

F5 Automation Toolchain Workshop

Contents

F5 Automation Toolchain Workshop	1
Getting Started	2
Starting with UDF	2
Starting the lab without UDF.....	2
Module 1 – Starting the lab environment.....	3
Task 1.1 Explore and start the lab environment	4
Task 1.2 Setup Postman	9
Task 1.3 – Authentication Tokens	10
Module 2 – Declarative Onboarding	11
Task 2.1 – Deploy Onboarding	11
Module 3 – Application Services 3 Extension (AS3)	15
Task 3.1 – HTTP Service with Service Discovery.....	15
Task 3.2 – Service HTTPS using HTTP to HTTPS redirect	19
Task 3.3 – ServiceMain and Generic.....	22
Task 3.4 – Use of Global Server Load Balancing (GSLB)	26
Module 4 – F5 Application Services Templates.....	33
Task 4.1 – Explore FAST via the GUI	33
Task 4.2 – Use FAST via the API.....	35
Module 5 – Telemetry Streaming (TS).....	38
Task 5.1 – Azure Sentinel and TS.....	39
Task 5.2 – Include WAF via an AS3 declaration.....	42
Task 5.3 – Azure Sentinel.....	46

Getting Started

This workshop is intended to give more inside in the use of F5 Automation Toolchain and uses Terraform to deploy infrastructure, BIG-IPs and backends automatically in Azure public cloud.

F5 Automation Toolchain delivers that a BIG-IP can be turned into code and deployed in a declarative and automated way to provide L4- L7 application services. When you are not familiar with F5 automation Toolchain, please go to: <https://clouddocs.f5.com/> and start reading.

Those who already have done the introduction A&O training are invited to use this workshop to gain more understanding of the entire toolchain F5 has developed.

Starting with UDF

When you are an F5 employee, select the “F5 EMEA Automation Toolchain Workshop” from the UDF Blueprint section and start deploying.

When you are invited by F5 into this workshop, you will get invited by email and through UDF classroom will be able to launch the blueprint.

The student will find that a resource_group ‘student1-f5-atc-workshop’ and location ‘West Europe’ is already configured in Azure.

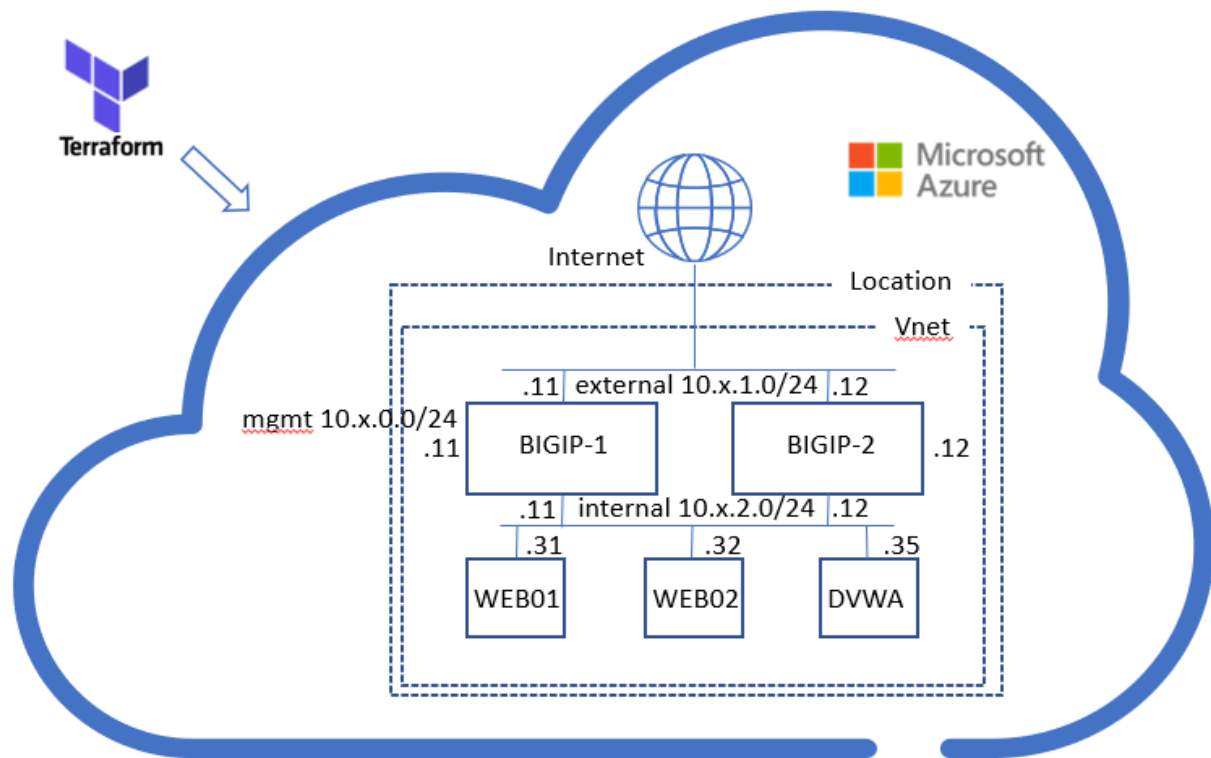
During the workshop which uses UDF, this is done for you. Once getting invited this resource group is there with the right location. Terraform will create its infrastructure here.

Starting the lab without UDF

When you don't have access to UDF, please follow the mentioned per-requisites:

- Azure account
- Configure a resource group
- Define a location

In the setup_changeme.yml the resource group and location are hardcoded and should be adjusted accordingly.



Device	Creds	IP Adresses
BIGIP-1	azureuser/F5emea2020! (ssh/https)	10.x.0.11 (mgmt) 10.x.1.11 (external) 10.x.2.11 (internal)
BIGIP-2	azureuser/F5emea2020! (ssh/https)	10.x.0.12 (mgmt) 10.x.1.12 (external) 10.x.2.12 (internal)
BIGIP-VIP1		10.x.1.20 (Virtual Server)
BIGIP-VIP2		10.x.1.30 (Virtual Server)
Web01	azureuser/F5emea2020! (ssh/https)	10.x.2.31
Web02	azureuser/F5emea2020! (ssh/https)	10.x.2.32
DVWA	azureuser/F5emea2020! (ssh/https)	10.x.2.35
Jumphost	F5student/f5student123 (ssh/rdp)	10.1.1.5 (UDF)

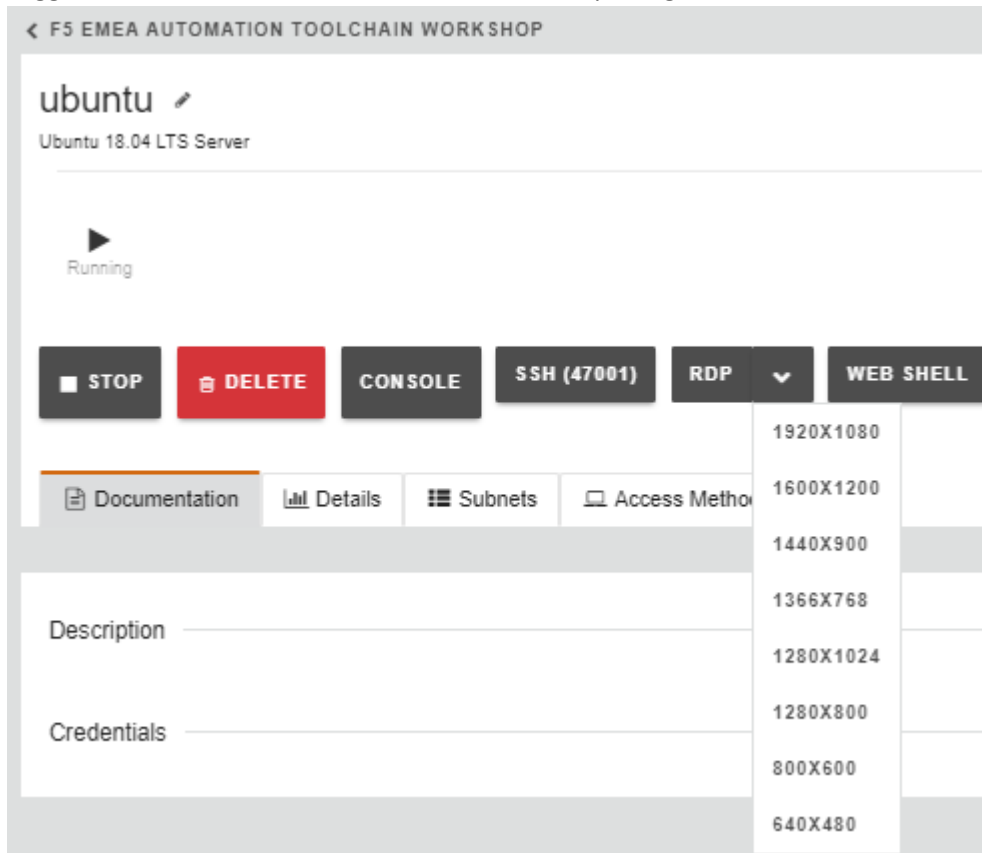
Module 1 – Starting the lab environment

In this module the environment will be set ready to deploy the pre-defined infrastructure and setup the used tools correctly to have a smooth lab experience. This module will cover:

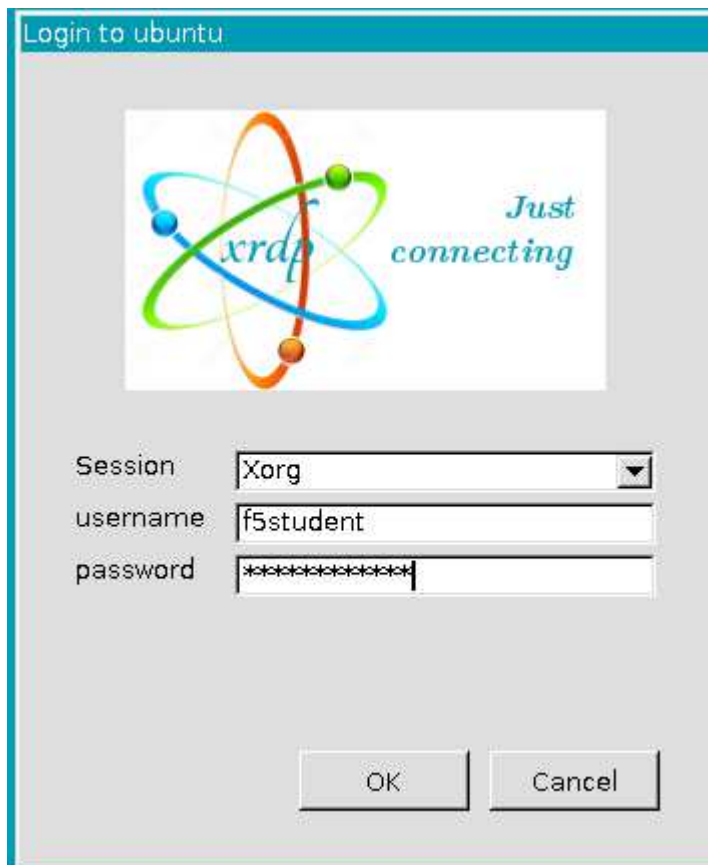
- Remote login the Jumphost
- Clone the relevant files from Github
- Make Visual Studio Code ready to use
- Deploy the infrastructure
- Make Postman ready to be used
- Create authentication tokens


Task 1.1 Explore and start the lab environment

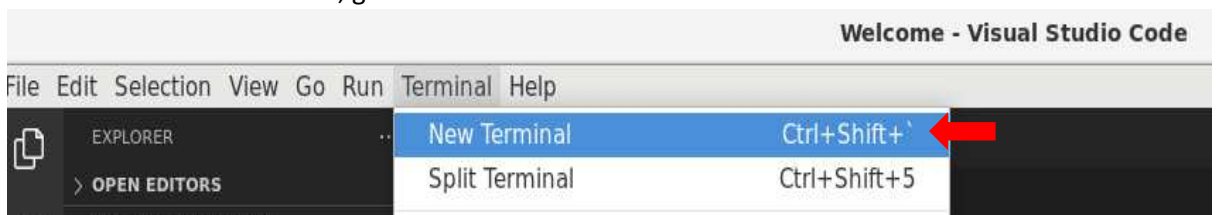
1. From UDF, start an RDP session to the Jumphost. For modern laptops it might be a suggestion to use the 1440x900 resolution to keep things readable in the remote session.



2. In the Remote Desktop Connection popup select 'Connect'. Ignore the warning that the identity could not be verified and click 'Yes'.
3. Login to Ubuntu with f5student/f5student123.



4. At the desktop, select 'Activities' in the upper left corner and select  from the menu on the left. This will start Visual Studio Code.
5. When no Terminal is shown, go to 'Terminal' and select 'new Terminal'.



This will open a Terminal window in the lower section of VSC.

6. In this terminal, which is bash based type:

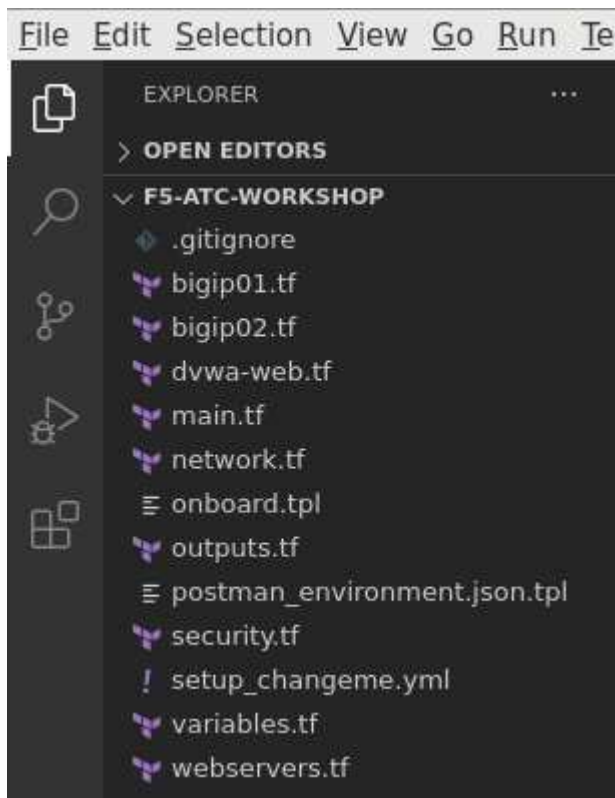
git clone <https://github.com/gwolfis/f5-atc-workshop.git>

