HITACHI
Inspire the Next

DevOpsを実現するためのWebアプリテンプレート開発とメンテナンス

2023/10/12

株式会社日立ソリューションズ
技術革新本部　生産技術部　Cloud Center of Excellence
ITアーキテクト
三好 秀徳

HITACHI
Inspire the Next

Contents

1. 背景・動機
2. 改善したいこと
3. 改善策の立案
4. 改善策の実現方法
5. 改善策の実装
6. ソフトウェア開発の変化と改善策への反映
7. 改善策実施に関する定量的評価とその効果
参考資料

HITACHI
Inspire the Next

1. 背景・動機

HITACHI
Inspire the Next

1. 当社ではDevOpsを実践するためのツールや環境を長期間提供
   - ソースコード管理：GitLab、自動ビルド／テスト：GitLab CI
   - 詳細情報
     - SPI Japan 2018『新しいITサービス開発への取り組み、DevOps実践を目指して！』
     - https://www.jaspic.org/events/sj/spi_japan_2018/

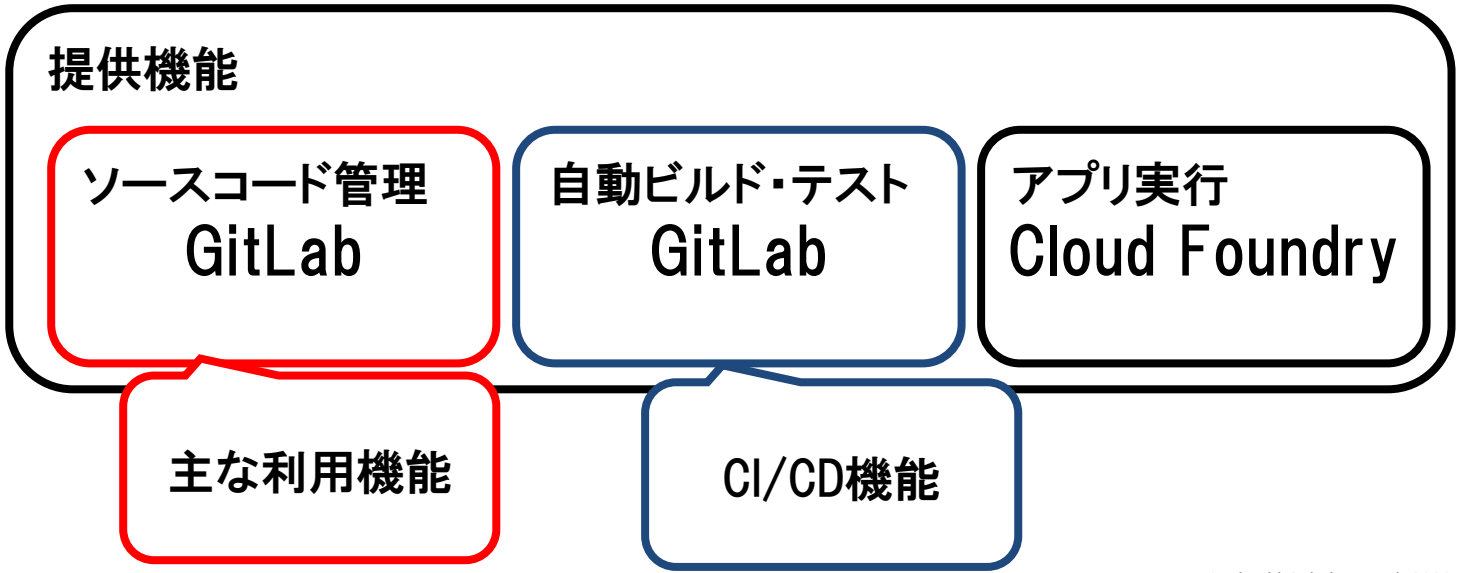
2. 改善したいこと

2-1　改善したいこと

HITACHI
Inspire the Next

ツールや環境はあるが、DevOpsの実践が低調

- 全GitLab利用プロジェクト中、CI/CD機能の利用は1割未満
  - GitLab利用のほとんどがソースコード管理としての利用



2-2 DevOpsの実践が低調な理由を探る

3. 改善策の立案

3-1 やらない理由とその対策案の思考

HITACHI
Inspire the Next

項番	やらない理由
1	これまでの手作業で困っていない
2	これまでの手作業と同じ結果が得られるのか？
3	具体的にどのように実装してよいか分からない
4	実装にかける工数に見合う成果が得られるのか不明

