JANOG56 Meeting in Matsue

ネットワーク運用自動化における コンフィグ設定差分への対策、その後の苦労話

2025年07月30日

NRIセキュアテクノロジーズ株式会社
マネージドセキュリティサービス開発本部

甘利 丈慈, Oyunbileg Chingun

自己紹介

■名前: 甘利 丈慈 (Amari Jorge)

■所属: NRIセキュアテクノロジーズ

■職歴：2023年4月入社
- ネットワーク運用/ネットワーク運用自動化
- DNS運用/DNS運用自動化

■出身：東京都港区

■趣味：スカッシュ, 将棋, Run

JANOG 初参加！🔰

■名前: Oyunbileg Chingun

■所属: NRIセキュアテクノロジーズ

■職歴：2017年4月キャリア入社
- GitOps, NetOps, IaC
- 共用基盤, レポートシステム設計開発

■出身：モンゴル (東京都中央区在住)

■趣味：読書,バスケ,水泳,映画,旅

JANOG 3回目

目次

1. 我々がこれまでJANOGで発表したネットワーク運用自動化について
2. ネットワーク運用自動化の課題：コンフィグ設定差分
3. コンフィグ同期機能のリリースと手ごたえ
4. ヒアリングで見えた課題とその対応についての議論

設計開発者

作業担当者

1. 我々がこれまでJANOGで発表したネットワーク運用自動化について

手動設定変更の自動化

▼

確認コマンド実行の自動化

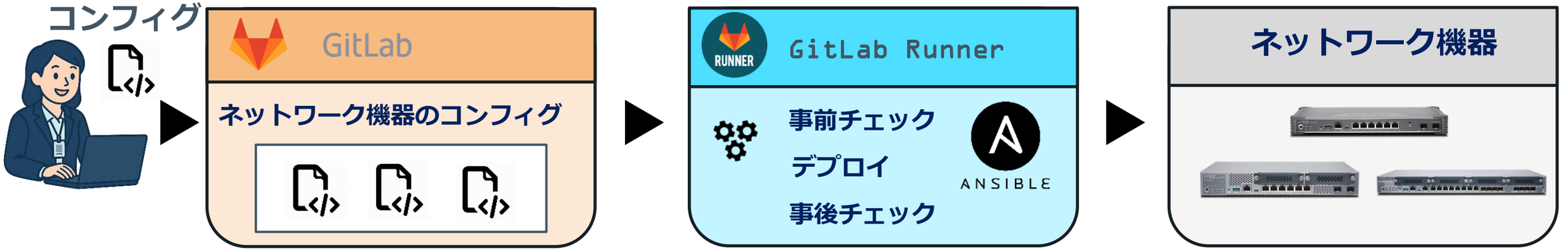
▼

今回のコンフィグ最新化の自動化へ

1-1. 手動設定変更から設定変更の自動化を実現

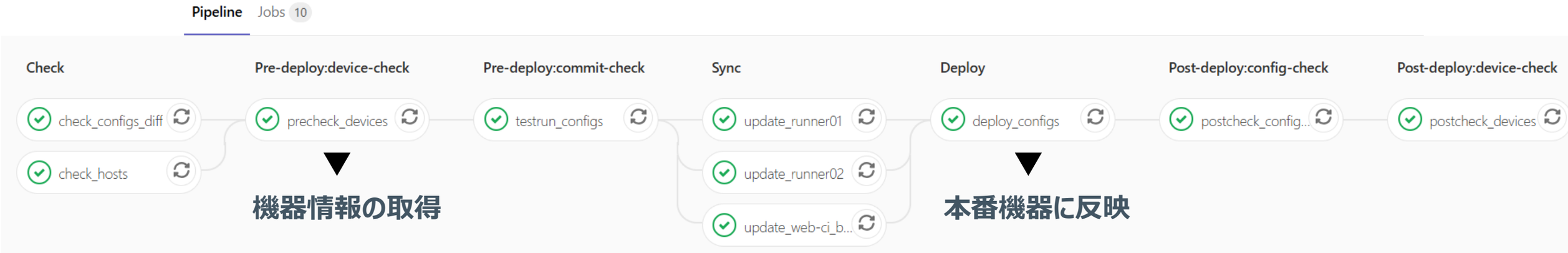
SNS投稿禁止
NO POST TO SNS

JANOG49 登壇　手動での設定変更の自動化



①Git masterへのマージ　②CI/CDパイプラインの動作　③実機への設定反映

★設定変更マージリクエストのマージをトリガに動作するCI/CDパイプライン



1-2. 実機への確認コマンド実行を自動化した補助機能リリース

JANOG52 登壇　GUI上から実機の確認コマンドを実行

Run for

master

Existing branch name or tag

Variables

Variable	WEB_CI_FUNCTION	check
Variable	WEB_CI_TARGET	netopssrx02
File	WEB_CI_COMMANDS	prefix 172.23.93.84 show security flow session destination-
Variable	Input variable key	Input variable value

Specify variable values to be used in this run. The values specified in CI/CD settings will b

Run Pipeline

Pipeline　Jobs 3

Check　Web-ci:check　Web-ci:execute

check_hosts　check_variables　execute_comma...

```
44  実行されるコマンドは下記です.
45  show security flow session destination-prefix 172.23.93.81
46  show security flow session destination-prefix 172.23.93.82
47  show security flow session destination-prefix 172.23.93.83
48  show security flow session destination-prefix 172.23.93.84
49  show security flow session destination-prefix 172.19.249.195
50  show security flow session destination-prefix 172.19.249.196
```

