

电子科技大学
计算机科学与工程学院

标准实验报告

(实验) 课程名称 云应用开发实践

电子科技大学教务处制表

电子科技大学

实验报告

学生姓名：朱若愚 学号：2022150501027 指导教师：余盛季

一、实验室名称： 计算机学院实验中心

二、实验项目名称：blog++应用开发以及云服务器高可用部署

三、实验学时：4 学时

四、实验原理：

1. 博客应用开发原理

博客前端通过 HTML、CSS、JavaScript 构建用户界面，用户在浏览器中发送请求。数据存储则依赖数据库管理系统（MONGODB），用于存储用户信息、博客文章内容等数据，实现数据的增删改查操作。

2. 云服务器部署原理

将博客应用部署到云服务器，云服务器本质上是远程的计算资源，提供计算、存储和网络资源。首先在云服务器上搭建运行环境，如安装操作系统、Web 服务器、后端运行环境（根据所选语言安装对应的运行时环境）以及数据库。然后将博客应用的代码上传至云服务器，配置 Web 服务器使其能够正确访问应用代码，启动后端服务与数据库服务，并确保网络设置允许外部访问，从而使用户能通过互联网访问部署在云服务器上的博客应用。

3. 弹性负载均衡（ELB）原理

弹性负载均衡是一种将流量分发到多个服务器的技术。当用户请求到达 ELB 时，它根据预先设定的负载均衡策略（如轮询、加权轮询、最少连接等），将请求分配到后端多台服务器上。这样可以分散流量压力，避免单点故障，提高应用的可用性和可靠性。同时，ELB 还具备健康检查功能，能自动检测后端服务器的健康状态，若发现有服务器出现故障，会将其从分发列表中移除，确保用户请求只发送到健康的服务器上。

4. 弹性伸缩（AS）原理

弹性伸缩根据业务负载自动调整服务器数量。通过设定特定的伸缩策略（如基于 CPU 使用率、内存使用率、网络流量等指标），监控系统持续收集服务器的性能指标数据。当负载升高，指标超过预设的阈值时，触发伸缩组自动增加服务器实例，快速扩展计算资源，以应对高峰流量；相反，当负载降低，指标低于阈值时，系统自动减少服务器实例，释放闲置资源，从而有效降低运营成本并确保业务始终具备良好的性能表现。

五、实验目的：

1.掌握博客应用的基本开发流程，包括前端界面设计、后端逻辑编写以及数据库交互操作，理解 Web 应用的架构原理和开发方法。

2.学习云服务器的租用、配置与管理，能够将开发好的应用成功部署到云服务器上并使其对外提供服务，了解云环境下的应用部署要点和注意事项。

3.深入理解弹性负载均衡和弹性伸缩的工作机制与应用场景，通过实际操作掌握在云平台上配置和使用 ELB 与 AS 的方法，提升应对高并发、可变负载场景下的应用运维能力，确保系统具备良好的稳定性和成本效益。

六、实验内容：

博客应用开发

使用 HTML、CSS 构建博客的页面布局，包括首页展示博客列表、详情页展示文章内容、编辑页用于创作和修改博客等页面；利用 JavaScript 实现页面的交互功能,使用 mongodb 存储数据。

云服务器部署

1. 云服务器环境搭建：在华为云上租用一台云服务器实例，选择合适的操作系统（Ubuntu）并完成初始化配置；安装 Web 服务器软件，配置虚拟主机使其指向博客应用的部署目录；安装后端运行环境以及数据库软件，并进行相应的配置初始化，确保各组件能够正常运行。
2. 应用部署与配置：将开发好的博客应用代码通过 FTP 上传至云服务器指定目录；配置 Web 服务器，使其能够加载和运行后端应用，并设置正确的静态文件路径，使前端资源能够被正确访问；启动后端应用服务以及数据库服务，测试应用在本地云服务器环境中的运行情况，确保无误后，配置云服务器的防火墙规则和安全组策略，开放必要的端口，使外部用户能够通过域名或 IP 地址访问博客应用。

弹性负载均衡（ELB）配置

1. 创建 ELB 实例：在云平台控制台中，选择弹性负载均衡服务，创建一个 ELB 实例，配置监听端口，选择合适的负载均衡算法。
2. 添加后端服务器：将之前部署好的云服务器添加到 ELB 的后端服务器列表中，并配置服务器的权重。
3. 配置健康检查：设置 ELB 的健康检查参数，包括检查间隔时间、超时时间、健康阈值和不健康阈值等，使 ELB 能够自动检测后端服务器的运行状态，确保只有健康的服务器接收用户请求。

弹性伸缩（AS）配置

1. 创建伸缩组：在云平台的弹性伸缩服务中，创建一个伸缩组，指定伸缩组的最小实例数和最大实例数，定义服务器的规格配置，并关联之前部署的博客应用镜像或启动配置，确保新创建的服务器实例能够按照要求安装和配置好博客应用环境。或基于监控指标（的策略；配置伸缩策略的触发条件、冷却时间等参数。
2. 关联弹性负载均衡：将创建好的 ELB 实例与伸缩组进行关联，使得伸缩组中新增的服务器实例能够自动添加到 ELB 的后端服务器列表中，而移除的实例也能从 ELB 中同步移除，实现自动化的流量分发和服务器伸缩协同工作。

