Confluent Platform review for self-service

Index

Current Stockholm Infrastructure 3

Components and services deployed 4

Current Application Workflow 4

OAUTH 4

MTLS 5

Challenges Encountered in Achieving Self-Service 5

Feasibility of Self-Service with Current Setup 6

Staged Tasks Required for Implementation 6

AD Group and Service Account 6

Customer Access to Code in GitLab 6

IP Whitelisting 6

Pull Request Process 6

Broker Restart 6

# Overview:

This task is created to document the current architecture of Confluent platform and how we can achieve self-service with current infrastructure as code.

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Author | Date created | Reviewer | Comments |
| 1.0 | Junaid Ahmed | 4th December 2023 | N/A | initial draft |
|  |  |  |  |  |
|  |  |  |  |  |

# Current Stockholm Infrastructure

* Currently Scania has deployed Confluent platform with version 7.1.1 as streaming platform solution.
* Below is the current architecture.

A computer screen shot of a computer

Description automatically generated

* Currently, access to the streaming platform is facilitated through both the internet and the Scania private network.
* Prior to gaining access to the streaming solution, the application undergoes an IP whitelisting process within AWS security groups.
* The application is provided with the Kafka broker endpoint, initiating a connection to the AWS application load balancer. Subsequently, the load balancer redirects the connection to the appropriate Kafka broker.
* Separate load balancers are dedicated to distinct services such as the REST proxy, schema registry, and Control Center.
* Deployed within a public subnet, these load balancers function as reverse proxies, mitigating the direct exposure of the underlying platform to the internet, thereby enhancing security measures.
* Six public load balancers are allocated for Kafka brokers for clients coming from Internet, complemented by an additional six private load balancers situated in a private subnet specifically designed for on-premises clients.
* To ensure high availability and fault tolerance, all components are strategically deployed across three distinct Availability Zones.
* For applications originating from the Scania data center, connectivity to the streaming platform is established via a direct connect mechanism. This approach is adopted to bolster security by segregating this traffic from other external flows.
* A dedicated internal Scania firewall governs the flow of traffic, necessitating the opening of ports for inbound and outbound communication.
* Positioned between the Scania data center and the streaming platform, a load balancer oversees and manages the traffic flow, contributing to a structured and secure connectivity architecture.

# Components and services deployed.

|  |  |  |
| --- | --- | --- |
| Service/Components | Spread over AZ | Comments |
| Kafka Broker | We have total of 6 pods, spread over 3 az. |  |
| Zookeeper | We have total of 3 pods spread over 3 az. |  |
| Confluent operator | We have 1 pod spread over 1 az. |  |
| Control Center | We have 1 pod spread over 1 az. |  |
| Connect | We have total of 9 pods, in 3 separate cluster of 3s |  |
| KsqlDB | We have 2 pods spread over 2 az. |  |
| Schema registry | We have total of 3 pods spread over 3 az. |  |
| Rest Proxy | We have 1 rest proxy spread over 1 az. | We should have more rest proxy for high availability |
| Load balancers | We have total of 15 load balancers deployed. |  |

# Current Application Workflow:

OAUTH:

* Key Cloak AD is designated as a trusted entity within the Confluent platform, enabling accounts under Key Cloak AD with mapped access and Role-Based Access Controls (RBACs) to the Confluent platform.
* Customers initiate the process by obtaining a Key Cloak ID and Secret.
* The managed service then creates a Key Cloak AD group and a corresponding service account.
* A new folder is created in the code repository for configurations, also append customer Config information into a global config file.
* The subsequent step involves pipeline execution, where terraform is invoked to generate all the necessary resources within the platform.

MTLS:

* For customers opting for MTLS, the process involves ordering a Scania-signed certificate or a LetsEncrypt certificate, both trusted by the Confluent platform.
* Like the OAUTH flow, a Key Cloak AD group and service account are created, mapped to the Common Name (CN) of the certificate.
* Authentication is done using certificate. But for authorization, we create a service account which get mapped to CNAME
* A new folder is created in the code repository for configurations, also append customer Config information into a global config file.
* The pipeline execution, powered by terraform, proceeds to create the platform resources.
* An additional step involves a broker restart due to the mapping of the service account to the Canonical Name (CNAME).

# Challenges Encountered in Achieving Self-Service:

1. Kafka Broker Restart:
   * A challenge arises when MTLS is employed, necessitating a broker restart, a step incompatible with a self-service model.
2. Modular Access to GitHub:
   * Providing customers with modular GitHub access, safeguarding against inadvertent code changes affecting other customers, poses a challenge.
3. Entity Management Service Refactoring:
   * Refactoring the entity management service code to eliminate redundant customer information entries is essential for granting modular code access without affecting other customers.
4. AD Resource Creation Limitation:
   * A limitation exists where Scania cannot create Azure AD resources with Terraform, requiring customers to perform specific actions in the case of MTLS.
   * In Confluent platform we map an AD with tenant id and its creds, but this is owned by streaming team, we cannot give them write access, as this may create problems later.

# Feasibility of Self-Service with Current Setup:

* Not completely, self-service is partially viable with the current setup; however, a shift in our approach to customer onboarding is necessary.

# Staged Tasks Required for Implementation:

1. AD Group and Service Account :
   1. Customers are required to order an AD group with RBACs.
   2. Customers must create a service account and add it to the groups.
   3. The streaming platform team will provide a template for the above processes.
   4. In MTLS cases, customers should map the service account to the CN name of the certificate.
2. Customer Access to Code in GitLab:
   1. Customers, upon obtaining access, should create a dedicated folder and a file with configurations, with read-only access to other customers' files.
   2. The streaming platform team will provide a template for configurations.
   3. Adaptations to the pipeline code are necessary to update or append the owners.yaml file.
3. IP Whitelisting:
   1. Develop a script to whitelist IPs from which customer applications access the streaming platform.
4. Pull Request Process:
   1. Customers, after making changes, will create pull requests.
   2. The managed service will review and approve these pull requests.
5. Broker Restart:
   1. With the current Confluent platform version (7.1.1), a manual broker restart is required for MTLS customers.
   2. Upon the release of a new version that handles this automatically, the manual step will become obsolete.