



生成AIファーストな ネットワークエンジニアリングを目指して

～検証例から考える次世代ネットワーク運用～

株式会社MIXI
開発本部 インフラ室 ネットワーク開発グループ
江川 潤

JANOG56 Day2 2025年7月31日(木) 11:15～12:00(45分)

自己紹介

株式会社MIXI
開発本部 インフラ室
ネットワーク開発グループ

江川 潤 / Jun Egawa

MIXI社歴 2023年10月 -
仕事内容 AS38651の運用、
LaLaアリーナ館内NW運用を主に担当

JANOG歴 Janog50が初参加、今回が初登壇！

SNS /

mixi2 [@junenu](#)

X [@JulianTima013](#)

GitHub [@junenu](#)



伝えたいこと：生成AIと共存しよう

- 1 検証に至るまで前提
- 2 生成AI活用検証結果の共有
- 3 検証結果から見えてきたもの
- 4 次世代のネットワーク運用に向けてやるべきこと

- JANOGでも生成AIの活用が本格的に議論
- 生成AI活用により運用効率化の動きが加速
 - 障害分析・対応など
- 一方で、多くの組織では具体的なユースケースを模索中
- 現実的な導入や運用に関するノウハウが求められている

この資料に登場する用語の補足

用語	一言で言うと	補足
LLM (大規模言語モデル)	あらかじめ学習した膨大な知識を活用して自然な言語を生成するAI	例: ChatGPTなど。汎用的な文章理解・生成が可能。
RAG (検索拡張生成)	外部情報を 検索・参照 しながら、より正確な回答を行う仕組み	社内ドキュメントやFAQといった実データに基づいた応答が可能。
MCP (Model Context Protocol)	生成AIに業務の背景やルールの文脈を明示的に与えるための仕組み	LLMが様々なデータソースやツールに接続するための標準化された方法

生成AIファースト？

生成AIファースト？

ネットワークエンジニアリングにおける「生成AIファースト」

- ネットワーク運用のパラダイムシフト
- 生成AIファーストの定義
 - 生成AIが主体となり、判断と実行を行う
 - 人間は生成AIの能力を最大限に引き出す
 - そして、より高度な業務へ...
- 理想の運用像
 - リアルタイムな状況把握、予兆検知
 - プロアクティブな改善提案/最適化

3つのフェーズに分ける

例：BGPルーター追加作業を例に

支援型

生成AI = “補佐役”
人間のサポートをする
例：

- 疎通確認スクリプト作成
- 作業手順書作成

半自律型

生成AI = “プレイヤー”
人間の指示のもと作業を実施
例：

- BGPルーター設定作成
- ドキュメント作成

完全自律型

生成AI = “マネージャー”
自分で物事を判断して作業
例：

- BGPルーター設計 - 構築
- トラシュー対応



支援型の活用は進んできた！
次は自律的な活用！

MIXIのNWでも活用したい！

MIXI AS38651のざっくり構成



- 運用上の課題の解決

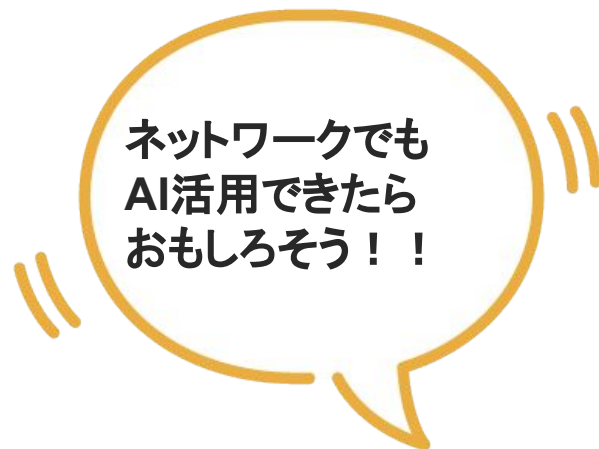
- ネットワークの運用効率化
- 非定型業務の人間依存を解消

- 余白時間の創出

- 我々人間は次の高度な挑戦をしていく！

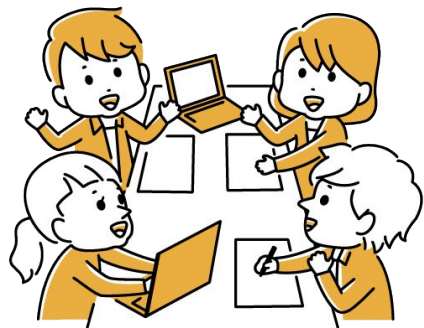
- 発明の連鎖をしていく

- 使ってみないとわからないことがたくさん



チーム内でディスカッション！！！！

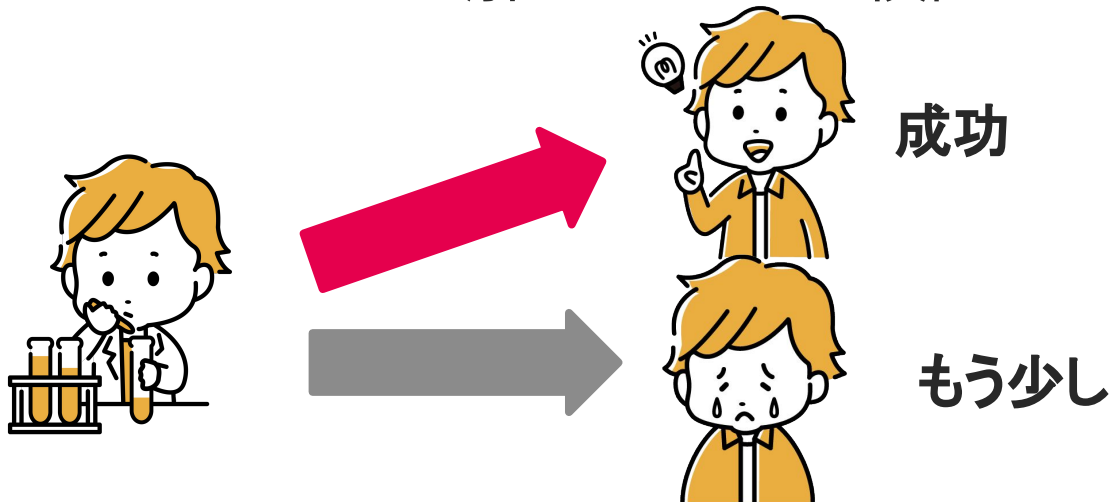
- 作業手順書作成
- セキュリティレポートのOSバージョンチェック
- 障害解析
- グラフ（トラフィックなど）の分析 などなど



