API Mgmt and Security Lab

WW SSA

Nov 10, 2020

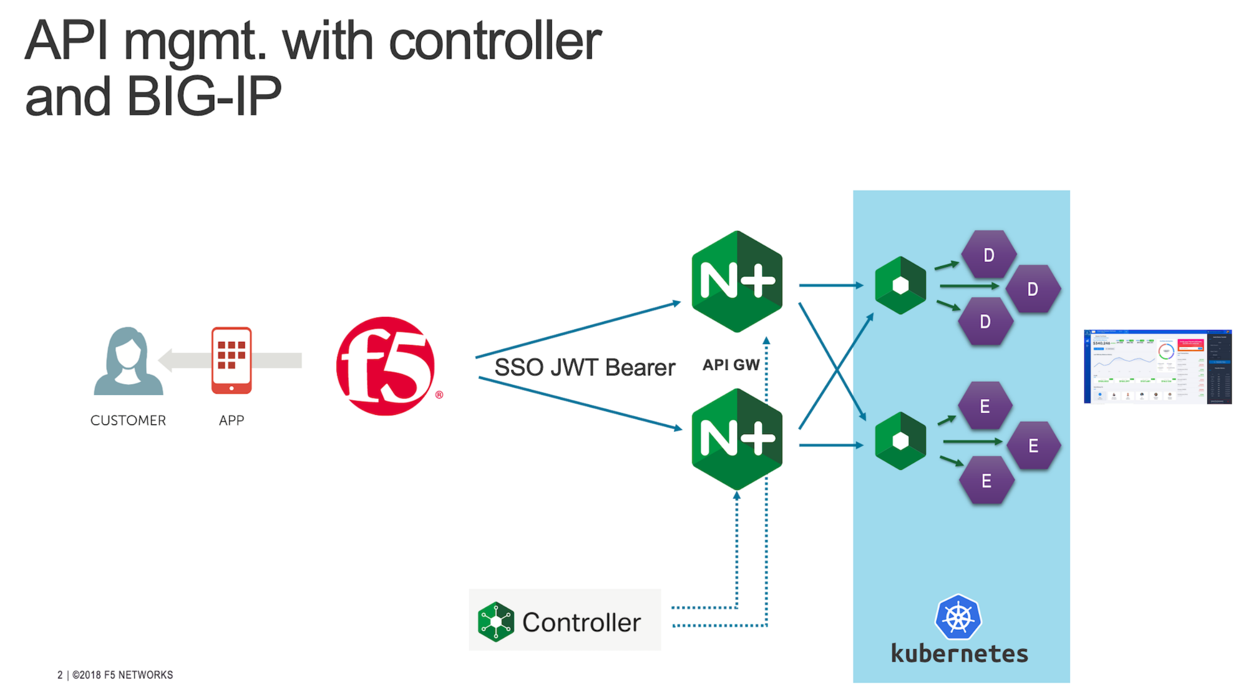
CONTENTS:

CHAPTER

ONE

PUBLISH AND PROTECT MODERN APPLICATIONS

Warning: For any remark or mistake in this lab, please reach out Matthieu DIERICK or fork this repo and fix it with a merge request.



1.1 Class 1 - Understand the infrastructure and the workflow

Welcome into the NGINX Controller 3.x with BIG-IP Lab

Warning: For any remark or mistake in this lab, please reach out Matthieu DIERICK or fork this repo and fix it with a merge request.

Note: The video below will explain you how Arcadia Finance application works and is structured. It is important to understand this part before configuring the lab. In the next section, we present the same if you don't want to watch the video.

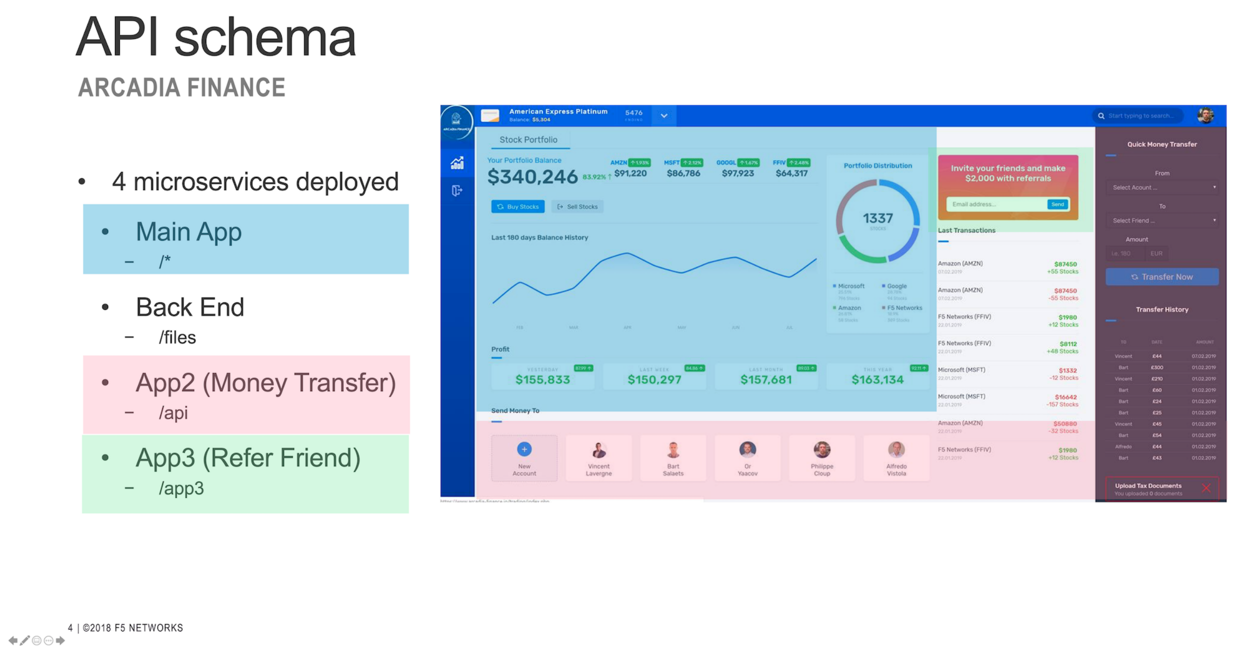
1.1.1 Architecture of Arcadia Application

Note: We will use the famous Arcadia Finance application in this lab. This application is based on 4 microservices. You can find below the different IP addresses and Ports used by NGINX and BIG-IP.

Note: This application is available in GitLab in case you want to build your own lab : https://gitlab.com/arcadia-application

First of all, it is important to understand how Arcadia app is split between microservices

This is how Arcadia App looks like when the 4 microservices are up and running, and you can notice how traffic is routed based on URI



API schema

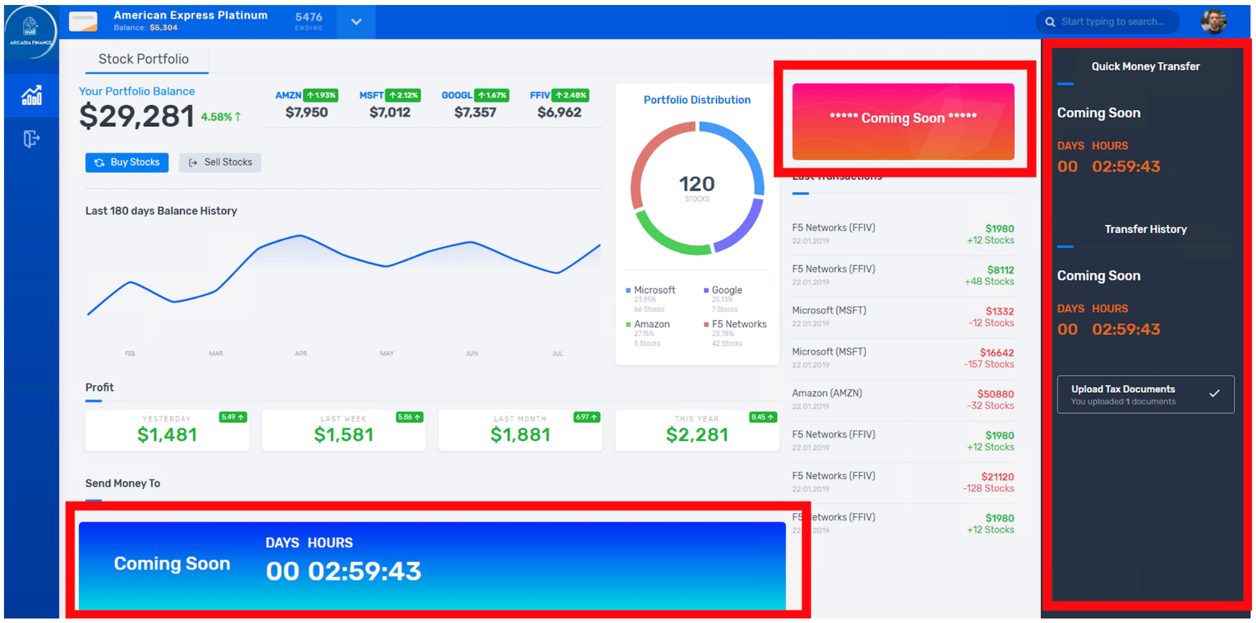
ARCADIA FINANCE

- 4 microservices deployed
  - Main App
    - /*
  - Back End
    - /files
  - App2 (Money Transfer)
    - /api
  - App3 (Refer Friend)
    - /app3

4 | ©2018 F5 NETWORKS

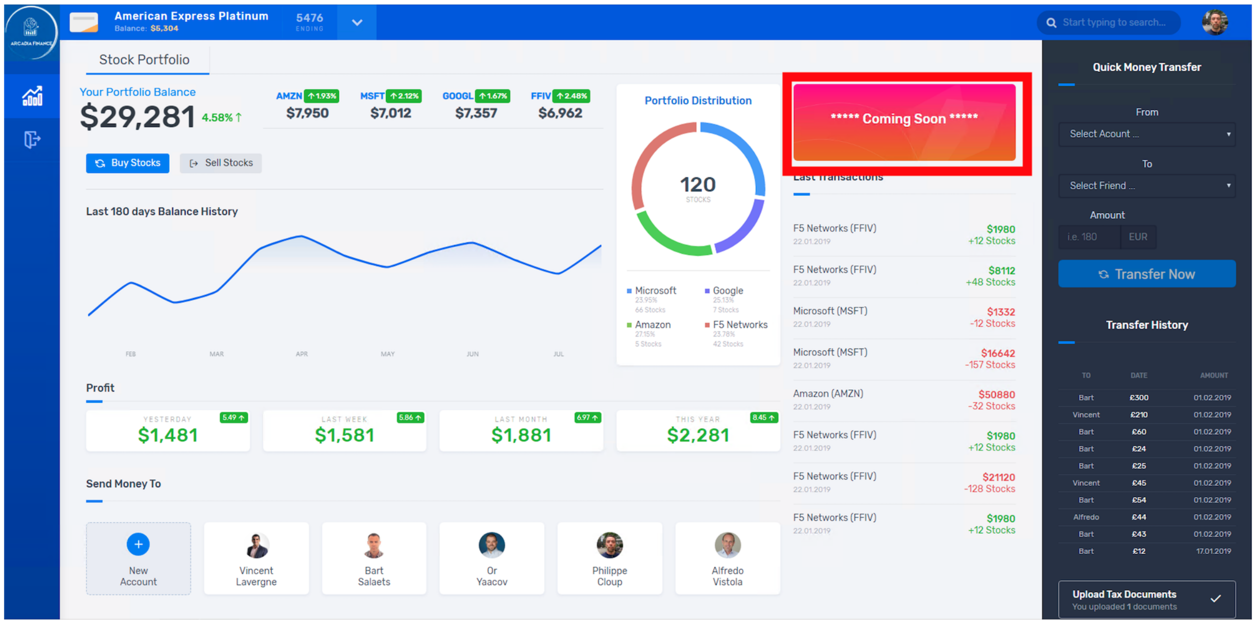
But you can deploy Arcadia Step by Step

If you deploy only `Main App` and `Back End` services.

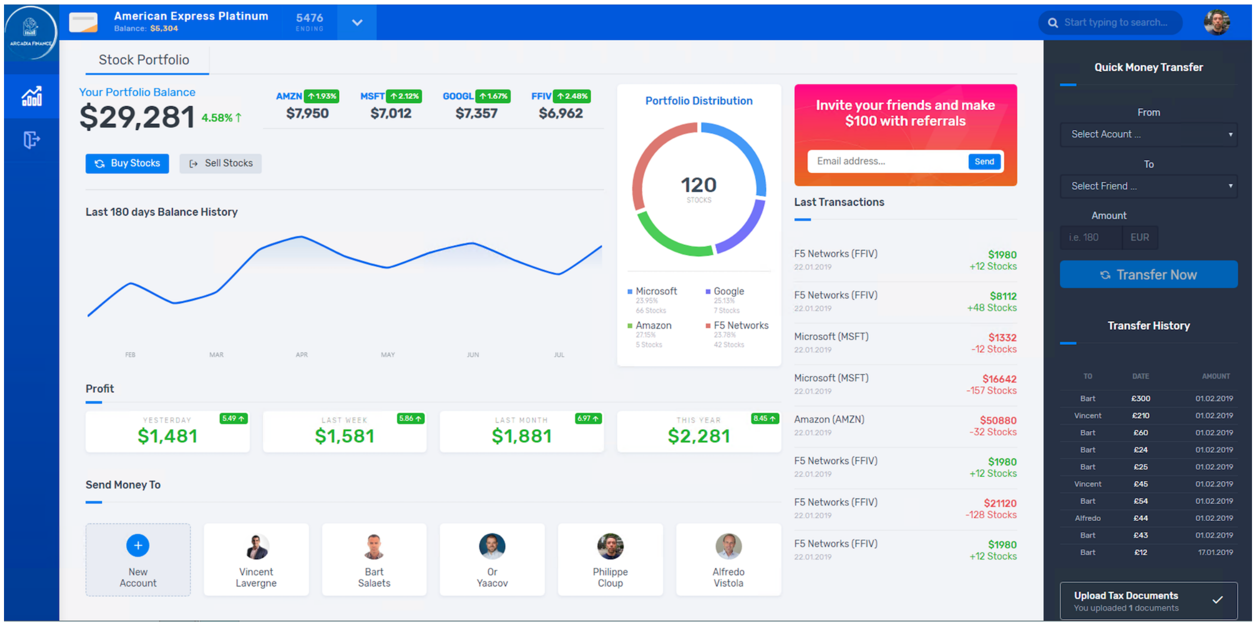


Note: You can see App2 (Money Transfer) and App3 (Refer Friend) are not available. There is dynamic content showing a WARNING instead of a 404 or blank frame.

If you deploy `Main App`, `Back End` and `Money Tranfer` services.



If you deploy `Main App`, `Back End`, `Money Tranfer` and `Refer Friend` services.



Last Transactions

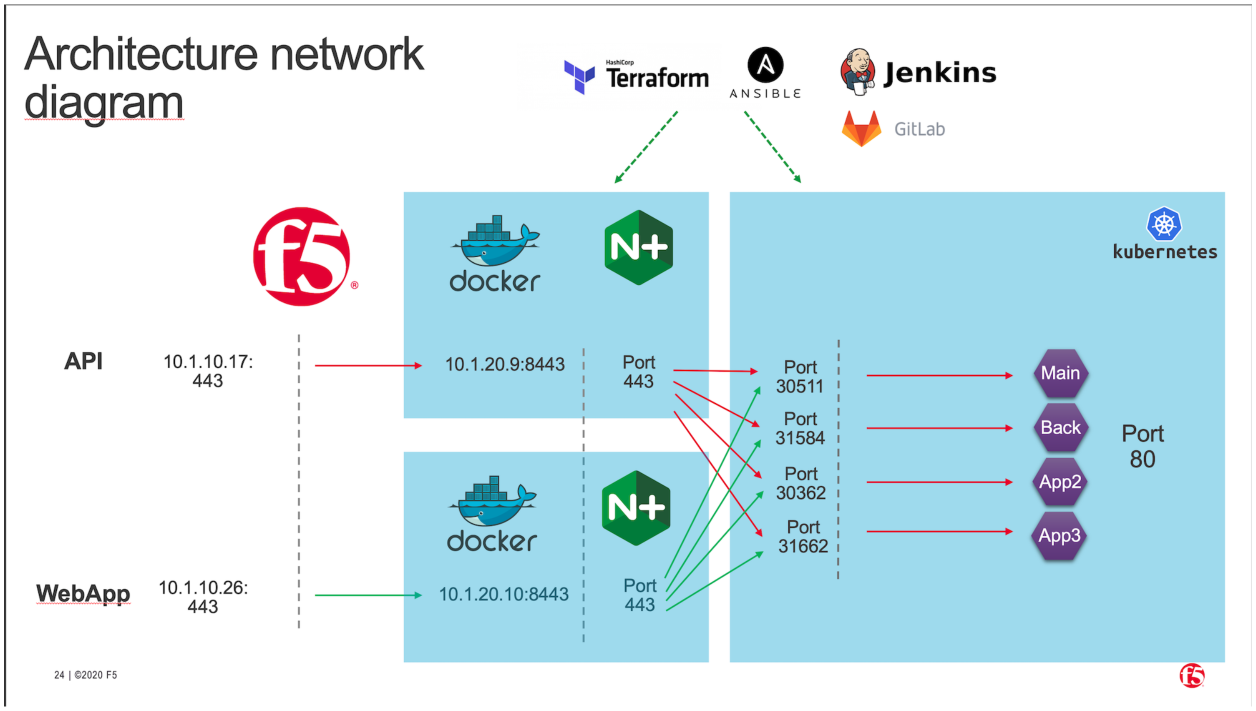
F5 Networks (FFIV)	22.01.2019	$1980 +12 Stocks
F5 Networks (FFIV)	22.01.2019	$8112 +48 Stocks
Microsoft (MSFT)	22.01.2019	$1332 -12 Stocks
Microsoft (MSFT)	22.01.2019	$16642 -157 Stocks
Amazon (AMZN)	22.01.2019	$50880 -32 Stocks
F5 Networks (FFIV)	22.01.2019	$1980 +12 Stocks
F5 Networks (FFIV)	22.01.2019	$21120 -128 Stocks
F5 Networks (FFIV)	22.01.2019	$1980 +12 Stocks

Transfer History

TO	DATE	AMOUNT
Bart	£300	01.02.2019
Vincent	£210	01.02.2019
Bart	£60	01.02.2019
Bart	£24	01.02.2019
Bart	£25	01.02.2019
Vincent	£45	01.02.2019
Bart	£54	01.02.2019
Alfredo	£44	01.02.2019
Bart	£43	01.02.2019
Bart	£12	17.01.2019

The diagram belows show the IP addresses and the ports used for all the routes

Note: For a lab standpoints, these IP addresses and ports does not change. But in a real life, they are dynamic.



1.1.2 Workflow of the demo

The demo is split in 3 classes

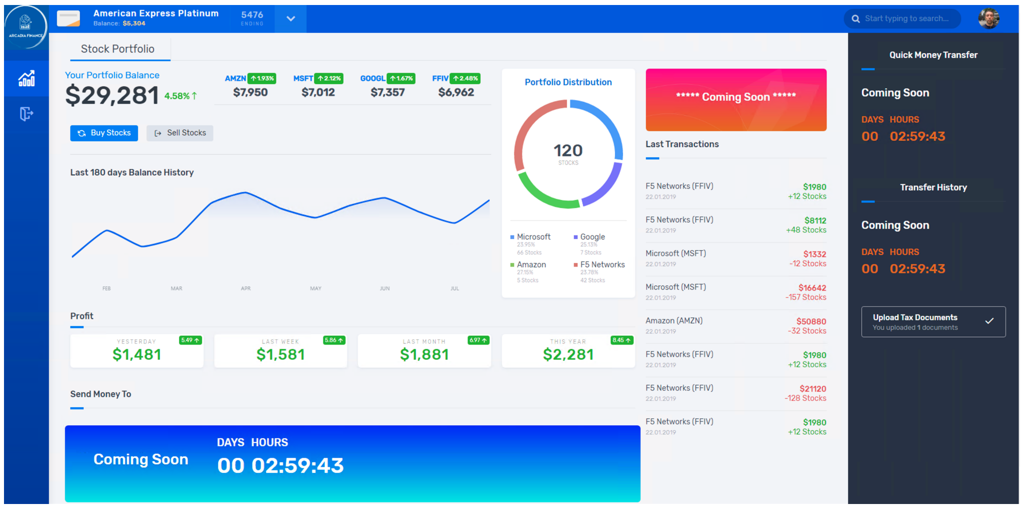
- **Deploy, publish and protect Arcadia Web application**
  - Deploy and publish Arcadia Main App
  - Deploy and publish Money Transfer App
  - Deploy and publish Refer Friends App
  - Apply WAF policy
- **Publish and protect Arcadia API**
  - Publish the API using an OpenAPI 3.0 spec file
  - Protect the API with Advanced WAF and APM using OpenAPI 3.0 spec file
  - Discover the new Developer Portal

Step 1 - DevOps deploy Arcadia application

Note: Goal is to use the GUI in the NGINX Controller for our traditional customers. NetOps will configure the services (MainApp and BackEnd) manually.

Tasks:

1. DevOps commit a new code in GitLab in order to publish a brand new application "Arcadia Bank"
2. **GitLab webhooks this commit and asks Jenkins to run a pipeline. This pipeline:**
   1. Deploy Arcadia application in Kubernetes (Terraform).
   2. Deploy nodeports in Kubernetes (but it could be KIC) (Terraform).
   3. Deploy NGINX+ instances (ADC) in Docker, in front of this K8S cluster (Terraform)
   4. Create Gateways in NGINX Controller for each NGINX+ instance (Ansible)
   5. Deploy AS3 template into front BIGIP to publish publically the application - without WAF (Ansible)
3. NetOps create ADC configuration in NGINX controller in order to "route" traffic to the right K8S service
   1. MainApp `/*` to service MainApp
   2. BackEnd `/file*` to service BackEnd



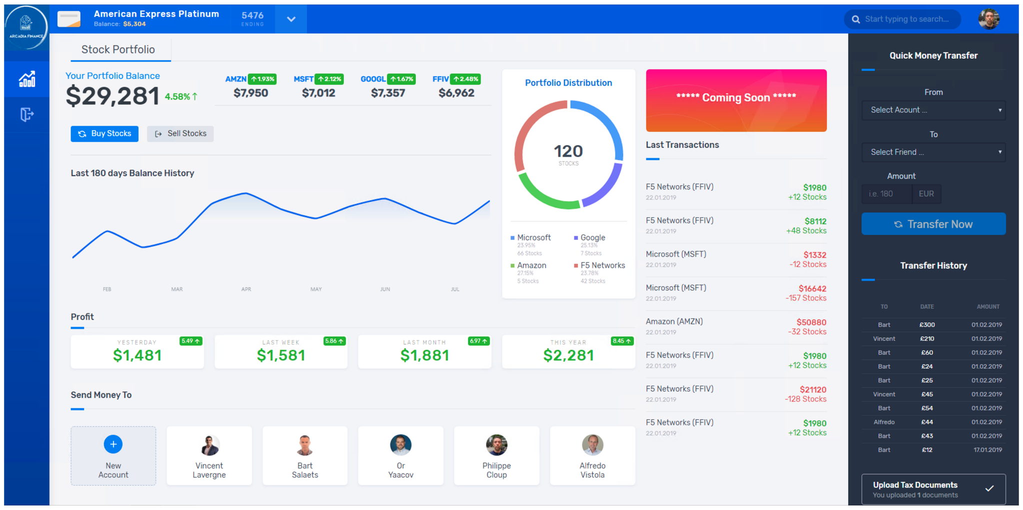
Warning: At this stage, the first part of the application is published and can be accessed and demonstrated. We can see Money Transfert application is not yet there, same for Refer Friends.

Step 2 - DevOps deploy Money Transfer application

Note: Goal is to demonstrate NGINX Controller has a REST API to configure objects. NetOps will configure the service (Money Transfer) via REST API.

Tasks:

1. DevOps commit a new code in GitLab in order to publish the second part of the Arcadia Bank website. This new application allows money transfer between friends.
2. **GitLab webhooks this commit and ask Jenkins to run a pipeline. This pipeline:**
    1. Deploy Money Transfer application in Kubernetes (Terraform)
    2. Deploy nodeports in Kubernetes (Terraform)
3. NetOps use REST API to publish this new app on NGINX+ instances



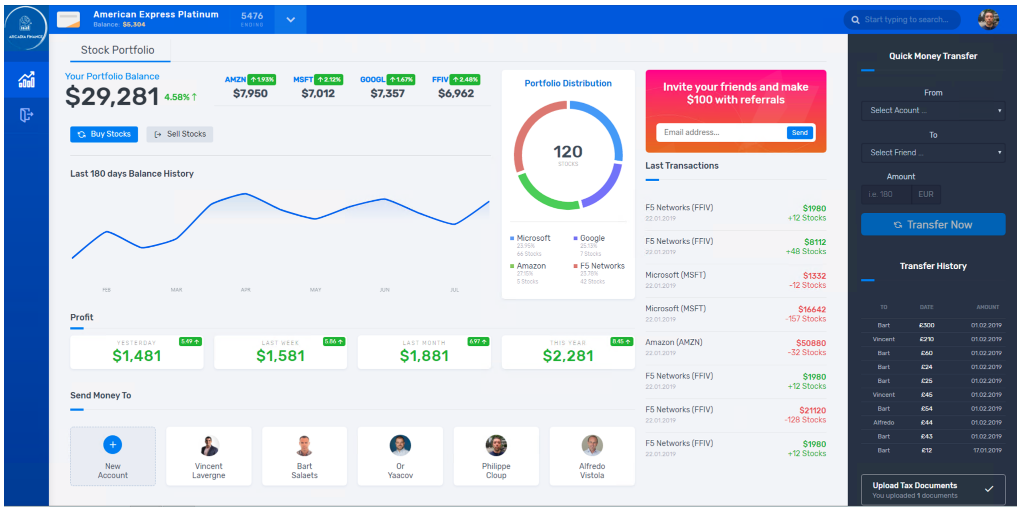
Warning: At this stage, the Money Transfer application is published and can be accessed and demonstrated

Step 3 - DevOps deploy Refer Friends application

Note: Goal is to demonstrate NGINX Controller can be part of the application lifecycle and CICD. NetOps don't configure anything.