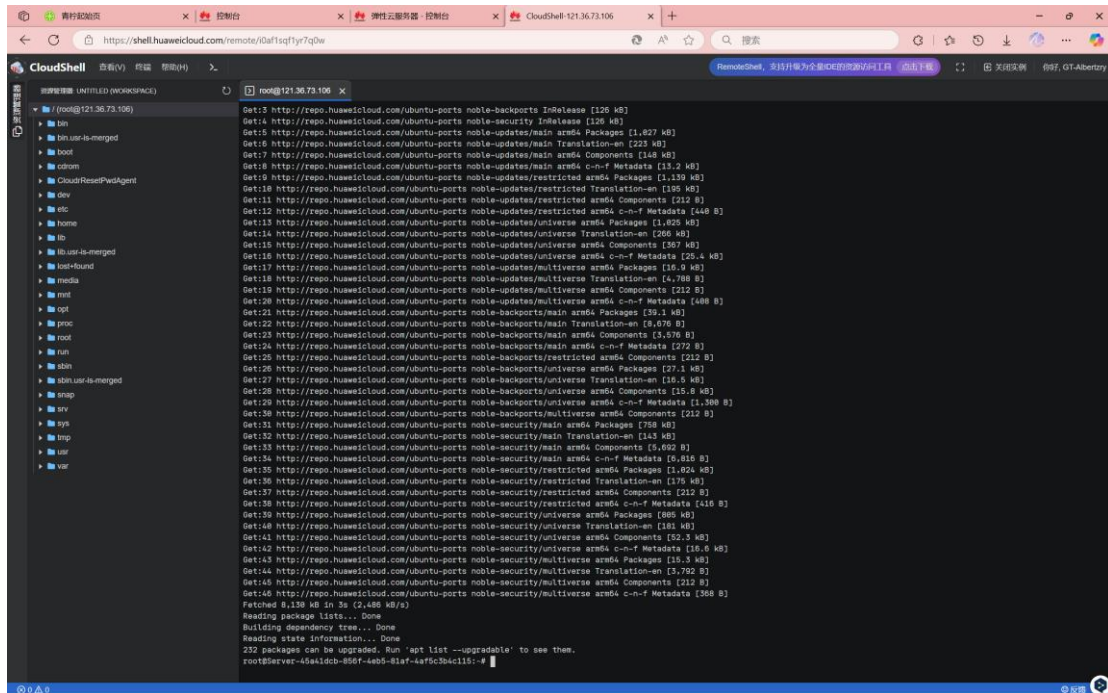
七、实验环境：

1vCPUs | 1GiB | kc1.small.1

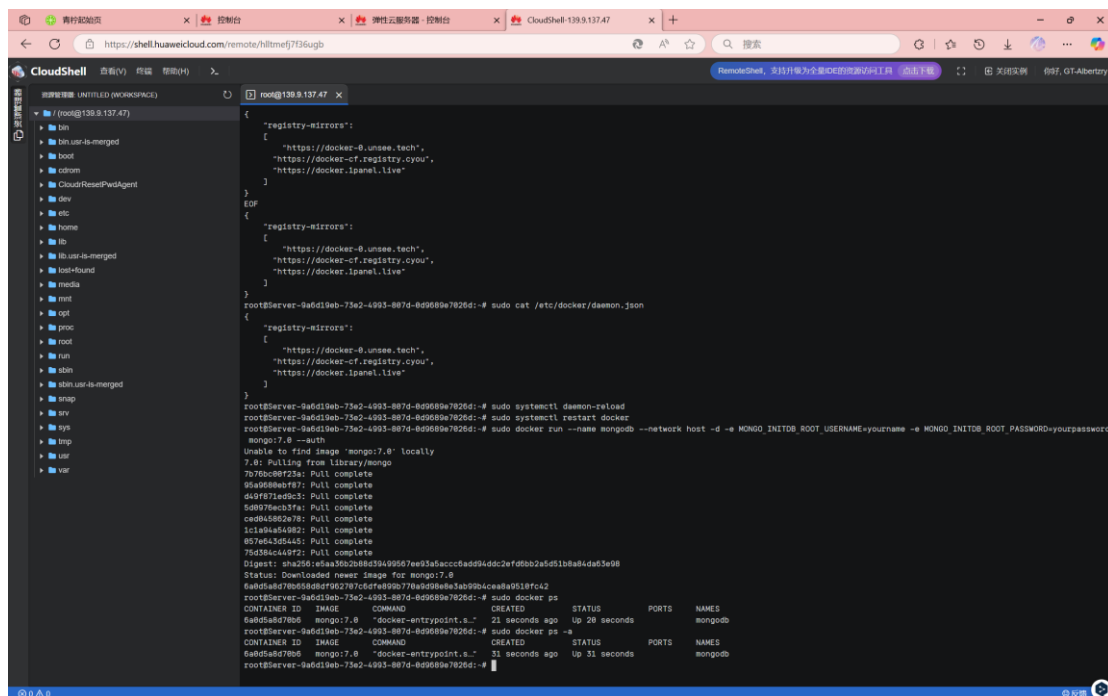
Ubuntu 24.04 server 64bit with ARM

八、实验步骤：

1. 购买云服务器并在此台服务器上安装 MongoDB



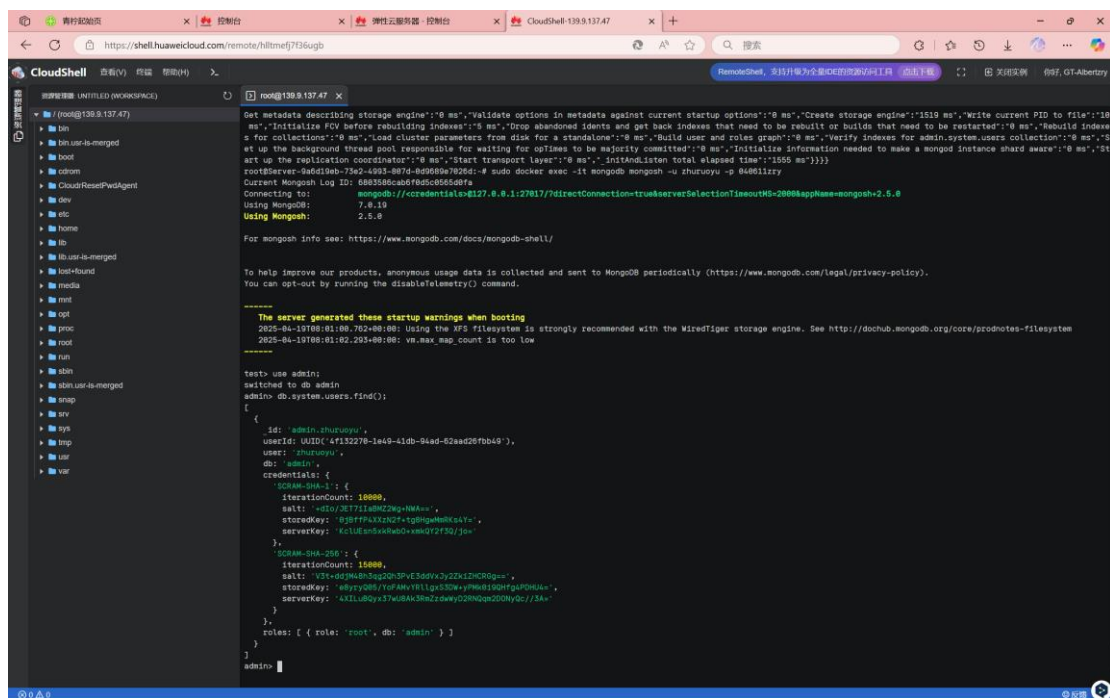
```
root@121.36.73.106:~# apt update
Get:1 http://repo.huaweicloud.com/ubuntu-ports noble-backports InRelease [126 kB]
Get:2 http://repo.huaweicloud.com/ubuntu-ports noble-security InRelease [126 kB]
Get:3 http://repo.huaweicloud.com/ubuntu-ports noble-updates/main arm64 Packages [1,627 kB]
Get:4 http://repo.huaweicloud.com/ubuntu-ports noble-updates/main Translation-en [223 kB]
Get:5 http://repo.huaweicloud.com/ubuntu-ports noble-updates/main arm64 Components [148 kB]
Get:6 http://repo.huaweicloud.com/ubuntu-ports noble-updates/main arm64 c-n-f Metadata [13.2 kB]
Get:7 http://repo.huaweicloud.com/ubuntu-ports noble-updates/restricted arm64 Packages [1,319 kB]
Get:8 http://repo.huaweicloud.com/ubuntu-ports noble-updates/restricted Translation-en [195 kB]
Get:9 http://repo.huaweicloud.com/ubuntu-ports noble-updates/restricted arm64 Components [212 B]
Get:10 http://repo.huaweicloud.com/ubuntu-ports noble-updates/restricted arm64 c-n-f Metadata [446 B]
Get:11 http://repo.huaweicloud.com/ubuntu-ports noble-updates/universe arm64 Packages [2,405 kB]
Get:12 http://repo.huaweicloud.com/ubuntu-ports noble-updates/universe arm64 Components [266 kB]
Get:13 http://repo.huaweicloud.com/ubuntu-ports noble-updates/universe arm64 c-n-f Metadata [16.9 kB]
Get:14 http://repo.huaweicloud.com/ubuntu-ports noble-updates/multiverse arm64 Packages [16.9 kB]
Get:15 http://repo.huaweicloud.com/ubuntu-ports noble-updates/multiverse arm64 Components [212 B]
Get:16 http://repo.huaweicloud.com/ubuntu-ports noble-updates/multiverse arm64 c-n-f Metadata [488 B]
Get:17 http://repo.huaweicloud.com/ubuntu-ports noble-backports/main arm64 Packages [39.1 kB]
Get:18 http://repo.huaweicloud.com/ubuntu-ports noble-backports/main arm64 Components [15.9 kB]
Get:19 http://repo.huaweicloud.com/ubuntu-ports noble-backports/main arm64 c-n-f Metadata [272 B]
Get:20 http://repo.huaweicloud.com/ubuntu-ports noble-backports/restricted arm64 Packages [1,624 kB]
Get:21 http://repo.huaweicloud.com/ubuntu-ports noble-backports/restricted Translation-en [175 kB]
Get:22 http://repo.huaweicloud.com/ubuntu-ports noble-backports/restricted arm64 Components [212 B]
Get:23 http://repo.huaweicloud.com/ubuntu-ports noble-backports/restricted arm64 c-n-f Metadata [416 B]
Get:24 http://repo.huaweicloud.com/ubuntu-ports noble-backports/universe arm64 Packages [2,405 kB]
Get:25 http://repo.huaweicloud.com/ubuntu-ports noble-backports/universe arm64 Components [266 kB]
Get:26 http://repo.huaweicloud.com/ubuntu-ports noble-backports/universe arm64 c-n-f Metadata [16.9 kB]
Get:27 http://repo.huaweicloud.com/ubuntu-ports noble-backports/multiverse arm64 Packages [16.9 kB]
Get:28 http://repo.huaweicloud.com/ubuntu-ports noble-backports/multiverse arm64 Components [212 B]
