Go Smarter Hands-on Lab Guide

Automating and Orchestrating Application Services

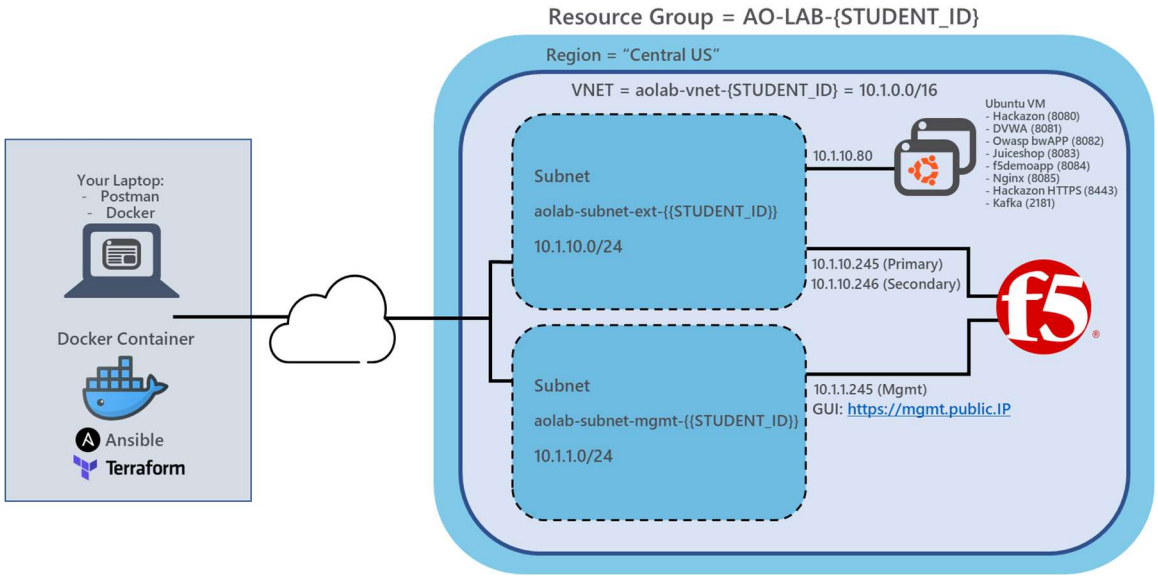
Exercise Guide - Version 1.2

Table of Contents

Contents

1. Getting Started

1.1 Lab Topology



- 1 x Ubuntu Server with a private IP address acting as our Pool Member(s).
- 1 x BIG-IP v15.1 instance (2-NIC, PAYG)

Lab Components:

Component	IP Address	Credentials
Linux Server	Private: 10.1.10.80	SSH: azureuser / f5DEMOs4uLATAM
BIG-IP	Private: • 10.1.10.245 (self-IP) • 10.1.10.246 (For Virtual Servers) • 10.1.1.245 (Management) Public IP: Check in Azure Portal or with the automation tool	SSH: azureuser / f5DEMOs4uLATAM GUI: azureuser / f5DEMOs4uLATAM

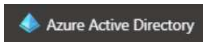
1.2 Prerequisites

- **Azure account**: Although it is possible to run the lab using a free Azure account, please have in mind that it requires the entry of billing information. For those desiring to set up an account refer to this tutorial on Setting up an Azure Development Account:
  https://clouddocs.f5.com/training/community/iam/html/class6/close.html

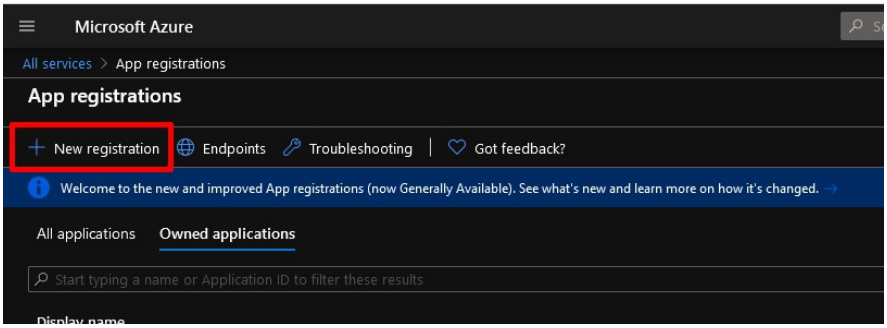
- Postman client in your laptop - https://www.postman.com/downloads/
- Ubuntu 18.04 Server with Internet access to deploy Azure Infrastructure using Ansible

2. Lab 1 – Azure Setup

Task 1 – Register your Client Credentials with the Azure AD v2.0 endpoint

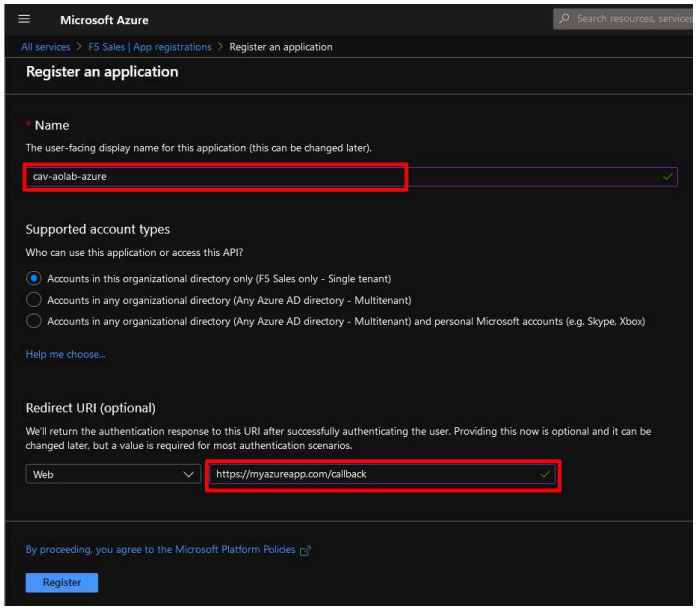
1. Sign-in to the Azure Portal https://portal.azure.com/#home
2. If your account gives you access to more than one tenant, select your account in the top right corner, and set your portal session to the Azure AD tenant that you want.
3. In the left-hand navigation pane, select the Azure Active Directory service 

and then select **App registrations** > **New registration**.



