JANOG55 発表資料

JANOG55 Meeting@京都 みやこめっせ

ネットワーク作業自動化の道: 信頼性と効率性の両立

KDDI株式会社

岸 祐斗　長野 知幸　藪原 穣

2025年1月23日

本日お話しすること

1. はじめに
- Executive Summary
- 自動化に立ちはだかる壁

2. 背景
- 背景
- 自動化対象

3. プロセスによる解決策
- 常識に捉われないクロスファンクショナルチーム
- トライアル&エラーによる開発手法
- チーム結成にあたっての苦労話
- ネットワークエンジニア×ソフトウェアスキル
- 継続的な自動化・品質向上の実施

4. 内製開発プロダクト概要
- プロダクトのコンセプト
- 自動化支援プラットフォーム
- 自動化の前後比較
- 工夫ポイント

5. デモンストレーション

6. おわりに
- 商用作業を終えて
- プロジェクト参画者の感想
- 今後の展望
- 議論ポイント

登壇者紹介

岸　祐斗
(Yuto Kishi)
IPネットワーク部
yu-kishi@kddi.com

長野　知幸
(Tomoyuki Nagano)
ソフトウェア開発部
to-nagano@kddi.com

藪原　穣
(Osamu Yabuhara)
ソフトウェア開発部
os-yabuhara@kddi.com

1. はじめに

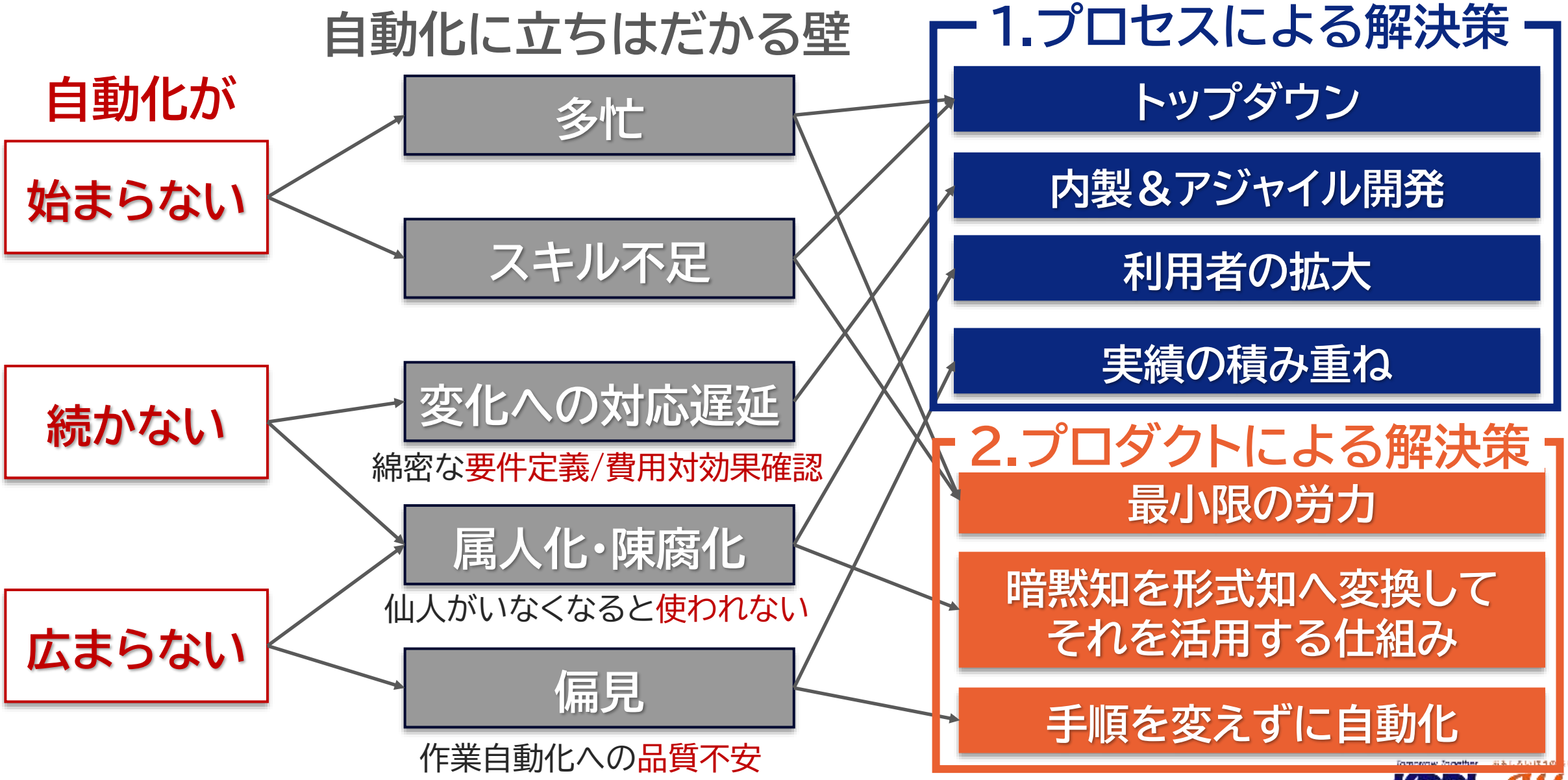
Executive Summary

大規模ネットワークであるがゆえに、**安定性・高信頼性**を求められ、
人手作業からの脱却にハードルを感じていたネットワークエンジニアが
ソフトウェアエンジニアとコラボレーションして乗り越えた話

成し遂げたこと

- ネットワークエンジニア×ソフトウェアエンジニアの
  内製クロスファンクショナルチームにより、
  固定概念に捉われないソリューションを導くことができた
- Jupyterを用いた自動化支援クラウドプラットフォームにより、
  ネットワークエンジニアが継続して自動化できる環境の整備と
  動機付けができた

自動化に立ちはだかる壁とその解決策



2. 背景

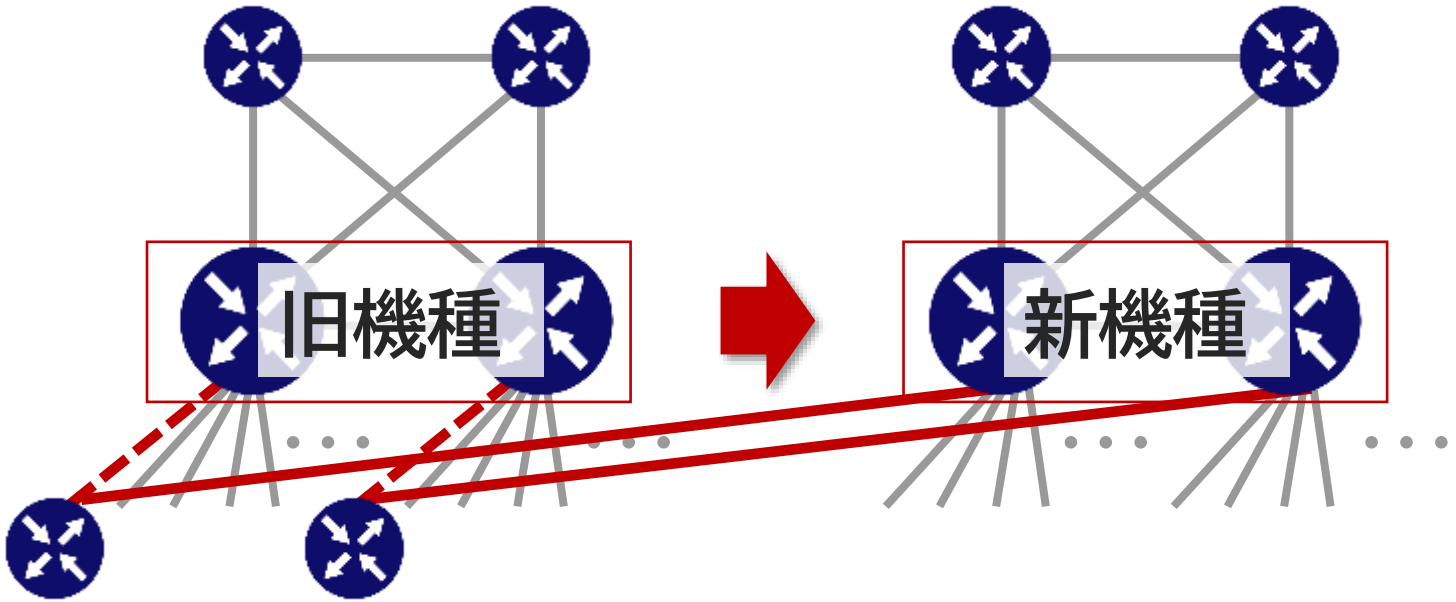
背景

- KDDIのネットワークは大規模であるがゆえに、少しの作業ミスが大規模障害に発展するリスクがあり、**実績のある手順を変えることをリスク**としていた
- 一方で、多様化するサービスニーズに応えるために、ネットワークの作業数も増えており、従来の**手作業では対応しきれなくなってきている**
- これまでも自動化を進めていたが・・・
  - ソフトウェアスキルのないメンバは**始められていなかった**
  - 属人化により担当者がいなくなると**続かなかったり**、他のメンバ/組織に**広まらなかった**

自動化対象

■ 対象作業

- モバイルバックホールルータの機器更改に伴うマイグレーション作業
- 作業回数は**数百**規模



■ 対象工程

- **作業準備**：手作業でExcel手順書を作成
- **作業実施**：実績のある慎重なステップを踏んだ手順を手作業で実施

3. プロセスによる解決策

常識に捉われないクロスファンクショナルチーム

- ネットワーク（NW）エンジニアとソフトウェア（SW）エンジニアで構成された**部門横断のクロスファンクショナルな開発チーム**を**トップダウン**で結成
- SWエンジニアによる客観的な視点により、NWエンジニアの**固定概念に捉われないソリューション**を導くことができた

