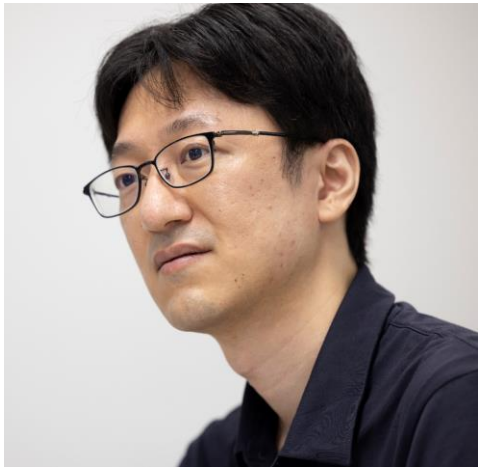
システム開発とネットワーク運用の両立に向けた取り組みと課題

JANOG51 Day3

久保仁史

自己紹介

久保 仁史 (Kubo Hitoshi)



所属：株式会社アット東京
　　ソリューション本部ネットワークサービス部

業務：データセンター内ネットワークサービス(L2/L3)
　　の開発・運用・保守

長男(10ヵ月)

話すこと

作成したツール
取り組み

話すこと

作成したツール

①通知メール作成ツール

②Zabbix通知抑止ツール(資料のみ)

取り組み

メンテナンス/障害についてお客様にメール連絡する

今まで

	実施者	作業内容
①	人	回線管理システムから影響箇所をキーワードに検索し、影響回線リストを作成する
②	人	顧客管理システムから連絡先リストをダウンロードし、影響回線だけ連絡先リストを作成する
③	人	①、②のリストをもとにメールを作成する
④	人	記載内容をチェックしたのち、メール送信する

課題

通知漏れのリスク

・回線検索手順が人によって様々

・コピペ誤り

時間がかかる

・１回の通知で複数回、回線検索が必要

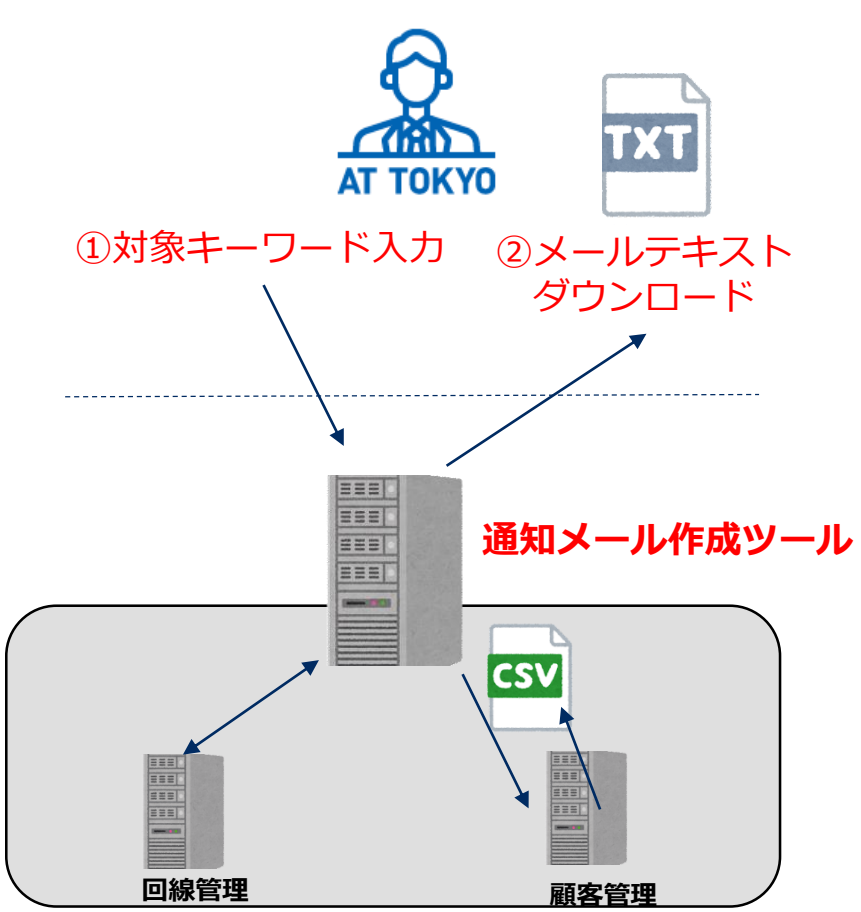
・回線と連絡先の突合

・メールの作成

・チェック

20min/1件×月10件 ⇒ 200min

通知メール作成ツール



	実施者	作業内容
①	人	影響箇所のキーワードを入力する
	ツール	キーワードから影響回線リストと連絡先リストを抽出し、メールテキストを作成する
②	人	メールテキストをダウンロードする
③	人	メールテキストに影響時間と内容を記載する
④	人	記載内容をチェックしたのち、メール送信する

