
Bab 6

AWS Basic Server Production Part 2



Detail Materi



Pengenalan Elastic Load Balancer

Indikator :

Mengenal Elastic Load Balancer, Topologi dan
Roadmap yang akan dibangun, Konfigurasi Webserver
EC2, Pengenalan Auto Scaling



Pengenalan Auto Scaling

Modul Sekolah DevOps Cilsy

Hak Cipta © 2020 **PT. Cilsy Fiolution Indonesia**

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronis maupun mekanis, termasuk mecopy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Adi Saputra, Irfan Herfiandana & Tresna Widiyaman
Editor: Taufik Maulana, Muhammad Fakhri Abdillah, Rizal Rahman & Tresna Widiyaman
Revisi Batch 9

Penerbit : **PT. Cilsy Fiolution Indonesia**
Web Site : <https://cilsyfiolution.com> , <https://devops.cilsy.id>

Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)

Daftar Isi

CoverModul Sekolah DevOps Cilsy.....	2
6. AWS Basic Server Production Part 2.....	5
Learning Outcomes.....	5
Outline Materi.....	5
6.1. Elastic Load Balancer.....	6
6.1.1. Apa Itu Elastic Load Balancer ?.....	6
6.1.2. Model-model Elastic Load Balancer.....	7
6.1.2.1. Application Load Balancers.....	7
6.1.2.2. Network Load Balancers.....	8
6.1.2.3. Classic Load Balancers.....	8
6.1.3. Fungsi dan Kegunaan Elastic Load Balancer.....	9
6.1.4. Exercise.....	10
6.2. Roadmap Pembelajaran.....	10
6.3. Praktek Elastic Load Balancer.....	11
6.3.1. Persiapan Instance.....	12
6.3.2. Setup Webserver dan Aplikasi.....	13
6.3.3. Setup Classic Load Balancer.....	14
6.3.4. Setup Application Load Balancer.....	20
6.3.5. Setup Networking Load Balancer.....	26
6.3.6. Exercise.....	33
6.4. Auto Scalling.....	34
6.4.1. Apa itu Auto Scaling ?.....	34
6.4.2. Exercise.....	35
6.5. Praktek AutoScalling.....	35
6.5.1. Persiapan Instance.....	35
6.5.2. Setup AutoDeployment.....	36
6.5.3. Setup AMI.....	38
6.5.4. Setup Launch Configuration.....	39

6.5.5. Konfigurasi Auto Scaling Group.....	42
6.5.6. Exercise.....	55
6.6. Integrasi Load balancing dan Auto Scaling.....	56
6.7. Summary.....	63

6.

AWS Basic Server Production Part 2

Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami Konsep Load Balancing pada AWS
2. Melakukan Konfigurasi Load balancing Pada Server Production
3. Memahami Konsep Auto Scalling pada AWS
4. Melakukan Konfigurasi Auto Scalling Pada Server Production
5. Melakukan Integrasi Auto Scalling dengan Load Balancer pada Server

Outline Materi

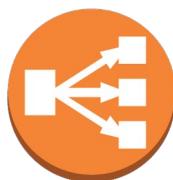
1. Pengenalan Amazon Load Balancer
2. Konfigurasu Load Balancer
3. Pengenalan Auto Scalling Group
4. Konfigurasi Auto Scalling Group
5. Integrasi Auto Scalling dan Load balancer

6.1. Elastic Load Balancer

6.1.1. Apa Itu Elastic Load Balancer ?

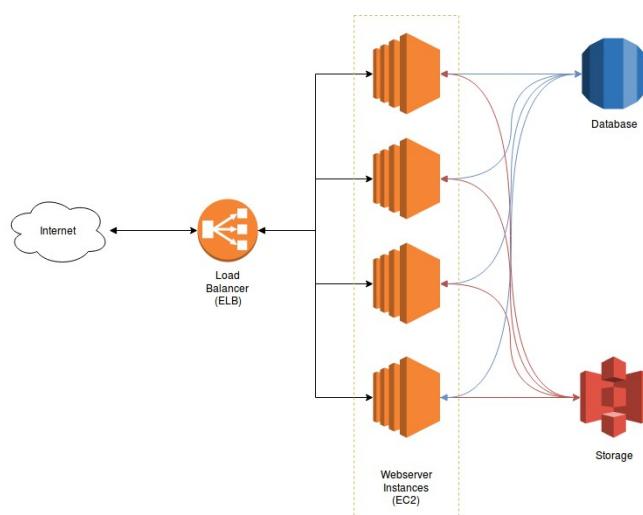
Elastic Load Balancing (ELB) adalah layanan yang masih satu bagian dengan **Amazon EC2**, layanan ini berfungsi untuk menyeimbangkan beban instance yang kita miliki. ELB mengatur dan mengarahkan setiap traffic yang masuk untuk ditugaskan kepada instance yang memiliki traffic yang rendah/tidak sibuk.

Semua traffic dari luar akan masuk kedalam ELB, maka dari itu ELB memegang public DNS yang dapat kita gunakan untuk record CNAME yang nantinya akan mengarah ke server yang trafficensya rendah/tidak sibuk tersebut.



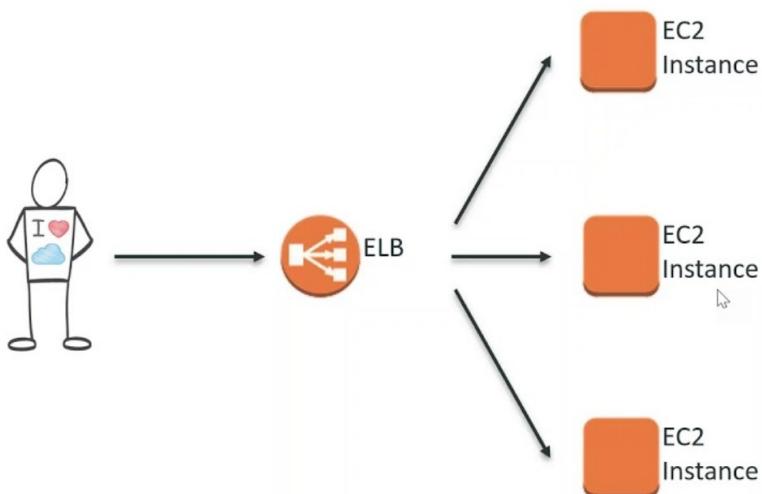
Logo Elastic Load Balancer

Elastic Load Balancing memiliki fasilitas **health check** sehingga pada saat salah satu EC2 Instance mengalami down maka dapat langsung di ketahui dan juga diarahkan ke server EC2 lain yang sedang tidak bermasalah. Load balancer kita akan menjadi alamat yang di tuju oleh pengguna, dengan demikian kita dapat dengan mudah mengatur server Instance.



Ilustrasi prinsip kerja load balancer

Jadi apabila kita menambahkan dan menghapus EC2 Instance, ini tidak akan menjadi masalah pada alamat yang kita tuju karena kita menggunakan alamat load balance, dan jika beban traffic tiba-tiba meningkat maka kita dapat membuat EC2 bertambah secara otomatis

*Contoh Penggunaan ELB pada EC2 Instance*

6.1.2. Model-model Elastic Load Balancer

Selain daripada kelebihannya yang dapat mengurangi beban traffic yang dialihkan ke instance lain, ELB ini memiliki beberapa macam model yang memiliki fungsinya tersendiri, model tersebut adalah sebagai berikut.

6.1.2.1. Application Load Balancers

Application Load Balancers beroperasi pada request level (layer 7), mengarahkan lalu lintas ke target Instance, container, dan alamat IP berdasarkan konten permintaan. Ideal untuk digunakan pada load balancing lanjutan dari traffic HTTP dan HTTPS.

Application Load Balancer menyediakan request routing lanjutan yang ditargetkan pada arsitektur pengiriman aplikasi modern, termasuk microservices dan aplikasi berbasis container. Application Load Balancer menyederhanakan dan meningkatkan keamanan aplikasi kita dengan memastikan bahwa cipher dan protokol SSL / TLS terbaru digunakan setiap saat.

6.1.2.2. Network Load Balancers

Network Load Balancer beroperasi pada level koneksi (Layer 4), melakukan routing koneksi ke target instance, container dan alamat IP berdasarkan data protokol IP. Ideal untuk digunakan pada load balancing traffic TCP.

Network Load Balancer mampu menangani jutaan permintaan per detik sembari mempertahankan **ultra-low latencies**. Network Load Balancer dioptimalkan untuk menangani pola lalu lintas yang tiba-tiba dan mudah berubah saat menggunakan satu alamat IP statis per Availability Zone. Model ini dapat terintegrasi dengan layanan AWS populer lainnya seperti Auto Scaling, Elastic Container Service (ECS), dan Amazon CloudFormation.

6.1.2.3. Classic Load Balancers

Classic Load Balancer menyediakan load balancing dasar di beberapa instance dan beroperasi pada level request dan level connection. Classic Load Balancer ditujukan untuk aplikasi yang dibangun dalam jaringan EC2-Classic. Application Load Balancer direkomendasikan untuk Layer 7 dan Network Load Balancer untuk Layer 4 saat menggunakan Virtual Private Cloud (VPC).

Untuk detail dari berbandingan ketiga model load balancing tersebut, kita dapat lihat pada tabel perbandingan dibawah ini.

Feature	Application Load Balancer	Network Load Balancer	Classic Load Balancer
Protocols	HTTP, HTTPS	TCP	TCP, SSL HTTP, HTTPS
Platforms	VPC	VPC	EC2-Classic, VPC
Health checks	✓	✓	✓
CloudWatch metrics	✓	✓	✓
Logging	✓	✓	✓
Zonal fail-over	✓	✓	✓
Connection draining (deregistration delay)	✓	✓	✓
Load Balancing to multiple ports on the same instance	✓	✓	
WebSockets	✓	✓	
IP addresses as targets	✓	✓	
Loadbalancer deletion protection	✓	✓	
Path-Based Routing	✓		
Host-Based Routing	✓		
Native HTTP/2	✓		
Configurable idle connection timeout	✓		✓
Cross-zone load balancing	✓	✓	✓
SSL offloading	✓		✓
Server Name Indication (SNI)	✓		
Sticky sessions	✓		✓
Back-end server encryption	✓		✓
Static IP		✓	
Elastic IP address		✓	
Preserve Source IP address		✓	
Resource-based IAM permissions	✓	✓	✓
Tag-based IAM permissions	✓	✓	
Slow start	✓		
User authentication	✓		
Redirects	✓		
Fixed response	✓		

Table Perbandingan Komponen ELB

6.1.3. Fungsi dan Kegunaan Elastic Load Balancer

Selain berbagai macam model dari ELB, adapula beberapa keuntungan yang akan kita dapatkan apabila menggunakan layanan ELB tersebut. Berikut beberapa keuntungan menggunakan Elastic Load Balancer.

1. Membagi beban traffic ke beberapa Instance / Server
2. Mendeteksi status Healthy Check EC2 instance-instance
3. Dapat di gunakan sebagai Auto Scaling EC2 ketika server banyak beban traffic

6.1.4. Exercise

1. Jelaskan kembali yang dimaksud dengan Elastic Load Balancer !
2. Apakah yang menjadi perbedaan antara ketiga model ELB ?

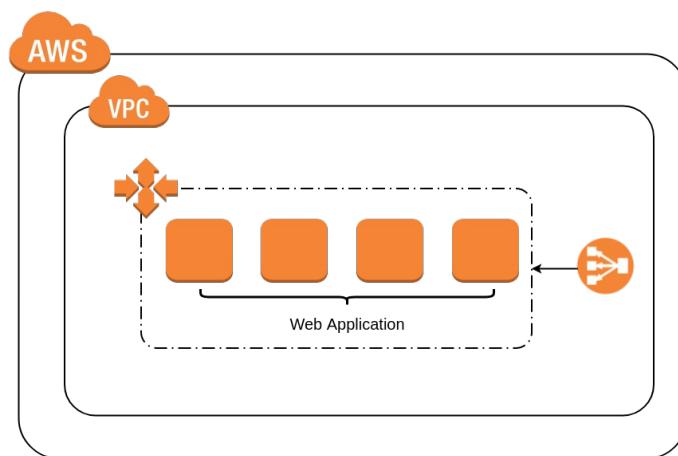
6.2. Roadmap Pembelajaran



Roadmap Pembelajaran

Jika kita lihat pada roadmap yang sudah kita bahas sebelumnya, kita kemarin sudah melewati **tahap pertama sampai dengan tahap ke-empat** yaitu **Membuat Webserver di Production Server EC2**. Sekarang kita masuk **tahap keempat** yaitu **Memasang Load Balancer dan Auto Scaling pada Server Production.**

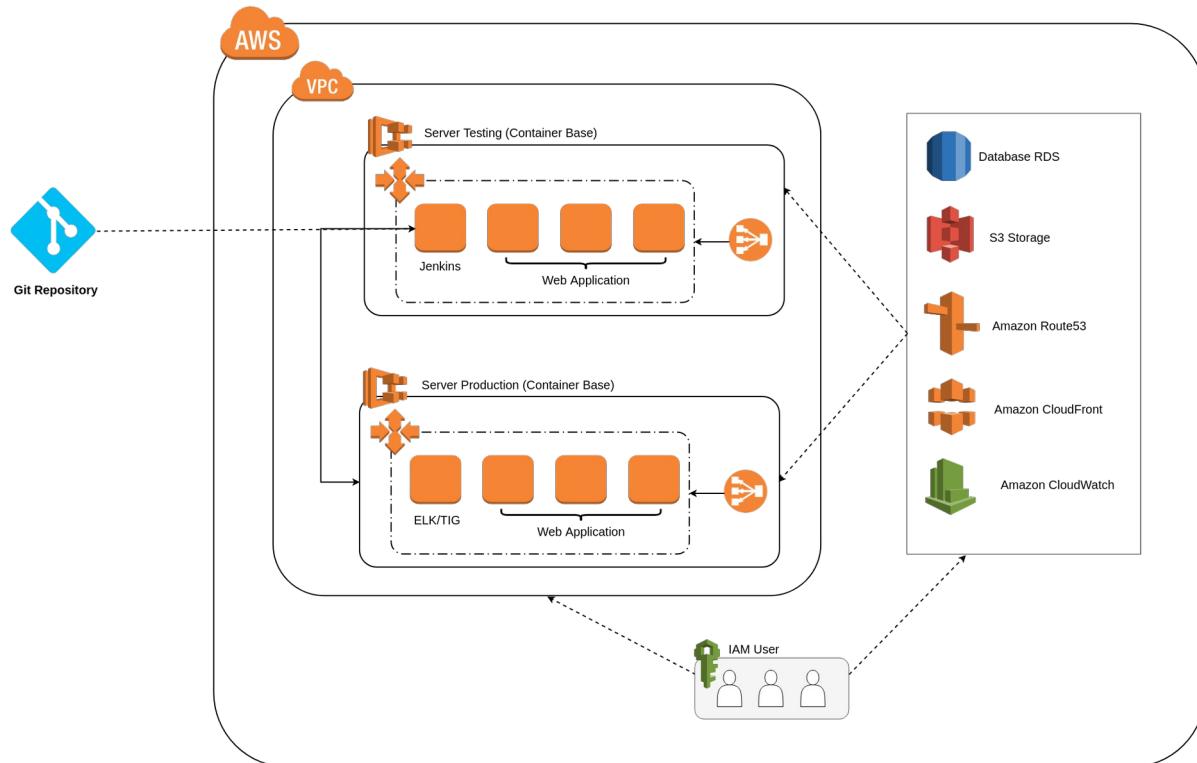
Berdasarkan topologinya sekarang, kita akan membangun infrastructure didalam layanan AWS menggunakan EC2 Instance, Load Balancing dan Auto Scaling Group.



Roadmap Topologi Load Balancing dan Auto Scaling

Topologi diatas merupakan sebagian dari topologi total yang akan kita buat pada Platfrom AWS. Dimana nanti sedikit demi sedikit yang sudah kita buat akan saling berintegrasi dan membuat infrastructure yang sempurna seperti hasil akhir dibawah ini.

AWS Environment Sekolah DevOps Cilsy



Gambar topologi AWS

6.3. Praktek Elastic Load Balancer

Pada bagian ini kita akan melakukan Setup pada Elastic Load Balancer yang dikombinasikan pada EC2 Instance, disini kita akan coba Setup pada ketiga macam model load balancer tersebut yaitu Application Load Balancer, Network Load Balancer, Classic Load Balancer.

6.3.1. Persiapan Instance

Sebelum melakukan setup pada Elastic Load Balancer, pertama kita harus menyiapkan terlebih dahulu 2 buah instance yang sudah terinstall webserver sebagai server production kita. Karena syarat minimal sebuah load balancer adalah dua buah instance, maka setelah membuat kedua instance tersebut kita bisa dengan mudah mengecek keberhasilan dari konfigurasi yang sudah kita buat.

Disini kami menganggap kalian sudah bisa membuat sebuah instance baru di AWS. Apabila kalian masih kesulitan pada saat membuat instance, kami sarankan untuk kembali membaca bab sebelumnya.

Berikut merupakan rincian instance yang harus kalian buat :

Instance 1

- Operating System : **Ubuntu 18.04**
- Name Tag : **namapeserta Server Production ELB1**
- VPC : Gunakan yang sudah dibuat sebelumnya
- Instance Type : **t2.micro**
- Sisanya bisa default disesuaikan

Instance 2

- Operating System : **Ubuntu 18.04**
- Name Tag : **namapeserta Server Production ELB2**
- VPC : Gunakan yang sudah dibuat sebelumnya
- Instance Type : **t2.micro**
- Sisanya bisa default disesuaikan

Apabila kalian sudah selesai membuat kedua buah instance tersebut, maka bisa kita lihat hasilnya akan menjadi seperti dibawah ini.

Instances (8) Info		C	Connect	Instance state ▼	Actions ▼	Launch instances	▼
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	A
<input type="checkbox"/>	-	I-010d04c7f2176eac9	Stopped Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	test1	I-03a1238bca488dbe5	Terminated Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	test2	I-0db0109aa3fb2d962	Terminated Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	Cilsy Server Production ELB1	I-01bf423943784249e	Running Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	Cilsy Server Production ELB2	I-04c9b1b2826a9692f	Running Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	Jenkins	I-0af73829c469b8a51	Stopped Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	Git-Jenkins-Docker	I-0390acbfb3454d9bb4	Stopped Q Q	t2.micro	-	No alarms	+
<input type="checkbox"/>	docker	I-0668fa96eb4e2dc5	Stopped Q Q	t2.micro	-	No alarms	+

Hasil persiapan instance

6.3.2. Setup Webserver dan Aplikasi

Pada bagian ini kita akan melakukan setup webserver pada kedua instance yang sudah kita buat tadi, kalian dapat menggunakan Script automasi yang sudah kalian buat pada bab sebelumnya. Adapun beberapa requirement yang harus di install adalah seperti berikut :

- Apache2
- php dan php-mysql
- mysql-server

Setelah melakukan setup pada webserver, sekarang kita akan melakukan setup pada web aplikasi pesbuk. Sama seperti webserver, kalian dapat menggunakan script karena kita akan menginstallkan webaplikasi seperti biasa.

Jangan lupa lakukan konfigurasi pada database dengan nama user **devopscilsy** dan database **dbsosmed** sehingga nantinya bisa kita akses.

Setelah setup berhasil, **buat satu sampai dua user yang sama detailnya** pada kedua instance sehingga nanti pada saat menggunakan load balancing, season login pada web aplikasi akan tetap sesuai dengan season sebelumnya.

Bila perlu edit bagian tittle pada file **index.php** dan **dashboard.php** di salah satu server instance untuk nantinya melihat perbedaan pada kedua server ketika load balance

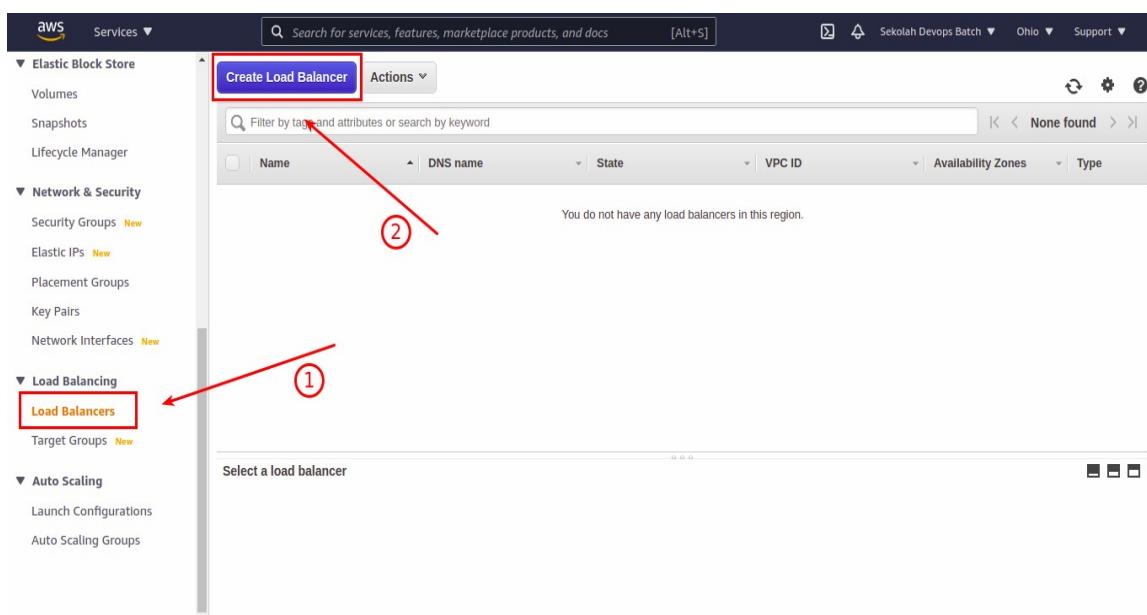


Ini sedikit kurang efisien karena kita menggunakan dua database pada kedua server, jangan khawatir karena nanti kita akan belajar Amazon RDS pada bab selanjutnya. Jadi kita bisa gunakan satu database saja untuk beberapa web aplikasi.

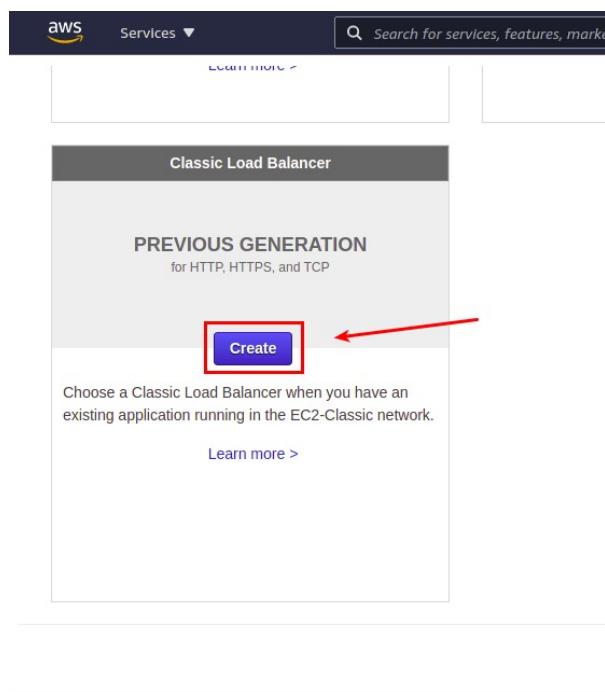
6.3.3. Setup Classic Load Balancer

Pertama kita akan coba melakukan **setup pada Classic Load Balancer**. Seperti yang sudah kita bahas sebelumnya, Classic Load Balancer ini mengatur traffic berdasarkan pada level http, https dan TCP atau level port. Juga melakukan pengecekan berdasarkan status http atau https. Berikut tahapan untuk melakukan setup Classic Load Balancer :

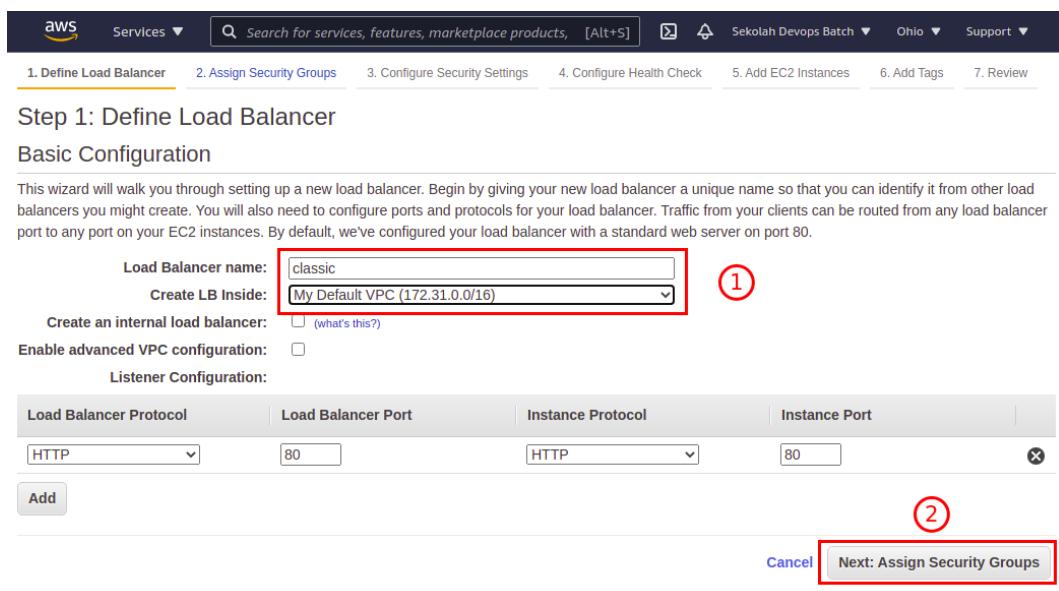
Pertama kita masuk kedalam **dashboard EC2**, setelah pilih menu **Load Balancers** pada bagian panel samping. Selanjutnya pilih tombol **Create Load Balancer** untuk membuat ELB baru.



Kita akan diberikan pilihan **model Load Balancer** mana yang akan kita buat, disini kita scroll ke bawah dan memilih **Classic Load Balancer**, lalu klik tombol **Create**.



Setelah itu kita akan diarahkan ke menu **Basic Configuration**. Isikan Load Balancer Name : classic. Pada bagian **Create LB inside**, pilih vpc dimana Instance berada. Lalu tambahkan **protocol http** dengan port 80 atau **https** dengan port 443. Klik tombol **Next : Assign Security Group** untuk melanjutkan.



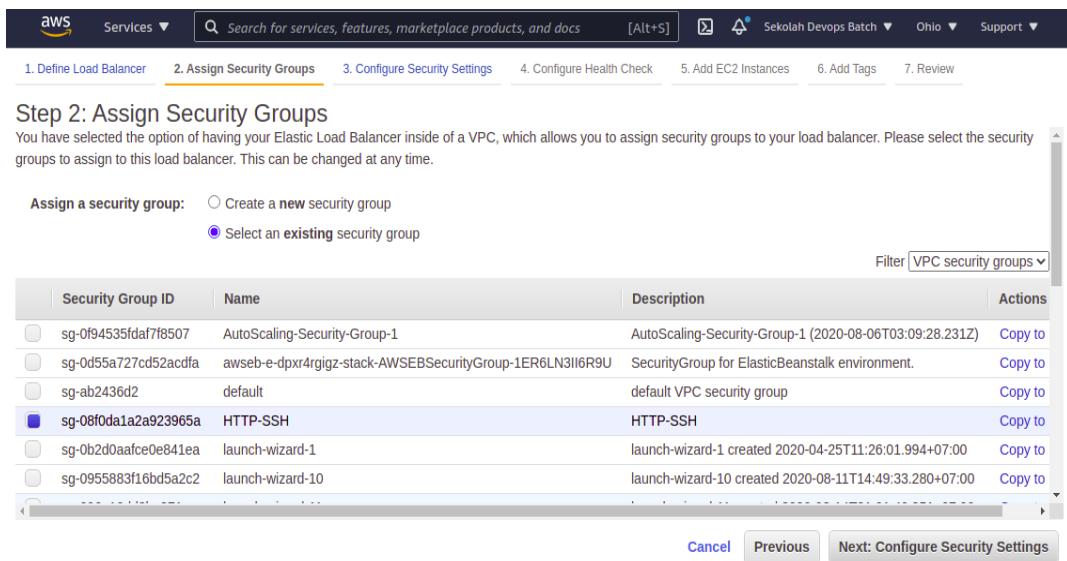
Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name:	classic	(1)	
Create LB Inside:	My Default VPC (172.31.0.0/16)	(1)	
Create an internal load balancer:	<input type="checkbox"/>		
Enable advanced VPC configuration:	<input type="checkbox"/>		
Listener Configuration:			
Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80
<input type="button" value="Add"/> (2)			
<input type="button" value="Cancel"/> <input style="border: 2px solid red; background-color: #e6f2ff; color: black; padding: 2px 10px; margin-left: 10px;" type="button" value="Next: Assign Security Groups"/>			

Selanjutnya kita masuk ke menu **security group**, pilih security group yang sudah kita buat sebelumnya kemudian klik **Next Configure Security Settings**.



Step 2: Assign Security Groups

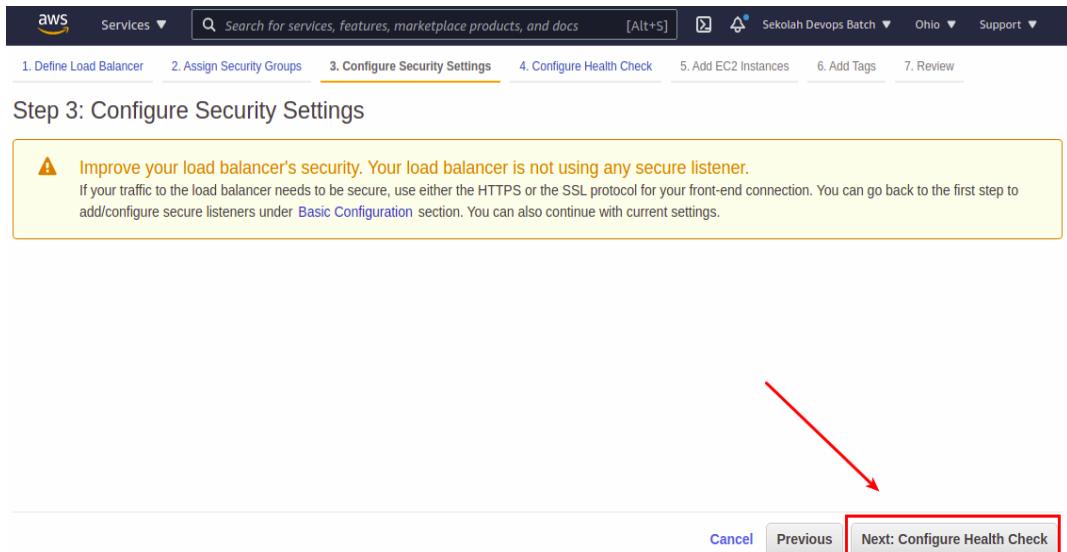
You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.