Get:29 http://repo.huaweicloud.com/ubuntu-ports noble-backports/multiverse arm64 c-n-f Metadata [488 B]
Get:30 http://repo.huaweicloud.com/ubuntu-ports noble-security/main arm64 Packages [758 kB]
Get:31 http://repo.huaweicloud.com/ubuntu-ports noble-security/main arm64 Components [143 kB]
Get:32 http://repo.huaweicloud.com/ubuntu-ports noble-security/main arm64 c-n-f Metadata [6,616 B]
Get:33 http://repo.huaweicloud.com/ubuntu-ports noble-security/restricted arm64 Packages [1,624 kB]
Get:34 http://repo.huaweicloud.com/ubuntu-ports noble-security/restricted Translation-en [175 kB]
Get:35 http://repo.huaweicloud.com/ubuntu-ports noble-security/restricted arm64 Components [212 B]
Get:36 http://repo.huaweicloud.com/ubuntu-ports noble-security/restricted arm64 c-n-f Metadata [416 B]
Get:37 http://repo.huaweicloud.com/ubuntu-ports noble-security/universe arm64 Packages [2,405 kB]
Get:38 http://repo.huaweicloud.com/ubuntu-ports noble-security/universe arm64 Components [266 kB]
Get:39 http://repo.huaweicloud.com/ubuntu-ports noble-security/universe arm64 c-n-f Metadata [16.9 kB]
Get:40 http://repo.huaweicloud.com/ubuntu-ports noble-security/multiverse arm64 Packages [16.9 kB]
Get:41 http://repo.huaweicloud.com/ubuntu-ports noble-security/multiverse arm64 Components [212 B]
Get:42 http://repo.huaweicloud.com/ubuntu-ports noble-security/multiverse arm64 c-n-f Metadata [488 B]
Fetched 9,138 kB in 3s (2,486 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
232 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@121.36.73.106:~#
```



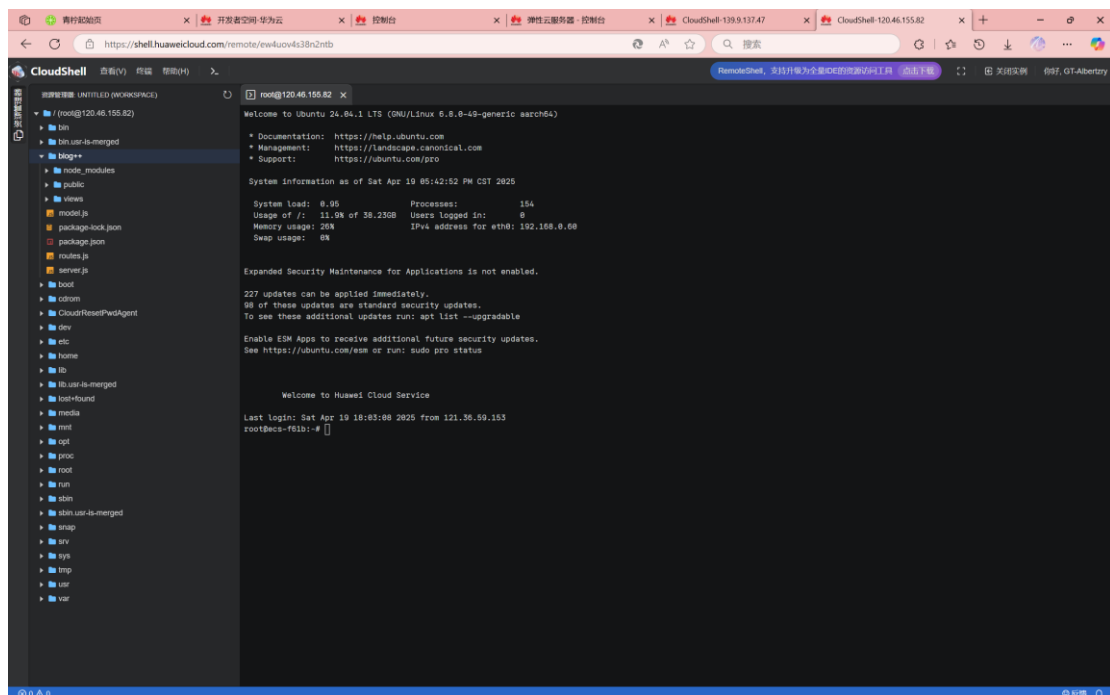
```
root@139.9.137.47:~# cat /etc/docker/daemon.json
{
  "registry-mirrors": [
    "https://docker-0.unsee.tech",
    "https://docker-cf.registry.cyou",
    "https://docker.lpanel.live"
  ]
}

root@139.9.137.47:~# sudo systemctl daemon-reload
root@139.9.137.47:~# sudo systemctl restart docker
root@139.9.137.47:~# docker run --name mongo --network host -d -e MONGO_INITDB_ROOT_USERNAME=yourname -e MONGO_INITDB_ROOT_PASSWORD=yourpassword mongo:7.0 --auth
Unable to find image 'mongo:7.0' locally
7.0: Pulling from library/mongo
7b76c88f23a: Pull complete
9a9688a9f87: Pull complete
4a48f74e623: Pull complete
5b8076ac03f: Pull complete
ced84862e78: Pull complete
1c1a4448a02: Pull complete
8074d45d445: Pull complete
7d586cc49f2: Pull complete
Digest: sha256:c8a362b8839499507ee31a5cc0c6d94dc2ef0b2a5d19b4da64a3698
Status: Downloaded newer image for mongo:7.0
6a8d5a879b5d8d8f9d2787c6f8e99b77a0d38e8e3ab9b4c6a8d518fca2
root@139.9.137.47:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
6a8d5a879b5d   mongo:7.0 "docker-entrypoint.s..." 21 seconds ago    Up 20 seconds    27017/tcp    mongo-0
root@139.9.137.47:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
6a8d5a879b5d   mongo:7.0 "docker-entrypoint.s..." 21 seconds ago    Up 31 seconds    27017/tcp    mongo-0
root@139.9.137.47:~#
```