通知メール作成ツール - 効果

通知漏れのリスク
- 回線検索手順が人によって様々
- コピペ誤り

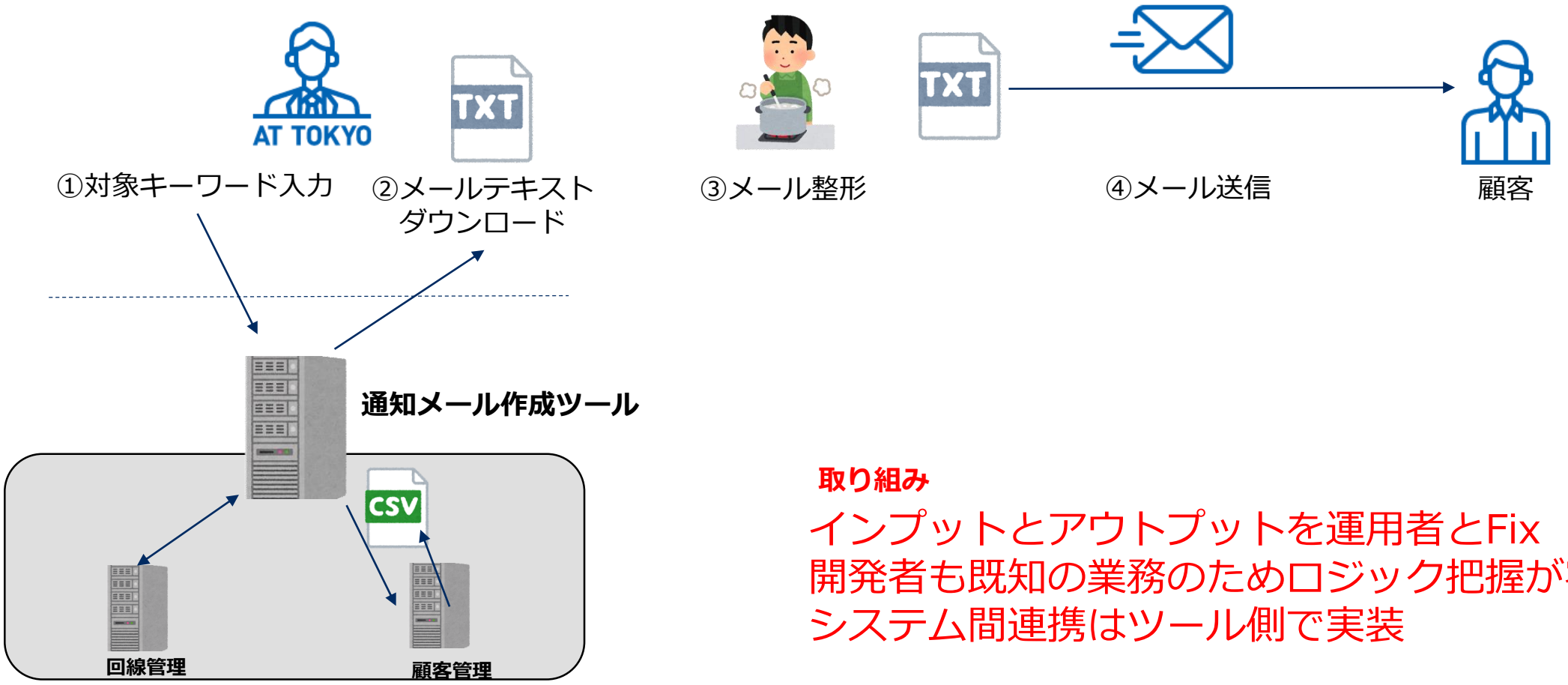
時間がかかる
- １回の通知で複数回、回線検索が必要
- 回線と連絡先の突合
- メールの作成
- チェック

