

Load Balanced Architecture with Advanced Request Routing

This lab guide creates a simple website, hosted on EC2, behind an Application Load Balancer. Host-based and path-based routing rules will then be configured to route based on information in the host header or URL path.

Host-based routing allows you to route to multiple domains on a single load balancer by routing to a different set of EC2 instances or containers based on information in the host header.

Path-based routing is also referred to as URL-based routing. The Application load Balancer will forward the requests to the specific targets based on the rules configured on the load balancer. We will be doing this first.

Requirements (Prerequisites)

- *AWS Free Tier Account*
- *Basic Exposure to the AWS Console and completion of prior days learning*
- *Follow the steps in Section 8 of the course to register your own domain name in Route 53*

Resources

Download the "[advanced-request-routing-code.zip](#)" file.

Exercise Overview

Exercise 1 – Create the Red and Blue EC2 instances

Exercise 2 – Enable Path-based Routing

Exercise 3 – Enable Host-based Routing

Exercise 4 – Clean up your resources

Exercise 1 - Create the Red and Blue EC2 instances

Task 1 - Create the S3 Bucket and Upload the code

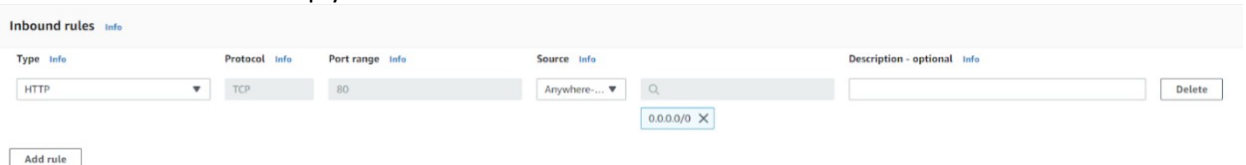
The first step is to upload the code we will use to create the websites into an S3 bucket.

1. Go to the S3 console and click 'Create Bucket'.
2. We will call out bucket 'arr-bucket-123456' with the numbers at the end being random to make the bucket globally unique. Take a note of the bucket name as we will be referring to it again shortly.
3. Scroll down and click 'Create Bucket'.
4. Locate the files you have downloaded as part of the course and select all files except the user data files and bucket-permissions file and upload them to the bucket.

Task 2 - Create your Security Group

The second step is to pre-emptively create the Security Group for your EC2 instances.

1. Go to the EC2 console, scroll down to the 'Security Group' column under 'Network Security'. Click 'Create Security Group'. We will call it 'WebsiteSG'. Populate the description field with 'WebsiteSG' also.
2. Under inbound rules - simply add one rule:



Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Anywhere...	

Add rule

3. This is an HTTP rule, with an 'Anywhere IPv4' source. Create the Security Group.

Task 3 - Create an IAM Role for EC2 and Launch the Red Instance

1. Head over to the IAM console and select 'Policies'.
2. Click 'Create policy' and move to the JSON section. Copy and paste the code from the bucket-permissions.json file into the code window, replacing 'YOUR-BUCKET-ARN' with the ARN of the bucket you have just created.
3. Click review policy, and Create policy, call it 'S3-ARR-Policy'.
4. Click 'Roles' and select 'Create role'.
5. Under common use cases, select 'EC2' and click next.
6. For permissions policies attach the 'S3-ARR-Policy'.
7. Click next and call it 'S3-ARR-Role'.

We will now launch the 'Red' EC2 instance and check to see if it has successfully retrieved the code from S3.

1. Head over to the EC2 console and select 'Launch Instance'.
2. Call your first instance 'Red'.
3. Select the AMI, and under 'Instance type', select 't2.micro'.
4. For 'Key pair', leave the default setting 'Proceed without a key pair'.
5. Under 'Network settings' select the security group you created earlier and choose the

subnet us-east-1a.

6. Expand 'Advanced Details' and under 'IAM instance profile' select the 'S3-ARR-Role' role created before.
7. Next, copy the user data from the 'user-data-red' file into the 'User data' field. It should look like the image below. You will also need to edit the name 'YOUR-BUCKET-NAME' in the user data to the name of your S3 bucket.

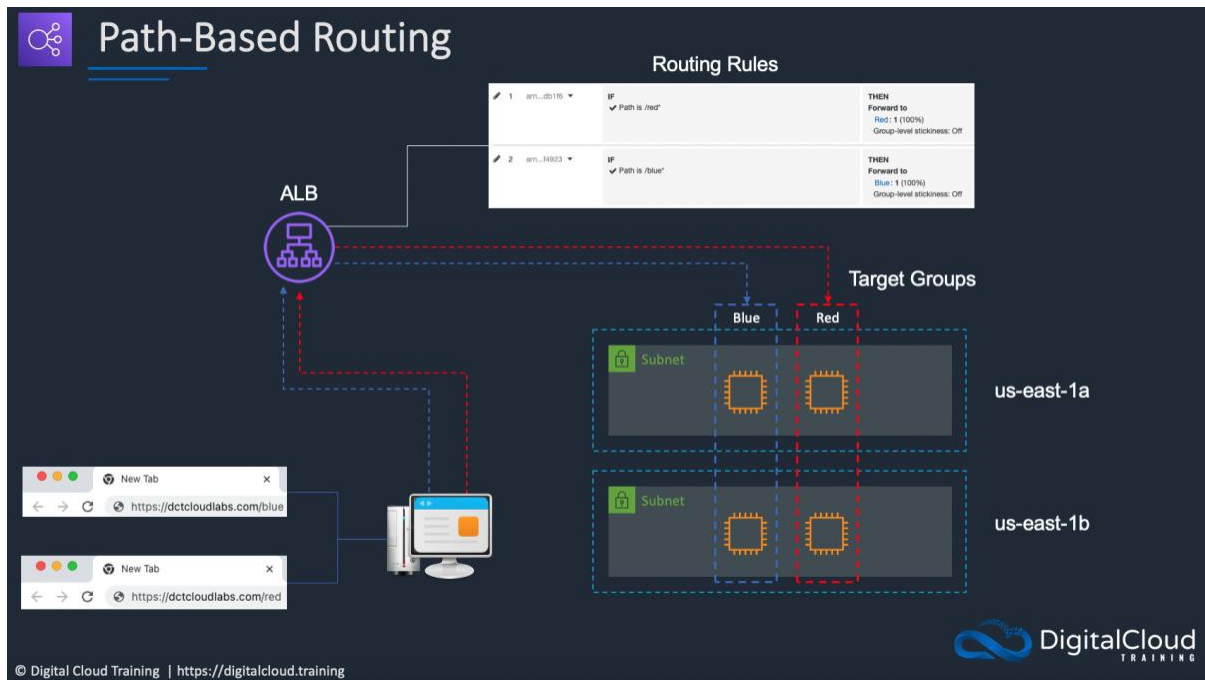
```
#!/bin/bash
yum update -y
yum install -y httpd
# below line starts httpd daemon
systemctl start httpd
# below line sets http daemon to start automatically at boot time
systemctl enable httpd
mkdir /var/www/html/red
# populate /red directory of server with web page
cd /var/www/html/red
aws s3 cp s3://YOUR-BUCKET-NAME/hw-red.css ./
aws s3 cp s3://YOUR-BUCKET-NAME/hw-red-py.css ./
aws s3 cp s3://YOUR-BUCKET-NAME/python.png ./
aws s3 cp s3://YOUR-BUCKET-NAME/apache.svg ./
aws s3 cp s3://YOUR-BUCKET-NAME/red-index.html ./index.html
# populate root of server with web page (replaces apache default page)
cd /var/www/html
aws s3 cp s3://YOUR-BUCKET-NAME/hw-red.css ./
aws s3 cp s3://YOUR-BUCKET-NAME/hw-red-py.css ./
aws s3 cp s3://YOUR-BUCKET-NAME/python.png ./
aws s3 cp s3://YOUR-BUCKET-NAME/apache.svg ./
aws s3 cp s3://YOUR-BUCKET-NAME/red-root-index.html ./index.html
# optional restart of httpd daemon
systemctl restart httpd
```

8. Create the instance.
9. Follow the exact same steps to create the Blue instance with the blue user data (user-data-blue) and select subnet us-east-1b.

You should then be able to get the public IP address from either instance to view the web pages created on each EC2 instance. You will need to add /blue and /red to the appropriate instance to return the custom web page.

Exercise 2 – Enable Path-based Routing

With path-based routing we will enter a path to our URL and the load balancer will route the request to the appropriate target group based on the rules we create.



Task 1 – Create your target groups

The first step is to set up the target groups; you need at least 2 target groups to configure Path-based routing.

1. Click on Target Groups under Load balancing.
2. Click 'Create target group.'
3. Set up 2 target groups, one is called 'Red' which will contain the red targets, and the other is called 'Blue' and will contain the blue targets.
4. Leave all the defaults, except changing the target group name to 'Red' / 'Blue' and change the health check to:
 - a. For Red: /red/index.html
 - b. For Blue: /blue/index.html
5. Register the correct instance i.e., Red and be sure to click 'include as pending below'.

Target groups (2) Info								
<input type="text" value="Search or filter target groups"/>								
<input type="checkbox"/>	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID	
<input type="checkbox"/>	Blue	arn:aws:elasticloadbalancin...	80	HTTP	Instance	None associated	vpc-0e71f18342474263f	
<input type="checkbox"/>	Red	arn:aws:elasticloadbalancin...	80	HTTP	Instance	None associated	vpc-0e71f18342474263f	

Task 2 – Create your Application Load Balancer

Next step is to create the Application Load Balancer.

1. On the left-hand side of the EC2 console you will find the link for Load Balancers.
2. Click 'Create Load Balancer' and choose the 'Application Load Balancer'.

3. Call the load balancer 'LabLoadBalancer' and leave the Scheme as Internet Facing.
4. Select both of us-east-1a and us-east-1b for the subnet mappings. This will allow routing of traffic across the instances in different AZs.
5. Choose the same Security Group as earlier (WebsiteSG).
6. Choose the listener and routing as per the image below; we will configure the routing rules later.

Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

Remove

Protocol

HTTP ▼

Port

80

1-65535

Default action

[Info](#)

Forward to

Blue

Target type: Instance, IPv4

HTTP ▼

↺

[Create target group](#)

Add listener

7. Click Create load balancer and wait a few minutes for it to turn from 'Provisioning' to 'Active'.
8. Once active, under 'Listeners and rules', select the checkbox for your first listener. Then, under the 'Manage Rules' menu, click 'Add rule'.

Listeners and rules

Network mapping

Security

Monitoring

Integrations

Attributes

Tags

Listeners and rules (1/1) [Info](#)

↺

Manage rules ▲

Manage listener ▼

Add listener

↺

Add rule

Edit rules

Reprioritize rules

< 1 > ⚙

Filter listeners by property or value

<input checked="" type="checkbox"/>	Protocol:Port ▼	Default action ▼	Rules ▼	ARN ▼	Security policy ▼	Default SSL/TLS certificate ▼	Tags ▼
<input checked="" type="checkbox"/>	HTTP:80	<div>Forward to target group</div> <ul style="list-style-type: none"> Blue ↗: 1 (100%) Group-level stickiness: Off 	1 rule	ARN	Not applicable	Not applicable	0 tags

9. Configure as follows:

- **Name and tags** – ignore and click next.
- **Conditions** – click 'Add condition' and select 'Path' from the dropdown menu and enter `"/red*"` for the path.
- **Actions** – Forward to the 'Red' target group.
- **Priority** – set to 1.

Actions

Action types

☒ Forward to target groups

☐ Redirect to URL

☐ Return fixed response

Forward to target group [Info](#)

Choose a target group and specify routing weight or [Create target group](#)

Red

Target type: Instance, IPv4

HTTP ▼

↺

Weight

1

Percent

100%

0-999

Add target group

You can add up to 4 more target groups.

☐ Turn on group-level stickiness [Info](#)

If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.

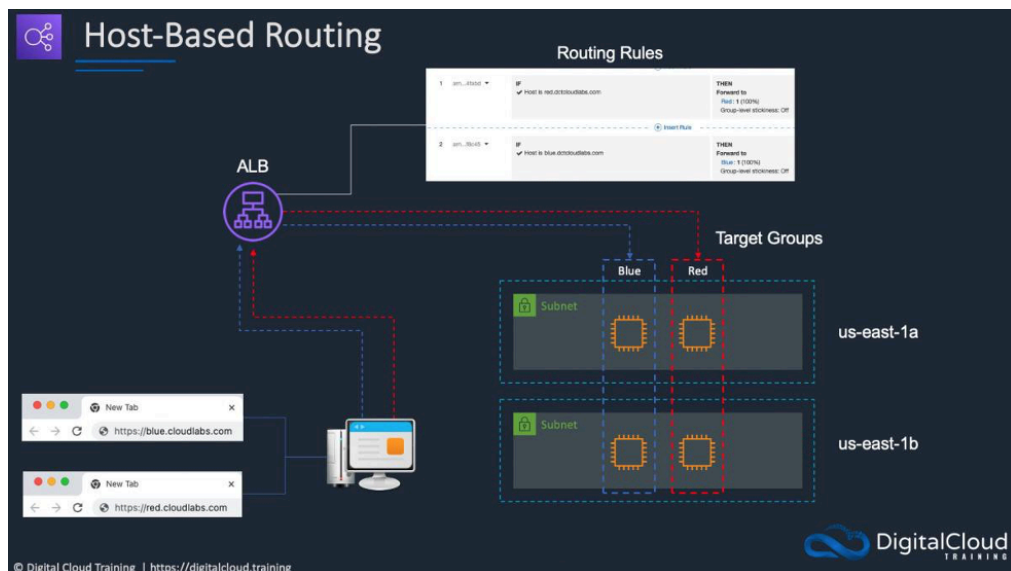
10. Repeat the above creating a rule for the blue target group. The path should be “/blue*” and priority can be 2. The listener rules should now look like this:

Listener rules (3) Info				
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.				
<input type="text" value="Filter rules by property or value"/>				
<input type="checkbox"/>	Name tag	Priority	Conditions (If)	Actions (Then) ✕
<input type="checkbox"/>	-	1	Path Pattern is /red*	Forward to target group <ul style="list-style-type: none"> Red: 1 (100%) Group-level stickiness: Off
<input type="checkbox"/>	-	2	Path Pattern is /blue*	Forward to target group <ul style="list-style-type: none"> Blue: 1 (100%) Group-level stickiness: Off
<input type="checkbox"/>	Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> Blue: 1 (100%) Group-level stickiness: Off

11. We can then test the load balancer’s path-based routing by copying the DNS name from the Application Load Balancer and append either /blue or /red on the URL and see what happens. You should see the different colored custom web pages we added to our instances.

Exercise 3 – Enable Host-based Routing

With host-based routing we will enter a subdomain to the domain name and the load balancer will route the request to the appropriate target group based on the rules we create.



Task 1 – Edit host-based forwarding rules

The first step is to set up the forwarding rules. You need at least 2 target groups to configure host-based routing.

1. Navigate back to the listener.
2. Click on ‘Edit rules’ under the ‘Manage rules’ menu.
3. Select and delete both path based rules.

4. Add a new rule to the listener and configure as follows:
 - **Conditions** – add a host header condition and for your value, enter your domain name with “red” as the subdomain, e.g. “red.yourdomainname.com” (swap the domain name for your registered domain).
 - **Actions** – Forward to the ‘Red’ target group.
 - **Priority** – set to 1.
5. Repeat the above creating a rule for the blue target group. The host header value should be “blue.yourdomainname.com” (swap the domain name for your registered domain) and priority can be set to 2.
6. The rules should look something like this:

Listener rules (3) Info				
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.				
<input type="text" value="Filter rules by property or value"/>				
<input type="checkbox"/>	Name tag	Priority	Conditions (If)	Actions (Then) [?]
<input type="checkbox"/>	-	1	HTTP Host Header is red.nealawslabs.link	Forward to target group <ul style="list-style-type: none"> Red: 1 (100%) Group-level stickiness: Off
<input type="checkbox"/>	-	2	HTTP Host Header is blue.nealawslabs.link	Forward to target group <ul style="list-style-type: none"> Blue: 1 (100%) Group-level stickiness: Off
<input type="checkbox"/>	Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> Blue: 1 (100%) Group-level stickiness: Off

Task 2 – Configure Records in Route 53

In this task we need to create the relevant subdomain DNS records in Amazon Route 53 and configure the load balancer as the target.

1. Open Route 53 dashboard from the management console, find your public domain name under hosted zones. Click on it and select ‘Create Record.’
2. Enter various details for this record:
 - **subdomain:** blue or red.
 - **Record type:** Select A type here.
 - **Value/Route traffic to:**
 - Select 'Alias to Application and Classic Load Balancer' Select region N.Virginia.
 - Select the target load balancer we made earlier.

Quick create record

[Switch to wizard](#)

▼ Record 1

[Delete](#)

Record name [Info](#)
 .nealawslabs.link

Keep blank to create a record for the root domain.

☒ Alias

Route traffic to [Info](#)

Alias hosted zone ID: Z35SXD0TRQ7X7K

Routing policy [Info](#)

Evaluate target health

☒ Yes

[Add another record](#)

- Make sure you repeat the above steps to ensure you have one DNS record for each of the blue and red subdomains.
- Create an additional record for the apex/naked domain name that points to the ALB.
- The DNS records should look something like this:

Record name ▼	Type ▼	Routing policy ▼	Differe... ▼	Alias ▼	Value/Route traffic to
nealawslabs.link	A	Simple	-	Yes	dualstack.labloadbalancer-832710974.us-east-1.elb.amazonaws.com
nealawslabs.link	NS	Simple	-	No	ns-1289.awsdns-33.org. ns-694.awsdns-22.net. ns-320.awsdns-40.com. ns-1748.awsdns-26.co.uk.
nealawslabs.link	SOA	Simple	-	No	ns-1289.awsdns-33.org. awsdns-hostmaster.amazon.com
blue.nealawslabs.link	A	Simple	-	Yes	dualstack.labloadbalancer-832710974.us-east-1.elb.amazonaws.com
red.nealawslabs.link	A	Simple	-	Yes	dualstack.labloadbalancer-832710974.us-east-1.elb.amazonaws.com

- To check if everything is working as expected, open a web browser, and use the following format for the URL:

[red.yourdomainname.com](#)

[blue.yourdomainname.com](#)

You should see the webpages of the relevant EC2 instance. You can also append the paths to the application page.

Exercise 4 – Clean up your resources

Task 1 – Delete your resources

You can now delete the resources you created:

- Application Load Balancer (chargeable).
- Target groups (not chargeable).
- Security Group (not chargeable).
- Amazon EC2 Instances (chargeable).
- Amazon S3 bucket (chargeable).
- Amazon Route 53 Alias records (not chargeable).