IaC Confluent Cloud AWS Private Linking with Cluster Linking Example

```
flowchart TB
    subgraph CONFLUENT["Confluent Cloud"]
        subgraph ENV["Environment: non-prod"]
            subgraph PLATT["Private Link Attachment"]
                PLService["PrivateLink Service"]
                DNSDomain["DNS Domain"]
            end

            subgraph SANDBOX_CLUSTER["Sandbox Cluster - Enterprise"]
                SandboxKafka["Kafka Brokers"]
                SandboxTopic["dev-stock_trades Topic"]
                DatagenConnector["Datagen Connector"]
            end

            subgraph SHARED_CLUSTER["Shared Cluster - Enterprise"]
                SharedKafka["Kafka Brokers"]
                MirrorTopic["dev-stock_trades Mirror"]
            end

            subgraph CLUSTER_LINK["Bidirectional Cluster Link"]
                LinkSandboxShared["sandbox to shared"]
            end

            subgraph STREAM_GOV["Stream Governance"]
                SchemaRegistry["Schema Registry"]
            end
        end
    end

    DatagenConnector --> SandboxTopic
    SandboxTopic --> LinkSandboxShared
    LinkSandboxShared --> MirrorTopic

    subgraph AWS["AWS Cloud"]
        subgraph TGW["Transit Gateway"]
            TGWCore["TGW Core"]
            TGWRT["Route Table"]
        end

        subgraph DNS_VPC["DNS VPC - Centralized"]
            R53Inbound["Route53 Inbound Resolver"]
        end

        subgraph VPN_VPC["Client VPN VPC"]
            VPNEndpoint["Client VPN Endpoint"]
            VPNClients["VPN Clients"]
```

```
        end

        subgraph TFC_AGENT_VPC["TFC Agent VPC"]
            TFCAgents["Terraform Cloud Agents"]
        end

        subgraph SANDBOX_VPC["Sandbox PrivateLink VPC - 10.0.0.0/20"]
            SandboxSub1["Subnet AZ-1"]
            SandboxSub2["Subnet AZ-2"]
            SandboxSub3["Subnet AZ-3"]
            SandboxVPCE["VPC Endpoint"]
            SandboxSG["Security Group"]
        end

        subgraph SHARED_VPC["Shared PrivateLink VPC - 10.1.0.0/20"]
            SharedSub1["Subnet AZ-1"]
            SharedSub2["Subnet AZ-2"]
            SharedSub3["Subnet AZ-3"]
            SharedVPCE["VPC Endpoint"]
            SharedSG["Security Group"]
        end

        subgraph ROUTE53["Route53 DNS Configuration"]
            PHZ["Private Hosted Zone"]
            ZonalRecords["Zonal CNAME Records"]
            WildcardRecord["Wildcard CNAME"]
            SystemRule["SYSTEM Resolver Rule"]
        end

        SecretsManager["AWS Secrets Manager"]
    end

    SandboxVPCE --> SandboxSG
    SharedVPCE --> SharedSG
    PHZ --> ZonalRecords
    PHZ --> WildcardRecord

    SandboxVPCE -->|PrivateLink| PLService
    SharedVPCE -->|PrivateLink| PLService
    PLService --> SandboxKafka
    PLService --> SharedKafka

    SANDBOX_VPC -->|TGW Attachment| TGW
    SHARED_VPC -->|TGW Attachment| TGW
    DNS_VPC -->|TGW Attachment| TGW
    VPN_VPC -->|TGW Attachment| TGW
    TFC_AGENT_VPC -->|TGW Attachment| TGW

    TFCAgents -->|DNS Query| R53Inbound
    VPNClients -->|DNS Query| R53Inbound
    R53Inbound --> PHZ
    PHZ -->|Returns Endpoint IPs| SandboxVPCE
    PHZ -->|Returns Endpoint IPs| SharedVPCE
```