NWエンジニア
SWエンジニア

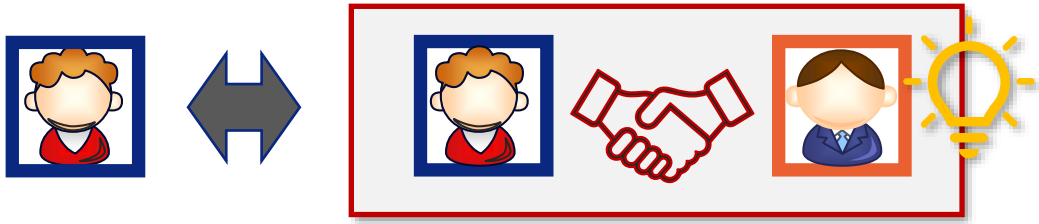
（これまで）NWエンジニアのみ

▲ すきま時間での開発
→自動化が**始まらない、完成しない**

▲ NWエンジニアの固定概念
→**特定作業単発**の自動化ツール

NWエンジニア×SWエンジニア

常識に捉われないクロスファンクショナルな開発チーム



○ 体制を組んで開発
→自動化に**注力できる、完成する**

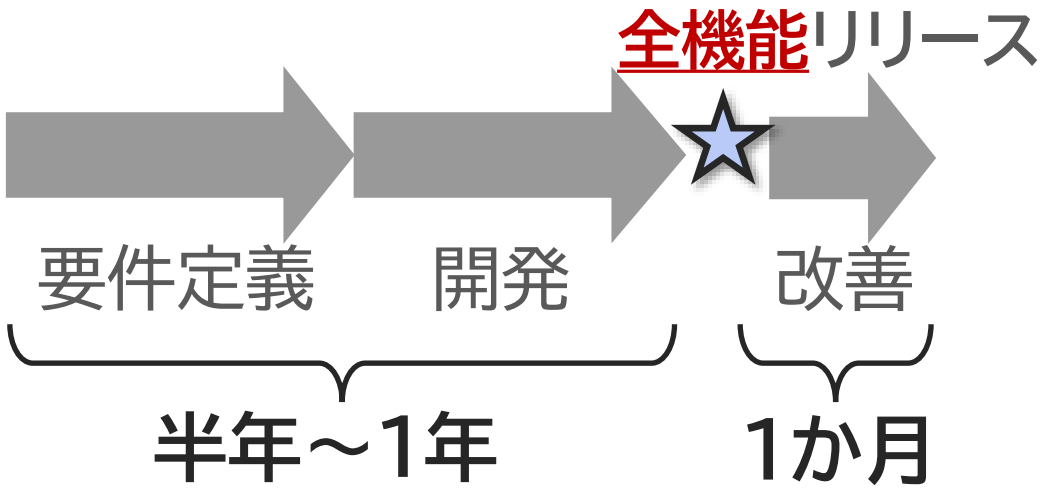
○ SWエンジニアによる客観的な視点
→**他作業でも利用可能**な自動化設計

トライアル&エラーによる開発手法

■ 内製でのアジャイル開発

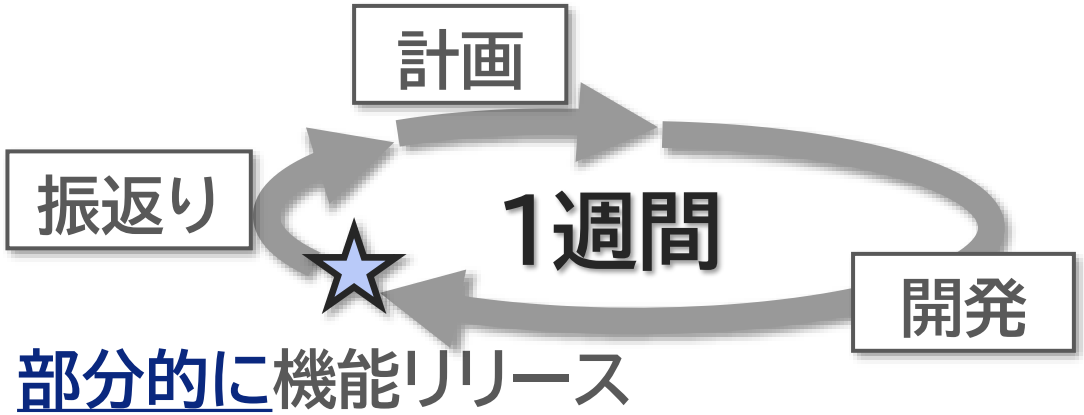
● 1週間単位で**機能を小リリース、PDCA方式**へ変えた

従来のベンダ開発（ウォーターフォール開発）



使えるようになるまで時間がかかる

内製アジャイル開発



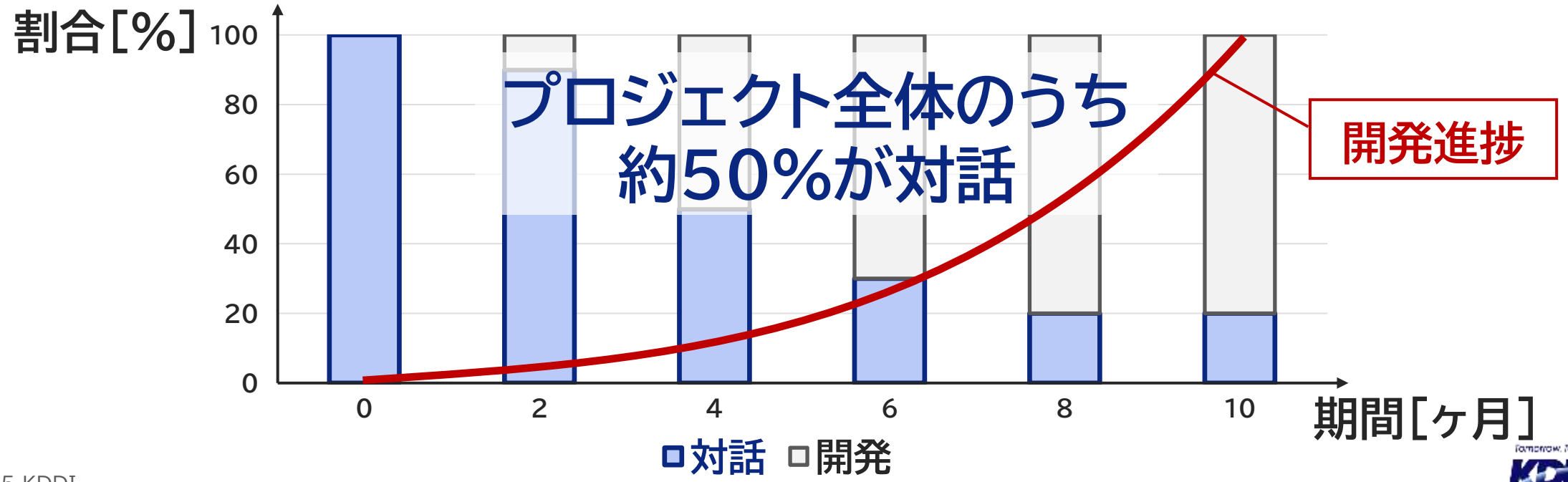
すぐに使える
→改善フィードバックにも迅速に対応

チーム結成にあたっての苦労話

バックグラウンドの違いによる衝突が発生

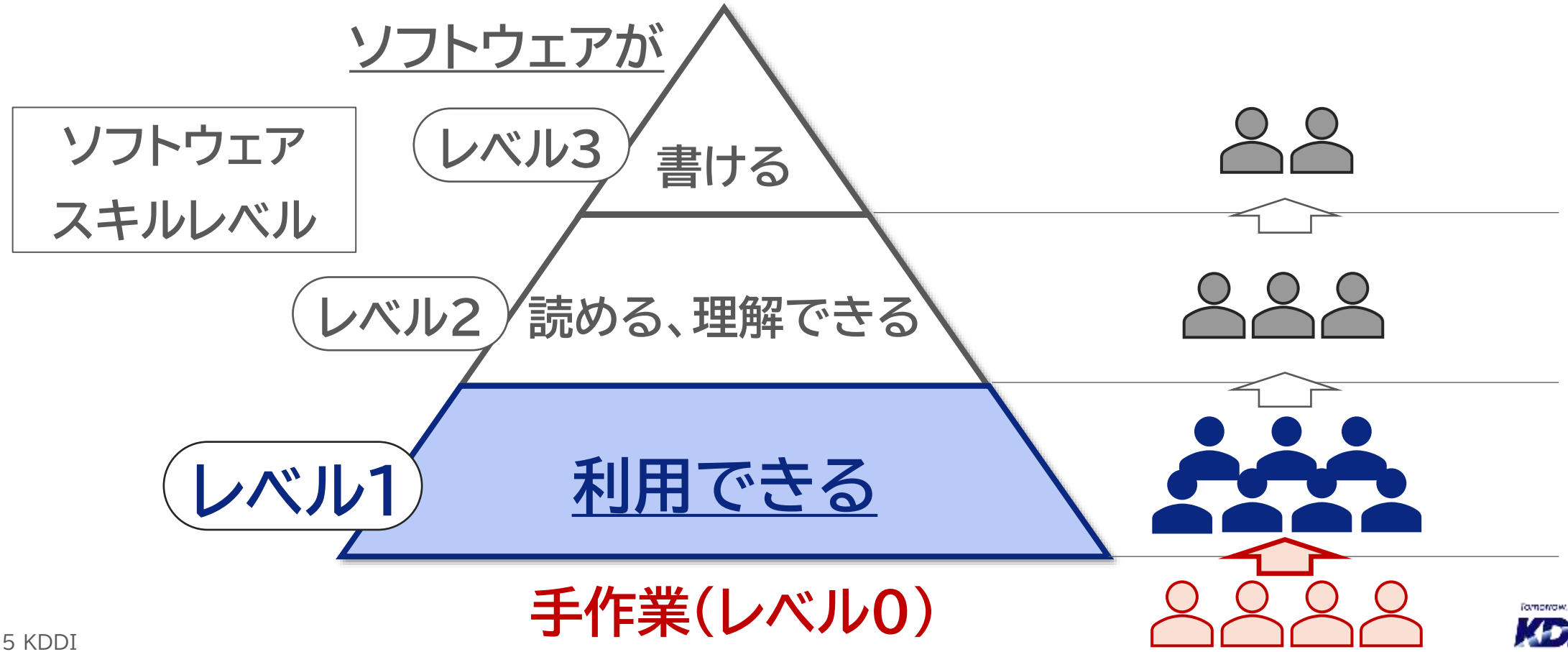
→ **対話**を重ね、両者の**意見のすり合わせ**を重視

NWエンジニア	SWエンジニア
目先の作業の迅速な自動化	費用対効果が最大となる自動化



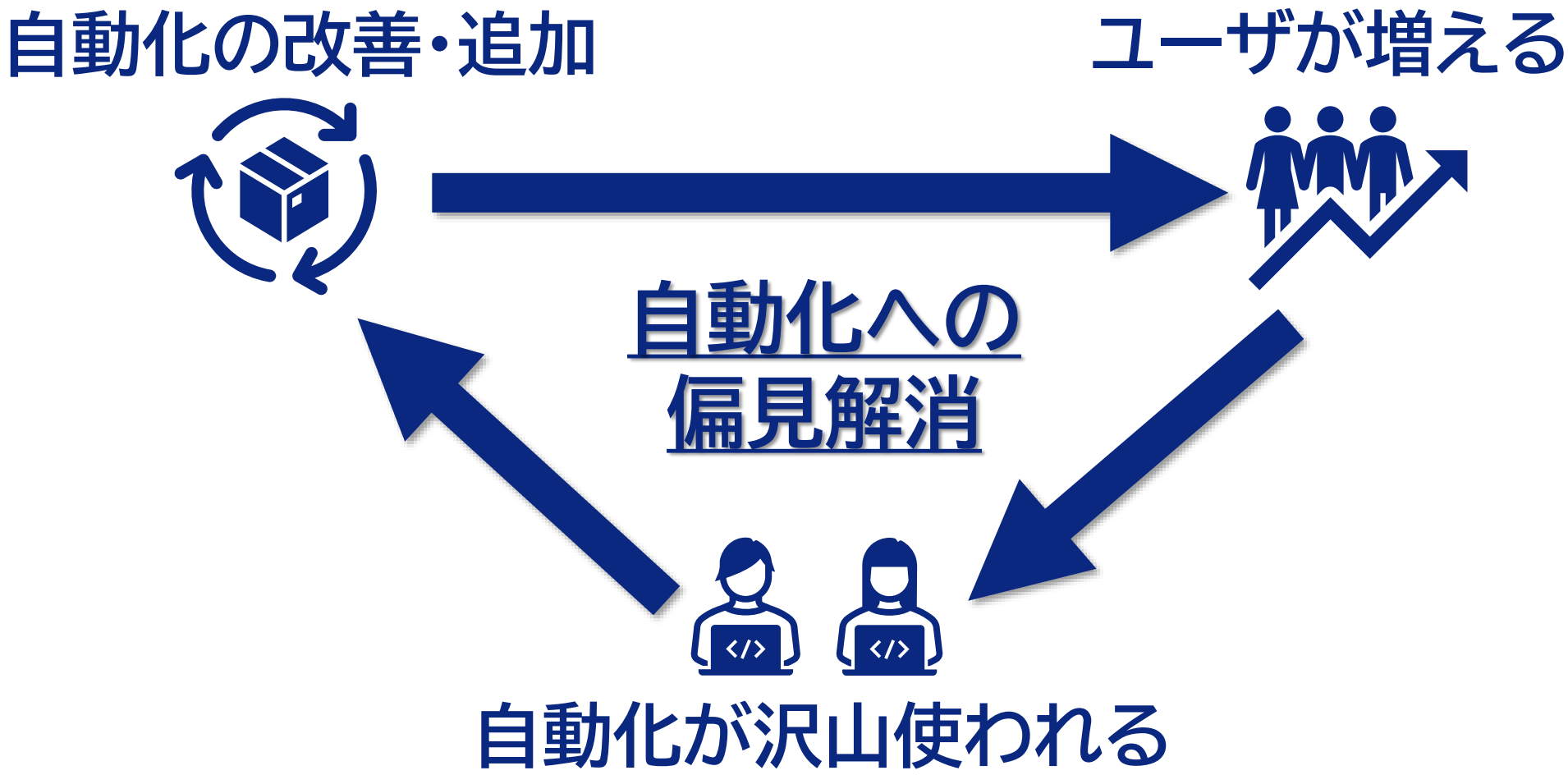
ネットワークエンジニア×ソフトウェアスキル

