JANOG58 Meeting

ネットワークOSの運用ライフサイクルを考える - AIを活用したネットワークOS選定の省力化

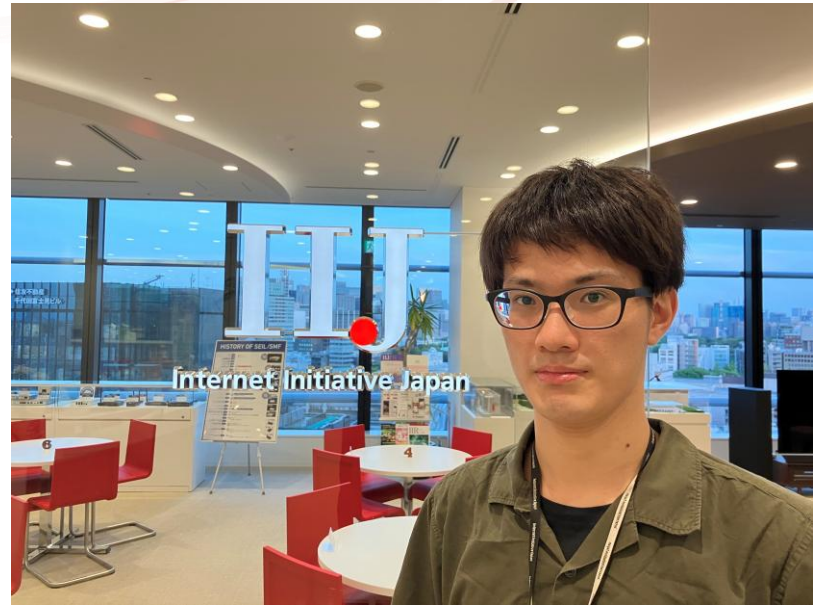
Internet Initiative Japan

2026年7月7日

株式会社インターネットイニシアティブ
ネットワークサービス事業本部 基盤エンジニアリング本部
ネットワーク技術部
小田川 匠, 蓬田 裕一

小田川　匠

株式会社インターネットイニシアティブ
ネットワークサービス事業本部 基盤エンジニアリング本部
ネットワーク技術部 ネットワーク技術1課



- **2023年度 IIJ新卒入社**

- **AS2497 Backbone 設備担当**
  - インターネット接続サービス(IP)のルータ設計・構築・運用
  - 新規機種・OSの動作検証・導入
  - 運用ツールの開発・保守

- **趣味**
  - 新しいことに取り組むこと
    - アプリケーション開発、LLMのSFT(ファインチューニング)、サーバ運用、イベント参加など
  - 旅行
    - 温泉地に行くこと

経歴

- 蓬田 裕一 (よもぎた ゆういち)
- 2008-2014 IIJ
  - トランジットサービス運営
  - バックボーン運用
- 2015-2019 JPNAP
  - IXサービスの運営・設計・構築・運用
- 2019- IIJ
  - IIJバックボーンチームマネージャー (AS2497)
  - Peering/Interconnection マネージャー

AS2497バックボーン

- バックボーンネットワーク
  - 新規POP設置・移設・廃止の企画・設計
  - 新機種選定、検証、新機能導入
  - インターネット接続系サービス設備の運営
  - 内製アプリ開発のProject Manager
- 日々の構築・運用業務
  - AS2497の機器オペレーションや障害対応
  - 運用フローの設計や導入
  - 承認業務

趣味

- おいしいごはんを探して食べる
- おいしかったご飯を自宅でもつくってみる
- たびに出る(温泉・観光・知らない街へ)
- プロレスを見る、実際に見に行く
- 好きなYOUTUBEを見る

対外対応

- Peering戦略検討と交渉
  - 顧客、ニーズを意識した戦略を検討
  - Peer接続の交渉
- 社外交渉
  - Peering以外にも、海外回線調達、海外機器の保守事業者選定、データセンター選定
- 社外コラボレーション
  - 他社とのサービス協業の検討

本日のアジェンダ

1. 背景と目的
2. AIの影響と昨今のNOSリリース状況
3. IIJのNOS選定ポリシー
4. NOSを導入するまでの工程、理想、課題感
5. 自動検証ツールの設計共有
6. デモ
7. まとめと今後の展望
8. ディスカッション

