阿里云+黎山

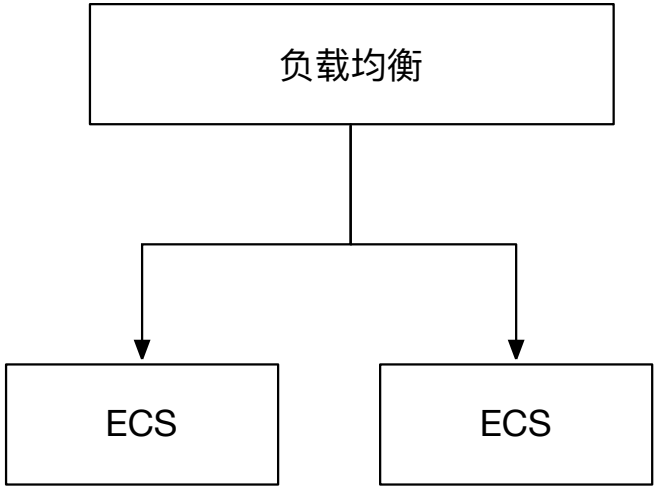
如何利用开源DevOps工具完成云上的自动运维

议题

- 通过实际应用场景讲解IaC的重要性
- Terraform/Packer的使用介绍
- 多个工具组合案例
- 操作演示

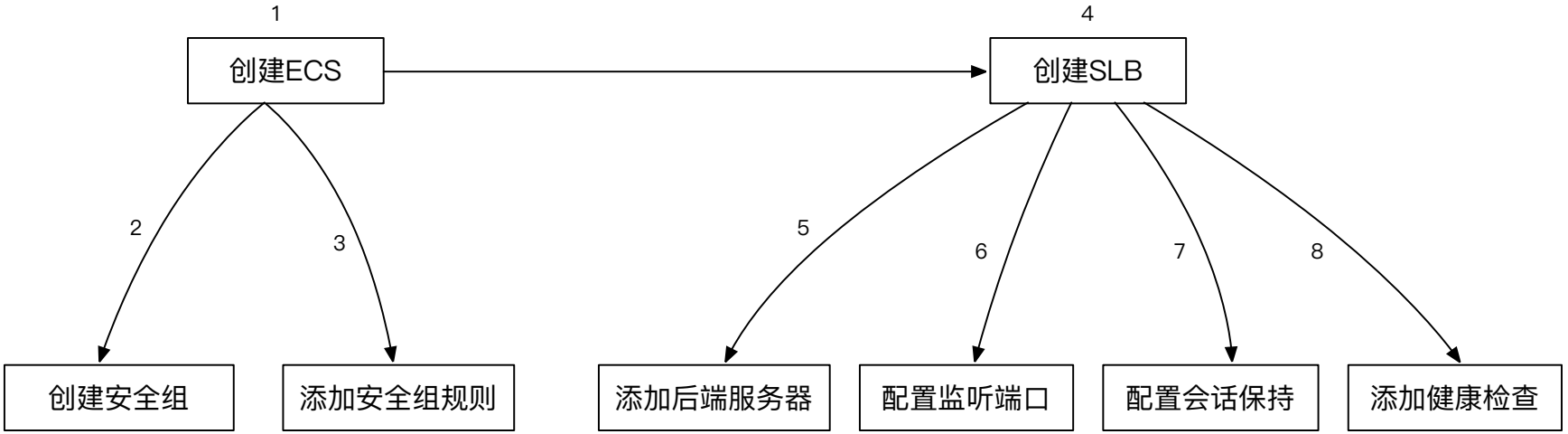
应用场景解析1

- 场景1： 某应用1，为了增大吞吐率，做了流量均衡处理、扩大并发数、缩短延迟，使用了负载均衡和ECS的组合，架构如下：



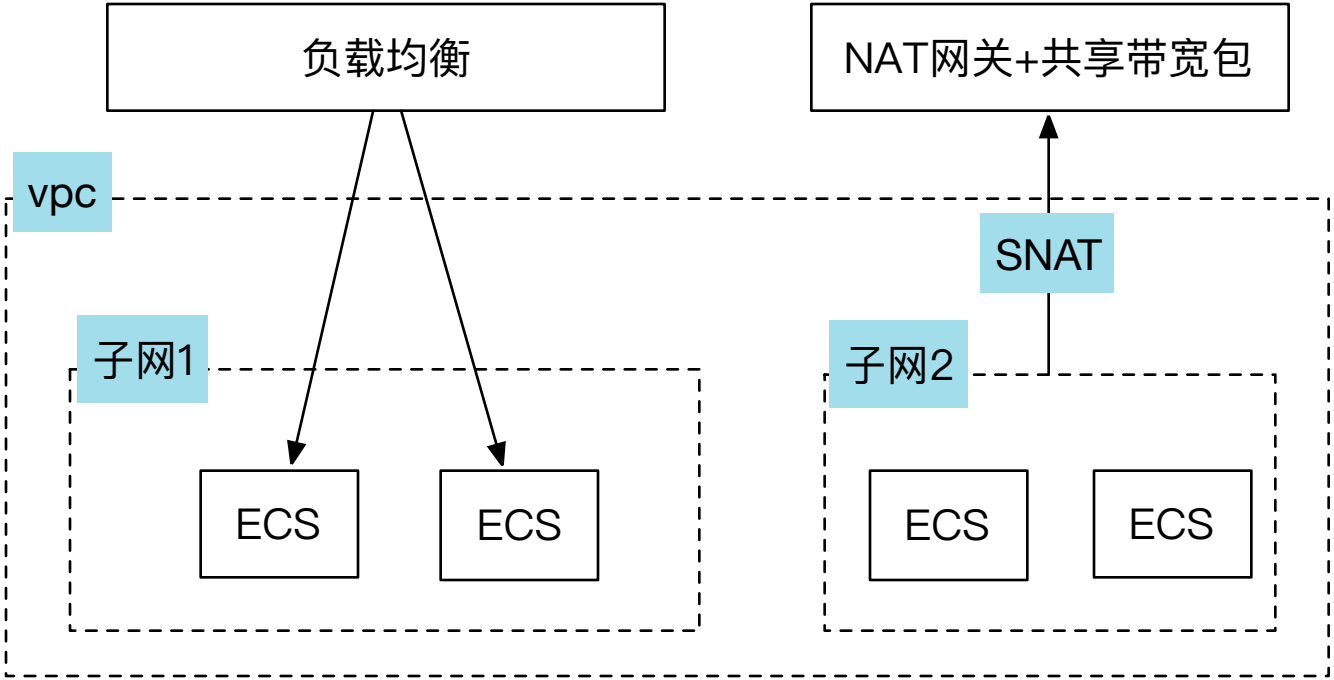
应用场景解析1