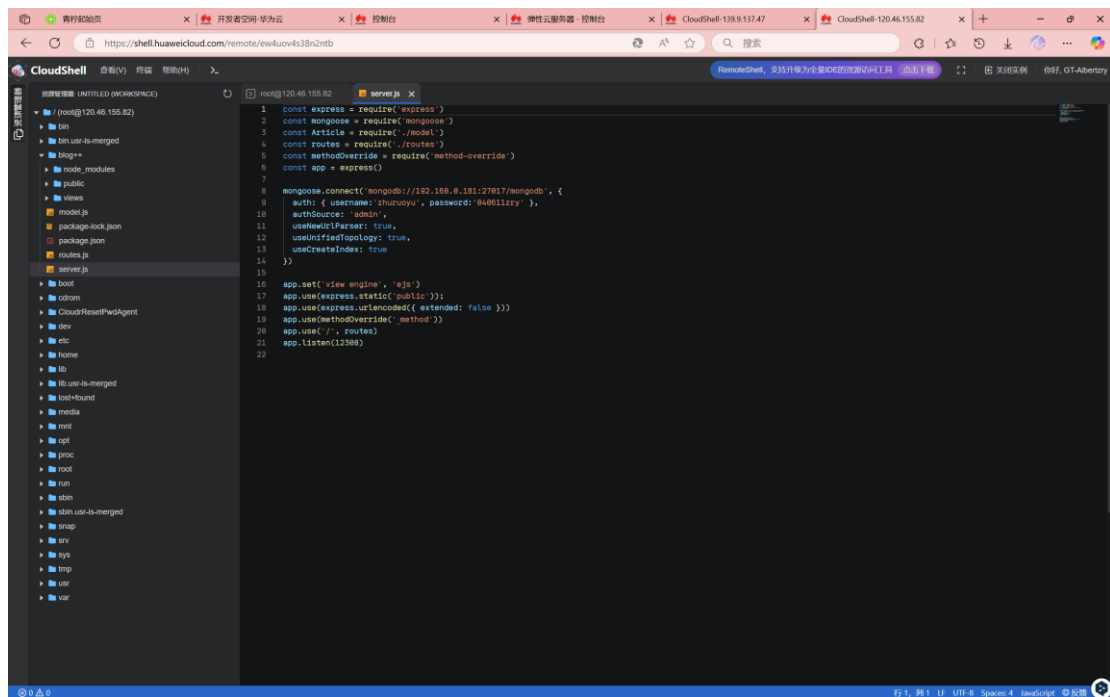
2. 检查数据库运行状态



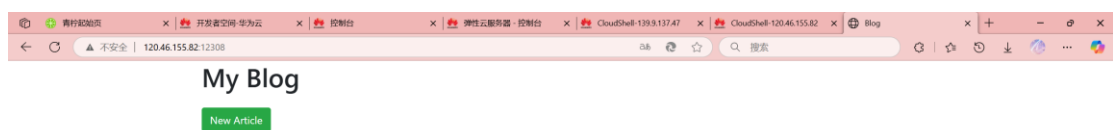
3. 购买另外一台云服务器并将开发完成的 blog++ 项目文件上传至 /root/blog++/ 目录内，使用 npm 安装相关所需模块