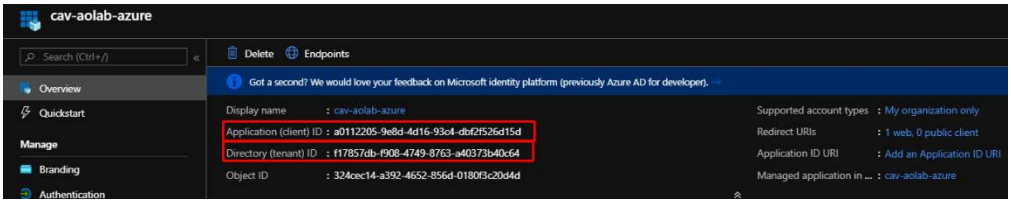
Note: For naming convention purposes, select a prefix which can be easily identified by you, such as your name initials. This will be used whenever you see [STUDENT_ID] mark in this section.

4. When the Register an application page appears, enter your application's registration information:
   - **Name**: [STUDENT_ID]-aolab-azure
   - **Redirect URI**: https://myazureapp.com/callback
   - Click Register

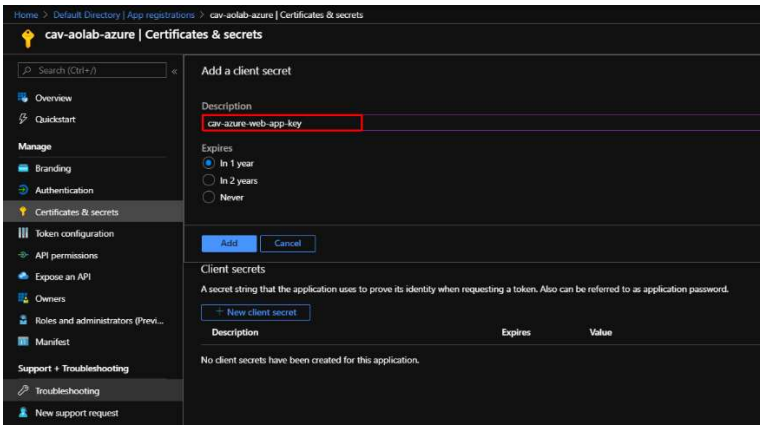


5. When the application is created, **make a note of the following parameters** to use in subsequent steps.
   - **Application ID (Client ID)**

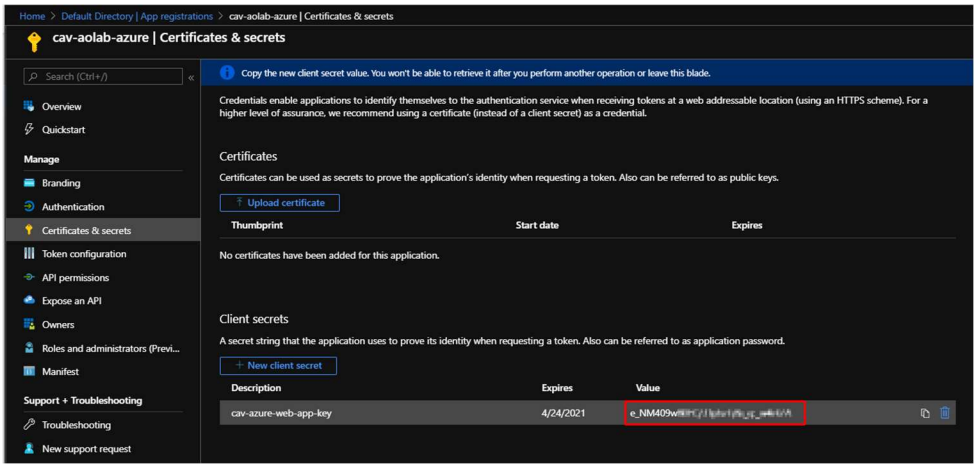
- **TENANT ID**



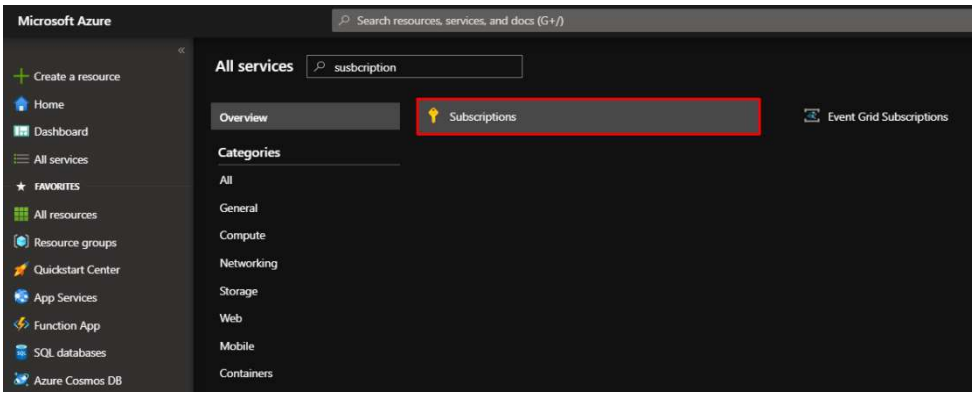
6. Select Certificates & Secrets > New client secret, then click on **Add**, while providing following information:
   - **Description**: [STUDENT_ID]-azure-web-app-key



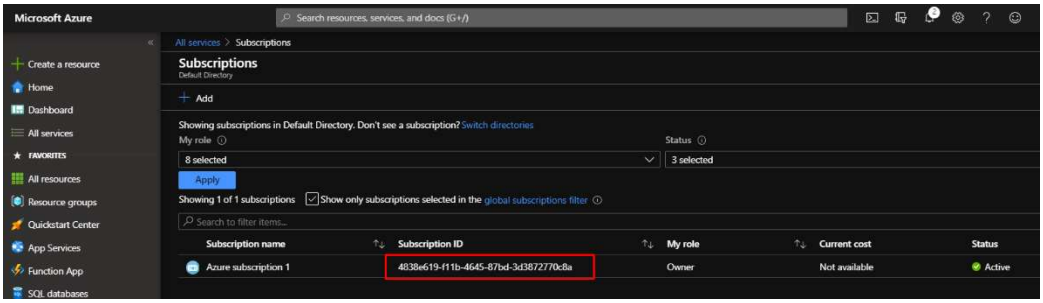
Note: After creation is complete, **make note of the secret** before closing **Certificates & Secrets** tab (After moving away from this page, you no longer can see the Client Secret and you will need to create a new one)