- 操作步骤：新建SLB、创建ECS，设定安全组，做安全组规则配置，然后在SLB增加后端服务器，配置监听端口、配置会话保持、添加健康检查。



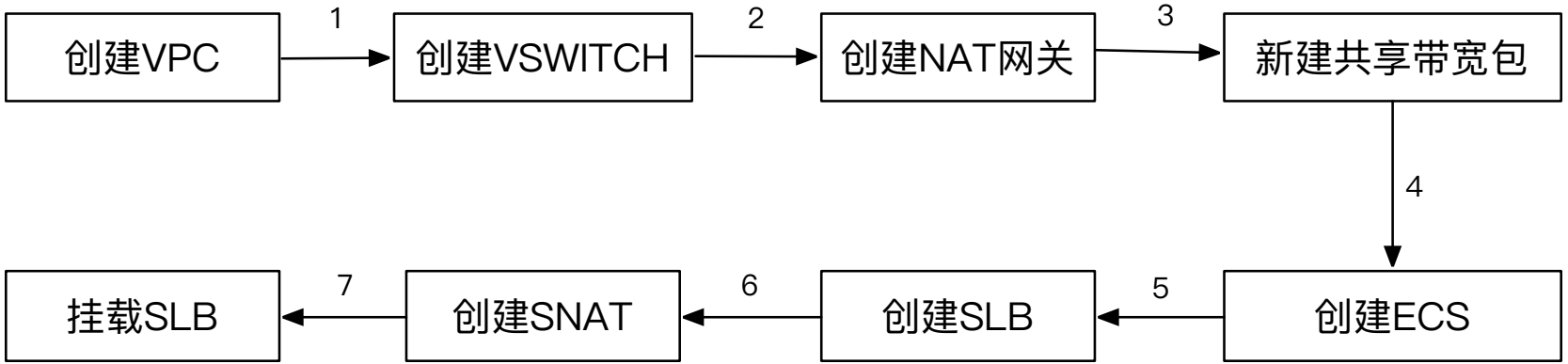
应用场景解析2

- 场景2： 某应用2，需要隔离的网络环境、或者需要使用VPC下的新功能，需要将应用搭建在VPC网络内，架构如下：



应用场景解析2

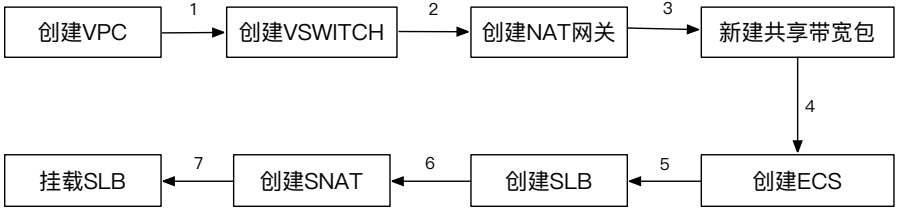
- 需要如下步骤 创建VPC、VSwitch、NAT网关、共享带宽包、ECS、SLB、SNAT、端品转发。其中还包括若干配置操作

