**实战题目**

**需求介绍：**

使用提供的可执行的GO二进制文件（server）部署一个高可用，可伸缩且高效的Web应用程序（请参考下面所述完成系统配置部署）。该应用程序可使用缓存等技术加快响应，并需具有水平缩放能力。选手需依据AWS的WA（Well-Architected）框架不断优化其架构，满足卓越运营、安全、可靠、性能效率及成本优化的基本要求。

二进制文件server是可以运行的文件，可部署到ec2上（t2-micro机型）其实现了应用的处理订单的功能。下载地址：

https://ws-china-gameday.s3-us-west-2.amazonaws.com/gd1/server-d1128.bin

配置文件server.ini用于server控制连接、操作数据库及读写系统日志等的配置，其模板下载地址：

https://ws-china-gameday.s3-us-west-2.amazonaws.com/gd1/server.ini

系统的数据库表定义：

CREATE TABLE IF NOT EXISTS unicorntable (

id SERIAL PRIMARY KEY,

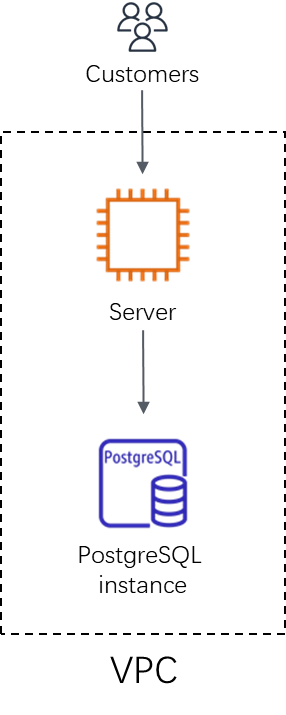
requestid VARCHAR(255),

requestvalue VARCHAR(255),

hits INT

);

