2026/03/06 ENOG89 MEETING

AWS MCP Serverで実践する環境構築とトラブルシューティング

菊池　之裕

Sr. Solutions Architect Network Specialist

Amazon Web Services Japan G.K.

自己紹介

名前：菊池　之裕(きくち　ゆきひろ)

アマゾン ウェブ サービス合同会社 ネットワークソリューション本部

シニアソリューションアーキテクト　ネットワークスペシャリスト

ロール：Network系サービスについてのご支援

経歴: ISP,IXP,VPN運用、開発を経てネットワーク機器、仮想ルータ販売会社のプリセールス、プロダクトSEからAWSへ（なんと10年目）

好きな AWS サービス:AWS Transit Gateway,AWS Direct Connect, AWS Marketplace

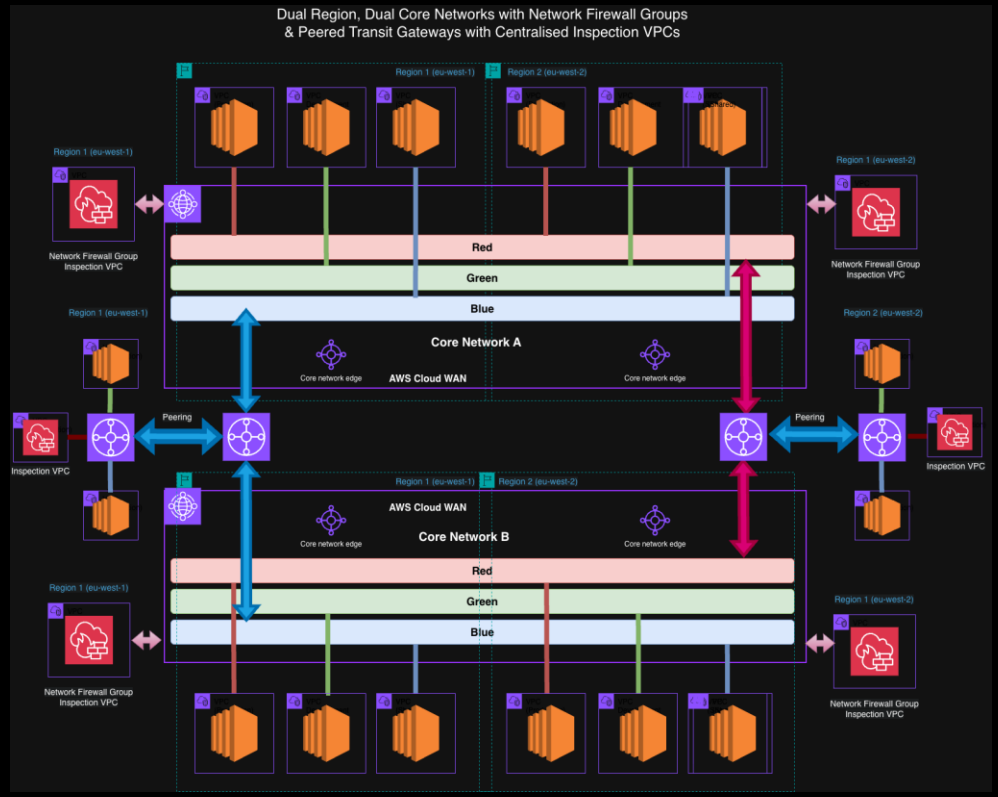
Agenda

- はじめに
- AWS MCP ServerとNetwork MCP Server
- ユースケース
- まとめ

はじめに

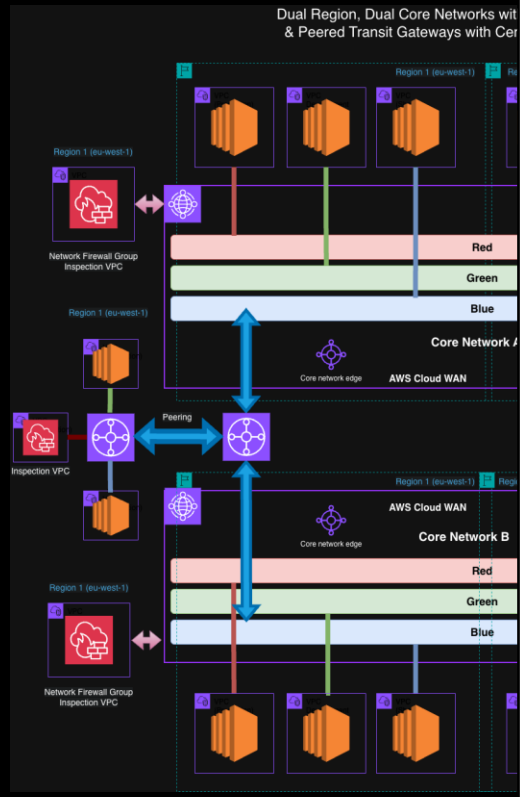
現在のネットワークにおける認知の限界

- 複雑なトポロジーと巨大なポリシー
- インスペクションによる直感的でない経路



現在のネットワークにおける認知の限界

- 複雑なトポロジーと巨大なポリシー
- インスペクションによる直感的でない経路
- ネットワークトポロジー記載のJSON　人の理解を超えている、、




```
{
 "version": "2025.11",
 "segment-actions": [
  {
   "action": "share",
   "mode": "attachment-route",
   "segment": "segmentA",
   "share-with": ["segmentB"],
   "routing-policy-names": ["filterAOutbound", "filterBInbound"]
  }
 ],
 "routing-policies": [
  {
   "routing-policy-name": "filterAOutbound",
   "routing-policy-direction": "outbound",
   "routing-policy-number": 100,
   "routing-policy-rules": [
    {
     "rule-number": 1,
     "rule-definition": {
      "match-conditions": [{"type": "prefix-equals", "value": "10.1.0.0/16"}],
      "condition-logic": "and",
      "action": {"type": "allow"}
     }
    },
    {
     "rule-number": 2,
