AWS Summit

AWS技术峰会 2015 · 上海

亚马逊 aws

ECS 功能演示

亚马逊高级技术客户经理

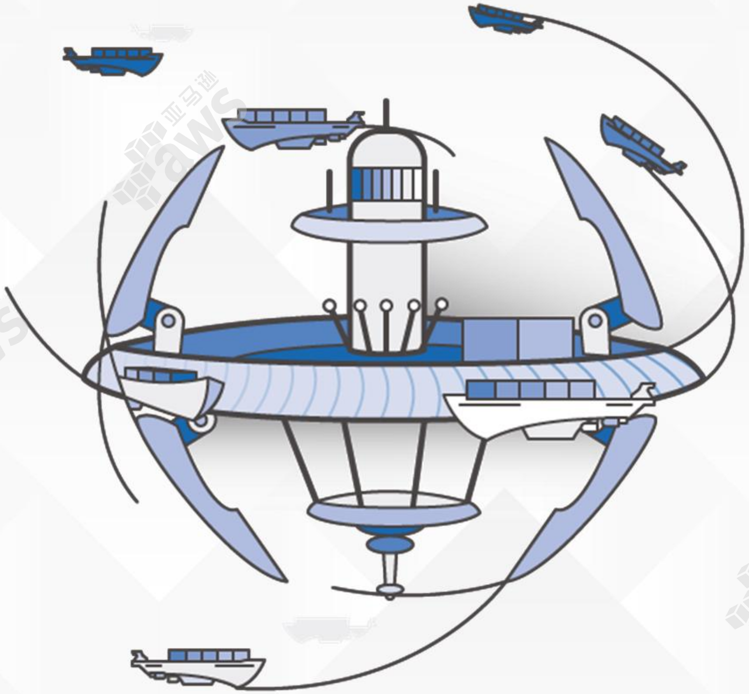
张浙

内容提要

- 从一个容器封装的简单应用程序开始
- 采用适用于ECS的实例创建集群
- 在ECS图形界面中部署容器应用
- 采用ecs cli创建集群及管理容器
- 构建基于ELB的弹性容器应用架构

ECS集群特性

- 简单易懂的集群管理方式
  - 无需自建集群管理器或容器配置管理系统，集中精力于开发容器化的应用程序
- 灵活的调度算法
  - 默认调度算法基于CPU内存等资源的使用情况找到最佳的容器运行位置
- 高性能的大规模集群
  - 数以万计的容器可以在几秒钟内完成部署任务而不增加额外的复杂性
- 安全性
  - 容器运行在自己的VPC中，不与其他用户共享计算资源
- 可扩展性
  - 集群支持通过最简单的API进行扩展
  - Auto Scaling & Multi-AZs



多容器应用样例

$ cat app.js

```
var express = require('express');
var morgan = require('morgan');
var app = express();
var PORT = process.env.PORT || 8080;

app.use(morgan('[:date[iso]] :method :url\t:status'));

// Redis Setup
var redis = require('redis'),
    client =
redis.createClient(process.env.DB_PORT,
process.env.DB_HOST, {});

client.on('connect', function() {
    console.log('Connected to Redis');
});

app.get("/", function (req, res) {
    getCount(function (err,reply) {
        var value = (reply == null ? 0 : parseInt(reply));
        res.status(200).send({count: value});
    });
});
app.put('/inc', function (req, res) {
    client.incr('count');
    res.status(204).end();
});
app.put('/dec', function (req, res) {
   client.decr('count');
…
```

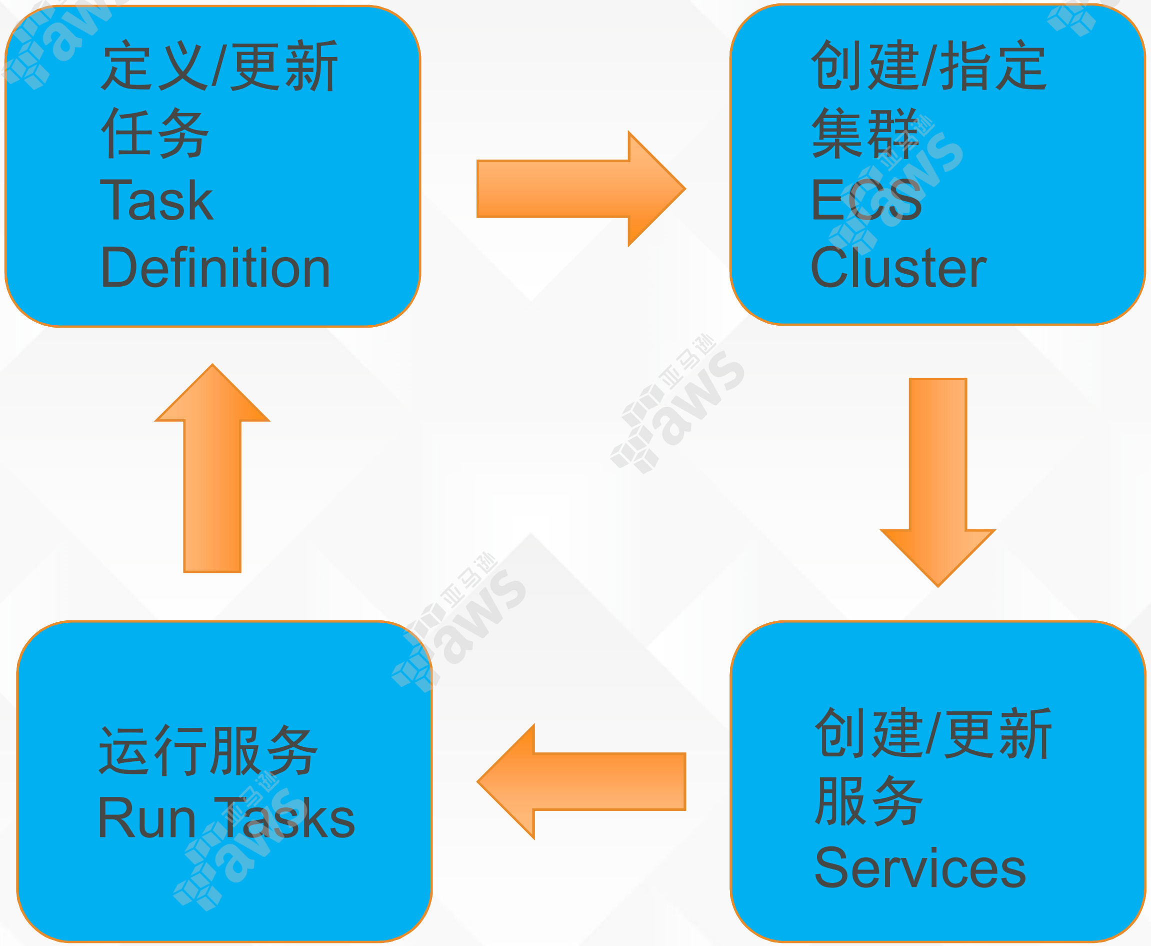
$ cat Dockerfile

```
FROM node:0.12.1-slim

EXPOSE 8080

ENV DB_HOST=redis

ENV DB_PORT=6379

ADD package.json package.json

RUN npm install --save

ADD app.js app.js

CMD node app.js
```