Tasks:

1. DevOps commit a new code in GitLab in order to publish the third and last part of the Arcadia Bank website. This new application allow a customer to refer friends with their email address.
2. **GitLab webhooks this commit and ask Jenkins to run a Pipeline. This pipeline:**
    1. Deploy Refer Friends application in Kubernetes (Terraform)
    2. Deploy nodeports in Kubernetes (Terraform)
    3. Configure all components in NGINX Controller (Ansible)



Warning: At this stage, the Refer Friends application is published and can be accessed and demonstrated. The Arcadia Bank website is finished, but not yet secured.

Step 4 - NetOps/SecOps publish WAF policy to protect Arcadia application

Note: Goal is to demonstrate BIG-IP Advanced WAF has a Declarative API interface to push WAF policies.

Task:

1. NetOps run a Jenkins pipeline that will push a new AS3 declaration with a WAF policy built by Secops

Warning: At this stage, the Arcadia Bank website is published and secured.

Step 5 - Publish Arcadia API

Note: Goal is to demonstrate the new Controller capabilties with API management and gateway

Task:

1. DevOps provide with an API specification file (OpenAPI 3.0 - OAS3)
2. NetOps import this file into the Controller APIm and publish the API
3. SecOps import his file into the BIG-IP and protect the API (WAF + Access)
4. Developpers can access the new Developper Portal

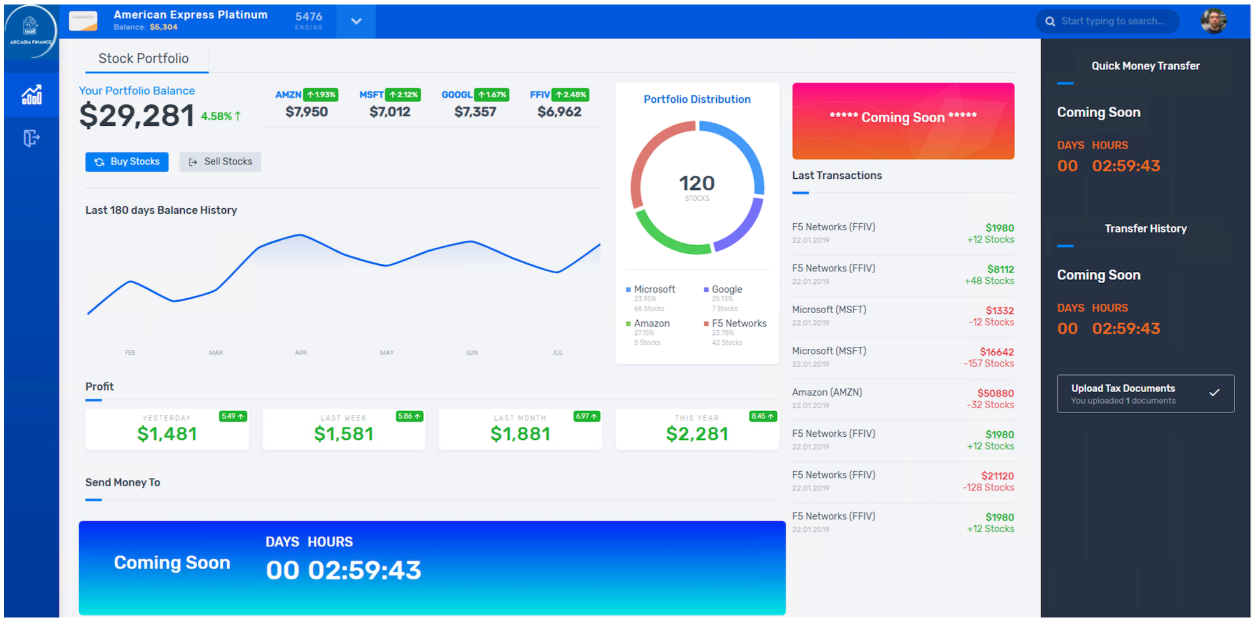
1.2 Class 2 - Deploy, Publish and Protect Arcadia Web Application

In this section, we will deploy, publish and protect Arcadia application

1.2.1 DevOps deploy Arcadia Application - Main app

In this module, we will deploy the 2 main containers for Arcadia Bank application and we will publish them.

Note: At the end of this module, Arcadia Bank application will look like this.



Note: As a DevOps, you will deploy Arcadia Application (main and back end pods) with an automation tool set

---

Step 1 - Deploy Arcadia Main app with a CI/CD pipeline like a DevOps

Connect to Jumhost RDP and Login as user / user

1. Open `Chrome`, you can notice Chrome opens all the tabs for you
2. Login to all tools
   1. Controller : admin@nginx-udf.internal / admin123!
   2. Jenkins : admin / admin
   3. GitLab : root / F5twister$

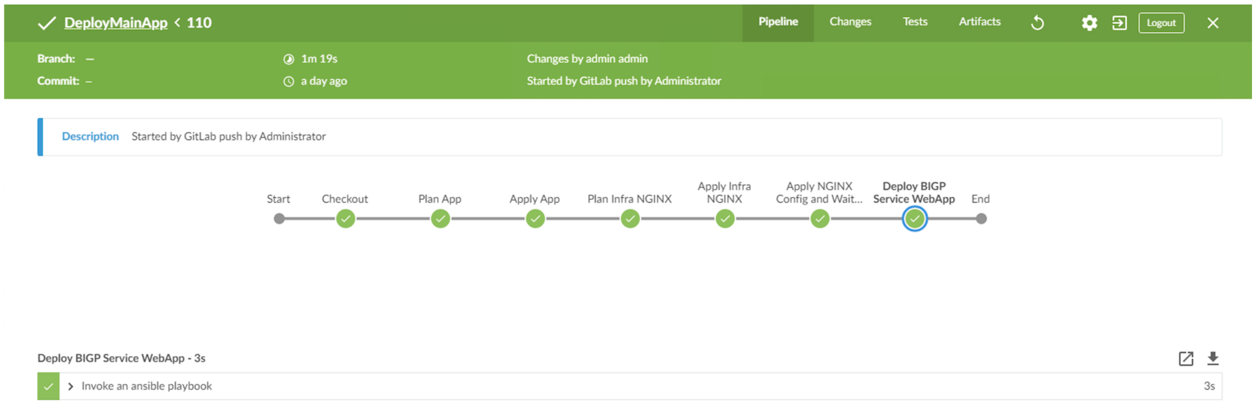
> **Warning:** If GitLab does not start, restart the docker in the GitLab VM (WebSSH > docker restart gitlab). Wait 5 minutes.

4. Kubernetes : click on `skip`
5. BIG-IP : admin / admin

3. In Gitlab, click on Administrator / Arcadia-MainApp
   1. Click on file `deploy`
   2. Click `edit` and make a modification, like `YES !!!!!`
   3. Click `Commit changes`

Note: At this moment, you simulate a commit like a DevOps. This `commit` will trigger a `webhook` to `Jenkins`, so that Jenkins executes a `pipeline`.

1. In Jenkins, click on `DeployMainApp` pipeline
2. A pipeline is running, click on it
3. You can follow the steps



Branch: –
Commit: –
1m 19s
a day ago
Changes by admin admin
Started by GitLab push by Administrator

Description Started by GitLab push by Administrator

Deploy BIGP Service WebApp - 3s

Invoke an ansible playbook 3s

Note: At this stage, Arcadia Main app and Back End app are deployed un K8S. But you need to publish them with NGINX+ via the controller.

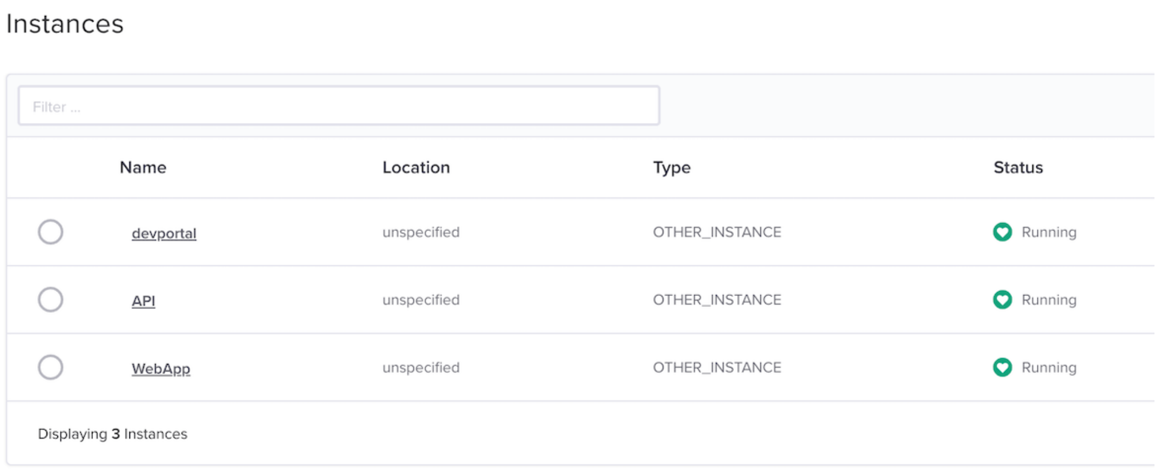
Step 2 - Publish Arcadia application with NGINX+ and Controller

The Jenkins pipeline did several things

1. Deployed Arcadia application (main and back end pods) in Kubernetes
   1. Connect to `Kubernetes` and check that.
   2. You can see 2 deployments (main and back) with nodeports services
2. Started 3 NGINX+ instances in a docker
   1. WebSSH to `CICD and DOCKER (NGINX API gw, Dev Portal)`
   2. Run a `docker ps`

```
ubuntu@ip-10-1-1-9:~$ docker ps
CONTAINER ID        IMAGE                            COMMAND                         ␣
↪CREATED              STATUS               PORTS                                     ␣
↪                      NAMES
bf86e23a9807        nginx-plus:36v1                  "sh /entrypoint.sh"       33␣
↪seconds ago        Up 31 seconds       10.1.20.9:8080->80/tcp, 10.1.20.9:8443->
↪443/tcp                NginxPlusAPI
74d679bdf5fb        nginx-plus:36v1                  "sh /entrypoint.sh"       33␣
↪seconds ago        Up 31 seconds       80/tcp, 10.1.20.12:8090->8090/tcp           ␣
↪                 NginxPlusDevPortal
ac12c0f3148a        nginx-plus:36v1                  "sh /entrypoint.sh"       33␣
↪seconds ago        Up 32 seconds       10.1.20.10:8080->80/tcp, 10.1.20.10:8443->
↪443/tcp                NginxPlusWebApp
ab75d7bd60bb        nginx                            "nginx -g 'daemon of..."   7␣
↪months ago         Up 13 hours          0.0.0.0:80->80/tcp                         ␣
↪                   lab-nginx
35ddc5adc34d        sameersbn/bind:9.11.3-20190706   "/sbin/entrypoint.sh..."   9␣
↪months ago         Up 13 hours          0.0.0.0:53->53/tcp, 0.0.0.0:10000->10000/
↪tcp, 0.0.0.0:53->53/udp   bind
```

3. Check if NGINX+ instances appears in the controller
   1. In the controller GUI, click top `left corner icon`, and `infrastructure`
   2. You can see 3 instances running



Instances

Filter ...

	Name	Location	Type	Status
○	devportal	unspecified	OTHER_INSTANCE	✓ Running
○	API	unspecified	OTHER_INSTANCE	✓ Running
○	WebApp	unspecified	OTHER_INSTANCE	✓ Running

Displaying 3 Instances

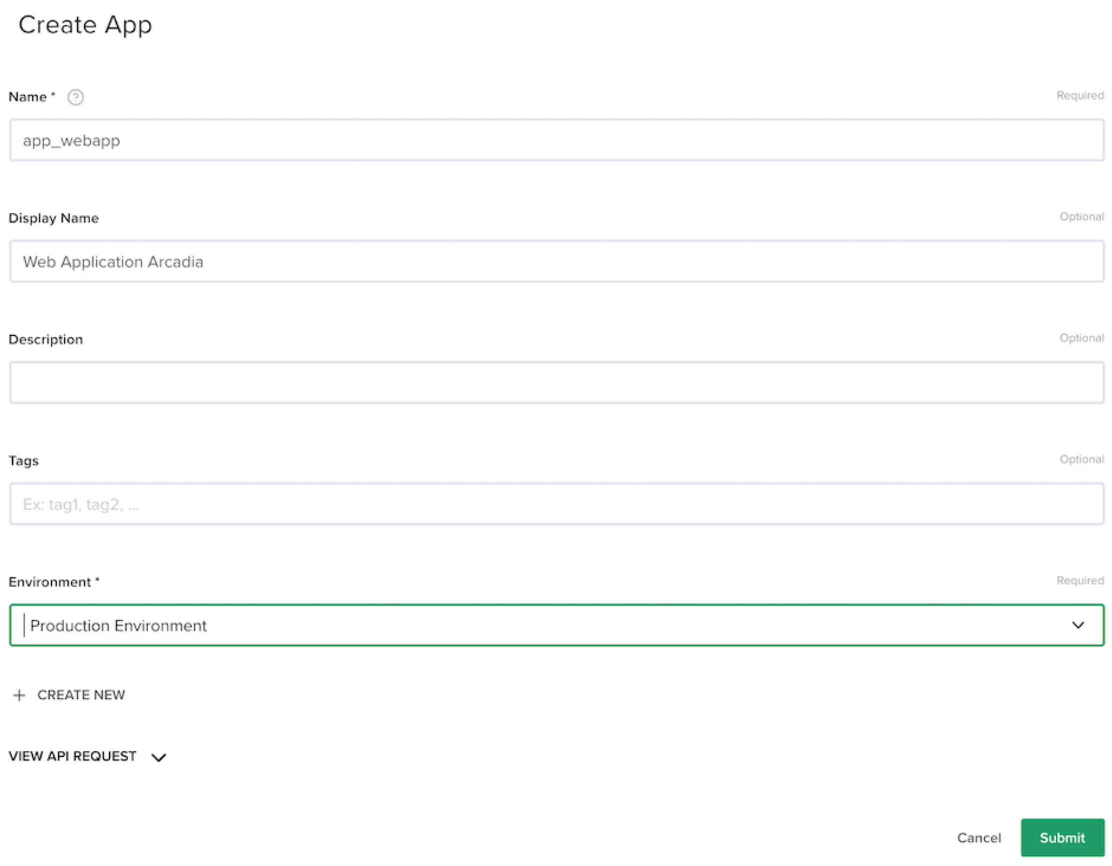
4. Deployed an AS3 declaration into the BIG-IP in order to publish the NGINX+ instance externally

Note: It is time to configure the NGINX+ instances in order to publish Arcadia application (main and back services)

Configure the Controller

Warning: For all the commands below, there are CASE SENSITIVE

1. Connect to the controller (admin@nginx-udf.internal / admin123!)
2. Click on top `left corner icon` and `Services`
3. Click on `Apps` and `create app`
   1. Application name : `app_webapp`
   2. Display name : `Web Application Arcadia`
   3. Environment : `Production Environment`
4. Click `submit`



Create App

Name * — Required: app_webapp

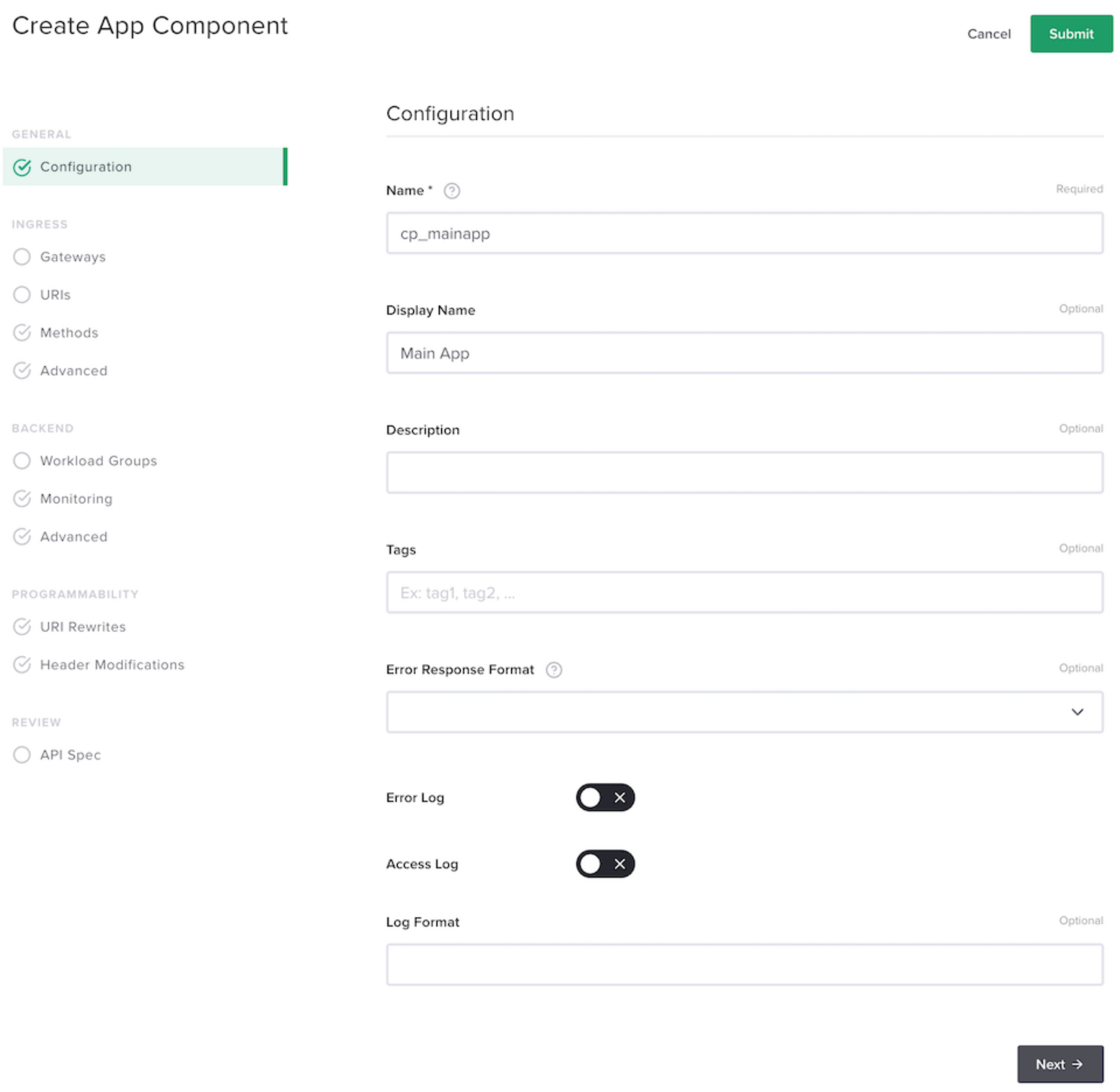
Display Name — Optional: Web Application Arcadia

Description — Optional:

Tags — Optional: Ex: tag1, tag2, ...

Environment * — Required: Production Environment

5. Click on `Create Component`
   1. Configure the component as below



Configuration

Name * (Required): cp_mainapp

Display Name (Optional): Main App

Description (Optional):

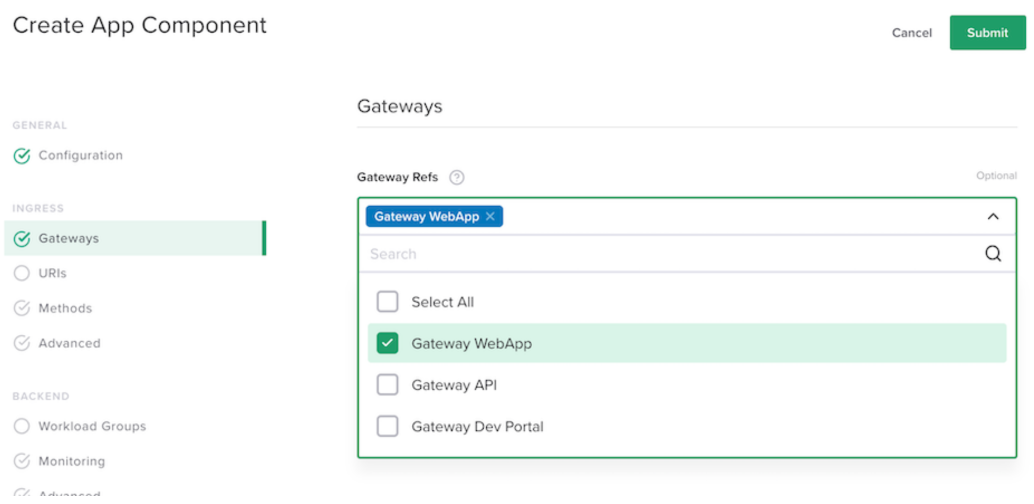
Tags (Optional): Ex: tag1, tag2, ...

