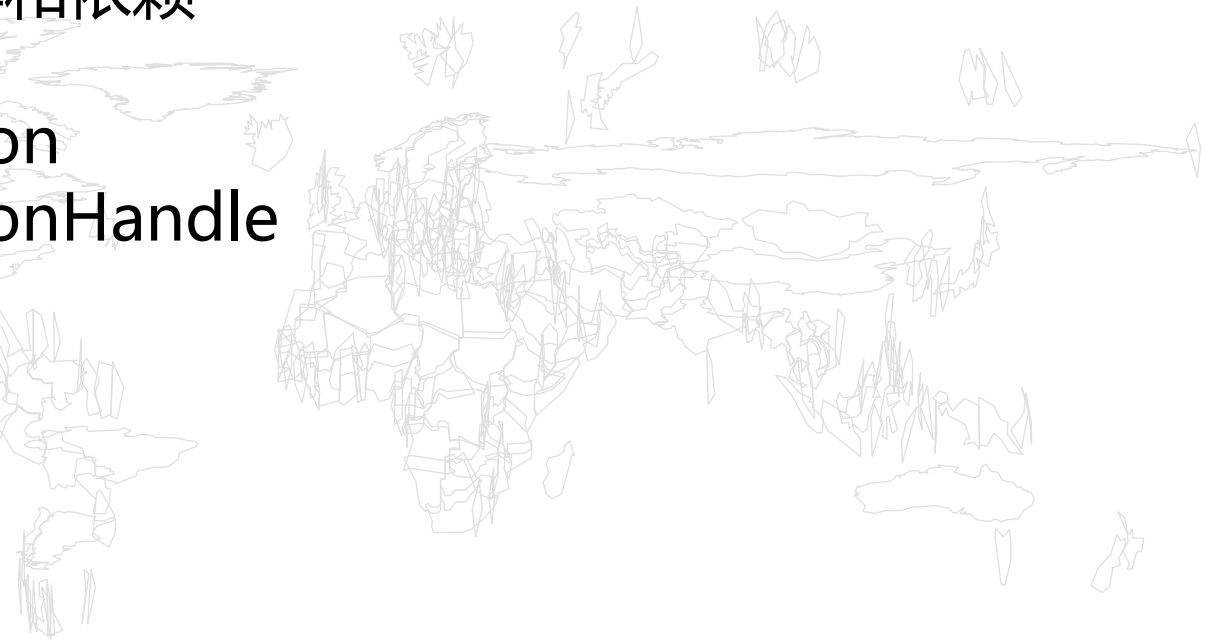
```
# define func to create playbook init config
def create_pb_init(host_ip, has_slaves, is_sentinel):
# config master
file_pb = open(pb_file_dir + '/' + pb_file_name, 'wb')
file_pb.write(redis_playbook_template.create_playbook(template='master', host_master_name=host_master, host_s
lave_name=host_slave, host_sentinel_name=host_sentinel, playbook_name=pb_name, master_ip=host_ip, master_port
='6379', master_name='master01'))
file_pb.close()

# config slaves
if has_slaves:
    file_pb = open(pb_file_dir + '/' + pb_file_name, 'ab')
    file_pb.write(redis_playbook_template.create_playbook(template='slaves', host_master_name=host_master, ho
st_slave_name=host_slave, host_sentinel_name=host_sentinel, playbook_name=pb_name, master_ip=host_ip, master_
port='6379', master_name='master01'))
    file_pb.close()

# config sentinel
if is_sentinel:
    file_pb = open(pb_file_dir + '/' + pb_file_name, 'ab')
    file_pb.write(redis_playbook_template.create_playbook(template='sentinel', host_master_name=host_master,
 host_slave_name=host_slave, host_sentinel_name=host_sentinel, playbook_name=pb_name, master_ip=host_ip, mast
er_port='6379', master_name='master01'))
    file_pb.close()
```