     "rule-definition": {
      "match-conditions": [{"type": "prefix-in-cidr", "value": "0.0.0.0/0"}],
      "condition-logic": "and",
      "action": {"type": "drop"}
     }
    }
   ]
  },
  {
   "routing-policy-name": "filterBInbound",
   "routing-policy-direction": "inbound",
   "routing-policy-number": 200,
   "routing-policy-rules": [
    {
     "rule-number": 1,
     "rule-definition": {
      "match-conditions": [{"type": "prefix-equals", "value": "10.2.0.0/16"}],
      "condition-logic": "and",
      "action": {"type": "allow"}
     }
    },
    {
     "rule-number": 2,
     "rule-definition": {
      "match-conditions": [{"type": "prefix-in-cidr", "value": "0.0.0.0/0"}],
      "condition-logic": "and",
      "action": {"type": "drop"}
     }
    }
   ]
  }
 ]
```

現在のネットワークにおける認知の限界

- **情報の断片化**： Console、CLI、Wikiを往復する 日常
- **巨大なポリシー**： 数千行に及ぶCloud WAN JSON設定の解読
- **見えない経路**： Inspection VPCを経由するトラフィックのブラックボックス化
- **属人化**： 「あの設定の意味、あの人しか知らない」問題

人間が暗記して管理できる限界はをもう超えている、、

台帳を検索してまわるのも一苦労、、、

スクリプトからエージェントへ — なぜ MCP か？

- トラブルは常に「想定外」から、、、

スクリプト（従来）
決定論的（Deterministic）
手順が固定されており、想定外のエラーで停止
人間が「次の手順」を書き直す必要がある

AIエージェント × MCP（新世代）
推論的（Reasoning）
エラーを観測し、ツールを組み替えて「次の一手」を自ら判断
MCP（Model Context Protocol）がエージェントとツールを標準インターフェースで接続

AWS MCP Serverと Network MCP Server

AWS MCP Serverって何ができるの？

- AWSのAPIをたたいてAWSを操作できる
- AWSの ドキュメンテーション(公式ドキュメント、ナレッジベース、Blogなど）を検索して最新の情報を検索できる