後からDevOpsを導入するのは困難

コピー＆ペーストできる実例が必要？

全員が頑張るDevOpsは導入が困難

DevOpsを標準とした実装をプロジェクト開始時から使えば課題は発生しない？

3-2 やらない理由とその対策案の整理

HITACHI
Inspire the Next

項番	やらない理由	対策
1	これまでの手作業で困っていない	ー
2	これまでの手作業と同じ結果が得られるのか？	● プロジェクト開始当初からDevOpsを標準とする
3	具体的にどのように実装してよいか分からない	● 日本語ドキュメントの作成 ● 実装例の提示
4	実装にかける工数に見合う成果が得られるのか不明	● プロジェクト開始当初からDevOpsを標準とする

DevOpsの実装例を盛り込んだテンプレートを開発・メンテナンス

3-2 テンプレートの開発・メンテナンスに依る想定波及効果

HITACHI
Inspire the Next

1. DevOpsを実践することに依る定量的指標の獲得
   - ミドルウェアやライブラリのバージョンアップ対応に要する時間の比較
     - ◆ 書籍・ネット情報ではなく、自身の実績値を得たい
2. テンプレートを構成する部品に関する知見の獲得と展開
   - ミドルウェア、ライブラリ、ツールに関する知見の獲得

4. 改善策の実現方法

4-1 DevOpsとは？

HITACHI
Inspire the Next

“DevOps（デブオプス）は、ソフトウェア開発手法の一つ。開発（Development）と運用（Operations）を組み合わせたかばん語であり、開発担当者と運用担当者が連携して協力する（さらに両担当者の境目もあいまいにする）開発手法をさす。ソフトウェアを迅速にビルドおよびテストする文化と環境により、確実なリリースを、以前よりも迅速に高い頻度で可能とする組織体制の構築を目指している。”

DevOps - Wikipedia 2023年8月8日(火)参照
https://ja.wikipedia.org/wiki/DevOps

ソフトウェアを迅速にビルドおよびテストすることに貢献するテンプレートを志向

4-2 テンプレートの実装方針

HITACHI
Inspire the Next

1. 当社プロジェクト実績・世間シェア・保有技術力を踏まえた技術選定
2. 手作業で実施している作業の洗い出しと自動化
3. The Twelve-Factor Appの実装
4. テンプレート自身が有する機能の単純化
   - メンテナンス性、利用可能性を高めるため
5. ツールは極力標準設定で利用する
   - 設定ファイルの設定に時間を費やすのを避けるため
6. カバレッジ100%を目標とした自動テストの実装
   - メソッド単位の単体テスト＋HTTPリクエスト単位での結合テスト

4-3 テンプレートの実装方針（本日ご説明するもの）

HITACHI
Inspire the Next

1. 当社プロジェクト実績・世間シェア・保有技術力を踏まえた技術選定
2. 手作業で実施している作業の洗い出しと自動化
3. The Twelve-Factor Appの実装
4. テンプレート自身が有する機能の単純化
   - メンテナンス性、利用可能性を高めるため
5. ツールは極力標準設定で利用する
   - 設定ファイルの設定に時間を費やすのを避けるため
6. カバレッジ100%を目標とした自動テストの実装
   - メソッド単位の単体テスト＋HTTPリクエスト単位での結合テスト

5. 改善策の実装

5-1　テンプレートの実装方針（本日ご説明するもの①）

HITACHI
Inspire the Next

1. **当社プロジェクト実績・世間シェア・保有技術力を踏まえた技術選定**
2. 手作業で実施している作業の洗い出しと自動化
3. The Twelve-Factor Appの実装
4. テンプレート自身が有する機能の単純化
    - メンテナンス性、利用可能性を高めるため
5. ツールは極力標準設定で利用する
    - 設定ファイルの設定に時間を費やすのを避けるため
6. カバレッジ100%を目標とした自動テストの実装
    - メソッド単位の単体テスト＋HTTPリクエスト単位での結合テスト

