Lab 5: Pager Duty

In startups and medium-size companies it’s not uncommon for software engineers to serve pager duty. That’s partially because there’s very little dedicated IT Ops resources and partially because pager duty might make engineers to be more responsible and *feel* the pain of their bugs.

Luckily, since your tech stack is built on AWS, you can take advantage of the automation with launch configurations and autoscaling groups. You can monitors any spikes in traffic or any other metric like failures, CPU load or not having enough disk space and then scale up or down accordingly.

Task

- Create an autoscaling group and launch an instance
- Have autoscaling group launch an instance when CPU load is > 15% for 1 min on the instance in the launch group.

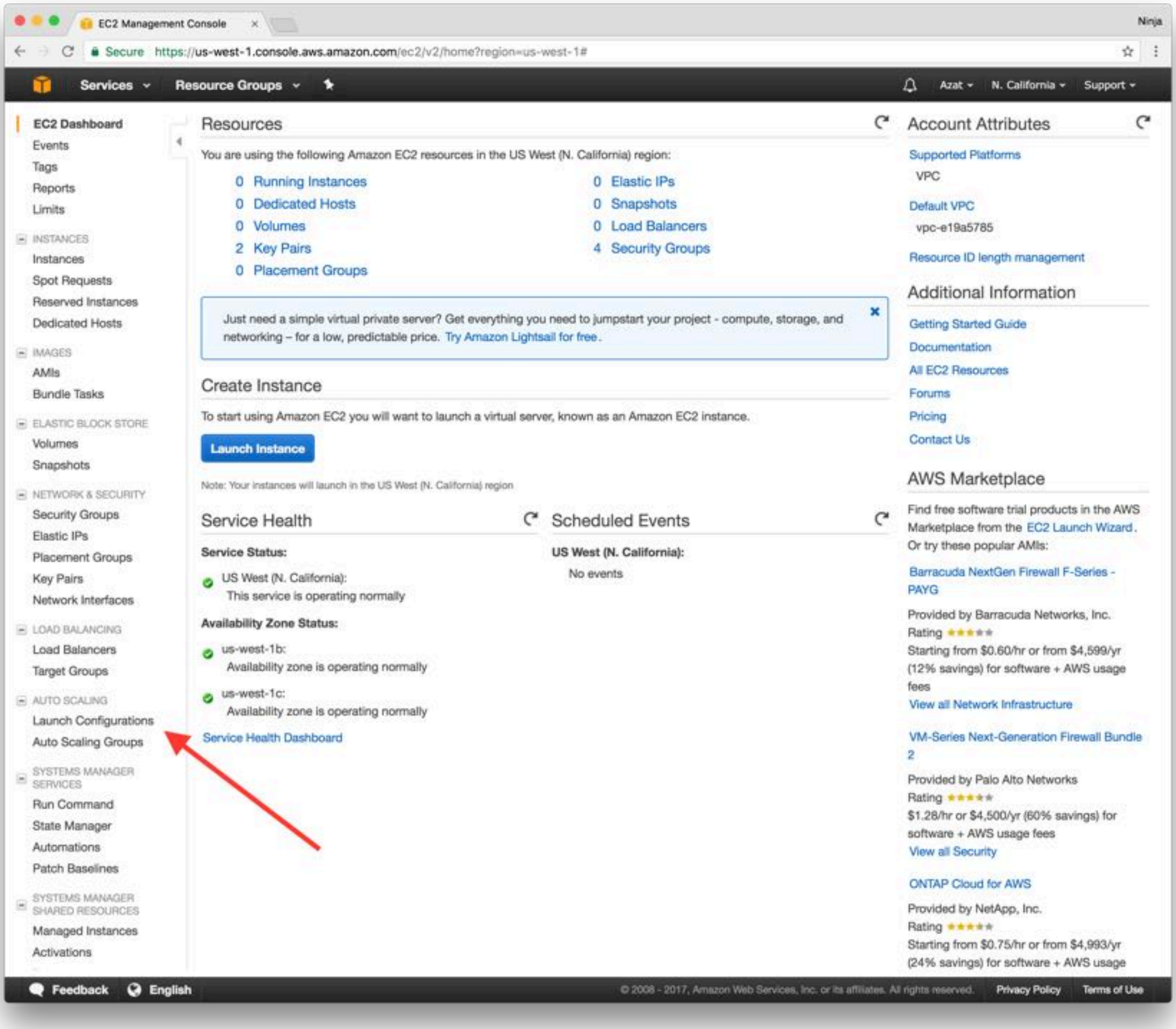
Walk-through

1. Create a launch configuration with 1 Node app with user data to start an HTTP server (config from lab 2, user data has modified Node Hello Wolrd app)
2. Create an autoscaling group
3. Create an autoscaling policy to increase instances by 1 when CPU load is > 15% for 1 min
4. Load test it with loadtest npm module to see a new instance is created
5. Remove autoscaling group
6. Terminate instances

If you would like to attempt the task, go skip the walk-through and for the task directly. However, if you need a little bit more hand holding or you would like to look up some of the commands or code or settings, then follow the walk-through.

1. Create an EC2 Launch Configuration for One Instance

Log in to the web console (use us-west-1 N California region) and navigate to the EC2 dashboard. Select “Launch Configuration” to start the wizard.



Resources

You are using the following Amazon EC2 resources in the US West (N. California) region:

0 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
0 Volumes	0 Load Balancers
2 Key Pairs	4 Security Groups
0 Placement Groups	

Just need a simple virtual private server? Get everything you need to jumpstart your project - compute, storage, and networking – for a low, predictable price. Try Amazon Lightsail for free.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (N. California) region

Service Health

Service Status:

US West (N. California):
This service is operating normally

Availability Zone Status:

us-west-1b:
Availability zone is operating normally

us-west-1c:
Availability zone is operating normally

Service Health Dashboard

Scheduled Events

US West (N. California):
No events

Account Attributes

Supported Platforms
VPC

Default VPC
vpc-e19a5785

Resource ID length management

Additional Information

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

AWS Marketplace

Find free software trial products in the AWS Marketplace from the EC2 Launch Wizard. Or try these popular AMIs:

Barracuda NextGen Firewall F-Series - PAYG
Provided by Barracuda Networks, Inc.
Rating ★★★★★
Starting from $0.60/hr or from $4,599/yr (12% savings) for software + AWS usage fees
View all Network Infrastructure