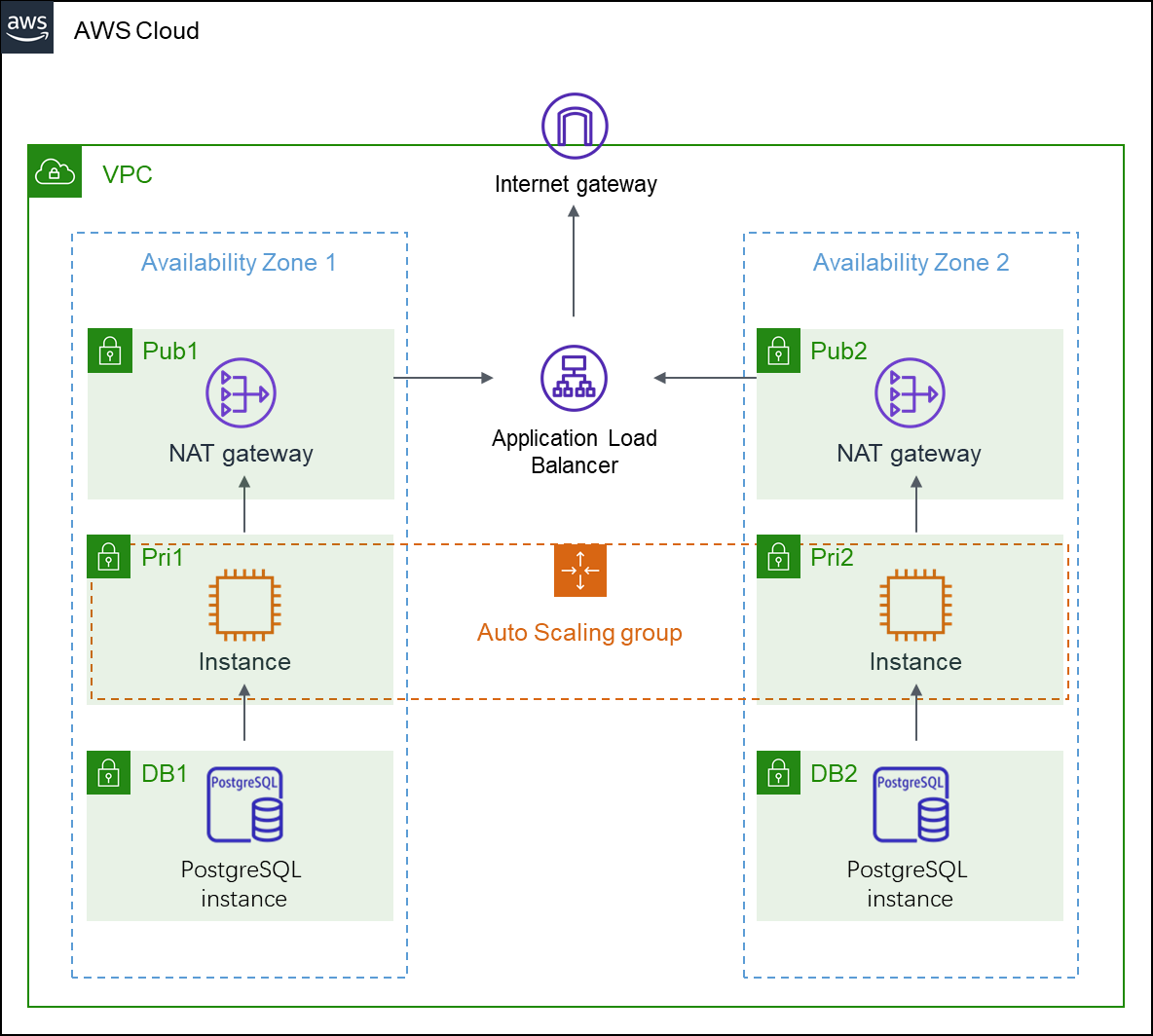
**部署架构图：**



**开始搭建**

**首先思考：**

由于题目所提供的只是初步架构图，所以我们应该根据WA架构思考一种高效经济的架构。在不使用容器的情况向，笔者构思了这么一个架构：



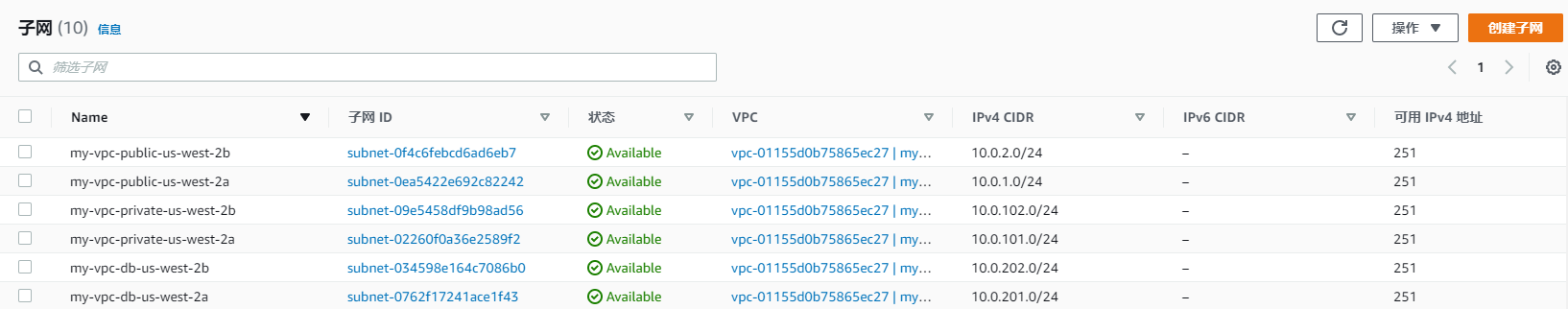
然后我们开始搭建这个架构。

**开始搭建：**

首先为了确保速度，一般比赛时我们会把很多公式化的设施用Cloudformation或是第三方搭建服务等进行脚本化搭建，既减少了错误率也提高了速度。所以这次演示不会详细介绍部分搭建步骤。

**第一步：搭建VPC**

VPC经典的划分方式，根据功能划分子网，目的是方便管理以及确保安全，因为这次需要用到数据库，所以除去共有子网和私有子网外，我们还应该搭建数据库专用的子网。



**第二步：创建安全组**

根据WA架构中安全部分“权限最小化”的原则，安全组因只开放最小端口，且限定流量的来源。而且每个组件应有自己唯一的安全组，绝对不可以多组件混用一个安全组。在前期测试时，可以给Server的安全组开放来源0.0.0.0/0的端口，但是架构部署完成时必须删除这些规则。所以这次需要三个安全组。

