

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"]
                SANDBOX_KAFKA["Kafka Brokers"]
                SANDBOX_TOPIC["dev-stock_trades Topic"]
                DATAGEN_CONNECTOR["Datagen Connector"]
            end

            subgraph SHARED_CLUSTER["Shared Cluster – Enterprise"]
                SHARED_KAFKA["Kafka Brokers"]
                MIRROR_TOPIC["dev-stock_trades Mirror"]
            end

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

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

    DATAGEN_CONNECTOR --> SANDBOX_TOPIC
    SANDBOX_TOPIC --> LINK_SANDBOX_SHARED
    LINK_SANDBOX_SHARED --> MIRROR_TOPIC

    subgraph AWS["AWS Cloud"]
        subgraph TGW["Transit Gateway"]
            TGW_CORE["TGW Core"]
            TGW_RT["Route Table"]
        end

        subgraph DNS_VPC["DNS VPC – Centralized"]
            R53_INBOUND["Route53 Inbound Resolver"]
        end

        subgraph VPN_VPC["Client VPN VPC"]
            VPN_ENDPOINT["Client VPN Endpoint"]
            VPN_CLIENTS["VPN Clients"]
        end
    end

```

```

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"]

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
```

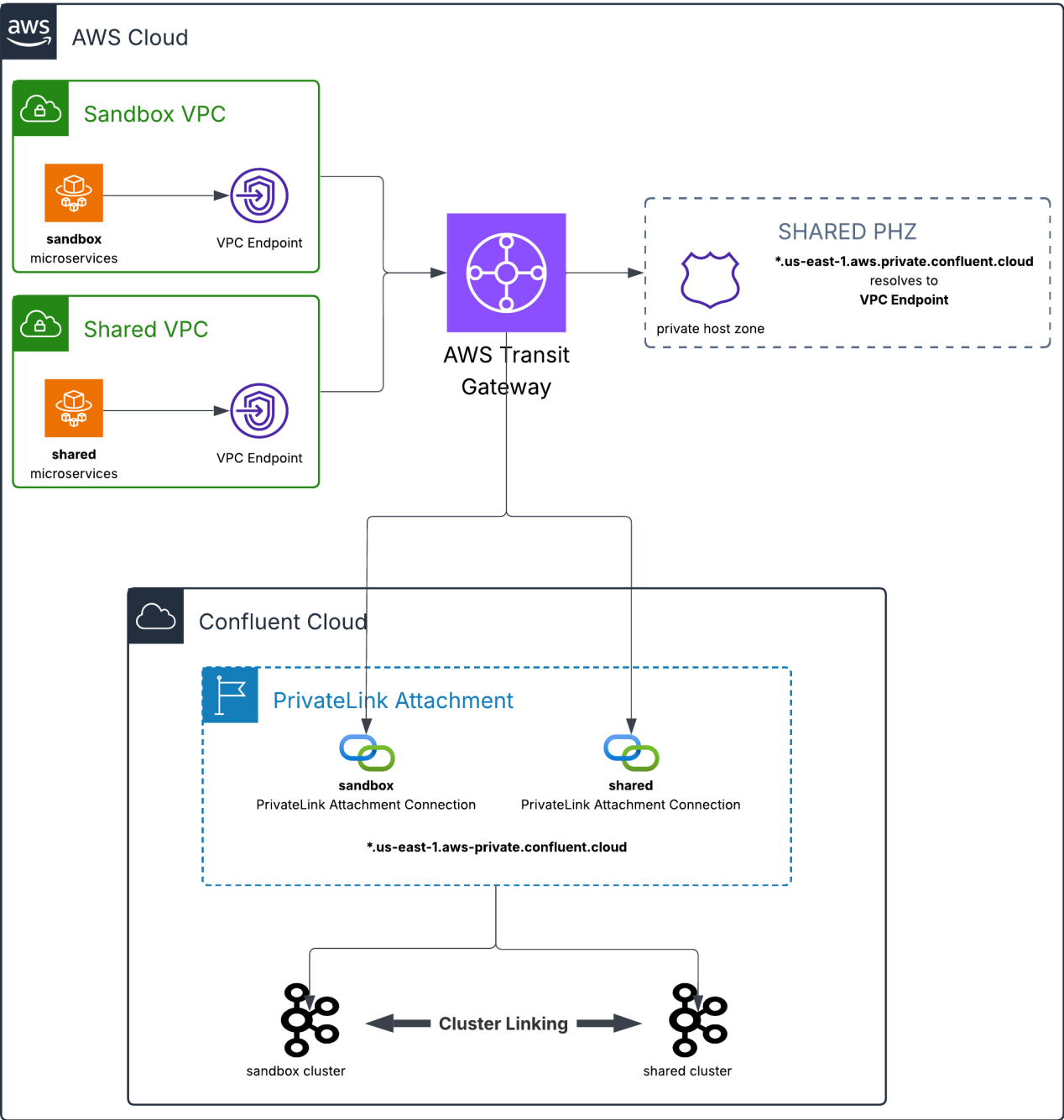
```
style PLService fill:#e94560,stroke:#ffffff,stroke-
width:2px,color:#ffffff
style SandboxVPCE fill:#d62828,stroke:#ffffff,stroke-
width:2px,color:#ffffff
style SharedVPCE fill:#d62828,stroke:#ffffff,stroke-
width:2px,color:#ffffff
style TGWCore fill:#ff9900,stroke:#000000,stroke-
```

