

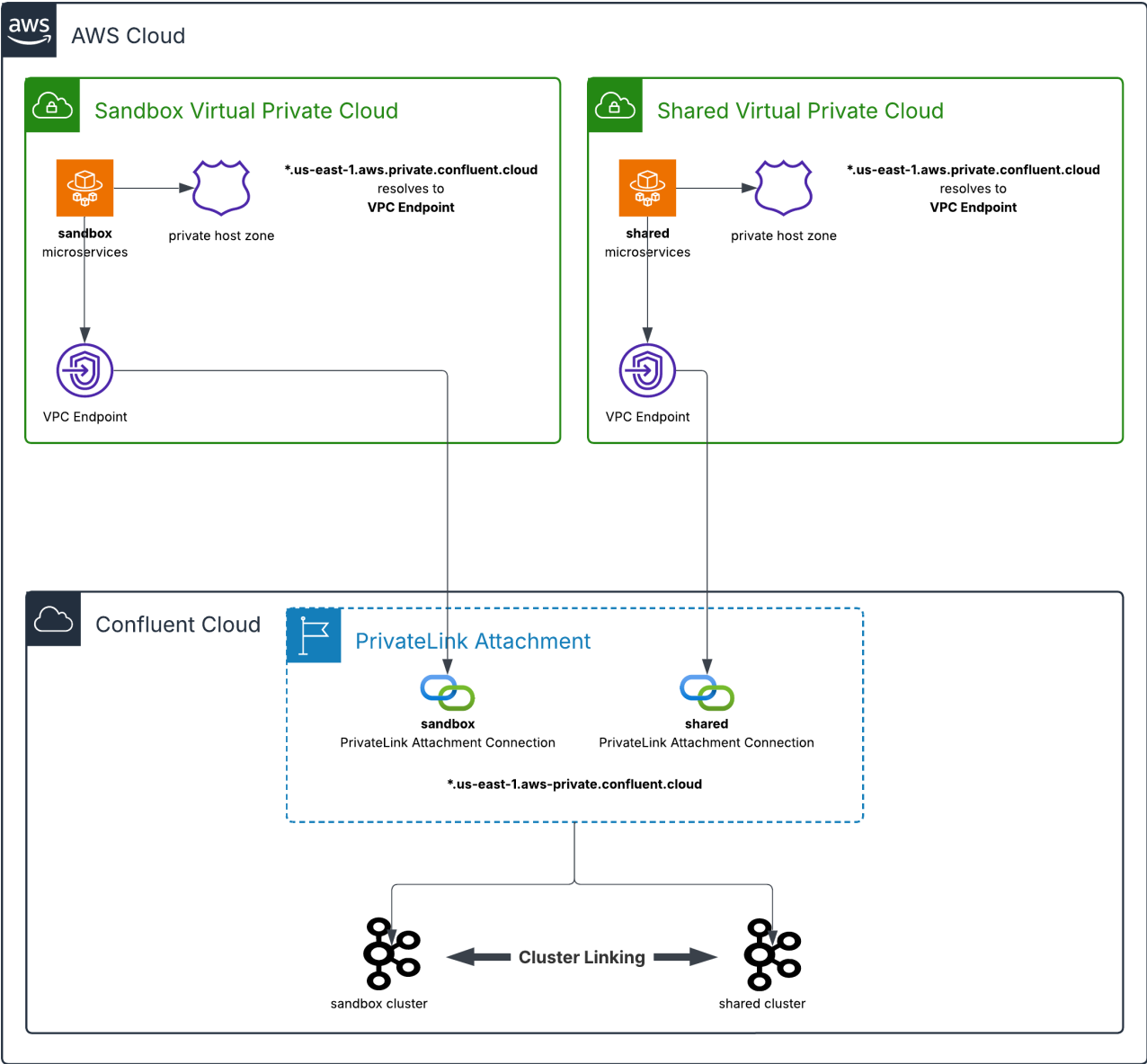
IaC Confluent Cloud AWS Private Linking with Cluster Linking Example

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:

