**Huawei Cloud Auto Scaling Demo**

[1. Description](http://dillinger.io/#Description)  
[2. Prerequisites](http://dillinger.io/#Prerequisites)  
[3. Web Application](http://dillinger.io/#WebApplication)  
[4. Auto Scaling Demo](http://dillinger.io/#ASDemo)

**Abstract**：Auto Scaling (AS) is a somewhat ***cool*** feature which is almost the standard of cloud providers. It is a service that automatically adjusts service resources based on your service requirements and configured AS policies.

**1. Description {#Description}**

In this auto scaling demo, I did the following:

* Create a simple **web application** which will print the ECS instance IP address on which it runs
* Create a **private image** with the pre-installed web application and some scripts to generate CPU/memory load to the ECS instance
* Create an **ELB instance** and a listener on it. AS service will cooperate with the ELB to register a newly created ECS instance to it, and de-register a removed ECS instance from it
* Create an **auto scaling group** with 2 alarm policies set:
  + as-policy-cpu-high: If average CPU usage >= 80%, then add 1 instance
  + as-policy-cpu-low: If average CPU usage <= 35%, then reduce 1 instance

The auto scaling demo is a very comprehensive example, since it relates to the following services:

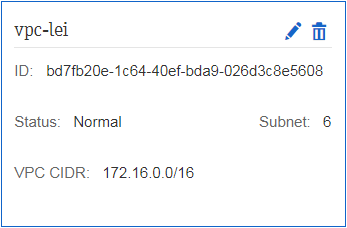
* ECS: Elastic Cloud Server (ECS) instances are added/Removed from AS Group by scaling actions
* IMS: Create a private image through Image Management Service (IMS), with web application deployed, ECS instances are created by it
* CES: If an alarm-triggering policy is configured, AS works together with Cloud Eye Service (CES) to trigger scaling actions in the event that a specified alarm is generated
* ELB: After Elastic Load Balance (ELB) is configured, AS automatically adds ECS instances to or removes ECS instances from the ELB service in a scaling action
* SMN: Simple Message Notification (SMN) can push messages about auto scaling group activities to users so they can get the current status of auto scaling group in a timely manner
* CTS: With Cloud Trace Service (CTS), you can record the activities of the auto scaling events, which can be used for future inquiries, audit and backtracking

**2. Prerequisites {#Prerequisites}**

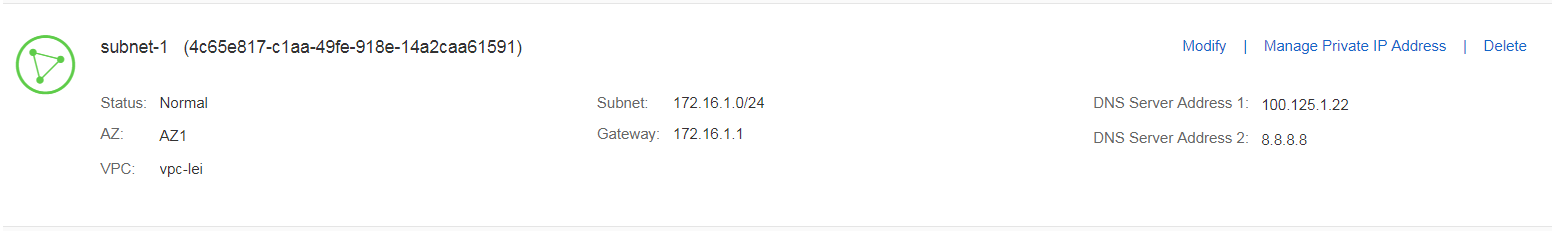
* **Huawei Cloud:** [Huawei Cloud](https://intl.huaweicloud.com/)
* **Prerequisites：** This article is **NOT** a replacement of *User Guide* of a certain service. That is to say, you have to familiar with basic Huawei Cloud operations, such as how to create a private image from an ECS instance, how to create an ELB instance and a listener, etc. If you do not know how to do these, please refer to the **User Guide** of a certain service: [Huawei Cloud Help Center](https://support.huaweicloud.com/intl/en-us/index.html)

The document with the following resources created on Huawei Cloud:

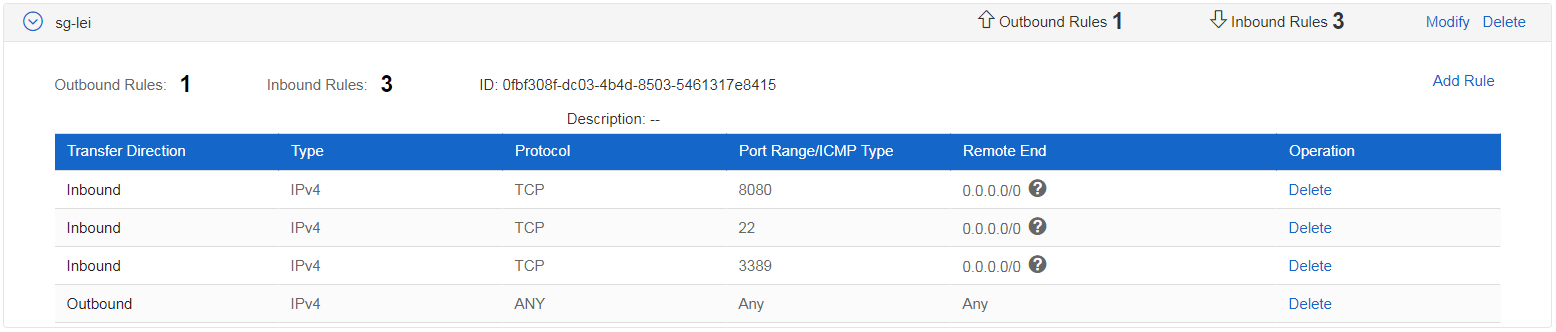
**1. VPC (vpc-demo)**

****

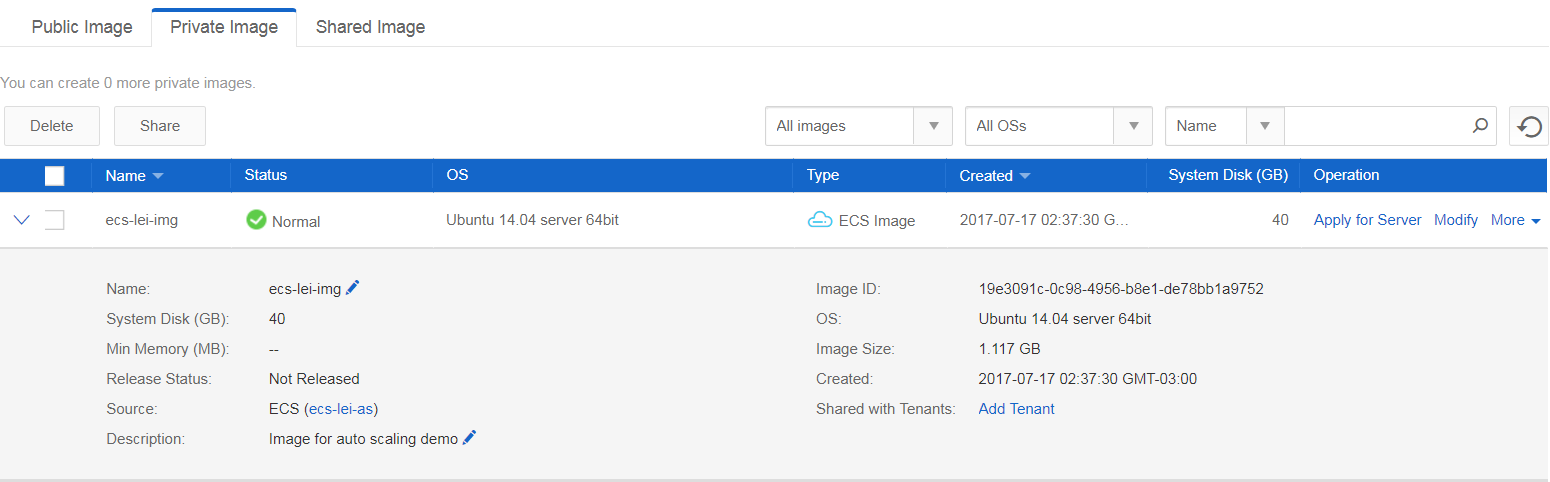
**2. Subnet (subnet-1)**

****

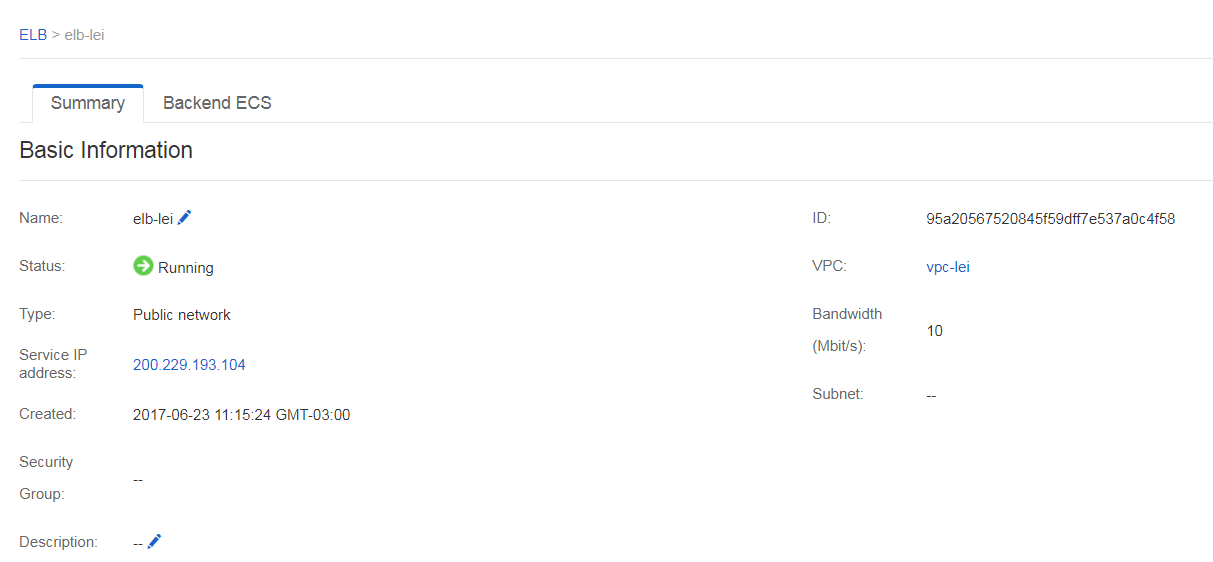
**3. Security Group (sg-demo)**

****

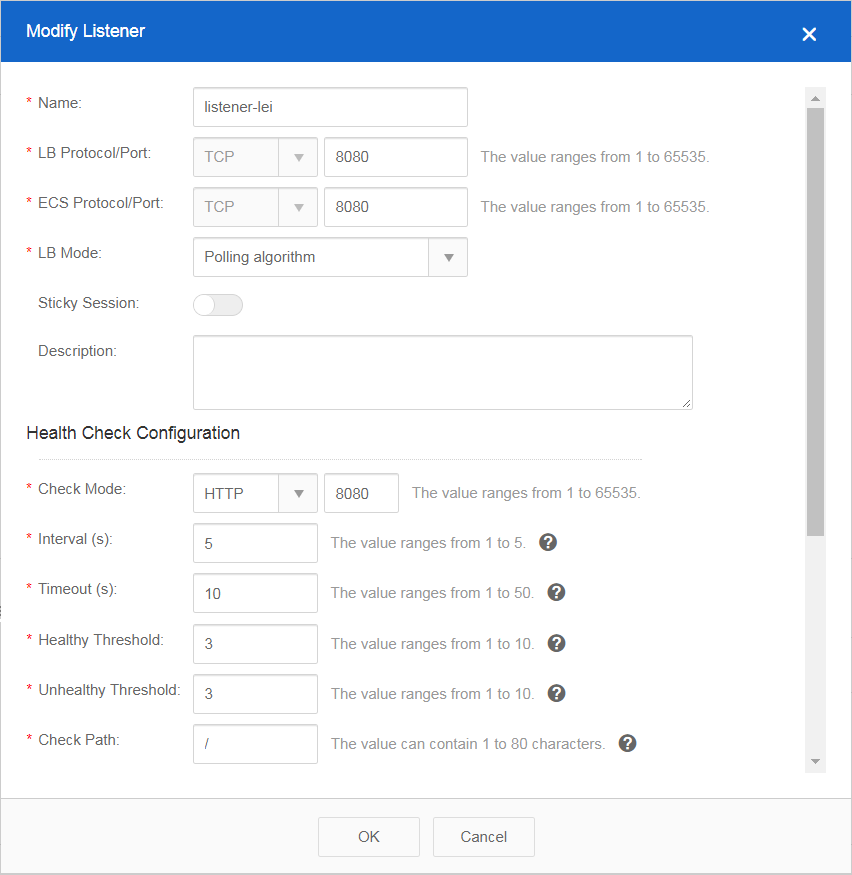
**4. Private Image (ecs-demo-img)**

****

**5. ELB (elb-demo)**

****

**6. ELB Listener (listener-demo)**

****

**3. Web Application {#WebApplication}**

First, please refer to [Install Tomcat 7 on ubuntu 14.04 Server](http://dillinger.io/#http://www.ubuntugeek.com/install-tomcat-7-on-ubuntu-14-04-server.html) to install Tomcat7 on your Ubuntu 14.04 ECS instance.