This grabs the files we will going to use for the lab from Github and places them on the Jumphost.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

f5student@ubuntu:~$ git clone https://github.com/gwolfis/f5-atc-workshop.git
Cloning into 'f5-atc-workshop'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 35 (delta 19), reused 29 (delta 13), pack-reused 0
Unpacking objects: 100% (35/35), done.
f5student@ubuntu:~$
```

7. Let's open the folder in VSC, by selecting **Open Folder** in the left pane and in the Home directory select 'f5-atc-workshop' and click 'OK'.



The folder contains all the files to fire up the infrastructure in Azure public cloud by using terraform.

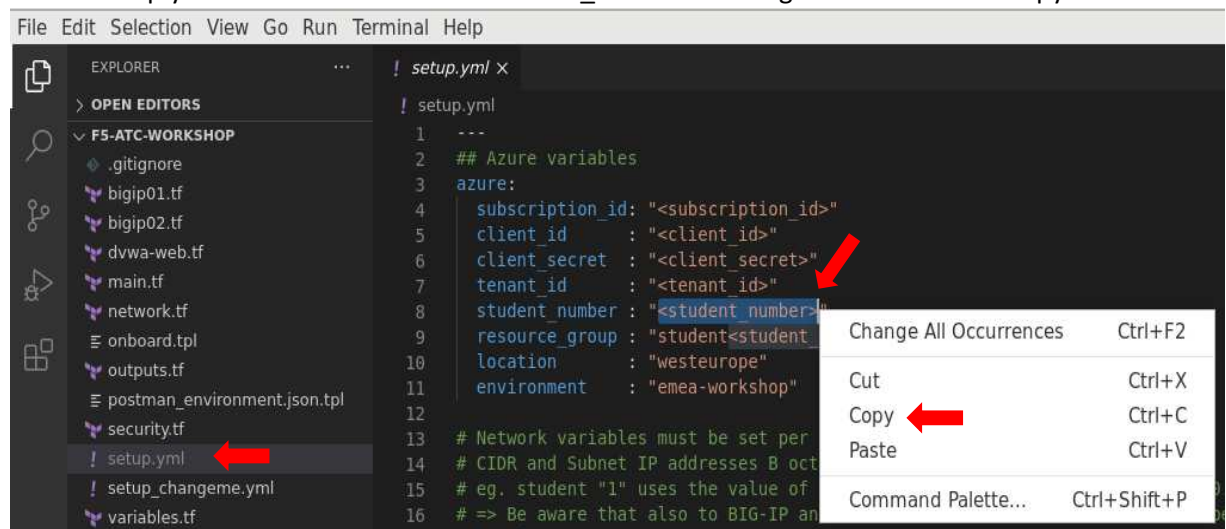
This lab does not contain a deep dive into Terraform since this is out of scope for the purpose of this workshop. The reason for showing these files and deliver guidance to leverage Terraform for deploying the lab infrastructure comes with the idea that now one can deploy this lab everywhere, even in its own Azure environment.

If you don't have Terraform skills, no worries, this guide will lead you through.

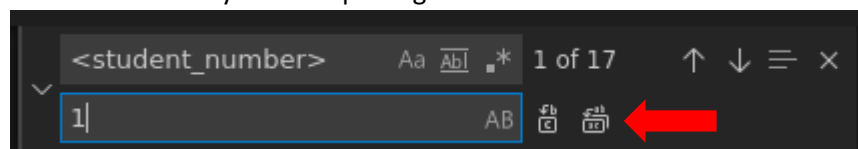
8. In the terminal, be sure you are in the directory 'f5-atc-workshop' and type the following and press enter:

cp setup_changeme.yml setup.yml

9. Select 'setup.yml' and in the file select <student_number> and right-click and select 'Copy'.



10. Type 'Ctrl+H' to search and replace the label into your actual student number. This has been assigned to you when being in the workshop. The picture shows '1', but this **most** be your assigned student number! And click to replace it either at once or piece by piece to understand what you are replacing.



11. As a result, you should see that everywhere the label <student_number> is replaced by '1'. This counts for the student_number, and all B-octets of defined IP ranges and IP host addresses.
12. Close the search and replace window and save the changes by pressing 'Ctrl+S'. Not saving the changes will make terraform fail!
13. In the terminal we first need to navigate to the right directory: **cd f5-atc-workshop**
14. In the terminal type: **terraform init**

```
To prevent automatic upgrades to new major versions that may contain breaking changes, we recommend adding version constraints in a required_providers block in your configuration, with the constraint strings suggested below.
```

```
* hashicorp/template: version = "~> 2.1.2"
```

```
Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.
```

```
If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
```

```
f5student@ubuntu:~/f5-atc-workshop$
```

You should see this output.

15. Next, type: **terraform plan**

16. Finally type: **terraform apply -auto-approve**

Terraform will take about ~5min to deploy the infrastructure. Time for coffee!

```
Apply complete! Resources: 41 added, 0 changed, 0 destroyed.
```

```
Outputs:
```

```
azure_resource_group = student1-f5-atc-workshop
```

```
bigip_1_ext_selfip_privip = 10.1.1.11
```

```
bigip_1_ext_selfip_pubip = 20.56.50.166
```

```
bigip_1_int_selfip_privip = 10.1.2.11
```

```
bigip_1_mgmt_privip = 10.1.0.11
```

```
bigip_1_mgmt_pubip = 20.56.50.176
```

```
bigip_2_ext_selfip_privip = 10.1.1.12
```

```
bigip_2_ext_selfip_pubip = 20.50.198.122
```

```
bigip_2_int_selfip_privip = 10.1.2.12
```

```
bigip_2_mgmt_privip = 10.1.0.12
```

```
bigip_2_mgmt_pubip = 20.50.198.120
```

```
bigip_ext_priv_vippip1 = 10.1.1.20
```

```
bigip_ext_priv_vippip2 = 10.1.1.30
```

```
bigip_ext_pub_vippip1 = 20.56.50.167
```

```
bigip_ext_pub_vippip2 = 20.50.198.121
```

```
dvwa_int_selfip_privip = 10.1.2.35
```

```
web_1_int_selfip_privip = 10.1.2.31
```

```
web_2_int_selfip_privip = 10.1.2.32
```

```
f5student@ubuntu:~/f5-atc-workshop$
```

When all is deployed, you should see this output.

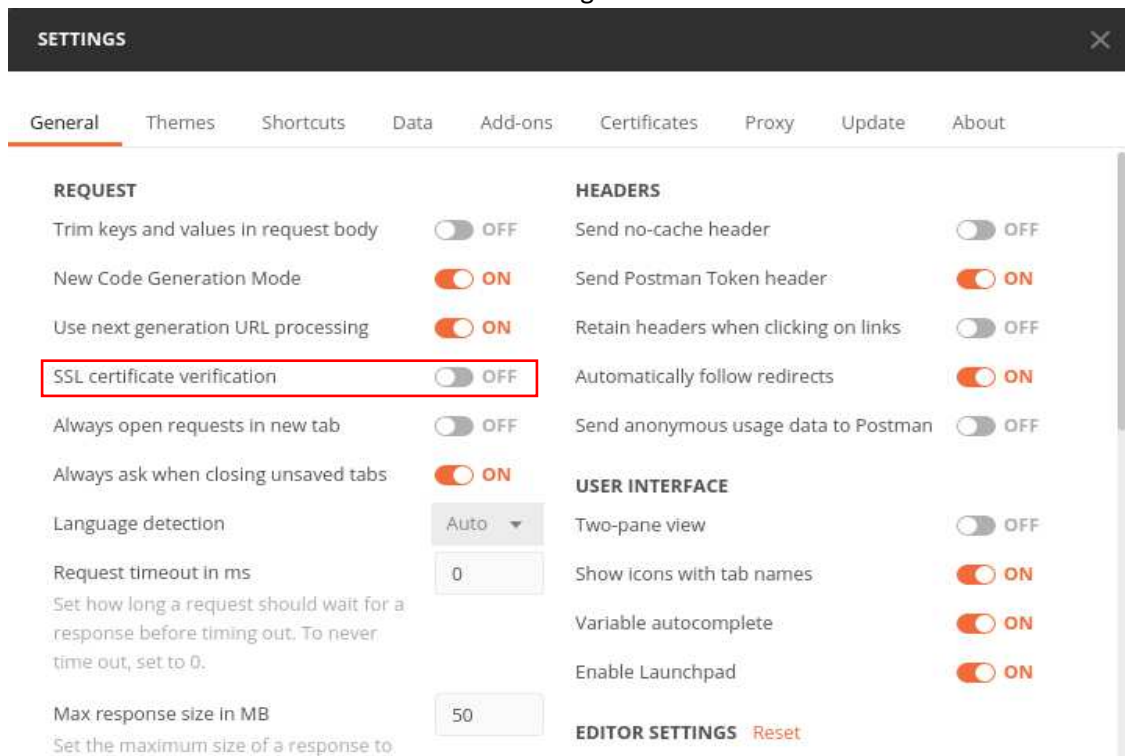
17. Use your Azure credentials to open the Azure portal and select your resource group referenced by your student number. You should see all the deployed infrastructure objects.

Task 1.2 Setup Postman

This task will make Postman ready for being used with this lab.

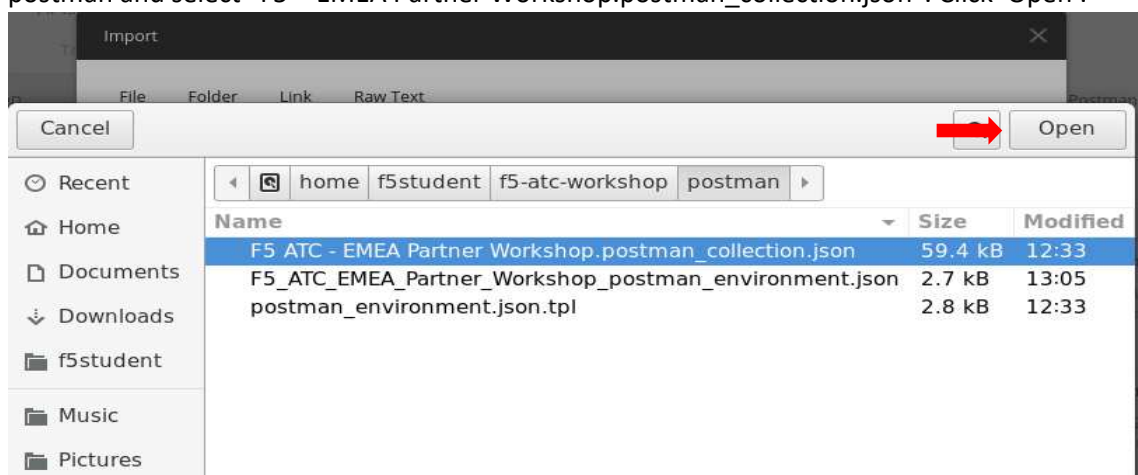


1. On the Jumphost, go to the Activities pane on the left and select Postman
2. Once Postman has start up, you can ignore the login screen and go to the bottom of the page and select “Skip signing in and take me straight to the app”.
3. Welcome to Postman! Select file and click settings and turn off “SSL certificate verification”.



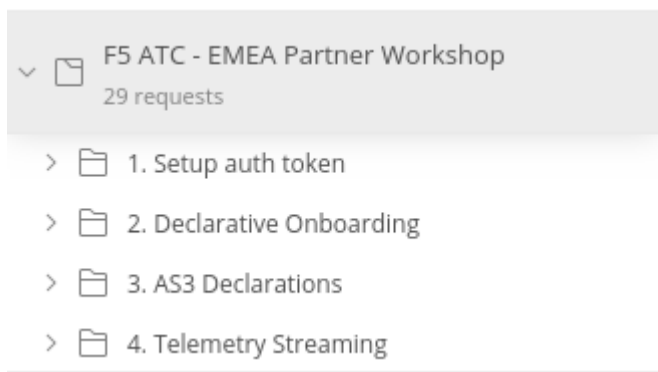
Next, we are going to import the Postman collection and environment files.

4. In Postman, select ‘Import’ from file and in the Home directory select: f5-atc-workshop > postman and select “F5 – EMEA Partner Workshop.postman_collection.json”. Click ‘Open’.

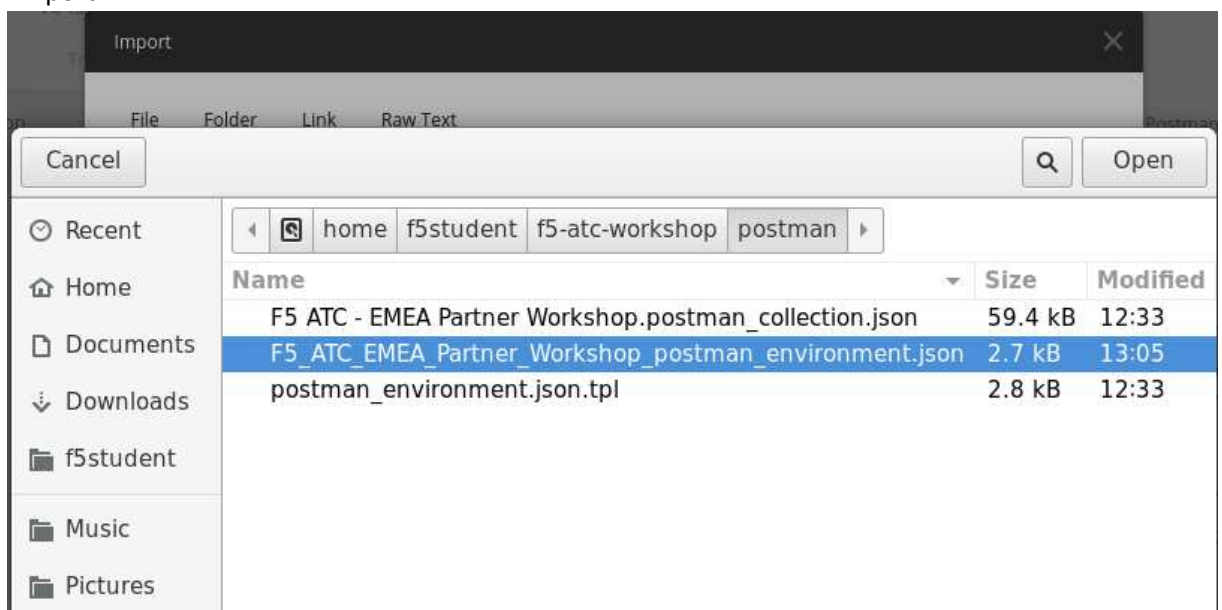



Next, hit ‘Import’.

In Postman select ‘Collections’ and open “F5 ATC -EMEA Partner Workshop”.



5. Now we need to set the Postman environment for this lab. So, once more click 'Import' from file and in the same postman directory and now select "F5_ATC_EMEA_Partner_Workshop_postman_environment.json" and hit 'Open' and 'Import'.

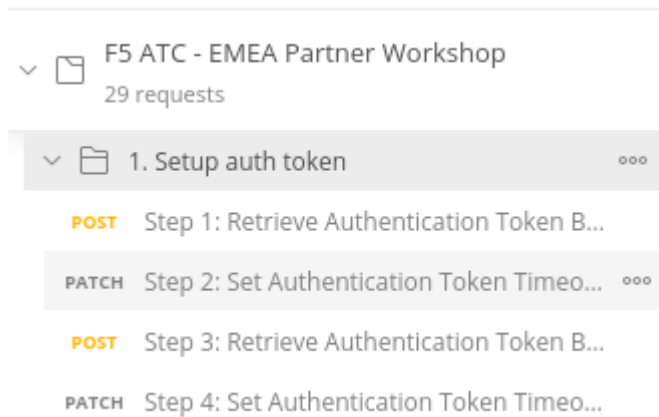


6. In Postman, go to the environment section in the top right and use the drop-down menu to switch from "No Environment" to "F5 ATC – EMEA Partner Workshop". Be aware that during the deployment of the infrastructure, this file has been auto-generated by Terraform.
7. Use the 'eye' icon  to watch the configured variables.

Postman is ready to be used within our lab.

Task 1.3 – Authentication Tokens

1. In Postman, open the collection and select "1. Setup auth token"



2. Select “Step 1: Retrieve Authentication Token BIGIP1” and check the Header and Body. As you can see in the Body, you don’t need to enter a username and password, these are part of the environment variables. Hit ‘Send’ to Post the declaration.
The authentication Token will get retrieved and via the Postman Test and put in the according variable in the environment.
3. Select “Step 2: Set Authentication Token Timeout BIGIP1” to Patch the token timer and click ‘Send’.
4. Repeat the previous steps for BIG-IP2 by using the defined Postman request in step 3 and 4.

Module 2 – Declarative Onboarding

The BIG-IP has gone through the bootstrap phase and is up and running by leveraging Terraform.

Included in the bootstrap phase are the installation of the F5 ATC packages for DO, AS3, TS and FAST.

With just finishing the bootstrap phase, the BIG-IP is still not aware of its environment. This is where the onboarding becomes essential

Let’s move into the onboarding phase and make the BIG-IP aware of its environment.

For more Declarative Onboarding details, please follow the link:

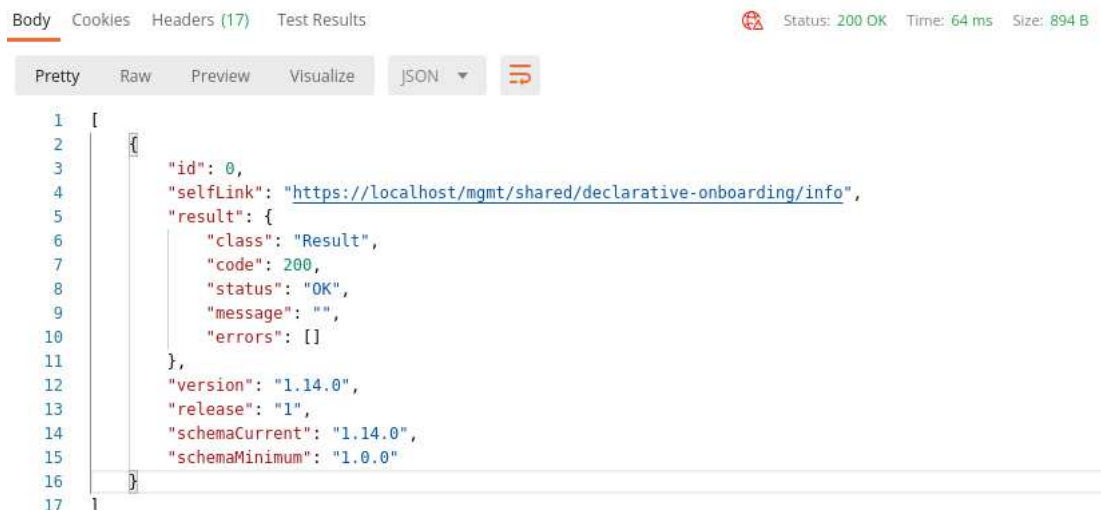
<https://clouddocs.f5.com/products/extensions/f5-declarative-onboarding/latest/>

Task 2.1 – Deploy Onboarding

1. Open Postman, select the F5 ATC collection and click “2. Declarative Onboarding”

First let’s check what is already on the BIG-IP from a DO perspective.

2. The first step is always to verify if DO is installed on the BIG-IP. Select “Step 2.0: GET DO info” and click ‘Send’.