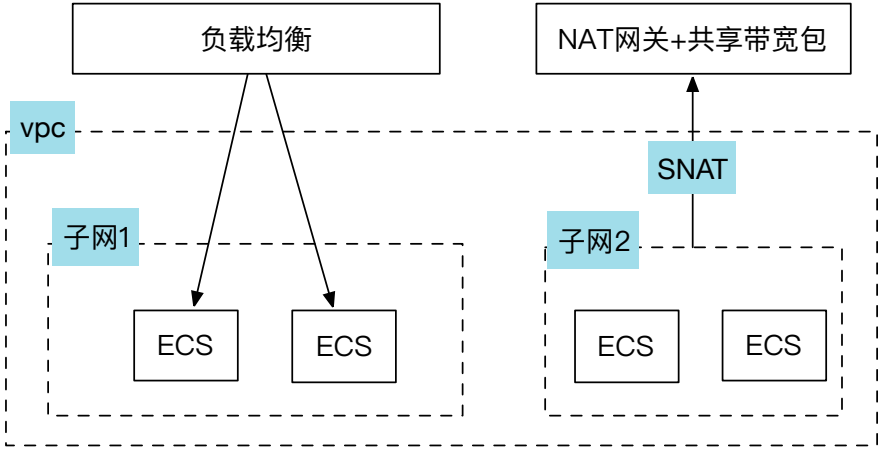


应用场景解析3

- 场景3： 某应用3，与应用2一样的基础设施需求，也需要隔离的网络环境、负载。

需要按照固定流程手工再做一次重复的工作，人员流动，文档不全，导致接手的人需要几天的时间熟悉环境及各种配置。

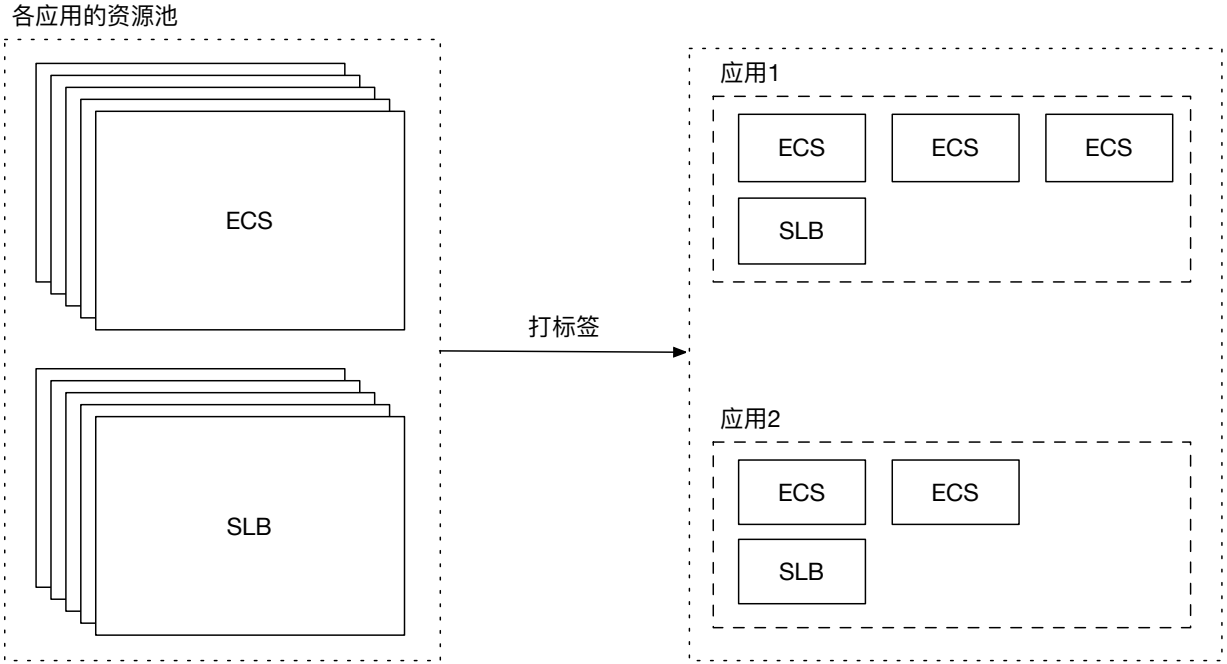




场景解析4

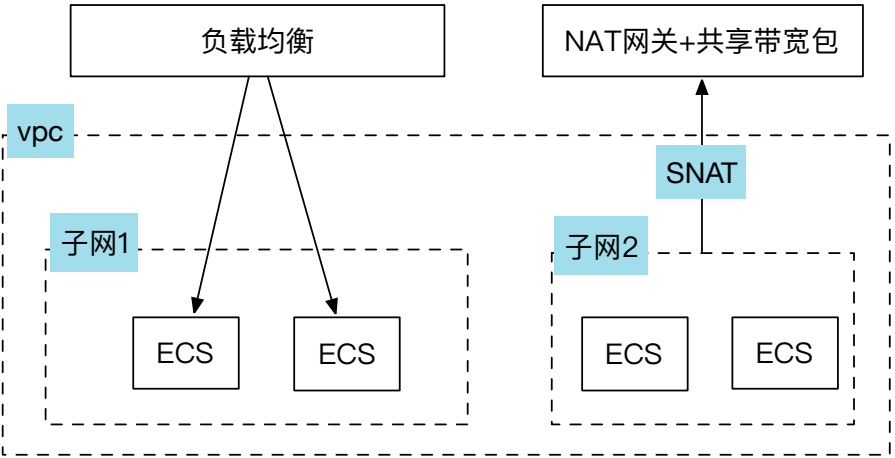
- 随着应用的增加，ECS、SLB等资源也在增加，希望通过“打标签”区分哪些资源属于哪些应用，将资源按照应用分组。

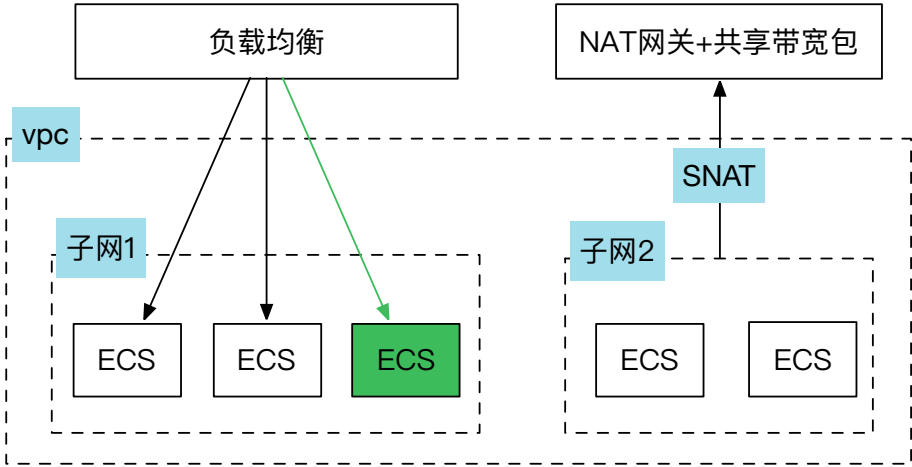
需要找到资源和应用的对应关系，再把每个资源都打上标签。



应用场景解析5

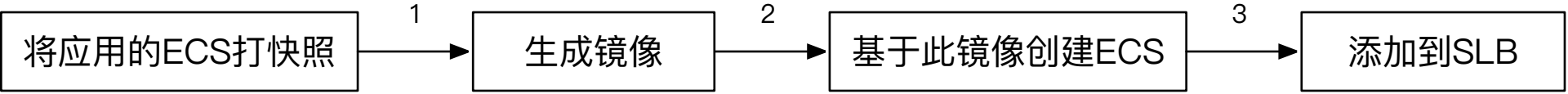
- 应用2深受市场欢迎，流量暴增，需要增加ECS以承载更多的并发和访问量，需要扩容一台与线上应用一致的ECS，挂载到SLB上。





应用场景解析5

- 需要如下步骤 基于线上应用的ECS打快照、生成镜像、创建基于此镜像的ECS、将此ECS添到SLB中。（若干配置步骤省略）



场景总结

- 以上场景的特点：
  - 操作流程、配置固定；
- 手工操作的缺点：
  - 效率低、时间长；
  - 可能导致错误；
  - 变更不能回滚；
  - 过程没有历史记录；
  - 过程不能审计（不知道是谁做了什么样的操作）