- ネットワークエンジニアもソフトウェアスキルを身に付ける必要がある
- まずはソフトウェア**利用者を増やし、**
  その後、ソフトウェアの読み書きができる人を増やしていく



継続的な自動化・品質向上の実施

ネットワークエンジニアが、業務の自動化・品質向上を**継続的に**実施していく



4. プロダクトによる解決策

プロダクトのコンセプト

■ ネットワークエンジニアが、**すぐに使えて、改善を継続的に実施できる自動化支援プラットフォーム**

＝開発チームが自動化を行う

プラットフォームの要件

多忙
スキル不足
属人化・陳腐化
偏見

最小限の労力

▲環境準備の手間、学習コスト
✓すぐに利用でき、誰が実行しても同じ結果に
✓理解が容易で汎用性が高い

**暗黙知を形式知へ変換して
それを活用する仕組み**

▲動作がブラックボックス
✓蓄積されているノウハウを容易に活用

手順を変えずに自動化

▲動作検証を一から再度行う手間
✓実績のある手順をそのまま自動化

ベースとなるプロダクトの比較

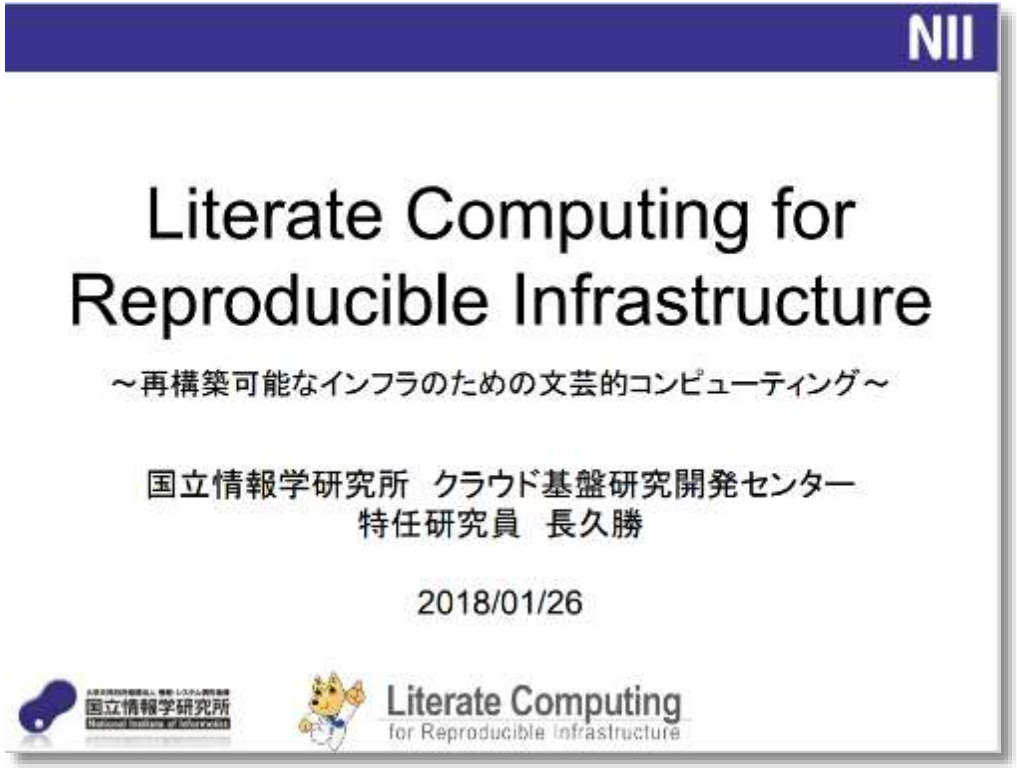
■ 逐次、作業者が結果を確認しながら手順を実行していく、という従来の手順を踏襲できる点より、Jupyterを採用することが、最も良いと判断

比較項目	Jupyter（Python）	汎用ベンダ製品	Ansible	既存の独自自動化
理解（習得）の容易性	○ 情報多＋研修	× 情報少	△ 情報多	× 情報少
他作業への拡張性（汎用性）	○ Python	× NW	△ ssh	× NW
形式知化、再利用化の容易性	○ Markdown+コード+実行結果	△ コード	△ コード	△ Excel
過去手順との親和性	○ 逐次	× 一斉	× 一斉	× 一斉