```
1  [
2    {
3      "id": 0,
4      "selfLink": "https://localhost/mgmt/shared/declarative-onboarding/info",
5      "result": {
6        "class": "Result",
7        "code": 200,
8        "status": "OK",
9        "message": "",
10       "errors": []
11      },
12      "version": "1.14.0",
13      "release": "1",
14      "schemaCurrent": "1.14.0",
15      "schemaMinimum": "1.0.0"
16    }
17  ]
```

The BIG-IP returns a 200 OK with the DO version currently installed.

3. In Postman, in the left pane, select “Step 2.1: GET DO declarations”, check the declaration, and click ‘Send’.

When everything is correct you got and “[]” back which is correct since we have not declared an onboarding declaration yet.



```
1  [ ]
```

4. Next, let's select “Step 2.2: Deploy Declarative Onboarding” and check the Body. Check underneath DO json schema and remark to following:
 - It does not declare a license, since we are using PAYG
 - Since we are using TMOS > v14, we need to define a password for the admin user, though the same is kept. Prior v14, this is not needed in a DO declaration.
 - This is a single BIG-IP, so no clustering is defined.
 - Variables are used to define the network items. Check them in the environment.

The JSON schema is printed underneath to have a better view.

```
{
  "schemaVersion": "1.2.0",
  "class": "Device",
  "async": true,
  "label": "Onboard BIG-IP",
  "Common": {
    "class": "Tenant",
    "mySystem": {
      "class": "System",
      "hostname": "bigip01.westeurope.azure.local"
    },
    "myDns": {
      "class": "DNS",
```

```
"nameServers": [
  "8.8.8.8"
],
},
"myNtp": {
  "class": "NTP",
  "servers": [
    "0.pool.ntp.org",
    "1.pool.ntp.org",
    "2.pool.ntp.org"
  ],
  "timezone": "Europe/Amsterdam"
},
"{{bigip_username}}": {
  "class": "User",
  "userType": "regular",
  "password": "{{bigip_password}}",
  "shell": "bash"
},
"myProvisioning": {
  "class": "Provision",
  "itm": "nominal",
  "asm": "nominal"
},
"external": {
  "class": "VLAN",
  "tag": "10",
  "mtu": "1500",
  "interfaces": [
    {
      "name": "1.1",
      "tagged": false
    }
  ]
},
"external-self": {
  "class": "SelfIp",
  "address": "{{bigip_1_ext_selfip_privip}}/24",
  "vlan": "external",
  "allowService": "default",
  "trafficGroup": "traffic-group-local-only"
},
"internal": {
  "class": "VLAN",
  "tag": "20",
  "mtu": "1500",
  "interfaces": [
    {
      "name": "1.2",
      "tagged": false
    }
  ]
}
```

```


    },
    "internal-self": {
      "class": "SelfIp",
      "address": "{{bigip_1_int_selfip_privip}}/24",
      "vlan": "internal",
      "allowService": "default",
      "trafficGroup": "traffic-group-local-only"
    },
    "default-gateway": {
      "class": "Route",
      "gw": "{{default_gateway}}",
      "network": "default",
      "mtu": 1500
    }
  }
}

```

5. Be aware that rendering this json schema will take a minute. Let's check how the declaration is processing by selecting "Step 2.3: GET DO task" and click 'Send'.

It will show that the declaration shows status 'running' with code 202.

Body Cookies Headers (17) Test Results

Pretty Raw Preview Visualize JSON 

```

1  [
2    {
3      "id": "05ab1d9c-73cb-4e43-90f8-51b462dbb9a7",
4      "selfLink": "https://localhost/mgmt/shared/declarat
5      "result": {
6        "class": "Result",
7        "code": 202,
8        "status": "RUNNING",
9        "message": "processing"
10     },

```

Keep refreshing by resending until you will show status 'OK' with code 200, this means that the declaration is processed, and the BIG-IP is ready.

```

{
  "id": "b9e00d25-e578-4cee-ba09-d3304f262551",
  "selfLink": "https://localhost/mgmt/shared/declarative-onboarding/task/b9e00d25-e578-4cee-ba09-d3304f262551",
  "result": {
    "class": "Result",
    "code": 200,
    "status": "OK",
    "message": "success"
  }
},

```

As an alternative you can use the used declaration in step 2.1, which is syntactical the same as the POST-ed declaration of step 2.2, of course with the exception that the used method is GET. Changing the POST into a GET will deliver the same declaration and response.

6. Log into the BIG-IP and check the hostname and network settings as it is declared.
When you forgot the public mgmt IP of BIG-IP, you can check the output in the terminal of Visual Code. When not visible type: **terraform output** to get them refreshed. Select the IP address and use 'Ctrl+C' to copy the IP address for using it in your browser.

Or

Check the environment variables in Postman.

7. For Onboarding BIGIP2, repeat steps 5 and 6, but use the respected Postman declarations defined in step 2.4 and 2.5.

Declarative Onboarding has finished.

Module 3 – Application Services 3 Extension (AS3)

Application Services 3 Extension drives the ability to automate BIG-IP application services via a declarative interface. Application services are all the L4 to L7 application services you can think of like: HTTP, SSL/TPS, URL route, tcp/udp load balancing, WAF, Access, Network/FW/DDoS security and DNS/GSLB.

AS3 supports both BIG-IP hardware and software and delivers declaration of configurations through templates, just like declarative onboarding.

Please check out the link for a more detailed reading:

Task 3.1 – HTTP Service with Service Discovery

In this exercise you will deploy an AS3 HTTP_Service declaration where service-discovery of poolmembers is taking place.

1. Let's check if AS3 has been installed on the BIG-IP by selecting "Step 3.0: Deploy AS3 info" and click 'Send'.



It will return a 200 OK with the installed version of AS3.

2. Next, you want to know if the BIG-IP already has AS3 declarations installed by using "Step 3.0.1: GET all AS3 declarations" and click 'Send'. This time it will return a '204 No Content'.

3. Open the Postman collection “F5 ATC – EMEA Partner Workshop” and select “3. AS3 Declarations”.
4. Select “Step3.1: Deploy HTTP_Service” and check the Body specifically at the poolmembers section. The declaration is included below for reference.

```
{
  "class": "AS3",
  "action": "deploy",
  "declaration": {
    "class": "ADC",
    "schemaVersion": "3.7.0",
    "id": "Deploy_App_Services",
    "label": "Deploy_App_Services",
    "remark": "Deploy_App_Services",
    "App_Services": {
      "class": "Tenant",
      "HTTP_Service": {
        "class": "Application",
        "template": "http",
        "serviceMain": {
          "class": "Service_HTTP",
          "virtualAddresses": [
            "{{bigip_ext_priv_vippip1}}"
          ],
          "snat": "auto",
          "pool": "Pool1"
        },
        "Pool1": {
          "class": "Pool",
          "monitors": [
            "http"
          ],
          "members": [
            {
              "servicePort": 80,
              "addressDiscovery": "azure",
              "updateInterval": 10,
              "tagKey": "service_discovery",
              "tagValue": "true",
              "addressRealm": "private",
              "resourceGroup": "{{resource_group}}",
              "subscriptionId": "{{subscription_id}}",
              "directoryId": "{{tenant_id}}",
              "applicationId": "{{client_id}}",
              "apiAccessKey": "{{client_secret}}",
              "credentialUpdate": false
            }
          ]
        }
      }
    }
  }
}
```

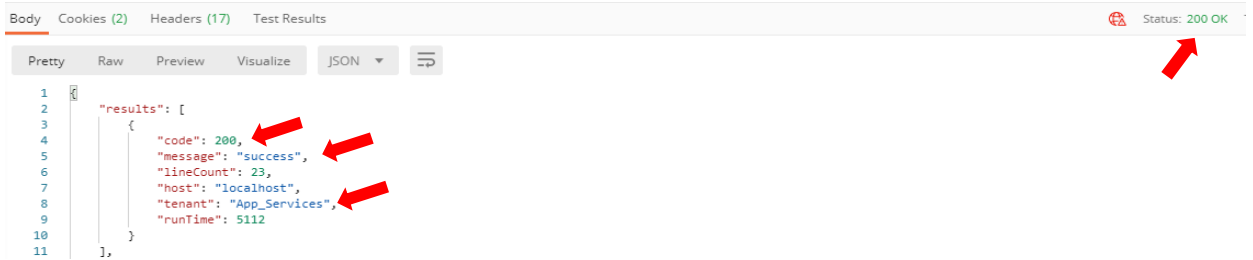


```

    }
  }
}
]