Error Response Format (Optional):

Error Log: off

Access Log: off

Log Format (Optional):


Create App Component

Cancel Submit

GENERAL
- Configuration

INGRESS
- Gateways
- URIs
- Methods
- Advanced

BACKEND
- Workload Groups
- Monitoring

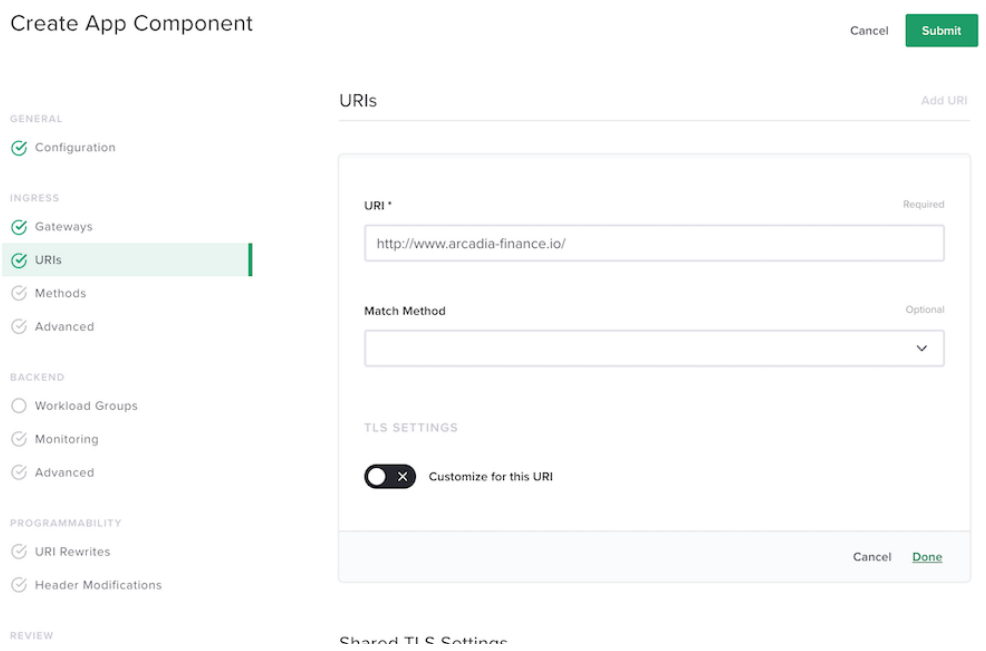
Gateways

Gateway Refs — Optional

Gateway WebApp ×

Search

- [ ] Select All
- [x] Gateway WebApp
- [ ] Gateway API
- [ ] Gateway Dev Portal


Create App Component

Cancel Submit

GENERAL
- Configuration

INGRESS
- Gateways
- URIs
- Methods
- Advanced

BACKEND
- Workload Groups
- Monitoring
- Advanced

PROGRAMMABILITY
- URI Rewrites
- Header Modifications

REVIEW

URIs — Add URI

URI * — Required

http://www.arcadia-finance.io/

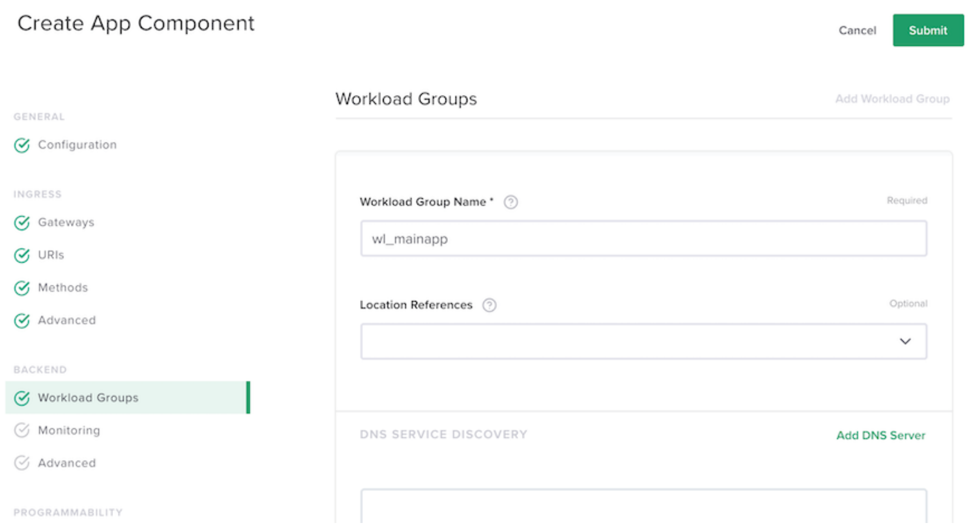
Match Method — Optional

TLS SETTINGS

Customize for this URI

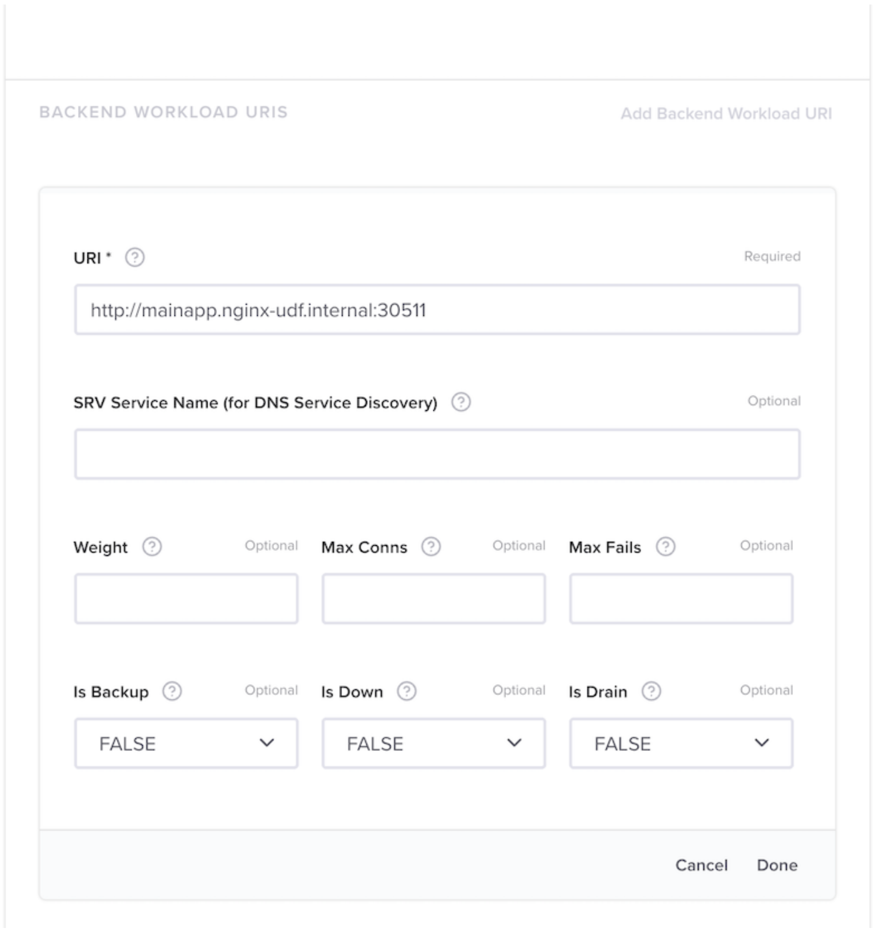
Cancel Done

Warning: Don't forget to click on `done`



Workload Group Name * Required: wl_mainapp

Location References Optional:



URI * Required: http://mainapp.nginx-udf.internal:30511

SRV Service Name (for DNS Service Discovery) Optional:

Weight Optional:

Max Conns Optional:

Max Fails Optional:

Is Backup Optional: FALSE

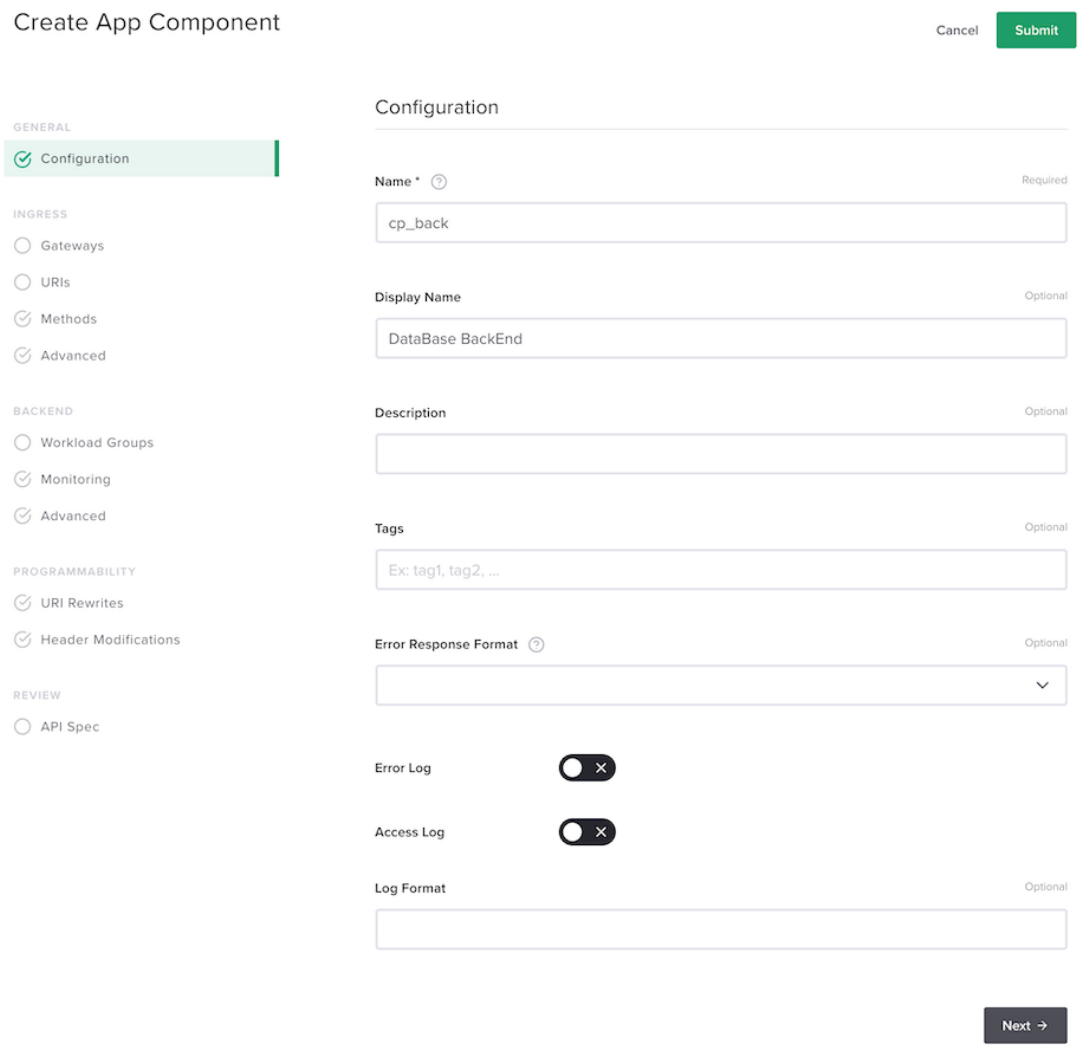
Is Down Optional: FALSE

Is Drain Optional: FALSE

Warning: Don't forget to click on `done` twice

Note: Click `submit`

1. Get back to `Web App` and add a new `Component`
2. Do the same, but for the back end service


Create App Component

Configuration

Name * : cp_back

Display Name: DataBase BackEnd

Description:

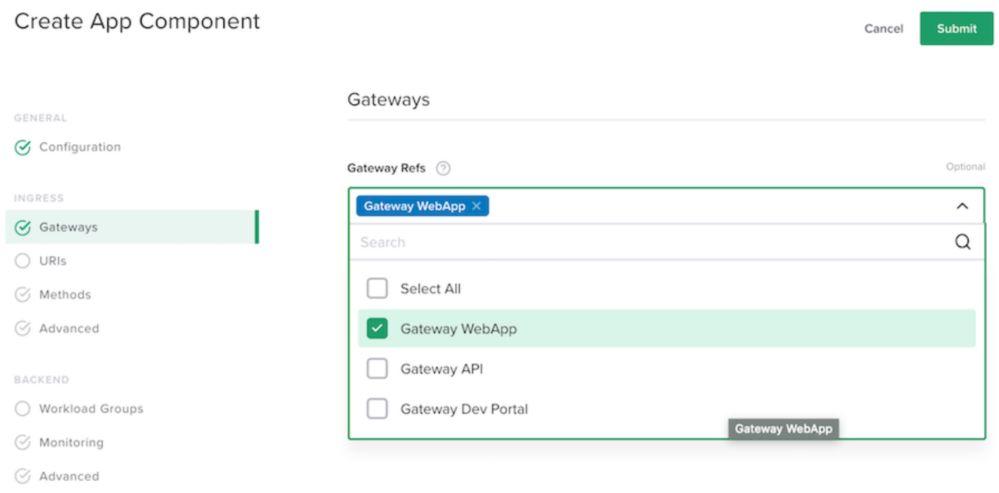
Tags: Ex: tag1, tag2, ...

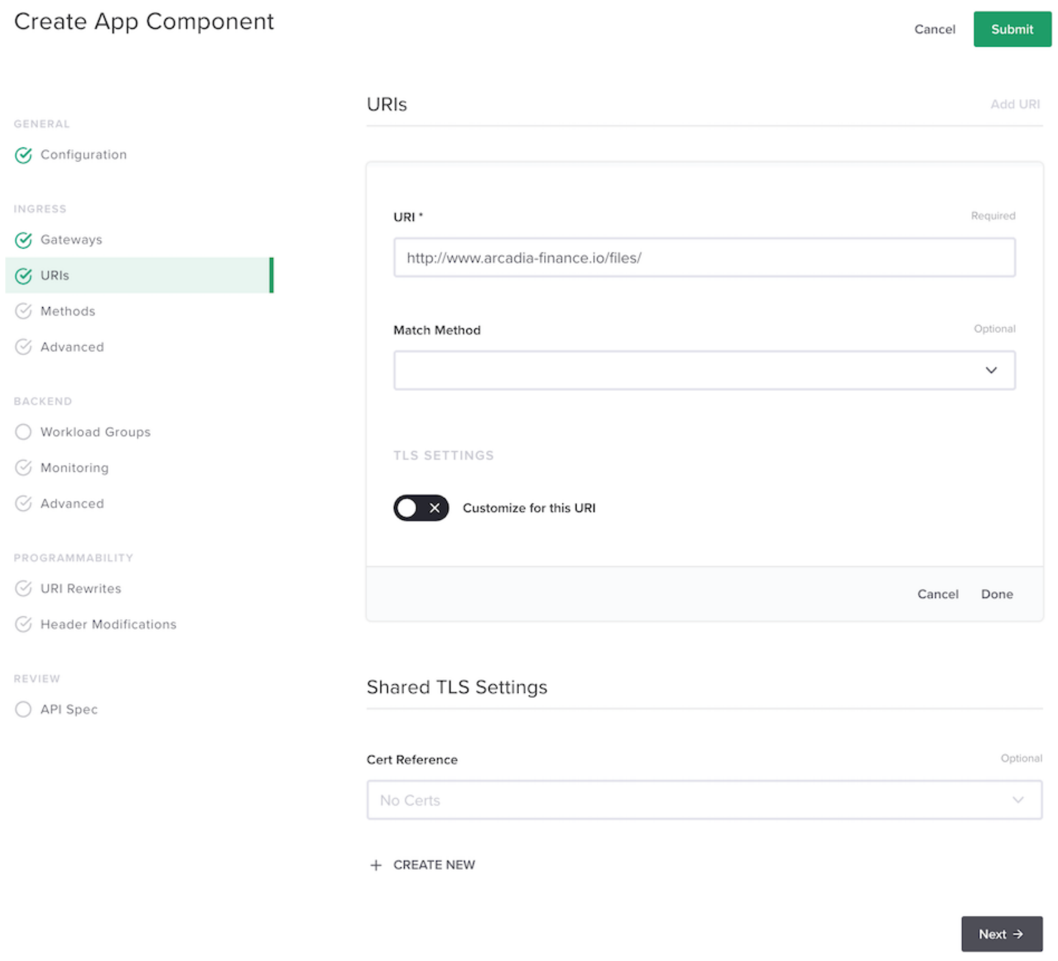
Error Response Format:

Error Log: off

Access Log: off

Log Format:

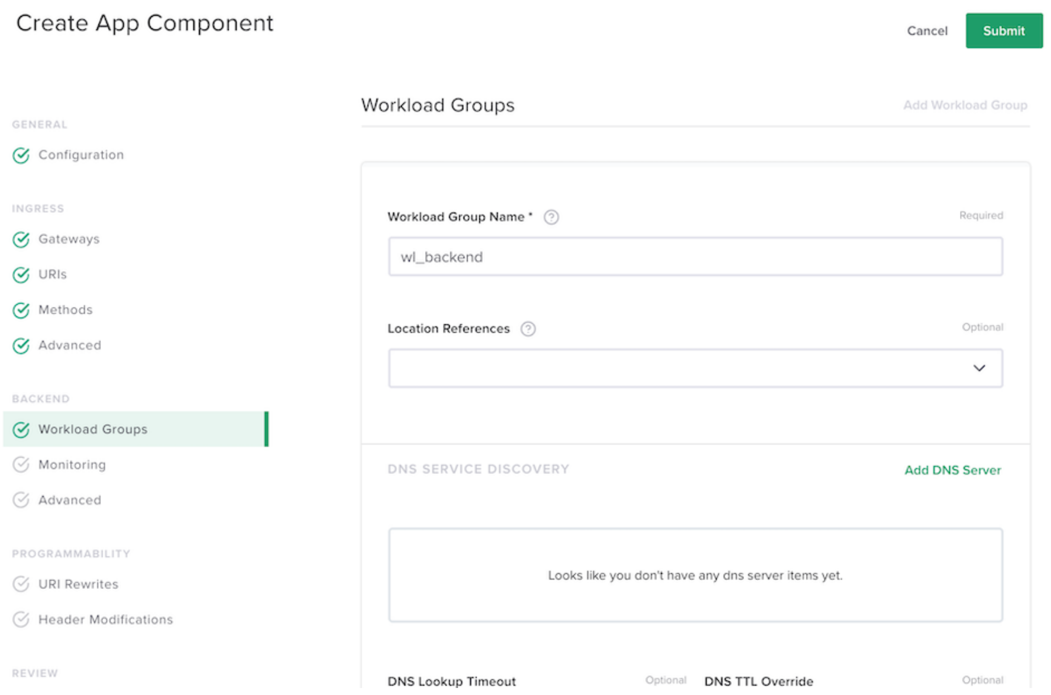




URI *

http://www.arcadia-finance.io/files/

Warning: Don't forget to click on `done`



Workload Group Name * Required

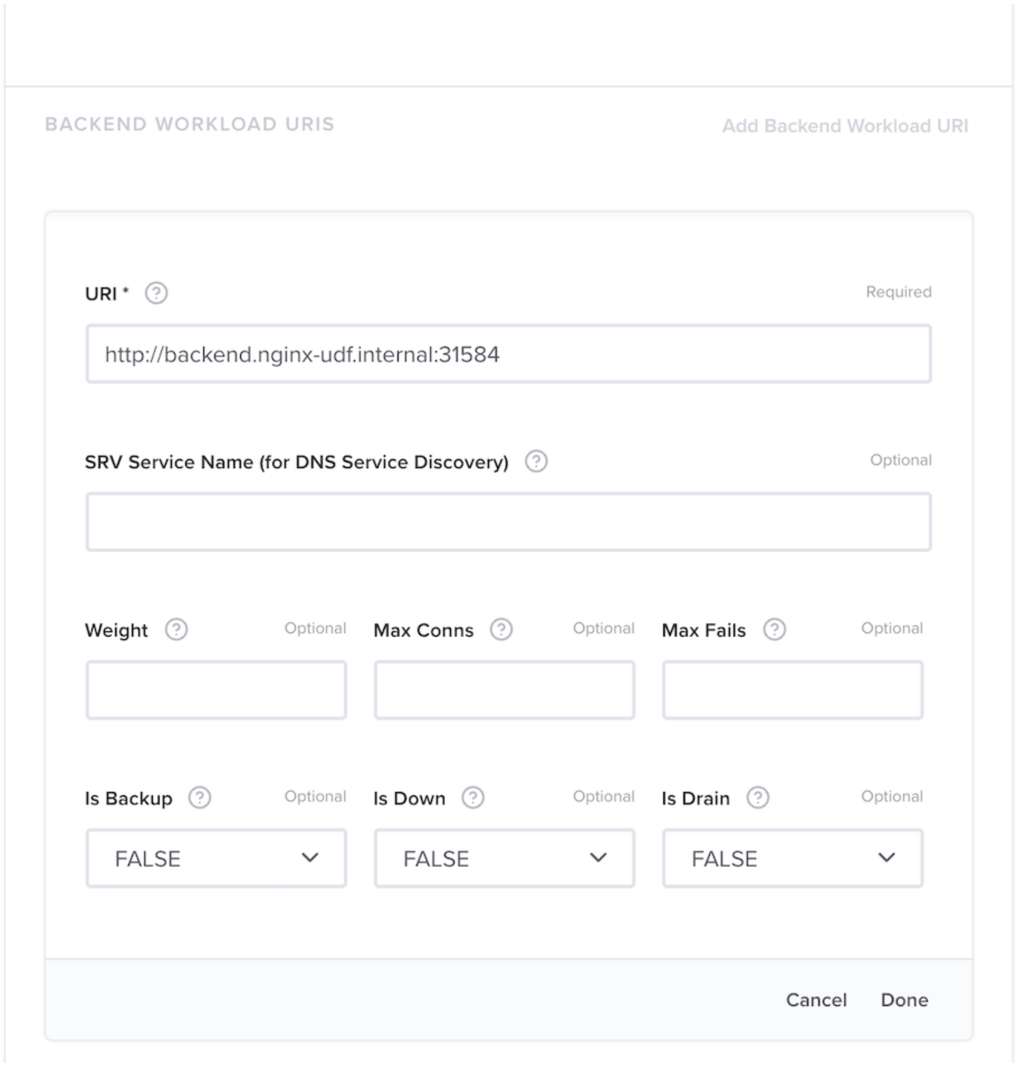
wl_backend

Location References Optional

DNS SERVICE DISCOVERY Add DNS Server

Looks like you don't have any dns server items yet.

DNS Lookup Timeout Optional DNS TTL Override Optional



URI * Required

http://backend.nginx-udf.internal:31584

SRV Service Name (for DNS Service Discovery) Optional

Weight Optional

Max Conns Optional

Max Fails Optional

Is Backup Optional: FALSE

Is Down Optional: FALSE

Is Drain Optional: FALSE

Warning: Don't forget to click on `done` twice

Note: Click `submit`

Step 3 - Test your Controller deployment

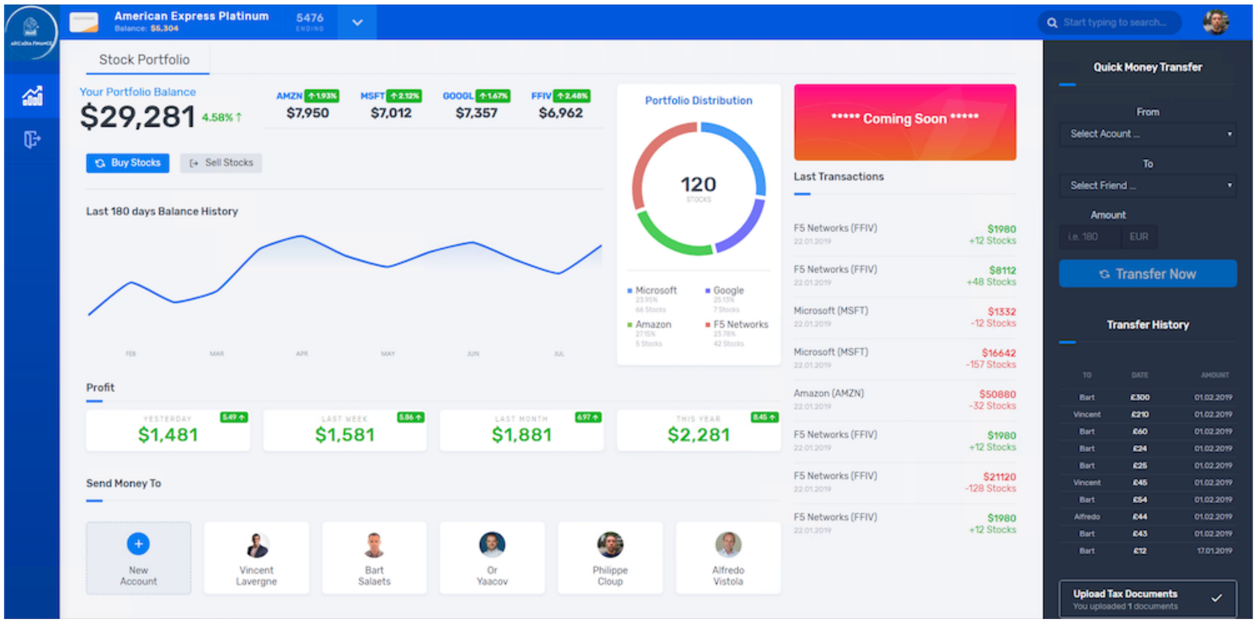
1. Open `Chrome` and click on the bookmark `Arcadia Finance`
2. Click on `Login`
3. Login as `matt` / `ilovef5`
4. You should see the `main app` wihtout `App2` nor `App3`

Warning: Congratulations, you have deployed your first modern app with NGINX+ and the NGINX Controller

1.2.2 DevOps deploy Money Transfer application

In this module, we will deploy the Money Tranfer container for Arcadia Bank application and we will publish it.

Note: At the end of this module, Arcadia Bank application will look like this.



TO	DATE	AMOUNT
Bart	£300	01.02.2019
Vincent	£210	01.02.2019
Bart	£60	01.02.2019
Bart	£24	01.02.2019
Bart	£25	01.02.2019
Vincent	£45	01.02.2019
Bart	£54	01.02.2019
Alfredo	£44	01.02.2019
Bart	£43	01.02.2019
Bart	£12	17.01.2019

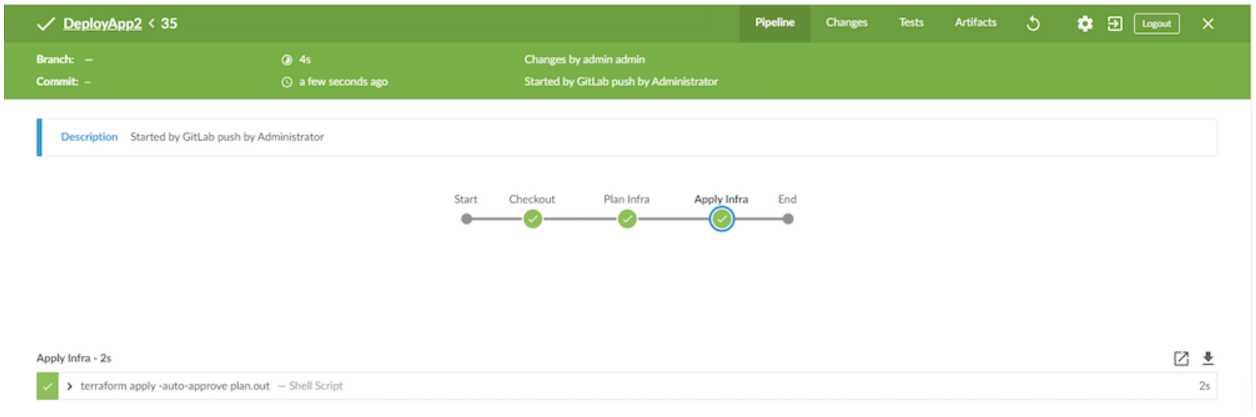
Note: In this lab, we will automate some tasks in the controller. As you noticed in the previous lab, it is long to create and you can make mistakes. We will deploy a new component using the NGINX Controller API.

Step 1 - Deploy Arcadia App2 with a CI/CD pipeline like a DevOps

1. In Gitlab, click on Administrator / Arcadia-App2
   1. Click on file `deploy`
   2. Click `edit` and make a modification - like YES !!!!!
   3. Click `Commit changes`

   Note: At this moment, you simulate a commit like a DevOps. This `commit` will trigger a `webhook` to `Jenkins`, so that Jenkins execute a `pipeline`.