“sg\_elb”：用于负载均衡器，开放8080端口，源0.0.0.0/0

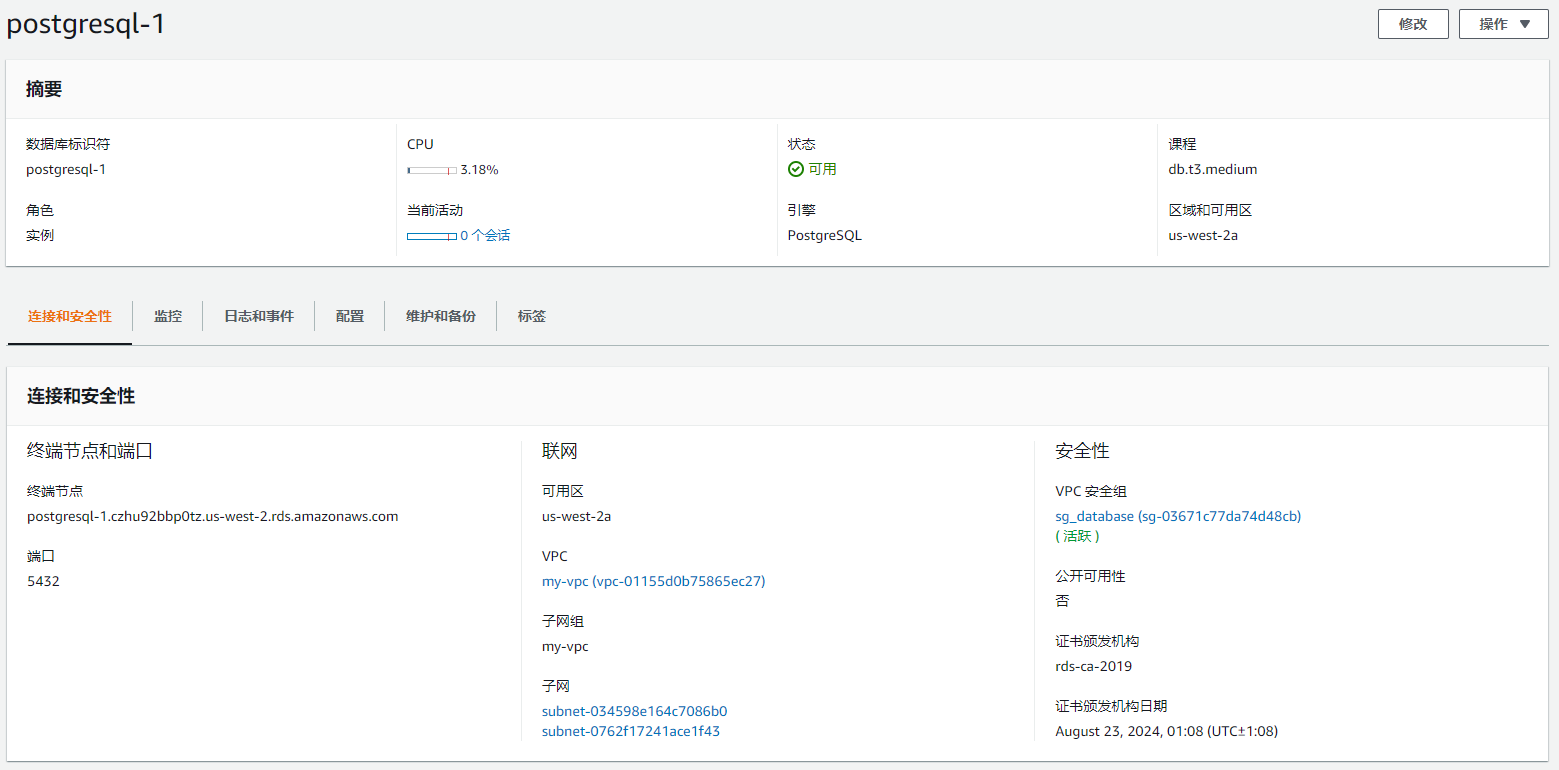
“sg\_server”：用于服务器，开放8080端口，源sg\_elb

“sg\_database”：用于数据库，开放5432端口，源sg\_server

****

**第三步：创建数据库**

创建一个postgre数据库，使用“sg\_database”安全组。根据WA架构“高可用”理念，应当开启replica备份。由于比赛时不限制数据库实例大小，但根据WA架构“经济”及“高可用”的理念，我们不应该无限制的选择超大数据库，因为性能强的数据库成本高昂且我们不一定会用到这么强的性能，但也不能使用最小型的数据库，否则可能在大流量来临时造成堵塞，所以选择一个中等大小的数据库较为适合。记下数据库host名，数据库用户名和密码，以及初始数据库名，这些参数要写入server.ini中。数据库创建成功后需要按题目要求创建数据表。



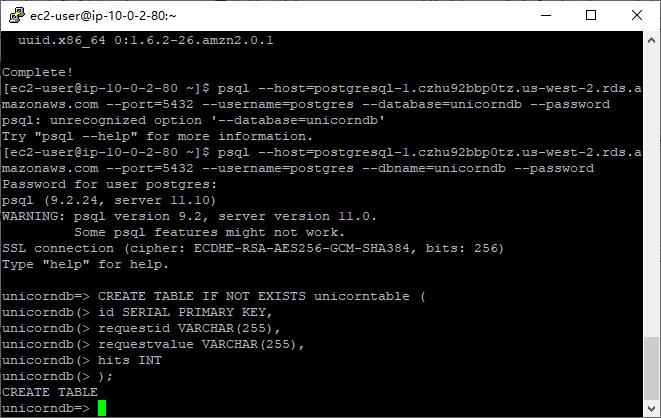
**第四步：创建服务器AMI**

在公网创建一台t2.micro服务器，开始制作服务器ami。

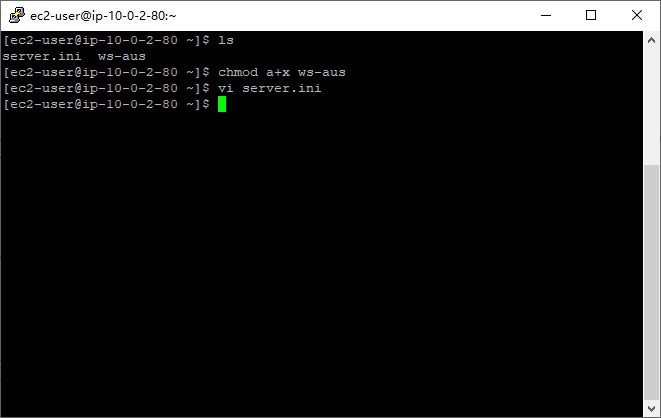
先连接数据库创建数据表。

安装psql命令：sudo yum install postgresql postgresql-server postgresql-devel postgresql-contrib postgresql-docs