API MCP ServerとKnowledge MCP Serverが統合された

汎用AWS MCP ServerでAWS環境を構築する

「VPCを作ってEC2を立ち上げて」とAIに頼む

ユーザー：

「ap-northeast-1に検証用VPCを作って。

CIDRは10.0.0.0/16、パブリックサブネットを2AZに1つずつ。

Amazon Linux 2023のt3.microを1台立ち上げてSSHできるようにして」

自然言語で環境構築ができる

AIの動き（汎用AWS MCP使用）：

1. describe_availability_zones でAZ一覧を確認

2. create_vpc → create_subnet × 2 → create_internet_gateway

3. create_route_table → create_route → associate_route_table × 2

4. describe_images で最新のAmazon Linux 2023 AMIを取得

5. create_security_group（SSH許可）→ create_key_pair

6. run_instances → describe_instances でパブリックIPを出力

接続コマンドも自動生成：

ssh -i ./my-key.pem ec2-user@10.1.1.10

MCP Server : AWS MCP ServerとNetwork MCP Server

- AWS が提供する MCP サーバー群 — 汎用ツールと専用ツール

汎用 AWS MCP Server
AWS の各種サービス API を MCP ツールとして提供
EC2 / IAM / S3 / CloudWatch など広範なサービスをカバー
開発、検証環境の構築が自然言語でできる
「何でもできる」が、ネットワーク調査には最適化されていない
書き込み権限があり、既存のネットワークを破壊することがある

AWS Network MCP Server
ネットワーク調査・トラブルシューティングに特化した専用 MCP
VPC / TGW / Cloud WAN / Network Firewall の深い解析が可能
シニアエンジニアの思考回路をツールとして実装
リードオンリーで、環境を破壊しない

安全性の担保：Read-Only

なぜRead-Onlyが重要か

- AIエージェントは推論をするが、推論は時に間違える。
- **二次災害の排除**： Read-Onlyによる推論ミスによる勝手な構成変更リスクがゼロ
- **診断と治療の分離**： AIに「診断（Read-Only）」をさせ、人間が「治療（IaC）」を判断
- **本番環境での安心運用**： 物理的にWriteできないから、安心して実行

運用サイクル

- 診断（AI: Read-Only）→ 判断（人間）→ 治療（IaC: Write）

安全性の担保：Read-Only

なぜRead-Onlyが重要か

- AIエージェントは推論をするが、推論は時に間違える。
- **二次災害の排除**： Read-Onlyによる推論ミスによる勝手な構成変更リスクがゼロ
- **診断と治療の分離**： AIに「診断（Read-Only）」をさせ、人間が「治療（IaC）」を判断
- **本番環境での安心運用**： 物理的にWriteできないから、安心して実行

運用サイクル

- 診断（AI: Read-Only）→ 判断（人間）→ 治療（IaC: Write）

誤った推論による意図しない構成変更リスクをゼロに、Read-Onlyだからこそ、AIを本番環境で信頼して使えます。

汎用AWS MCPとの比較 ― 「辞書」と「専門医」

汎用AWS MCP Server を使ってパストレースを依頼する
ユーザー：「VPC-AからVPC-Bへの経路を確認して」

AIの動き：
1. describe-vpcs を呼び出す（全VPC取得）
2. describe-route-tables を呼び出す（全ルートテーブル取得）
3. describe-transit-gateways を呼び出す（TGW取得）
4. describe-transit-gateway-route-tables を呼び出す
5. ... さらに10回以上のAPI呼び出し
6. 途中で文脈を見失い、的外れな回答を返すことがある

Network MCP Server を使うと…
ユーザー：「VPC-AからVPC-Bへの経路を確認して」

AIの動き：
1. get_path_trace_methodology（調査手順をロード）
2. get_vpc_network_details（経路を解析）
3. 結果を論理的にまとめて出力