**NOTE:** In this example, I used **Ubuntu 14.04** and **Tomcat7**. However, it is not a mandatory, you can use your own Operating Sytem and Tomcat version.

Second, deploy the attached as-demo web application under /var/lib/tomcat7/webapps/ROOT, and use service tomcat7 restart to restart your tomcat.

Third, bind an EIP to your ECS instance, and try to access: http://<EIP>:8080/as-demo. If you can see a web page with **Auto Scaling Demo** displayed, congratulations, your web application is successfylly deployed.

Fourth, create a directory scripts under /home/linux and put the scripts files in attached scripts.zip under /home/linux/scripts

Fifth, edit the /etc/rc.local and add the following commands so they will be executed at system startup:

chmod +x /home/linux/scripts/\*.sh

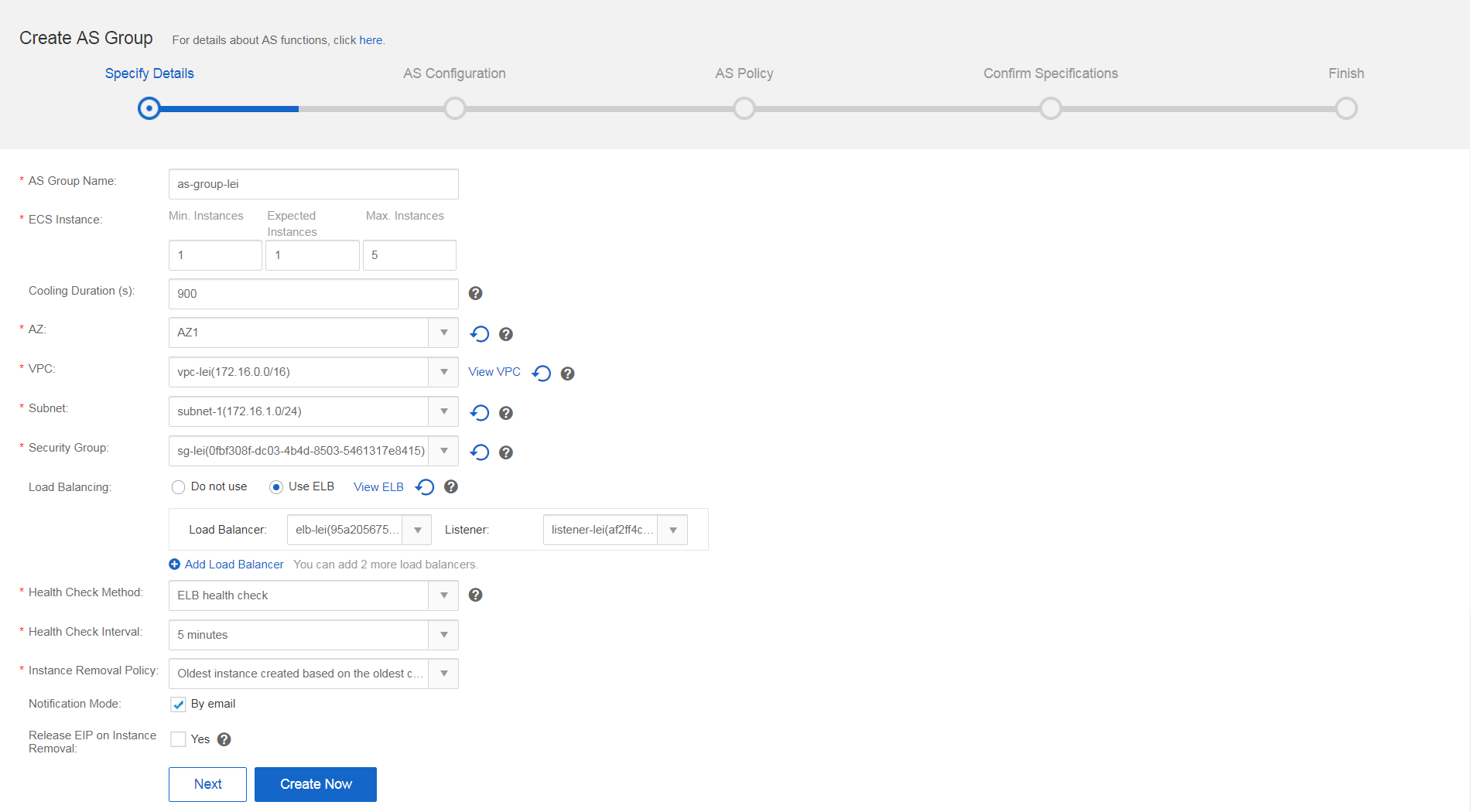
bash /home/linux/scripts/replace\_ipaddress\_index.sh

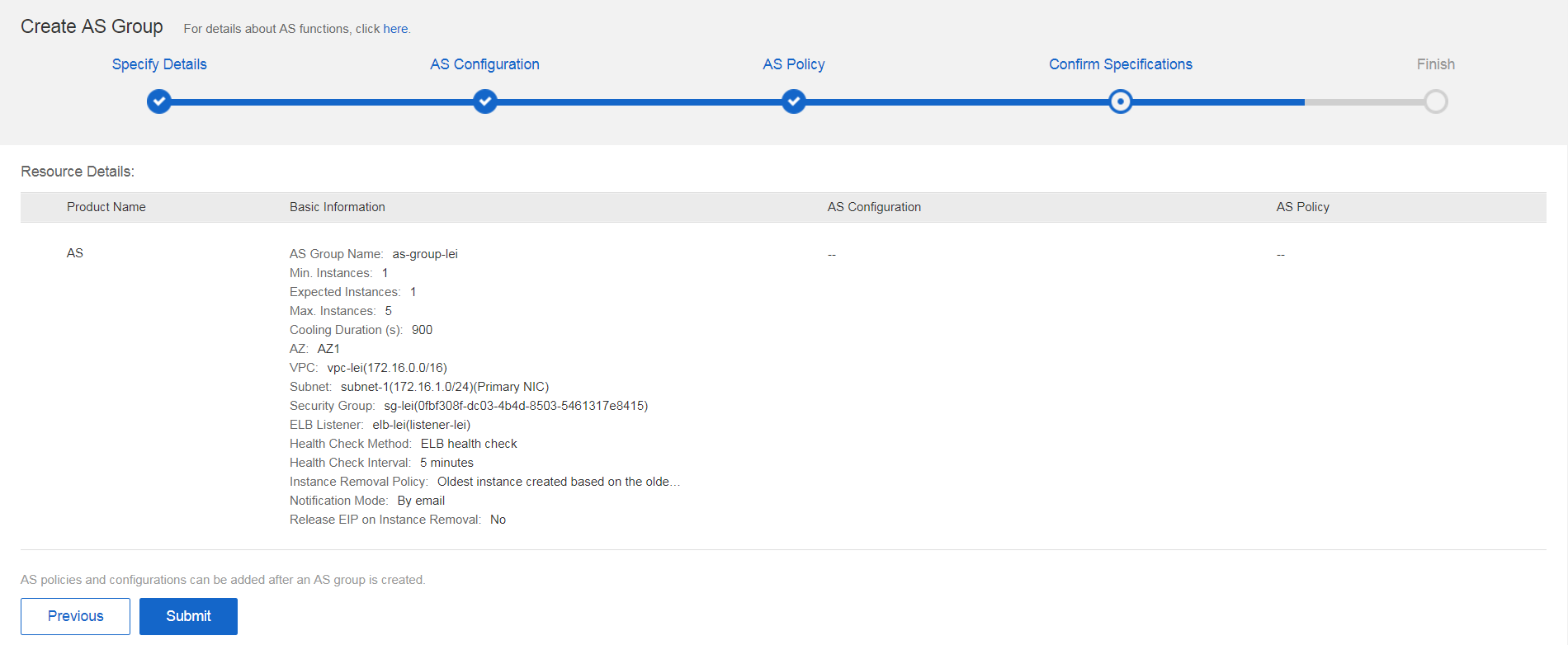
Now, stop the ECS instance, and make a private image from this ECS instance. In this example, I made a private image named ecs-demo-img.

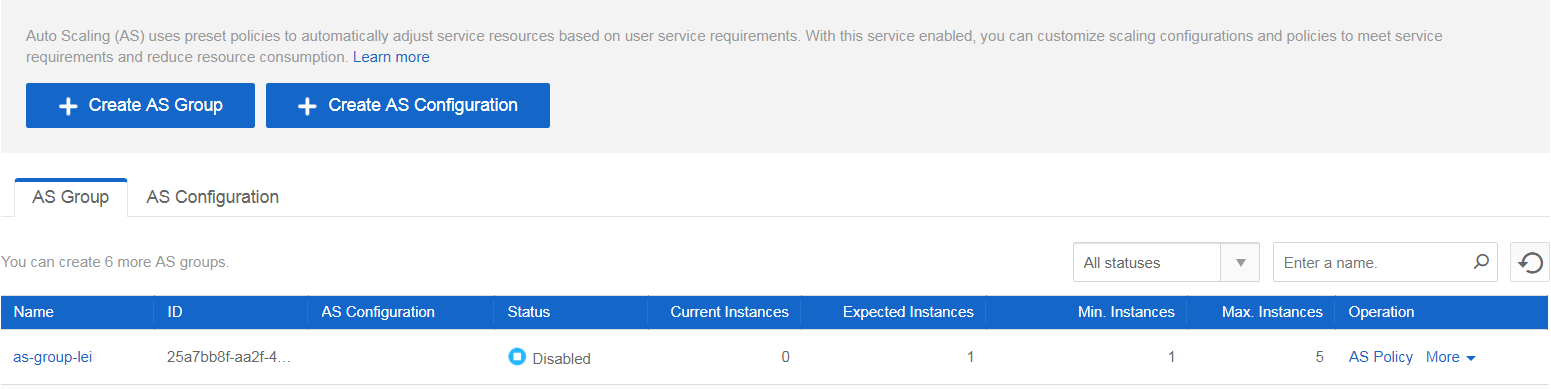
**4. Auto Scaling Demo {#ASDemo}**

**1. Create AS Group**

An AS group consists of a collection of ECS instances and AS policies that have similar attributes and apply to the same application scenario. An AS group is the basis for enabling or disabling AS policies and performing scaling actions. The pre-configured AS policy automatically adds or deletes instances to or from an AS group, or maintains a fixed number of instances in an AS group.