成功体験を積んで、ステップアップ
本当に使えるのだろうか...?
早速検証!!!!

目次

1. チャットボットをカスタマイズする検証
2. コーディングAIエージェントを使う検証
3. NWのコンテキストを理解できるのか検証



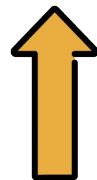
検証その1

セキュリティレポート該当チェック

支援型

半自律型

完全自律型



セキュリティレポート該当チェック

- 定期的に公開されるセキュリティレポートの確認は必須
- 手作業チェックの負担をどう減らすか？が課題に

解決策

専用のカスタム AIチャットボット

ノーコードでカスタマイズ可能なAIチャットボット
今回は、OpenAI ChatGPTのカスタムGPTを利用



セキュリティレポート該当チェック

生成AIに指示する

POINT

命名規則をインプットする

- OSによってバージョン命名規則に特徴がある

OSバージョン構造（解析ルール）

バージョン形式：`Lx.yZz`

例：`L6.2S1`

|要素|意味|例|

|-----|-----|-----|

|`Lx`|レイヤー（大まかな世代）|L6|

|`yZ`|マイナー+リリースタイプ|2S（2=マイナー, S=安定）|

|`z`|ノード（小規模な修正）|1|

リリースタイプの優先度：K < S < X

セキュリティレポート該当チェック

生成AIに指示する

POINT

機器の情報の入力形式をインプットする

- OSバージョン
- セキュリティレポートに該当する設定（必要あれば）

入力方法

まず、あなたが管理している機器の情報を以下の形式で入力してください：

...

device/CORE/CoreRouter1:# OS: L5.5K2

device/EDGE/EdgeSwitch2:# OS: L6.2S1

device/CAMPUS/CampusLeaf4:# OS: L7.0S1

...

※必要に応じて、該当機器に適用されている設定ブロック（CLI形式）を追加してください：

...

device/TEST/TestBox6:#

username user {

role admin;

password-hash "\$9\$testbox";

}

...

セキュリティレポート該当チェック

生成AIに指示する

POINT

出力形式を指定

- ハルシネーションを防ぐ注釈

出力形式（判定結果）

機器名	OSバージョン	該当設定	判定	判定理由
CoreRouter1	L5.5K2	⚠️あり	❌影響あり	範囲内&設定一致
EdgeSwitch2	L6.2S1	✅なし	✅影響なし	条件不成立
CampusLeaf4	L7.0S1	⚠️あり	❌影響あり	範囲内&設定一致

備考

- 構成情報が不明な場合、影響判定は「⚠️不明」となります。

セキュリティレポート該当チェック

結果、最初は失敗でしたが・・・
コードインタプリター機能をON

➡ 間違いがなくなった！

	機器名	OSバージョン	該当設定	判定	判定理由
1	CoreRouter1	L5.5K2	⚠️あり	❌影響あり	範囲内&設定一致
2	EdgeSwitch2	L6.4S3	⚠️あり	✅影響なし	設定一致だが範囲外
3	DistRouter3	L6.3S2	✅なし	✅影響なし	範囲内だが設定不一致
4	CampusLeaf4	L6.2S0	⚠️あり	❌影響あり	範囲内&設定一致
5	CampusLeaf5	L6.3S0	✅なし	✅影響なし	範囲内だが設定不一致
6	TestBox6	L5.9X1	⚠️あり	❌影響あり	範囲内&設定一致

コードインタプリター機能とは
機器情報などプロンプトの内容を
Pythonコードに変換して処理

```
python
import pandas as pd

# Define device data
devices = [
    {"機器名": "CoreRouter1", "OSバージョン": "L5.5K2", "設定": "user"},
    {"機器名": "EdgeSwitch2", "OSバージョン": "L6.4S3", "設定": "user"},
    {"機器名": "DistRouter3", "OSバージョン": "L6.3S2", "設定": "netadmin"},
    {"機器名": "CampusLeaf4", "OSバージョン": "L6.2S0", "設定": "user"},
    {"機器名": "CampusLeaf5", "OSバージョン": "L6.3S0", "設定": "guest"},
    {"機器名": "TestBox6", "OSバージョン": "L5.9X1", "設定": "user"},
]

# Define the function to parse OS version
def parse_version(version):
    layer = int(version[1])
    minor = int(version[3])
    release_type = version[4]
    node = int(version[5])
    return (layer, minor, release_type, node)

# Define comparison function for OS versions
def version_in_range(v, lower, upper):
    def to_tuple(ver):
        layer = int(ver[1])
        minor = int(ver[3])
        release_type = ver[4]
        node = int(ver[5])
        release_order = {'K': 0, 'S': 1, 'X': 2}
```