※https://github.com/awslabs/mcp/tree/main/src/aws-network-mcp-server

汎用AWS MCPとの比較 — 「辞書」と「専門医」

汎用AWS MCP Server を使ってパストレースを依頼する
ユーザー：「VPC-AからVPC-Bへの経路を確認して」

AIの動き：
1. describe-vpcs を呼び出す（全VPC取得）
2. describe-route-tables を呼び出す（全ルートテーブル取得）
3. describe-transit-gateways を呼び出す（TGW取得）
4. describe-transit-gateway-route-tables を呼び出す
5. ... さらに10回以上のAPI呼び出し
6.コンテキストウィンドウの制限により、途中で文脈が失われ、不正確な回答が返される場合がある

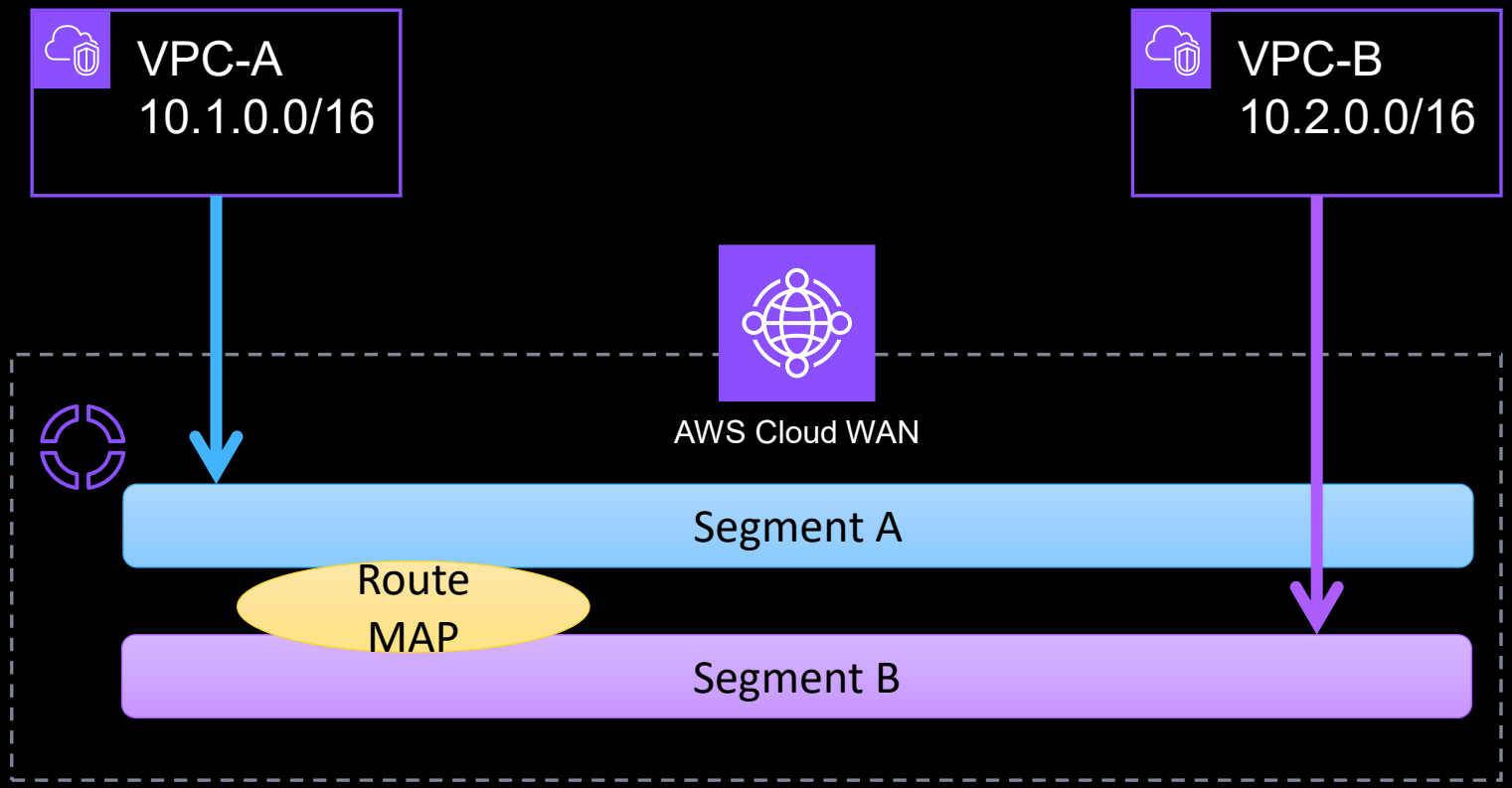
Network MCP Server を使うと…
ユーザー：「VPC-AからVPC-Bへの経路を確認して」