封装样例到容器中运行

```
$ docker build –t “zhang1980s/ecs-demo:redis-app” .

$ docker run –detach –p 6379:6379 –name=redis redis

$ docker run –p 8080:8080 –name=redis-app zhang1980s/ecs-demo:redis-
app

$ curl -X PUT http://localhost/inc
$ curl -X GET http://localhost | jq '.'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    11  100    11    0     0    868      0 --:--:-- --:--:-- --:--:--   916
{
  "count": 1
}

$ docker login

$ docker push zhang1980s/ecs-demo:redis-app
```

定义ECS容器化应用基本步骤



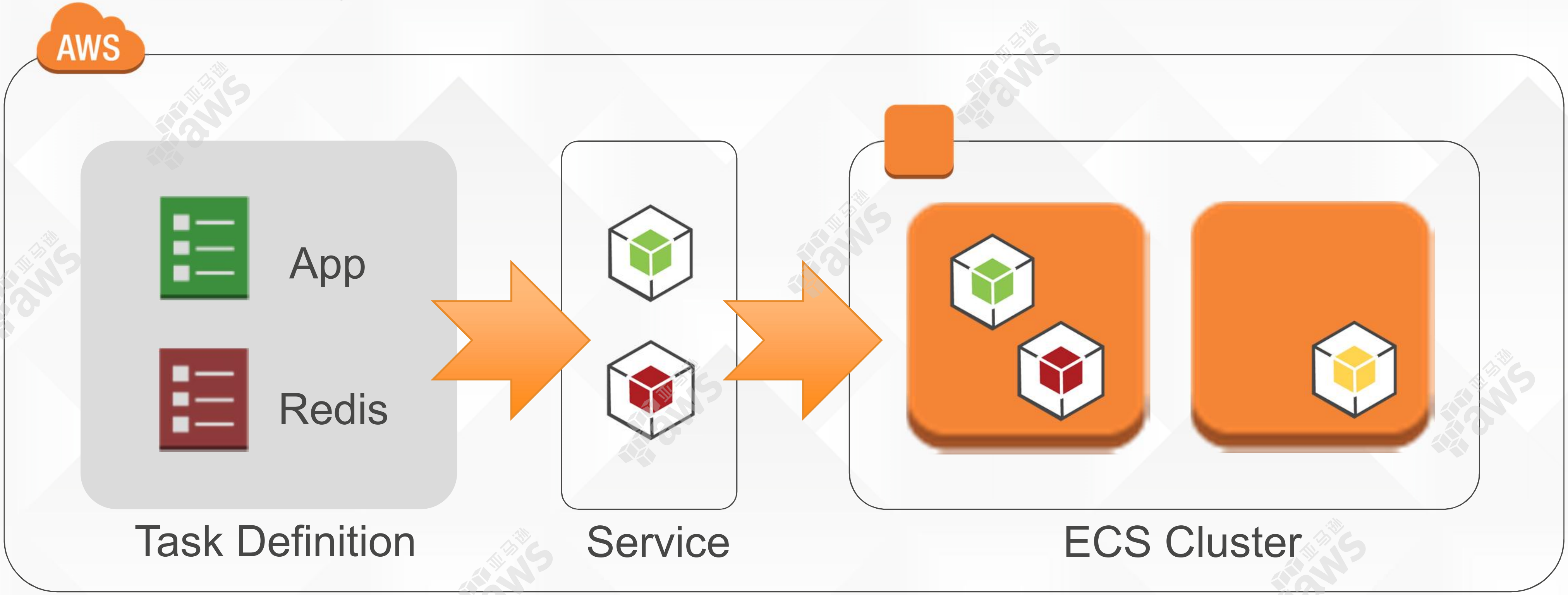
定义/更新任务 Task Definition

创建/指定集群 ECS Cluster

创建/更新服务 Services

运行服务 Run Tasks

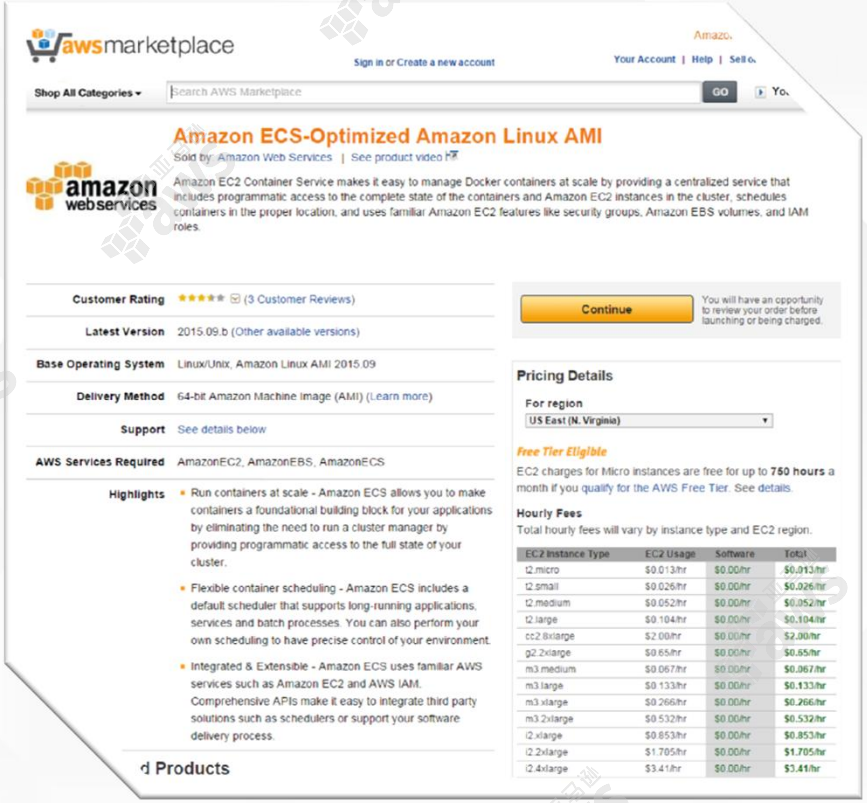
定义ECS容器化应用基本步骤



AWS cloud

构建基于ECS集群管理的应用

- 指定集群及集群节点
  - Amazon ECS-optimized Amazon Linux AMI
  - ecs-agenthttps://github.com/aws/amazon-ecs-agent
- User data
  - echo ECS_CLUSTER=ecs-demo-1 >> /etc/ecs/ecs.config



awsmarketplace

Amazo. Sign in or Create a new account Your Account | Help | Sell o.

Shop All Categories ▾ Search AWS Marketplace GO Yo.

Amazon ECS-Optimized Amazon Linux AMI

Sold by: Amazon Web Services | See product video

Amazon EC2 Container Service makes it easy to manage Docker containers at scale by providing a centralized service that includes programmatic access to the complete state of the containers and Amazon EC2 instances in the cluster, schedules containers in the proper location, and uses familiar Amazon EC2 features like security groups, Amazon EBS volumes, and IAM roles.

