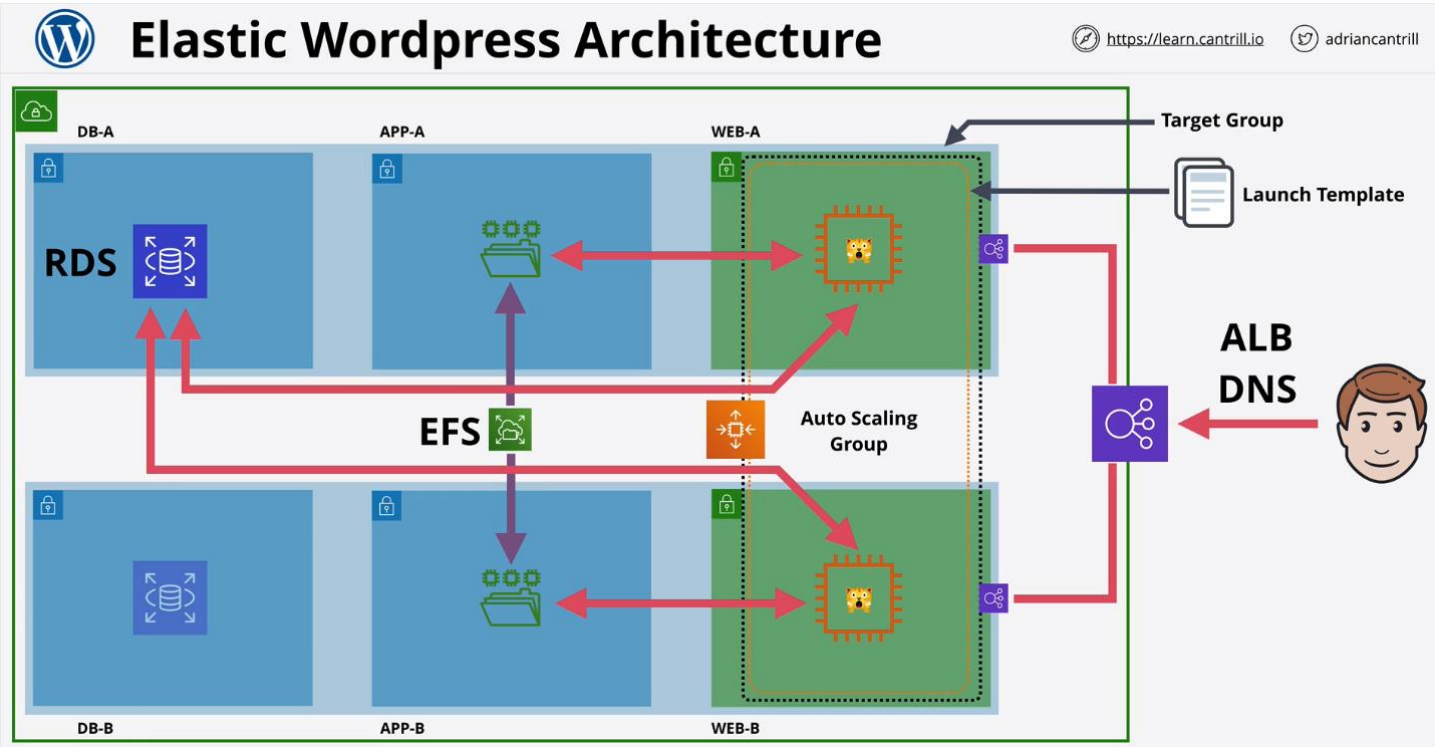
SUMMARY:

This Lab covers HA & Scaling: Creating launch templates, ASGs, Load Balancers

Contents

1. HA and Scaling Architecture for Demo:

In this [DEMO] lesson series you will implement a fully scalable, highly available and self-healing architecture.



2. PART 1 — Launch Template

In PART 1 of this [DEMO] series you will learn how to create a Launch Template which defines WHAT is launched and WHAT is configured within an Elastic Architecture.

Use CFN files in lesson to create stacks.

SSM parameter used:

```
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "root"
    },
    "logs": {
```

```
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/var/log/secure",
                        "log_group_name": "/var/log/secure",
                        "log_stream_name": "{instance_id}"
                    },
                    {
                        "file_path": "/var/log/httpd/access_log",
                        "log_group_name": "/var/log/httpd/access_log",
                        "log_stream_name": "{instance_id}"
                    },
                    {
                        "file_path": "/var/log/httpd/error_log",
                        "log_group_name": "/var/log/httpd/error_log",
                        "log_stream_name": "{instance_id}"
                    }
                ]
            }
        }
    },
    "metrics": {
        "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
        },
        "metrics_collected": {
            "collectd": {
                "metrics_aggregation_interval": 60
            },
            "cpu": {
                "measurement": [
                    "cpu_usage_idle",
                    "cpu_usage_iowait",
                    "cpu_usage_user",
                    "cpu_usage_system"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ],
                "totalcpu": false
            },
            "disk": {
                "measurement": [
                    "used_percent",
                    "inodes_free"
                ],
                "metrics_collection_interval": 60,
```

```
                "resources": [
                    "*"
                ]
            },
            "diskio": {
                "measurement": [
                    "io_time",
                    "write_bytes",
                    "read_bytes",
                    "writes",
                    "reads"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            },
            "mem": {
                "measurement": [
                    "mem_used_percent"
                ],
                "metrics_collection_interval": 60
            },
            "netstat": {
                "measurement": [
                    "tcp_established",
                    "tcp_time_wait"
                ],
                "metrics_collection_interval": 60
            },
            "statsd": {
                "metrics_aggregation_interval": 60,
                "metrics_collection_interval": 10,
                "service_address": ":8125"
            },
            "swap": {
                "measurement": [
                    "swap_used_percent"
                ],
                "metrics_collection_interval": 60
            }
        }
    }
}
```

Parameter store values:

```
# Parameter Store Values
NAME TYPE VALUE

/A4L/DefaultInstance STRING t2.micro
```

