CONFLUENT

Enterprise Cluster with Private Network Interface

Set up and configure an Enterprise cluster

Objectives

Confluent Private Network Interface (PNI), powered by AWS Elastic Network Interface (ENI), enables you to attach an ENI from your AWS account to a network service in the Confluent AWS account. This allows you to access Confluent Cloud services, such as Freight and Enterprise clusters, through the ENI that resides in your AWS account and offers PrivateLink-like one way connectivity with user controlled security groups. In this lab, you will create an Enterprise cluster and connect it via private endpoints to an AWS VPC. You will also learn how to configure your clients to be able to access the Enterprise cluster.

Labs

Create an AWS VPC

- ☐ Confluent Cloud Enterprise Clusters currently support a subset of AWS regions. All regions in AWS that support Enterprise clusters also support PNI access. Pick your AWS region out of the following:

  Africa:

  - af-south-1 (Cape Town)

  Asia Pacific:

  - ap-east-1 (Hong Kong)
  - ap-northeast-1 (Tokyo)
  - ap-northeast-2 (Seoul)
  - ap-northeast-3 (Osaka)
  - ap-south-1 (Mumbai)
  - ap-south-2 (Hyderabad)
  - ap-southeast-1 (Singapore)
  - ap-southeast-2 (Sydney)

- ap-southeast-3 (Jakarta)

Canada:

- ca-central-1 (Canada Central)

Europe:

- eu-central-1 (Frankfurt)
- eu-central-2 (Zurich)
- eu-north-1 (Stockholm)
- eu-south-1 (Milan)
- eu-south-2 (Spain)
- eu-west-1 (Ireland)
- eu-west-2 (London)
- eu-west-3 (Paris)

Middle East:

- me-south-1 (Bahrain)
- me-central-1 (UAE)

South America:

- sa-east-1 (São Paulo)

United States:

- us-east-1 (N. Virginia)
- us-east-2 (Ohio)
- us-west-2 (Oregon)

□ If you do not have a VPC in a supported region, create another VPC in your chosen region as before for the Private Link Enterprise Cluster. You will also need 3 subnets across the first three availability zones (AZ).

You should use the Terraform scripts at https://github.com/sknop/simple-vpc if you are still getting familiar with AWS. Here are the instructions:

- □ `git clone https://github.com/sknop/simple-vpc`
- □ `cd simple-vpc`
- □ `cd aws`
- □ Copy `terraform.tfvars.template` to `terraform.tfvars`.
- □ Edit `terraform.tfvars` and pick the AWS region of your choice.

  - □ Adjust other parameters as required.
  - □ Run `terraform init -upgrade`
  - □ Verify your settings with terraform plan
  - □ Run `terraform apply`
  - □ Check the output for clues on the details of your setup.

Create the Enterprise Cluster

- □ Create an Enterprise cluster in the Confluent Cloud Console (UI).
  - □ Choose the same AWS region where you created your VPC instance.
  - □ Choose the 99.99% SLA (this invokes deployment across 3 AZ).
  - □ Skip the networking configuration for now.
  - □ Skip the payment.
  - □ Pick a name for your cluster and launch it.

Connect the cluster to AWS

- □ In your new cluster, navigate to the Network Management panel (it is flagged with an exclamation mark and a yellow triangle).
- □ Click on “Create network configuration”
  - □ Choose “Private Network Interface"
  - □ Give the Gateway a name.
    - □ Give the gateway a name like “enterprise-PNI-gateway”
    - □ Choose 3 zones for your gateway.
- □ In Configure gateway, create an access point
  - □ Give it a name like “enterprise-PNI-access-point”
  - □ Download the script, making a note of the Confluent AWS account ID.

- □ Run the script on your local machine.
    - □ You need to make sure that your AWS CLI is configured correctly. If you were able to run your Terraform script that created your VPC, you are set up correctly.
    - □ You need to specify 10 arguments to the script:

      ```
      <subnet-id1> <base-ip1>
      <subnet-id2> <base-ip2>
      <subnet-id3> <base-ip3>
      <security-group-id>
      <confluent-aws-account-id>
      <aws-region>
      <num_pni_per_subnet>
      ```

      For example

      ```
      ./create_aws_enis_with_permission.sh subnet-00000000000000001
      100.251.1.10 subnet-00000000000000002 100.251.2.10
      subnet-00000000000000003 100.251.3.10 sg-9999999999999999 012345678901
      eu-west-1 17
      ```

        - □ You can find the subnets in the output of the VPC Terraform creation script.
        - □ Ensure there are no conflicts with the IP address range for each subnet. If the script encounters an IP address already in use, it will simply fail.
          The recommendation is to create 17 ENI per subnet. In the above example, the script would create ENIs with IP addresses between 100.251.3.10 and 100.251.3.26.
    - □ The security group can be a simple open security group for this lab. In reality, the security group can be used to prevent access of the VPC from Confluent Cloud and turn the link from bi-directional into uni-directional.
    - □ The script returns the identifiers of the ENI created. Copy that list and add it to the “Create Access Point” panel in step 3.

    - □ Add the AWS account ID of your AWS account in step 4. You can find your account ID in the AWS Console top right in the drop-down menu.
- □ Complete the creation of the gateway and access point.