2. In Jenkins, click on `DeployApp2` pipeline
3. A pipeline is running, click on it
4. You can follow the steps



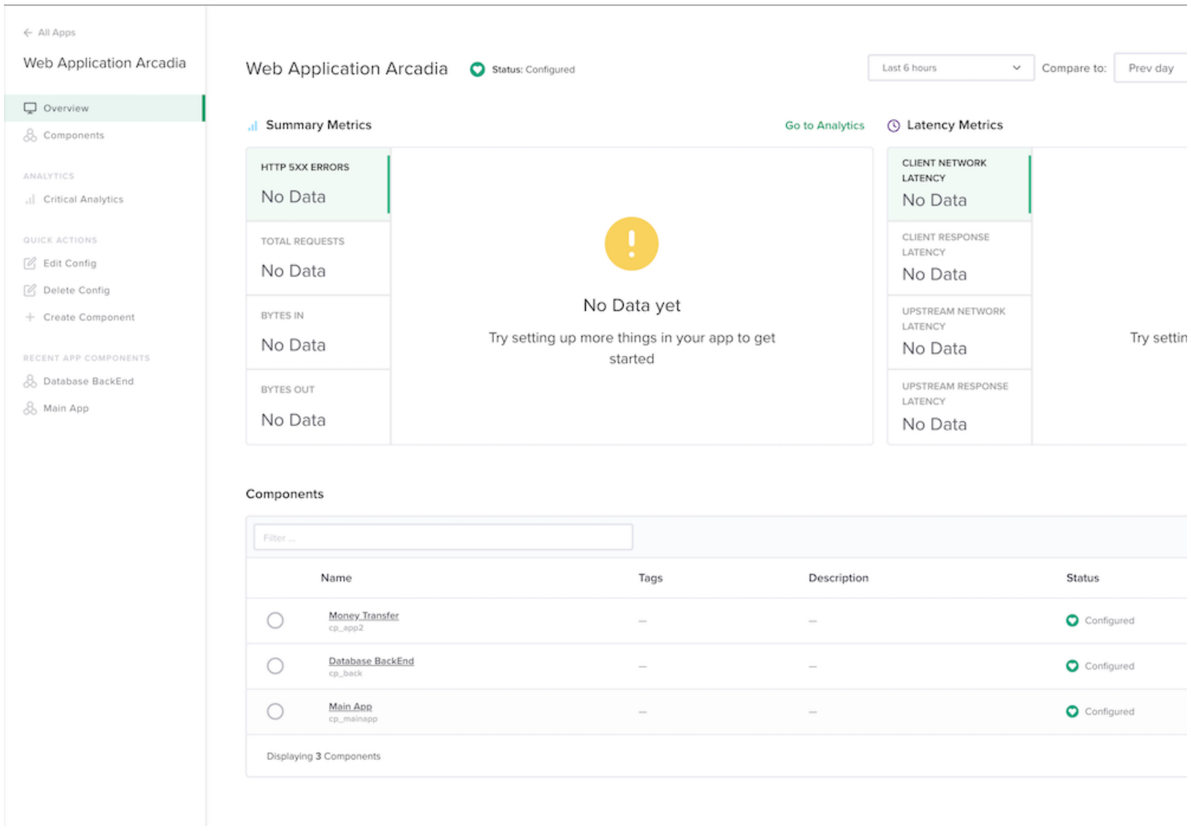
Note: At this stage, App2 (Money Transfer app) is deployed un K8S. But you need to publish it via the controller.

Step 2 - Publish Money Transfer App with NGINX+ and Controller

1. In the `Jumphost` open `Postman`
2. Open collection `Deploy Component App2`
   1. Send the first call `Log in NGINX Controller`
   2. Send the second call `Create App2 Component`

   Note: With one click, you created the component. Fast and no human mistake.

3. Connect to Controller GUI and check the new component in `web application arcadia`


← All Apps

Web Application Arcadia

- Overview
- Components

ANALYTICS

- Critical Analytics

QUICK ACTIONS

- Edit Config
- Delete Config
- Create Component

RECENT APP COMPONENTS

- Database BackEnd
- Main App

Web Application Arcadia — Status: Configured

Last 6 hours — Compare to: Prev day

Summary Metrics — Go to Analytics

HTTP 5XX ERRORS	No Data
TOTAL REQUESTS	No Data
BYTES IN	No Data
BYTES OUT	No Data

No Data yet

Try setting up more things in your app to get started

Latency Metrics

CLIENT NETWORK LATENCY	No Data
CLIENT RESPONSE LATENCY	No Data
UPSTREAM NETWORK LATENCY	No Data
UPSTREAM RESPONSE LATENCY	No Data

Try settin

Components

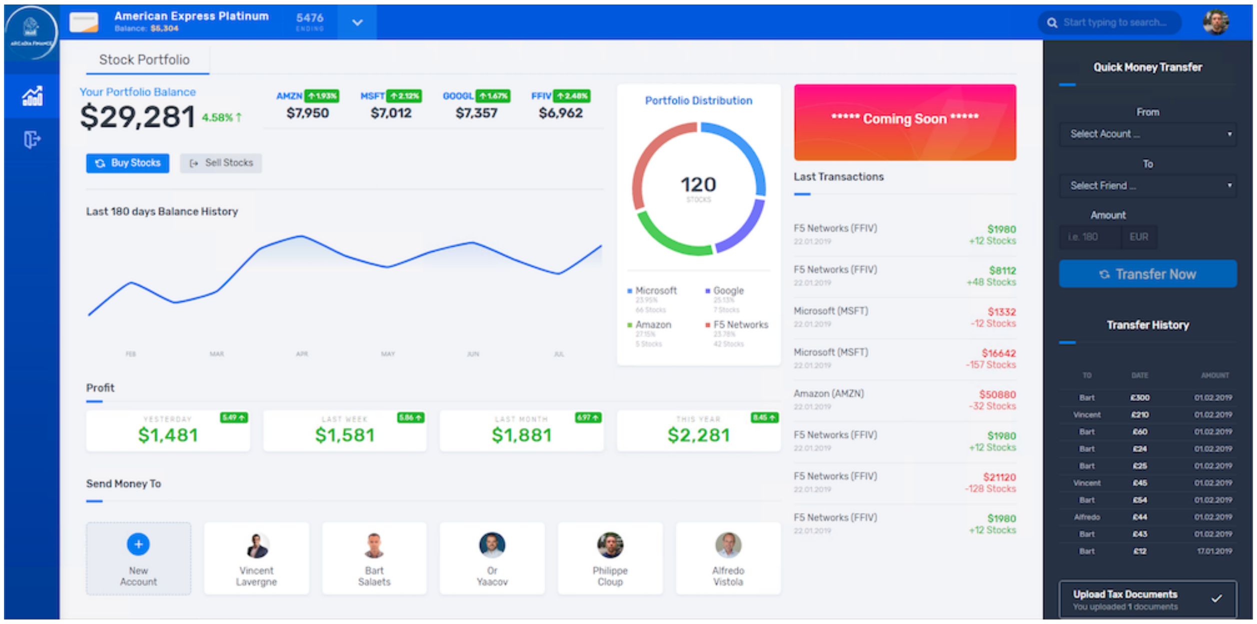
Filter ...

Name	Tags	Description	Status
Money Transfer cp_app2	–	–	Configured
Database BackEnd cp_back	–	–	Configured
Main App cp_mainapp	–	–	Configured

Displaying 3 Components

Note: You can notice the new `Money Transfer` component is created

4. In `Chrome` refresh the page. You can see the new App `Money Transfer`
5. Transfer some money to your friends in order to populate analytics


American Express Platinum — Balance: $5,304 — 5476 ENDING

Start typing to search...

Stock Portfolio

Your Portfolio Balance

$29,281 4.58% ↑

AMZN ↑1.93%	MSFT ↑2.12%	GOOGL ↑1.67%	FFIV ↑2.48%
$7,950	$7,012	$7,357	$6,962

Buy Stocks — Sell Stocks

Last 180 days Balance History

Portfolio Distribution

Microsoft 23.95% 66 Stocks	Google 25.13% 7 Stocks
Amazon 27.15% 5 Stocks	F5 Networks 23.78% 42 Stocks

Profit

YESTERDAY	LAST WEEK	LAST MONTH	THIS YEAR
$1,481 (5.49 ↑)	$1,581 (5.86 ↑)	$1,881 (6.97 ↑)	$2,281 (8.45 ↑)

Send Money To

New Account — Vincent Lavergne — Bart Salaets — Or Yaacov — Philippe Cloup — Alfredo Vistola

***** Coming Soon *****

Last Transactions

F5 Networks (FFIV) 22.01.2019	$1980 +12 Stocks
F5 Networks (FFIV) 22.01.2019	$8112 +48 Stocks
Microsoft (MSFT) 22.01.2019	$1332 -12 Stocks
Microsoft (MSFT) 22.01.2019	$16642 -157 Stocks
Amazon (AMZN) 22.01.2019	$50880 -32 Stocks
F5 Networks (FFIV) 22.01.2019	$1980 +12 Stocks
F5 Networks (FFIV) 22.01.2019	$21120 -128 Stocks
F5 Networks (FFIV) 22.01.2019	$1980 +12 Stocks

Quick Money Transfer

From: Select Acount ...

To: Select Friend ...

Amount: i.e. 180 EUR

Transfer Now

Transfer History

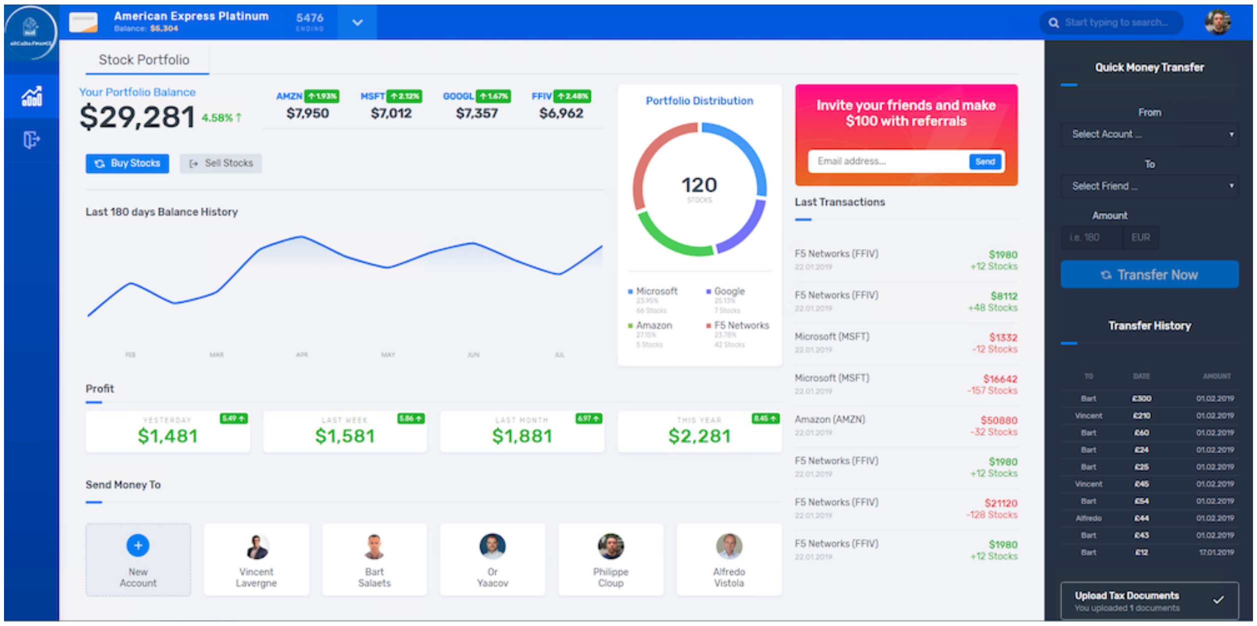
TO	DATE	AMOUNT
Bart	£300	01.02.2019
Vincent	£210	01.02.2019
Bart	£60	01.02.2019
Bart	£24	01.02.2019
Bart	£25	01.02.2019
Vincent	£45	01.02.2019
Bart	£54	01.02.2019
Alfredo	£44	01.02.2019
Bart	£43	01.02.2019
Bart	£12	17.01.2019

Upload Tax Documents — You uploaded 1 documents

1.2.3 DevOps deploy Refer Friends Application

In this module, we will deploy the Refer Friends container for Arcadia Bank web application and we will publish it.

Note: At the end of this module, Arcadia Bank application will look like this.



Note: In this lab, all tasks will be automated. DevOps will deploy the app in K8S, and NetOps will create the new component at the same time.

Step 1 - Deploy Arcadia App3 and the new componenent with a CI/CD pipeline

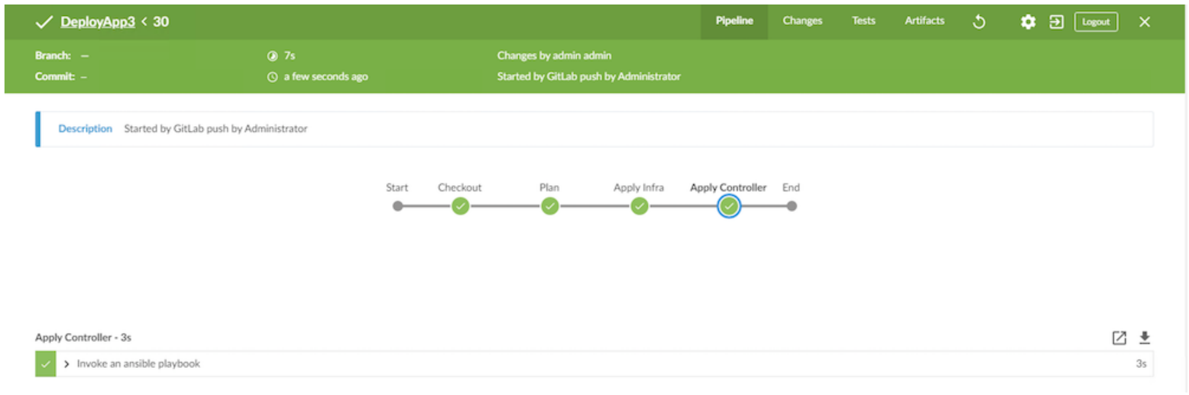
1. In Gitlab, click on Administrator / Arcadia-App3
    1. Click on file `deploy`
    2. Click `edit` and make a modification - like YES !!!!!
    3. Click `Commit changes`

    Note: At this moment, you simulate a commit like a DevOps. This `commit` will trigger a `webhook` to `Jenkins`, so that Jenkins execute a `pipeline`.

    ---

2. In Jenkins, click on `DeployApp3` pipeline

3. A pipeline is running, click on it
4. You can follow the steps

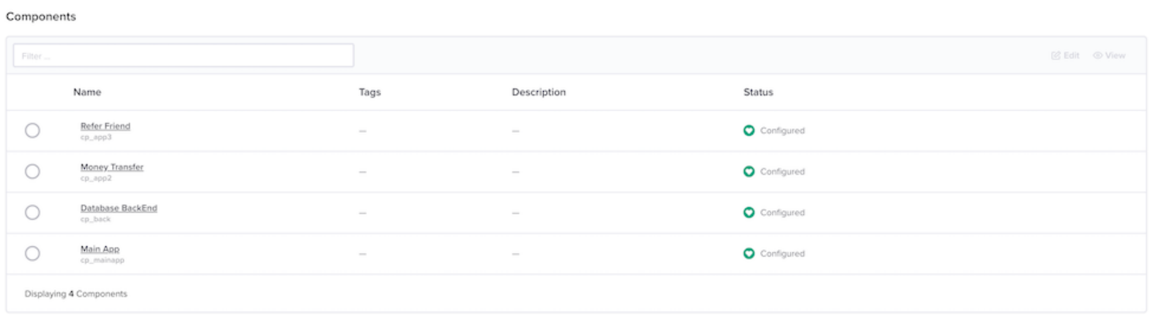


Branch: –
Commit: –
7s
a few seconds ago
Changes by admin admin
Started by GitLab push by Administrator

Description Started by GitLab push by Administrator

Apply Controller - 3s
✓ > Invoke an ansible playbook 3s

5. Connect to Controller GUI and check the new component in `web application arcadia`

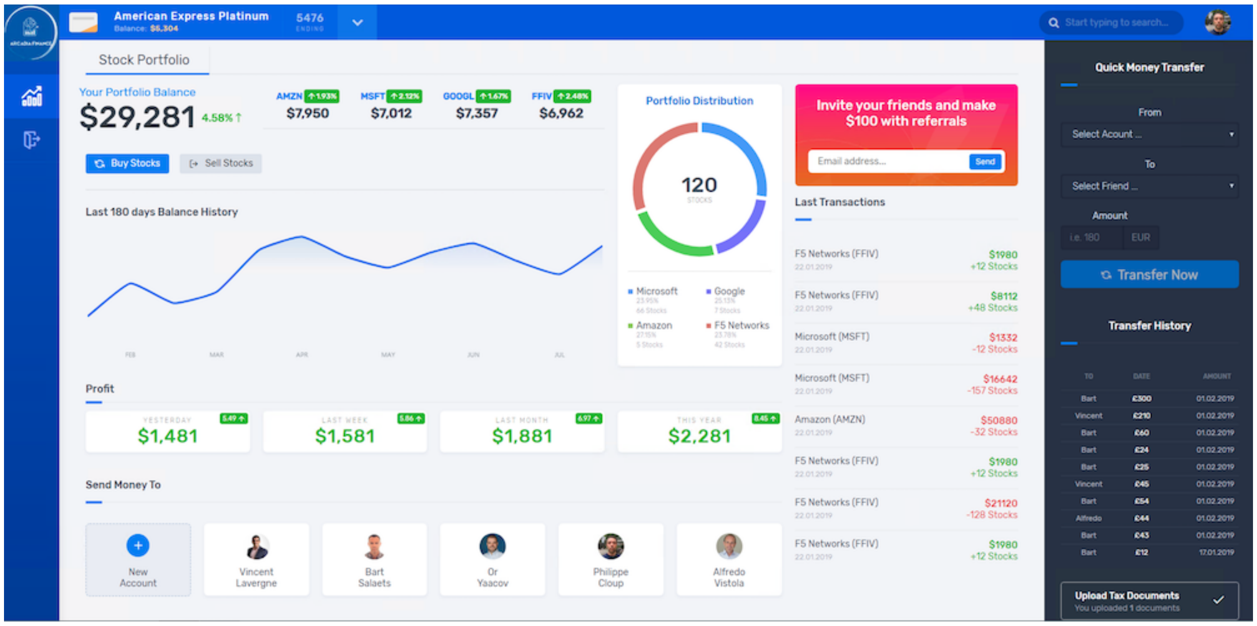


Components

	Name	Tags	Description	Status
○	Refer Friend cp_app3	–	–	✓ Configured
○	Money Transfer cp_app2	–	–	✓ Configured
○	Database BackEnd cp_back	–	–	✓ Configured
○	Main App cp_mainapp	–	–	✓ Configured

Displaying 4 Components

6. In `Chrome` refresh the page. You can see the new App `Refer friends`



Last Transactions

F5 Networks (FFIV) 22.01.2019	$1980 +12 Stocks
F5 Networks (FFIV) 22.01.2019	$8112 +48 Stocks
Microsoft (MSFT) 22.01.2019	$1332 -12 Stocks
Microsoft (MSFT) 22.01.2019	$16642 -157 Stocks
Amazon (AMZN) 22.01.2019	$50880 -32 Stocks
F5 Networks (FFIV) 22.01.2019	$1980 +12 Stocks
F5 Networks (FFIV) 22.01.2019	$21120 -128 Stocks
F5 Networks (FFIV) 22.01.2019	$1980 +12 Stocks

Transfer History

TO	DATE	AMOUNT
Bart	£300	01.02.2019
Vincent	£210	01.02.2019
Bart	£60	01.02.2019
Bart	£24	01.02.2019
Bart	£25	01.02.2019
Vincent	£45	01.02.2019
Bart	£54	01.02.2019
Alfredo	£44	01.02.2019
Bart	£43	01.02.2019
Bart	£12	17.01.2019

Note: Congrats, as you can notice, with one `commit` in Gitlab, you triggered a webhook that deployed the `app` and the `infrastructure`

Warning: Now, it's time to protect Arcadia Finance web application with a BIG-IP.

1.2.4 Protect Arcadia Application with Declarative WAF

Warning: ONLY IF YOU START THE LAB FROM HERE - ELSE DON'T READ THIS WARNING. If you want to start from here (because you are only interested by Declarative WAF), and do not want to run all the steps before, you can use `Postman` and `Jenkins` to create everything for you. To do so, follow the steps below.

1. Open `Jenkins` and run the pipeline `DeployMainApp`
2. Open `Postman`, and select the collection `Arcadia Manual Pipeline - no CICD`
3. Run the calls
    1. Login to NGINX Controller
    2. Create WebApp Application
    3. Create MainApp Component
    4. Create BackEnd Component
4. Open `Jenkins` and run the pipeline `DeployApp2`
5. Open `Postman`, and select the collection `Arcadia Manual Pipeline - no CICD`
6. Run the call
    1. Create App2 Component
7. Open `Jenkins` and run the pipeline `DeployApp3`

Now, Arcadia App is fully deployed and the NGINX Controller is set up.

In this module, we will deploy a WAF policy to protect Arcadia Bank application and we will publish it. With v16.0 (and in draft in v15.1), the WAF policy can be deployed via a declarative call, and the WAF policy itself is a JSON file.

Note: We use the new v15.1/v16.0 Declarative WAF policy. You can retrieve the JSON Policy in the GitLab repo and below.

Note: You can learn more on the Declarative WAF policy here : https://f5.sharepoint.com/sites/EMEASystemsEngineering/SitePages/Adv.-WAF-v16.0-Declarative-API.aspx

```
{
    "policy": {
        "name": "policy-fund-1",
        "description": "Policy Example - Rapid Deployment",
        "template": {
            "name": "POLICY_TEMPLATE_RAPID_DEPLOYMENT"
```

(continues on next page)

(continued from previous page)

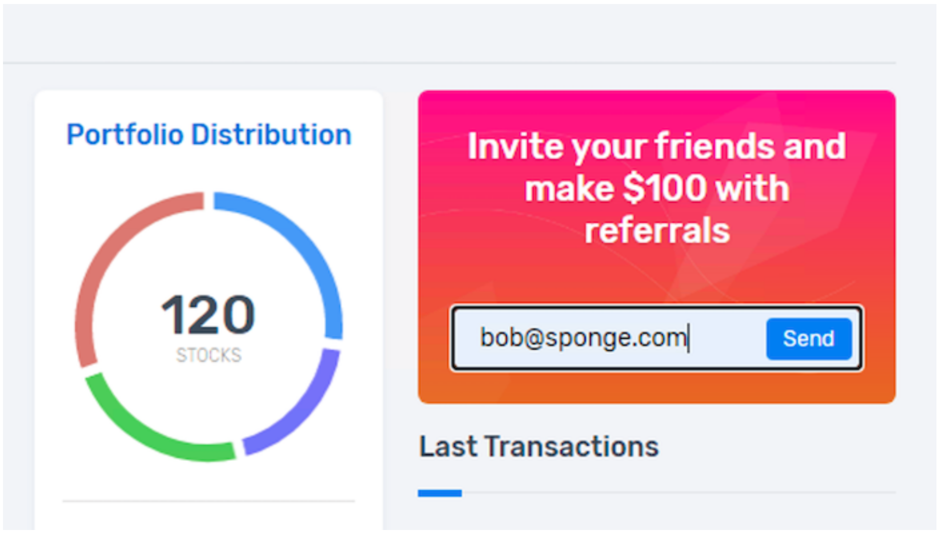
```
        },
        "enforcementMode": "blocking",
        "server-technologies": [
            {
                "serverTechnologyName": "MySQL"
            },
            {
                "serverTechnologyName": "Unix/Linux"
            },
            {
                "serverTechnologyName": "MongoDB"
            }
        ],
        "signature-settings": {
            "signatureStaging": false
        },
        "policy-builder": {
            "learnOnlyFromNonBotTraffic": false
        },
        "response-pages": [
            {
                "responsePageType": "ajax",
                "ajaxEnabled": true,
                "ajaxPopupMessage": "My customized popup message! Your support ID is:
↪<%TS.request.ID()%>"
            }
            ]
    }
}
```

Note: You can notice this JSON policy is based in Rapid Deployment template and we added few things like Server-Technologies, Signature Staging, Policy Buidler and Response Page.

Step 1 - Send an attack

1. In `Chrome`, in Arcadia web application, refer a friend

    1. Refer bob@sponge.com

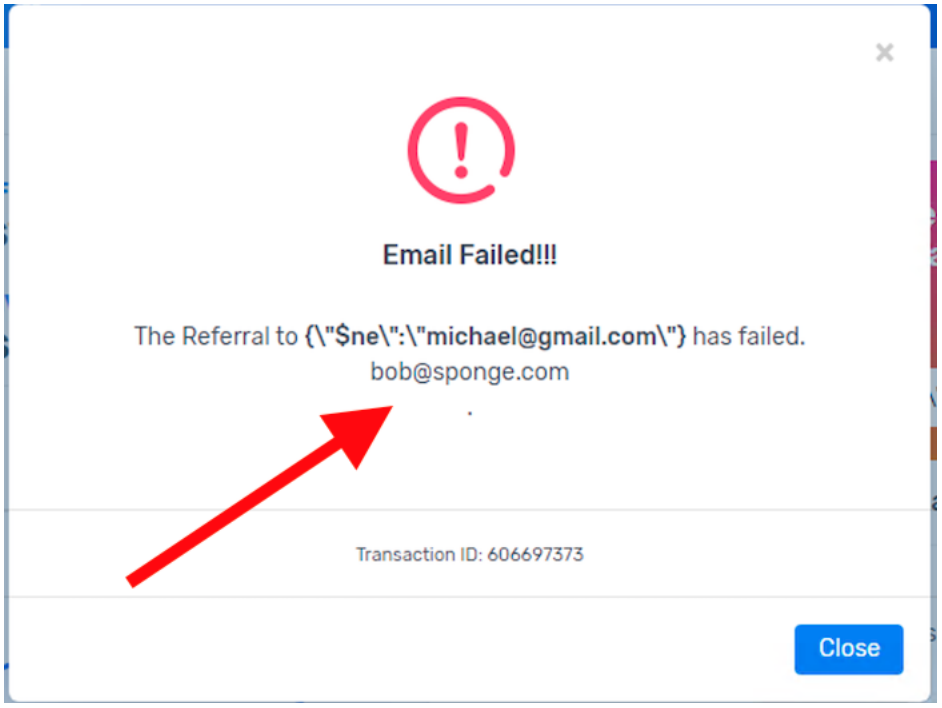


2. Send an attack with the below payload in the `refer friend` field

```
{\"$ne\":\"michael@gmail.com\"}
```

Note: This attacks means return everything not equals to `michael@gmail.com`

3. Attack succeed and you can get the DB content


Email Failed!!!