怎么办？自动化能自动化的一切！！！

自动化能自动化的一切

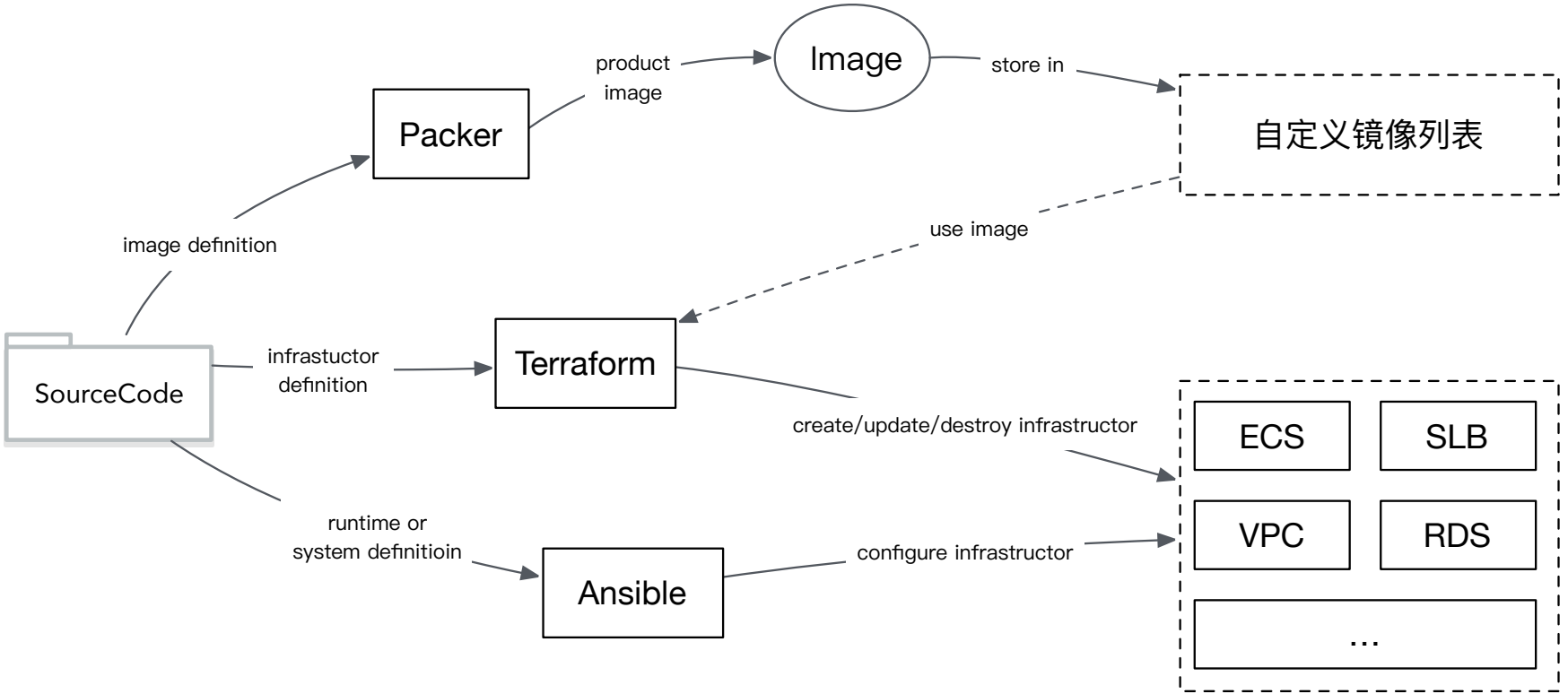
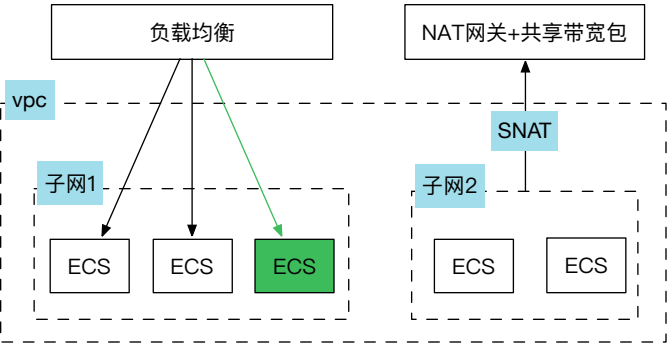


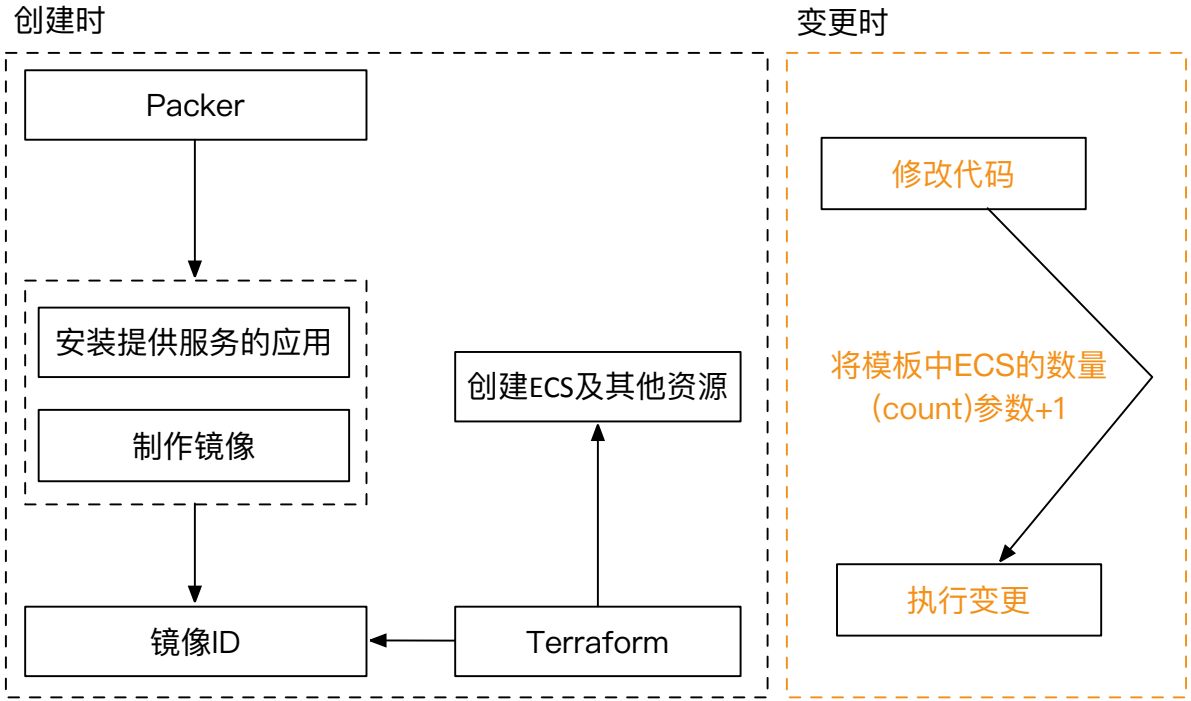
Image
product
image
Packer
store in
自定义镜像列表
image definition
use image
SourceCode
infrastuctor
definition
Terraform
create/update/destroy infrastructor
ECS
SLB
VPC
RDS
...
runtime or
system definitioin
Ansible
configure infrastructor