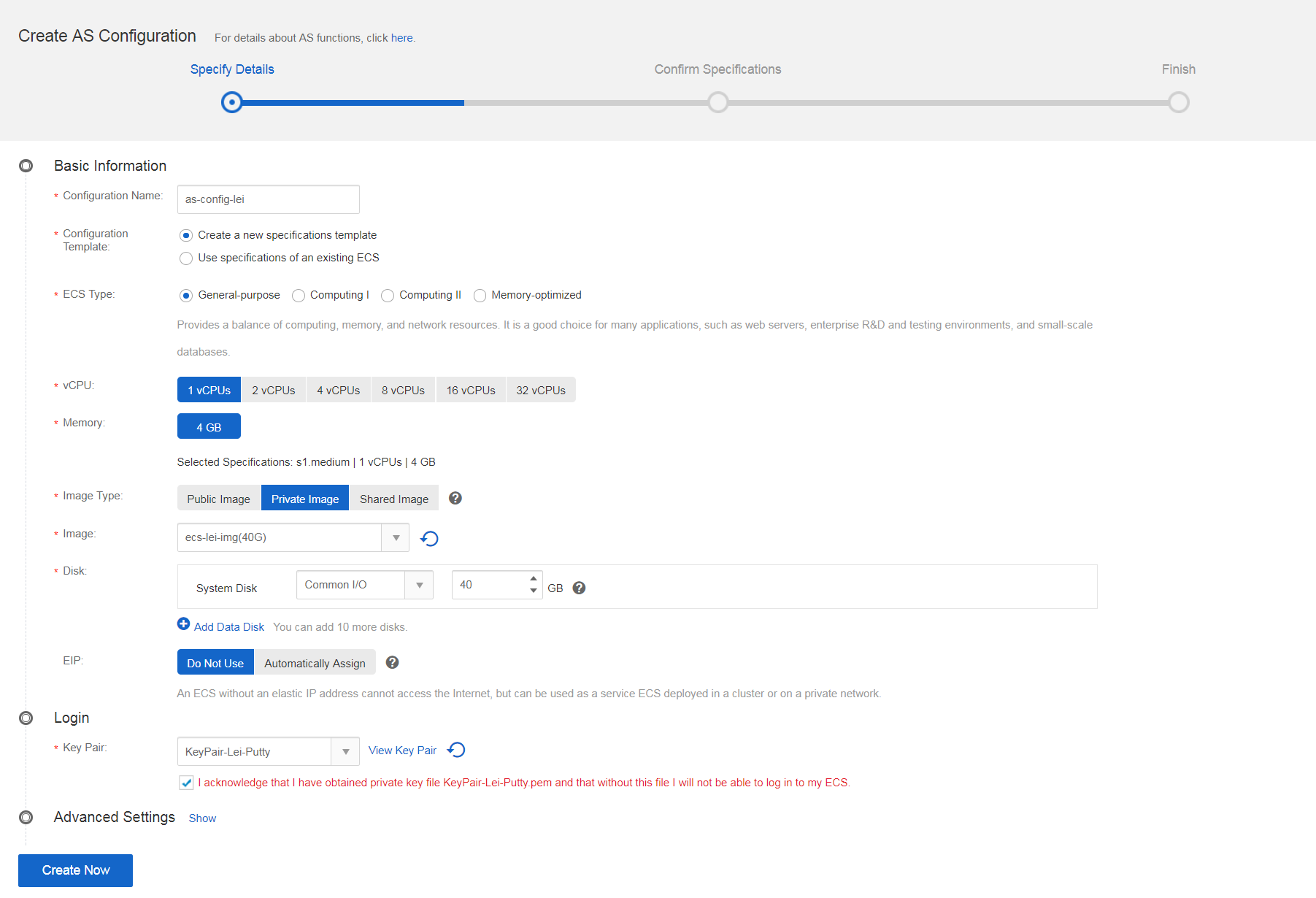
Go to Auto Scaling and create an AS group. I used some information in Prerequisites section, please adjust the parameters according to your own environment.  
**NOTE:** I choose Use ELB in this example, so all the ECS instances created will be added to ELB as backend ECS instances.  


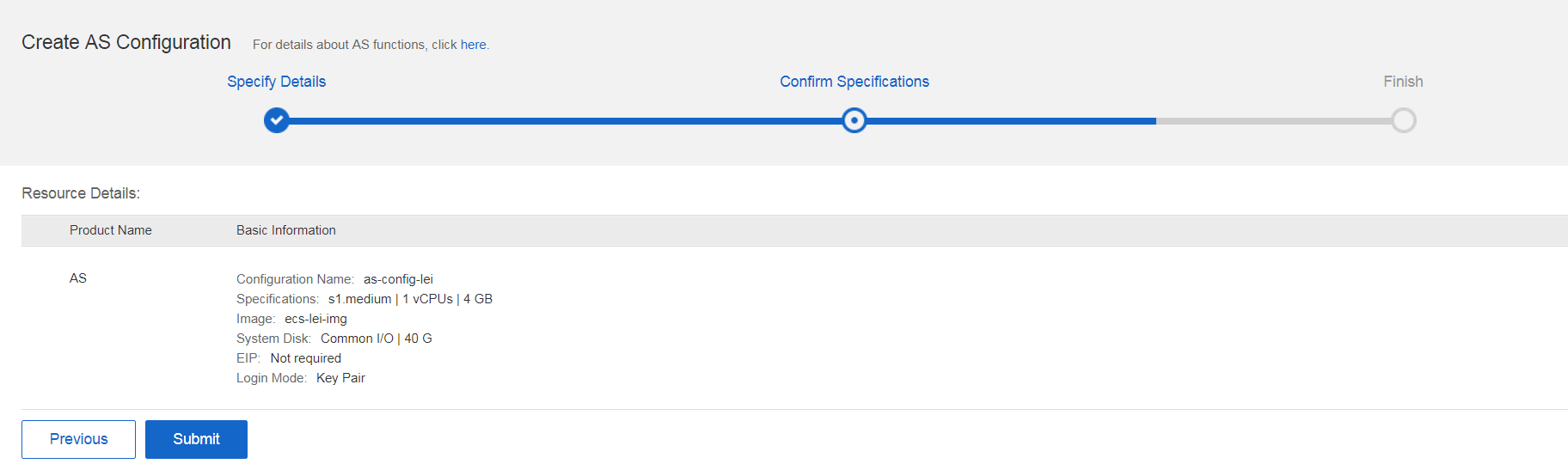
Create AS Group Summary. If everything is correct, click Submit.  


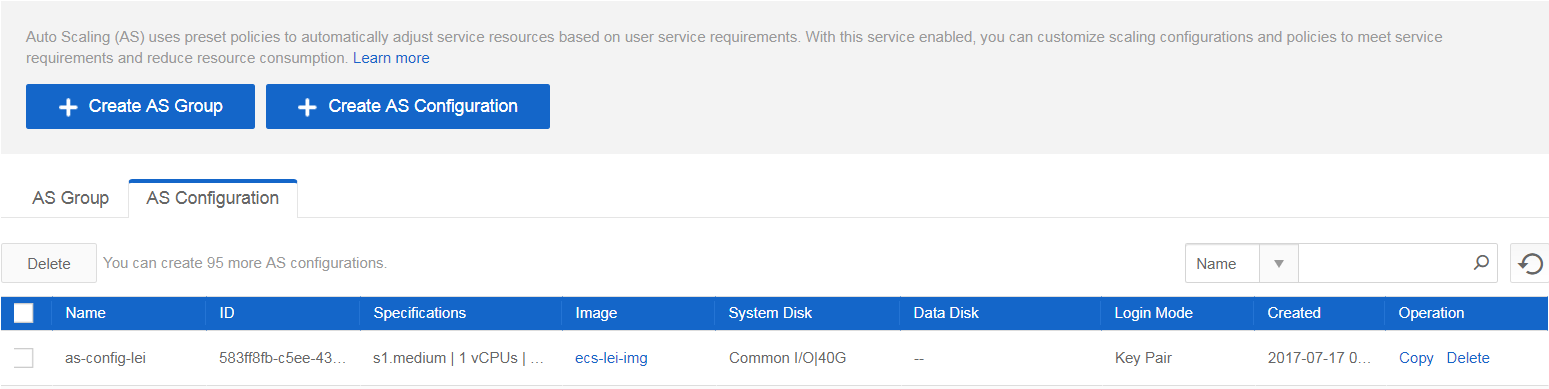
After successful creation, you can see the newly created AS Group in the list, with status is Disabled and Current Instances is 0.  


**2. Create AS Configuration**

An AS configuration specifies the specifications of the ECSs to be added to an AS group.

The most important thing is to choose the Image which is the private image you created.  


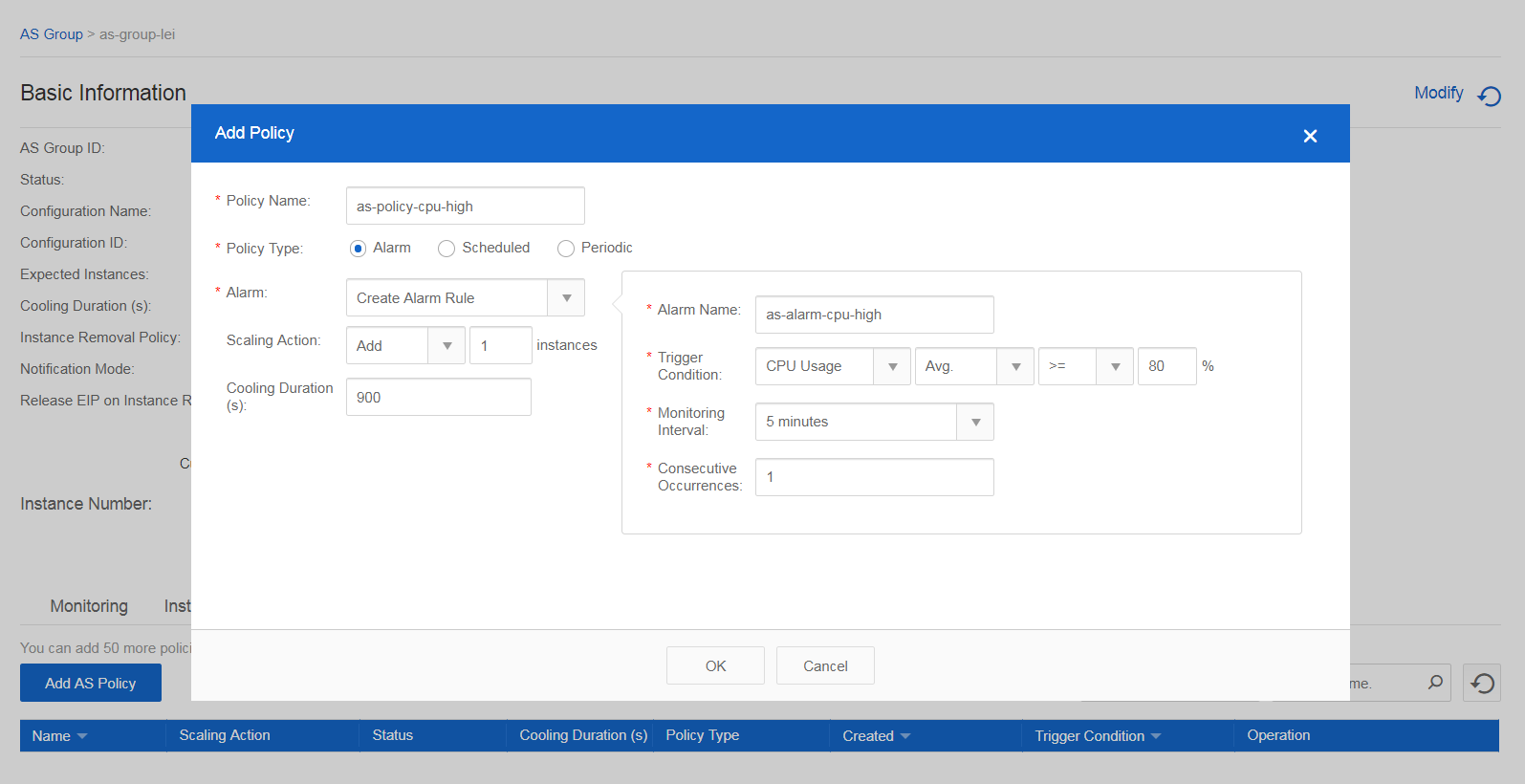
Create AS Group Configuration Summary. If everything is correct, click Submit.  


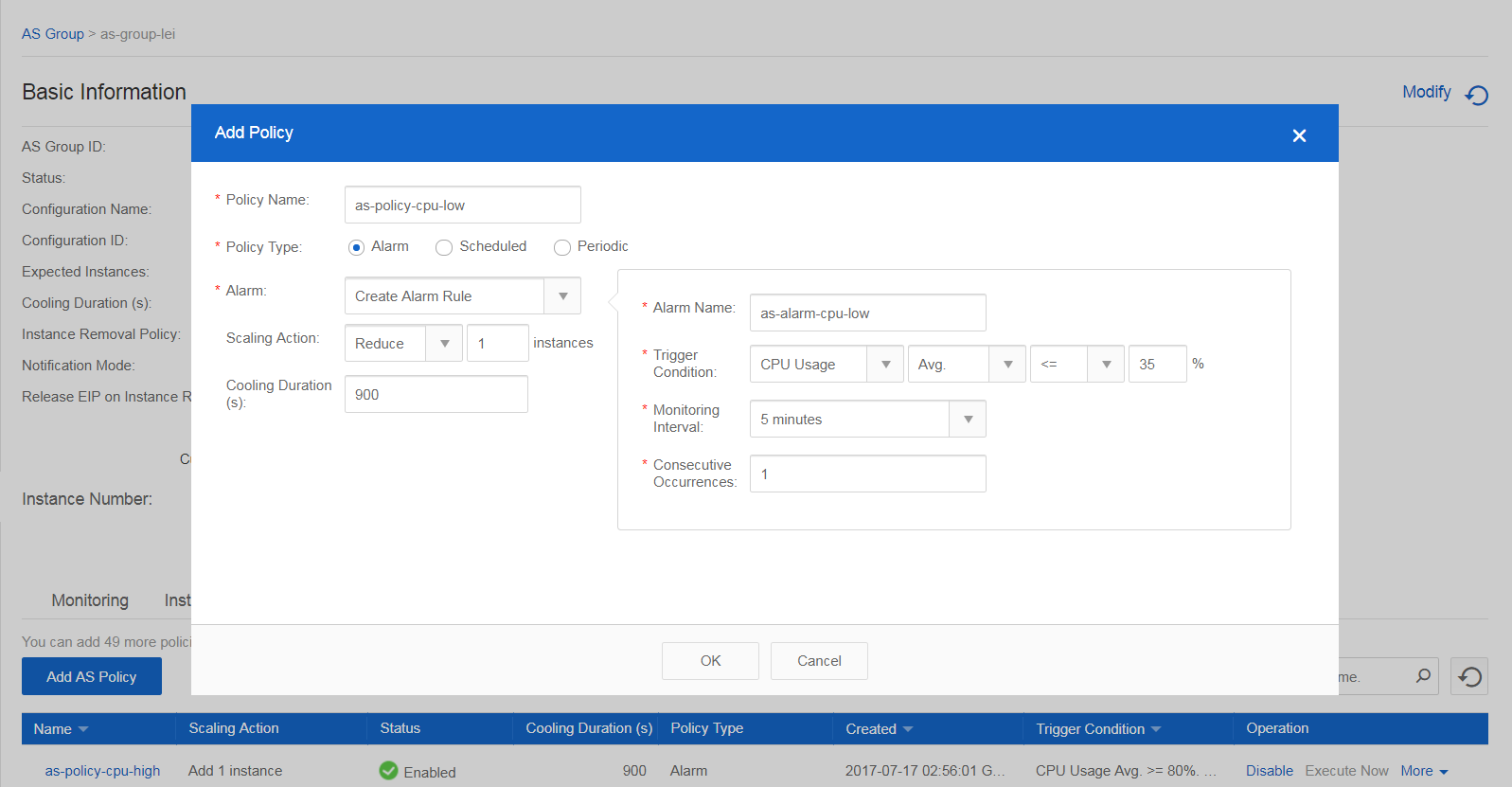
After successful creation, you can see the newly created AS Configuration in the list.  