1-3. 時系列でみる我々がJANOGで発表したネットワーク運用自動化

新しい補助機能「コンフィグ同期機能」を開発・リリースしました！

2021年　2022年　2023年　2024年　2025年

★ JANOG49 登壇

★ JANOG52 登壇

手動設定変更 → 設定変更を自動で運用

周辺コマンド実行機能

補助機能 → コンフィグ同期機能

今回の対象

2. ネットワーク運用自動化の課題
コンフィグ設定差分

ネットワーク運用自動化の現状と課題

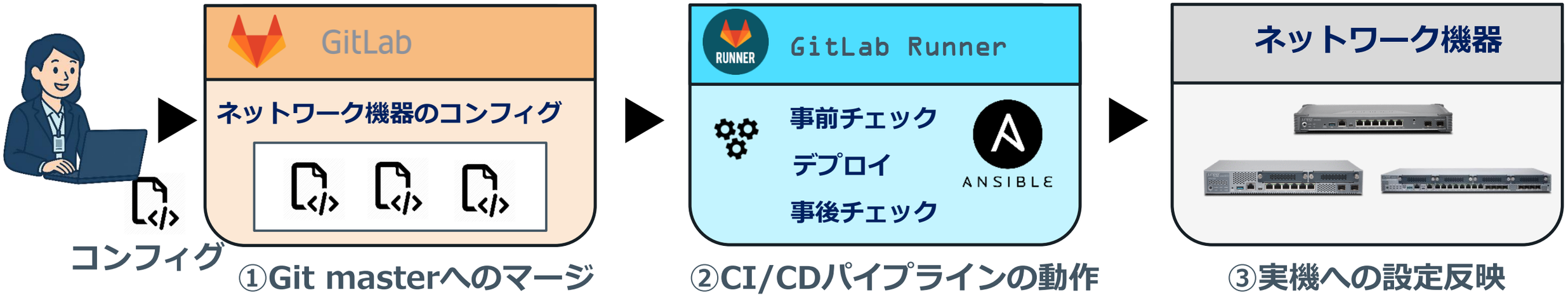
▼

コンフィグ差分に対する問題意識

2-1. ネットワーク運用自動化の現状と課題（1 / 2）

ネットワーク運用自動化の通常運用時

Gitのマージをトリガーに CI/CDパイプラインで実機に設定変更を反映しているため、Git管理のコンフィグは実機と同期しており、常に最新の状態



2-2. ネットワーク運用自動化の現状と課題（2/2）

SNS投稿禁止
NO POST TO SNS

イレギュラーな対応

Gitを介さず実機に直接設定変更が行われた場合、Gitのコンフィグが更新されず、実機とGit管理のコンフィグに差分が生じてしまう

2-3. コンフィグ差分に対する問題意識

コンフィグが非同期状態になるケースは、ネットワーク運用自動化を開始した当初から問題視していた課題

過去のJANOGの質疑応答でもどうするべきか議論になりました

■ 我々の前回・前々回のJANOG登壇テーマ

ネットワーク運用自動化(NetOps)の実運用: 苦労/解決/妥協した話