7. From the left-side menu, select All Services, then look for Subscriptions at the search field.



8. When the subscription screen appears, **take note of the SUBSCRIPTION ID**.

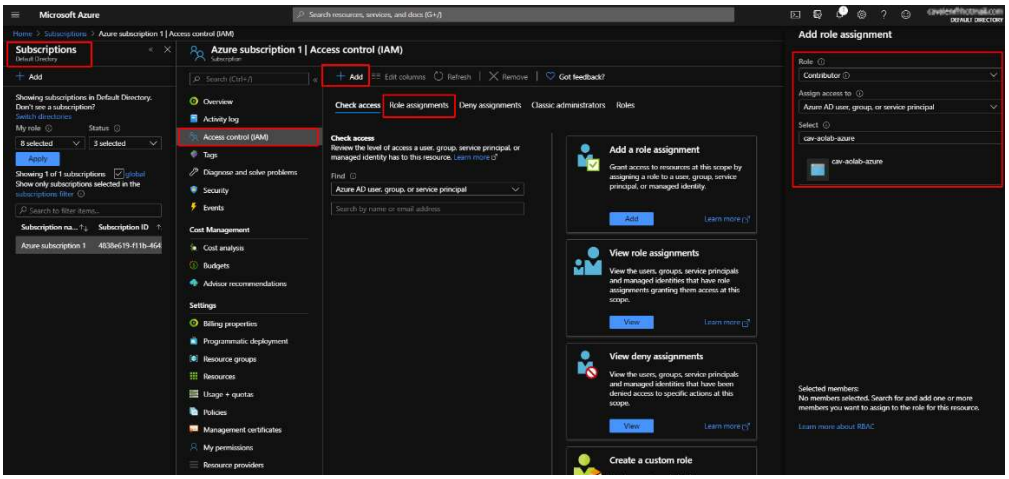


At this point you should have note of 4 important parameters:

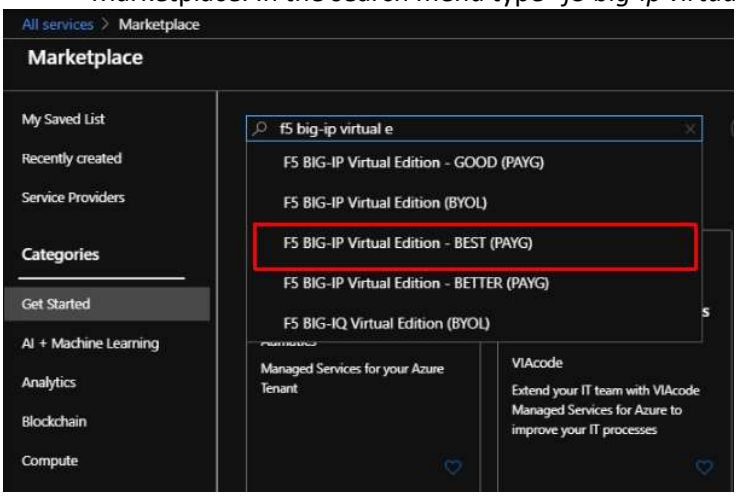
- Application (Client) ID
- Directory (Tenant) ID
- Client Secret
- Subscription ID

9. Now we need to provide proper roles for the application we just registered in order to be able to run the automation scripts. From the left-side menu, select All Services, then Subscriptions as in step 7, then select your subscription ID, and finally click on **Access control (IAM)** on the left-side.

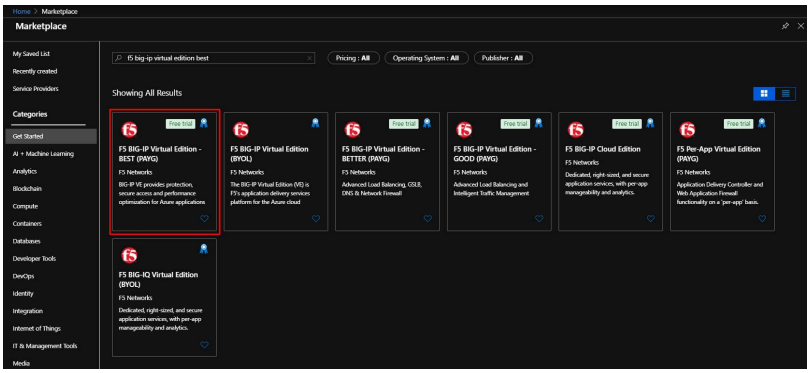
10. Click on **Add**, then select Role Assignments, then select "Contributor" in the Role drop down menu in the right side. Type in the Application name created in step 4, select your Application and when done click on **Save**.



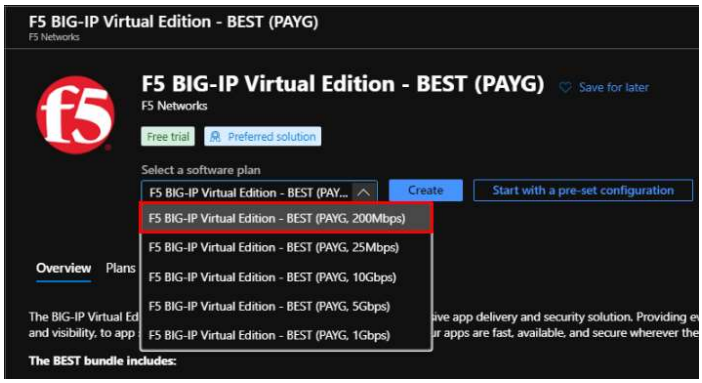
11. Finally, we need to subscribe to the BIG-IP instance in the **Azure Marketplace** in order to be able to automate the BIG-IP VE deployment. From the left-side menu, select All Services, then Marketplace. In the search menu type "*f5 big-ip virtual*"



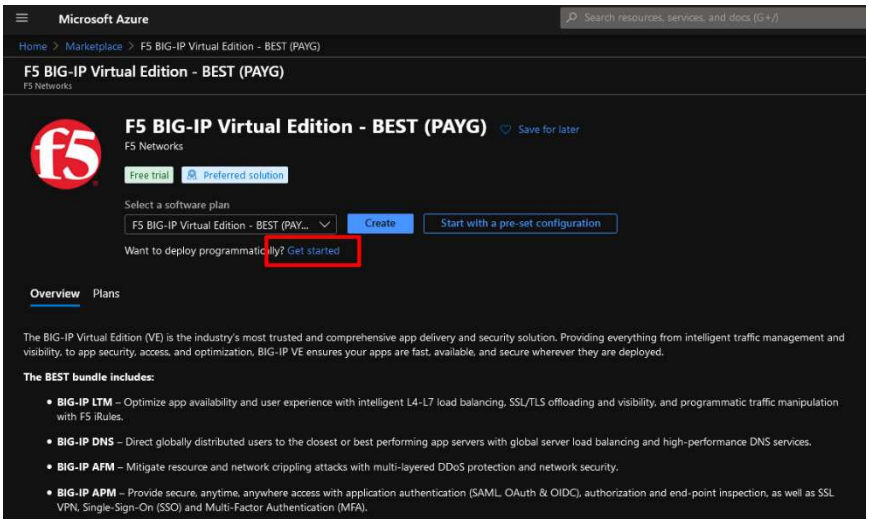
Then select the **F5 BIG-IP Virtual Edition - BEST (PAYG)** from the results.