Assign a security group: Select an existing security group

Security Group ID	Name	Description	Actions
sg-0f94535fdfaf7f8507	AutoScaling-Security-Group-1	AutoScaling-Security-Group-1 (2020-08-06T03:09:23Z)	Copy to
sg-0d55a727cd52acdfa	awseb-e-dpxr4rgjz-stack-AWSEBSecurityGroup-1ER6LN3II6R9U	SecurityGroup for ElasticBeanstalk environment.	Copy to
sg-ab2436d2	default	default VPC security group	Copy to
sg-08f0da1a2a923965a	HTTP-SSH	HTTP-SSH	Copy to
sg-0b2d0aafce0e841ea	launch-wizard-1	launch-wizard-1 created 2020-04-25T11:26:01.994+07:00	Copy to
sg-0955883f16bd5a2c2	launch-wizard-10	launch-wizard-10 created 2020-08-11T14:49:33.280+07:00	Copy to

Cancel Previous Next: Configure Security Settings

Pada bagian selanjutnya langsung saja klik **Next Configure Health Check**.

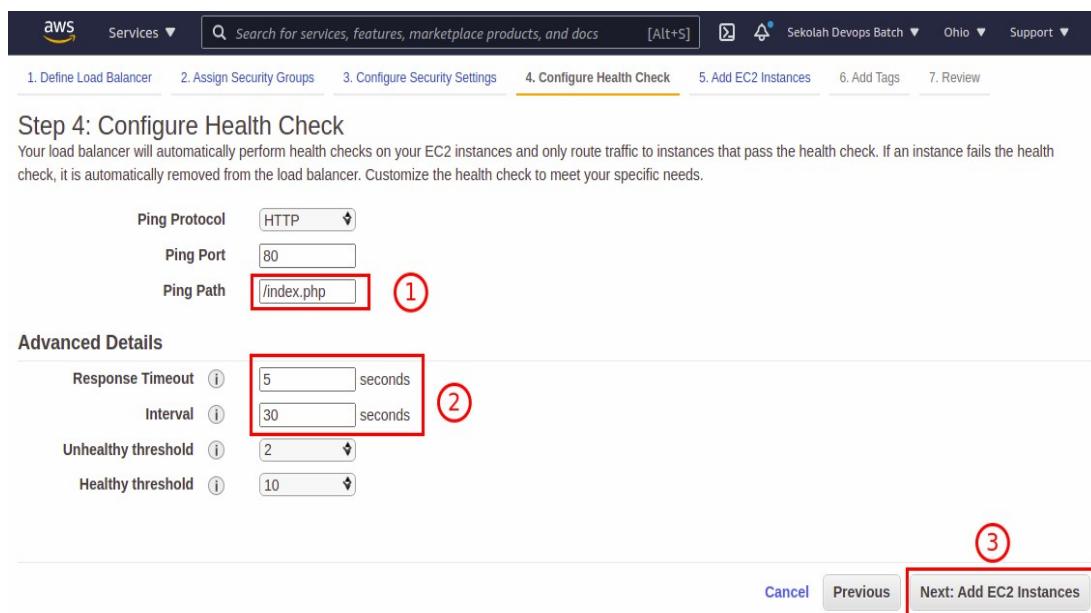


Step 3: Configure Security Settings

⚠ Improve your load balancer's security. Your load balancer is not using any secure listener.
If your traffic to the load balancer needs to be secure, use either the HTTPS or the SSL protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under [Basic Configuration](#) section. You can also continue with current settings.

Cancel Previous Next: Configure Health Check

Selanjutnya kita dapat mengatur **health check** dengan parameter ping protocol HTTP, ping port 80 dan ping path **index.php**. Pada bagian Advanced setting kita dapat mengatur response time agar menjadi 5 second dalam waktu interval 30 second.



Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

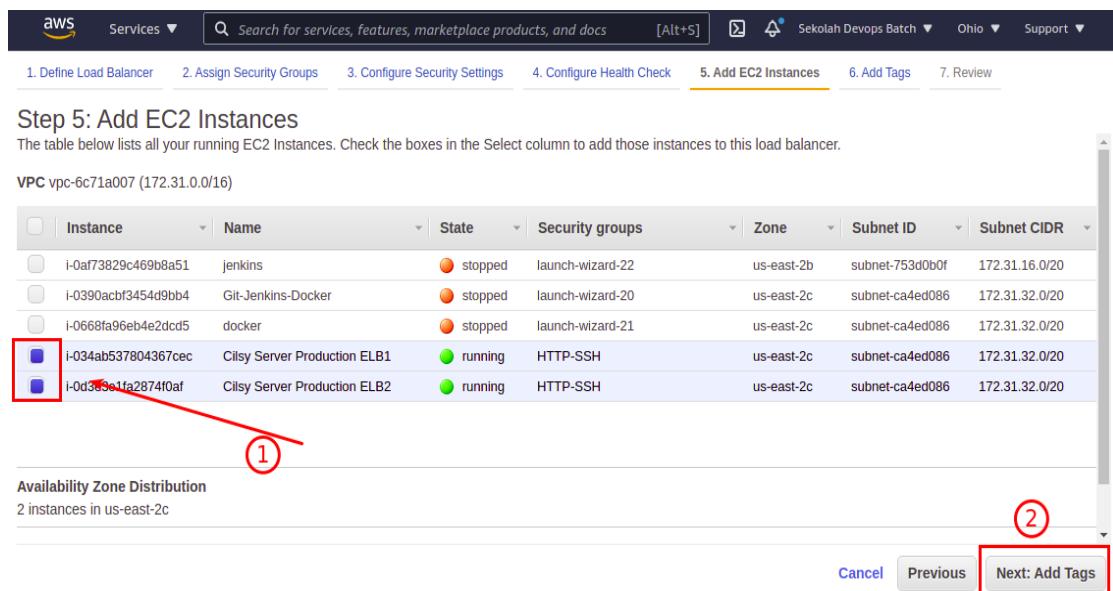
Ping Protocol: HTTP
 Ping Port: 80
 Ping Path: /index.php (1)

Advanced Details

Response Timeout (i)	5 seconds
Interval (i)	30 seconds (2)
Unhealthy threshold (i)	2
Healthy threshold (i)	10

(3) Next: Add EC2 Instances

Selanjutnya kita pilih **Instance** yang telah kita buat untuk dimasukan sebagai **Load Balancer**, pilih dua instance yang sudah kita buat tadi. Setelah selesai klik **Next Add Tags**.



Step 5: Add EC2 Instances

The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

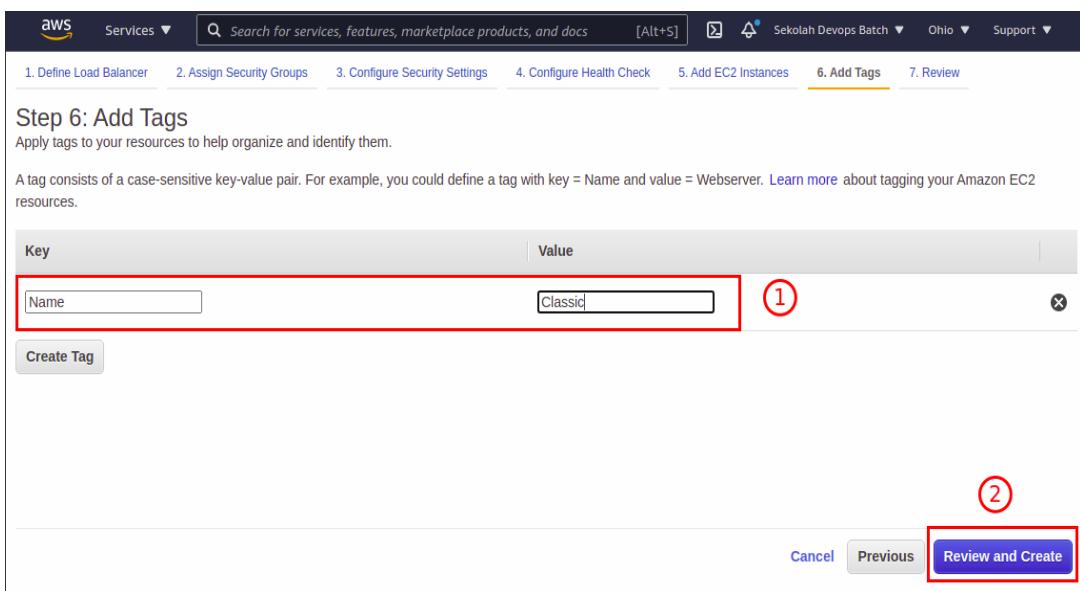
VPC vpc-6c71a007 (172.31.0.0/16)

Select	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
<input type="checkbox"/>	i-0af73829c469b8a51	jenkins	stopped	launch-wizard-22	us-east-2b	subnet-753d0b0f	172.31.16.0/20
<input type="checkbox"/>	i-0390acbfb3454d9bb4	Git-Jenkins-Docker	stopped	launch-wizard-20	us-east-2c	subnet-ca4ed086	172.31.32.0/20
<input type="checkbox"/>	i-0668fa96eb4e2ddc5	docker	stopped	launch-wizard-21	us-east-2c	subnet-ca4ed086	172.31.32.0/20
<input checked="" type="checkbox"/>	i-034ab537804367cec	Cilisy Server Production ELB1	running	HTTP-SSH	us-east-2c	subnet-ca4ed086	172.31.32.0/20
<input checked="" type="checkbox"/>	i-0d3891fa28740af	Cilisy Server Production ELB2	running	HTTP-SSH	us-east-2c	subnet-ca4ed086	172.31.32.0/20

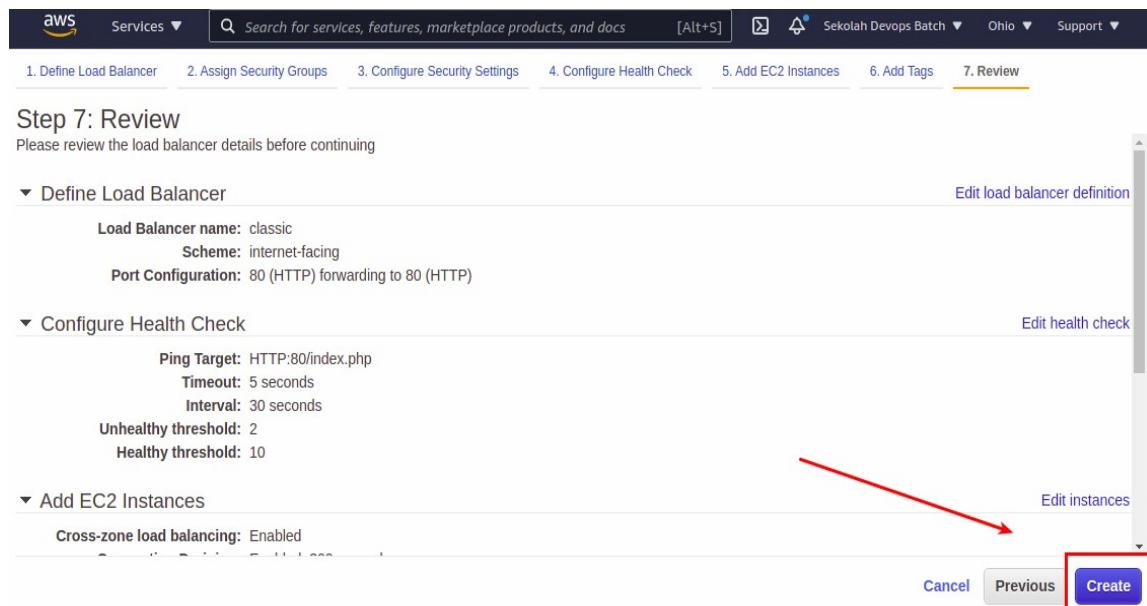
Availability Zone Distribution
2 instances in us-east-2c (1)

(2) Next: Add Tags

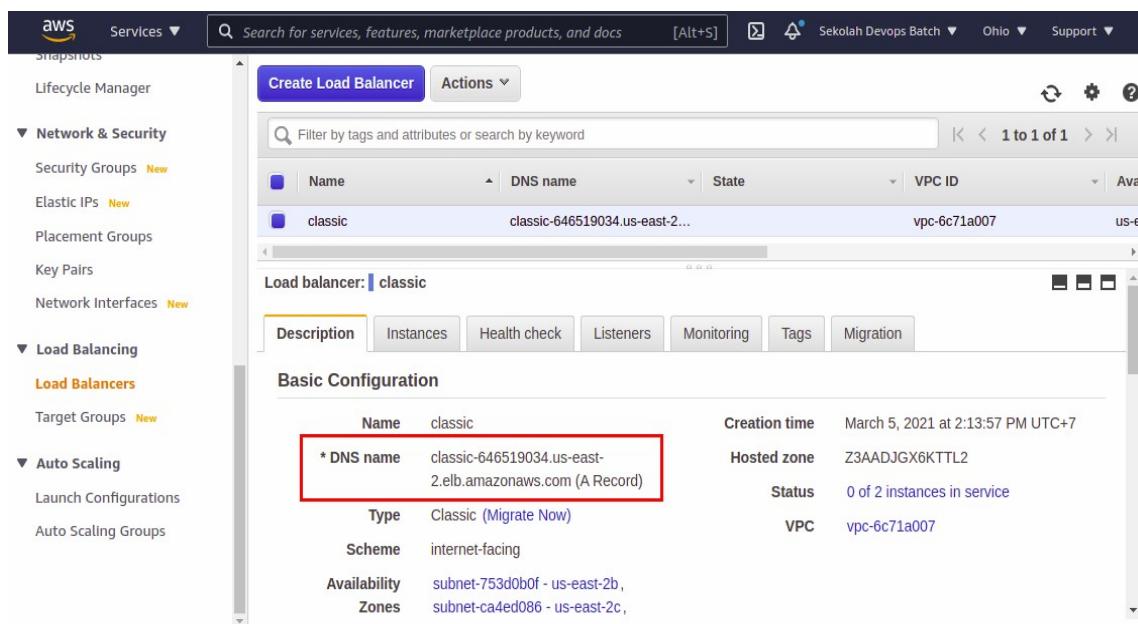
Buat mebuah tag baru dengan menekan tombol **Create Tag** kemudian isi Key dengan **Name** dan Value dengan **classic**, tag ini berguna untuk memberikan nama pada ELB classic.



Setelah semua konfigurasi dan setting telah di lakukan, maka kita data melihat kembali rincian dari sudah kita konfigurasi tadi sebelumnya. Untuk melanjutkan klik **Create**.



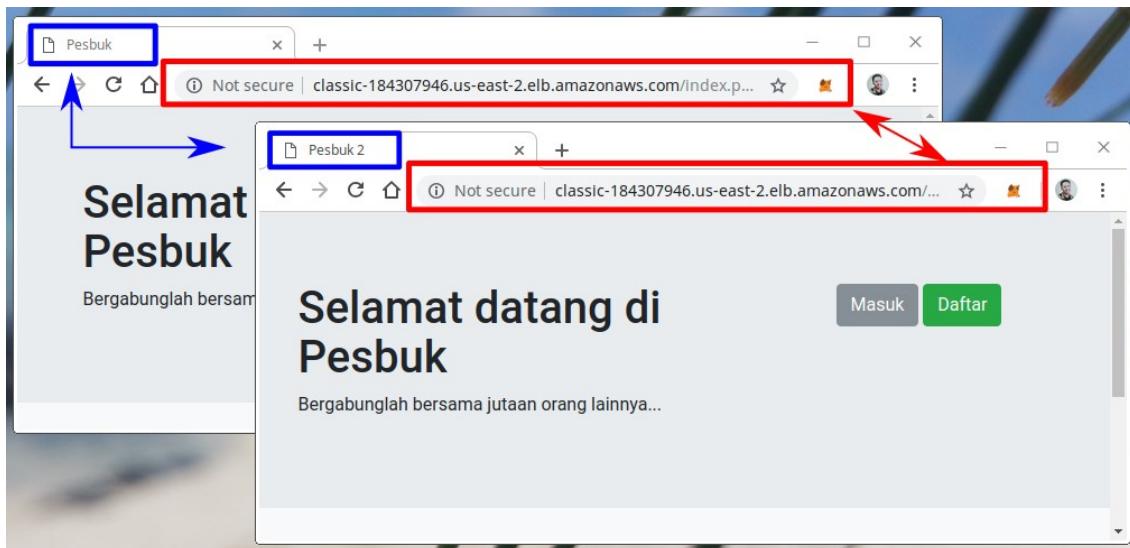
Setelah selesai, kita akan diarahkan ke dalam menu **dashboard Elastic Load Balancer** dan kita dapat lihat konfigurasi yang sudah kita buat tadi. Untuk melihat DNS name url, kita dapat menklik tombol description yang ada di menu bawah ELB.



The screenshot shows the AWS CloudFormation console with the 'classic' stack selected. The 'Outputs' section displays the following information:

Name	Description
classic-646519034.us-east-2.elb.amazonaws.com	The public DNS name of the ELB.

Coba akses alamat DNS Name tersebut. Refresh terus menerus dan lihat halaman webserver kita yang akan berubah ubah tergantung dengan server yang kita dapatkan pada saat kita akses. Kita bisa membedakan pada tittle aplikasi pada kedua server yang berbeda (warna biru).

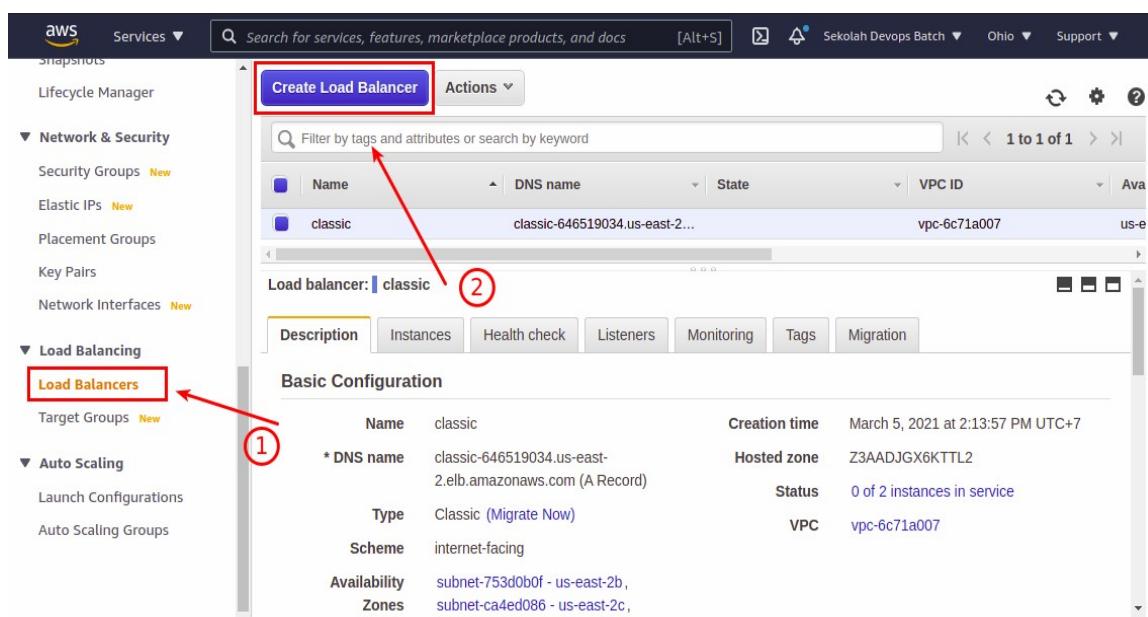


Hasil Classic Load Balancer

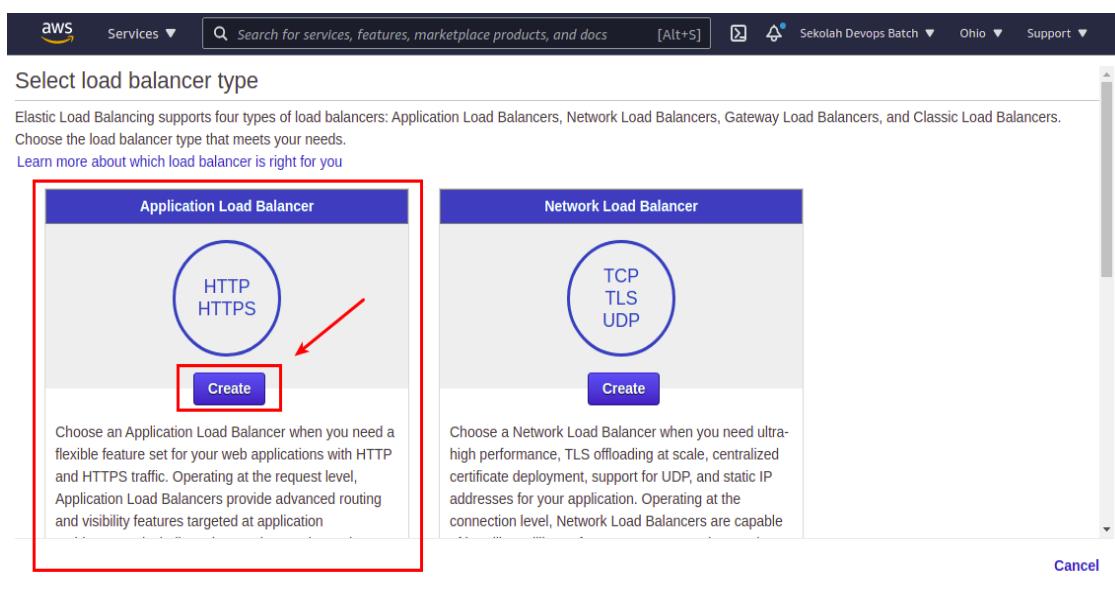
6.3.4. Setup Application Load Balancer

Selanjutnya kita akan lakukan **setup pada Application load balancer**. Seperti yang sudah kita bahas sebelumnya ALB berfungsi untuk mengatur traffic berdasarkan pada level Aplikasi yang digunakan. mengecek aplikasi apakah berjalan dengan baik atau mengalami kendala. Berikut tahapan untuk melakukan setting Application load balancer :

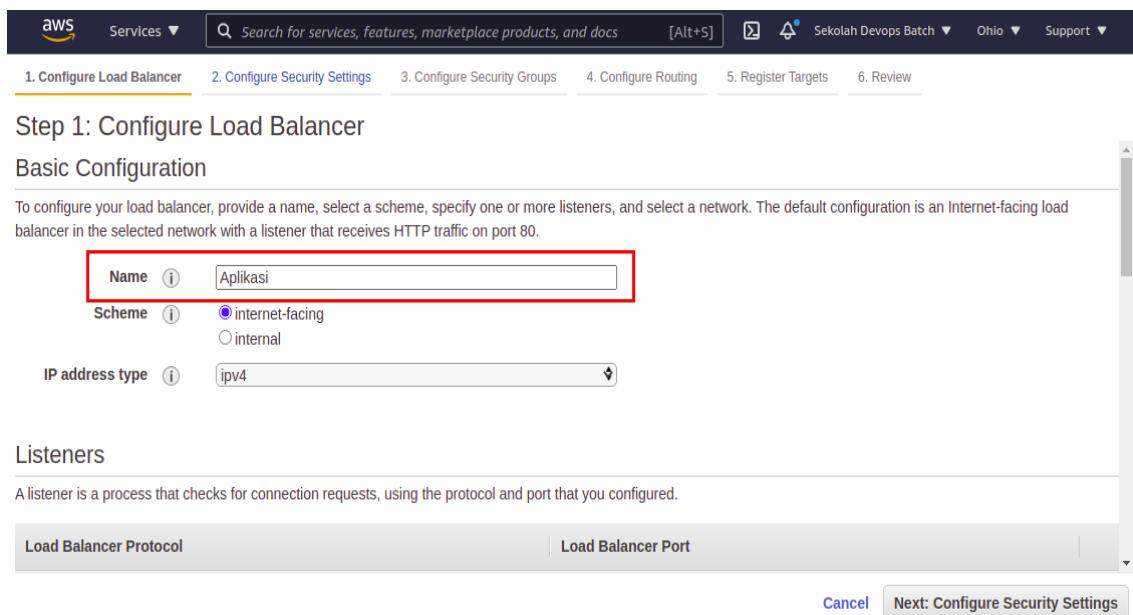
Pertama kita masuk kedalam **dashboard EC2**, setelah pilih menu **Load Balancers** pada bagian panel samping. Setelah itu pilih tombol **Create Load Balancer** untuk membuat ELB baru.



Kita akan diberikan pilihan **model Load Balancer** mana yang akan kita buat, disini kita pilih **Application Load Balancer**, lalu klik tombol **Create**.



Selanjutnya kita masuk pada menu **Configure Load Balancer**. Disini kita isikan beberapa form seperti **Name : Aplikasi**.



Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.

Name	Aplikasi
Scheme	<input checked="" type="radio"/> internet-facing <input type="radio"/> internal
IP address type	ipv4

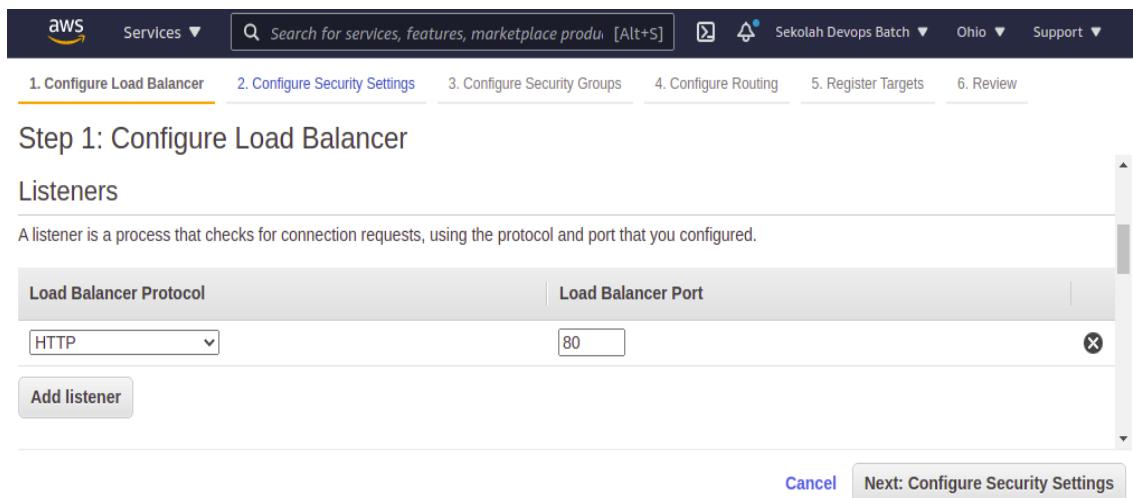
Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
------------------------	--------------------

[Cancel](#) [Next: Configure Security Settings](#)

Scroll ke bawah untuk mengisi protocol HTTP dengan port 80 pada bagian **Listeners**.

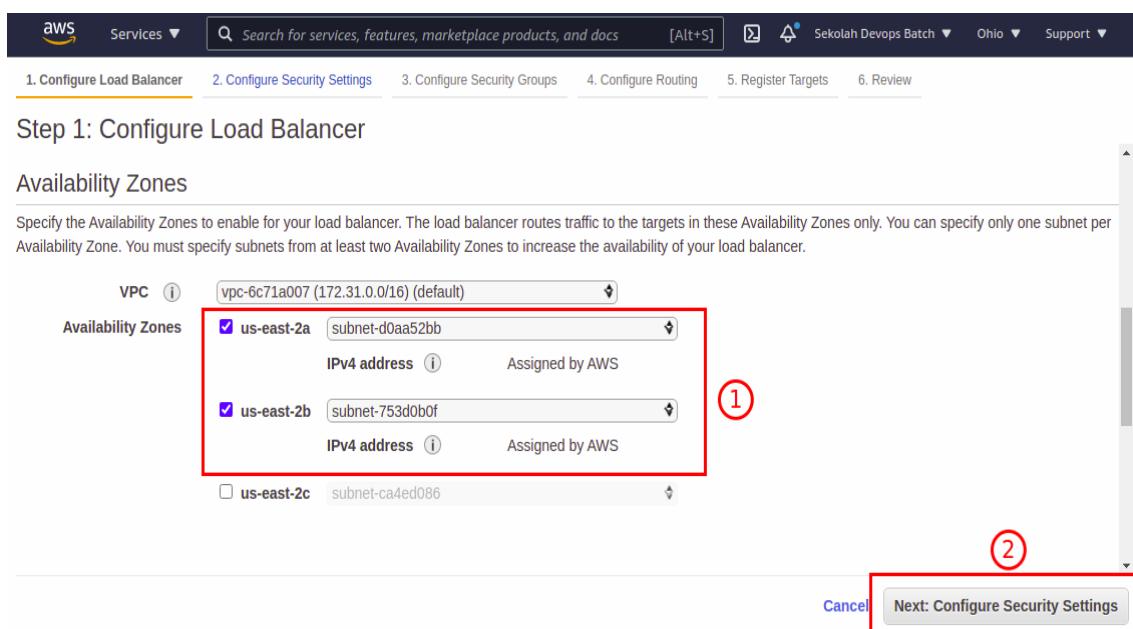


Load Balancer Protocol	Load Balancer Port
HTTP	80

Add listener

Cancel Next: Configure Security Settings

Pada bagian Availability Zone dibawah pilih dua zona yang kita akan gunakan. Lalu klik Next **Configure Security Setting**.