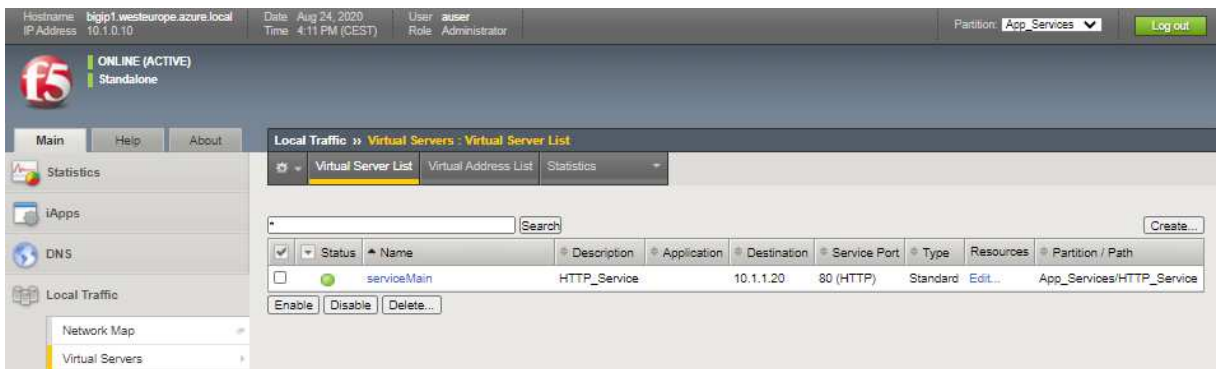
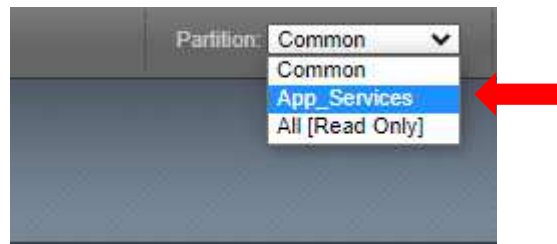
```

- Click 'Send' to POST "Step 3.1: Deploy HTTP_Service"
You should receive a 200 OK



- Log into the BIG-IP and check the local traffic configuration.

Be SURE to change the partition into the created App_Services partion to check the created VS.

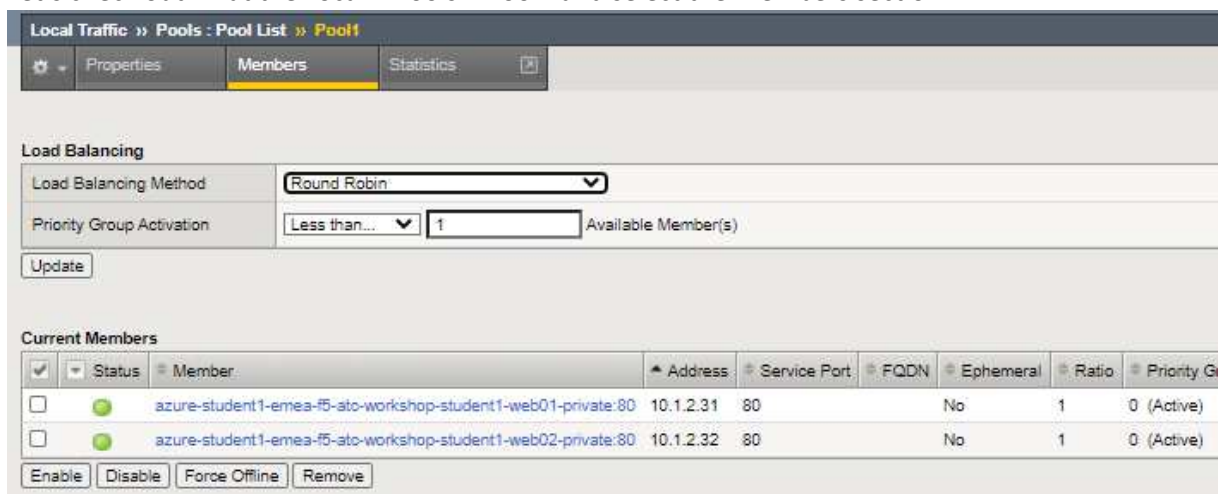


You will notice that the Virtual Server is marked 'Green'. Let's grab the VS (VIP) public IP address from the Terraform output and test it. You will see either web01 or web02 respond.



Remember we did not include pool members into our initial configuration (JSON declared AS3 schema).

- Let's check out what the Local > Pools > Pool1 and select the Members section:

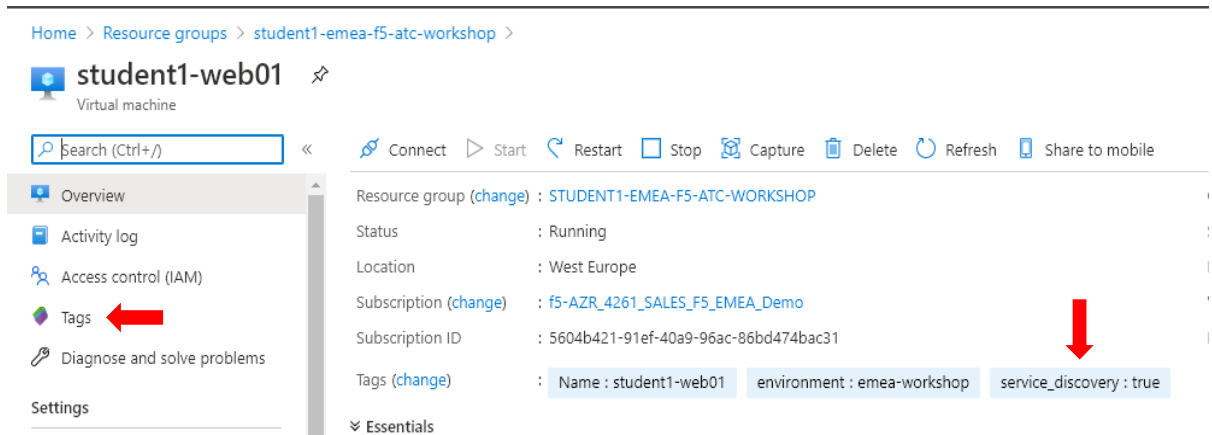


This pool includes two poolmembers and those are discovered via service-discovery config declared via AS3.

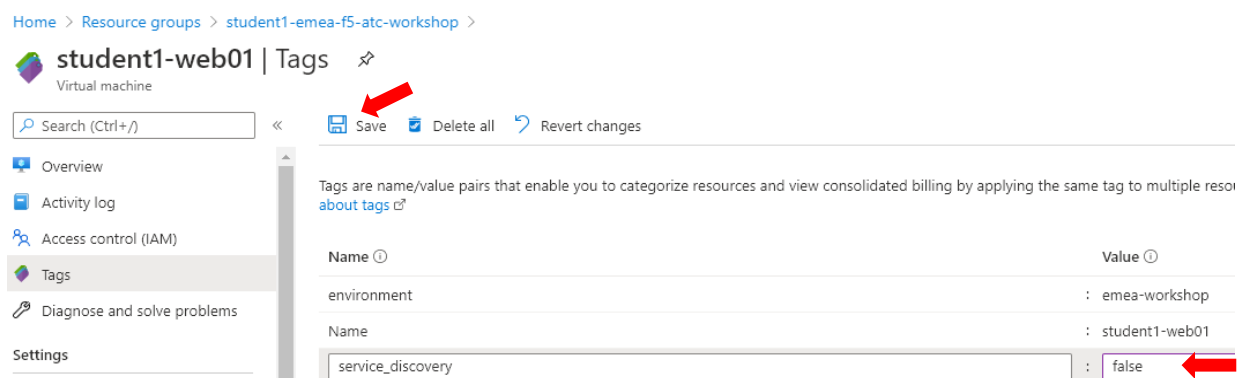
Service-discovery works by defining tagging to either the virtual machine or the interface and for this setup it was already included during the Terraform deployment of the web servers.

You can check this in Visual Studio Code > bigip-3nic-setup > web servers.tf and check the Web01 and Web02 tags. To see what the tag values are, checkout setup.yml.

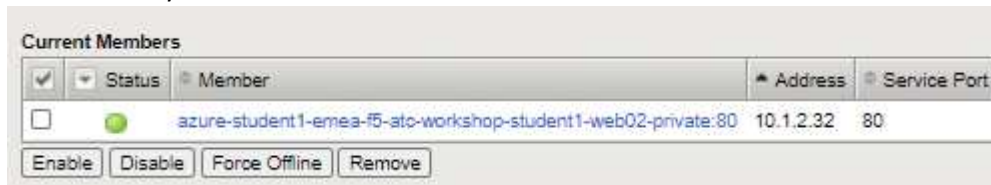
- Now let's verify these tags in Azure by open a browser and click the Azure portal link and select your personal resource group referenced by your student number. Select virtual machine web01 and verify the name tag and value.



9. In the same window, select tags and change the tag 'service-discovery' value from 'true' to 'false' and click 'Save'.



10. Now, go back to your BIG-IP and check your poolmembers in Pool1. How many poolmembers do you see?



11. Change back the tag value to 'true' in the Azure Portal and check the poolmembers section again.

Service-discovery of poolmembers can serve use cases, like:

- Taking a server backend from the pool for maintenance.
- Switch poolmember between 'prod' and 'test' environment.
- Automatically scale poolmembers when more get ramped up due to public cloud auto-scaling services.

Task 3.2 – Service HTTPS using HTTP to HTTPS redirect

This exercise will deploy application service using the Service HTTPS class and include adding a HTTP to HTTPS redirect.

If you left the previous HTTP Service deployed, then remember that a POST will update the current deployed application service with new deployed declaration.

1. In Postman, open the collection “F5 ATC – EMEA Partner Workshop” and next select “3. AS3 Declarations”. Select “Step 3.2: Deploy HTTP to HTTPS redirect Service” and verify underneath JSON schema.

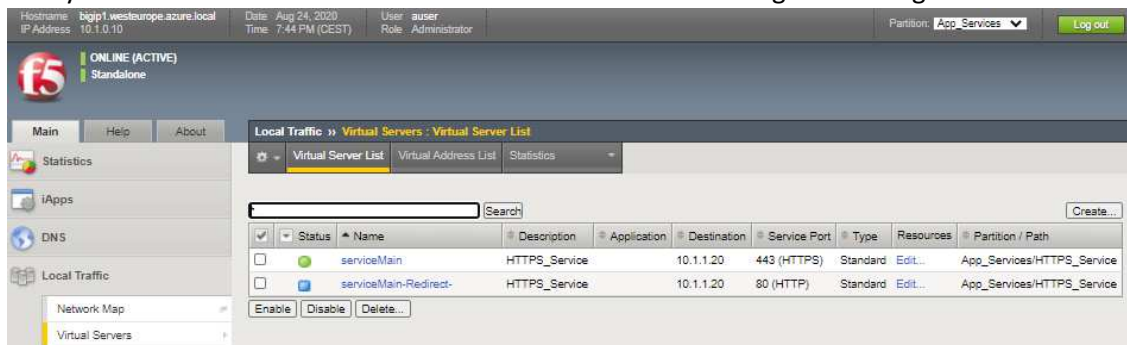
```
{
  "class": "AS3",
  "action": "deploy",
  "declaration": {
    "class": "ADC",
    "schemaVersion": "3.7.0",
    "id": "Deploying_App_Services",
    "label": "Deploying_App_Services",
    "remark": "Deploying_App_Services",
    "App_Services": {
      "class": "Tenant",
      "HTTPS_Service": {
        "class": "Application",
        "template": "https",
        "serviceMain": {
          "class": "Service_HTTPS",
          "virtualAddresses": [
            "{{bigip_ext_priv_vippip1}}"
          ],
          "snat": "auto",
          "pool": "web-pool",
          "profileHTTP": "basic",
          "serverTLS": "webtls"
        },
        "web-pool": {
          "class": "Pool",
          "monitors": [
            "http"
          ],
          "members": [
            {
              "servicePort": 80,
              "serverAddresses": [
                "{{webserver_1}}",
                "{{webserver_2}}"
              ]
            }
          ]
        }
      },
      "webtls": {
        "class": "TLS_Server",
        "certificates": [
          {
            "certificate": "webcert"
          }
        ]
      }
    }
  }
}
```

```

    },
    "webcert": {
      "class": "Certificate",
      "certificate": {
        "bigip": "/Common/default.crt"
      },
      "privateKey": {
        "bigip": "/Common/default.key"
      }
    }
  }
}

```

2. Click 'Send'.
3. Verify the declaration in the BIG-IP. You should have the same sight as the figure below.



4. Test the VS opening a web browser and go http://<BIG-IP_external_public_vip_address>
5. Notice the redirect to HTTPS.
6. Another thing you will notice is the fact that once a backend has been chosen, it will stick with that one. Go the web browser web01 or web02 depending which backend responded and click Demos > View Request and Response Headers and check the cookie section.

View Request and Response Headers

Virtual server address: 20.56.194.80
 Pool member address/port: 10.1.2.31:80
 Client IP address/port: 10.1.2.10
 Protocol: HTTP

Client Headers

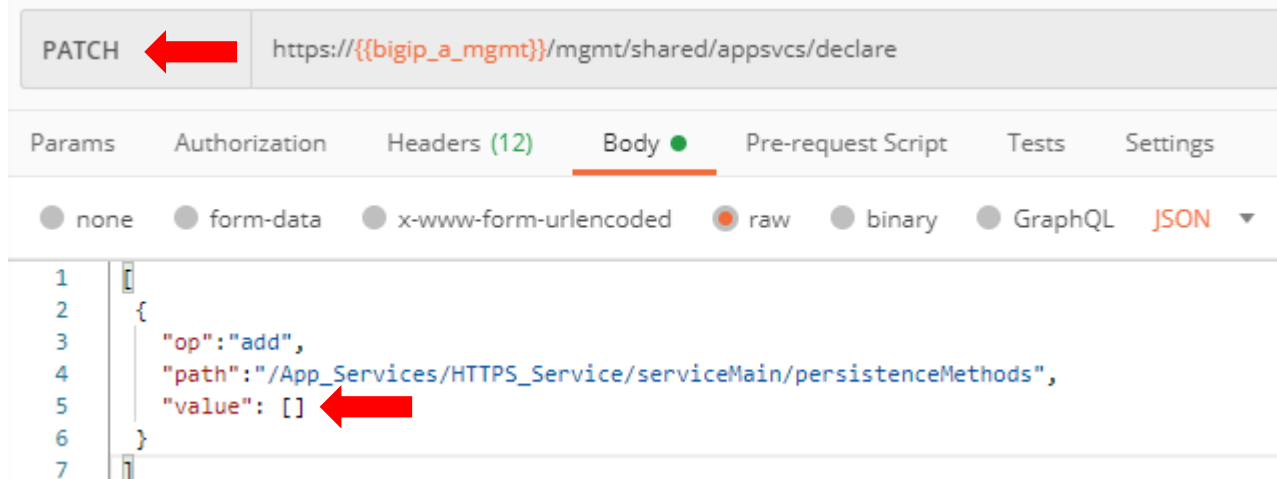
Request Headers Received at the Server

User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36
Cookie	BigipServer~App_Services~HTTPS_Service~web-pool=520225034.20480.0000; BigipServer~App_Services~HTTP_Service~Pool1=537002250.20480.0000

Cookie persistence is added as a default value to a Service_HTTP and Service_HTTPS.

7. In the collection, AS3 section, select “Step 3.2.1: Disable persistence ServiceMain” and click ‘Send’.

The persistence method gets changed by using a PATCH.



PATCH can be used on several occasions where configuration should be modified to change the POST-ed declaration. The PATCH method will use an operator, in this case “add”. It needs a path and a value. For no persistence, the value of ‘persistenceMethods: []’.

8. Re-test the backends and notice that you now will get switched between both backends. Be aware that cached objects of both backends will be shown.
9. Delete the declaration by selecting “Step 3.2.3: Delete Service_HTTPS declaration” in Postman. Before doing so, take a minute, to check the body and notice:
 - Deleting a declaration happens by declaring a POST
 - The declaration needs to include the “id” and “Tenant” name at a minimum. As you can see the rest stays empty.
10. Press ‘Send’ and check the BIG-IP. What got removed?

Task 3.3 – ServiceMain and Generic

Service_HTTP and Service_HTTPS where the default templates until AS3 v3.20. These templates enforce to name each created virtual service to be named serviceMain. From v3.20 this has changed from serviceMain into service. Also, the generic template has become the default now which allows to use any name.

Task 3.3 covers:

- Show ServiceMain and Generic differences.
- Declare multiple applications in one AS3 deployment.
- Use of PATCH to modify applications within a tenant.

1. In Postman, select “Step 3.3: Deploy serviceMain and service Generic” and copy and paste the JSON schema after you changed the IP address and aligning them with your student number.