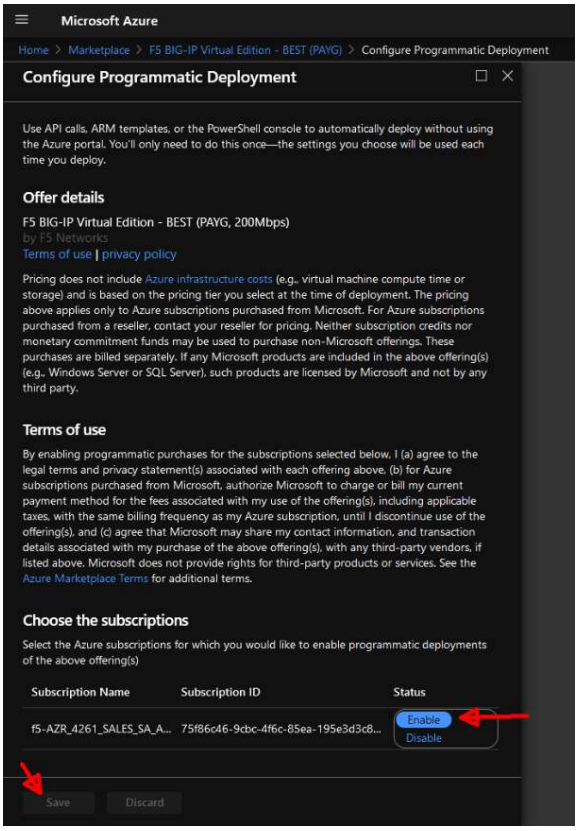
Select **F5 BIG-IP Virtual Edition – BEST (PAYG, 200Mbps)**



Click on **Get Started**



Then finally click on **Enable**, then **Save** and you are ready to go.



Task 2 – Deploy Azure Infrastructure

To deploy the Lab Infrastructure, please follow the instructions in the GitHub repository and continue to the next part.

https://github.com/cavalen/aolab-azure

You can use Ansible or Terraform to deploy the lab's infrastructure. Run one of the tools and wait until you see output with the information we need.

Ansible Output

```
ubuntu@46f2b37669d9:~/aolab-azure/deploy-lab$ cat info.txt

 Lamp Server IP: 52.165.155.198
 Lamp Server DNS: lampserver-cav.centralus.cloudapp.azure.com
 BIG-IP Mgmt IP: 52.143.249.207
 BIG-IP Mgmt DNS: https://bigiplab-cav.centralus.cloudapp.azure.com
 BIG-IP (Virtual Server) IP: 52.143.251.157
 BIG-IP (Virtual Server) DNS: bigiplab-cav0.centralus.cloudapp.azure.com

 Lamp user: f5student/f5DEMOs4uLATAM
 BIG-IP user: azureuser/f5DEMOs4uLATAM

ubuntu@46f2b37669d9:~/aolab-azure/deploy-lab$
```

Terraform Output

```
Apply complete! Resources: 24 added, 0 changed, 0 destroyed.

Outputs:

BIG-IP_Private_VIP = 10.1.10.245
BIG-IP_Public_VIP = 20.62.158.50
BIG-IP_mgmt_fqdn = f5-student.eastus.cloudapp.azure.com
BIG-IP_mgmt_private_ip = 10.1.1.245
BIG-IP_mgmt_public_ip = 20.62.158.151
BIG-IP_password = f5DEMOs4uLATAM
BIG-IP_username = azureuser
Resource_Group_name = aolab-student
Security_Group_name = bigip-student-nsg
Server_IP = 13.90.36.53
Your_Public_IP = [redacted]/32
```

Take note of this information, you will need this to get to the BIG-IP GUI and test services.

3. F5 Automation and Orchestration Toolchain Lab

The **F5 Automation Toolchain** is a product family that brings together our Automation & Orchestration components under one umbrella.

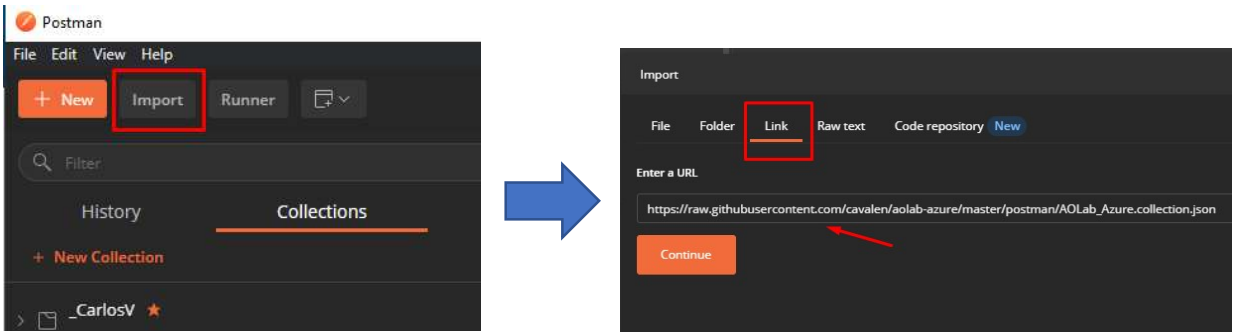
The F5 Automation Toolchain is made up of composable components that allow customers and partners to easily integrate F5 platforms into automation pipelines using modern automation patterns.

