Python, Cloud and Automation

3. Automation, Cloud and Examples

Jack Minchin

Tourism Economics

2022

Table of Contents

What are cloud solutions?

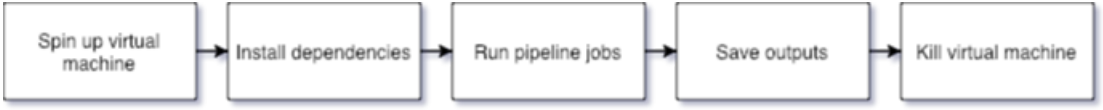
- At the most basic level, cloud computing is the ability to access computers that are not on local premises.
- In practise, cloud technology is an umbrella term for a variety of different computing services (file storage, compute, Platform as a Service (PaaS), cloud functions and pipelines)
- There are a number of cloud providers, at OE we have access to Azure and AWS. Azure is my preferred choice because of its integration with Active Directory (Microsoft) and the more verbose naming conventions (Azure App Service vs. AWS Elastic Beanstalk)

Some examples

- **Azure Portal** The Azure portal is a website where you can manage cloud products.
- **Azure Storage Accounts / Blob Storage** Storage accounts allow the upload and storage of files in the cloud. Blobs (or files) can be downloaded from the Azure Portal or programmatically.
- **Virtual Machines** VMs are computers that run in the cloud they can be used to run application or speed up run times if you are dealing with large datasets, you can interact them with the command line (SSH) or remote desktop (if the VM is running windows)

Pipelines

- Pipelines are automated tasks that run in the cloud they can be triggered by pushing to a git repository, or through an API call.
- In simple terms, they take an ordered set of commands and execute them on a remote virtual machine, the machine is set up specifically for the task and then killed immediately after the pipeline is complete.
- There are number of providers that offer pipelines but I have been using CircleCI because of their free tier.



Setting up a pipeline

- When using circleci, you define pipelines in a text format called YAML.
- This file will define the type of machine that should be created and the steps that the pipeline will include.
- Sometimes, it will also include how the pipeline will be triggered.

```yaml
# .circleci/config.yml

version: 2.1

# Define the jobs we want to run for this project
jobs:
  exampleJob:
    executor: # This step defines what type of environment to use
      name: 'win/default'
      size: 'medium'
    steps: # Below we set out the steps to use in the pipeline
      - checkout
      - run:
          name: 'Print some text' # The name of the step
          command: echo "this is the build job" # This will just return the text

# Orchestrate our job run sequence
workflows:
  build_and_test:
    jobs:
      - exampleJob
```

A completed pipeline

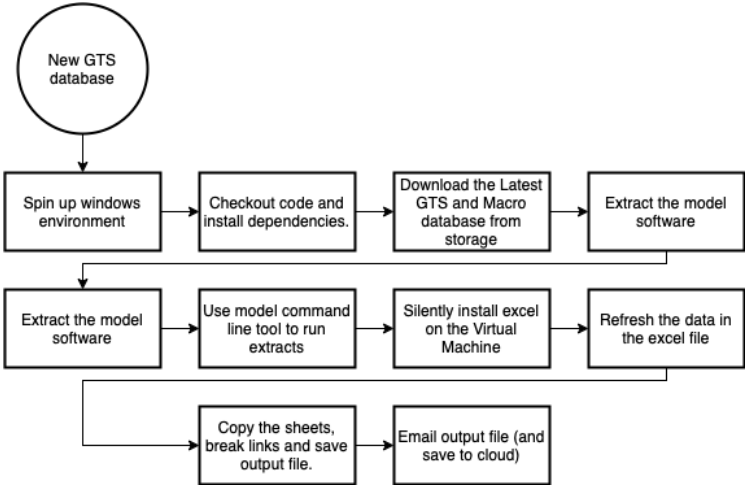
Some considerations

- Pipelines do not only run Python, and many cases Python isn't the best tool to use for orchestrating pipelines. The base level for interacting with pipelines is the command line meaning 99% of tasks can be acheived.
- Any coding language can be used, and there will be a solution allow most programs to be interacted with:
  - Excel $\rightarrow$ Powershell
  - File structure changes $\rightarrow$ Powershell
  - Stata $\rightarrow$ Do files
  - EViews $\rightarrow$ Scripts
  - OE Models $\rightarrow$ MDL Command line tool