针对于场景5的IaC思路

关键词：与线上应用一致的ECS

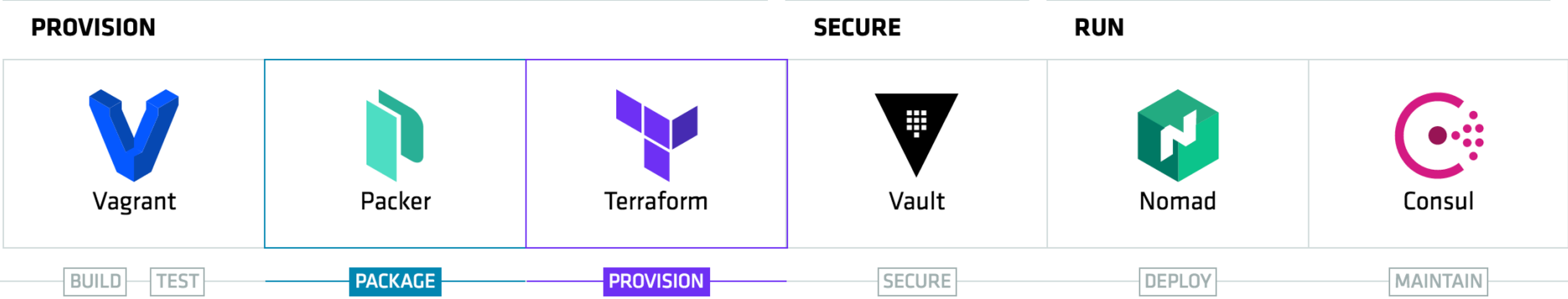
场景5的架构





Terraform/Packer介绍

HashiCorp家族



支持多平台

Aliyun、AWS、Azure、
OpenStack、VMWare…

开源

成熟、透明、
可自增强

Terraform安装

- 如何安装使用
  - https://www.terraform.io/downloads.html 下载对应平台的文件，放在指定目录中，如/usr/local/terraform0.9.2
  - 配置环境变量，以mac为例：
    - 在 /etc/profile 文件添加 “export TERRAFORM_HOME=/usr/local/terraform0.9.2”
    - 更新PATH，export PATH=$TERRAFORM_HOME:$PATH
    - 执行 source /etc/profile，使配置生效
  - 执行“terraform version”出现如下提示则表示安装成功：

```
➜ [redacted] git:(dev) ✗ terraform version
Terraform v0.9.2
```

Terraform模板

```
resource "alicloud_security_group" "group" {
  name = "sg-worker"
}
```

资源 别名

```
resource "alicloud_security_group_rule" "http-in" {
  type = "ingress"
  ip_protocol = "tcp"
  nic_type = "internet"
  policy = "accept"
  port_range = "80/80"
  priority = 1
  security_group_id = "${alicloud_security_group.group.id}"
  cidr_ip = "0.0.0.0/0"
}
```

安全组

```
resource "alicloud_instance" "instance" {
  instance_name = "worker-${format("%02d", count.index+1)}"
  image_id = "ubuntu_140405_64_40G_cloudinit_20161115.vhd"
  instance_type = "ecs.n1.small"
  count = "1"
  security_groups = ["${alicloud_security_group.group.*.id}"]
  internet_charge_type = "PayByTraffic"
  internet_max_bandwidth_out = "5"
  io_optimized = "optimized"
  password = "12345abc"
  allocate_public_ip = true
  availability_zone = ""
  instance_charge_type = "PostPaid"
  system_disk_category = "cloud_efficiency"

  tags {
    role = "worker"
    dc = "beijing"
  }
}
```

ECS

```
resource "alicloud_slb" "instance" {
  name = "slb_worker"
  internet_charge_type = "paybytraffic"
  internet = true

  listener = [
    {
      "instance_port" = "22"
      "lb_port" = "22"
      "lb_protocol" = "tcp"
      "bandwidth" = "10"
      "health_check_type" = "http"
      "persistence_timeout" = 3600
      "healthy_threshold" = 8
      "unhealthy_threshold" = 8
      "health_check_timeout" = 8
      "health_check_interval" = 5
      "health_check_http_code" = "http_2xx,http_3xx"
      "health_check_timeout" = 8
    }]
}
```

SLB

```
resource "alicloud_slb_attachment" "default" {
  slb_id = "${alicloud_slb.instance.id}"
  instances = ["${alicloud_instance.instance.*.id}"]
}
```