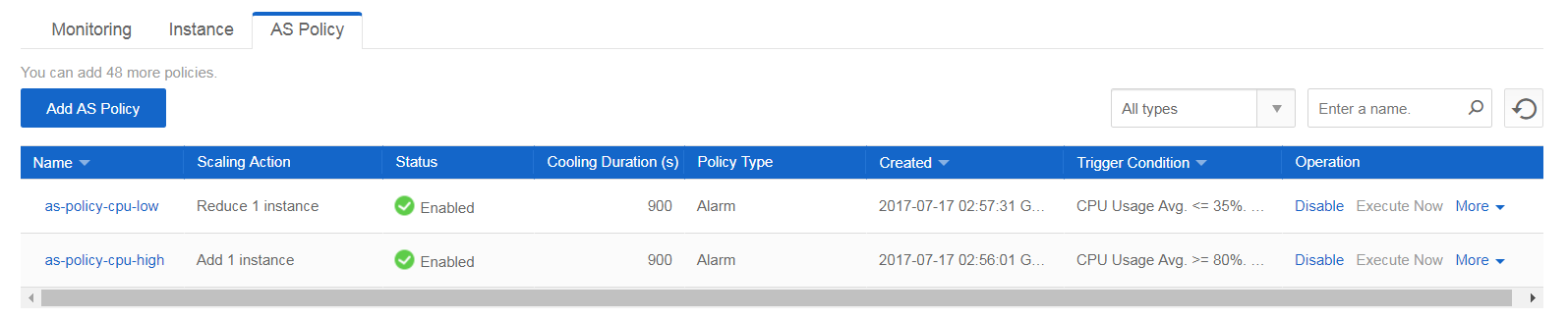
**3. Add Policy**

A scaling policy specifies the conditions for triggering a scaling action as well as the triggered operation. If the conditions are met, a scaling action is triggered to perform the required operation.  
A policy can be of **Alarm**, **Scheduled**, or **Periodic** type.

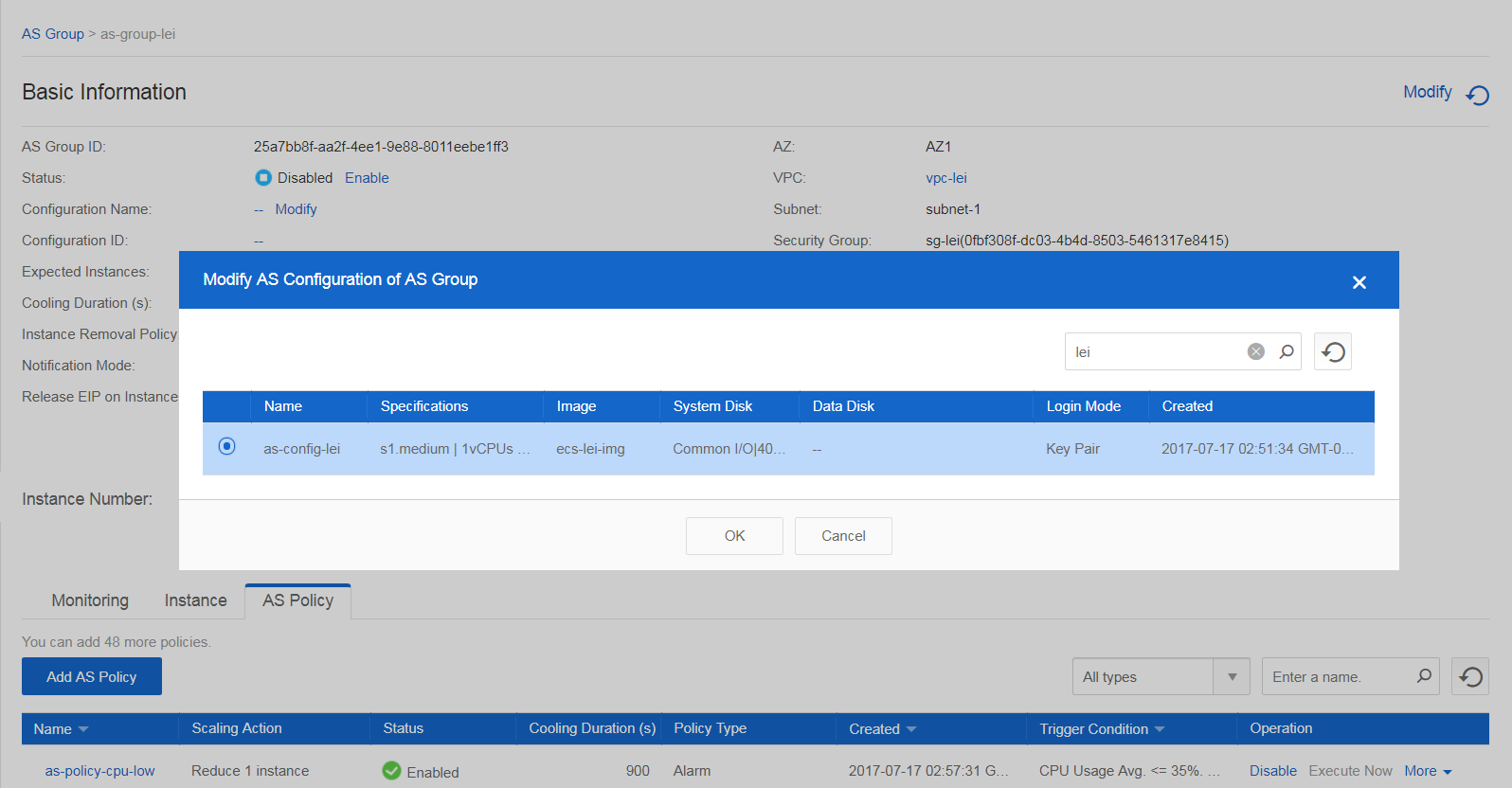
* **Alarm**: AS automatically increases or decreases the number of ECS instances in an AS  
  group or sets the number of ECS instances to a specified value if Cloud Eye (CES)  
  generates an alarm for a configured metric, such as CPU usage.
* **Periodic**: AS increases or decreases the number of ECS instances in an AS group or sets the number of ECS instances to a specified value at a configured interval, such as one  
  day, one week, or month.
* **Scheduled**: AS automatically increases or decreases the number of ECS instances in an  
  AS group or sets the number of ECS instances to a specified value at a specified time

Policy as-policy-cpu-high tells the AS Group: If the average CPU Usage of AS Group is higher or equal 80%, then add 1 instance  


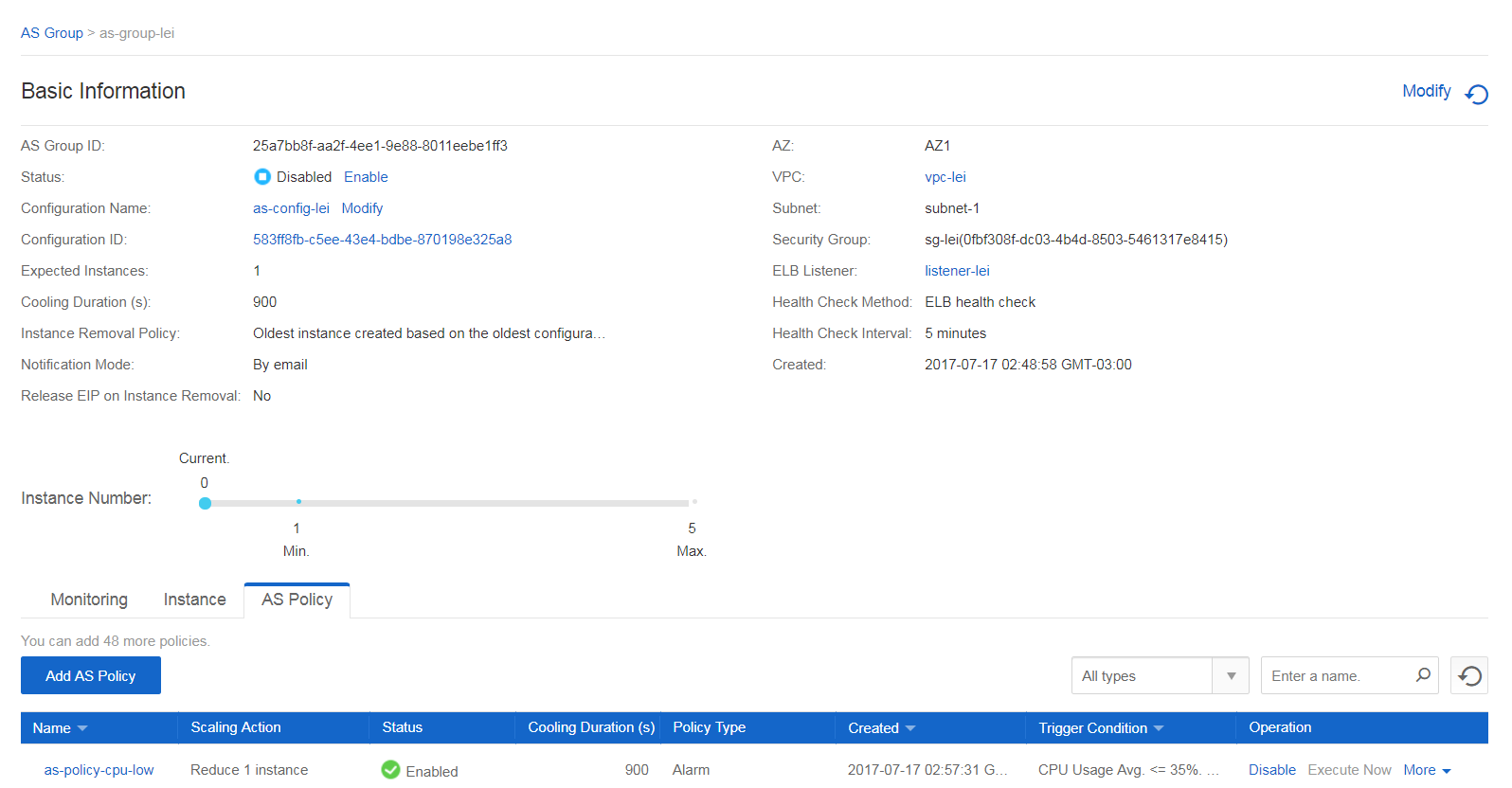
Policy as-policy-cpu-low tells the AS Group: If the average CPU Usage of AS Group is lower or equal 80%, then reduce 1 instance  


The following policies are created.  
**NOTE:** In this example I used CPU Usage, you can created memory usage alarm to trigger scaling action.  