连接psql命令：psql --host=<hostname> --port=5432 --username=<username> --password --dbname=<dbname>



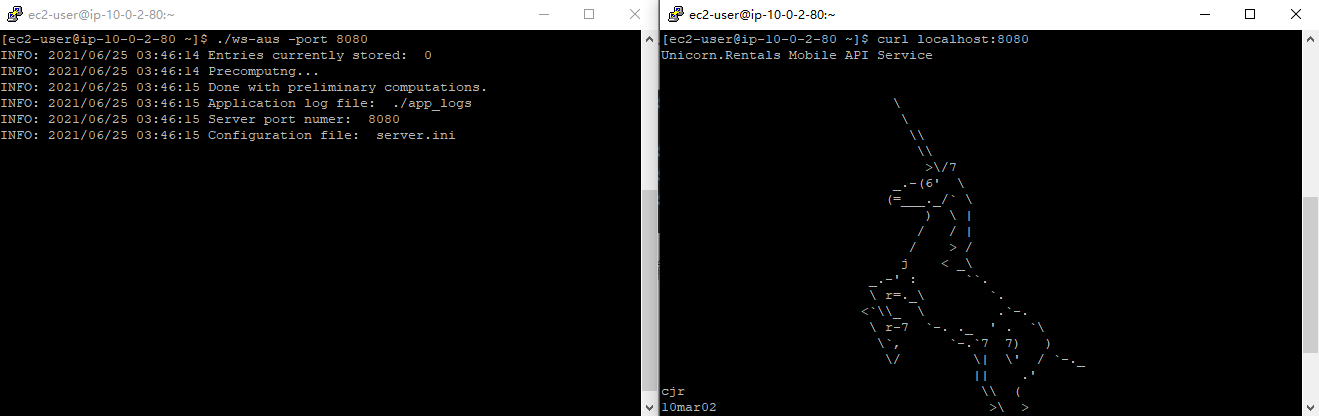
下载server.ini和Web应用程序。根据数据库参数修改server.ini，并给Web程序赋予可执行权限。



启动web服务器，试试能否启动成功。

启动命令:./ws-aus -port 8080

可以启动的话就可以先关闭服务了，接下来我们要设置自启动服务



我们需要写一个systemd的服务，命名为ws-aus.service

[Unit]

Description=gameday root service

After=network.target remote-fs.target

[Service]

Type=simple

User=ec2-user

WorkingDirectory=/home/ec2-user

ExecStart=/home/ec2-user/ws-aus -port 8080

ExecReload=/bin/kill -s -HUP $MAINPID

ExecStop=/bin/kill -s QUIT $MAINPID

KillMode=process

Restart=on-failure

RestartSec=42s

[Install]

