使い方

1. このテンプレートリポジトリから新規リポジトリを作成
2. 設定に従って変更
3. streamlitのエントリーポイントとしてapp.pyを作成する。
   ※エントリーポイントはDockerfile側で変更できます
4. actionsから手動実行するかmainへpushをすることで自動でデプロイが走ります。デプロイ先URLはgithubのabout欄に出力されます。

設定

変数名説明

TokenはSecretsに置いてください

```
YOUR_ARTIFACT_REPOSITORY_NAME: 作られるArtifact Repositoryのリポジトリ名になります。
YOUR_CLOUD_RUN_SERVICE_NAME: 作られるCloud Runのサービス名になります。
GITHUB_ACESS_TOKEN: GitHub ActionsがAboutにデプロイ先URLを書き込むために使うトークン。必要なのはAdministrationへのRead/Writeです。
YOUR_GCS_BUCKET_NAME: デプロイ時に作成されるGCSのバケット名です。
Streamlit側からは環境変数GCS_BUCKET_NAMEを通じて参照できます。
```

1. cloudbuild.yaml

```
substitutions:
  _REPO: YOUR_ARTIFACT_REPOSITORY_NAME
  _SERVICE: YOUR_CLOUD_RUN_SERVICE_NAME
  _GCS_BUCKET_NAME: YOUR_GCS_BUCKET_NAME
```

2. GitHub Variables

```
CLOUD_RUN_SERVICE: YOUR_CLOUD_RUN_SERVICE_NAME
```

3. GitHub Secrets

```
GH_ADMIN_TOKEN: GITHUB_ACESS_TOKEN (used to update url in
"About")
```

GCSとの接続

デプロイ時にYOUR_GCS_BUCKET_NAMEに設定した名前のバケットが作成されています。これはStreamlit側からは環境変数GCS_BUCKET_NAMEを通じて参照できます。
また、GCSとのやり取りを簡単に行うためにGCSWrapperというラッパークラスが用意されています。
以下はGCSWrapperを用いてGCSにファイルをアップロード・ダウンロードするシンプルなサンプルコードです。

```
import streamlit as st
from GCSWrapper import GCSWrapper

gcs = GCSWrapper()
uploaded = st.file_uploader("アップロードするファイルを選択")

if uploaded:
    if st.button("GCSにアップロード"):
        gcs.upload(uploaded,uploaded.name)

names = gcs.get_file_names()
if not names:
    st.write("ファイルがありません")
else:
    selected_file = st.selectbox("ダウンロードするファイルを選
択",names)
    if selected_file:
        data = gcs.download(selected_file)
        st.download_button(label="ローカルに保
存",data=data,file_name=selected_file)
```