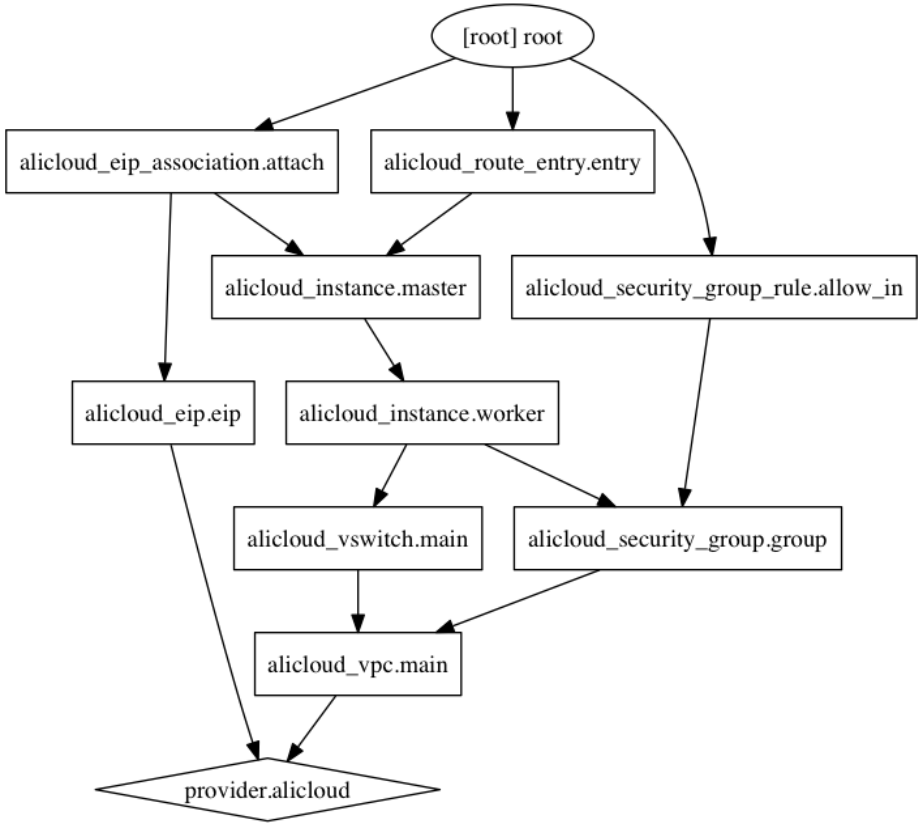
SLB挂载

Terraform命令

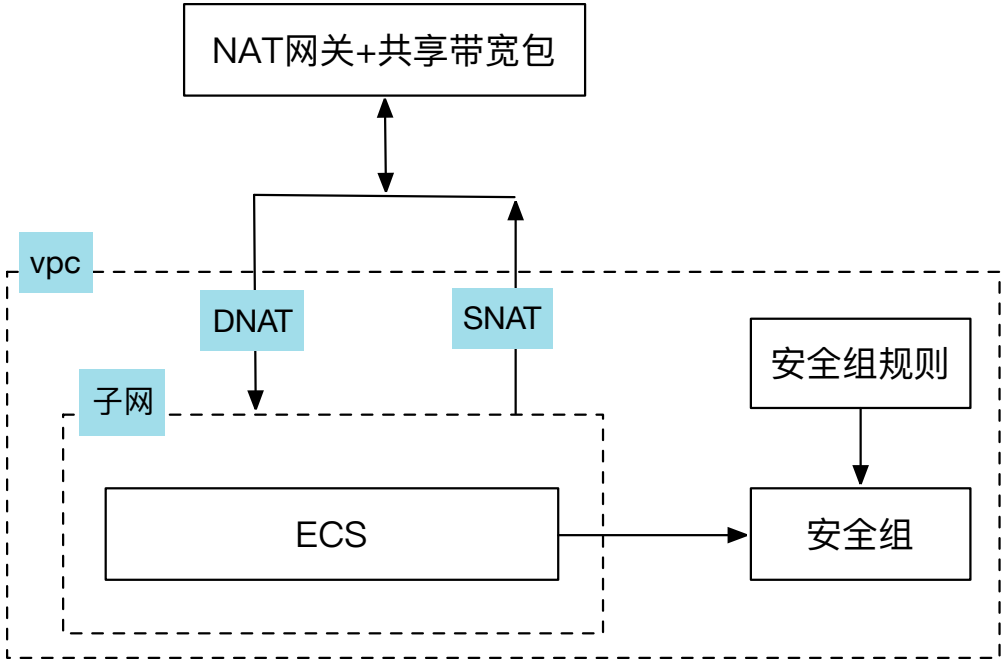
```
alicloud_instance.default: Creating...
  allocate_public_ip:          "" => "false"
  availability_zone:           "" => "cn-hangzhou-e"
  host_name:                   "" => "<computed>"
  image_id:                    "" => "ubuntu_140405_64_40G_cloudinit_20161115.vhd"
  instance_charge_type:        "" => "PostPaid"
  instance_name:               "" => "tf_vpc_snat"
  instance_type:               "" => "ecs.n1.small"
  internet_max_bandwidth_out:  "" => "0"
  io_optimized:                "" => "optimized"
  private_ip:                  "" => "<computed>"
  public_ip:                   "" => "<computed>"
  security_groups.#:           "" => "1"
  security_groups.1555784864:  "" => "sg-bp148zbcfw5tjob1m6sp"
  status:                      "" => "<computed>"
  subnet_id:                   "" => "<computed>"
  system_disk_category:        "" => "cloud_efficiency"
  system_disk_size:            "" => "<computed>"
  vswitch_id:                  "" => "vsw-bp1nnovl74ed84vzt9geg"
alicloud_nat_gateway.default: Creation complete
alicloud_snat_entry.default: Creating...
  snat_ip:           "" => "116.62.79.185"
  snat_table_id:     "" => "stb-bp1roq2ilyp5y6s7xn25y"
  source_vswitch_id: "" => "vsw-bp1nnovl74ed84vzt9geg"
alicloud_snat_entry.default: Creation complete
alicloud_instance.default: Still creating... (10s elapsed)
alicloud_instance.default: Still creating... (20s elapsed)
alicloud_instance.default: Still creating... (30s elapsed)
alicloud_instance.default: Creation complete
```

- 基本命令
  - terraform plan (预览)
  - terraform apply (执行)
  - terraform destroy (销毁)

资源拓普图


[root] root
alicloud_eip_association.attach
alicloud_route_entry.entry
alicloud_instance.master
alicloud_security_group_rule.allow_in
alicloud_eip.eip
alicloud_instance.worker
alicloud_vswitch.main
alicloud_security_group.group
alicloud_vpc.main
provider.alicloud