入力は特定キーワード１回

出力は回線リストと宛先が
記入されたメールテキスト

~~20min/1件×月10件 ⇒ 200min~~
5min/1件　×月10件 ⇒ 50min

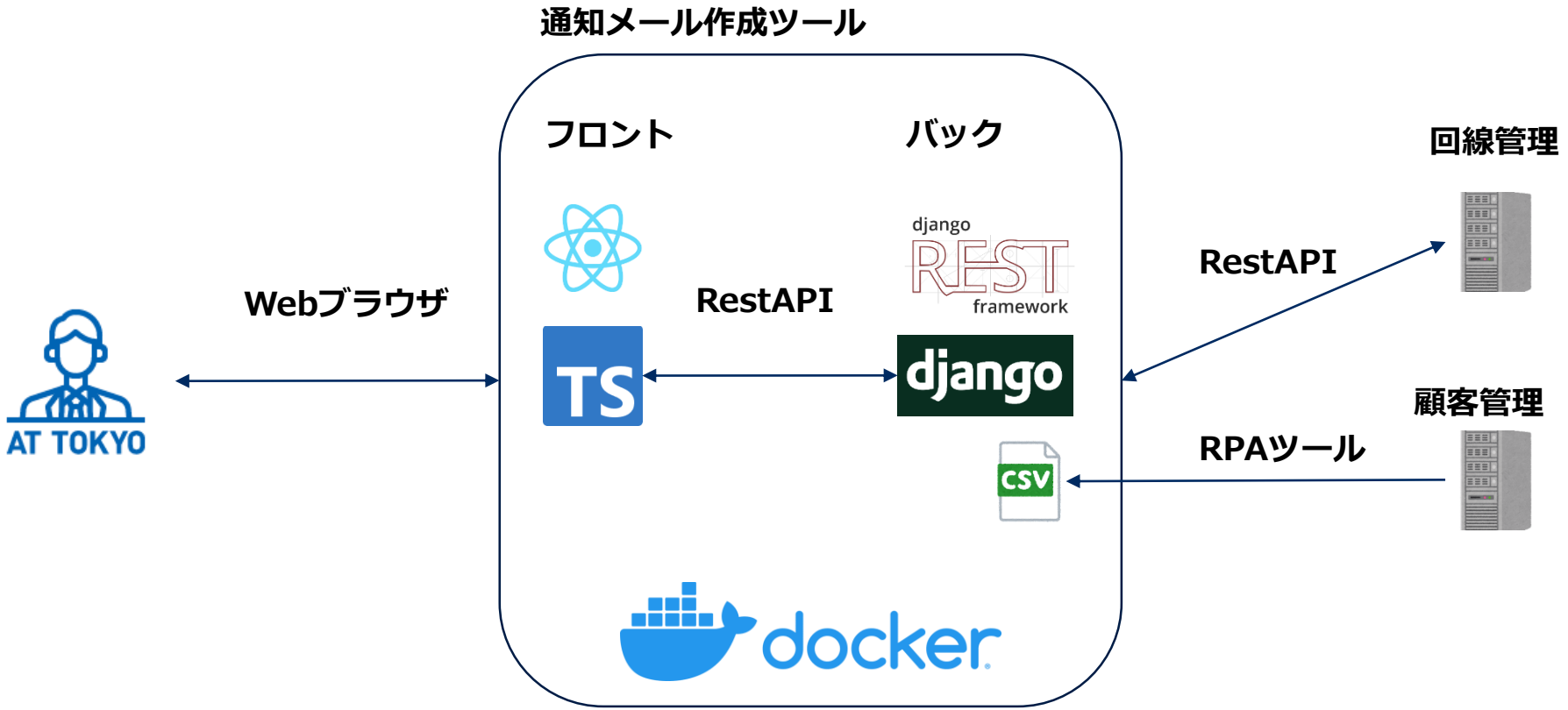
通知メール作成ツール - 取り組み



取り組み