These components include the following:

- Application Services 3 Extension (AS3): Declarative L4-L7 BIG-IP Application Services
- Declarative Onboarding Extension (DO): Declarative L1-L3 BIG-IP Onboarding
- Telemetry Streaming Extension (TS): Automated BIG-IP Telemetry Streaming to Analytics Systems

Task - Import Postman Collection and Environment

If you have not done this already, import the Postman Collection and Environment from the URLs:



You need to do this twice, first for the collection, next for the environment

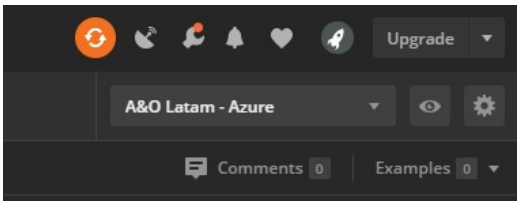
Environment: ***AO_Latam_Azure.environment.json***

https://raw.githubusercontent.com/cavalen/aolab-azure/master/postman/AOLab_Azure.collection.json

Collection: ***AO_Latam_Azure.collection.json***

https://raw.githubusercontent.com/cavalen/aolab-azure/master/postman/AOLab_Azure.environment.json

Note: Make sure the "**A&O Latam – Azure**" Environment is selected in upper-right section of Postman before proceeding.



Important Note when working with JSON Declarations (AS3/DO/TS):

> While a dash/hyphen (-) is a valid character for naming objects directly on a BIG-IP system, **it is not a valid character** in a declaration object name. Likewise, a dot (.) is not a valid character in a declaration object name. Underscore (_) is a valid character.

Lab 1 - Authentication and Initial Settings using REST.

This lab will explore the basics of setting BIG-IP parameters and configuration using REST API calls. These are standard API calls, no AS3 declarations.

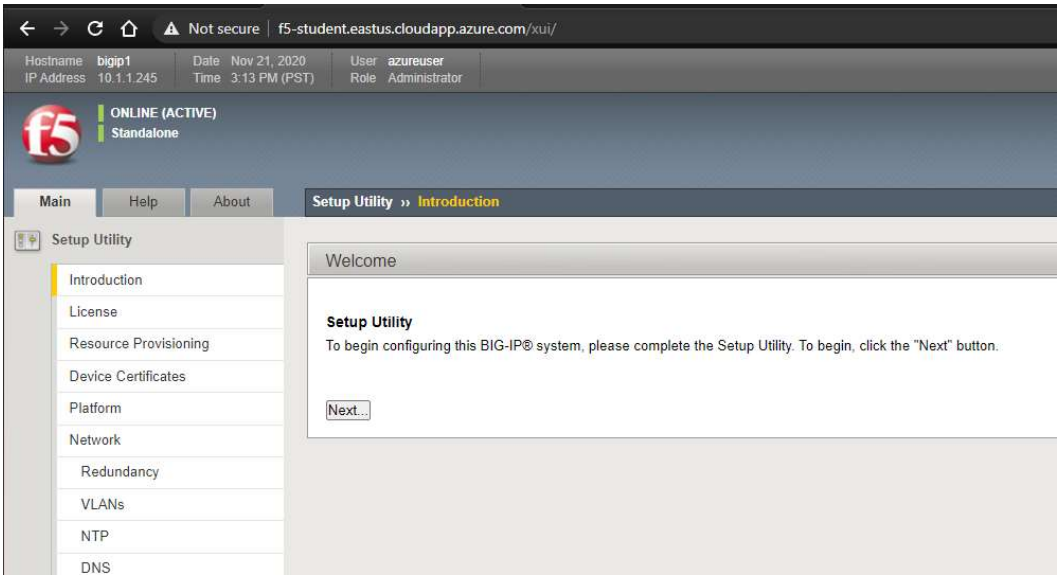
- Step 1 is used to provide Token Based authentication. Check the URL, and Body Tabs. Username and Password variables are configured in Postman Environment and the Token is set during the "Tests" phase.
- Step 2 edit the token's timeout using a PATCH method. Examine the REST endpoints, and the "Body" tab.
- Step 3 is used to validate TS/DO/TS are installed on the BIG-IP.

Note: Any time you need to interact with the BIG-IP using REST API calls during this lab, make sure that to first run steps 1 and 2, to obtain an authentication Token. (or in case you get a 401 Error from the BIG-IP)

Lab 2 – Declarative Onboarding (DO)

F5 Declarative onboarding uses a declarative model to initially configure a BIG-IP device with all of the required settings to get up and running. This includes system settings such as licensing and provisioning, network settings such as VLANs and Self IPs, and clustering settings if you are using more than one BIG-IP system.

If you go to your BIG-IP Setup Utility, you will see it is un-provisioned.



Open Postman, go to Lab 2 Folder

- Step 1: Validate DO is installed, there should be an empty response from the BIG-IP ([ ]) but it is ok if you see something that is not an error
- Step 2: Here we send a JSON Declaration, specifying settings like hostname, DNS, NTP, VLANs, SelfIP, Users, Provisioning, DB Values and Routes. Review the Body, the different sections

  Send the request (Response should be a "*202 Accepted*" with the full declaration)

- Step 3: Get previous declaration and status.
  Send the request. If DO is still running you should see a 202 Status Code.
  Refresh until you get a 200 OK Code.

"result": { "class": "Result", "code": 202, "status": "RUNNING", "message": "processing" },	"result": { "class": "Result", "code": 200, "status": "OK", "message": "success" },

  Move to the BIG-IP GUI and validate it is now configured (Check provisioning, SELF-IPs)

Lab 3 – Application Services 3 (AS3).

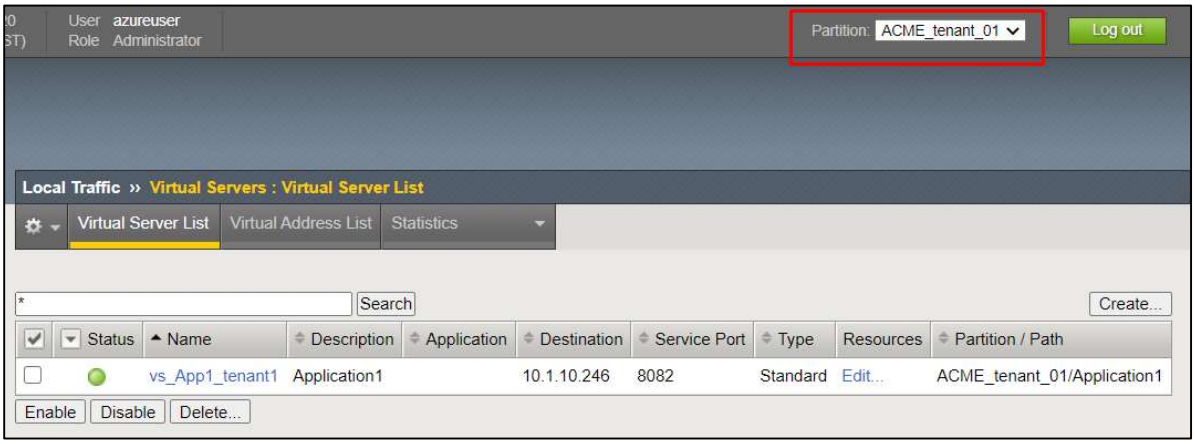
AS3 is installed by default in BIG-IPs deployed in Azure.

