Shabna Nazar

FEB 2023

Section1: DataBASE CONFIGURATIOn WITh TERRAFORM and Python for ETL

Table of Contents

[1. Use Case Details 2](#_Toc127822827)

[2. Pre-requisites 2](#_Toc127822828)

[3. Approach and Execution 2](#_Toc127822829)

[3.1 Azure Resource Provisioning Using Terraform: 2](#_Toc127822830)

[3.2 Python ETL: 3](#_Toc127822831)

[4. Process Diagram: 4](#_Toc127822832)

[5. How to scale the Data Pipeline for a large dataset? 5](#_Toc127822833)

# Use Case Details

**Section 1 Details: Database Configuration & Python ETL**

Provision one database of your choosing (SQL, NoSQL, Graph).  Write a python ETL that ingests the provided data, transforms it in some way, and loads it into the database. This should be reproducible code with documentation. (Terraform / Cloudformation / Ansible, docker-compose etc).

This use case has two sections:

1. Azure resource provisioning using Terraform, running from local in VSC.
2. Python code running locally using Anaconda to ingest source file from local and load to Azure DB in cloud.

# Pre-requisites

* An Azure account.
* Visual Studio Code GUI available in local.
* Terraform installed.
* Azure plugin installed in VSC.
* Anaconda or any Python GUI in local.

# Approach and Execution

## 3.1 Azure Resource Provisioning Using Terraform:

Approach: Execute sql.tf file from VSC.

-> Assign resource group name and location.

-> Create SQL Server

-> Create SQL DB in the newly created server.

-> Add firewall rule to provide access to client machine.

Execution:

-> Pre-requisites - locally installed visual studio code.

-> Install Azure tools in VSC.

-> Terraform installed and plugged in.

Run Command ‘az login’ in terminal

Text

Description automatically generated

-> Initialize terraform: Command- terraform init

Text

Description automatically generated

-> Verify the plan using: Command- terraform plan

-> Execute and provision: Commanmd- terraform Apply->enter 'yes'

## 3.2 Python ETL:

Approach:

-> Copy the source file in local. Change the file path in code to match your local

soure path.

-> upload the SSL file from GIT to your working directory (same as source file path).

-> Ingest the file to DataFrame using Anaconda notebook.

-> Perform transformations:

1) Basic Checks

2) Sorting by name

3) Add column for full name and move the position

4) Reset and adjust the index

5) email column verification

-> Connect to mySQL server/DB

-> Load Data !!! GOAL ACHIEVED

-> Extract from table for proof.

Note: Modules needed: Please install these if not available in your local.

import os

import time

import pandas as pd

from email\_validator import validate\_email, EmailNotValidError

import mysql.connector

# Process Diagram:

VSC/local

Azure Cloud

Terraform Terminal

SQL

Load Data and Extract

Source File/local

Anaconda

Python ETL

# How to scale the Data Pipeline for a large dataset?

-> Auto create the table using Data Frame structure.

-> Update ETL section.

-> Provision Azure storage account using Terraform

-> Ingestion of source file from Azure storage.

-> Create separate layers for large datasets, Raw, Clean and Processed