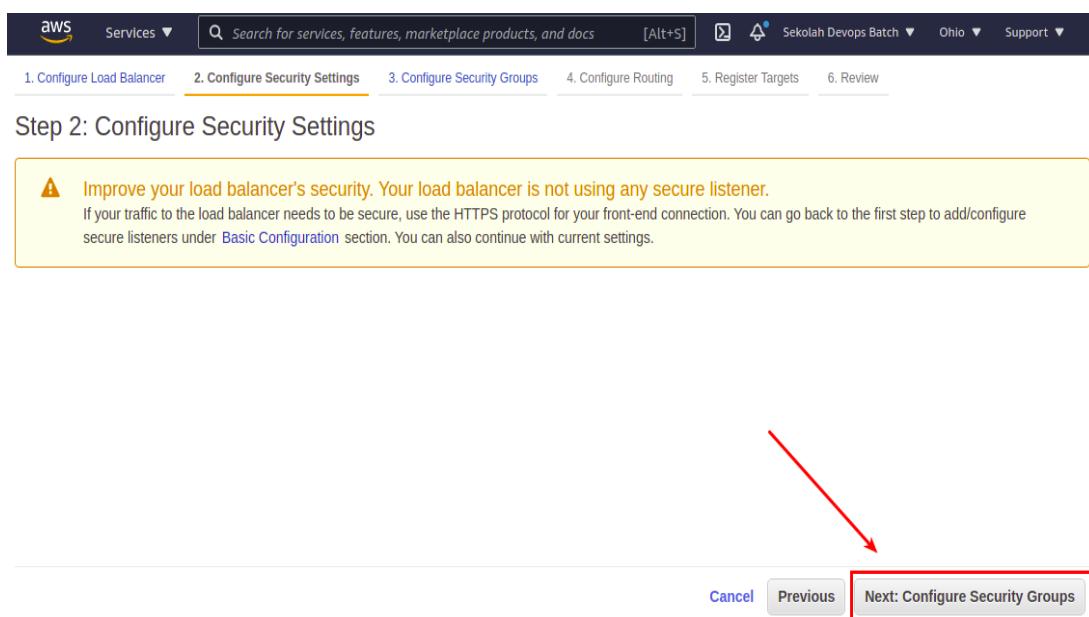


VPC	vpc-6c71a007 (172.31.0.0/16) (default)
Availability Zones	<input checked="" type="checkbox"/> us-east-2a subnet-d0aa52bb IPv4 address Assigned by AWS <input checked="" type="checkbox"/> us-east-2b subnet-753d0b0f IPv4 address Assigned by AWS <input type="checkbox"/> us-east-2c subnet-ca4ed086

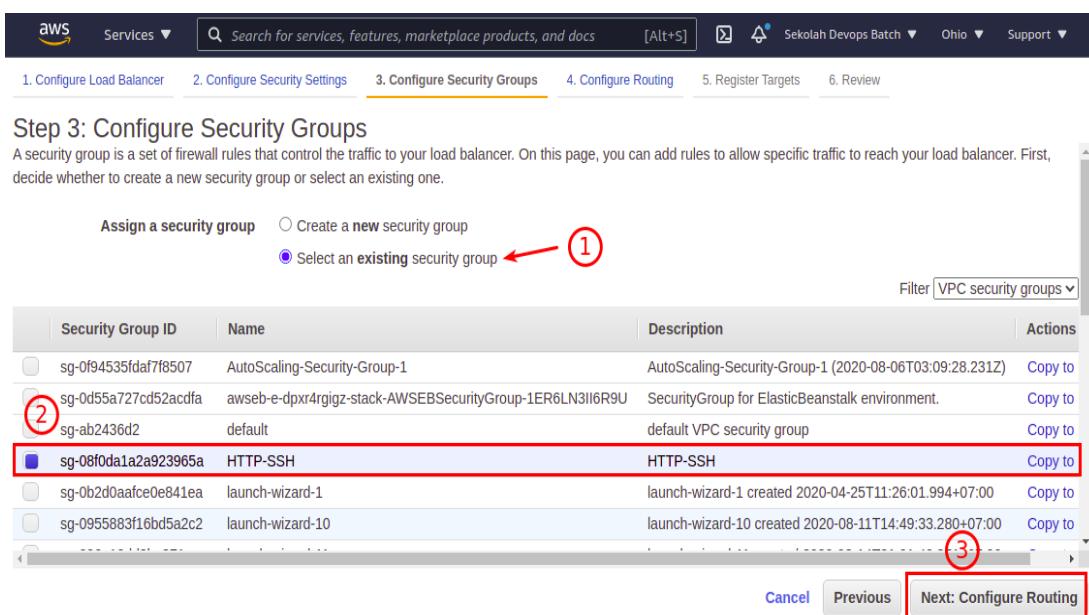
Cancel Next: Configure Security Settings

Pada bagian selanjutnya langsung saja klik **Next Configure Security Group**.





Selanjutnya kita masuk ke menu **security group**, pilih security group yang sudah kita buat sebelumnya kemudian klik **Next Configure Security Settings**.



Selanjutnya buat **target group baru** dengan klik box **Target group > New target group**, kemudian isikan **Name : routing**, lalu pilih protocol dengan HTTP, isikan port dengan 80 lalu target type dengan instance. Pada health check pilih protokol dengan HTTP dan path dengan index.php agar Load Balance bisa mengecek server dari aplikasi yang berjalan.

Kita dapat mengatur path sebagai health checks, pada pengaturan advanced kita dapat atur timeout menjadi 5 second dan interval menjadi 30 second, selanjutnya klik **Next Register Targets**.

Step 4: Configure Routing
Target group

Target group (1) New target group
Name (1) routing

Target type
 Instance
 IP
 Lambda function

Protocol (1) HTTP
Port (1) 80

Protocol version (1)
 HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
 HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
 gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks

Protocol (2) HTTP
Path (2) /index.php

Cancel Previous Next: Register Targets (3)

Selanjutnya pada menu register target kita pilih **Instance** yang telah kita buat untuk dimasukan sebagai **Load Balancer**, pilih dua instance yang sudah kita buat tadi dengan klik **Add to register**. Setelah selesai klik **Next Review**.

Step 5: Register Targets

	Instance	Name	Port	State	Security groups	Zone
No instances available.						

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.

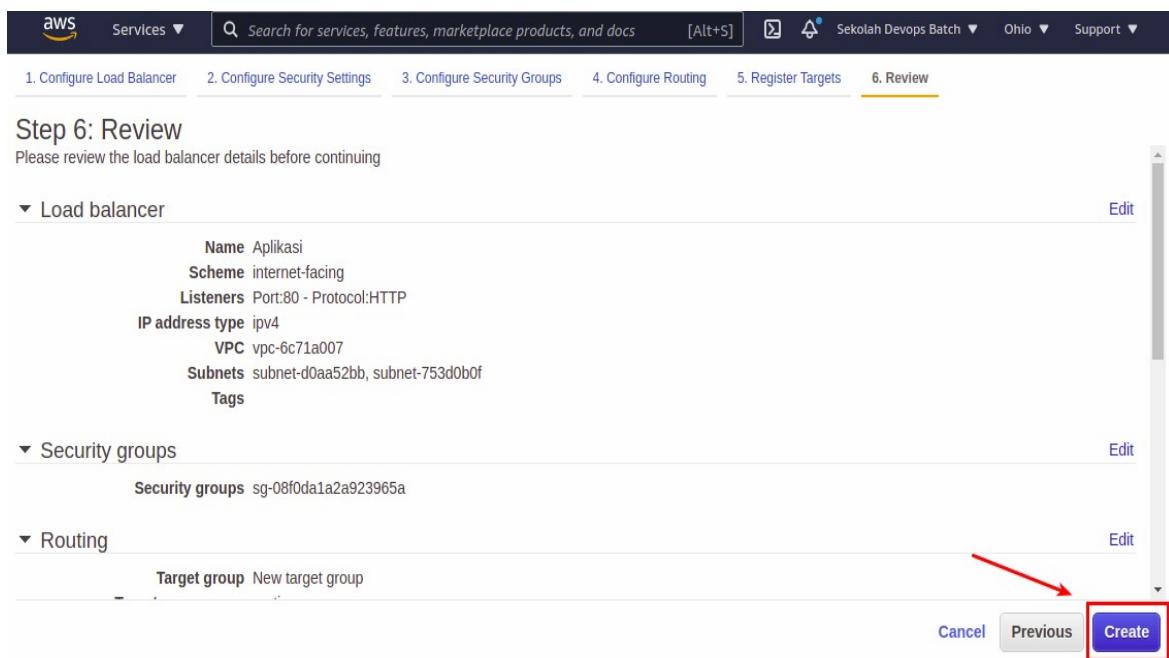
Add to registered on port 80 (2)

Search Instances X

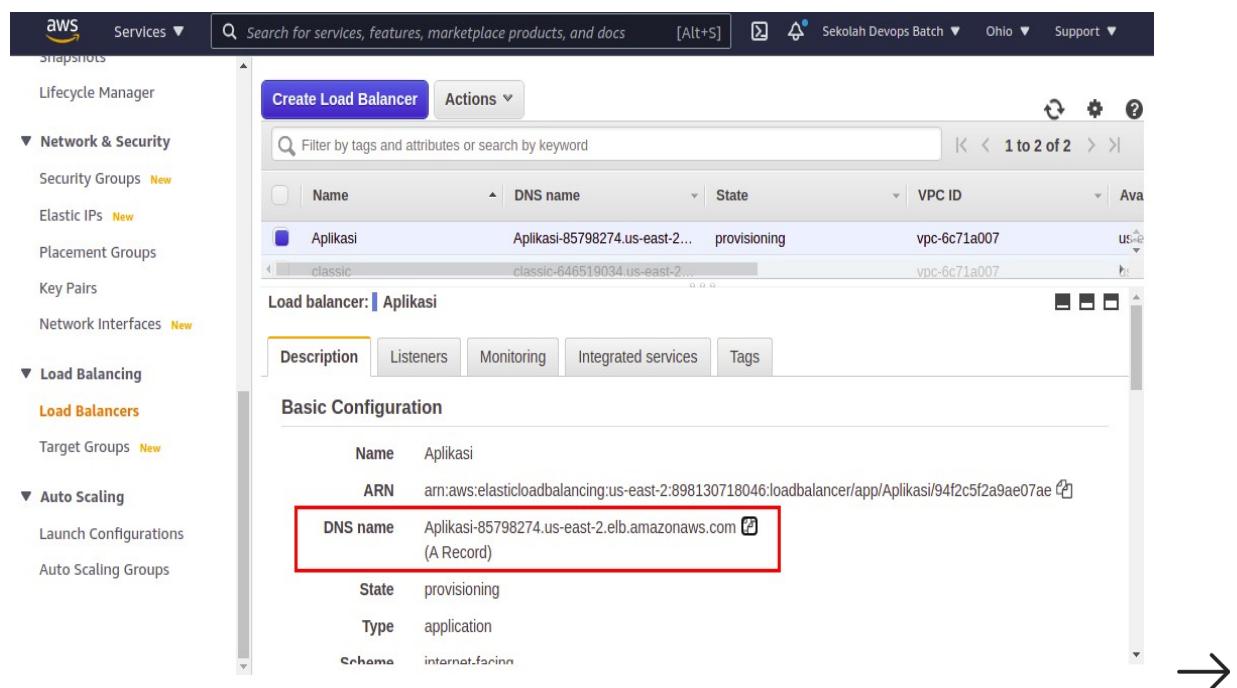
	Instance	Name	State	Security	Zone	Subnet ID	Subnet CIDR
<input checked="" type="checkbox"/>	i-034ab537804...	Cilsy Server Pr...	running	HTTP-SSH	us-east-2c	subnet-ca4ed086	172.31.32.0/20
<input checked="" type="checkbox"/>	i-0d3e3e1fa28...	Cilsy Server Pr...	running	HTTP-SSH	us-east-2c	subnet-ca4ed086	172.31.32.0/20

Cancel Previous Next: Review (3)

Setelah semua konfigurasi dan setting telah di lakukan, maka kita data melihat kembali rincian dari sudah kita konfigurasi tadi sebelumnya. Untuk melanjutkan klik **Create**.

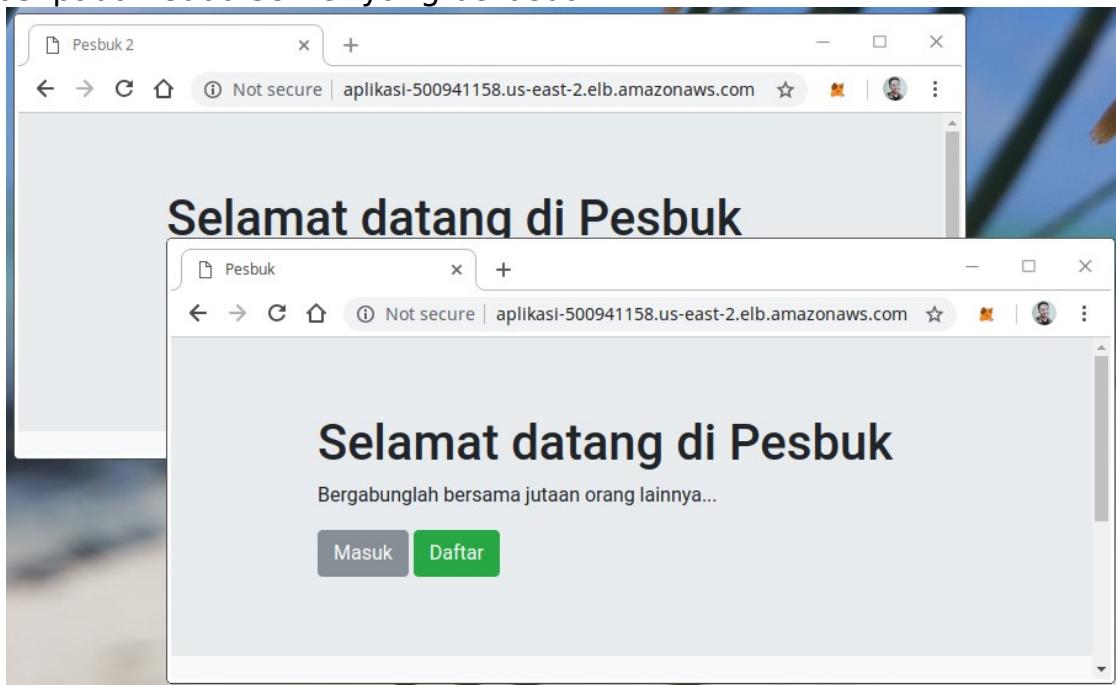


Setelah selesai, kita akan diarahkan ke dalam menu **dashboard Elastic Load Balancer** dan kita dapat lihat konfigurasi yang sudah kita buat tadi. Untuk melihat DNS name url, kita dapat menklik tombol description yang ada di menu bawah ELB.



The screenshot shows the AWS Elastic Load Balancer dashboard. On the left, there's a navigation sidebar with options like Lifecycle Manager, Network & Security, Load Balancing, Auto Scaling, and Snapshots. The main area shows a table of existing load balancers. One row is selected, showing details: Name: Aplikasi, DNS name: Aplikasi-85798274.us-east-2..., State: provisioning, VPC ID: vpc-6c71a007. Below the table, there are tabs for Description, Listeners, Monitoring, Integrated services, and Tags. The 'Description' tab is active. Under 'Basic Configuration', there are fields for Name (Aplikasi), ARN (arn:aws:elasticloadbalancing:us-east-2:898130718046:loadbalancer/app/Aplikasi/94f2c5f2a9ae07ae), DNS name (Aplikasi-85798274.us-east-2.elb.amazonaws.com), State (provisioning), Type (application), and Schema (internet-facing). A large red box highlights the 'DNS name' field.

Coba akses alamat **DNS Name** tersebut. Refresh terus menerus dan lihat halaman webserver kita yang akan berubah ubah tergantung dengan server yang kita dapatkan pada saat kita akses. Kita bisa membedakan pada tittle aplikasi pada kedua server yang berbeda.



Hasil Setup Application Load Balancer

6.3.5. Setup Networking Load Balancer

Model terakhir yang akan kita setup adalah **Netwroking Load Balancer**. Sebelumnya sudah kita bahas mengenai NLB yang berfungsi mengatur traffic berdasarkan pada level http, https dan TCP atau level port. Pengecekan dilakukan berdasarkan status http atau https.

Berikut tahapan untuk melakukan setting Netwroking Load Balancer :

Pertama kita masuk kedalam **dashboard EC2**, setelah pilih menu **Load Balancers** pada bagian panel samping. Selanjutnya pilih tombol **Create Load Balancer** untuk membuat ELB baru.



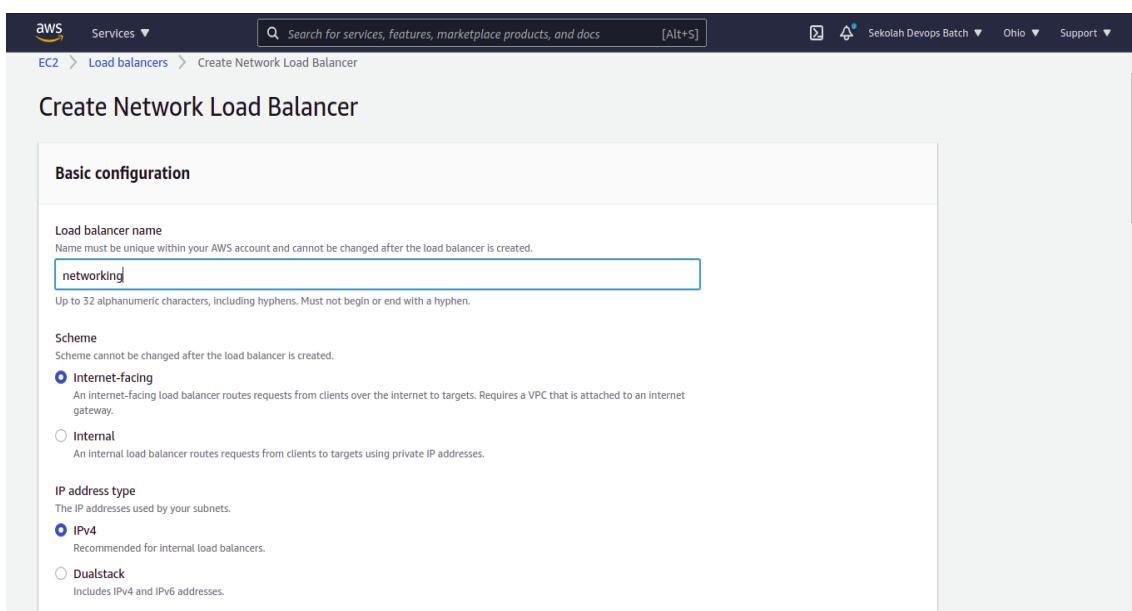
The screenshot shows the AWS CloudFormation service page. On the left, there's a navigation sidebar with sections like Services, Network & Security, Load Balancing, Auto Scaling, and more. Under Load Balancing, the 'Load Balancers' section is selected and highlighted with a red box, with a red number 1 next to it. At the top center, there's a large blue 'Create Load Balancer' button. Below it, the 'Actions' dropdown menu is circled with a red number 2. The main area displays a table of existing load balancers, with one entry named 'Aplikasi' highlighted.

Kita akan diberikan pilihan **model Load Balancer** mana yang akan kita buat, disini kita pilih **Network Load Balancer**, lalu klik tombol **Create**.

The screenshot shows a modal dialog titled 'Select load balancer type'. It contains two sections: 'Application Load Balancer' and 'Network Load Balancer'. The 'Network Load Balancer' section is highlighted with a red box and a red arrow points to its 'Create' button. Both sections provide descriptions of their features and target use cases.

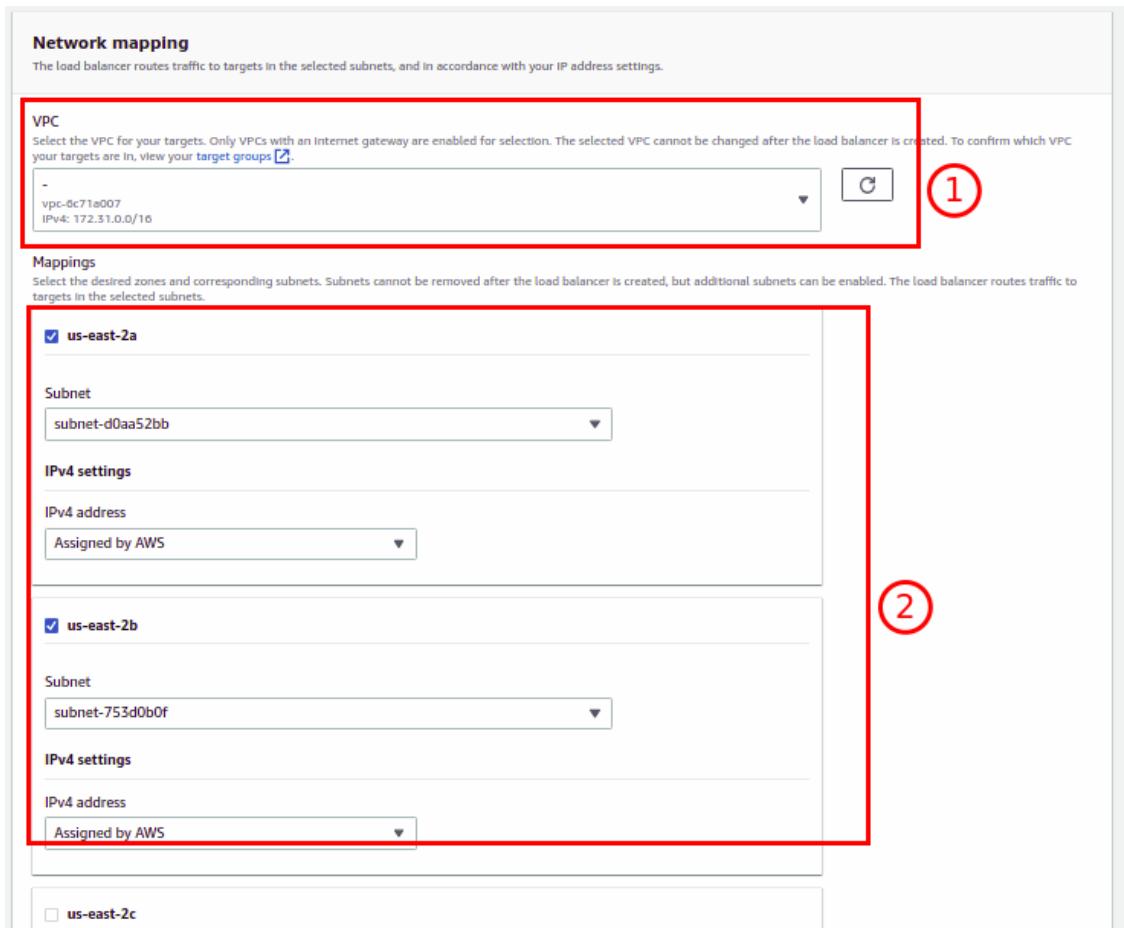
Disini kita isikan beberapa form seperti **Name : networking**





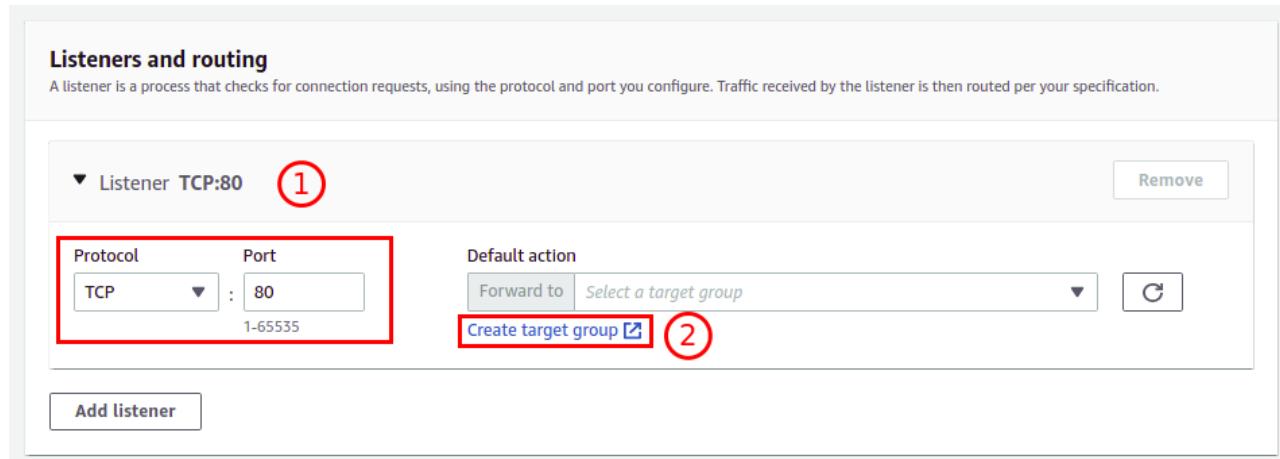
The screenshot shows the 'Create Network Load Balancer' page in the AWS Management Console. Under 'Basic configuration', the 'Load balancer name' is set to 'networking'. The 'Scheme' is set to 'Internet-facing'. The 'IP address type' is set to 'IPv4'. There are sections for 'Mappings' and 'SSL certificate' which are currently empty.

Jika discroll kebawah, isikan **VPC** tempat **Instance** berada lalu pilih 2 zona yang menjadi load balancer.



The screenshot shows the 'Network mapping' section of the AWS Network Load Balancer creation wizard. A red box labeled '1' highlights the 'VPC' dropdown, which contains a single entry: 'vpc-0c71a007 IPv4: 172.31.0.0/16'. A red box labeled '2' highlights the 'Mappings' section, which lists two availability zones: 'us-east-2a' and 'us-east-2b'. Each zone has its own 'Subnet' dropdown (containing 'subnet-d0aa52bb' and 'subnet-753d0b0f') and 'IPv4 settings' section ('IPv4 address: Assigned by AWS'). The 'us-east-2c' row is partially visible at the bottom.

Pada bagian **Listeners and routing**, kita isikan **Protocol TCP** dengan port **80**. lalu pada Default action, kita harus buat terlebih dahulu target group dengan mengeklik link **Create target Group**.



Listeners and routing
 A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specification.

▼ Listener TCP:80 (1)

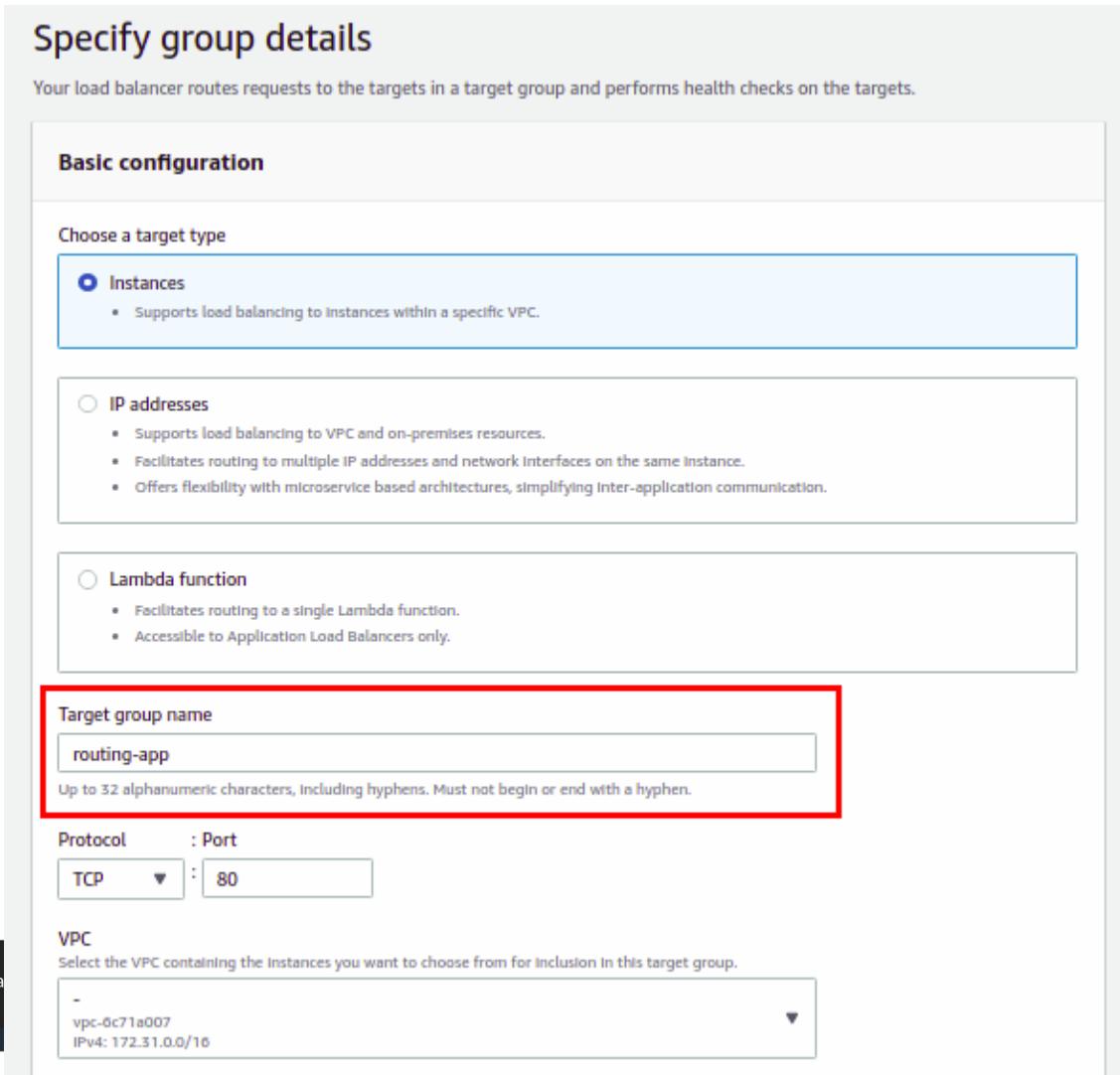
Protocol	Port
TCP	: 80 1-65535

Default action

Forward to Select a target group (2)

Add listener

Kita akan diarahkan pada tab lain. Kita isikan nama dari **Target Group** ini routing-app



Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Basic configuration

Choose a target type

- Instances
 - Supports load balancing to instances within a specific VPC.
- IP addresses
 - Supports load balancing to VPC and on-premises resources.
 - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
 - Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Lambda function
 - Facilitates routing to a single Lambda function.
 - Accessible to Application Load Balancers only.

