

運用改善を始めたら

コードが書けるエンジニアが誕生した話

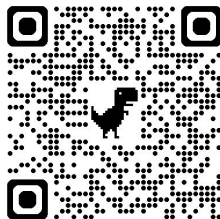
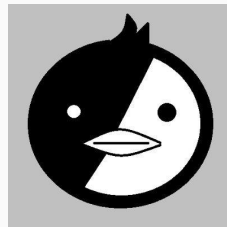
2024年09月17日 吉祥寺.pm

Who am I?

あぶかわと言います。X等では kabukawa というアカウント名で活動しています。

普段はとあるスタートアップでインフラ/運用/社内ITというか、開発以外の諸々をやる球拾い係のような仕事をしてます。

気になると思った方は <https://about.me/kabukawa> からフォロー等、ご自由に！



運用を楽しにしたかった。

手作業で運用は辛いよねという話

手作業で運用する、というのは辛い。

- 作業すべきことを決める
- 環境などの条件を確認する
- 作業を間違えないように2名以上で
- 報告用に作業証跡を作成
- 作業完了報告

そして最も大きいのは、やることは多いけどできる人は少ないという現実。

やることが多いのに、日中常にシステム止めて作業はできないからとか、バッチの動いていない時間帯じゃないと不味い、という事で実際の作業は夜中にやったりすることになりますよね。🤔

そしてそこでミスが有ると。。。😱

チームが疲弊

始めた当時はチームメンバーが自分以外に3名いたが、全てパートナー企業から来ていたので夜間作業が依頼しづらかった。

運用作業は放置して良いものではなく、やらないとサービスが動かなくなるが、負担が特定メンバーに偏っていてチームメンバーが疲弊していた。

そしてもう一つ。メンバーはコードを書くのが得意というわけではなかった。

改善しないと不味い状況であることは明らかだけど、どこから始めるのが良いだろう？

改善しよう


まずは手順書から

手順書をきちんと書こう

作業を必ず書き留めておく事から。

- 全ての実行コマンド
- 作業時間(開始から終了迄)
- 再鑑での指摘
- 間違えたところもメモ
- 前準備すべき内容/申請など
- 証跡の作成/提出手順

本当はコードで自動化とかしたいけど

体制(メンバーのスキル)的に、いきなりコード書けは無理。まずは手順を整理して何をしなければならないかを明確にする必要があった。

これがきちんとできていれば、少なくともミスと精神的な負担がだいぶ削減できるはず。(けど負担も大きい😭)

目的を持って手順書を作ると

- 目的とゴールの明確化
- チームメンバーが作業前にレビュー、知見を共有
- 無駄な手順を削減
- 作成者が手順テスト環境でデバッグ
- 次回実施時に他のメンバーが手順を再現可能
- 他チームとの作業調整で活用
- 手順書に結果も載せると提出用証跡
- 作業者が自信を持って作業できるように

明確にする

何を目的にして何をどうやるのか、というのが明確になっていれば、間違いに早く気づくことができる。

目的からやるべきことをブレイクダウンして手順として記述していくのは、ソフトウェア開発でいう、要件定義から設計のところでやっていることと考え方は大きくは変わらない。

ちゃんと自分で考えること。記録を残して次に繋げること。ここではこれを大事にした。

明確にしても手順は減らない。むしろ増えている。それでもその時は必要だった。

改善しよう


コードにしてみる


手順が有るのだからコードにできる

手順書をそのままシェルスクリプトに。

- 事前準備するデータ
- 起動/停止/再起動
- 状態確認
- データ取得/更新
- 証跡としての実行ログ
- 定期作業はcron

手順が明確であれば準備はできている

手順が明確になれば、シェルスクリプトにして手順を確実に実行したり、実行ログを残したりできるようになる。

手作業でのミスを防げるし、コードを書いたことがない初心者でもちょっと覚えれば改善に取り組めるはず。

コードにして良かったこと

- 手順のコピペミスなどが無くなる
- 何を書けばいいか迷わない
- モブプログラミングで皆で作ることができる
 - スキルレベルの平準化、知見共有、会話の活性化
- 制御構造など、基本的な事柄をマスター
 - 条件分岐、繰り返し、ファイルI/O、例外処理
- 動く楽しい→もっとコードで改善の流れ
- 躓かないように丁寧なフォロー
- 面倒な事はコードに置き換えようという意識に変化

丸投げしない

- 参考にできるスクリプトは用意しておく

```
# Get schema list
SCHEMA_LIST=`mysql -u ${DB_USER} -h ${DB_HOST} -p${DB_PASS} -P ${DB_PORT} -N -B -e
"SHOW DATABASES Like 'CM%';" | tr -d "\r"`

# Get sql file list
LIST=`ls sql/*.sql | sort`

# read tsv file and select column data
KY_LIST=`cat ${1} | grep ${CANCEL_TYPE}`
COMPANY_LIST=$(echo "${KY_LIST}" | grep ${kinko} | cut -f3 | sort -u)
```