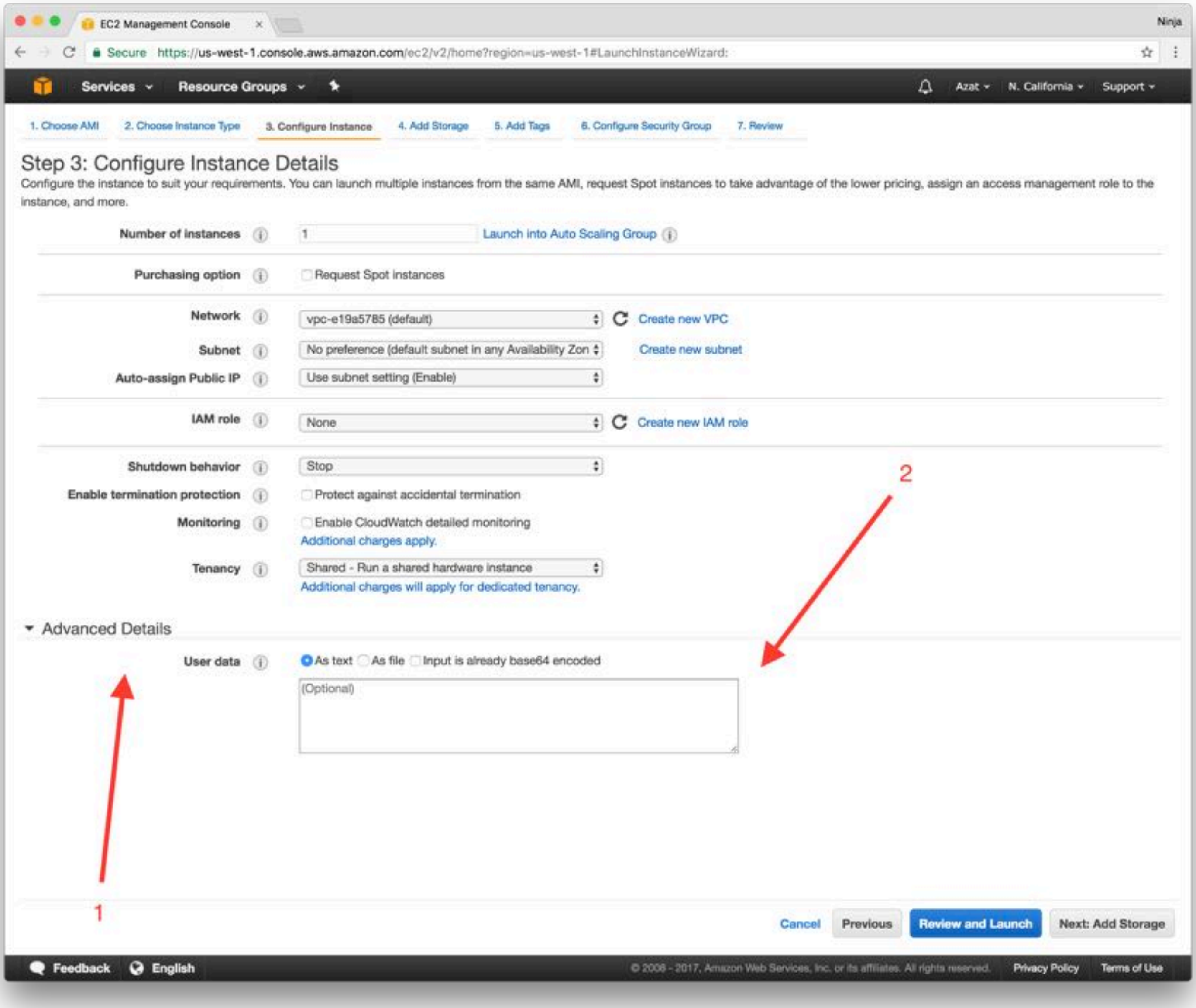
VM-Series Next-Generation Firewall Bundle 2
Provided by Palo Alto Networks
Rating ★★★★★
$1.28/hr or $4,500/yr (60% savings) for software + AWS usage fees
View all Security

ONTAP Cloud for AWS
Provided by NetApp, Inc.
Rating ★★★★★
Starting from $0.75/hr or from $4,993/yr (24% savings) for software + AWS usage

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The Launch Configuration wizard is *very* similar to the Launch instance wizard. You will need to specify image(s), instance(s) type(s), volume(s), etc.

On the first screen of the instance wizard, find in Quick Start Amazon Linux. We recommend using *“Amazon Linux 64-bit, HVM, SSD, EBS”* because it’s eligible for free tier on **t2.micro**.



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1

Purchasing option: □ Request Spot instances

Network: vpc-e19a5785 (default)

Subnet: No preference (default subnet in any Availability Zon

Auto-assign Public IP: Use subnet setting (Enable)

IAM role: None

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.