```
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:

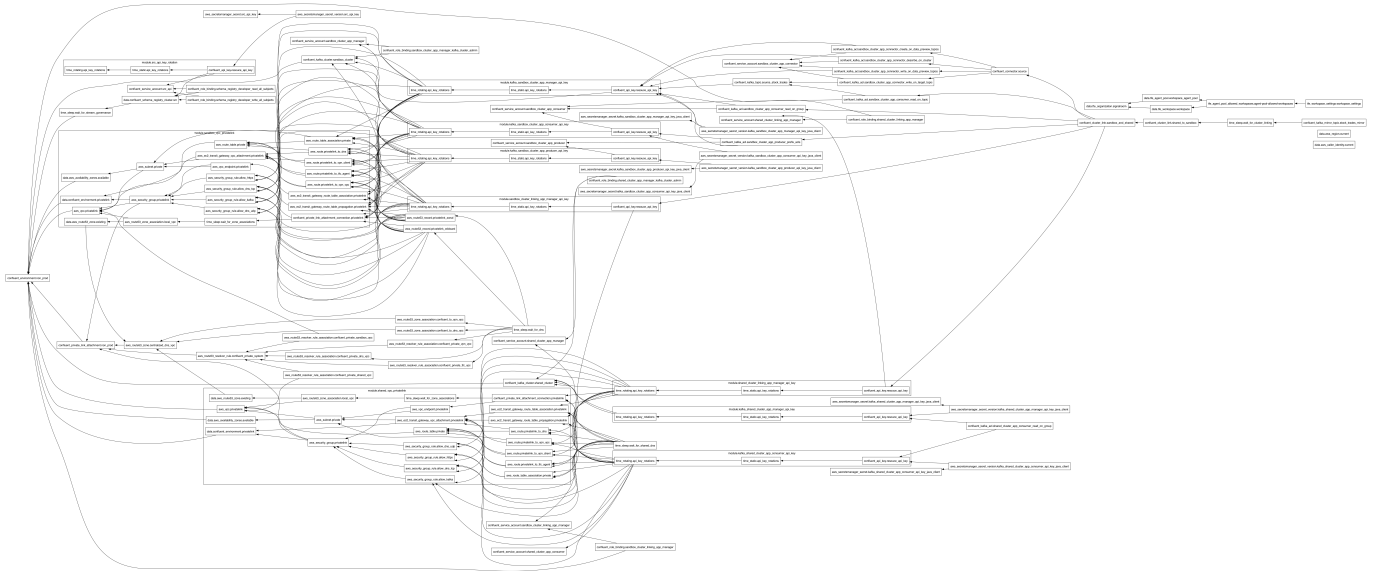


Table of Contents

- **1.0 Prerequisites**
 - **1.1 Client VPN, Centralized DNS Server, and Transit Gateway**
 - **1.1.1 Key Features Required for Confluent PrivateLink to Work**
 - **1.1.1.1 Hub-and-Spoke Network Architecture via Transit Gateway**
 - **1.1.1.2 Centralized DNS Resolution (Critical for PrivateLink)**
 - **1.1.1.3 DNS Forwarding Chain**
 - **1.1.1.4 VPC Endpoints (AWS PrivateLink)**
 - **1.1.1.5 Client VPN Integration**
 - **1.1.1.6 Cross-VPC Routing**
 - **1.1.1.7 Security & Observability**
 - **1.2 Terraform Cloud Agent**
 - **1.2.1 Key Features of the TFC Agent Setup**
 - **1.2.1.1 Custom DHCP Options for DNS Resolution**
 - **1.2.1.2 Transit Gateway Connectivity**
 - **1.2.1.3 Security Group Configuration for Kafka/PrivateLink Traffic**
 - **1.2.1.4 AWS VPC Endpoints for Private Service Access**
 - **1.2.1.5 ECS Fargate Deployment Pattern**
 - **1.2.1.6 IAM Permissions for Infrastructure Management**
 - **1.2.1.7 Network Architecture Summary**
- **2.0 Project's Architecture Overview**
 - **2.1 Key Architecture Components**
 - **2.1.1 Confluent Private Link Attachment (Environment-Level)**
 - **2.1.2 AWS VPC Endpoint Configuration**
 - **2.1.3 Confluent Private Link Attachment Connection**
 - **2.1.4 Centralized Private Hosted Zone (PHZ) Strategy**
 - **2.1.5 Route53 SYSTEM Resolver Rule**
 - **2.1.6 Transit Gateway Routing**
 - **2.1.7 Multi-Cluster Architecture with Cluster Linking**
 - **2.1.8 Service Account & API Key Management**
 - **2.1.9 DNS Propagation Timing**
 - **2.1.10 Schema Registry Integration**

- **3.0 Let's Get Started**
 - **3.1 Deploy the Infrastructure**
 - **3.2 Teardown the Infrastructure**
- **4.0 References**
 - **4.1 Terminology**
 - **4.2 Related Documentation**

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

```
%%{init: {'theme': 'base', 'themeVariables': { 'primaryColor': '#1a73e8',
'primaryTextColor': '#fff', 'primaryBorderColor': '#1557b0', 'lineColor':
'#5f6368', 'secondaryColor': '#34a853', 'tertiaryColor': '#fbbc04'}}}%
```

```
flowchart TB
```