4. 修改 server.js 内数据库相关配置信息

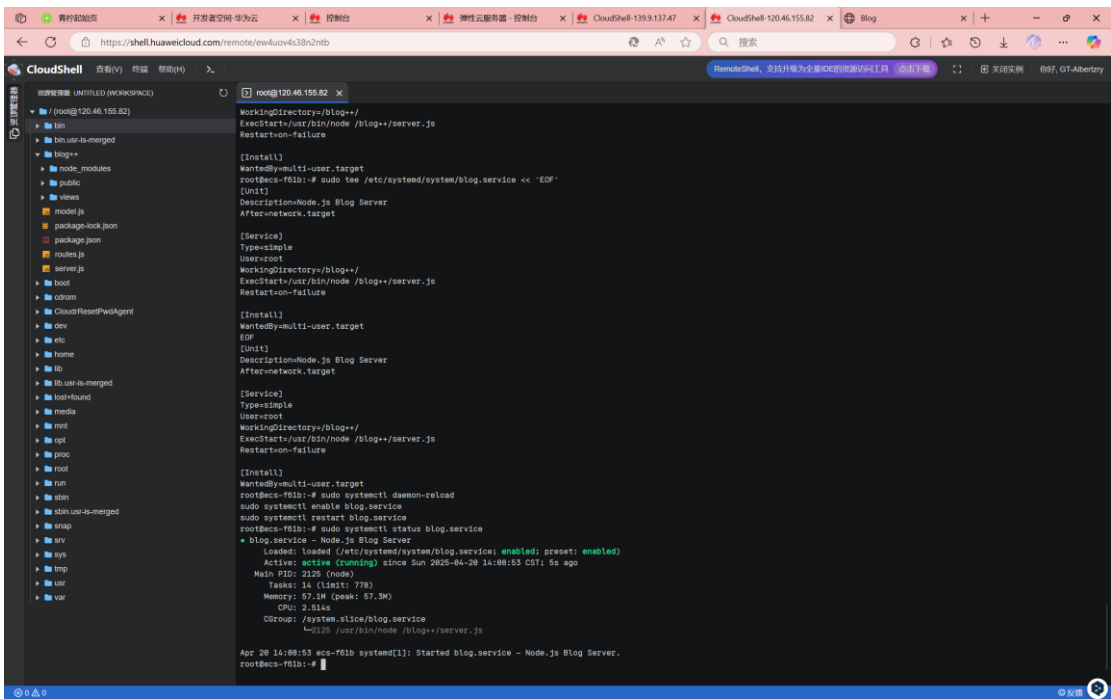


5. 使用 node 运行 server.js，访问该云服务器公网 IP 并测试

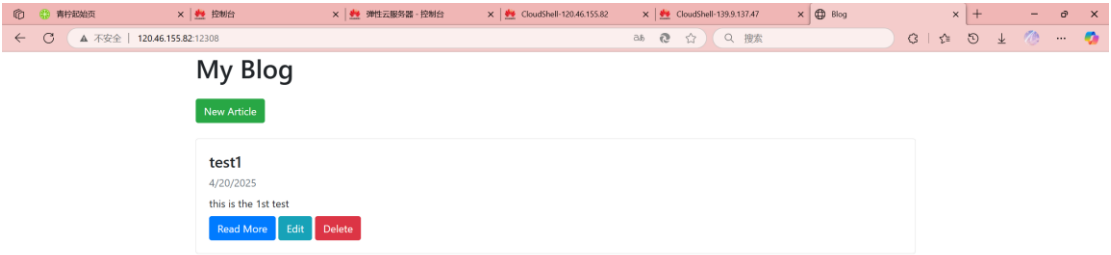
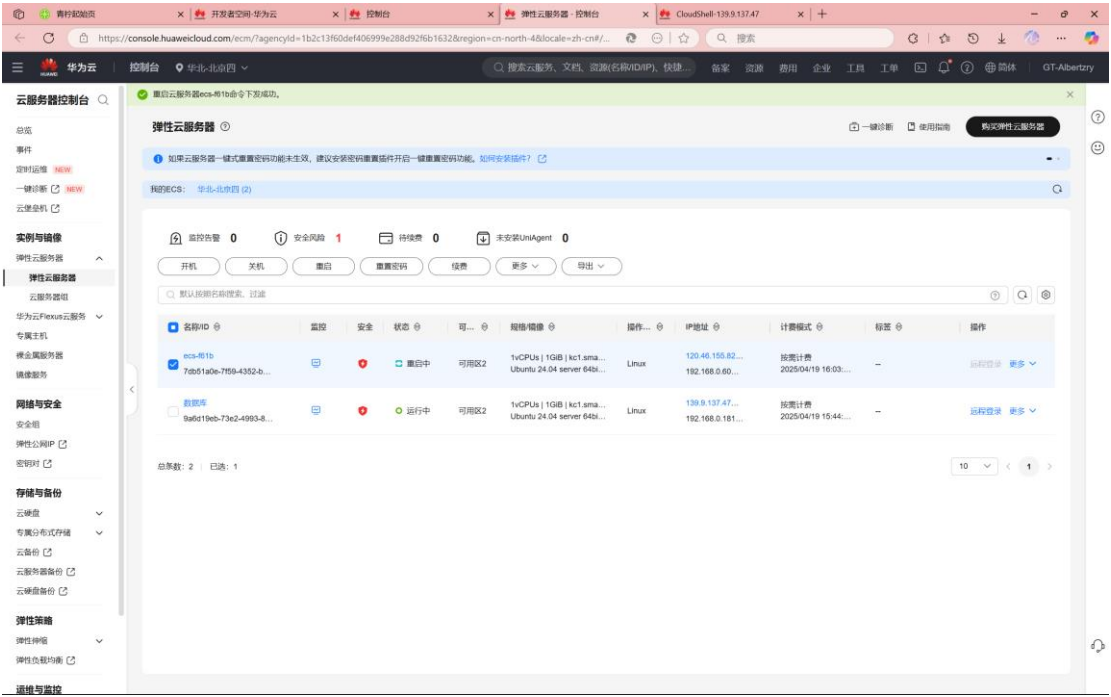