- **本日はネットワーク機器(主にルータ)について話します**
- **我々は定期的なネットワークOS(以降NOS)のアップグレードが求められている**
  - 機能改善や機能追加
  - NOSが古いとサポートが受けられない
  - セキュリティ的な問題が内在し続けるとマズイ

- **そこに、昨今はフロンティアAIの影響が..**

フロンティアAIがNOSライフサイクルへ及ぼす影響を考察

1 脆弱性発見の自動化

AIがNOSのソースコードやバイナリを解析し、人間では発見できない脆弱性を自動で発見

2 リリースサイクルの高速化

AIを使ってリリースサイクルを高速化し、脆弱性に追従したNOSのリリース対応を行う可能性

3 ライフサイクル検討の自律化

NOSリリース状況やバグ調査を自律的に実施。アップグレード計画の策定と作成をAIが行う

4 ゼロデイ攻撃の事前検知

運用者自身がAIで自身のネットワークノードを検査し、ゼロデイ攻撃へ備える

1 脆弱性・設定ミスの悪用

AIがNOSの脆弱性や設定ファイルを解析し、攻撃に利用可能な設定ミスを発見・悪用

2 アップグレードサイクルの高速化

メンテナンスや顧客調整といった人間側のボトルネックがあり、実施が追いつかない

3 攻撃ペイロードの生成

フロンティアAIが標的のネットワークOSに特化した攻撃ペイロード・コードを自動生成し利用

4 攻撃の自動化・大規模化

AIが攻撃プロセス全体を自動化し、複数ベンダー・複数OSへの同時攻撃を行う可能性

脅威は攻撃者だけにあるのではない。防御側もAIを活用していく必要がある

昨今のNOSリリース状況

Arista EOS	**年2～3本のペースでリリース**トレインを提供。各メジャートレインは初回リリースから**36か月間のサポート**。EOSはトレイン単位で管理され、例えば EOS 4.30 は 2026年4月14日、EOS 4.32 は 2027年4月9日にサポート終了予定。
Cisco IOS XE	17.x系まで年3回のタイムベースリリースを採用し、3リリースごと(17.3、17.6・・)に**48か月のExtended Supportリリース**を提供。2026年から「26.x」命名へ移行し、**リリース頻度を年2回へ変更**。すべてのリリースをGA後**48か月サポート**のExtended Supportリリースとして提供する方針。
Cisco IOS XR	2024年の24.1.1リリースから四半期ごとのリリースモデルへ移行し、年間4本のFeature Release（FR）と**年間4本のExtended Maintenance Release（EMR）**を提供。FRは約24か月、**EMRは約36か月のメンテナンスサポート期間**。EMRはFRの約90日後に提供され、Ciscoは新規導入・長期運用用途ではEMRを推奨
HPE Junos OS	23.2R1以降、**年2回（6月・12月）**の**EEOL（Extended End of Life）リリース**モデルへ。EEOLリリースはGAから60か月のエンジニアリングサポートと、さらに6か月のカスタマーサポート期間あり。23.2、23.4、24.2、24.4、25.2といったEEOL版は**実質5年超の長期サポートリリース**として提供。
Nokia SR OS	**年3回（x.3、x.7、x.10）のリリースサイクル**。各リリースでR2、R3などのメンテナンスリリースを継続提供、運用上は各年最後の**x.10系列が長期運用向けトレイン**。23.10.R2、23.10.R3などの継続的なメンテナンスリリースを提供。

- **マルチバージョン**
  - 0系と1系でメジャーバージョンをわける
  - 0系と1系が同時に同じ脆弱性の影響を受けるリスク低減
- **安定バージョンの選択**
  - メジャーバージョンの1st Releaseは避ける
  - できればマイナーバージョンが2回以降リリースされたものを選択。長期運用トレインが理想
- **事前の検証が必須**
  - 開発環境 および 本番環境の動作確認と判定

1 OSバージョンの選定
使用するOSを決定

1-1 リリースノートの調査
新機能・変更点・既知の不具合を調査

2 導入前検証の実施
選定したOSバージョンが要件を満たすか確認

2-1 バグスクラブ
既知バグの存在確認と影響評価

2-2 必要機能検証
必要機能がIIJの利用構成で想定通り動くことを確認

3 本番環境での実働試験

実環境と同等の条件での一定期間の動作確認
(ルーティング、監視、リソース情報取得)