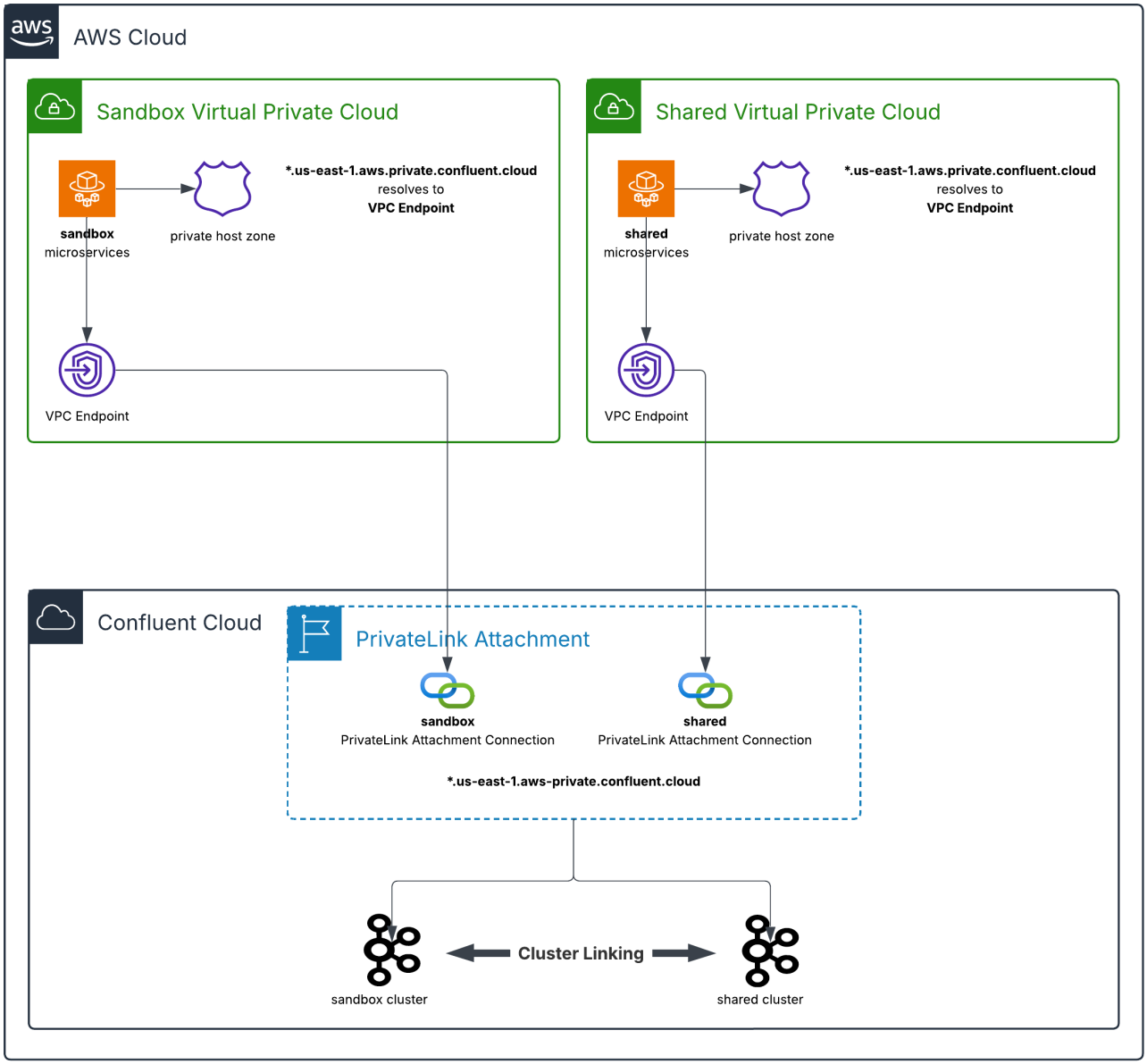
```
    PHZ -.->|Zone Association| TFC_AGENT_VPC
    PHZ -.->|Zone Association| DNS_VPC
    PHZ -.->|Zone Association| VPN_VPC
    PHZ -.->|Zone Association| SANDBOX_VPC
    PHZ -.->|Zone Association| SHARED_VPC

    SystemRule -.->|Rule Association| TFC_AGENT_VPC
    SystemRule -.->|Rule Association| DNS_VPC
    SystemRule -.->|Rule Association| VPN_VPC
    SystemRule -.->|Rule Association| SANDBOX_VPC
    SystemRule -.->|Rule Association| SHARED_VPC

    TFCAgents -->|Kafka 9092 via TGW| SandboxVPCE
    TFCAgents -->|Kafka 9092 via TGW| SharedVPCE
    VPNClients -->|Kafka 9092 via TGW| SandboxVPCE

    TFCAgents -.->|API Keys| SecretsManager

    %% Styling - High Contrast Colors
    style CONFLUENT fill:#1a1a2e,stroke:#e94560,stroke-
width:3px,color:#ffffff
    style ENV fill:#16213e,stroke:#e94560,stroke-width:2px,color:#ffffff
    style PLATT fill:#e94560,stroke:#ffffff,stroke-width:2px,color:#ffffff
    style SANDBOX_CLUSTER fill:#0f3460,stroke:#00d9ff,stroke-
width:2px,color:#ffffff
    style SHARED_CLUSTER fill:#0f3460,stroke:#00d9ff,stroke-
width:2px,color:#ffffff
    style CLUSTER_LINK fill:#533483,stroke:#e94560,stroke-
width:2px,color:#ffffff
    style STREAM_GOV fill:#0f3460,stroke:#00d9ff,stroke-