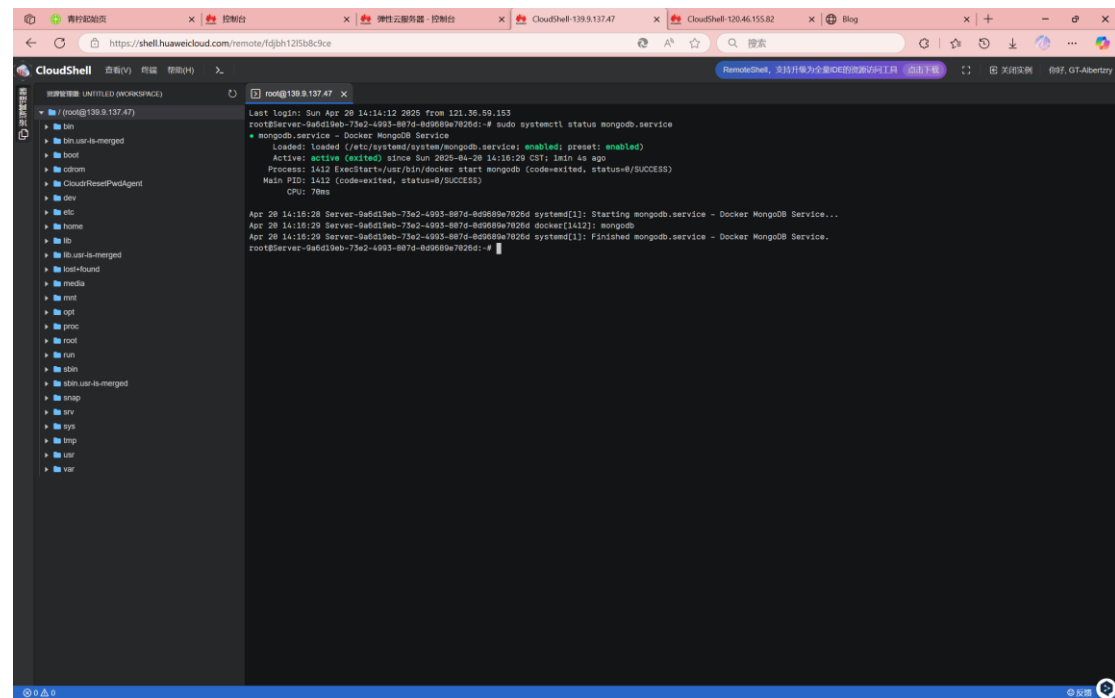
6. 配置开机自启并检查服务状态



7. 重启服务器并直接访问公网 IP 查看是否方位成功

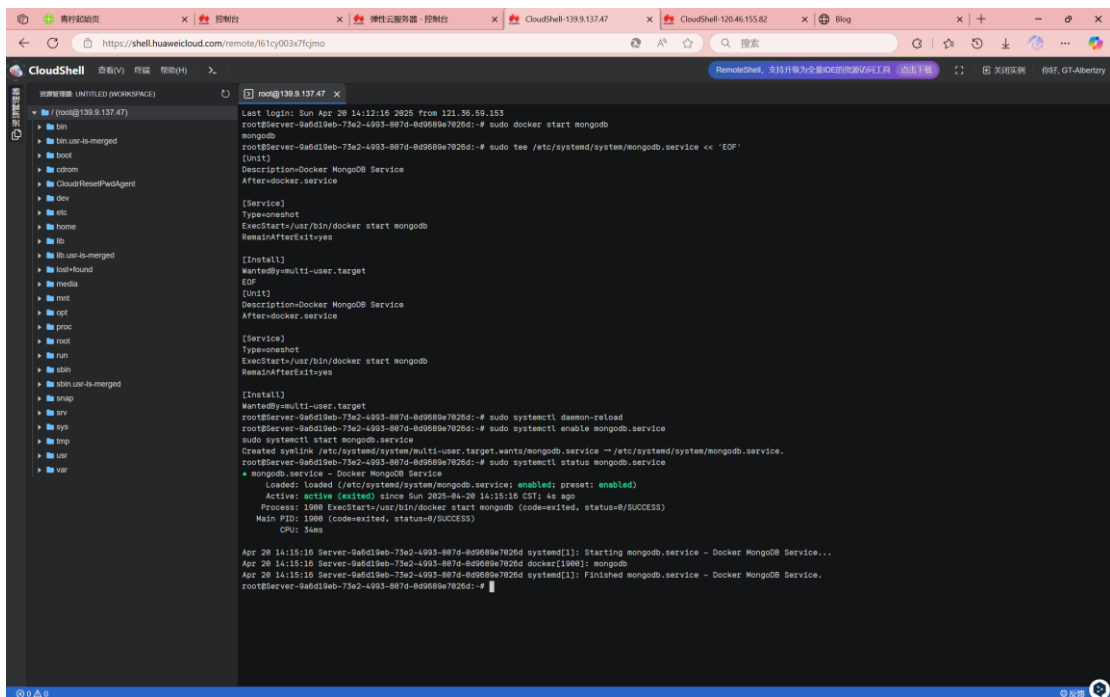


8. 同理配置数据库开机自启动服务



```
root@139.8.137.47:~# sudo systemctl status mongod.service
Last login: Sun Apr 20 14:14:12 2025 from 121.36.59.153
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~# sudo systemctl status mongod.service
● mongod.service - Docker MongoDB Service
   Loaded: loaded (/etc/systemd/system/mongod.service; enabled; preset: enabled)
   Active: active (exited) since Sun 2025-04-20 14:16:29 CST; 1min 4s ago
   Process: 1412 ExecStart=/usr/bin/docker start mongod (code=exited, status=0/SUCCESS)
   Main PID: 1412 (code=exited, status=0/SUCCESS)
   CPU: 7ms

Apr 20 14:16:29 Server-9a5d19eb-75a2-4993-887d-8d9689e7826d systemd[1]: Starting mongod.service - Docker MongoDB Service...
Apr 20 14:16:29 Server-9a5d19eb-75a2-4993-887d-8d9689e7826d docker[1412]: mongod
Apr 20 14:16:29 Server-9a5d19eb-75a2-4993-887d-8d9689e7826d systemd[1]: Finished mongod.service - Docker MongoDB Service.
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~#
```