4 リリース対応

検証結果に基づいた承認プロセスの実施

- **2年1回のNOSのバージョンアップ**
  - 0系 → 1系 → 0系 → 1系 → ・・・ で定期化

- **全NOSのリリースに合わせて全部検証する**
  - パッチも含めてリリースされたら全部検証
  - 計画的な入れ替え以外でのNOS利用に備える
  - できる限り早めに使えるか使えないかを判定する

NOSを導入するまでの工程 (それぞれの課題感)(1/2)

1 **OSバージョンの選定**

使用

1-1 **リリ**

新機能

マルチベンダー・マルチバージョン対応
による稼働増
リリースノートの解析難易度高

2 **導入**

選定

複数機器や機能による膨大なバグをスクラブする稼働捻出

2-1 **バグ**

既知バグの存在確認と影響評価

2-2 **必要**

必要

そもそも機能検証に時間がかかる
全リリースバージョンを検査できない

3 **本番環境での実働試験**

実環

(ルー

NOSの入れ替え対応と確認もオペレータによるマニュアル実行

4 **リリース対応**

検証

厳密な承認プロセスに向けたドキュメント作成とレビューも作る手間がかかる

- **業務でもAIが普通に使える時代に！**

- **AIが得意なことはAIへ任せよう！**
  - 24h/365dの稼働
  - 膨大な情報の処理と判定
  - ドキュメントの生成

NOS導入までの既存のスケジュール

- **機能検証に多く時間がかかっている**
  - 手動で全ての対応を実施しているため工数が膨大
  - AI・ツールを使って機能検証の工数を削減したい

- **設計方針**

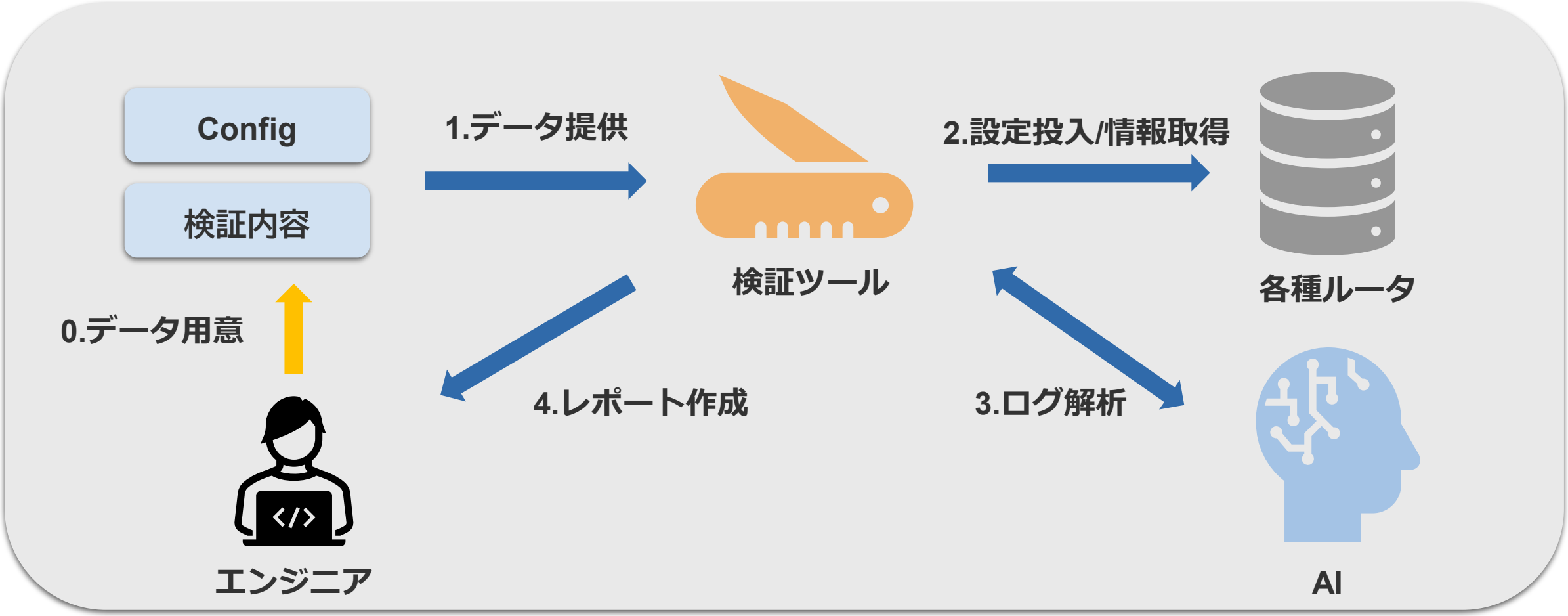
検証を自動・並列で実行できること

- 業務時間外・別業務中に機能検証を自動・並列実行
- 使用者の工数を気にせずに実行可能

ツールのメンテナンス・利用が容易であること

- 保守にかかる工数 > 減った工数 では削減出来ていない
- 理想は誰でも管理・利用ができる状態

- **ツールの処理について**
  - 使用するConfigテンプレートと検証内容は人が用意
  - それ以外はAI+ツールで全て自動実行