width:2px,color:#ffffff

    style AWS fill:#232f3e,stroke:#ff9900,stroke-width:3px,color:#ffffff
    style TGW fill:#ff9900,stroke:#232f3e,stroke-width:3px,color:#000000
    style DNS_VPC fill:#1b998b,stroke:#ffffff,stroke-
width:2px,color:#ffffff
    style VPN_VPC fill:#3066be,stroke:#ffffff,stroke-
width:2px,color:#ffffff
    style TFC_AGENT_VPC fill:#7209b7,stroke:#ffffff,stroke-
width:2px,color:#ffffff
    style SANDBOX_VPC fill:#2d6a4f,stroke:#95d5b2,stroke-
width:2px,color:#ffffff
    style SHARED_VPC fill:#2d6a4f,stroke:#95d5b2,stroke-
width:2px,color:#ffffff
    style ROUTE53 fill:#1b998b,stroke:#ffffff,stroke-
width:2px,color:#ffffff

    style PLService fill:#e94560,stroke:#ffffff,stroke-
width:2px,color:#ffffff
    style SandboxVPCE fill:#d62828,stroke:#ffffff,stroke-
width:2px,color:#ffffff
    style SharedVPCE fill:#d62828,stroke:#ffffff,stroke-
width:2px,color:#ffffff
    style TGWCore fill:#ff9900,stroke:#000000,stroke-
```

```
width:2px,color:#000000
    style TGWRT fill:#ff9900,stroke:#000000,stroke-width:2px,color:#000000
    style PHZ fill:#1b998b,stroke:#ffffff,stroke-width:2px,color:#ffffff
    style SecretsManager fill:#dd6b20,stroke:#ffffff,stroke-
width:2px,color:#ffffff
```