Customer Rating	★★★★★ (3 Customer Reviews)
Latest Version	2015.09.b (Other available versions)
Base Operating System	Linux/Unix, Amazon Linux AMI 2015.09
Delivery Method	64-bit Amazon Machine Image (AMI) (Learn more)
Support	See details below
AWS Services Required	AmazonEC2, AmazonEBS, AmazonECS

Highlights

- Run containers at scale - Amazon ECS allows you to make containers a foundational building block for your applications by eliminating the need to run a cluster manager by providing programmatic access to the full state of your cluster.
- Flexible container scheduling - Amazon ECS includes a default scheduler that supports long-running applications, services and batch processes. You can also perform your own scheduling to have precise control of your environment.
- Integrated & Extensible - Amazon ECS uses familiar AWS services such as Amazon EC2 and AWS IAM. Comprehensive APIs make it easy to integrate third party solutions such as schedulers or support your software delivery process.

You will have an opportunity to review your order before launching or being charged.

Pricing Details

For region: US East (N. Virginia) ▾

Free Tier Eligible

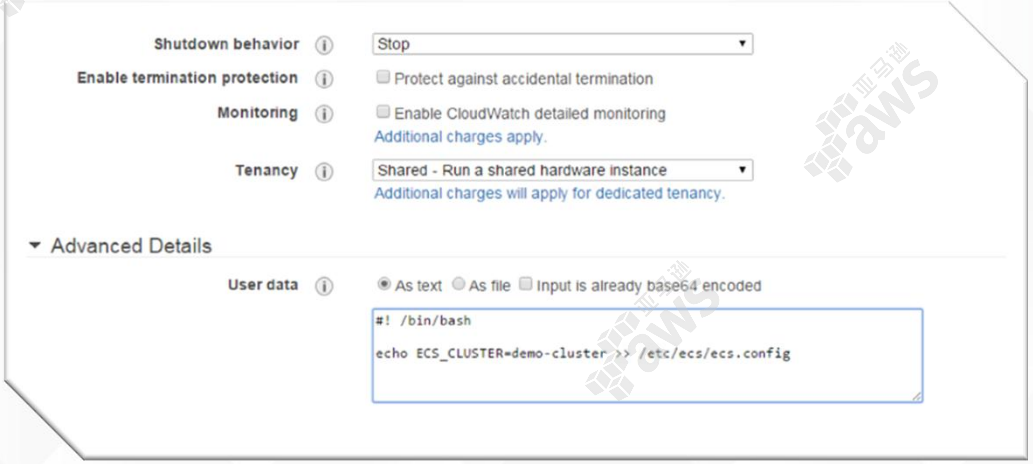
EC2 charges for Micro instances are free for up to **750 hours** a month if you qualify for the AWS Free Tier. See details.

Hourly Fees

Total hourly fees will vary by instance type and EC2 region.