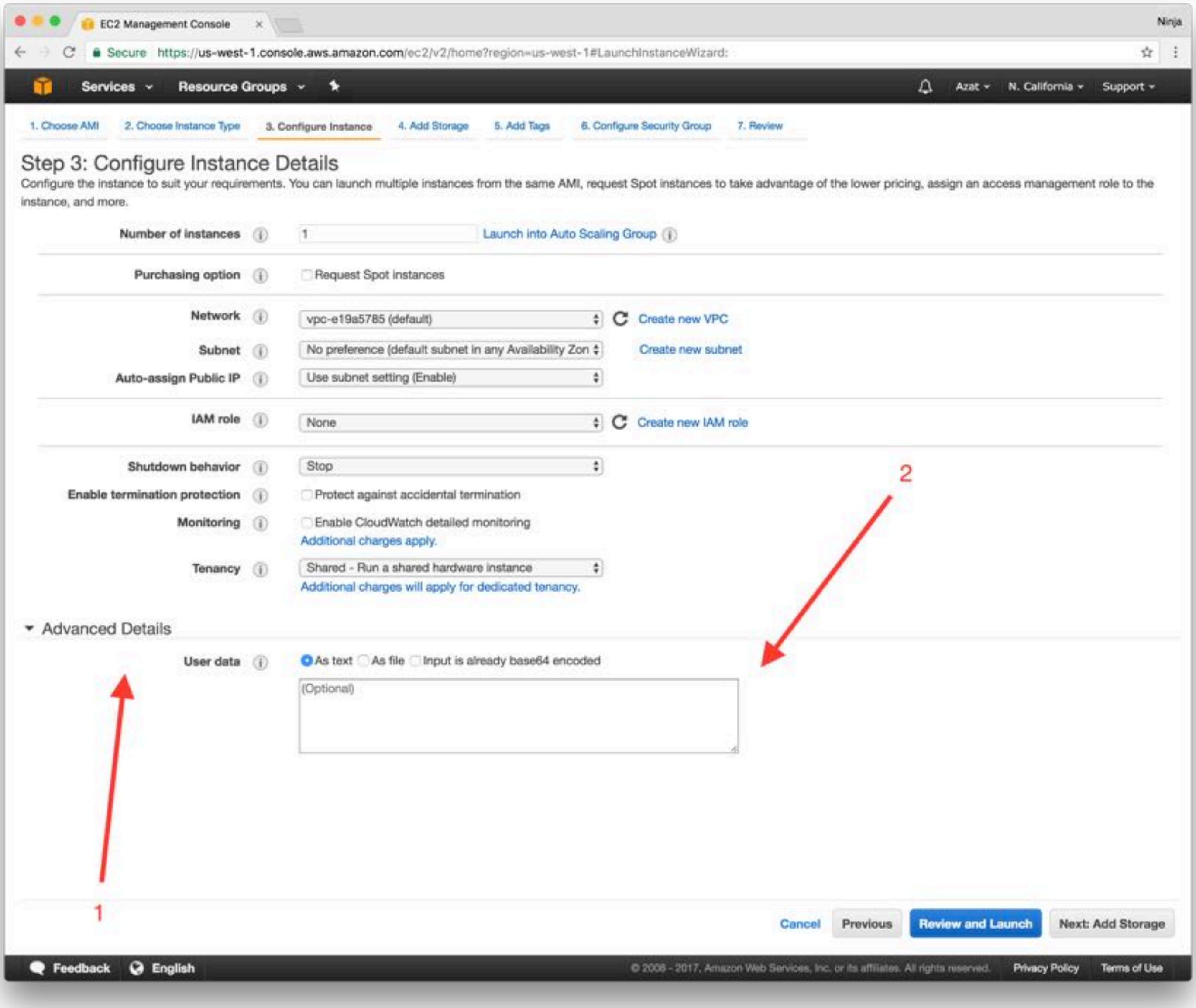
▾ Advanced Details

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

(Optional)

On the next screen *2. Choose an instance type*, select “t2.micro” (Free tier eligible).

Select Next to proceed to the step *3 Create Launch Configuration*. Enter the configuration name (arbitrary, could be foobar, but try to give it a meaningful name). Now configure the User Data which will install and run Node web server. The code was made slow so we can test autoscaling easier. To do so open Advanced settings as shown on the screen capture below:



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1 — Launch into Auto Scaling Group

Purchasing option: □ Request Spot instances

Network: vpc-e19a5785 (default) — Create new VPC

Subnet: No preference (default subnet in any Availability Zon — Create new subnet

Auto-assign Public IP: Use subnet setting (Enable)

IAM role: None — Create new IAM role

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
(Optional)

Paste the following code for bash command into User Data. For bonus, change the Hello World to Hello {YOUR_NAME}.

```
#!/bin/bash -ex
# output user data logs into a separate place for debugging
exec > >(tee /var/log/user-data.log|logger -t user-data -s 2>/dev/console) 2>&1
# get node into yum
curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -
# install node (and npm) with yum
yum -y install nodejs
# install pm2 to restart node app
npm i -g pm2@2.4.3
# get source code for SLOW node server from GitHub's gist (could be private GitHub
 repo or private S3)
curl "https://gist.githubusercontent.com/azat-co/ee4038e32db3581b8b4fb34ef15ff00d/
```

```
raw/e74557029d0acdc2d9dab0d90c5172ebcf29a649/slow-hello-world.js" > /home/ec2-user
/hello-world-server.js
sudo chmod 755 /home/ec2-user/hello-world-server.js # optional
# restart pm2 and thus node app on reboot
crontab -l | { cat; echo "@reboot sudo pm2 start /home/ec2-user/hello-world-server
.js -i 0 --name \"node-app\""; } | crontab -
# start the server with sudo because it is port 80, not 3000
sudo pm2 start /home/ec2-user/hello-world-server.js -i 0 --name "node-app"
```

Leave other fields default on step 3. Also, leave screen 4 (storage) with default settings.

On the step 5: Security Groups Add have *at the very least* these protocols and ports open:

- HTTP 80
- HTTPS 443
- SSH 22

Review everything and select **Create launch configuration**.

If you are curious, here's the code for the slow Node Hello World server. I intentionally put a for loot there to consume extra CPU cycles. 😈

```
const port = 80
require('http')
  .createServer((req, res) => {
    console.log('url:', req.url)
    console.log('now we will slow down and block and consume CPU...')
    for (var i=0; i< 1000000; i++) {}
    res.end('hello world')
  })
  .listen(port, (error)=>{
    console.log(`server is running on ${port}`)
  })
```

Note: Don't forget to select an existing key pair for which you have the private key saved or create a new key pair and *download* the private key.

2. Create autoscaling group

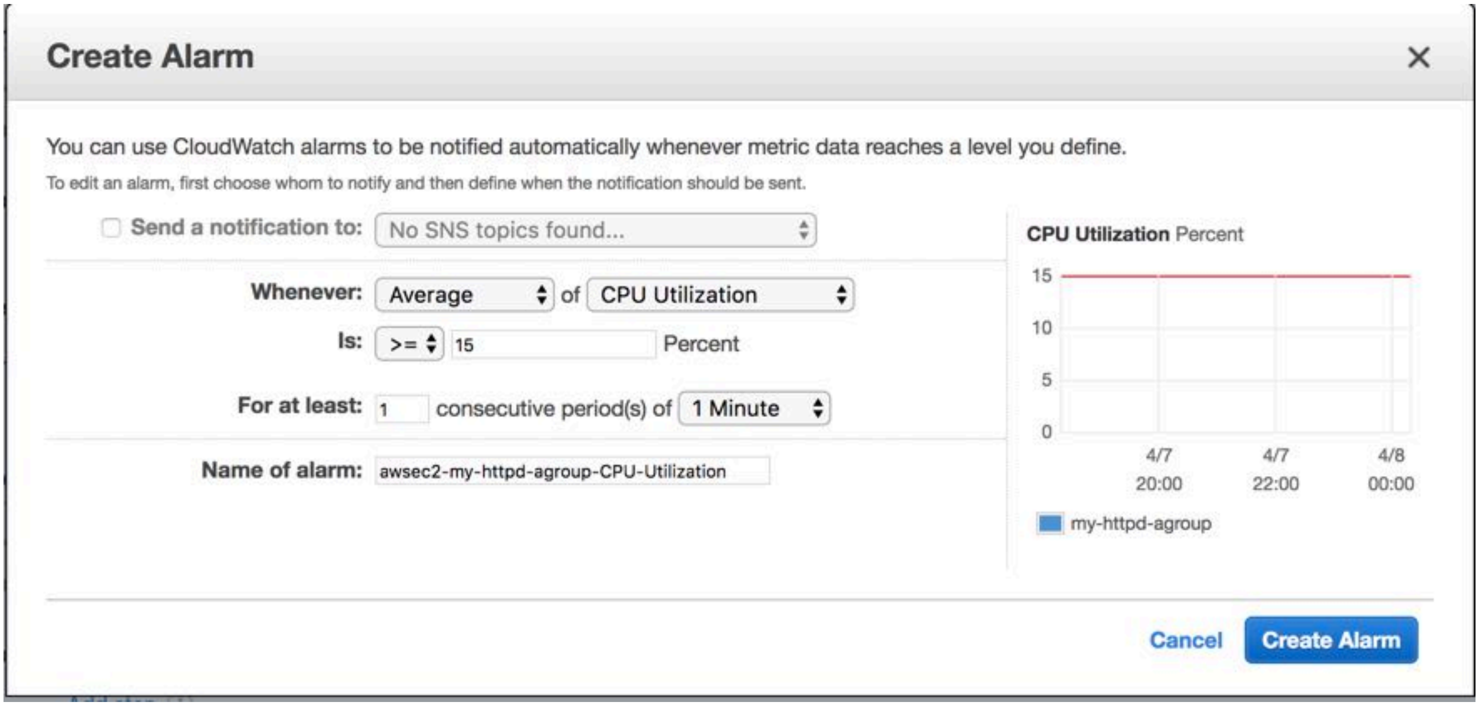
Once you create the launch configuration (think of it as a blue print), create an autoscaling group. On the first screen *1. Configure Auto Scaling group details*, select the following:

- Group name: pick something arbitrary but meaningful, e.g., my-awesome-httpd-autogroup
- Group size: 1
- Network: default
- Subnet: us-west-1b and us-west-1c (make sure you are in us-west-1 region)
- Advanced I Load Balancing: ✓ Check (Receive traffic from one or more load balancers)
- Health Check Grace Period: 1second
- Others: leave default

On *2. Configure scaling policies* select “Use scaling policies to adjust the capacity of this group” - of course we want to have the auto scaling! :)

Now you should see the range (max and min instances) and two policies. Increase the max range to 2 as shown in the screenshot… then create two policies: increase and decrease which will increase and decrease instance count by 1 when average CPU load is higher and lower than 15% for 1 min respectively.

You will need to create two alerts (AWS CloudWatch). You can do it right from this screen (see screencaptures). Make sure to use 1minutes, otherwise you’ll have to wait *longer* for the alert to kick in (15min, 1hour, etc.).



Create Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

□ Send a notification to: No SNS topics found...

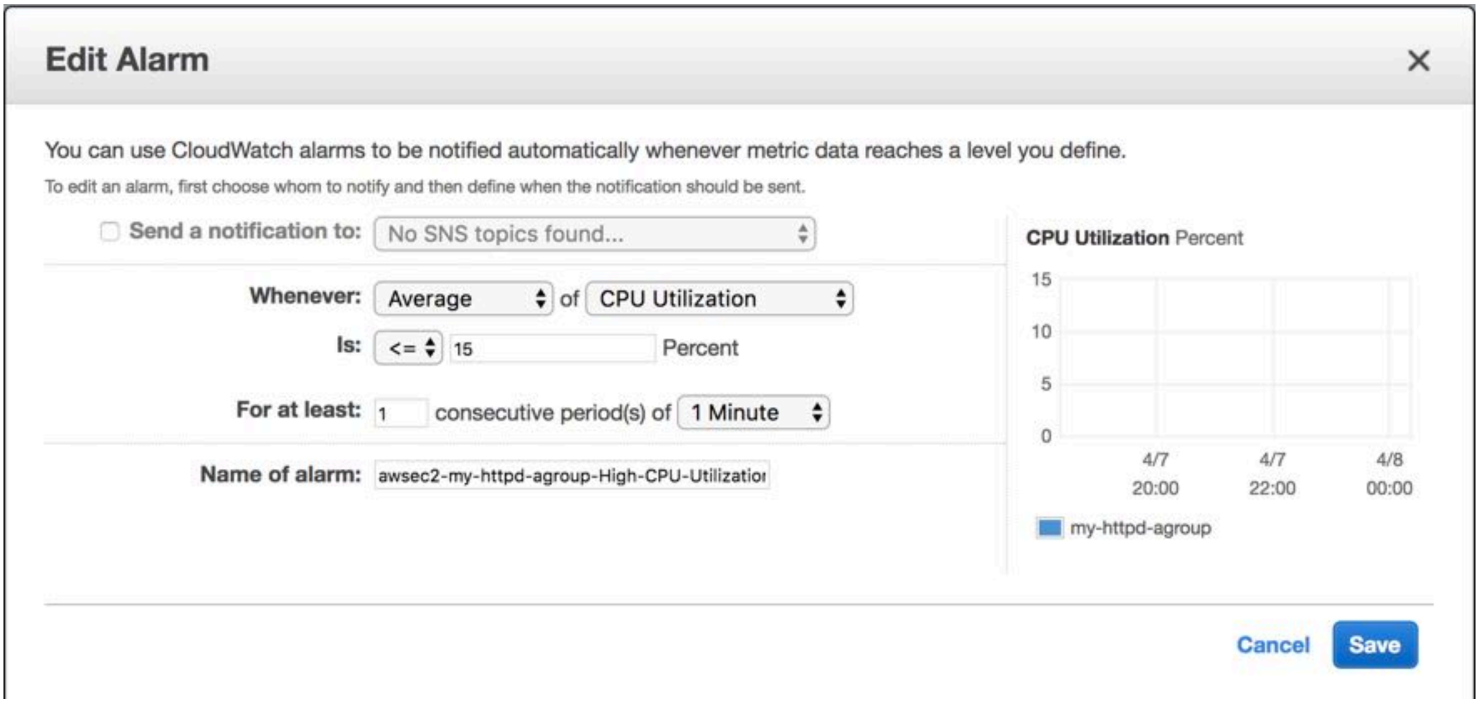
Whenever: Average of CPU Utilization

Is: >= 15 Percent

For at least: 1 consecutive period(s) of 1 Minute

Name of alarm: awsec2-my-httpd-agroup-CPU-Utilization

And this is the decrease alert which is basically the opposite of the increase alert. You don’t have to set up notification but it’s rather easy to do if you want to get email (or any other) alerts via AWS SNS.



You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

□ Send a notification to: No SNS topics found...