- **AIとコードでの実装は得意分野が異なる**
  - それぞれの得意分野を処理させる

	AI	コードでの実装
得意	• 膨大な情報の処理と判定 • ドキュメントの生成 • 人が読むことを想定された情報の処理	• 精度が求められる処理 • 実行速度が求められる処理
苦手	• NOSのConfig作成(2026/07現在) • 精度が求められる処理 • 実行速度が求められる処理	• 人が読むことを想定された情報の処理 • ドキュメントの生成

- **検証の際に使用するConfigテンプレート**
  - AIに作成させると複雑なものは間違いが発生
    - 人が作成してプログラムで投入する形式に
    - 既存の運用しているConfigがあるため、テンプレート作成は容易

- **機器の動作確認**
  - プログラムで書こうとすると膨大な手間が発生
    - 大量の正規表現、バージョン更新による出力結果の変更対応
  - AIで判定・レポートを作成し、人がレビューする形式に

- **ツールの良い点**
  - 機能検証期間の短縮
    - 1.5~2ヵ月から数日へ
    - 自動実行のため人がほとんど介在せず処理可能
  - AIを最大限活用して、情報処理
    - AIがログを処理して判断している
      - ex1：show interfaceと検証情報をAIに入れることで、AIが判定を実施
      - ex2：NOSのバージョンに関わらず、ログの解析が可能
  - ツールとAIを組み合わせたレポート作成機能
    - エンジニアは検証結果を読むだけで完了

- **ツールの悪い点・改善点**
  - 検証レポートの精度
    - 90%以上は正しいが、間違った結果を返してくる場合がある
    - 似たようなパラメータがある項目は間違えやすい
    - 検証内容(指示内容)が曖昧な場合も勝手な解釈をすることがある
      - ex：ACLで特定のトラフィックを弾く場合
    - より高性能なモデルへの変更や、モデルに対してCPT/SFTの実施、MCP/RAG等を用いることで改善可能
  - AIやハードウェアの性能に左右されやすい
    - LLMを動作させるハードウェアが必要。モデルの性能が低いと適切な評価が出来ない

