サービス開発初心者チームの DevOps を用いた新サービス開発の進め方について

英文タイトル

木村　慎吾
kimura_singo@intec.co.jp
株式会社インテック クラウドプロダクトサービス部

発表要旨：

本発表はこれまでサービス運営の経験がないチームが、自社の新サービス立ち上げにあたり、サービスリリース直後からサービス運用を問題なくできることを目標としてサービス開発に取り組んだ事例紹介となる。

DevOps の考え方を意識してサービス開発を実施したことで､新サービス開発の目標であった、リリース直後から運用が問題なくできていることを達成できたと考えている。開発方法を自分たちなりにアレンジしすぎるのは問題もあるが、今回のサービス開発においては自分たちの状況をとらまえて、利用できるサービスマネジメントシステムを活用し、またアジャイル開発とウォーターフォールのハイブリットで進めていったことは問題ないと考えている。

サービス開発方法が今までのやり方と変わり、運用チームも巻き込んだ DevOps という考え方が今後は主流になっていくと考えられる。ただし、アジャイル開発におけるスクラムのようなわかりやすいフレームワーク（方法論）が DevOps においては現状ないと考えられる。よって､DevOps を進めたいがどこから手をつけてよいかわからないチームにおいては、今回の事例によって DevOps を推進できるのではないかと考えている。

キーワード：

DevOps、クラウドネイティブ開発、アジャイル開発、サービスマネジメントシステム

想定している聴衆

これからサービス開発を行う人

発表者の紹介（全角１００文字）：

サービス開発、アプリケーション開発のリーダーして従事。ここ数年はアジャイル開発へ取り組み、その経験について社内・社外において発表を行い啓蒙活動を実施している。

サービス開発初心者チームのDevOpsを用いた新サービス開発の進め方について

株式会社インテック

クラウドプロダクトサービス部

〇木村　慎吾

E-mail：kimura_shingo@intec.co.jp

セッションの概要

- 主題
  - 新規にサービスを立ち上げるときの進め方
  - 私たちが実践した経験発表
- ポイント
  - 開発と運用を分離するのではなく、開発当初からサービスの運用を見越してサービス開発を実施
- 想定対象
  - これからサービス開発を行っていく方々

自己紹介

木村 慎吾

インテック
クラウドプロダクトサービス部

開発との関わり
- サービスオーナーとして従事
- ここ数年はアジャイル開発を実践

対象となるサービスについて

- サービス概要
  - サービス名：ID認証サービス
  - リリース：2021年4月（開発期間：2019年11月～2021年2月）
- 体制
  - サービスオーナー：木村
  - 開発メンバ：9名
  - 運用メンバ：2名
  - 営業・企画など：数名
- チームの特徴
  - アジャイル開発はパッケージ開発で経験あり
  - サービスの開発・運用は今回が初めて

2021年04月07日

報道関係各位

株式会社インテック

インテック、複数クラウドサービスのID認証を一括で管理できる「ID認証サービス」に新機能を追加
～多要素認証やユーザ管理機能を提供～

TISインテックグループの株式会社インテック（本社：富山県富山市、代表取締役社長：北岡隆之、以下インテック）は、複数クラウドサービスのID認証を一括で管理できるクラウド型「ID認証サービス」に、よりセキュアな認証を実現する多要素認証機能、クラウドサービス運用をスムーズに行うためのユーザ管理機能やモニタリング・レポート機能などを追加し、本日より提供することを発表します。

「ID認証サービス」は、複数クラウドサービスのID認証を一括で管理できるクラウド型の認証基盤サービスで、クラウドサービスを開発するサービス事業者や、クラウドサービスの利用を推進する企業の情報システム部門の方などにご利用いただいております。
今回新たに追加した、証明書認証やワンタイムパスワード認証などの多要素認証機能を利用することで、よりセキュアな認証が可能となります。
また、クラウドサービス開発者は、ユーザ管理機能のAPIを利用することで、認証機能の開発ノウハウがなくても自社サービスでセキュリティレベルの高い認証機能が提供可能となります。自社で認証機能の開発の必要がないため、自社サービスのコア開発に専念することができます。

サービスの開発目標

サービスの開発時の目標

- サービスを継続的に発展させていきたい

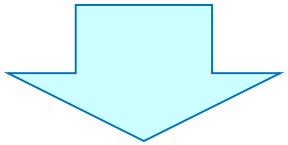
具体的な取り組み

以下の2点を意識して開発を計画

- セキュリティ系のサービスであるためリリース直後から安定した運用ができること
- クラウドネイティブ開発を取り入れること

サービス運用におけるこれまでの課題と今後の対応

- 過去のやりかたについて
  - 開発工程後に運用へ引き継ぎをする
- 問題点
  - バラバラに設計され、それぞれの観点のみで対応される
    - ギャップが発生し、スピード感がなくなる
    - 運用側でむりやり運用回避で逃げるため、サービス全体として正しいものがわかりにくくなり、改修が次第に困難になっていく
- やりたいこと
  - スピード感を持った対応
  - サービスの継続を邪魔しない対応