```
{
  "class": "AS3",
  "persist": false,
  "declaration": {
    "class": "ADC",
    "schemaVersion": "3.20.0",
    "id": "serviceMain-and-generic",
    "shared_tenant": {
      "class": "Tenant",
      "my_generic_app": {
        "class": "Application",
        "myApp": {
          "class": "Service_Generic",
          "virtualAddresses": [
            "{{bigip_ext_priv_vippip1}}"
          ],
          "virtualPort": 8080,
          "shareAddresses": false,
          "pool": "web-pool8080"
        },
        "web-pool8080": {
          "class": "Pool",
          "monitors": [
            "http"
          ],
          "members": [{
            "servicePort": 80,
            "serverAddresses": [
              "{{webserver_1}}"
            ]
          }]
        }
      }
    },
    "HTTPS_Service": {
      "class": "Application",
      "template": "https",
      "serviceMain": {
        "class": "Service_HTTPS",
        "virtualAddresses": [
          "{{bigip_ext_priv_vippip1}}"
        ],
        "shareAddresses": false,
        "snat": "auto",
        "pool": "web-pool",
        "profileHTTP": "basic",
        "serverTLS": "webtls"
      }
    }
  }
}
```

```

    },
    "web-pool": {
      "class": "Pool",
      "monitors": [
        "http"
      ],
      "members": [{
        "servicePort": 80,
        "serverAddresses": [
          "{{webserver_2}}"
        ]
      }]
    },
    "webtls": {
      "class": "TLS_Server",
      "certificates": [{
        "certificate": "webcert"
      }]
    },
    "webcert": {
      "class": "Certificate",
      "certificate": {
        "bigip": "/Common/default.crt"
      },
      "privateKey": {
        "bigip": "/Common/default.key"
      }
    }
  }
}

```

2. Click 'Send' and after the 200 OK in the response, check the BIG-IP.

The screenshot shows the F5 BIG-IP management console interface. At the top, system information is displayed: Hostname (bigip1.westeurope.azure.local), IP Address (10.1.0.10), Date (Aug 25, 2020), Time (11:02 AM (CEST)), User (auser), Role (Administrator), Partition (shared_tenant), and a Log out button. The main interface features a left-hand navigation menu with options like Main, Help, About, Statistics, iApps, DNS, Local Traffic, Network Map, and Virtual Servers. The main content area is titled 'Local Traffic >> Virtual Servers: Virtual Server List' and contains a table of virtual servers. The table has columns for Status, Name, Description, Application, Destination, Service Port, Type, and Resources. Three virtual servers are listed: 'myApp' (my_generic_app, 10.1.1.20, 8080), 'serviceMain' (HTTPS_Service, 10.1.1.20, 443 (HTTPS)), and 'serviceMain-Redirect' (HTTPS_Service, 10.1.1.20, 80 (HTTP)). Below the table are buttons for Enable, Disable, and Delete.

Status	Name	Description	Application	Destination	Service Port	Type	Resources
<input type="checkbox"/>	myApp	my_generic_app		10.1.1.20	8080	Standard	Edit...
<input type="checkbox"/>	serviceMain	HTTPS_Service		10.1.1.20	443 (HTTPS)	Standard	Edit...
<input type="checkbox"/>	serviceMain-Redirect	HTTPS_Service		10.1.1.20	80 (HTTP)	Standard	Edit...

Within one tenant both Service_HTTPS and a Service_Generic are deployed.

Notice the following:

- Service_Generic uses a random name, where Service_HTTPS is tied to the enforced naming.
 - All services are using the same VS IP address, but a different port
 - Tenants or partitions can include multiple applications.
 - Our webserver have been divided between the declared applications.
 - o Web01 is used as a poolmember for my_generic_app.
 - o Web02 is used as a poolmember for ServiceMain.
3. Test the configuration by opening a web browser and test the different defined virtual servers:
- http://<BIG-IP_external_public_vip_address>
 - http://<BIG-IP_external_public_vip_address>:8080
4. What if we would want to reuse the same poolmembers and share it among the applications...
- We are going to use a POST to add a function by copying the previous JSON schema into the body of “Step 3.3: Deploy serviceMain and service Generic with shared poolmembers”.
- Next in the body find pool member section of my_generic_app and add the poolmember as shown in the picture.

```

"members": [{
  "servicePort": 80,
  "serverAddresses": [
    "{{webserver_1}}",
    "{{webserver_2}}"
  ]
}]

```



Don't forget to obey the JSON syntax set a comma after the first IP address, before adding “{{webserver_2}}”.

5. Click 'Send' and check the response by browsing to http://<BIG-IP_external_public_vip_address>:8080

Use 'Ctrl+F5' to refresh and switch between the poolmembers.

Adding a poolmember could be done by using a PATCH as well.

6. Now, let's delete the deployment, but this time we only delete one application within the existing tenant. We can do this in two different ways, either by modifying the previous POST and delete the application section which we want to be removed from the tenant by re-POST-ing the declaration in its new existence or by using a PATCH.

Select “Step 3.3.1: Remove application from tenant through AS3” and select the body.

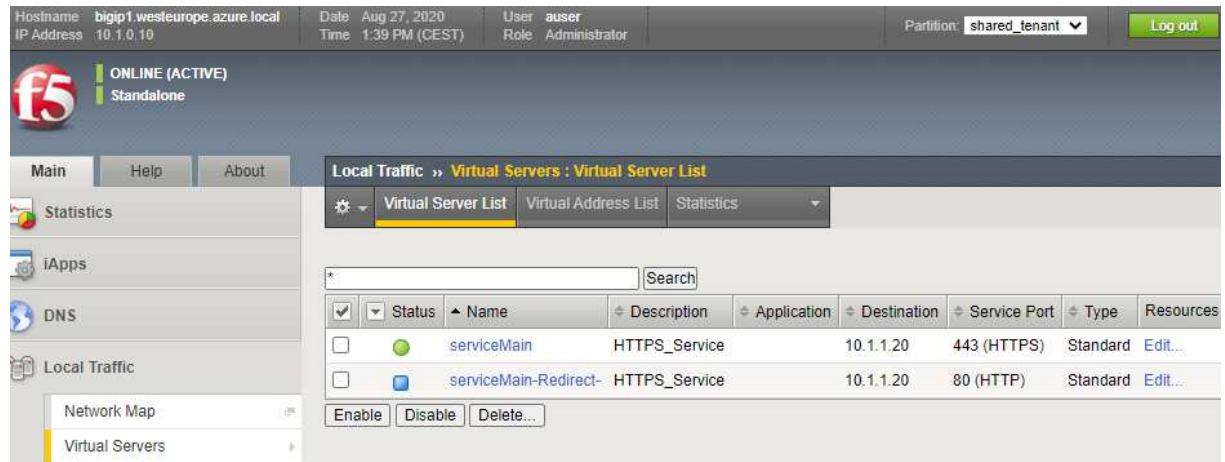
```

[
  {
    "op": "remove",
    "path": "shared_tenant/my_generic_app",
    "value": []
  }
]

```

The operator task is now to 'remove' and notice the path which is pointing to my_generic_app.
Now copy and paste and click 'Send'.

7. Check the declaration via GET, this can be achieved by grabbing the declaration from Step 3.3 and change the declaration method from POST to GET and hit 'Send' and watch the response.
8. Check the BIG-IP. It should look similar as the below picture, where my_generic_app has been removed.



9. Remove the entire config by selecting "Step 3.3.3: Delete AS3 declaration serviceMain and generic" and press 'Send'.
10. Check the BIG-IP if the tenant has been removed.

Task 3.4 – Use of Global Server Load Balancing (GSLB)

This task will teach how to setup Global Server Load Balancing (GSLB) using AS3 declarations.

To accomplish this, the following actions need to take place:

- Setup BIGIP2 DO
- Create an AS3 declared L4-L7 config at both BIG-IP's
- Create an AS3 config for GSLB
- Test it

Task 3.4.1

1. In module 2 declarative onboarding for BIGIP2 should have taken place. When this is not the case, please go back and complete this task first before moving any further.

Once both BIG-IP's are past the phase of onboarding, next they need to have some L4-L7 application services deployed to get towards demonstrating GSLB.

2. In Postman, go to “Step 3.4.1: BIGIP-1 AS3 deployment” and select underneath body to copy and paste in the body of the AS3 declaration and hit ‘Send’.
Before doing so, think what this AS3 declaration will do?

```
{
  "class": "AS3",
  "action": "deploy",
  "declaration": {
    "class": "ADC",
    "schemaVersion": "3.7.0",
    "id": "Deploying_App_Services",
    "label": "Deploying_App_Services",
    "remark": "Deploying_App_Services",
    "App_Services": {
      "class": "Tenant",
      "HTTPS_Service": {
        "class": "Application",
        "template": "https",
        "serviceMain": {
          "class": "Service_HTTPS",
          "virtualAddresses": [
            "{{bigip_ext_priv_vippip1}}"
          ],
          "snat": "auto",
          "pool": "web-pool",
          "profileHTTP": "basic",
          "serverTLS": "webtls"
        },
        "web-pool": {
          "class": "Pool",
          "monitors": [
            "http"
          ],
          "members": [
            {
              "servicePort": 80,
              "serverAddresses": [
                "{{webserver_1}}"
              ]
            }
          ]
        },
        "webtls": {
          "class": "TLS_Server",
          "certificates": [
            {
              "certificate": "webcert"
            }
          ]
        }
      }
    }
  }
}
```

```
    },  
    "webcert": {  
      "class": "Certificate",  
      "certificate": {  
        "bigip": "/Common/default.crt"  
      },  
      "privateKey": {  
        "bigip": "/Common/default.key"  
      }  
    }  
  }  
}  
  
}
```

- Wait for the 200 response and check the declared AS3 deployment on BIG-IP01 and test if get answer from the backend when hitting the virtual server in a browser.
- Select “Step 3.4.2: BIGIP-02 AS3 deployment” and copy and paste underneath body and click ‘Send’.

```
{
  "class": "AS3",
  "action": "deploy",
  "declaration": {
    "class": "ADC",
    "schemaVersion": "3.7.0",
    "id": "Deploying_App_Services",
    "label": "Deploying_App_Services",
    "remark": "Deploying_App_Services",
    "App_Services": {
      "class": "Tenant",
      "HTTPS_Service": {
        "class": "Application",
        "template": "https",
        "serviceMain": {
          "class": "Service_HTTPS",
          "virtualAddresses": [
            "{{bigip_ext_priv_vippip2}}"
          ],
          "snat": "auto",
          "pool": "web-pool",
          "profileHTTP": "basic",
          "serverTLS": "webtls"
        },
        "web-pool": {
          "class": "Pool",
          "monitors": [
            "http"
          ],

```

```

        "members": [
            {
                "servicePort": 80,
                "serverAddresses": [
                    "{{webserver_2}}"
                ]
            }
        ],
        "webtls": {
            "class": "TLS_Server",
            "certificates": [
                {
                    "certificate": "webcert"
                }
            ]
        },
        "webcert": {
            "class": "Certificate",
            "certificate": {
                "bigip": "/Common/default.crt"
            },
            "privateKey": {
                "bigip": "/Common/default.key"
            }
        }
    }
}
}
}
}

```

5. Check the config on BIG-IP02 and test it just like step 3.
6. Both BIG-IPs have the same 'local' configuration and if those should be able back each other up for disaster recovery reasons, drive migration scenarios or just deliver scale, that's where the use of GSLB becomes handy.

Select "Step 3.4.3: Deploy GSLB" and check the JSON body below to understand what get's declared.

```

{
    "class": "ADC",
    "schemaVersion": "3.6.0",
    "id": "GSLB_Sample",
    "GSLB_App_Services": {
        "class": "Tenant",
        "Application": {

```

```

    "class": "Application",
    "template": "generic",
    "testDomain": {
        "class": "GSLB_Domain",
        "domainName": "lab.f5atc.local",
        "resourceRecordType": "A",
        "poollbMode": "ratio",
        "pools": [
            { "use": "service-azure-eu-west1-pool" },
            { "use": "service-azure-eu-west2-pool" }
        ]
    },
    "service-azure-eu-west1-pool": {
        "class": "GSLB_Pool",
        "enabled": true,
        "members": [
            {
                "ratio": 1,
                "server": {
                    "use": "/Common/Shared/bigip1-azure-west-eu"
                },
                "virtualServer": "0"
            }
        ],
        "resourceRecordType": "A"
    },
    "service-azure-eu-west2-pool": {
        "class": "GSLB_Pool",
        "enabled": true,
        "members": [
            {
                "ratio": 1,
                "server": {
                    "use": "/Common/Shared/bigip2-azure-west-eu"
                },
                "virtualServer": "0"
            }
        ],
        "resourceRecordType": "A"
    }
},
"Common": {
    "class": "Tenant",
    "Shared": {
        "class": "Application",
        "template": "shared",
        "azure-west1-eu": {
            "class": "GSLB_Data_Center"
        },
        "azure-west2-eu": {

```

```

        "class": "GSLB_Data_Center"
    },
    "bigip1-azure-west-eu": {
        "class": "GSLB_Server",
        "dataCenter": {
            "use": "azure-west1-eu"
        },
        "devices": [
            {
                "address": "{{bigip_1_ext_selfip_pubip}}",
                "addressTranslation": "{{bigip_1_ext_selfip_privip}}"
            }
        ],
        "virtualServers": [
            {
                "address": "{{bigip_ext_pub_vippip1}}",
                "addressTranslation": "{{bigip_ext_priv_vippip1}}",
                "addressTranslationPort": 443,
                "port": 443,
                "monitors": [{"bigip": "/Common/https"}]
            }
        ]
    },
    "bigip2-azure-west-eu": {
        "class": "GSLB_Server",
        "dataCenter": {
            "use": "azure-west2-eu"
        },
        "devices": [
            {
                "address": "{{bigip_2_ext_selfip_pubip}}",
                "addressTranslation": "{{bigip_2_ext_selfip_privip}}"
            }
        ],
        "virtualServers": [
            {
                "address": "{{bigip_ext_pub_vippip2}}",
                "addressTranslation": "{{bigip_ext_priv_vippip2}}",
                "addressTranslationPort": 443,
                "port": 443,
                "monitors": [{"bigip": "/Common/https"}]
            }
        ]
    }
}

```

7. Copy and paste it into the body of step 3.4.3 and select 'Send'.

8. Check your BIG-IP and be sure to check the right one and select the right tenant and go to:
 - GSLB > Wide IPs and select the created WIP.
 - GSLB > Pools and check out the pool members, they should be 'green'.
 - GSLB > Datacenters, two datacenters have been defined.
 - GSLB > Servers, check the device and virtual servers.

All should be green, so let's start testing.

TESTING

9. In the Jumpshot open a Linux terminal or the VSC terminal.
10. In bash type the following: **dig @<server> name <wideIP>**

Server = BIGIP-2 external IP address

wideIP = lab.f5atc.local

```
f5student@ubuntu:~/f5-atc-workshop$ dig @20.50.198.77 lab.f5atc.local

; <<>> DiG 9.11.3-ubuntu1.13-Ubuntu <<>> @20.50.198.77 lab.f5atc.local
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57859
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: de2ea35cdf9af4cd5cf1a04f5f5b94319ad89a4d85354ac2 (good)
;; QUESTION SECTION:
;lab.f5atc.local.                IN      A

;; ANSWER SECTION:
lab.f5atc.local.                0       IN      A       20.56.48.217
lab.f5atc.local.                0       IN      A       20.50.198.70

;; AUTHORITY SECTION:
f5atc.local.                    0       IN      NS      this.name.is.invalid.