丸投げしない

- 参考にできるスクリプトは用意しておく

```
for SCHEMA in ${SCHEMA_LIST}
do
    mkdir output/${SCHEMA}
    for sql in ${LIST}
    do
        echo -n "."
        OUTNAME=`basename ${sql} .sql`.csv
        mysql -u ${DB_USER} -h ${DB_HOST} -p${DB_PASS} -P ${DB_PORT} -B ${SCHEMA} <
        ${sql} 2> /dev/null | sed -e 's/\t/,/g' -e 's/NULL//g' > output/${SCHEMA}/${OUTNAME}
    done
    echo ""
done
```

書籍やテキスト

- Efficient Linuxコマンドライン
 - <https://www.oreilly.co.jp/books/9784814400485/>
- マスタリングLinuxシェルスクリプト 第2版
 - <https://www.oreilly.co.jp/books/9784814400119/>



設計から実装へ

手順書でやること(設計)が明確になっているから、コード実装はちょっとした書き方だけ教えれば見様見真似でできるようになる。

手順書と付き合わせて動きを確認して問題を確認するのも自分でできるので、小さなことであっても「何かを作って改善した」というの実感しやすい。

コードを書くのは楽しい、成長しているというのを実感してもらう。

繰り返される作業はコードによって単純化でき、実行時のミスという負担は軽減できた。

改善しよう


更にコードで


意外と早かった限界

シェルスクリプトはとっつきやすいが、難しいことをやろうとすると実は面倒。

- CSVファイルをjoinや編集
- jsonをテーブル形式に加工
- Lambdaで動かせない
- 数が増えると管理が大変
- 認証情報の管理

前に進むには新しい武器が必要

複雑なことをやろうとすると、逆に面倒になるということに意外と早くメンバーが気づいた。

本格的に何かプログラミング言語を使えるようにしないと、成長が止まってしまう。

言語とか環境をどうするか？

Google ColaboとPythonで

- [Google Colaboratory](#)はとにかく楽
 - 環境周りでの躓き無し
 - メソッドの補完なども効く
- ifやfor等は理解されているので書き方の違いだけ
- 最初にPandasを教えた
 - データベース、CSVファイル、json等を扱える
 - データ操作周りは基本これで完結
 - dataframeをMarkdownにして出力もできる
- 最初はググって出てきたコードのコピペかもしれない
 - ChatGPTに書かせたコードとかも

丸投げしない

- ブートストラップコードは用意しておく

```
import os
from google.colab import drive
drive.mount('/content/drive')
config_file = "/content/drive/MyDrive/config/config"
os.environ['AWS_CONFIG_FILE'] = config_file
credentials_file = "/content/drive/MyDrive/config/credentials"
os.environ['AWS_SHARED_CREDENTIALS_FILE'] = credentials_file
```

- 必要なライブラリインストール

```
!pip install boto3
```

書籍やテキスト

- Pythonプログラミング入門
 - https://utokyo-ipp.github.io/IPP_textbook.pdf
- Overview of Colaboratory Features
 - https://colab.research.google.com/notebooks/basic_features_overview.ipynb
- Pandas DataFrame UltraQuick Tutorial.ipynb
 - https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/pandas_dataframe_ultraquick_tutorial.ipynb

本格的に入門

プログラミングなどを始める時に最初に躓くのは環境構築。そこに時間を書けている余裕は無い。

Google Colaboratoryという無料で使えるPython(Jupyter Notebook)環境がある。Google Driveとの接続もできるし、できたものの共有もやりやすいので、これを使って手っ取り早くPythonプログラミングができるように。

最初のブートストラップだけ幾つか用意して、後はモブプログラミングでフォロー。

既に繰り返しや条件分岐など基本は理解できているので、意外に躓かなかった。

改善しよう

どんなものを作ったか

作ったもの達(約1年)

Python

- ECSの定期起動/停止
 - Lambda関数として作成、コスト削減のためにテスト環境のサービスを日次で制御
- AWSリソースのパラメータシート出力
 - jsonをPandasでテーブル形式にしたうえでMarkdown形式でファイル出力
- サービス間のデータ移行ツール
 - 別クラウド上のデータを取得して編集、加工、データ移行)

シェルスクリプト

- 月次のSE作業サポートツール
 - CSVファイルで定義された作業を実行
 - データバックアップ
 - 事前データ取得(比較用)
 - データ更新
 - 更新後データ取得
 - 差分出力
- 分析用ログ/データ取得
- 障害調査用ツール



60%

手作業をコードに変えることで減らせた作業時間

でも、できていないことはまだまだ有る。



道半ばです

まだまだ改善したいものはあるが、手が回っていないのが実情。

改善しようというところに共感がないと進まないのも事実。チームがチームとして機能していることが前提。

焦っても仕方がない。できることから始めるのが、やっぱり大事。始めなければ何も変わらなかった。

メンバーにどこまで任せて行けるのかは未だにいい感じのところが見つからない。

短期間(半年)でここまで来られたのは運が良かっただけかもしれない。

これから

コードの先にあるもの

It is never too late to start something. But life is not long enough to wait for you to start. So, it is now only to start. Keep it laughing for those who want to laugh. The first step to take now is more precious than anything else.

何かを始めるのに遅すぎるということはない。しかし、人生は始めるのを待っているほど長くはない。だから、始めるのは今しかない。笑いたい人のために笑い続けよう。今踏み出す最初の一步は、何よりも尊い。

まずはやる
理由を与え考える
いつか必ず
世界が変わる

ありがとうございました