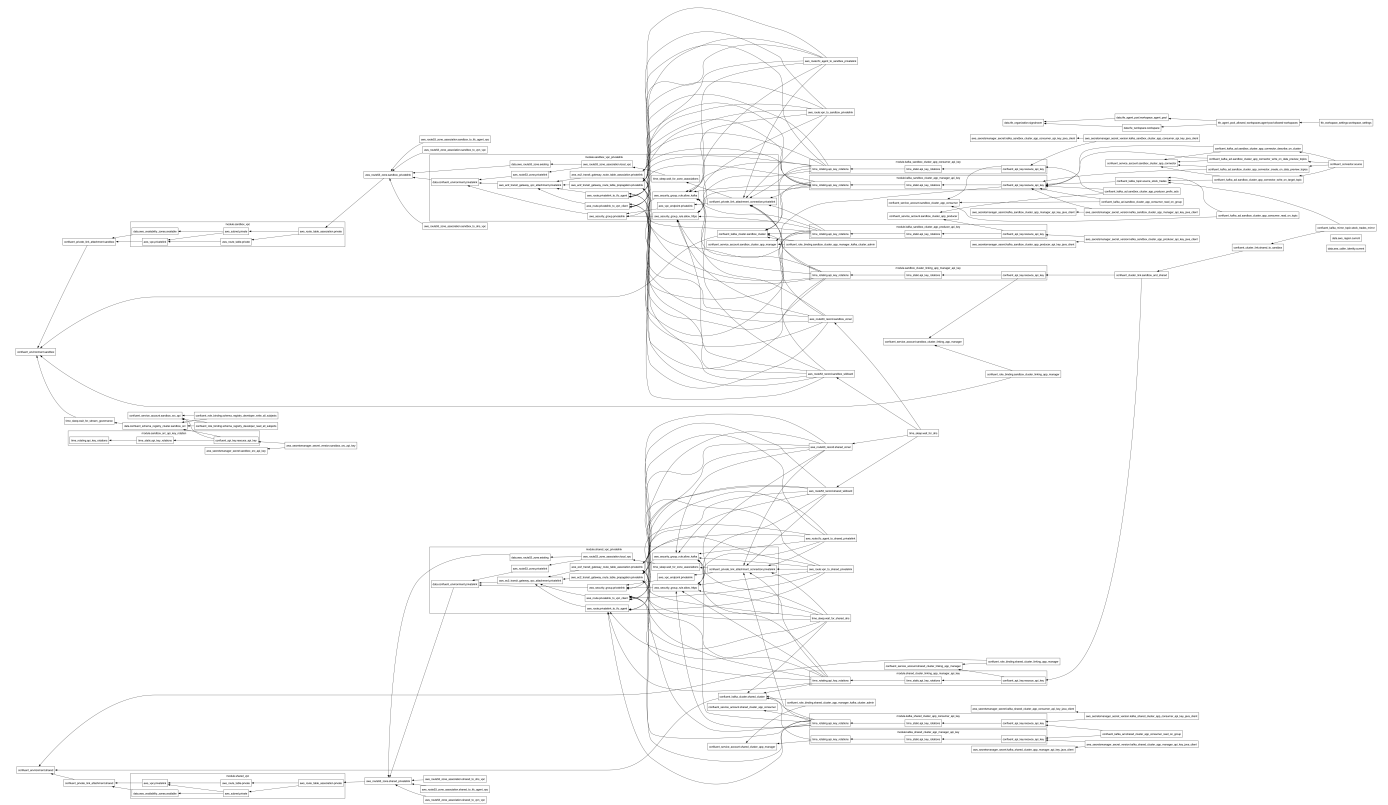


Table of Contents

- [1.0 Resources](#)
 - [1.1 Terminology](#)
 - [1.2 Related Documentation](#)

1.0 Resources

1.1 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.