ネットワーク運用自動化(NetOps)の実運用: その後の改善・拡大、そして残課題について

当時いただいた意見

→ 差分は避けられないもので、気にせずに新規設定で上書きするのが吉

▼

セキュリティ運用では、イレギュラー対応で投入された緊急遮断など重要な設定変更が消失してしまうため、気にせず上書きすることは許容されない

3. コンフィグ同期機能のリリースとリリース後の手ごたえ

コンフィグ同期機能のリリース

▼

コンフィグ同期機能の紹介

▼

コンフィグ差分認知から解消までのフロー

▼

新機能リリース後の手ごたえ

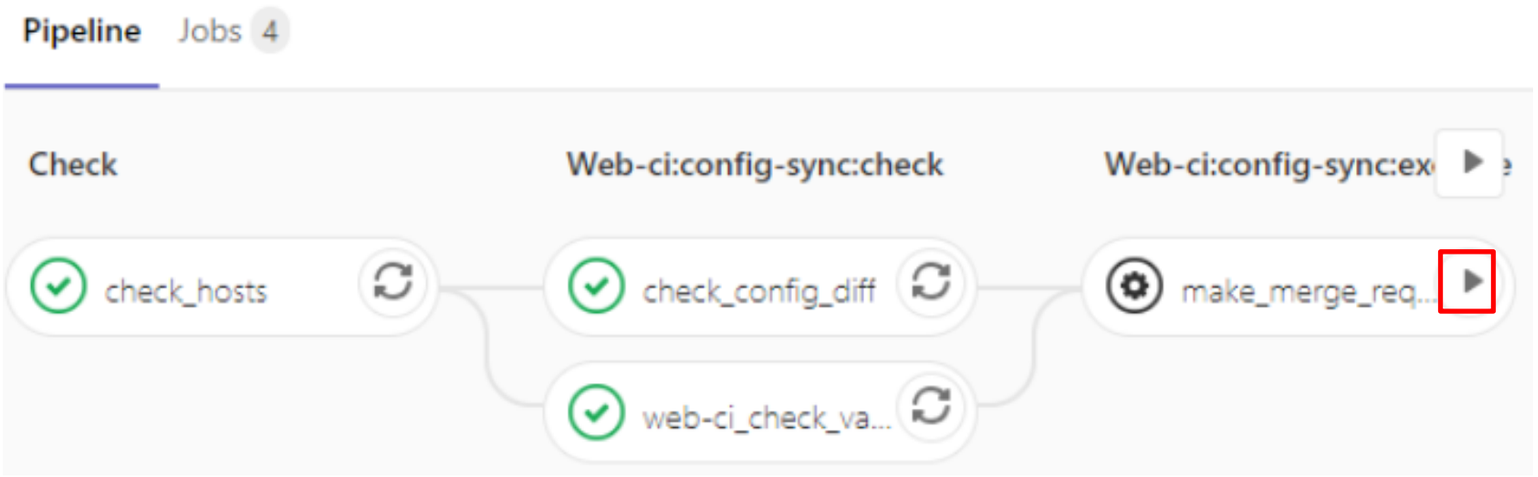
▼

効果測定とヒアリング

3-1. 対策：補助機能としてコンフィグ同期機能をリリース

設定変更作業者はGUI画面から簡単な操作をするだけ
Git管理のコンフィグを実機に合わせて更新するマージリクエストを作成

3-2. コンフィグ同期機能の紹介：ジョブ構成

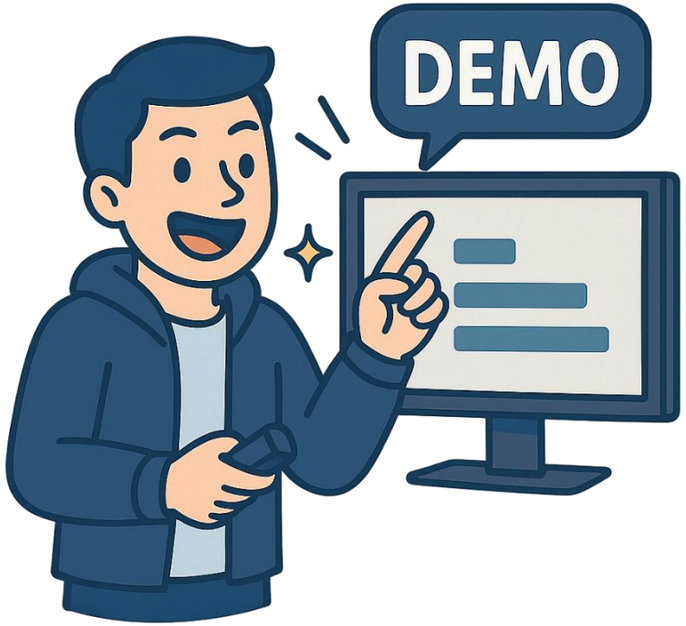


executeステージは
コンフィグ差分が生じている
場合のみ手動実行可能

名前	トリガー	役割
check_hosts	userの手動	IPアドレス・ホスト名の重複を確認
check_config_diff	自動	実機のコンフィグとGitlab管理のコンフィグに差分が生じているかどうかを確認
check_variables	自動	GUI画面から入力した各変数に不備が無いことを確認
make_merge_request	userの手動	Gitlab管理のコンフィグを実機に合わせて同期するマージリクエストを作成