Target group name

Up to 32 alphanumeric characters, including hyphens. Must not begin or end with a hyphen.

Protocol : Port

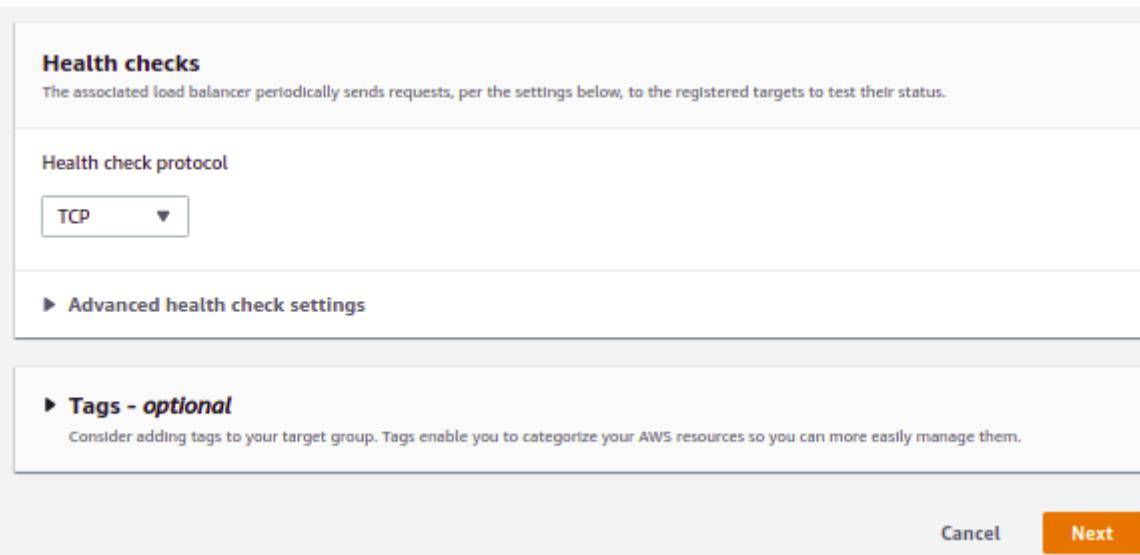
TCP	:	80
-----	---	----

VPC

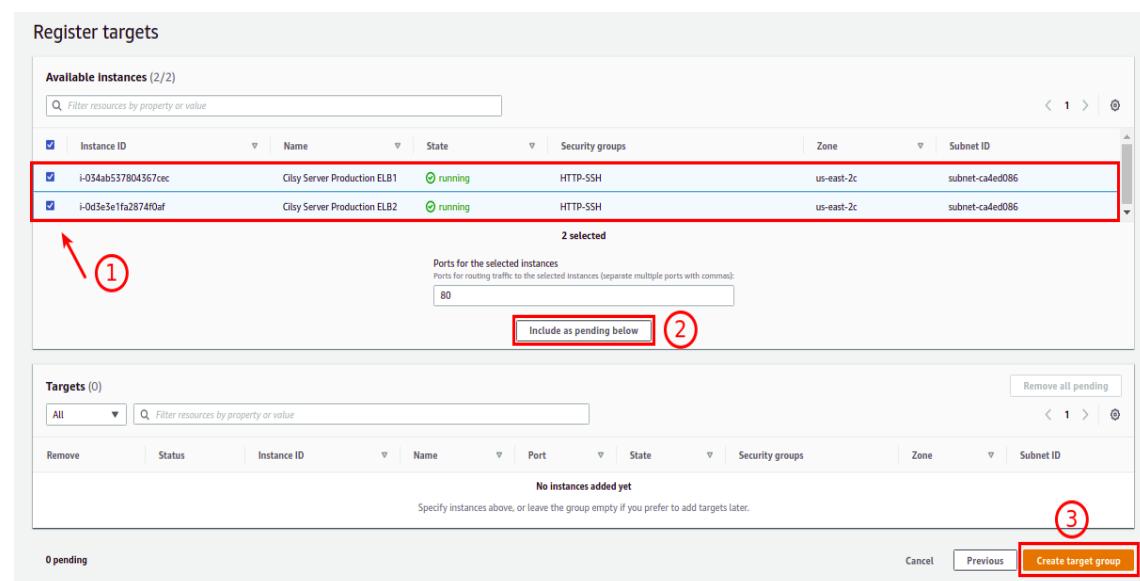
Select the VPC containing the instances you want to choose from for inclusion in this target group.

vpc-0c71a007
IPv4: 172.31.0.0/16

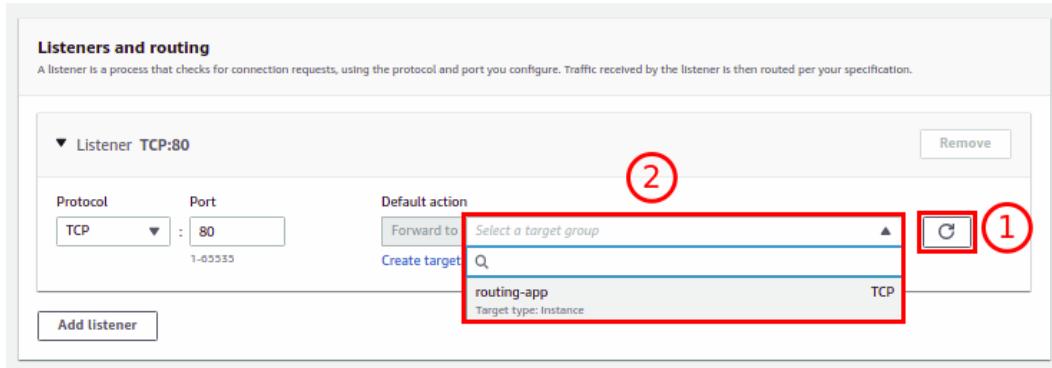
Jika discroll ke bawah, kita klik **Next**



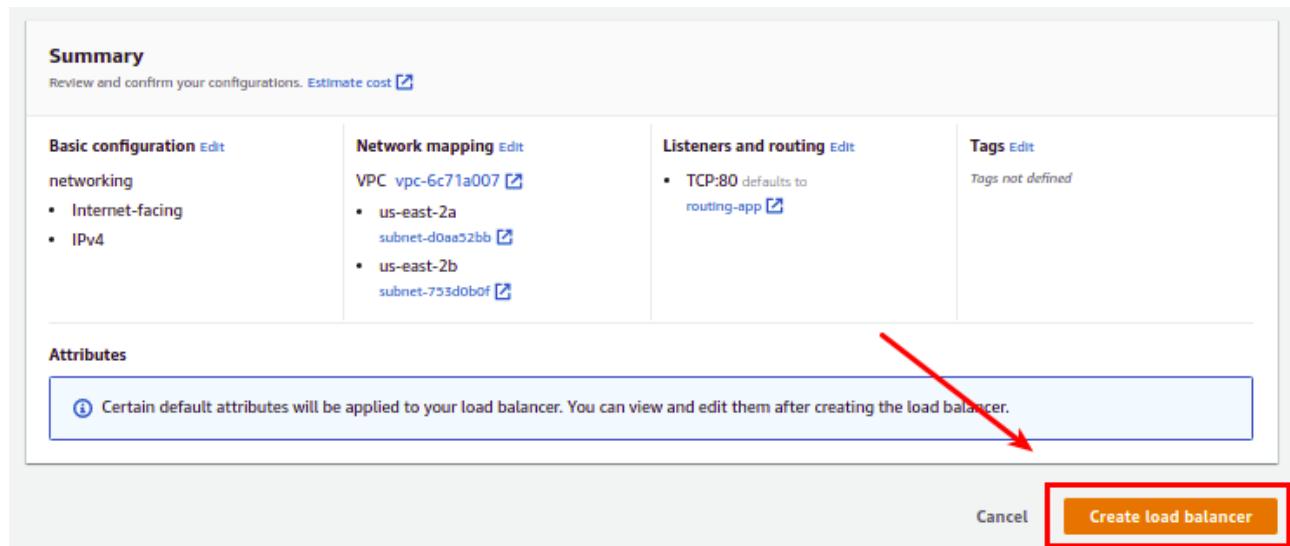
Lalu langkah selanjutnya pada menu **Register Target**, kita isikan **Instance** yang akan diberlakukan **Network Load Balancing**. Lalu klik **Include as pending below**. Setelah semua beres, klik **Target Group**.



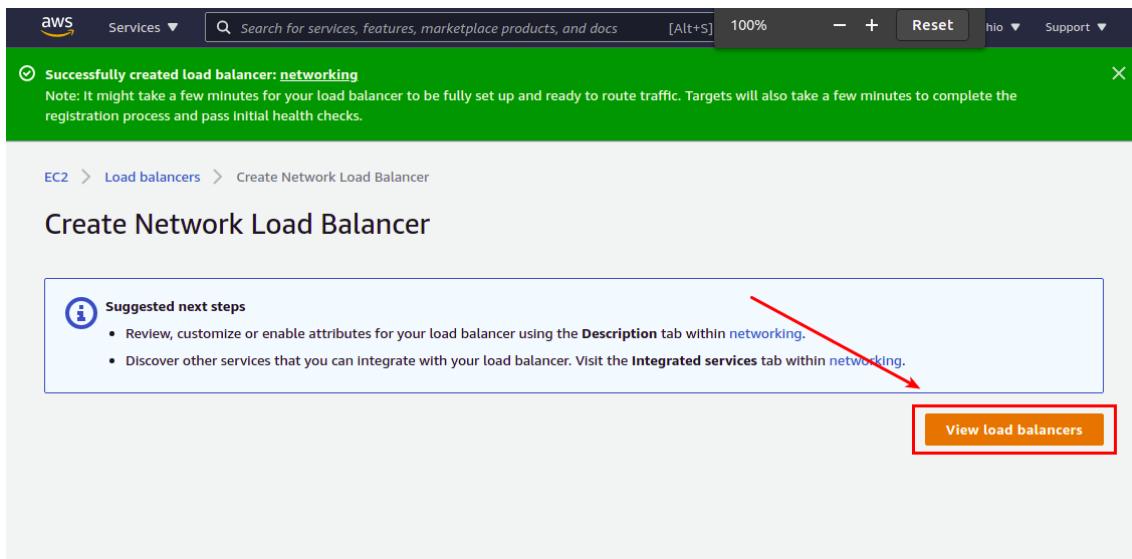
Kembali ke tab awal pembuatan **Network Load Balancing**, refresh **Default action** dengan klik ikon refresh seperti pada gambar. Lalu pilih **Target Group** yang sudah dibuat sebelumnya.



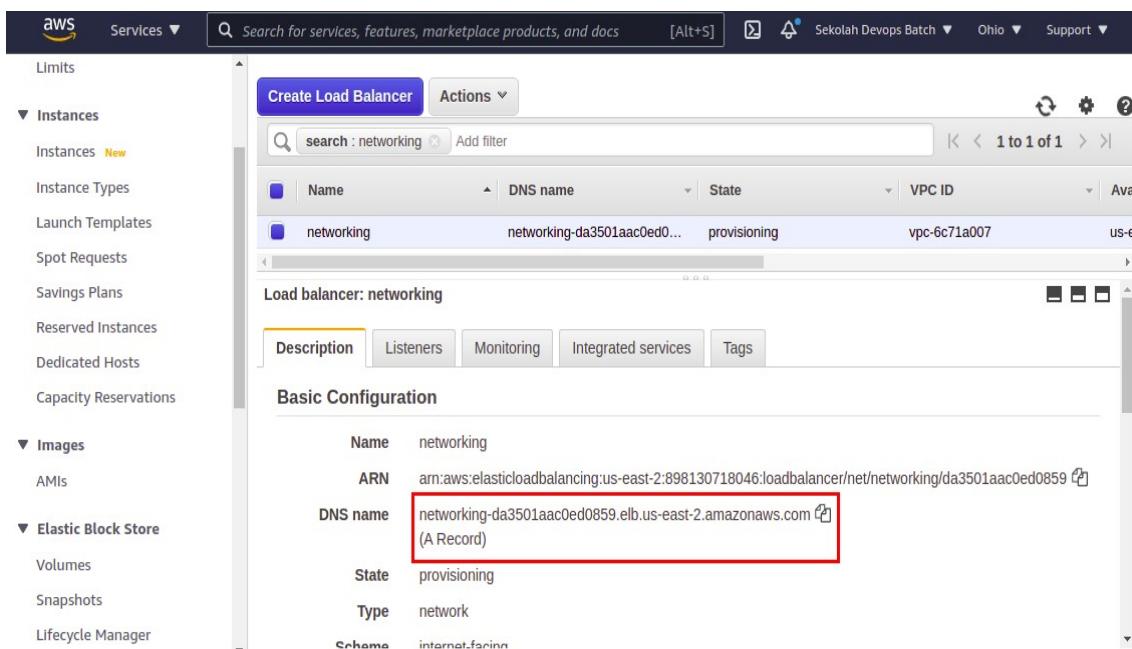
Scroll ke bawah untuk melihat review konfigurasi **Network Load Balancing** yang akan dibuat. Klik **Create load balancer**.



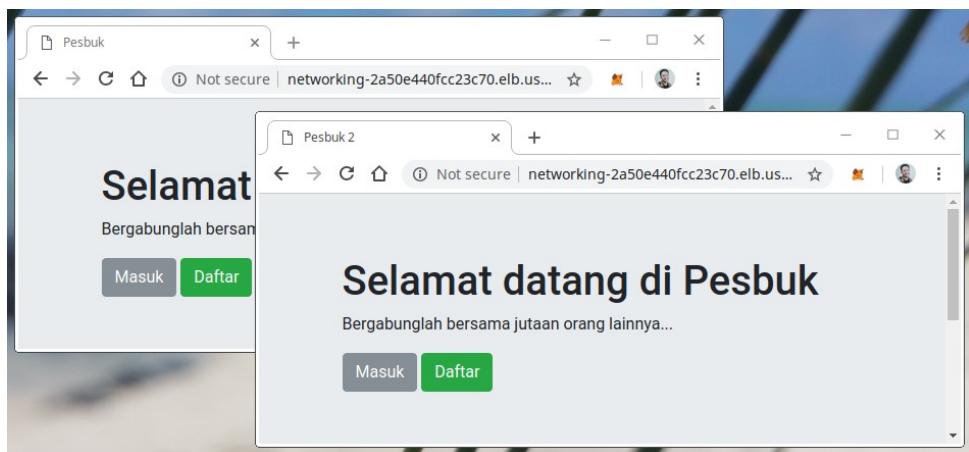
Setelah selesai, kita mendapatkan notifikasi bahwa **Load balancer** sudah dibuat. Klik **View load balancer** agar diarahkan ke dalam menu **dashboard Elastic Load Balancer** dan kita dapat lihat konfigurasi yang sudah kita buat tadi. Untuk melihat DNS name url, kita dapat menklik tombol description yang ada di menu bawah ELB.



Pada menu **dashboard Elastic Load Balancer**, kita dapat lihat konfigurasi yang sudah kita buat tadi. Untuk melihat DNS name url, kita dapat menklik tombol description yang ada di menu bawah ELB.



Coba akses alamat DNS Name tersebut. Refresh terus menerus dan lihat halaman webserver kita yang akan berubah ubah tergantung dengan server yang kita dapatkan pada saat kita akses. Kita bisa membedakan pada tittle aplikasi pada kedua server yang berbeda.



Hasil Setup Network Load Balancer

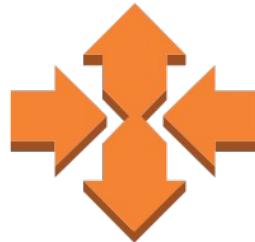
6.3.6. Exercise

1. Buat dua buah server instance yang sudah terinstall webserver dan web aplikasi.
2. Buat Load Balancer yang menurut kalian cocok untuk digunakan pada aplikasi yang kalian miliki !

6.4. Auto Scalling

6.4.1. Apa itu Auto Scaling ?

Auto Scaling merupakan salah satu layanan dari AWS yang bekerja memonitoring aplikasi kita dan secara otomatis menyesuaikan kapasitas untuk mempertahankan kinerja yang stabil. Dengan memanfaatkan informasi traffik yang masuk kedalam layanan instance kita, Auto Scaling dapat mengetahui kebutuhan yang dimiliki oleh pengguna sehingga dia dapat langsung melakukan scaling instance agar kebutuhan traffik terlayani dengan baik.



Logo AutoScalling

Auto Scaling memungkinkan layanan web kita bisa menambah jumlah instance ketika load tinggi dan menguranginya ketika load rendah secara otomatis. Dengan Auto Scaling, kita tidak perlu lagi melakukan capacity guessing.

Untuk memaksimalkan layan ini, kita harus mengkombinasikan auto scaling dengan load balancing, sehingga semua server yang melakukan scale up dan scale down akan tetap bisa kita akses menggunakan alamat yang sama dari Load Balancer tersebut.

Sebelum membuat konfigurasi Auto Scaling, kita harus melakukan konfigurasi pada **Launch Configuration**. Auto Scaling akan secara **otomatis me-launch dan men-delete instance** sesuai beban load dengan mengambil informasi dari template nya yaitu yang sudah dibuat pada launch configuration.

Pada dasarnya Launch Configuration adalah template EC2 yang akan digunakan oleh Auto Scaling, maka pembuatan Launch Configuration pun sangat mirip seperti ketika kita men-setup instance EC2.

6.4.2. Exercise

1. Cari tahu salah satu contoh pengaplikasian Load balancer pada dunia nyata !

6.5. Praktek AutoScalling

6.5.1. Persiapan Instance

Pada bagian ini kita akan coba melakukan praktek untuk membuat sebuah Auto Scalling dari Instance yang kita miliki. Kita akan membuat instance minimal yang aktif pada auto scaling ini adalah 2 dan maksimumnya adalah 6 instance, sehingga nantinya ketika CPU server nya naik maka server akan bertambah 1 persatu hingga mencapai 6 instance.

Pertama yang harus kita siapkan adalah sebuah instance yang akan kita setup untuk AutoScalling dengan rincian berikut :

- Operating System : **Ubuntu 18.04**
- Name Tag : **namapeserta AutoScalling**
- VPC : Gunakan yang sudah dibuat sebelumnya
- Instance Type : **t2.micro**
- Sisanya bisa default disesuaikan

Setelah instance selesai dibuat, selanjutnya kita akan lakukan setup webserver dan webaplikasi pada server instance tersebut. Kalian dapat menggunakan Script yang sudah dibuat sebelumnya untuk menginstall webserver dengan requirement tersebut :

- Apache 2
- Php dan php-mysql
- mysql-server

Setup web aplikasi, jangan lupa untuk membuat user database mysql dan import database dari web aplikasi tersebut.

6.5.2. Setup AutoDeployment

Selanjutnya kita akan melakukan setup auto deployment pada instane yang sudah kita buat, karena nanti instance yang sudah kita buat sebelumnya akan dibuat menjadi sebuah image yang akan di load ketika server menggandakan diri. Maka konten yang ada pada server baru yang digandakan tersebut haruslah sama dengan konten terbaru yang kita miliki.

Sehingga kita buat sebuah automasi dimana ketika server hidup atau digandakan, dia akan otomatis melakukan deploy pada server dengan mengambil source dari github.

Pertama buat script berikut dengan nama **autodeploy.sh** di home server tersebut.

```
#!/bin/bash

echo "=====>"
echo "Downloading Data"
echo "=====>"
cd
rm -f master.zip
rm -R sosial-media-master
wget https://github.com/sdcilsy/sosial-media/archive/master.zip
echo "=====>"
echo "Ekstrak File"
echo "=====>"
sudo apt-get install -y unzip
unzip master.zip
```

```
echo "=====>"  
echo "Memindahkan data"  
echo "=====>"  
sudo rm -R /var/www/html/*  
sudo rm /var/www/html/*  
sudo mv sosial-media-master/* /var/www/html/  
echo "Setup selesai"  
exit 0
```

Bagian hijau tersebut bisa diganti dengan repository yang kalian miliki sendiri. Setelah itu simpan dan beri hak akses pada file tersebut.

```
chmod +x autodeploy.sh
```

Selanjutnya kita akan buat script tersebut berjalan pada saat booting, pada ubuntu 18.04 ini kita akan menggunakan cronjob untuk konfigurasinya seperti berikut.

```
crontab -e
```

Akan muncul pilihan seperti dibawah ini, pilih editor sesuai yang kalian pahami. Tapi saya sarankan menggunakan nano.

```
ubuntu@ip-172-31-44-30:~$ crontab -e  
no crontab for ubuntu - using an empty one  
  
Select an editor. To change later, run 'select-editor'.  
1. /bin/nano      <---- easiest  
2. /usr/bin/vim.basic  
3. /usr/bin/vim.tiny  
4. /bin/ed  
  
Choose 1-4 [1]: 1
```

Setelah masuk kedalam editor cronjob, kita pindah ke line paling akhir dan tambahkan script berikut dibawahnya.

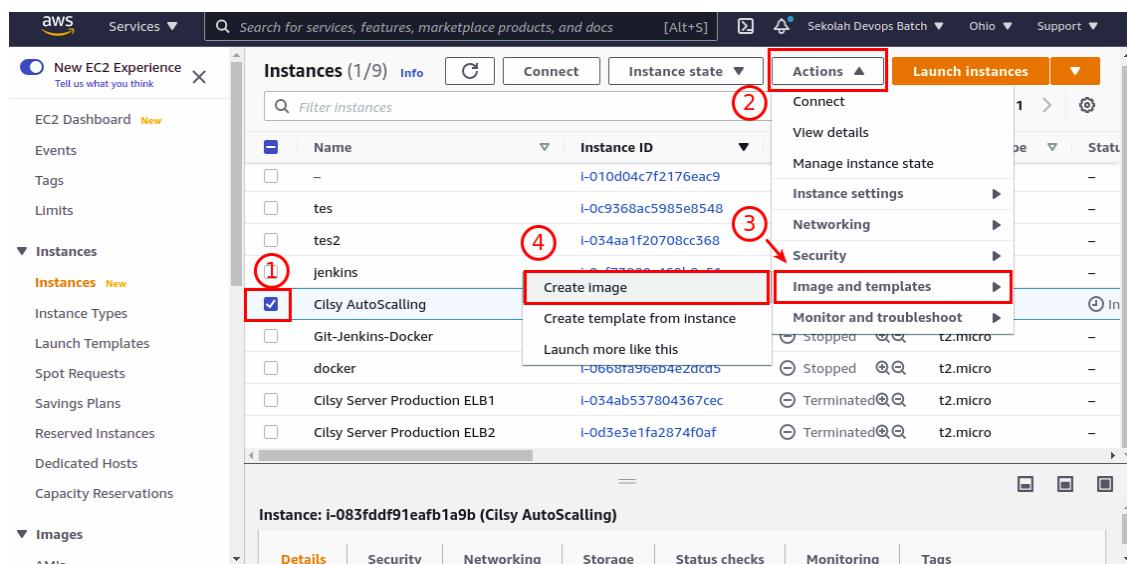
```
@reboot sh /home/ubuntu/autodeploy.sh
```

Setelah itu simpan konfigurasi tersebut, dan kita bisa ujicoba keberhasilannya dengan melakukan reboot pada instance tersebut.

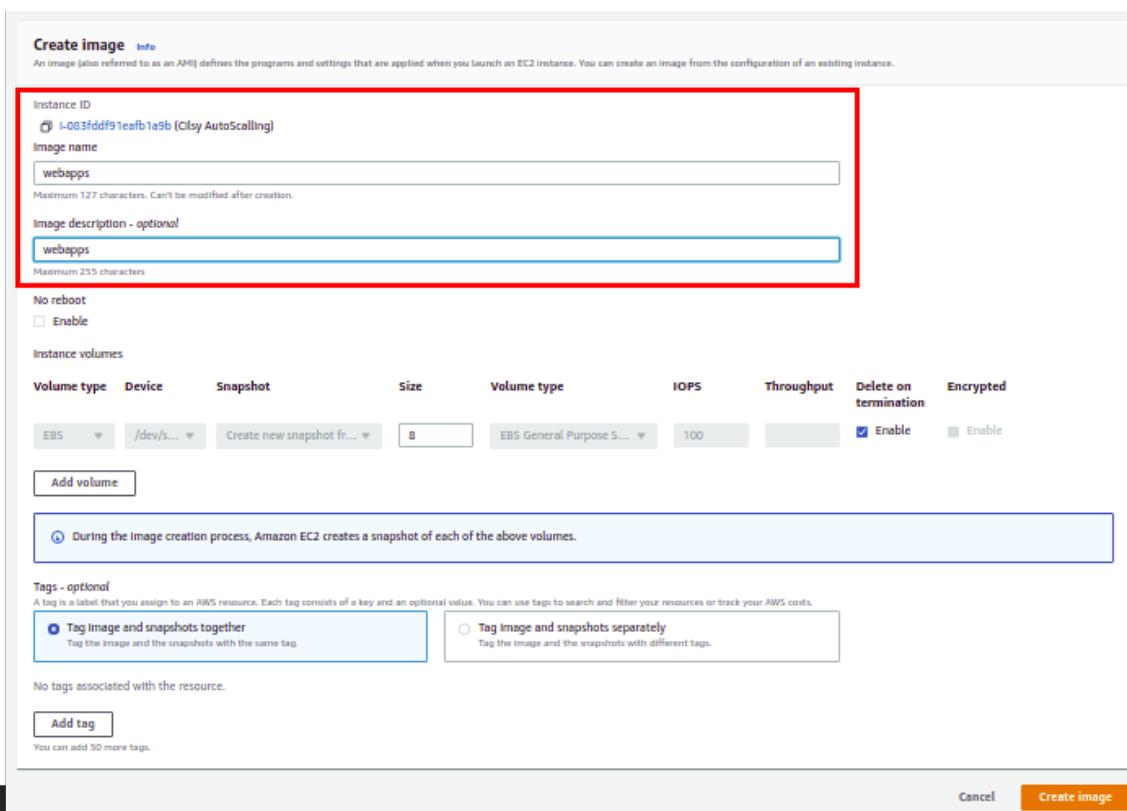


6.5.3. Setup AMI

Pada bagian ini kita akan coba membuat image dari instance tadi, image ini disebut dengan AMI. Untuk membuat AMI, kita hanya perlu memilih instance lalu klik **Action > Image and templates > Create Image**.



Setelah itu akan masuk ke menu Create image, isikan deskripsi AMI yang akan kita buat dan setelah itu klik **Create Image**.



Create image Info

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID:

Image name:

Maximum 127 characters. Can't be modified after creation.

Image description - optional:

Maximum 255 characters.

No reboot

Enable

Instance volumes:

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/s...	Create new snapshot fr...	8	EBS General Purpose S...	100		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

Add volume

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional:

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Tag Image and snapshots together
Tag the Image and the snapshots with the same tag.

Tag Image and snapshots separately
Tag the Image and the snapshots with different tags.

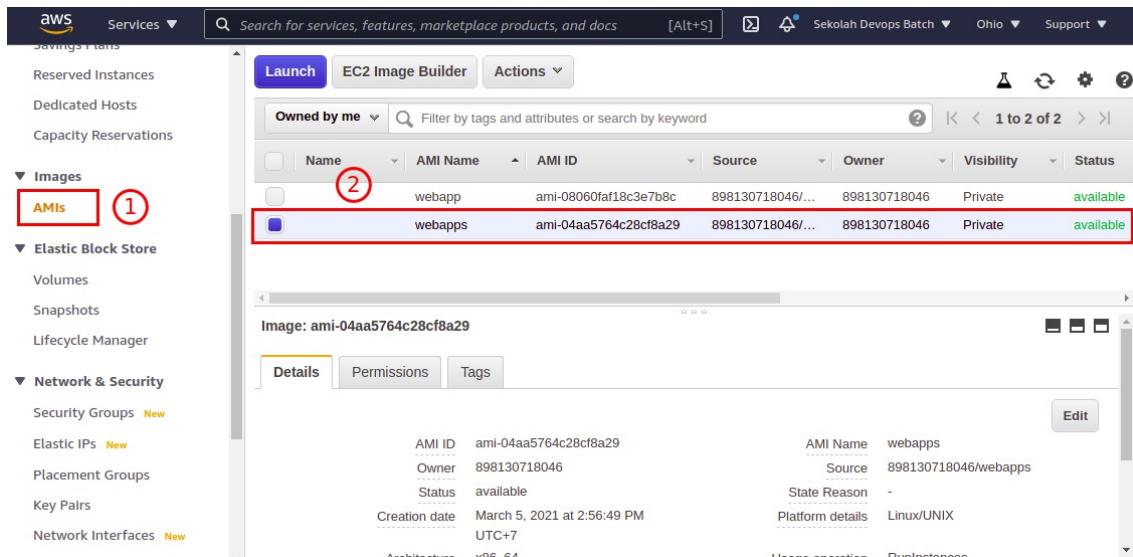
No tags associated with the resource.

Add tag

You can add 50 more tags.

Create image

Untuk mengecek AMI yang sudah kita buat, kita bisa masuk kedalam menu AMI yang ada pada **dashboard EC2** seperti berikut. Tunggu proses pending pada AMI hingga selesai menjadi **available**.