This repository provides **production-grade Terraform infrastructure-as-code** that implements a **secure, multi-network Confluent Cloud architecture**. It demonstrates **AWS PrivateLink connectivity from a single Confluent Cloud environment to multiple AWS VPCs**, enabling private, network-isolated access without exposing traffic to the public internet.

The solution also showcases **in-region Cluster Linking between two Confluent Cloud Kafka clusters**, enabling **low-latency, fully managed data replication** across teams, lines of business, or isolated environments (for example, development, staging, and production) within the same AWS region.

Cluster Linking maintains an **in-sync mirror of selected topics** on the consuming cluster. This isolation allows consuming teams to independently scale **large numbers of consumers, stream processing applications, and downstream sinks** without impacting the producing cluster. From the producer's perspective, the load is equivalent to **a single additional consumer**, regardless of downstream scale.



Access control and ownership remain cleanly separated: the producing team grants **scoped read credentials** to approved topics, while the consuming team **creates, owns, monitors, and manages the cluster link**. This pattern enables secure, scalable data sharing with clear operational boundaries and minimal coupling.

Below is the Terraform resource visualization of the infrastructure that's created:

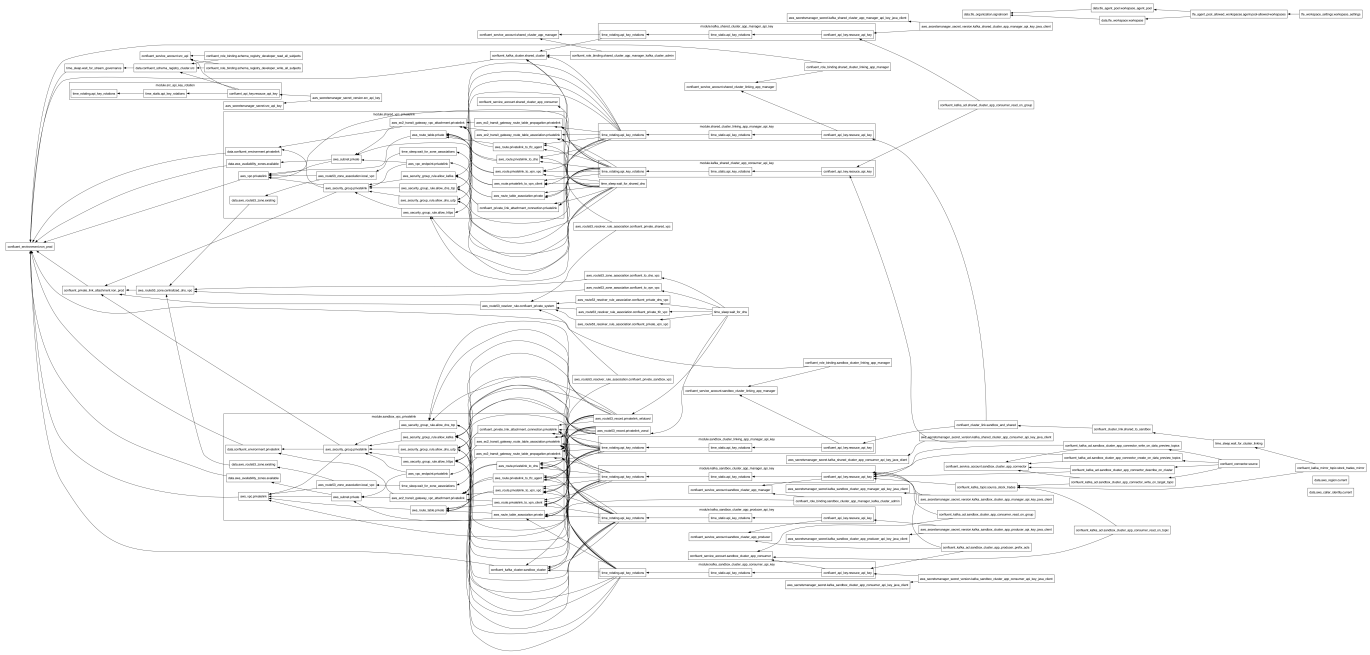


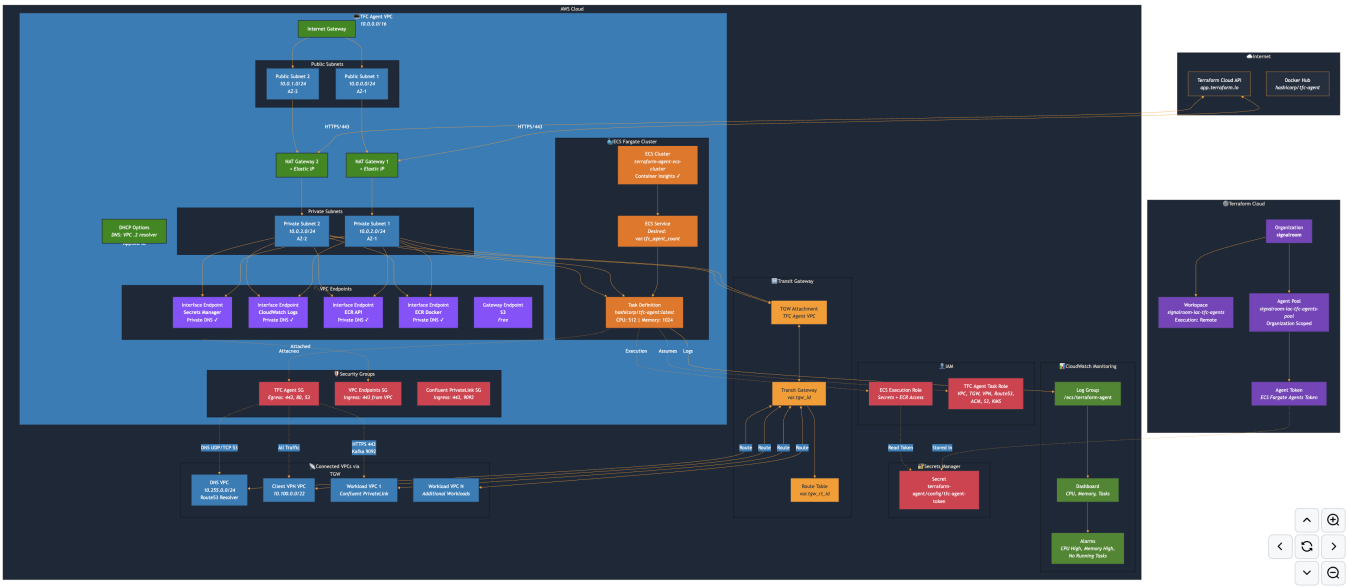
Table of Contents

1.0 Prerequisites

This project assumes you have the following prerequisites in place:

- Client VPN, Centralized DNS Server, and Transit Gateway
- Terraform Cloud Agent

1.1 Client VPN, Centralized DNS Server, and Transit Gateway



The diagram illustrates:

Core Components:

- **Transit Gateway Hub** — Central connectivity point with ASN 64512, DNS support, and VPN ECMP enabled. Includes main and additional route tables for traffic isolation
- **Client VPN VPC** — Hosts the VPN endpoint with mutual TLS authentication using ACM server/client certificates, with TGW attachment for routing

Connected VPCs:

- **Workload VPCs** — Multiple VPCs with distinct CIDR ranges connected via TGW attachments
- **TFC Agent VPC** — Dedicated VPC for Terraform Cloud agents (local execution mode)
- **DNS VPC** — Centralized Route 53 resolver for cross-VPC DNS resolution