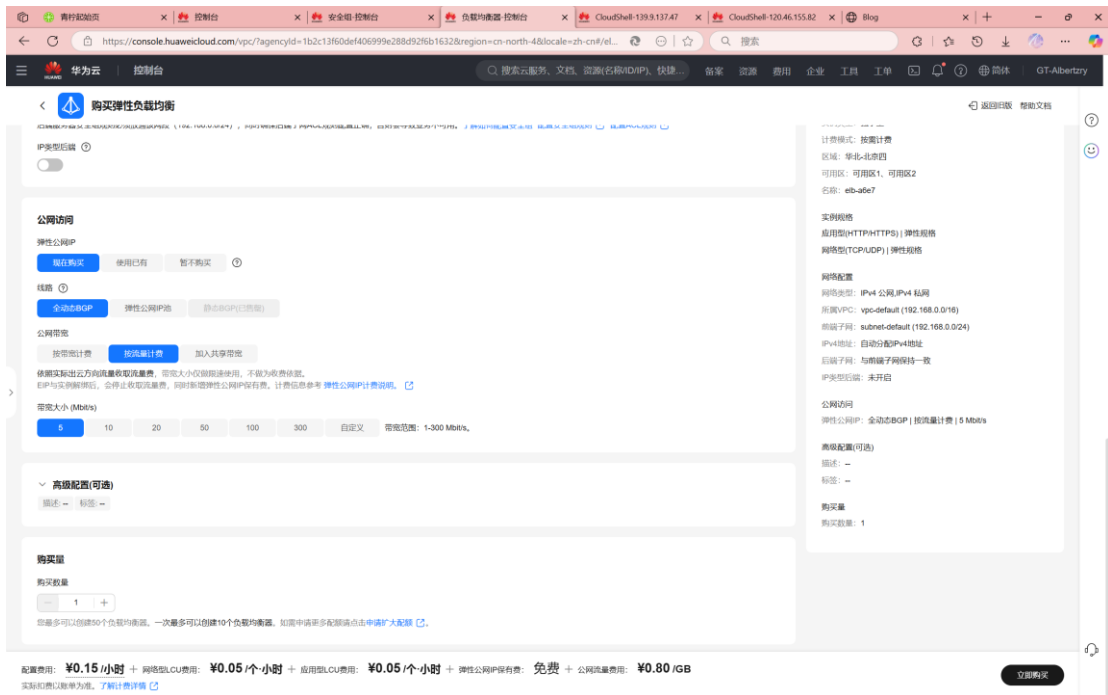
```
root@139.8.137.47:~# sudo docker start mongod
mongod
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~# sudo tee /etc/systemd/system/mongod.service << 'EOF'
[Unit]
Description=Docker MongoDB Service
After=docker.service

[Service]
Type=oneshot
ExecStart=/usr/bin/docker start mongod
RemainAfterExit=yes

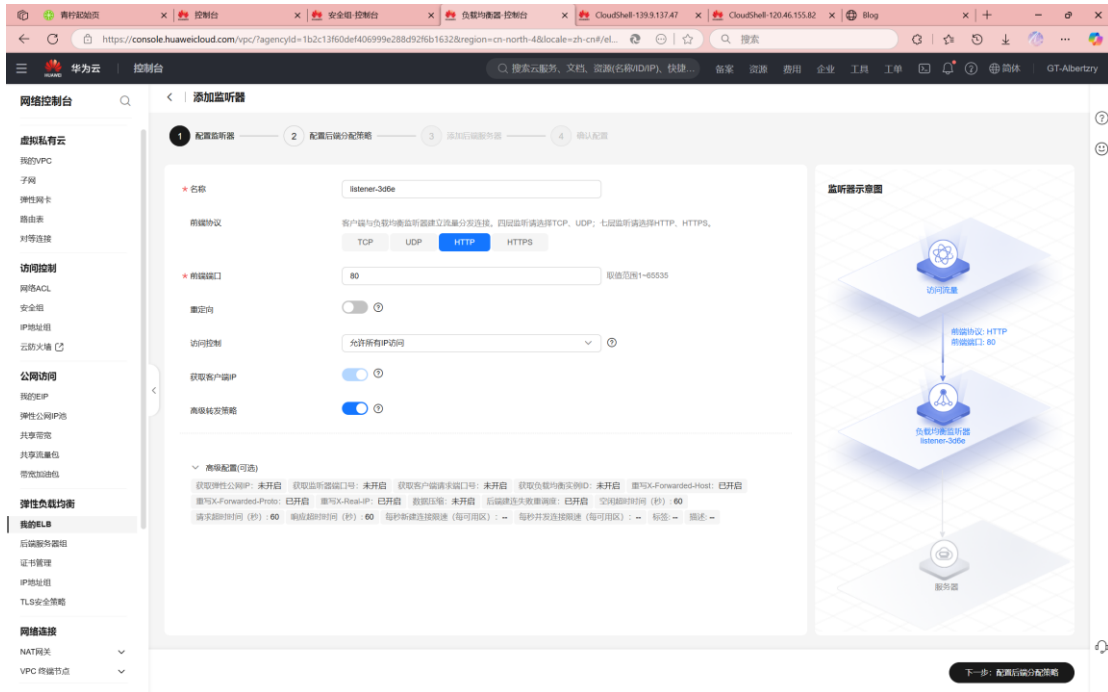
[Install]
WantedBy=multi-user.target
EOF
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~# sudo systemctl daemon-reload
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~# sudo systemctl enable mongod.service
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~# sudo systemctl start mongod.service
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service → /etc/systemd/system/mongod.service.
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~# sudo systemctl status mongod.service
● mongod.service - Docker MongoDB Service
   Loaded: loaded (/etc/systemd/system/mongod.service; enabled; preset: enabled)
   Active: active (exited) since Sun 2025-04-20 14:15:16 CST; 4s ago
   Process: 1988 ExecStart=/usr/bin/docker start mongod (code=exited, status=0/SUCCESS)
   Main PID: 1988 (code=exited, status=0/SUCCESS)
   CPU: 34ms

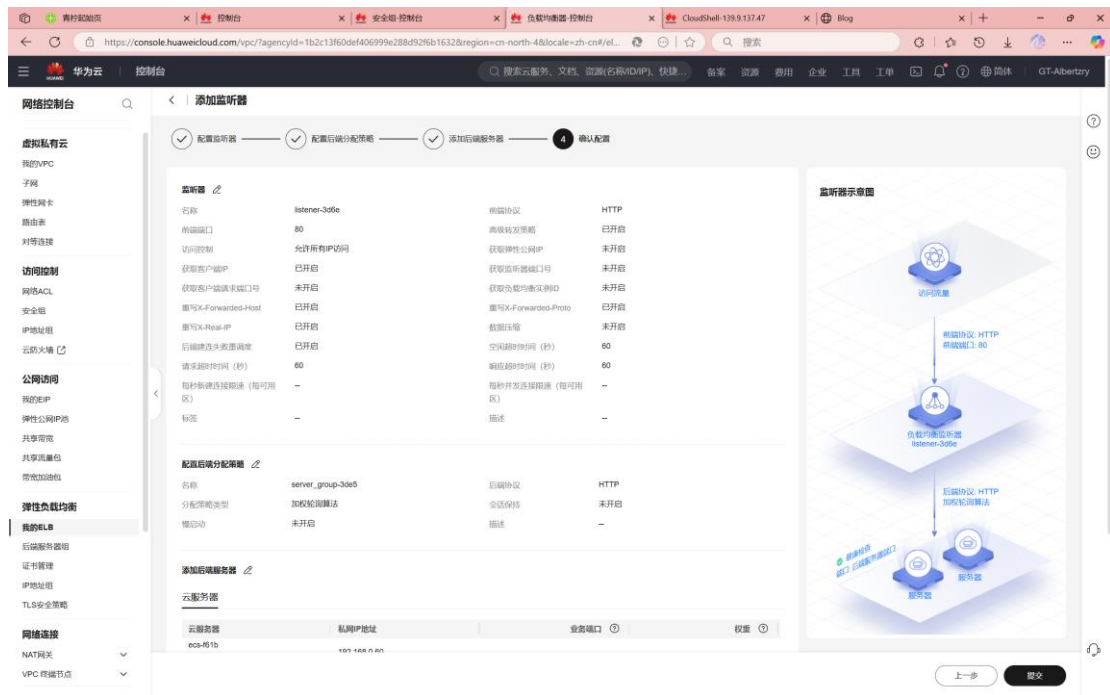
Apr 20 14:15:16 Server-9a5d19eb-75a2-4993-887d-8d9689e7826d systemd[1]: Starting mongod.service - Docker MongoDB Service...
Apr 20 14:15:16 Server-9a5d19eb-75a2-4993-887d-8d9689e7826d docker[1988]: mongod
Apr 20 14:15:16 Server-9a5d19eb-75a2-4993-887d-8d9689e7826d systemd[1]: Finished mongod.service - Docker MongoDB Service.
root@Server-9a5d19eb-75a2-4993-887d-8d9689e7826d:~#
```

