Lab 9 Alternative:   
Private Link for Managed Connectors

# Prerequisites

To set up a private link in the Confluent Cloud to point to a private link endpoint service in AWS, we need several preparatory steps:

* AWS VPC with at least one private subnet
* Instance running within the private subnet, large enough for the source database
* Installation and configuration of the database, preferably PostgreSQL, via Ansible (scripts adaptable from [github.com/sknop/bootcamp-connect](http://github.com/sknop/bootcamp-connect)
* Dedicated cluster in Confluent Cloud in the same region as the AWS VPC

# Lab steps

* Ensure VPC has been created (use <https://github.com/sknop/simple-vpc>). You should have performed the steps for the Dedicated Cluster lab, but here they are again:
  + Copy terraform.tfvars.template to terraform.tvars
  + Pick the AWS region of your choice
  + Run terraform init -upgrade
  + Verify your settings with terraform plan
  + Run terraform apply
  + Check the output for clues on the details for your setup
* Test your jumphost is working and reachable by logging into it from the directory you run your Terraform script from:
  + ssh -A -i bootcamp.pem ubuntu@<jumphost ip address in the output>  
    The option -A enables forwarding of connections from an authentication agent such as ssh-agent. This is useful when logging into the to database instance from the jumphost.
* Install database
  + From your jumphost, ssh into the database instance  
    ssh ubuntu@<database ip address in the terraform output>
    - If this command fails, you might have issues with your ssh-agent (this is what the -A option in the ssh command above is for). If you cannot fix that, copy the bootcamp.pem file from your laptop to the jumphost, move the file into your .ssh directory and ensure you have only read permission for your user - chmod 400 ~/.ssh/bootcamp.pem.
  + On the database instance, clone the repository that contains the setup script  
    git clone <https://github.com/sknop/bootcamp-connect>
  + cd bootcamp-connect/docker
  + Run the setup script:   
    ./setup.sh  
    The script will install docker, and other required tools, then spin up a Postgres container and configure the database. It will also install Java and Maven needed for the next step.
* Build and run the uploader service
  + cd back to your home directory
  + cd bootcamp-connect/movie\_datagen
  + mvn clean package
  + java -cp target/movie\_datagen-1.0.0.jar io.confluent.ps.datagen.DataGen jdbc:postgresql://localhost:5432/movielens ml-latest/ admin adminsecret  
      
    This is one line!
  + Let this run for a few seconds or a minute, then you have sufficient data for the next step. End the process with CTRL-C. You can also keep the process running if you like, the amount of data it generates is bounded.

Now that the database is running, we can run the actual lab

Note: we assume you are using the simple-vpc script to create your VPC, otherwise replace the name simple-vpc with your VPC.

* Create an AWS Endpoint Service
  + Log into the AWS console
  + Create an open security group
    - Navigate to the EC2 panel
    - On the left-hand side, go to Security Groups
    - Create a new security group for your VPC (simple-vpc)
    - Give it a useful name like “Confluent Cloud Access”
    - Add a new Inbound rule for all TCP for the source 0.0.0.0/0
  + Create the endpoint service
    - Navigate to the VPC panel
    - On the left-hand side, go to Endpoint Services
    - Create a new endpoint service
      * Give it a name.
      * It is a Network load balancer, not a gateway
    - Create a new load balancer by clicking on the button
      * Again, it is a Network Load Balancer
      * Give it a name
      * It is an Internal Scheme
      * Choose the correct VPC (simple-vpc)
      * Map all three AZ to the private subnets, let AWS choose the IP address out of their range
      * Select the security group that you have created earlier
      * Set up the listener as TCP pointing to port 5432
    - Create a new target group by clicking on the “Create target group” link
      * Target type IP addresses
      * Give it a speaking name
      * Pick port 5432 (the postgres default port)
      * Choose the correct VPC (simple-vpc)
      * Pick the private IP address of your database host , add it as pending, then create the target group.
    - Refresh the target groups in the listener section of the load balancer, then choose your new target group
    - Create the load balancer
    - Important: set up cross-zone balancing
      * Navigate to the details map of the load balancer
      * Open the Attributes tab
      * You should see cross-zone balancing set to Off by default
      * Press Edit, then change the cross-zone balancing to “Enable cross-zone load balancing”
      * Save changes
    - Wait until the load balancer has been provisioned (this might take a few minutes)
    - Choose the new load balancer in your endpoint service panel
    - Choose “Acceptance required”
    - Support IP address type IPv4.
    - Create the endpoint service
  + Now you need to add the allowed principals. These come in ARN (AWS Resource Number) format.
    - Head to the Confluent Cloud to your dedicated cluster, then to the networking tab. Choose your network, then open the “Egress access points” tab. The correct ARN number can be found in the text with a convenient copy button to the right of it.
    - Copy the ARN, then head back to the AWS console and add this principal to your endpoint. Ensure you press “Allow principals” when finished.

Your endpoint service is now ready to receive endpoint connections.

* Set up an Egress Access Point in your dedicated cluster
  + In the Confluent Cloud Network tab you have just been, press “Create Access Point”
  + Give the access point a name, and add the private link service name from your endpoint service. You can find the service name in the details tab in AWS as “Service Name”. It will have the form “com.amazonaws.vpce.<region>.<service ID>
  + Enable Highly Available if you have chosen multiple AZs (if you followed this guide, you have)
  + The endpoint takes a few seconds to provision, then it will switch to “Pending Accept”
  + You will find the pending endpoint in the “Endpoint connections” tab in the details for your Endpoint Service.
    - Select the endpoint connection from the Confluent Cloud, then choose Actions>Accept pending connection request
    - Copy the VPC endpoint DNS name for safe keeping, you will need this for the connector
    - Once the state has changed to Available, you can return to the Confluent Cloud Console
* It will take several minutes for the endpoint to be acknowledged. Once up and running it should show “Ready”

Now it is time to set up the connector

* Go to your dedicated cluster, then select Connectors
* Create a new Debezium Postgres (v2) connector
* You can use a My Account Access key for this exercise
* Set up the database connection
  + Database host is the VPC endpoint DNS name you copied above
  + Port is 5432
  + Username is admin
  + Password is adminsecret
  + Database is movielens
  + Keep SSL mode to prefer
* Output messages setup
  + Choose AVRO from value format
  + STRING for Kafka record key
  + You can define your own topics prefix, I tend to use “postgres”
  + Leave slots and publication name and Connector Config values at default
* Leave tasks at 1
* Give the connector your own name if you like. It might be useful to copy the JSON config somewhere locally for debugging purposes if something goes wrong.

Now we need to wait until the connector is provisioned. If your private link is set up incorrectly or you mistyped the username or password, it will fail quickly, but it can take several minutes to start up successfully.

If the connector claims it is running, you can check the topics - if you have the proxy or dynamic proxy running, you can use the Confluent Cloud console.