WantedBy=multi-user.target

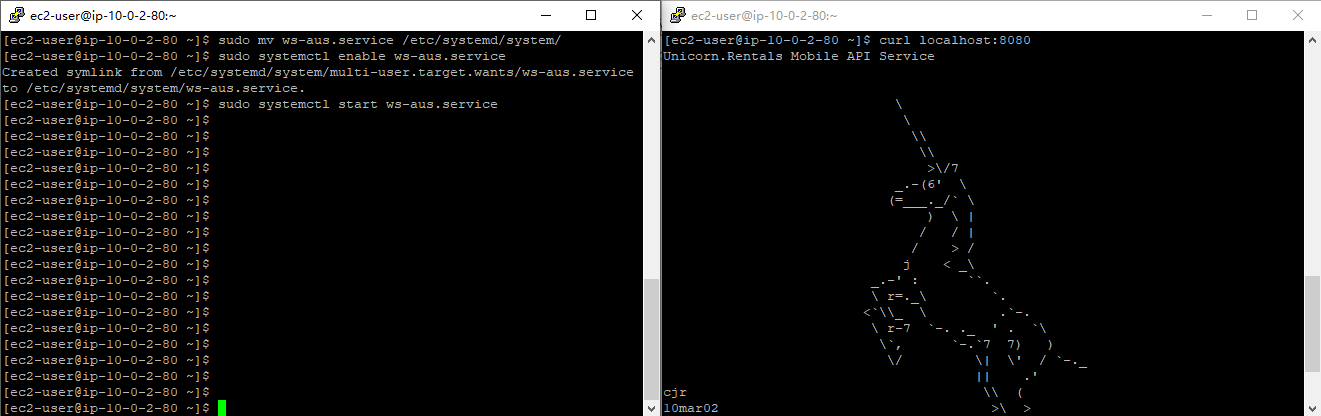
写完后将其放入/etc/systemd/system/

然后使得我们写完的服务建立符号链接关系

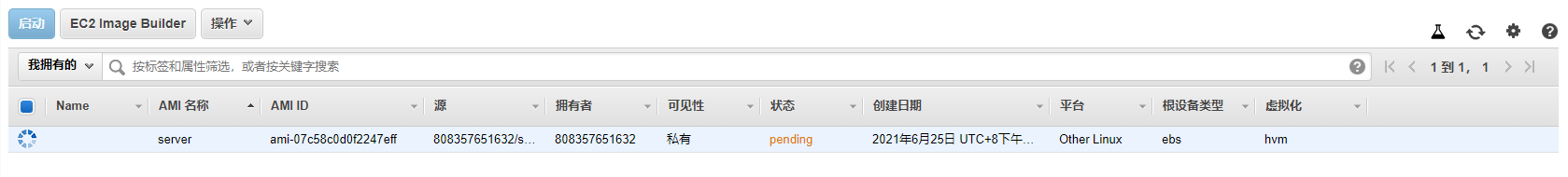
命令：sudo systemctl enable ws-aus.service

启动服务

命令：sudo systemctl start ws-aus.service

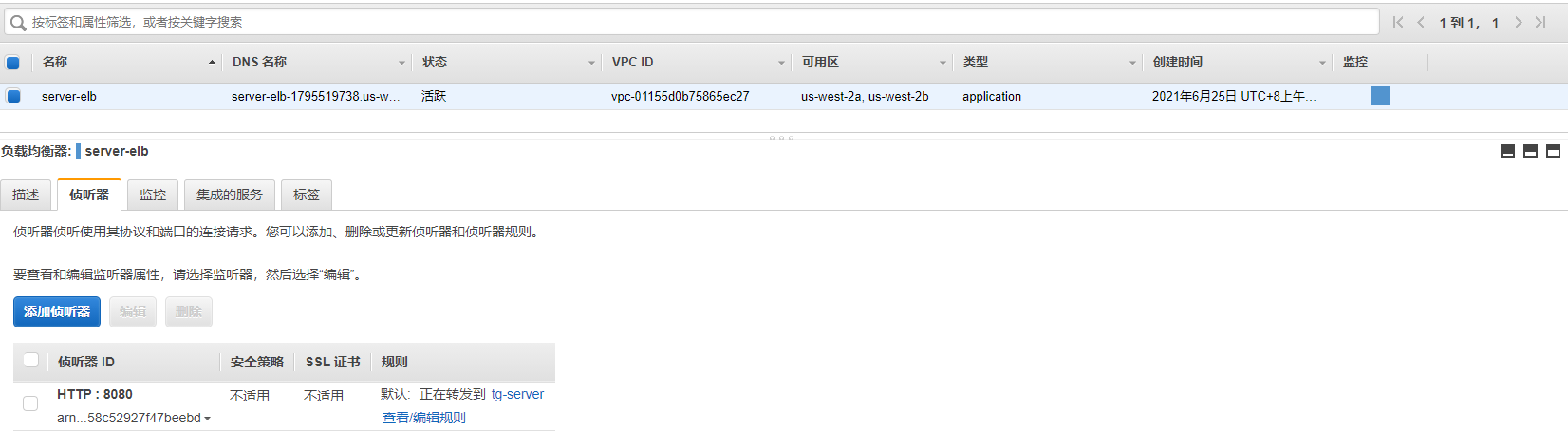


服务配置完成，去创建AMI



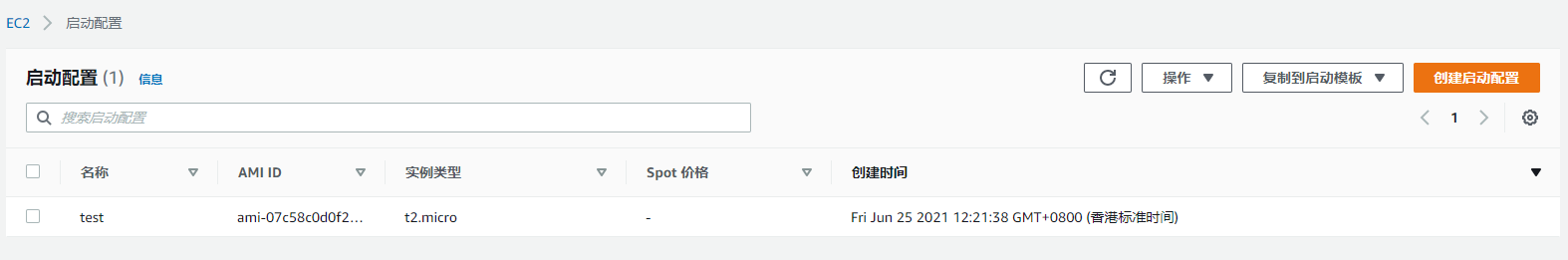