```
    subgraph USERS["👤 Remote Users"]
        VPNClient["VPN Client
(OpenVPN/AWS Client)"]
    end

    subgraph AWS["☁ AWS Cloud"]
        subgraph VPN_VPC["Client VPN VPC
var.vpn_vpc_cidr"]
            VPNEndpoint["AWS Client VPN
Endpoint"]
            VPNSubnets["VPN Subnets
(Multi-AZ)"]
            VPNSG["Security Group
client-vpn-sg"]
            VPNResolver["Route53 Outbound
Resolver Endpoint"]
            VPNEndpoint --> VPNSubnets
            VPNSubnets --> VPNSG
            VPNSubnets --> VPNResolver
        end

        subgraph TGW["Transit Gateway
signalroom-tgw"]
            TGWCore["TGW Core
ASN: 64512"]
            TGWRouteTable["Custom Route
Tables"]
            TGWCore --> TGWRouteTable
        end
    end
```

```

        subgraph DNS_VPC["DNS VPC (Centralized)"]
var.dns_vpc_cidr"]
            R53Inbound["Route53 Inbound
Resolver Endpoint"]
            R53PHZ["Private Hosted Zones
*.aws.confluent.cloud"]
            R53Inbound --> R53PHZ
        end

        subgraph TFC_VPC["TFC Agent VPC"]
var.tfc_agent_vpc_cidr"]
            TFCAgent["Terraform Cloud
Agent"]
        end

        subgraph WORKLOAD_VPCs["Workload VPCs"]
            subgraph WL1["Workload VPC 1"]
                VPCE1["VPC Endpoint
(PrivateLink)"]
            end
            subgraph WL2["Workload VPC N..."]
                VPCEN["VPC Endpoint
(PrivateLink)"]
            end
        end

        ACM["ACM Certificates
(Server & Client)"]
        CWLogs["CloudWatch Logs
VPN & Flow Logs"]

        subgraph CONFLUENT["▲ Confluent Cloud"]
            PrivateLinkService["PrivateLink Service
Endpoint"]
            Kafka["Kafka Cluster
(Private)"]
            PrivateLinkService --> Kafka
        end

        %% Connections
        VPNClient -->|"Mutual TLS
Authentication"| VPNEndpoint
        ACM -->|"Certificate Auth"| VPNEndpoint

        VPN_VPC -->|"TGW Attachment"| TGW
        DNS_VPC -->|"TGW Attachment"| TGW
        TFC_VPC -->|"TGW Attachment"| TGW
        WL1 -->|"TGW Attachment"| TGW
        WL2 -->|"TGW Attachment"| TGW

        VPNResolver -->|"DNS Forwarding
Rule"| R53Inbound
        R53PHZ -->|"Returns Private