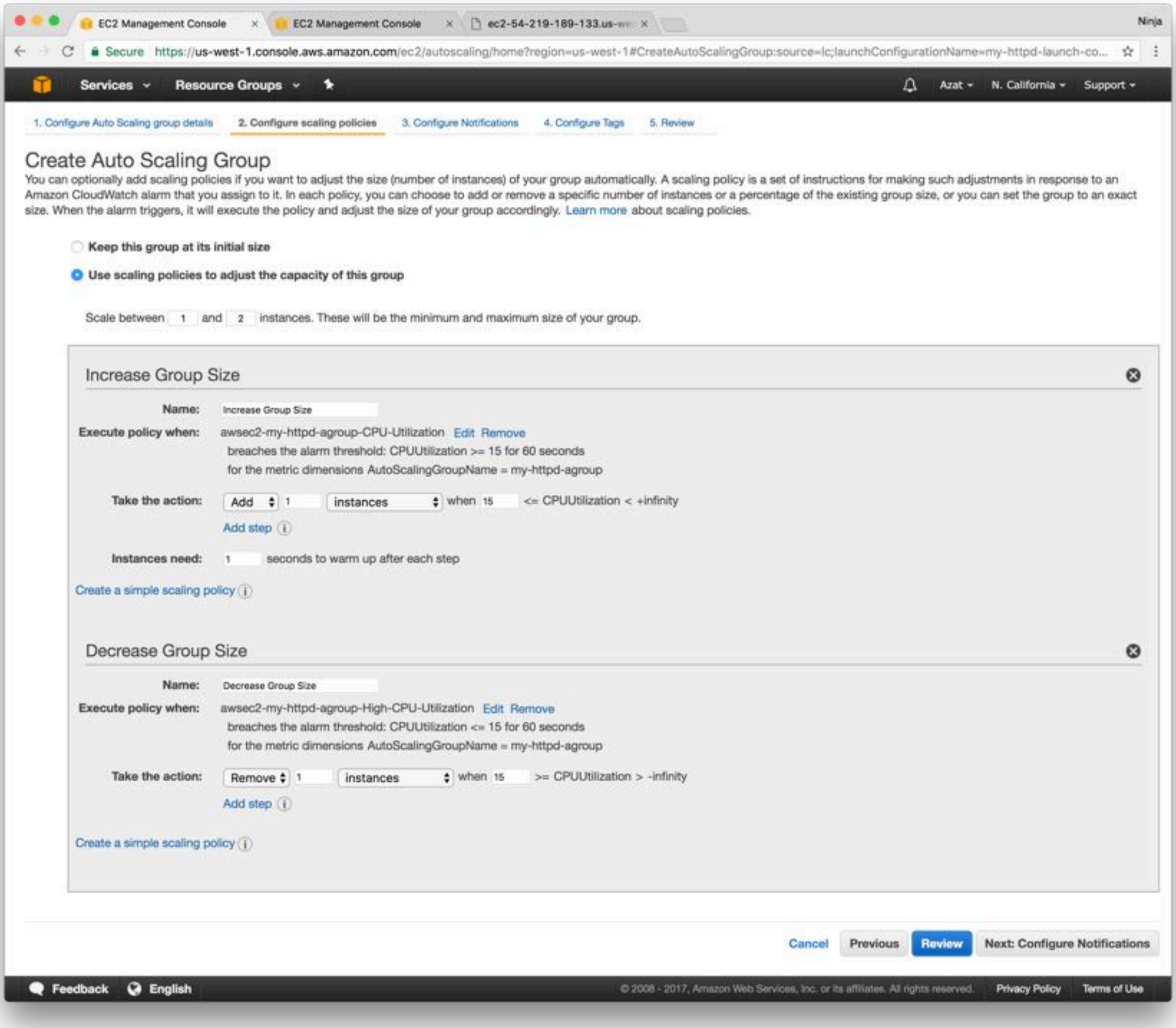
Whenever: Average of CPU Utilization

Is: <= 15 Percent

For at least: 1 consecutive period(s) of 1 Minute

Name of alarm: awsec2-my-httpd-agroup-High-CPU-Utilizatior

The end result should look like this with 2 policies and their alerts (and instance range 1-2):


EC2 Management Console
EC2 Management Console
ec2-54-219-189-133.us-we
Ninja
Secure https://us-west-1.console.aws.amazon.com/ec2/autoscaling/home?region=us-west-1#CreateAutoScalingGroup:source=lc;launchConfigurationName=my-httpd-launch-co...
Services
Resource Groups
Azat
N. California
Support
1. Configure Auto Scaling group details
2. Configure scaling policies
3. Configure Notifications
4. Configure Tags
5. Review

Create Auto Scaling Group

You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. Learn more about scaling policies.

○ Keep this group at its initial size

● Use scaling policies to adjust the capacity of this group

Scale between 1 and 2 instances. These will be the minimum and maximum size of your group.

Increase Group Size

Name: Increase Group Size

Execute policy when: awsec2-my-httpd-agroup-CPU-Utilization Edit Remove
breaches the alarm threshold: CPUUtilization >= 15 for 60 seconds
for the metric dimensions AutoScalingGroupName = my-httpd-agroup

Take the action: Add 1 instances when 15 <= CPUUtilization < +infinity

Add step

Instances need: 1 seconds to warm up after each step

Create a simple scaling policy

Decrease Group Size

Name: Decrease Group Size

Execute policy when: awsec2-my-httpd-agroup-High-CPU-Utilization Edit Remove
breaches the alarm threshold: CPUUtilization <= 15 for 60 seconds
for the metric dimensions AutoScalingGroupName = my-httpd-agroup

