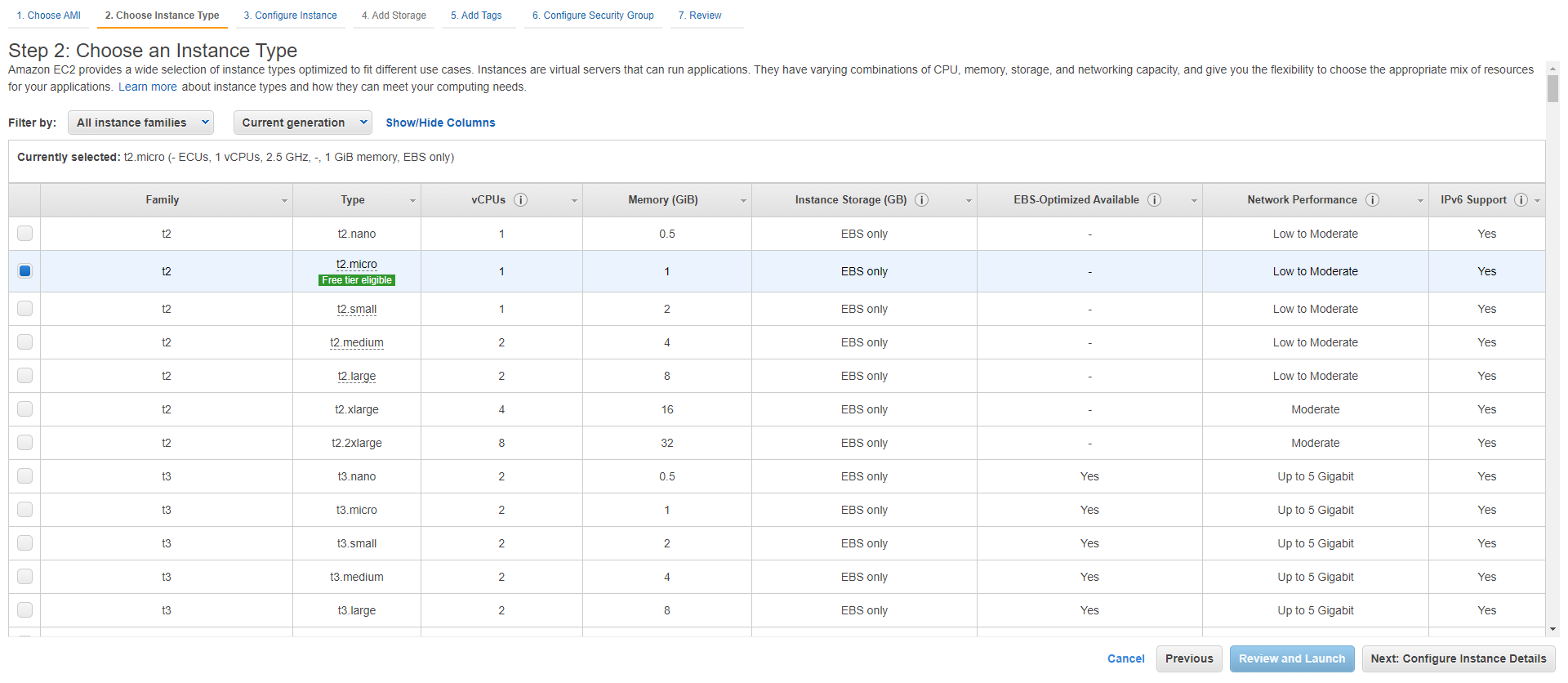
EC2

什么是EC2

Amazon Elastic Compute Cloud (Amazon EC2) 在 Amazon Web Services (AWS) 云中提供可扩展的计算容量。使用 Amazon EC2 可避免前期的硬件投入，因此您能够快速开发和部署应用程序。简单的理解就是虚拟服务器。在这章我们会对它的几个部分进行介绍。

实例类型（Instance Type）

Amazon EC2 提供多种经过优化，适用于不同使用案例的实例类型以供选择。实例类型由 CPU、内存、存储和网络容量组成不同的组合，可让您灵活地为您的应用程序选择适当的资源组合。每种实例类型都包括一种或多种实例大小，从而使您能够扩展资源以满足目标工作负载的要求。



目前实例被分为五种类型，分别是：

* 通用型

通用实例提供计算、内存和联网资源三方面的平衡，可用于各种不同的工作负载。这些实例非常适合于以相同比例使用这些资源的应用程序，如 Web 服务器和代码存储库。

* 计算优化型

计算优化型实例非常适用于从高性能处理器获取的受计算限制的应用程序。属于此系列的实例非常适用于批处理工作负载、媒体转码、高性能 Web 服务器、高性能计算 (HPC)、科学建模、专用游戏服务器和广告服务器引擎、机器学习推理和其他计算密集型应用程序。

* 内存优化型

内存优化型实例旨在提高可处理内存中大型数据集的工作负载的性能。

* 加速计算型

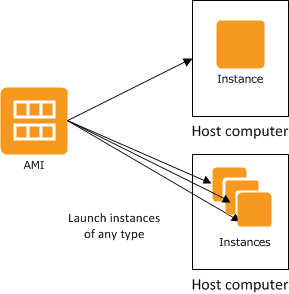
加速计算实例使用硬件加速器或协同处理器来执行浮点数计算、图形处理或数据模式匹配等功能，比使用在 CPU 上运行的软件更高效。

* 存储优化型

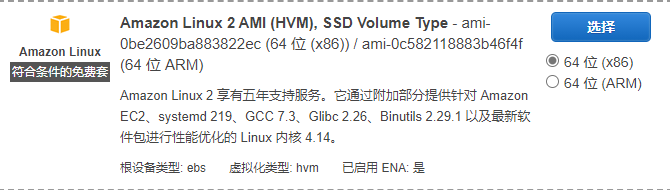
存储优化型实例旨在用于需要对本地存储上的大型数据集进行高速连续读写访问的工作负载。它们经过了优化，每秒可以向应用程序交付数以万计的低延迟、随机 I/O 操作 (IOPS)。

AMI

Amazon 系统映像 (AMI) 是一种包含软件配置 (例如，操作系统、应用程序服务器和应用程序) 的模板。可以简单地理解为系统镜像。你可以通过一个AMI启动多台配置相同的实例。



亚马逊官方推出过多款AMI，而我们实验和比赛经常使用的AMI是Amazon Linux 2 AMI，它的操作类似CentOS，且包含了大部分AWS所需的服务。



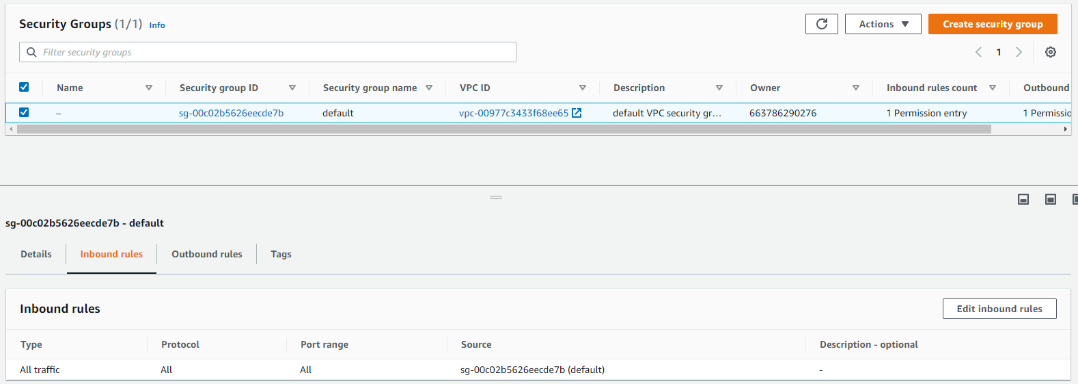
安全组(Security groups)

安全组充当 EC2 实例的虚拟防火墙，用于控制传入和传出流量。入站规则控制传入到实例的流量，出站规则控制从实例传出的流量。启动实例时，您可以指定一个或多个安全组。如果您未指定安全组，则 Amazon EC2 将使用默认安全组。您可以为每个安全组添加规则，规定流入或流出其关联实例的流量。您可以随时修改安全组的规则。新规则和修改后的规则将自动应用到与安全组相关联的所有实例。在 Amazon EC2 确定是否允许流量到达实例时，它评估与实例关联的所有安全组中的所有规则。

安全组的特性：

* 在安全组里没有任何规则时，安全组是默认禁止所有入站流量的
* 默认情况下，安全组允许所有出站流量。
* 安全组规则始终是宽松的；无法创建拒绝访问的规则。
* 安全组规则允许您根据协议和端口号筛选流量。
* 安全组是有状态的 — 如果您从实例发送一个请求，则无论入站安全组规则如何，都将允许该请求的响应流量流入。对于 VPC 安全组，这还意味着，无论出站规则如何，都允许对允许的入站流量的响应流出。有关更多信息，请参阅连接跟踪。
* 您可以随时添加和删除规则。您所做的更改将会自动应用到与安全组关联的实例中。

安全组界面



* Security group ID:安全组的ID，未来在API调用中会经常用到
* Security group name:安全组的名称，注意Name和Security group name并不一样，Security group name在第一次输入后便无法修改，未来在API调用时所使用的名字时是使用Security group name而非Name，Name的作用是方便管理者进行分类。
* VPC ID:安全组所属VPC的ID
* Inbound rules:入站规则
* Outbound rules:出站规则
* Type:AWS预设的协议和端口
* Protocol: 允许的协议。最常见的协议为 6 (TCP)、17 (UDP) 和 1 (ICMP)。
* Port range: 对于 TCP、UDP 或自定义协议，允许的端口范围。您可以指定单个端口号 (例如 22) 或端口号范围 (例如7000-8000)。
* Source: 流量的源 (入站规则) 或目标 (出站规则)。请指定以下选项之一：

1、一个单独的 IPv4 地址。您必须使用 /32 前缀长度；例如 203.0.113.1/32。

2、一个单独的 IPv6 地址。您必须使用 /128 前缀长度；例如 32001:db8:1234:1a00::123/128。

3、采用 CIDR 块表示法的 IPv4 地址范围；例如，203.0.113.0/24。

4、采用 CIDR 块表示法的 IPv6 地址范围；例如，2001:db8:1234:1a00::/64。

5、一个前缀列表 ID，例如 pl-1234abc1234abc123。

6、其他安全组。

安全组的工作方式

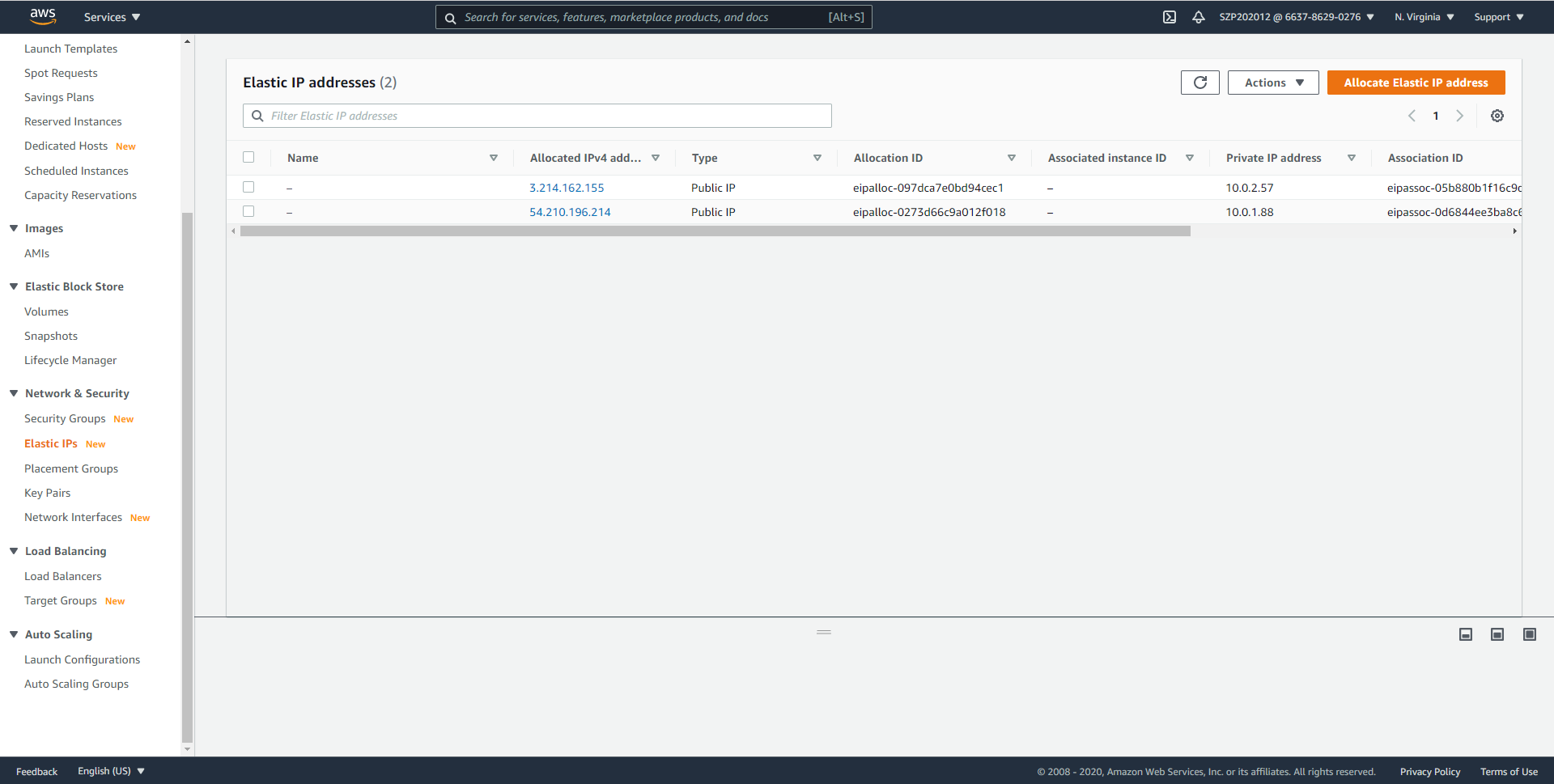
您的安全组使用连接跟踪来跟踪有关进出实例的流量的信息。将基于流量的连接状态应用规则以确定允许还是拒绝流量。这种方法允许安全组具有状态。这意味着无论出站安全组规则如何都允许对入站流量的响应流出实例，反之亦然。

并非所有通信流都会被跟踪。如果有一个安全组规则允许所有通信（0.0.0.0/0 或 ::/0）的 TCP 或 UDP 流，而另一个方向上存在一个对应的规则允许所有端口 (0-65535) 的所有响应通信（0.0.0.0/0 或 ::/0），则不会跟踪该通信流。因此，允许响应流量基于允许响应流量的入站或出站规则流动，而不是基于跟踪信息流动。

如果删除或修改了支持该流的规则，则会立即中断未被跟踪的通信流。例如，如果您有一个开放 (0.0.0.0/0) 出站规则，并且删除了允许所有 (0.0.0.0/0) 入站 SSH（TCP 端口 22）流量传输到实例的规则（或修改它以使不再允许连接），则您与实例的现有 SSH 连接将立即删除。该连接以前未跟踪，因此更改将断开连接。另一方面，如果您有一个较细化的入站规则，最初允许 SSH 连接（意味着已跟踪该连接），但将该规则更改为不再允许来自当前 SSH 客户端地址的新连接，则更改规则不会断开现有连接。也就是说之前已经被跟踪的连接不会因当前更改而断开

五、弹性IP(Elastic IP)

弹性 IP 地址 是专为动态云计算设计的静态 IPv4 地址。使用弹性 IP 地址，您可以快速将地址重新映射到您的账户中的另一个实例，从而屏蔽实例故障。弹性 IP 地址会分配给您的 AWS 账户，并且在您释放它之前一直属于您。弹性 IP 地址是公有 IPv4 地址，可通过 Internet 访问。如果您的实例没有公有 IPv4 地址，则可以将弹性 IP 地址与您的实例关联以启用与 Internet 的通信。例如，这允许您从本地计算机连接到您的实例。

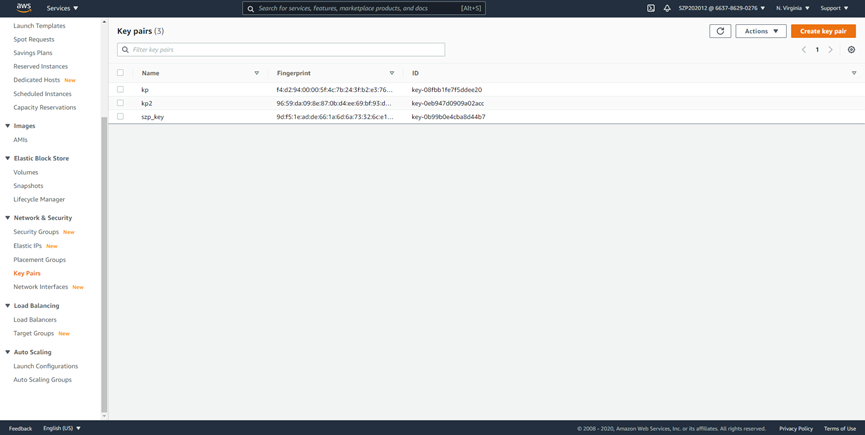


弹性IP的特征：

* 弹性 IP 地址是静态地址，不会随着时间的推移而改变。
* 当您将弹性 IP 地址与实例关联时，该地址也会与实例的主网络接口相关联。当您将弹性 IP 地址与连接到实例的网络接口关联时，它也与该实例关联。
* 当您将弹性 IP 地址与实例或其主网络接口关联时，实例的公有 IPv4 地址 (如果有) 将释放回 Amazon 的公有 IPv4 地址池中。您不能重用公有 IPv4 地址，并且不能将公有 IPv4 地址转换为弹性 IP 地址。
* 您可以取消弹性 IP 地址与资源的关联，然后重新将此地址与其他资源关联。为避免意外行为，请确保在进行更改之前关闭到现有关联中指定资源的所有活动连接。将弹性 IP 地址关联到其他资源后，可以重新打开与新关联的资源的连接。
* 为确保弹性 IP 地址的有效使用，如果弹性 IP 地址未与正在运行的实例关联，或者它已与停止的实例或未附加的网络接口关联，我们将强制收取小额的小时费用。当您的实例正在运行时，您无需为与该实例关联的某个弹性 IP 地址付费，但需为与该实例关联的所有其他弹性 IP 地址付费。
* 在将弹性 IP 地址与之前具有公有 IPv4 地址的实例关联时，该实例的公有 DNS 主机名将发生更改以匹配弹性 IP 地址。
* 我们会将公有 DNS 主机名解析为实例所在网络外部的该实例的公有 IPv4 地址或弹性 IP 地址，以及实例所在网络内部的该实例的私有 IPv4 地址。
* 弹性 IP 地址来自 Amazon 的 IPv4 地址池，或来自已引入到您的 AWS 账户的自定义 IP 地址池。
* 从已引入到您 AWS 账户的 IP 地址池分配弹性 IP 地址时，该地址不会计入弹性 IP 地址限制。有关更多信息，请参阅 弹性 IP 地址限制。
* 分配弹性 IP 地址时，您可以将弹性 IP 地址与网络边界组关联。这是我们公布 CIDR 块的位置。设置网络边界组会将 CIDR 块限制到此组。如果您未指定网络边界组，我们将设置包含区域中所有可用区的边界组（例如，us-west-2）。
* 弹性 IP 地址仅用于特定网络边界组。
* 一个弹性 IP 地址只能在一个特定区域中使用，不能移动到其他区域。

六、密钥对（Key pair）

密钥对（由私有密钥和公有密钥组成）是一组安全凭证，可在连接到实例时用来证明您的身份。Amazon EC2 存储公有密钥，您存储私有密钥。您可以使用私有密钥而不是密码来安全地访问您的实例。拥有您的私有密钥的任何人都可以连接到您的实例，因此将您的私有密钥存储在一个安全位置非常重要。



启动实例时，系统会提示您输入密钥对。如果您计划使用 SSH 连接到实例，则必须指定密钥对。您可以选择现有密钥对或创建新的密钥对。当您的实例首次启动时，您在启动时指定的公有密钥的内容将放置在 Linux 实例中的 ~/.ssh/authorized\_keys 的条目中。在使用 SSH 连接到 Linux 实例时，要进行登录，您必须指定与公有密钥内容对应的私有密钥。

由于 Amazon EC2 不保存私有密钥的副本，因此，如果您丢失私有密钥，则无法恢复它。

如果实例是使用EBS卷作为根存储，还是有方法恢复对丢失密钥的实例的访问，但是依然建议保存好密钥。

https://docs.aws.amazon.com/zh\_cn/AWSEC2/latest/UserGuide/replacing-lost-key-pair.html