**第五步：创建负载均衡器**

创建目标群组，安全检查8080端口，创建应用负载均衡器，监听8080端口。



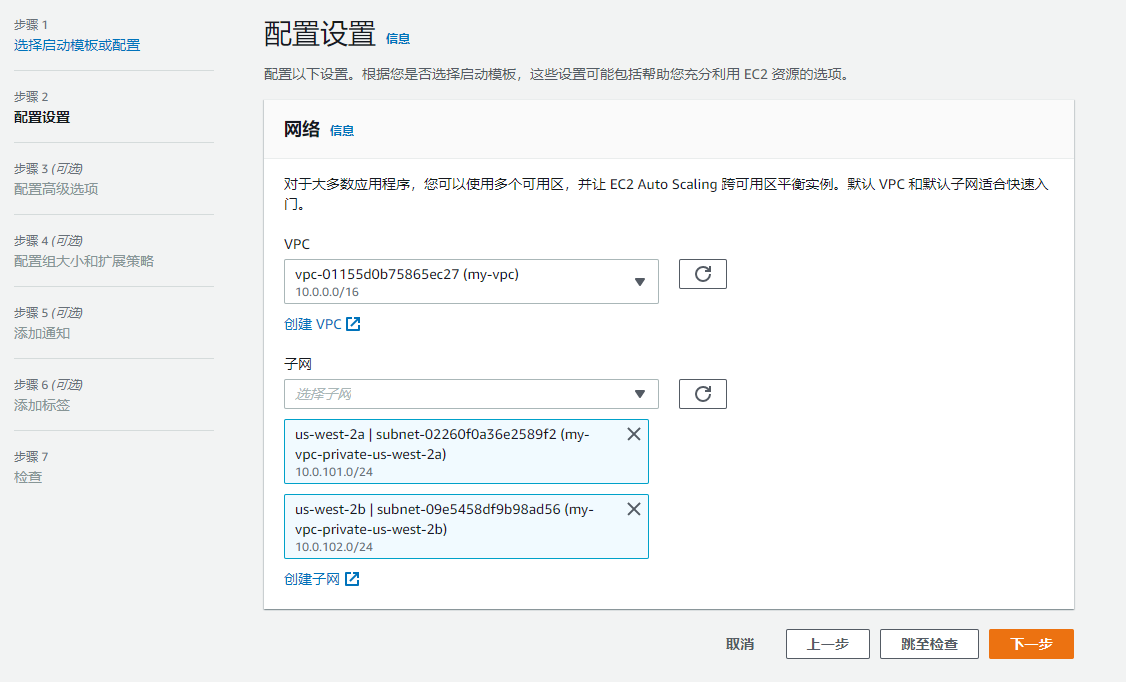
**第六步：创建Auto Scaling Group**

创建启动配置



创建ASG

选择私有子网

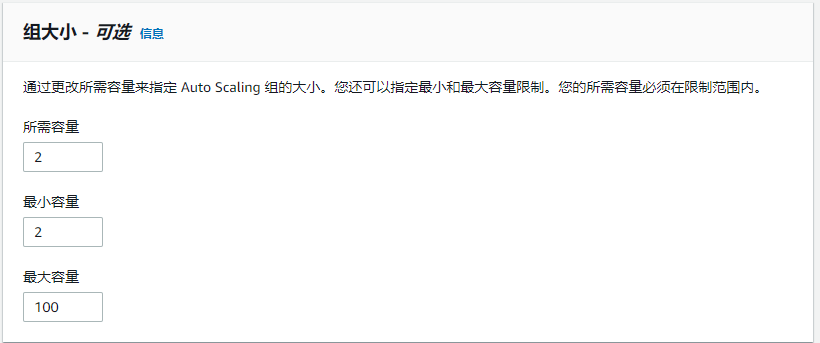


负载均衡参数如下图



配置组大小和扩展策略

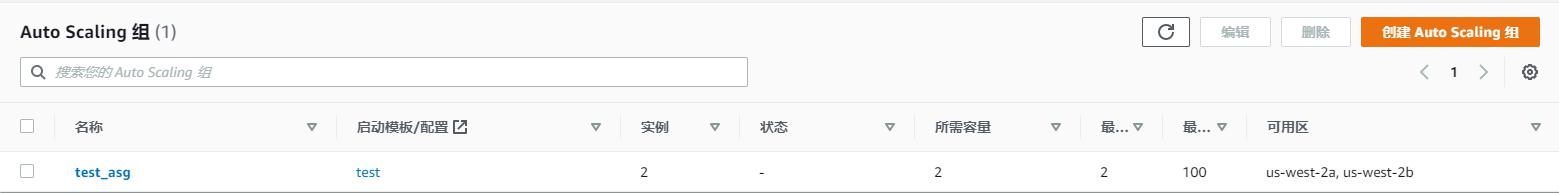
根据WA高可用的理念，组大小中最小容量应当大于一台，不免出现服务不可用的情况



扩展策略如图