インプットとアウトプットを運用者とFix
開発者も既知の業務のためロジック把握が容易
システム間連携はツール側で実装

通知メール作成ツール - システム図



話すこと

作成したツール

①通知メール作成ツール

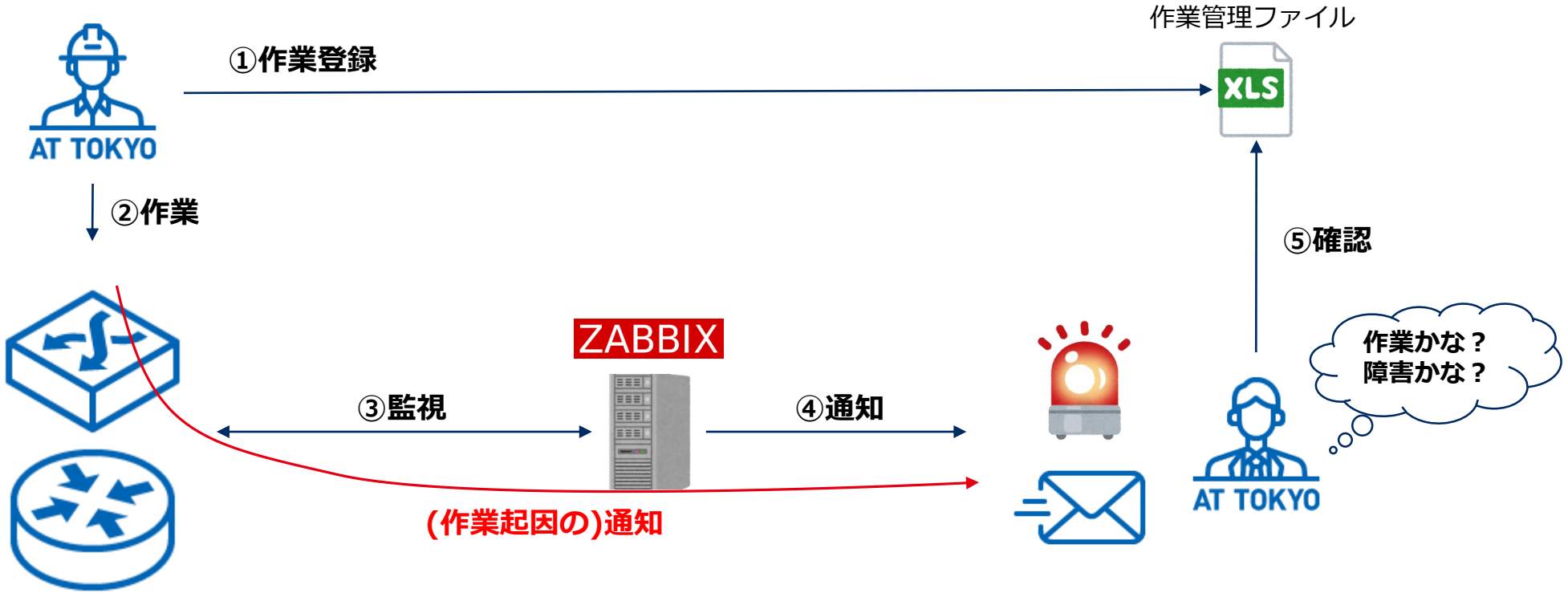
②Zabbix通知抑止ツール(資料のみ)