The Referral to **{\"$ne\":\"michael@gmail.com\"}** has failed.
bob@sponge.com

Transaction ID: 606697373

Step 2 - Push AS3 declaration to deploy the WAF policy

Note: It is important to understand what we are doing here. We are leveraging all the new v16.0 Adv. WAF Declarative policy features. With one API call (done by Jenkins and Ansible), we will deploy a new AS3 declaration with a WAF policy.

Check the files used here

1. In `Gitlab`, open `Administrator / as3-waf` project
2. You can see several files, but the most important are
   1. playbook-v16.yaml
   2. as3-v16.json
   3. policy-fund-1.json
3. Open `policy-fund-1.json`

```
{
    "policy": {
        "name": "policy-fund-1",
        "description": "Policy Example - Rapid Deployment",
        "template": {
            "name": "POLICY_TEMPLATE_RAPID_DEPLOYMENT"
        },
        "enforcementMode": "blocking",
        "server-technologies": [
            {
                "serverTechnologyName": "MySQL"
            },
            {
                "serverTechnologyName": "Unix/Linux"
            },
            {
                "serverTechnologyName": "MongoDB"
            }
        ],
        "signature-settings": {
            "signatureStaging": false
        },
        "policy-builder": {
            "learnOnlyFromNonBotTraffic": false
        },
        "response-pages": [
            {
                "responsePageType": "ajax",
                "ajaxEnabled": true,
                "ajaxPopupMessage": "My customized popup message! Your␣
↪support ID is: <%TS.request.ID()%>"
            }
            ]
    }
}
```

Note: This is our declarative JSON WAF policy

4. Open `as3-v16.json`

```
{
    "class": "AS3",
    "action": "deploy",
    "persist": true,
    "declaration": {
        "class": "ADC",
        "schemaVersion": "3.2.0",
        "id": "Prod_Web_AS3",
        "Web-Prod": {
            "class": "Tenant",
            "defaultRouteDomain": 0,
            "arcadia": {
                "class": "Application",
                "template": "generic",
                "VS_WebApp": {
                    "class": "Service_HTTPS",
                    "remark": "Accepts HTTPS/TLS connections on port 443",
                    "virtualAddresses": ["10.1.10.26"],
                    "redirect80": false,
                    "pool": "pool_NGINX_WebApp",
                    "policyWAF": {
                        "use": "Arcadia_WAF_policy"
                    },
                    "securityLogProfiles": [{
                        "bigip": "/Common/Log all requests"
                    }],
                    "profileTCP": {
                        "egress": "wan",
                        "ingress": { "use": "TCP_Profile" } },
                    "profileHTTP": { "use": "custom_http_profile" },
                    "serverTLS": { "bigip": "/Common/arcadia_client_ssl" }
                },
                "Arcadia_WAF_policy": {
                    "class": "WAF_Policy",
                    "url": "http://10.1.20.4/root/as3-waf/-/raw/master/
↪policy-fund-1.json",
                    "ignoreChanges": true
                },
                "pool_NGINX_WebApp": {
                    "class": "Pool",
                    "monitors": ["http"],
                    "members": [{
                        "servicePort": 8080,
                        "serverAddresses": ["10.1.20.10"]
                    }]
                },
                "custom_http_profile": {
                    "class": "HTTP_Profile",
                    "xForwardedFor": true
                },
                "TCP_Profile": {
                    "class": "TCP_Profile",
```

(continues on next page)

(continued from previous page)

```
                    "idleTimeout": 60 }
                }
            }
        }
}
```

Note: In this AS3 declaration, you can notice the new v16.0 Adv. WAF Reference section (`Arcadia_WAF_policy`). This section refers to our external JSON policy file, and will upload, import and apply the policy in the BIG-IP.

5. Open `playbook-v16.yaml`\`

```
---

- hosts: bigip
connection: local
gather_facts: false
vars:
    my_admin: "admin"
    my_password: "admin"
    bigip: "10.1.1.12"

tasks:
- name: Deploy AS3 WebApp
    uri:
    url: "https://{{ bigip }}/mgmt/shared/appsvcs/declare"
    method: POST
    headers:
        "Content-Type": "application/json"
        "Authorization": "Basic YWRtaW46YWRtaW4="
    body: "{{ lookup('file','as3-v16.json') }}"
    body_format: json
    validate_certs: no
    status_code: 200
```

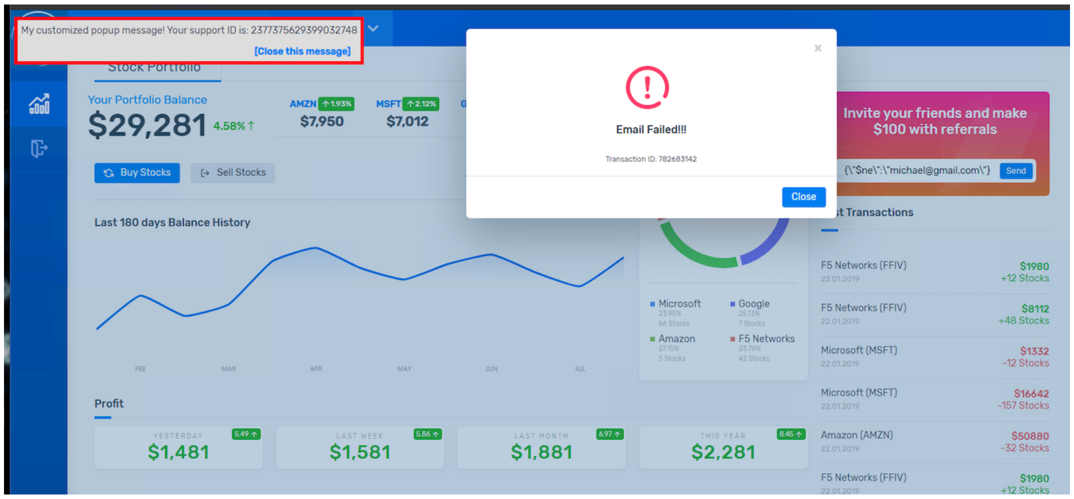
Note: You can see the playbook is very simple in v16.0 thanks to the AS3 call. It will do all the job for us. This playbook is just sending an AS3 declaration call to the BIGIP.

Run the CI/CD pipeline

1. In `Jenkins`, click on `DeployWAF` pipeline
2. Run the `pipeline`
3. In `Chrome`, launch an incognito window, and retry the attack

```
{\"$ne\":\"michael@gmail.com\"}
```

4. Attack fails and you can notice the `AJAX blocking page` set in the JSON declarative WAF policy



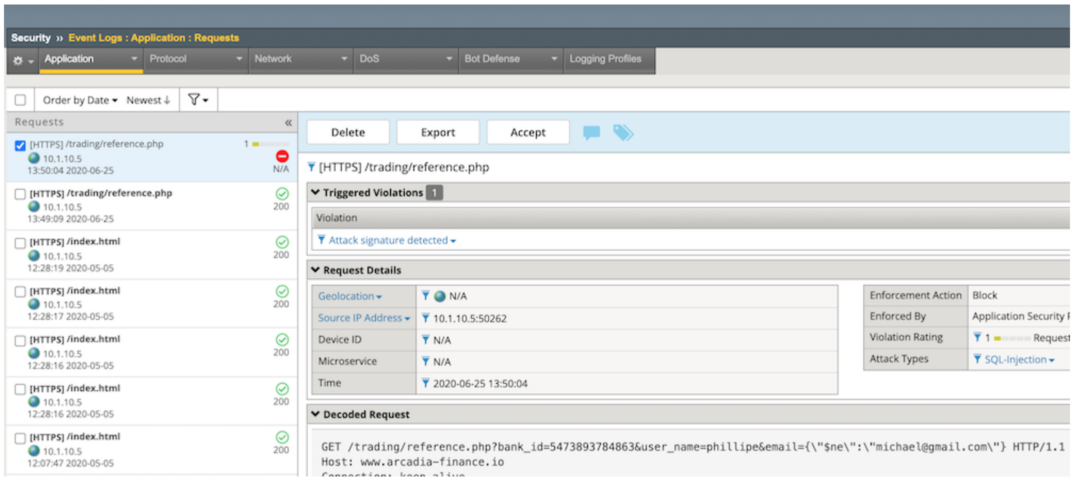
My customized popup message! Your support ID is: 2377375629399032748

[Close this message]

Email Failed!!!

Transaction ID: 782683142

5. Check logs in the BIG-IP



Security » Event Logs : Application : Requests

Requests	
[HTTPS] /trading/reference.php 10.1.10.5 13:50:04 2020-06-25	N/A
[HTTPS] /trading/reference.php 10.1.10.5 13:49:09 2020-06-25	200
[HTTPS] /index.html 10.1.10.5 12:28:19 2020-05-05	200
[HTTPS] /index.html 10.1.10.5 12:28:17 2020-05-05	200
[HTTPS] /index.html 10.1.10.5 12:28:16 2020-05-05	200
[HTTPS] /index.html 10.1.10.5 12:28:16 2020-05-05	200
[HTTPS] /index.html 10.1.10.5 12:07:47 2020-05-05	200

[HTTPS] /trading/reference.php

Triggered Violations 1

Violation
Attack signature detected

Request Details

Geolocation	N/A
Source IP Address	10.1.10.5:50262
Device ID	N/A
Microservice	N/A
Time	2020-06-25 13:50:04

Enforcement Action	Block
Enforced By	Application Security
Violation Rating	1 Request
Attack Types	SQL-Injection

Decoded Request

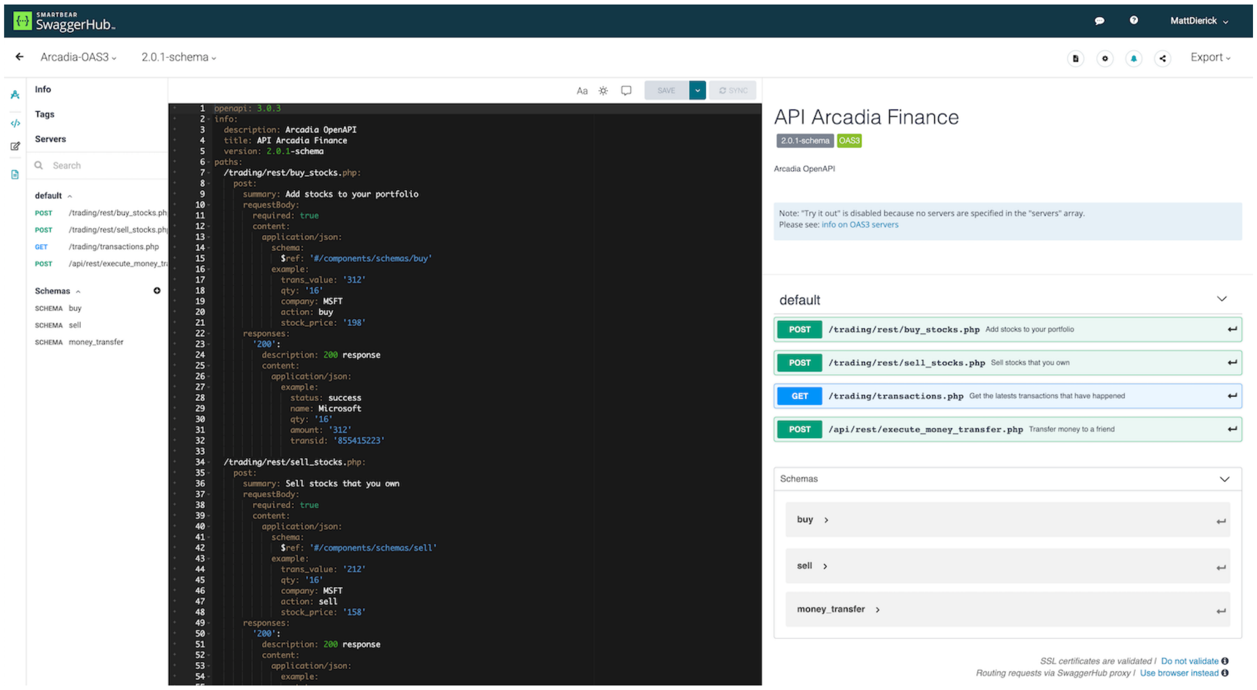
```
GET /trading/reference.php?bank_id=5473893784863&user_name=phillipe&email={\"$ne\":\"michael@gmail.com\"} HTTP/1.1
Host: www.arcadia-finance.io
```

1.3 Class 3 - Publish and Protect Arcadia API

In this section, we will publish and protect Arcadia API. There are 4 API allowing us to :

- See last transactions
- Buy stocks
- Sell stocks
- Make a money transfer

The API specification is available here : https://app.swaggerhub.com/apis/F5EMEASSA/Arcadia-OAS3/2.0.1-schema



```
openapi: 3.0.3
info:
  description: Arcadia OpenAPI
  title: API Arcadia Finance
  version: 2.0.1-schema
paths:
  /trading/rest/buy_stocks.php:
    post:
      summary: Add stocks to your portfolio
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/buy'
            example:
              trans_value: '312'
              qty: '16'
              company: MSFT
              action: buy
              stock_price: '198'
      responses:
        '200':
          description: 200 response
          content:
            application/json:
              example:
                status: success
                name: Microsoft
                qty: '16'
                amount: '312'
                transid: '855415223'

  /trading/rest/sell_stocks.php:
    post:
      summary: Sell stocks that you own
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/sell'
            example:
              trans_value: '212'
              qty: '16'
              company: MSFT
              action: sell
              stock_price: '158'
      responses:
        '200':
          description: 200 response
          content:
            application/json:
              example:
```

API Arcadia Finance

2.0.1-schema OAS3

Arcadia OpenAPI

Note: "Try it out" is disabled because no servers are specified in the "servers" array. Please see: info on OAS3 servers

default

POST /trading/rest/buy_stocks.php Add stocks to your portfolio

POST /trading/rest/sell_stocks.php Sell stocks that you own

GET /trading/transactions.php Get the latests transactions that have happened

POST /api/rest/execute_money_transfer.php Transfer money to a friend

Schemas: buy, sell, money_transfer

SSL certificates are validated / Do not validate
Routing requests via SwaggerHub proxy / Use browser instead

1.3.1 Module 1 - Publish API with OAS3 spec file from the Controller GUI

Note: In this section we will push OAS3 specification file into the controller GUI in order to create the API

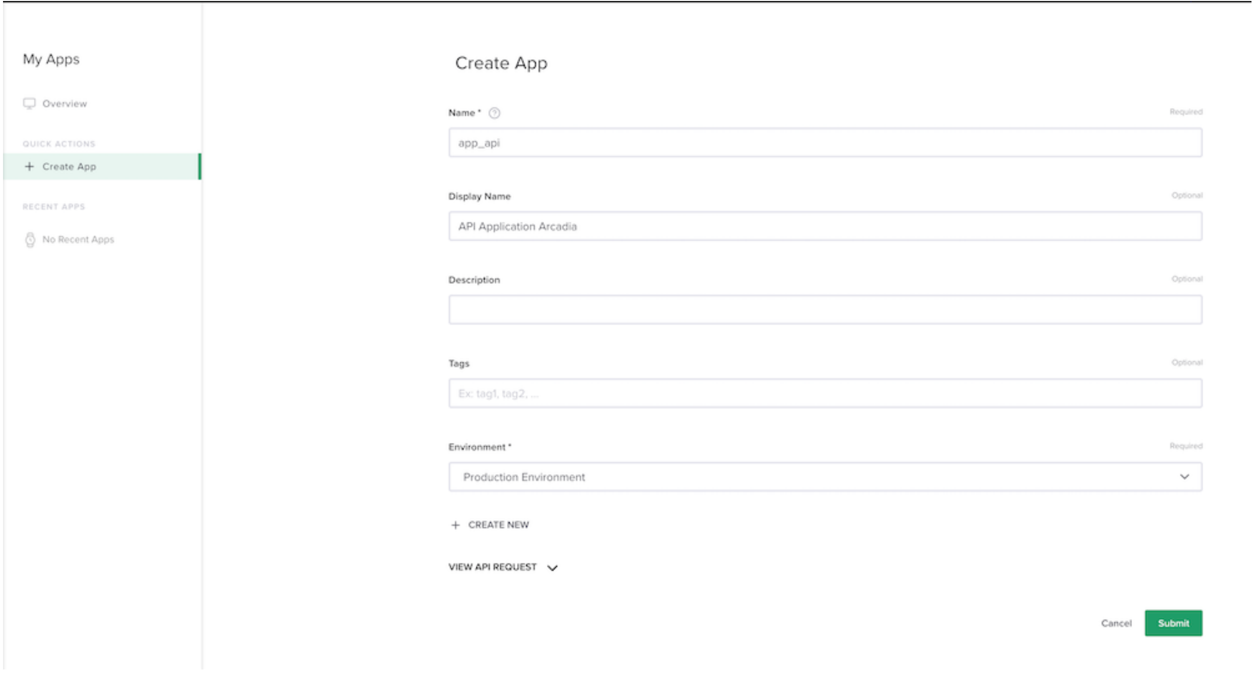
Connect to Controller GUI via your laptop's browser or the jumphost

login: admin@nginx-udf.internal

password: admin123!

Step 1 - Create an New Application

1. Click on top left corner icon, and click on `Apps`
2. Click `Create`
3. Create a new Application
   1. name : `app_api`
   2. display name : `API Application Arcadia`
   3. Environment : `Production Environment`
4. Click `Summit`



Create App

Name * : app_api (Required)

Display Name : API Application Arcadia (Optional)

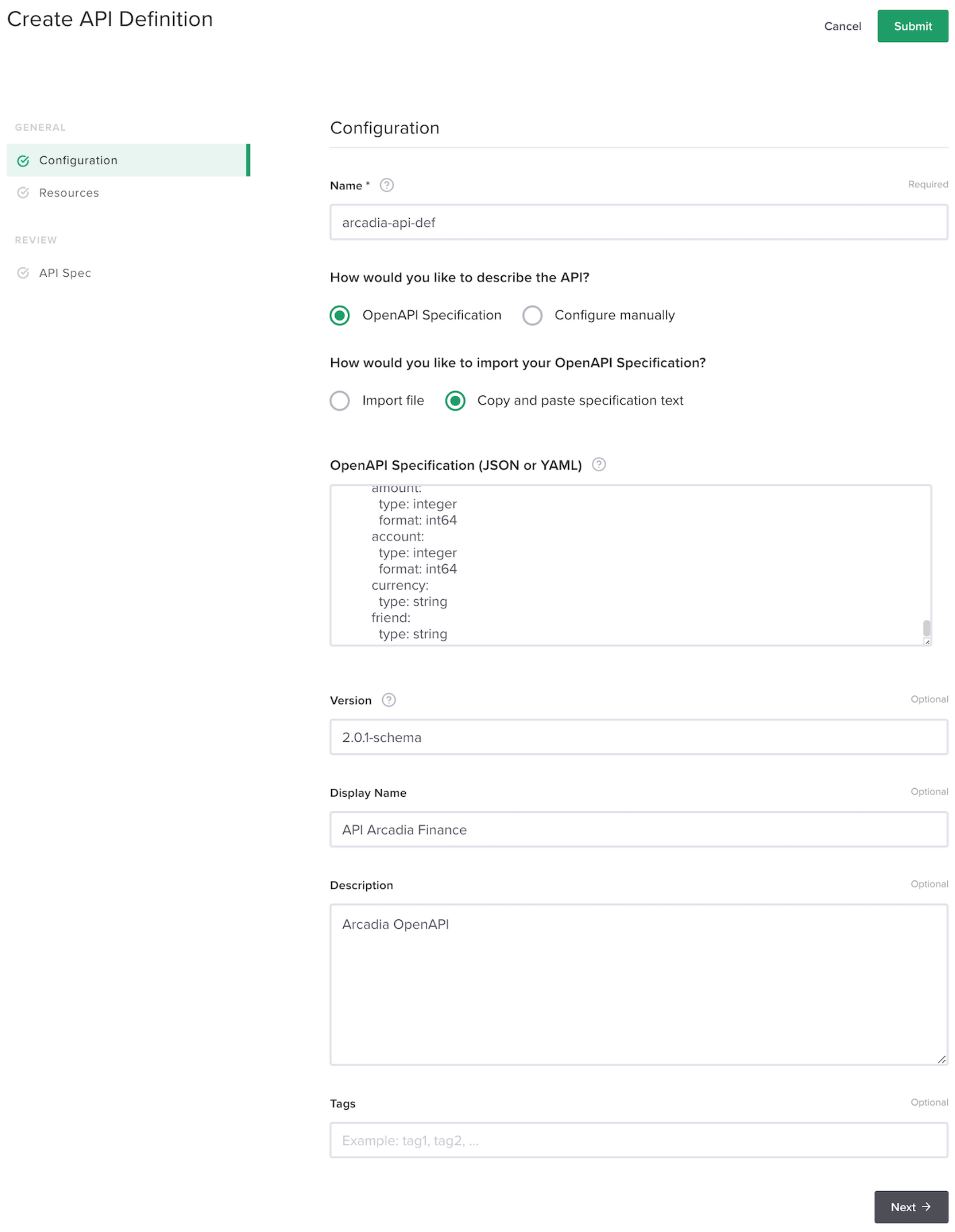
Description : (Optional)

Tags : Ex: tag1, tag2, ... (Optional)

Environment * : Production Environment (Required)

Step 2 - Create an API Definition

1. Click on the left menu `APIs`
2. Click `Create API Definition`
   1. Name : `arcadia-api-def`
   2. Display Name : `Arcadia API Definition`
   3. Version : `v1`
   4. Select `OpenAPI specification`
      1. and `Copy and paste specification text` **if you are not connected in the jumphost** from here : https://app.swaggerhub.com/apis/F5EMEASSA/Arcadia-OAS3/2.0.1-schema
      2. or `Import file` **if your are connected in the jumphost**, the file is located in the `Desktop` folder and its name is `OAS3-Arcadia.yaml`


Create API Definition

Cancel Submit

GENERAL

Configuration

Resources

REVIEW

API Spec

Configuration

Name * Required

arcadia-api-def

How would you like to describe the API?

[✓ OpenAPI Specification] Configure manually

How would you like to import your OpenAPI Specification?

Import file [✓ Copy and paste specification text]

OpenAPI Specification (JSON or YAML)

```
      amount:
        type: integer
        format: int64
      account:
        type: integer
        format: int64
      currency:
        type: string
      friend:
        type: string
```

Version Optional

2.0.1-schema

Display Name Optional

API Arcadia Finance

Description Optional

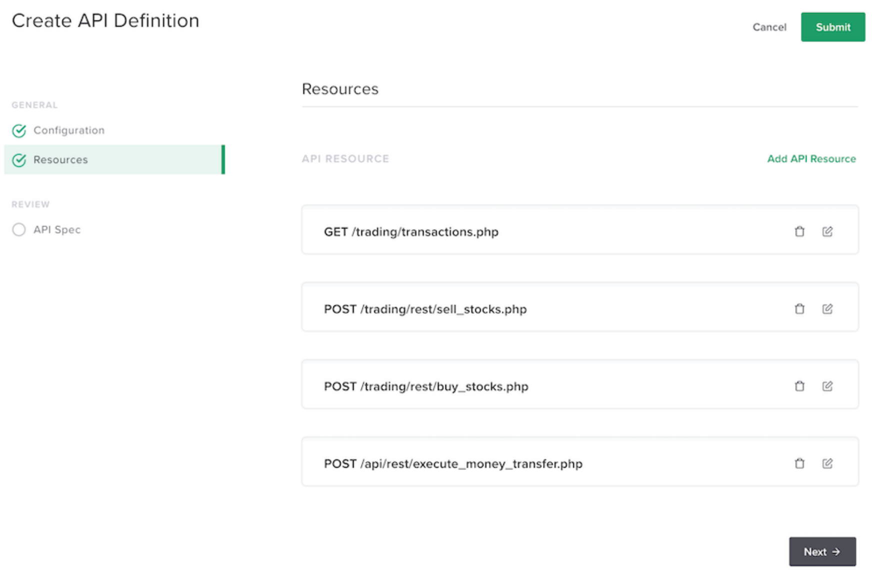
Arcadia OpenAPI

Tags Optional

Example: tag1, tag2, ...

Next →

5. Click `Next`
6. You can see all the resources have been imported from the `swagger file` and please open one resource to check its content.