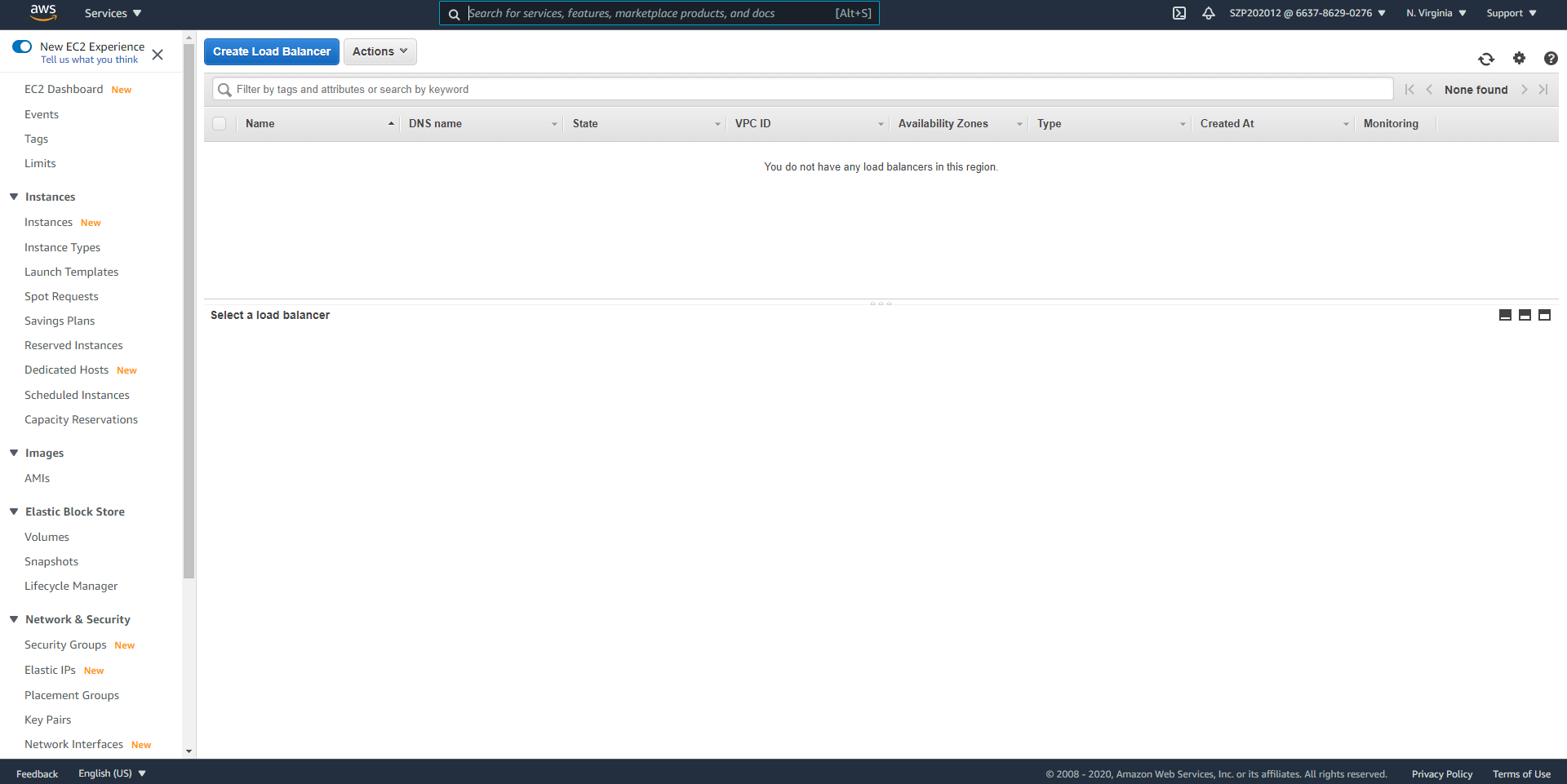
七、负载均衡器（Load Balancing）

Elastic Load Balancing 跨多个可用区中的多个目标（如 Amazon EC2 实例、容器和 IP 地址）分发传入应用程序或网络流量。Elastic Load Balancing 会在应用程序的传入流量随时间的推移发生更改时扩展负载均衡器。它可以自动扩展来处理绝大部分工作负载。


                基本 应用程序负载均衡器 的组成部分
            

负载均衡器的种类：

* Application Load Balancer
* Network Load Balancer
* Classic Load Balancer



三种负载均衡器的区别：

ALB作用于应用层，NLB作用于网络层，CLB是一种旧的负载均衡器，一般已不再使用。在没有特别提及的情况下，我们一般使用ALB。

负载均衡器两种常见的组件：

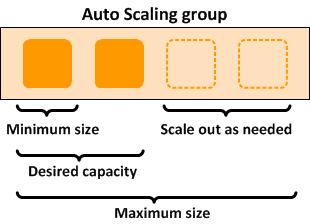
侦听器：侦听器是一个使用您配置的协议和端口检查连接请求的进程。您为侦听器定义的规则确定负载均衡器如何将请求路由到其已注册目标。

目标组：每个目标组均用于将请求路由到一个或多个已注册的目标。在创建每个侦听器规则时，可以指定目标组和条件。满足规则条件时，流量会转发到相应的目标组。您可以为不同类型的请求创建不同的目标组。

八、Auto Scaling

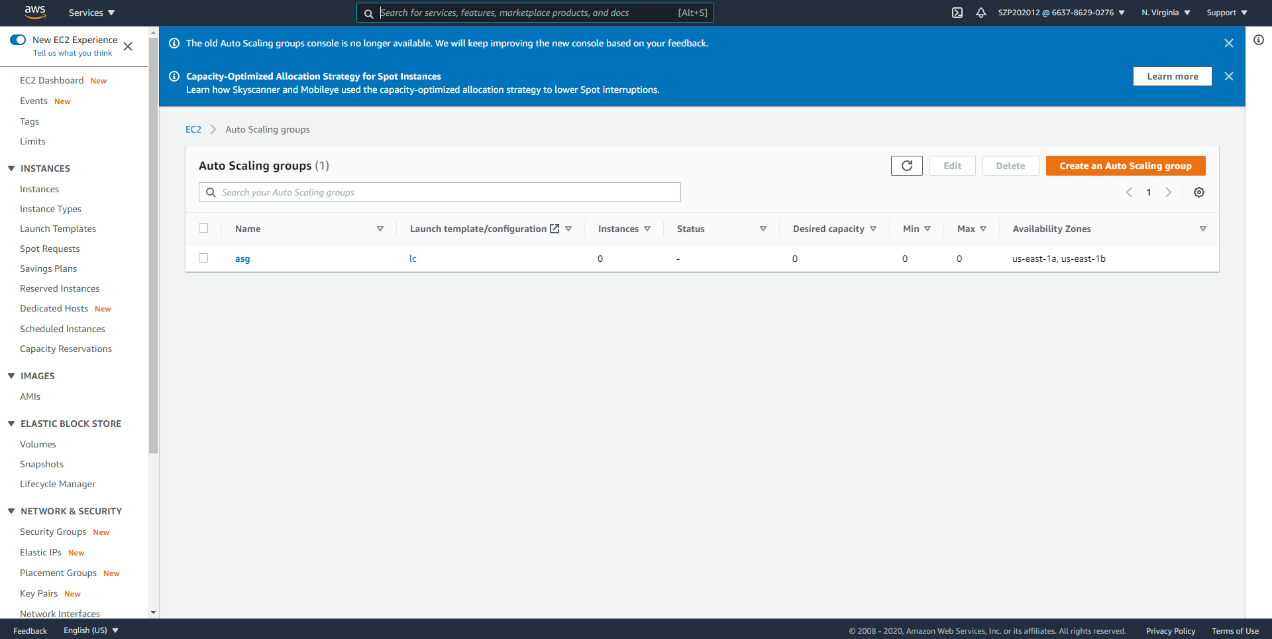
Amazon EC2 Auto Scaling 助您确保拥有适量的 Amazon EC2 实例，用于处理您的应用程序负载。您可创建 EC2 实例的集合，称为 Auto Scaling 组 。您可以指定每个 Auto Scaling 组中最少的实例数量，Amazon EC2 Auto Scaling 会确保您的组中的实例永远不会低于这个数量。您可以指定每个 Auto Scaling 组中最大的实例数量，Amazon EC2 Auto Scaling 会确保您的组中的实例永远不会高于这个数量。如果您在创建组的时候或在创建组之后的任何时候指定了所需容量，Amazon EC2 Auto Scaling 会确保您的组一直具有此数量的实例。如果您指定了扩展策略，则 Amazon EC2 Auto Scaling 可以在您的应用程序的需求增加或降低时启动或终止实例。

例如，以下 Auto Scaling 组的最小大小为 1 个实例，所需的容量为 2 个实例，最大大小为 4 个实例。您制定的扩展策略是按照您指定的条件，在最大最小实例数范围内调整实例的数量。



启动配置

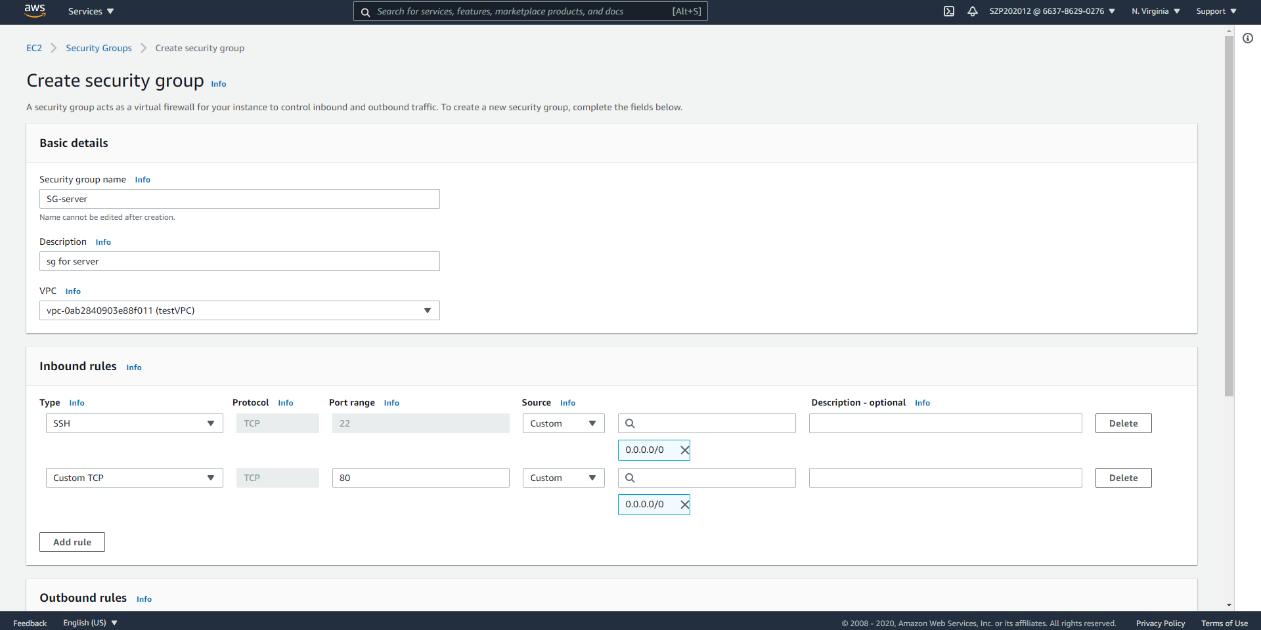
启动配置是 Auto Scaling 组用于启动 EC2 实例的实例配置模板。在创建启动配置时，您需要指定实例的信息。包括 Amazon 系统映像 (AMI) ID、实例类型、密钥对、一个或多个安全组以及块储存设备映射。如果之前已启动过 EC2 实例，可以指定相同的信息来启动实例。



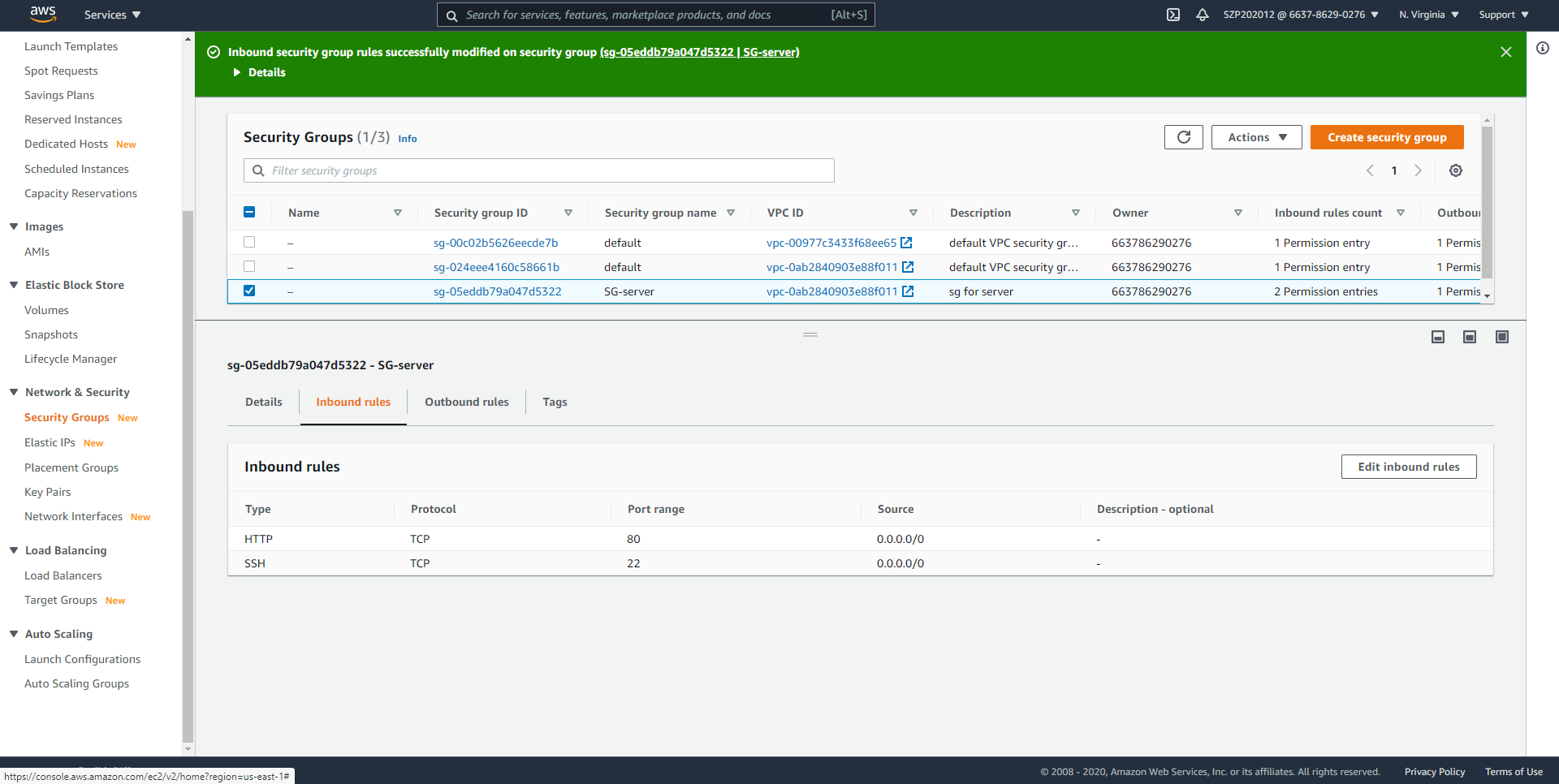
九、启动一台EC2

1.确认自己有一个能正常使用的VPC

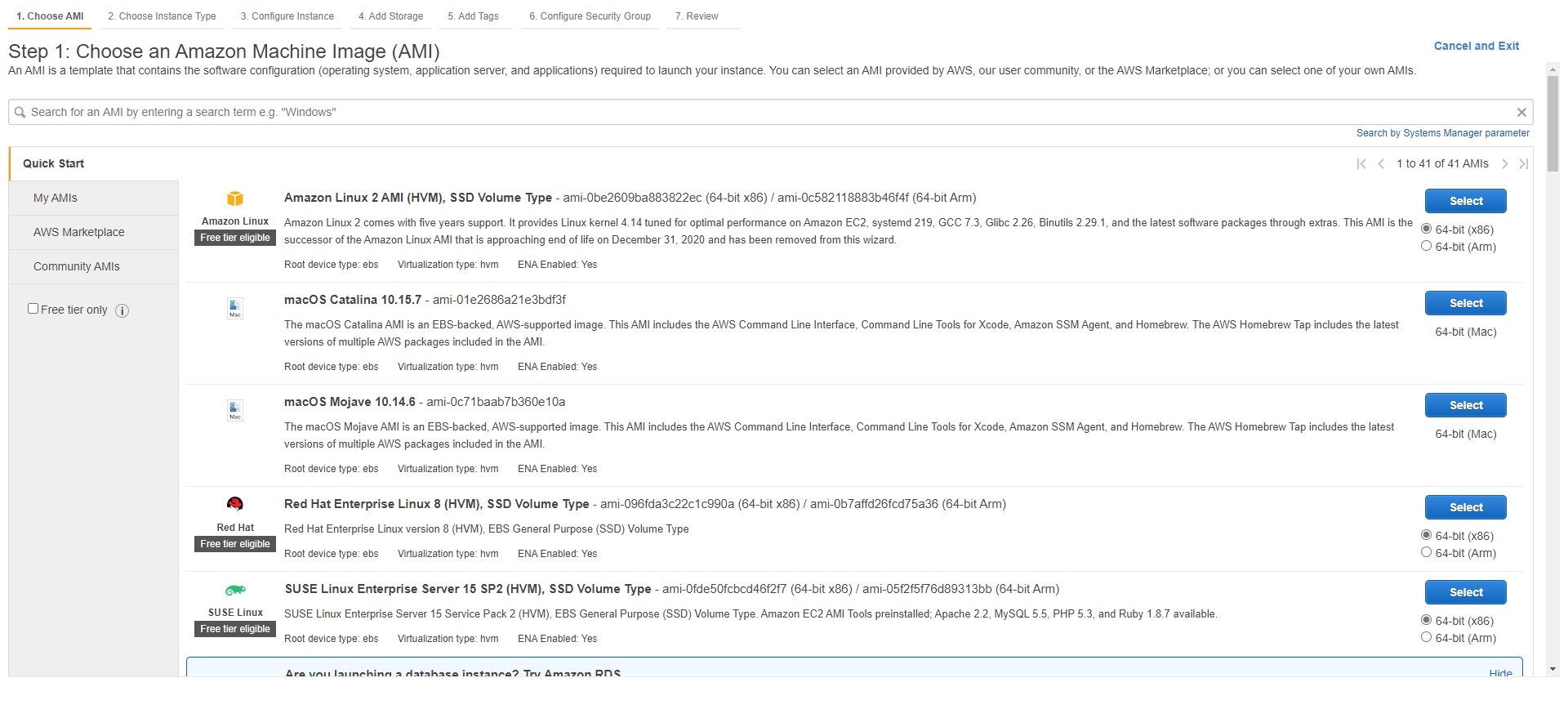
2.创建一个安全组，请规范命名，填写描述，注意选择自己使用的VPC，入站规则开放22和80端口，出站规则保持默认。22端口用于SSH访问，80端口用于未来访问EC2上的服务



