

74th

Shell Scriptの代わりに
Pythonタスクランナー
Fabric & Invoke を
活用する技術



74th 著

Shell Script の代わりに Python タスクランナー Fabric & Invoke を活用する技術

74th 著

2018-10-08 版 発行

目次

第 1 章	シェルスクリプト、Make、だんだん黒魔術化してきた	5
1.1	Fabric をちょっと覗く	8
第 2 章	タスクランナー Invoke の機能解説	11
2.1	Quick Start	12
2.2	ディレクトリを一時的に移動する <code>invoke.Context.cd()</code>	16
2.3	ヘルプを表示する	17
2.4	様々なパラメータの指定	19
2.5	終了コードで分岐する <code>invoke.Result.ok/failed</code>	21
2.6	標準出力を使う <code>invoke.Result.stdout</code>	23
2.7	引数を指定しない場合に実行するタスクを指定する <code>@task(default=True)</code> 24	
2.8	依存関係を指定する <code>@task(depended_task)</code>	26
2.9	<code>sed</code> の代わりに使う	28
2.10	<code>jq</code> の代わりに <code>JSON</code> を処理したり、 <code>YAML</code> を扱う	30
2.11	環境変数を設定する	32
2.12	ユーザ入力を伴うコマンドを自動入力する	33
2.13	ユーザ入力を伴うタスクを作る	35
2.14	<code>sudo</code> を実行する	36
2.15	Name Space を使う	38
2.16	設定ファイルを使う、設定を変更する	41
	使い方	41
	設定項目例	43
第 3 章	リモートサーバタスクランナー Fabric	44
3.1	Quick Start	44

3.2	Invoke のライブラリとして Fabric を使う	48
3.3	複数のホストに対して同時に実行する	50
3.4	リモートサーバにポートを繋げて作業する	52
第 4 章	Tips	54
4.1	コンテナで動かす	54
4.2	MacOS で Pyenv/VirtualEnv 環境と共存する	59
4.3	型補完と一緒に使う	60
4.4	Fabric(v1) と Fabric2 と Fabric3 について	61
4.5	Linux でユーザ権限でインストールする場合	62
4.6	Windows で動かす	64
付録 A	リンク集	66
A.1	Invoke	66
A.2	Fabric	66
付録 B	リファレンス	67
B.1	Invoke.Context リファレンス	67
	メソッド	67
	プロパティ	68
B.2	invoke.Context.run() メソッドリファレンス	69
	パラメータ	69
B.3	invoke.Result クラス リファレンス	72
B.4	fabric2.Connection クラスリファレンス	73
	メソッド	73
終わりに		76
著者紹介		77
	74th	77

第 1 章

シェルスクリプト、Make、だんだん黒魔術化してきた

どんな環境でも動作するプログラミング言語として、Shell Script は 2018 年現在も人気である。そしてそれは今後 10 年もそうであろう。しかし、積極的に Shell Script を書きたい！と思っている人はどれほどいるのだろうか。Shell Script は便利ではあるが、私は以下の点で扱いにくいと考えている。

- 文字列の操作にも `cat`、`sed` などのコマンドが必要であり、Shell Script でタスクを処理させるにはそれらのコマンドを使いこなすことも必要である。
- 多くのエディタの場合、型チェックや書式のチェック機能を持っておらず、動作させたときに初めてエラーやロジックのミスに気づく。
- すべてが文字列表現を介してロジックが動作する。変数の中身が数値であることなどを保証しない。
- Windows 上で満足に動作させるには、Windows Subsystem Linux や Cygwin など別のツールが必要になる。
- オブジェクト指向的な扱いは難しい。

Shell Script はそれ自体が直接的な役割を持つことは少なく、一連のコマンドを呼び出すことが役割である事が多い。そのような、コマンドを呼び出したり、タスクをこなす役割で登場したものには、以下のようなものがある。

- Make ソースコードのビルドのためのタスクランナー。ファイル間の依存関係を解決することに特化している。
- Gulp Javascript のためのタスクランナー。プラグインシステムを持ってお

1.1 Fabric をちょっと覗く

Fabric とは、Jeff Forcier 氏 (@bitprophet) が作成した、Python で動作するタスクランナーである。2018 年 5 月に、Version2 がリリースされ、Python2 に加えて Python3 がサポートされた。現在でも活発に開発が行われている（ただし、@bitprophet による仕事がほとんど）。なお、ライセンスは 2 条項 BSD である。

Fabric は、Invoke というタスクランナーを内包している。役割としては以下のように分かれている。

- ローカル環境でのタスクランナー: Invoke
- SSH 接続したサーバ上で実行するタスクランナー: Fabric

ここでは、まず Invoke を簡単に紹介する。Invoke ではタスクを tasks.py という名前のファイルを作成する必要がある。

たとえば、Google Compute Engine のインスタンスを削除するスクリプトを Shell Script で実装すると、以下ようになる。

リスト 1.1: delete_instance.sh

```
#!/bin/bash
if [ $# -ne 1 ]; then
    echo "delete_instance.sh <NAME>"
    exit 1
fi
gcloud compute instances delete --quiet $1
```

```
./delete_instance.sh hoge_instance
```

一方、Invoke で記述すると以下ようになる。

リスト 1.2: tasks.py

第 2 章

タスクランナー Invoke の機能解説

Fabric は、タスクランナー Invoke を内包しており、ほとんどの機能は Invoke が提供している。Invoke は、ローカルコンピュータ上で実行されるタスクランナーである。

Fabric と Invoke の違いは以下の通りである。

- Fabric: SSH 接続先のコンピュータ上でタスクを実行する。接続先の環境変数を引き継いでコマンドを実行する。
- Invoke: ローカルコンピュータ上でタスクを実行する。実行環境の環境変数を引き継いでコマンドを実行する。

SSH 接続を要する用途以外では Invoke の機能で基本的に十分である。

2.1 Quick Start

Invoke は pip からインストールできる。

```
PY3 $ pip3 install invoke
PY2 $ pip install invoke
```

タスクを記述するファイル名は、**tasks.py** である必要がある。

tasks.py には、タスク名で関数を定義する。関数には、@task のデコレータを付ける。

リスト 2.1: tasks.py

```
from invoke import task

@task
def build(c):
    print("Building!")
```

このタスクは、下記のコマンドで実行できる。

```
$ invoke build
Building!
```

ここでタスクの関数の第一引数には `invoke.Context` が格納される。この `invoke.Context` の `run` メソッドでコマンドを実行できる。

Golang のビルドを行う例を示す。

リスト 2.2: tasks.py

```
from invoke import task
```