3-3. コンフィグ同期機能の紹介：デモ

開発環境でのコンフィグ同期機能のデモ



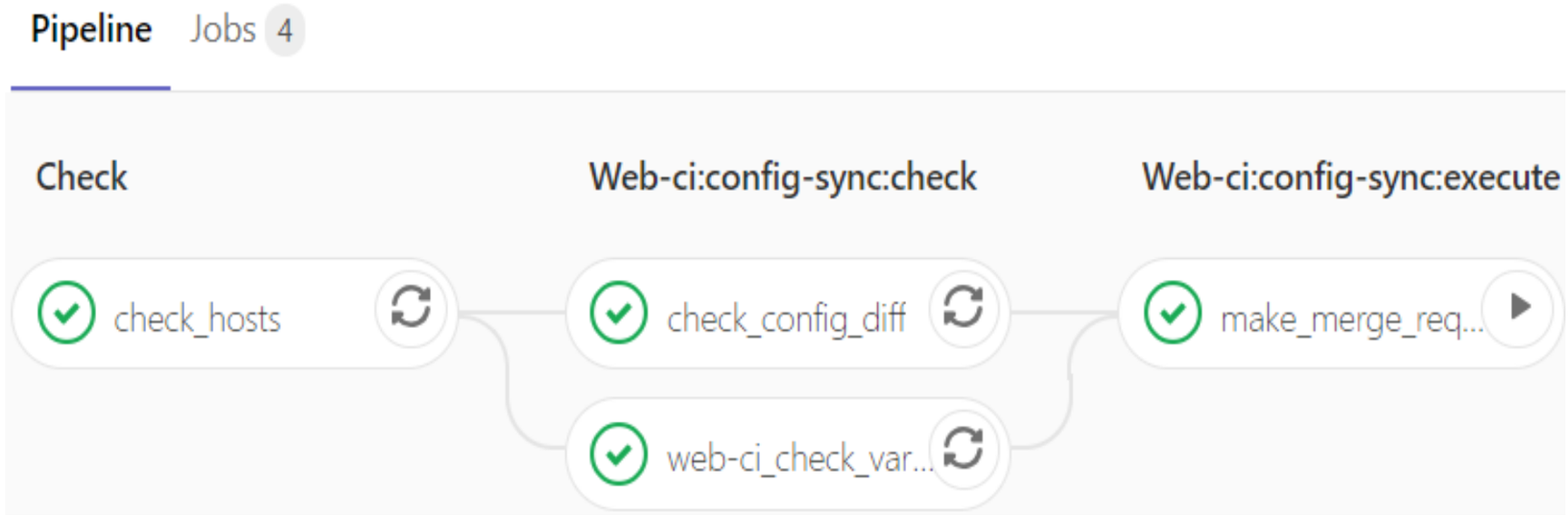
デモの流れ：

1. 実機を直接設定変更（事前に実施済み）
2. 設定変更（自動）
3. コンフィグ差分の検出
4. コンフィグ同期機能の実施
5. 設定変更（自動）のリトライ

3-4. コンフィグ差分認知から解消までのフロー

実機ログイン不要 ＋ 作業時間&作業の手間を縮小

3-5. リリース後の手ごたえ



◎ GUI上で完結
→ 実機のログインは不要
◎ CI/CD自動パイプライン

作業担当者からのPositiveな声

■実機ログインが不要になり、操作ミスなどインシデントのリスクが減った
■10~20分のつきっきりの作業が、5分未満の自動処理となり、楽になった！

3-6. 効果測定とヒアリング

■ 効果測定：2024年9月~2025年4月 → 7割使われている！！！

コンフィグ差分が起きた件数: 99件

66件	33件
コンフィグ同期機能	従来の追いつき対応