Take the action: Remove 1 instances when 15 >= CPUUtilization > -infinity

Add step

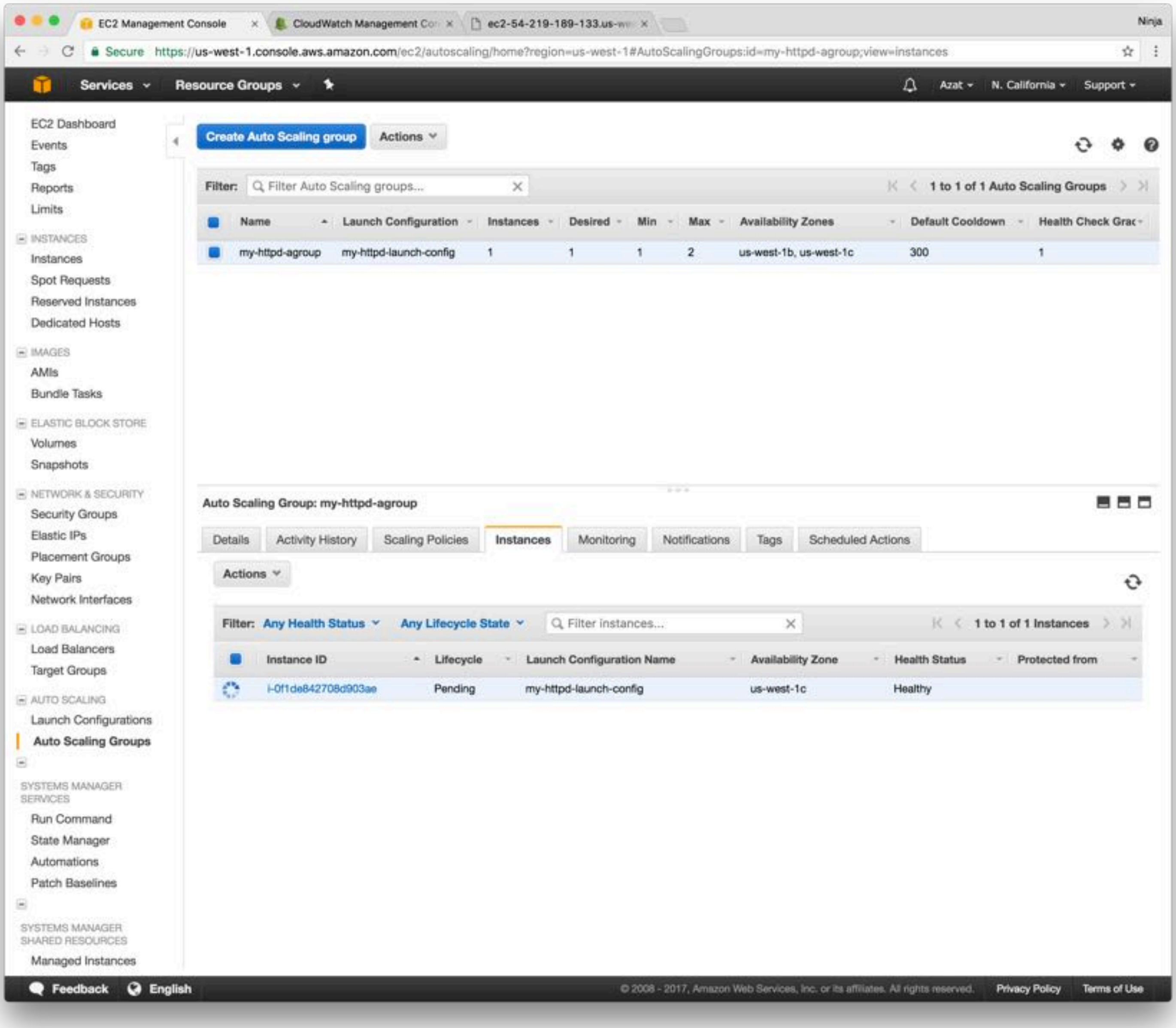
Create a simple scaling policy

Cancel
Previous
Review
Next: Configure Notifications
Feedback
English
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Select Next and skip (leave empty) *3. Configure Notifications*. On *4. Configure Tags*, fill tag with named role with value “aws-course”.

On *5. Review*, review everything and then select “Create Auto Scaling group”.

Select “Close” to get back to the EC2 dashboard… on the Auto Scaling Groups, you’ll see a new instance in the bottom menu (Instances tab) as shown below:



Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grac
my-httpd-agroup	my-httpd-launch-config	1	1	1	2	us-west-1b, us-west-1c	300	1

Auto Scaling Group: my-httpd-agroup

Instance ID	Lifecycle	Launch Configuration Name	Availability Zone	Health Status	Protected from
i-0f1de842708d903ae	Pending	my-httpd-launch-config	us-west-1c	Healthy	

Feedback
English
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

You can click on the instance ID or find it in the Instance. Copy the public URL for the newly created EC2 instance. Paste it in the browser to navigate to the web server. You'll see the index.html page with the "This is my cool HTML page" heading.

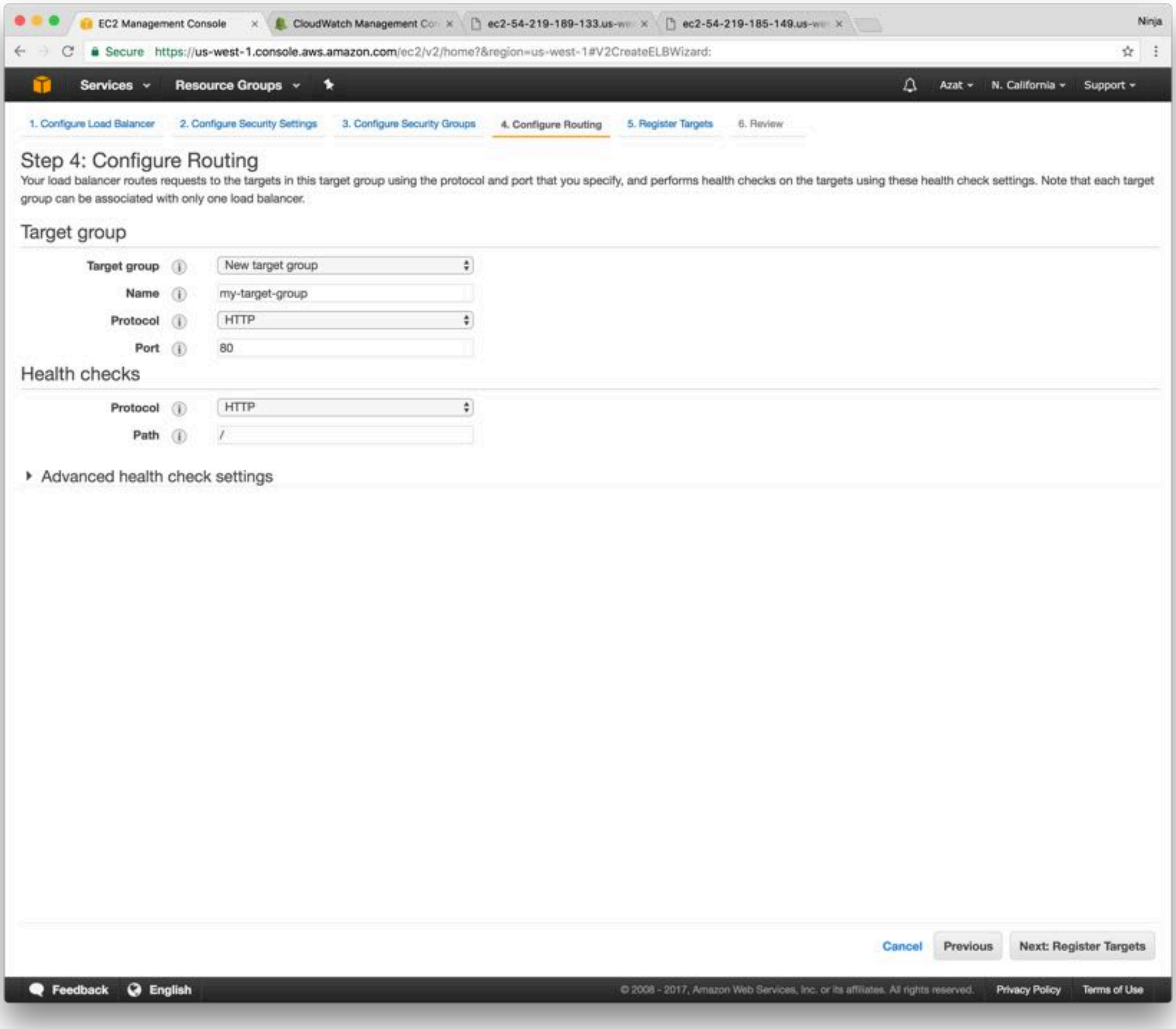
Alternatively, execute curl from your developer machine to see the HTML response:

```
curl PUBLIC_URL
```

2.a. Create App Load Balancer and Add Target Group (Optional - Lab 4 is on ELB)

The next lab will be all about ELB, but if you are already familiar with ELB, then implement app ELB, add target group and attache it to the autoscaling group.