5-2 テンプレートの具体化

HITACHI
Inspire the Next

当社プロジェクト実績・世間のシェア・技術カバレッジを踏まえ選定・具体化

項番	項目	値
1	何を作るか？	Webアプリ(簡易ブログ＋認証機能)
2	言語は何にするか？	Java
3	フレームワークは何にするか？	Spring Boot

5-3 テンプレートの実装方針（本日ご説明するもの②）

HITACHI
Inspire the Next

1. 当社プロジェクト実績・世間シェア・保有技術力を踏まえた技術選定
2. **手作業で実施している作業の洗い出しと自動化**
3. The Twelve-Factor Appの実装
4. テンプレート自身が有する機能の単純化
   - メンテナンス性、利用可能性を高めるため
5. ツールは極力標準設定で利用する
   - 設定ファイルの設定に時間を費やすのを避けるため
6. カバレッジ100%を目標とした自動テストの実装
   - メソッド単位の単体テスト＋HTTPリクエスト単位での結合テスト

5-4 ヒアリングから得られた手作業の洗い出しと自動化の実装

HITACHI
Inspire the Next

わずかな人の努力でプロジェクト全体の効率化が望めるものを中心に自動化

1. ビルド
2. コーディングルール確認・修正
3. ステップカウント
4. カバレッジ取得
5. 詳細設計書作成
6. ライブラリ依存管理
7. ライブラリ更新確認
8. ライセンス情報取得
9. データベース構成図作成
10. テスト(単体・結合)

5-5 手作業の自動化で活用したツール

HITACHI
Inspire the Next

項番	項目	活用ツール
1	ビルド	Gradle
2	コーディングルール	Google Java Format
3	ステップカウント	cloc
4	カバレッジ取得	Jacoco
5	詳細設計書作成	Javadoc
6	ライブラリ依存管理	Gradle
7	ライブラリ更新確認	Gradle plugin
8	ライセンス情報取得	Gradle plugin / sbom-tool
9	データベース構成図作成	Flyway / SchemaSpy
10	テスト(単体・結合)	JUnit / Docker

5-6 テンプレートの実装方針（本日ご説明するもの③）

HITACHI
Inspire the Next

1. 当社プロジェクト実績・世間シェア・保有技術力を踏まえた技術選定
2. 手作業で実施している作業の洗い出しと自動化
3. The Twelve-Factor Appの実装
4. テンプレート自身が有する機能の単純化
   - メンテナンス性、利用可能性を高めるため
5. ツールは極力標準設定で利用する
   - 設定ファイルの設定に時間を費やすのを避けるため
6. カバレッジ100%を目標とした自動テストの実装
   - メソッド単位の単体テスト＋HTTPリクエスト単位での結合テスト

5-7 The Twelve Factor App

1. モダンなWebアプリとしてあるべき姿をまとめた12個のベストプラクティス
   - https://12factor.net/ja/
2. 抽象的かつ難解な記述で理解が難しい
   - 特定の言語やフレームワークに適した記述が求められている

5-8 The Twelve Factor App

HITACHI
Inspire the Next

1. 最近のWebフレームワークであれば標準で実装
2. 注意が必要なのは以下の点。サンプル実装を提示
   - Ⅲ.設定 設定を環境変数に格納する
     - https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.external-config
   - Ⅵ.プロセス 1つもしくは複数のステートレスなプロセスとして実行する
   - Ⅺ. ログ ログをイベントストリームとして扱う
     - https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.logging

HITACHI
Inspire the Next

6. ソフトウェア開発の変化と改善策への反映

HITACHI
Inspire the Next

1. クラウド対応

- **再試行の考慮**

“プライベート クラウド システムを含め、クラウド ホスティングには、低コストで入手できる多数のコンピューティング ノードを使った動的リソース割り当て、共有リソース、冗長性、自動フェールオーバーによって、全体的な可用性を引き上げる効果があります。 しかし、クラウド環境の性質上、一時的な障害が発生する可能性はより高くなります。”

一時的な障害の処理 - Best practices for cloud applications 2023年8月8日(火)参照
https://learn.microsoft.com/ja-jp/azure/architecture/best-practices/transient-faults