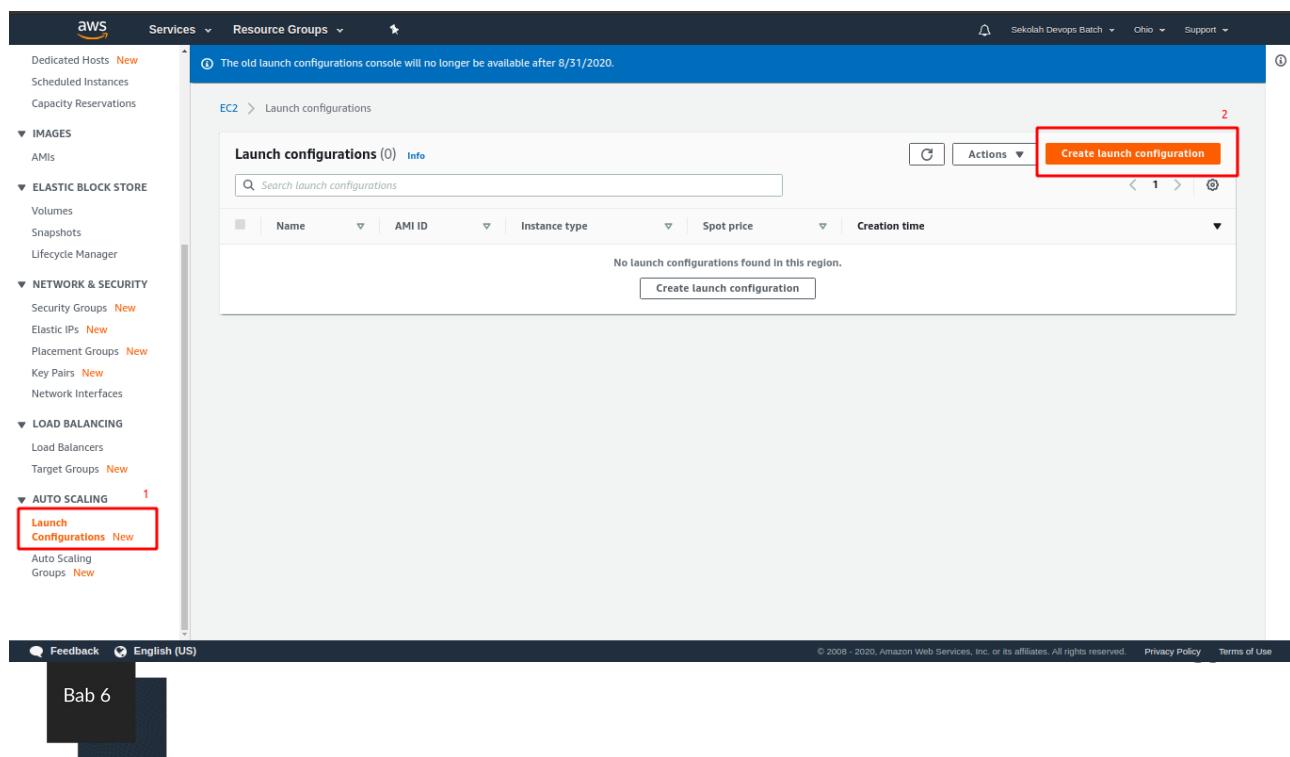


Name	AMI Name	AMI ID	Source	Owner	Visibility	Status
webapp	ami-08060faf18c3e7b8c	898130718046...	898130718046	Private		available
webapps	ami-04aa5764c28cf8a29	898130718046...	898130718046	Private		available

Instance yang sudah menjadi AMI

6.5.4. Setup Launch Configuration

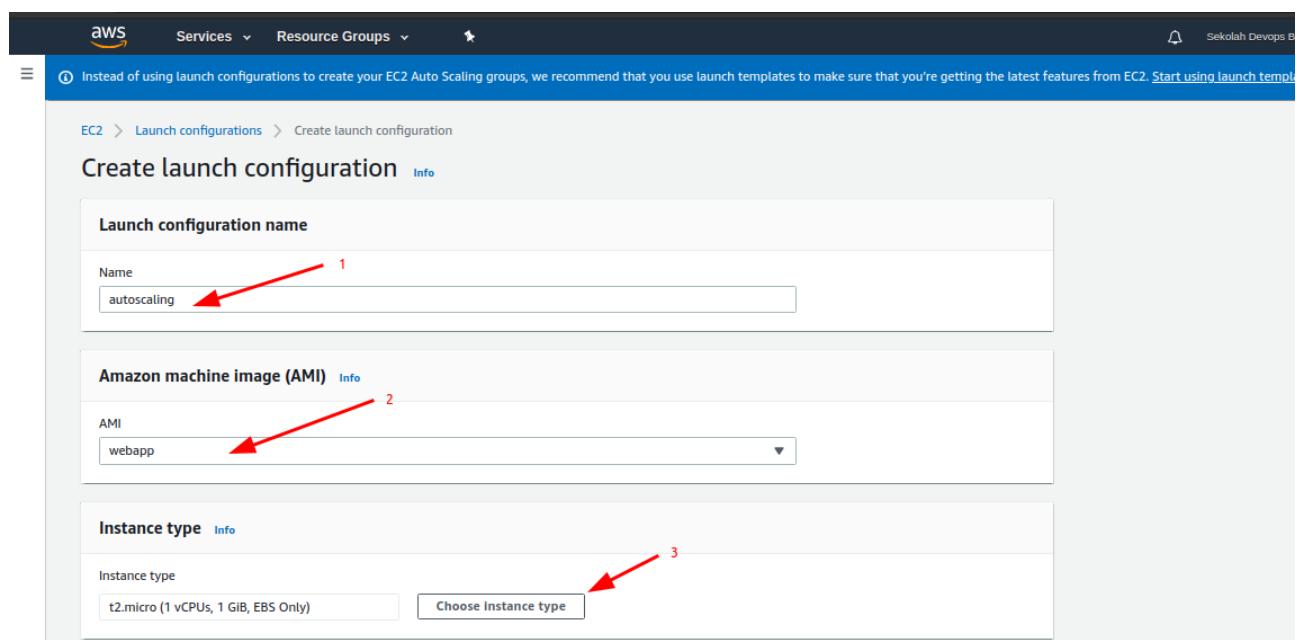
Sekarang kita akan masuk kedalam setup autoscaling tahap awal, disini kita akan melakukan konfigurasi pada **launch configuration** dan mendaftarkan IAM yang sudah kita buat tadi. Pertama kit masuk kedalam menu **Launch Configuration** lalu pilih tombol **Create launch configuration**.



Selanjutnya kita akan diarahkan pada launch konfiguration, pilih AMI yang sudah kita sediakan sebelumnya.

Setelah itu pilih instance type yang akan kita gunakan, disini saya memilih **t2.micro**.

Selanjutnya isikan detail konfigurasi launch instance yang akan kita gunakan, jangan lupa check monitoring cloudwatch agar kita bisa melihat traffick yang keluar dari server tersebut.



Create launch configuration

Launch configuration name

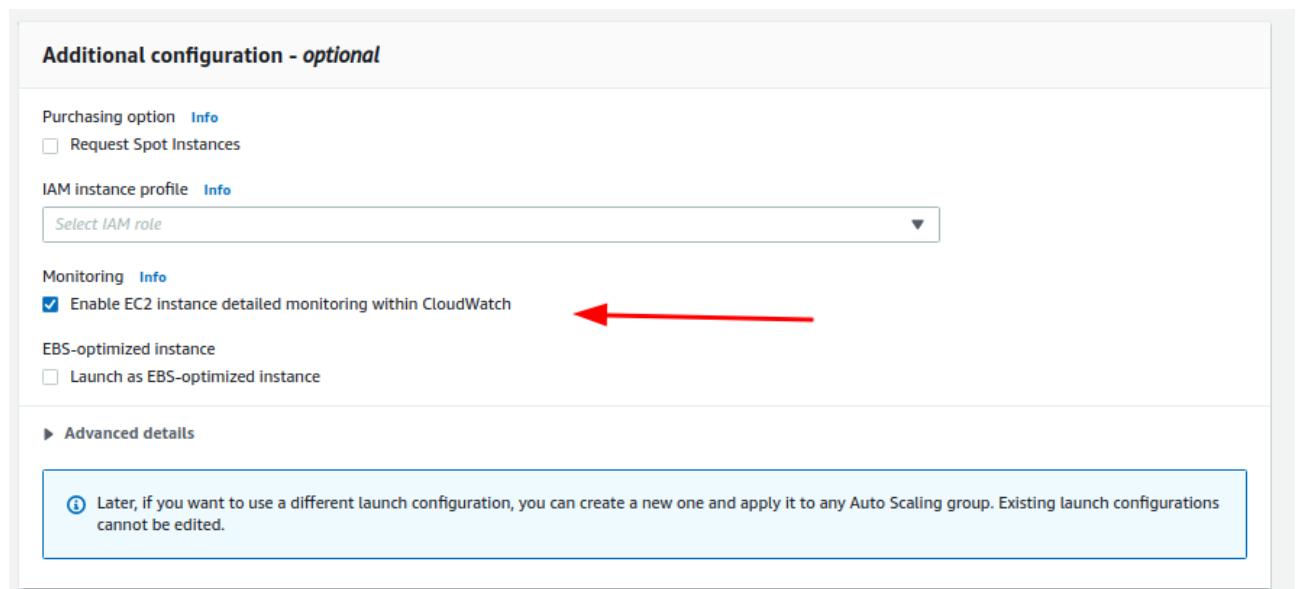
Name: autoscaling 1

Amazon machine image (AMI)

AMI: webapp 2

Instance type

Instance type: t2.micro (1 vCPUs, 1 GiB, EBS Only) 3



Additional configuration - optional

Purchasing option [Info](#)
 Request Spot Instances

IAM instance profile [Info](#)
 Select IAM role

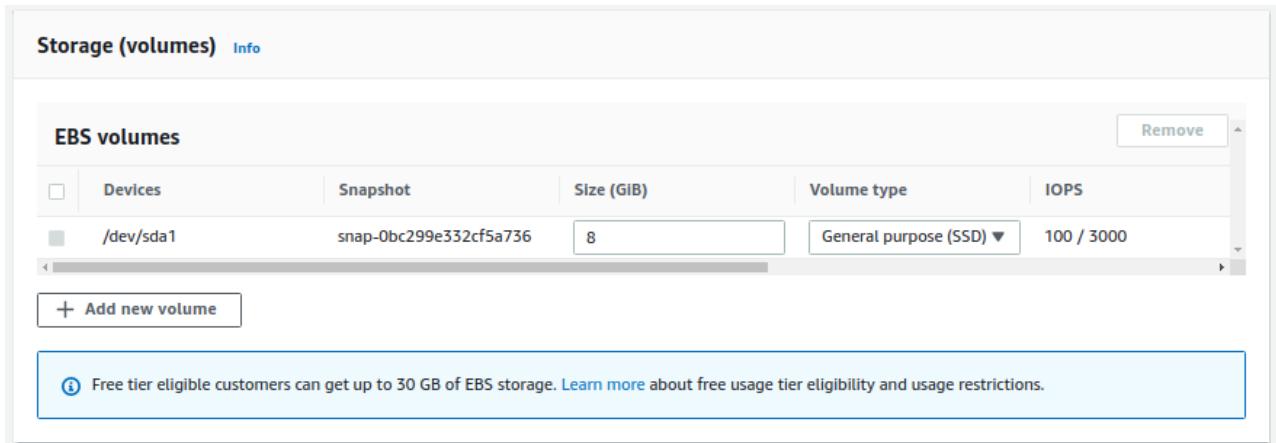
Monitoring [Info](#)
 Enable EC2 instance detailed monitoring within CloudWatch →

EBS-optimized instance
 Launch as EBS-optimized instance

► Advanced details

(i) Later, if you want to use a different launch configuration, you can create a new one and apply it to any Auto Scaling group. Existing launch configurations cannot be edited.

Selanjutnya sesuaikan storage yang akan digunakan pada server yang nanti akan kita gunakan.



Storage (volumes) [Info](#)

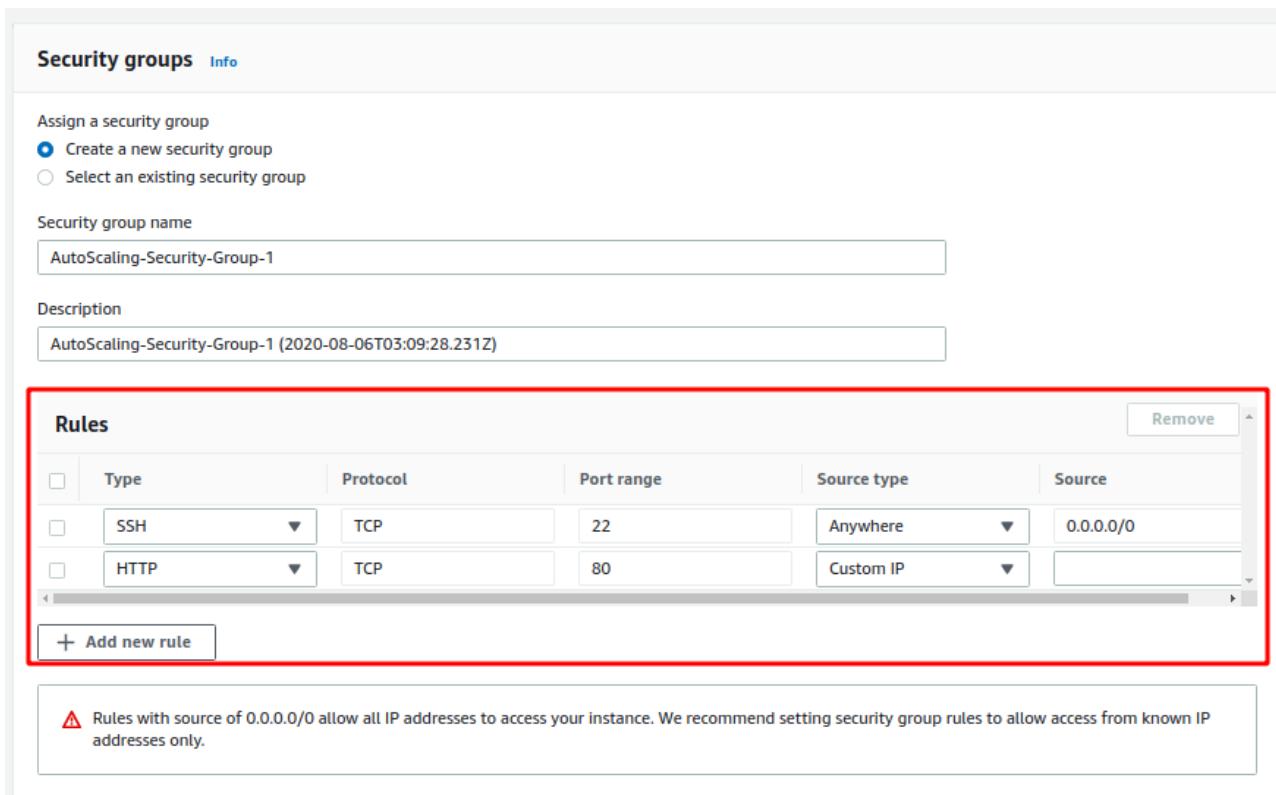
EBS volumes

Devices	Snapshot	Size (GiB)	Volume type	IOPS
/dev/sda1	snap-0bc299e332cf5a736	8	General purpose (SSD)	100 / 3000

+ Add new volume

Free tier eligible customers can get up to 30 GB of EBS storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Selanjutnya, buat **Security Group** baru dengan spesifikasi seperti dibawah ini, kalian juga dapat menggunakan security group yang sudah ada sebelumnya.



Security groups [Info](#)

Assign a security group

Create a new security group

Select an existing security group

Security group name

AutoScaling-Security-Group-1

Description

AutoScaling-Security-Group-1 (2020-08-06T03:09:28.231Z)

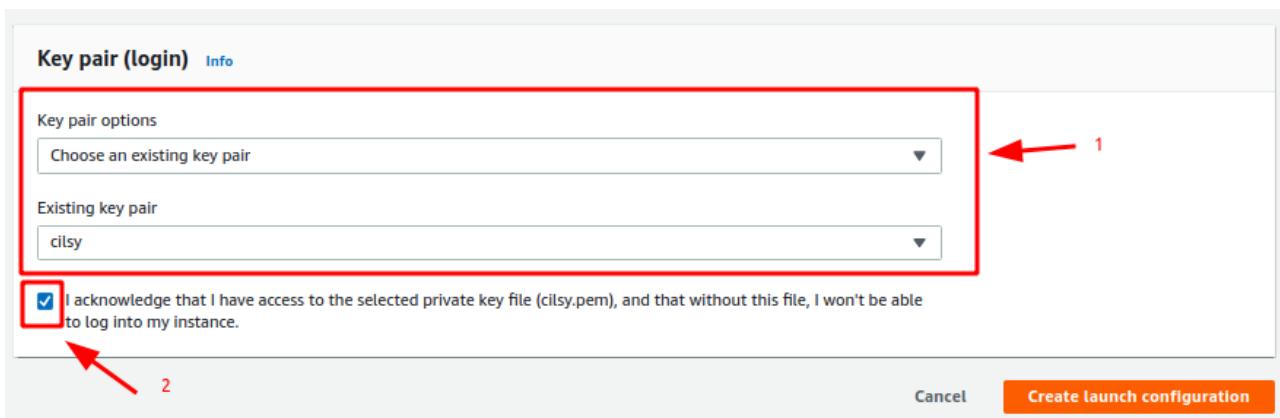
Rules

Type	Protocol	Port range	Source type	Source
SSH	TCP	22	Anywhere	0.0.0.0/0
HTTP	TCP	80	Custom IP	

+ Add new rule

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Kemudian kita lanjut ke bagian untuk memilih keypair. Gunakan yang sudah dibuat sebelumnya, disini kita memilih **cilsy** untuk keypairnya.



Key pair (login) Info

Key pair options

Choose an existing key pair ▾

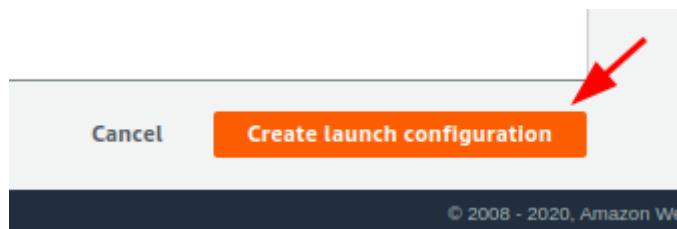
Existing key pair

cilsy ▾

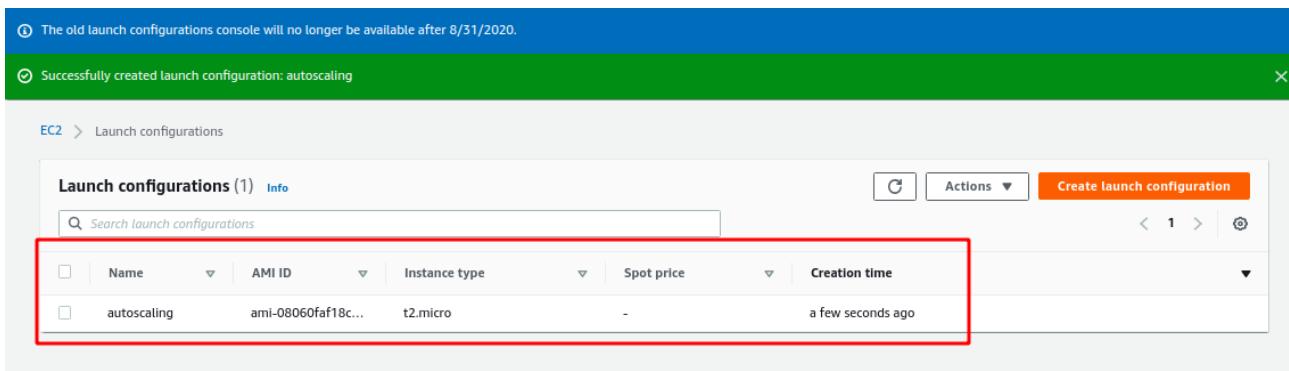
I acknowledge that I have access to the selected private key file (cilsy.pem), and that without this file, I won't be able to log into my instance.

Cancel **Create launch configuration**

Selanjutnya, klik Create launch configuration



Maka setelah semua tahap konfigurasi dilakukan, kita telah berhasil membuat sebuah launch configuration yang akan kita gunakan pada autoscaling group nanti, hasilnya seperti berikut.



The old launch configurations console will no longer be available after 8/31/2020.

Successfully created launch configuration: autoscaling

EC2 > Launch configurations

Launch configurations (1) Info

Actions ▾ **Create launch configuration**

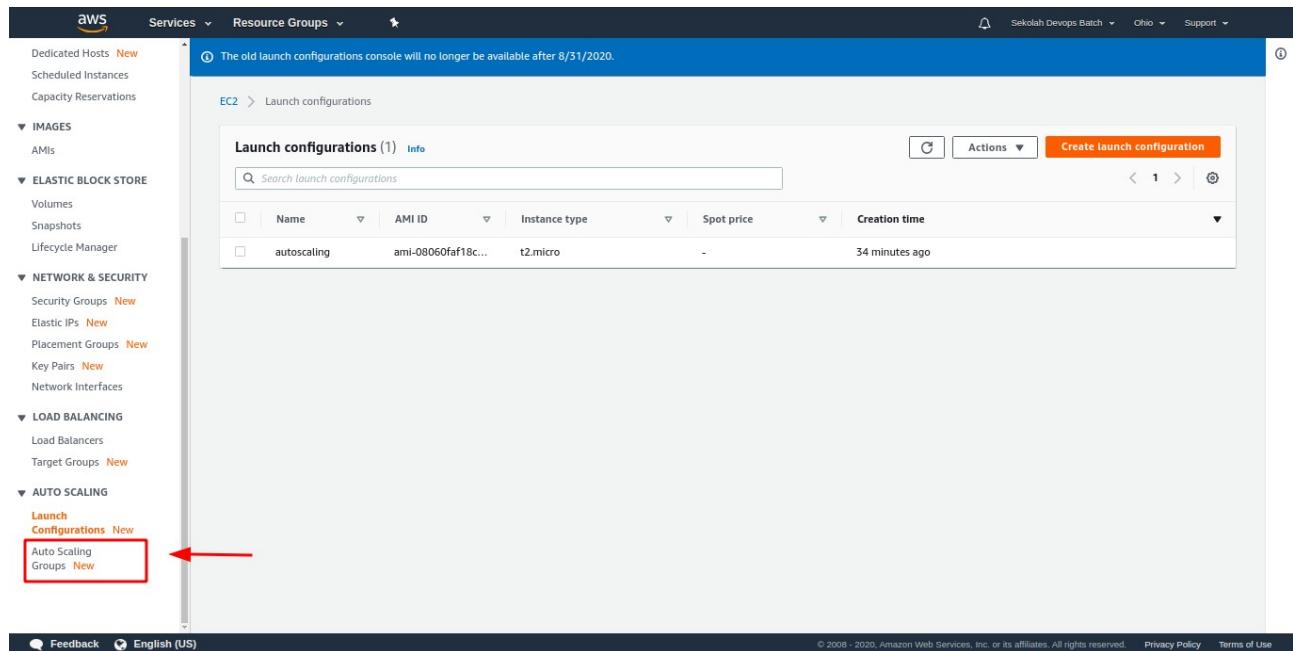
Name	AMI ID	Instance type	Spot price	Creation time
autoscaling	ami-08060faf18c...	t2.micro	-	a few seconds ago

Hasil konfigurasi launch configuration

6.5.5. Konfigurasi Auto Scaling Group

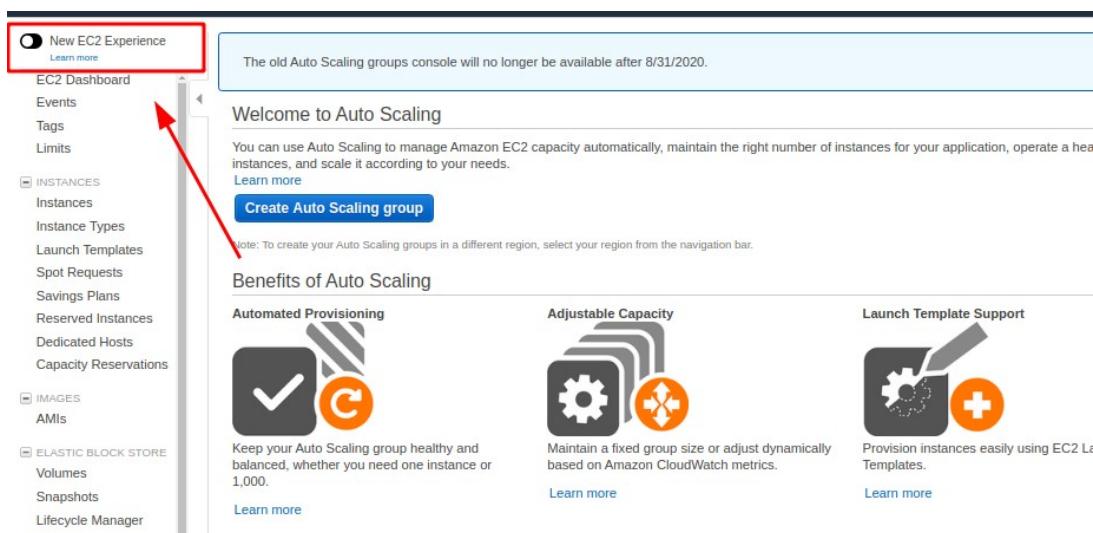
Setelah kita melakukan persiapan dengan membuat **Launch Configurations**, sekarang kita akan masuk kedalam konfigurasi Auto Saling Group. Untuk mulai

membuatnya, kita harus masuk kedalam menu Auto Scaling Groups lalu tekan tombol **Create Auto Scaling Group**.



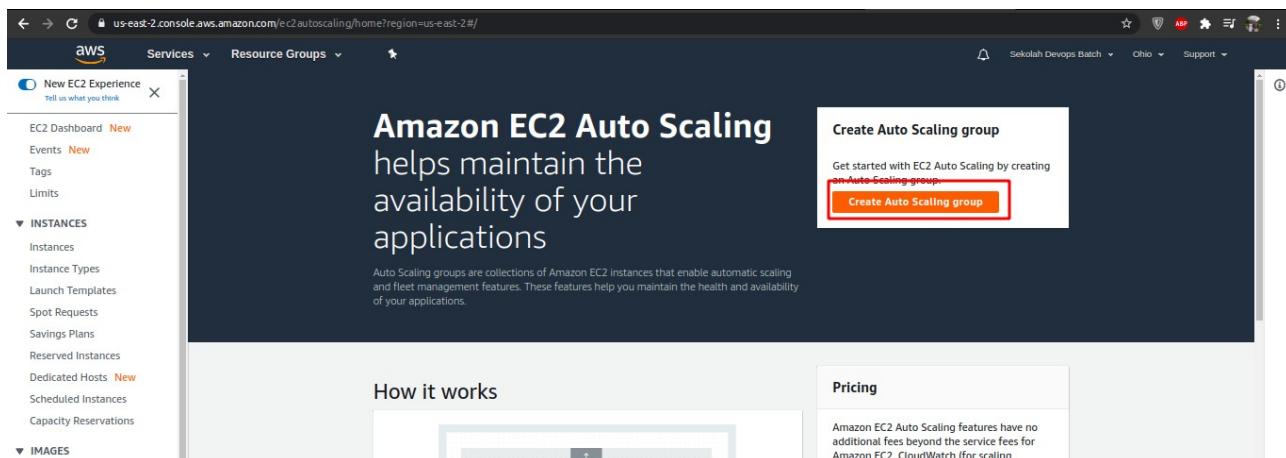
The screenshot shows the AWS EC2 Launch Configurations page. On the left, there is a navigation sidebar with several sections: Dedicated Hosts, Scheduled Instances, Capacity Reservations, IMAGES (AMIs), ELASTIC BLOCK STORE (Volumes, Snapshots, Lifecycle Manager), NETWORK & SECURITY (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), LOAD BALANCING (Load Balancers, Target Groups), and AUTO SCALING (Launch Configurations, Auto Scaling Groups). A red arrow points from the text in the previous paragraph to the 'Auto Scaling Groups' link in the sidebar.

Pada modul ini, akan dijelaskan 2 cara membuat Auto Scaling Group, mengingat ada 2 jenis versi UI yang digunakan pada AWS Console. Secara **default**, Anda akan menggunakan **tampilan baru**. Namun, apabila Anda ingin menggunakan tampilan lama, Anda harus menon-aktifkan terlebih dahulu tampilan baru dari AWS Console di pojok kiri atas.