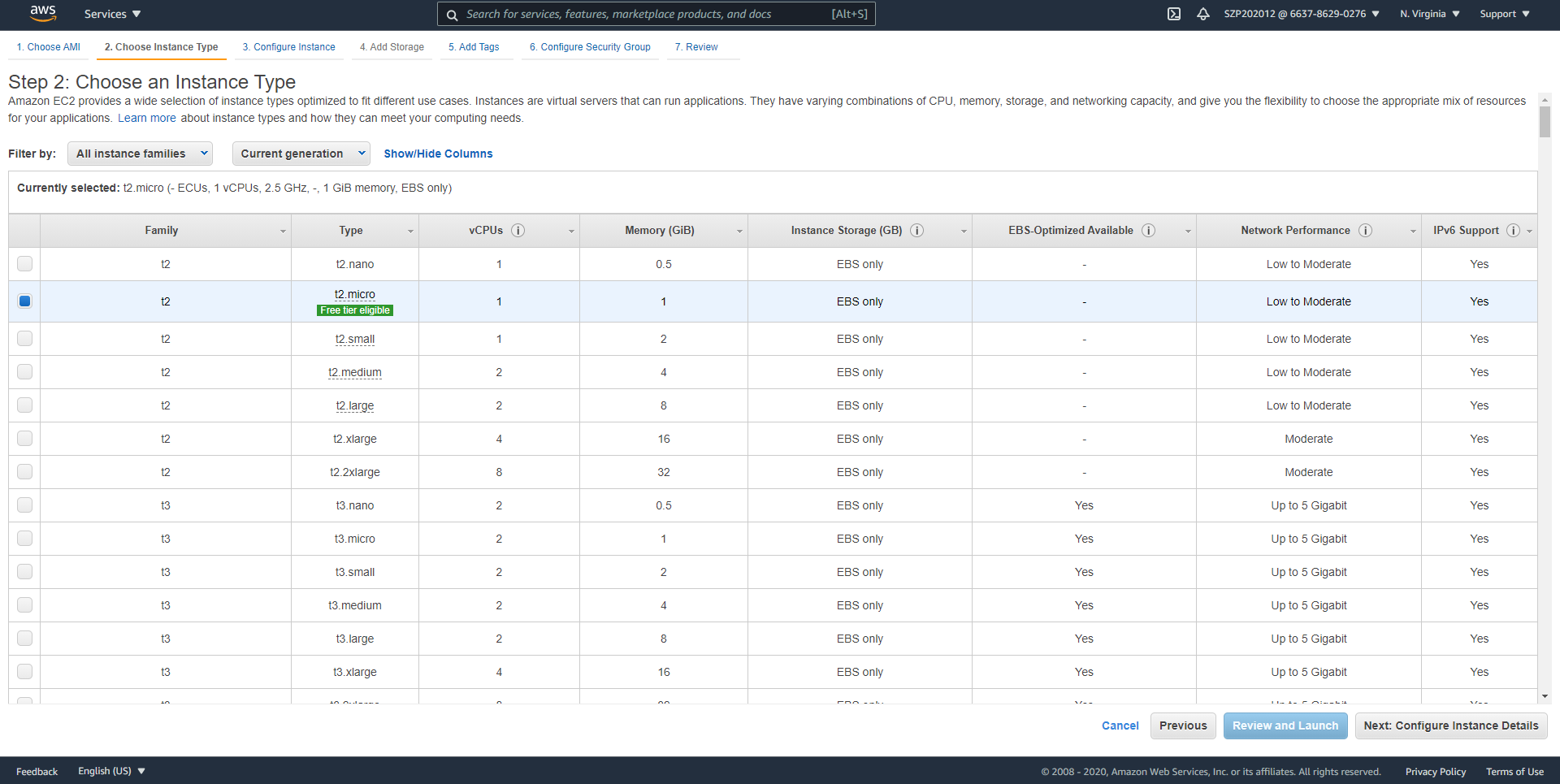
创建完成后



3.控制栏选择Instances，点击Launch Instances，进入选择AMI的界面选择Amazon Linux 2 AMI



4.选择实例类型，因为是实验，所以选择t2.micro

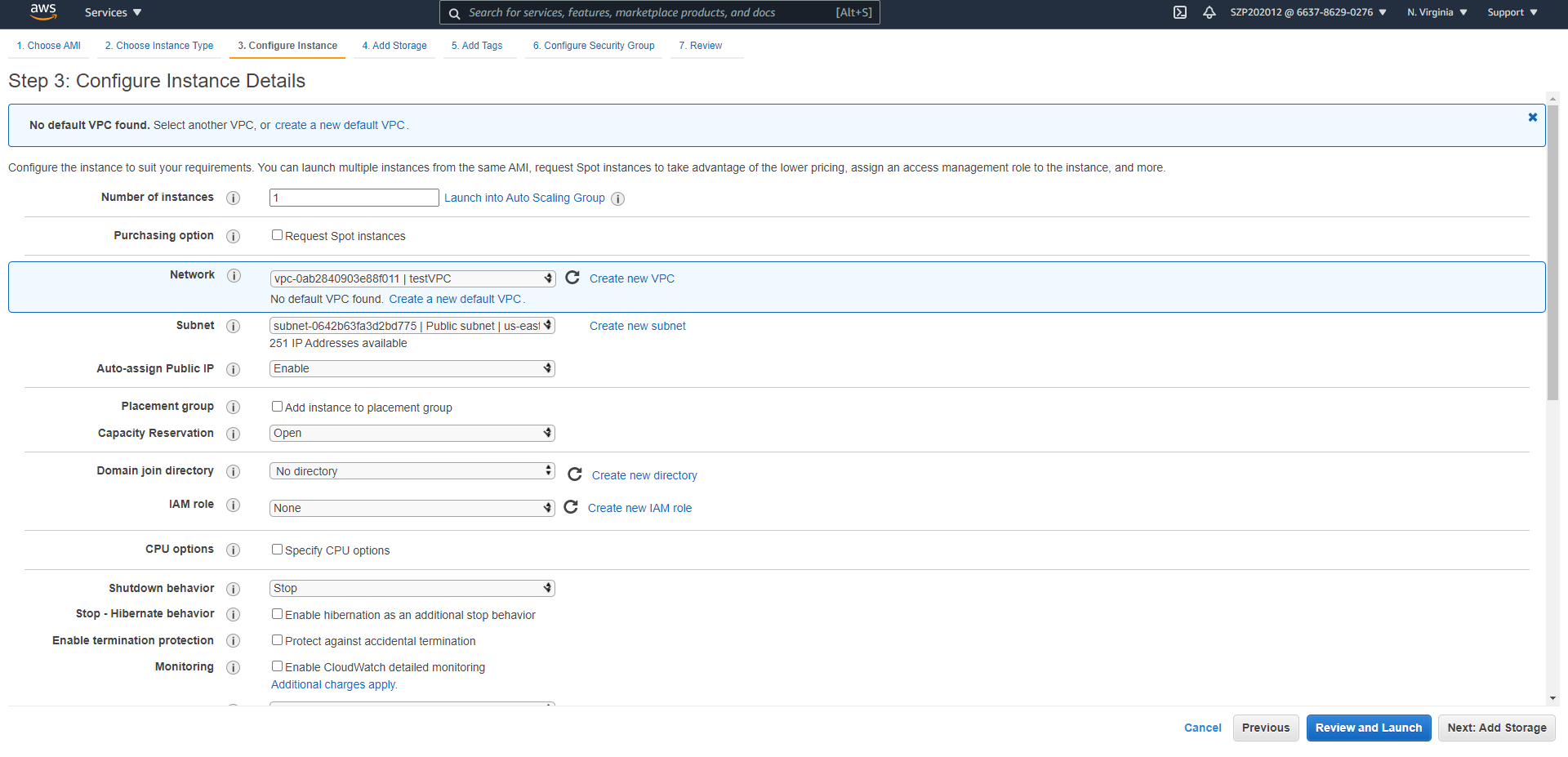


5.配置实例细节

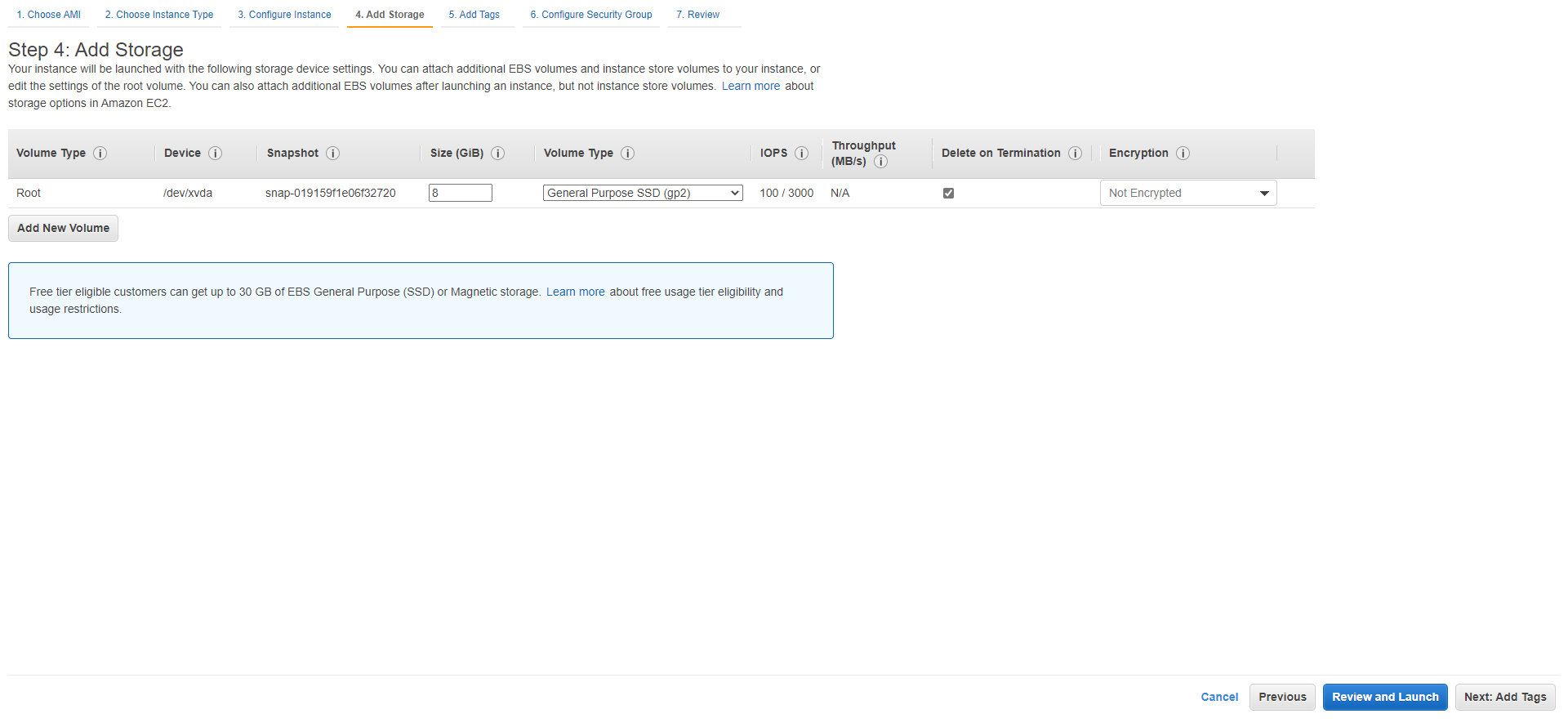
Network：选择使用的VPC

Subnet：选择一个共有子网

Auto-assign Public IP：自动绑定公有IP，选择enable（可选）



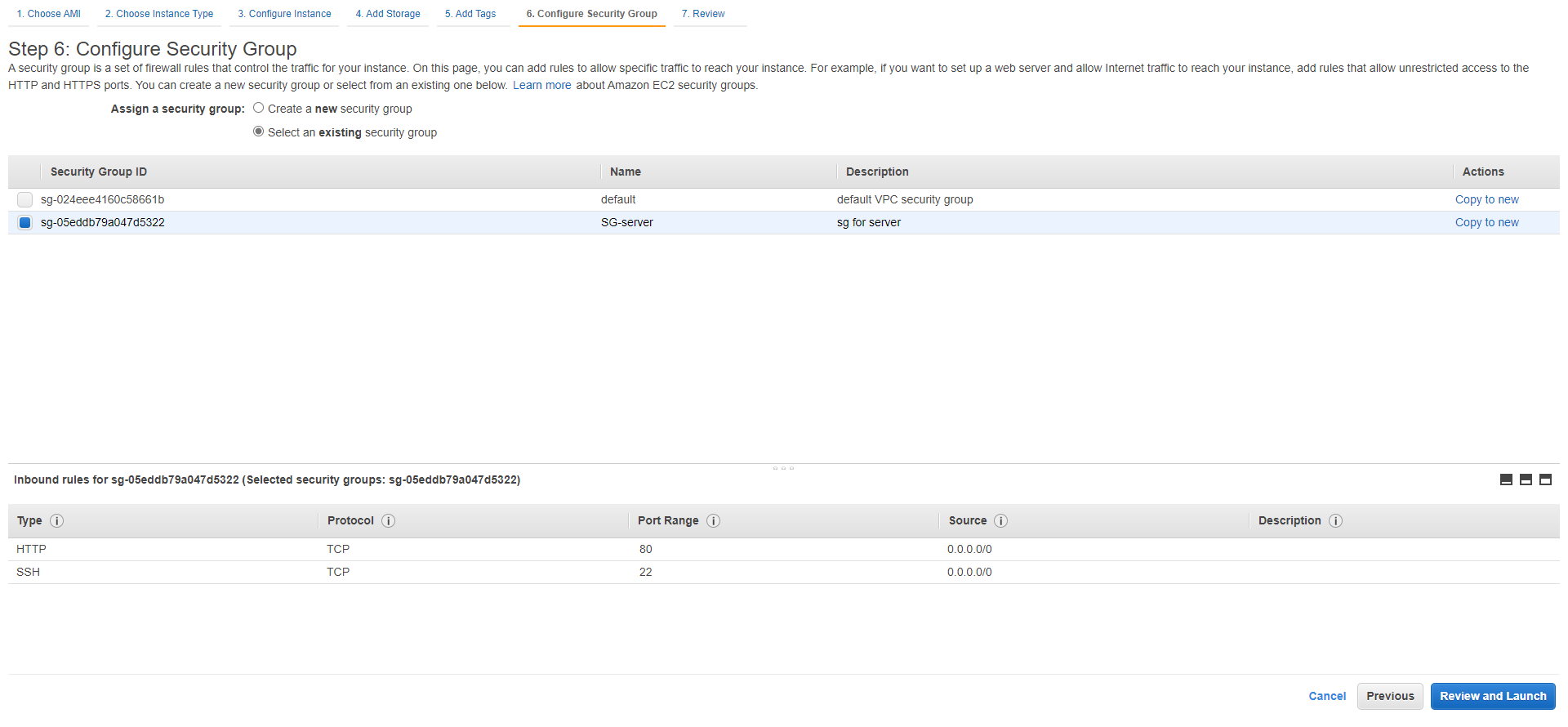
6.选择存储，即选择虚拟服务器挂载的块存储大小，保持默认即可



7.添加标签，可以不添加，也可以添加Name标签用来标记实例名字



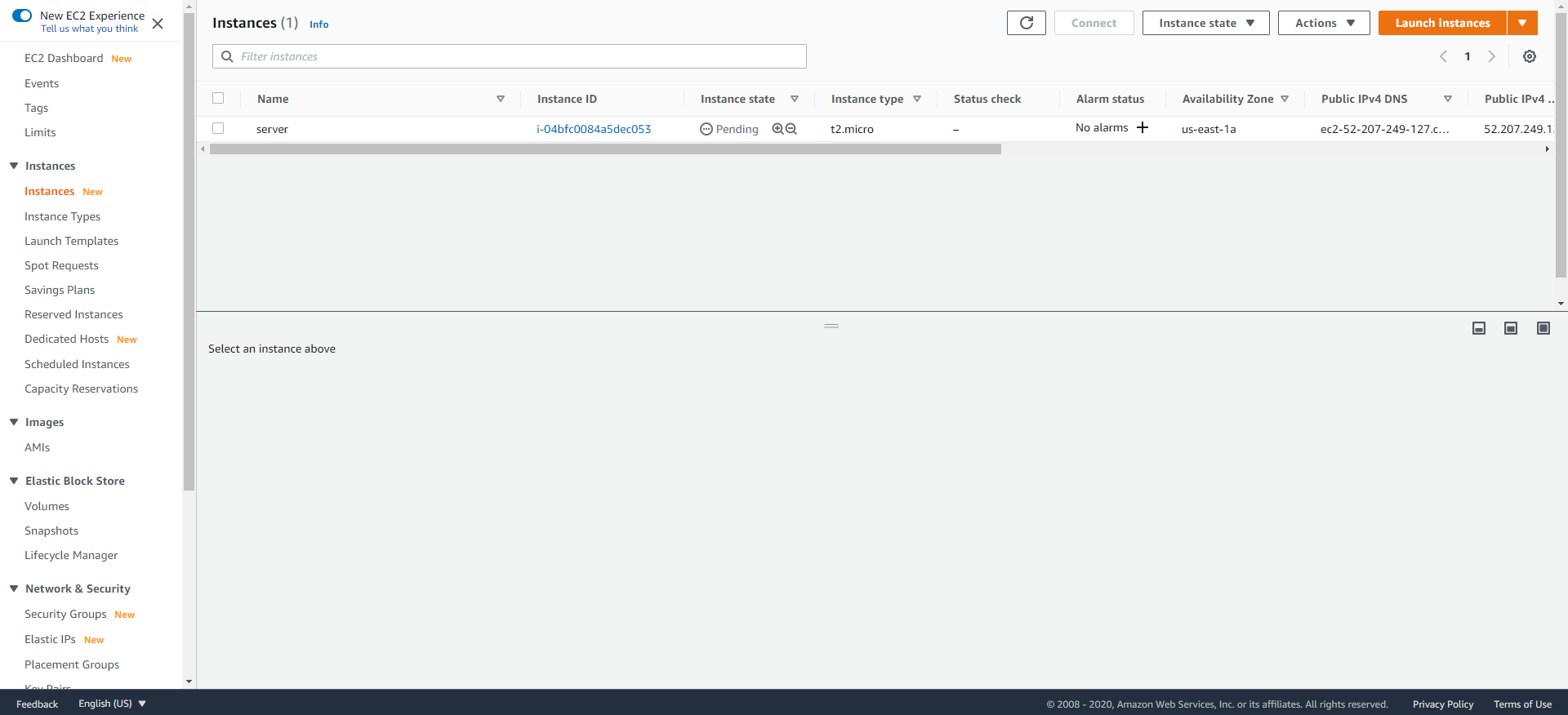
8.选择安全组，选择Select an existing security group，选择之前创建的安全组



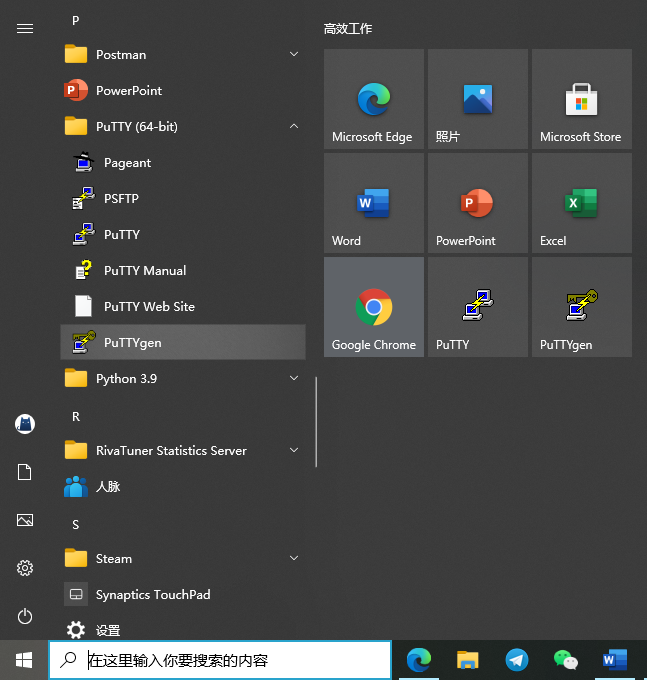
9.概览，确认各项配置无误后选择launch，会跳出先择密钥对，选择Create a new key pair，输入密钥对名字后点击Download Key Pair，然后点击Launch Instances，启动实例