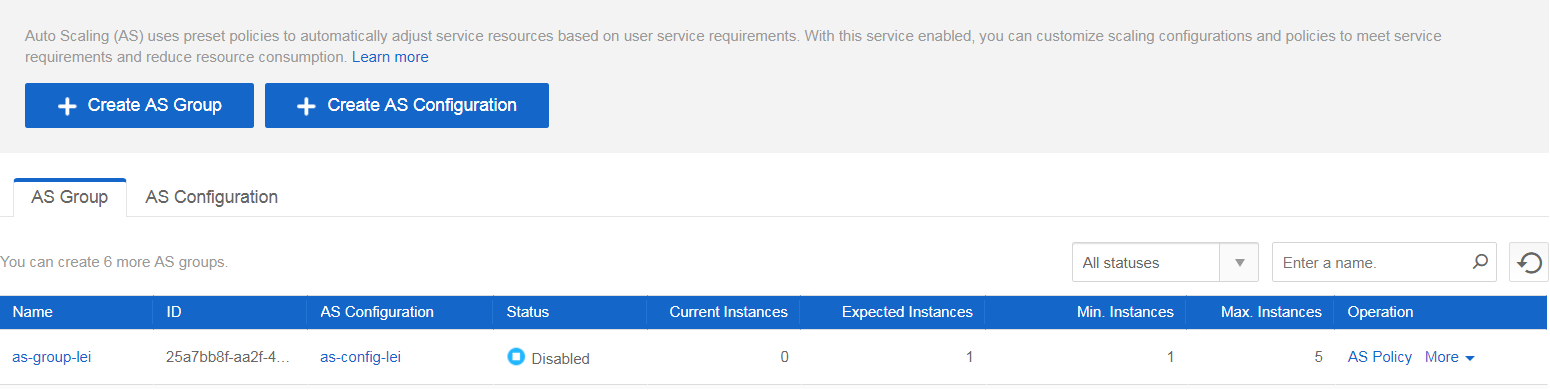
**4. Modify Configuration of AS Group**

Now the Configuration Name is empty for the AS Group, please click Modify next to Configuration Name and choose the AS Configuration as-config-demo to relate it to AS Group as-group-demo.  


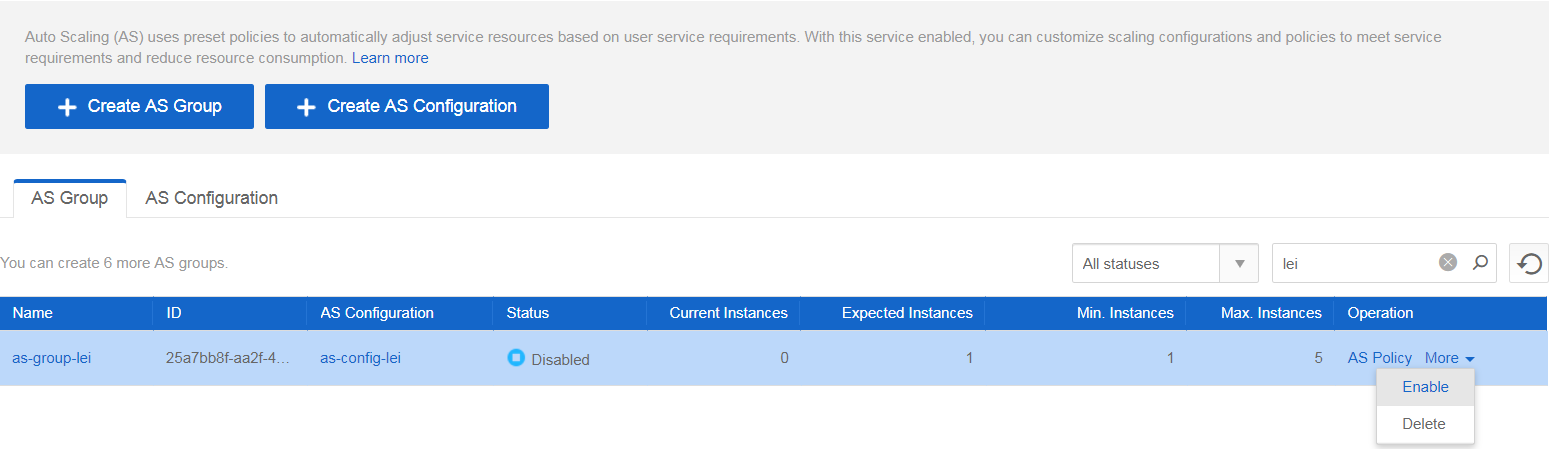
AS Group basic information:



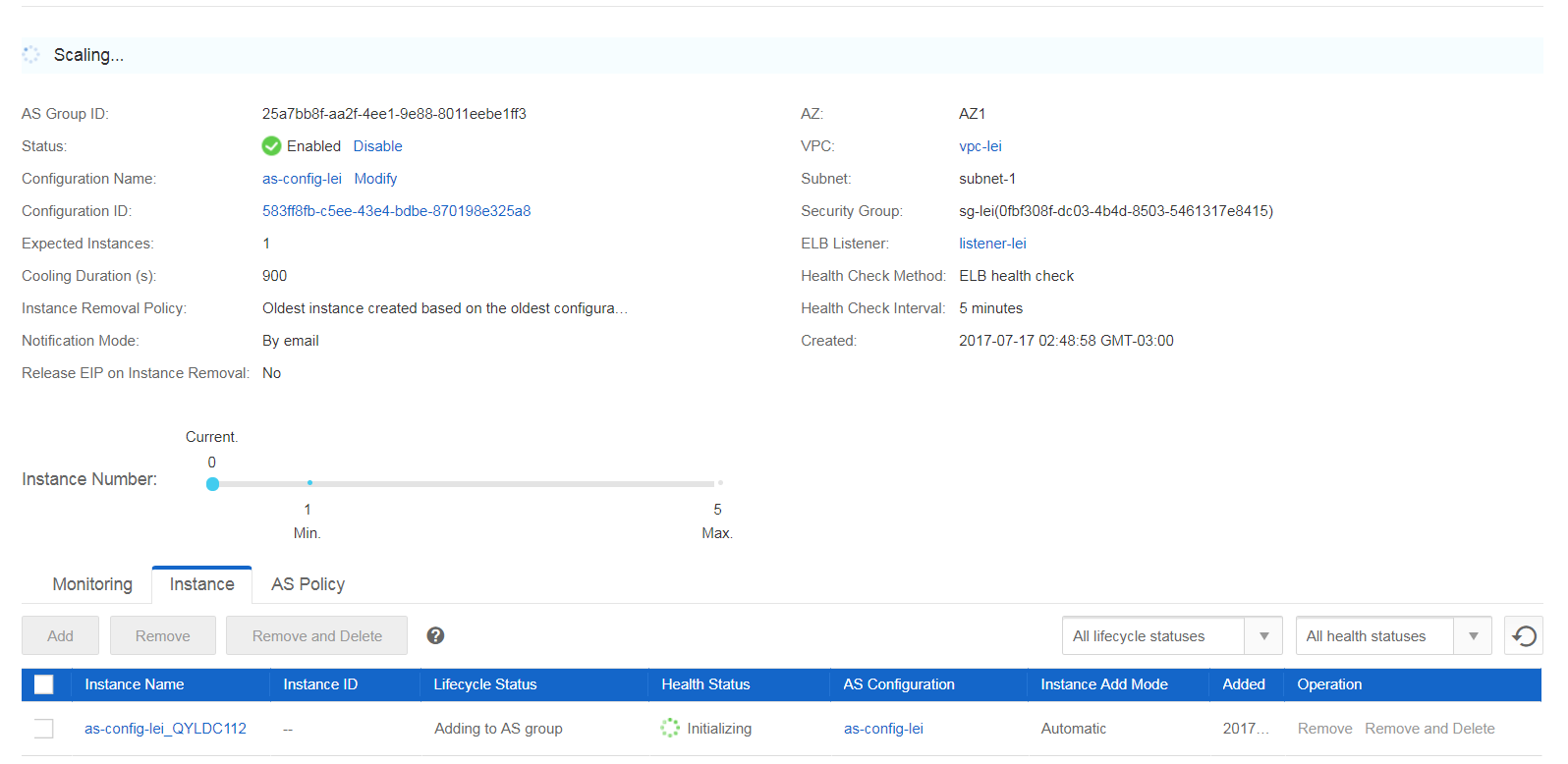
**5. AS Group Summary**

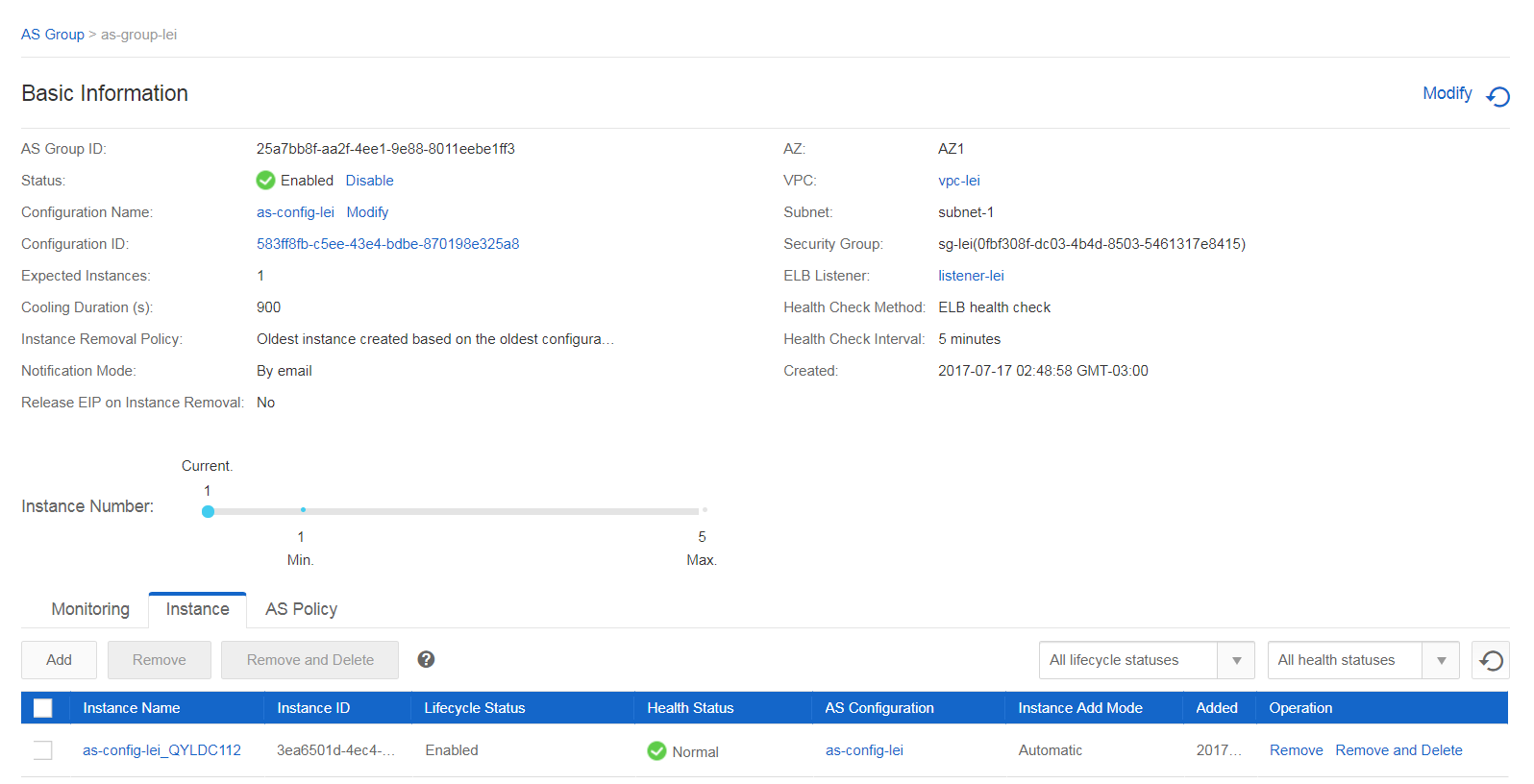
Go back to AS Group you can see as-group-demo is related to as-config-demo.  