EC2 Instance Type	EC2 Usage	Software	Total
t2.micro	$0.013/hr	$0.00/hr	$0.013/hr
t2.small	$0.026/hr	$0.00/hr	$0.026/hr
t2.medium	$0.052/hr	$0.00/hr	$0.052/hr
t2.large	$0.104/hr	$0.00/hr	$0.104/hr
cc2.8xlarge	$2.00/hr	$0.00/hr	$2.00/hr
g2.2xlarge	$0.65/hr	$0.00/hr	$0.65/hr
m3.medium	$0.067/hr	$0.00/hr	$0.067/hr
m3.large	$0.133/hr	$0.00/hr	$0.133/hr
m3.xlarge	$0.266/hr	$0.00/hr	$0.266/hr
m3.2xlarge	$0.532/hr	$0.00/hr	$0.532/hr
i2.xlarge	$0.853/hr	$0.00/hr	$0.853/hr
i2.2xlarge	$1.705/hr	$0.00/hr	$1.705/hr
i2.4xlarge	$3.41/hr	$0.00/hr	$3.41/hr

d Products



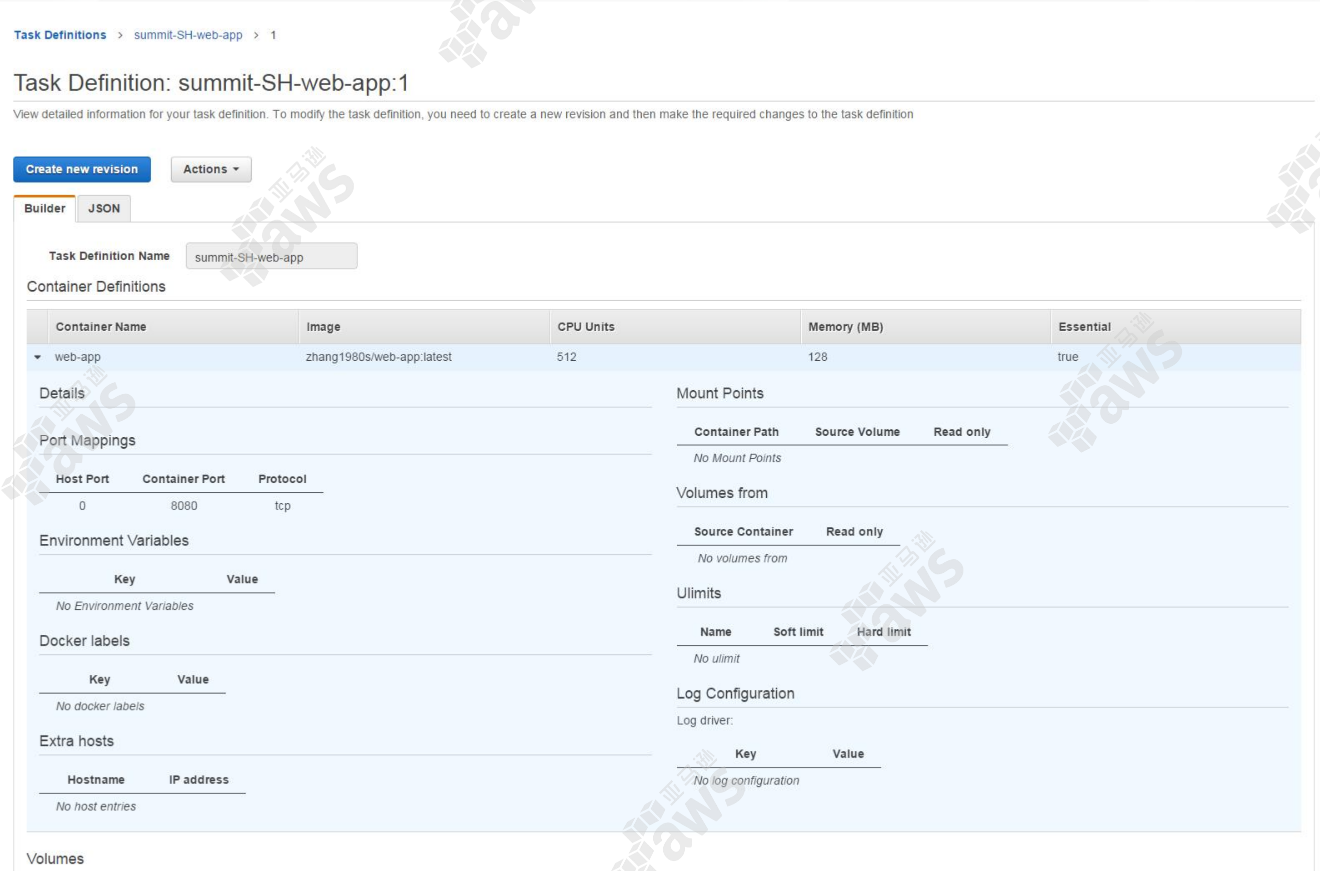
Shutdown behavior	Stop ▾
Enable termination protection	□ Protect against accidental termination
Monitoring	□ Enable CloudWatch detailed monitoring Additional charges apply.
Tenancy	Shared - Run a shared hardware instance ▾ Additional charges will apply for dedicated tenancy.

▾ Advanced Details

User data: ● As text ○ As file □ Input is already base64 encoded

```
#! /bin/bash

echo ECS_CLUSTER=demo-cluster >> /etc/ecs/ecs.config
```

创建Task Definition / Task



Task Definitions > summit-SH-web-app > 1

Task Definition: summit-SH-web-app:1

View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the required changes to the task definition

Task Definition Name: summit-SH-web-app

Container Definitions

Container Name	Image	CPU Units	Memory (MB)	Essential
web-app	zhang1980s/web-app:latest	512	128	true