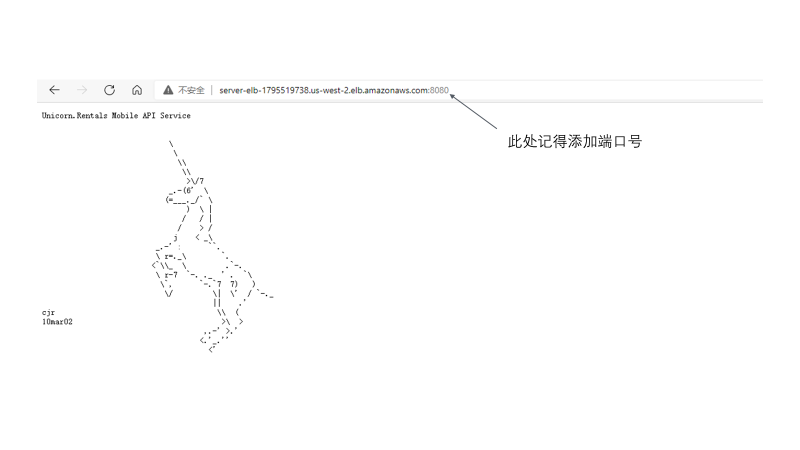


ASG创建完成



**第七步：验证并提交链接**

复制负载均衡器的DNS链接，在浏览器上粘贴，因为我们是在8080端口提供服务，所以后缀需要加上端口号。

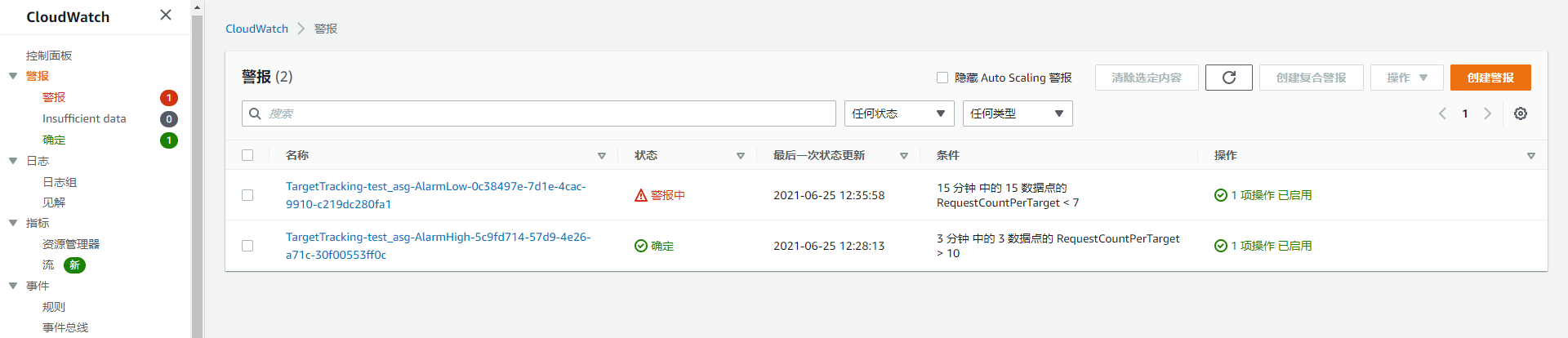


随后提交链接。

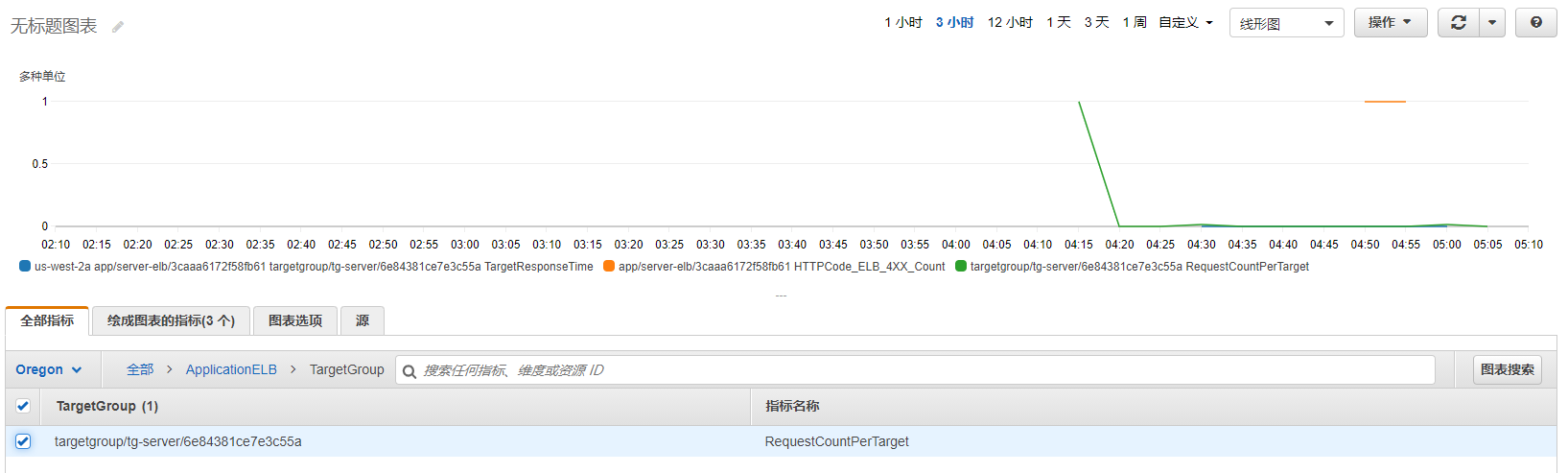
**第八步：监控及动态调整架构**

做完整个架构后我们并非无事可做，虽然架构现在已经拥有了自我调节的能力，但是比赛中流量是动态的，我们依旧需要监控流量并作出及时的调整，因为有时候系统的调整还是不如人为调整更迅速。