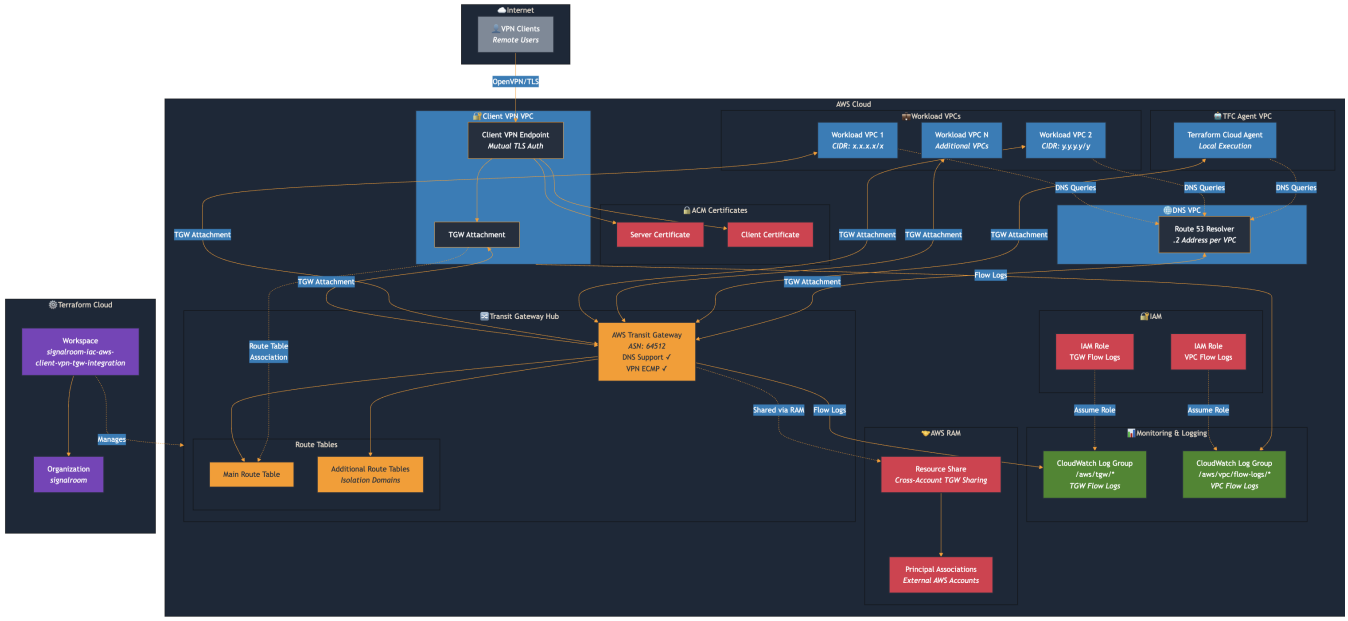
Supporting Services:

- **AWS RAM** — Cross-account sharing of the Transit Gateway to external AWS accounts
- **CloudWatch** — Flow logs for both TGW and VPC traffic with dedicated IAM roles
- **Terraform Cloud** — signalRoom organization managing the infrastructure via the `signalroom-iac-aws-client-vpn-tgw-integration` workspace

Traffic Flow:

1. Remote VPN clients connect via OpenVPN/TLS to the Client VPN endpoint
2. Traffic routes through the TGW to reach workload VPCs, TFC agents, or DNS services
3. VPN client traffic routes back through the TGW via the dedicated route table association

1.2 Terraform Cloud Agent



The diagram illustrates:

Terraform Cloud Integration:

- **signalroom** organization with remote execution workspace
- **Agent Pool** (organization-scoped) with token stored in AWS Secrets Manager
- Agents poll Terraform Cloud API via NAT Gateway

TFC Agent VPC (10.0.0.0/16):

- **Public Subnets** — Host NAT Gateways with Elastic IPs for outbound internet
- **Private Subnets** — Run ECS Fargate tasks (no public IPs)
- **DHCP Options** — Configure DNS to use VPC's .2 resolver

ECS Fargate Cluster:

- Runs `hashicorp/tfc-agent:latest` containers (512 CPU / 1024 MB)
- Container Insights enabled for monitoring
- Circuit breaker with auto-rollback on deployment failures

VPC Endpoints (PrivateLink):

- **Secrets Manager** — Retrieve TFC agent token privately
- **CloudWatch Logs** — Stream logs without NAT
- **ECR API + Docker** — Pull container images privately
- **S3 Gateway** — Free endpoint for ECR layer storage

Transit Gateway Connectivity:

- Routes to **DNS VPC** (10.255.0.0/24) for Confluent domain resolution
- Routes to **Client VPN VPC** (10.100.0.0/22) for admin access
- Routes to **Workload VPCs** for Confluent PrivateLink (ports 443, 9092)

Security Groups:

- **TFC Agent SG** — Outbound HTTPS/HTTP/DNS to internet + specific VPC routes
- **VPC Endpoints SG** — Inbound 443 from VPC CIDR
- **Confluent PrivateLink SG** — Inbound 443/9092 from agents

2.0 Project's Architecture Overview

Key Features Required for Confluent PrivateLink to Work (Confluent Cloud Configuration)