基于ROS 和Ansible 部署Redis 集群-Infrastructure

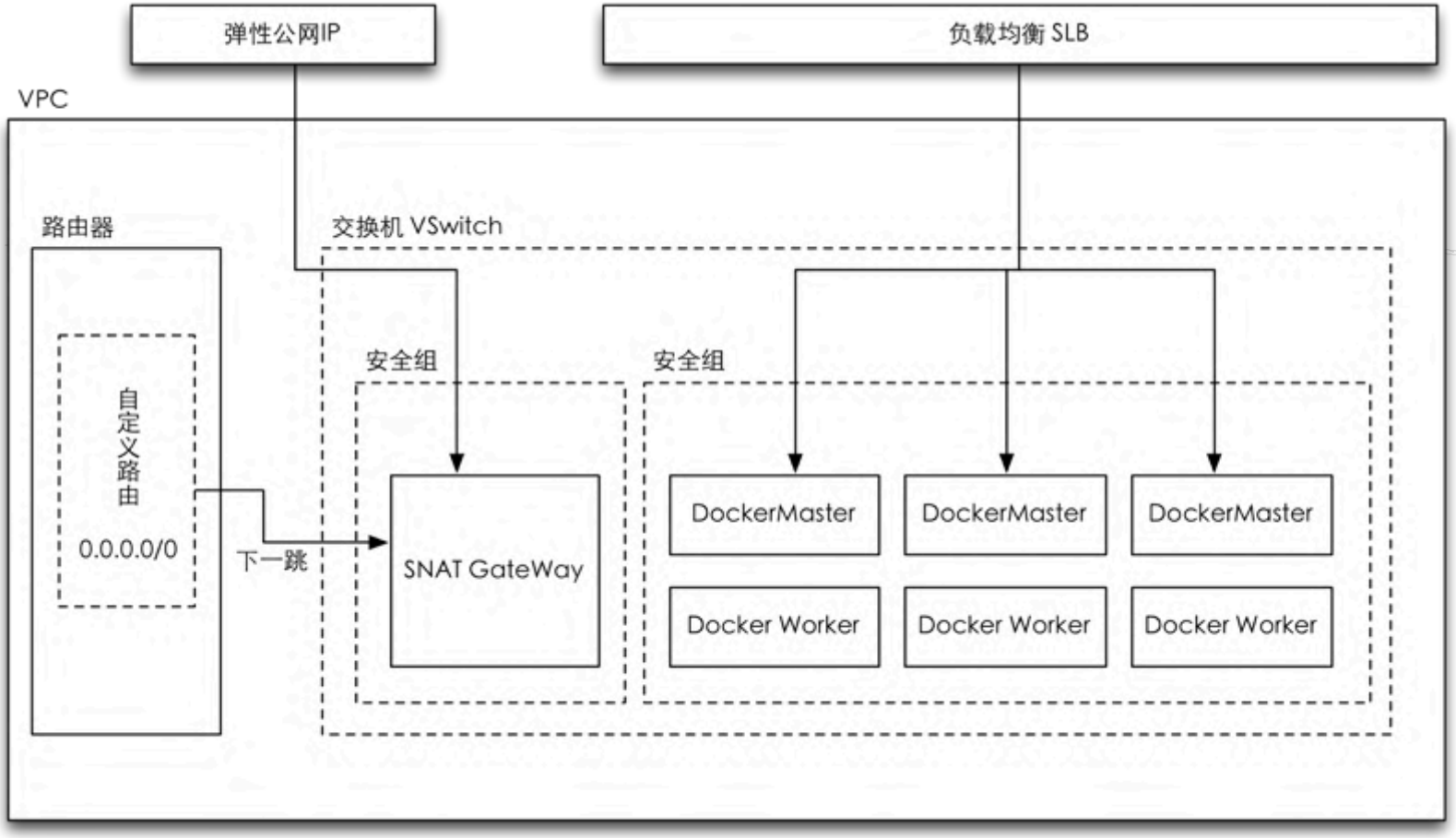


自动的通知机制

- 资源创建编排和依赖
- 信号机制
- WaitCondition
- WaitConditionHandle



在云端快速搭建Docker Swarm



Summary

- 一键部署
- 高度复用
  - 私有
  - 云市场
- Devops
- Infrastructure AS Code

谢谢聆听

阿里云资源编排团队博客
https://yq.aliyun.com/teams/12