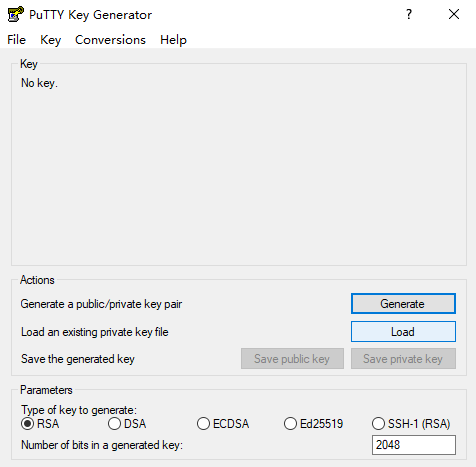
10.返回主页，发现有一台实例进入Pending状态



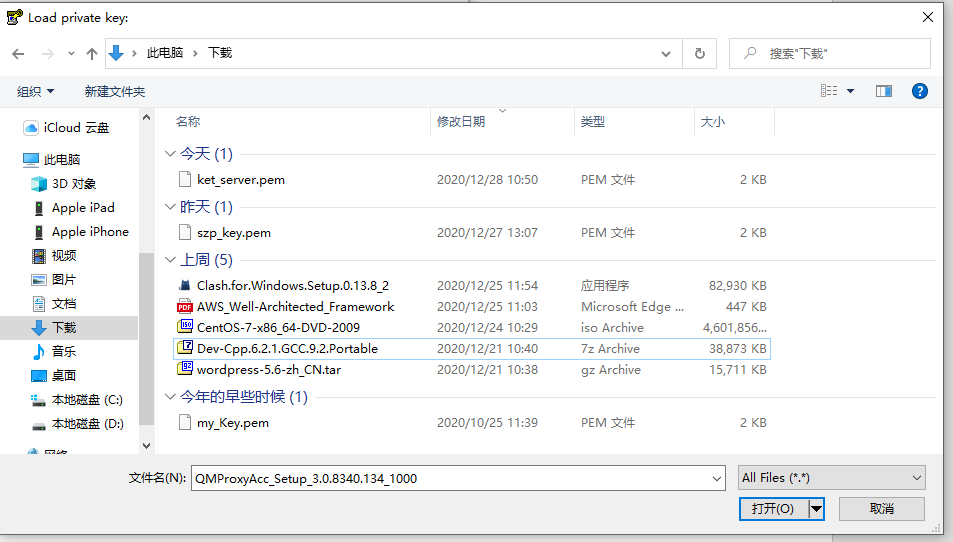
11.在自己的电脑上打开Puttygen



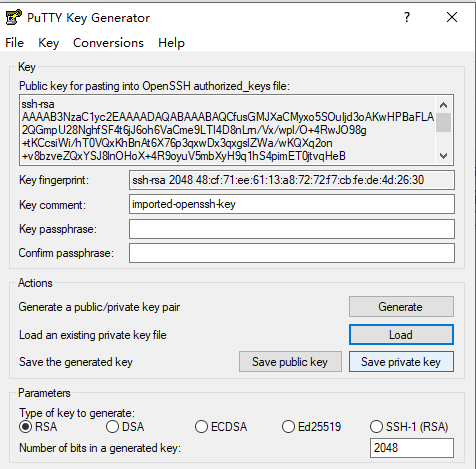
点击Load



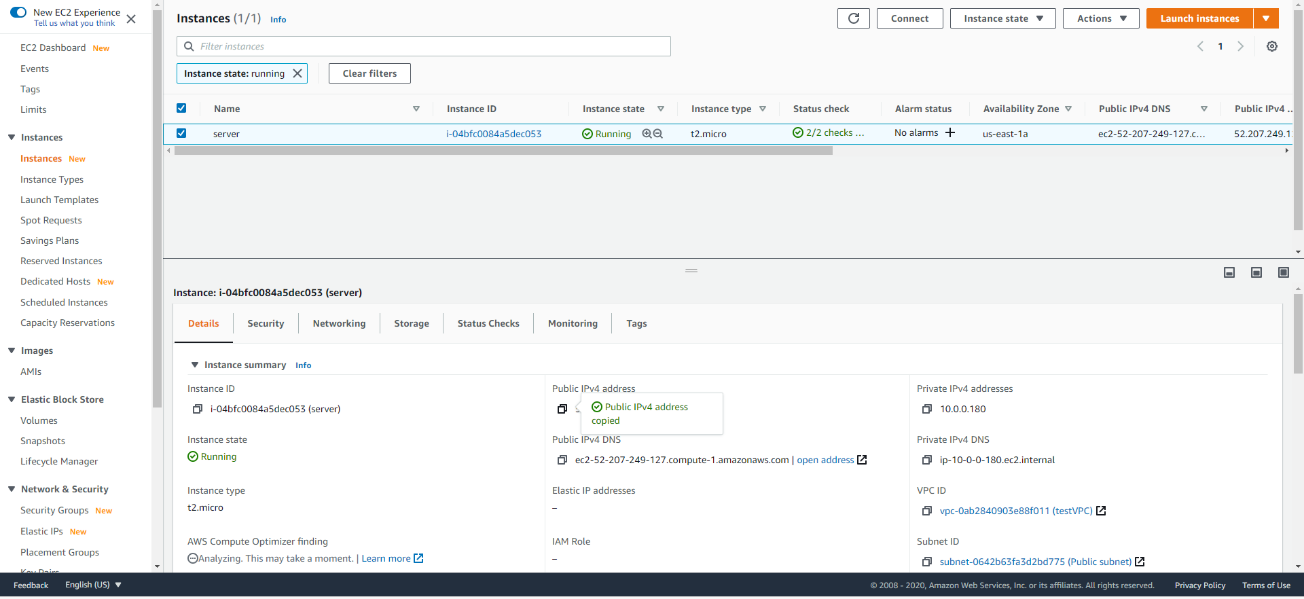
点击All files，会发现刚刚下载的密钥，点击打开



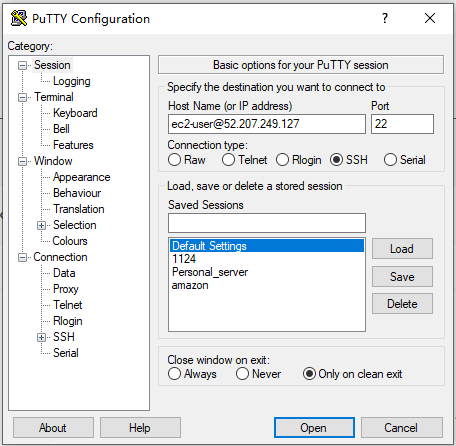
点击Save private key选择一个保存位置，密钥转换成功



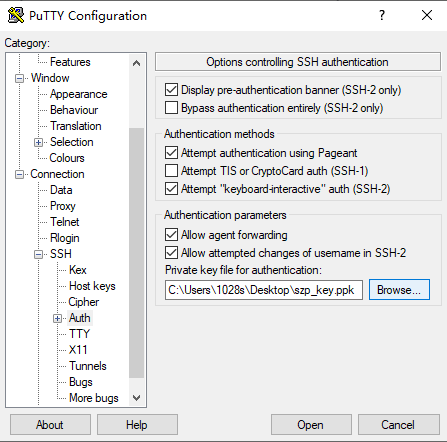
12.打开AWS控制台，并复制实例的公有IP



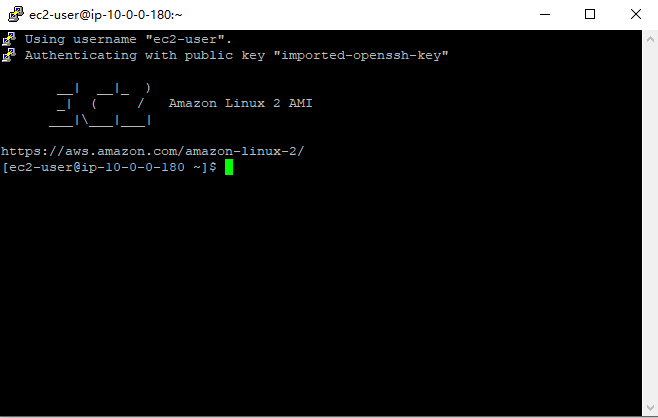
13.打开Putty，Host name输入“ec2-user@实例公有IP”



点击Connection->SSH->Auth，点击Browse，选择自己的密钥，然后点击open



实例启动成功，恭喜！



14.让我们启动一个apache服务来体现实例的工作吧

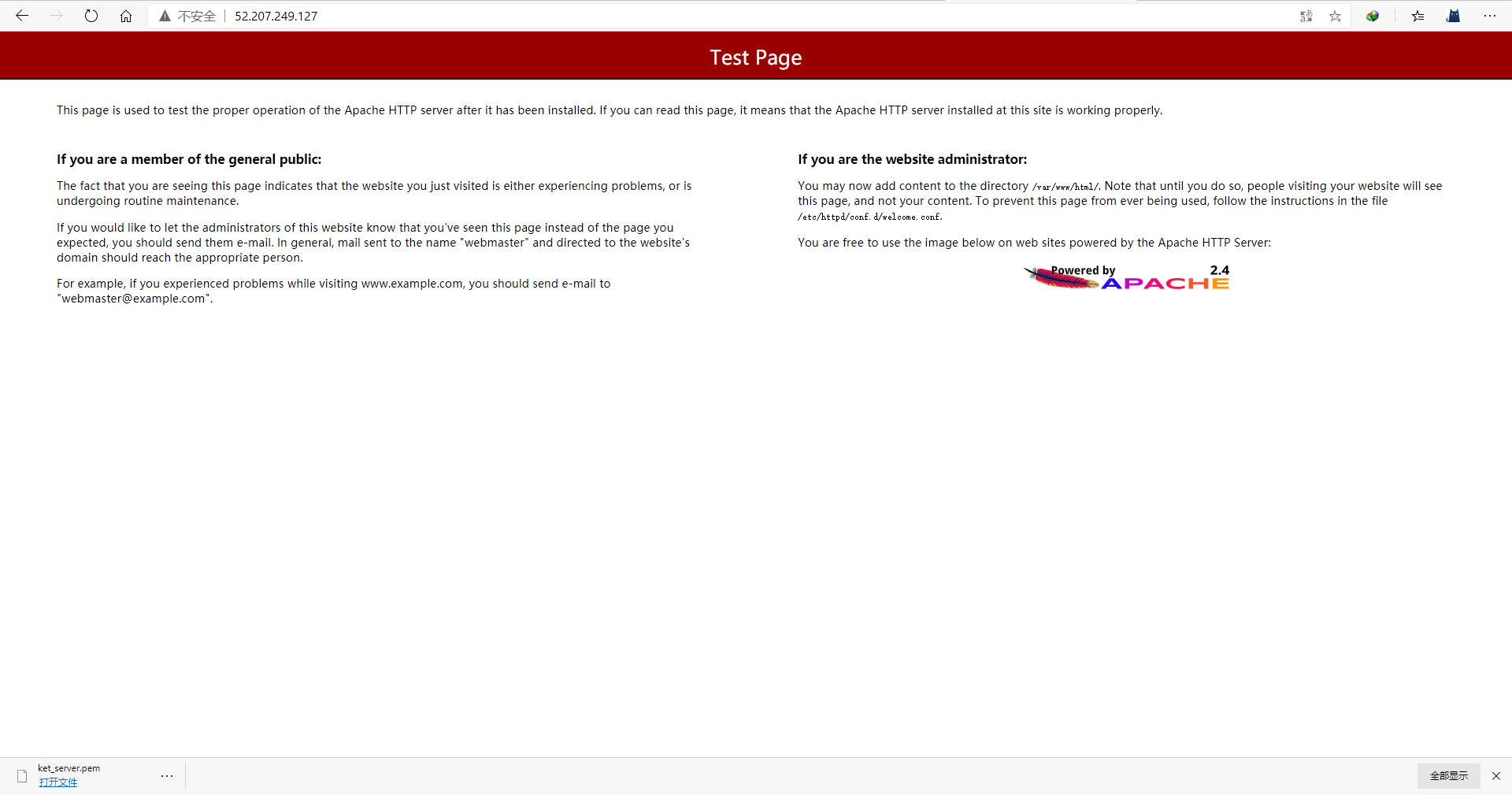
输入sudo -i进入root用户，输入yum install -y httpd



安装成功后输入service httpd start



复制实例公有IP粘贴在浏览器并访问，出现测试页面，这说明你的实例已经启动了一个简单的apache服务

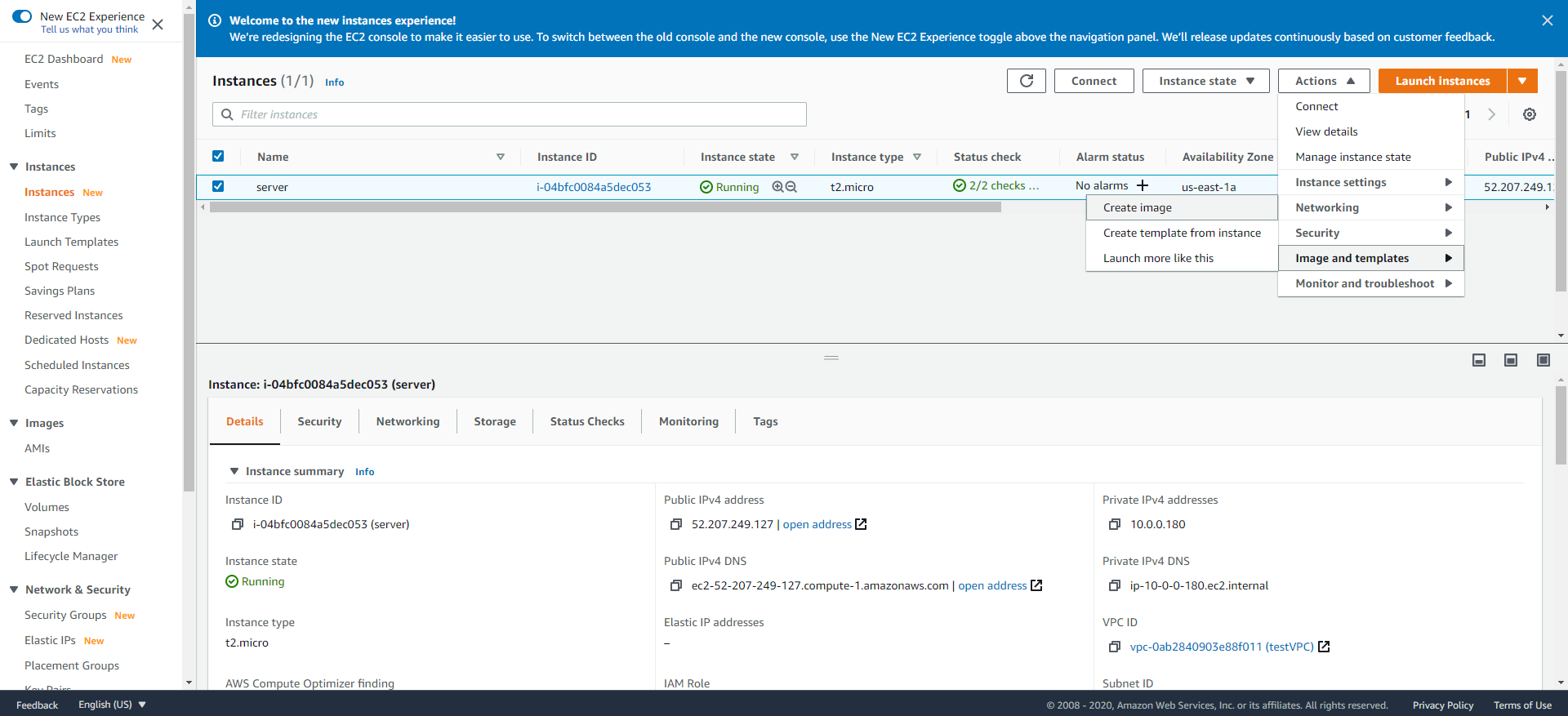


十、启动一个带自动拓展和负载均衡能力的实例集群

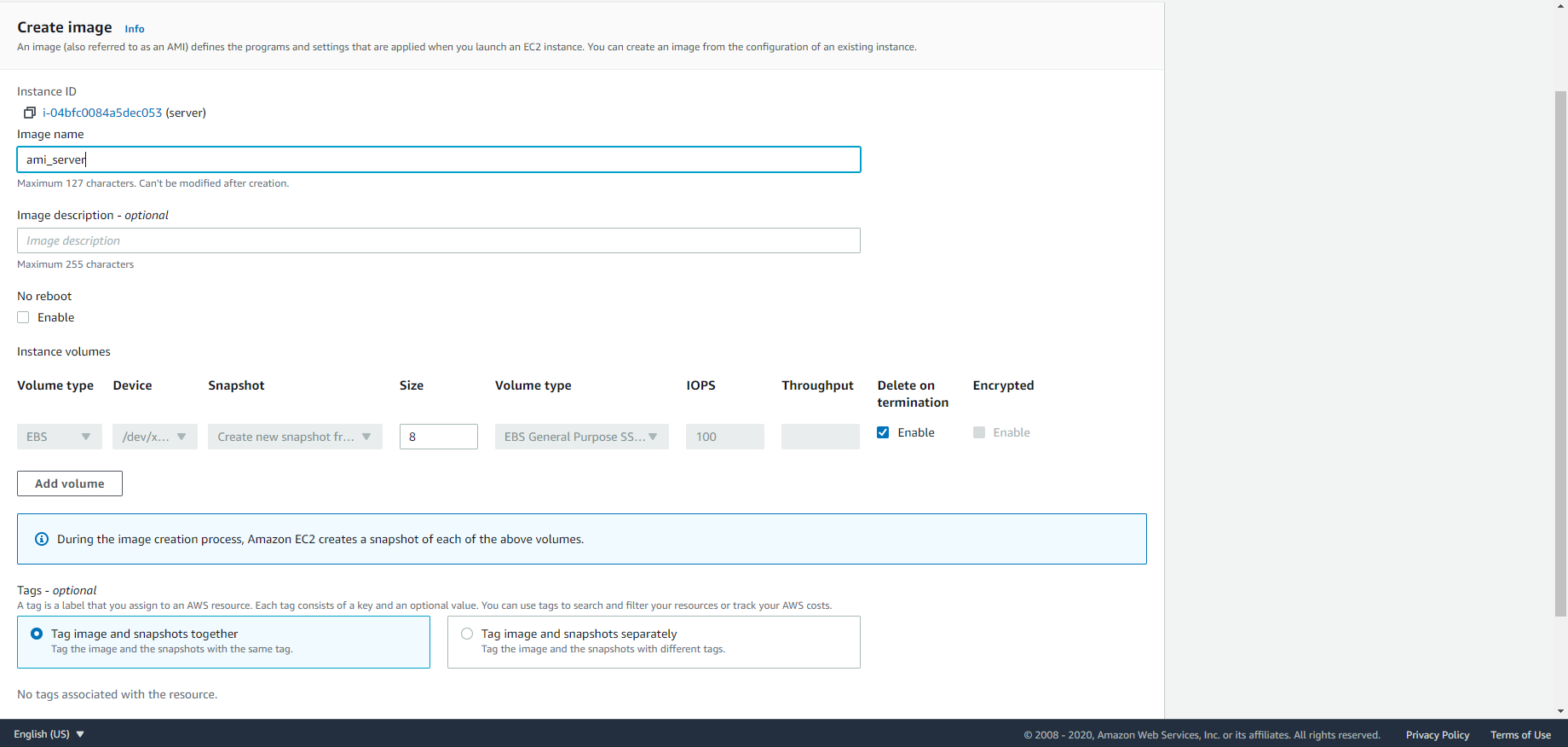
在做这个实验之前，先确保自己的VPC有两个不同可用区的公有子网

1.继续使用刚刚的实例，我们给他创建一个属于自己的AMI

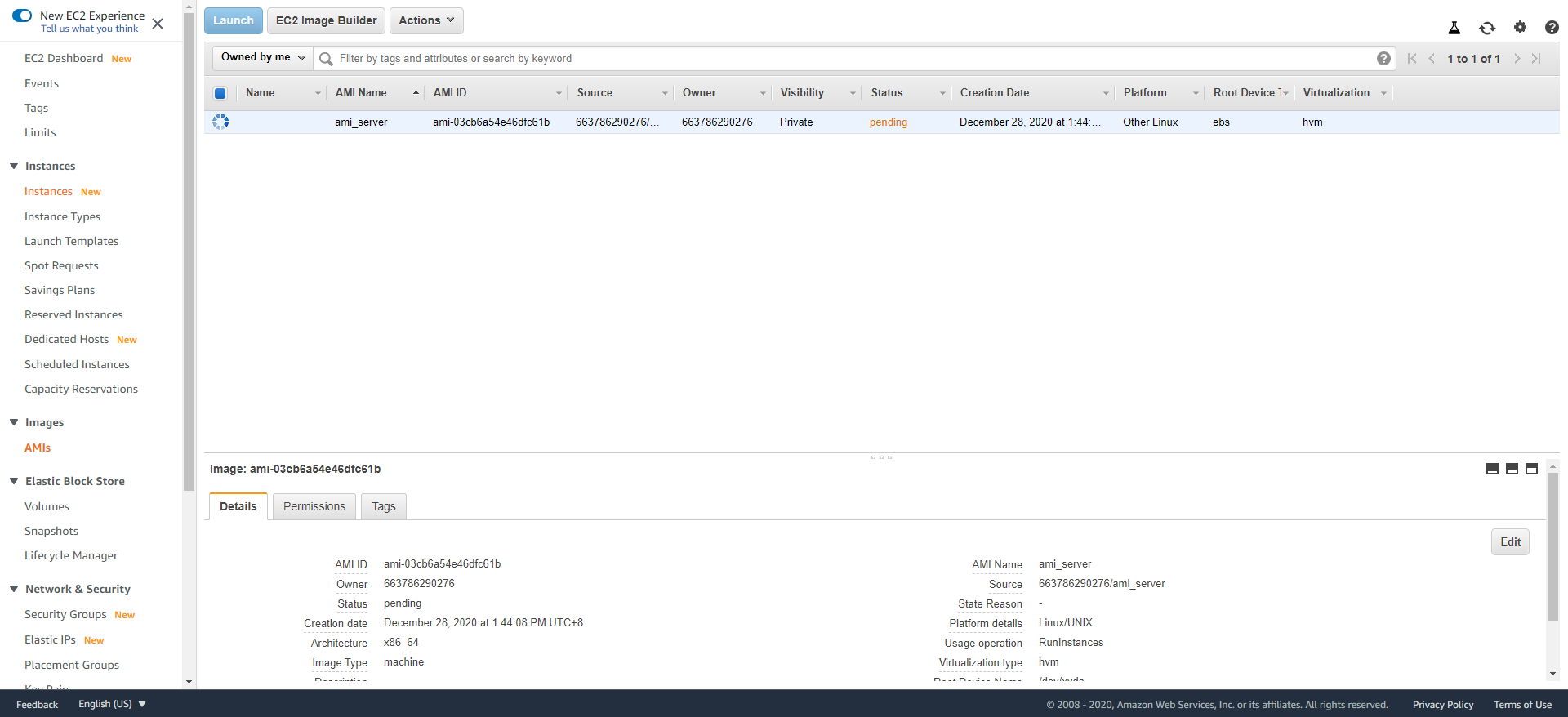
勾选实例，点击Action->Image and templates->Create image