- Step 1: Get the AS3 version information from the REST API. You should get a 200 OK response like this:

```
{
    "version": "3.23.0",
    "release": "5",
    "schemaCurrent": "3.23.0",
    "schemaMinimum": "3.0.0"
}
```


- Step 2a: In this step we create a Simple HTTP Application in a Tenant "**ACME_cloud_01**" using a "generic" template. This will lead to the creation of a Virtual Server along with some defaults for the service

  Review the declaration parts, including the Tenant, the Application class, the service, and pool definitions.



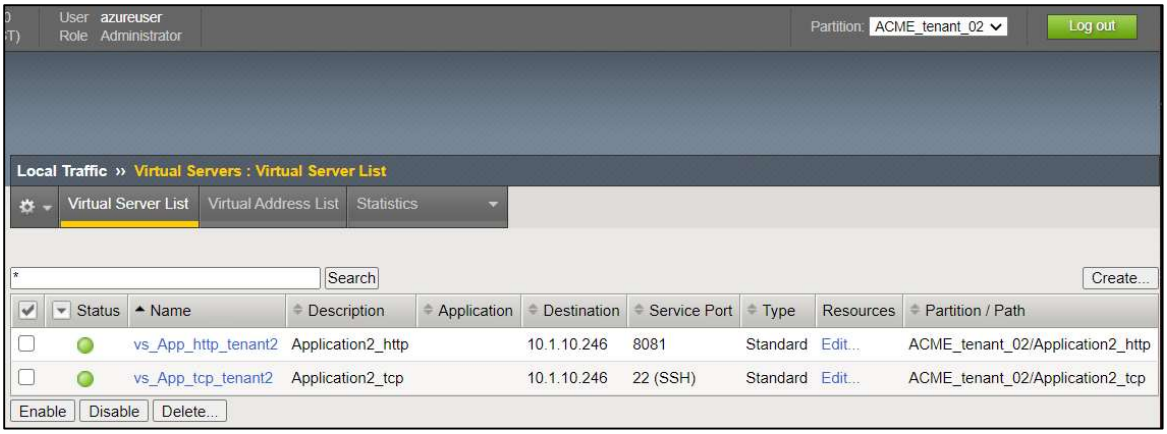
  After testing the service, review and send Step 2b (PATCH command) to update the current declaration and add a new pool member.

  Finally, in Step 2c, delete ACME_tenant_01 sending the "Remove Tenant" API Call

  Note that there are many ways to delete AS3 configurations, one is to send a declaration with an "empty" Tenant config (as we just did) and another way is using the DELETE verb (we will do this in the last step)

- Step 3a: In this step we send an AS3 declaration using a Generic template to create two apps (HTTP and TCP) in a Tenant "**ACME_tenant_02**"

  Review the declaration and send the configuration to the BIG-IP. You should get a 200 OK response. Validate the configuration and test the application browsing the Public VIP IP address and port

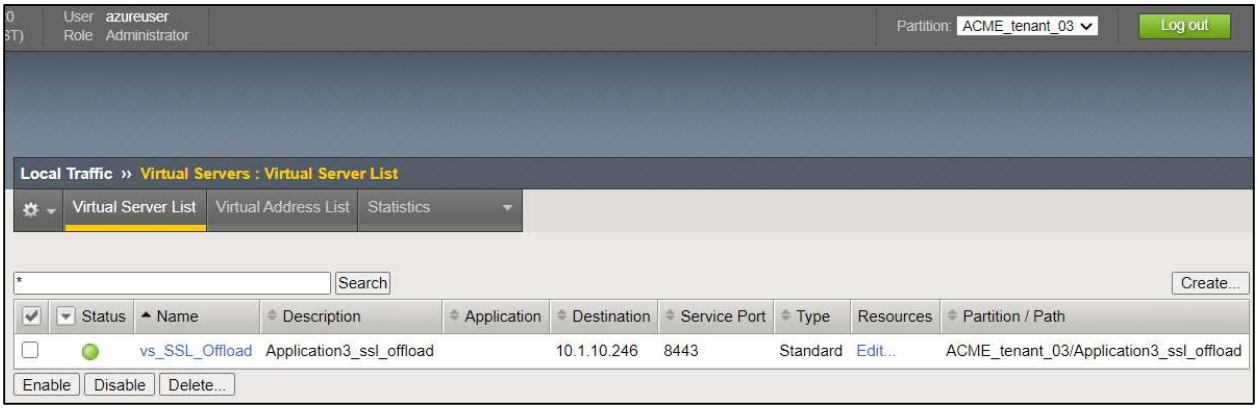


- Step 3b: Delete ACME_tenant_02 by sending the "Remove Tenant" API Call.

- Step 4a: This step creates a HTTPS application with a ClientSSL profiles (SSL Offload) with an existing SSL certificate. No encryption between the BIG-IP and the pool member.

  In the declaration, review the "**frontside**" and "**webcert**" statements

  This should result in a 200 OK status and a new Tenant "**ACME_tenant_03**". Test this HTTPS application using the browser.