**6. Enable AS Group**

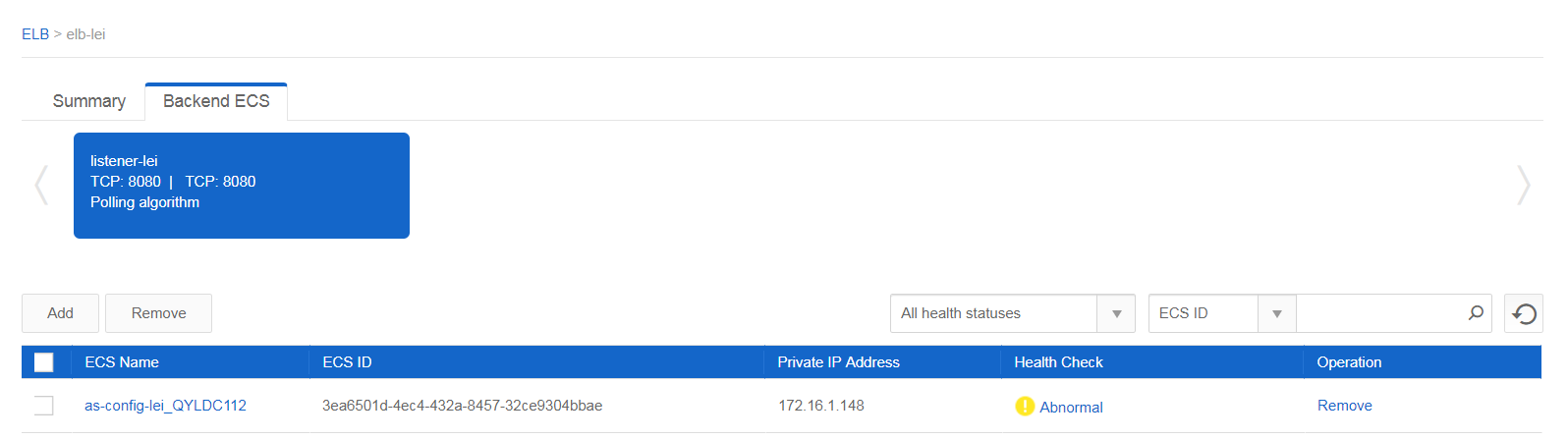
The current status of AS Group as-group-demo is Disabled, please click Enable in the Operation column to enable it.  


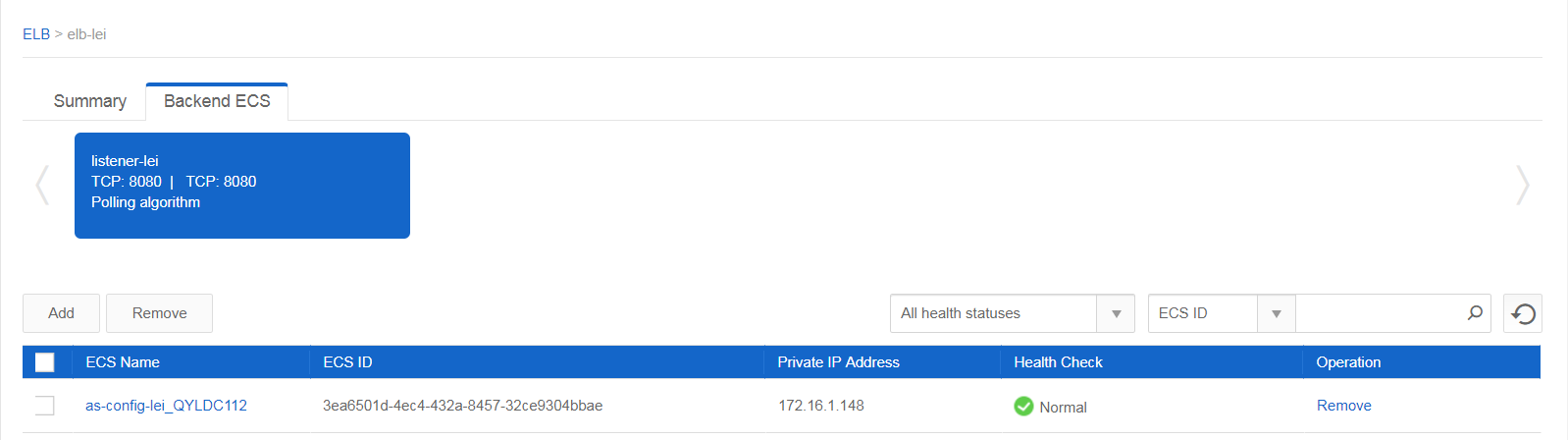
**7. AS Group Instance**

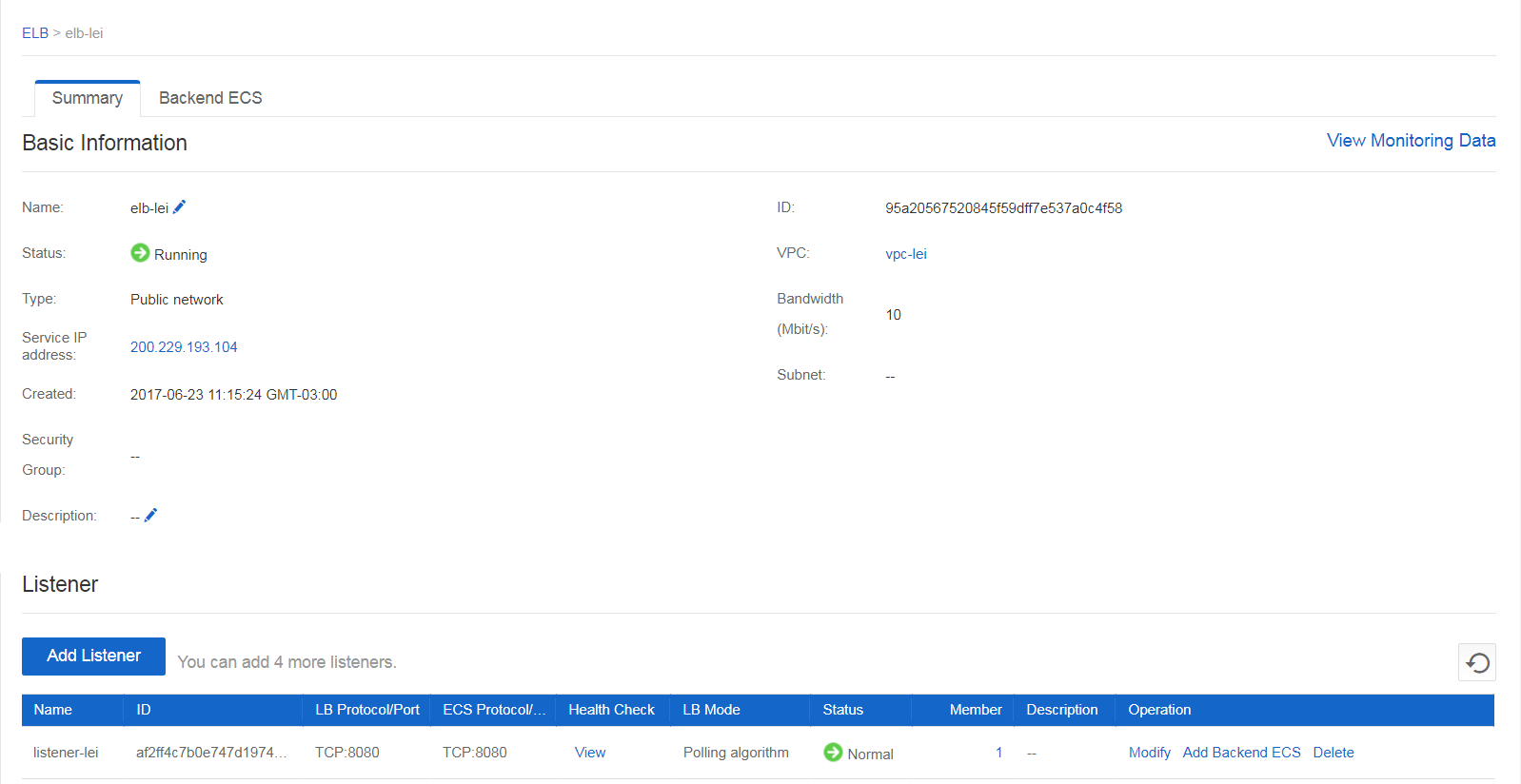
Click AS Group name as-group-demo, choose Instance tab, you can see an instance is in Initializing status.  


Wait for some minutes, the instance status will be Normal.  


**8. ELB Backend ECS Instance**

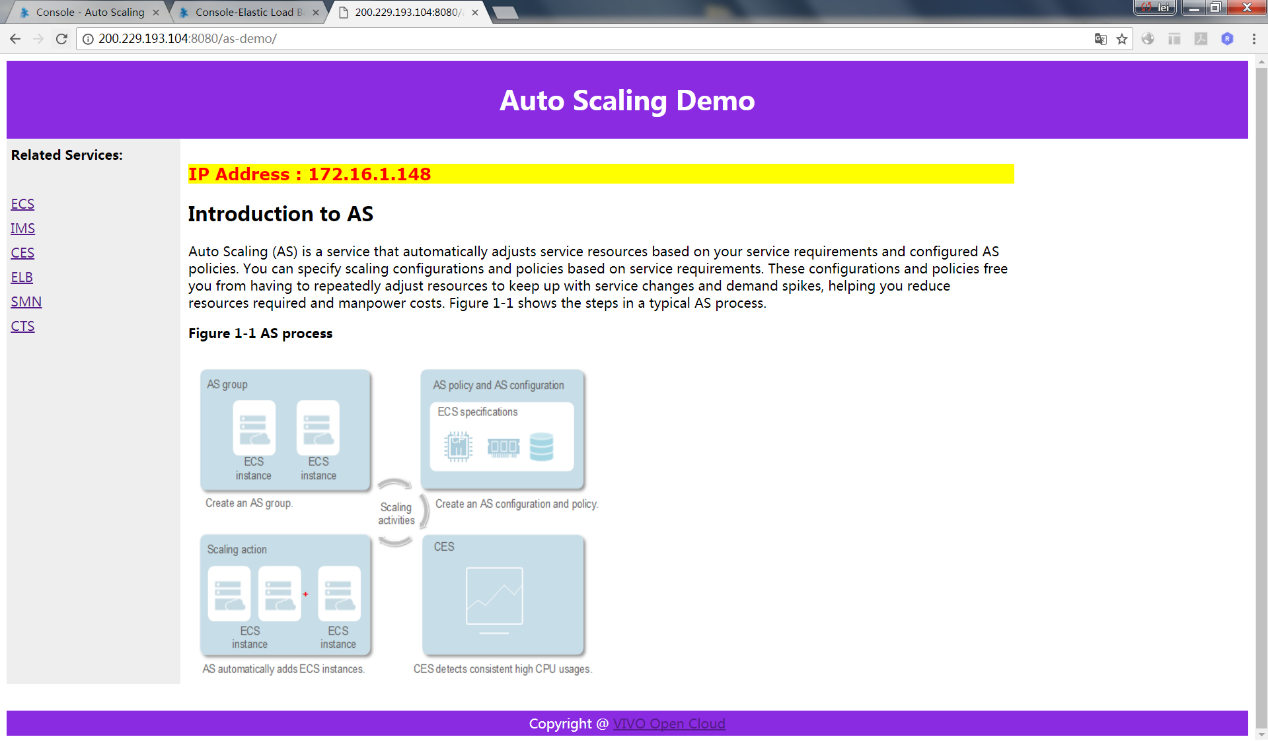
Check the ELB instance elb-demo and choose Backend ECS tab, you can see an ECS instance is registered to it. If you see the Health Check is Abnormal, do not worry, wait for some minutes.  


Eventually the backend ECS instance’s Health Check will be Normal.  


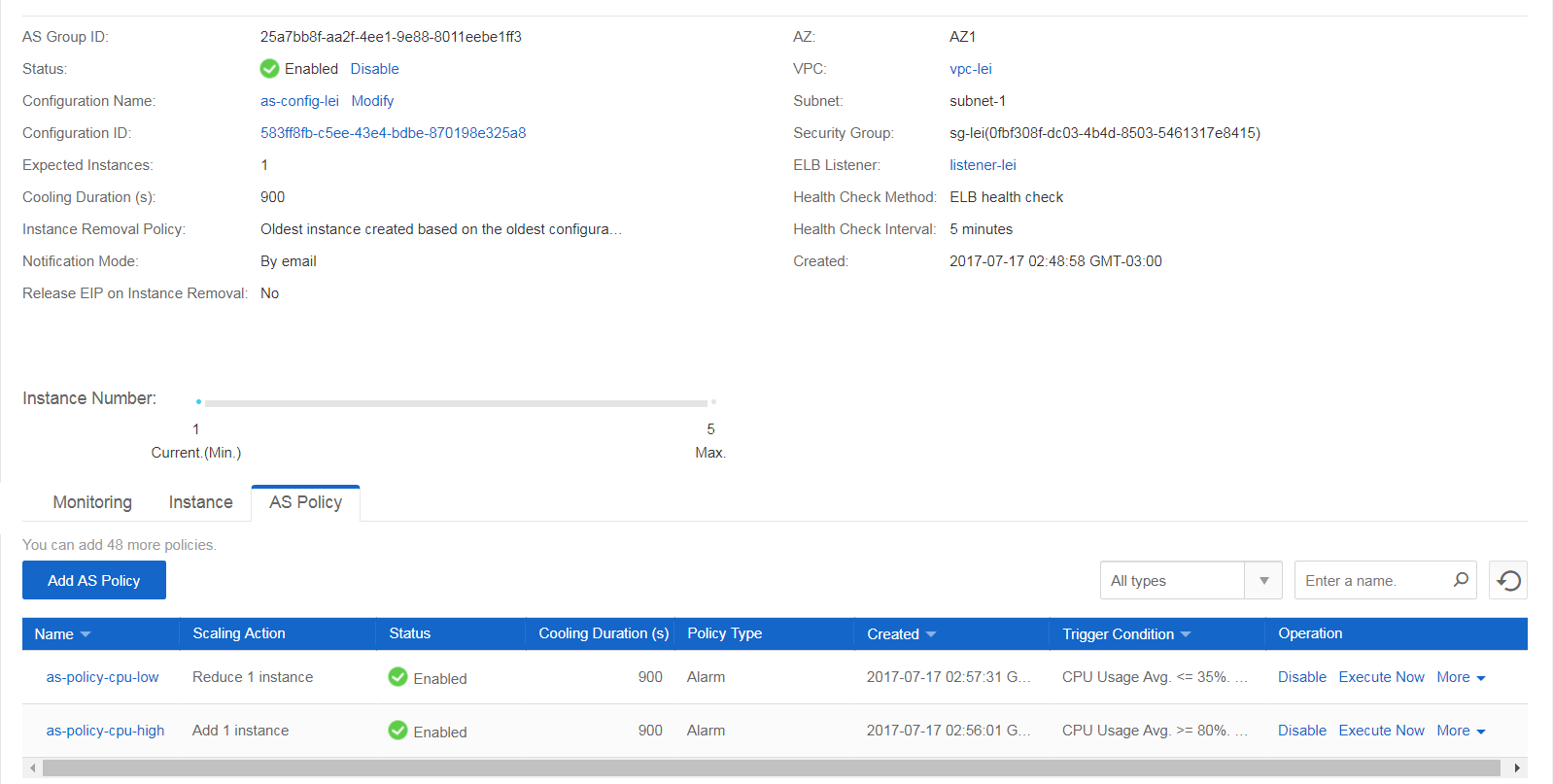
Check ELB basic information, especially the Service IP address, it will be used to access the web application. In this example, it is 200.229.193.104.  