Details

Port Mappings

Host Port	Container Port	Protocol
0	8080	tcp

Environment Variables

Key	Value
No Environment Variables	

Docker labels

Key	Value
No docker labels	

Extra hosts

Hostname	IP address
No host entries	

Mount Points

Container Path	Source Volume	Read only
No Mount Points		

Volumes from

Source Container	Read only
No volumes from	

Ulimits

Name	Soft limit	Hard limit
No ulimit		

Log Configuration

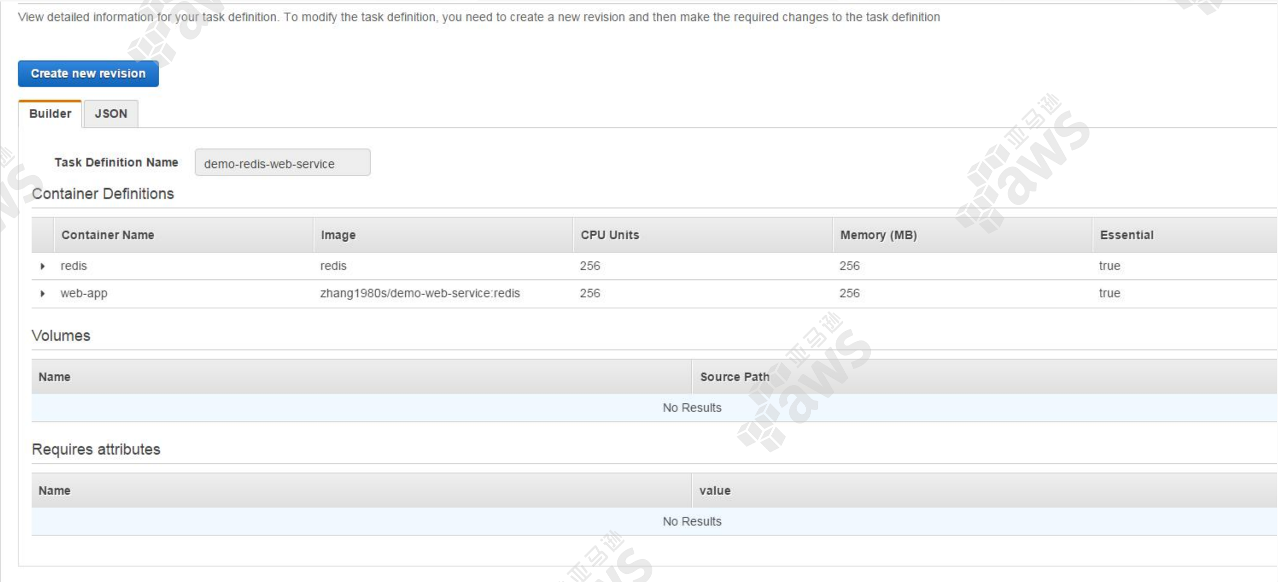
Log driver:

Key	Value
No log configuration	

Volumes

- Task Definition
  - 定义容器镜像
  - CPU/内存使用
  - 端口映射信息
  - 其他容器连接信息
  - 容器执行的命令
  - 传递到容器中的环境变量
  - 数据卷定义
  - 任务和容器的关联
- 运行Task
  - Task Definition实例化运行

定义多容器器的Task Definition



View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the required changes to the task definition

Task Definition Name: demo-redis-web-service

Container Definitions

Container Name	Image	CPU Units	Memory (MB)	Essential
redis	redis	256	256	true
web-app	zhang1980s/demo-web-service:redis	256	256	true

Volumes

Name	Source Path
No Results	

Requires attributes

Name	value
No Results	

- 从AWS CLI创建/更新Task Definition

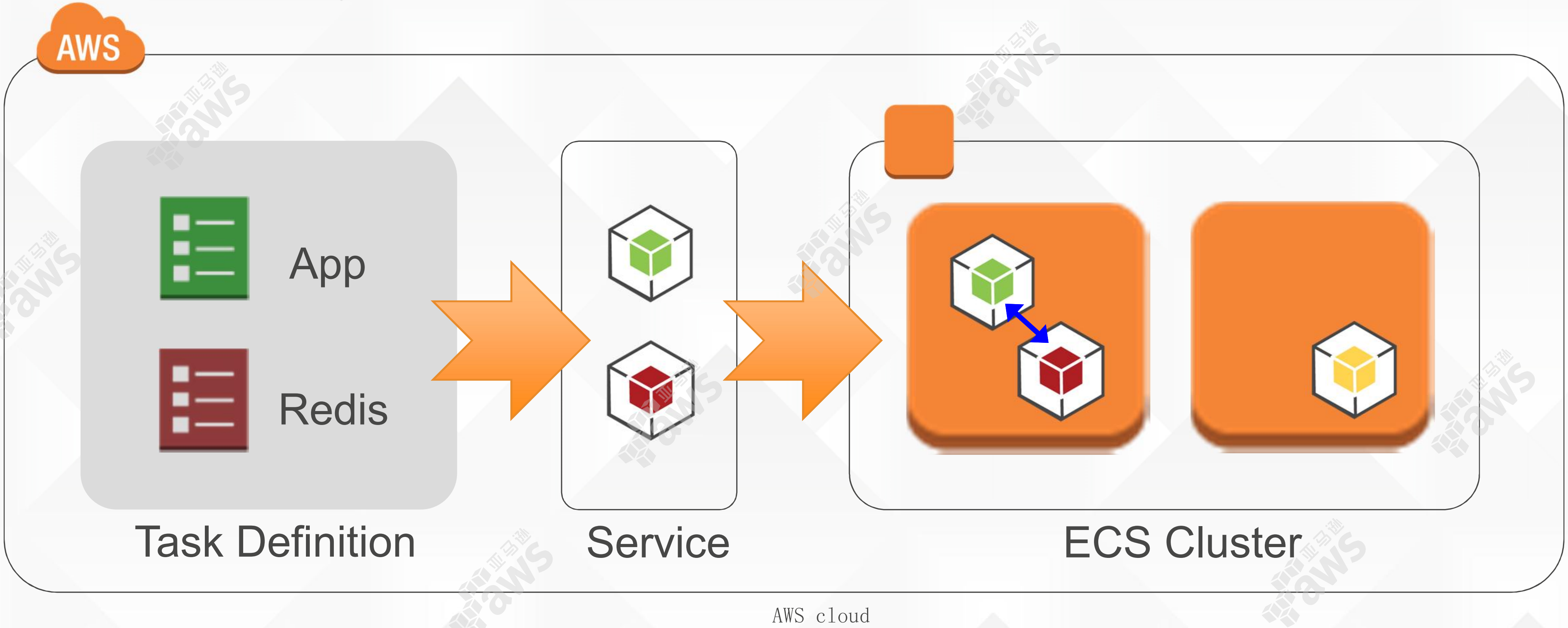
```
$ aws ecs register-task-definition --cli-input-json
file://path/to/file/task-definition.json
```