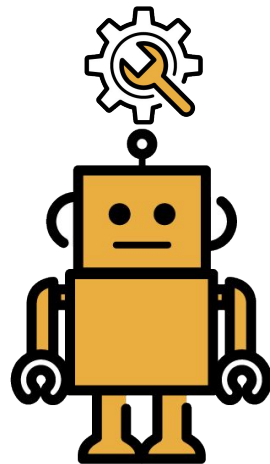
セキュリティレポート該当チェック

得られたメリット

- 0から作ってみて3時間で完成
- 約3時間かかっていた作業が1時間程度で完了
- 出力がまとまっているので共有しやすい

学んだこと

生成AIに間違いが発生しない
仕組みを与える

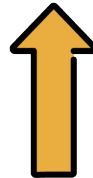


検証その2 NW作業の準備業務

支援型

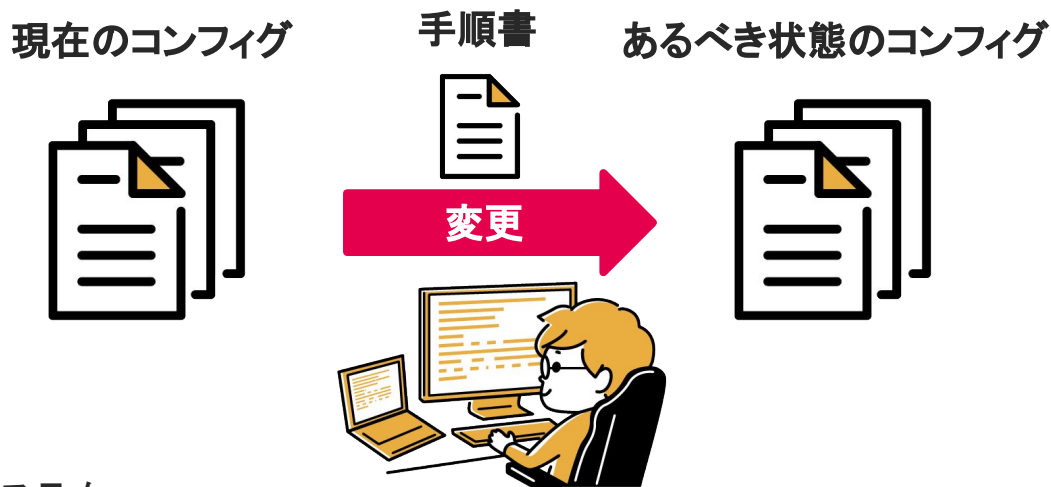
半自律型

完全自律型



MIXIの作業準備方法

1. テンプレートシステムであるべき状態のコンフィグを生成
2. あるべき状態のコンフィグへ変更する手順書を作る



参考

テンプレートシステム:

[Janog47「走りながら作るネットワークテンプレートシステム」ってなんだ!?](#)

実例だとこんな感じ！

```
- - {name: Ethernet33/1, speed: 100g, state: disabled}
+ - name: Ethernet33/1
+   speed: 100g
+   description: janog56 in Matsue
+   mtu: 1600
+   lldp: disabled
+   ipv4:
+     address: 192.0.2.1/24
+   ipv6:
```

```
interface Ethernet33/1
- shutdown
+ description janog56 in Matsue
+ mtu 1600
+ speed forced 100gfull
+ no switchport
+ ip address 192.0.2.1/24
+ ipv6 enable
+ no lldp transmit
+ no lldp receive
```



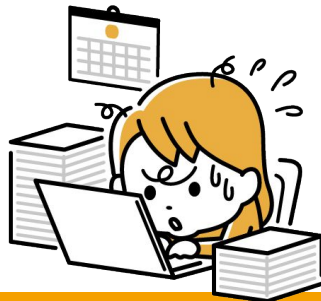
yamlファイルを編集して
スクリプト実行するとコンフィグ生成

課題

- 定型化作業が少なく準備工数がかかる
 - NW構成のアップデートが早い
 - 標準化に落とし込む前に次の案件が・・・
 - ビジネスを止めないことが優先

解決策

AIエージェントに任せる



使ってみる！

- Interop ShowNetとのペアリング作業で検証
- コーディングAIエージェントを利用
 - OpenAI ChatGPTのCodex



ShowNetと
ペアリングするための
NW作業準備をして

実際はこんなプロンプト

JPIXのPeeringを追加するPRを作成してください

AS番号: 290

名前: ShowNet

ピアリングIPアドレス:

210.171.224.9

2001:de8:8::20:82:1

prefix:

45.0.0.0/16

130.128.0.0/15

202.17.220.0/23

2001:3e8::/32

<(省略)テンプレートシステムの使い方を指示>

<(省略)過去の同様作業の手順書を貼り付け>



テンプレートシステム

RT1.yaml

```
- name: Port1
  description: XYZ
  ipv4:
    address: 210.171.225.107/23
```

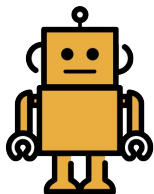
ASN.yaml

```
- asn: 38651
  name: MIXI ASN
  max_prefixes:
    ipv4: 3
    ipv6: 1
- asn: 290
  name: ShowNet
  max_prefixes:
    ipv4: 3
    ipv6: 1
```

コンフィグ
生成

```
2972 3042 no neighb
2973 3043 neighbor
2974 3044 neighbor
2975 3045 neighbor
2976 3046 neighbor
3047 + neighbor jpix-as290-ipv4 peer group
3048 + neighbor jpix-as290-ipv4 remote-as 290
3049 + neighbor jpix-as290-ipv4 remove-private-as
3050 + neighbor jpix-as290-ipv4 local-v4-addr 210.171.225.107
3051 + neighbor jpix-as290-ipv4 description AS290 ShowNet
3052 + neighbor jpix-as290-ipv4 route-map jpix-as290-in-ipv4 in
3053 + neighbor jpix-as290-ipv4 route-map jpix-as290-out-ipv4 out
```