```


Endpoint IPs"| VPCE1

```
VPCE1 -->|"AWS PrivateLink"| PrivateLinkService
VPCEN -->|"AWS PrivateLink"| PrivateLinkService
```

```
VPNEndpoint -. ->|"Logs"| CWLogs
TGW -. ->|"Flow Logs"| CWLogs
```

%% Styling

```
classDef userStyle fill:#4285f4,stroke:#1557b0,stroke-
width:2px,color:#fff
classDef vpcStyle fill:#e8f0fe,stroke:#1a73e8,stroke-width:2px
classDef tgwStyle fill:#fef7e0,stroke:#f9ab00,stroke-width:3px
classDef dnsStyle fill:#e6f4ea,stroke:#34a853,stroke-width:2px
classDef confluentStyle fill:#f3e8fd,stroke:#9334e6,stroke-width:2px
classDef serviceStyle fill:#fff,stroke:#5f6368,stroke-width:1px

class USERS userStyle
class VPN_VPC,TFC_VPC,WORKLOAD_VPCs,WL1,WL2 vpcStyle
class TGW tgwStyle
class DNS_VPC dnsStyle
class CONFLUENT confluentStyle
```

1.1.1 Key Features Required for Confluent PrivateLink to Work

1.1.1.1 Hub-and-Spoke Network Architecture via Transit Gateway

- Transit Gateway serves as the central routing hub connecting all VPCs
- Disabled default route table association/propagation for explicit routing control
- DNS support enabled on the TGW (`dns_support = "enable"`)
- Custom route tables for fine-grained traffic control between VPCs

1.1.1.2 Centralized DNS Resolution (Critical for PrivateLink)

- **Dedicated DNS VPC** with Route53 Inbound Resolver endpoints
- **Private Hosted Zones** for `*.aws.confluent.cloud` domain
- DNS forwarding rules route Confluent queries from all VPCs to the central DNS VPC
- Route53 Outbound Resolver in VPN VPC forwards to DNS VPC resolver IPs

1.1.1.3 DNS Forwarding Chain (as documented in your outputs)

1. Client queries `lkc-xxxxx.us-east-1.aws.private.confluent.cloud`
2. VPN VPC's default DNS forwards to Route53 Outbound Resolver
3. Outbound Resolver forwards to DNS VPC Inbound Resolver
4. DNS VPC checks Private Hosted Zones → returns VPC Endpoint private IPs

1.1.1.4 VPC Endpoints (AWS PrivateLink)

- VPC Endpoints in workload VPCs connecting to Confluent's PrivateLink service

- Security groups allowing traffic from authorized sources (VPN clients, TFC agents)

1.1.1.5 Client VPN Integration

- Mutual TLS authentication using ACM certificates (server + client)
- Split tunnel configuration for routing only Confluent traffic through VPN
- Authorization rules controlling which CIDRs VPN clients can access
- Routes added to VPN endpoint for all workload VPC CIDRs via Transit Gateway

1.1.1.6 Cross-VPC Routing

- TGW attachments for: VPN VPC, DNS VPC, TFC Agent VPC, and all Workload VPCs
- Route tables in each VPC with routes to other VPCs via TGW
- Workload VPC CIDRs aggregated and distributed to VPN client routes

1.1.1.7 Security & Observability

- Dedicated security groups per component (VPN endpoint, etc.)
- VPC Flow Logs and TGW Flow Logs to CloudWatch
- VPN connection logging for audit trails
- IAM roles with least-privilege for flow log delivery