DevOpsを実践

開発手段を活かす

開発で意識したこと

- スピード感と継続性を確保できる仕組みをつくる

開発における仕組み

- テストの自動化
- リリースの自動化（ビルド、デプロイのパイプライン作成）
- ソースのバージョン管理

※開発自体はアジャイル開発（スクラム）で実施

※開発系の仕組み（自動化など）については、本発表の対象外

この仕組みを運用でも有効活用させていく

サービス開発開始時の課題と対応

- どこから初めてよいかわからない

  スタート時に見本となるようなものがなかった

  - DevOpsにはスクラムのようなフレームワーク的ものがない
  - 自社の事例もない

- 推進のポイント

  模索しながら進めるなかで以下の2点が進める糸口になった

  - マネジメントシステムを意識した取り組み
  - システムテスト工程の設置

サービスマネジメントシステムを意識した取り組み

- サービスマネジメントシステムからのアプローチ
  - 自社のマネジメントシステムへ準拠
    - ISO20000への対応
    - ただ、これまでの慣習による考えて方もあり調整が発生
  - 一派的なISO20000、ITILなどから取り組みを整理
    - 新しい取り組みを活かすためにそもそも論を学ぶ
    - ITIL4などの新しいものを活かす

- 取り組みポイント

  開発と運用の接点となる3つのプロセスを重要視した。
  - リリース管理（リリースプロセス）
  - 構成管理（管理プロセス）
  - インシデント管理（解決プロセス）

リリース管理について

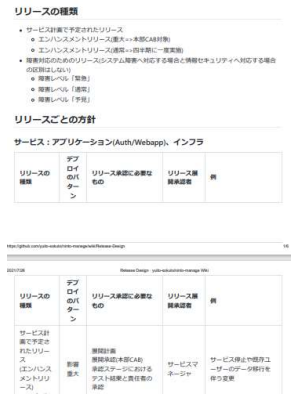
❑ 課題

社内のリリースルールと開発側の仕組みにギャップがあり、リリースごとに都度対応するための作業が発生し運用に負荷がかかる。

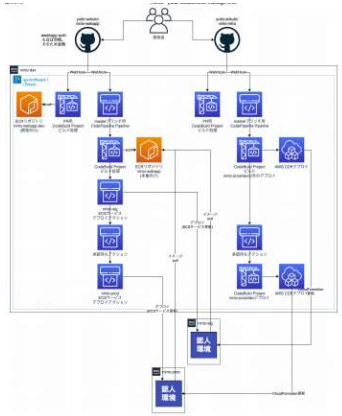
❑ 対応

社内ルールを開発主体のビルド・デプロイパイプラインに組み込む。

■リリースのルール（Ops）



■デプロイパイプライン（Dev）



構成管理について

課題

バグなどの緊急対応が発生するとインフラ面の変更管理が後回しになり、最終的にメンテナンスされず、現状がわからなくなる。

変更点

インフラのコード化

- アプリケーション開発と同様の管理方法を採用
  - Githubによるソース管理
- デプロイプロセスによって現状を管理
  - インフラ環境の現状についてデプロイプロセスのログで確認

インシデント管理について

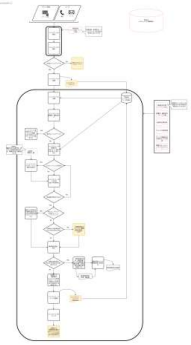
課題

- 監視項目とインシデントフローがマッチしていなく、監視アラート発生ごとに都度対応しなければならない
- アプリケーションログの定義があいまいで、運用回避ということで、監視側での対応が発生する

対応

- インシデントフローを中心に関連する処理のギャップを埋めるように認識合わせを実施
- 主に以下の項目を意識して認識あわせを実施した
  - インシデントフロー
  - 監視設定項目
  - アプリケーションログ