YAMLを編集
スクリプト実行



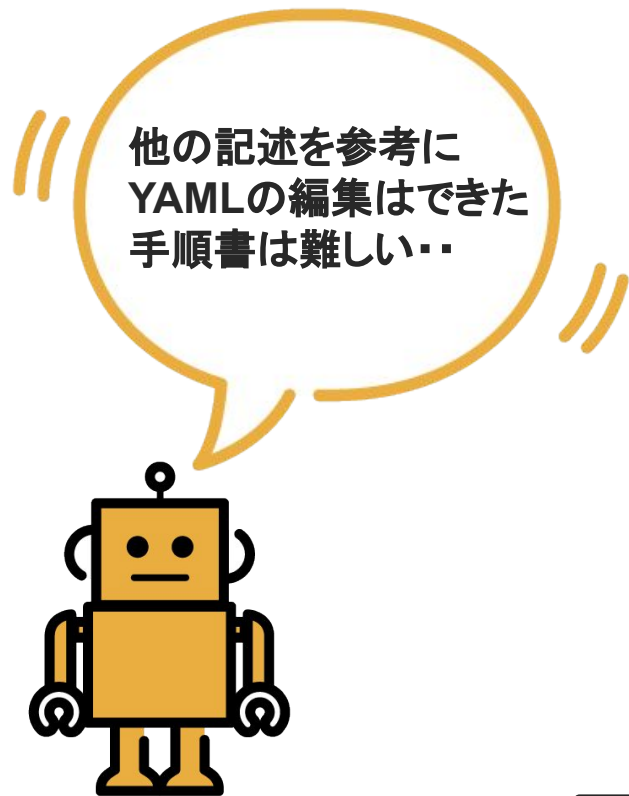
(仕組みのおかげで)
コンフィグ準備を
完全自動化 🎉

一方で

手順書作成は、難易度高い

- ほとんど型通りにやるだけだが・・・
- 最終的にほとんど自分で書いた

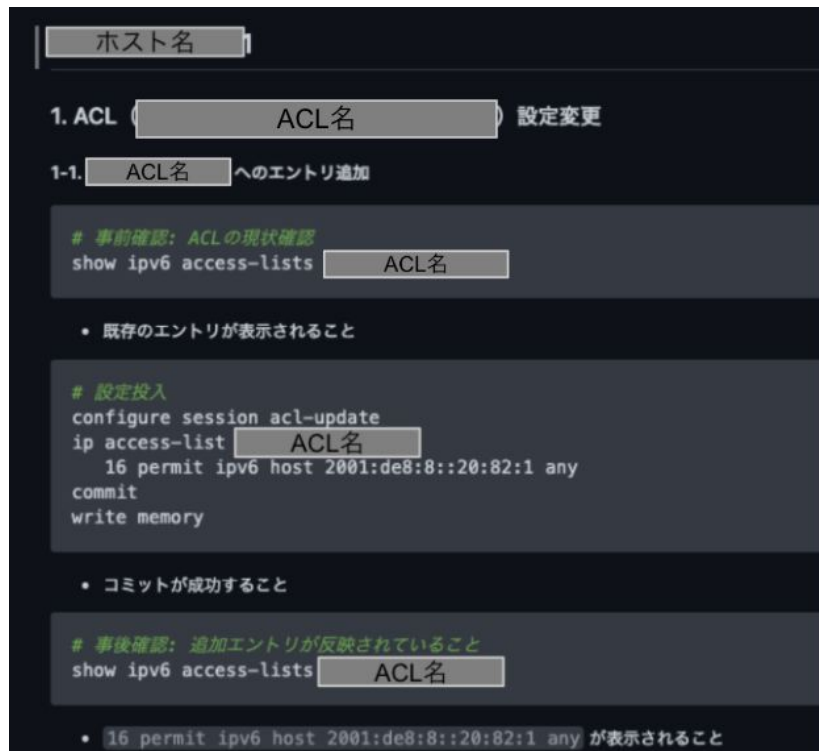
**ゼロから何かを生成されるのではなくて
仕組みで解決する**



後日、再挑戦

別のツール(GitHub Copilot)で
手順書作成に再挑戦

これは成功！！
70%くらいの出来、だけど十分
精度を上げれば運用にのりそう



Host Name: [Host Name]

1. ACL [ACL Name] [設定変更]

1-1. [ACL Name] へのエントリ追加

事前確認: ACLの現状確認

```
show ipv6 access-lists [ACL Name]
```

- 既存のエントリが表示されること

設定投入

```
configure session acl-update
ip access-list [ACL Name]
  16 permit ipv6 host 2001:de8:8::20:82:1 any
commit
write memory
```

- コミットが成功すること

事後確認: 追加エントリが反映されていること

```
show ipv6 access-lists [ACL Name]
```

- 16 permit ipv6 host 2001:de8:8::20:82:1 any が表示されること

指示書はできるだけ具体的に

ACL行番号の入れ替えルール

目的

- シーケンス番号の欠番をなくし、可読性と管理性を維持する。元のACLの評価順序は絶対に変更しないこと。

実施ルール

- ACLの差分（行の削除や追加）がある場合、以下のロジックで新しい設定コマンドを生成すること。

1. 現状の把握：まず、変更対象のACLの全構成（行番号とルール）を把握する。
2. 削除の実行：`diff`で`-`となっている行を削除（`no <行番号> ...`）するコマンドを生成する。
3. 再ナンバリングと追加：
 - 既存行の維持：元のACLから削除対象行を除いたリストを作成する。このリストの順序（元の評価順序）は維持する。
 - 新規採番：このリストの各行に対して、`10`から`10`刻みなど、規則的な新しいシーケンス番号を割り振る。
 - 追加行の挿入：`diff`で`+`となっている行を、diffで示された位置関係（どの行の後に追加されたか）を維持しつつ、
 - コマンド生成：2で生成した削除コマンドに続き、再採番・挿入されたすべてのACL行を再設定するコマンドを生成する。