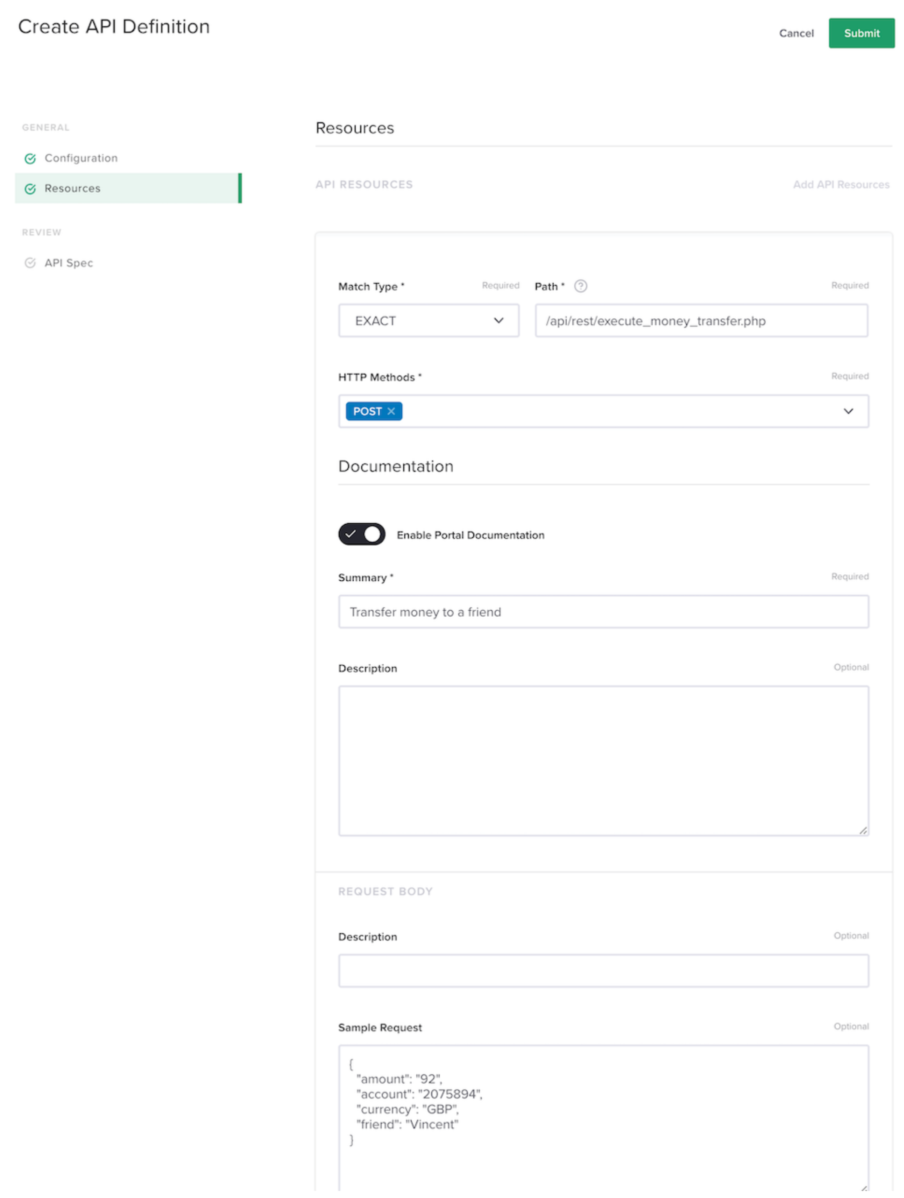


GET /trading/transactions.php

POST /trading/rest/sell_stocks.php

POST /trading/rest/buy_stocks.php

POST /api/rest/execute_money_transfer.php



Match Type * (Required): EXACT

Path * (Required): /api/rest/execute_money_transfer.php

HTTP Methods * (Required): POST

Enable Portal Documentation

Summary * (Required): Transfer money to a friend

Description (Optional):

Description (Optional):

Sample Request (Optional):

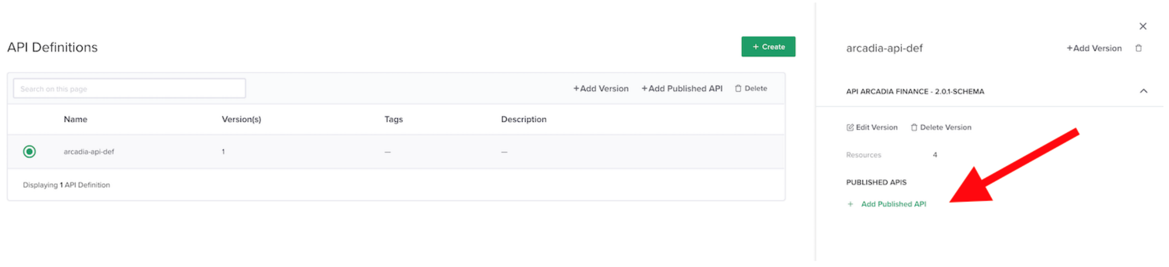
```
{
  "amount": "92",
  "account": "2075894",
  "currency": "GBP",
  "friend": "Vincent"
}
```

7. Click `Summit`

Step 3 - Publish the API

Note: At this stage, the API definition is created. So the controller knows the differents URI but doesn't know yet where to forward the traffic to.

1. Click on the API definition raw, and on the right frame, click on `+ Add Published API`

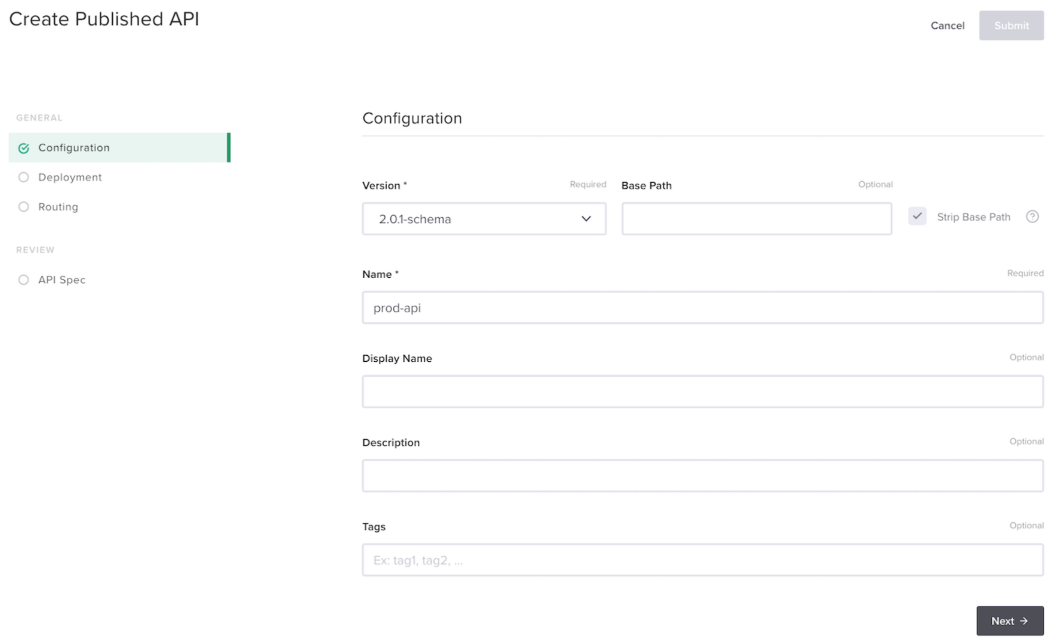


	Name	Version(s)	Tags	Description
●	arcadia-api-def	1	–	–

Displaying 1 API Definition

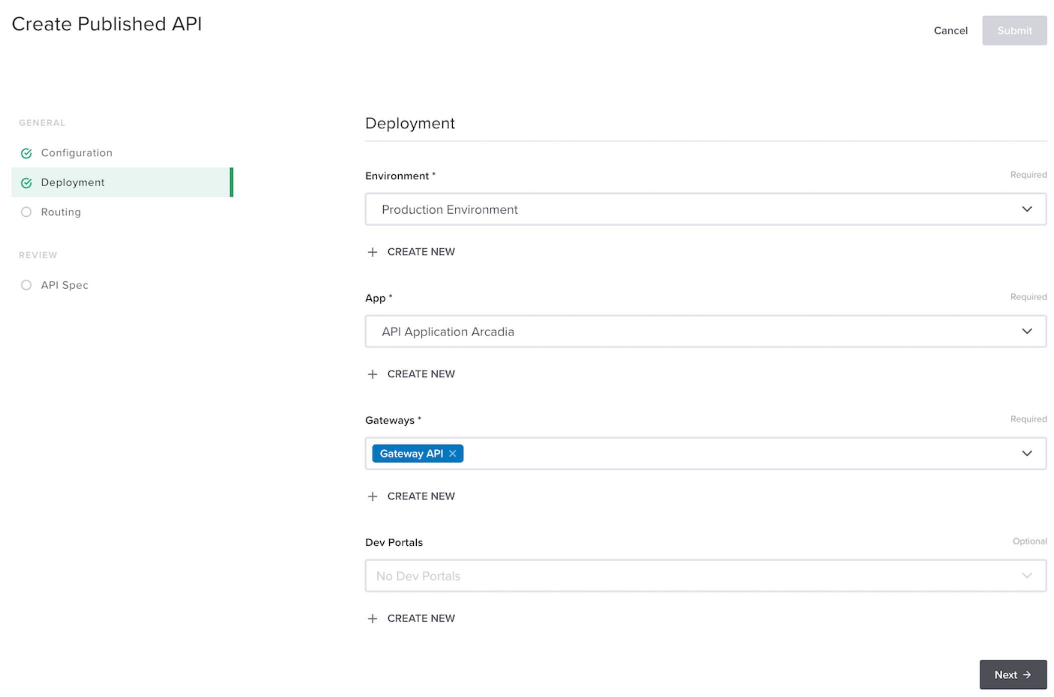
Resources 4

2. Configure the mandatory settings
   1. Name: `prod-api`
   2. Display Name: `Production API`



Version * Required: 2.0.1-schema
Base Path Optional:
Name * Required: prod-api
Display Name Optional:
Description Optional:
Tags Optional: Ex: tag1, tag2, ...

3. Click `Next`
4. Configure the `deployment`
   1. Environment: `Production Environment`
   2. App: `API Application Arcadia`
   3. Gateways: `Gateway API`


Create Published API

Cancel Submit

GENERAL
- Configuration
- Deployment
- Routing

REVIEW
- API Spec

Deployment

Environment * — Required: Production Environment

+ CREATE NEW

App * — Required: API Application Arcadia

+ CREATE NEW

Gateways * — Required: Gateway API

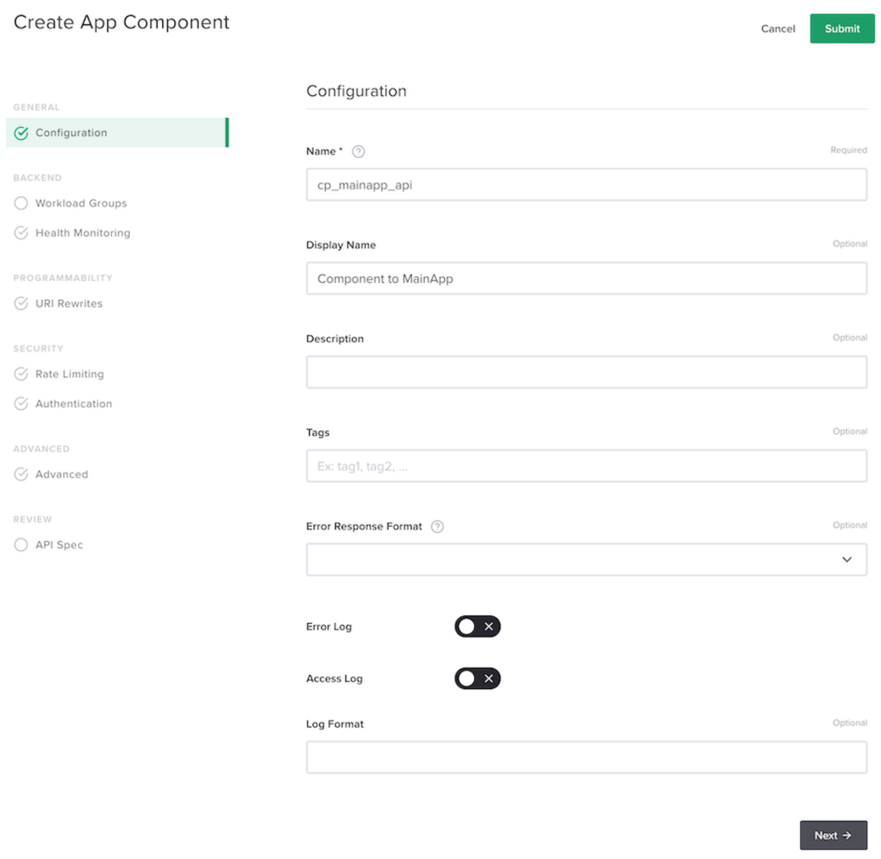
+ CREATE NEW

Dev Portals — Optional: No Dev Portals

+ CREATE NEW

Next →

5. It is time to configure the `Routing`. It is similar to the `components` in the WebApp configuration
6. Create a new component, routing the traffic to the `MainApp`
    1. Click `Add New` in the `Components section` and configure it as below



Configuration

Name * (Required): cp_mainapp_api

Display Name (Optional): Component to MainApp

Description (Optional):

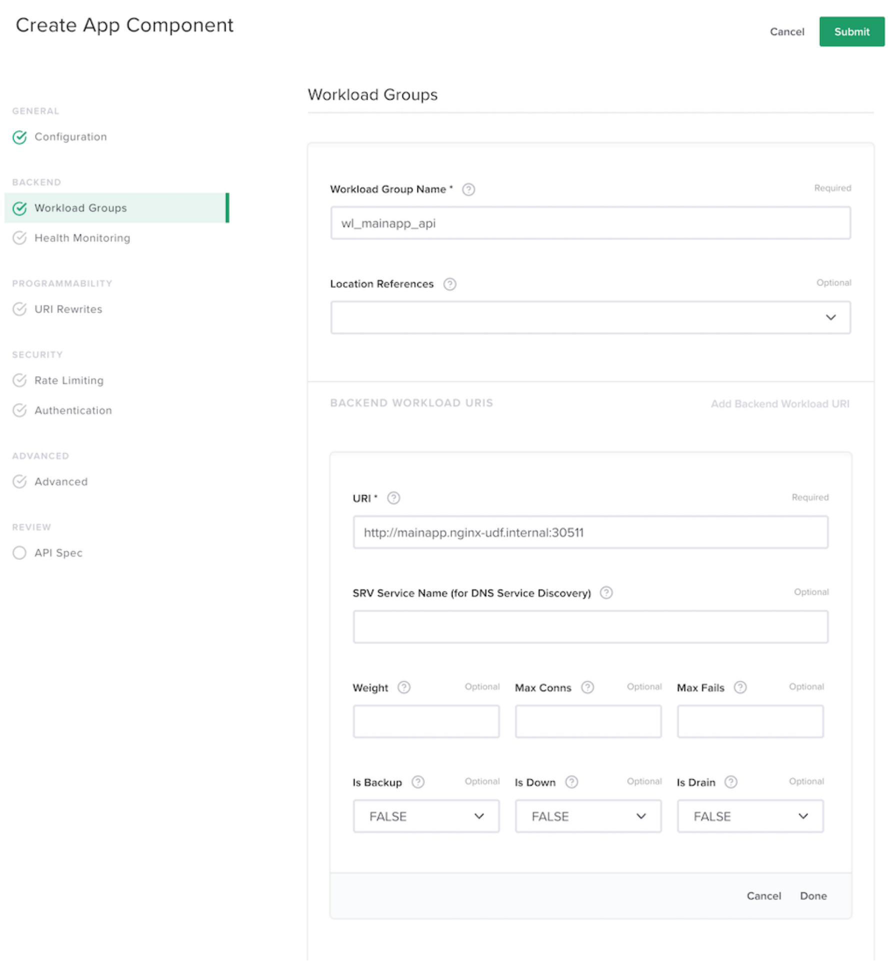
Tags (Optional): Ex: tag1, tag2, ...

Error Response Format (Optional):

Error Log: ✕

Access Log: ✕

Log Format (Optional):



Workload Group Name * — Required: wl_mainapp_api

Location References — Optional:

URI * — Required: http://mainapp.nginx-udf.internal:30511

SRV Service Name (for DNS Service Discovery) — Optional:

Weight — Optional:
Max Conns — Optional:
Max Fails — Optional:

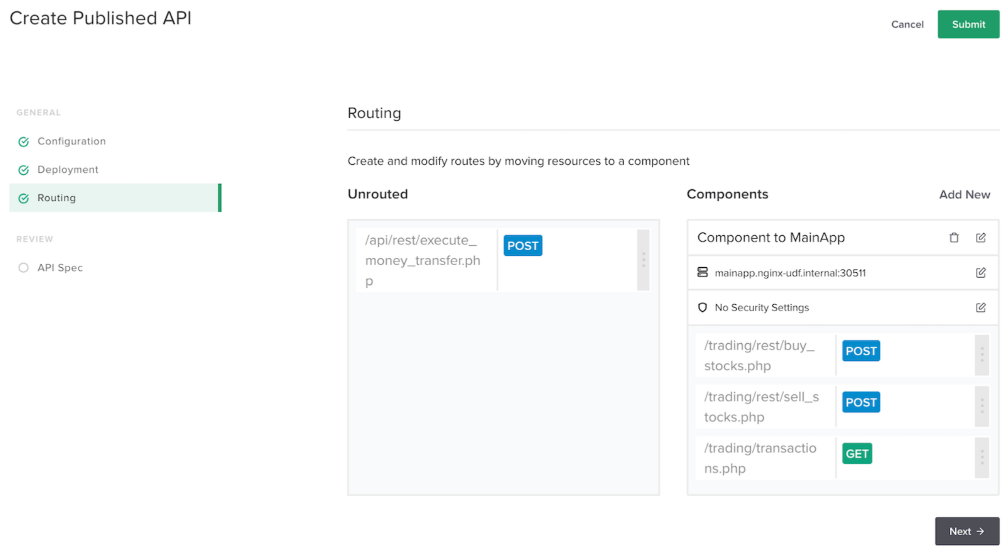
Is Backup — Optional: FALSE
Is Down — Optional: FALSE
Is Drain — Optional: FALSE

Note: Click `Done` then click `Next`

Note: We only configure one workload as the API we will test is hosted in the main app K8S service (sell stocks and buy stocks)

2. Click `Next` until the end and click `Submit`

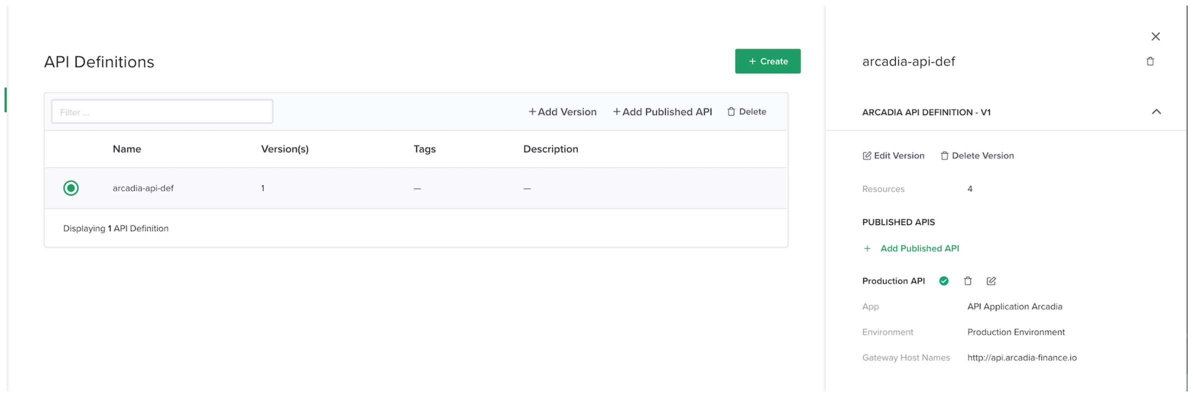
7. Now, drag and drop the 3 URI starting by `/trading` to the right `Component MainApp`



Create and modify routes by moving resources to a component

Unrouted	Components
/api/rest/execute_money_transfer.php POST	Component to MainApp
	mainapp.nginx-udf.internal:30511
	No Security Settings
	/trading/rest/buy_stocks.php POST
	/trading/rest/sell_stocks.php POST
	/trading/transactions.php GET

8. Click `Next` and `Submit`



Name	Version(s)	Tags	Description
arcadia-api-def	1	–	–

Displaying 1 API Definition

Resources	4

PUBLISHED APIS

Production API

App	API Application Arcadia
Environment	Production Environment
Gateway Host Names	http://api.arcadia-finance.io

Step 4 - Test your API

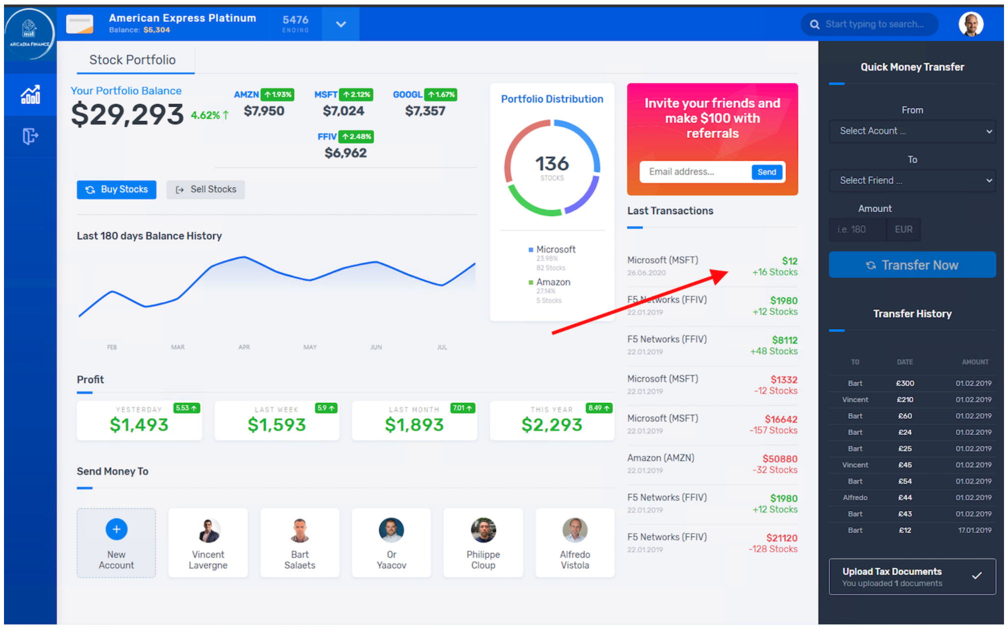
RDP to the jumphost

login: user

password: user

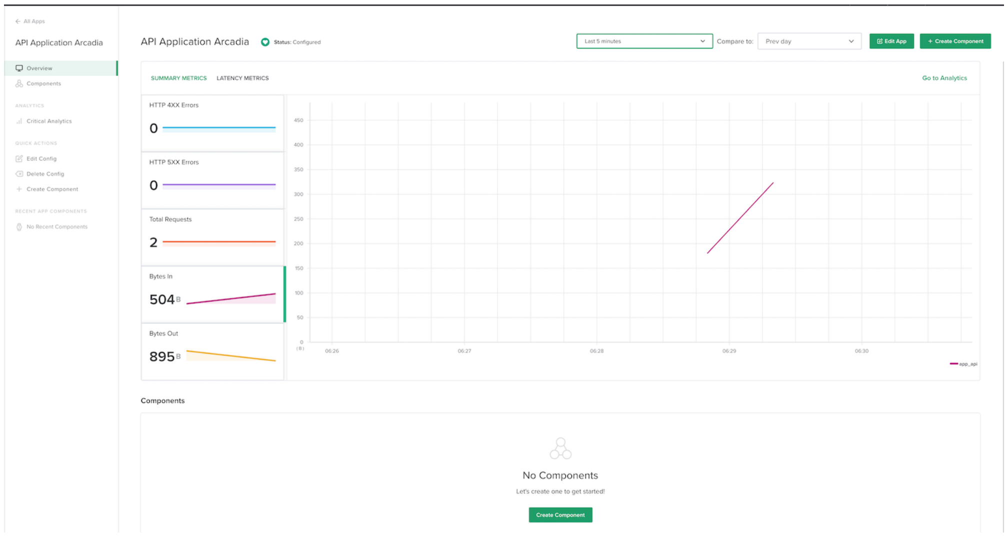
1. Open `Postman`
2. Open up the collection `Arcadia API`
3. Make 2 calls
   1. Last transactions
   2. POST Buy Stocks
4. Both works and are routed to the `MainApp pod` in K8S thanks to the NIGNX+ API GW.

5. You can check in the Web Application in `Chrome` if your Buy Stock call passed. It should appear in the last transaction GUI.