ECS Command Line Interface – ecs-cli

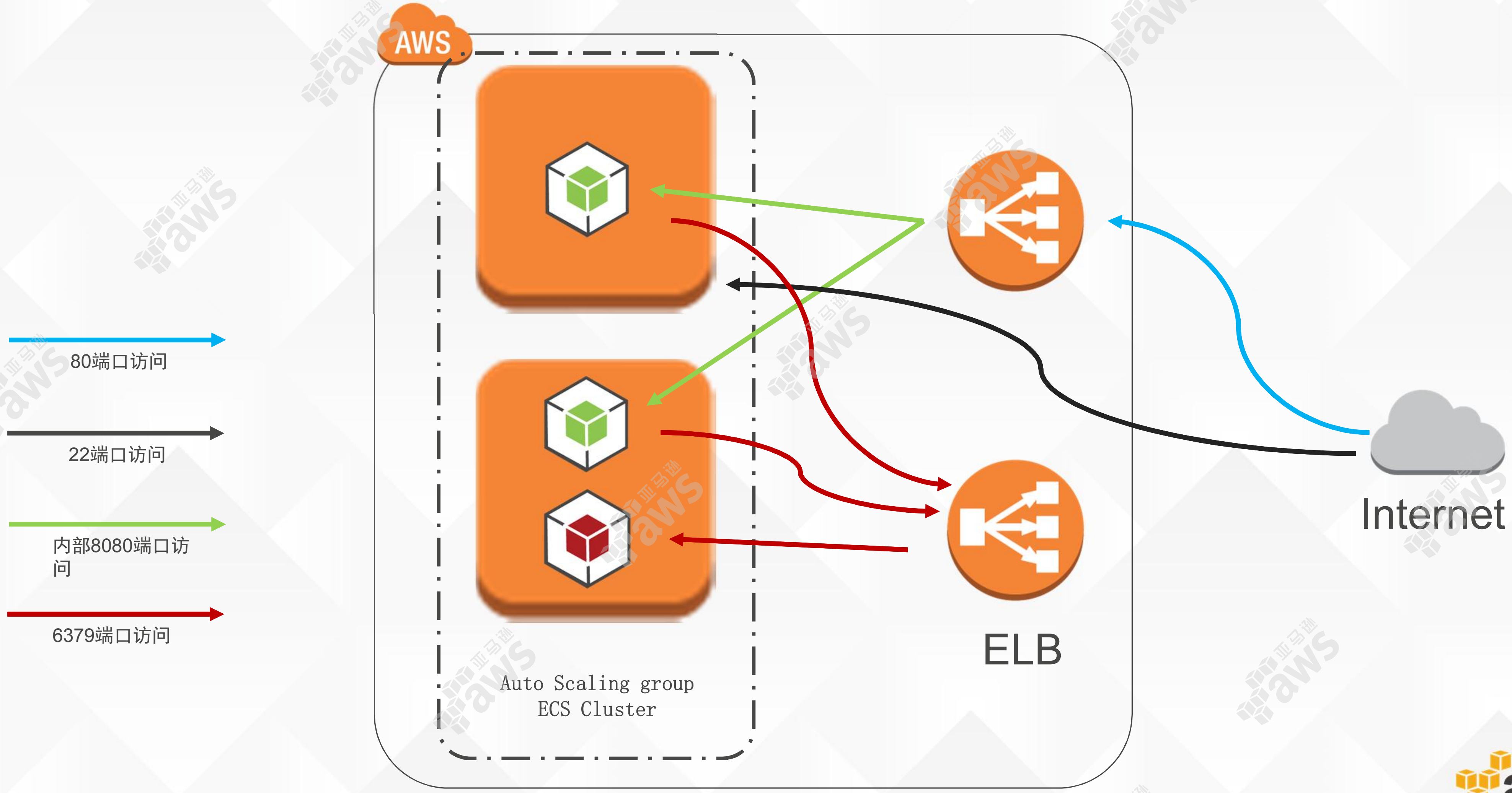
- 提供简化的ECS集群操作功能
- 支持集群管理/更新，服务管理/更新，任务管理/更新以及集群监控功能
- 从Linux/Mac本地直接远程操作ECS集群
- 加入Compose支持，更好兼容现有的多容器应用
- 开源项目 https://github.com/aws/amazon-ecs-cli

```
$ ecs-cli configure –region us-west-2 –access-key
<ACCESS_KEY_ID> -secret-key <SECRET_ACCESS_KEY> --
cluster <CLUSTER_NAME>

$ ecs-cli up –keypair <mykey> --capability-iam –size 2 –instance-type
t2.medium

$ ecs-cli compose –file docker-compose.yml up

$ ecs-cli ps

$ ecs-cli compose –file docker-compose.yml scale 2

$ ecs-cli compose –file docker-compose.yml down

$ ecs-cli compose –file docker-compose.yml service up

$ ecs-cli compose –file docker-compose.yml service rm

$ ecs-cli compose down --force
```