AIの動き：
1. get_path_trace_methodology（調査手順をロード）
2. get_vpc_network_details（経路を解析）
3. 結果を論理的にまとめて出力

※https://github.com/awslabs/mcp/tree/main/src/aws-network-mcp-server

実際にデモしてみます。

ネットワーク構成



Demo: AWS MCP ServerとNetwork MCP Server

ユースケース

ユースケース: IP Discovery（数十秒の調査）

課題： 数百のVPCから特定IPを探すのは苦行

従来の方法： Console → VPC一覧 → サブネット確認 → ENI検索 → SG確認 → 約10分

MCP使用時： 「10.1.1.168は誰ですか？」

→ find_ip_address が全プロファイル・全リージョンを並行スキャン

→ 数十秒でENI、インスタンス、セキュリティグループを特定

得られる情報：

- リソースの正体（EC2 / RDS / Lambda ENI など）
- 所属VPC・サブネット・アベイラビリティゾーン
- アタッチされたセキュリティグループとルール

ユースケース:「ルーティングは正しいのに通らない」

ユーザー：「VPC-AからVPC-Bに繋がらない。ルートは合ってるはずなんだけど」

AIの動き：

1. VPC-A・VPC-Bのルートテーブルを確認 → 経路は存在する
2. TGWルートテーブルを確認 → 転送設定も正しい
3. detect_tgw_inspection を実行
   → TGWがアプライアンスモードで動作していることを検出
   → Inspection VPC内のNetwork Firewallを経由していることを特定
4. Firewallポリシーを確認
   → 対象ポートのルールが「DENY」になっていることを発見

AIの出力：

「ルーティング設定は正しいですが、Inspection VPC内のNetwork FirewallがTCPポート443をDenyしています。Firewallルールの修正を推奨します。」

従来： 複数のコンソールを行き来しながら1〜2時間の調査

MCP使用時： 数分で根本原因を特定

アウトプット — Mermaid による可視化

AIに「図解して」と指示するだけで、Mermaid形式の図が自動生成

生成できる図の種類：

シーケンス図： パケットの旅路（送信元 → TGW → Inspection VPC → 宛先）

フローチャート： ルーティング判定ロジック

グラフ図： Cloud WANセグメント間の接続関係マトリクス

例：パストレース結果のシーケンス図

```
sequenceDiagram
  VPC-A ->> TGW: 10.172.16.0/16 (ルートあり)
  TGW ->> Inspection VPC: アプライアンスモード
  Inspection VPC ->> Network Firewall: ルール評価
  Network Firewall ->> VPC-B: ACCEPT
```

https://bit.ly/3OPbh8s

mermaid.live/edit#pako:eNpdkM9LwzAUx_-V8E4K3ei6rj9yGMxuyi4ydChoRcKabcU2nVnL1LGD7UW9i2eHl7GjJ0H8Z-LBP8MmOEEPgfdePC4y...