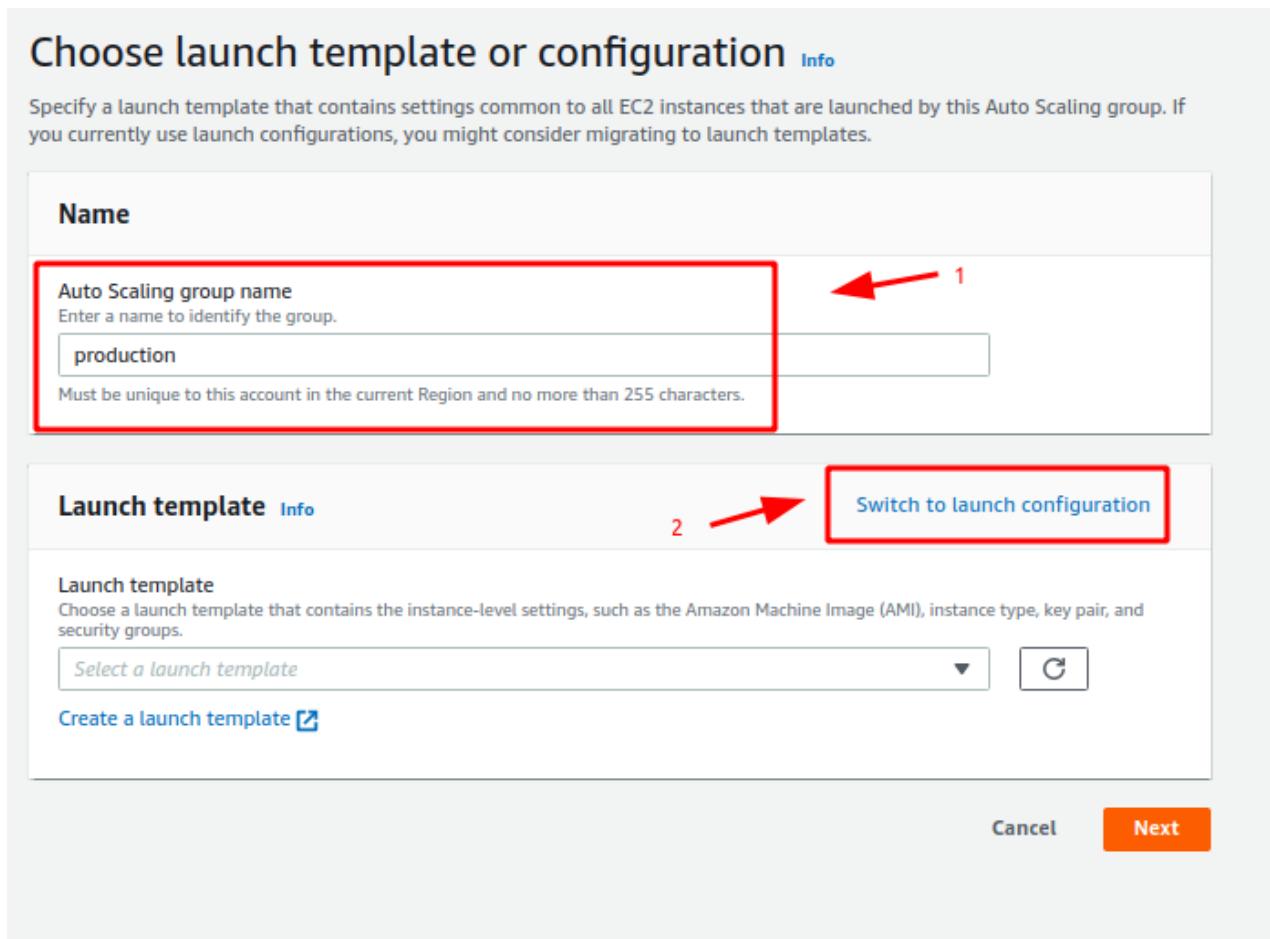


The screenshot shows the AWS Auto Scaling Groups page. At the top left, there is a 'New EC2 Experience' button with a red box and an arrow pointing to it. The main content area includes a message about the old Auto Scaling groups console being deprecated, followed by a 'Welcome to Auto Scaling' section, a 'Create Auto Scaling group' button, and a note about selecting a region. Below this are three 'Benefits of Auto Scaling' cards: 'Automated Provisioning' (keep the group healthy and balanced), 'Adjustable Capacity' (maintain a fixed group size or adjust dynamically), and 'Launch Template Support' (provision instances easily using EC2 Templates).

Berikut adalah langkah membuat Auto Scaling Group menggunakan **UI versi baru.**



Pada awal menu kita akan diminta untuk memilih launch configuration yang kita miliki, disini pilih yang sudah kita buat sebelumnya.



Choose launch template or configuration Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name
Auto Scaling group name Enter a name to identify the group. production
Must be unique to this account in the current Region and no more than 255 characters.

Launch template <small>Info</small>	Switch to launch configuration
Launch template Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups. Select a launch template	2
Create a launch template <input checked="" type="checkbox"/>	

Cancel

Next

Choose launch template or configuration Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name

Auto Scaling group name
Enter a name to identify the group.
production
Must be unique to this account in the current Region and no more than 255 characters.

Launch configuration Info [Switch to launch template](#)

Launch configuration
Choose a launch configuration that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

autoscaling	▼													
Create a launch configuration <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <tr> <td style="width: 30%; padding: 5px;">Launch configuration</td> <td style="width: 30%; padding: 5px;">AMI ID</td> <td style="width: 40%; padding: 5px;">Date created</td> </tr> <tr> <td>autoscaling</td> <td>ami-08060faf18c3e7b8c</td> <td>Thu Aug 06 2020 10:28:24 GMT+0700 (Western Indonesia Time)</td> </tr> <tr> <td>Security groups</td> <td>Instance type</td> <td>Key pair name</td> </tr> <tr> <td>sg-0f94535fdaf7f8507</td> <td>t2.micro</td> <td>-</td> </tr> </table>			Launch configuration	AMI ID	Date created	autoscaling	ami-08060faf18c3e7b8c	Thu Aug 06 2020 10:28:24 GMT+0700 (Western Indonesia Time)	Security groups	Instance type	Key pair name	sg-0f94535fdaf7f8507	t2.micro	-
Launch configuration	AMI ID	Date created												
autoscaling	ami-08060faf18c3e7b8c	Thu Aug 06 2020 10:28:24 GMT+0700 (Western Indonesia Time)												
Security groups	Instance type	Key pair name												
sg-0f94535fdaf7f8507	t2.micro	-												

[Cancel](#) [Next](#)

Setelah itu, pilih VPC yang akan digunakan, dan juga subnetnya.

Configure settings Info

Configure the settings below. Depending on whether you chose a launch template, these settings may include options to help you make optimal use of EC2 resources.

Network Info

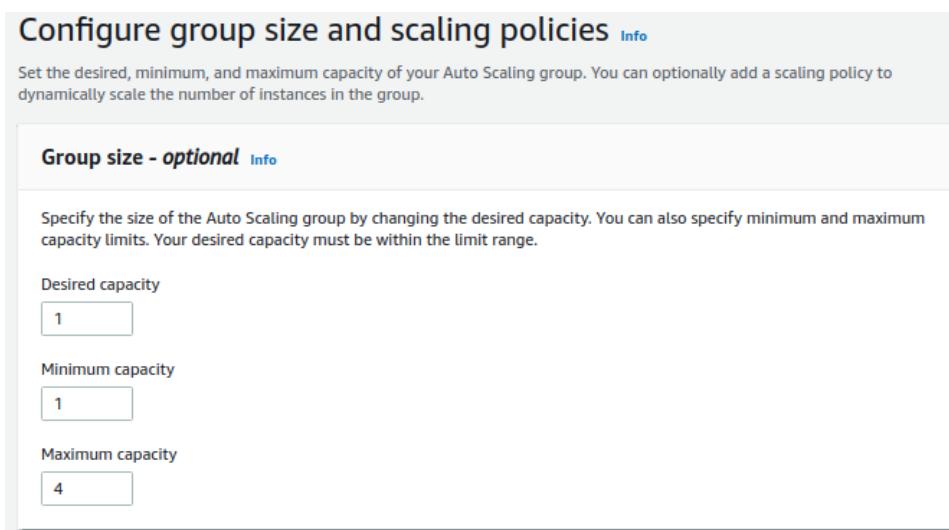
For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC 1
 Select: **vpc-6c71a007** 172.31.0.0/16 Default C
[Create a VPC](#)

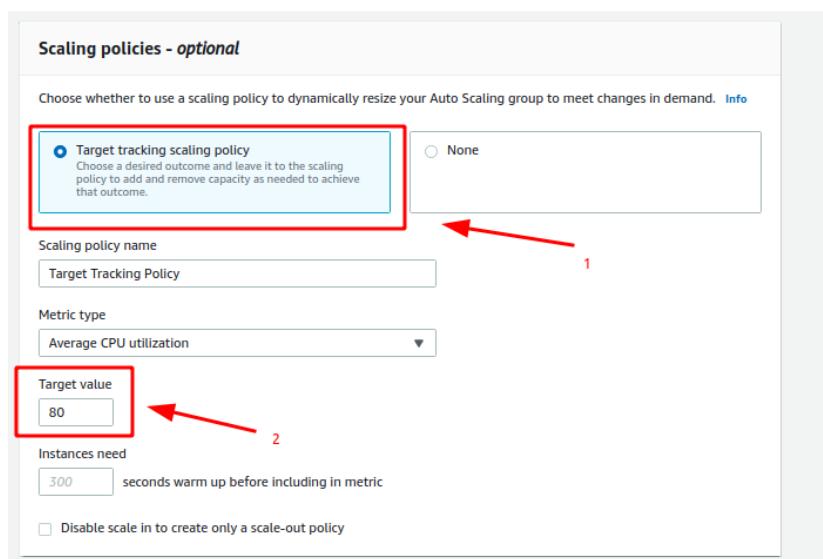
Subnets 2
 Select subnets: Select subnets C
 Subnets:
 us-east-2a | subnet-d0aa52bb X 172.31.0.0/20 Default
 us-east-2b | subnet-753d0b0f X 172.31.16.0/20 Default
 us-east-2c | subnet-ca4ed086 X 172.31.32.0/20 Default
[Create a subnet](#)

[Cancel](#) Previous Skip to review [Next](#)

Bagian selanjutnya pilih **Use Scaling Policies**. Setelah itu sesuaikan bagian **Group Size** untuk maximum dan minimum instance, disini saya menentukan **minimal 1 dan maximal 4** instance. Untuk bagian **Desired Capacity**, kita akan isi dengan 1. Sehingga apabila akan dilakukan scale up, jumlah instance yang akan ditambahkan adalah 1.

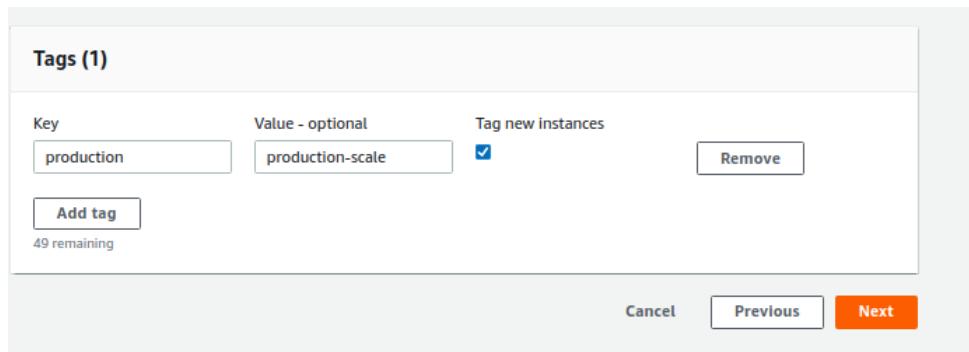


Selanjutnya kita set parameter **Scaling policies**, dimana nantinya ini akan jadi parameter ketika instance tersebut harus bertambah apabila CPU mengalami kenaikan. Pilih **Target tracking scaling policy**. Kemudian isikan informasi seperti dibawah ini, lalu isikan target value CPU menjadi **80**, setelah selesai klik **Next**.



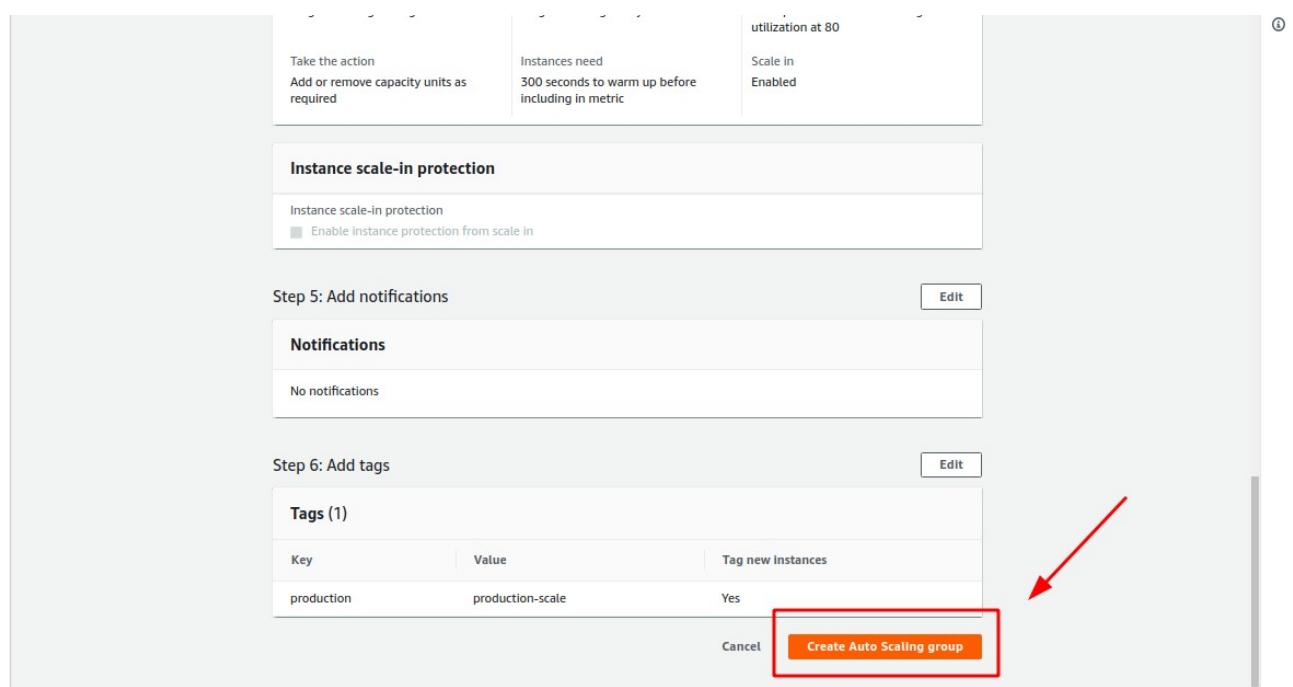
Selanjutnya adalah bagian notifikasi, kalian dapat mensetting notifikasi untuk scaling ini. Akan tetapi pada bagian ini kita tidak akan melakukannya. Maka selanjutnya pilih **Next**.

Selanjutnya isikan **tag** pada instance yang akan di scale oleh auto scaling, disini kita akan berikan tag berupa nama, sehingga ketika ada penambahan instance baru maka akan memiliki nama sesuai tag yang kita buat. Setelah itu klik **Next**.

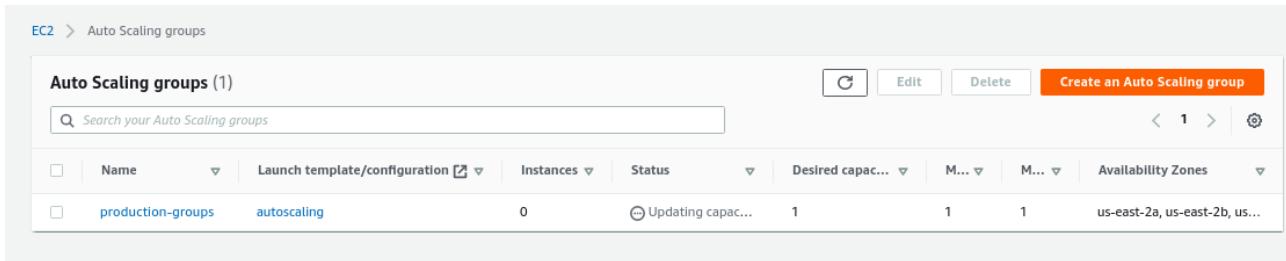


Berikutnya akan muncul bagian review, bagian ini hanya menampilkan sekilas review dari konfigurasi yang sudah kita buat tadi. Klik **Create Auto Scaling group** untuk melanjutkan.

Jika berhasil maka akan muncul status berhasilnya seperti dibawah ini, klik **Close** untuk melanjutkan.



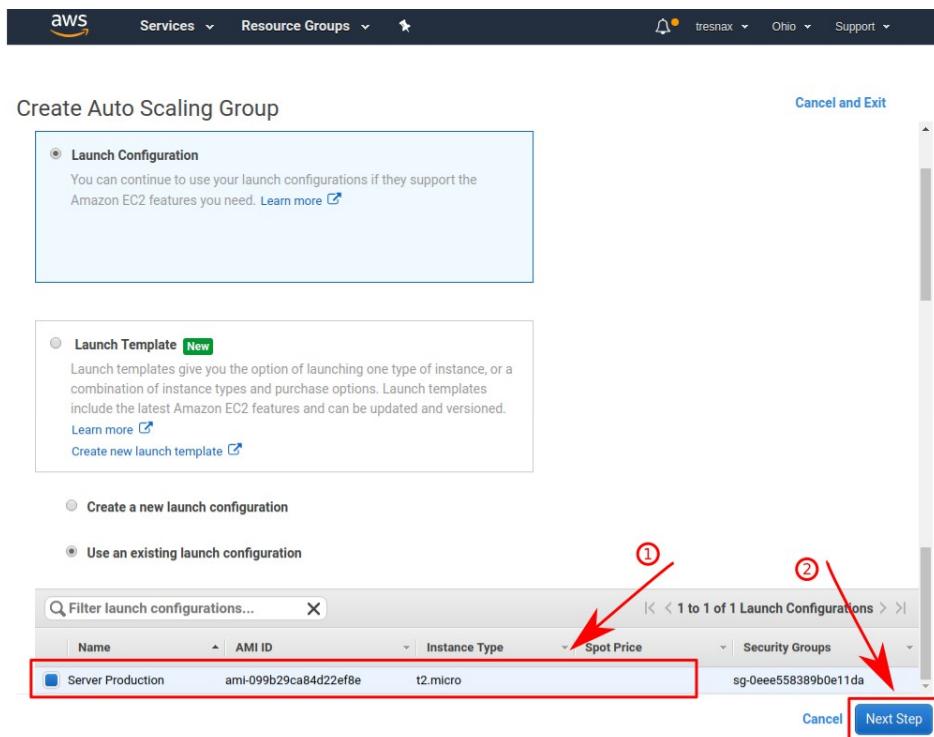
Maka hasil dari konfigurasi Auto Scaling Grup yang sudah dibuat tadi akan terlihat seperti dibawah ini.



The screenshot shows the AWS EC2 Auto Scaling groups interface. At the top, there's a search bar labeled "Search your Auto Scaling groups". Below it is a table with columns: Name, Launch template/configuration, Instances, Status, Desired capacity, Min., Max., and Availability Zones. One row is visible for the group "production-groups" which uses the launch template "autoscaling", has 0 instances, and is in the status "Updating capacity...".

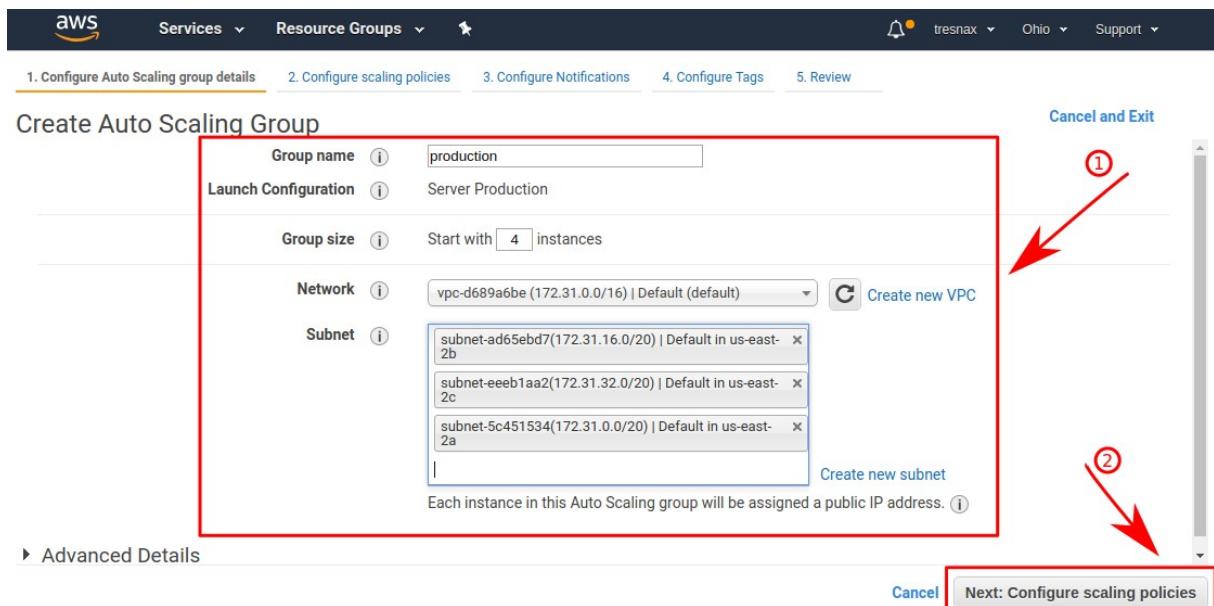
Seperti yang telah dijelaskan di awal sub bab ini bahwa ada 2 interface yang bisa digunakan, yaitu versi lama dan versi baru. Berikut kami tampilkan konfigurasi pada UI versi lama, karena ada beberapa perbedaan dalam step-stepnya, namun secara umum masih sama.

Pada awal menu kita akan diminta untuk memilih launch configuration yang kita miliki, disini pilih yang sudah kita buat sebelumnya tadi. Selanjutnya **Next Step.**

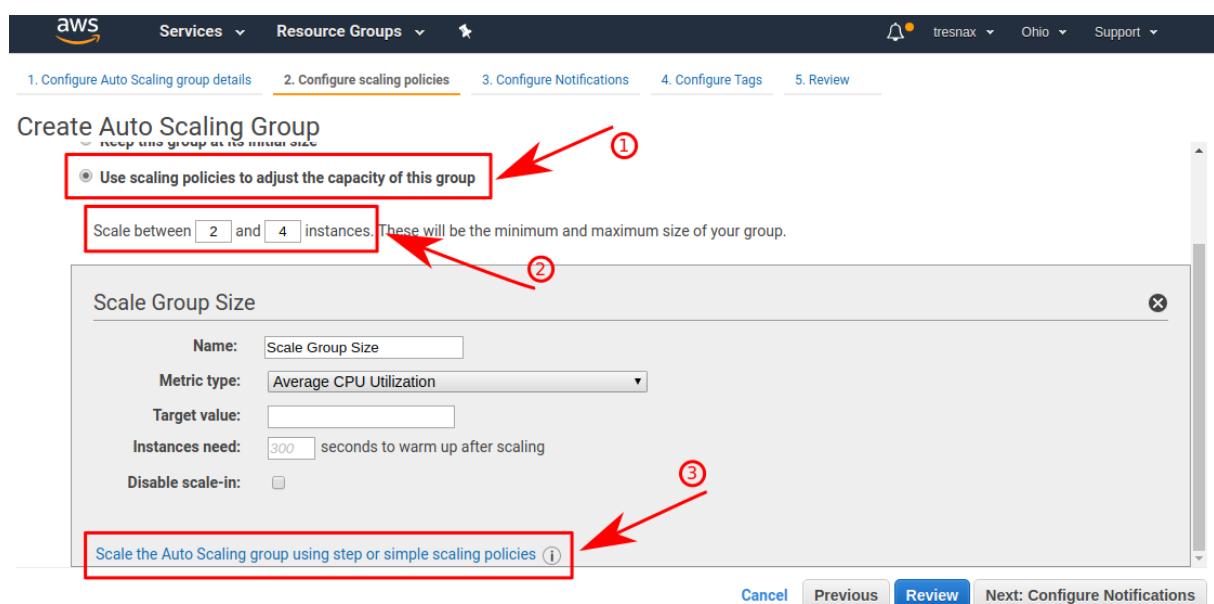


The screenshot shows the "Create Auto Scaling Group" wizard, Step 1: Launch Configuration. There are three tabs at the top: "Launch Configuration" (selected), "Launch Template", and "Create a new launch configuration". The "Launch Configuration" tab has a note: "You can continue to use your launch configurations if they support the Amazon EC2 features you need. [Learn more](#)". The "Launch Template" tab has a note: "Launch templates give you the option of launching one type of instance, or a combination of instance types and purchase options. Launch templates include the latest Amazon EC2 features and can be updated and versioned. [Learn more](#) [Create new launch template](#)". The "Create a new launch configuration" tab is empty. At the bottom right, there is a "Next Step" button.

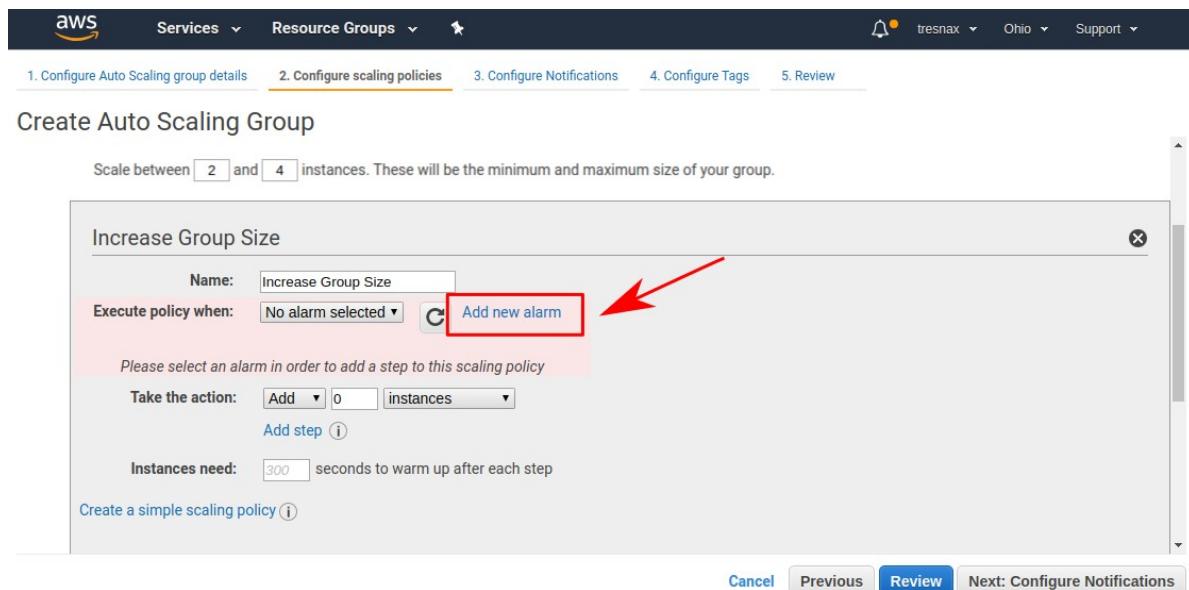
Isikan informasi detail mengenai Auto Scaling yang akan kita gunakan. Mulai dari group name, Group Size, Network dan Subnet. Group Size disini adalah jumlah instance yang akan kita scale nantinya. Jika sudah klik **Next : Configure Scaling Policies.**



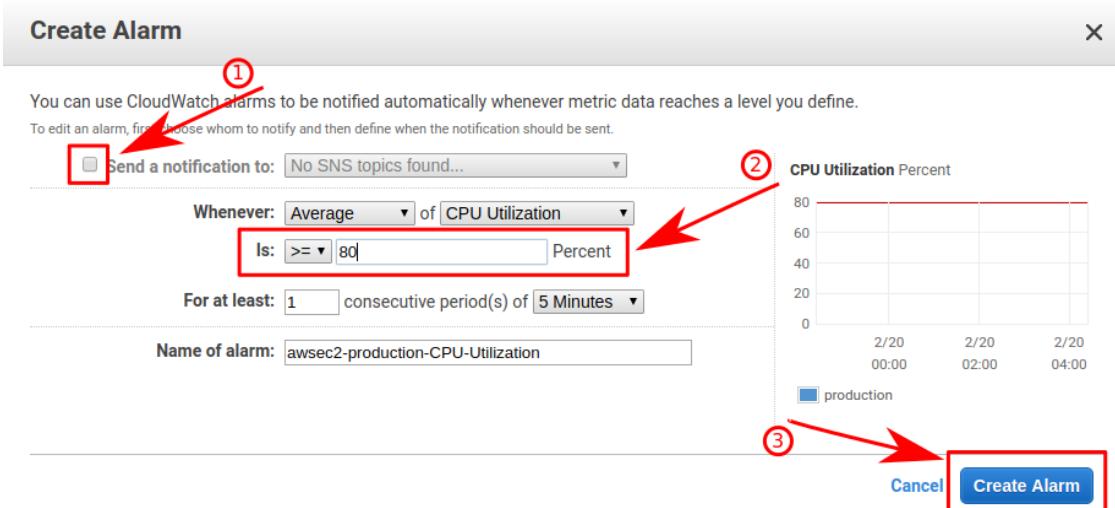
Bagian selanjutnya pilih **Use Scaling Policies**. Setelah itu sesuaikan bagian **Scale Between** untuk maximum dan minimum instance, disini saya menentukan **minimal 2 dan maximal 4** instance. Untuk pengaturan lanjutan pilih **Scale the Auto Scaling Group dibawah**.



Selanjutnya kita buat sebuah **alarm baru** untuk **Increase Group Size**, dimana nantinya ini akan jadi parameter ketika instance tersebut harus bertambah apabila CPU mengalami kenaikan.

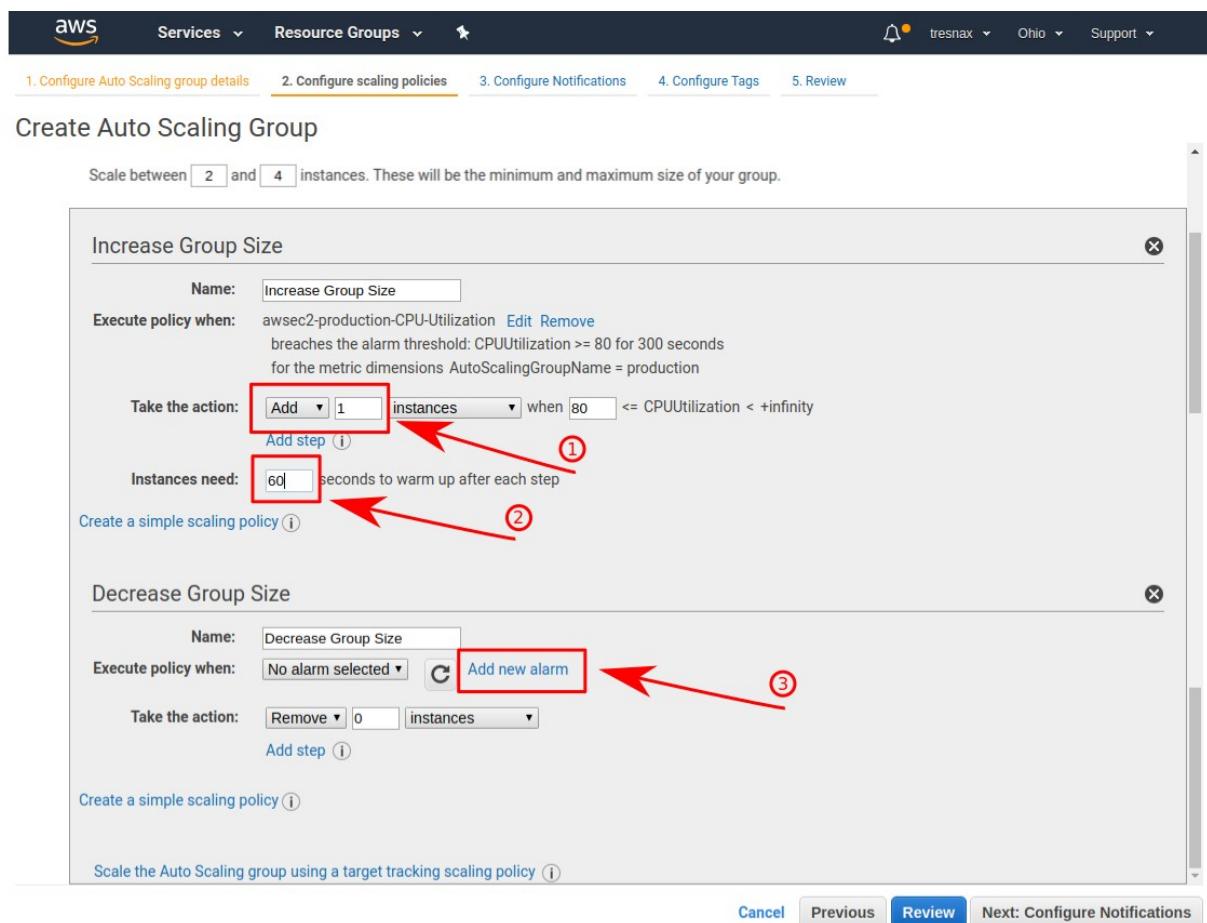


Isikan informasi seperti dibawah ini, **unchecklist untuk notification SNS**, lalu isikan persentase CPU menjadi ≥ 80 , setelah selesai klik **Create Alarm**.