JANOG41:LC4RI

■ JANOG41にて、Jupyterによる手順書が有用と発表済み

→ 我々の判断の後押しになった

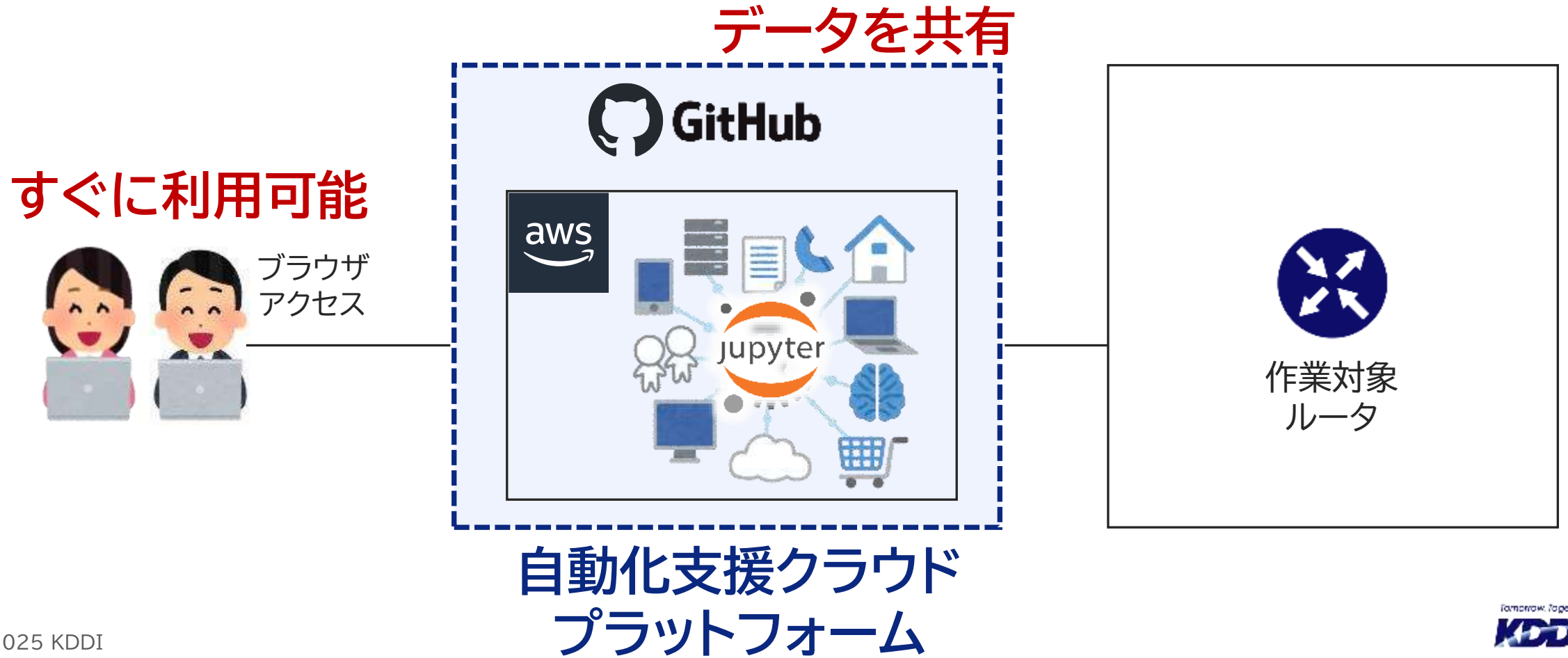


長久勝:” Literate Computing for Reproducible Infrastructure”, JANOG41

https://www.janog.gr.jp/meeting/janog41/program/sp8lcri

アーキテクチャー

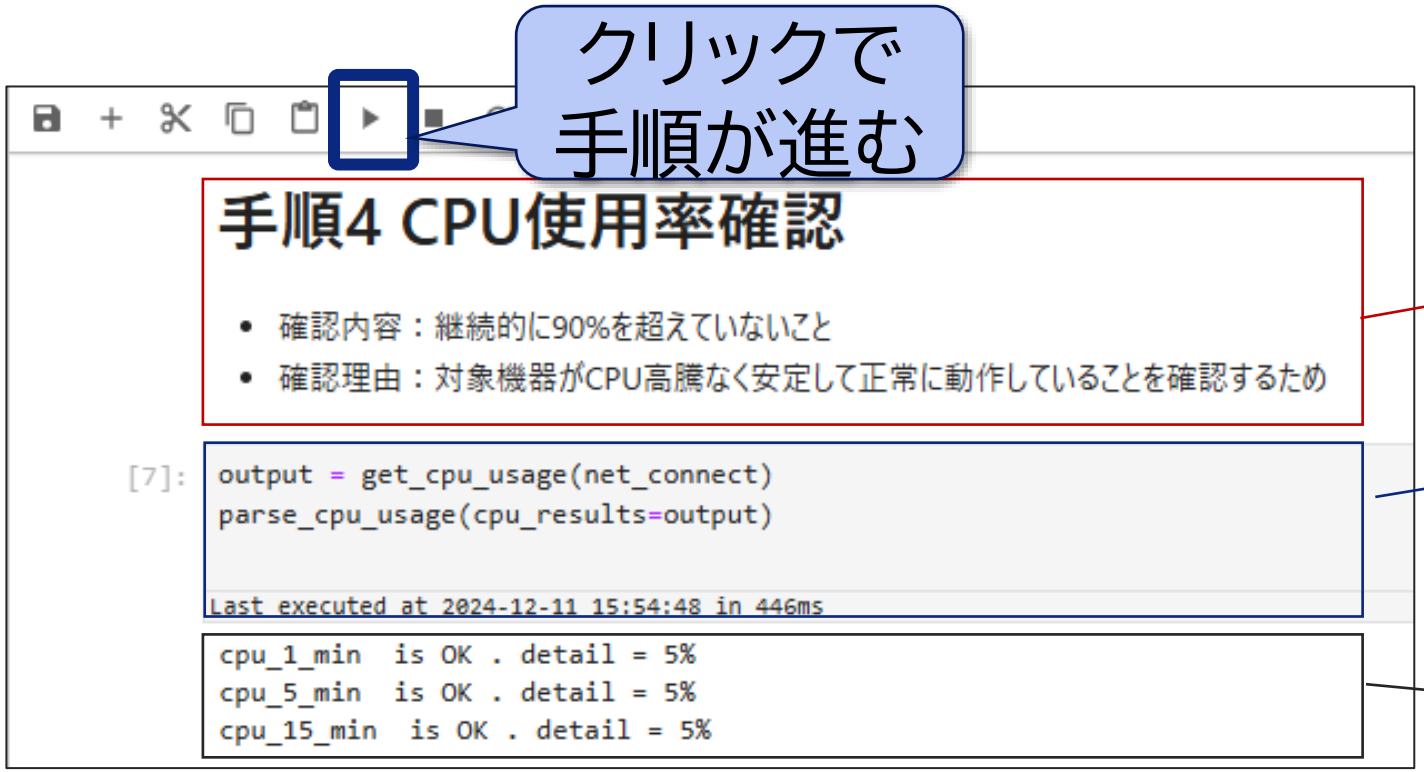
■ クラウド（AWS）上にJupyterを構築し、データも共有可能な自動化支援クラウドプラットフォームを提供



Jupyter手順書

■ 従来の手順を、そのままJupyter手順書(※1)とすることで、作業品質を維持したうえで、効率化を実現

（※1）Jupyter手順書：Jupyter Notebook形式の作業手順書。
MarkdownによるメモとPythonコードが一体化したもの。



Jupyter手順書作成の効率化

■手順書は固定値と可変値で構成され、ステップごとに実施。

■再利用可能な「パーツ」に分解し、可変値を別途定義する仕組みにより、効率的な手順書作成を実現

Before

従来の手順書イメージ

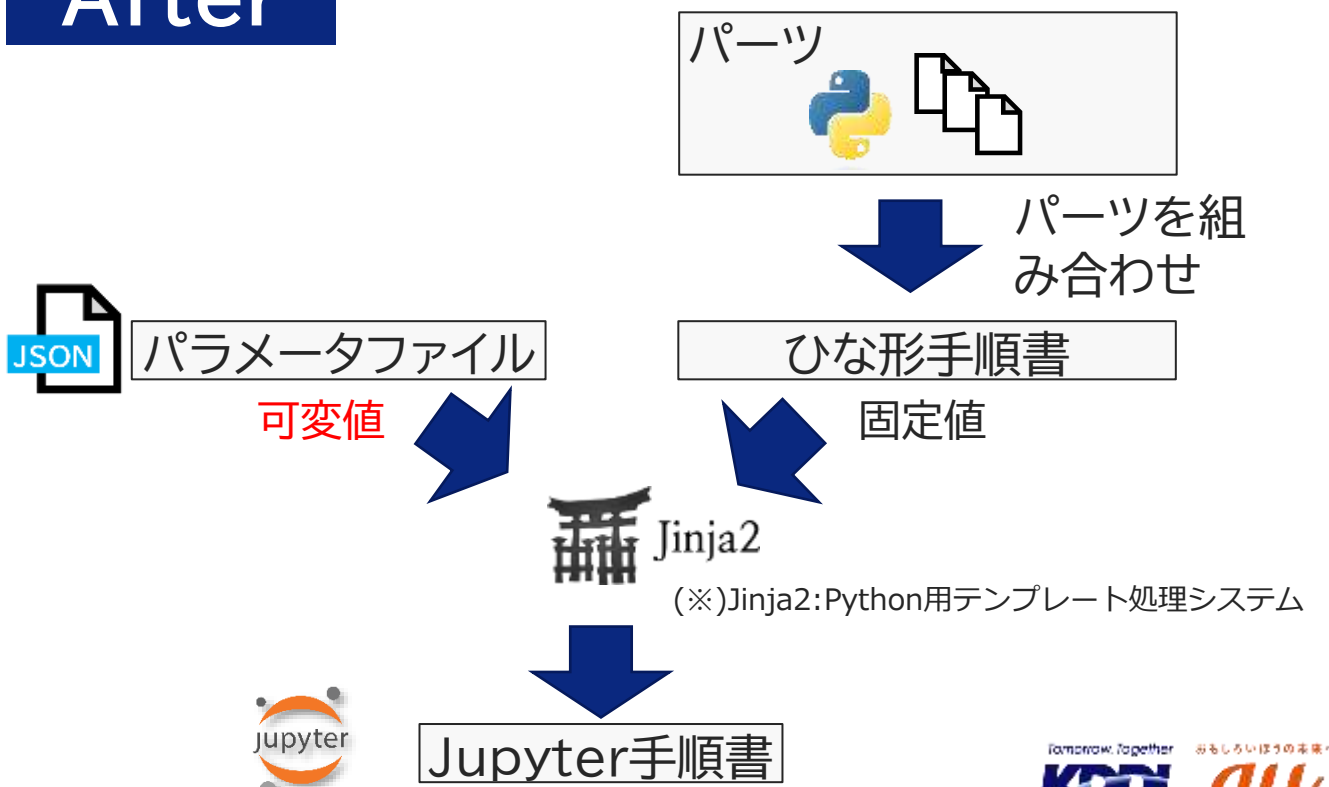
設定変更	configure terminal	
	stepX-X.txt	※設定コンフィグ
	show configuration	設定内容が正し
	commit	
	end	
事後状態確認	show running-config	直前取得のcon
	show bgp XX.XX.XX.XX/XX bestpath-compare	BestPath#が"
	show bgp YY.YY.YY.YY/YY bestpath-compare	