Terraform操作演示

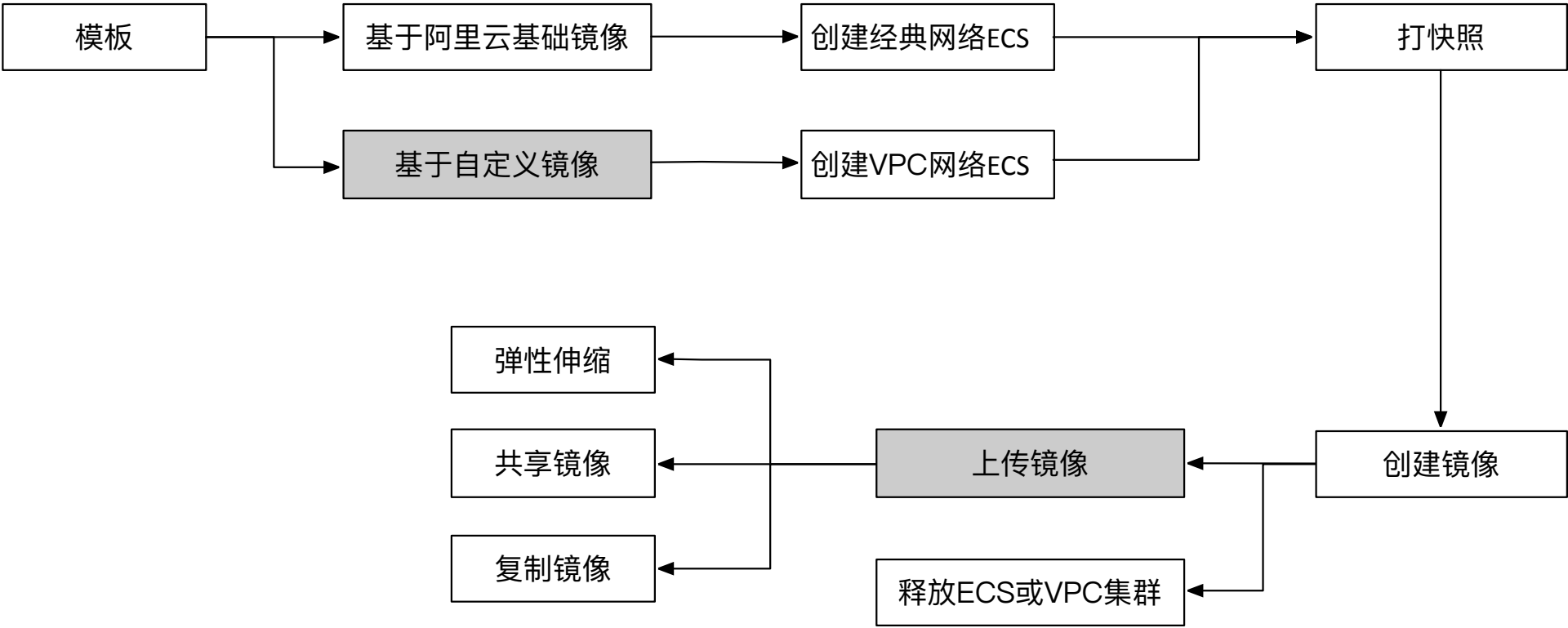


VPC集群：
NAT网关+SNAT+端口转发(DNAT)

Terraform操作演示

```
➜ alicloud-vpc-snat git:(dev) ✗ te
```

Packer介绍


模板
基于阿里云基础镜像
基于自定义镜像
创建经典网络ECS
创建VPC网络ECS
打快照
创建镜像
上传镜像
弹性伸缩
共享镜像
复制镜像
释放ECS或VPC集群

Packer模板

```
{
  "builders": [{
    "type":"alicloud-ecs",
    "access_key":"{{user `access_key`}}",
    "secret_key":"{{user `secret_key`}}",
    "region":"cn-beijing",
    "image_name":"packer_basic",
    "source_image":"centos_7_2_64_40G_base_20170222.vhd",
    "ssh_username":"root",
    "instance_type":"ecs.n1.tiny",
    "io_optimized":"true"
  }],
  "provisioners": [{
    "type": "shell",
    "inline": [
      "sleep 30",
      "yum install redis.x86_64 -y"
    ]
  }]
}
```

builder的类型

镜像中要处理的任务

packer build examples/alicloud/basic/alicloud.json

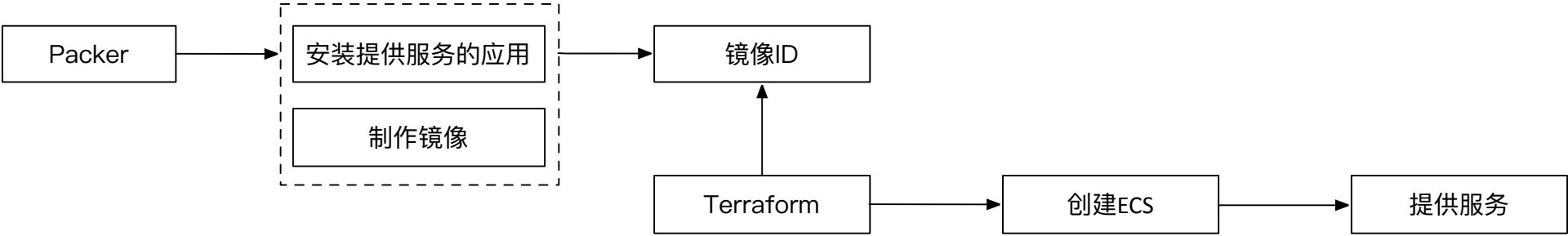
```
    alicloud-ecs: Complete!
==> alicloud-ecs: Start delete alicloud image snapshots
==> alicloud-ecs: Creating alicloud images: packer_basic
==> alicloud-ecs: Clean the created EIP
==> alicloud-ecs: Clean the created instance
==> alicloud-ecs: Clean the created security group
==> alicloud-ecs: Clean the created vSwitch
==> alicloud-ecs: Clean the created VPC
Build 'alicloud-ecs' finished.

==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:

cn-beijing: m-2ze9dy2omzlpspcca7zm
```