基于Link的容器间通信（紧耦合）



基于ELB创建弹性的容器化服务



80端口访问

22端口访问

内部8080端口访问

6379端口访问

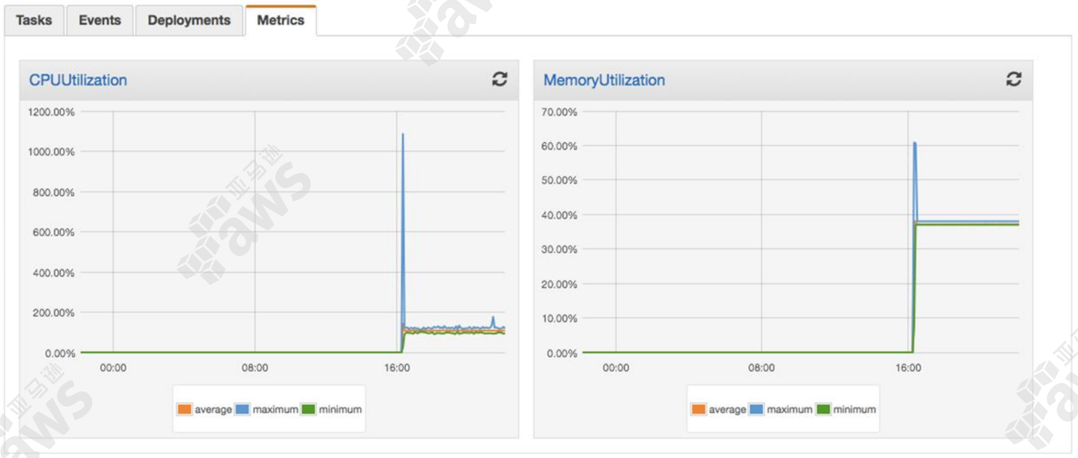
AWS

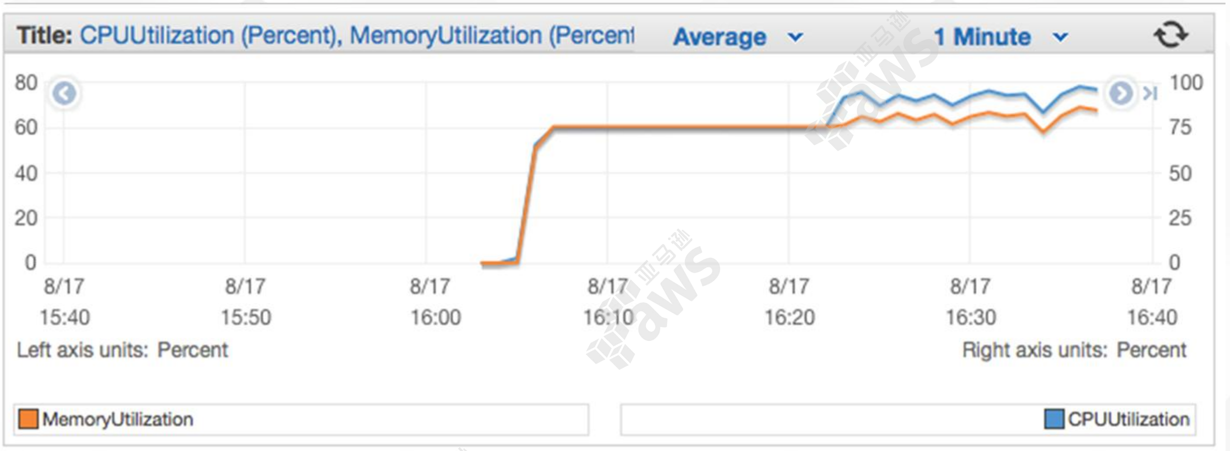
Auto Scaling group
ECS Cluster

ELB

Internet

ECS状态监控

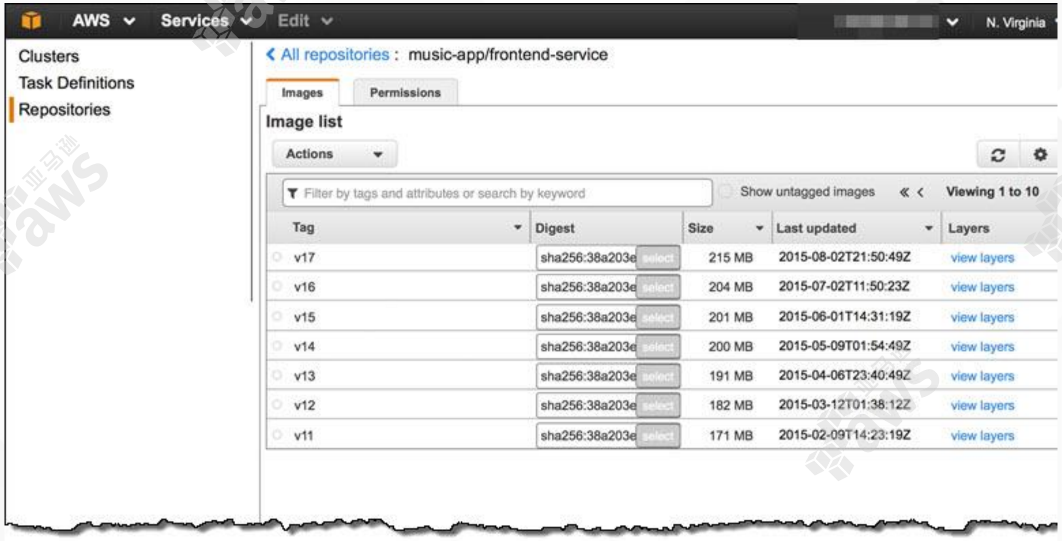




- 性能监控
  - ECS Cluster
  - Cloud Watch
- API监控
  - Cloud Trail
- 事件监控
  - ECS Cluster event

即将发布： Amazon EC2 Container Registry

- 完全托管
  - 无需安装任何管理软件及提供基础设施资源即可获得的Docker容器Registry。
- 安全
  - 通过HTTPS传输容器镜像，自动对容器镜像进行加密
  - 通过AWS IAM策略管理镜像访问权限
- 高可用
  - ECR基于高可用高持久性及冗余架构设计，实现对应用程序的可靠部署
- 简化工作流
  - ECR和ECS以及Docker CLI高度集成，实现开发和生产环境工作流的简化