凡例

可変値

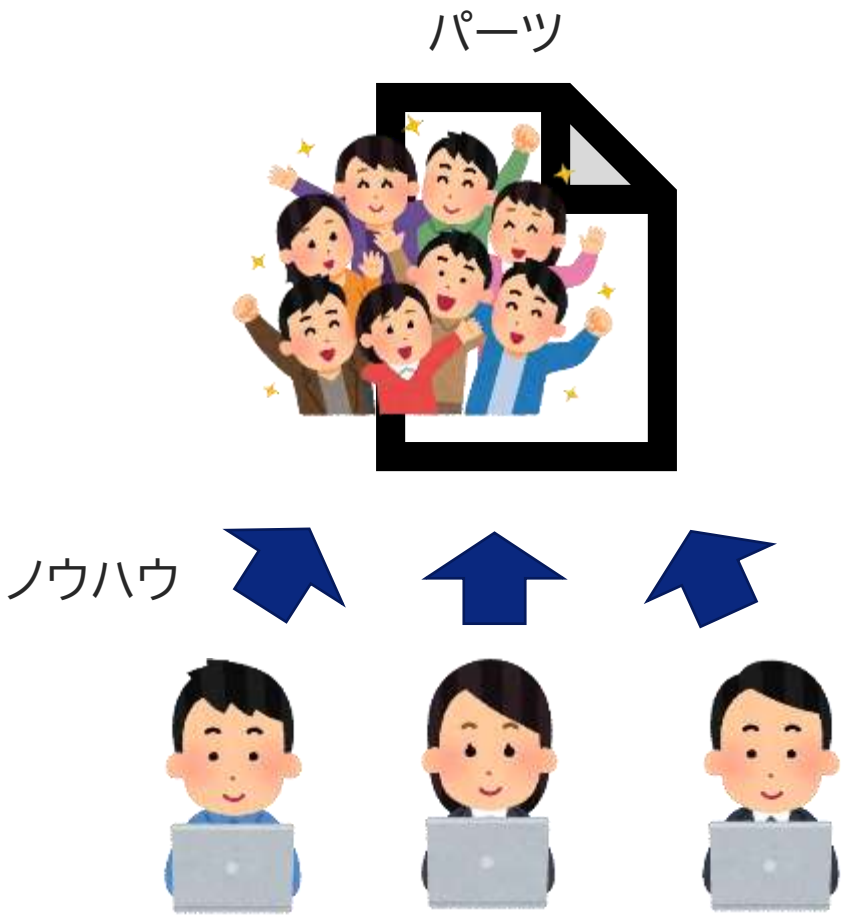
固定値

After



暗黙知から形式知へ

■ パーツ(Markdownメモ)にノウハウを記録することで、個人の暗黙知を形式知として共有可能に。



Markdownメモ

手順4 CPU使用率確認

- 確認内容：継続的に90%を超えていないこと
- 確認理由：対象機器がCPU高騰なく安定して正常に動作していることを確認するため

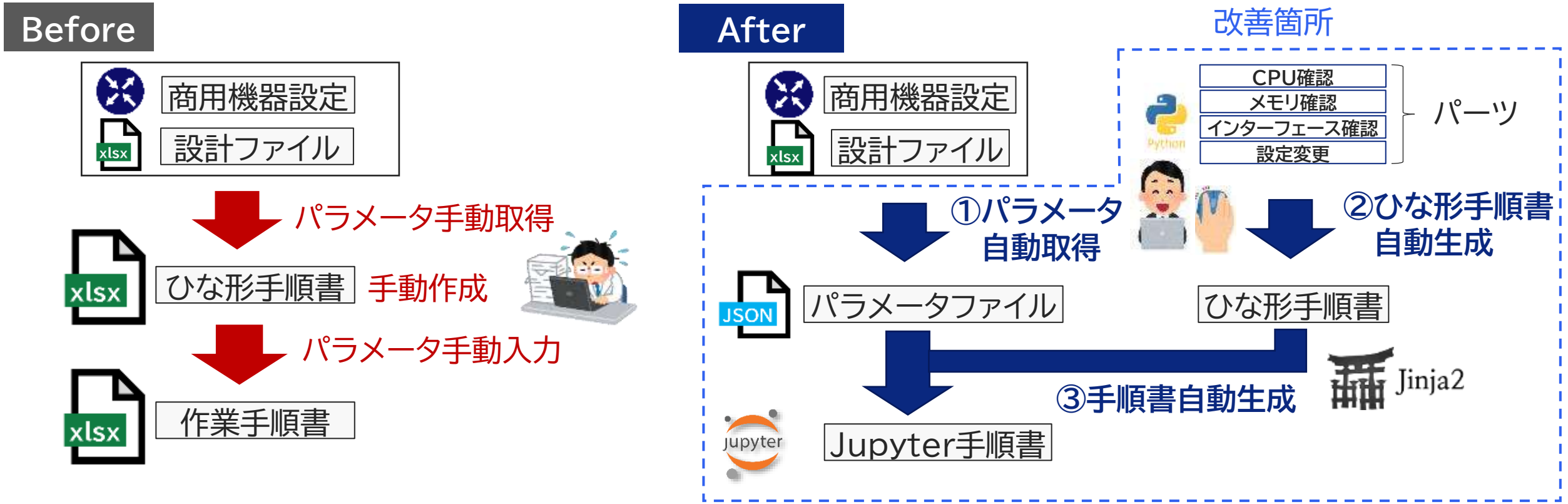
```
[7]: output = get_cpu_usage(net_connect)
     parse_cpu_usage(cpu_results=output)

Last executed at 2024-12-11 15:54:48 in 446ms

cpu_1_min  is OK . detail = 5%
cpu_5_min  is OK . detail = 5%
cpu_15_min  is OK . detail = 5%
```

作業準備の変化(Before→After)

①パラメータの自動取得(パラメータファイル自動作成)