取り組み

計画作業で発生する通知を止めたい

資料のみ

今まで



※Zabbixに抑止機能は具備しているが、作業単位で管理するには少々煩雑

課題

資料のみ

通知が作業起因か障害起因か判別を誤るリスク

時間がかかる

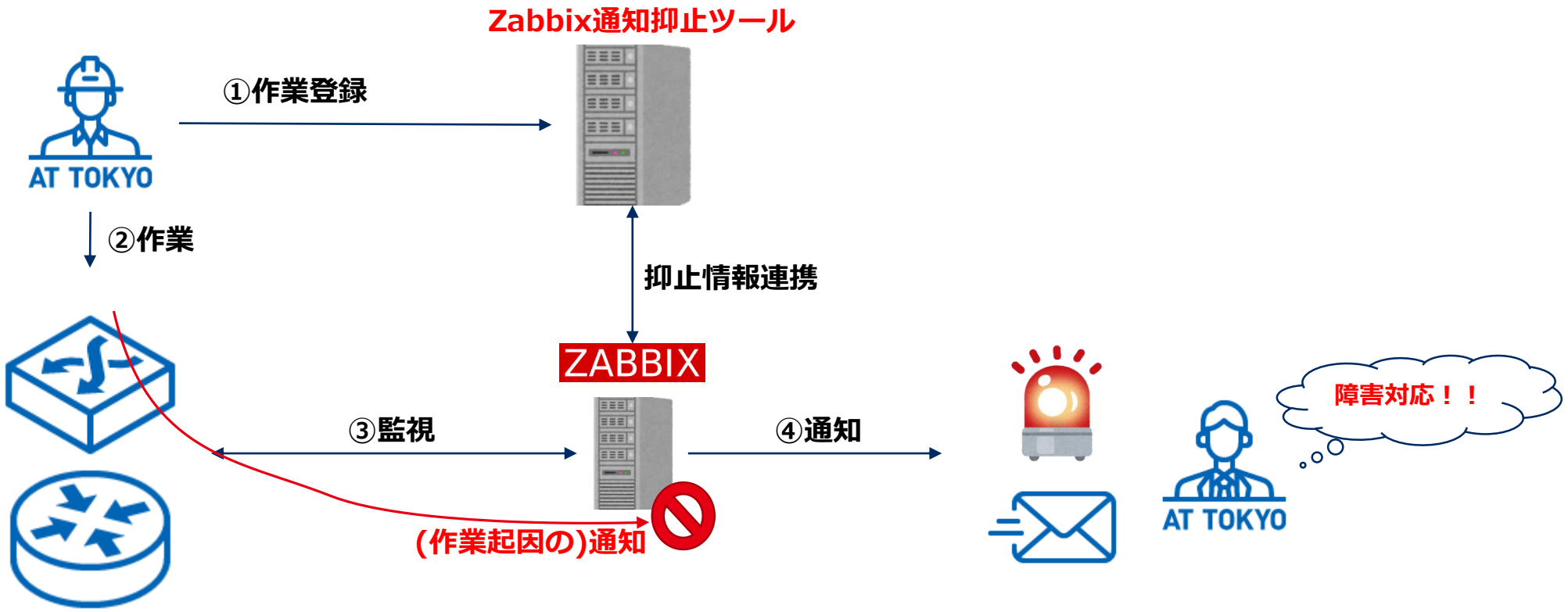
(我々が運用する上での)Zabbixの抑止機能の課題

・作業単位で管理することが困難で1作業につき複数の登録が必要
・抑止の開始終了時間は1登録単位のため変更が大変
・バリデーションがない（インタフェースやIPアドレス）
・作業の進捗(開始前、作業中、終了)のステータスが無い