;; Query time: 6 msec
;; SERVER: 20.50.198.77#53(20.50.198.77)
;; WHEN: Fri Sep 11 15:13:53 UTC 2020
;; MSG SIZE rcvd: 138
```

You should see two A records getting resolved, which means that both VS in the GTM pool are responding.

Optional, you can twist steps 11 and 12 and delete one config and re-test GSLB.

11. Once you are convinced that GSLB works and actually has been configured through AS3. Let's delete the declaration by selecting "Step 3.4.4: Delete GSLB" and press 'Send'.
12. Delete the configuration of the BIG-IP pp services by using Postman step 3.4.5 and 3.4.6.

Module 4 – F5 Application Services Templates

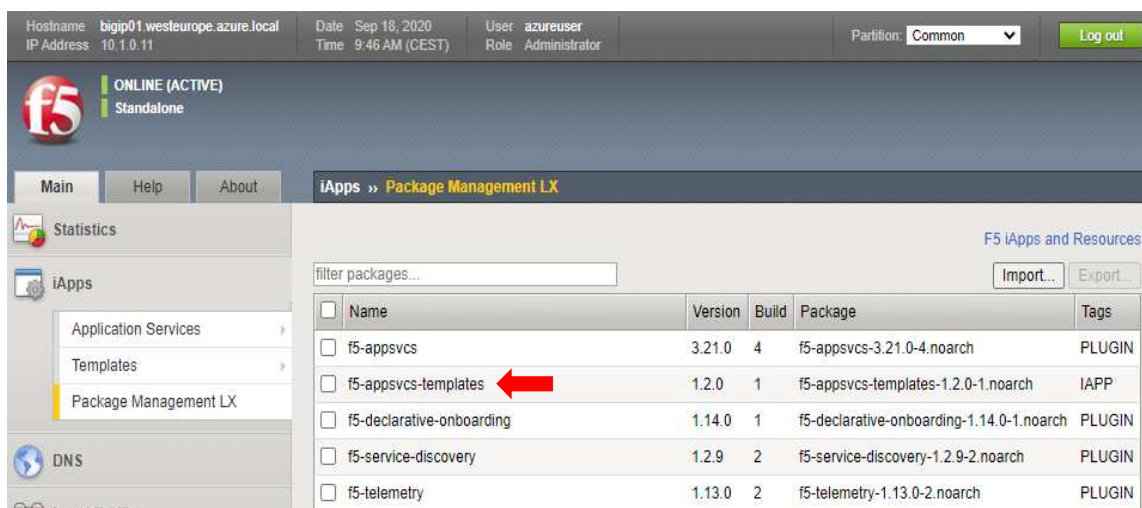
F5 Application Services Templates (FAST) are an easy and effective way to deploy applications on the BIG-IP system using [AS3](#).

The FAST Extension provides a toolset for templating and managing AS3 Applications on BIG-IP. FAST will replace the TCL based iApps over time.

Throughout this lab we only touched the BIG-IP to verify API declared configuration. During this lab we will use the BIG-IP GUI to deploy FAST at first.

Task 4.1 – Explore FAST via the GUI

1. Login BIG-IP1 and go to iApps > Package Management LX and check if the FAST .rpm is installed.



FAST needs its own .rpm installed, but also the AS3 .rpm.

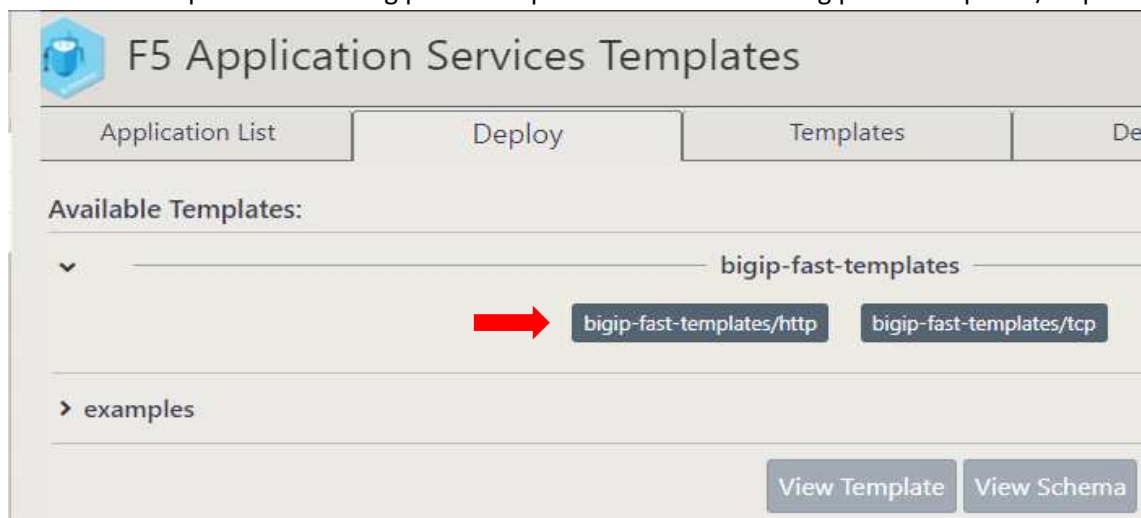
Next, go to iApps > Application Services > Applications LX and Select the F5 Application Services Templates UI extension.



- You must Sign in before getting access to the FAST extension (same as BIGIP username/password).
- The start window does not show much at this point but take your time to go through the tabs before continuing.

Application List	A list of AS3 deployed applications. Applications deployed via FAST can be modified or deleted.
Deploy	Create new applications using FAST
Templates	List of installed templates
Deploy Log	Summary of AS3 async task results
API	Documentation how to use FAST via REST API

- Let's deploy our first template using the FAST extension. Select tab 'Deploy' and in the "Available Templates" select 'bigip-fast-templates' and then click 'bigip-fast-templates/http'.



- Fill in the template with the following values:

What is the tenant name?	Task4_1_FAST
What is the application name?	http-service
What IP address do you want to use for the virtual server?	10.<your_student_number>.1.20
Server Addresses	
Add the IP addresses of a pool member	10.<your_student_number>.1.31 10.<your_student_number>.1.32

Leave everything else at the default values.

Click 'Submit'.



The Deploy log will show a successful creation of the template.

- Check if the virtual server is deployed and be sure to change to the right partition.

7. Test the deployed config by using the public IP of the VIP (bigip_ext_pub_vippip1).
8. Let's go to the 'Application List' and watch our first template being in the list.
Scroll down to the bottom of the page and use the different buttons to view the Template, Schema, Inputs and Rendered.
Output is shown below the buttons.

Task 4.2 – Use FAST via the API

1. Open Postman and go to the “4. F5 Application Services Templates (FAST)” in the “Collections” and select “Step 4.2.1: GET FAST info” and hit ‘Send’.
Since FAST depends on AS3 you will see info about both.

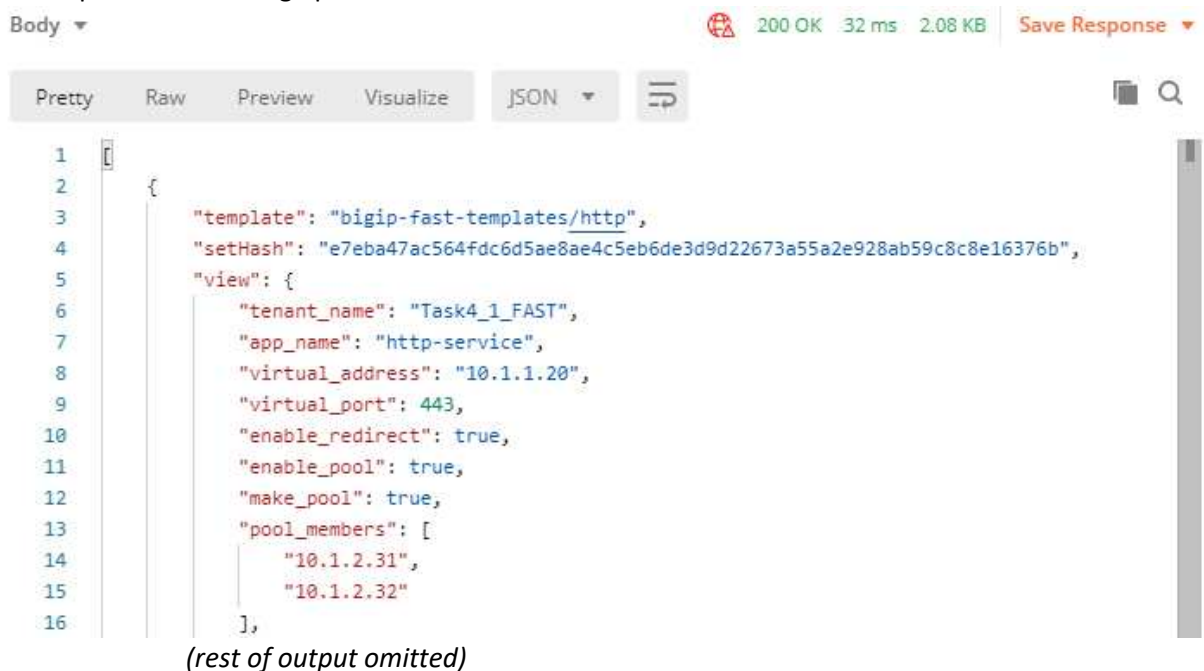


The screenshot shows the Postman interface with the 'Body' tab selected. The response is in JSON format, showing the FAST version and AS3 info. The JSON is as follows:

```
{
  "version": "1.3.0",
  "as3Info": {
    "version": "3.21.0",
    "release": "4",
    "schemaCurrent": "3.21.0",
    "schemaMinimum": "3.0.0"
  },
  (rest of output omitted)
}
```

Read through the response to understand what gets delivered.

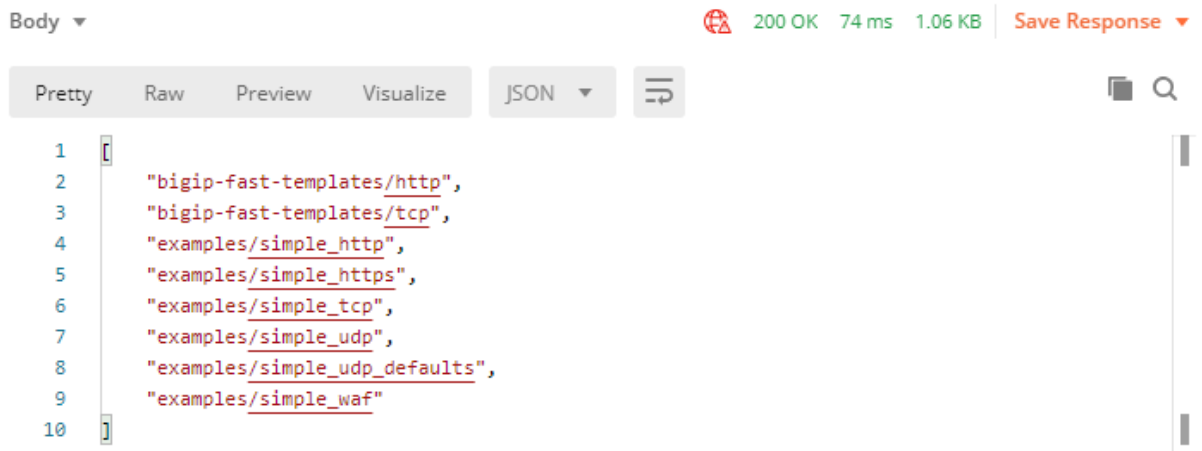
2. Select “Step 4.2.2: GET FAST applications” to show if any applications are deployed using FAST.
When you did not delete the FAST app deployment from the previous task, you should see this http-service showing up.



The screenshot shows the Postman interface with the 'Body' tab selected. The response is in JSON format, showing the details of the FAST application. The JSON is as follows:

```
{
  "template": "bigip-fast-templates/http",
  "setHash": "e7eba47ac564fdc6d5ae8ae4c5eb6de3d9d22673a55a2e928ab59c8c8e16376b",
  "view": {
    "tenant_name": "Task4_1_FAST",
    "app_name": "http-service",
    "virtual_address": "10.1.1.20",
    "virtual_port": 443,
    "enable_redirect": true,
    "enable_pool": true,
    "make_pool": true,
    "pool_members": [
      "10.1.2.31",
      "10.1.2.32"
    ]
  },
  (rest of output omitted)
}
```

3. To be able to deploy an application using FAST we need a template.
In Postman, select “Step 4.2.3: GET FAST templates” and hit ‘Send’.



4. Copy “examples/simple_waf” from the response and paste it into the URI after /templates.

Like this: https://{{bigip_1_mgmt}}/mgmt/shared/fast/templates/examples/simple_waf

And hit ‘Send’.

The examples/simple_waf template layout is shown in the response. Scroll through to understand which options you can configure and pay special attention to the ‘required’ list.

```
    "required": [
      "tenant_name",
      "application_name",
      "virtual_port",
      "virtual_address",
      "WAF_policy_path",
      "server_port",
      "server_address",
      "certificate",
      "private_key"
    ],
```

5. Before moving on with creating the template, FAST expects the WAF policy already to be present on the BIG-IP. So, first let’s install this policy by going into Postman and select “Step 4.2.4: Create local WAF policy” and check the URI and Body. This is not an AS3 nor a FAST-related REST call. Be aware that with all the ‘magic’ created by F5 automation toolchain, the BIG-IP still has a full functioning ‘Native’ iControl REST API.
This request just creates a WAF policy which gets stored in the partition /Common.

► Step 4.2.4: Create local WAF policy

POST ▼ https://{{bigip_1_mgmt}}/mgmt/tm/asm/policies

Params Auth Headers (11) **Body** ● Pre-req. Tests Settings

raw ▼ JSON ▼

```

1 {
2   "name": "base_waf_policy"
3 }
```

Hit 'Send'.

- Now, based on the 'required' list a template has been created in the below section. Copy and paste it into the body of "Step 4.2.5: Deploy FAST simple_waf" and click 'Send'.

```

{
  "name": "examples/simple_waf",
  "parameters": {
    "tenant_name": "Task4_2_FAST",
    "application_name": "waf-service",
    "virtual_port": 443,
    "virtual_address": "10.1.1.30",
    "WAF_policy_path": "/Common/base_waf_policy",
    "server_port": 80,
    "server_address": ["10.1.2.35"],
    "certificate": "-----BEGIN CERTIFICATE-----
\nMIIDrjCCApagAwIBAgIEEDZqSTANBgkqhkiG9w0BAQsFADCBmDELMakGA1UEBhMC\nVVMxZCzAJBgNVBA
gTAldBMRAwDgYDVQQHEwdTZWF0dGxIMRIwEAYDVQQKEw1NeUNv\nnbXBhbnkxCzAJBgNVBAsTAK1UMR4wHA
YDVQQDExVsb2NhbkGhvc3QubG9jYXxkb21h\nnaW4xKTAnBgkqhkiG9w0BCQEWGnJvb3RABG9jYXxob3N0Lm
xvY2FsZG9tYWlUMB4X\nnDTE4MDgxNTEyMDAwOV0XDTI4MDgxMjEyMDAwOVowgZgxZCzAJBgNVBAYTA1VTMQ
sw\nnCQYDVQQIEwJXQTEQMA4GA1UEBxMHU2VhdHRsZTESMBAGA1UEChMJTX1Db21wYW55\nnMQswCQYDVQQLEwJ
JVD EeMBwGA1UEAxMVbG9jYXxob3N0LmXvY2FsZG9tYWlUMSkw\nnJwYJKoZIhvcNAQkBFhpyb290QGxv
Y2Fsag9zdC5sb2NhbkGRvbwFpbjCCASIwDQYJ\nnKoZIhvcNAQEBBQADggEPADCCAQoCggEBALhgsSNqnxeC
YeinyxwPpUW379NE08aV\nnHdR/RCNYbFlmXhtsIRWsTtxt0pSNUK91b5MEY707zo/01rW8QoINS74c6kKde
EByu\nnOcaTrn8+VFF9i46J5/1TiTb1ag7QZfs/B1jZt4+VMwkcL3S3tqN/ffX6FF0tuNiW\nnC8V+9l2NWq
Zg9hynR0mqwxSmyBRhewDtGcHHbW77q27yc4pb5Wr9nyJxUqlnqZMZ\nn73T5WMyjpVZx/3qW0CuimpWKhT
ET5j4G7Vv9YGkABd33EULwcCLoRGBhZ7I7qORP\nnJAIHvpt/AJyMTJXHhV7Krp8cKZF3sgG6NveuCbwvK
3CEI5Q80XQ4UCAwEAATAN\nnBgkqhkiG9w0BAQsFAAOCAQEAX8k0XpXCP2josNyCE50YNB63SvMSOU9Mo2
1bkWKz\nnEXmn+SYg+wx0RlvfcxUBjPolbVklDZRBcB6xk5AjlJSDx+3Jz039K/Ypj4wcF+Ia\nnLIHlkaEn
T0eMsUsCPRXPnnRfyVER7+ER7TQURWRhrw+9QYLmwFX1bLzDXwEX7Wb4\nnLzjxFTZT5UoQsoE94XGScjeZ
JDdvDjpm1o3R5ag57MfxAFcS9rmW/tNs2xNYh3U8\nnGzeJgPH4Rm8zRdWrVeq/AEyq0CwWZxRI1S3YrqjA
c+nfpVQbC06yGRRLW1CKUZwf\nnpx+z/vcZaA9+e2LW3MafBNAEvCTXCoE0X0Ax38AYWRCq7g==\nn-----
END CERTIFICATE-----\n",
    "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIIEvIBADANBgkqhkiG9w0BAQEFAASCBBKwggSlAgEAAoIBAQC4YLEjap8XgmHo\nnsp8cD6VFt+/TRN
PG1R3Uf0QjWgXZZ1x7bCEVrE8bdKUjbivdw+TBG09086P9Ja1v\nnEKCDUu+HOpCnXhAcrjnGk65/P1RRfY
uOief9U4k25Wo00GX7PwY2beP1TMJHC90\nnt7ajf331+hRdLbjYlgvFfvZdjVqmYPYcp0dJqsMUPsgUYX
sA7RnBx21u+6tu8nOK\nnW+Vq/Z8icVKpZ6mTGe90+VjMo6VWcf961tAropqVioUxE+Y+Bu1b/WBpAAXd9x
-----"
```