②ひな形手順書を自動作成

③Jinja2を用いて、Jupyter手順書を自動生成

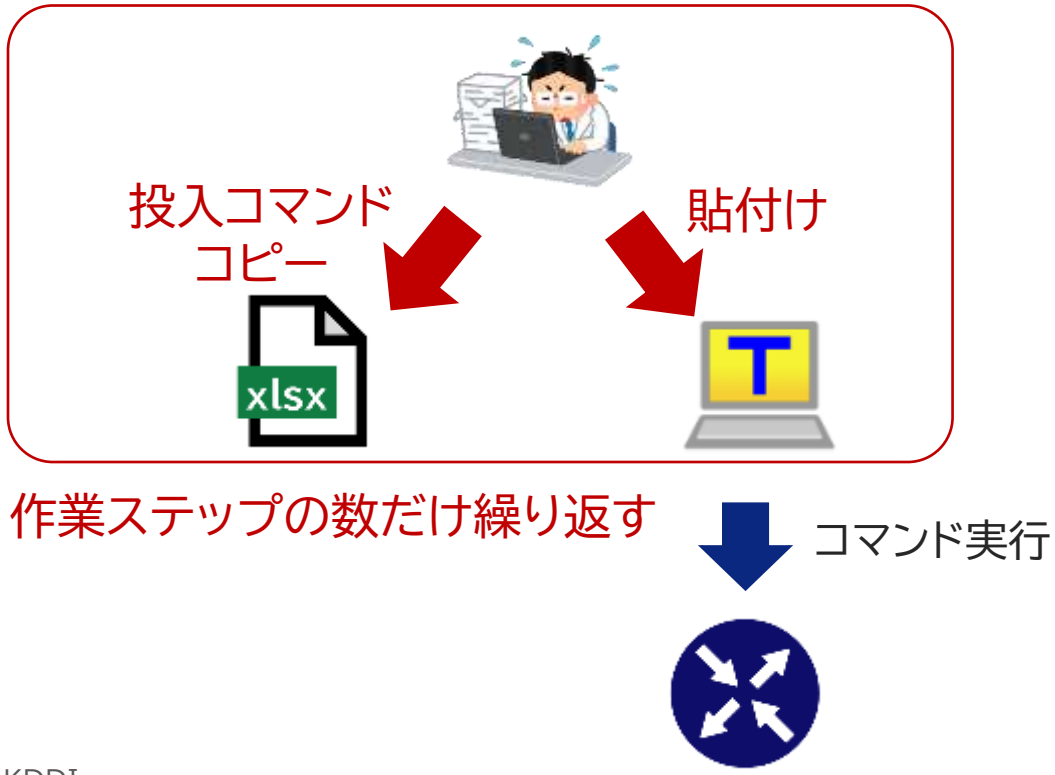


作業実施の変化（Before→After）

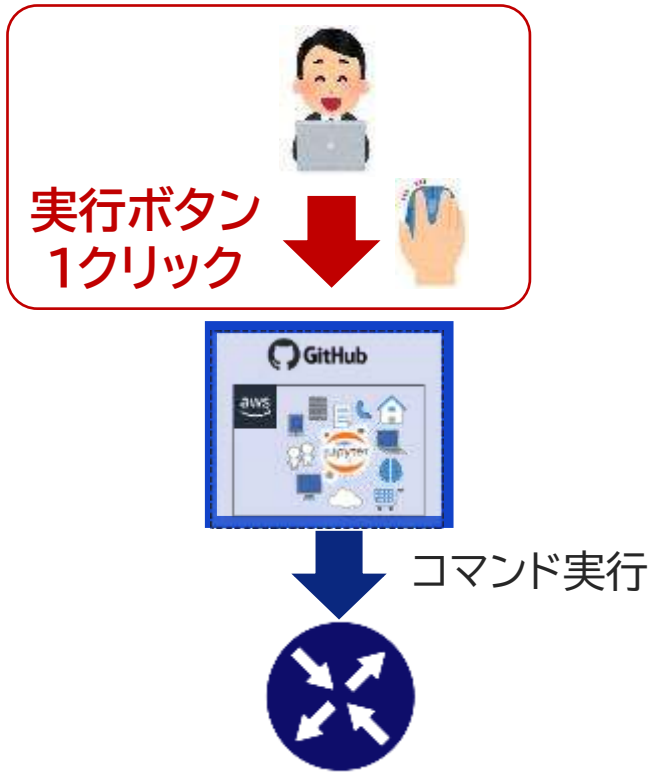
■従来のExcel手順書を、Jupyter手順書に刷新

■Excel手順書のコピーおよびTeraTermへの貼り付け作業をなくし、クリックでコマンド実行可能に

Before



After



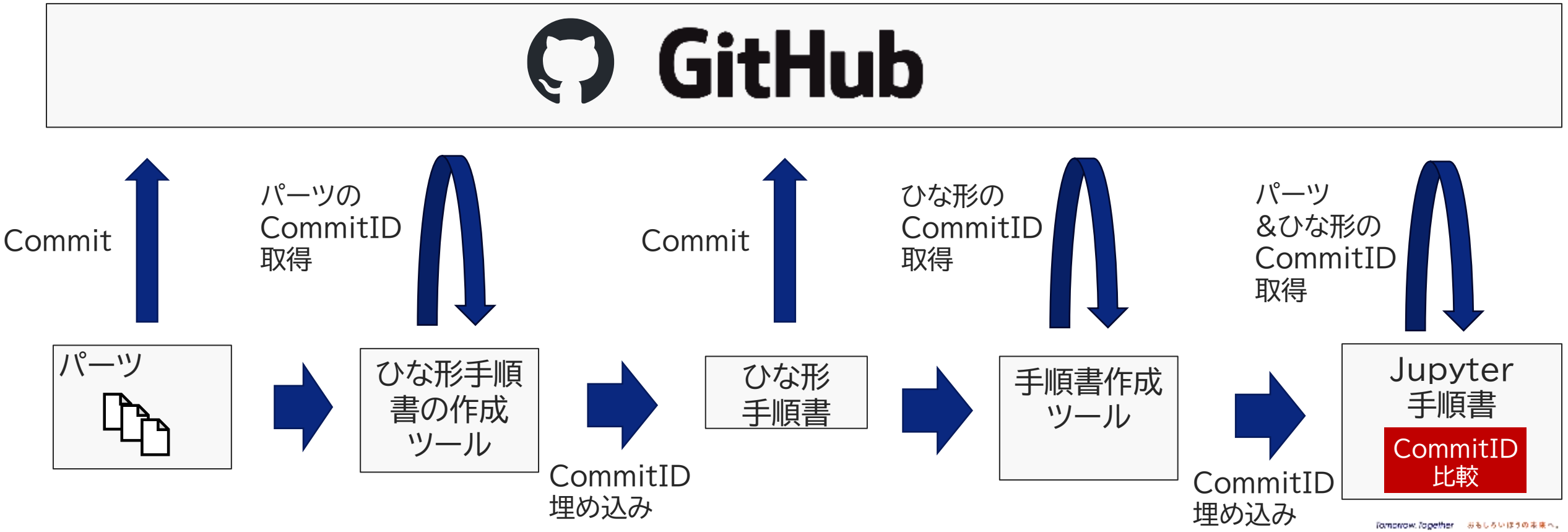
凡例

手動

自動

工夫ポイント1：バージョン確認

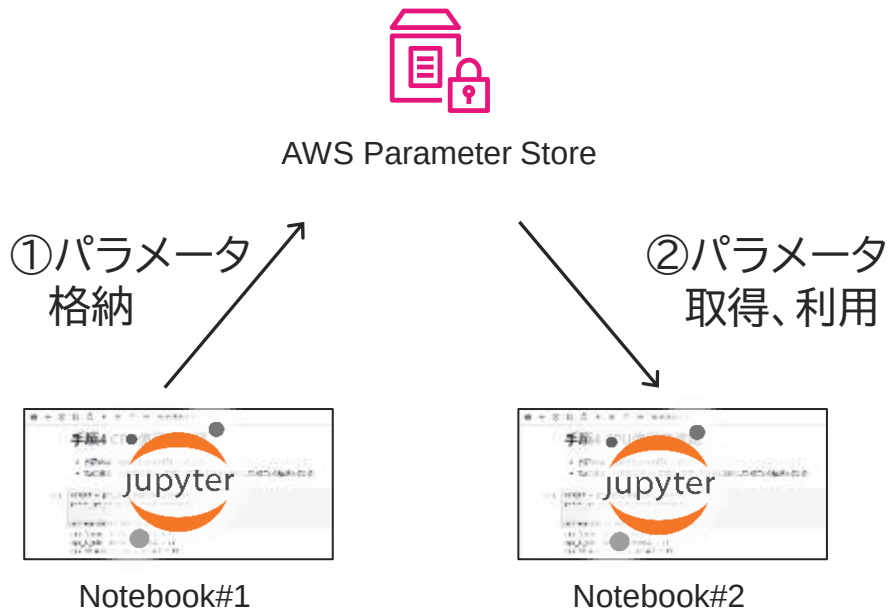
■ 常に最新バージョンのパーツやひな形手順書を利用できるよう、バージョン確認機能を実装



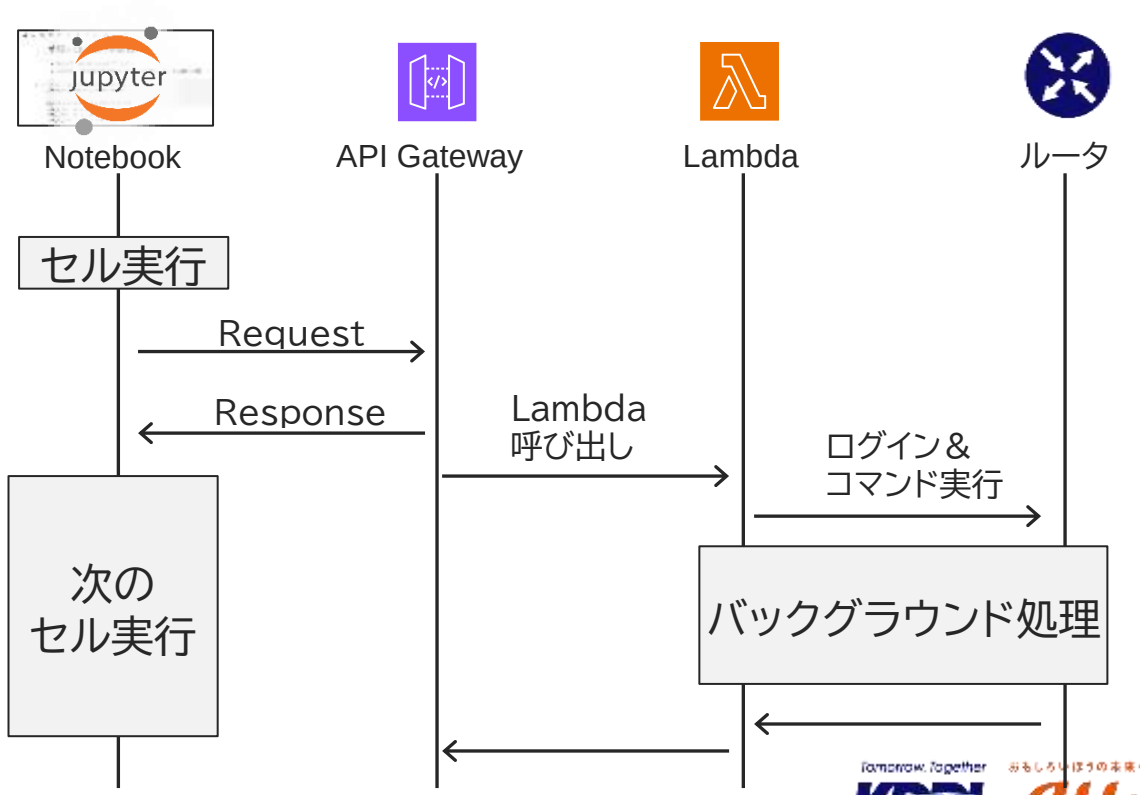
工夫ポイント2：AWSの活用

■ Jupyterでは対応できない要件を解決するため、AWSを活用

Notebook間のパラメータ共有



バックグラウンド処理



5. デモンストレーション

デモンストレーション

作業実施

手順書版数管理の徹底

正常性判断手順の改善

コンフィグ変更結果確認手順の改善

作業準備

自動化部品からひな形手順書を作成

パラメータを収集してひな形手順書に反映

デモンストレーション　作業実施

そもそも、

作業手順書版数管理
コンフィグ全比較
何度も正常性監視 …

大変なのに、なんで必要なのか？

【2. 令和４年度に発生した事故から得られた教訓等】　**主な教訓等①**　12

■令和４年度に発生した重大な事故について、当事者である電気通信事業者から事故の内容等の説明を受け、検証を行い、当該事故等から得られる教訓等を整理。主なものは次のとおり。