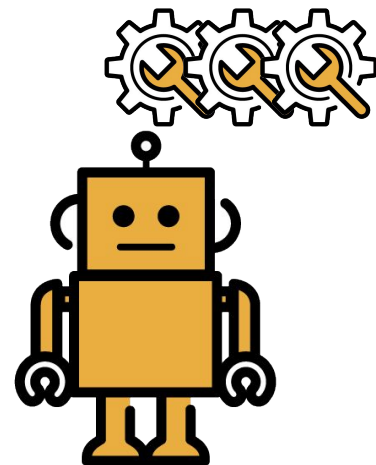
NW作業の準備業務

得られたメリット

- AIが作業している裏で別のことができる
- 間違いはあれど、0から作る必要がなくなったこと
- 1作業につき約1時間稼働削減 🎉

学んだこと

やはり仕組みがあると確実性アップ
できるだけ的確な指示をすること



検証その3

NWのコンテキストを理解させる

支援型

半自律型

完全自律型

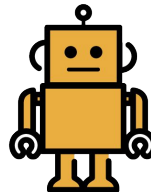


目指して...

NWのコンテキストを理解させる



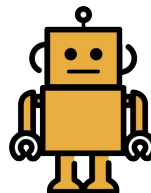
RT1のOSは？



RT1とは？



RT1のOSは？
<context>
RT1は、Z社のA-OS
</context>



RT1はZ社のA-OSです

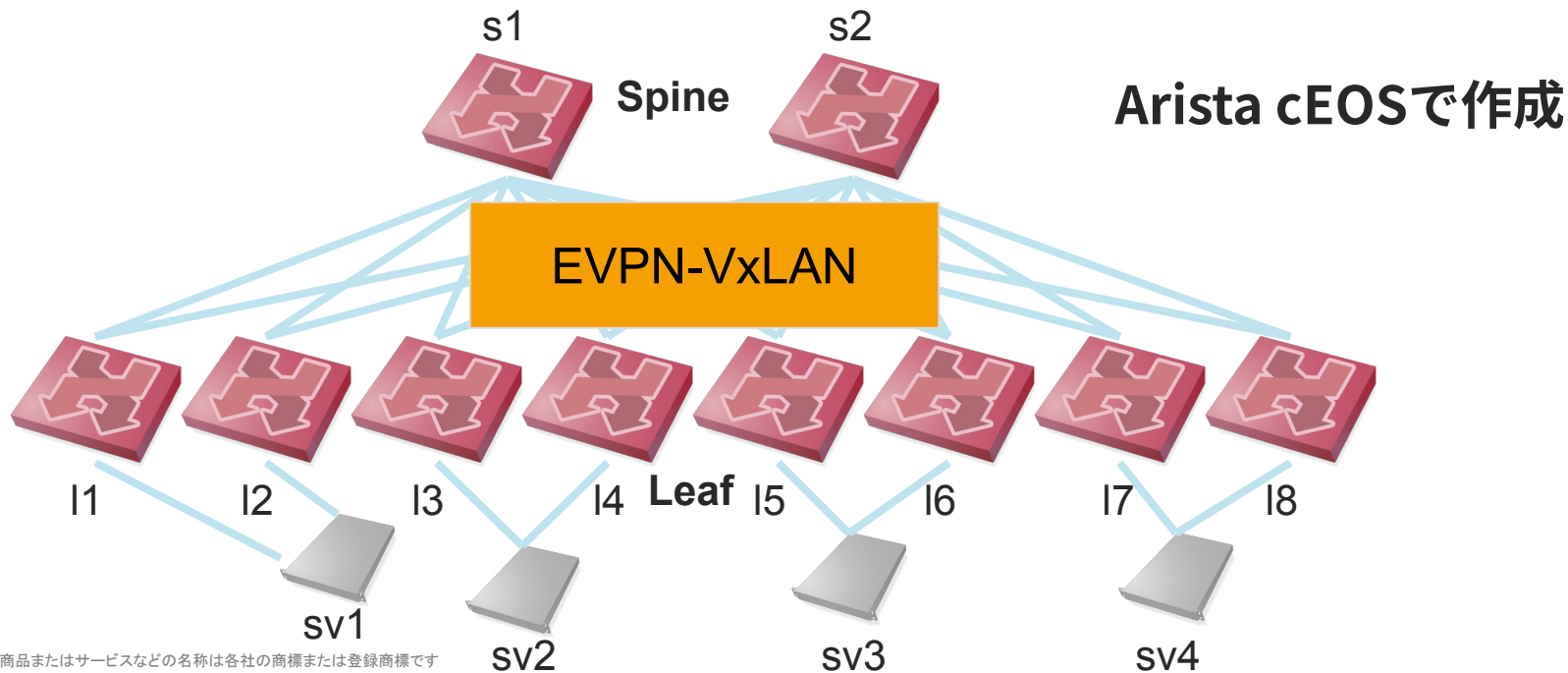
コンテキスト：
文章などの前後の脈絡。文脈。

コンテキストを与える手段として
RAGとMCPを検証しました

NWのコンテキストを理解させる

検証

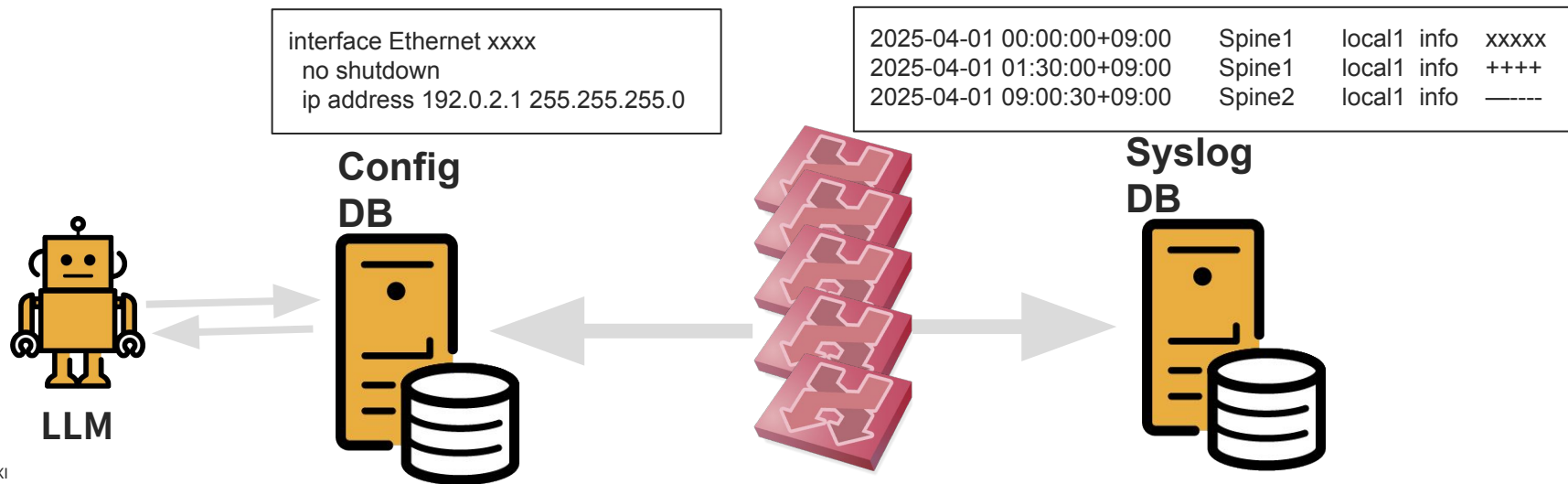
- containerlabで仮想環境を作成して検証



NWのコンテキストを理解させる

RAGを作ってみよう！

- コンフィグとシスログファイルを用意
- それぞれベクトルDBを作成してRAGとして利用



NWのコンテキストを理解させる

RAGとは？

RAG（検索拡張生成）

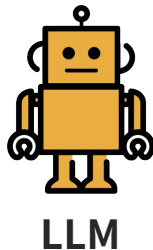
外部情報を検索・参照しながら、
より正確な回答を行う仕組み

OS Version
Z: A-OS 2025.07.31

(question)
RT1のOSは？



(answer)
RT1はZ社のA-OSです



関連するコンフィグの
分割単位を検索
(上の文章を探す)



Config
DB

NWのコンテキストを理解させる

コンフィグRAGを参照して質問

 「s1のloopback interfaceの設定を教えてください」



s1のLoopback0インターフェースの設定は以下の通りです。

```
interface Loopback0
  ip address 10.0.0.1/32
  ip ospf cost 1
  ip ospf area 0.0.0.0