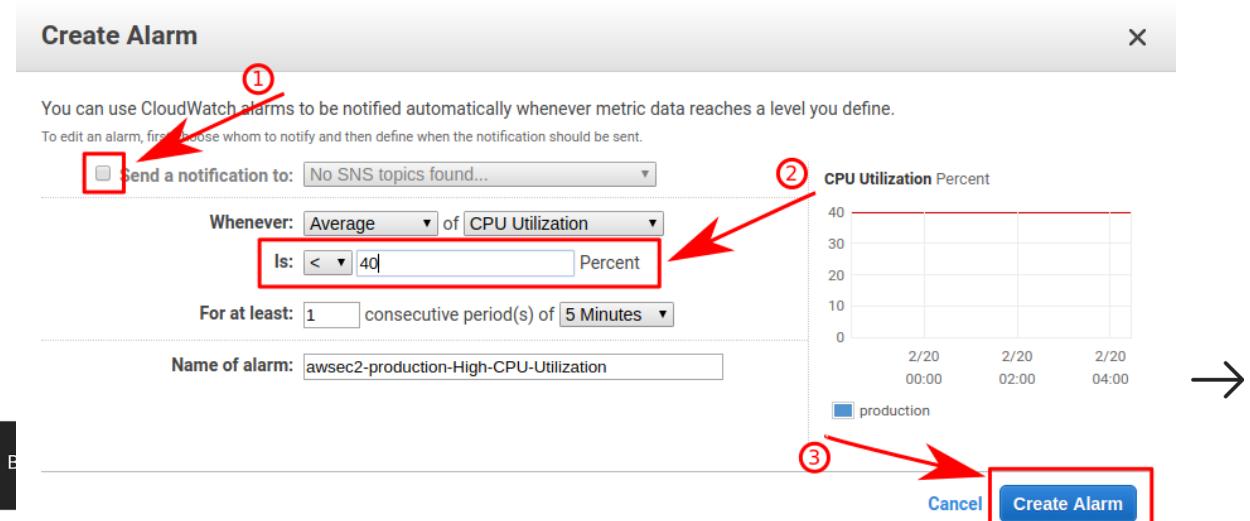


Setelah selesai menambah alarm, selanjutnya pada bagian **Take the action** pilih Add dengan angka 1 untuk menambah 1 instance setiap increase. Lalu waktu pada **instance need** diubah menjadi 60 detik.

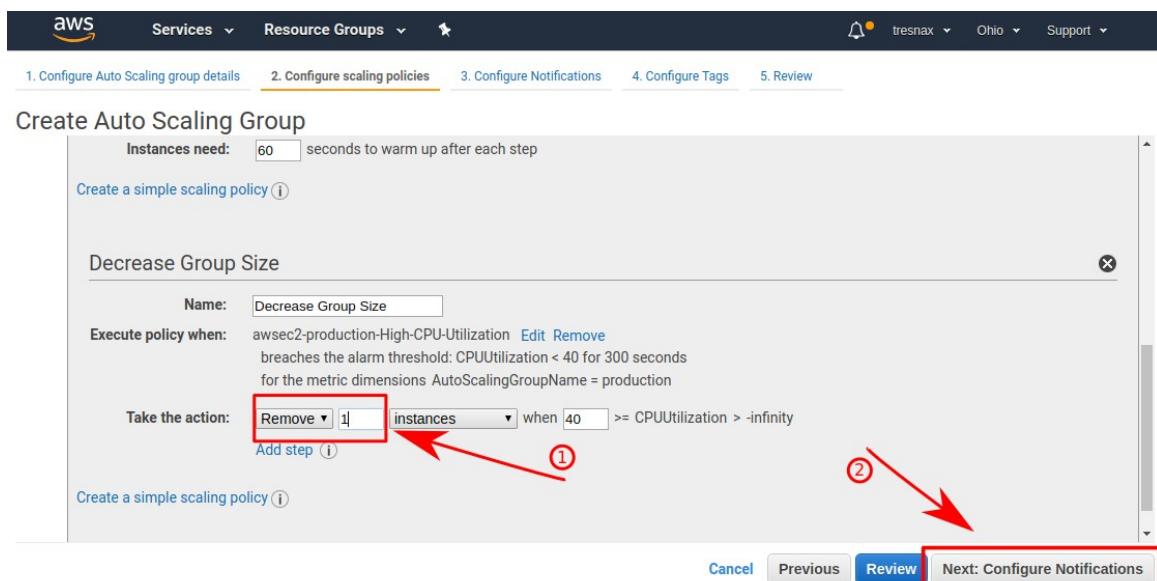
Pada bagian **Decrease Group Size** pilih **Add new alarm** untuk membuat alarm baru ketika server berada pada CPU terendah yang membuat instance berkurang.



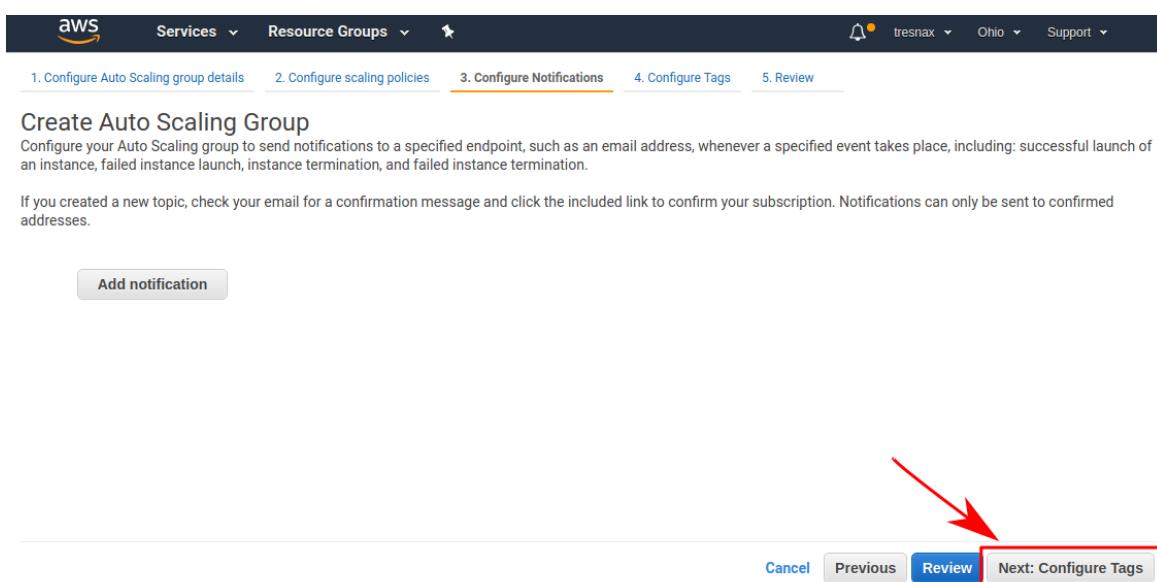
Pada bagian alarm ini berbeda dengan sebelumnya, karena kita akan mengurangi instance. Maka parameternya adalah pada saat CPU memiliki **< 40 percent**. Setelah itu **Create Alarm**.



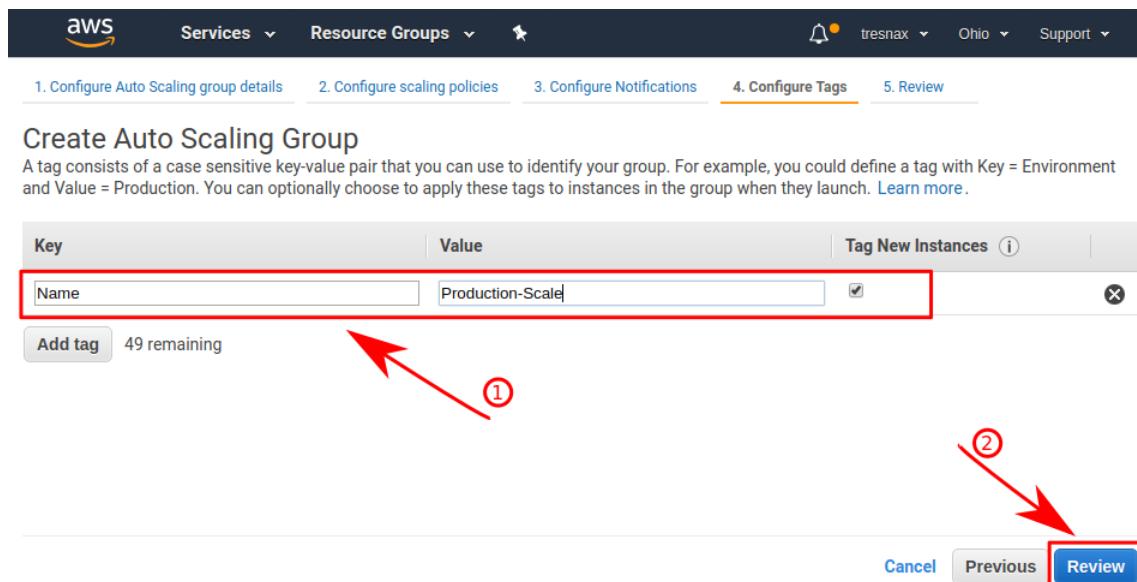
Setelah kita buat alarm tersebut, maka hasilnya akan seperti dibawah ini. Jangan lupa pada bagian **Take the action** pilih **remove dengan angka 1**, supaya nantinya setiap CPU instance turun menjadi 40 maka scale akan dikurangi 1. Selanjutnya pilih **Next : Configure Notifications** untuk melanjutkan.



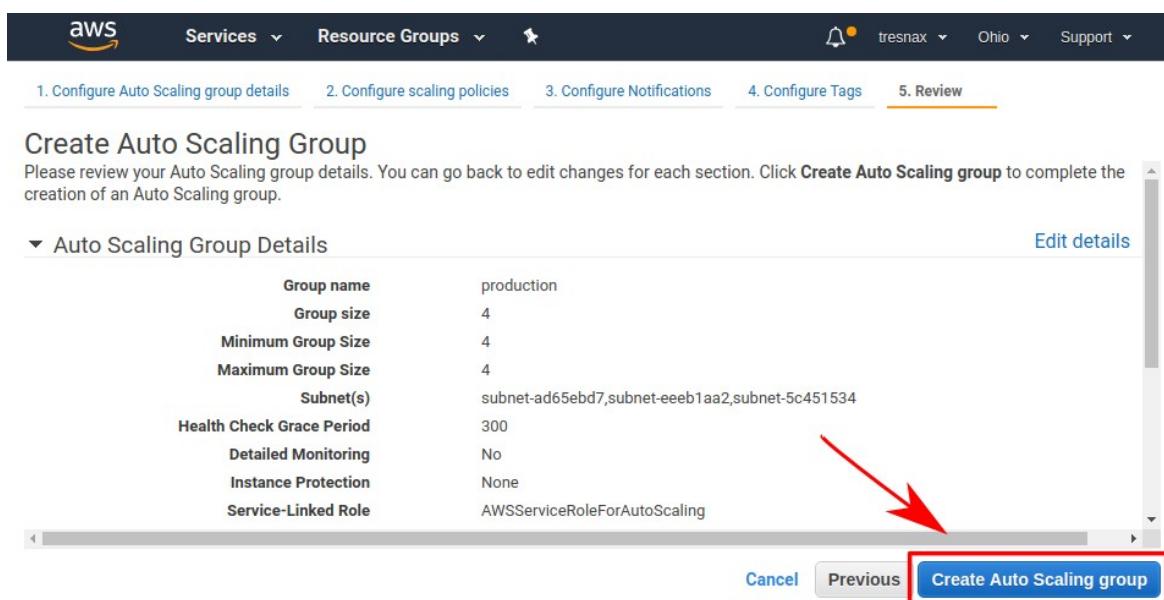
Selanjutnya adalah bagian notifikasi, kalian dapat mensetting notifikasi untuk scaling ini. Akan tetapi pada bagian ini kita tidak akan melakukannya. Maka selanjutnya pilih **Next : Configure Tags**.



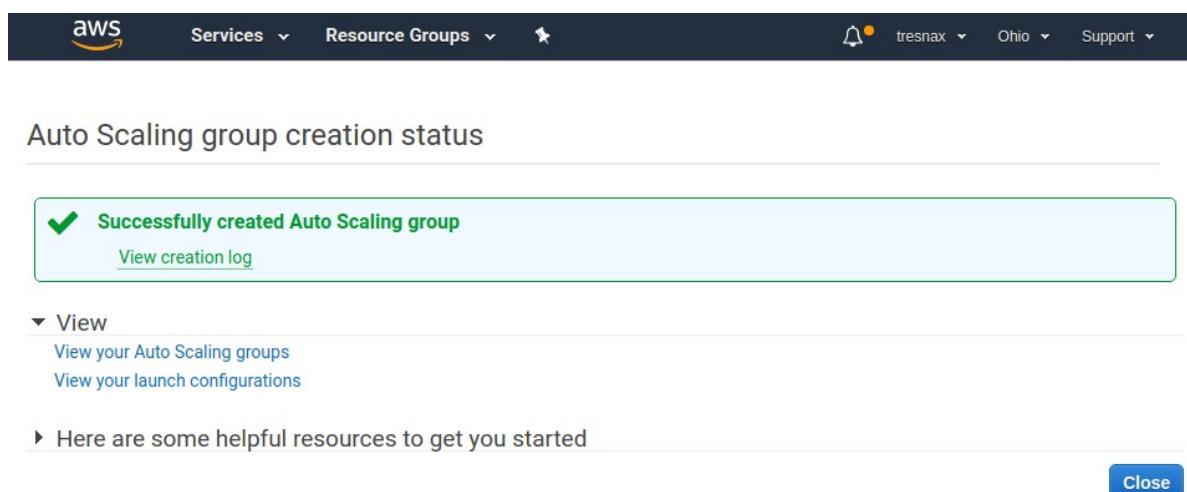
Selanjutnya isikan **tag** pada instance yang akan di scale oleh auto scaling, disini kita akan berikan tag berupa nama, sehingga ketika ada penambahan instance baru maka akan memiliki nama sesuai tag yang kita buat. Setelah itu klik **Review**.



Berikutnya akan muncul bagian review, bagian ini hanya menampilkan sekilas review dari konfigurasi yang sudah kita buat tadi. Klik **Create Auto Scaling group** untuk melanjutkan.



Jika berhasil maka akan muncul status berhasilnya seperti dibawah ini, klik **Close** untuk melanjutkan.



Auto Scaling group creation status

✓ Successfully created Auto Scaling group

[View creation log](#)

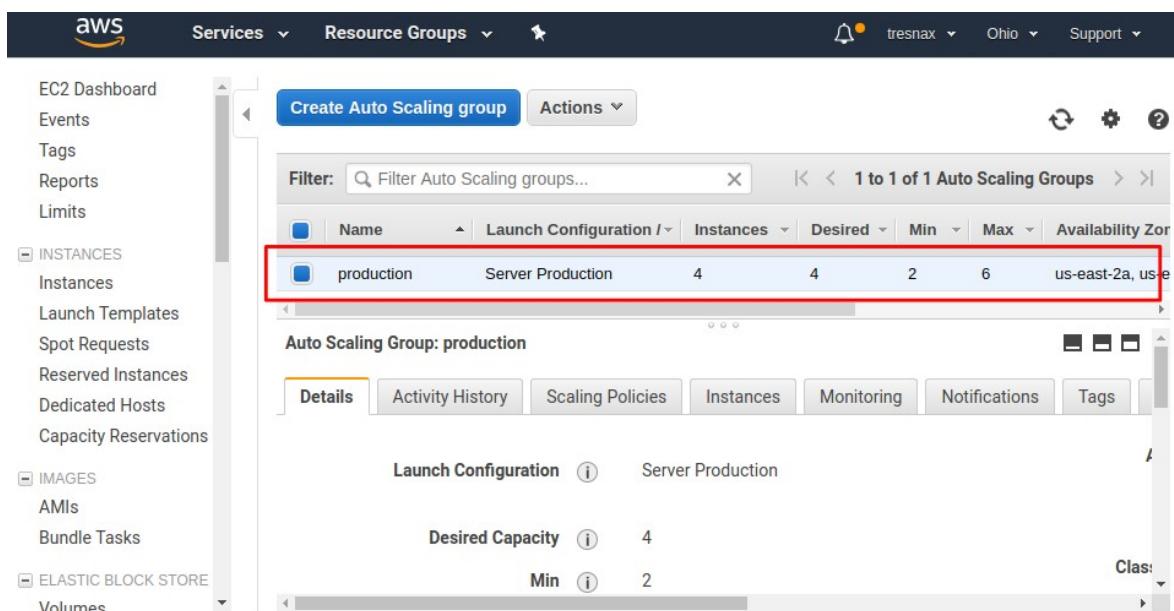
▼ View

- [View your Auto Scaling groups](#)
- [View your launch configurations](#)

► Here are some helpful resources to get you started

[Close](#)

Maka hasil dari konfigurasi Auto Scaling Grup yang sudah dibuat tadi akan terlihat seperti dibawah ini.



Name	Launch Configuration /	Instances	Desired	Min	Max	Availability Zon
production	Server Production	4	4	2	6	us-east-2a, us-e

Auto Scaling Group: production

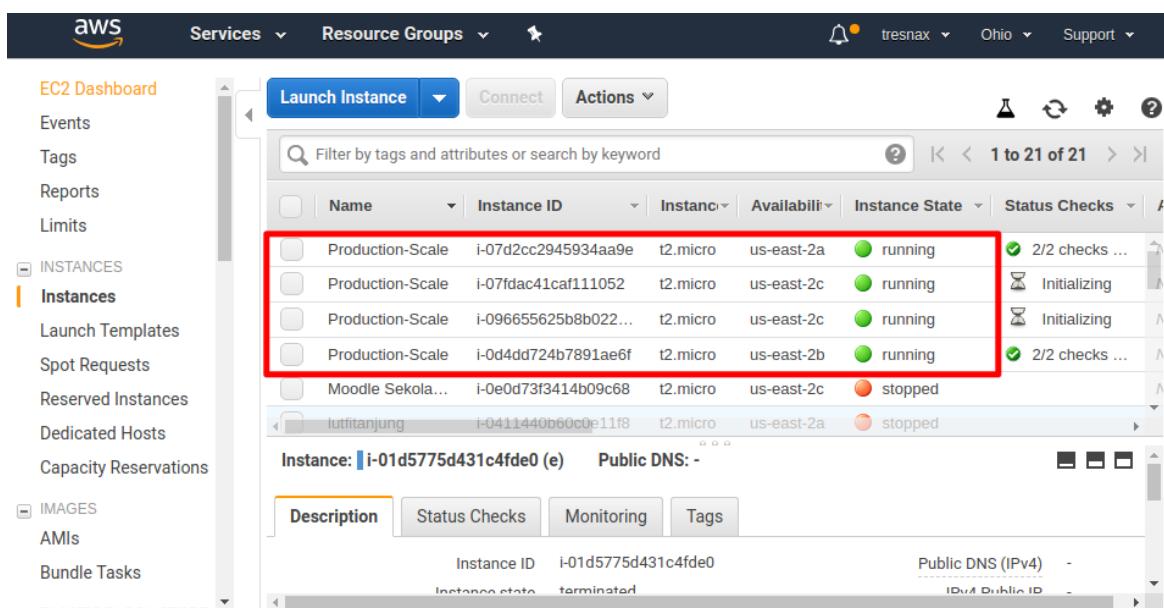
Details Activity History Scaling Policies Instances Monitoring Notifications Tags

Launch Configuration Server Production

Desired Capacity 4

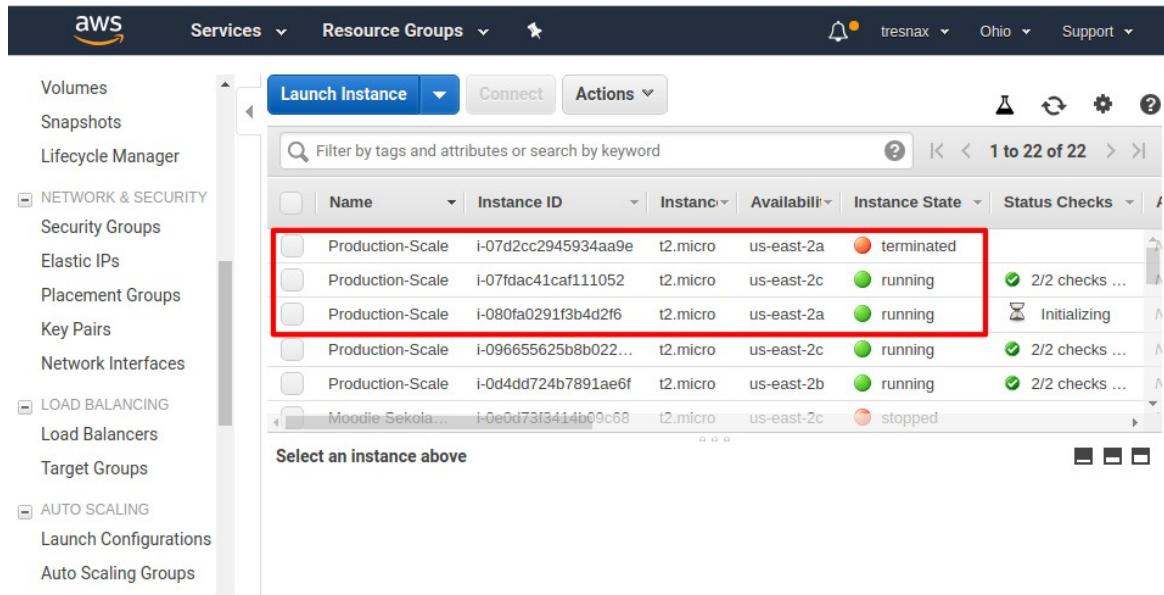
Min 2

Jika kita lihat di bagian menu Instance maka akan terlihat server yang sudah kita siapkan Auto Scaling akan muncul seperti dibawah ini. Akan muncul 4 server karena tadi kita sudah mengeset menjadi 4 server.



Name	Instance ID	Type	Region	State
Production-Scale	i-07d2cc2945934aa9e	t2.micro	us-east-2a	running
Production-Scale	i-07fdac41caf111052	t2.micro	us-east-2c	running
Production-Scale	i-096655625b8b022...	t2.micro	us-east-2c	running
Production-Scale	i-0d4dd724b7891ae6f	t2.micro	us-east-2b	running
Moodle Sekola...	i-0e0d73f3414b09c68	t2.micro	us-east-2c	stopped

Kita dapat **mengujicoba server tersebut** dengan melakukan **terminate** pada 3 server yang ada dan menyisakan 1 server. Maka akan dengan sendirinya server dibuat kembali menjadi 2 server. Itu karena minimal servernya adalah 2.



Name	Instance ID	Type	Region	State
Production-Scale	i-07d2cc2945934aa9e	t2.micro	us-east-2a	terminated
Production-Scale	i-07fdac41caf111052	t2.micro	us-east-2c	running
Production-Scale	i-080fa0291f3b4d2f6	t2.micro	us-east-2a	running
Production-Scale	i-096655625b8b022...	t2.micro	us-east-2c	running
Production-Scale	i-0d4dd724b7891ae6f	t2.micro	us-east-2b	running
Moodle Sekola...	i-0e0d73f3414b09c68	t2.micro	us-east-2c	stopped

Hasil setelah mencoba instance dimatikan

6.5.6. Exercise

- Buat sebuah AMI server baru untuk digunakan Auto Scaling.



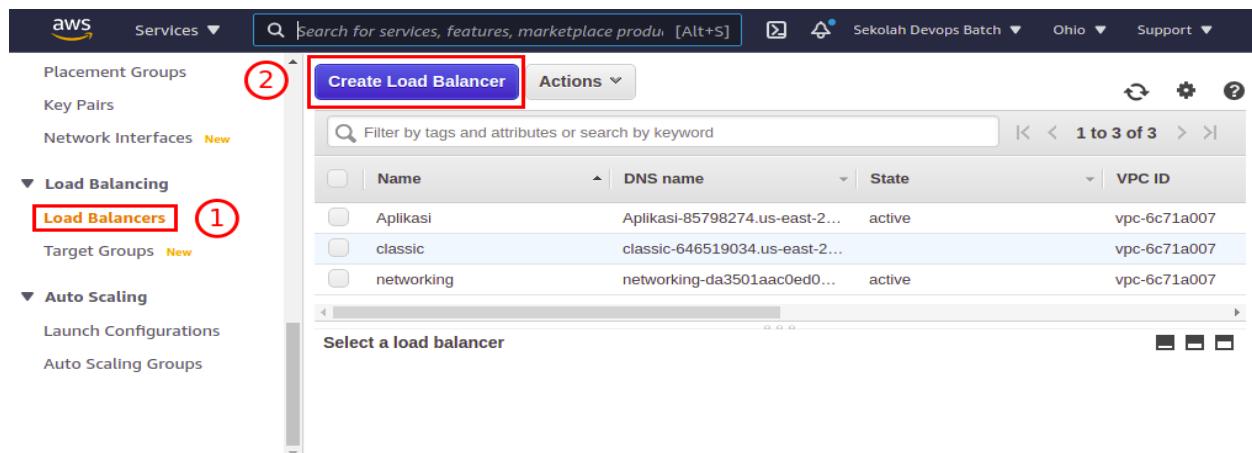
2. Buat sebuah Auto Scaling group baru dengan minimal instance 2 dan maximal 8
3. Atur penambahan instance ketika CPU naik 60% dan pengurangan instance ketika CPU turun menjadi 30%.

6.6. Integrasi Load balancing dan Auto Scaling

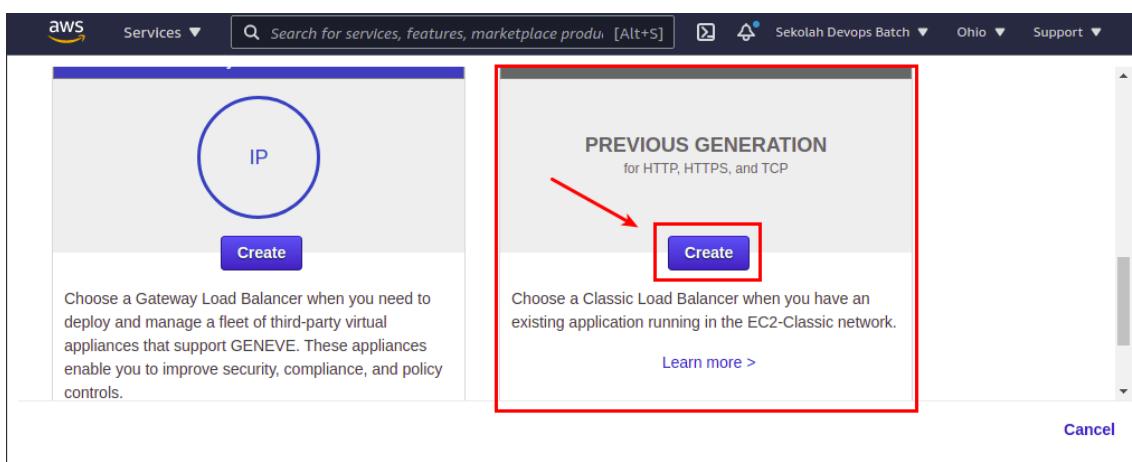
Setelah kedua materi antara Load Balancing dan Auto Scaling telah kita pelajari, sekarang kita akan coba kombinasikan keduanya. Karena Auto Scaling tidak akan berjalan maksimal tanpa adanya load balancing. Maka kita perlu membuat load balancing dan memasangkannya pada auto scaling yang sudah kita buat tadi.

Pertama kita akan membuat load balancing terlebih dahulu, disini kita akan gunakan **classic load balancing**, tahap pembuatannya tidak akan berbeda dengan sebelumnya. Hanya ada beberapa yang diubah saja.