1. Confluent Private Link Attachment (Environment-Level)

- Single `confluent_private_link_attachment` resource created at the environment level for AWS region
- Provides the `vpc_endpoint_service_name` that AWS VPC Endpoints connect to
- Provides the `dns_domain` (e.g., `*.aws.private.confluent.cloud`) for DNS configuration
- Multiple VPCs can share the same PrivateLink attachment via separate VPC Endpoints

2. AWS VPC Endpoint Configuration

- Interface VPC Endpoints (`vpc_endpoint_type = "Interface"`) in each workload VPC
- **Critical**: `private_dns_enabled = false` — DNS handled via centralized Private Hosted Zones instead
- Security groups allowing inbound on ports 443 (HTTPS), 9092 (Kafka), and 53 (DNS) from TFC Agent VPC, VPN VPC, VPN Client CIDR, and local VPC CIDR
- Endpoints deployed across multiple AZs (3 subnets) for high availability

3. Confluent Private Link Attachment Connection

- `confluent_private_link_attachment_connection` links the AWS VPC Endpoint ID to the Confluent PrivateLink attachment
- Creates the bidirectional connection between AWS and Confluent Cloud
- Depends on Route53 zone associations being complete first (`time_sleep` for propagation)

4. Centralized Private Hosted Zone (PHZ) Strategy

- Single PHZ created for the Confluent DNS domain, associated with **all VPCs** that need access
- **Zonal CNAME records**: `*.{availability-zone-id}.{dns_domain}` → AZ-specific VPC Endpoint DNS
- **Wildcard CNAME record**: `*.{dns_domain}` → Primary VPC Endpoint DNS

5. Route53 SYSTEM Resolver Rule

- `rule_type = "SYSTEM"` tells Route53 to use Private Hosted Zones for the Confluent domain
- Rule associated with every VPC that needs Confluent access

6. Transit Gateway Routing

- Each PrivateLink VPC attached to TGW with DNS support enabled
- Route table association AND route propagation configured
- Routes added from PrivateLink VPCs back to all consumer VPCs

7. Multi-Cluster Architecture with Cluster Linking

- Two Enterprise Kafka clusters (Sandbox and Shared) in the same environment
- Bidirectional Cluster Link with mirror topics for data replication

8. Service Account & API Key Management

- Separate service accounts per role with API key rotation
- ACLs granting specific permissions per service account
- API keys stored in AWS Secrets Manager

9. DNS Propagation Timing

- `time_sleep` resources ensuring DNS propagates before dependent resources (1-2 minutes)