■ 残り3割に使われない原因 (担当者にヒアリングを実施)

①承認プロセスが効率性の弊害となるケースがある

②周知不足
Teamsなどでの全体周知+機能説明会は実施したが、充分に浸透せず

4. ヒアリングで見えた課題とその対応についての議論

コンフィグ同期機能における承認プロセスの考え方の違い

▼

承認プロセスが弊害で
コンフィグ同期機能が使われない

▼

まとめ、議論 (悩みポイント)

4-1. コンフィグ同期機能における承認プロセスの考え方の違い

コンフィグ同期機能の使用フロー

①コンフィグ差分認知 ▶ ②コンフィグ差分解消機能の実行 ▶ ③承認プロセス ▶ ④マージリクエストのマージ ▶ コンフィグ差分解消対応完了

我々開発者の設計思想
Git上のコンフィグを実機に合わせるということは実機へ変更ではない（本番作業ではない）ため、クロスチェックと承認は不要

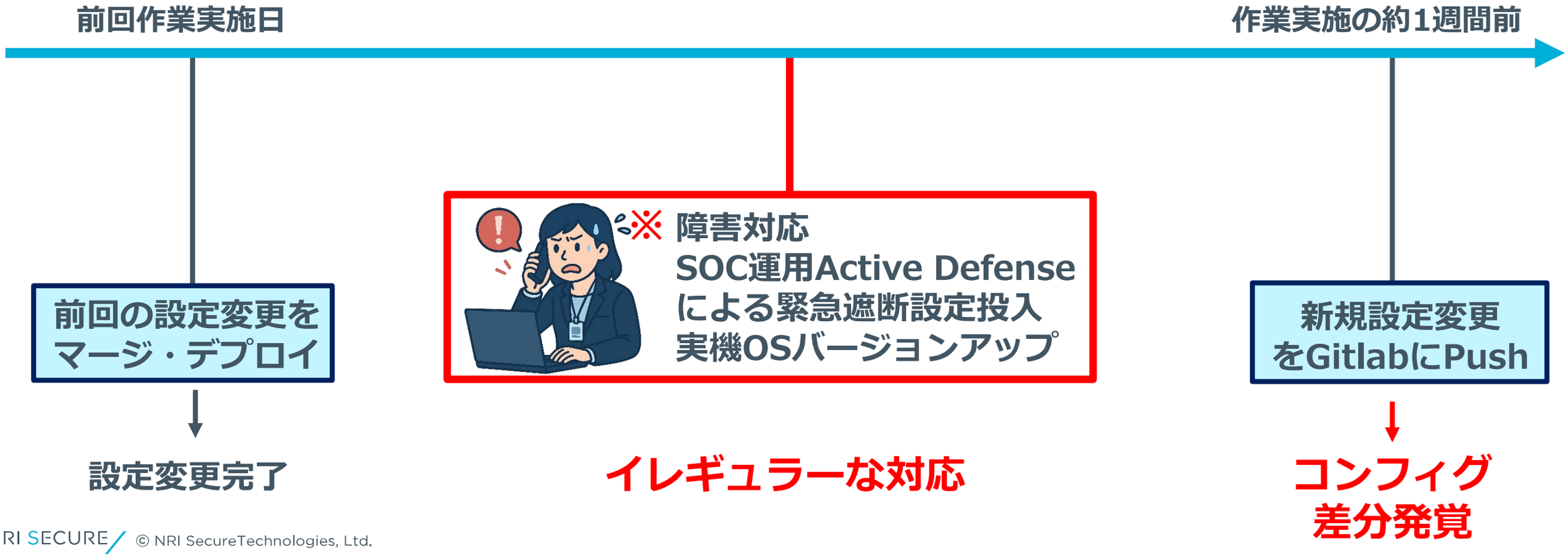
ユーザ側@現場の観点
手間ではあるが「コンフィグ同期機能」リリース以前の慣習に則って、新しいマージリクエストには、クロスチェックと承認が必要