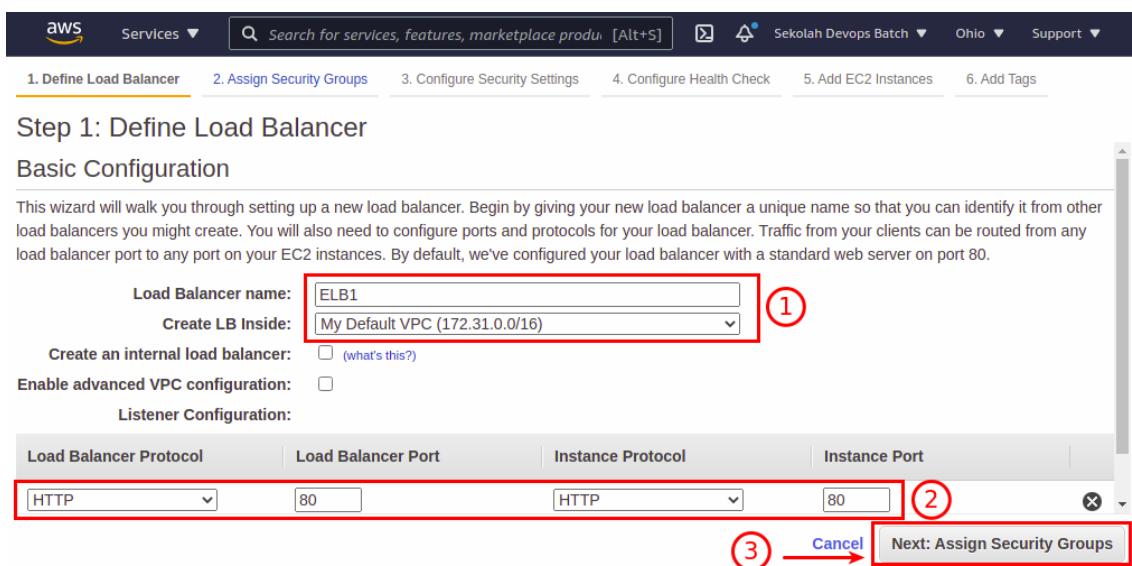
Masuk kedalam menu **Load balancing** dan pilih tombol **Create Load Balancer** untuk membuat ELB baru.



Kita akan diberikan pilihan **model Load Balancer** mana yang akan kita buat, disini kita pilih **Classic Load Balancer**, lalu klik tombol **Create**.

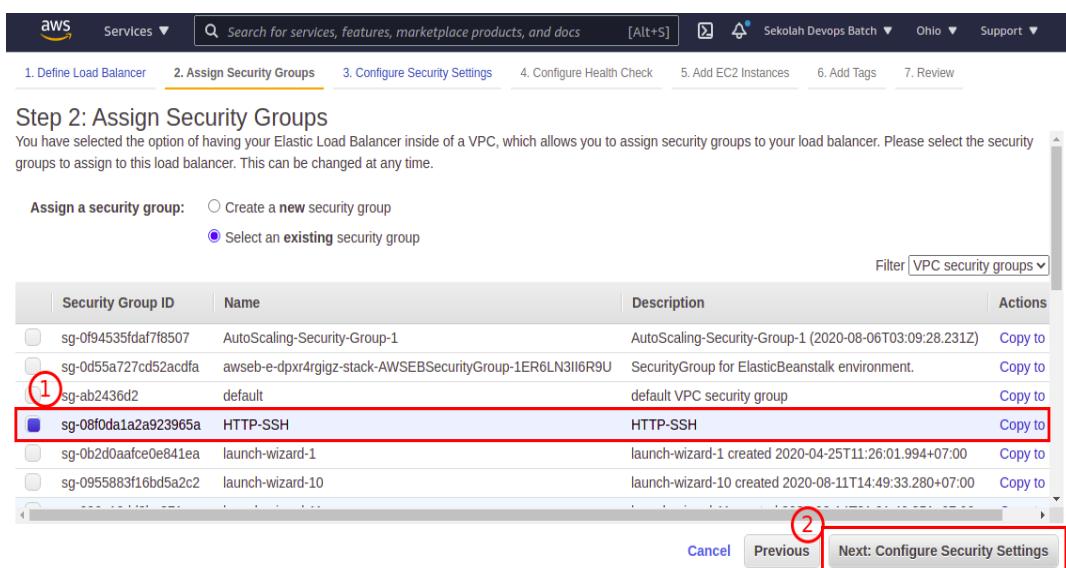


Setelah itu kita akan diarahkan ke menu **Basic Configuration**. Isikan Load Balancer Name : ELB1. Lalu tambahkan **protocol http** dengan port 80. Klik tombol **Next : Assign Security Group** untuk melanjutkan.



Selanjutnya kita masuk ke menu **security group**, pilih security group yang sudah kita buat sebelumnya kemudian klik **Next Configure Security Settings**.





Step 2: Assign Security Groups

You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.

Assign a security group:

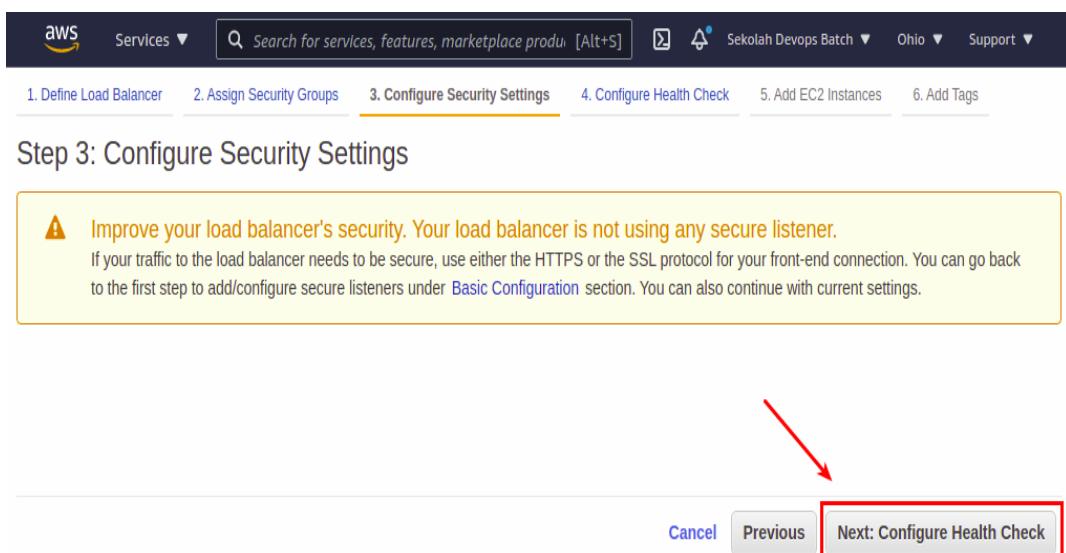
- Create a new security group
- Select an existing security group

Filter: VPC security groups

Security Group ID	Name	Description	Actions
sg-0f94535fdaf7f8507	AutoScaling-Security-Group-1	AutoScaling-Security-Group-1 (2020-08-06T03:09:28.231Z)	Copy to
sg-0d55a727cd52acdfa	awseb-e-dpxr4rgigz-stack-AWSEBSecurityGroup-1ER6LN3II6R9U	SecurityGroup for ElasticBeanstalk environment.	Copy to
1 sg-ab2436d2	default	default VPC security group	Copy to
sg-0f0da1a2a923965a	HTTP-SSH	HTTP-SSH	Copy to
sg-0b2d0aafe0e841ea	launch-wizard-1	launch-wizard-1 created 2020-04-25T11:26:01.994+07:00	Copy to
sg-0955883f16bd5a2c2	launch-wizard-10	launch-wizard-10 created 2020-08-11T14:49:33.280+07:00	Copy to

Cancel Previous Next: Configure Security Settings (2)

Pada bagian selanjutnya langsung saja klik **Next Configure Health Check**.



Step 3: Configure Security Settings

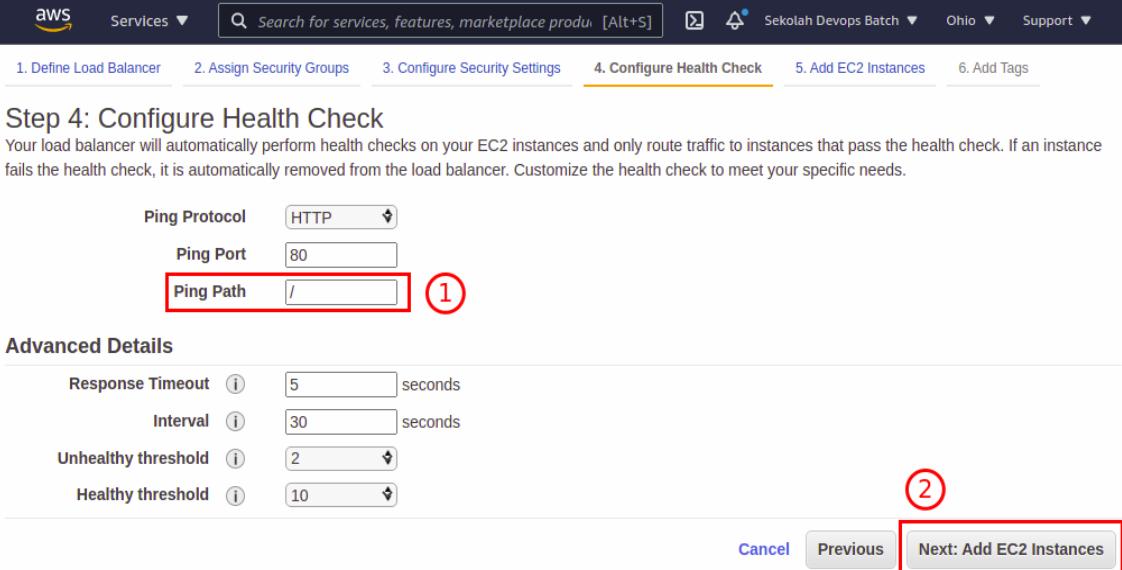
A Improve your load balancer's security. Your load balancer is not using any secure listener.
 If your traffic to the load balancer needs to be secure, use either the HTTPS or the SSL protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under [Basic Configuration](#) section. You can also continue with current settings.

Cancel Previous Next: Configure Health Check (2)

Selanjutnya kita dapat mengatur **health check** dengan parameter ping protocol HTTP, ping port 80 dan ping path dengan "/". Pada bagian Advanced setting kita dapat menyesuaikan seperti pengaturan dibawah ini.

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.



Pada bagian ini sedikit berbeda, karena kita tidak perlu memilih dan meregister instance yang akan kita gunakan, langsung kita lewati saja dengan klik **Next Add Tags.**

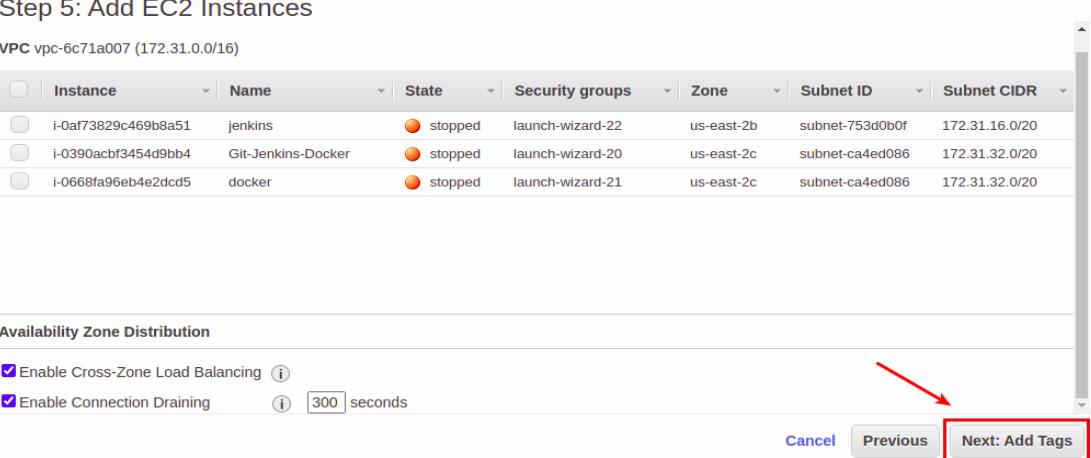
Step 5: Add EC2 Instances

VPC `vpc-6c71a007 (172.31.0.0/16)`

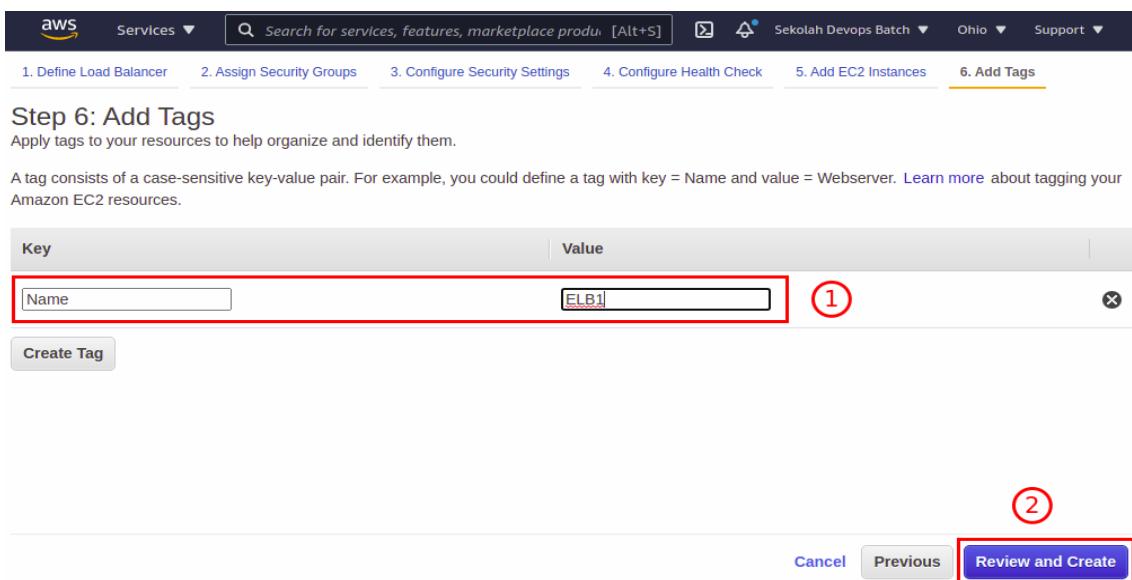
Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
i-0af73829c469b8a51	jenkins	stopped	launch-wizard-22	us-east-2b	subnet-753d0b0f	172.31.16.0/20
i-0390acbf3454d9bb4	Git-Jenkins-Docker	stopped	launch-wizard-20	us-east-2c	subnet-ca4ed086	172.31.32.0/20
i-0668fa96eb4e2dc5	docker	stopped	launch-wizard-21	us-east-2c	subnet-ca4ed086	172.31.32.0/20

Availability Zone Distribution

Enable Cross-Zone Load Balancing (i)
 Enable Connection Draining (i) [300] seconds



Buat mebuah tag baru dengan menekan tombol **Create Tag** kemudian isi Key dengan **Name** dan Value dengan **ELB1**, tag ini berguna untuk memberikan nama pada ELB classic.



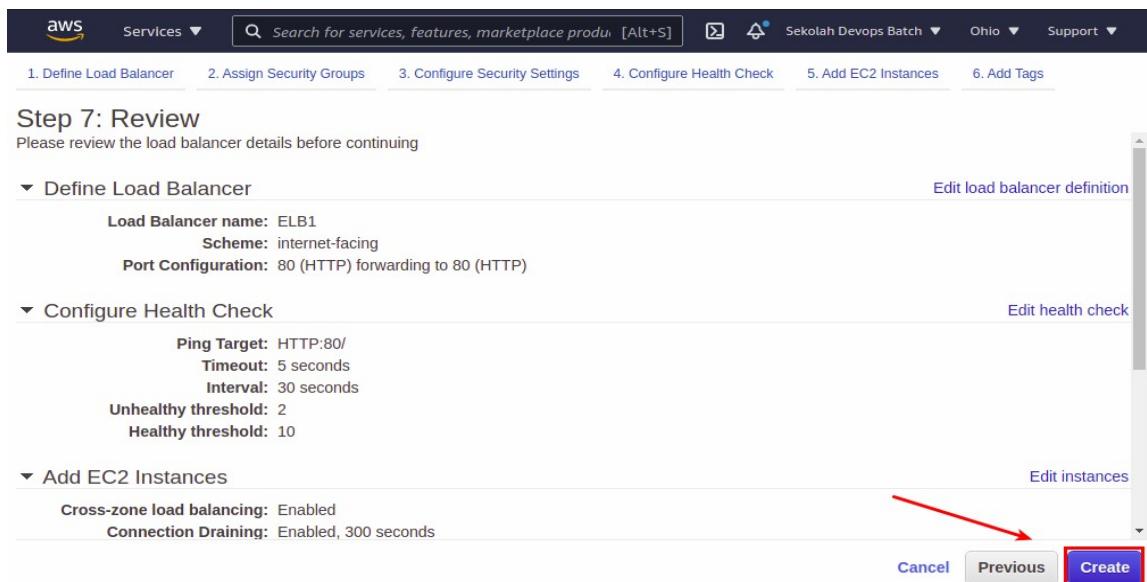
Step 6: Add Tags
Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value
Name	ELB1

Create Tag

Setelah semua konfigurasi dan setting telah dilakukan, maka kita dapat melihat kembali rincian dari sudah kita konfigurasi tadi sebelumnya. Untuk melanjutkan klik **Create**.



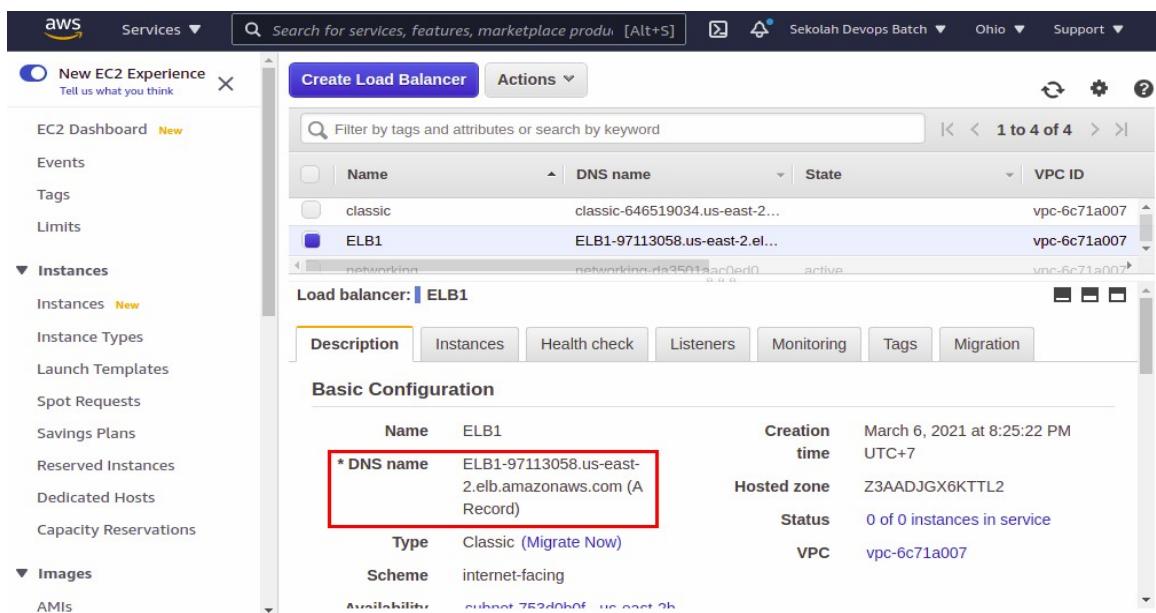
Step 7: Review
Please review the load balancer details before continuing

- Define Load Balancer**
 - Load Balancer name: ELB1
 - Scheme: internet-facing
 - Port Configuration: 80 (HTTP) forwarding to 80 (HTTP)
- Configure Health Check**
 - Ping Target: HTTP:80/
 - Timeout: 5 seconds
 - Interval: 30 seconds
 - Unhealthy threshold: 2
 - Healthy threshold: 10
- Add EC2 Instances**
 - Cross-zone load balancing: Enabled
 - Connection Draining: Enabled, 300 seconds

Edit load balancer definition **Edit health check** **Edit instances**

Create

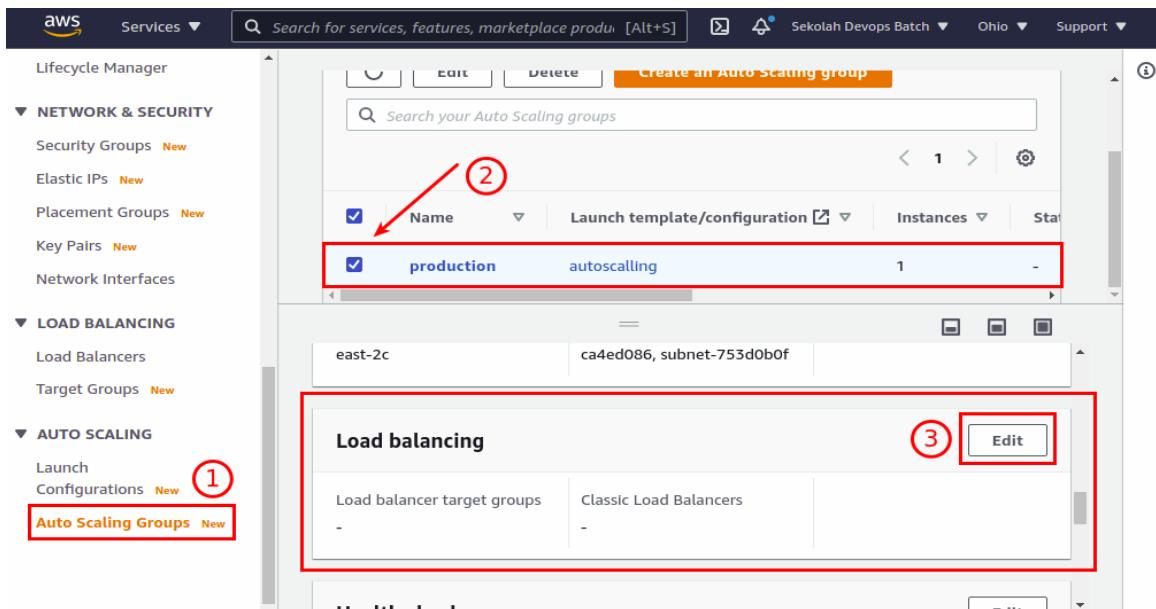
Setelah selesai, kita akan diarahkan ke dalam menu **dashboard Elastic Load Balancer** dan kita dapat lihat konfigurasi yang sudah kita buat tadi. Untuk melihat DNS name url, kita dapat meneklik tombol description yang ada di menu bawah ELB.



The screenshot shows the AWS Load Balancer configuration page. In the 'Basic Configuration' section, the 'DNS name' field is highlighted with a red box. The configuration includes:

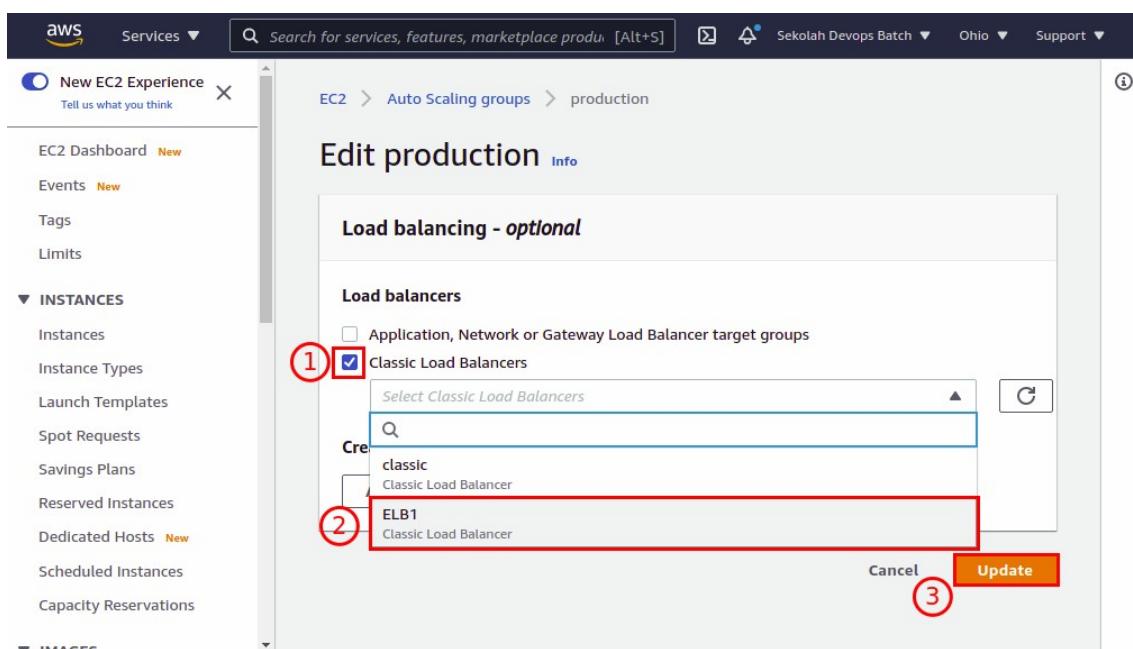
- Name: ELB1
- * DNS name: ELB1-97113058.us-east-2.elb.amazonaws.com (A Record)
- Type: Classic (Migrate Now)
- Scheme: internet-facing
- Creation time: March 6, 2021 at 8:25:22 PM UTC+7
- Hosted zone: Z3AADJGX6KTTL2
- Status: 0 of 0 instances in service
- VPC: vpc-6c71a007

Selanjutnya pindah kedalam menu **Auto Scaling Group**, lalu pada bagian tab **Details** scroll ke bawah hingga menemukan menu **Load Balancing**, klik tombol **Edit**.



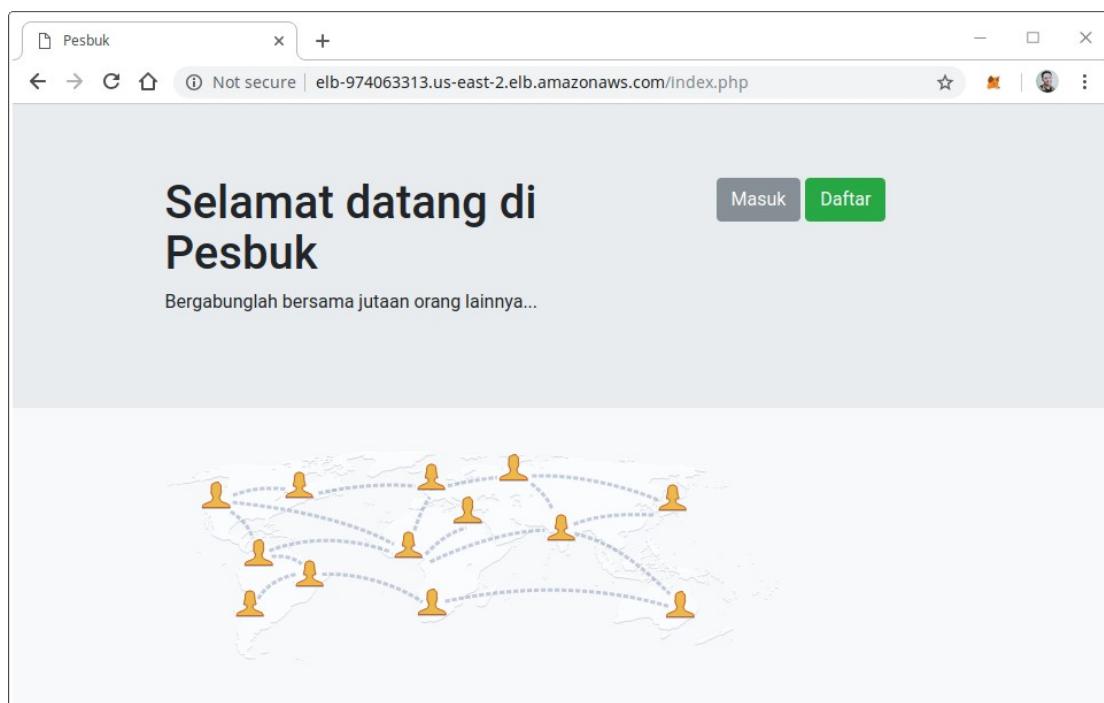
The screenshot shows the AWS Auto Scaling Groups configuration page. Step 1 highlights the 'Auto Scaling Groups' link. Step 2 highlights the 'production' group. Step 3 highlights the 'Edit' button in the 'Load balancing' section of the pop-up window.

Akan muncul sebuah pop-out konfigurasi. Check pada **Classic Load Balancer** seperti pada gambar lalu pilih load balancer yang sudah kita buat tadi. Lalu klik **Update** untuk menyimpan.



The screenshot shows the 'Edit production' page for an Auto Scaling group named 'production'. In the 'Load balancing - optional' section, there is a dropdown menu titled 'Select Classic Load Balancers'. The menu lists 'classic' and 'ELB1' under 'Classic Load Balancer'. The 'ELB1' option is highlighted with a red box and circled with the number 2. A red box also highlights the checked checkbox for 'Classic Load Balancers' (circled with number 1). The 'Update' button at the bottom right is circled with the number 3.

Setelah selesai menyimpan, selanjutnya ujicoba load balance yang sudah kita buat dengan memanggil alamat load balancer tersebut. Harusnya akan muncul web aplikasi dari instance auto scaling yang kita buat load balancer.



Hasil Load Balancer dan AutoScaling

6.7. Summary

1. Pada bab ini kita sudah membahas mengenai dasar load balancing dan autoscaling, harusnya kita sudah paham konsep dari penerapan kedua layanan ini di AWS.
2. Kita sudah coba mengaplikasikan 3 model load balancer, yaitu classic, application, dan network load balancer. Ketiga model tersebut memiliki fungsional sesuai dengan kebutuhan aplikasi yang kita miliki.
3. Auto Scaling pada bab ini sudah kita ujicoba dengan membuat sebuah server yang akan otomatis melakukan deployment aplikasi dari github, sehingga ketika server mulai melakukan scale, maka server baru yang dibuat akan otomatis mengaktifkan aplikasi baru dari scaling tersebut.
4. Auto Scaling yang kita buat menggunakan minimal 2 server dan maksimal 4 server untuk penggunaannya. Server akan bertambah ketika beban CPU mencapai 80% dan akan berkurang ketika beban CPU 40%.
5. Melakukan kombinasi Load Balancing pada auto scaling memberikan performa yang maksimal, dimana semua server yang sudah di scale oleh auto scaling akan otomatis disatukan dengan menggunakan load balancing.