10. Schema Registry Integration

- Stream Governance (Essentials) enabled at environment level with AVRO support

3.0 Let's Get Started

3.1 Deploying the Infrastructure

3.2 Destroying the Infrastructure

```
|
| Error: error deleting Kafka ACLs "lkc-
j6wj9w/TOPIC#sandbox_aws_privatelink_example_#LITERAL#User:sa-
w7xo5n9#*#CREATE#ALLOW": Delete "https://lkc-j6wj9w.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-j6wj9w/acls?
host=%2A&operation=CREATE&pattern_type=LITERAL&permission=ALLOW&principal=
User%3Asa-
w7xo5n9&resource_name=sandbox_aws_privatelink_example_&resource_type=TOPIC
": dial tcp: lookup lkc-j6wj9w.us-east-1.aws.private.confluent.cloud on
10.2.0.2:53: no such host
|
|
|

|
| Error: error deleting Kafka ACLs "lkc-
j6wj9w/TOPIC#sandbox_aws_privatelink_example_#LITERAL#User:sa-
w7xo5n9#*#WRITE#ALLOW": Delete "https://lkc-j6wj9w.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-j6wj9w/acls?
host=%2A&operation=WRITE&pattern_type=LITERAL&permission=ALLOW&principal=U
ser%3Asa-
w7xo5n9&resource_name=sandbox_aws_privatelink_example_&resource_type=TOPIC
": dial tcp: lookup lkc-j6wj9w.us-east-1.aws.private.confluent.cloud on
10.2.0.2:53: no such host
|
|
|

|
```

```
│ Error: error deleting Kafka ACLs "lkc-j6wj9w/CLUSTER#kafka-
cluster#LITERAL#User:sa-w7xo5n9#*#DESCRIBE#ALLOW": Delete "https://lkc-
j6wj9w.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
j6wj9w/acls?
host=%2A&operation=DESCRIBE&pattern_type=LITERAL&permission=ALLOW&principa
l=User%3Asa-w7xo5n9&resource_name=kafka-cluster&resource_type=CLUSTER":
dial tcp: lookup lkc-j6wj9w.us-east-1.aws.private.confluent.cloud on
10.2.0.2:53: no such host
│
│
╵

╷
│ Error: error waiting for Kafka Mirror Topic "lkc-99gmp5/bidirectional-
between-sandbox-and-shared/dev-stock_trades" to be deleted: Get
"https://lkc-99gmp5.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
99gmp5/links/bidirectional-between-sandbox-and-shared/mirrors/dev-
stock_trades": dial tcp: lookup lkc-99gmp5.us-east-
1.aws.private.confluent.cloud on 10.2.0.2:53: no such host; could not
parse error details; raw response body: ""
│
│
╵

╷
│ Error: error deleting Kafka ACLs "lkc-j6wj9w/TOPIC#dev-
stock_trades#LITERAL#User:sa-w7xo5n9#*#WRITE#ALLOW": Delete "https://lkc-
j6wj9w.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
j6wj9w/acls?
host=%2A&operation=WRITE&pattern_type=LITERAL&permission=ALLOW&principal=U
ser%3Asa-w7xo5n9&resource_name=dev-stock_trades&resource_type=TOPIC": dial
tcp: lookup lkc-j6wj9w.us-east-1.aws.private.confluent.cloud on
10.2.0.2:53: no such host
│
│
╵
Operation failed: failed running terraform apply (exit 1)
```

If you encounter DNS resolution errors during the destroy process, do the following:

Navigate to the Terraform directory:

```
cd terraform
```

***Remove the unreachable resources from the Terraform state:**

```
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_connector_describe_on_cluster'
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_connector_write_on_target_topic'
```

```
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_connector_create_on_data_preview_
topics'
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_connector_write_on_data_preview_t
opics'
terraform state rm 'confluent_cluster_link.sandbox_and_shared'
terraform state rm 'confluent_cluster_link.shared_to_sandbox'
terraform state rm 'confluent_kafka_topic.source_stock_trades'
terraform state rm 'confluent_kafka_mirror_topic.stock_trades_mirror'
```

Navigate back to the root directory:

```
cd ..
```

Rerun the destroy command:

```
```

4.0 Resources

4.1 Terminology

- **PHZ**: Private Hosted Zone - AWS Route 53 Private Hosted Zone is a DNS service that allows you to create and manage private DNS zones within your VPCs.
- **TFC**: Terraform Cloud - A service that provides infrastructure automation using Terraform.
- **VPC**: Virtual Private Cloud - A virtual network dedicated to your AWS account.
- **AWS**: Amazon Web Services - A comprehensive cloud computing platform provided by Amazon.
- **CC**: Confluent Cloud - A fully managed event streaming platform based on Apache Kafka.
- **PL**: PrivateLink - An AWS service that enables private connectivity between VPCs and services.
- **IaC**: Infrastructure as Code - The practice of managing and provisioning computing infrastructure through machine-readable definition files.

4.2 Related Documentation

- AWS PrivateLink Overview in Confluent Cloud
- Use AWS PrivateLink for Serverless Products on Confluent Cloud
- GitHub Sample Project for Confluent Terraform Provider PrivateLink Attachment
- Geo-replication with Cluster Linking on Confluent Cloud
- Use the Confluent Cloud Console with Private Networking
- IP Filtering on Confluent Cloud
- AWS/Azure PrivateLink Networking Course
- Hands On: Configuring a PrivateLink Cluster