```

実際の設定情報を答えることができた！！！！

****設定の目的と意図:****

- * ****IPアドレス:**** Loopback0インターフェースにIPアドレス `10.0.0.1/32` を設定しています。Loopbackインターフェース
- * ****OSPFコスト:**** `ip ospf cost 1` コマンドは、OSPFルーティングプロトコルにおけるこのインターフェースのコストを
- * ****OSPFエリア:**** `ip ospf area 0.0.0.0` コマンドは、このインターフェースをOSPFエリア0.0.0.0（バックボーンエ

シスログRAGとコンフィグRAGを合わせて障害解析

擬似障害:s1でBGPクリアを実施



「何か問題は発生していますか？」



=== Debug Information ===

質問: 何か問題が発生していますか？

問題の概要: BGPピアリングがリセットされている。

関連する設定/ログ情報の分析: s1のログに、BGPピアリングがAdminResetによってIdleになっていることが記録されている。対象のピアは、10.0.1.2

対応策: BGPピアリングがリセットされた原因を特定する必要がある。s1でBGP hard resetを実行した理由を調査する。また、10.0.0.1で何が起きたのか

分類: ['bgp']

影響を受けている機器: ['s1', 'l1', 'l2', 'l3', 'l4', 'l5', 'l6', 'l7', 'l8']

調査期間: 2025-03-31 16:08:58

問題の概要:

BGPピアリングがリセットされている。

関連する設定/ログ情報の分析:

s1のログに、BGPピアリングがAdminResetによってIdleになっていることが記録されている。対象のピアは、10.0.1.2, 10.0.1.3, 10.0.1.4, 10

対応策:

BGPピアリングがリセットされた原因を特定する必要がある。s1でBGP hard resetを実行した理由を調査する。また、10.0.0.1で何が起きたのかも調査

NWのコンテキストを理解させる

RAGの検証結果は良好だった！・・・だけど

- データは最新に保つ運用がづらい...
- RAGに向いている使い方ではなかった

じゃあ、MCPも検証！！

NWのコンテキストを理解させる

MCPとは？

MCP (Model Context Protocol)