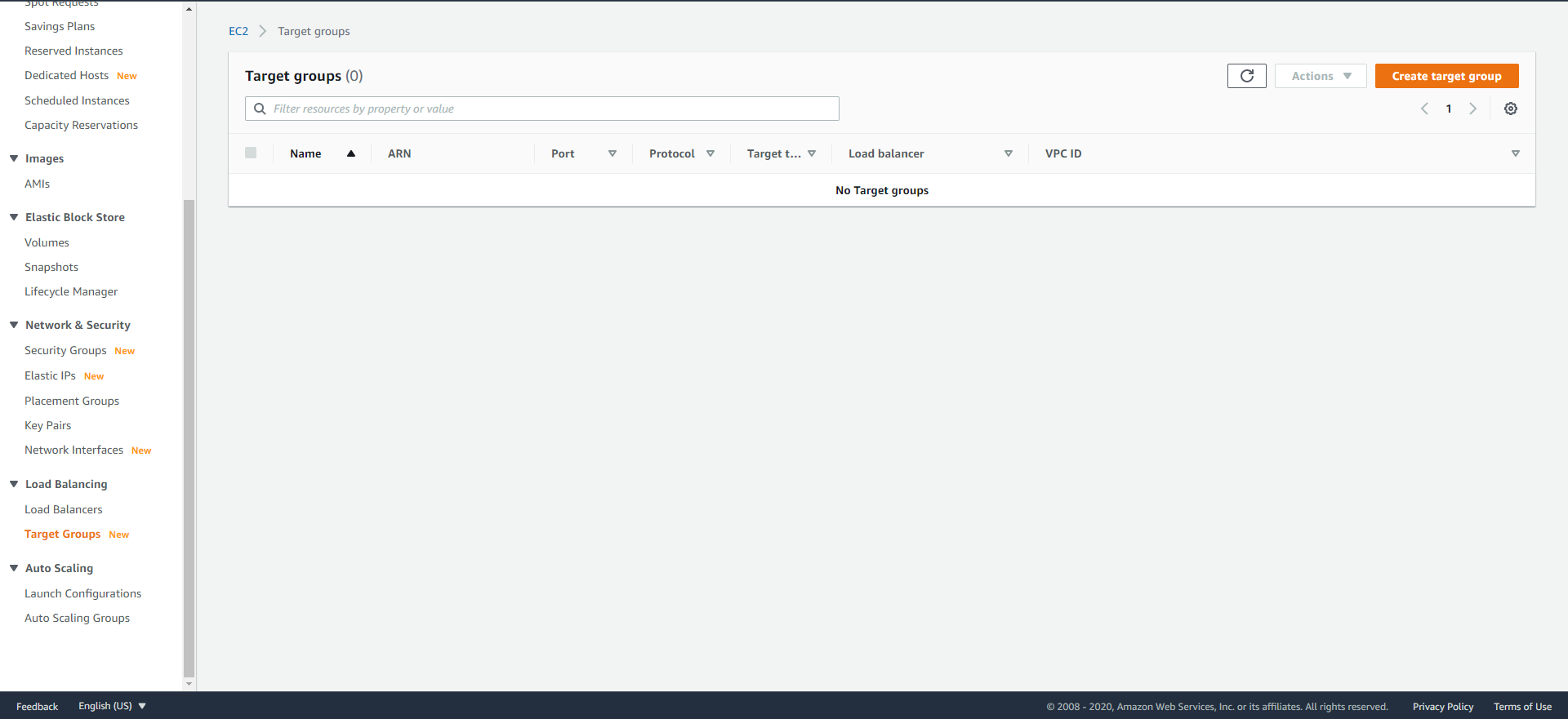
输入AMI的名字，点击Create Image



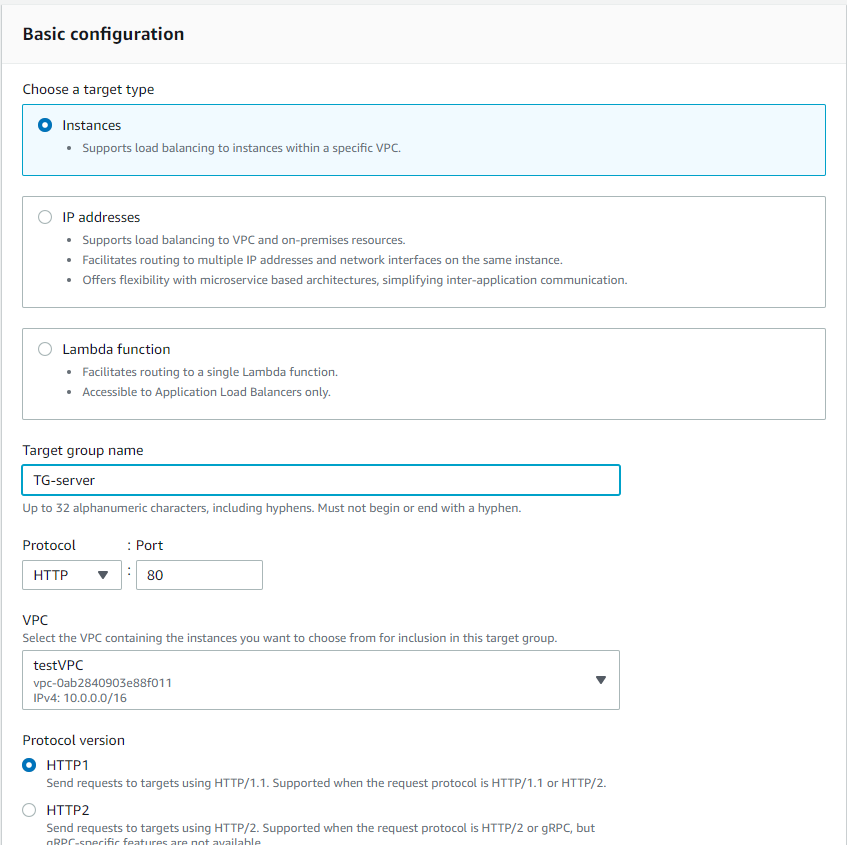
工具栏点击AMIs，会发现AMI正在生成，等待片刻，此时实例会重启



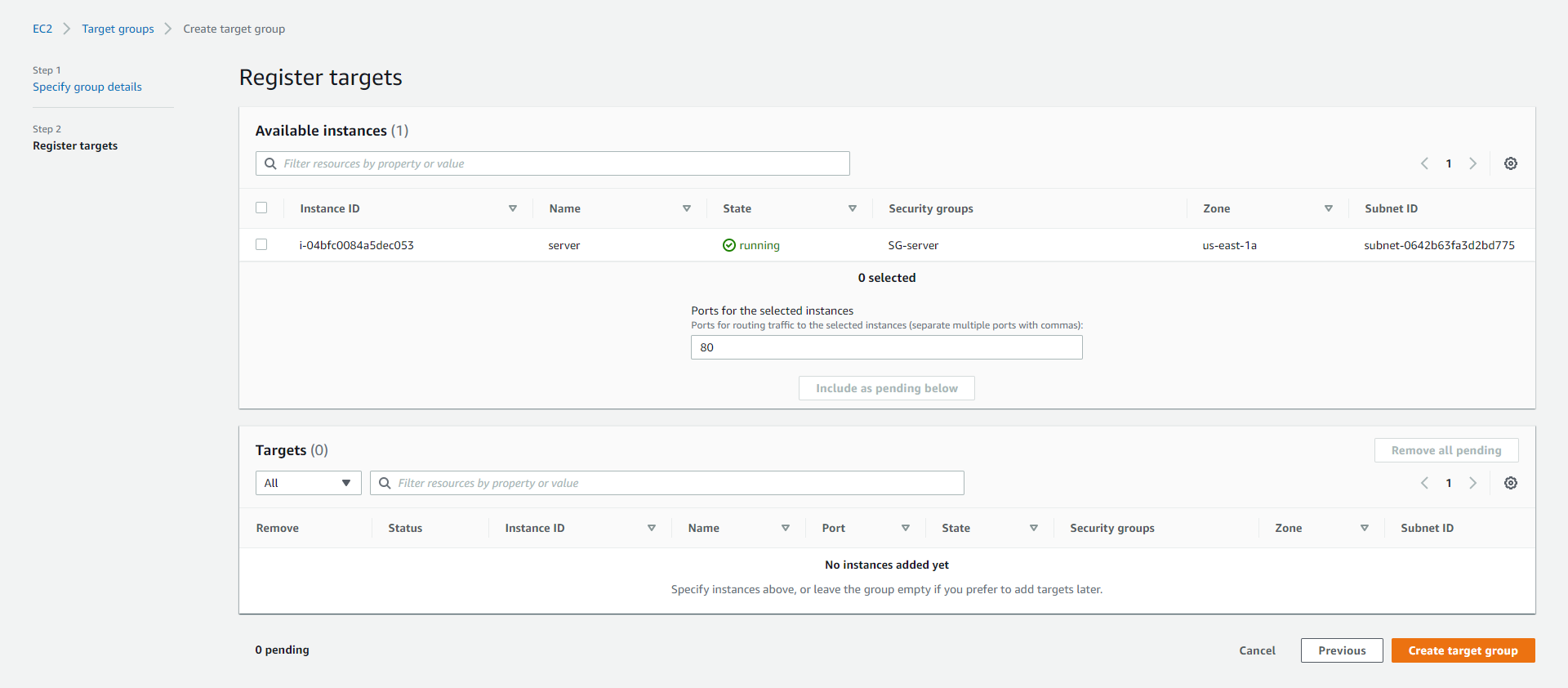
2.此时我们先来创建负载均衡器，首先先创建目标组，工具栏点击Target Groups->Create Target group



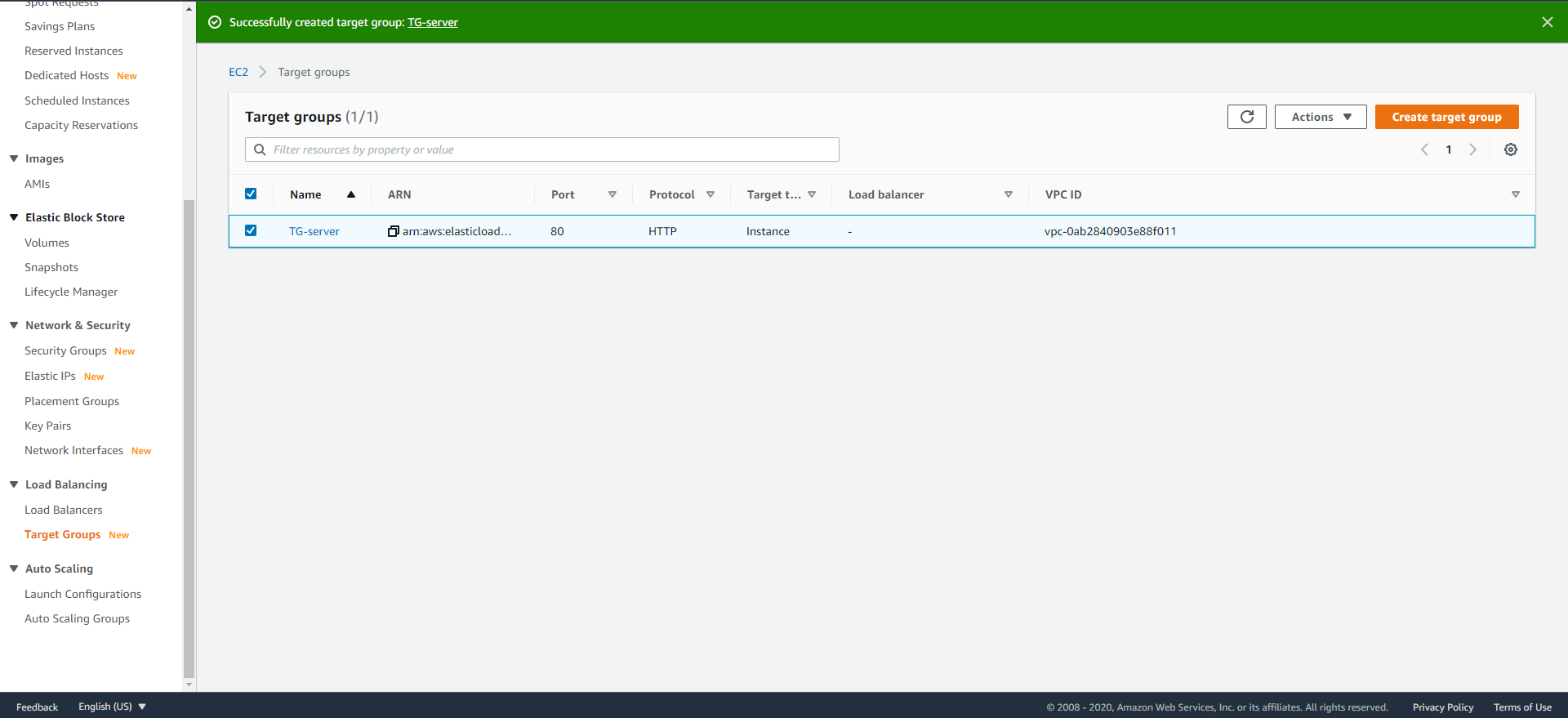
在基本配置里填写Target group name，Port选择80（因为Apache使用的是80端口），选择VPC，其他保持默认，点击Next