1.2 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](#)

This command can be used to verify that the appropriate routes have been added to the Terraform Cloud Agent VPC route tables:

```
(echo -e "RouteTableId\tDestination\tGateway\tTransitGateway\tState"; \
aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=<TFC_AGENT_VPC_ID>" \
  --output json | jq -r '.RouteTables[] | . as $rt | .Routes[] | \
    [$rt.RouteTableId, .DestinationCidrBlock, .GatewayId, .TransitGatewayId, \
    .State] | @tsv') | \
column -t -s $'\t'
```

Great questions! Let me explain both in the context of what we've been troubleshooting:

EOF - "End of File"

EOF stands for **End of File**. In networking contexts (like your Terraform errors), it means the connection was **unexpectedly terminated** before the data transfer completed.

What EOF Means in Your Errors

```
Error: Get "https://lkc-2753xq...": EOF
```

This means:

1. ✅ TFC Agent resolved the DNS successfully
2. ✅ TFC Agent initiated a TCP connection to the cluster
3. ❌ The connection got established but then **abruptly closed** before receiving a response
4. No data was received - just "end of file" (connection closed)

Common Causes of EOF Errors

Client → [Network] → Server

↑

Connection drops here (EOF)

Reasons:

- Routing black hole (packets go nowhere)
- Firewall/Security Group blocking return traffic
- Network timeout
- Server immediately closes connection
- Intermediate device (like NAT gateway) drops connection

In your case, the EOF errors were happening because:

- TFC Agent could reach out to the cluster endpoint
- But couldn't get responses back due to missing routes

L3 Hop - "Layer 3 Hop"

L3 refers to **Layer 3** of the OSI networking model - the **Network Layer** (IP routing).

An **L3 hop** is each **router or gateway** that a packet passes through on its journey from source to destination.

The OSI Model Quick Reference

```
Layer 7: Application (HTTP, DNS, SSH)
Layer 6: Presentation (Encryption, Compression)
Layer 5: Session (Session Management)
Layer 4: Transport (TCP, UDP) – Ports
Layer 3: Network (IP Routing) ← L3 is here!
Layer 2: Data Link (MAC addresses, Switches)
Layer 1: Physical (Cables, Wireless)
```

L3 Hop Example

```
Your Laptop → Router1 → Router2 → Router3 → Destination
                ↑ Hop 1  ↑ Hop 2  ↑ Hop 3
```

Each hop = one L3 device making a routing decision

In Your AWS Architecture

```
TFC Agent VPC (10.2.0.0/20)
  ↓ Hop 1: Route table says "10.0.0.0/20 → TGW"
Transit Gateway (TGW)
  ↓ Hop 2: TGW routes to Sandbox VPC attachment
Sandbox VPC (10.0.0.0/20)
  ↓ Hop 3: VPC routing to endpoint subnet
VPC Endpoint (vpce-0a51e878ac71a2af8)
  ↓
Confluent Cluster
```

Each arrow (↓) represents an **L3 hop** - a routing decision at the IP layer.

Why L3 Hops Matter

Troubleshooting:

- You can trace the path with **traceroute** to see each hop
- Helps identify where packets are getting dropped
- Shows routing delays/problems

Performance:

- Each hop adds latency (usually ~1-5ms per hop)
- More hops = more potential failure points

Your Scenario: The missing route from TFC Agent VPC meant packets had **no next hop** defined:

```
TFC Agent: "I need to send to 10.0.1.15..."
Route Table: "I have no route for 10.0.0.0/20!"
Result: Packet dropped, EOF error
```

After adding the route:

```
TFC Agent: "I need to send to 10.0.1.15..."
Route Table: "That's in 10.0.0.0/20, send via TGW!"
TGW: "Route to Sandbox VPC"
Sandbox VPC: "Route to endpoint subnet"
✓ Connection succeeds!
```

Related Networking Terms You Might See

TTL (Time To Live): Maximum number of hops a packet can take before being discarded (prevents routing loops)

MTU (Maximum Transmission Unit): Largest packet size that can traverse a network path

L2 Adjacent: Two devices directly connected at Layer 2 (same network segment, no router in between)

Next Hop: The next router/gateway in the routing path

Default Route: Where to send packets when no specific route exists (often `0.0.0.0/0` → internet gateway)