1.2 Terraform Cloud Agent

```
%%{init: {'theme': 'base', 'themeVariables': { 'primaryColor': '#1a73e8',
'primaryTextColor': '#fff', 'primaryBorderColor': '#1557b0', 'lineColor':
'#5f6368', 'secondaryColor': '#34a853', 'tertiaryColor': '#fbbc04'}}}%
```

```
flowchart TB
```

```
    subgraph TERRAFORM_CLOUD["└ Terraform Cloud (HCP)"]
        TFC["Terraform Cloud
API & Workspaces"]
        AgentPool["Agent Pool
(signalroom)"]
    end
```

```
    subgraph AWS["└ AWS Cloud"]
        subgraph TFC_AGENT_VPC["TFC Agent VPC
var.vpc_cidr"]
            subgraph PUBLIC_SUBNETS["Public Subnets (Multi-AZ)"]
                IGW["Internet
Gateway"]
                NAT1["NAT Gateway
AZ-1"]
                NAT2["NAT Gateway
AZ-2"]
            end
        end
```

```
        subgraph PRIVATE_SUBNETS["Private Subnets (Multi-AZ)"]
            subgraph ECS["ECS Fargate Cluster"]
```

```

        TFCAgent1["TFC Agent
Container"]
        TFCAgent2["TFC Agent
Container"]
    end

    subgraph AWS_ENDPOINTS["AWS VPC Endpoints"]
        VPCE_SM["Secrets Manager
Endpoint"]
        VPCE_CW["CloudWatch Logs
Endpoint"]
        VPCE_ECR["ECR API/DKR
Endpoints"]
        VPCE_S3["S3 Gateway
Endpoint"]
    end

    CONFLUENT_SG["Confluent PrivateLink
Security Group"]
    end

    DHCP["DHCP Options
(Custom DNS)"]
    TFC_AGENT_SG["TFC Agent
Security Group"]
    end

    subgraph TGW["Transit Gateway
signalroom-tgw"]
        TGWCore["TGW Core"]
        TGWRT["Route Table"]
    end

    subgraph DNS_VPC["DNS VPC (Centralized)
var.dns_vpc_cidr"]
        R53Inbound["Route53 Inbound
Resolver"]
        PHZ["Private Hosted Zones
*.aws.confluent.cloud"]
    end

    subgraph CLIENT_VPN_VPC["Client VPN VPC
var.client_vpn_vpc_cidr"]
        VPNEndpoint["Client VPN
Endpoint"]
    end

    subgraph WORKLOAD_VPCs["Workload VPCs
(Confluent PrivateLink)"]
        subgraph WL1["Workload VPC 1"]
            VPCE1["PrivateLink
Endpoint"]
        end
        subgraph WL2["Workload VPC N"]

```

```

        VPCEN["PrivateLink
Endpoint"]
        end
    end

    SecretsManager["AWS Secrets Manager
(TFC Agent Token)"]
    CloudWatch["CloudWatch Logs"]
    ECR_Registry["ECR Registry
(hashicorp/tfc-agent)"]
    end

    subgraph CONFLUENT["▲ Confluent Cloud"]
        PrivateLinkSvc["PrivateLink
Service"]
        Kafka["Kafka Cluster
(Private)"]
    end

    %% External Connections
    TFC <-->|"HTTPS/443
via NAT"| TFCAgent1
    TFC <-->|"HTTPS/443
via NAT"| TFCAgent2
    AgentPool -.->|"Agent Registration"| TFCAgent1

    %% Internal VPC Connections
    TFCAgent1 --> TFC_AGENT_SG
    TFCAgent2 --> TFC_AGENT_SG
    TFCAgent1 --> VPCE_SM
    TFCAgent2 --> VPCE_CW

    VPCE_SM -.->|"Private DNS"| SecretsManager
    VPCE_CW -.->|"Private DNS"| CloudWatch
    VPCE_ECR -.->|"Private DNS"| ECR_Registry

    NAT1 --> IGW
    NAT2 --> IGW
    TFCAgent1 -->|"0.0.0.0/0"| NAT1
    TFCAgent2 -->|"0.0.0.0/0"| NAT2

    %% DHCP & DNS Flow
    DHCP -->|"DNS Servers:
VPC + Centralized"| TFCAgent1
    TFCAgent1 -->|"DNS Query:
*.confluent.cloud"| R53Inbound

    %% Transit Gateway Connections
    TFC_AGENT_VPC -->|"TGW Attachment"| TGW
    DNS_VPC -->|"TGW Attachment"| TGW
    CLIENT_VPN_VPC -->|"TGW Attachment"| TGW
    WL1 -->|"TGW Attachment"| TGW
    WL2 -->|"TGW Attachment"| TGW