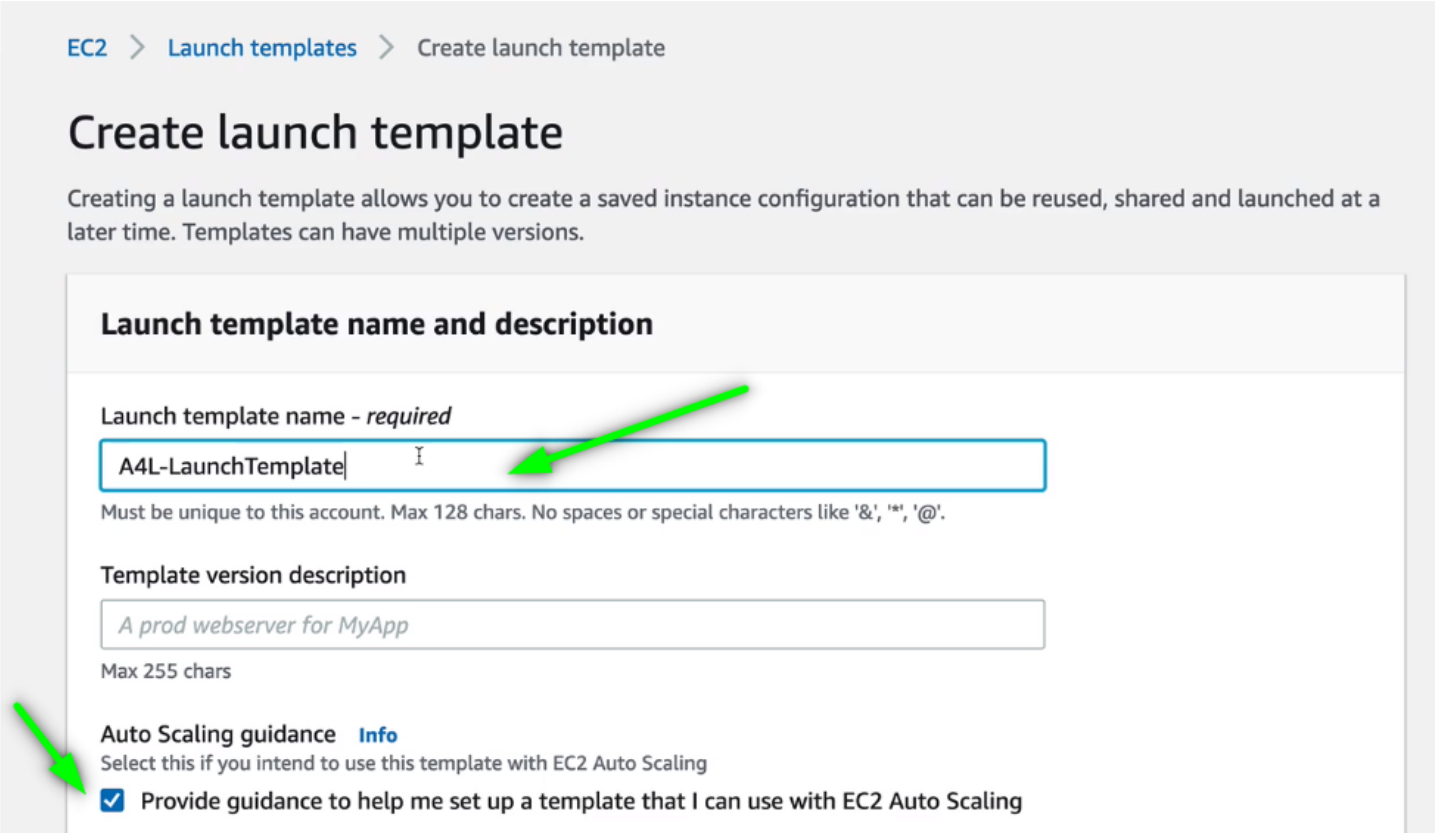
```
/A4L/Wordpress/DBName STRING a4lwordpress
/A4L/Wordpress/DBUser STRING a4lwordpress
/A4L/Wordpress/DBPassword SECURESTRING XXXXXX
/A4L/Wordpress/DBRootPassword SECURESTRING XXXXXX
```

Launch template:

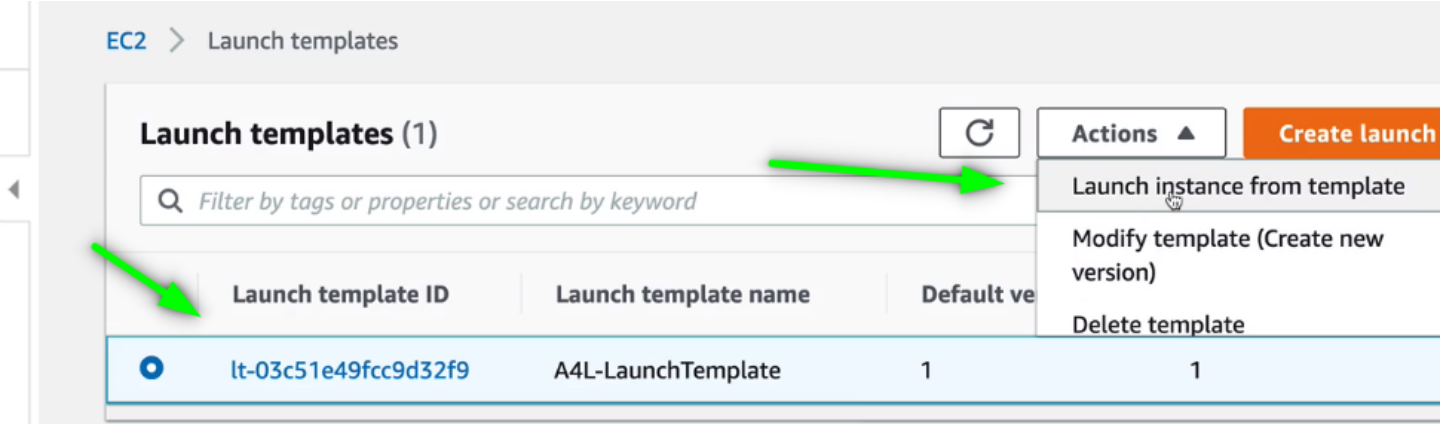
```
#!/bin/bash -xe
yum -y update
# STEP 1 - Set Config variables
DBPassword=$(aws ssm get-parameters --region us-east-1 --names /A4L/Wordpress/DBPassword --
with-decryption --query Parameters[0].Value)
DBPassword=`echo $DBPassword | sed -e 's/^"//' -e 's/"$//'`
DBUser=$(aws ssm get-parameters --region us-east-1 --names /A4L/Wordpress/DBUser --
query Parameters[0].Value)
DBUser=`echo $DBUser | sed -e 's/^"//' -e 's/"$//'`
DBName=$(aws ssm get-parameters --region us-east-1 --names /A4L/Wordpress/DBName --
query Parameters[0].Value)
DBName=`echo $DBName | sed -e 's/^"//' -e 's/"$//'`

a4ldbendpoint=$(aws cloudformation list-exports --region us-east-1 --
query 'Exports[?Name==`a4l-db-endpoint`].Value' --output text)
a4lvpc1wordpressefs=$(aws cloudformation list-exports --region us-east-1 --
query 'Exports[?Name==`a4l-vpc1-wordpress-efs`].Value' --output text)

# STEP 2 - Begin Configuration
yum -y install httpd wget cowsay mariadb amazon-efs-utils
amazon-linux-extras install -y php7.2
amazon-linux-extras install epel -y
yum install stress -y
systemctl enable httpd
systemctl start httpd

mkdir -p /var/www/html/wp-content
chown -R ec2-user:apache /var/www/
echo -e "$a4lvpc1wordpressefs:/ /var/www/html/wp-
content efs _netdev,tls,iam 0 0" >> /etc/fstab
mount -a -t efs defaults

# STEP 3 - CWAgent Install
rpm -Uvh https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-
cloudwatch-agent.rpm
mkdir -p /usr/share/collectd/
touch /usr/share/collectd/types.db
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -
c ssm:AmazonCloudWatch-linux -s

# STEP 4 - Install Wordpress
wget http://wordpress.org/latest.tar.gz -P /var/www/html
cd /var/www/html
tar -zxvf latest.tar.gz
cp -rvf wordpress/* .
```

```
rm -R wordpress
rm latest.tar.gz
echo "<html><head><title>test</title></head><body>test</body></html>" > /var/www/html/healthc
heck.html

# STEP 5 - Configure Wordpress
cp ./wp-config-sample.php ./wp-config.php
sed -i "s/'localhost'/'$a4ldbendpoint'/g" wp-config.php
sed -i "s/'database_name_here'/'$DBName'/g" wp-config.php
sed -i "s/'username_here'/'$DBUser'/g" wp-config.php
sed -i "s/'password_here'/'$DBPassword'/g" wp-config.php

# Step 6a - permissions
usermod -a -G apache ec2-user
chown -R ec2-user:apache /var/www
chmod 2775 /var/www
find /var/www -type d -exec chmod 2775 {} \;
find /var/www -type f -exec chmod 0664 {} \;

# STEP 6 COWSAY
echo "#!/bin/sh" > /etc/update-motd.d/40-cow
echo 'cowsay "Amazon Linux 2 AMI - Animals4Life"' > /etc/update-motd.d/40-cow
chmod 755 /etc/update-motd.d/40-cow
rm /etc/update-motd.d/30-banner
update-motd

# Step 7 Create update_wp_id.sh file
cat >> /home/ec2-user/update_wp_ip.sh<< 'EOF'
#!/bin/bash
source <(php -r 'require("/var/www/html/wp-
config.php"); echo("DB_NAME=".DB_NAME."; DB_USER=".DB_USER."; DB_PASSWORD=".DB_PASSWORD."; DB
_HOST=".DB_HOST); ')
SQL_COMMAND="mysql -u $DB_USER -h $DB_HOST -p$DB_PASSWORD $DB_NAME -e"
OLD_URL=$(mysql -u $DB_USER -h $DB_HOST -p$DB_PASSWORD $DB_NAME -
e 'select option_value from wp_options where option_id = 1;' | grep http)
HOST=$(curl http://169.254.169.254/latest/meta-data/public-ipv4)
$SQL_COMMAND "UPDATE wp_options SET option_value = replace(option_value, '$OLD_URL', 'http://
$HOST') WHERE option_name = 'home' OR option_name = 'siteurl';"
$SQL_COMMAND "UPDATE wp_posts SET guid = replace(guid, '$OLD_URL','http://$HOST');"
$SQL_COMMAND "UPDATE wp_posts SET post_content = replace(post_content, '$OLD_URL', 'http://$H
OST');"
$SQL_COMMAND "UPDATE wp_postmeta SET meta_value = replace(meta_value,'$OLD_URL','http://$HOST
');"
EOF

chmod 755 /home/ec2-user/update_wp_ip.sh
echo "/home/ec2-user/update_wp_ip.sh" >> /etc/rc.local
/home/ec2-user/update_wp_ip.sh
```