生成AIに業務の背景やルールの文脈を明示的に与えるための仕組み

LLMが選択したツールを
MCPサーバーが
実行して情報取得

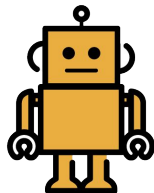
(question)
RT1のOSは？

(answer)
RT1はZ社のA-OSです

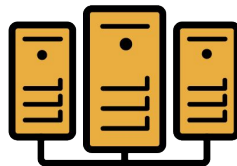
RT1にSSHして
コマンド実行する
ツールを使います。



MCPの説明



LLM



MCPサーバー

SSH

“show version”

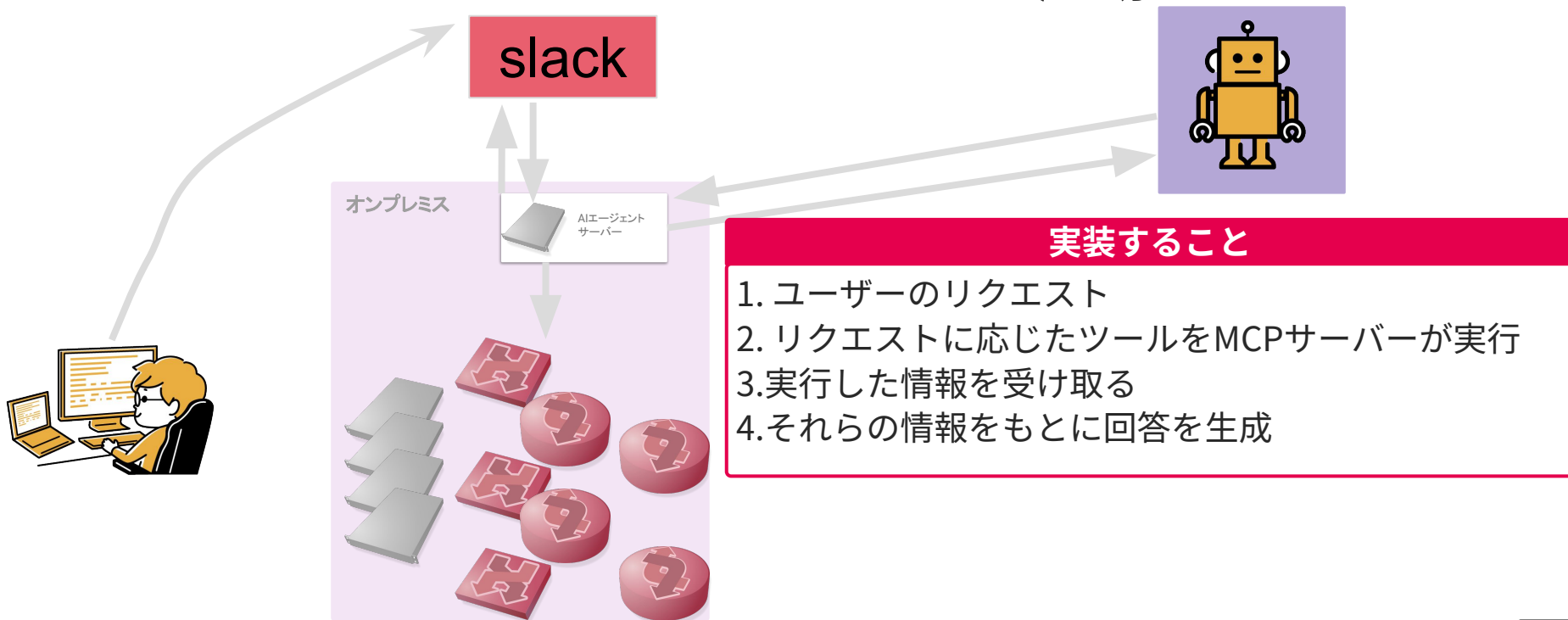


NW機器
RT1

[Introduction - Model Context Protocol \(MCP\)](#)

NWのコンテキストを理解させる

SDK AIエージェントとMCPサーバーを起動



NWのコンテキストを理解させる

SDK AIエージェントとMCPサーバーを起動

擬似障害:s1でBGPクリアを実施

 jun.egawa 🧐 18:11
netbot ai agent NW機器のログを解析して何か問題が起きていないか解析してください

やります 1 🧐

 netbot-staging アプリ 18:12
各ネットワーク機器 (I1~I8, s1, s2) のログを収集・解析しました。
直近～過去のログ傾向を踏まえ要点を整理します。

🔍 診断結果サマリー

****障害箇所**:** I系 (I1~I8) およびs1にてBGPとOSPF隣接の断続的な切断・再確立

****根本原因**:**

- 主要な要因は「管理者/ツールによるBGPピアの一括リセット」「物理I/F Downによる隣接断」等の運用操作・テストによる一時的な切断
- 子ルータでも複数回"interface went down"や"OSPF adjacency dropped"が同時多発
- "Hard Reset"等のイベント後、正常にEstablished状態へ戻っている

MCPサーバーが
ネットワーク機器にSSH接続
ログコマンドで情報取得

NWのコンテキストを理解させる

検証していく中で課題もあった

- 言葉の使い方で精度にブレがあった
- これを解決するために、定型プロンプトを用意
- 例えば、
 - リーフの下に接続しているサーバーの接続性チェック
 - -sでサーバー名を指定、サーバーが接続しているリーフの診断をするようにプロンプトとツールを実装

```
# If --server/-s is specified, generate the query automatically
if args.server:
    prompt = f"Please check the status of the leaf switches and interfaces hosting {args.server}."
```