First, create an app ELB (not classic) from the EC2 console. Select public and IPv4 and proper listeners to open port 80 (at least). Then, pick a name for the new target group as shown below.



Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the targets using these health check settings. Note that each target group can be associated with only one load balancer.

Target group

Target group	New target group
Name	my-target-group
Protocol	HTTP
Port	80

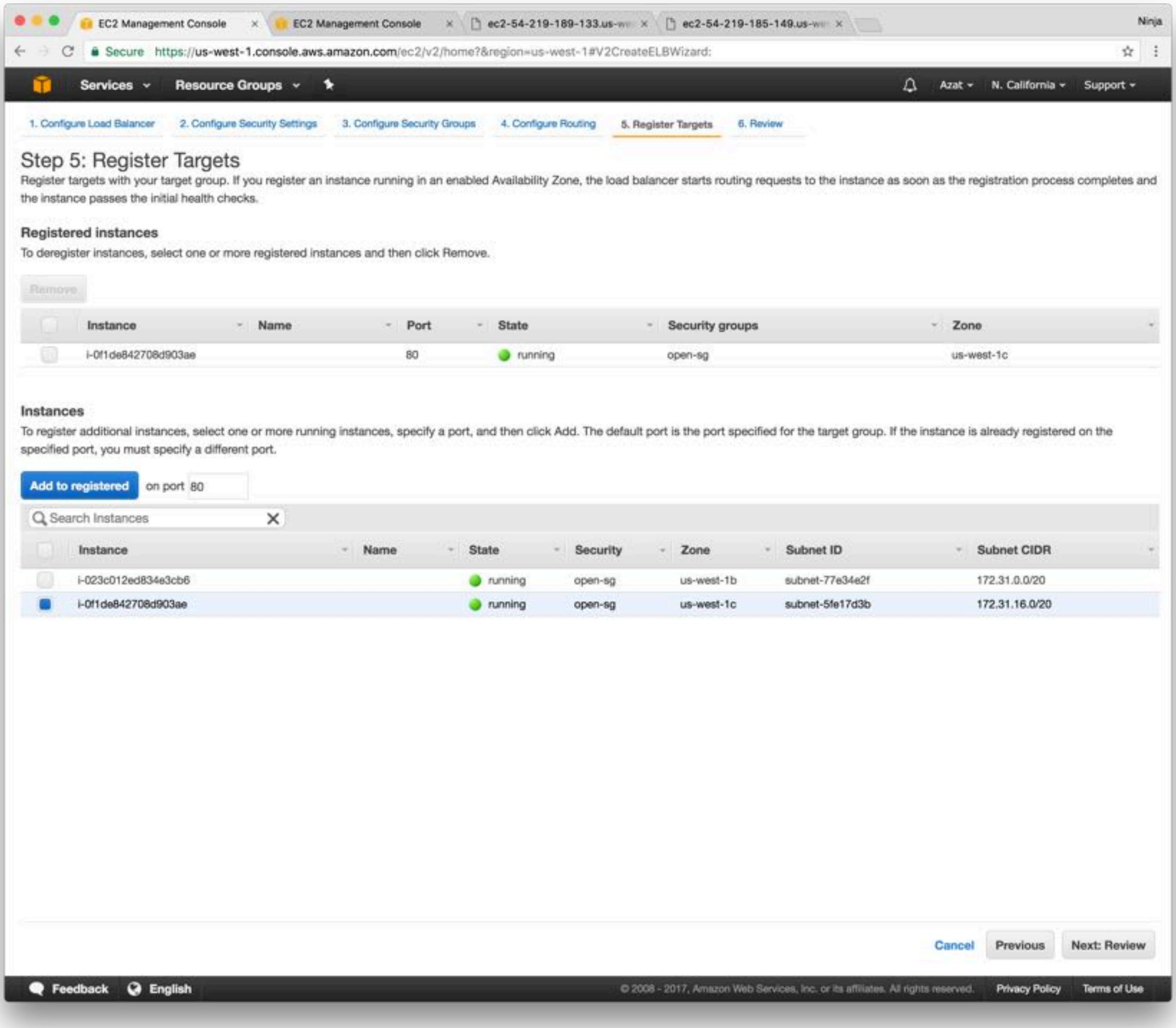
Health checks

Protocol	HTTP
Path	/

Advanced health check settings

Cancel
Previous
Next: Register Targets
Feedback
English
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

On the next step, register the instances launched by your autoscaling group (use ID if you have more than one instance running). See sceenshot as an example—your IDs and names *can and probably will* vary:



Step 5: Register Targets

Register targets with your target group. If you register an instance running in an enabled Availability Zone, the load balancer starts routing requests to the instance as soon as the registration process completes and the instance passes the initial health checks.

Registered instances

To deregister instances, select one or more registered instances and then click Remove.

□	Instance	Name	Port	State	Security groups	Zone
□	i-0f1de842708d903ae		80	● running	open-sg	us-west-1c