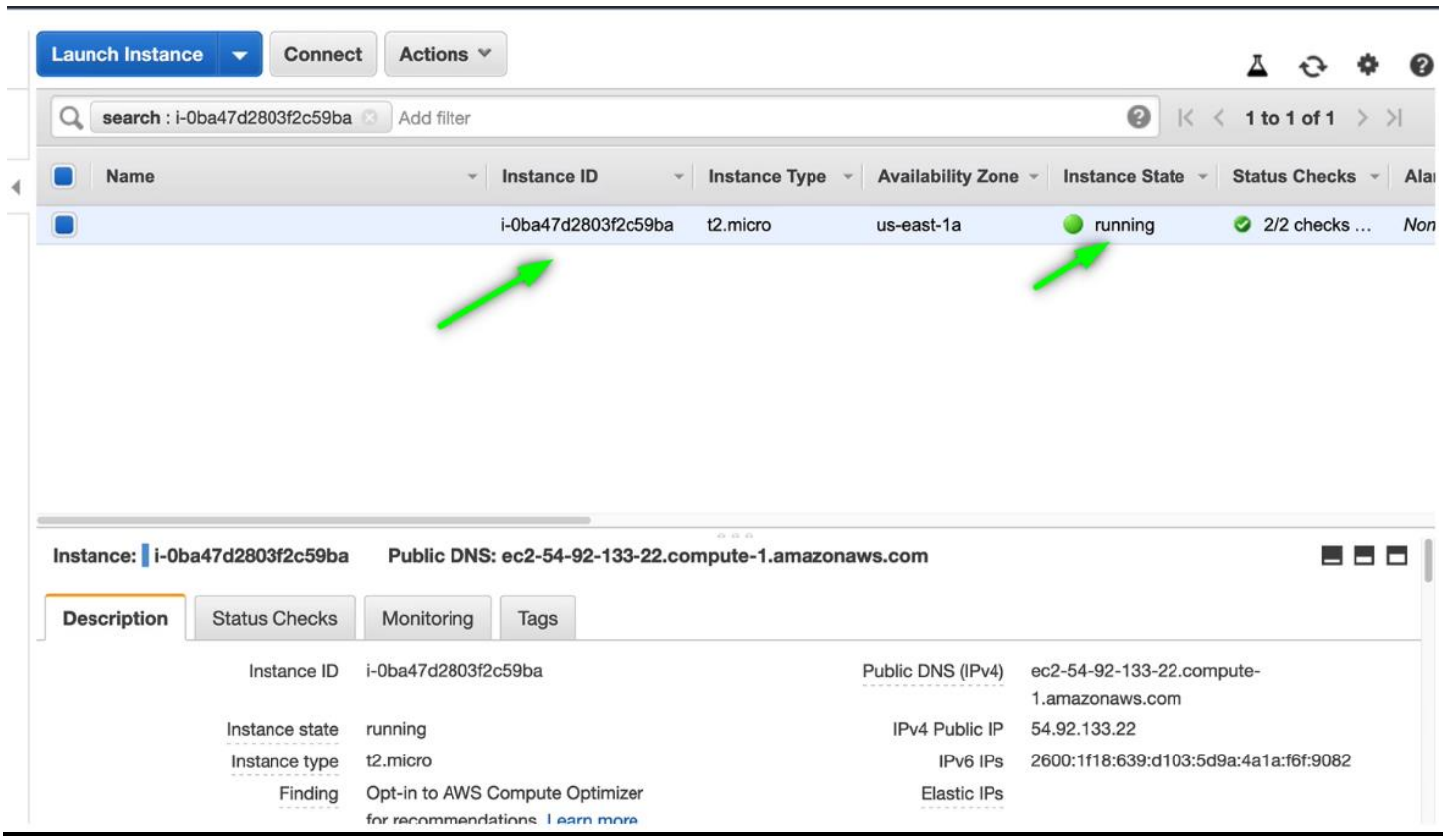
Create launch template in console



Launch template



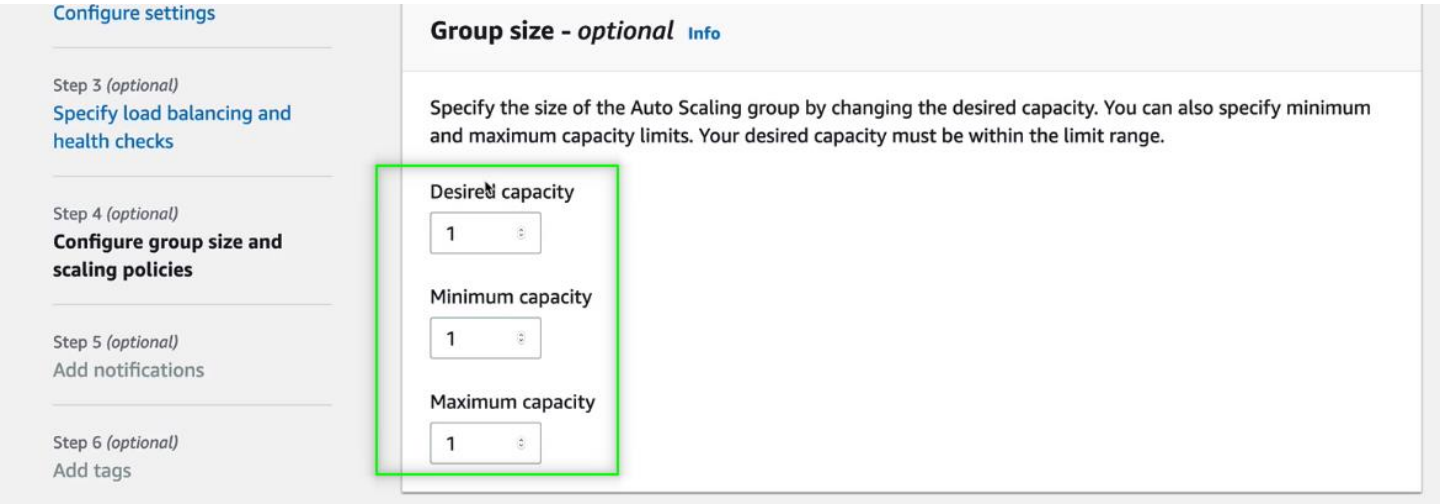
Wordpress template now up:

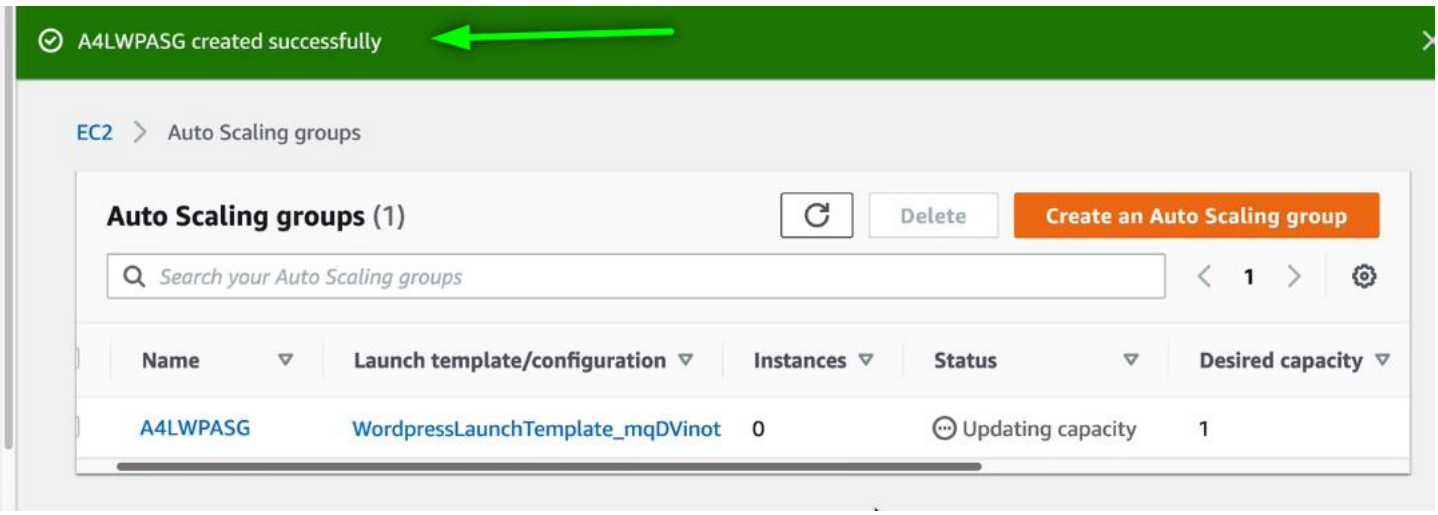


3. PART 2 — Autoscaling Group

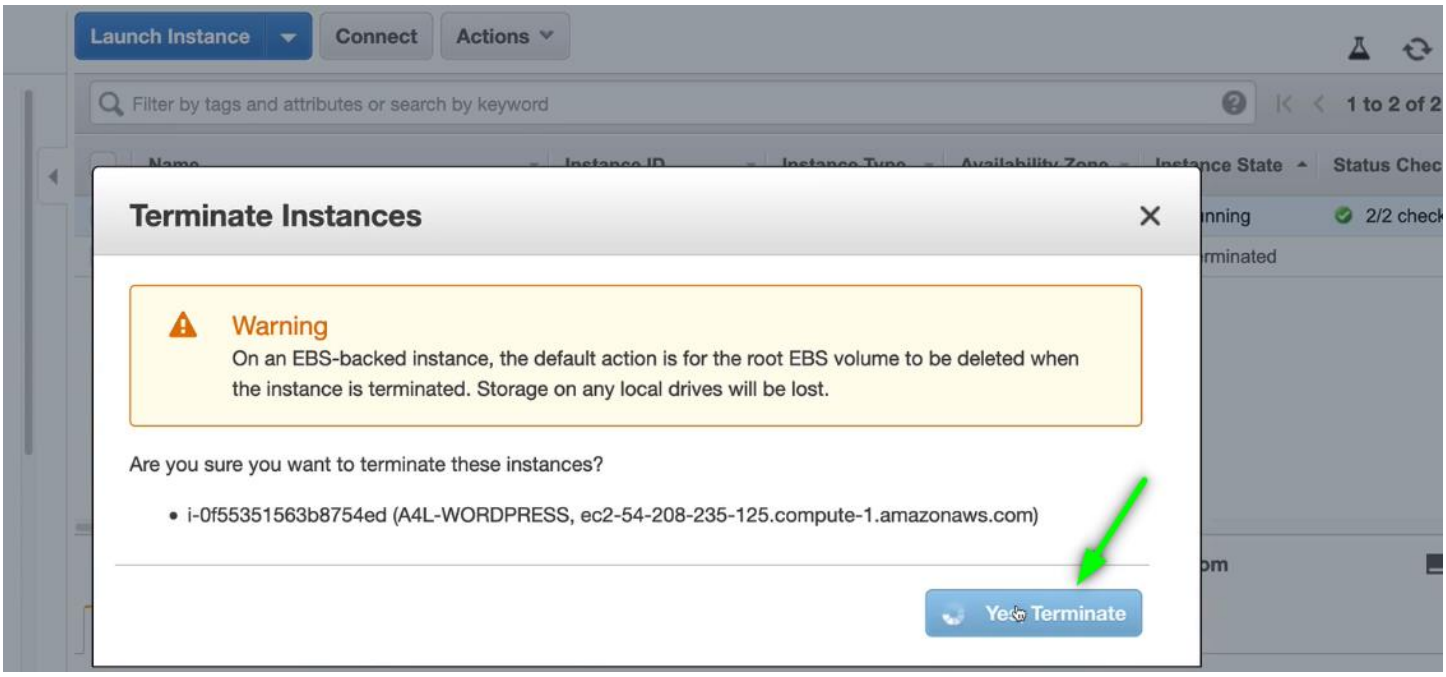
In Part 2 of this [DEMO] series you will learn how to create an auto scaling group to automatically provision instances based on the launch template. The ASG controls the WHEN and WHERE (when to provision things and in what subnets), the LT controls the WHAT (what configuration). You will experience manual scaling, auto scaling and self-healing features of ASGs in this DEMO.

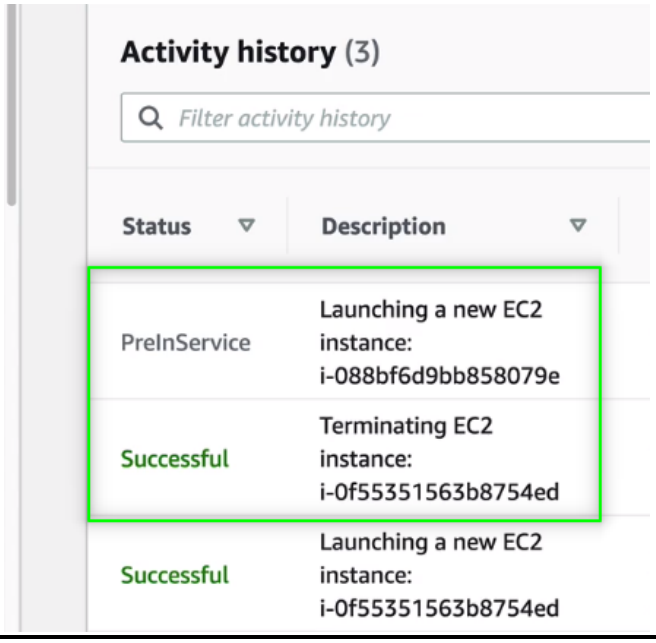
Create and Configure ASG to use 1:1:1 group sizes





Terminate current running EC2 instance of wordpress, to have the ASG kick-in:



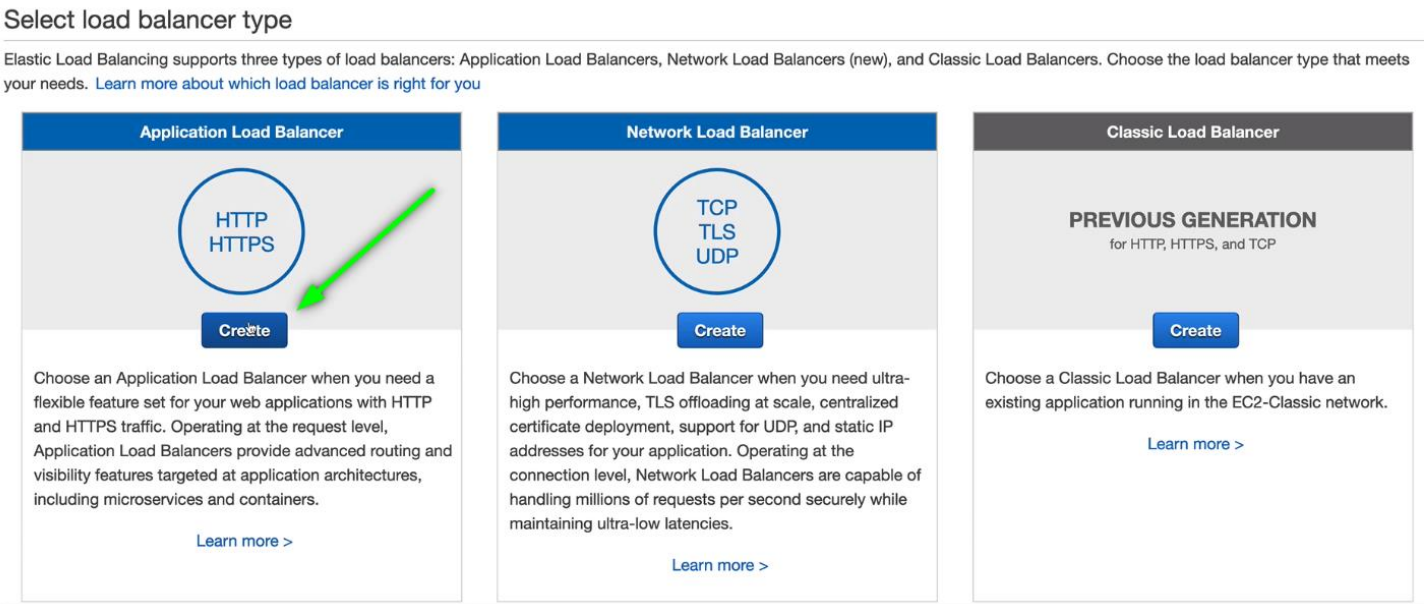


4. PART 3 — Load Balancer

In PART 3 of this [DEMO] series you will create an application load balancer and integrate it with the auto scaling group to automatically provision, terminate and scale instances all while automatically adding these to the Application Load Balancer.

Use previous stacks from Part 1 & 2.

Create ALB and configure:



1. Configure Load Balancer | 2. Configure Security Settings | 3. Configure Security Groups | 4. Configure Routing | 5. Regist

Step 1: Configure Load Balancer

Name A4L-Wordpress-LB

Scheme internet-facing / internal

IP address type ✓ipv4 / dualstack

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
HTTP	80

Add listener

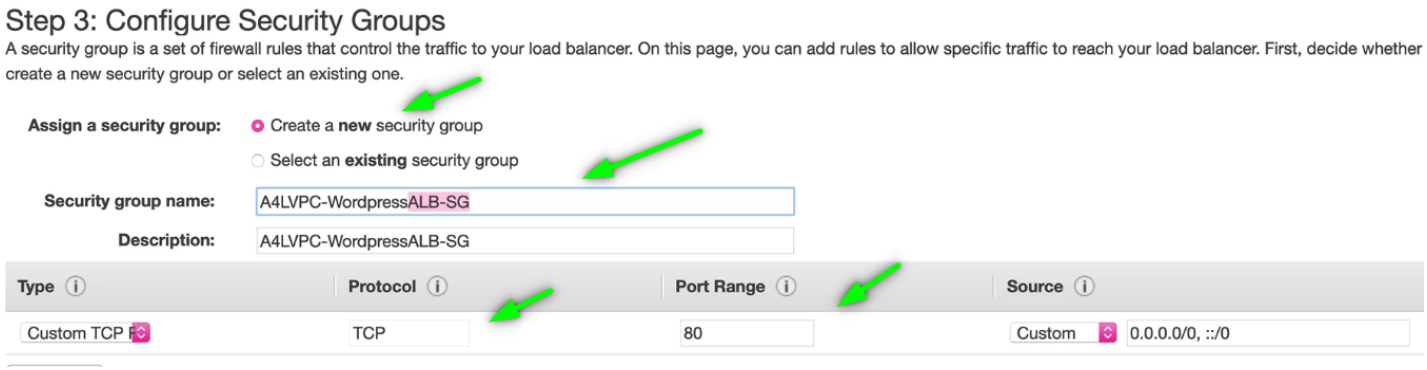
Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Ava
You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

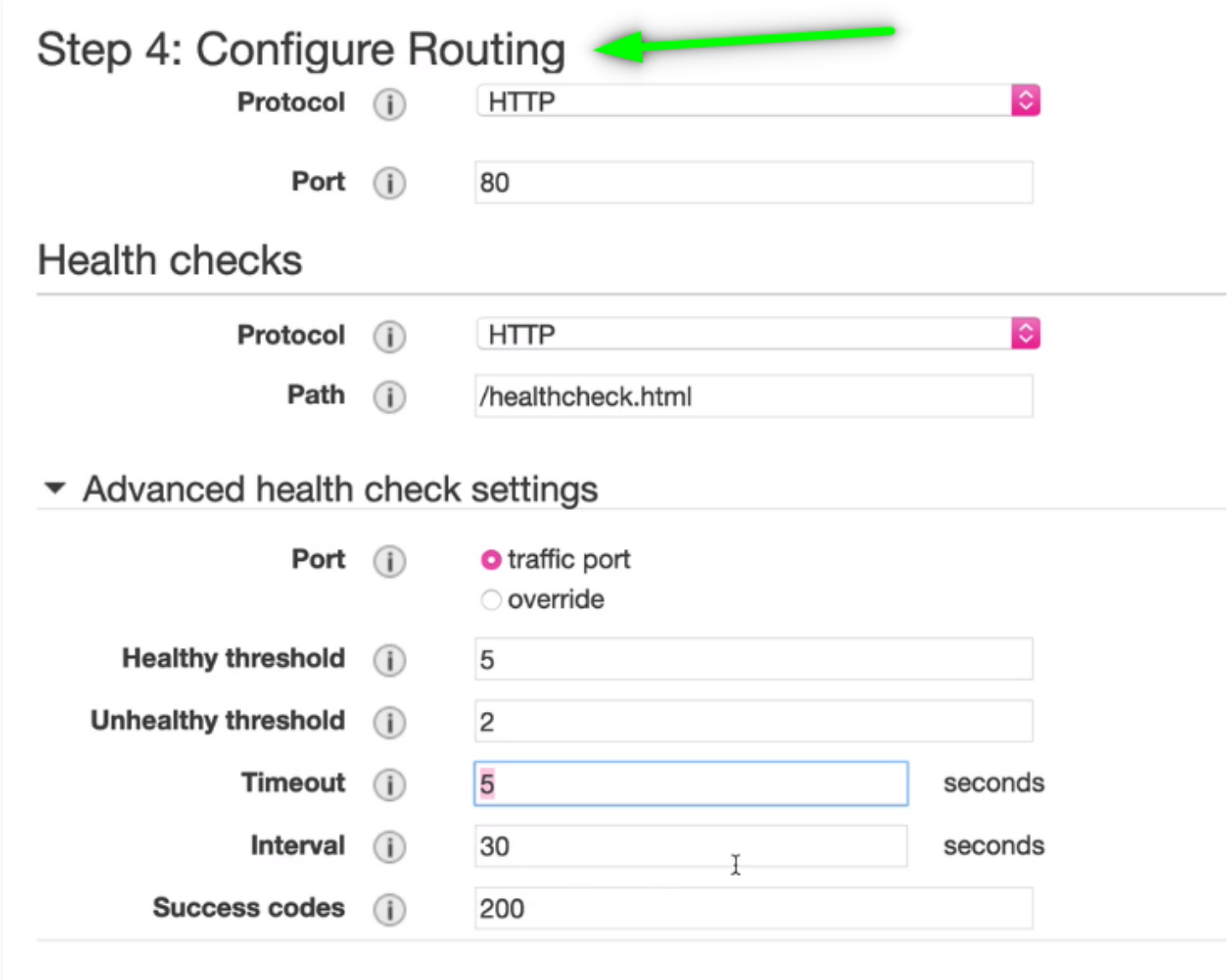
VPC vpc-06df09599aabb8c3c (10.16.0.0/16) | a4l-vpc1

Availability Zones

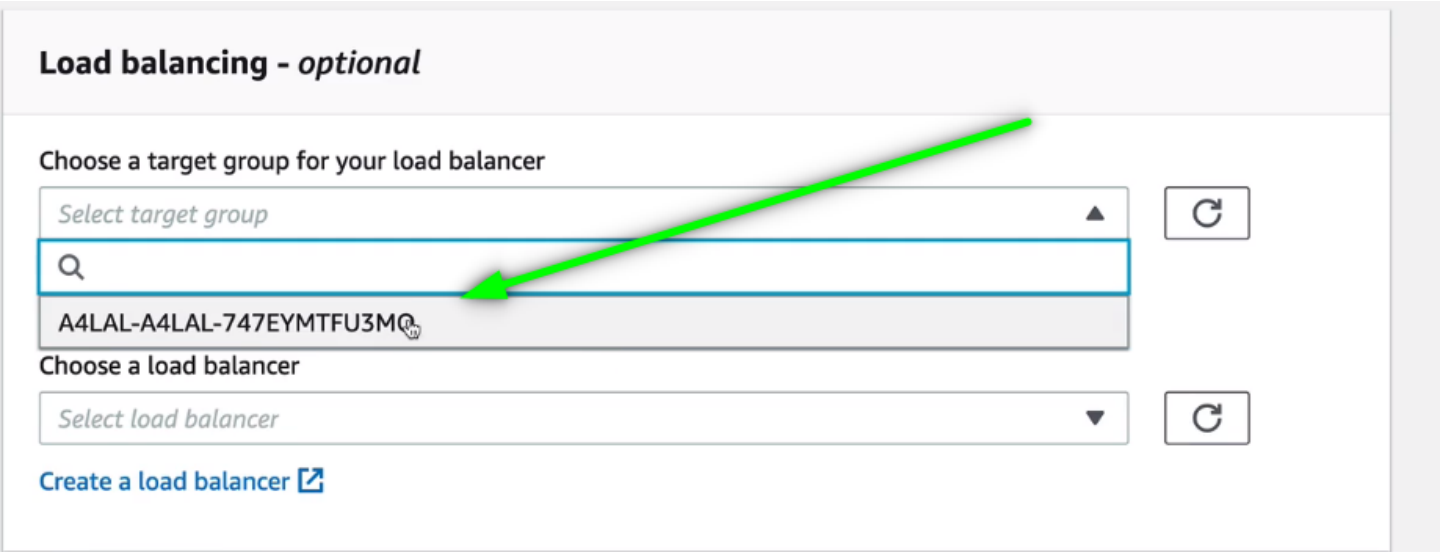
- ☑ us-east-1a — subnet-0a97015b4c2508379 (sn-web-A) — IPv4 address: Assigned by AWS
- ☑ us-east-1b — subnet-001638d919b404d43 (sn-web-B) — IPv4 address: Assigned by AWS
- ☐ us-east-1c — Select a subnet

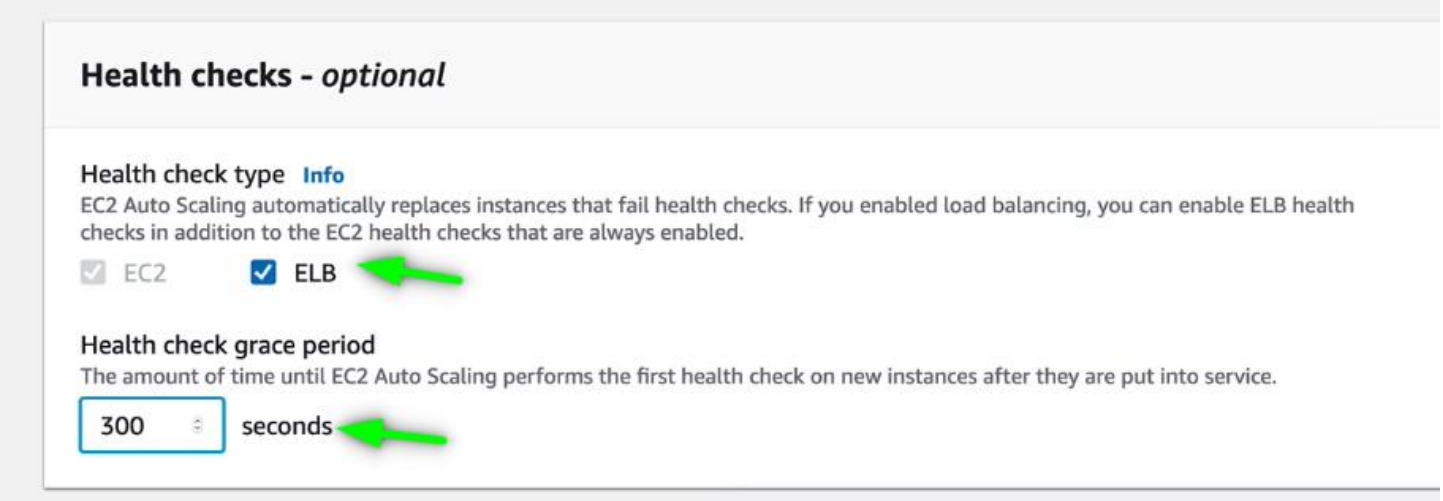


Setup health checks on ALB

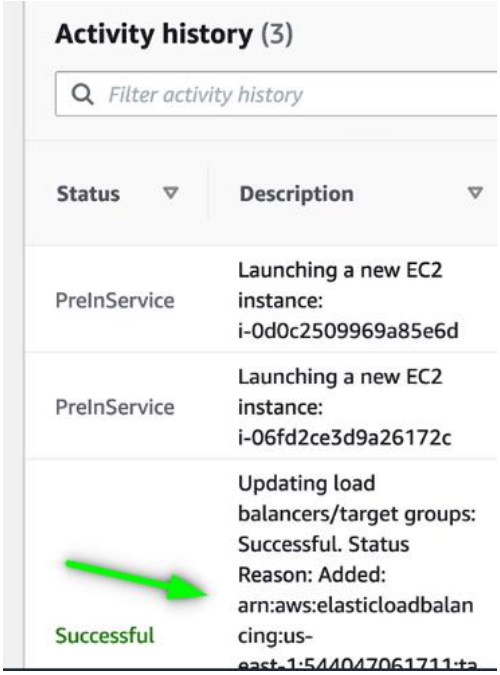


Go to ASG and enable load balancing, map to new ALB:





Check that ASG is utilizing ALB

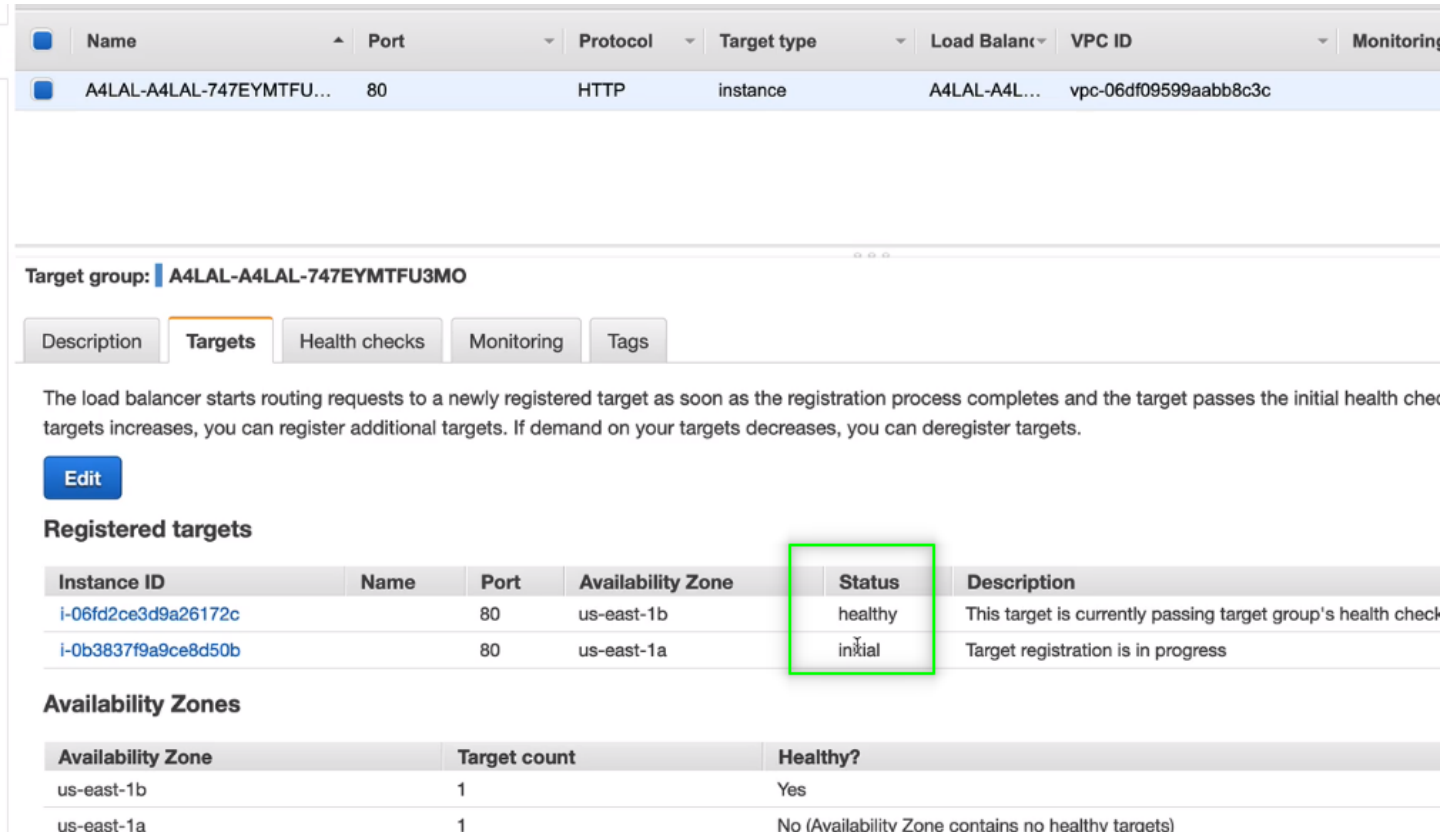


Activity history (3)

Filter activity history

Status	Description
PreInService	Launching a new EC2 instance: i-0d0c2509969a85e6d
PreInService	Launching a new EC2 instance: i-06fd2ce3d9a26172c
Successful	Updating load balancers/target groups: Successful. Status Reason: Added: arn:aws:elasticloadbalancing:us-east-1:544047061711:ta

Health checks:



Name	Port	Protocol	Target type	Load Balanc	VPC ID	Monitoring
A4LAL-A4LAL-747EYMTFU...	80	HTTP	instance	A4LAL-A4L...	vpc-06df09599aabb8c3c	

Target group: A4LAL-A4LAL-747EYMTFU3MO

Description | Targets | Health checks | Monitoring | Tags

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health chec
targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

Edit

Registered targets

Instance ID	Name	Port	Availability Zone	Status	Description
i-06fd2ce3d9a26172c		80	us-east-1b	healthy	This target is currently passing target group's health check
i-0b3837f9a9ce8d50b		80	us-east-1a	initial	Target registration is in progress

Availability Zones

Availability Zone	Target count	Healthy?
us-east-1b	1	Yes
us-east-1a	1	No (Availability Zone contains no healthy targets)