- **機能検証の期間を大幅に短縮**
  - AI+ツールが機能検証を行うことで工数を大幅に削減

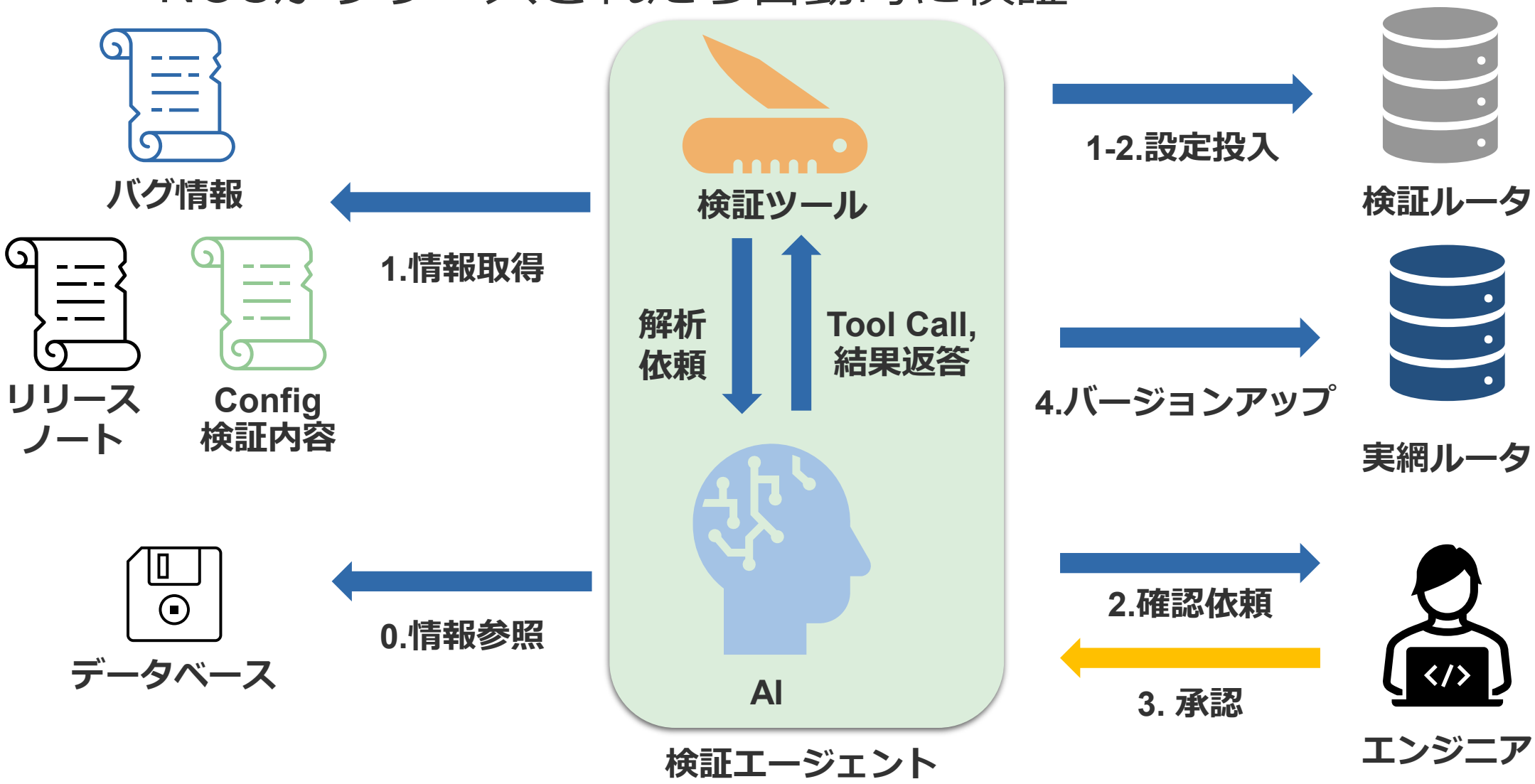
今後の展望(1/2)

- **NOS選定業務の100%自動化**
  - 検証が問題なかったNOSを自由に選べる環境
  - 現在バグスクラブ効率化の開発を実施中

- **検証ツールの適用範囲の拡大**
  - AS2497で運用している全てのNOSへの適用

今後の展望(2/2)

- **NOS選定業務の100%自動化**
  - NOSがリリースされたら自動的に検証



ディスカッション

- **NOSのライフサイクルについてどのようなポリシーをお持ちですか？またどういった基準でNOSのバージョンを選定していますか？**
- **機器検証やNOS検証を実施する際に、どういったフローで行っていますか？**
- **どの程度の工数や人数、期間をかけてNOSの検証を実施していますか？**
- **AI(やAI Agent)での検証やバグサーチ、ログ解析を実施していますか？ユースケースがあれば教えてください。**
- **自動化ツールのメンテナンス・コードの保守管理はどういった体制で実施していますか？**

日本のインターネットは1992年、IIJとともにはじまりました。以来、IIJグループはネットワーク社会の基盤をつくり、技術力でその発展を支えてきました。インターネットの未来を想い、新たなイノベーションに挑戦し続けていく。それは、つねに先駆者としてインターネットの可能性を切り拓いてきたIIJの、これからも変わることのない姿勢です。IIJの真ん中のIはイニシアティブ――IIJはいつもはじまりであり、未来です。