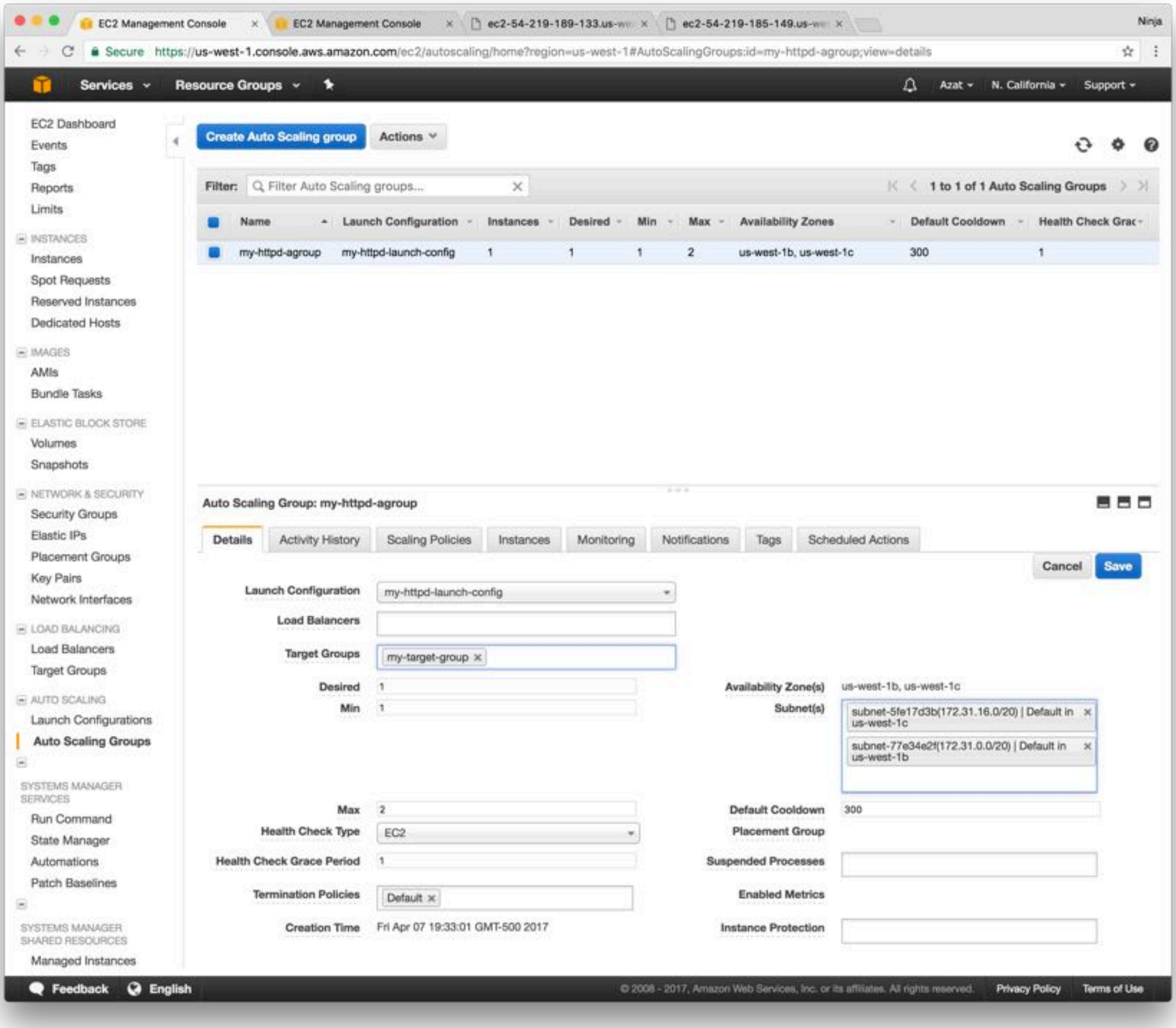
Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

□	Instance	Name	State	Security	Zone	Subnet ID	Subnet CIDR
□	i-023c012ed834e3cb6		● running	open-sg	us-west-1b	subnet-77e34e2f	172.31.0.0/20
☑	i-0f1de842708d903ae		● running	open-sg	us-west-1c	subnet-5fe17d3b	172.31.16.0/20

Cancel
Previous
Next: Review
Feedback
English
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Lastly, associate this target group (ELB) with your autoscaling group by editing your autoscaling group and adding the target group by name (there would be an auto complete drop down). Ignore the field “Load balancers”. That’s for the classic ELB and we are using app ELB. See an example below:



https://us-west-1.console.aws.amazon.com/ec2/autoscaling/home?region=us-west-1#AutoScalingGroups:id=my-httpd-agroup;view=details

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grac
my-httpd-agroup	my-httpd-launch-config	1	1	1	2	us-west-1b, us-west-1c	300	1

Auto Scaling Group: my-httpd-agroup

Launch Configuration	my-httpd-launch-config		
Load Balancers			
Target Groups	my-target-group		
Desired	1	Availability Zone(s)	us-west-1b, us-west-1c
Min	1	Subnet(s)	subnet-5fe17d3b(172.31.16.0/20) | Default in us-west-1c subnet-77e34e2f(172.31.0.0/20) | Default in us-west-1b
Max	2	Default Cooldown	300
Health Check Type	EC2	Placement Group	
Health Check Grace Period	1	Suspended Processes	
Termination Policies	Default	Enabled Metrics	
Creation Time	Fri Apr 07 19:33:01 GMT-500 2017	Instance Protection	

Again. This step is optional because we are really working an an autoscaling group but typically developers use ELB with autoscaling feature so that's why there's this step 2.a. but it's optional. We will do another lab just on ELB soon.

3. Stress Test Web server

If you did step 2.a., then use the LB URL (DNS name). If you skipped 2.a. that means you don't have a load balancer so just use the DNS or public IP of your single instance under the autoscaling group.

Stress test the heck of your web server for at least one minute and then check in the EC2 console that the new instance was launched by the autoscaling group.

I recommend using loadtest library which is light weight CLI tool written in Node and available via npm. To install loadtest, simply (assuming you have Node and npm—npm comes with Node) run in your terminal / command prompt:

```
npm i -g loadtest@2.3.0
```

You can also use Apache ab or JMeter. Good tools for stress testing but not as simple as loadtest in my opinion.

The run the stress test with this command (my LB URL is my-httpd-lb-1208730027.us-west-1.elb.amazonaws.com):

```
loadtest -c 100 --rps 200 http://my-httpd-lb-1208730027.us-west-1.elb.amazonaws.com
```

Or if you don't use ELB, the command will look something like this

```
loadtest -c 100 --rps 200 http://ec2-54-183-255-49.us-west-1.compute.amazonaws.com
```

Keep in mind that because the port is 80, there's no need to specify it in the URL (like we did with 3000 in the previous lab). 80 is the default port for HTTP.

In the command above `-c` means concurrency and `--rps` means requests per second for each client. 100 and 200 should give enough troubles to t2.micro instance to ramp up the CPU load to hit the 15% threshold. This in turn will trigger the alarm to increase instance count by one. Wait for 1 min (make sure you alert period is 1 min). Go to EC2 console and check for the new instance create by the auto group. Adjust rps and concurrency as needed if the alarm hasn't been triggered.

If you used ELB (2.a.) then CPU load will decrease and decrease alert might be triggered with time. If you didn't implement ELB, then simple stop stress testing and observed the removal of one instance so the total count is back to one.

To avoid extra charges or running out of the trial compute time, terminate the autoscaling group first, thin EBL and target groups (if you implemented them) and then instances. If you terminate instances first, autoscaling group will spawn new instances to get to the desired size.

Troubleshooting

No new instances: If you need to edit an alert because you noticed a mistake (maybe you forgot to select 1 min instead of 15min), you can still edit the alert but you'll need to open a new tab and navigate to the AWS CloudWatch I Alerts. Find that alert by the name and edit it there, then come back and review.

No Hello World: SSH to the EC2 instance and find the node process by

```
ps aux | grep node
```

You might see something like this:

```
[ec2-user@ip-172-31-2-188 ~]$ ps aux | grep 'node'
root      2668  1.8  2.6 909804 27416 ?        Sl   03:52   0:00 node /home/ec2-us
er/hello-world-server.js
ec2-user  2681  0.0  0.2 110460  2188 pts/0    S+   03:52   0:00 grep --color=auto
 node
```

Also, check security group to see if port 80 is open for HTTP.