6-2 再試行に関するクラウドベンダーの情報

HITACHI
Inspire the Next

クラウドベンダー提供のSDK利用、リトライ処理用ライブラリの使用

- Amazon Web Services：ジッターを伴うタイムアウト、再試行、およびバックオフ
  - https://aws.amazon.com/jp/builders-library/timeouts-retries-and-backoff-with-jitter/
- Azure：再試行パターン
  - https://learn.microsoft.com/ja-jp/azure/architecture/patterns/retry
- リトライ処理用ライブラリ：Spring Retry
  - https://github.com/spring-projects/spring-retry

2. Docker／コンテナ対応

- Twelve Factor Appsに沿った設計・実装が前提
- コンテナイメージ作成ツールJibの活用
  - Dockerfileの作成、メンテナンスが不要
  - ベストプラクティスに沿ったコンテナイメージが作成できる
  - https://github.com/GoogleContainerTools/jib

7. 改善策実施に関する定量的評価とその効果

7-1 作業工数

HITACHI
Inspire the Next

1. テンプレートの作成、メンテナンス
   - 初期バージョン作成：20日 （2019年4月～2019年5月）
   - メンテナンス(2019年5月～)
     - メジャーバージョンアップ：1日～3日 / 6か月毎
     - マイナーバージョンアップ：3時間 / 1か月毎
2. 各種ドキュメント作成
   - 5日/ドキュメント × 10本 = 50日
     - Dockerセットアップ方法＋使い方
     - The Twelve Factor Apps解説＋実装例

7-2 テンプレートの利用数とDevOps実践に対する貢献

HITACHI
Inspire the Next

1. テンプレートの直接的な利用
   - 6プロジェクト
2. DevOps実践の推移
   - DevOps機能利用プロジェクト、実行回数ともに増加
     - 2019年4月時点： 5プロジェクト、573回実行

       ↓
     - 2023年7月時点： 18プロジェクト、2,504回実行

7-3 想定波及効果に対する定量的評価

HITACHI
Inspire the Next

1. DevOps実践に依る改善効果の定量化
   - メジャーバージョンアップ(6か月ごと) 5日 → 1日～3日
     - ◆ 非互換性候補の確認／検証に時間がかかる
   - マイナーバージョンアップ(1か月ごと) 3日 → 3時間
     - ◆ 自動テストの実行で検証の大半が終了
2. テンプレートを構成する部品に関する知見の獲得と展開
   - 作成した資料の閲覧数 累計1,000 view (2022年7月～)

参考情報

参考情報①

HITACHI
Inspire the Next

1. Spring Bootを使ったWebアプリケーションのテンプレートを社内OSSとして開発・メンテナンス
   - https://qiita.com/hidenori-miyoshi/items/e95d4f639ebc98a93084
2. Spring Boot
   - https://spring.io/projects/spring-boot
3. Google Java Format
   - https://github.com/google/google-java-format
4. Flyway
   - https://flywaydb.org/

参考情報②

HITACHI
Inspire the Next

5. SchemaSpy
   - https://schemaspy.org/
6. Jib
   - https://github.com/GoogleContainerTools/jib
7. sbom-tool
   - https://github.com/microsoft/sbom-tool
8. The Twelve-Factor App
   - https://12factor.net/ja/

HITACHI
Inspire the Next

9. 再試行に関するクラウドベンダーの資料

- Amazon Web Services：ジッターを伴うタイムアウト、再試行、およびバックオフ
  - https://aws.amazon.com/jp/builders-library/timeouts-retries-and-backoff-with-jitter/
- Azure：再試行パターン
  - https://learn.microsoft.com/ja-jp/azure/architecture/patterns/retry

HITACHI
Inspire the Next

END

DevOpsを実現するためのWebアプリテンプレート開発とメンテナンス

- Cloud Foundryは、Cloud Foundry.org Foundation , Inc.の米国及びその他の国における商標または登録商標です
- Spring Bootは、Pivotal Software, Inc.の米国及びその他の地域における登録商標または商標です
- Javaは、Oracle International Corporationとその子会社および関連企業の米国およびその他の国における登録商標です
- GitLabは、GitLab B.V. の米国及びその他の国における登録商標または商標です
- Dockerは、Docker, Inc. の米国及びその他の国における登録商標または商標です
- その他記載の会社名、製品名は、それぞれの会社の商号、商標もしくは登録商標です

HITACHI

Inspire the Next