**9. Access Web Application**

Now, it’s time to access the web application. Open a web browser and input: http://<ELB Service IP address>:8080/as-demo.  
**NOTE:** Please use the correct ELB Service IP address in your environment. For this example, the url is: http://100.100.100.100:8080/as-demo

If there is nothing wrong, you will see the following web page. Check the IP Address it printed if the same as your ECS instance private IP address.  


**10. Make sure CPU high policy is enabled**

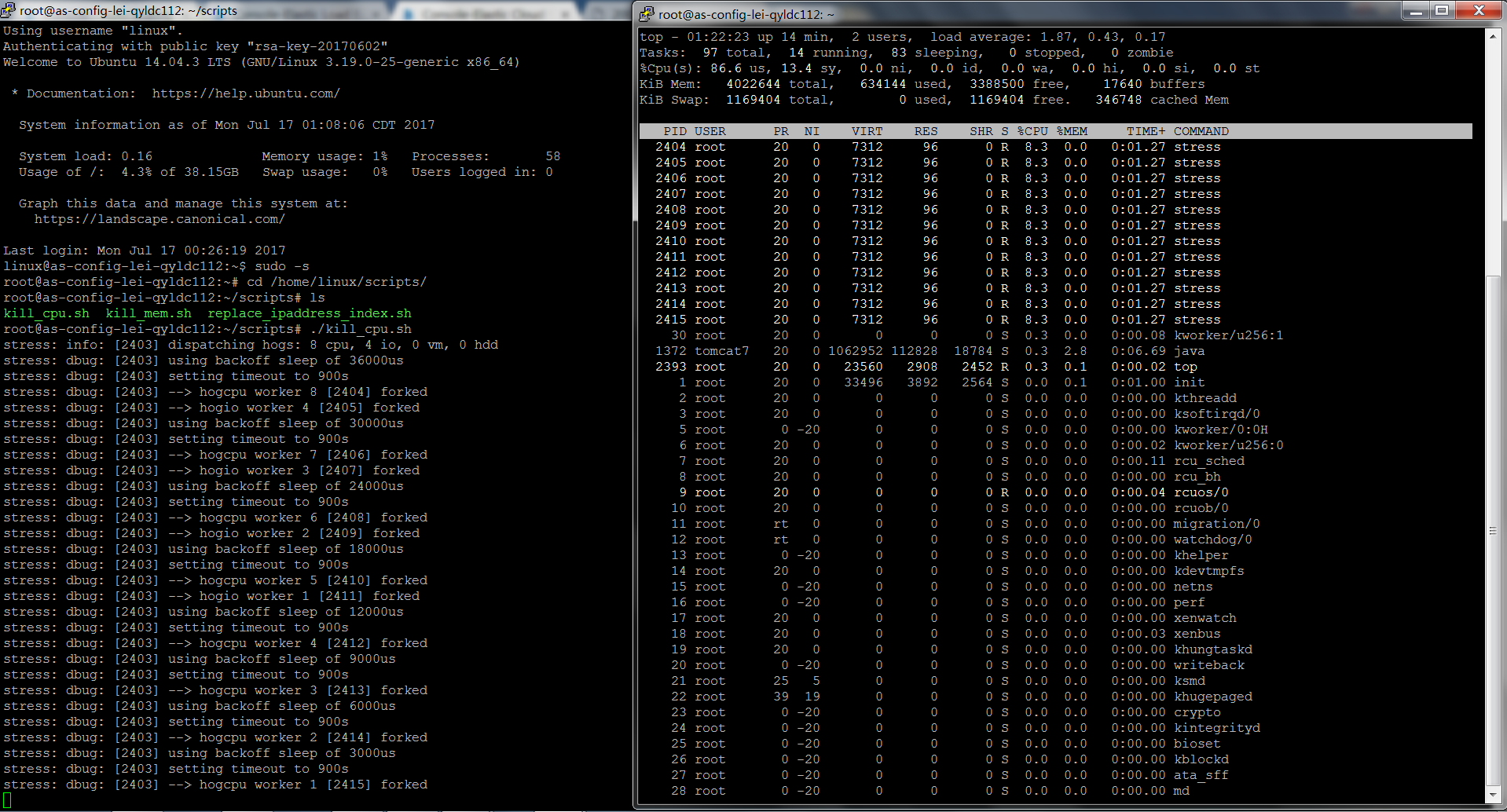
Now, go to AS Group->Policy to make sure the polices created are Enabled. If not, please enable them.  


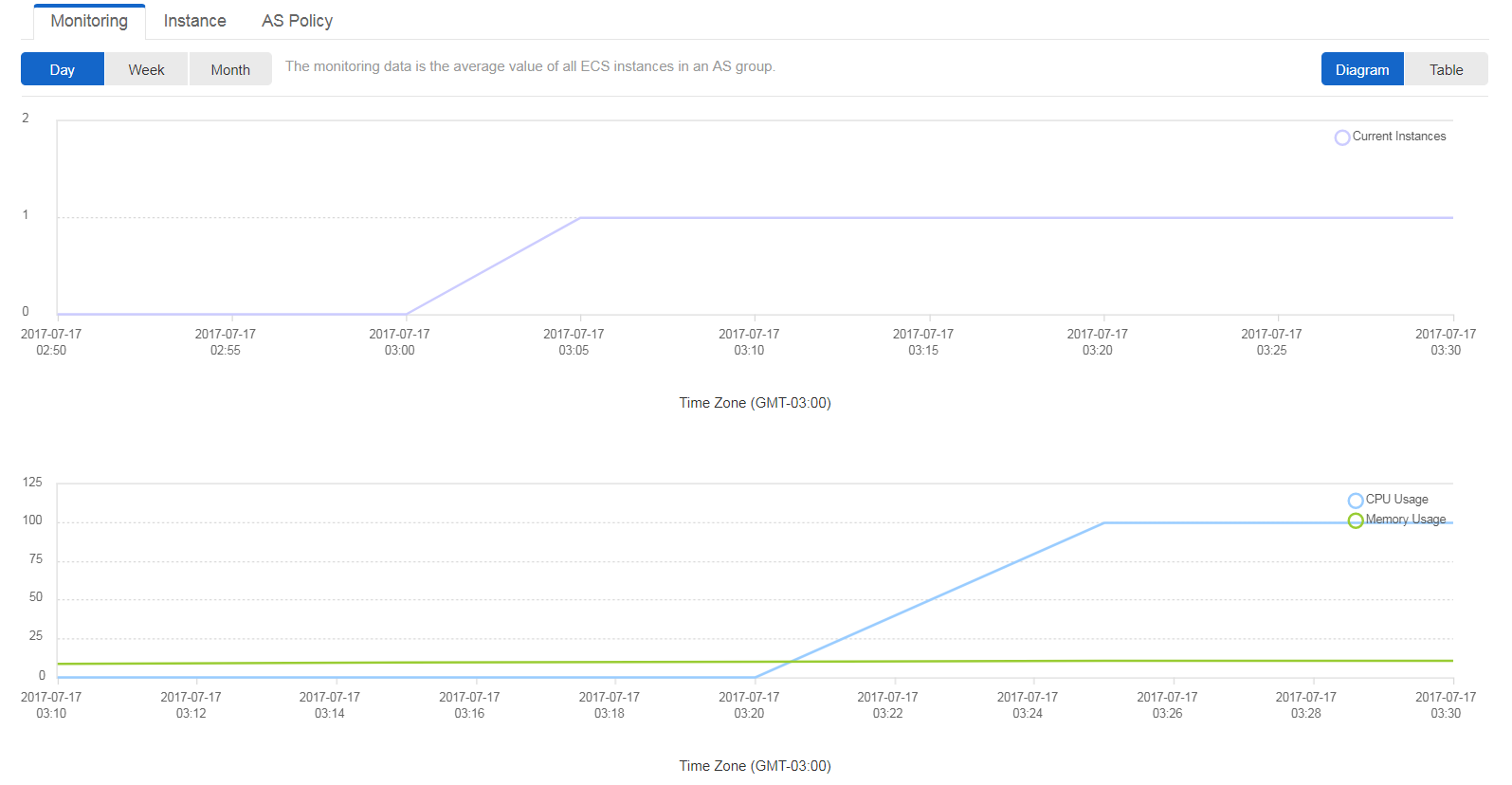
**11. Kill CPU**

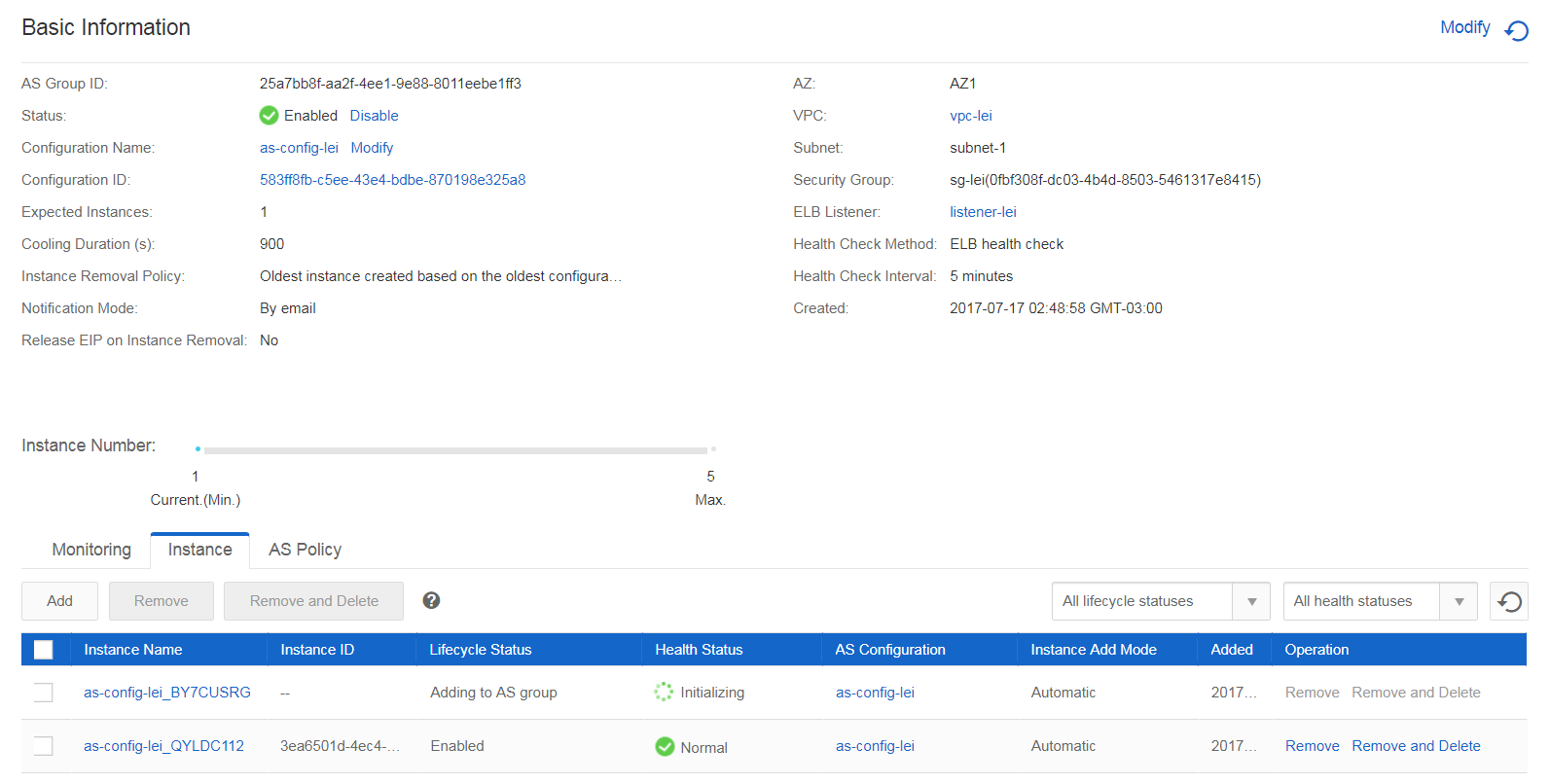
**Congratulations!** Now it’s time to show how auto scaling works.