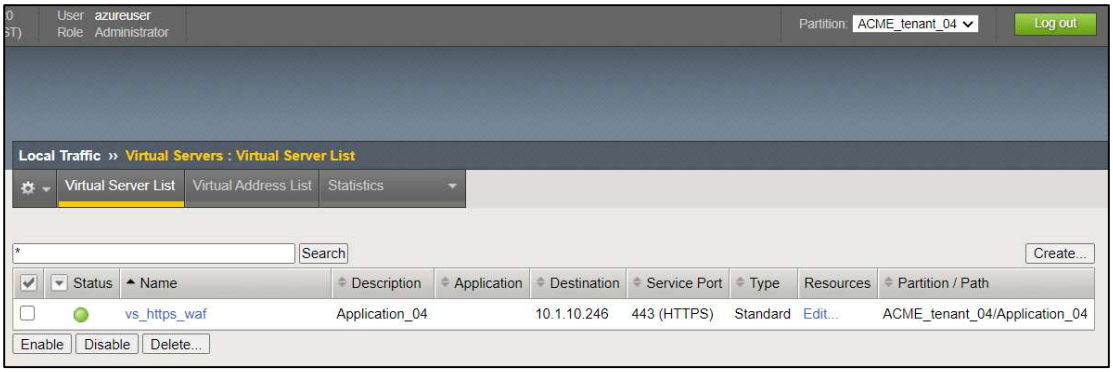


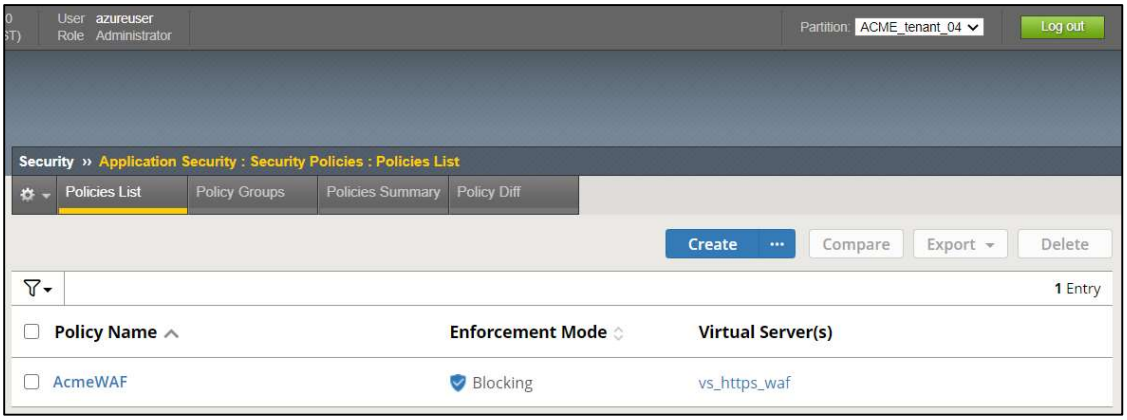
- Step 4b: After reviewing the declaration and the application, delete ACME_tenant_03 Tenant by sending the "Remove Tenant" API Call

- Step 5a: This step creates an HTTPS Application, an SSL certificate embedded in the declaration and a Declarative WAF policy imported from a JSON file in a remote repository (e.g., GitHub).

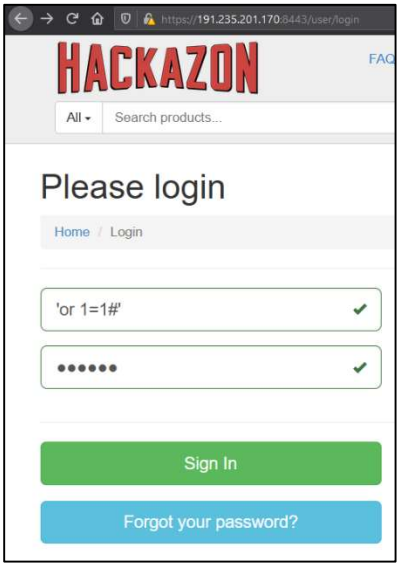
  Following the philosophy of API first, F5 BIG-IP WAF can be configured using a Declarative approach, using JSON files that can be easily integrated into a CI/CD Pipeline. Same a AS3.
  Review the declarative policy. It includes WAF configurations as Server Technologies, Blocking Settings, Data Guard, Custom Response Pages, Webhooks, Login Page, Brute-force Prevention and others.

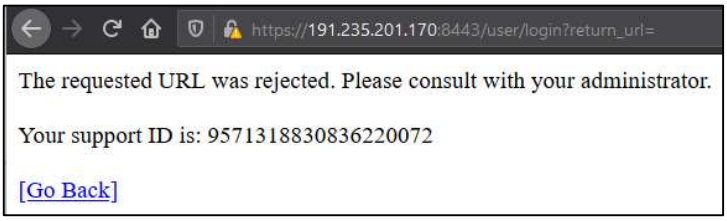
  Validate the BIG-IP configuration, The Virtual Server and the ASM Policy:



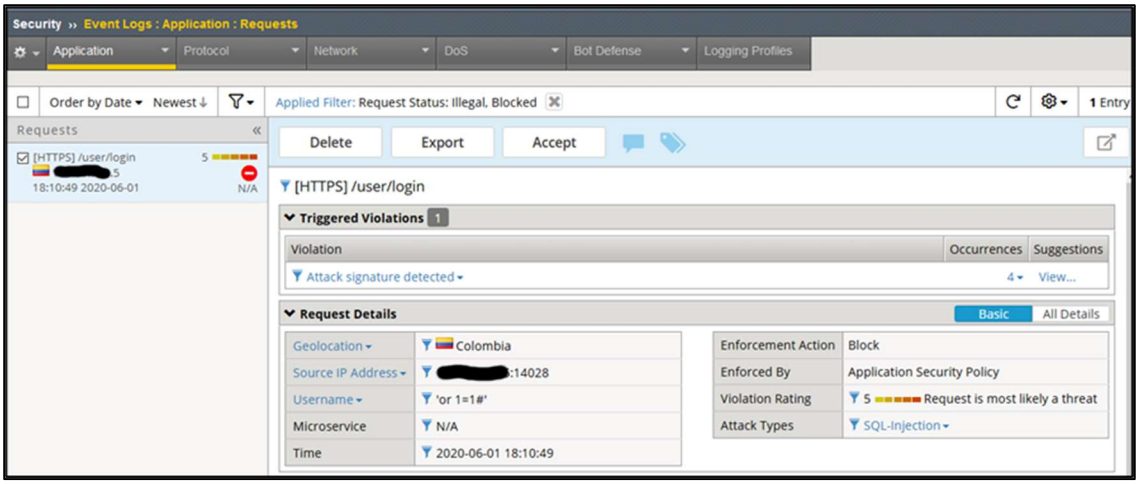


Now, using your browser go to https://bigipvip.public.ip/user/login and try to do a SQL injection in the username field.



