# QATIP Intermediate
# Azure Lab02
# HCL versus JSON

## Contents



## Lab Objective

1. This lab will compare a Terraform HCL (HashiCorp Configuration Language) configuration with a JSON formatted terraform file . You will compare an existing HCL file (main.tf) with its JSON equivalent (json/main.tf.json), apply the JSON-based Terraform configuration, and modify it to add an additional container.

## Teaching Points

1. Terraform supports both HCL and JSON formats, each catering to different use cases. HCL is designed to be human-readable and easy to work with for developers, while JSON provides a structured format that integrates well with automation tools and APIs. Understanding how to translate HCL into JSON helps students work in environments where

JSON configurations are required for interoperability with external systems.

2. One of the key differences between HCL and JSON is their structure. HCL allows for more concise and readable configurations using blocks, whereas JSON enforces strict key-value formatting and explicit nesting. This makes JSON more verbose but also ensures consistency in machine-parsed configurations. By comparing the two formats, students will develop a stronger grasp of how Terraform configurations are structured and processed.
3. Another crucial concept in this lab is modifying JSON-based Terraform configurations. JSON configurations often require explicit syntax, including the use of arrays for lists and clear key-value assignments. Students will learn how to make structured modifications, such as adding additional containers, while maintaining correct JSON formatting. This reinforces best practices in handling Terraform configurations in JSON and avoiding syntax errors.
4. Deploying a Terraform configuration from JSON follows the same principles as HCL-based deployments, but it demonstrates the importance of maintaining structured and well-organized files. Through this hands-on experience, students will gain confidence in applying Terraform configurations in different formats and troubleshooting potential issues in JSON-based deployments.

## Before you begin

1. Ensure you have completed Lab0 before attempting this lab.
2. In the IDE terminal pane, enter the following command...

   **cd c:\azure-tf-int\lab\02**

3. This shifts your current working directory to labs\02. Ensure all commands are executed in this directory
4. Close any open files and use the Explorer pane to navigate to and open the labs\02 folder.

## Step-by-Step Instructions

### Step 1: Open and compare the two provisioned files

1. In your IDE, open both:
    - \02\main.tf (HCL format)
    - \02\json\main.tf.json (JSON format)

2. Compare the structure:
    - HCL uses blocks ({}), while JSON uses explicit key-value pairs.
    - HCL lists are inline, whereas JSON represents them as arrays ([]).
    - HCL is more compact, whereas JSON requires explicit declarations.

3. Considerations
    - Which structure do you find easier to read?
        - HCL is often considered more readable due to its concise syntax, but JSON is standardized for programmatic use.
    - How does JSON handle nesting compared to HCL?
        - JSON requires explicit key-value pairs, making deeply nested structures more verbose than HCL.
    - Why does Terraform support both formats?
        - HCL is developer-friendly, while JSON allows integration with automation tools.

### Step 2: Apply the JSON Configuration

1. Navigate to the JSON directory: **cd c:\azure-tf-int\lab\02\json**
2. Initialize Terraform: **terraform init**
3. Apply the JSON-based configuration: **terraform apply -auto-approve**
4. Verify the deployed container: **docker ps**
    - The output should show the webserver container running on port 88.
    - Browse to http://localhost:88
5. Verify the downloaded images: **docker images**
    - The output shows the downloaded httpd image details

### Step 3: Modify the JSON to Add Another Container

1. Add a **second container** by duplicating the "**webserver**" block inside the "**docker_container**" block
2. Insert a comma between the 2 sub-blocks
3. In the duplicated block, change the identifier and name to "**webserver_2**" and change the external port from 88 → **89**
4. Updated main.tf.json Snippet:

```
"docker_container": {
 "webserver": {
  "image": "${resource.docker_image.httpd.image_id}",
  "name": "webserver",
  "ports": [
   {
    "internal": 80,
    "external": 88
   }
  ]
 },
 "webserver_2": {
  "image": "${resource.docker_image.httpd.image_id}",
  "name": "webserver_2",
  "ports": [
   {
    "internal": 80,
    "external": 89
   }
  ]
 }
}
```

### Step 4: Apply the Modified JSON Configuration

1. Save the updated main.tf.json
2. Run **terraform apply -auto-approve**

3. Verify that two containers are running using **docker ps**
    - The output should list webserver:88 and webserver_2:89
4. Considerations
    - How does JSON handle nested structures differently from HCL?
        - JSON requires explicit key-value nesting, making deep structures more verbose.
    - What challenges did you face modifying the JSON?
        - JSON requires strict syntax, including commas and proper indentation, which can be error-prone.
    - Why does Terraform support both HCL and JSON?
        - HCL is more human-readable, while JSON is suited for automation and integrations.

### Step 5: Lab Clean-Up

1. Remove all resources using **terraform destroy** followed by **yes**
2. Confirm the containers have been destroyed using **docker ps**
3. Confirm the image has been destroyed using **docker images**

**### Congratulations, you have completed this lab ###**