4-2. 承認プロセスが弊害でコンフィグ同期機能が使われない

■注目ケース：緊急度の高い対応で生じるコンフィグ差分

→ 直後にコンフィグ差分解消対応がされず、新規設定変更の際に気が付く

→ 作業担当者は、設定変更とコンフィグ差分解消対応の両方の対応が必要



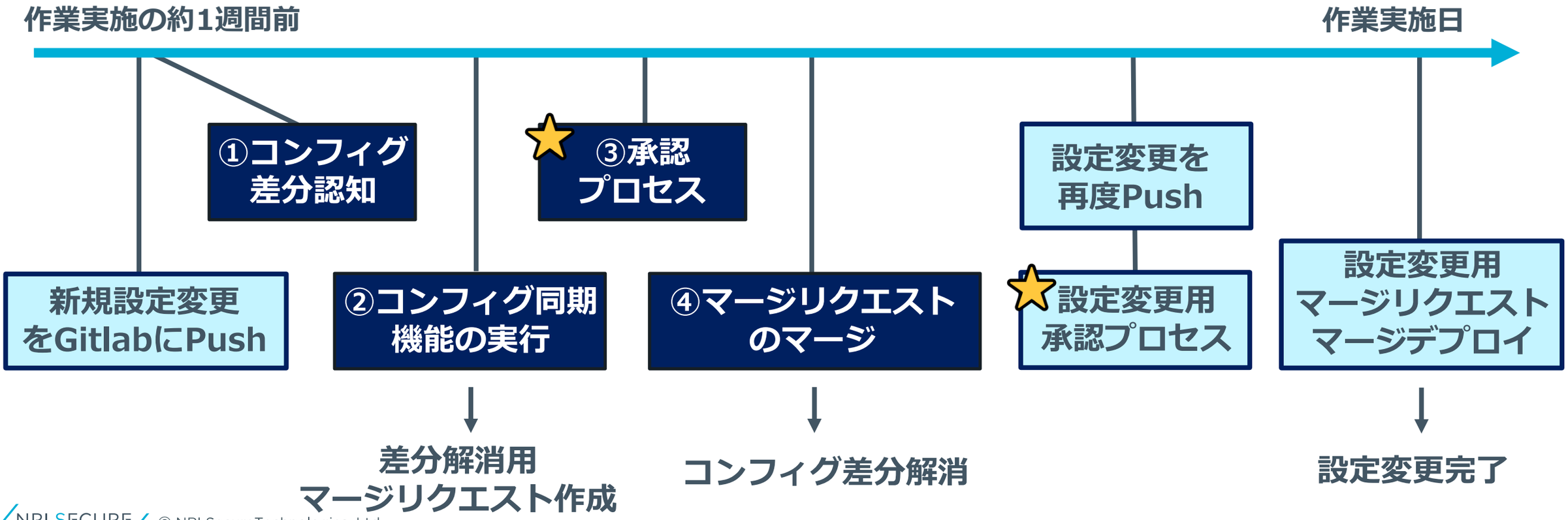
4-3. 承認プロセスが弊害でコンフィグ同期機能が使われない

SNS投稿禁止
NO POST TO SNS

■ 設定変更完了までに二回の承認プロセスが発生

→ 承認依頼の待ち時間が長い (3日~5日)

