



# Route 53 Resolver Endpoints

**Amazon Route 53 Resolver** is a scalable and highly available DNS (Domain Name System) service in Amazon Web Services (AWS). It is part of the Route 53 service, which is AWS's domain name system (DNS) web service. The Route 53 Resolver acts as a DNS resolver for Amazon VPCs (Virtual Private Clouds), allowing resources within your VPC to resolve domain names to IP addresses and vice versa.

## Key Features of Route 53 Resolver

### 1. DNS Resolution:

- The Route 53 Resolver provides DNS resolution for both forward (resolving domain names to IP addresses) and reverse (resolving IP addresses to domain names) lookups within your VPC.

### 2. Hybrid DNS:

- Route 53 Resolver supports hybrid cloud scenarios by enabling DNS resolution between on-premises networks and AWS VPCs. This is done through the use of inbound and outbound endpoints.
- **Inbound Endpoints:** Allow on-premises systems to resolve domain names for AWS resources.
- **Outbound Endpoints:** Allow AWS resources to resolve domain names hosted in on-premises networks or another cloud provider.

### 3. Private DNS for VPCs:

- Route 53 Resolver allows you to use private DNS for resources within your VPC, enabling the use of custom domain names within your private network.

### 4. Resolver Rules:

- You can create resolver rules to define how DNS queries should be handled. For example, you can specify that queries for certain domain names should be forwarded to specific IP addresses or DNS servers.

### 5. Security:

- Route 53 Resolver integrates with AWS Identity and Access Management (IAM) and AWS CloudTrail, allowing you to manage access and monitor DNS queries.

### 6. Scalability and High Availability:

- Route 53 Resolver is designed to be highly available and can scale to handle DNS queries efficiently across multiple VPCs and regions.

## Use Cases

- **Hybrid Cloud DNS Resolution:** Connecting on-premises environments with AWS, enabling seamless DNS resolution between the two.
- **Private DNS Management:** Providing DNS resolution for internal domain names within a VPC, without exposing DNS queries to the public internet.
- **Custom DNS Routing:** Directing specific DNS queries to custom DNS servers or endpoints, allowing for customized DNS behavior.

## How It Works

1. **DNS Queries from VPC:** When an AWS resource within a VPC needs to resolve a domain name, it sends a DNS query to the Route 53 Resolver.
2. **Resolver Rules:** The resolver uses the rules you have defined to determine how to handle the query, whether to resolve it within the VPC, forward it to another DNS server, or respond with a custom IP address.
3. **Inbound and Outbound Endpoints:** Depending on the configuration, queries may be routed to on-premises DNS servers (via outbound endpoints) or received from on-premises systems (via inbound endpoints).

In summary, Route 53 Resolver is an essential component for managing DNS resolution within AWS environments, supporting both internal and hybrid cloud scenarios with flexibility and security.



## What are we doing in this Lab?

In this lab, you're setting up a hybrid DNS resolution system between two AWS regions (Ohio and Frankfurt) using Amazon Route 53 Resolver. The goal is to enable seamless communication and DNS name resolution between resources in different regions, simulating an on-premises and cloud hybrid environment.

### Summary of Steps:

1. **Setup CloudFormation Stacks:**
  - Deploy templates in the Ohio and Frankfurt regions to create necessary infrastructure, including VPCs, EC2 instances, and a DNS zone.
2. **Establish VPC Peering:**
  - Create a VPC peering connection between the VPCs in the Ohio and Frankfurt regions to allow network communication.
3. **DNS Resolution Setup:**
  - Create **Route 53 Resolver inbound endpoints** in the Ohio region to allow DNS queries from Frankfurt.
  - Update DNS server configurations in the Frankfurt region to use these endpoints.

- Configure **Route 53 Resolver outbound endpoints** in the Ohio region to forward DNS queries for a custom domain.

#### 4. Verification:

- Verify that instances in both regions can communicate and resolve DNS queries correctly.

#### 5. Cleanup:

- Delete the created infrastructure and configurations to avoid ongoing costs.

#### End Goal:

The end goal is to demonstrate how to set up cross-region, hybrid DNS resolution between AWS regions. This setup allows instances in different regions to resolve each other's DNS names securely and privately as if they were within the same network. This is useful for hybrid cloud scenarios where seamless and secure communication is required between on-premises systems and AWS cloud resources.

## 😊 To begin with the Lab:

### 😊 Step 1: Creating Environment

1. In this lab you need to download these 4 files shown below from GitHub.
2. Now you can see that two files are in YAML format because they are going to be used as templates in Cloud formation.
3. So, in your AWS Console go to cloud formation in the Ohio Region, then upload your template and create your stack. The template you have to use in the Ohio Region is **AWS\_DNS**.

Also, it would help if you had a domain name from any provider with you. Then open the **AWS\_DNS** file and update the name of your domain here. Then you have to upload the template.

4. Similarly, you have to go to the Frankfurt region and go to cloud formation then you need to upload the template **OnPrem\_AWS** and create your stack.

 zonefile	03-08-2024 16:21	Text Document	1 KB
 OnPrem_AWS	03-08-2024 16:21	Yaml Source File	11 KB
 Commands	03-08-2024 16:21	Text Document	1 KB
 AWS_DNS	03-08-2024 16:28	Yaml Source File	7 KB

5. Below you can see that our template has been created in the Ohio region and here you can see the resources that have been created. So, here this template has created a VPC, an EC2 instance, and a DNS zone in Route 53.

CloudFormation > Stacks > awsDNS

**awsDNS**

Stacks (1)

awsDNS  
2024-08-03 16:30:55 UTC+0530  
CREATE\_COMPLETE

Resources (16)

Logical ID	Physical ID	Type	Status	Module
AWSecurityGroup	sg-0f0aa4386447692596	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-
DefaultInstanceSecurityGroupSelfReferenceRule	sgr-0e98512eefad4863f	AWS::EC2::SecurityGroupIngress	CREATE_COMPLETE	-
DNSCDCCOM	2056580234WHJLWDKOT	AWS::Route53::HostedZone	CREATE_COMPLETE	-
DNSCDCCOMWEB	cloudservicesdemo.in	AWS::Route53::RecordSet	CREATE_COMPLETE	-
EC2A	i-0fbfe3eb3e37d25d6	AWS::EC2::Instance	CREATE_COMPLETE	-
EC2InstanceProfile	awsDNS-EC2InstanceProfile-0SJ0VWZc5Sj	AWS::IAM::InstanceProfile	CREATE_COMPLETE	-
EC2Role	awsDNS-EC2Role-mXgzeZJPdtTa	AWS::IAM::Role	CREATE_COMPLETE	-

- Below you can see that our template in the Frankfurt region has also been created. Here we have created a VPC, 3 EC2 instances.

CloudFormation > Stacks > onPrem

**onPrem**

Stacks (1)

onPrem  
2024-08-03 16:36:28 UTC+0530  
CREATE\_COMPLETE

Resources (17)

Logical ID	Physical ID	Type	Status	Module
APP	i-0d9377ba12a7ba7c9	AWS::EC2::Instance	CREATE_COMPLETE	-
DefaultInstanceSecurityGroupSelfReferenceRule	sgr-0e939034ae8e942b	AWS::EC2::SecurityGroupIngress	CREATE_COMPLETE	-
DNSA	i-065b3825b8fdbb49c	AWS::EC2::Instance	CREATE_COMPLETE	-
DNSB	i-03f68339cea839d44	AWS::EC2::Instance	CREATE_COMPLETE	-
EC2InstanceProfile	onPrem-EC2InstanceProfile-AVHxGw41fTl	AWS::IAM::InstanceProfile	CREATE_COMPLETE	-
EC2Role	onPrem-EC2Role-q1UgefWv2ak1	AWS::IAM::Role	CREATE_COMPLETE	-
ONPREMSecurityGroup	sg-0ee82e80fbb8408d	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-

- Now we are going to create a peering connection from the Frankfurt region to the Ohio region. For that in the search bar search for Peering Connection or go to VPC then navigate to there.
- Here you need to click on Create. Then you need to choose the Requester which is on-prem and scroll down.

# Create peering connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them privately.

Info

## Peering connection settings

### Name - optional

Create a tag with a key of 'Name' and a value that you specify.

my-pc-01

### Select a local VPC to peer with

#### VPC ID (Requester)

vpc-0c6f5754088a5939c (CDD-onprem)

#### VPC CIDRs for vpc-0c6f5754088a5939c (CDD-onprem)

CIDR	Status	Status reason
192.168.10.0/24	<input checked="" type="checkbox"/> Associated	-

9. After that you need to choose My account, then in the region choose Another region and mention the VPC ID of the Ohio region VPC.

10. Then click on Create peering connection.

### Select another VPC to peer with

#### Account

- My account
- Another account

#### Region

- This Region (eu-central-1)
- Another Region

US East (Ohio) (us-east-2)

#### VPC ID (Acceptor)

vpc-0df226de6ad01e396

## Tags

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.

No tags associated with the resource.

Add new tag

You can add 50 more tags.

Cancel

Create peering connection

11. Then a request will be sent to the Ohio region VPC to accept the Peering connection request.

A screenshot of the AWS VPC Peering connections details page. At the top, a green banner says: "A VPC peering connection pcx-036425e83713d1191 has been requested. Remember to change your region to us-east-2 to accept the peering connection." Below the banner, the peering connection ID is shown as "pcx-036425e83713d1191". The "Actions" dropdown menu is open, showing options like "Accept request", "Reject request", "Edit DNS settings", "Manage tags", and "Delete peering connection". The "Accept request" option is highlighted with a red box.

Details		Info	
Requester owner ID 878893308172	Acceptor owner ID 878893308172	VPC Peering connection ARN arn:aws:ec2:eu-central-1:878893308172:vpc-peering-connection/pcx-036425e83713d1191	
Peering connection ID pcx-036425e83713d1191	Requester VPC vpc-0c6f5754088a5939c / CDD-onprem	Acceptor VPC vpc-0df226de6ad01e396	
Status Initiating Request to 878893308172	Requester CIDRs 192.168.10.0/24	Acceptor CIDRs -	
Expiration time Saturday, August 10, 2024 at 17:01:41 GMT+5:30	Requester Region Frankfurt (eu-central-1)	Acceptor Region Ohio (us-east-2)	

12. Now go to Peering connection in the Ohio region, here you will see the status as Pending Acceptance. Then choose it and click on action then click on accept request.

A screenshot of the AWS VPC Peering connections list page in the Ohio region. It shows one peering connection entry: "pcx-036425e83713d1191" with status "Pending acceptance". The "Actions" dropdown menu is open, showing options like "View details", "Accept request" (which is highlighted with a red box), "Reject request", "Edit DNS settings", "Manage tags", and "Delete peering connection".

13. Then you need to go to route tables in both of the regions and add a route for both where the destination is CIDR blocks, and the target is Peering connection.

A screenshot of the AWS Route Tables details page for route table "rtb-0e2483161fc31085f". The "Routes" tab is selected, showing two routes:

Destination	Target	Status	Propagated
10.16.0.0/16	local	Active	No
192.168.10.0/24	pcx-036425e83713d1191	Active	No

**Details** [Info](#)

Route table ID <a href="#">rtb-02a738a9ad482ac90</a>	Main <input checked="" type="checkbox"/> No	Explicit subnet associations 2 subnets	Edge associations -
VPC <a href="#">vpc-0c6f5754088a5939c   CDD-onprem</a>	Owner ID <a href="#">878893308172</a>		

[Routes](#) [Subnet associations](#) [Edge associations](#) [Route propagation](#) [Tags](#)

**Routes (3)**

Filter routes		Both	Edit routes
Destination	Target	Status	Propagated
<a href="#">pl-gea54007</a>	<a href="#">vpce-0f728720751d8c26b</a>	<span style="color: green;">Active</span>	No
<a href="#">10.16.0.0/16</a>	<a href="#">ppx-036425e83713d1191</a>	<span style="color: green;">Active</span>	No
<a href="#">192.168.10.0/24</a>	local	<span style="color: green;">Active</span>	No

14. Now we are going to connect to our ON PREM APP instance in the Frankfurt region and try to ping the EC2 instance in the Ohio region. So, choose your instance and click on connect. Then choose Session Manager to connect with your instance.

**Instances (1/3) [Info](#)**

[Find Instance by attribute or tag \(case-sensitive\)](#) [All states](#)

[Clear filters](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
CDD-ONPREM-DNSA	i-069b3825b8fdbb49c	<span style="color: green;">Running</span>	t2.micro	<span style="color: green;">2/2 checks passed</span>	<a href="#">View alarms</a> +	eu-central-1a	-
<b>CDD-ONPREM-APP</b>	i-0d9377ba12a7ba7c9	<span style="color: green;">Running</span>	t2.micro	<span style="color: green;">2/2 checks passed</span>	<a href="#">View alarms</a> +	eu-central-1b	-
CDD-ONPREM-DNSB	i-03f68339cea839d44	<span style="color: green;">Running</span>	t2.micro	<span style="color: green;">2/2 checks passed</span>	<a href="#">View alarms</a> +	eu-central-1b	-

[EC2](#) > [Instances](#) > [i-0d9377ba12a7ba7c9](#) > [Connect to instance](#)

### Connect to instance [Info](#)

Connect to your instance i-0d9377ba12a7ba7c9 (CDD-ONPREM-APP) using any of these options

[EC2 Instance Connect](#) [Session Manager](#) [SSH client](#) [EC2 serial console](#)

**Session Manager usage:**

- Connect to your instance without SSH keys, a bastion host, or opening any inbound ports.
- Sessions are secured using an AWS Key Management Service key.
- You can log session commands and details in an Amazon S3 bucket or CloudWatch Logs log group.
- Configure sessions on the Session Manager [Preferences](#) page.

[Cancel](#) [Connect](#)

15. Then you need to copy the Private IP address of your EC2 instance in Ohio and then try to ping that in Session Manager as shown below. Here you can see that we are able to make the connection with our instance.

```
sh-4.2$ ping 10.16.45.114
PING 10.16.45.114 (10.16.45.114) 56(84) bytes of data.
64 bytes from 10.16.45.114: icmp_seq=1 ttl=255 time=99.7 ms
64 bytes from 10.16.45.114: icmp_seq=2 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=3 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=4 ttl=255 time=99.9 ms
64 bytes from 10.16.45.114: icmp_seq=5 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=6 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=7 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=8 ttl=255 time=99.7 ms
64 bytes from 10.16.45.114: icmp_seq=9 ttl=255 time=99.7 ms
64 bytes from 10.16.45.114: icmp_seq=10 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=11 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=12 ttl=255 time=99.7 ms
64 bytes from 10.16.45.114: icmp_seq=13 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=14 ttl=255 time=99.8 ms
64 bytes from 10.16.45.114: icmp_seq=15 ttl=255 time=99.7 ms
64 bytes from 10.16.45.114: icmp_seq=16 ttl=255 time=99.7 ms
64 bytes from 10.16.45.114: icmp_seq=17 ttl=255 time=99.7 ms
^C
--- 10.16.45.114 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16020ms
rtt min/avg/max/mdev = 99.712/99.814/99.908/0.422 ms
sh-4.2$
```

16. And here if we try to ping our DNS server then it is saying that Name or service not known.

```
sh-4.2$ ping cloudservicesdemo.in
ping: cloudservicesdemo.in: Name or service not known
sh-4.2$
```

17. So, currently we have created our environment and in it our instances can talk to each other, but they cannot resolve the DNS name of each other.

## 😊 Step 2: Create Inbound Endpoints

1. Now we will create two Route 53 resolver Endpoints and then update our DNS and App servers.
2. Now you need to go to Route 53 in Ohio region because here our DNS zone has been hosted and go to Inbound endpoints under Resolver as shown in the snapshot below. Click on create inbound endpoints.

You are signed in to the following Region: us-east-2 (Ohio)  
To change your Region, use the Region selector in the upper-right corner.

Inbound endpoints (0) [Info](#)

ID	Name	Status	Host VPC	IP addresses	Resolver Endpoint Type	Transmission protocols
No resources You don't have any resources. <a href="#">Create inbound endpoint</a>						

3. Here you need to give it an endpoint name of your choice, then choose the VPC, after that, you need to choose the same Security group as shown below and in the Endpoint, type Choose IPv4 and scroll down.

### General settings for inbound endpoint

**Endpoint name**  
A friendly name lets you easily find your endpoint on the dashboard.

The endpoint name can have up to 64 characters. Valid characters: a-z, A-Z, 0-9, space, \_ (underscore), and - (hyphen)

**VPC in the Region: us-east-2 (Ohio) | Info**  
All inbound DNS queries will flow through this VPC on the way to Resolver. You can't change this value after you create an endpoint.

**Security group for this endpoint | Info**  
A security group controls access to this VPC. The security group that you choose must include one or more inbound rules. You can't change this value after you create an endpoint.

**Endpoint Type**  
Route 53 Resolver endpoints support IPv4, IPv6, and Dual-stack IP addresses. For a Dual-stack connection one endpoint can use both IPv4 and IPv6 addresses to connect to a VPC.

**Protocols for this endpoint | Info**  
The protocols for this endpoint determine how data is transmitted to this endpoint. Choose the data transmission protocol with the level of security required for your inbound endpoint.

Do53

4. Then in the IP address 1 you have to choose the availability zone, so choose us-east-2a and the subnet, similarly in the IP address 2 choose AZ as us-east-2b and the subnet. Then click on create endpoint.

**IP addresses** [Info](#)

To improve reliability, Resolver requires that you specify two IP addresses for DNS queries. We recommend that you specify IP addresses in two different Availability Zones. After you add the first two IP addresses, you can optionally add more in the same or different Availability Zones.

**▼ IP address #1** [Remove IP address](#)

**Availability Zone** [Info](#)  
The Availability Zone that you choose for inbound DNS queries must be configured with a subnet.

▼

**Subnet** [Info](#)  
The subnet that you choose must have an available IP address.

▼

**IPv4 address** [Info](#)  
For inbound DNS queries, you can either let the service choose an IP address for you from the available IP addresses in the subnet, or you can specify the IP address yourself.

Use an IPv4 address that is selected automatically  
 Use an IPv4 address that you specify

**▼ IP address #2** [Remove IP address](#)

**Availability Zone** [Info](#)  
The Availability Zone that you choose for inbound DNS queries must be configured with a subnet.

▼

**Subnet** [Info](#)  
The subnet that you choose must have an available IP address.

▼

**IPv4 address** [Info](#)  
For inbound DNS queries, you can either let the service choose an IP address for you from the available IP addresses in the subnet, or you can specify the IP address yourself.

Use an IPv4 address that is selected automatically  
 Use an IPv4 address that you specify

[Add another IP address](#)

5. Here it has created an inbound endpoint and two IP addresses in different Az. You need to copy these IP addresses and paste them in your notepad for later use.

Inbound endpoint: AWS-OnPrem-Endpoint [Info](#)

AWS-OnPrem-Endpoint Configuration																											
ID rs1vr-in-bbdbda462cf42438f9	Status <span style="color: green;">Operational</span>	Host VPC vpc-0df226de6ad01e396																									
Name AWS-OnPrem-Endpoint	Security group sg-0f0a4386447692596	Resolver Endpoint Type IPv4																									
Transmission Protocols Do53																											
<b>IP addresses (2)</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="5">IP address</th> <th colspan="2">Availability Zone</th> </tr> <tr> <th></th> <th>IP address ID</th> <th>Status</th> <th>Subnet</th> <th></th> <th>Availability Zone</th> </tr> </thead> <tbody> <tr> <td><input type="radio"/></td> <td><a href="#">10.16.106.180</a></td> <td>rni-3323d11a08ac4aca</td> <td><span style="color: green;">Attached</span></td> <td>subnet-042836e19620a564c</td> <td>subnet-042836e19620a564c</td> </tr> <tr> <td><input type="radio"/></td> <td><a href="#">10.16.36.212</a></td> <td>rni-ff49247837a84cc48</td> <td><span style="color: green;">Attached</span></td> <td>subnet-083aef19e0cbf741f</td> <td>subnet-083aef19e0cbf741f</td> </tr> </tbody> </table>			IP address					Availability Zone			IP address ID	Status	Subnet		Availability Zone	<input type="radio"/>	<a href="#">10.16.106.180</a>	rni-3323d11a08ac4aca	<span style="color: green;">Attached</span>	subnet-042836e19620a564c	subnet-042836e19620a564c	<input type="radio"/>	<a href="#">10.16.36.212</a>	rni-ff49247837a84cc48	<span style="color: green;">Attached</span>	subnet-083aef19e0cbf741f	subnet-083aef19e0cbf741f
IP address					Availability Zone																						
	IP address ID	Status	Subnet		Availability Zone																						
<input type="radio"/>	<a href="#">10.16.106.180</a>	rni-3323d11a08ac4aca	<span style="color: green;">Attached</span>	subnet-042836e19620a564c	subnet-042836e19620a564c																						
<input type="radio"/>	<a href="#">10.16.36.212</a>	rni-ff49247837a84cc48	<span style="color: green;">Attached</span>	subnet-083aef19e0cbf741f	subnet-083aef19e0cbf741f																						

6. Now we are going to update the zone in our EC2 instance DNS A which is in Frankfurt. By using the zone file that you have seen at the start.
7. So, inside the DNS server you are going to update this file in you need to paste the IP address in place of inbound endpoint IP1 and IP2.

```
zone "aws.clouddeepdive.org" {
    type forward;
    forward only;
    forwarders { INBOUND_ENDPOINT_IP1; INBOUND_ENDPOINT_IP2; };
};
```

8. Now connect with your instance and run some commands. Below you can see the commands. First, you need to enter the first command and then enter the second command to add some data in your configuration file.
9. After entering the second command you need to update the zone file now so copy the content from the file and paste it here as you can see below. After that just save it and then run the restart command.

```
cd /etc
sudo nano /etc/named.conf
sudo service named restart
```

```
GNU nano 2.9.8

options {
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query { any; };
    allow-transfer { localhost; 192.168.10.179; };
    recursion yes;
    forward first;
    forwarders {
        192.168.10.2;
    };
    dnssec-enable yes;
    dnssec-validation yes;
    dnssec-lookaside auto;
    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";
    managed-keys-directory "/var/named/dynamic";
};

zone "corp.clouddeepdive.org" IN {
    type master;
    file "corp.clouddeepdive.org.zone";
    allow-update { none; };
};

zone "aws.clouddeepdive.org" {
    type forward;
    forward only;
    forwarders { 10.16.106.180; 10.16.36.212; };
};
```

10. Now we need to go to the EC2 instance App server which is running in the Frankfurt region. In this server also we are going to update a file. So, open the session manager and run this command

**sudo nano /etc/sysconfig/network-scripts/ifcfg-eth0**

11. Then you will be in this type of window inside your instance, now you need to update the DNS server in your App server for that go to bottom and write the private IP address of your 2 DNS servers which you can find in Frankfurt region. Now just save it.
12. In the end you need to reboot your server for that run this command

**sudo reboot**

```
GNU nano 2.9.8
```

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
TYPE=Ethernet
USERCTL=yes
PEERDNS=yes
DHCPV6C=yes
DHCPV6C_OPTIONS=-nw
PERSISTENT_DHCLIENT=yes
RES_OPTIONS="timeout:2 attempts:5"
DHCP_ARP_CHECK=no
DNS1=192.168.10.20
DNS2=192.168.10.179
```

13. Similarly, you need to update your DNS B server also, like you did with your DNS A server
14. After doing that you need to connect with your App Server.
15. So, below you can see that when we try to ping the private IP of our EC2 instance in the Ohio region then we are getting a response but this time we also get the response from the DNS zone.
16. This time it could resolve the DNS query.

```
sh-4.2$ ping 10.16.45.114
PING 10.16.45.114 (10.16.45.114) 56(84) bytes of data.
64 bytes from 10.16.45.114: icmp_seq=1 ttl=255 time=101 ms
64 bytes from 10.16.45.114: icmp_seq=2 ttl=255 time=101 ms
64 bytes from 10.16.45.114: icmp_seq=3 ttl=255 time=101 ms
64 bytes from 10.16.45.114: icmp_seq=4 ttl=255 time=101 ms
64 bytes from 10.16.45.114: icmp_seq=5 ttl=255 time=101 ms
64 bytes from 10.16.45.114: icmp_seq=6 ttl=255 time=101 ms
^C
--- 10.16.45.114 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 101.655/101.713/101.809/0.266 ms
sh-4.2$ ping cloudservicesdemo.in
PING cloudservicesdemo.in (10.16.45.114) 56(84) bytes of data.
64 bytes from 10.16.45.114 (10.16.45.114): icmp_seq=1 ttl=255 time=101 ms
64 bytes from 10.16.45.114 (10.16.45.114): icmp_seq=2 ttl=255 time=101 ms
64 bytes from 10.16.45.114 (10.16.45.114): icmp_seq=3 ttl=255 time=101 ms
64 bytes from 10.16.45.114 (10.16.45.114): icmp_seq=4 ttl=255 time=101 ms
64 bytes from 10.16.45.114 (10.16.45.114): icmp_seq=5 ttl=255 time=101 ms
64 bytes from 10.16.45.114 (10.16.45.114): icmp_seq=6 ttl=255 time=101 ms
^C
--- cloudservicesdemo.in ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 101.629/101.692/101.758/0.263 ms
sh-4.2$ █
```



## Step 3: Create Outbound Endpoint

1. Now go to route 53 and under resolver you will find outbound endpoints click on it and create it.
2. We created an inbound endpoint earlier, similarly, we need to create these. Follow the snapshots below.

## General settings for outbound endpoint

### Endpoint name

A friendly name lets you easily find your endpoint on the dashboard.

OnPrem-Outbound-EP

The endpoint name can have up to 64 characters. Valid characters: a-z, A-Z, 0-9, space, \_ (underscore), and - (hyphen)

### VPC in the Region: us-east-2 (Ohio) | [Info](#)

All outbound DNS queries will flow through this VPC on the way from other VPCs. You can't change this value after you create an endpoint.

vpc-0df226de6ad01e396 (CDD-aws)



### Security group for this endpoint | [Info](#)

A security group controls access to this VPC. The security group that you choose must include one or more outbound rules. You can't change this value after you create an endpoint.

awsDNS-AWSSecurityGroup-iGcNzUeLbTp (sg-0f0a4386447692596)



### Endpoint Type

Route 53 Resolver endpoints support IPv4, IPv6, and Dual-stack IP addresses. For a Dual-stack connection one endpoint can use both IPv4 and IPv6 addresses to connect to a VPC.

IPv4



### Protocols for this endpoint | [Info](#)

The protocols for this endpoint determine how data is transmitted to this endpoint. Choose the data transmission protocol with the level of security required for your outbound endpoint.

Choose protocol



Do53

## IP addresses [Info](#)

To improve reliability, Resolver requires that you specify two IP addresses for DNS queries. We recommend that you specify IP addresses in two different Availability Zones. After you add the first two IP addresses, you can optionally add more in the same or different Availability Zones.

### ▼ IP address #1

[Remove IP address](#)

#### Availability Zone | [Info](#)

The Availability Zone that you choose for outbound DNS queries must be configured with a subnet.

us-east-2a



#### Subnet | [Info](#)

The subnet that you choose must have an available IP address.

subnet-083aef19e0cbf741f (CDD-private-A) (10.16.32.0/20)



#### IPv4 address | [Info](#)

For outbound DNS queries, you can either let the service choose an IP address for you from the available IP addresses in the subnet, or you can specify the IP address yourself.

Use an IPv4 address that is selected automatically

Use an IPv4 address that you specify

▼ IP address #2

[Remove IP address](#)

Availability Zone | [Info](#)  
The Availability Zone that you choose for outbound DNS queries must be configured with a subnet.

us-east-2b

Subnet | [Info](#)  
The subnet that you choose must have an available IP address.

subnet-042836e19620a564c (CDD-private-B) (10.16.96.0/20)

IPv4 address | [Info](#)  
For outbound DNS queries, you can either let the service choose an IP address for you from the available IP addresses in the subnet, or you can specify the IP address yourself.

Use an IPv4 address that is selected automatically  
 Use an IPv4 address that you specify

[Add another IP address](#)

- Once it is created then you will see that an outbound endpoint has been created with two IP addresses attached to it in different AZs.

Route 53 > Resolver > Outbound endpoints > OnPrem-Outbound-EP

Outbound endpoint: OnPrem-Outbound-EP [Info](#)

[Edit](#) [Delete](#)

OnPrem-Outbound-EP Configuration

ID rslvr-out-10a62adaa291443bb	Status <span style="color: green;">Operational</span>	Host VPC vpc-0df226de6ad01e396
Name OnPrem-Outbound-EP	Security group sg-0f0a4386447692596	Resolver Endpoint Type IPV4
Transmission Protocols Do53		

IP addresses (2)

IP address	IP address ID	Status	Subnet	Availability Zone
<a href="#">10.16.110.139</a>	rni-5d0771aa0314a77bc	<span style="color: green;">Attached</span>	subnet-042836e19620a564c	subnet-042836e19620a564c
<a href="#">10.16.34.49</a>	rni-2c7e7907144249b19	<span style="color: green;">Attached</span>	subnet-083aef19e0cbf741f	subnet-083aef19e0cbf741f

- After that you are going to create rules. For that under the outbound endpoint you will see an option for rules click on it and click on create.

Route 53 > Resolver > Rules								
 You are signed in to the following Region: us-east-2 (Ohio) To change your Region, use the Region selector in the upper-right corner.								
<a href="#">Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#" style="background-color: orange; color: white; border: 1px solid orange;">Create rule</a>								
<a href="#">Rules (1)</a> <a href="#">Info</a>								
Name	ID	Status	Outbound endpoint	Type	Sharing status	Owner	Domain name	
<input type="radio"/> Internet Resolver	rslvr-autodefined-rr-internet-resolver	 Complete	-	Recursive	Not shared	Route 53 Resolver	.	

5. Here you need to give it a name then choose rule type as forward and put your domain name. After that choose VPC and then choose your outbound endpoint.

### Rule for outbound traffic

For queries that originate in your VPC, you can define how to forward DNS queries out of the VPC.

#### Name

A friendly name helps you find your rule on the dashboard.

The rule name can have up to 64 characters. Valid characters: a-z, A-Z, 0-9, space, \_ (underscore), and - (hyphen)

#### Rule type | [Info](#)

Choose **Forward** to forward DNS queries to the IP addresses that you specify in **Target IP addresses** section near the bottom of this page. Choose **System** to have Resolver handle queries for a specified subdomain. You can't change this value after you create a rule.

#### Domain name | [Info](#)

DNS queries for this domain name are forwarded to the IP address that you specify in the **Target IP addresses** section near the bottom of the page. If a query matches multiple rules (example.com and www.example.com), outbound DNS queries are routed using the rule that contains the most specific domain name (www.example.com). You can't change this value after you create a rule.

#### VPCs that use this rule - *optional* | [Info](#)

You can associate this rule with as many VPCs as you want. To remove a VPC, choose the X for that VPC.




#### Outbound endpoint | [Info](#)

Resolver uses the outbound endpoint to route DNS queries to the IP addresses that you specify in the **Target IP addresses** section near the bottom of this page.

6. Then in the target IP address put the Private IP addresses of your DNS server instances and click on create rule.

**Target IP addresses**

DNS queries are forwarded to the following IP addresses:

IPv4 address	Port	Transmission Protocol	
192.168.10.20	53	Do53	<input type="button" value="Remove target"/>
192.168.10.179	53	Do53	<input type="button" value="Remove target"/>
<input type="button" value="Add target"/>			

7. So, this rule will help us to resolve the DNS query for our EC2 instance in the Ohio region.
8. So, now you need to connect with your EC2 instance in your Ohio region and you will see that it is able to connect with your App server instance.
9. Now try the DNS name and you will see that it is able to resolve the DNS query.

Session ID: root-2h5dd7wkbxum5zwu5gk4cukyxy

Instance ID: i-0fbfe3eb3e37d25d6

```
sh-4.2$ ping 192.168.10.180
PING 192.168.10.180 (192.168.10.180) 56(84) bytes of data.
64 bytes from 192.168.10.180: icmp_seq=1 ttl=255 time=102 ms
64 bytes from 192.168.10.180: icmp_seq=2 ttl=255 time=102 ms
64 bytes from 192.168.10.180: icmp_seq=3 ttl=255 time=102 ms
64 bytes from 192.168.10.180: icmp_seq=4 ttl=255 time=102 ms
^C
--- 192.168.10.180 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 102.017/102.128/102.421/0.426 ms
```

**To delete the environment first delete the rule, inbound and outbound endpoint from your route 53. Then you need to go and delete the peering connection, after that just delete your cloud formation stack.**