Note that you do not have to create a DNS entry. The routing will be done automatically by Confluent.

Test your cluster

- □ SSH into your jumphost in the VPC you created and to which you set up the private link endpoint.
- □ Verify that you can connect to your new cluster from your jumphost.
    - □ In the Confluent Cloud UI, go to your cluster settings. Look for the "Endpoints" section. Expand the access point you just created to show more details. Notice the Bootstrap and REST endpoint URLs. They contain the string "`accesspoint.glb`." These are the endpoints you use to access your cluster.
    - □ In the field labeled "REST endpoint for Confluent Cloud console used to access this cluster", select the URL that contains the string "`accesspoint.glb`." (You can ignore the error "Endpoint is not available for console use.")
    - □ Use `openssl s_client -connect <YOUR_BOOTSTRAP_URL>` for a first test.
    - □ Create a global API key for your cluster (select "My account" in the Create key wizard). Write the key and secret down. You use them in the next steps.

- □ Create a client configuration file for a C++ client. You can use the following template:

None

```
# Required connection configs for Kafka producer, consumer, and admin
bootstrap.servers=<YOUR_BOOTSTRAP_URL>
security.protocol=SASL_SSL
sasl.mechanisms=PLAIN
sasl.username=<YOUR_API_KEY>
sasl.password=<YOUR_API_SECRET>

# Best practice for higher availability in librdkafka clients prior to 1.7
session.timeout.ms=45000
```

- □ Install kafkacat on your jumphost.
  `sudo apt install kafkacat`
- □ Test the connection with `kafkacat -F <YOUR_PROPERTY_FILE> -L`
- □ Install Java (11 or 17) on your jumphost.
  `sudo apt install openjdk-17-jre-headless`
- □ Install the Kafka Java Clients (kafka-topics, kafka-console-producer, etc.) on your jumphost.

□ Create a client configuration for Java. You can use the following template:

None

```
# Required connection configs for Kafka producer, consumer, and admin
bootstrap.servers=<YOUR_BOOTSTRAP_URL>
security.protocol=SASL_SSL
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username='<YOUR_API_KEY>' password='<YOUR_API_SECRET>';
sasl.mechanism=PLAIN
# Required for correctness in Apache Kafka clients prior to 2.6
client.dns.lookup=use_all_dns_ips

# Best practice for higher availability in Apache Kafka clients prior to 3.0
session.timeout.ms=45000

# Best practice for Kafka producer to prevent data loss
acks=all
```

□ Create a topic, then produce and consume from it on your Enterprise cluster. Reuse the API and secret you created for the C++ client.

□ Test the REST endpoint of your cluster with curl. Use a command like the following example to verify you can list the topics in your cluster.

None

```
curl --request GET --url
https://<YOUR_BOOTSTRAP_URL>:443/kafka/v3/clusters/<YOUR_CLUSTER_ID>/topics -u
"<YOUR_API_KEY>:<YOUR_API_SECRET>"
```

Proxy configuration

If you navigate in the UI to the topic, you can see that the topics are not accessible via the console. The problem is that the console needs to communicate with the cluster in a different network, and this network is only accessible from your VPC via the private endpoint.
In a production environment, a network administrator could set up a VPN to your private VPC and set up a DNS entry to reroute the traffic. For this bootcamp, we can use a workaround via Proxy.

Either:

- □ Install a proxy service on your jumphost, for example, nginx or haproxy. Configure this proxy to point to your Kafka cluster. The documentation below will help you figure out the required settings.

Or:

- □ Alternatively, you can use a dynamic (SOCKS v5) proxy.
  - □ For this, you need to set up a dynamic proxy by adding the option "-D port" to your SSH command when connecting to your jumphost, for example:

    ```
    ssh -D 8081 jumphost
    ```
  - □ Install a dynamic proxy in your browser, for example, Standard FoxyProxy (https://getfoxyproxy.org/).
  - □ Configure the Proxy to point to localhost:port (here, port 8081) with a URL pattern that matches your cluster and a regular expression at the beginning and end, for example like this https://.*lkc-kmzodp-ap4llk04.us-east-1.aws.accesspoint.glb.confluent.cloud*
    (Do not forget to enable your Proxy for your Patterns)
- □ Test the proxy solution with your cluster.
- □ After completing these labs, you can delete the Enterprise cluster again. It consumes resources and hence credits.
- □ Do not forget to destroy your VPC as well (ideally using "terraform destroy").

References

https://docs.confluent.io/cloud/current/networking/aws-pni.html

https://docs.confluent.io/cloud/current/clusters/cluster-types.html#cluster-types

https://www.confluent.io/blog/introducing-private-network-interface

https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html#get-the-software

https://docs.confluent.io/cloud/current/networking/ccloud-console-access.html#configure-a-proxy

https://docs.confluent.io/cloud/current/networking/ccloud-console-access.html

Expected Outcomes

Create an Enterprise Cluster and successfully connect it via Private Networking Interface to your AWS VPC.

Verify access via OpenSSL and Kafka tools.

Successfully create a Proxy solution to enable adding and modifying topics in the Confluent Cloud UI.

Check your understanding

This colour marks advanced questions.

- □ Which networking options are available for an Enterprise cluster?
- □ **How would you change the PNI connection from one-way access to bi-directional access?**