→ 承認者の手間が増大する

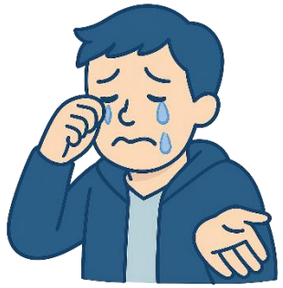


4-4. 承認プロセスが弊害でコンフィグ同期機能が使われない

■従来の手動方法でコンフィグを更新されてしまう…

→ 実機にログイン → 人的ミスのリスクが増加

→ 手作業で非効率



作業実施の約1週間前

作業実施日

①コンフィグ差分認知

新規設定変更をGitlabにPush

②実機にログインしてコンフィグを取得

③設定変更用投入コンフィグを手動修正

コンフィグ差分を修正しつつ、設定変更内容を反映したマージリクエストを作成

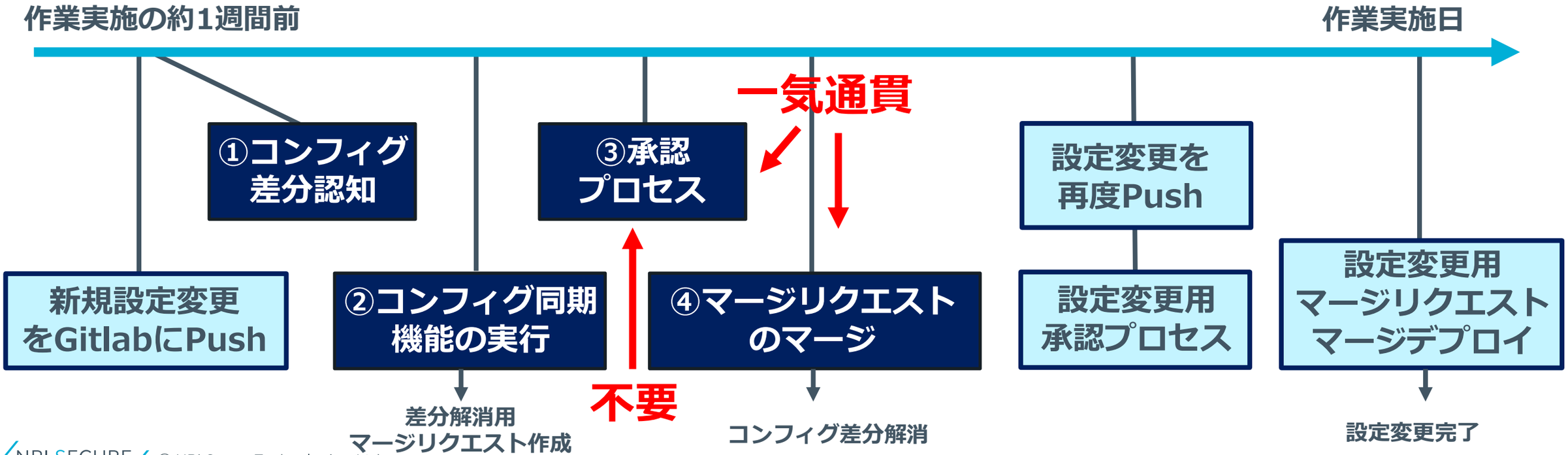
設定変更用承認プロセス

設定変更用マージリクエストマージデプロイ

設定変更完了
コンフィグ差分も解消

4-5. 議論したいポイント (悩みポイント)

- 本番機器の変更じゃないので、③の変更管理やレビューは必要ない
- 新しいマージリクエスト＝変更管理、承認が必要
- 変更管理やレビューを省略するルールにしていきます
- それなら③④もなくして一気通貫にできないか？
- 非常に気持ちはわかるが、GitOpsの基本として違和感ある…



4-6. まとめ・議論したいポイント

- ネットワーク運用自動化におけるコンフィグ差分を解消する新機能リリース
- 本機能は実機へのログインが不要であり、パイプライン上で自動で実行されるため、安全性・効率性の観点での寄与が確認されている
- 一方で、我々の設計思想と現場の実情に相違あり、作業担当者の負担を軽減しきれていないケースもあることが判明している

コンフィグ同期のマージリクエストにクロスチェック/承認必要と思いますか
承認不要とは言え、マージまで一気通貫させることはどう思いますか

現場での作業担当者の利用ハードルを下げるには他に何ができると思いますか

機能開発者の想定と現場の実情のギャップをどのように埋めていますか
皆さんに類似事例/失敗談/横展開例などあればアドバイスください

Appendix

ネットワーク運用自動化(NetOpsについて)
過去のJANOG登壇資料から抜粋

詳細は以下をご覧ください

JANOG49 登壇 https://www.janog.gr.jp/meeting/janog49/netops/

JANOG52 登壇 https://www.janog.gr.jp/meeting/janog52/netops/