Day 15 Task: Python Libraries for DevOps

Reading JSON and YAML in Python

- As a DevOps Engineer you should be able to parse files, be it txt, json, yaml, etc.

.JSON (JavaScript Object Notation) and YAML (YAML Ain't Markup Language) are both commonly used formats for storing and exchanging data in Python.

JSON is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language-independent but uses conventions that are familiar to programmers of the C family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, etc. These properties make JSON an ideal data-interchange language.

JSON is a standard format for data exchange, which is often used for exchanging data between a server and a web application. In Python, the json module provides functions for working with JSON in Python. This module provides two methods for encoding Python objects into JSON format, json.dump() and json.dumps(), and two methods for decoding JSON, json.load() and json.loads()

YAML, on the other hand, is a human-readable data serialization format. It is often used for configuration files and data exchange between languages that have different data structures. YAML is a superset of JSON and can be used to represent complex data structures such as lists and dictionaries. It is also less verbose than JSON, making it easier to read and write.

Both JSON and YAML can be easily read and written in Python using built-in libraries such as json and yaml. For example, to read a JSON file in Python, you can use the json.load() method, and to write a JSON file, you can use the json.dump() method. Similarly, to read a YAML file in Python, you can use the yaml.load() method, and to write a YAML file, you can use the yaml.dump() method.

In summary, both JSON and YAML are popular formats for storing and exchanging data in Python. JSON is a lightweight, easy-to-parse data interchange format, while YAML is a human-readable data serialization format that is often used for configuration files. Both can be easily read and written in Python using built-in libraries.

Tasks

1. Create a Dictionary in Python and write it to a json File.

   To create a dictionary in Python and write it to a JSON file, you can use the json library. Here is an example:

```
>>> import json
>>> #Create a dictionary
>>> data = {
... "Name": "DevOps Engineer",
... "Age" : 17,
... "address": {
... "city": "Pythontown"
... },
... "phone_number":[22-45-77852]
... }
>>> #write the dictionary too a JSON file
>>> json_string = json.dumps(data)
>>> print(json_string)
{"Name": "DevOps Engineer", "Age": 17, "address": {"city": "Pythontown"}, "phone_number": [-77875]}
>>>
```

2. Read a json file services.json kept in this folder and print the service names of every cloud service provider.

```
>>> data_dict = {
...      "services": {
...      "debug": "on",
...      "aws": {
...      "name": "EC2",
...      "type": "pay per hour",
...      "instances": 500,
...      "count": 500
...      },
...      "azure": {
...      "name": "VM",
...      "type": "pay per hour",
...      "instances": 500,
...      "count": 500
...      },
...      "gcp": {
...      "name": "Compute Engine",
...      "type": "pay per hour",
...      "instances": 500,
...      "count": 500
...      }
...      }
...      }
>>> print(data_dict['services']['aws']['name'])
EC2
>>> print(data_dict['services']['azure']['name'])
VM
>>> print(data_dict['services']['gcp']['name'])
Compute Engine
```