■インシデントフロー（Ops）



■監視設計（Dev＆Ops）

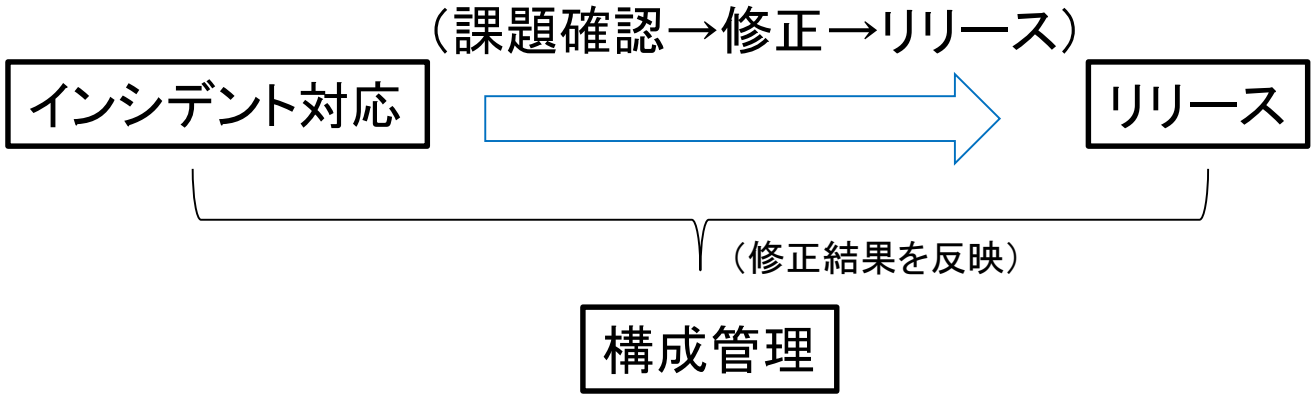
インシデント管理での気づき

- よくある運用中の課題（誤報アラート問題）への対応
  - 運用上問題がない監視アラートが発生
    - 意図していないアクセスに対して重大エラーとしてアラート検知
      - アプリケーション側が必要以上のレベルでログを出力
  - 過去の対応方法
    - 監視側で無視のような対応
    - その結果以下の課題を作り出す
      - これを繰り返すと正しい状態が何かわからなくなっていく
      - アプリケーション側は今後も似たような意味のないログを作り出す
  - 今回の対応

    サービスの継続的な発展を考えて以下の対応を実施
    - アプリケーション側の問題として修正させた
      - FATALレベルをINFOレベルとしてログ変更させた
    - 監視側は変更を加えない

サービスマネジメントシステム対応へのまとめ

- 開発と運用の接点には重要な3プロセスがある
  - 開発側が採用した手段の有効活用を一緒に考える



リリース前のシステムテスト工程について

- リリースへの課題
  - リリース後安定稼働への対策
    - 運用も開発期間で作り上げているため、イテレーションの中で全体を通して見ていくことが計画的に困難であった
  - クラウドネイティブ開発で利用する全体確認
    - イテレーションの実施中にアーキテクチャ全体を見ていくことが計画的に困難であった
- 対応
  - ウオーターフォール開発に近いがリリース前に全体確認を行うためにシステムテスト工程を設けた
  - 特に以下の確認に注力
    - インシデント管理を中心とした運用手順の確認
    - アーキテクチャ全体の確認

運用手順の確認ついて

- 開発中の課題
  - イテレーションの中では、運用目線で確認することは難しい
- システムテスト工程での対応
  - （自社）サービスマネジメントシステムの対応確認
    - インシデント対応
    - 年次、日次の運用対応
- 具体例
  - 発生したバグへ対応確認
    - システムテスト中に発生したバグで対応実施
      - インシデント管理→リリース→構成管理の流れの確認
      - インシデントフローの詳細が詰められていない場所に気づけた

    ⇒サービス開始後は問題なく運用ができている

アーキテクチャ全体の確認ついて

- 開発中の課題
  - 各イテレーションでは成果物を機能（業務）目線で確認しているため、非機能面など多角的に確認することが難しい

- システムテスト工程での対応
  - アーキテクチャ全体を確認するテストを実施
    - セキュリティテスト
    - パフォーマンステスト

- 具体例
  - WAFのアクセス検知に対する認識不足
    - セキュリティテストで検知
      - WAFは問題あるアクセスを当たり前に検知しただけ
      - ログが把握できておらず、FATALエラーとしてマッピングしていた

    ⇒サービス後、誤報がなくって安定して運用ができている

システムテスト工程まとめ

- サービス品質を守るためにアジャイル開発的ではないが必要な工程になる

初心者チームは各イテレーションで全体が見れない

- 運用手順が出来上がってからの確認が必要
- アーキテクチャが安定してからパフォーマンスやセキュリティを確認する必要がある

まとめ

- サービス開発を進めるために、以下の2つの施策が初心者チームでは有効でした。

  - 開発と運用の接続点を意識する
    - サービスマネジメントシステムのプロセスを一緒に検討
      - リリース管理、構成管理、インシデント管理が重要となる
      - 開発側の自動化などを活かす運用を考える

  - 全体が見れていない場合はそれを補う工程を設ける
    - システムテスト工程として最後にゲートを設ける
      - 運用の手順を確認する
      - アーキテクチャ全体をいろいろな角度で確認する