新しい Chrome をご利用いただけます

Replace ChatGPT Pro, Mermaid.live, and Lucid Chart with Mermaid Chart Try now

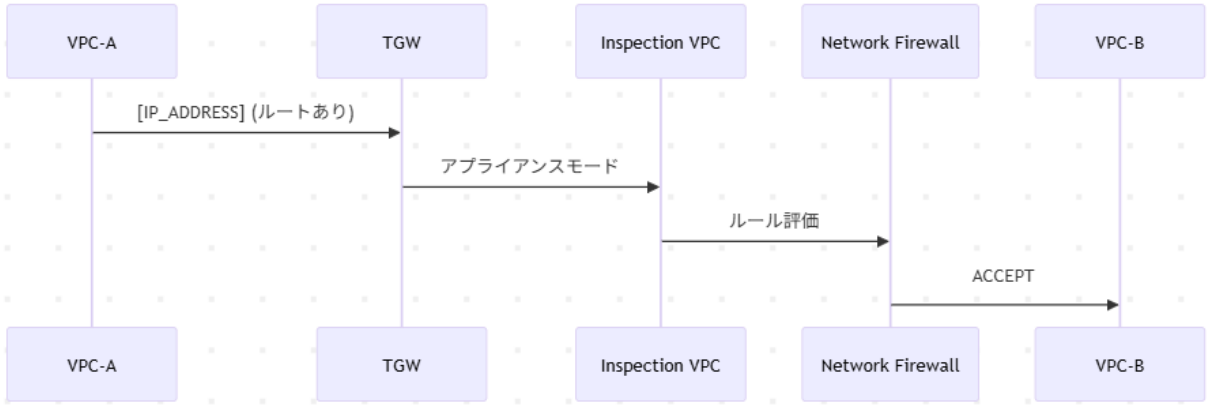
Mermaid Live Editor

Share Save diagram

Code Config Docs

```
sequenceDiagram
  VPC-A ->> TGW: [IP_ADDRESS] (ルートあり)
  TGW ->> Inspection VPC: アプライアンスモード
  Inspection VPC ->> Network Firewall: ルール評価
  Network Firewall ->> VPC-B: ACCEPT
```