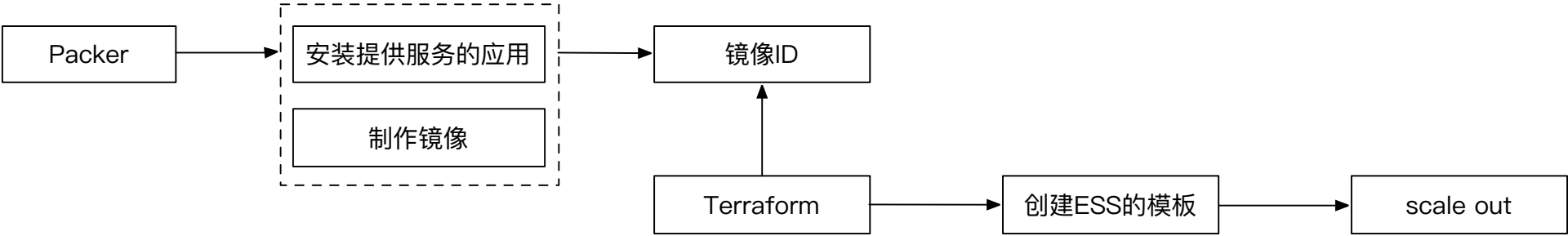
多个工具组合案例1

- 一次制作，重复使用：免去每次创建机器、安装服务的重复过程。



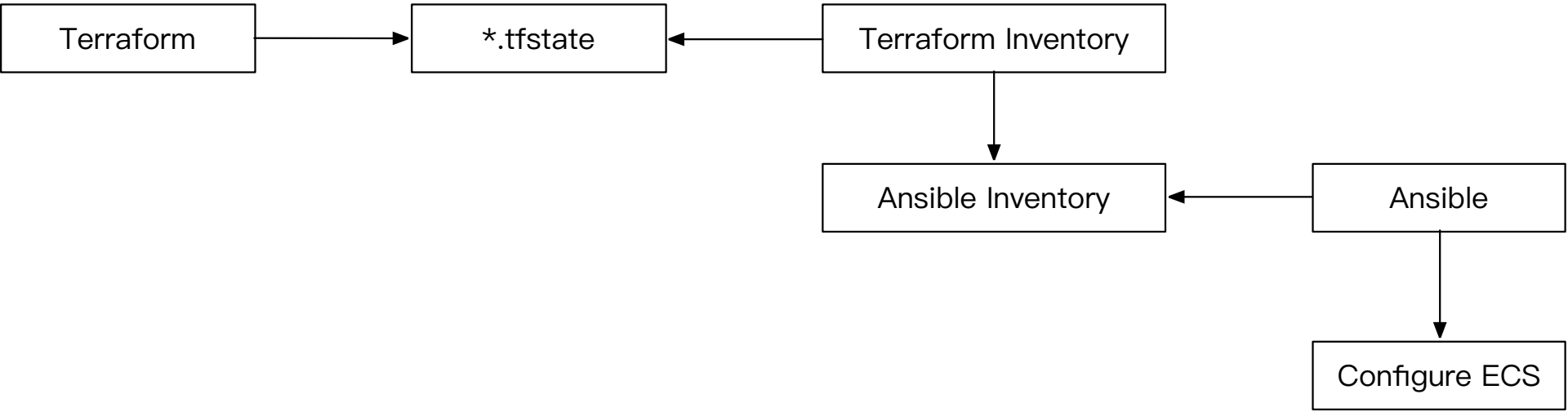
多个工具组合案例2

- 利用Packer制作需要弹性伸缩的应用镜像，Terraform的模板可以直接指定scale out时的镜像ID，scale out之后应用即可直接提供服务。

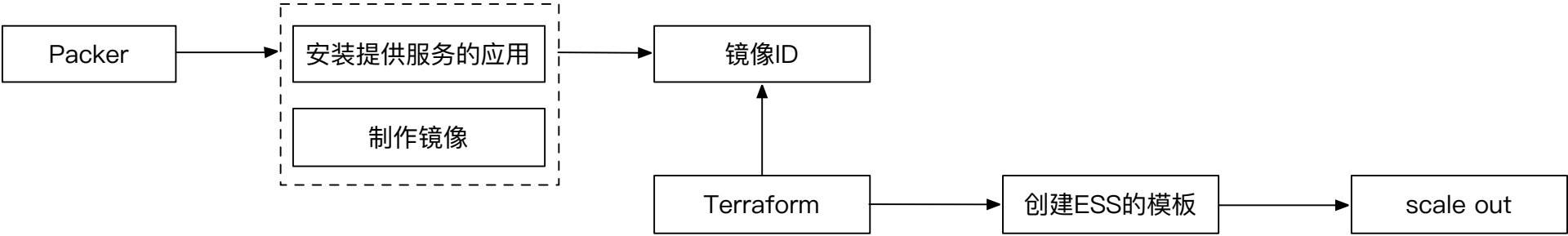


多个工具组合案例3

- Terraform生成tfstate文件，利用Terraform Inventory生成Ansible Inventory，Ansible可读取Ansible Inventory文件将应用或配置安装在指定的分组中。

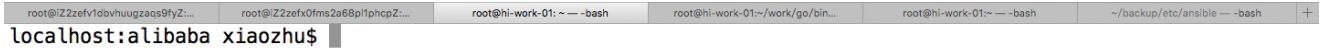


操作案例



- Packer制作Jenkins的自定义镜像，利用Terraform实现ESS的弹性伸缩，做扩容。

Packer(Jenkins)+Terraform组合演示



自动化能自动化的一切

- 用代码描述基础设施的创建、变更、销毁。
- 代码编写好，验证也是正确的，之后每次执行任务都不会出错。
- 快速，高效。
- 代码代替文档，有历史记录，可回滚。不用担心文档更新不及时或人员流动带来的“黑盒”问题。
- 不用通过访问生产环境，就能知道生产环境上的配置情况。
- 提高整个团队的DevOps能力。

参考

- 云栖公众号：https://yq.aliyun.com/opstools
- Terraform Github地址：https://github.com/alibaba/terraform-provider
- Packer Github地址：https://github.com/alibaba/packer-provider
- 开源DevOps工具的Github地址：https://github.com/alibaba/opstools
- Terraform 官方地址：https://www.terraform.io
- Packer官方地址：https://www.packer.io

扫码关注云栖公众号

THANKS