(1) 作業手順書の適切な管理

- 作業手順書については、レビュー観点や承認フローを明確にするとともに、取り違えの起こらないよう管理することが重要
- 遵守すべき作業手順書を明確に認識させるための訓練を、作業を行う全ての担当者に対して継続的に行うべき

(2) 誤設定情報の確実な検出

- 設定情報の投入前後での比較・確認対象を、追加・変更した設定値のみだけでなく、全ての設定値とすることが重要

(3) 迅速な異常検知

- システムの可用性について、内部・外部双方からの常時監視を行い、基準を下回った場合にアラートが発せられる仕組みが構築されていることが重要

(4) ネットワーク設計の適切なレビュー

- OSPF（Open Shortest Path First）通信においては、経路計算における処理負荷増大を防止するため、１グループ内における伝送装置の収容数が推奨諸元値を超えない構成とすることが重要

総務省 令和4年度電気通信事故に関する検証報告 別紙2より抜粋
https://www.soumu.go.jp/menu_news/s-news/01kiban05_02000302.html

デモンストレーション　作業実施

手順書版数管理の徹底

※じゅもん:Jupyter手順書の通称

これまで

手順書をシステムに登録
人の目で最新か判断→承認

取り違えないかどうかは
登録者、承認者の確認次第

改善後

手順書自身で改版有無検出
ひな形手順書の更新も検知

```
本じゅもん生成に利用したテンプレートのコミットID: 88c91
本じゅもん生成に利用したテンプレートの最新コミットID: 88
テンプレートは最新です

じゅもん生成時の本じゅもんのハッシュ値： 8f99b6fd7ddece
本じゅもんの再計算したハッシュ値： 8f99b6fd7ddece5956c
じゅもんは変更されていません
```

目の前の手順が最新かを
確認できる

デモンストレーション　作業実施

正常性判断手順の改善

これまで

正常性確認 わかりづらい

```
node:      node0_0_CPU0
------------------------------------------------------------
Physical Memory: 24576M total
 Application Memory : 24239M (19462M available)
 Image: 95M (bootram: 95M)
 Reserved: 224M, IOMem: 0, flashfsys: 0
 Total shared window: 188M
```

```
node:      node0_2_CPU0
------------------------------------------------------------
Physical Memory: 32768M total
 Application Memory : 32431M (28460M available)
 Image: 95M (bootram: 95M)
 Reserved: 224M, IOMem: 0, flashfsys: 0
 Total shared window: 144M
```

改善後

ひと目で判断可能

memory_results

	NodeName	Total	Available	Usage	AvaResult	UsageResult
0	node0_0_CPU0	24239	19462	20.0%	NG	OK
1	node0_2_CPU0	32431	28460	12.0%	OK	OK
2	node0_4_CPU0	24239	19912	18.0%	NG	OK
3	node0_9_CPU0	24239	20572	15.0%	OK	OK
4	node0_RP0_CPU0	16014	12754	20.0%	NG	OK
5	node0_RP1_CPU0	16014	13421	16.0%	NG	OK

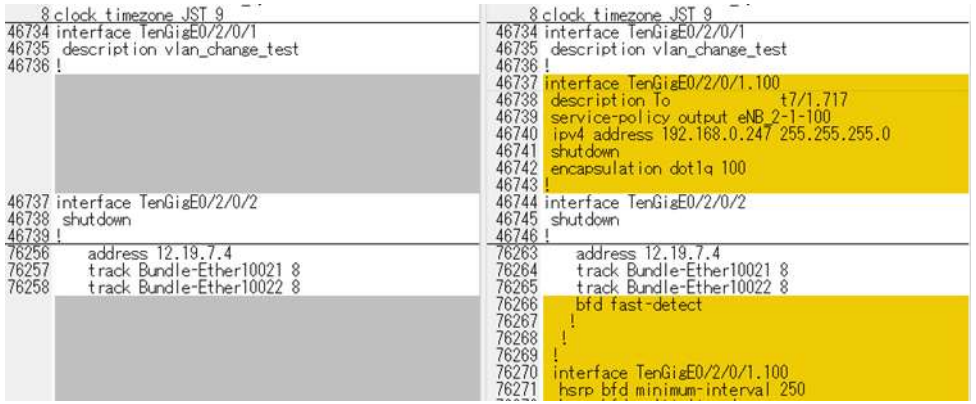
今後は自動で作業続行可否を
判断し、完全自動化に

デモンストレーション　作業実施

コンフィグ変更結果確認手順の改善

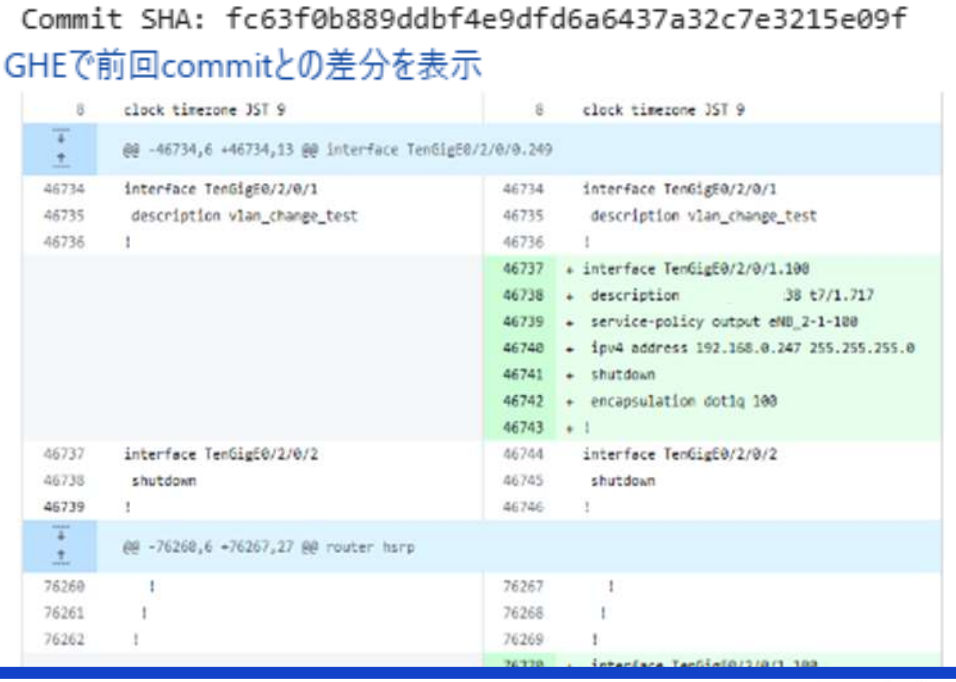
これまで

copy running-config ftp
からの、
GET /running-config.txt
アプリ立ち上げで比較



改善後

コンフィグ取得後に
比較表示リンクを生成



デモンストレーション　作業準備

自動化パーツからひな形手順書を作成

これまで

1作業1ファイル 再利用しにくい

コードに作業実施理由はなく、
ノウハウ継承が難しい

```
#ログイン
#show running-config取得
#IF情報取得
#コンフィグレーションモード移行
#Sub-IF設定変更
…
```

改善後

手順単位でパーツ化

部品を組み合わせて手順書化

```
"""
IF確認(show hsrp [interface])
- 確認内容：対象SubIFのPriorityが、{{ show_hsrpコマンド結
- 確認理由：HSRPグループのPriorityが設定した値であり、状態が
"""
for
```

手順11 IF確認(show hsrp [interface])

- 確認内容：対象SubIFが表示されること。Priority=100、state=Init
- 確認理由：HSRPグループのPriorityが設定した値であり、状態が想定のステータスで

```
for interface in show_hsrp_interfaces:
    output = show_hsrp_interface(net_connect, interface)
    print(output)
    print("-----------")
```