```

FC\n8HAI6ERgYWey06jkTyQCB76bfwCcJyEyVx4Veyq6Z/HCMrd7IBujVXrgm8LytwhCO\nUPNEF00FAGMB
AAECggEAI6Mi0ezjm7RqjsLpsI+QmxlqnDwHL/C2D5LavXczmBMK\nKJv1iB4i0zUQTzkV9ubyX+VdKfY7
OeycqWU+FSwEWq+18neipxyA09JLLdKDEy4w\nOYiLw+Hv2WG93mem8o1T9vc7/N9yeh4NySJSRadYliD0
ay8xdYGI/G8mGN5/yT0L\nUgj91AHKA2UJDQ1RzYFbZxF/eCWwZ6tosHraEwGKDdlqSkuZ62fQu5IoN97L
IX2+\nXPZm6t+ROdUAURE2N4zhOperh3UYxmHTTzJTn/h0f6Lhb0VVbEHj5WjxKJVp3deH\nnmBof5IZQxb
m1iTzzLKP3ERwIpaalX+dIi9Q0pzhSQKBgQD4KqgHmkN1j8vikF3N\nseL8JqTveQd6AKPeB+TlWH/NWu
bpSK9f+EUQv+gN99u79GQg2N+gIiJc101S0i0j\nY/xdIcpbFf7qQzaZ1W0USi3nakryInc7s3NNCWIEX/
h5YsyxD/01vwg570YXfEle\nASNLDiIz3w6wX/ebP7CpVjc3rQKBgQC+MpS1ws1dSwrr3CvWMkC1bAc2n3
20A9oE\nlr/YPKgQ/1CRprp1I+dF51iHxIHDnZqwnYTEYphitxj3RqTuppXu56FG3xmJWkV1\nnkxivn+E4
xahqLp9aaMRvFTTw/rvUUMzxLGU/3sTW6js/EcaI+u0PY9qKXDKJwv9Z\nYGGby28W0QKBgQCg8NkEBTFW
nqj1B/ctnP291ToF7BHDN4MOTUR00HQhs6ApDnBd\n5t6znCfcXa/tVvNQshVk/n4WwKRwh8mqN//ET10z
erJVLr3MILiBHH8qdxTmtOk\nnrh5KiZk9iRfVcR0aiOPerEMjf1+Pf5T9F/PRixr3VONr0vD7h/SD/VvW
CQKBgQCU\nvN8j+Cwgg/UEqgoeiQYGcvowFu3GEdLiXeahZ1D2VPDzvzPV2KwGX0CfehJKdKpBb\nDhBeR3
QJEjujqXK8PCuwsQx1xV4addDI2jymaVE7U0EvCZHko0bApRPHqn5DKLr\nnu6+gmDQ4V2osL8YEUomQx1
XOMkisj+0vHXAj3hBfaQKBgQCNi4IMcCPCh/BSGgb\n9nBU0YG0Y7856ca7nLo4x2G67NigGAp1EgvYW6i
4wD/1zo70/af8UzYjTh7+Bse08\nZy3N+h6eajbzu6j3I3Sp+3TGGZcw/8s5EU2SPM13Rf25wW0M33Xn0A
wxGLzUY/dX\nnNCffdm6CVXWk73y++Xahj5AiIw==\n-----END PRIVATE KEY-----\n"
}
}

```

7. Check the FAST deployed applications on the BIG-IP.

Tenant	Application	Template	Actions
Task4_1_FAST	http-service	bigip-fast-templates/http	[Edit] [Delete]
Task4_2_FAST	waf-service	examples/simple_waf	[Edit] [Delete]

Check the deploy log and after this select the right partition and test if you have connection to the backend.

8. To clean up what has been created by FAST. Go into the FAST template section and in the 'Application List' delete both FAST deployments.

This concludes FAST for this workshop.

Be aware that FAST goes beyond this point by letting you create your own FAST templates. More details can be found here: <https://clouddocs.f5.com/products/extensions/f5-appsvcs-templates/latest/userguide/template-authoring.html>

Task 3.6– Deploy a second BIG-IP and create high availability using CFE

Within this exercise we are going to take the following actions:

Module 5 – Telemetry Streaming (TS)

This time we will use an AS3 declaration to add a WAF policy to the configuration. This WAF policy does not exist on the BIG-IP and will get pulled from an external resource. Next exercise will demonstrate how F5 Big-IP can integrate with Azure Sentinel and leverage Telemetry Streaming.

This task will cover:

- Deploy Azure Sentinel
- Create an event listener
- Deploy AS3 declaration, including a WAF policy
- Deploy a Telemetry Streaming declaration

Task 5.1 – Azure Sentinel and TS

This task will setup:

- Azure Sentinel
 - Grab the Azure Sentinel Workspace ID and Primary Key
 - Deploy Telemetry Streaming Declaration
1. As we done before with DO and AS3, first check if TS installed. Select “Step 4.0: GET Telemetry Streaming info” and click ‘Send’.

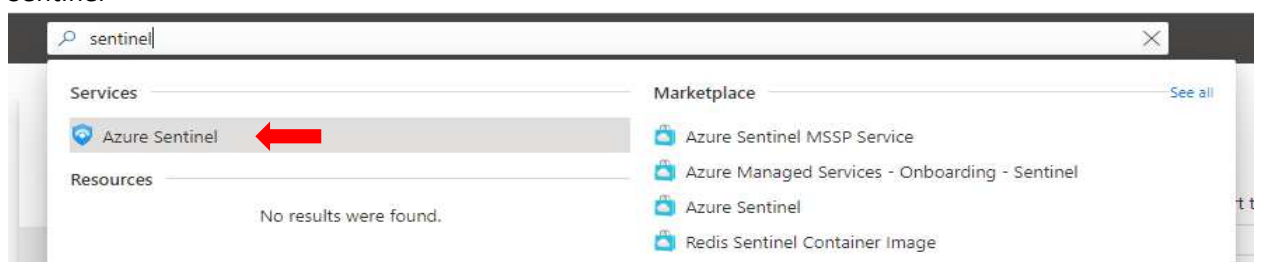


2. Login to the Azure portal and go to Resource groups and select your created resource group recognized by your student number.
3. In the selected resource group find the Log Analytics Workspace called 'student#-log'.

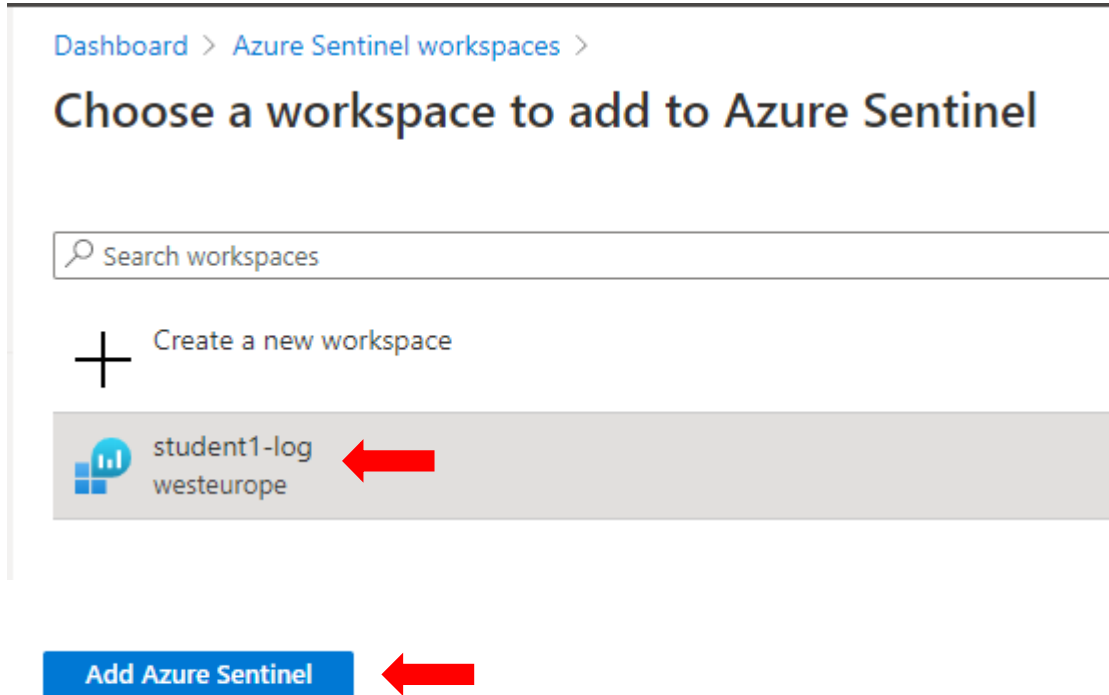
A Log Analytics workspace is a unique environment for Azure Monitor log data. Each workspace has its own data repository and configuration, and data sources and solutions are configured to store their data in a particular workspace.

This object has been created with the deployment of the Azure infrastructure using Terraform. To verify go to Visual Studio Code and checkout main.tf. Otherwise you need to create it first to be able to get the next step.

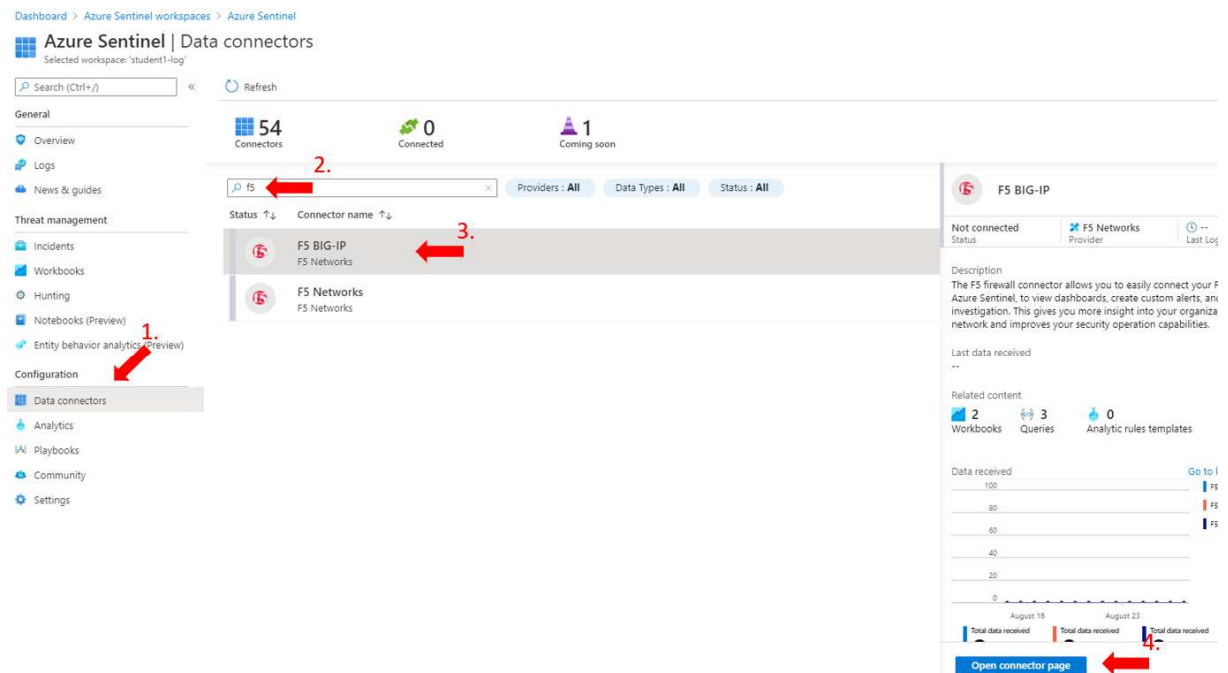
4. In Azure portal, use the search field at the top of the page to type sentinel and select 'Azure Sentinel'



5. In Azure Sentinel workspaces click **+ Add** and select your Log Analytics workspace and press 'Add Azure Sentinel'



6. Once Azure Sentinel is created select 'Data connectors' and type 'f5' in the search field and hit enter.



Select the top F5 BIG-IP and click 'Open connector page'

7. Here we are going to copy the Workspace ID and Primary Key and paste it into our TS declaration

F5 BIG-IP

F5 BIG-IP

Not connected

Status

F5 Networks

Provider

⌚ --

Last Log Received

Description

The F5 firewall connector allows you to easily connect your F5 logs with Azure Sentinel, to view dashboards, create custom alerts, and improve investigation. This gives you more insight into your organization's network and improves your security operation capabilities.

Last data received

--

Related content

2

Workbooks

3

Queries

0

Analytic rules templates

Data received

100

80

60

40

20

0

Go to log analytics

F5TELEMETRY...

F5TELEMETRY...

F5TELEMETRY...

August 16

August 23

Total data received

0

Total data received

0

Total data received

0

Data types

F5Telemetry_LTM_CL --

F5Telemetry_system_CL --

F5Telemetry_ASM_CL --

Instructions Next steps



Prerequisites

To integrate with F5 BIG-IP make sure you have:

- ✓ **Workspace:** read and write permissions are required.
- ✓ **Keys:** read permissions to shared keys for the workspace are required. [See the documentation](#)



Configuration

Configure and connect F5 BIGIP

To connect your F5 BIGIP, you have to post a JSON declaration to the system's API endpoint. For inst [Sentinel](#).

Workspace ID

67130d71-bca3-499e-b7a9-d2915de07dc9

Primary Key

s/c+LKO42yGLAKT4s5ruXm0DSEYWLWdbmjnyVVVLqWMp/H51iPbNY+CluIOr+...

- Open in Postman "Step 5.1.1: Deploy Telemetry Streaming with Azure Sentinel" and copy and paste the Workspace ID and Primary Key into the body of the Telemetry Streaming declaration.