```

```

%% Route Propagation
TGWCore --> TGWRT

%% DNS Resolution
R53Inbound --> PHZ
PHZ -->|"Returns Private IPs"| VPCE1

%% PrivateLink Connections
VPCE1 -->|"AWS PrivateLink"| PrivateLinkSvc
VPCEN -->|"AWS PrivateLink"| PrivateLinkSvc
PrivateLinkSvc --> Kafka

%% TFC Agent to Workload VPCs
TFC_AGENT_SG -->|"HTTPS/443
Kafka/9092"| CONFLUENT_SG
CONFLUENT_SG -->|"via TGW"| VPCE1
CONFLUENT_SG -->|"via TGW"| VPCEN

%% Styling
classDef tfcStyle fill:#5c4ee5,stroke:#3d32a8,stroke-
width:2px,color:#fff
classDef vpcStyle fill:#e8f0fe,stroke:#1a73e8,stroke-width:2px
classDef tgwStyle fill:#fef7e0,stroke:#f9ab00,stroke-width:3px
classDef dnsStyle fill:#e6f4ea,stroke:#34a853,stroke-width:2px
classDef confluentStyle fill:#f3e8fd,stroke:#9334e6,stroke-width:2px
classDef endpointStyle fill:#fce8e6,stroke:#ea4335,stroke-width:1px
classDef ecsStyle fill:#fff3e0,stroke:#ff9800,stroke-width:2px

class TERRAFORM_CLOUD tfcStyle
class TFC_AGENT_VPC,CLIENT_VPN_VPC,WORKLOAD_VPCs,WL1,WL2 vpcStyle
class TGW tgwStyle
class DNS_VPC dnsStyle
class CONFLUENT confluentStyle
class AWS_ENDPOINTS,VPCE_SM,VPCE_CW,VPCE_ECR,VPCE_S3 endpointStyle
class ECS ecsStyle

```

1.2.1 Key Features Required for Confluent PrivateLink to Work (TFC Agent Configuration)

1.2.1.1 Custom DHCP Options for DNS Resolution

- DHCP Options Set configured with **dual DNS servers**: VPC default DNS (`cidrhost(vpc_cidr, 2)`) AND centralized DNS VPC resolver IPs
- Region-aware domain name configuration (`ec2.internal` for us-east-1, `{region}.compute.internal` for others)
- Associates TFC Agent VPC with custom DHCP options to route Confluent domain queries to the central DNS infrastructure

1.2.1.2 Transit Gateway Connectivity

- TFC Agent VPC attached to shared Transit Gateway with DNS support enabled
- Explicit route table association and route propagation (not using TGW defaults)

- Routes added from private subnets to: DNS VPC, Client VPN VPC, and all Workload VPCs containing PrivateLink endpoints
- Flattened route map pattern (`for_each`) ensures routes are created for every workload VPC CIDR

1.2.1.3 Security Group Configuration for Kafka/PrivateLink Traffic

- **TFC Agent Security Group** with egress rules for:
 - HTTPS (443) and Kafka (9092) to each workload VPC CIDR
 - DNS (UDP/TCP 53) to DNS VPC CIDR specifically
 - General HTTPS/HTTP for Terraform Cloud API and package downloads
- **Confluent PrivateLink Security Group** allowing inbound from TFC Agent SG on ports 443 and 9092

1.2.1.4 AWS VPC Endpoints for Private Service Access

- **Interface endpoints** with private DNS enabled for: Secrets Manager, CloudWatch Logs, ECR API, ECR DKR
- **S3 Gateway endpoint** (required for ECR image layer pulls)
- Dedicated security group for VPC endpoints allowing HTTPS from within VPC
- Eliminates NAT Gateway dependency for AWS service calls

1.2.1.5 ECS Fargate Deployment Pattern

- TFC Agents run in private subnets with `assign_public_ip = false`
- NAT Gateways per AZ for outbound internet (Terraform Cloud API communication)
- Agent token stored in Secrets Manager, fetched via VPC Endpoint
- Container health checks and deployment circuit breaker for reliability

1.2.1.6 IAM Permissions for Infrastructure Management

- Task role with Transit Gateway, VPC, Route53 Resolver, and Client VPN management permissions
- Execution role with Secrets Manager access for agent token retrieval
- KMS permissions scoped to Secrets Manager service for encryption/decryption

1.2.1.7 Network Architecture Summary

- **Hub-and-spoke model:** TGW connects TFC Agent VPC → DNS VPC → Workload VPCs
- **DNS resolution chain:** TFC Agent → Custom DHCP → Centralized DNS VPC → Private Hosted Zones → PrivateLink Endpoint IPs
- **Traffic flow:** TFC Agent → TGW → Workload VPC → PrivateLink Endpoint → Confluent Cloud Kafka

2.0 Project's Architecture Overview

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

2.1.1 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

2.1.2 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

2.1.3 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)