デモンストレーション　作業準備

パラメータを収集してひな形手順書に反映

これまで

設計データやコンフィグを基に
パラメータをExcelに入力

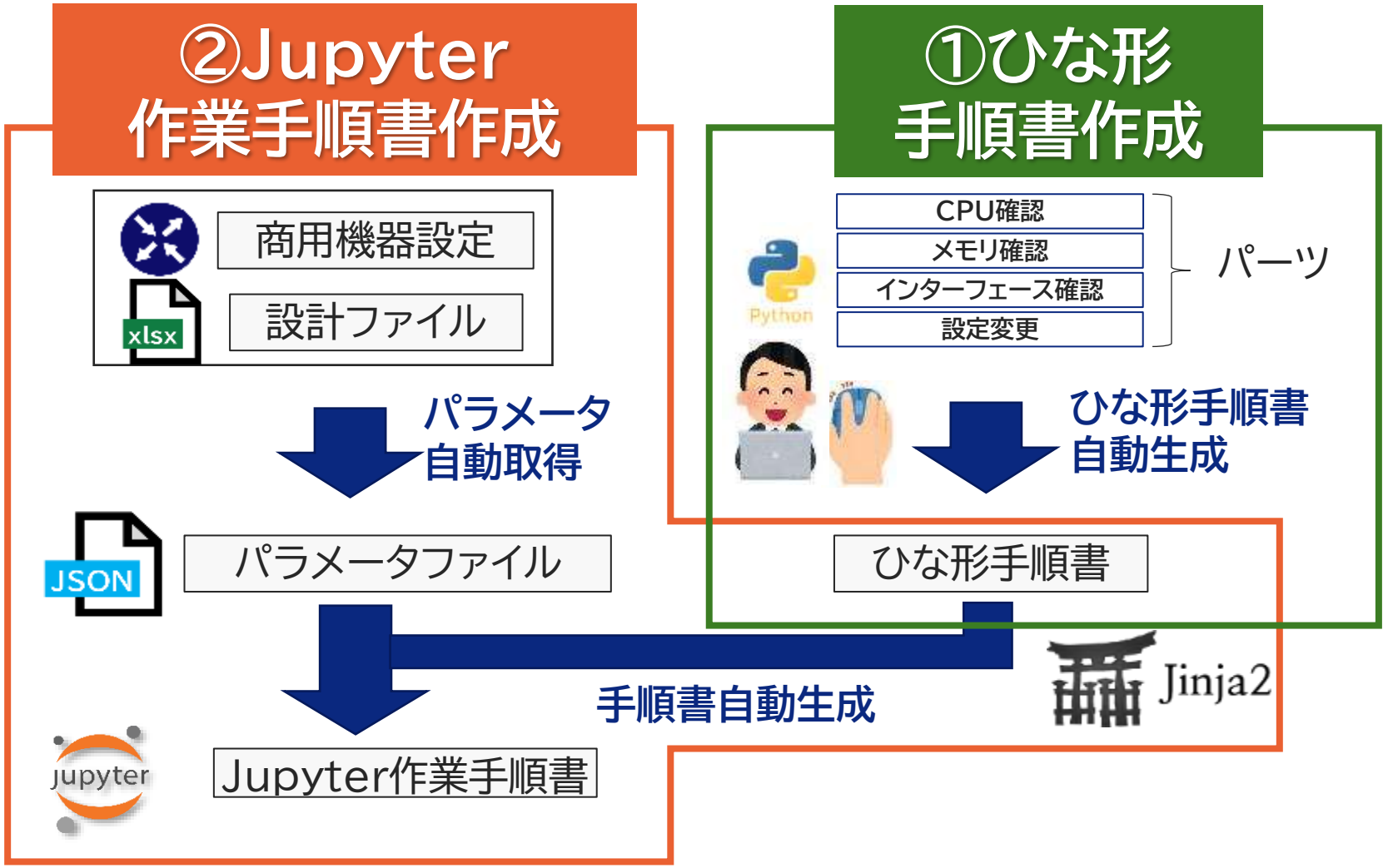
xlsx VLAN番号はExcelから…

集約ルート情報は
実機にログインして…

改善後

設計データとコンフィグから
必要なデータ抽出し
ひな形手順書に代入

デモンストレーション　作業準備



6. おわりに

【再掲】自動化に立ちはだかる壁とその解決策

自動化に立ちはだかる壁

自動化が

- **始まらない**
  - 多忙
  - スキル不足
- **続かない**
  - 変化への対応遅延
    - 綿密な要件定義/費用対効果確認
  - 属人化・陳腐化
    - 仙人がいなくなると使われない
- **広まらない**
  - 偏見
    - 作業自動化への品質不安

1.プロセスによる解決策

- トップダウン
- 内製＆アジャイル開発
- 利用者の拡大
- 実績の積み重ね

2.プロダクトによる解決策

- 最小限の労力
- 暗黙知を形式知へ変換して
  それを活用する仕組み
- 手順を変えずに自動化

商用作業を終えて

数字で見る成果

	これまで	導入後
作業手順書作成	約4営業日	約0.5営業日
作業実施	1夜間作業 約6時間	約3時間

社内の他部門からも「使ってみたい！」の声を多くいただく

プロジェクト参画者の感想

ディベロッパー（Dev） プロダクトを設計/実装する人

- 社内のユーザのため、NWサービス利用者のためになる
  アプリをリリースできて嬉しい
- **もっと高度な技術**
  （AI活用・デジタルツイン検証環境構築）に触れたい
- 設計ミスを通じて今まで知らなかった
  ルータやNWのことが少しわかるようになった

NW装置に関わる業務プロセス知り、通信キャリアである自社内の業務を改善できるソフトウェアエンジニアとして、一歩を踏み出せた

プロジェクト参画者の感想

プロダクトオーナー（PO）

プロダクトのビジョンや要件を定義して
ステークホルダーと開発者の間を取り持つ

- 自動化の壁を超えるために、何が必要か分からなかった。
  ユーザ＆Devと対話を重ね、方向性が見えた
- 開発チームとユーザの距離が近く、要望をすぐに反映できる**内製開発の醍醐味を感じられた**
- **ユーザ自身がプロセス改善に乗り気**になり、さらにプロダクト利用の機運が高まったのが嬉しい

別の部門にも同じ進め方でプロセス改善を行い、会社全体の業務効率化を促進できると確信

プロジェクト参画者の感想

ユーザ　手順書を作る人/作業する人

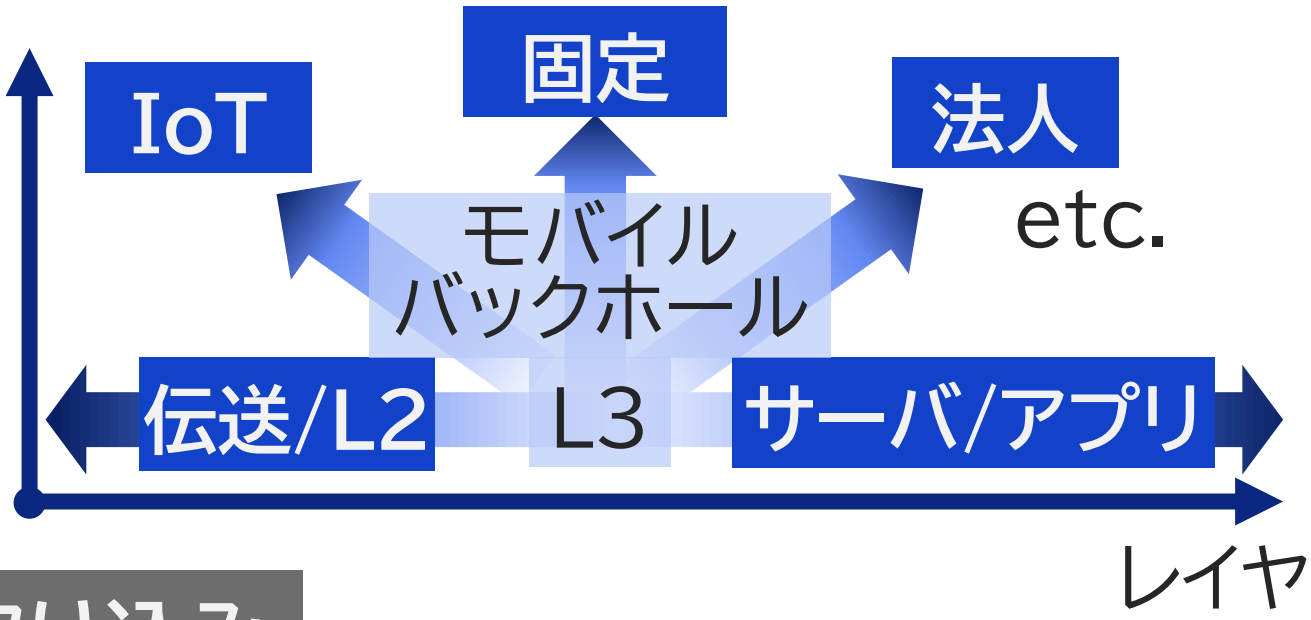
- 夜間作業時の**判断ミスを減らせそう**
  体力を温存できるようになった(切実)
- **何百とある今後の作業の完了見通しを立てられた**
  自律型NWへ移行するときも、きっと役立つはず
- **自動化しやすい作業手順に変えたくなった**
- **もっと自動化できる** 希望を持てた
  もっと自動化してやる という競争心も芽生えた

自動化の壁を越えはじめた！

今後の展望

適用領域の拡大

楽に安全でスピーディーに
完了できる作業手順書を
様々な領域で量産したい



野良 既存有用スクリプトの取り込み

各々が自分たちの
ために書いたスクリプト

みんなで有効活用
（社内/社外問わず）

本日お話ししたこと

自動化が		自動化を
始まらない	→	始めた!!
続かない	→	続けられそう!!
広まらない	→	広めるぞ!!

議論ポイント

皆さんの現場にも自動化の壁はありますか？ありましたか？

どの様な壁ですか？
今回の発表を聞いて、乗り越えてみようと思えましたか？

皆さんの現場で利用している自動化ツールは何ですか？

私たちはJupyterを活用しましたが、Ansibleの方が良い、
TeraTermマクロで十分なのか、どんなツールが人気なのか、など･･･

ネットワークエンジニアはソフトウェアスキルを持つべき？

どのぐらいのレベルが必要だと思いますか？
スキル習得に壁を感じますか？

Tomorrow, Together

KDDI

「つなぐチカラ」を進化させ、
誰もが思いを実現できる社会をつくる。

KDDI VISION 2030