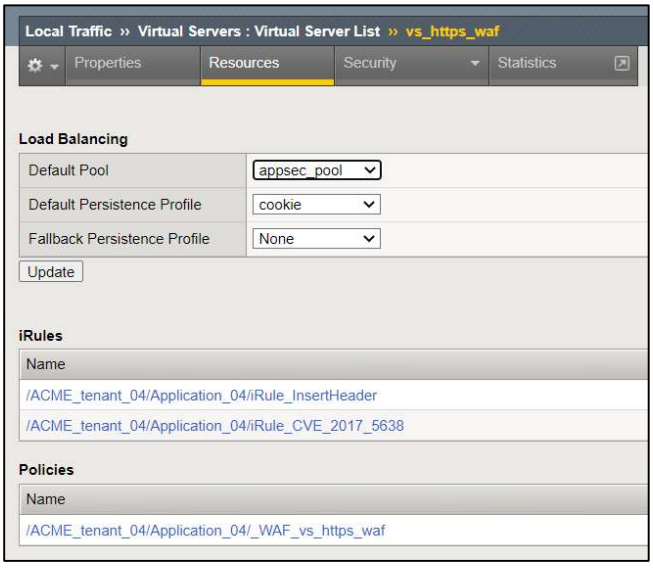


Then review the security logs in the BIG-IP:



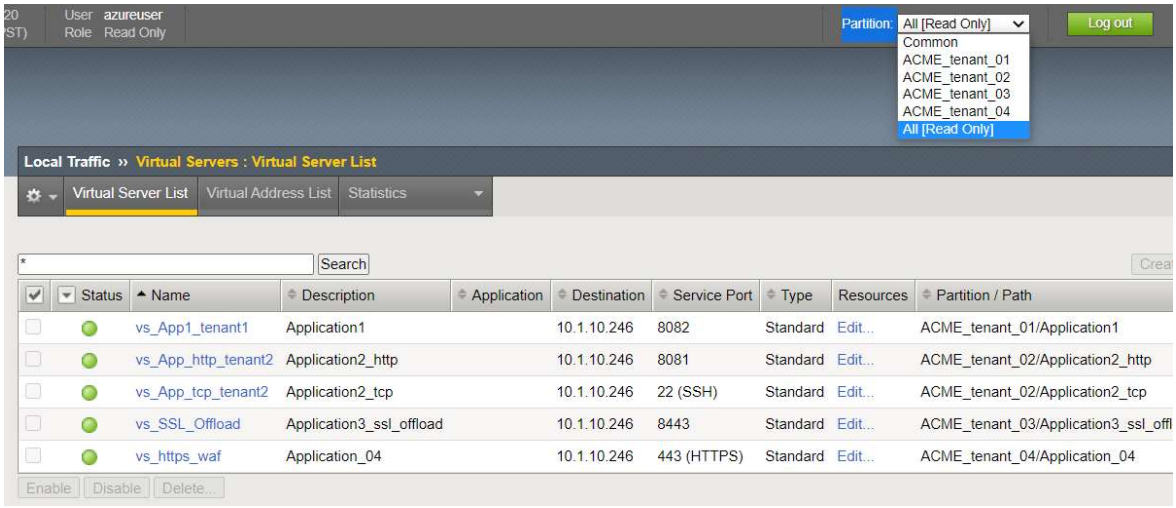
- Step 5b: Here we will update the Declaration from the previous step to add two iRules. The first one is defined in the Declaration and the second imported from a remote repository.

```
"iRule_InsertHeader": {
    "class": "iRule",
    "remark": "Insert headers to response",
    "iRule": "when HTTP_RESPONSE {\nHTTP::header insert X-CustomHeader [IP::client_addr]\n HTTP::header replace Server GOSMARTER\n}"
},
"iRule_CVE_2017_5638": {
    "class": "iRule",
    "iRule": {
        "url": "https://raw.githubusercontent.com/cavalen/aolab-azure/master/files/CVE-2017-5638-ApacheStruts2.irule"
    }
},
```



- Step 5c: Remove ACME_tenant_04

- Step 6: Now let's create all previous tenants at one in big declaration. Review the different part of the Body in Postman and click Send



- Step 7: We can query the AS3 API to retrieve the latest declarative configuration from the BIG-IP using a GET command to the API entry point.
  Due to the declarative nature of AS3, any manual changes done to the BIG-IP will not be reflected here, because AS3 is now the "owner" (or the Source of Truth) of the configuration and manual modifications should be avoided. Manual configurations and automation is not a good mix

- Step 8: Finally, delete all AS3 configurations using a DELETE statement to AS3 API Endpoint.

Lab 4 – Telemetry Streaming (TS).

In this lab using a declarative approach, we will create a simple HTTP application and for every client-side request the BIG-IP will generate a log to a Kafka consumer.

Telemetry Streaming is not installed by default, but is a simple task:

- Get the latest version from here:
  https://github.com/F5Networks/f5-telemetry-streaming/releases
- Follow the installation instructions from:
  https://clouddocs.f5.com/products/extensions/f5-telemetry-streaming/latest/installation.html

- Step 1:
  After the installation, we need to validate if the TS service is running. You should see something simiar to this:

```
{
    "nodeVersion": "v8.11.1",
    "version": "1.15.0",
    "release": "4",
    "schemaCurrent": "1.15.0",
    "schemaMinimum": "0.9.0"
}
```

- Step 2:

Before running this step, do the following:

- Open an SSH window to the Ubuntu Server (Check the public IP in Azure Console or with the Automation tool)
  user = azureuser
  password = f5DEMOs4uLATAM

```
cd /home/azureuser/kafka-docker/
sudo docker-compose up -d
sudo ./start-kafka-shell.sh 10.1.10.80 10.1.10.80:2181
kafka-console-consumer.sh --bootstrap-server 10.1.10.80:9092 --topic f5-telemetry --from-beginning
```

**** The last command runs inside the Kafka container**

⚠ Leave this window open to review after we generate some traffic to a Web application.

This is the actual TS declaration, and is composed of several parts:

A System Poller: The system poller collects and normalizes statistics from a system, such as BIG-IP, on a configurable interval.

```
"My_Poller": {
    "class": "Telemetry_System_Poller",
    "interval": 60
},
```

A Listener: The listener is configured in our BIG-IP in the port specified. It is not a Virtual Server.

```
"My_Listener": {
    "class": "Telemetry_Listener",
    "port": 6514
},
,
```

A Consumer: This is the Third party consuming our telemetry. In our case we will be using Kafka. There are several consumers like S3, Splunk, CloudWatch, Azure Log Analytics, ElasticSearch and others.

```
"My_Consumer": {
    "class": "Telemetry_Consumer",
    "type": "Kafka",
    "host": "10.1.10.80",
    "protocol": "binaryTcp",
    "port": 9092,
    "topic": "f5-telemetry"
}
```

- Step 3: Here we will deploy a Web Application with a “Request Logging” profile, using an HSL pool whose destination is the BIG-IP listener created in the previous step.
  Browse the application to http://f5vip.public.ip:8080 , generate some traffic and you should see messages in the Kafka console.

- Step 4: Delete TS configuration. To delete Telemetry Streaming configuration, we need to send a POST to TS API endpoint with an empty configuration.
  Review the request body and send.

  Fin.