点击Create target group

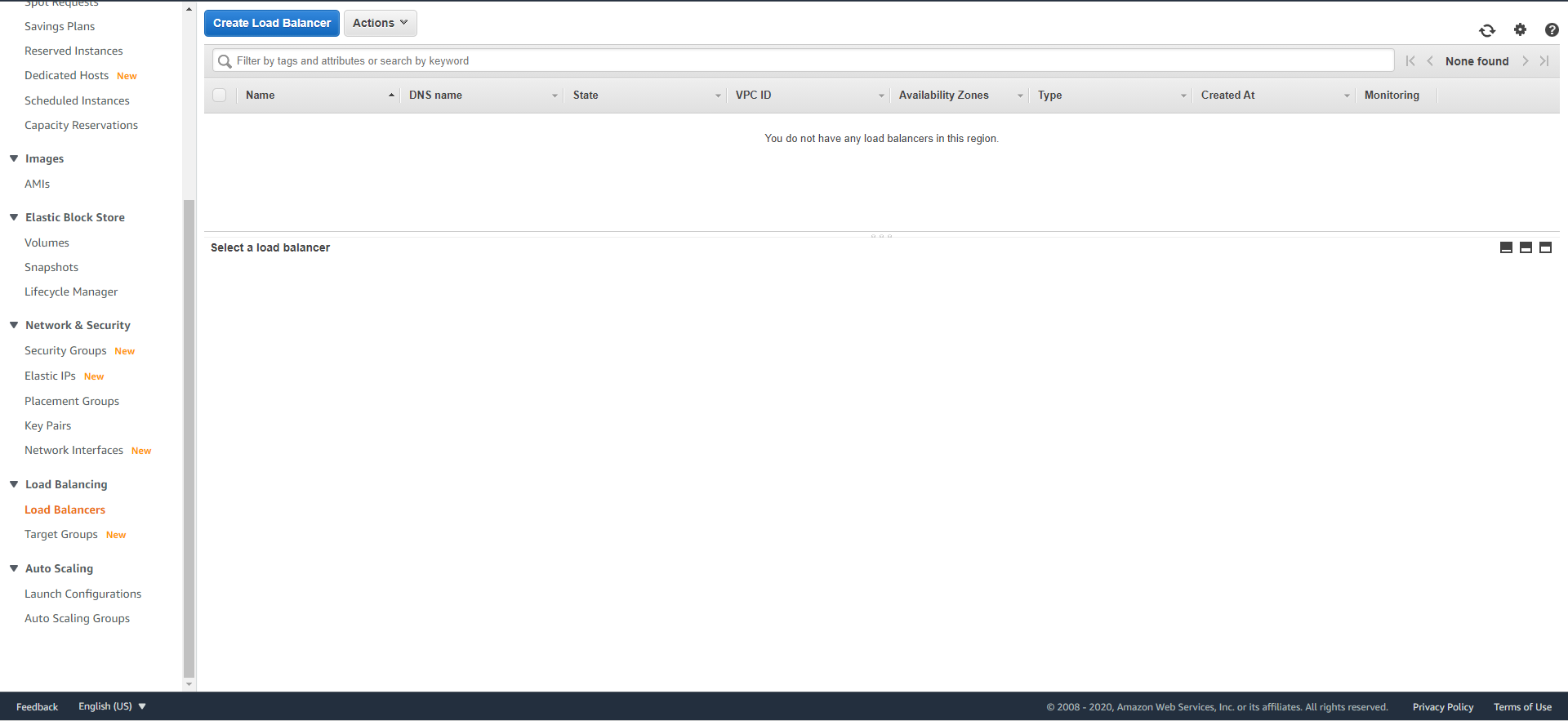


创建完成

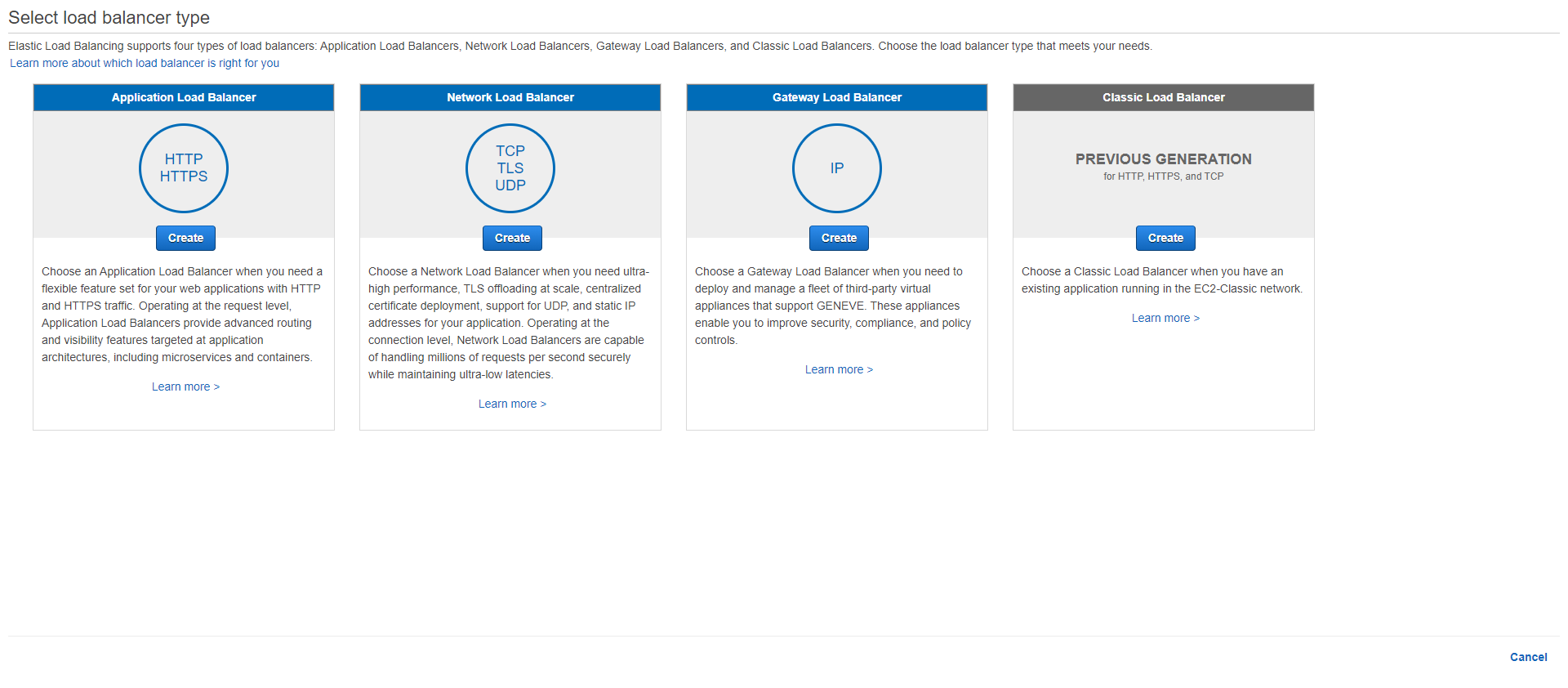


3.创建负载均衡器

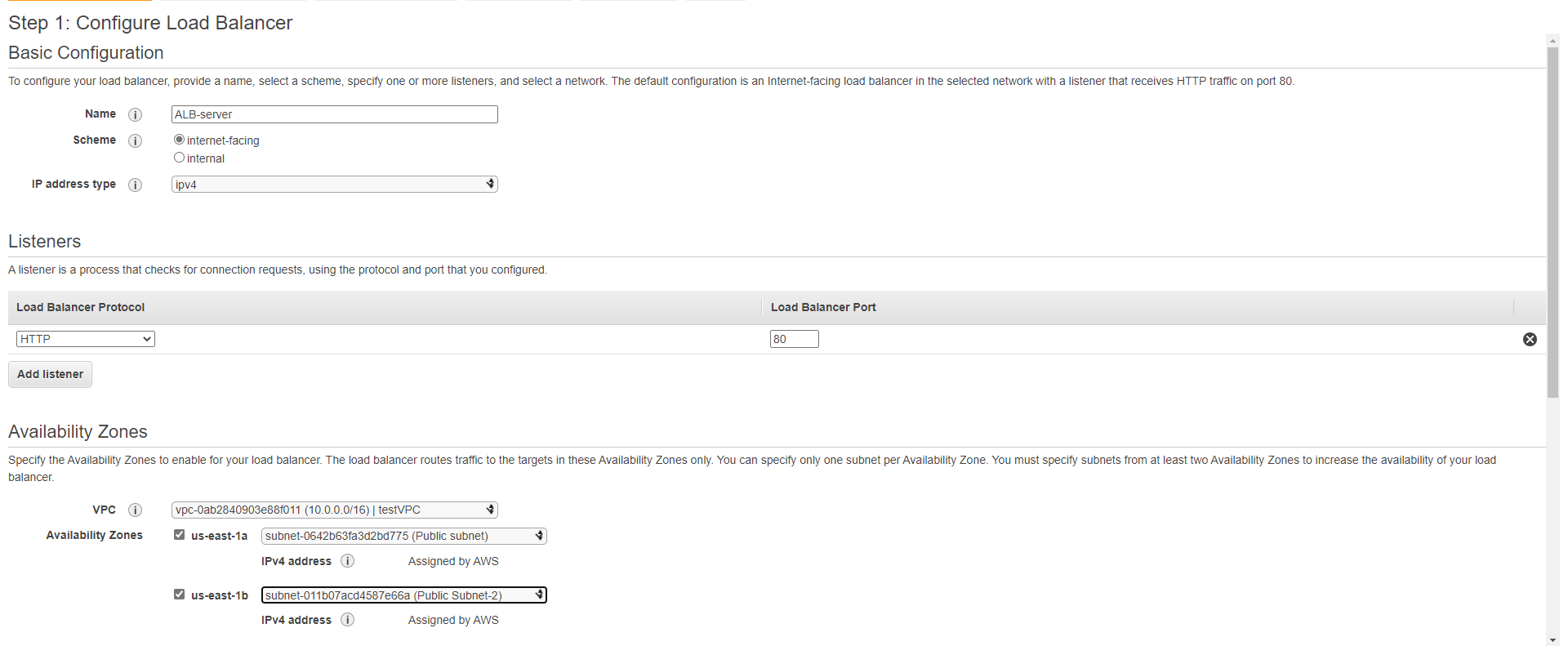
点击Load Banlancers->Create Load Balancer



选择Application Load Balancer



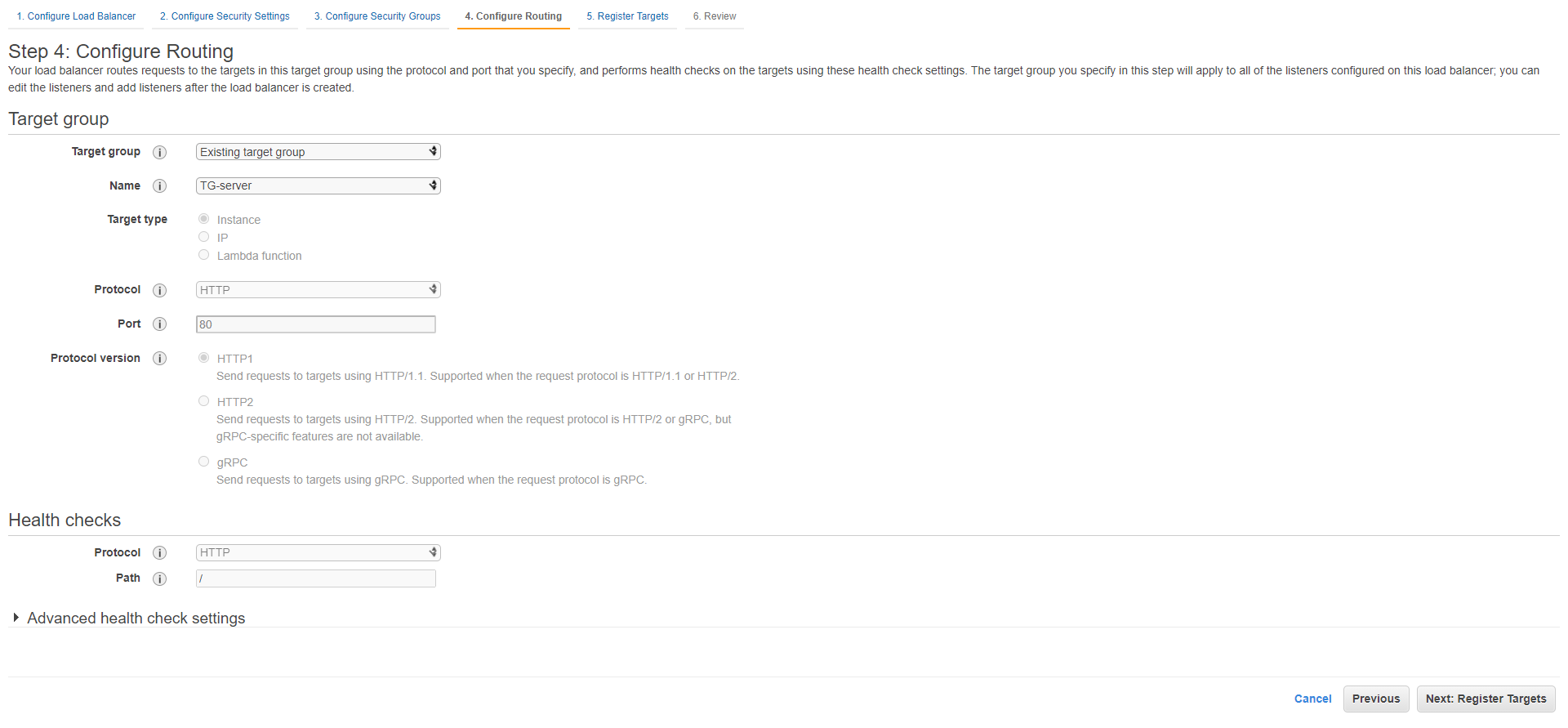
输入ALB的名字，监听器端口选择80，即负载均衡器对外暴露80端口，选择VPC，选择两个公有子网



第二步保持默认，跳到第三步，选择安全组，选择之前实例所使用的安全组



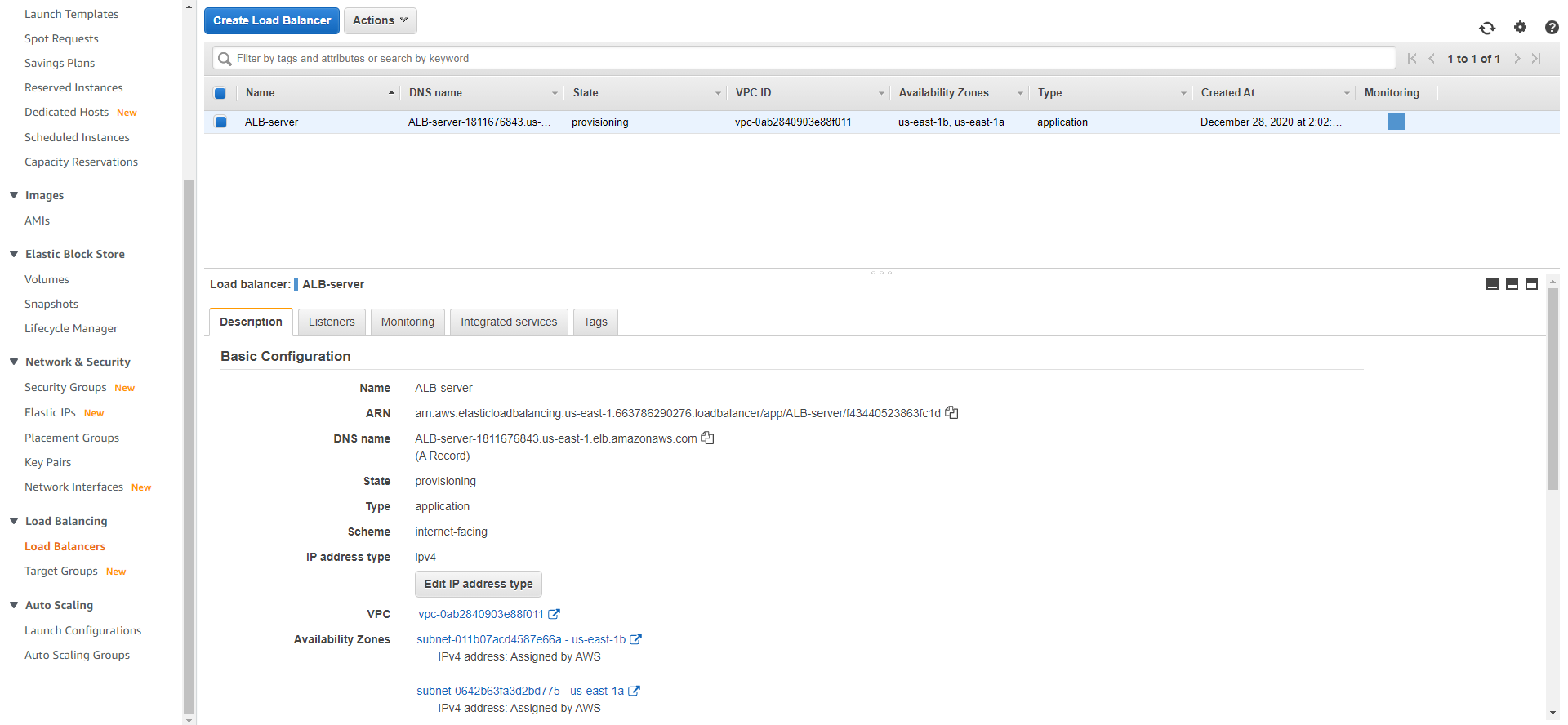
第四步，选择目标组，点击Existing target group，选择我们之前创建的一个目标组



确保配置无误后点击创建

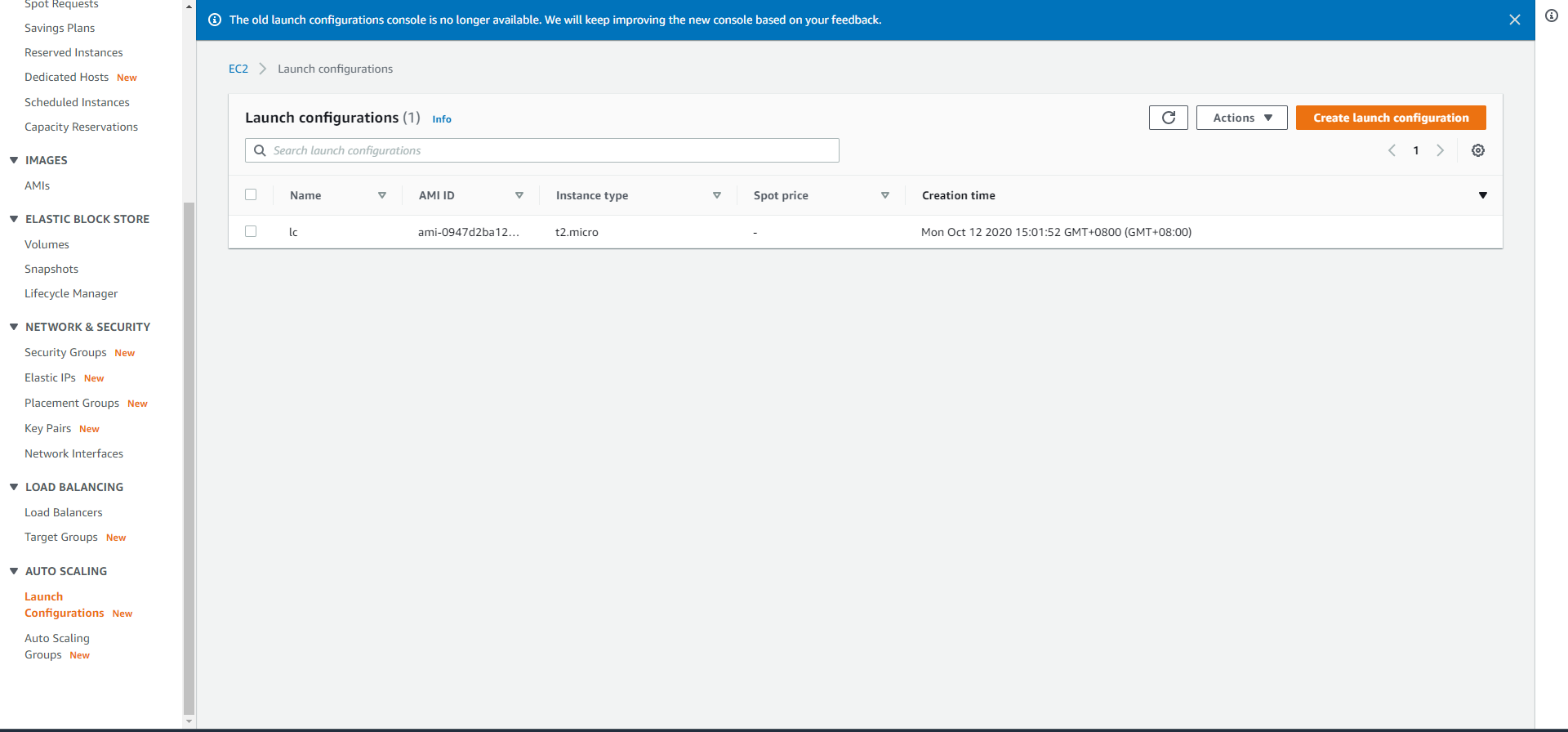


负载均衡器创建完成

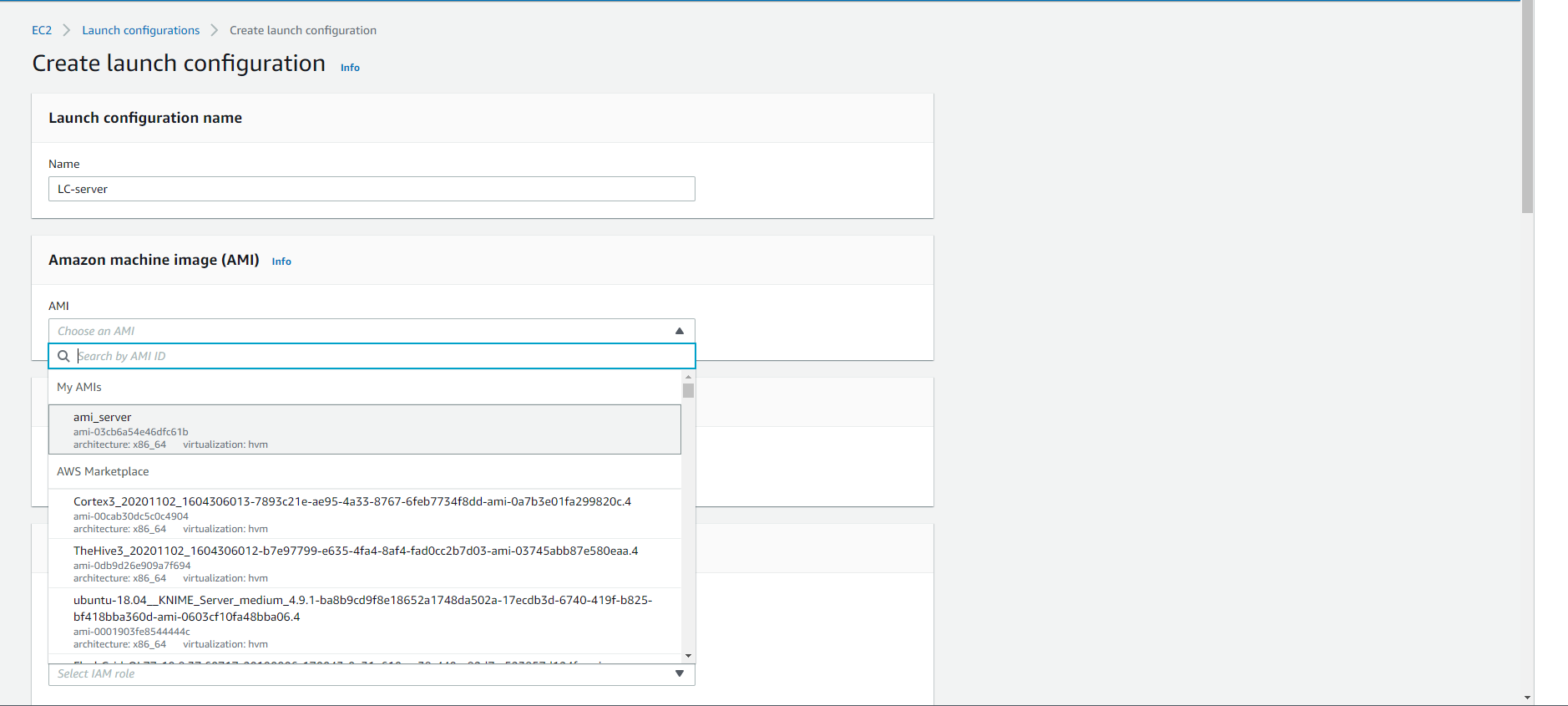


4.创建启动配置

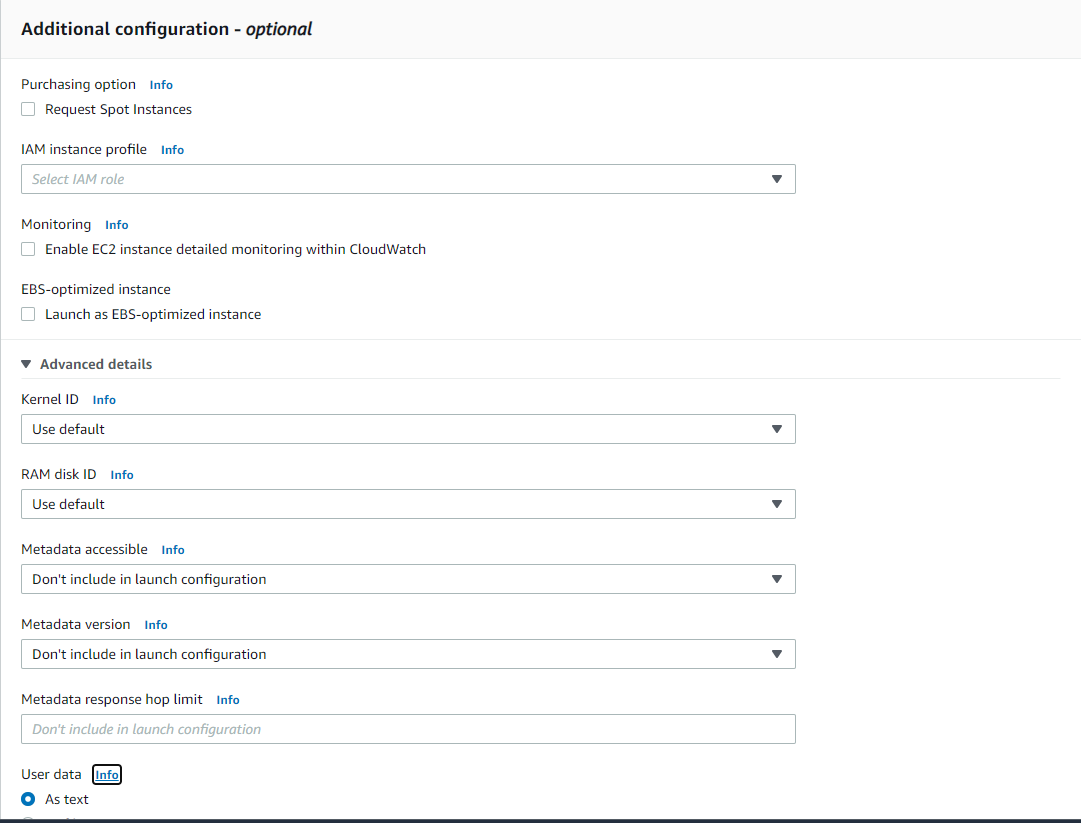
点击Launch Configurations->Create Launch Configuration