```
{
  "class": "Telemetry",
  "controls": {
    "class": "Controls",
    "logLevel": "debug",
    "debug": true
  },
  "My_System": {
    "class": "Telemetry_System",
    "systemPoller": {
      "interval": 60
    }
  },
  "My_Listener": {
    "class": "Telemetry_Listener",
    "port": 6514
  },
  "My_Consumer": {
    "class": "Telemetry_Consumer",
    "type": "Azure_Log_Analytics",
    "workspaceId": "<Workspace_ID>",
    "passphrase": {
      "cipherText": "<Primary_Key>"
    },
    "useManagedIdentity": false,
  }
}
```

```

    "region": "westeurope"
  }
}

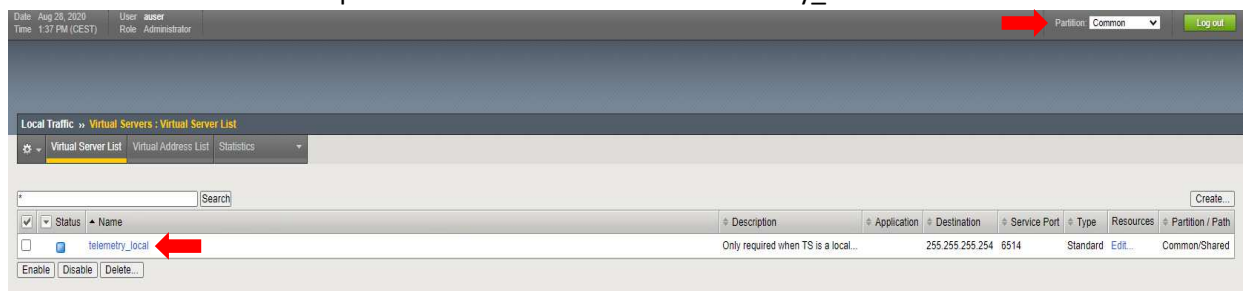
```

9. Press 'Send' and check if the response returns a 200 OK.

At this point we have defined that a TS declaration should deliver its logs to Azure Sentinel, but we did not define an event listener and also didn't define a log profile. Let's use AS3 to complete the task.

10. In Postman select "Step 4.1.2: Configure Event listener" and scroll through its Body and notice the following:
 - Virtual address used is 255.255.255.254
 - Port 6514 was already used in the TS declaration as the telemetry listener.
 - The Telemetry local rule is defined and only used when TS is a local listener.
 - The definition of a telemetry traffic log profile which includes a request template and makes use of HSL to get the logging delivered to Sentinel data connectors.
 - Includes a security log profile which uses Splunk
11. Hit 'Send' and watch the response returning a 200 OK.
12. Check the BIG-IP to understand what has been deployed.

This time we need to check partition Common to find VS 'telemetry_local'



13. Go to Local Traffic > Profiles > Other > Request Logging and select "telemetry_traffic_log_profile". Check the configured profile and recognize the declared template.
14. Go to Security > Event Logs > Logging Profiles and select telemetry_security_log_profile. Check to log profile and recognize the logging format definition and used server address.

(Optional)

Go to System > Logs > Configuration > Log Destinations and checkout 'telemetry_formatted' and 'telemetry_hsl'

Finally go to System > Logs > Configuration > telemetry_publisher and checkout telemetry_publisher.

Task 5.2 – Include WAF via an AS3 declaration

1. Log into the BIG-IP and check:
 - The partitions section if besides partition Common, no other partition exists.
 - The Local Traffic section that no VS, pool or poolmember exists.
 - The Security section that no WAF policy exists.

2. Go through below JSON schema and change the B-octet of each IP address to align with your student number.

```
{
  "class": "AS3",
  "action": "deploy",
  "declaration": {
    "class": "ADC",
    "schemaVersion": "3.7.0",
    "id": "Deploy_WAF_Service_HTTPS",
    "label": "Deploy_WAF_Service_HTTPS",
    "remark": "Deploy_WAF_Service_HTTPS",
    "App_Services": {
      "class": "Tenant",
      "HTTPS_Service": {
        "class": "Application",
        "template": "https",
        "serviceMain": {
          "class": "Service_HTTPS",
          "virtualAddresses": [
            "{{bigip_ext_priv_vippip1}}"
          ],
          "securityLogProfiles": [{
            "bigip": "/Common/Log all requests"
          },
          {
            "bigip": "/Common/Shared/telemetry_security_log_profile"
          }
        ],
        "snat": "auto",
        "pool": "web-pool",
        "profileHTTP": "basic",
        "serverTLS": "webtls",
        "policyWAF": {
          "use": "my_waf_policy"
        },
        "profileTrafficLog": {
          "bigip": "/Common/Shared/telemetry_traffic_log_profile"
        }
      },
      "web-pool": {
        "class": "Pool",
        "monitors": [
          "icmp"
        ],
        "members": [{
          "servicePort": 80,
          "serverAddresses": [
            "{{dwa_server}}"
          ]
        }
      ]
    }
  }
}
```

```





    ]
  },
  "webtls": {
    "class": "TLS_Server",
    "certificates": [{
      "certificate": "webcert"
    }]
  },
  "webcert": {
    "class": "Certificate",
    "certificate": {
      "bigip": "/Common/default.crt"
    },
    "privateKey": {
      "bigip": "/Common/default.key"
    }
  },
  "my_waf_policy": {
    "class": "WAF_Policy",
    "url": "https://raw.githubusercontent.com/f5devcentral/f5-asm-policy-templates/master/owasp_ready_template/owasp-no-auto-tune-v1.1.xml",
    "ignoreChanges": true
  }
}
}
}
}
}

```

Notice the following:

- The AS3 declaration makes use of Service_HTTPS.
- Check the WAF policy and understand that it gets pulled from an external source.

3. In Postman, select “Step 5.2: Deploy HTTP_Service with WAF” press ‘Send’.
4. Wait for the 200 OK and go into the BIG-IP to check the config.
Be sure to select the right tenant or you won’t see any configuration.
Go to Local Traffic > serviceMain and check the used Request Logging Profile

Classification Profile	None  Note: Changes
NAT64	<input type="checkbox"/> Enabled
Request Logging Profile	telemetry_traffic_log_profile  
VS Score	0
Immediate Action On Service Down	None 

5. Next, in the virtual server section, select the tab 'Security' and click 'Policies'.

Local Traffic » Virtual Servers : Virtual Server List » serviceMain

Properties Resources **Security** Statistics

Policy Settings

Destination	10.1.1.20:443
Service	HTTPS
Application Security Policy	Enabled... Policy: my_waf_policy
Service Policy	None
IP Intelligence	Disabled
DoS Protection Profile	Disabled
Bot Defense Profile	Disabled
Application Cloud Security Services	Disabled
Log Profile	Enabled... Selected: /Common Log all requests /Common/Shared telemetry_security_log_profile Available: /Common Log illegal requests global-network local-bot-defense local-dos

Be sure you see the WAF policy enabled and the log profile 'Log all requests' and 'telemetry_security_log_profile' has been Selected. Optional, you can check the WAF policy in the security section.

6. Test the website by using the following URL: http://<BIG-IP_external_public_vip_address>. You will get redirected to HTTPS and ignore the warning.
7. Login to the DVWA server with the credentials admin/password and do the following actions:
- Click 'Create / Reset Database'.
 - Login again.
 - From the left menu, select 'Instructions'.
 - From the left menu, select 'SQL Injection' and
 - o fill in User ID '10' press 'Submit',
 - o fill in User ID '20' press 'Submit',
 - o fill in User ID '10 OR 1=1' and press 'Submit'. This action was rejected, click 'Go Back'
 - From the left menu select 'PHP Info', click 'Go Back'.
 - Finally click 'About'.

All we just did was generating some traffic to get or logs filled.

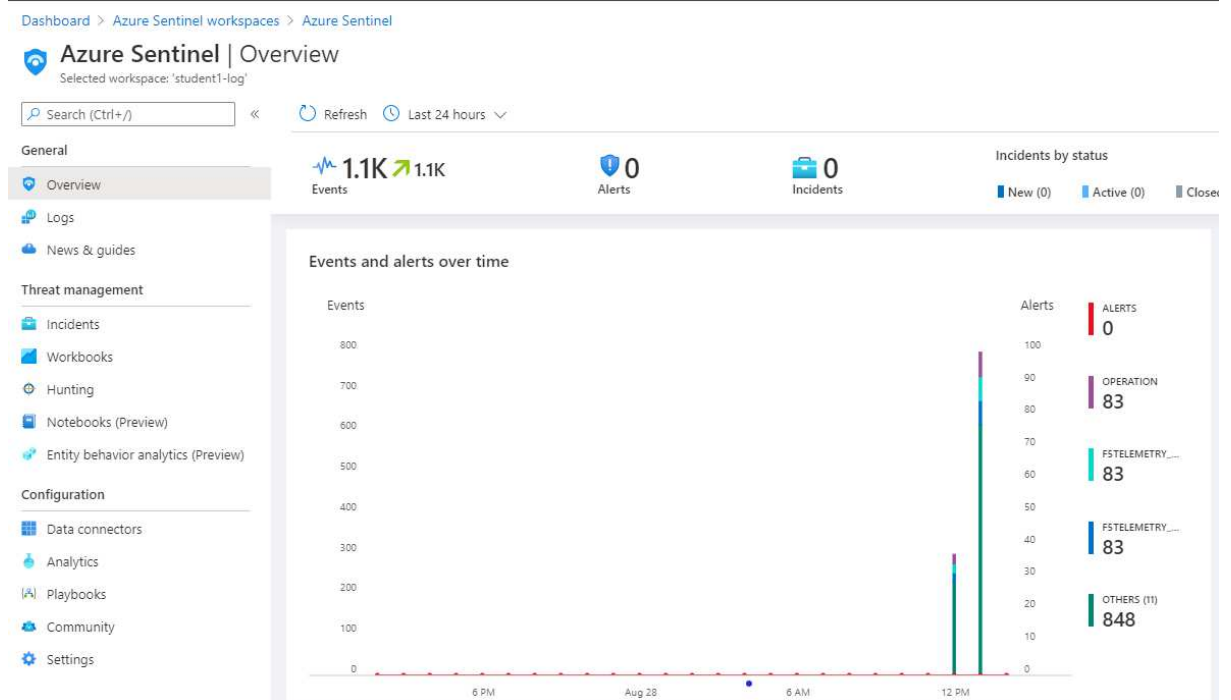
8. Go back to the BIG-IP and check Security > Event Logs and you should see something similar as below.

The screenshot displays the Azure Security Center interface. On the left, a sidebar shows navigation options like Local Traffic, Acceleration, Device Management, Shared Objects, Security, and Network. The main area shows a list of triggered violations. One violation is highlighted, showing details for an SQL injection attack. The 'Request Details' section includes information about the source IP (Netherlands), session ID, and the decoded request body, which contains a malicious payload targeting a vulnerable application. The 'Decoded Request' section shows the raw HTTP request, including headers like User-Agent, Accept, and cookies.

At this point it is proven that we have a working setup where a WAF policy is protecting a DVWA server from being compromised.

Task 5.3 – Azure Sentinel

1. Go back to the Azure Portal and search Azure Sentinel in the upper search field.
2. Select your Log Analytics workspace and you will be in the Sentinel Overview and you should already see some events and alerts.



3. From the overview. Select 'Logs', close the opened window 'Example queries' and select 'Custom Logs'.

Dashboard > Azure Sentinel workspaces > Azure Sentinel

Azure Sentinel | Logs

Selected workspace: 'student1-log'

Search (Ctrl+/)

New Query 1*

student1-log

Tables Queries Filter

Search

Group by: Solution Filters: not selected

Favorites

You can add favorites by clicking on the star icon

Azure Sentinel

LogManagement

Custom Logs

- FSTelemetry_ASM_CL
- FSTelemetry_clientSslProfi...
- FSTelemetry_deviceGroup...
- FSTelemetry_httpProfiles...
- FSTelemetry_iRules_CL
- FSTelemetry_LTM_CL
- FSTelemetry_ItemPolicies_CL
- FSTelemetry_networkTunn...
- FSTelemetry_poolCL
- FSTelemetry_sslCerts_CL
- FSTelemetry_system_CL
- FSTelemetry_telemetryEve...
- FSTelemetry_telemetrySer...
- FSTelemetry_virtualServer...

Run Time range: Set in query Save Copy link New alert rule Export Pin to dashboard Format query

```
1 FSTelemetry_ASM_CL
2 where TimeGenerated > ago(24h)
3 limit 10
```

Results Chart Columns Add bookmark Display time (UTC+00:00) Group columns

Completed 00:00:00.304 10 records

TimeGenerated [UTC]	response_s	hostname_s	management_ip_address_s	management_ip_address_2_s	http_class_n
8/28/2020, 2:05:49.263 PM		bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:01:33.189 PM	Response logging disabled	bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:01:35.773 PM		bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:01:40.906 PM		bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:01:53.385 PM	Response logging disabled	bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:01:39.245 PM	Response logging disabled	bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:01:42.893 PM		bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:00:32.634 PM	Response logging disabled	bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:00:41.102 PM		bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic
8/28/2020, 2:00:42.897 PM		bigip1.westeurope.azure.local	10.1.0.10	N/A	/App_Servic

This is the point where you can check if all required logs are actually being received.

4. Select 'Workbooks' and in the search field type 'f5' and select 'F5 BIG-IP ASM' and click 'Save' and when asked for the location leave it at its default of 'West Europe' and hit OK.

Dashboard > Azure Sentinel workspaces > Azure Sentinel

Azure Sentinel | Workbooks

Selected workspace: 'student1-log'

Search (Ctrl+/)

Refresh Add workbook

0 Saved workbooks 71 Templates 0 Updates

My workbooks Templates

Search f5

F5 BIG-IP ASM F5 NETWORKS

F5 BIG-IP System Metrics F5 NETWORKS

F5 BIG-IP ASM F5 NETWORKS

Gain insights into F5 BIG-IP Application Security Manager (ASM), by analyzing traffic and activities. This workbook provides insight into F5's web application firewall events and identifies attack patterns across multiple ASM instances as well as overall BIG-IP health.

Required data types: ☐ FSTelemetry_LTM_CL ☒ FSTelemetry_system_CL ☒ FSTelemetry_ASM_CL

Relevant data connectors: ☐ F5Bigip

View template Save

5. View the saved workbook and when prompted for missing required data types, hit OK.
6. Your F5 BIG-IP ASM Insights should look like this.

Dashboard > Azure Sentinel workspaces > Azure Sentinel

F5 BIG-IP ASM - student1-log

student1-log

Edit

F5 BIG-IP ASM Insights

TimeRange: Last 28 days

HostName events

bigip1.westeurope.azure... All 23 23

The number of shown events might be different.

7. Scroll down though the dashboard to see more events. You should seem more events and even some violations.

Voila, your F5 BIG-IP ASM integration in Azure Sentinel is finished.

8. (Optional) repeat creating more logging by hitting the DVWA server with more traffic to see more dashboard movement.