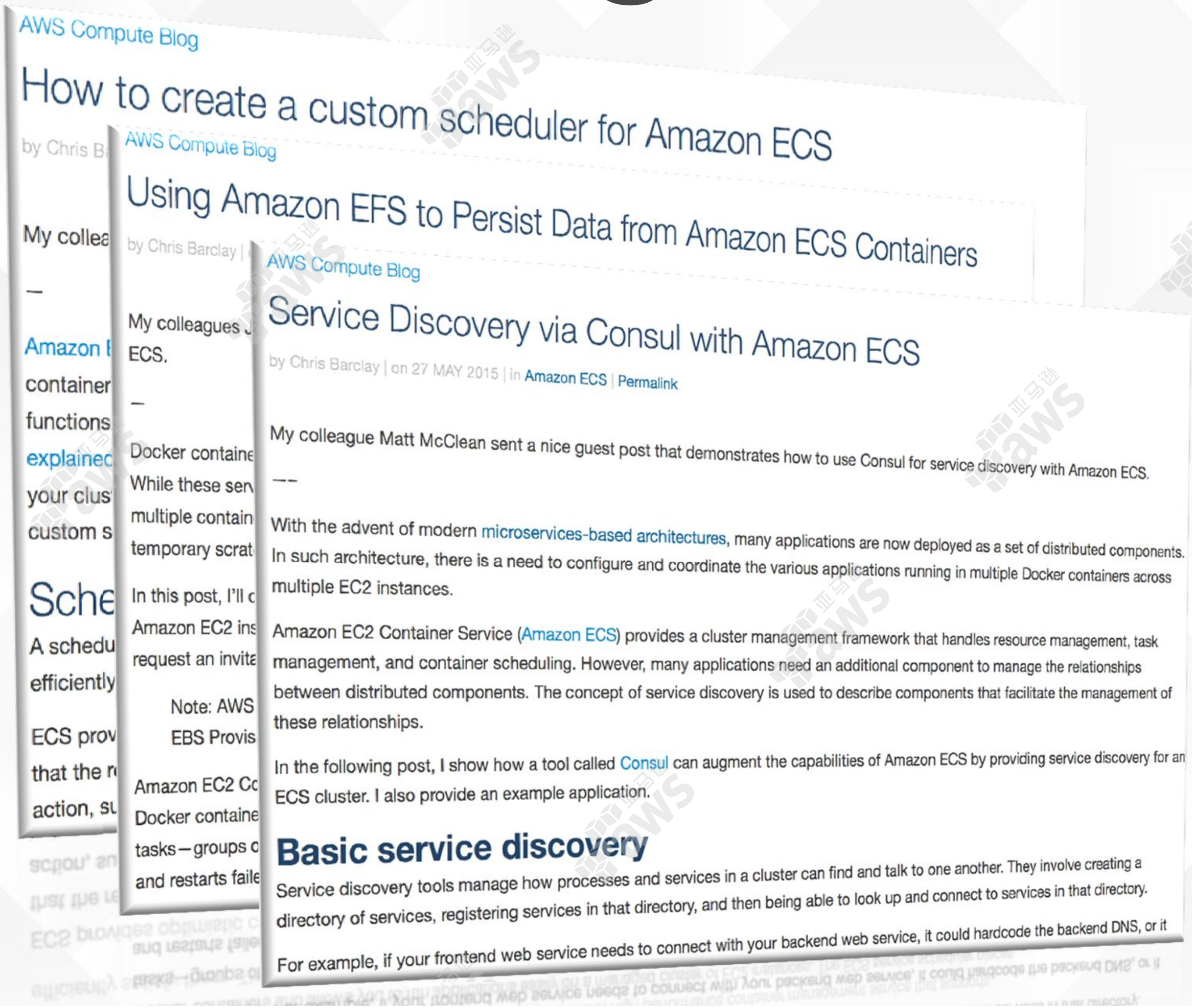


All repositories : music-app/frontend-service

Image list

Tag	Digest	Size	Last updated	Layers
v17	sha256:38a203e	215 MB	2015-08-02T21:50:49Z	view layers
v16	sha256:38a203e	204 MB	2015-07-02T11:50:23Z	view layers
v15	sha256:38a203e	201 MB	2015-06-01T14:31:19Z	view layers
v14	sha256:38a203e	200 MB	2015-05-09T01:54:49Z	view layers
v13	sha256:38a203e	191 MB	2015-04-06T23:40:49Z	view layers
v12	sha256:38a203e	182 MB	2015-03-12T01:38:12Z	view layers
v11	sha256:38a203e	171 MB	2015-02-09T14:23:19Z	view layers

更多Container@AWS技术方案



AWS Compute Blog

How to create a custom scheduler for Amazon ECS

by Chris B

My collea

–

Amazon I
container
functions
explained
your clus
custom s

Sche

A schedu
efficiently

ECS prov
that the r
action, su

AWS Compute Blog

Using Amazon EFS to Persist Data from Amazon ECS Containers

by Chris Barclay |

My colleagues
ECS.

–

Docker containe
While these ser
multiple contain
temporary scrat

In this post, I'll c
Amazon EC2 ins
request an invita

Note: AWS
EBS Provis

Amazon EC2 Co
Docker containe
tasks – groups c
and restarts faile

AWS Compute Blog

Service Discovery via Consul with Amazon ECS

by Chris Barclay | on 27 MAY 2015 | in Amazon ECS | Permalink

My colleague Matt McClean sent a nice guest post that demonstrates how to use Consul for service discovery with Amazon ECS.

—

With the advent of modern microservices-based architectures, many applications are now deployed as a set of distributed components. In such architecture, there is a need to configure and coordinate the various applications running in multiple Docker containers across multiple EC2 instances.

Amazon EC2 Container Service (Amazon ECS) provides a cluster management framework that handles resource management, task management, and container scheduling. However, many applications need an additional component to manage the relationships between distributed components. The concept of service discovery is used to describe components that facilitate the management of these relationships.

In the following post, I show how a tool called Consul can augment the capabilities of Amazon ECS by providing service discovery for an ECS cluster. I also provide an example application.

Basic service discovery

Service discovery tools manage how processes and services in a cluster can find and talk to one another. They involve creating a directory of services, registering services in that directory, and then being able to look up and connect to services in that directory.

For example, if your frontend web service needs to connect with your backend web service, it could hardcode the backend DNS, or it

- http://aws.amazon.com/documentation/ecs/
- https://aws.amazon.com/ecs/
- https://aws.amazon.com/blogs/compute/
- http://aws.amazon.com/whitepapers/
- https://aws.amazon.com/docker/

Thank You