9. 购买弹性负载均衡

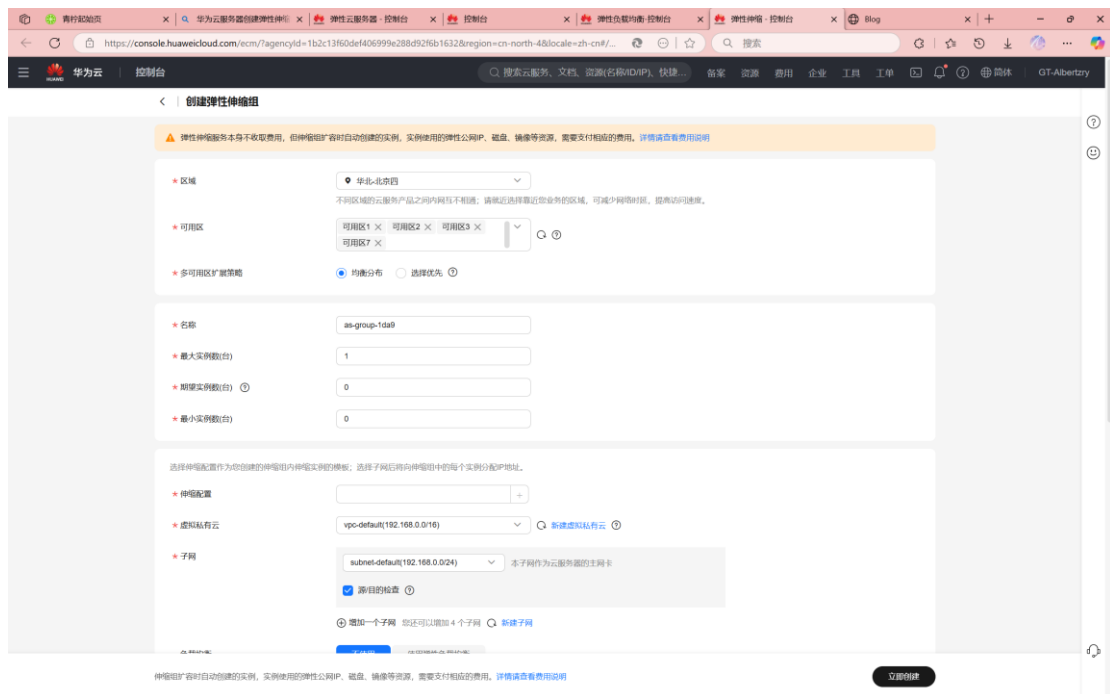


10. 添加监听器

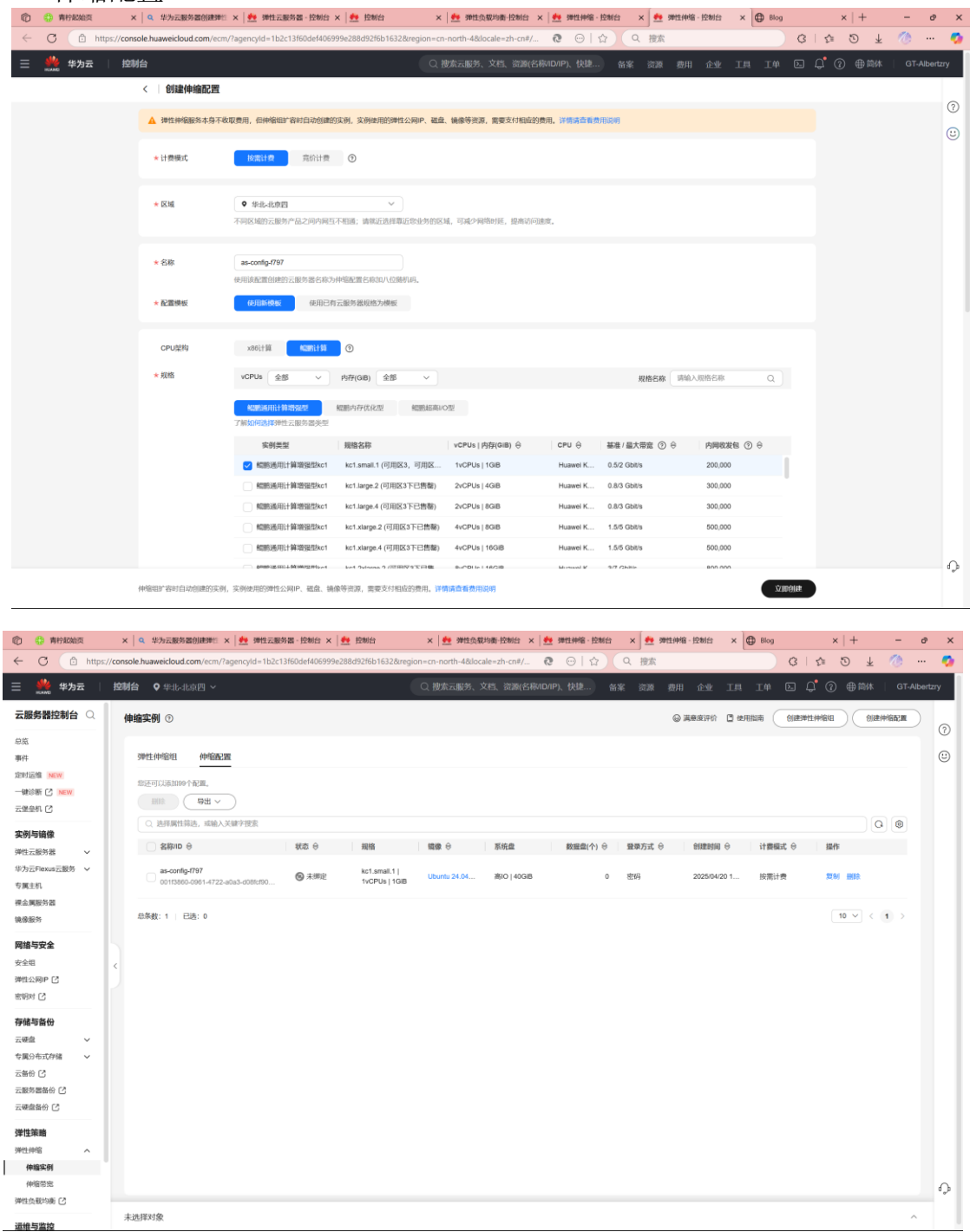




11. 创建弹性伸缩组



12. 伸缩配置



九、数据分析：

成功以公网 IP 通过 12308 端口访问页面并添加博客



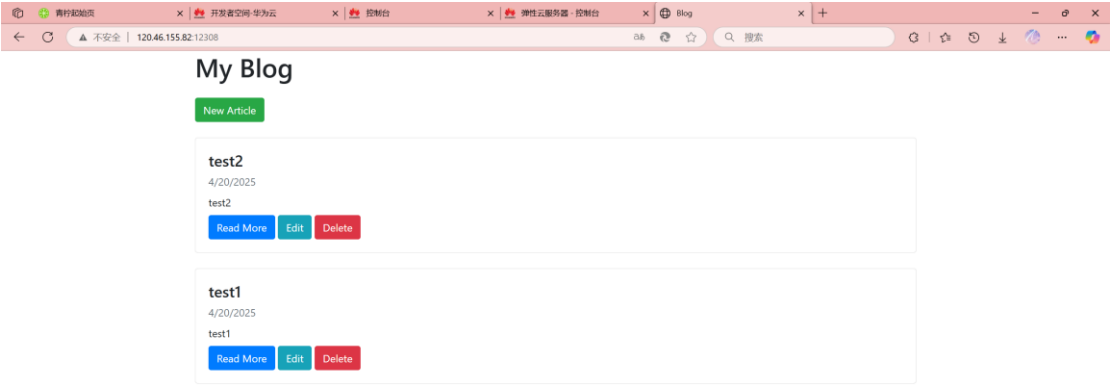
实现博客的修改



博客系统的详情查看



删除博客文章





十、实验结论：

将 blog++部署到云服务器上后，能够通过云服务器公网 IP 成功通过 12308 端口访问到该博客系统并且通过 Mongoddb 正确存储数据，在网页上实现博客的增删改。在完成配置弹性负载均衡（ELB）和配置弹性伸缩（AS）实现了高可用部署。

十一、总结及心得体会：

在本次实验中，我成功开发并部署了一个简单的 blog 应用，同时实现了弹性负载均衡和弹性伸缩功能。通过这个实验，我全面了解了云应用的开发流程，包括前端界面设计、后端逻辑编写和数据库交互，并且掌握了云服务器环境搭建、应用部署以及相关的网络配置技巧。在实现弹性负载均衡的过程中，学会了如何合理分配流量，保障系统的稳定性和可靠性；而弹性伸缩的配置则让我深入理解了如何根据业务负载自动调整资源，以优化成本和性能。此次实验不仅提升了我在云计算和系统运维方面的能力，还让我对实际项目中如何构建高效、可扩展的 Web 应用有了更深刻的认识，为今后处理类似复杂应用场景奠定了坚实的基础。

朱若久