AMI选择我们自己创建的AMI，其他配置和在创建实例时的配置相似



此处有一个不同点击Additional configuration，点开Advanced details，

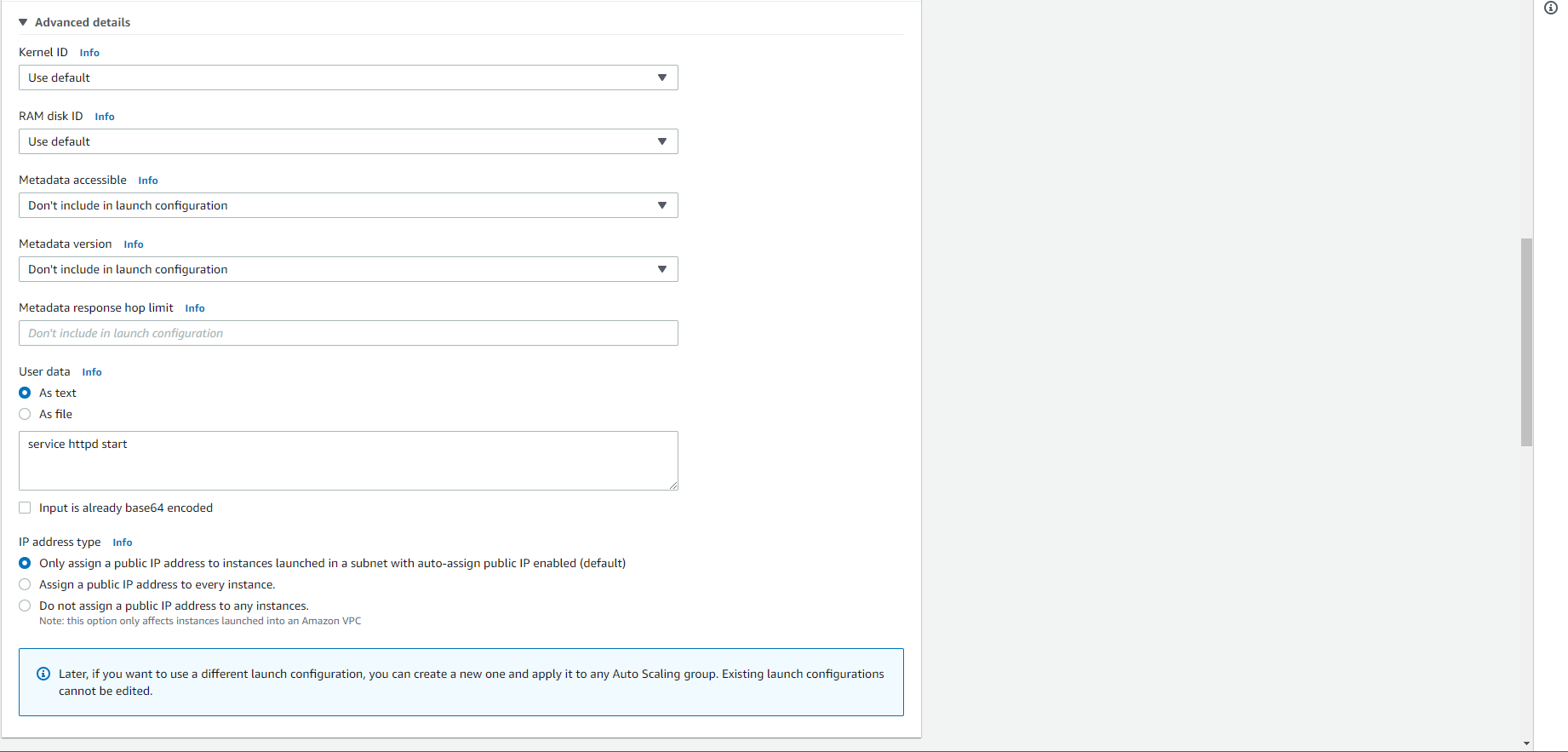


这里有一个User data，输入

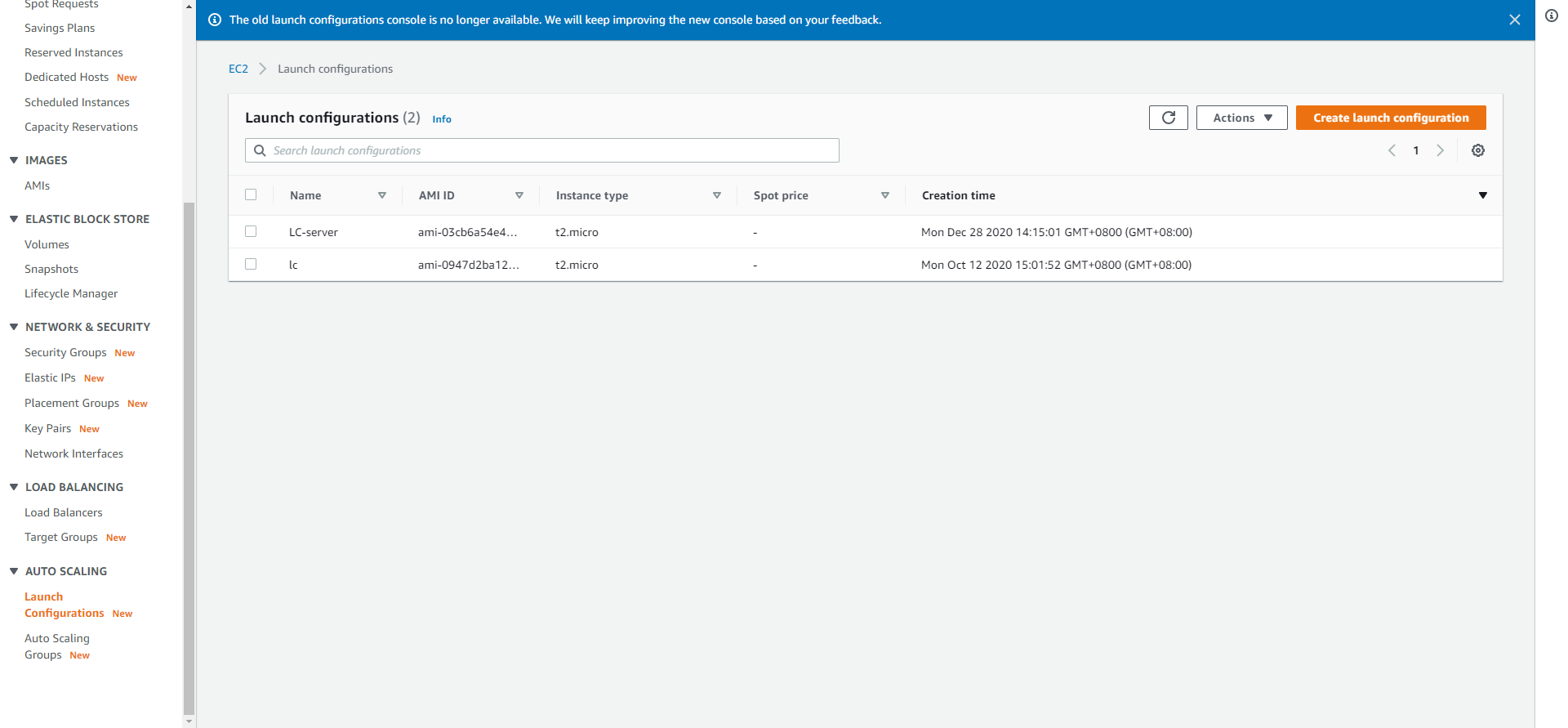
#!/bin/bash

service httpd start

User data内指定shell指令，EC2创建后启动进行预配置，因为我们之前创建的AMI没有设置apache自启动，所以需要输入命令使其启动服务，通过user data向实例传递启动apache服务的命令，就免除了手动登录到每台实例输入命令的操作。User data只在实例启动时有作用，已启动的实例无法再配置user data

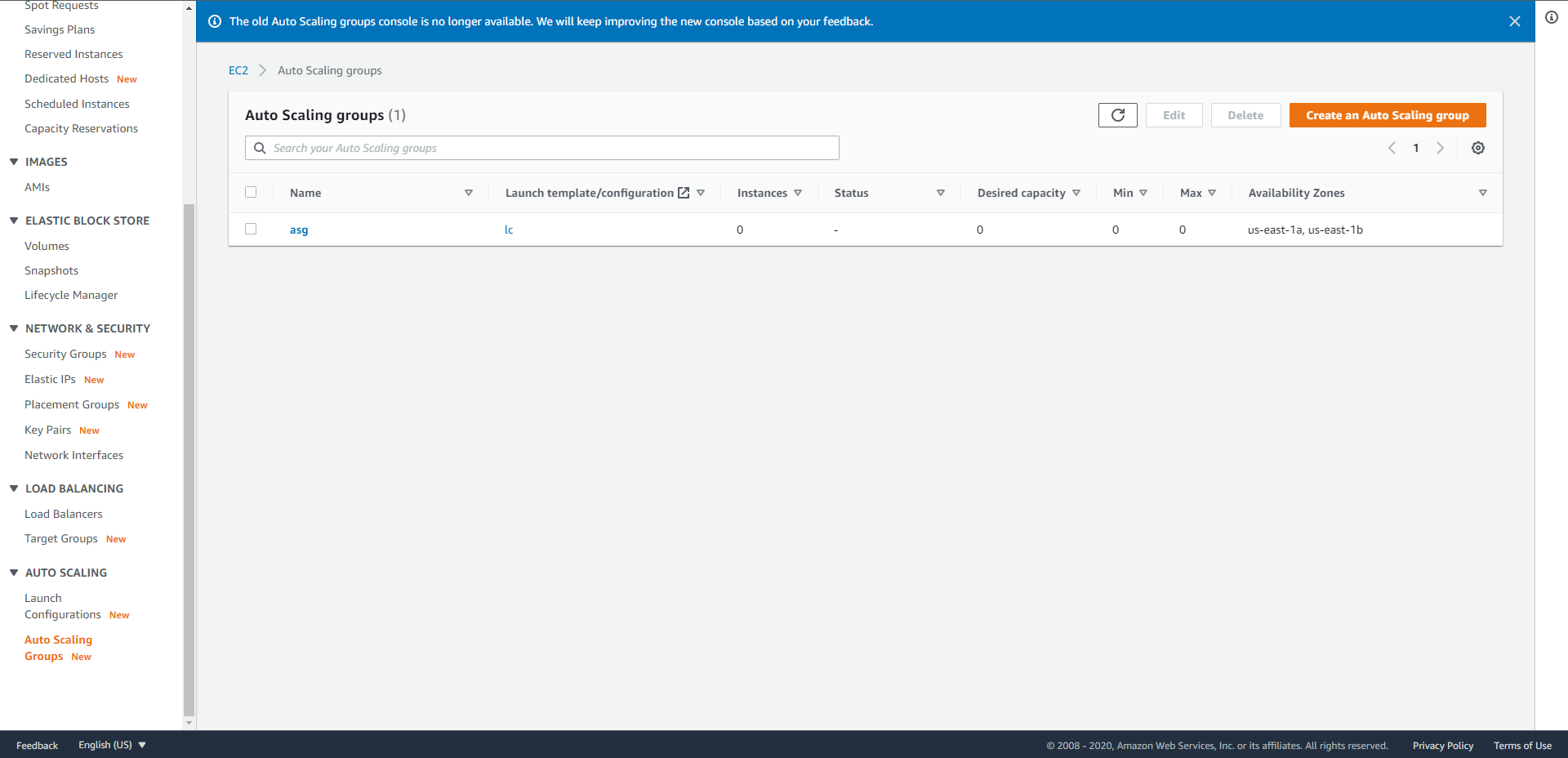


点击创建，启动配置设置完成，注意启动配置创建后无法更改，如果输入错误只能复制原有启动配置重新创建一个新的启动配置。

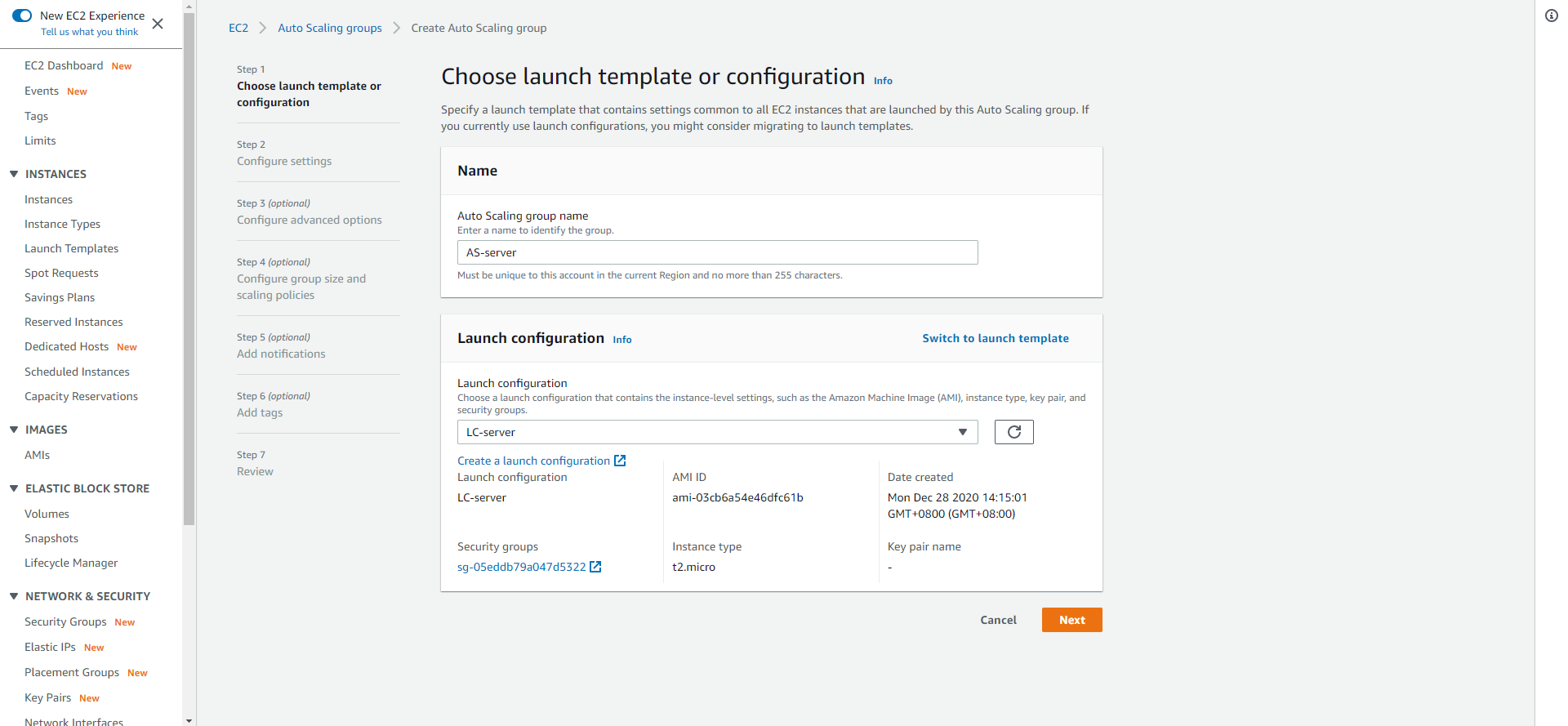


5.创建Auto Scaling

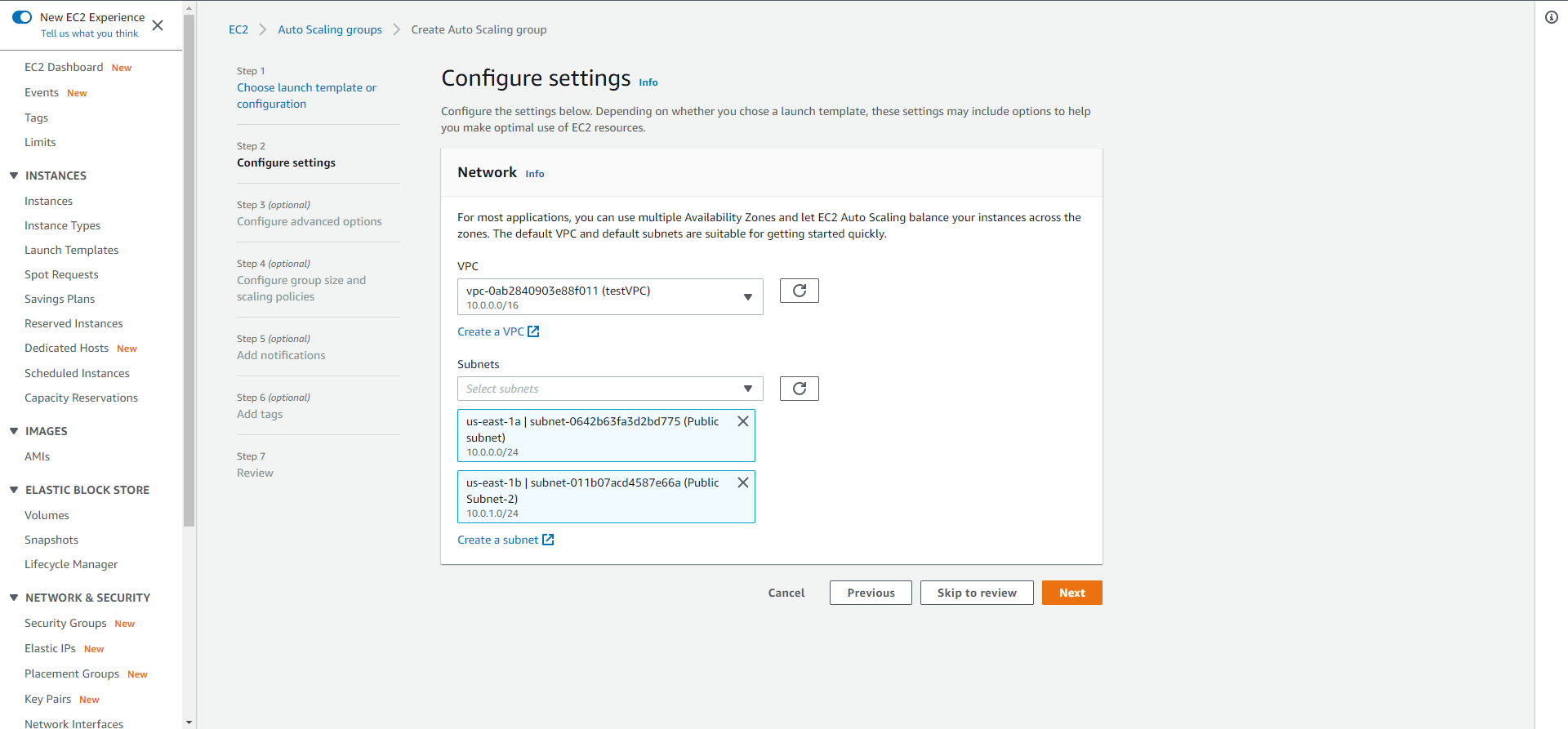
点击Auto Scaling Groups->Create Auto Scaling Group



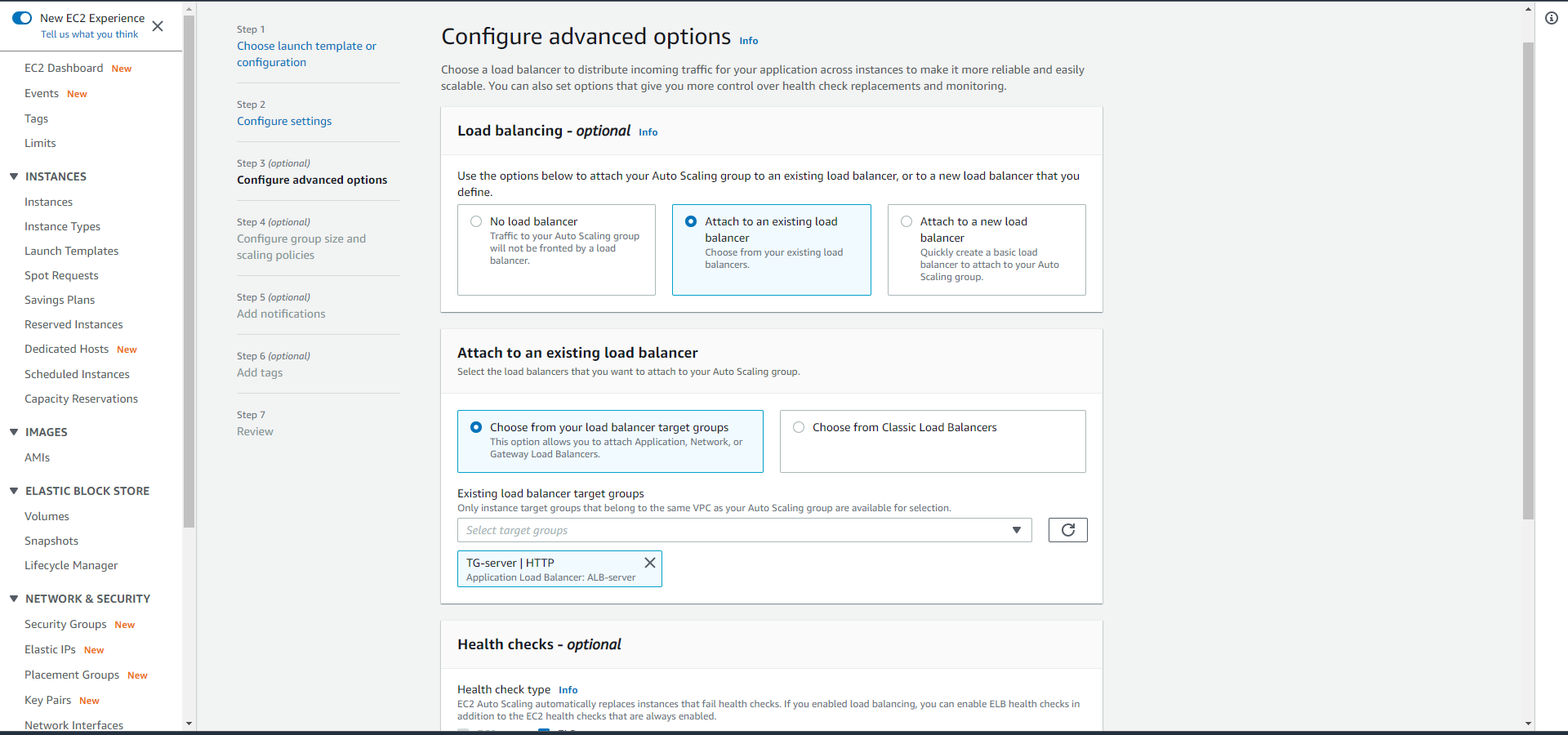
输入ASG的名字，再点击Switch to launch configuration，选择之前创建的启动配置，下一步