Step 5 - Look at the analytics

1. In the controler GUI
2. Click on the left icon `Apps`
3. Click on your `API Application Arcadia`
4. You can see your analytics for this API

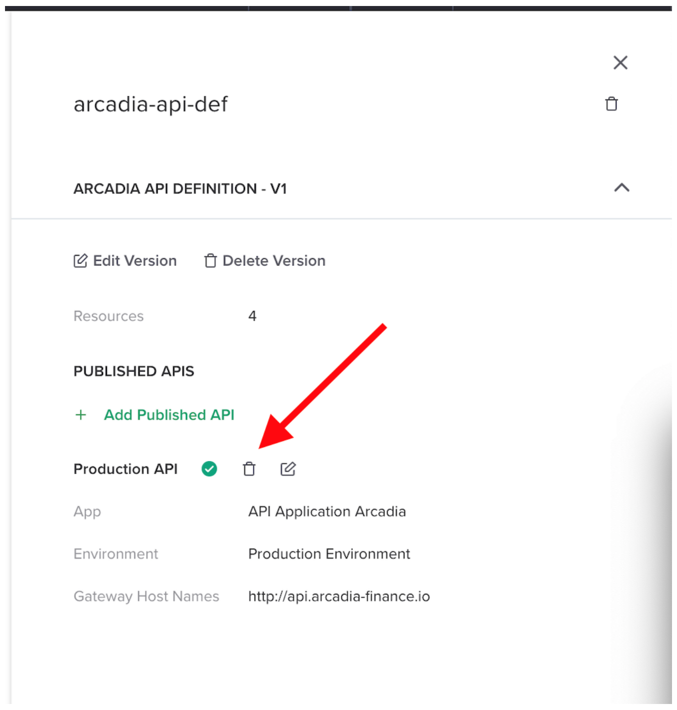


1.3.2 Module 2 - Publish API with OAS3 spec file via API

Note: We will clean up the previous lab in order to configure exactly the same objetcts but with API calls only.

Clean up the APIm configuration

1. In the controller GUI, go to `APIs` left menu and click on the existing API definintion `arcadia-api-def`
2. On the right panel
   1. Delete the `Production API` by clicking the `trash` button



arcadia-api-def

ARCADIA API DEFINITION - V1

Edit Version   Delete Version

Resources	4

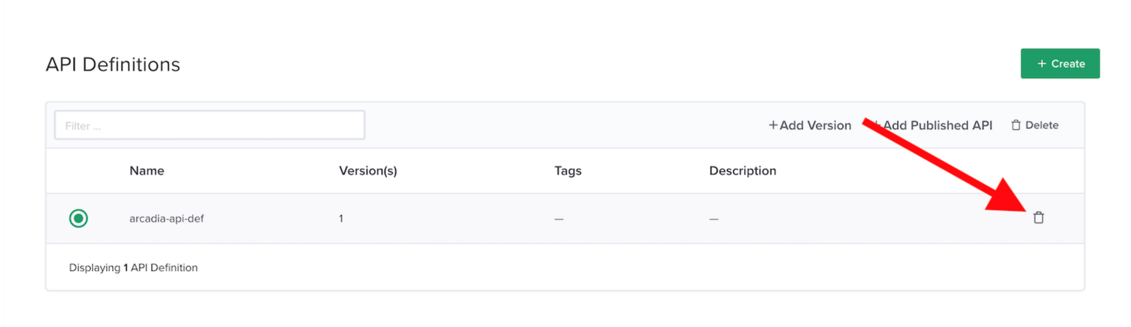
PUBLISHED APIS

+ Add Published API

Production API

App	API Application Arcadia
Environment	Production Environment
Gateway Host Names	http://api.arcadia-finance.io

   2. Delete the API definition `arcadia-api-def`



API Definitions

+ Create

Filter ...   +Add Version   +Add Published API   Delete

	Name	Version(s)	Tags	Description
●	arcadia-api-def	1	–	–

Displaying 1 API Definition

Note: Your API Definitions environment is clean.

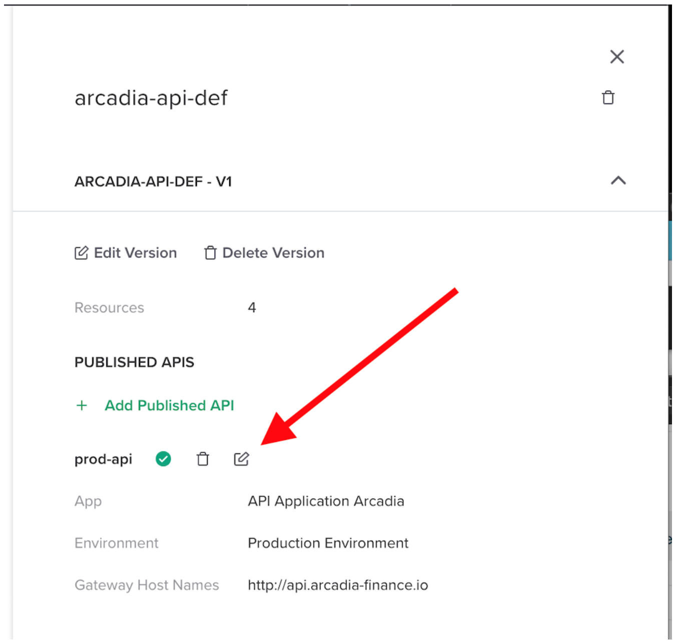
Create and publish an API Definition with the Controller API

Note: We will execute exactly the same job but by using the NGINX Controller control plane only. No GUI.

1. Connect to the Jumphost (user / user)
2. Launch `Postman`
    1. Open `Arcadia OAS` collection
    2. And `run` all calls from top to bottom

Note: For every call, check what is happening in the controller GUI

3. At then end, you should have the same results as the previous lab.
4. Edit the `Published API`



arcadia-api-def

ARCADIA-API-DEF - V1

Edit Version   Delete Version

Resources	4

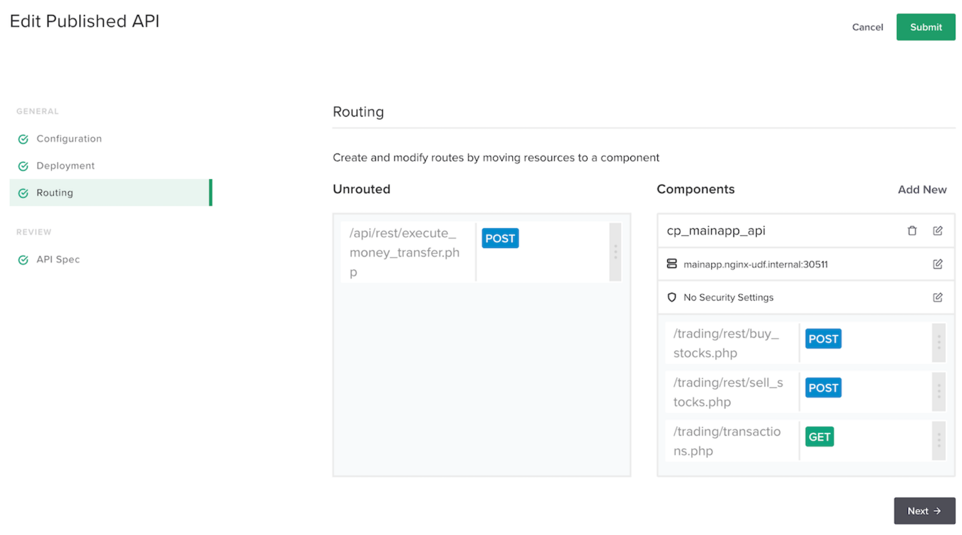
PUBLISHED APIS

+ Add Published API

prod-api

App	API Application Arcadia
Environment	Production Environment
Gateway Host Names	http://api.arcadia-finance.io

5. Check the `Routing`. You can see the routes are imported from the OAS3 file and the mapping is done with the components.


Edit Published API
Cancel
Submit
GENERAL
Configuration
Deployment
Routing
REVIEW
API Spec
Routing
Create and modify routes by moving resources to a component
Unrouted
/api/rest/execute_money_transfer.php
POST
Components
Add New
cp_mainapp_api
mainapp.nginx-udf.internal:30511
No Security Settings
/trading/rest/buy_stocks.php
POST
/trading/rest/sell_stocks.php
POST
/trading/transactions.php
GET
Next

6. Make a quick test with the `Arcadia API` postman collection

Warning: Check the call `Import API Definition OAS3`, we imported an OAS3 YAML File directly in the Controller with all the definitions and documentations

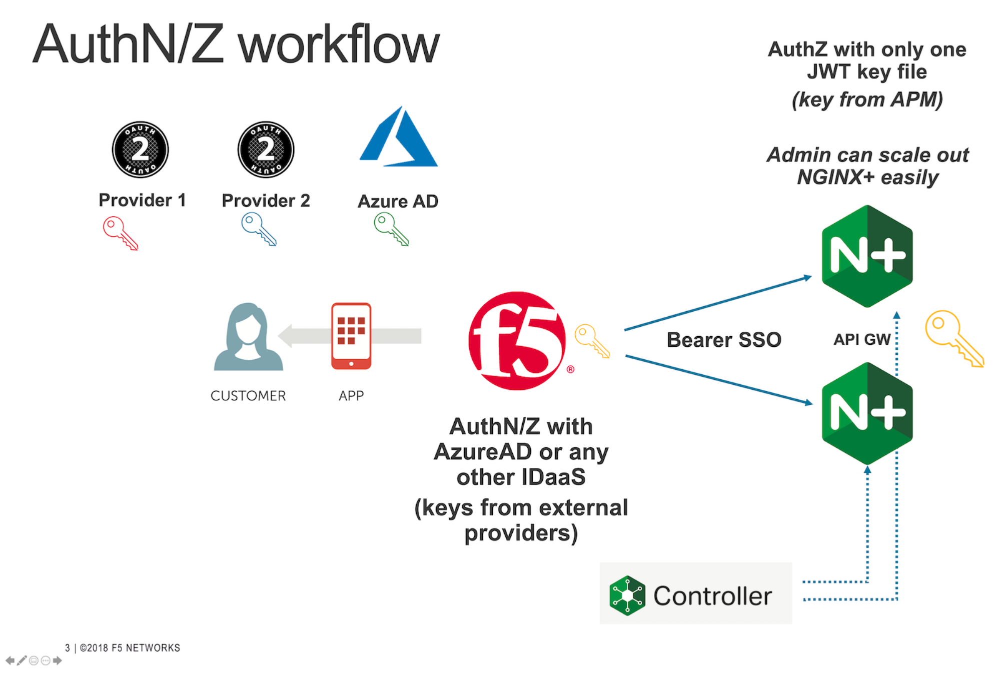
Note: In a near future, we will learn more on API definition versions

Warning: As you can notice, there is no security applied on the `Component`. Let's move to the next lab to assign a BIG-IP and Controller security policy

1.3.3 Module 3 - Protect Arcadia API with Adv. Waf and APM (Bearer SSO)

In this lab we will deploy a BIG-IP security policy based on Adv. WAF and APM, in front of the NGINX+ API GW. In order to make life better and simple for DevOps, we will delegate all the Authentication layer to APM. APM will authenticate JWT tokens coming from different providers with different keys, and we will use APM Bearer SSO in order to share a unique JWT key with the API gateways.

Note: APM will download keys from external providers automatically (by using OIDC discovery process) and will use another an unique key for internal SSO with NGINX API Gateways. This will allow DevOps to know only one key for all their deployments. And SecOps will manage the external providers.

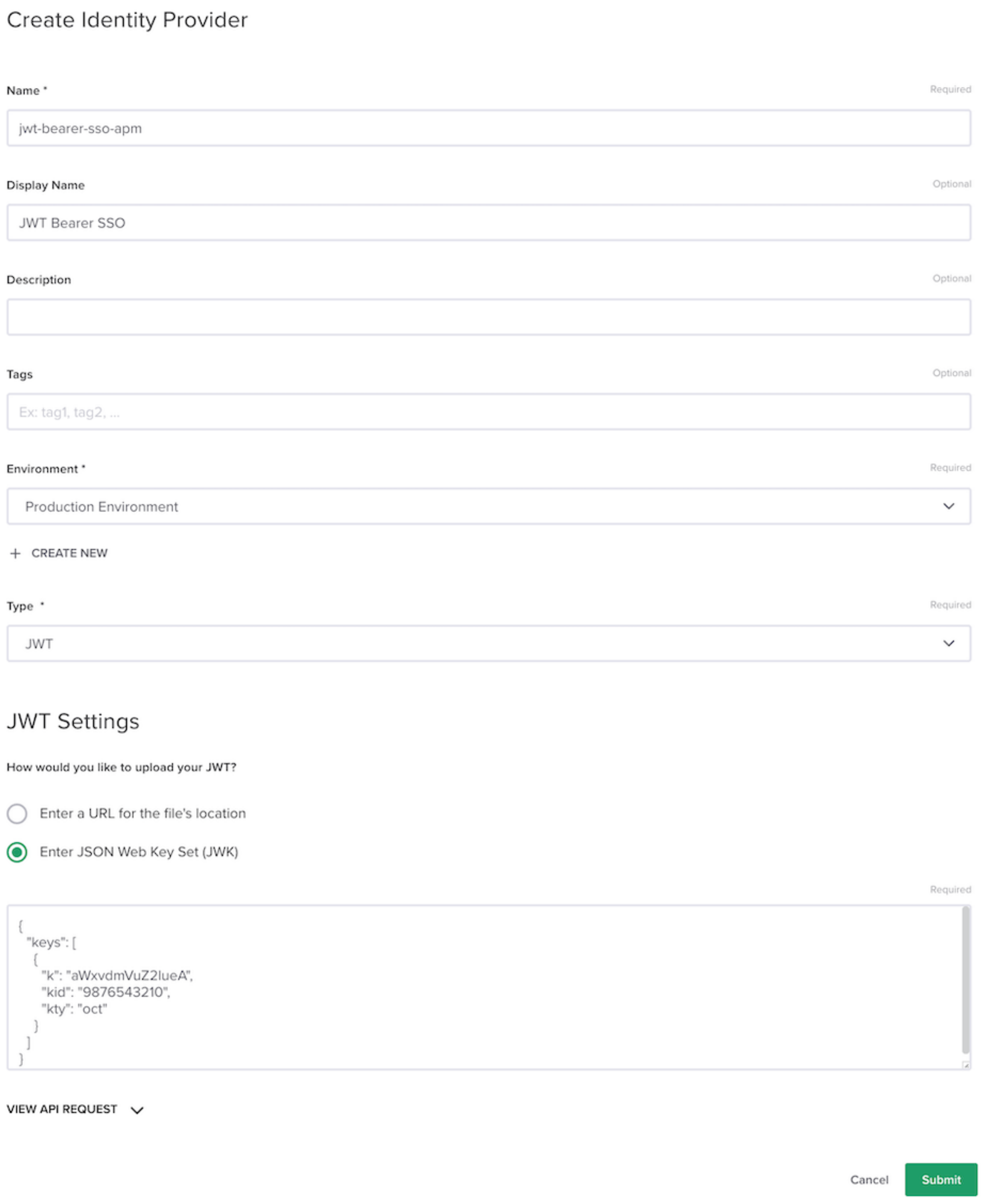

AuthN/Z workflow
Provider 1
Provider 2
Azure AD
CUSTOMER
APP
AuthN/Z with AzureAD or any other IDaaS (keys from external providers)
Bearer SSO
AuthZ with only one JWT key file (key from APM)
Admin can scale out NGINX+ easily
API GW
Controller
3 | ©2018 F5 NETWORKS

Configure NGINX Controller with a new Identity Provider

1. In the left menu, click on `Identity Provider` icon
2. Create a new Identity Provider as below. Use the JSON code below for the JWK

```
{
"keys": [
        {
        "k": "aWxvdmVuZ2lueA",
        "kid": "9876543210",
        "kty": "oct"
        }
    ]
}
```

Note: I invite you to decode the "k" value to know what is the `key`. As you can notice, we don't use a RSA key, but a secret (just to simplify the lab). This secret is BASE64 encoded.


Create Identity Provider

Name * (Required): jwt-bearer-sso-apm

Display Name (Optional): JWT Bearer SSO

Description (Optional):

Tags (Optional): Ex: tag1, tag2, ...

Environment * (Required): Production Environment

+ CREATE NEW

Type * (Required): JWT

JWT Settings

How would you like to upload your JWT?

( ) Enter a URL for the file's location

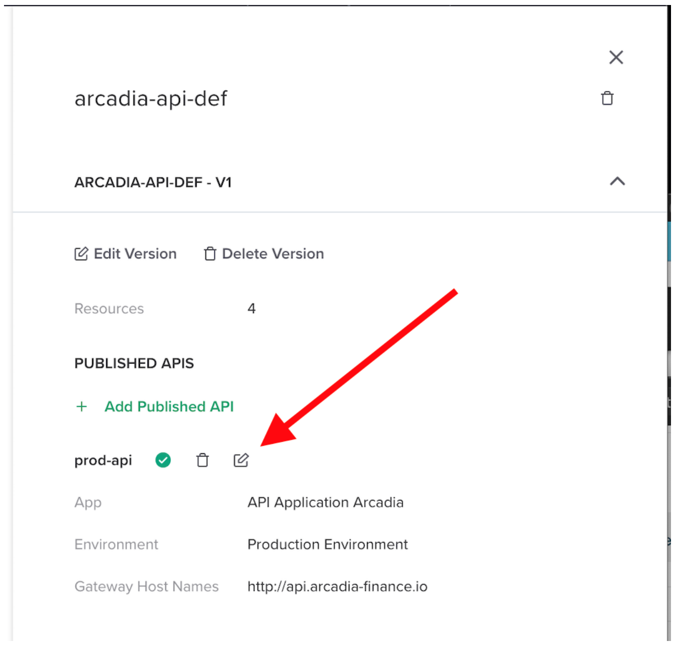
(●) Enter JSON Web Key Set (JWK)

Required

```
{
 "keys": [
  {
    "k": "aWxvdmVuZ2lueA",
    "kid": "9876543210",
    "kty": "oct"
  }
 ]
}
```

3. Assign this `Identity Provider` with your API Definition

   1. Get back to your `API definition` and `edit` the `Published API`


arcadia-api-def

ARCADIA-API-DEF - V1

Edit Version   Delete Version

Resources   4

PUBLISHED APIS

+ Add Published API

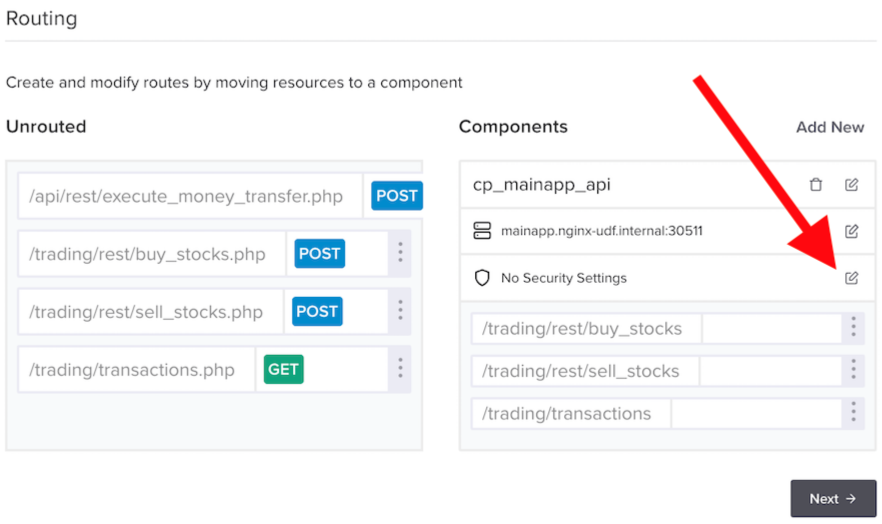
prod-api

App   API Application Arcadia

Environment   Production Environment

Gateway Host Names   http://api.arcadia-finance.io

2. Click on `routing` and edit the `Security Settings`


Routing

Create and modify routes by moving resources to a component

Unrouted

/api/rest/execute_money_transfer.php POST

/trading/rest/buy_stocks.php POST

/trading/rest/sell_stocks.php POST

/trading/transactions.php GET

Components   Add New

cp_mainapp_api

mainapp.nginx-udf.internal:30511

No Security Settings

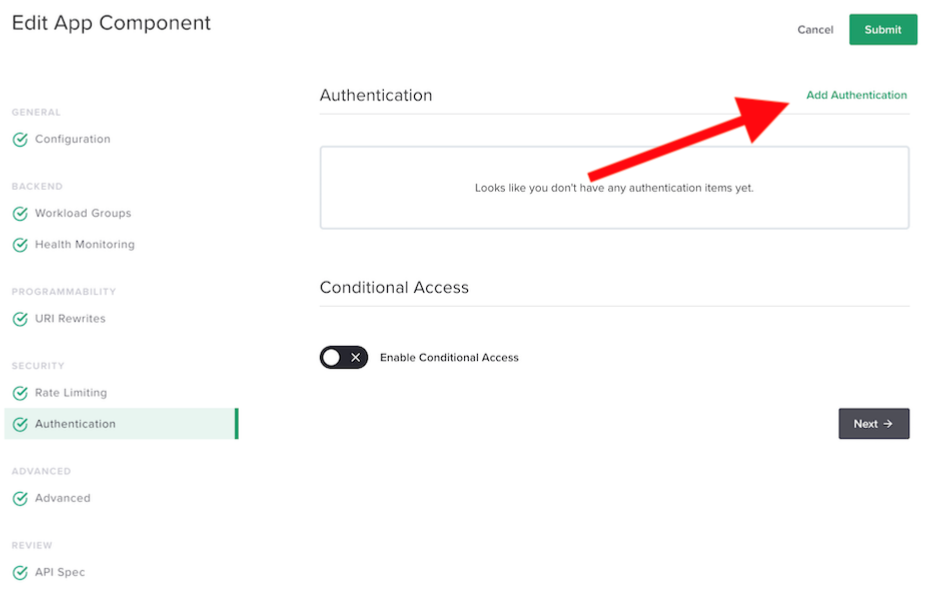
/trading/rest/buy_stocks

/trading/rest/sell_stocks

/trading/transactions

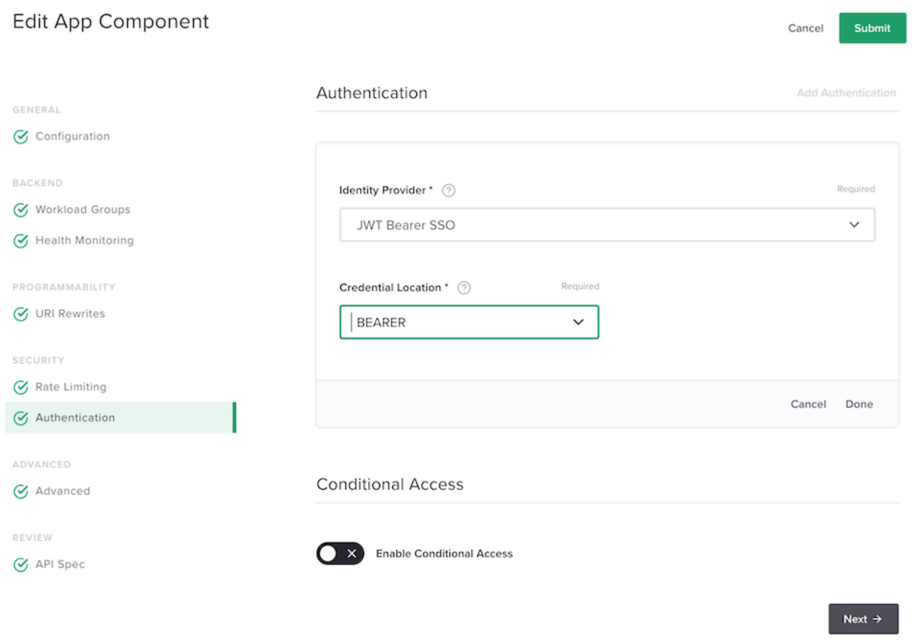
Next →

3. Click on `Add Authentication`



Looks like you don't have any authentication items yet.