通知数：月1100件

Zabbix通知抑止ツール

資料のみ



Zabbix通知抑止ツール - 効果

資料のみ

通知が作業起因か障害起因か判別を誤るリスク

時間がかかる

(我々が運用する上での)Zabbixの抑止機能の課題

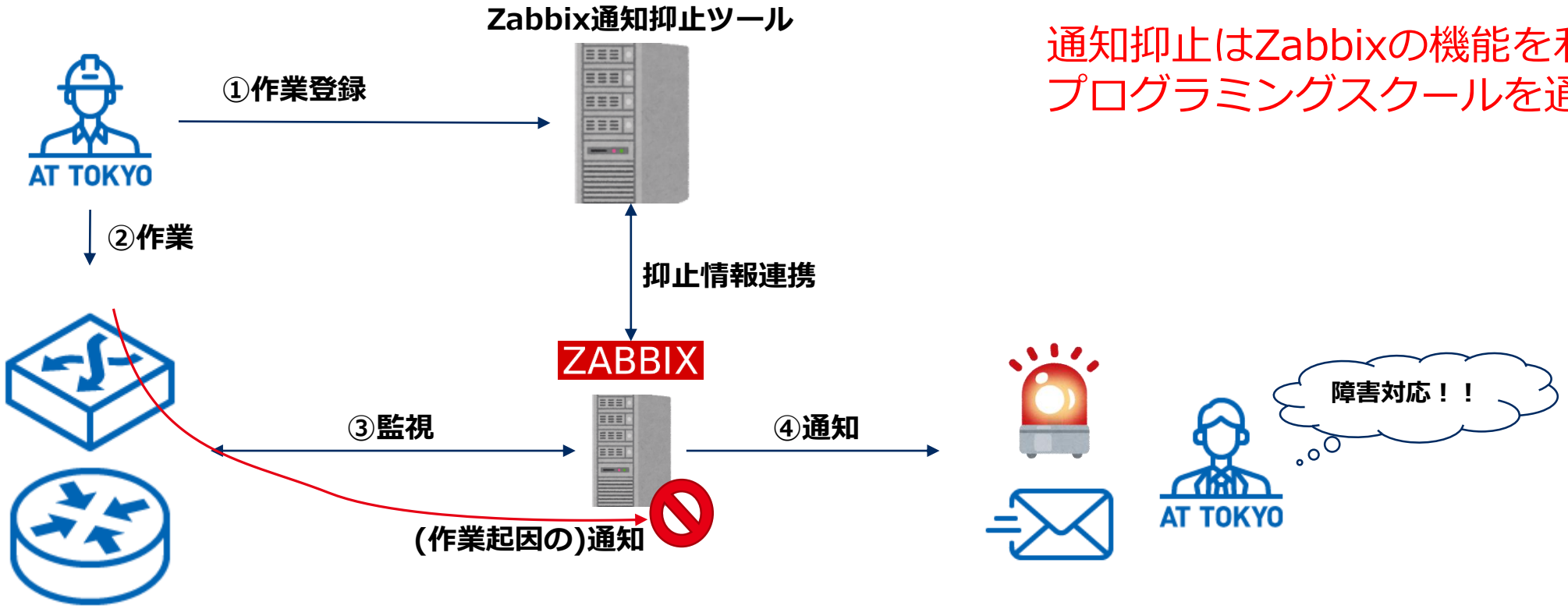
・作業単位で管理することが困難で1作業につき複数の登録が必要
・抑止の開始終了時間は1登録単位のため変更が大
・バリデーションがない（インタフェースやIPアドレス）
・作業の進捗(開始前、作業中、終了)のステータスが無い