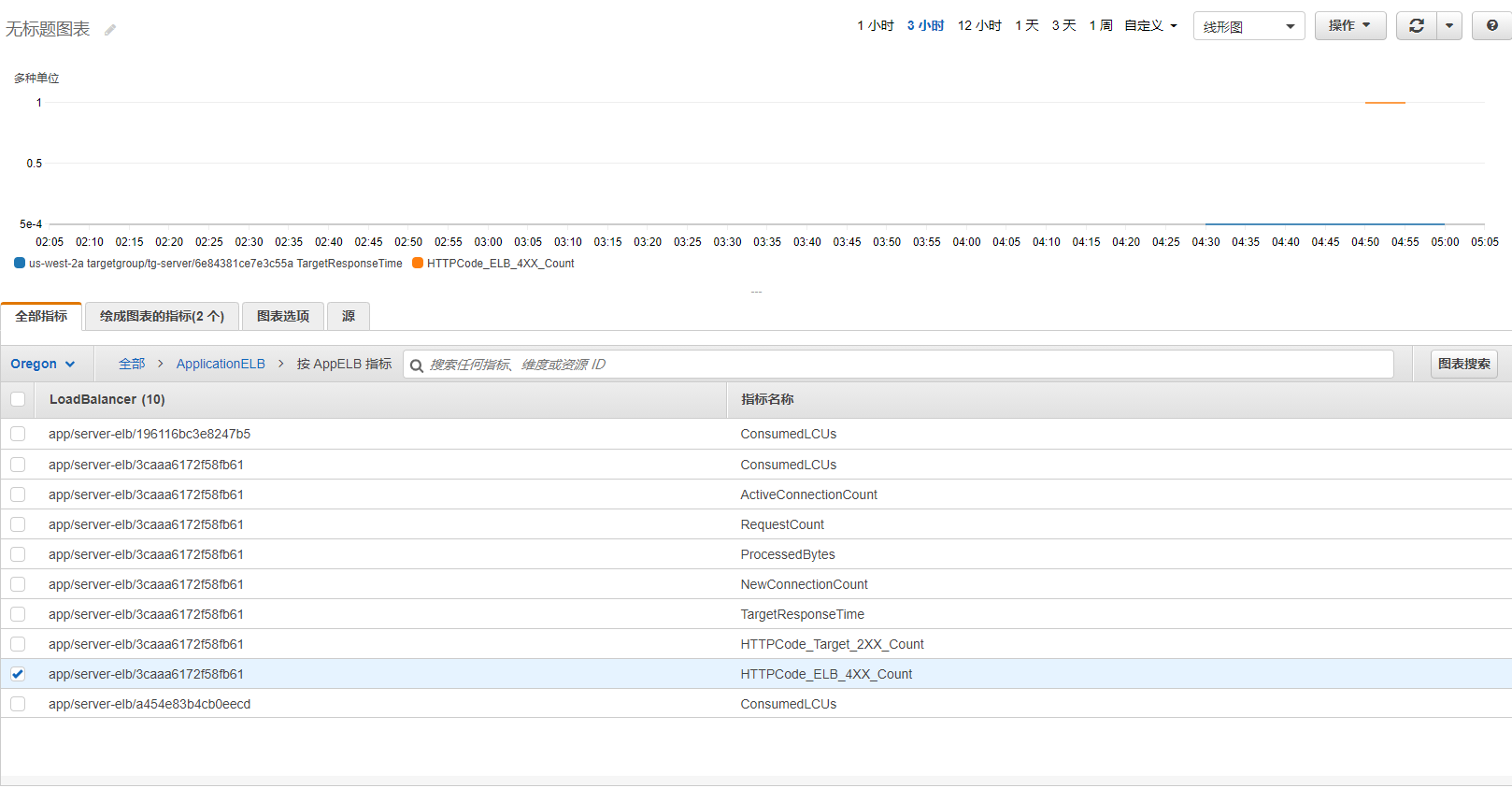
我们进入Cloudwatch面板,选择警报，可以看见ASG自动创建的两个警报，我们也可以自己创建更精确的警报。



点击指标，选择ApplicationELB > TargetGroup 便可以看见RequestCountPerTarget指标。这个指标体现了在目标组中的每台实例收到单位时间内的平均请求数



选择ApplicationELB > 按AppELB指标 便可以看见HTTPCode\_ELB\_4XX\_Count指标。这个指标体现了负载均衡器中的服务器收到了多少http4XX错误，如果比赛中有一段时间会出现403错误很有可能是服务器负载过高，那时就需要增加服务器数量。



同一界面下的RequestCount指标就是负载均衡器收到单位时间内的总请求数，可以最直观的体现总体流量变化。



如何手动编辑服务器数量呢？来到Auto Scaling Group的界面，点击“编辑”，修改所需容量，点击“更新”即可



**实战总结：**

本次实战涉及了基础架构设计，AWS组件的使用。考察了VPC的搭建，建立RDS数据库，AMI的制作以及自动化流程ASG的设置。关键需要注意的是初期需要根据WA（Well-Architected）框架来设计优化架构，中期注意“权限最小化”的安全组设置理念，EC2配置程序时如何更高效地配置自启动和错误重启服务，后期选择哪些更好的指标来监控服务的运行。