2.1.4 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

2.1.5 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

2.1.6 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

2.1.7 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

2.1.8 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

2.1.9 DNS Propagation Timing

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

2.1.10 Schema Registry Integration

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

3.0 Let's Get Started

3.1 Deploy the Infrastructure

The `deploy.sh` script handles authentication and Terraform execution:

```
./deploy.sh create \
  --profile=<SSO_PROFILE_NAME> \
  --confluent-api-key=<CONFLUENT_API_KEY> \
  --confluent-api-secret=<CONFLUENT_API_SECRET> \
  --tfe-token=<TFE_TOKEN> \
  --tgw-id=<TGW_ID> \
  --tgw-rt-id=<TGW_RT_ID> \
  --tfc-agent-vpc-id=<TFC_AGENT_VPC_ID> \
  --tfc-agent-vpc-cidr=<TFC_AGENT_VPC_CIDR> \
  --dns-vpc-id=<DNS_VPC_ID> \
  --vpn-vpc-id=<VPN_VPC_ID> \
  --vpn-vpc-cidr=<VPN_VPC_CIDR> \
  --vpn-client-vpc-cidr=<VPN_CLIENT_VPC_CIDR> \
  [--dns-vpc-cidr=<DNS_VPC_CIDR>] \      # Default: 10.255.0.0/24
  [--day-count=<DAY_COUNT>]             # Default: 30 (API key rotation
interval)
```

Handling DNS Resolution Errors: If you encounter DNS resolution errors during the apply process, simply re-run the `deploy.sh` script with the `create` command.

Cluster Linking Error:

```
| Error: error creating Cluster Link: 400 Bad Request: A cluster link
| already exists with the provided link name: Cluster Link
|_fA8DRTZSvGrLkTur7e8-Q already exists.
|
|   with confluent_cluster_link.shared_to_sandbox,
|   on setup-confluent-cluster_linking.tf line 113, in resource
| "confluent_cluster_link" "shared_to_sandbox":
| 113: resource "confluent_cluster_link" "shared_to_sandbox" {
```


|

If you see the above error, it indicates that the previous failed attempt left the cluster link in place. To resolve, delete the existing cluster link via the Confluent CLI:

```
confluent kafka link delete bidirectional_between_sandbox_and_shared --
cluster <SANDBOX_CLUSTER_ID> --environment <NON_PROD_ENV_ID> --force
```

Then re-run the `deploy.sh` script with the `create` command.

3.2 Teardown the Infrastructure

```
./deploy.sh destroy \
  --profile=<SSO_PROFILE_NAME> \
  --confluent-api-key=<CONFLUENT_API_KEY> \
  --confluent-api-secret=<CONFLUENT_API_SECRET> \
  --tfe-token=<TFE_TOKEN> \
  --tgw-id=<TGW_ID> \
  --tgw-rt-id=<TGW_RT_ID> \
  --tfc-agent-vpc-id=<TFC_AGENT_VPC_ID> \
  --tfc-agent-vpc-cidr=<TFC_AGENT_VPC_CIDR> \
  --dns-vpc-id=<DNS_VPC_ID> \
  --vpn-vpc-id=<VPN_VPC_ID> \
  --vpn-vpc-cidr=<VPN_VPC_CIDR> \
  --vpn-client-vpc-cidr=<VPN_CLIENT_VPC_CIDR>
```

Handling Cluster Link Deletion Error(s):

If you encounter a Cluster Link deletion error during the destroy process, you may see an error message similar to the following:

```
| Error: error deleting Cluster Link "lkc-
m1ox27/bidirectional_between_sandbox_and_shared": 401 Unauthorized: Not
authorized: the authenticated user didn't have the right access to the
resource: Cluster authorization failed.
```

|

Navigate to the Terraform directory:

```
cd terraform
```

Remove the unreachable resources from the Terraform state:

```
terraform state rm 'confluent_cluster_link.sandbox_and_shared'
terraform state rm 'confluent_cluster_link.shared_to_sandbox'
```

Navigate back to the root directory:

```
cd ..
```

Rerun the `deploy.sh` script with the `destroy` command.

Handling DNS Resolution Errors:

If you encounter DNS resolution errors during the destroy process, you may see error messages similar to the following:

```
| Error: error deleting Kafka ACLs "lkc-y08xyo/TOPIC#dev-
stock_trades#LITERAL#User:sa-81j9gv0##DESCRIBE#ALLOW": Delete
"https://lkc-y08xyo.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-y08xyo/acls?
host=%2A&operation=DESCRIBE&pattern_type=LITERAL&permission=ALLOW&principa
l=User%3AAsa-81j9gv0&resource_name=dev-stock_trades&resource_type=TOPIC":
dial tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no such
host
|
|
|
| Error: error deleting Kafka ACLs "lkc-y08xyo/TOPIC#dev-
stock_trades#LITERAL#User:sa-zmr663##READ#ALLOW": Delete "https://lkc-
y08xyo.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
y08xyo/acls?
host=%2A&operation=READ&pattern_type=LITERAL&permission=ALLOW&principal=Us
er%3AAsa-zmr663&resource_name=dev-stock_trades&resource_type=TOPIC": dial
tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no such host
|
|
|
| Error: error deleting Kafka ACLs "lkc-
y08xyo/GROUP#sandbox_aws_privatelink_example_#LITERAL#User:sa-
zmr663##READ#ALLOW": Delete "https://lkc-y08xyo.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-y08xyo/acls?
host=%2A&operation=READ&pattern_type=LITERAL&permission=ALLOW&principal=Us
er%3AAsa-
zmr663&resource_name=sandbox_aws_privatelink_example_&resource_type=GROUP
": dial tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no
such host
```

```
|
|
| Error: error deleting Kafka ACLs "lkc-
y08xyo/TOPIC#sandbox_aws_privatelink_example_#LITERAL#User:sa-
w7xq219#*#WRITE#ALLOW": Delete "https://lkc-y08xyo.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-y08xyo/acls?
host=%2A&operation=WRITE&pattern_type=LITERAL&permission=ALLOW&principal=U
ser%3Aa-
w7xq219&resource_name=sandbox_aws_privatelink_example_&resource_type=TOPIC
": dial tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no
such host
|
|
```

```
|
| Error: error deleting Kafka ACLs "lkc-y08xyo/CLUSTER#kafka-
cluster#LITERAL#User:sa-w7xq219#*#DESCRIBE#ALLOW": Delete "https://lkc-
y08xyo.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
y08xyo/acls?
host=%2A&operation=DESCRIBE&pattern_type=LITERAL&permission=ALLOW&principa
l=User%3Aa-w7xq219&resource_name=kafka-cluster&resource_type=CLUSTER":
dial tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no such
host
|
|
```

```
|
| Error: error deleting Kafka ACLs "lkc-y08xyo/TOPIC#dev-
stock_trades#LITERAL#User:sa-w7xq219#*#WRITE#ALLOW": Delete "https://lkc-
y08xyo.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
y08xyo/acls?
host=%2A&operation=WRITE&pattern_type=LITERAL&permission=ALLOW&principal=U
ser%3Aa-w7xq219&resource_name=dev-stock_trades&resource_type=TOPIC": dial
tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no such host
|
|
```

```
|
| Error: error deleting Kafka ACLs "lkc-
y08xyo/TOPIC#sandbox_aws_privatelink_example_#LITERAL#User:sa-
w7xq219#*#CREATE#ALLOW": Delete "https://lkc-y08xyo.us-east-
1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-y08xyo/acls?
host=%2A&operation=CREATE&pattern_type=LITERAL&permission=ALLOW&principal=
User%3Aa-
w7xq219&resource_name=sandbox_aws_privatelink_example_&resource_type=TOPIC
": dial tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no
such host
|
|
```

```
| Error: error deleting Kafka ACLs "lkc-y08xyo/TOPIC#dev-
```

```
stock_trades#LITERAL#User:sa-81j9gv0#*#READ#ALLOW": Delete "https://lkc-
y08xyo.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
y08xyo/acls?
host=%2A&operation=READ&pattern_type=LITERAL&permission=ALLOW&principal=Us
er%3A81j9gv0&resource_name=dev-stock_trades&resource_type=TOPIC": dial
tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no such host
|
|
|
| Error: error deleting Kafka ACLs "lkc-y08xyo/TOPIC#dev-
stock_trades#LITERAL#User:sa-81j9gv0#*#WRITE#ALLOW": Delete "https://lkc-
y08xyo.us-east-1.aws.private.confluent.cloud:443/kafka/v3/clusters/lkc-
y08xyo/acls?
host=%2A&operation=WRITE&pattern_type=LITERAL&permission=ALLOW&principal=U
ser%3A81j9gv0&resource_name=dev-stock_trades&resource_type=TOPIC": dial
tcp: lookup lkc-y08xyo.us-east-1.aws.private.confluent.cloud: no such host

Operation failed: failed running terraform apply (exit 1)
```

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_kafka_acl.sandbox_cluster_app_producer_prefix_acls["DESCRIBE"]'
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_producer_prefix_acls["READ"]'
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_producer_prefix_acls["WRITE"]'
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_consumer_read_on_topic'
terraform state rm
'confluent_kafka_acl.sandbox_cluster_app_consumer_read_on_group'
terraform state rm 'confluent_kafka_topic.source_stock_trades'
```

Navigate back to the root directory:

```
cd ..
```

Rerun the `deploy.sh` script with 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](#)