Example 1: Whitbread Outputs (Basic)

- Whitbread take a very basic quarterly breakdown of some GTS and Macro model indicators.
- The process is:
  1. Run the macro and GTS extract
  2. Update the links in the model file
  3. Copy the output sheets into a new workbook.
  4. Break links and save.
- Manual process time 40 minutes - 1 hour.

Creating a pipeline



Languages & Technologies Used

- Python - for running some of the orchestration
- PowerShell - Interacting with Windows & Excel
- Azure Storage - storing the database files
- Git - Managing the project and codebase
- CircleCI YAML - Writing the pipelines steps

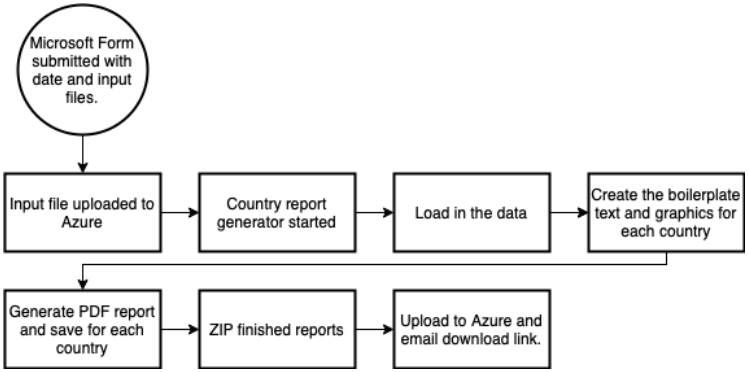
Time saving

Manual method: 1 hour.
Pipeline method: 8 minutes

Example 2: Country Report Generation (Advanced)

- The country report generator create $\sim$ 40 PDF reports with text and plots.
- The manual process:
  1. Creation of input file.
  2. Load excel file for each report.
  3. Update the links in each excel file.
  4. Rewrite boiler plate text to reflect new data.
  5. Fix plots where necessary.
  6. Write custom commentary.
  7. Save to PDF.

Creating a pipeline



Languages & Technologies Used

- Python - for orchestration, creating text and plots.
- Azure Storage - storing the input and output files
- Azure Functions - triggering the process when a new input file is added.
- Node.JS - application architecture and PDF generation.
- Git - Managing the project and codebase
- CircleCI YAML - Writing the pipelines steps.

Time saving

Manual method: 3 people 3 days.
Pipeline method: 10 minutes + time to write custom text (1 day?).

Sweden Regional TSA PowerPoints

- Creation of 8 regional TSA reports in PowerPoint. The text and graphics are entirely boilerplate.
- Using Python and a package called python-pptx, the code takes the regular output from the Excel model file and outputs the reports in .pptx and PDF.

Time saving

Manual method: Multiple days
Pipeline method: 40 seconds

APF IATA Data Inputs

- We receive a monthly 2020 - To Date data file for PAX and RPK figures from IATA.
- It is in a long form structure with full country names.
- Process:
  1. Receive input files
  2. Convert to wide form, change country names to model codes.
  3. Extrapolate to the end of the current quarter.
  4. Export readin file for model.
- Uses Python notebooks.

APF Automated Checks

- The number of variables in the APF model mean that it is impossible to check each variable series manually.
- We can load an extract from APF and automatically run tests, e.g:
  1. Ensure no negative values.
  2. Ensure that current release is with x% of previous release.
  3. Ensure that PAXALL (sum of bilateral indicators) is not greater than PAX (total pax from IATA)