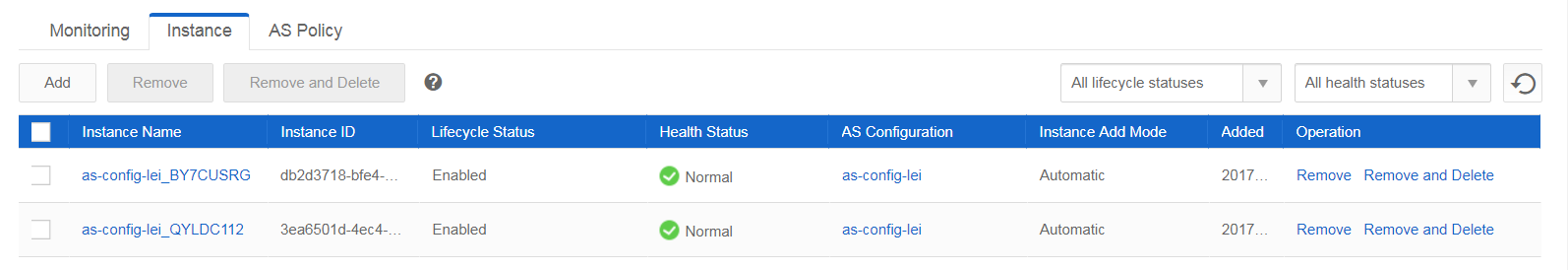
First, login to your ECS instance, and put some CPU load to it. I used the following:  
stress -c $[$(grep "processor" /proc/cpuinfo | wc -l) \* 8] -i 4 --verbose --timeout 15m

Then monitor its CPU Usage. If it is under 80%, then increase the parameter for -i option.

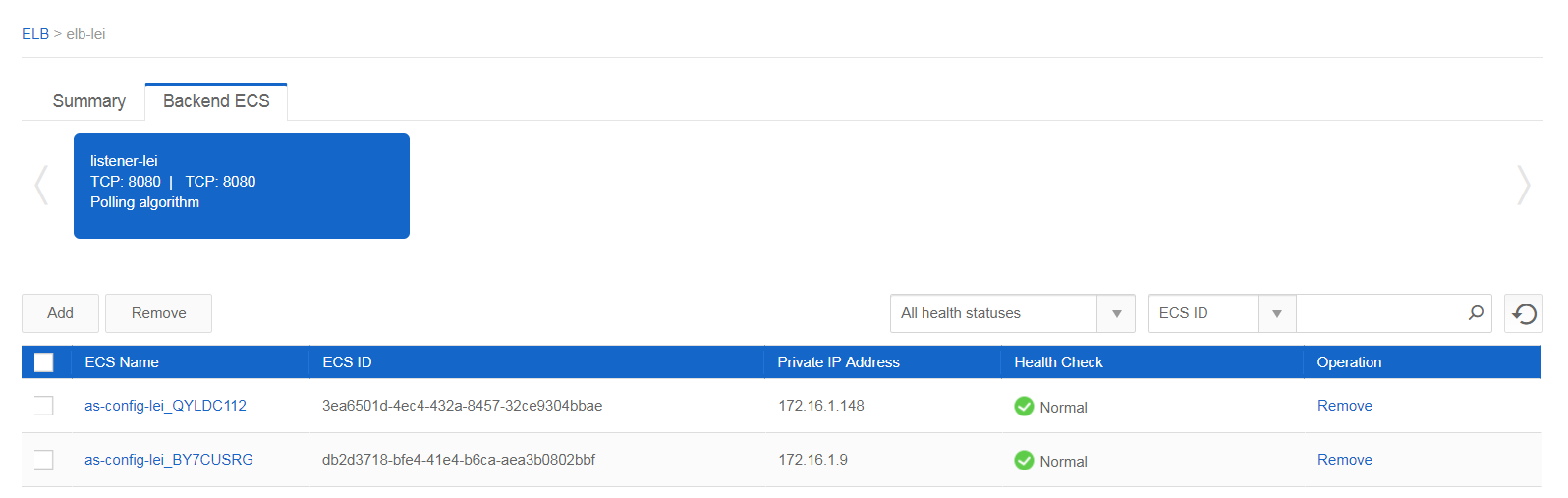


Or you can got to AS Group->Monitoring to monitor CPU usage, Memory Usage, Network Traffic etc.  


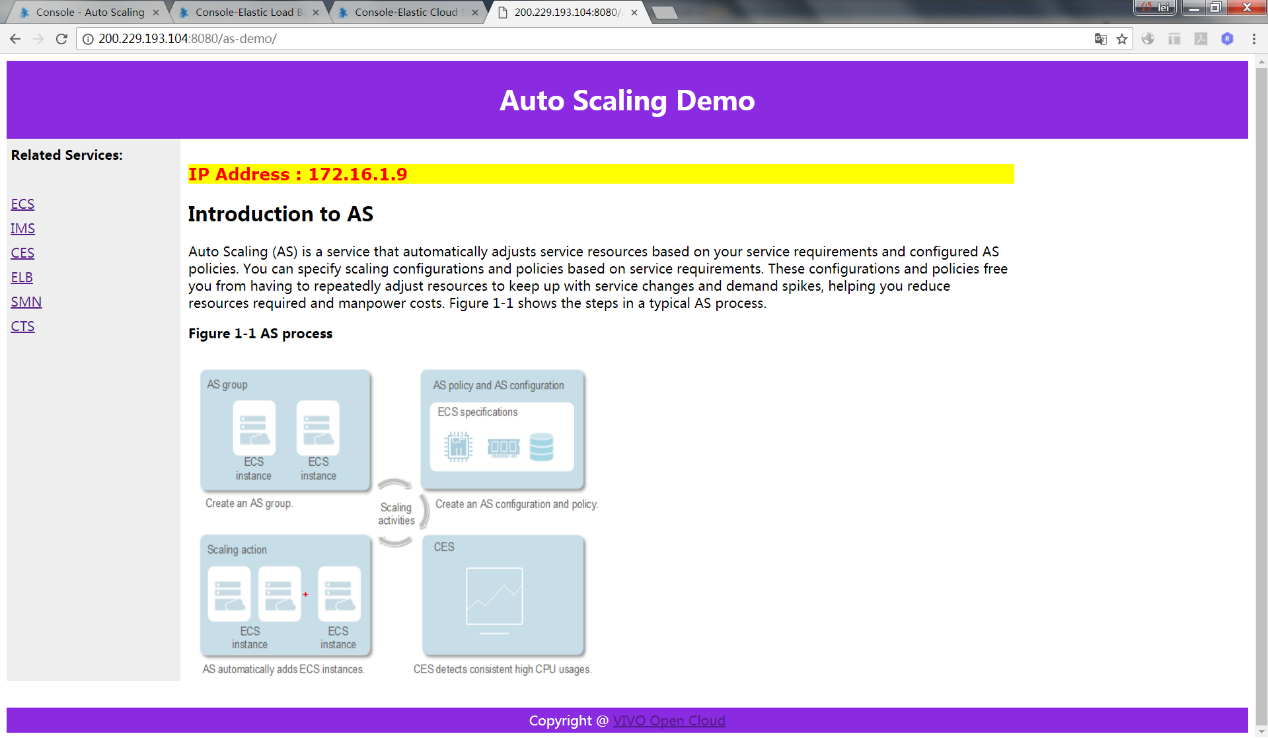
Wait for some time (at least 5 minutes), you can see a new instance is initializing.  


Wait for some minutes, the Health Status will be Normal.  


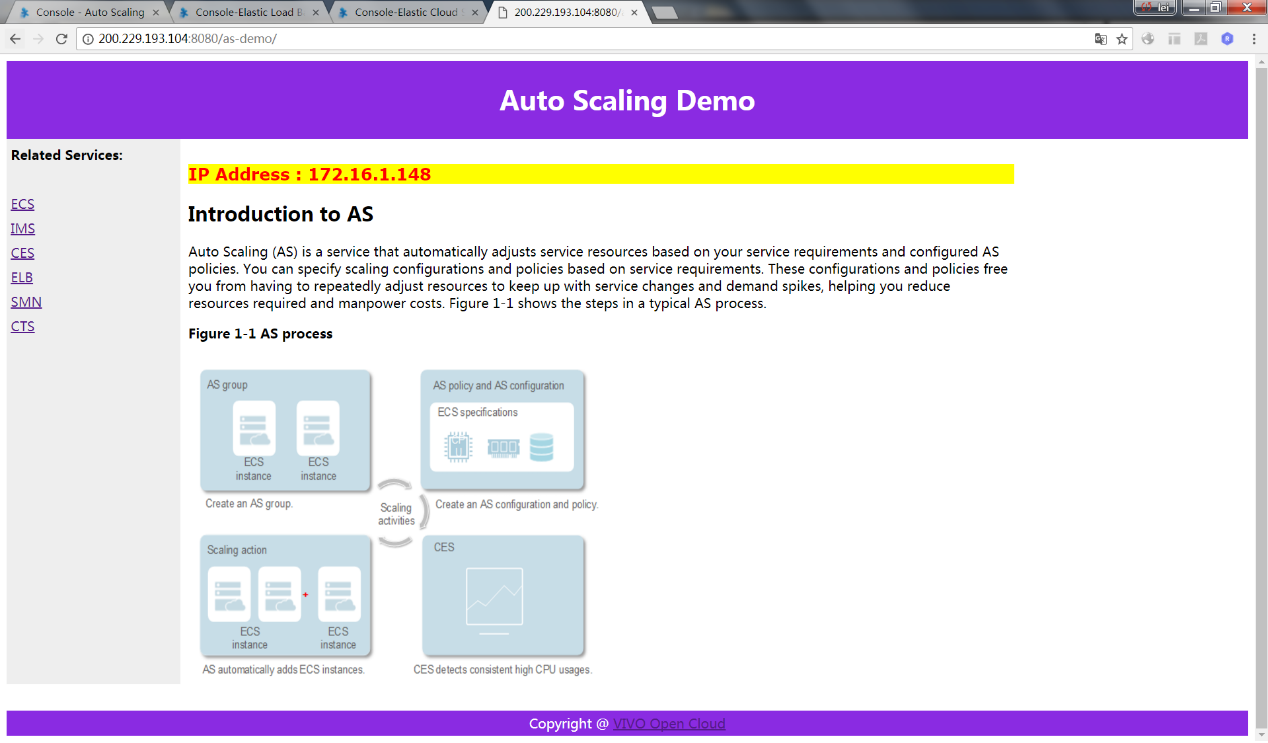
**12. ELB Backend ECS Instances**

Go to ELB elb-demo and choose Backend ECS, you will see the newly created ECS instance is registered to the ELB. If the “Health Check” status is not Normal, wait for some minutes.  


**13. Access Web Application after Scaling**

Now, it’s time to check auto scaling result. Open a web browser and input: http://<ELB Service IP address>:8080/as-demo, refresh the page for several times, you will see the IP Address changes each time when you refresh.  
**NOTE:** Please use the correct ELB Service IP address in your environment. For this example, the url is: http://200.229.193.104:8080/as-demo  


Refresh the page:



**14. AS Group CPU Low Reduce 1 Instance**

The kill\_cpu.sh will run for 15 minutes. After the script stop running, wait for some time (5 minutes), you will see the ECS instances in AS Group as-group-demo reduce 1 and there is only 1 instances(expected instance number is 1) running in the AS Group.  


When CPU usage is below 35% for some time (Monitoring Interval \* Consecutive Occurrences), AS Group will reduce 1 instance: