Using AWS PrivateLink to Connect to ElastiCache Redis

Hello!

Introduction

Imagine you have a database service that shares information between many different customers. You would like to set up a caching service so that customer queries can be serviced faster and to save latency and database costs. Since you are hosting this cache as part of your service, you will need to grant permissions to your customers so that they can access the cache.

This tutorial will help you create all the AWS resources necessary for you to connect a client app in a Virtual Private Cloud (VPC) with a Redis cluster hosted in another VPC. Then, it will provide step-by-step instructions on establishing that connection. We will be using AWS PrivateLink to achieve this.

Why PrivateLink? The reason PrivateLink is so useful in this scenario is because it allows resources in different VPCs to interact in a controlled manner without ever routing through the public internet. It ensures a secure connection between your cloud-based resources and allows you to skip setting up route tables and internet gateways; your VPCs will be able to communicate as if they were on the same network. In my experience, PrivateLink has been crucial because the information I had stored in my Redis cluster was confidential and could not have any internet-facing endpoints for security reasons.

However, if PrivateLink turns out not to be the correct technology for your solution, you can also check out VPC Peering Connections. Furthermore, this document will not go into detail about choosing a caching solution; it is assumed that you have already made the design decision to use Redis.

Prerequisites

Before we begin, there are some prerequisites to cover to guarantee your success in completing this tutorial.

A general understanding of cloud-based services is important to follow along why we are provisioning, giving access to, and configuring certain resources as described here. With no background knowledge, it may be difficult to understand the reasoning behind the steps taken. I would recommend first reading up on: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html.

Of course, you must also have access to at least one AWS account. Since this tutorial is meant to help you connect an app in one VPC to a Redis cluster in another VPC, you will need to have the Redis cluster ready. If not, please refer to

https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Clusters.Create.CON.Redis.html for setting up a Redis cluster. Furthermore, it is not absolutely necessary for you to have two separate accounts with a VPC in each, but for the purposes of this tutorial, you will need at least two separate VPCs in one account. That said, if you already have an app in a different VPC than the one your Redis cluster lives in, perfect! If not, I will be referring you to this page later to create an EC2 instance for demonstrative purposes: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html.

No coding experience is required.

Learning Objectives

By the end of this tutorial, you will be able to complete these three things:

1) Create a Load Balancer, Target Group, and Target(s)
2) Create a VPC Endpoint Service and grant connection permissions
3) Create a VPC Endpoint and connect to an Endpoint Service via PrivateLink

It's really that simple! So, without further ado, let's begin!

Part 1: Create a Load Balancer, Target Group, and Target(s)

If we picture the PrivateLink connection like an old 1960's telephone network, you have the client making the call and the service receiving the call, but you need the operator in the middle that says "Operator, number please?" The Load Balancer acts like this operator by intercepting the client's call and directing it to the correct service.

For this part of the tutorial, please be signed into the AWS account and working in the region that the Redis cluster is in.

1) Navigate to the ElastiCache Management Console. Select the Redis cluster that you want to create the PrivateLink connection for and then find the node endpoint. There may be multiple nodes in the cluster; choose the primary endpoint only if you want to grant write-access as well as read-access.

2) Obtain the IP address of the node endpoint by resolving the endpoint DNS. You can do this by just typing "ping [your node endpoint here]" in a terminal window (without the square brackets). It should be 4 sets of numbers between 0 and 255 separated by periods. Save this IP address! (I've blocked out some of mine just in case)

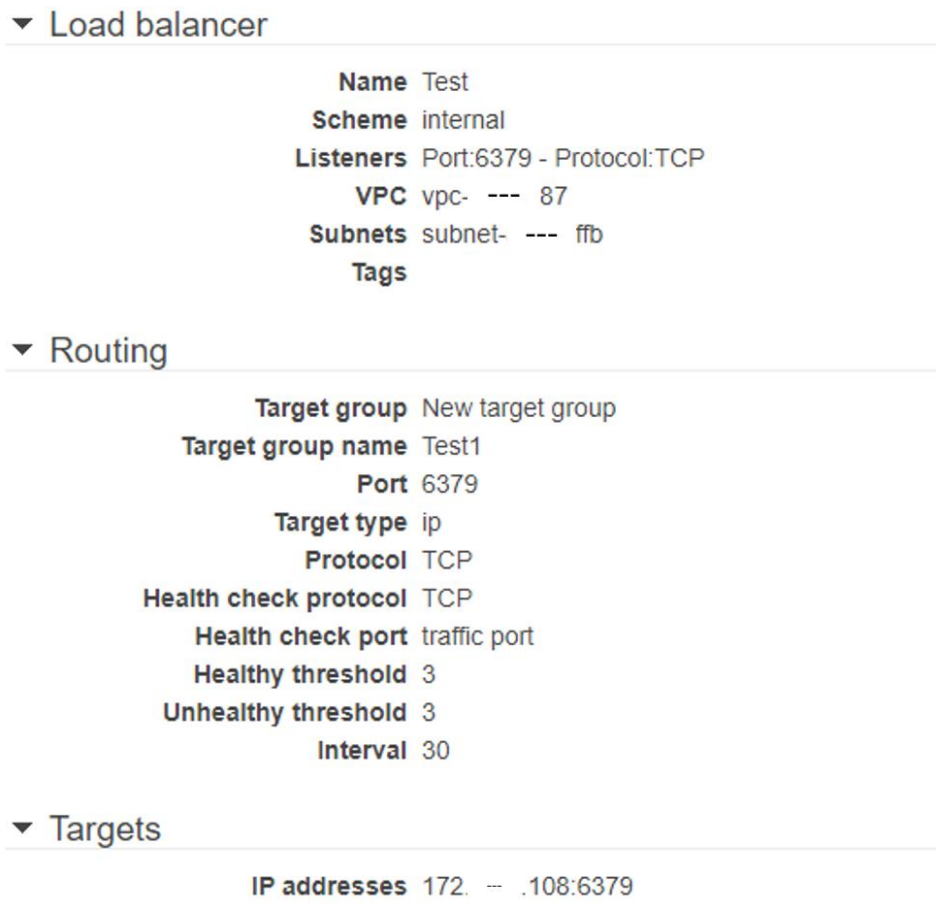
3) Navigate to the EC2 Management Console. Scroll down on the EC2 Dashboard (on the left) and the Load Balancers menu is near the bottom. Let's create a new Load Balancer:
    a. Select the Network Load Balancer option that supports TCP, TLS, and UDP.
    b. After deciding on a name, you can choose the "internal" option under Scheme if you want to only allow internal connections (through AWS servers and not the internet). Since we are using PrivateLink connections, select "internal".
    c. Add a Listener that uses TCP protocol and connects to port 6379. Listeners are our operator of the telephone company; when they "hear" a request of a certain type/port, they will route it forward. For consistency and ease of understanding, we will use port 6379, the default port for Redis clusters, for the listener as well.

    d. Select the VPC that your Redis cluster belongs to, and select the VPC Subnets in the availability zones that your Redis cluster is in. You can find this information in the ElastiCache console when you select your Redis cluster. See Appendix A for help if you can't find it.
    e. Click Next. It may tell you to improve your load balancer's security. For the purpose of this tutorial, you can safely ignore this for now, but you may want to look into it here: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/create-tls-listener.html
    f. Click next again if needed. At the top of the page, you should be at '*Step 3: Configure Routing*'.

4) We're now going to create a new Target Group. Give it a suitable name!
    a. For the Target type, we will have to choose "IP". This is because AWS Load Balancers are designed for use with EC2 instances; if not to an EC2 instance, it can only route traffic to an IP address, and not a DNS.
    b. Protocol and Port number should be the same as the Listener (TCP 6379)
    c. You can leave Health checks as-is, on the TCP setting.
    d. Click next. At the top of the page, you should be at '*Step 4: Register Targets*'.

5) We're now going to add Targets. Almost done with Part 1!
    a. Select the VPC Network that your Redis cluster lives in.
    b. Enter the IP address that you saved from earlier in Step 2.
    c. Change the Port number to 6379.

d. Click Add to list. Move to the last '*Step 5: Review*' and ensure your configurations match up with mine. (Again, some values have been blanked out)



6) Navigate to the Security Groups menu (halfway down the EC2 Dashboard). You'll want to select the security group that manages your Redis cluster. Again, that information is available in the ElastiCache details page (see Appendix A).
    a. Add an inbound rule that allows traffic from TCP protocol port 6379. The source should be from the Subnet CIDR that your Network Load Balancer is in. Now you're done with Part 1!

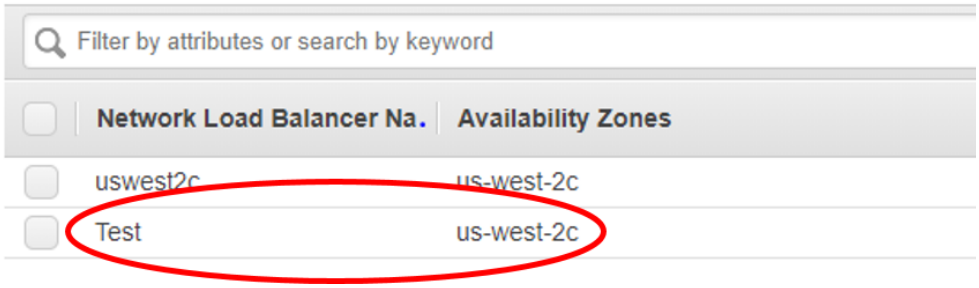
So, what did we just do? We created a Load Balancer that has a Listener that listens for requests from TCP port 6379. The Listener will route these requests to the Load Balancer's Target Group which contains a Target that points to our Redis cluster's IP address. We also had to allow inbound traffic from the Load Balancer to our Redis cluster through security groups. Ultimately, the TCP port 6379 request at our Load Balancer will end up connecting to our Redis cluster using TCP at port 6379. Neat!

Part 2: Create a VPC Endpoint Service and grant connection permissions

Now that we have our telephone company operator set up, we need to prepare our receiver's phone. This is the VPC Endpoint Service.

For this part of the tutorial, please stay signed into the AWS account and working in the region that the Redis cluster is in. This should be the same account that the Load Balancer we just made belongs to.

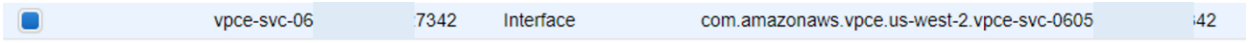
1) Navigate to the VPC Management Console. About halfway down on the left you'll find the Endpoint Services menu. We're going to create a new Endpoint Service:
    a. Find the Load Balancer you just made. Yours may be in a different Availability Zone, depending on where your Redis cluster lives.

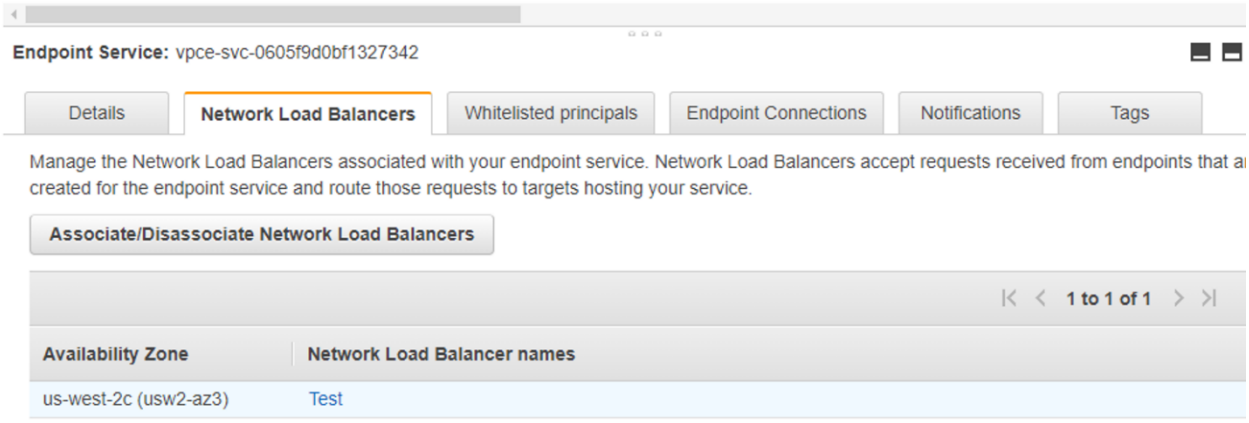


    b. You can choose the option of requiring acceptance for endpoint. This means you can choose if you want to manually accept all PrivateLink requests. If not, anyone who has access to your Endpoint Service can create a connection. Depending on your security and use-case, you can choose what works best. For now, let's keep the checkbox marked.
    c. You can also add private DNS names and Tags for further customization. For now, let's just skip them and go ahead and click "Create service". Done!

2) You should see your Endpoint Service in the menu now. If you have multiple, you can use Tags to add a unique name to help you keep track of them. Click on your Endpoint Service and go the Network Load Balancers menu. You should be able to see your Load Balancer here, as well as the Availability Zone. You can always add more Load Balancers

to this Endpoint Service at a later point in time.





3) Now, go to the Whitelisted Principals menu. You will need to add entries to the whitelist. These principals will be one of three: the IAM User, IAM Role, or AWS Account your client application is associated with. It is a best practice to give the least permissions possible, so as to not introduce security vulnerabilities.
    a. For the purpose of this tutorial, we can cautiously grant root permissions to an AWS account, if IAM User or IAM Role information is unavailable. To do so, add a principal to your whitelist like: “arn:aws:iam::42xxxxxxxx42:root” replacing with your client’s AWS account.
        i. If you do not actually have a client, and will be trying to connect to your Redis cluster from a different VPC in the same account, use your own account.
        ii. If you do have IAM User or Role information, please use that instead.
    b. After adding the principal, click “Add to whitelisted principals”.

4) This last step is easy. Go back to the Details page of your Endpoint Service. Find the Service name and save this somewhere. You have completed Part 2 of the tutorial!

So, what was all that? We created a VPC Endpoint Service and attached our Network Load Balancer to it. Then we added some principals to a whitelist. This means that when an app associated with a whitelisted principal tries to connect to our Endpoint Service, their request will be picked up by our Listener. Our Listener will forward the request to a Target in our Target Group. Ultimately, any valid requests to our Endpoint Service will end up at our Redis cluster. Awesome!

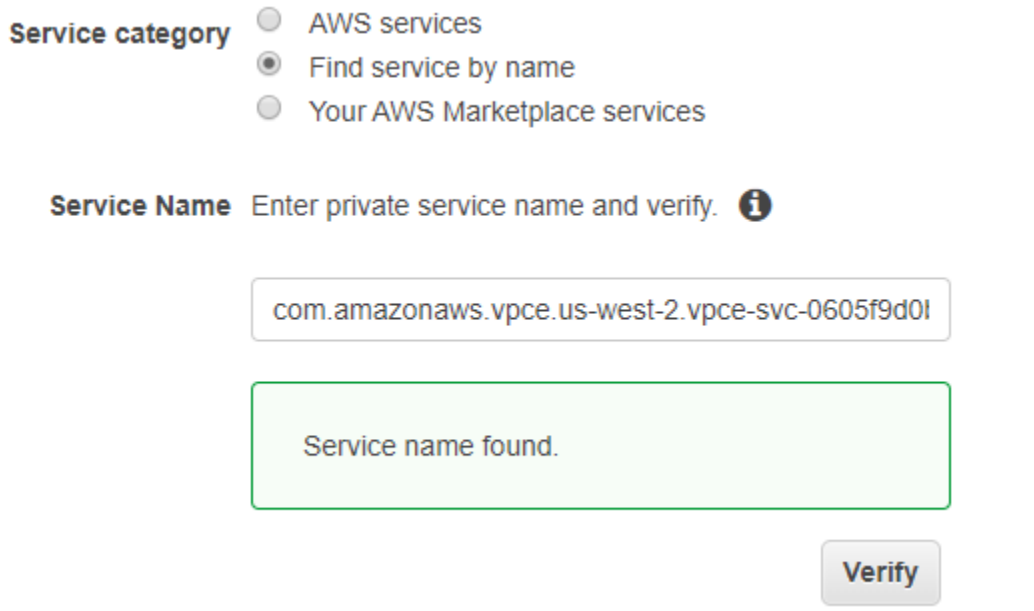
Part 3: Create a VPC Endpoint and connect to an Endpoint Service via PrivateLink

Now that we have our telephone operator prepared and our receiving phone ready to take calls, we need somebody to call us! That is what a VPC Endpoint is. The Endpoint will allow a VPC to communicate with another VPC's Endpoint Service (what we just provisioned). Since our Endpoint Service uses an internal Network Load Balancer, any connecting Endpoints will never route through the public internet! Instead, using AWS PrivateLink, we will be able to access our Redis cluster through a secure connection through AWS' servers.

For this part of the tutorial, please now sign in to the other AWS account and work in the region that the client application is in. I'd suggest using a different browser or an incognito window because you'll have to switch back and forth between the two AWS accounts a little bit.

If you do not have a client account, and are creating a PrivateLink from a different VPC in the same account, I'd suggest provisioning an EC2 instance for demonstration purposes: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html. This is what I will use to demonstrate the connection at the end. Otherwise, if you have another application you're using, you can continue with that instead.

1) Navigate to the VPC Management Console, and find the Endpoints section (right above Endpoint Services). Let's create an Endpoint:
    a. You will want to find a service by name. This name is the service name that you saved at the end of Part 2.



    b. Select the VPC that the client app lives in (or the second VPC without the Redis cluster), and select the Availability Zone/Subnet(s) that is supported. If none are available, you will have to create a new subnet in your VPC.

i. Note: region codes (us-east-1a) may be different between accounts, and only availability zone IDs are consistent (us1-az1).

c. Select the Security Group that manages the client app (or the EC2 instance that you provisioned at the start of Part 3).

d. You can add Tags for further customization, but we'll go ahead and create the endpoint.

2) Navigate to the EC2 Management Console and find the Security Groups menu a little over halfway down on the left. Find the security group that manages the client app (the same one you selected in Step 1.c just before).

a. Add a new inbound rule using Custom TCP, port 6379. The source should be the CIDR block(s) of the subnet(s) that you used in Step 1.b just before. If you have trouble finding this, you can refer to Appendix B.

Custom TCP	TCP	6379	192.16	---	92/26

i. This is to allow the traffic from the Endpoint Service to communicate back into your client, which you need to receive the responses from the Redis cache.

Almost done! There is one more step left. For this step, you will have to sign back into the AWS account that owns the Endpoint Service (and the Load Balancer and Redis cluster).

Note: If you turn off requiring acceptance for connection requests (as in Step 1.b of Part 2), you can skip this final step (it doesn't apply to your configuration).

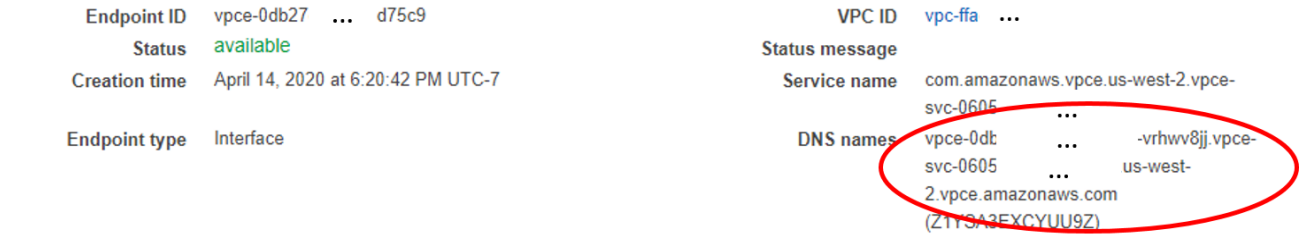
3) Navigate back to your Endpoint Services in the VPC Management Console. In the Endpoint Connections page, you should see a new connection with a Pending Acceptance state from the Endpoint you created earlier in Part 3. The owner of the connection should be the principal that you had added to the whitelist from before. Under Actions, click "Accept endpoint connection request".

	Endpoint ID		Owner		State
☑	vpce-0db27	d75c9	42	42	Pending Acceptance

It will change the state to Pending, and then to Available when it's ready. Now you're done!

Whew! Let's recap what we accomplished in Part 3. We set up an Endpoint that connects to our Endpoint Service, ensuring that communication can flow both ways, and finalized the PrivateLink by accepting the connection request. Now, how do we verify it works?

We'll have to move over to our client account. Let's find the PrivateLink DNS name under the details page for the Endpoint.



This is the DNS address you will be using to connect to your Redis cluster. What happens when you send a request to this address? Well, let's open up our EC2 instance that we're using in place of our client application.

When I run this command:

telnet vpce-0db … -vrhwv8jj.vpce-svc-0605 … us-west-2.vpce.amazonaws.com 6379

this tries to open a connection to that Endpoint address through port 6379 using telnet, a terminal connection application protocol. The response is:

```
Trying 192.1  ---  92...
Connected to vpce-0db27  ---  d75c9-vrhwv8jj.vpce-svc-
0605     ---     .us-west-2.vpce.amazonaws.com.
Escape character is '^]'.
ping
+PONG
set foo bar
+OK
get foo
$3
bar
```

We can see the connection has gone through in the second line that says "Connected to …". I also ran a couple Redis commands afterwards just to be sure. For the unfamiliar, "ping" is a Redis command that returns "PONG" when a connection is successful. The subsequent commands are basic set/get commands for Redis. It works!

Conclusion

Congratulations! You now have a fully functioning PrivateLink between a client VPC and your ElastiCache Redis cluster in a different AWS account and/or separate VPC. Let's go over one more time exactly how this works.

When your client makes a request to the Endpoint address at port 6379, it travels through AWS' secure servers using PrivateLink to the Endpoint Service of your service host VPC. There, the request is picked up by the Network Load Balancer's Listener (recall the phone operator analogy)

where it is forwarded to a Target. The Target points to the IP address where our Redis cluster node is. So, ultimately, when your client makes a request to the Endpoint, it gains a connection to your Redis cluster.

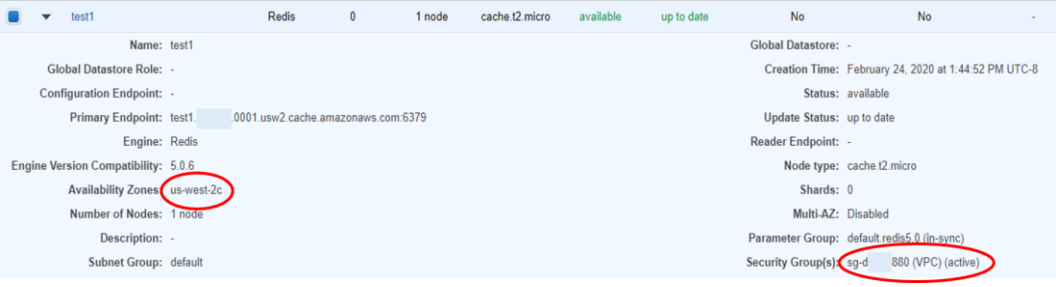
You are now ready to provide a caching infrastructure for your customers!

Next steps

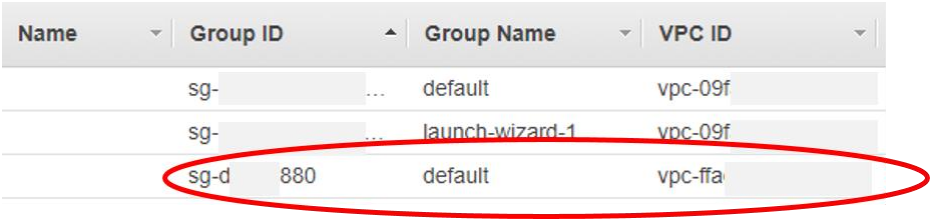
Now that you are able to use PrivateLink to connect VPCs and serve customers, you can design more infrastructure and/or services that share resources between different clients. You can also go back and explore using private DNS names for your Endpoint services. Good luck!

Appendix A

Find the Redis cluster you wish to use. Find the Security Group(s) the cluster is managed by. There should be at least one security group that is associated with a VPC. Save this security group ID. Also, you can find the Availability Zone(s) here; save those too.



Now, go to the VPC Management Console and find the Security Groups menu halfway down the bar on the left. Find the security group ID that you had just saved, and find the VPC ID.



Appendix B

Go to your Endpoints in the VPC Management Console. Under the Subnets page of your Endpoint, you will see what subnets are associated with your Endpoint. Click on them to be directed to your Subnet page where the subnet's CIDR blocks will be available.