Sample Diagrams



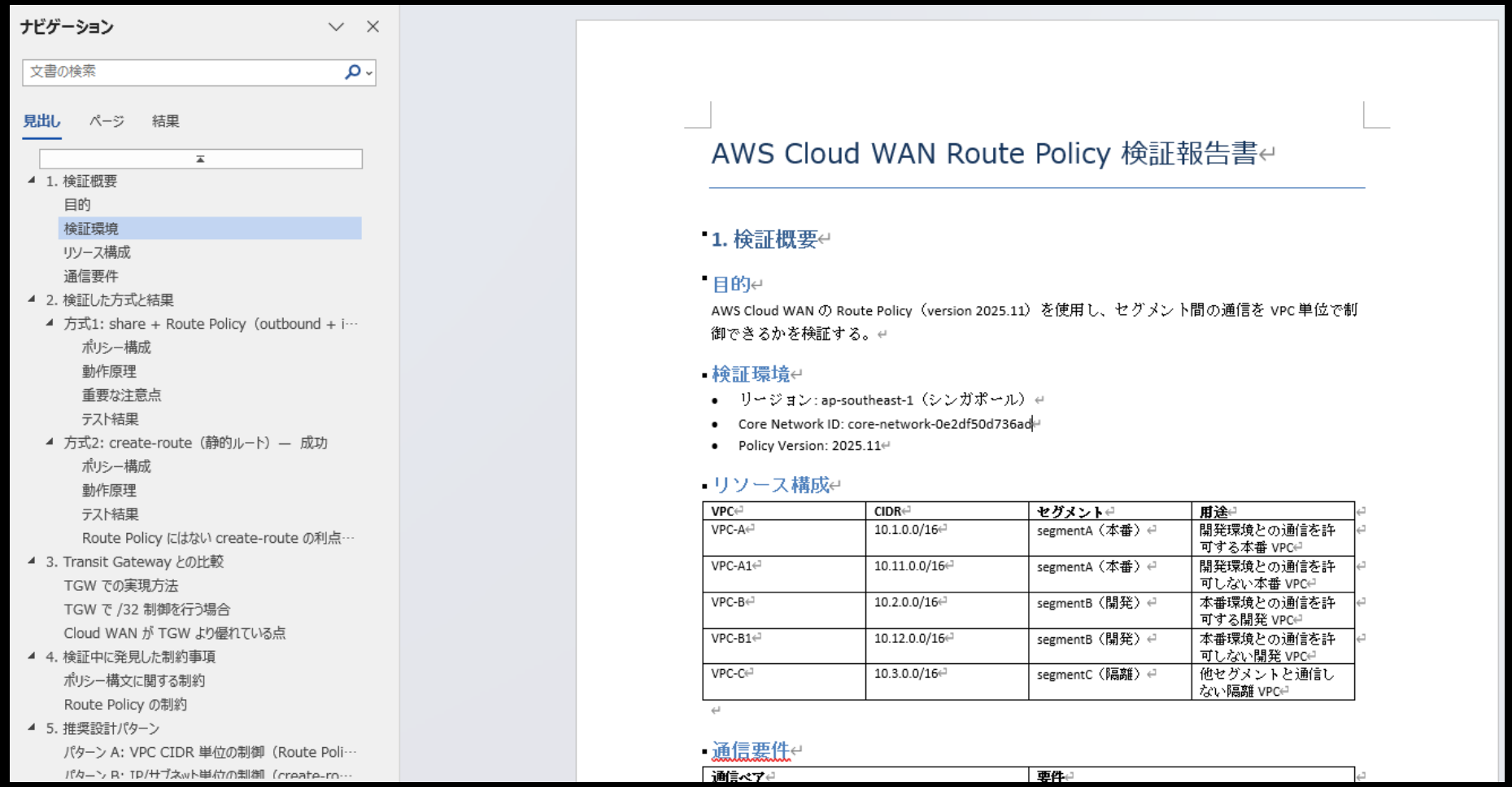
調査 = ドキュメント作成: 自動ドキュメント化

MCP + AIを使った調査は、そのまま構成ドキュメントに

自動生成できるドキュメント：
Cloud WANセグメント接続マトリクス（どのセグメントがどこと通信できるか）
TGWルートテーブルのサマリー（全アタッチメントと経路の一覧）
Firewallルールの人間語サマリー（「このルールは〇〇を許可/拒否している」）
インシデント調査レポート（調査手順・発見事項・推奨アクションを含む）

従来： 調査後に別途ドキュメントを書く（1〜2時間）
MCP使用時： 調査と同時にドキュメントが完成

調査 = ドキュメント作成: 自動ドキュメント化



汎用AWS MCPとNetwork MCPの連携

2つのMCPはそれぞれ得意領域が異なり、組み合わせることで調査が完結する。

典型的なトラブルシューティングでは、問題の原因が「ネットワーク層」なのか「その周辺（権限・設定・リソース）」なのかが最初は分からない。

汎用AWS MCPが担う領域：
EC2インスタンスの状態・設定の確認
セキュリティグループのルール確認
VPCエンドポイントの存在確認
IAMロール・ポリシーの確認

Network MCPが担う領域：
ルートテーブルの論理検証
TGW・Cloud WANの経路解析
Firewallポリシーの解読
フローログによる実測値の確認

調査の流れ：

「繋がらない」という事象

→ 汎用MCP：リソースと設定の全体像を把握

→ Network MCP：経路とポリシーを深掘り

→ 根本原因を特定 → 人間がIaCで修正

ベストプラクティス — 診断と治療の分離

診断（AI: Read-Only）→ 判断（人間）→ 治療（IaC: Write）

AIに任せること：

- 全リソースの横断スキャン
- ルーティングロジックの論理検証
- ポリシー矛盾の検知
- ログの収集・集計・解釈

人間が担うこと：

- 調査結果の最終判断
- 変更の承認とリスク評価
- IaCによる構成変更の実施
- 設計の意図と文脈の保持

ベストプラクティス — 診断と治療の分離

診断（AI: Read-Only）→ 判断（人間）→ 治療（IaC: Write）

AIに任せること：

- 全リソースの横断スキャン
- ルーティングロジックの論理検証
- ポリシー矛盾の検知
- ログの収集・集計・解釈

人間が担うこと：

- 調査結果の最終判断
- 変更の承認とリスク評価
- IaCによる構成変更の実施
- 設計の意図と文脈の保持

AIにすべてを任せるのではなく。AIに診断をさせ、人間が責任を持って治療を行う

まとめ

まとめ

AIと共に歩むネットワークエンジニアの未来

- ルーティーン調査をAIに委譲、エンジニアは高度な設計・判断に集中
- 環境の構築は AWS MCP Server
- 本番の検証は Read-onlyのNetwork MCPで
- 報告書の作成もLLMにお任せ

本来の設計をエンジニアの手に

Q&A

Thank you!