NWのコンテキストを理解させる

精度のブレは定型プロンプトで解決

 jun.egawa 🌞 16:17
netbot ai agent -s sv1
やり
ます 1 🔄

 netbot-staging アプリ 16:17
状態確認結果

デバイス名: leaf switches (l1, l2)

確認日時: 2025-07-10 16:17:28

確認項目: サーバー sv1 に接続されている両方の leaf スイッチのインターフェース状態およびBGP状態

確認結果:

- l1 スイッチの Ethernet3 と Ethernet4 インターフェースは両方とも「up」で「connected」状態です。入力・出力エラーは検出されておらず、通信に問題はありません。

- l1 のBGPピアは 10.0.0.1 (s1) および 10.0.0.2 (s2) と確立状態 (Established) で、メッセージの送受信も正常です。

- l2 スイッチの Ethernet3 と Ethernet4 インターフェースも両方とも「up」で「connected」状態です。こちらもエラーは発生しておらず正常稼働しています。

- l2 のBGPピアも同様に 10.0.0.1, 10.0.0.2 と確立状態で問題ありません。

総合的に、サーバー sv1 に接続された leaf スイッチの物理インターフェースおよびBGP状態は正常であり、現時点で特段の問題は検出されていません。

定義したツール

- リーフとサーバーの接続表検索
- それらを参照してコマンド実行

NWのコンテキストを理解させる

参考：こんなこともあった・・・

```
s1#show ip bgp summary
BGP summary information for VRF default
Router identifier 10.0.0.1, local AS number 65000
Neighbor Status Codes: m - Under maintenance
  Description                Neighbor V AS      MsgRcvd  Msg
s1#
```

加工なしのデータ

「デバイスI1のBGPサマリは特に異常なく、
ネイバー状態関連の情報にエラーは見られませんでした」

```
s1#show ip bgp summary | json
{
  "vrfs": {
    "default": {
      "vrf": "default",
      "routerId": "10.0.0.1",
      "asn": "65000",
      "peers": {}
    }
  }
}
s1#
```

JSONのデータ

「BGPサマリーではピアは存在せず、BGPは確立されていない(peers: {})。」

JSON側は
peersが空白のため
何もない！と理解しやすい

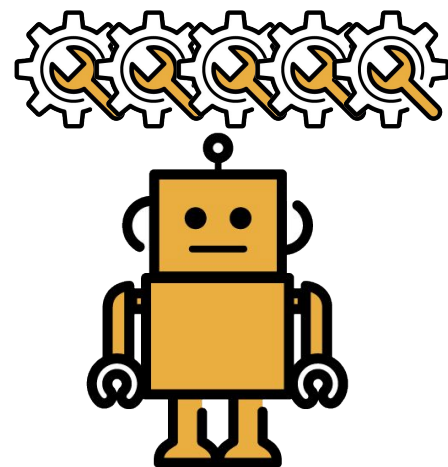
NWのコンテキストを理解させる

振り返り

- 残念ながら、まだ本番環境で動作してません
- ただ、解析や一次窓口の役割を果たしてくれそう
 - 文字通りのAIエージェント（代理人）

学んだこと

**正確でリアルタイムな情報を与える
精度がぶれない仕組みを作る**



得られたもの / お支払いしたもの

得られたもの

- 生成AI活用へのレッジ
- 余白時間の創出
- 業務のあり方を見直すきっかけ

お支払いしたもの

- 時間
- コスト（お金）

検証結果のまとめ

- プロンプトより仕組みで解決する
- 与えたい情報を最新に保つ情報管理
- 責任は人間が全て負う
- 確実に作業効率が上がった！



unenu commented last month

CodexでPRを作って、Copilotにレビューしてもらいました
内容は確認しましたので責任は江川にあります。
レビューお願いします。

**まだまだ道半ばですが
生成AIと共存する環境が
整ってきました！！**

これから

課題

- データの質と量
- ハルシネーション対策
- セキュリティとガバナンス
- ワークフローが回る=作業が増えるのでトラブル発生増？

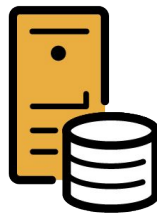
それでも

- 人間の余白時間がどんどん増えてくる
- 挑戦してどんどんネットワークの造詣を深めよ！
- より一層人間はクリエイティブになる意識を持つ

生成AI時代のデータ管理へシフトする覚悟



機密情報



コンフィグファイル



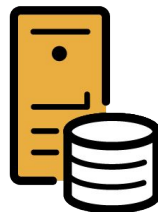
プロジェクト管理
チケット



監視データ



- データを最新に保つ工夫
- セキュリティや権限管理が課題



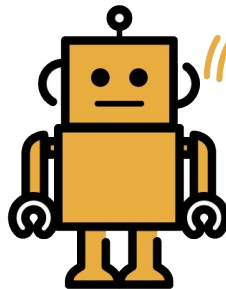
シスログデータ

未来のネットワークエンジニアリングの形

- チームに生成AIがジョインして一緒に働く
 - 私たちは監督者のスキルが求められる
- インフラをコードで表現することがもっと増えてくる
- 開発と運用の境目がなくなってくる（DevOps統合）



顧客や別部署
担当者



一次対応
AIエージェント



分からなければ
NW担当者呼び出し！



NW担当者

議論

- 生成AI活用で具体的にやっていることは何ですか？
 - 活用を始めてみて良かったこと
 - 苦労したこと、悩んでいること
- 生成AI活用の始めるまでの壁はありますか？
- ネットワークエンジニアが生成AIを活用していくために
 - やるべきことは何だと思いますか？
 - あると嬉しいサービス / ツールは何だと思いますか？
- 今後、生成AIに置き換えられることは何だと思いますか？
- 逆に置き換えられないことは何だと思いますか？

心もつなごう。

MIXI