作業登録の簡易化と入力ミスの防止

通知数：月~~1100件~~ ⇒ 月500件

Zabbix通知抑止ツール

資料のみ

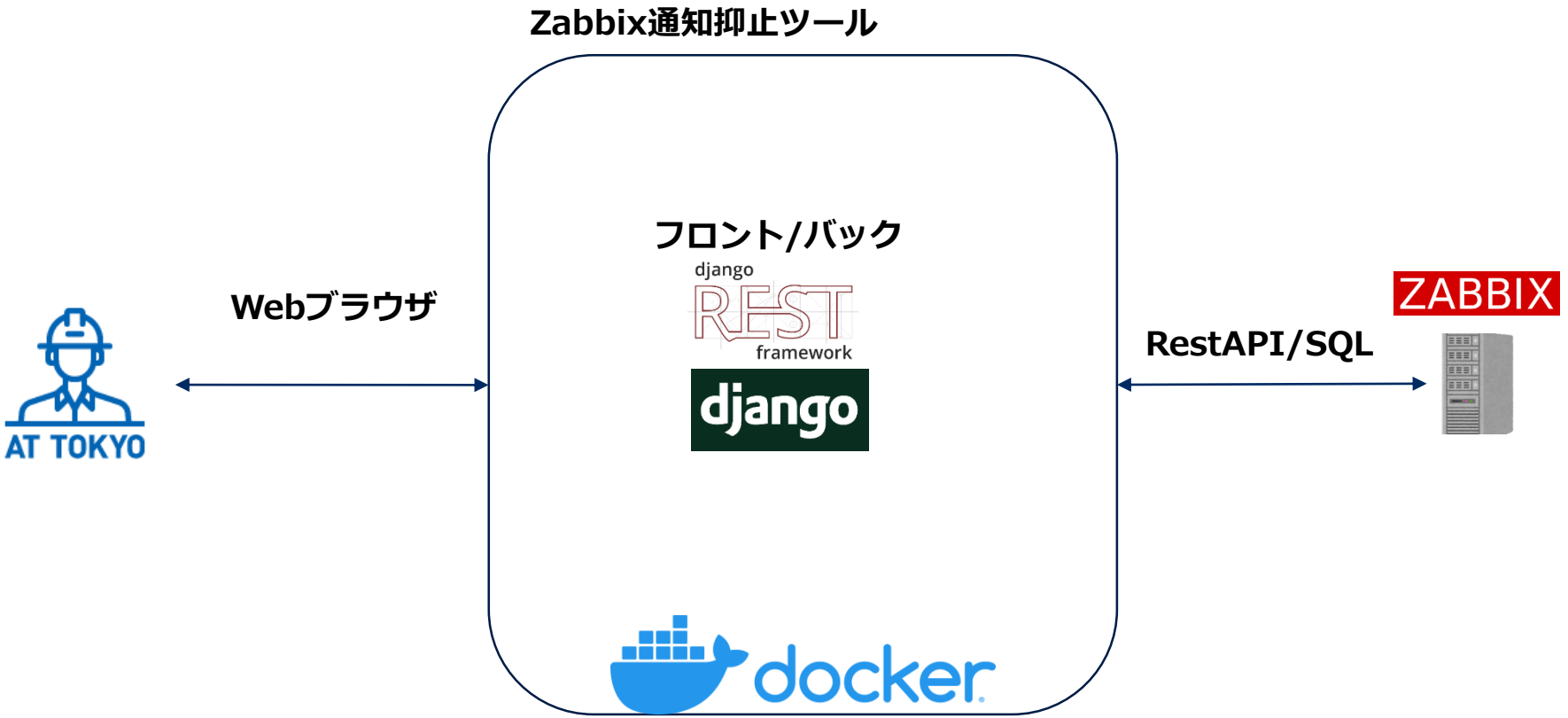


取り組み

通知抑止はZabbixの機能を利用
プログラミングスクールを通じて作成

Zabbix通知抑止ツール - システム図

資料のみ



話すこと

作成したツール

取り組み

- 組織
- 開発環境(資料のみ)

ツール・システムに関するナレッジの変化

ATLASSIAN Confluence

思考錯誤の結果を残す
⇒プロダクト開発のナレッジ蓄積

継続的な運用・フィードバックのための心がけ

開発グループと運用グループは障害対応などの業務を一緒に行っているため
距離が近くコミュニケーションは取りやすい関係です

とはいえ（だからこそ）、心掛けたこともあります

要望や実装ロジックを一緒に考える

- 開発者の独りよがりにならない
- 運用フローに負荷をかけない

要望を放置しない

- 迅速に対応することで次のフィードバックを得やすくなる

話すこと

作成したツール

取り組み

　チーム

　開発環境

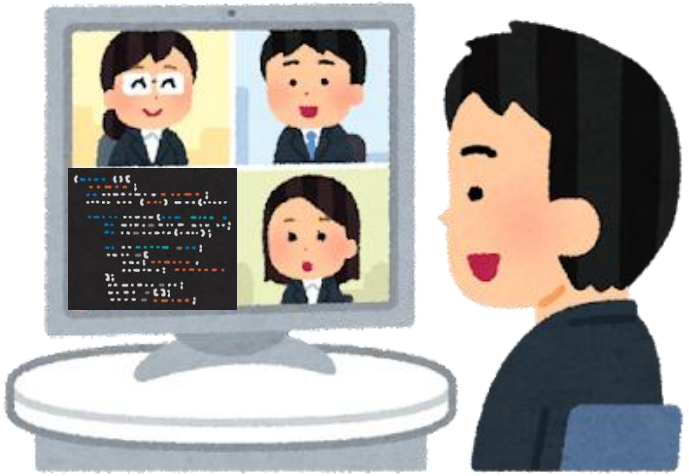
はじめました：ソースコード管理

GitLab

Issueごとにブランチを作成し、作業後マージする
マージリクエスト時にコードレビューを実施

効果

コードの共有
リファクタリング
デバッグ



はじめました：テスト

CIツール　Gitlab Runner

	テストツール
Django	TestCase
React	Jest, React Testing Library

はじめてみたが・・・

正しいテストってなんだろう
どこまでやればいいんだろう

テストムズカシイ（個人の感想です）

試験パターンはどこまで網羅する？
外部システムと連携している場合、モックが必要？
テストを書くのに時間がかかる

手動テストが残っている
習慣になっていない

はじめました:デプロイ

コードの反映はgit 、実行環境はdocker
　　更新もロールバックもスムーズに行える

ステージング環境を用意
　商用データでの動作確認
　デプロイのテスト
　　商用デプロイに失敗してロールバックを何度もやりました