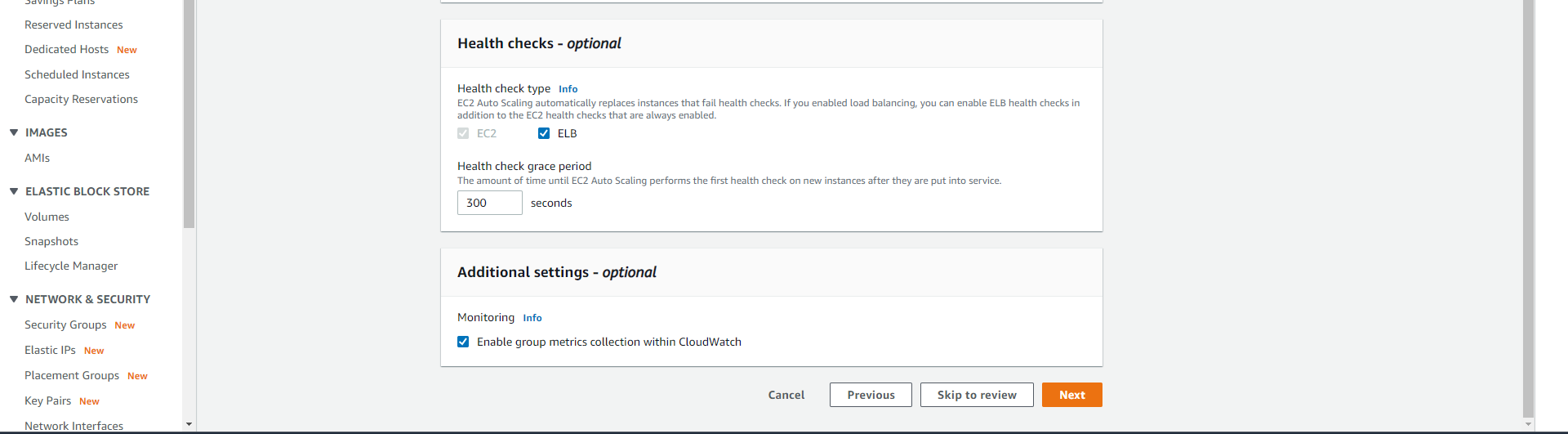
选择VPC和子网，下一步



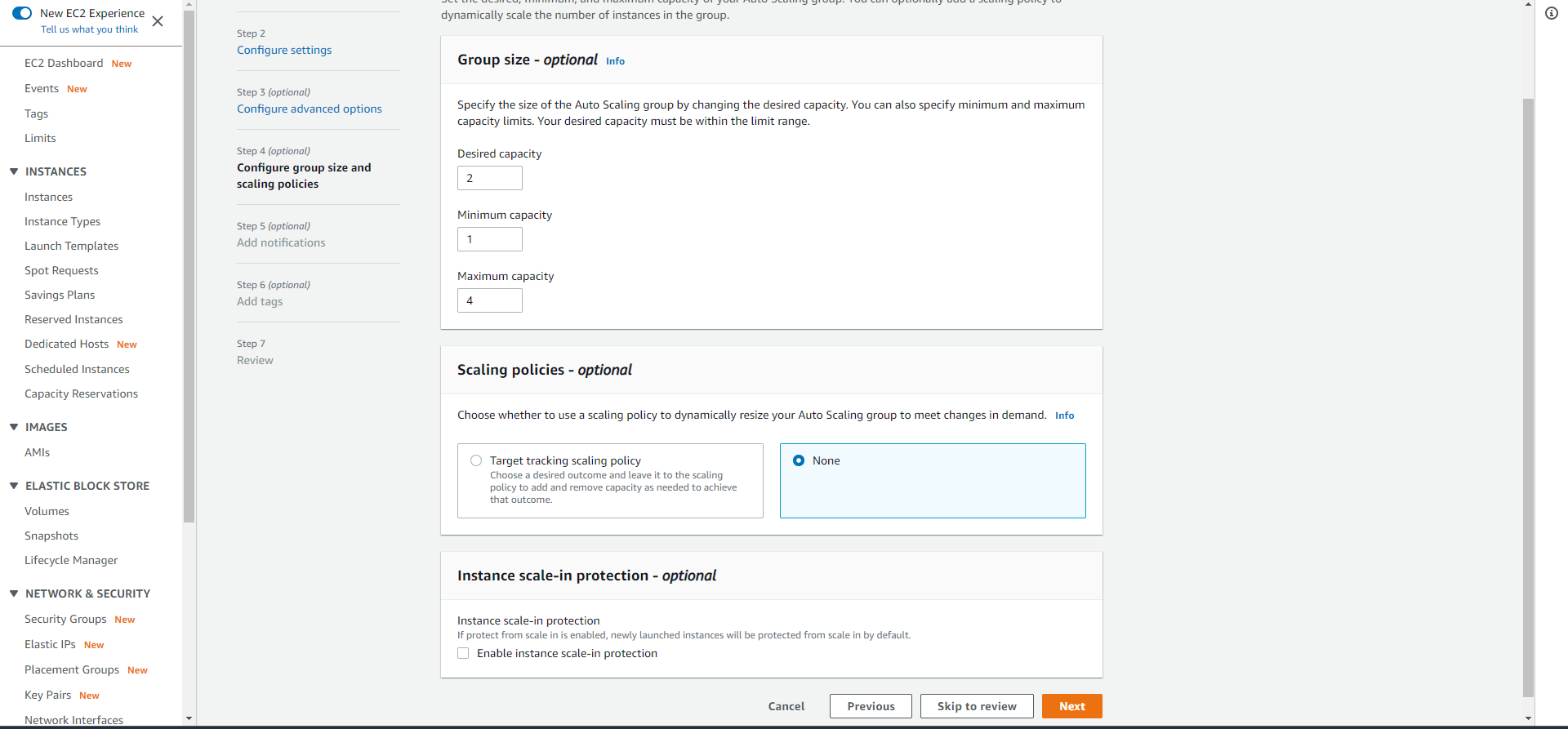
点击Attach to an existing load balancer，再在下面点击Choose from your load balancer target groups，选择ALB的目标组



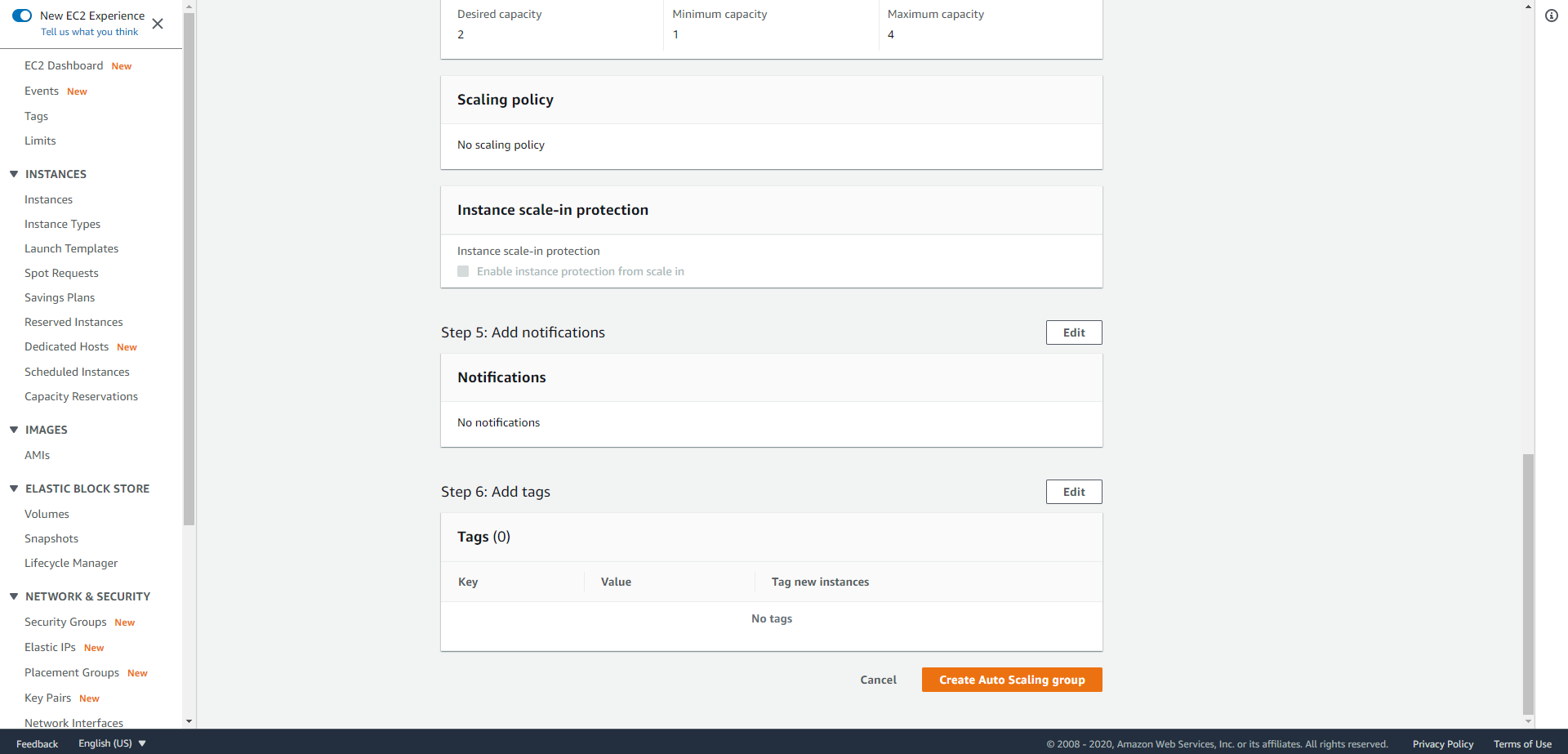
勾选两个选项，下一步



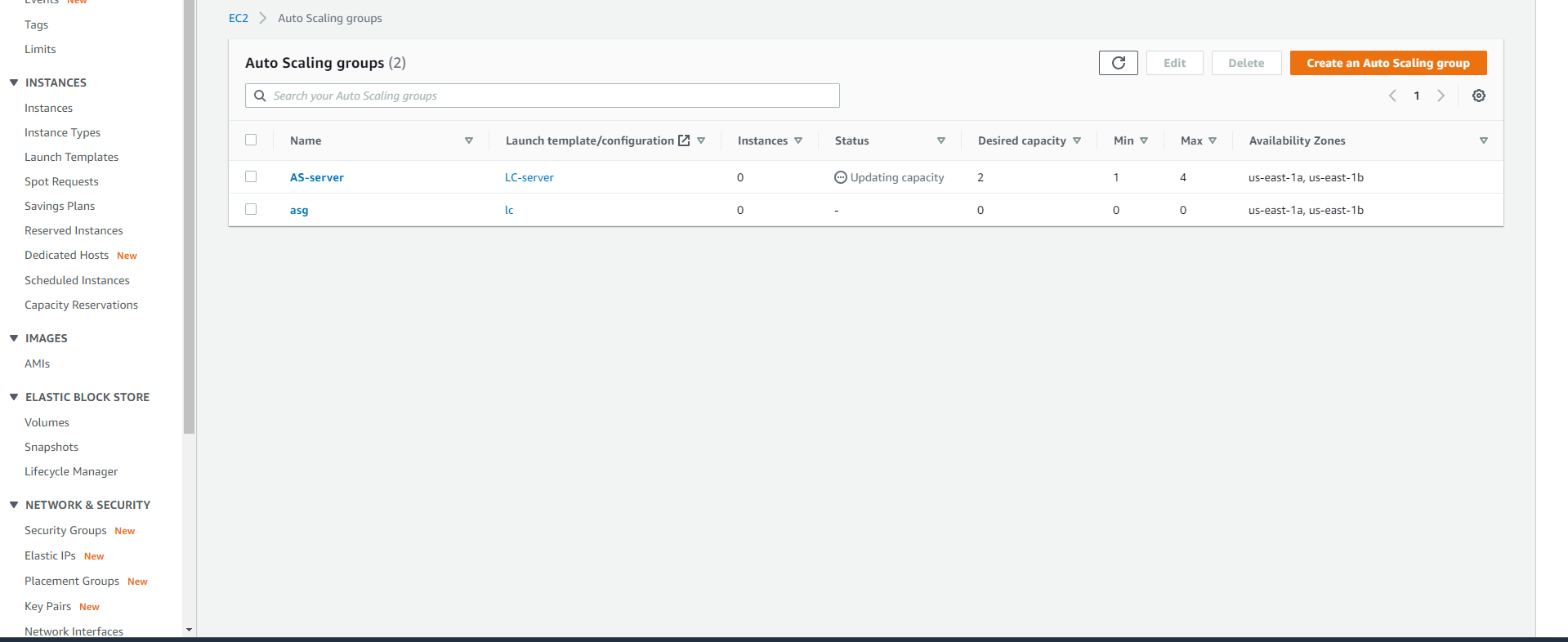
此次实验中Desired capacity（所需容量）选择2，Minimum capacity（最小容量）选择1，Maximum capacity（最大容量）选择4，下一步



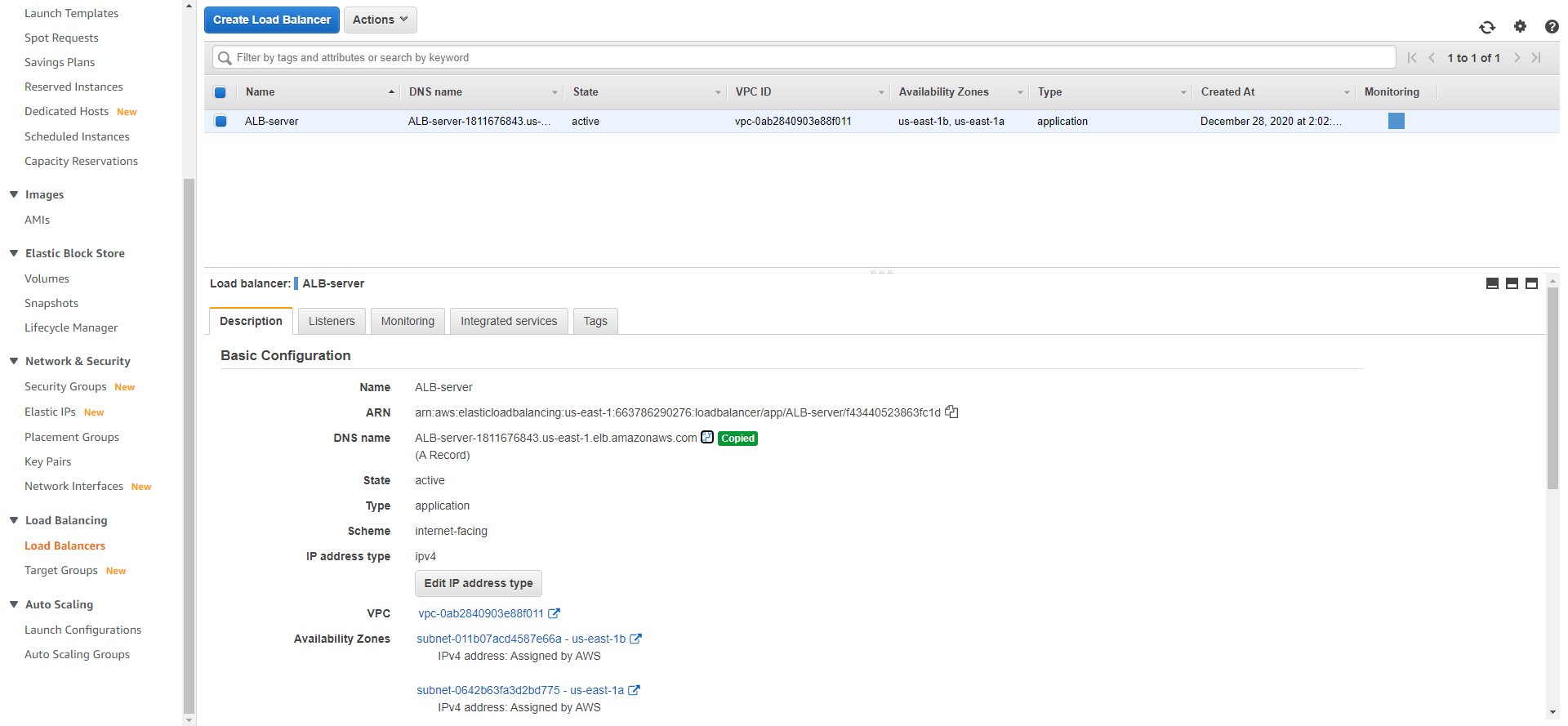
接着都是下一步即可，确认无误后点击创建



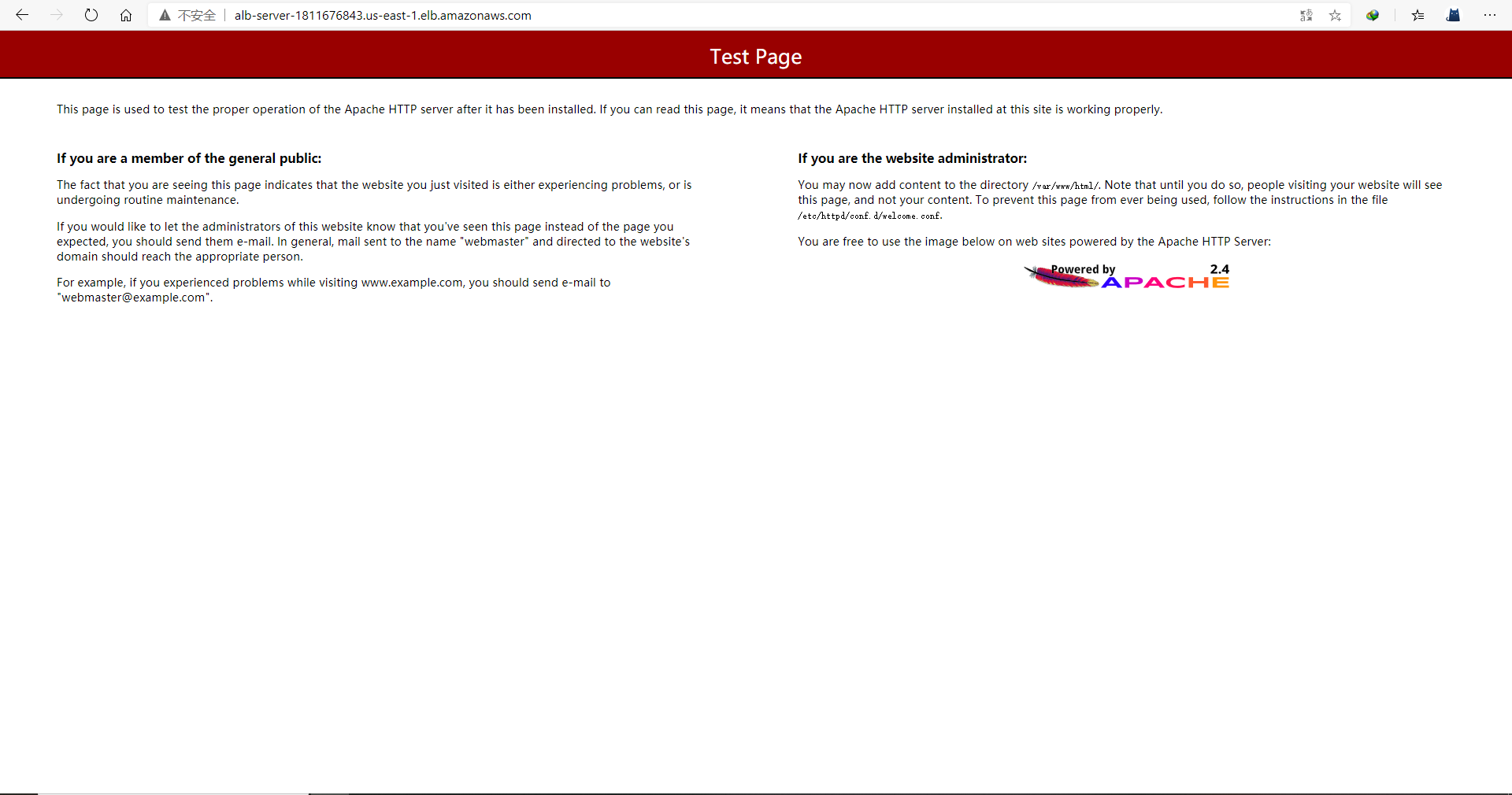
Auto Scalling组开始更新实例容量，ASG所启动的实例会自动加入ALB的目标组中



回到Load Balancers界面，复制DNS name到浏览器进行访问



出现测试界面的话，那么恭喜，这个实验做成功了



Note：由于Apache的测试界面返回状态值是403，所以尽管我们可以成功访问ALB的域名了但是目标组的健康检查依旧是无法通过的，在配置其他服务的时候如果发现Unhealthy那么说明实例的服务没有启动成功，或是目标组无法确定服务运行情况