4. Select the provider created previously `JWT Bearer SSO` and `Bearer`



Identity Provider * — Required: JWT Bearer SSO

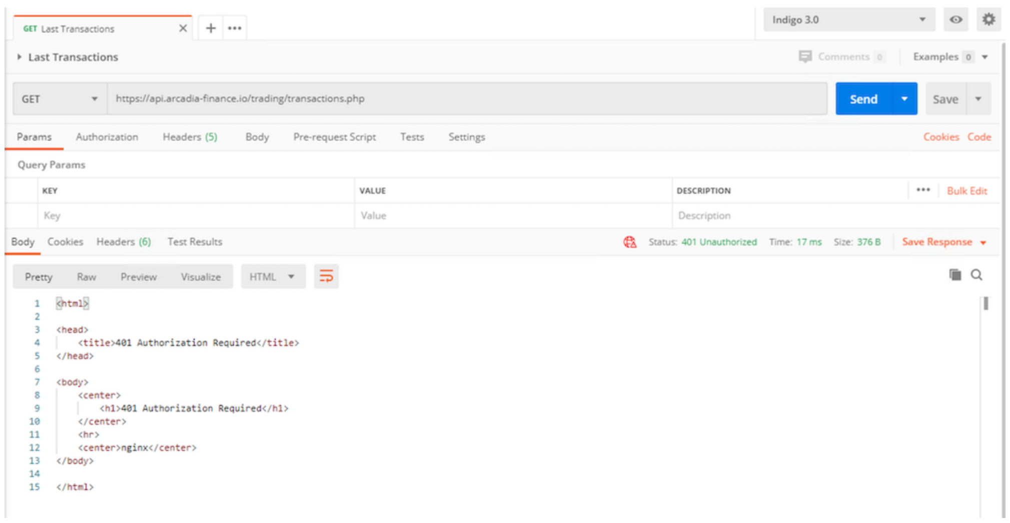
Credential Location * — Required: BEARER

5. Click `Done` and `Submit`

6. Click `Submit` again

4. Make a quick test with `Postman` by sending a request to the Arcadia API like `Last Transactions` or `Buy stocks`

   1. You can see a `401 Unauthorized`



GET https://api.arcadia-finance.io/trading/transactions.php

Status: 401 Unauthorized Time: 17 ms Size: 376 B

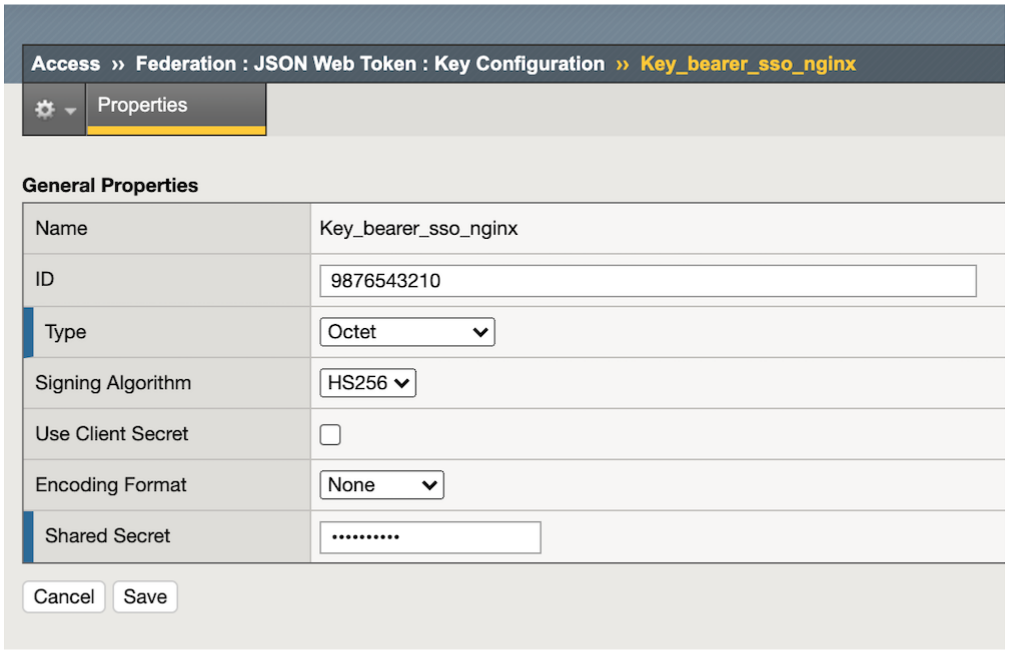
```
<html>

<head>
    <title>401 Authorization Required</title>
</head>

<body>
    <center>
        <h1>401 Authorization Required</h1>
    </center>
    <hr>
    <center>nginx</center>
</body>

</html>
```

Note: As you don't insert any JWT token in your request, the API GW rejected the request. It is time to configure APM to inject this JWT Bearer SSO

Configure Adv. WAF and APM

Note: In this lab we will use Access Guided Configuration and we will do some custom tuning in the policies. There are several ways to protect API with BIG-IP, but at the moment, we will focus on AGC so that you can understand how it works. GSA team is working on a dedicated UDF Blueprint for API Declarative WAF policy with v16.0

1. Connect to the Jumhost (user / user)
2. Open `Chrome` and connect to the BIG-IP (admin / admin)
3. Delete the existing `vs-arcadia-api` Virtual Server in the BIG-IP. We are going to create a new one from the Guided Configuration.
4. Create a JWK Bearer SSO key. If you remember below, the key (encoded64) was `aWxvdmVuZ2lueA`, and decoded64 `ilovenginx`
    1. Click `Access` > `Federation` > `JSON Web Token` > `Key Configuration`
    2. Create a new key as below with the value `ilovenginx` as Shared Secret


Access » Federation : JSON Web Token : Key Configuration » Key_bearer_sso_nginx

Properties

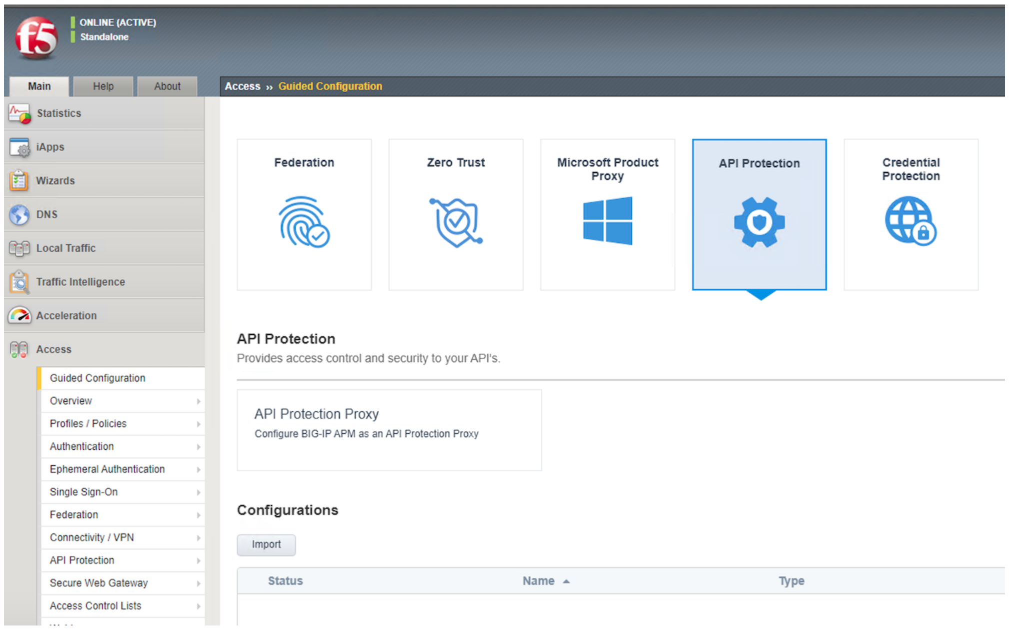
General Properties

Name	Key_bearer_sso_nginx
ID	9876543210
Type	Octet
Signing Algorithm	HS256
Use Client Secret	☐
Encoding Format	None
Shared Secret	••••••••••

Cancel Save

Warning: Don't forget to set an ID. It is mandatory in order to use this key in the Bearer SSO profile

5. In `Access`, click on `Guided Configuration` and select the template `API Protection Proxy` in `API Protection` group



API Protection

Provides access control and security to your API's.

API Protection Proxy

Configure BIG-IP APM as an API Protection Proxy

Configurations

Import

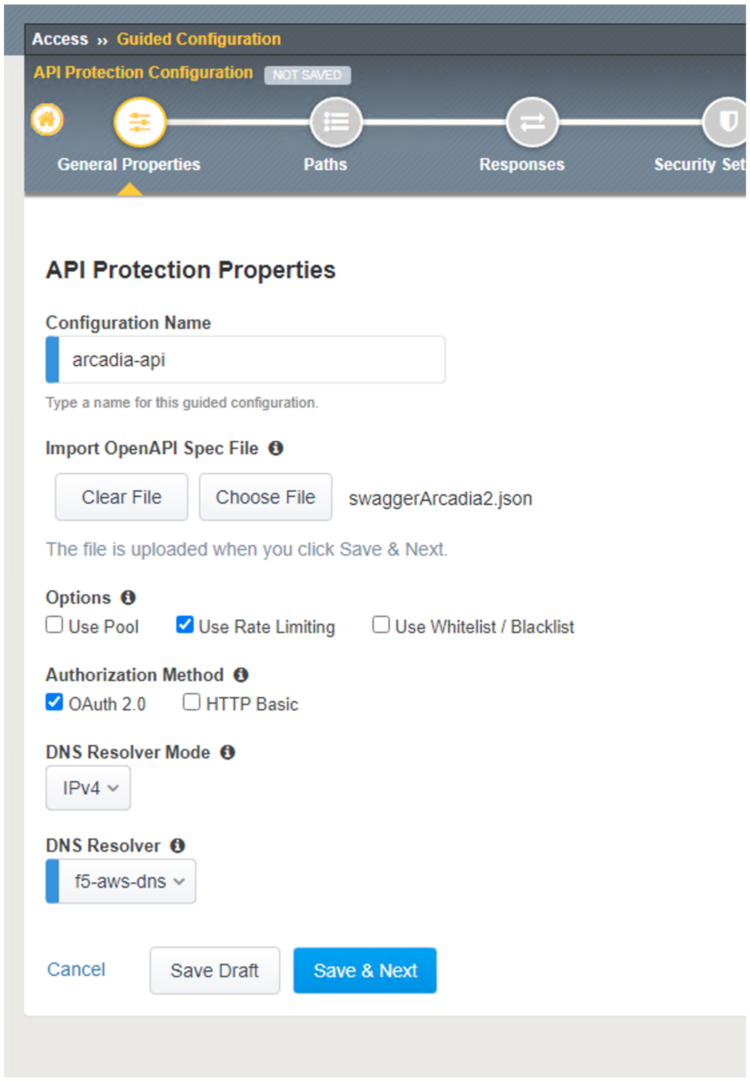
Status	Name	Type

6. Configure the template as below.

Warning: The AGC template does not support yet OpenAPI spec file Version 3. But only Version 2. We will use another version of the OAS file.

Note: The OAS file is located in `Downloads` directory and its name is `swaggerArcadia2.json`

1. Check the boxes `Use Rate Limiting` and `OAuth 2.0`



API Protection Properties

Configuration Name

arcadia-api

Type a name for this guided configuration.

Import OpenAPI Spec File

swaggerArcadia2.json

The file is uploaded when you click Save & Next.

Options

☐ Use Pool ☑ Use Rate Limiting ☐ Use Whitelist / Blacklist

Authorization Method

☑ OAuth 2.0 ☐ HTTP Basic

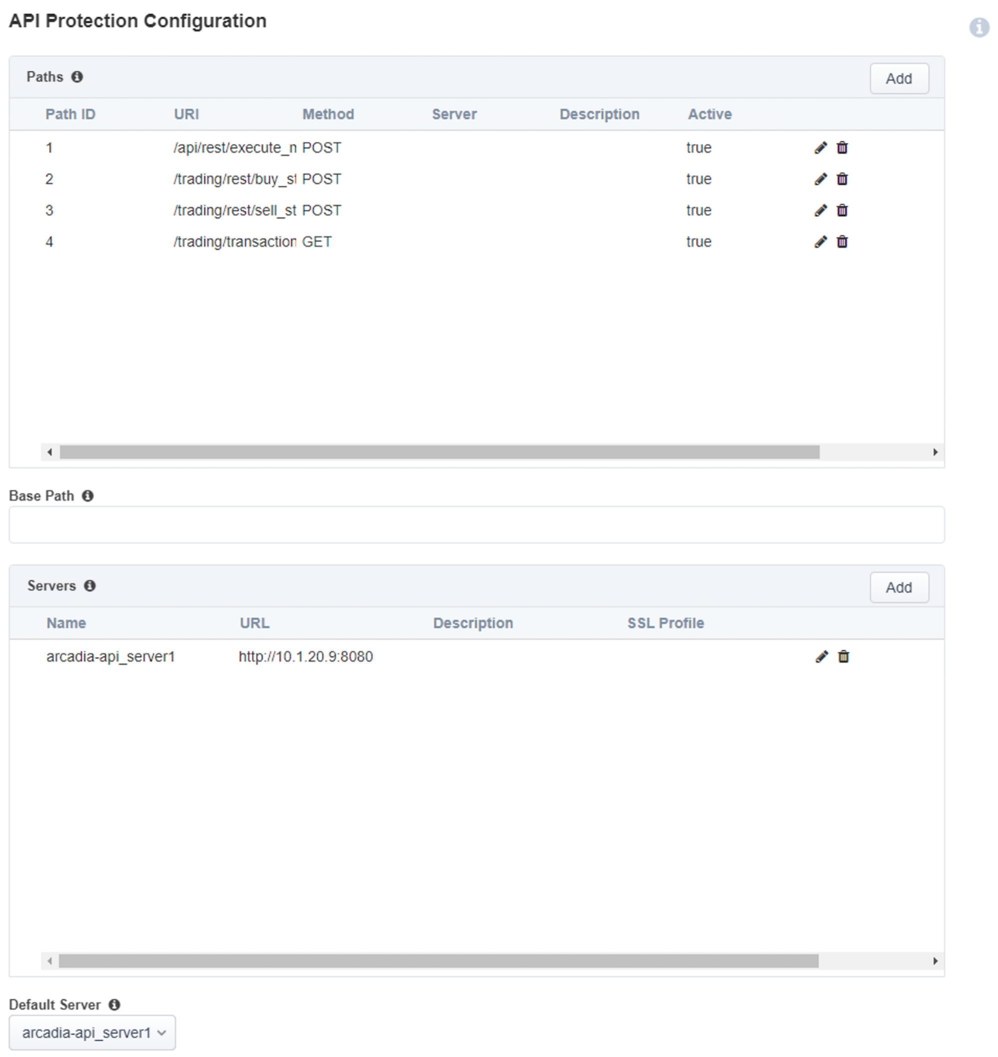
DNS Resolver Mode

IPv4

DNS Resolver

f5-aws-dns

- Select the default Servrer at the bottom of the screen



API Protection Configuration

Paths

Path ID	URI	Method	Server	Description	Active
1	/api/rest/execute_n	POST			true
2	/trading/rest/buy_st	POST			true
3	/trading/rest/sell_st	POST			true
4	/trading/transaction	GET			true

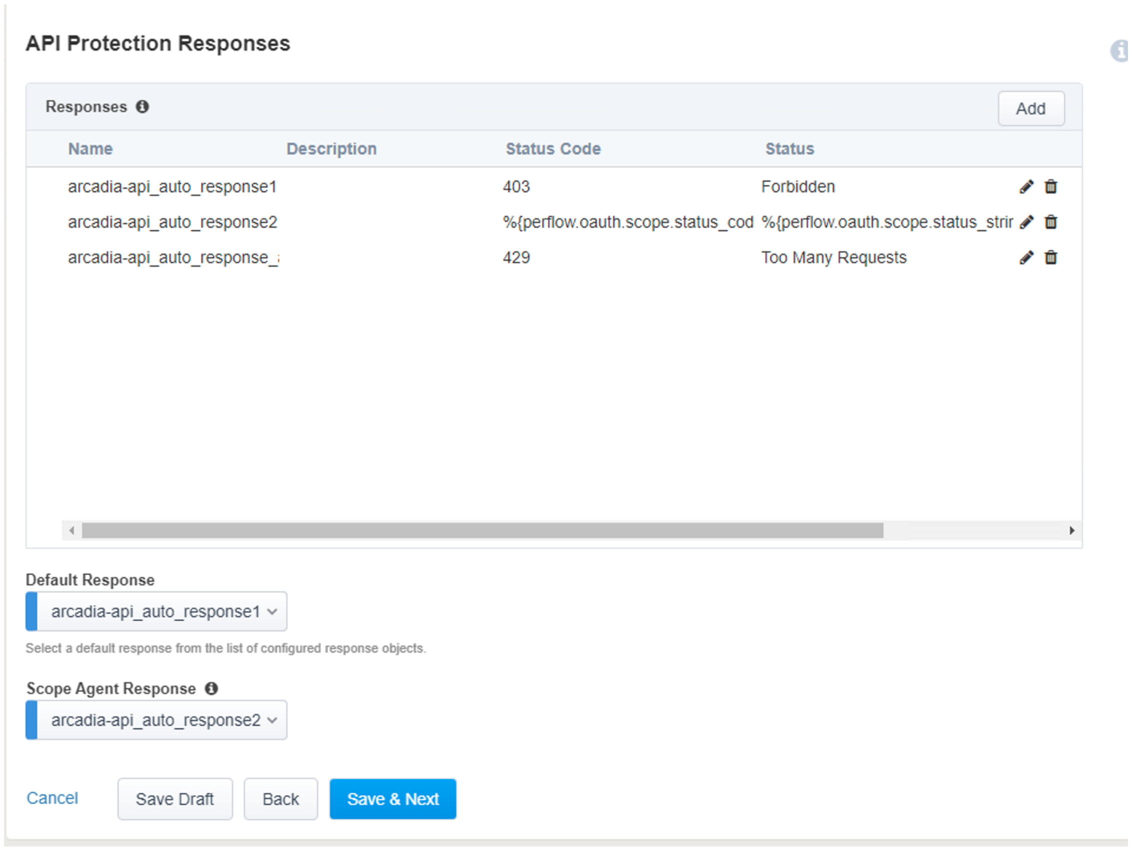
Base Path

Servers

Name	URL	Description	SSL Profile
arcadia-api_server1	http://10.1.20.9:8080		

Default Server: arcadia-api_server1

Note: You can notice the URI and the back server have been imported from the OAS2 file



API Protection Responses

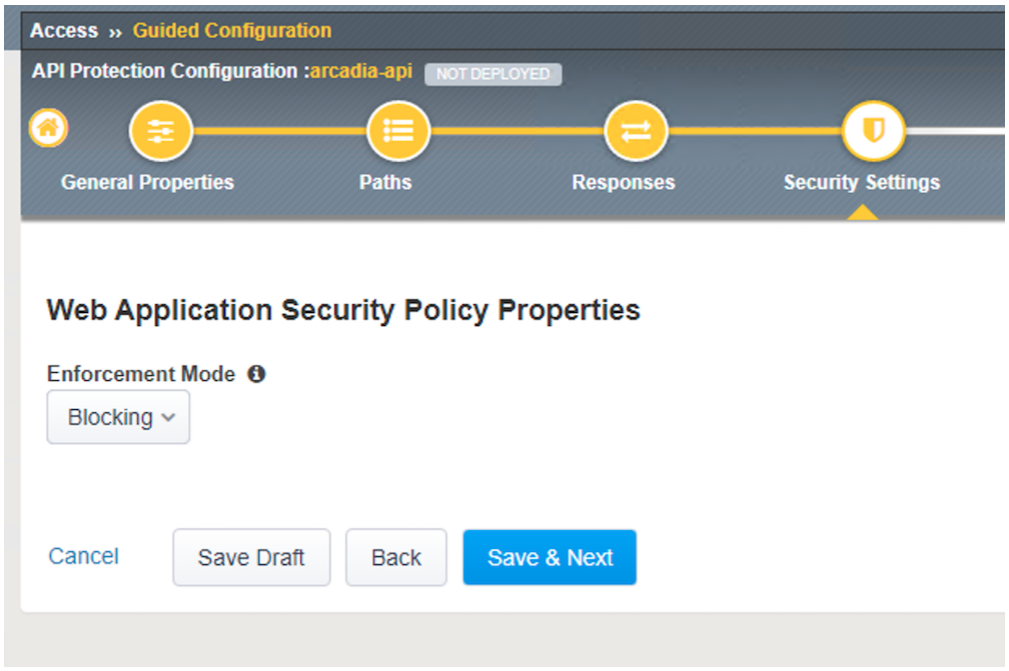
Responses

Name	Description	Status Code	Status
arcadia-api_auto_response1		403	Forbidden
arcadia-api_auto_response2		%{perflow.oauth.scope.status_cod	%{perflow.oauth.scope.status_strir
arcadia-api_auto_response_:		429	Too Many Requests

Default Response: arcadia-api_auto_response1

Select a default response from the list of configured response objects.

Scope Agent Response: arcadia-api_auto_response2



Access » Guided Configuration

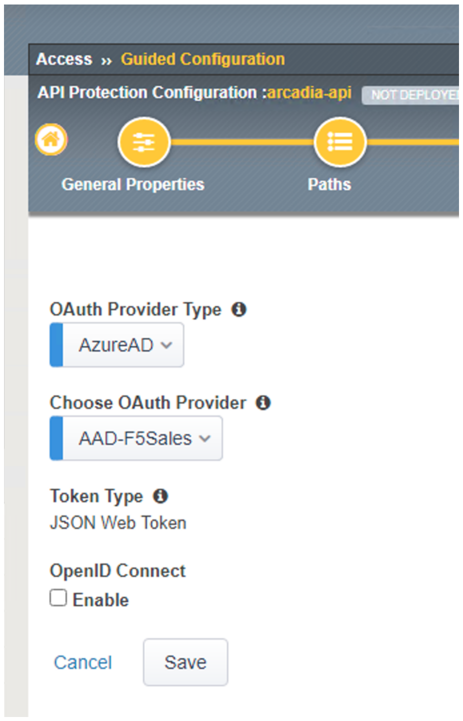
API Protection Configuration :arcadia-api NOT DEPLOYED

Web Application Security Policy Properties

Enforcement Mode: Blocking

2. Select `AzureAD AAD-F5Sales` as provider

> **Warning:** Due to a bug in AGC, we can't add more providers here. We will modify the list later on directly in the APM configuraiton (ID 835509)


Access » Guided Configuration
API Protection Configuration :arcadia-api
NOT DEPLOYE
General Properties
Paths

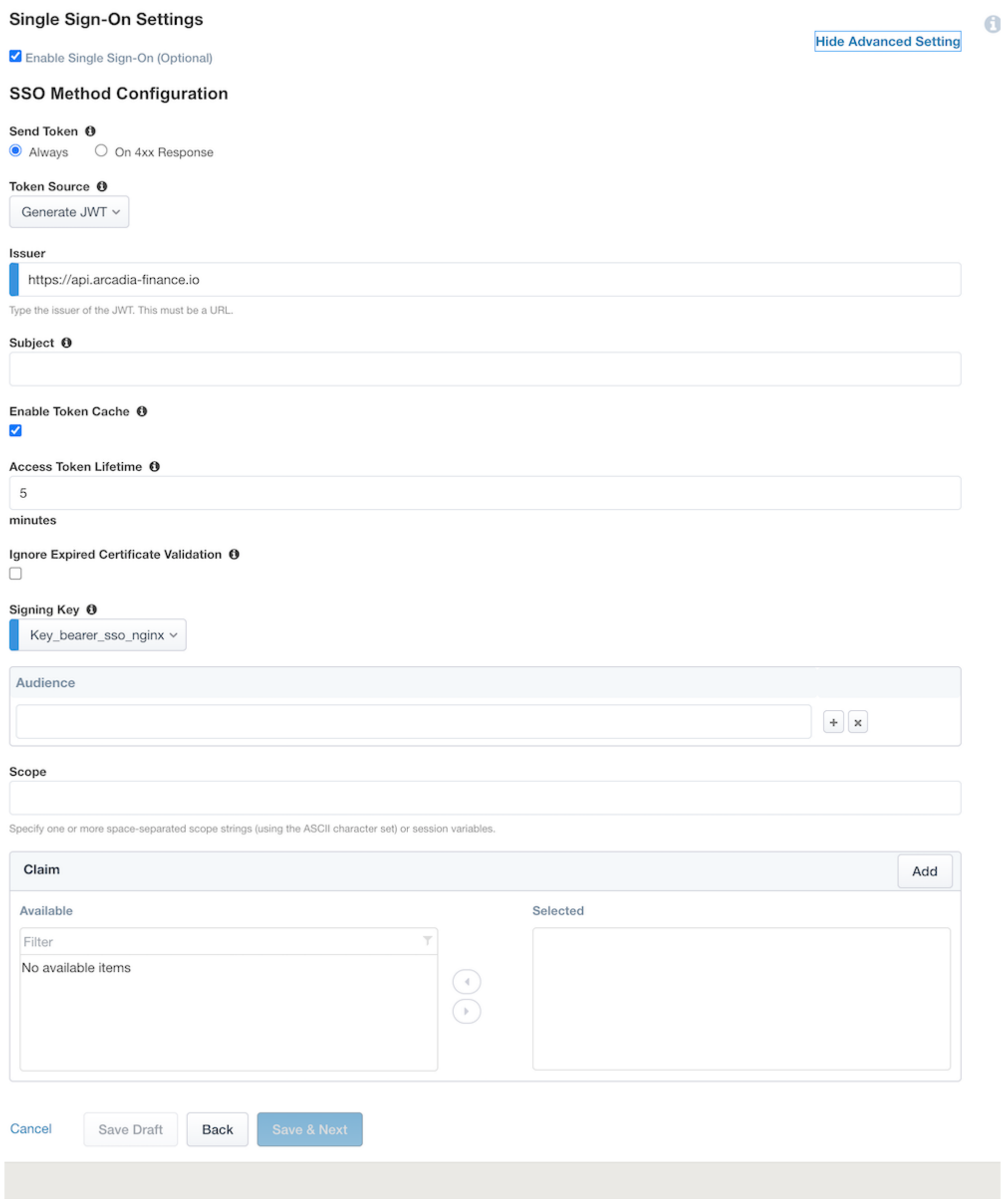
OAuth Provider Type: AzureAD

Choose OAuth Provider: AAD-F5Sales

Token Type: JSON Web Token

OpenID Connect: ☐ Enable

3. Configure `Signle Sign-On Settings` as below


Single Sign-On Settings

☑ Enable Single Sign-On (Optional)

SSO Method Configuration

Send Token: ● Always ○ On 4xx Response

Token Source: Generate JWT

Issuer: https://api.arcadia-finance.io

Type the issuer of the JWT. This must be a URL.

Subject:

Enable Token Cache: ☑

Access Token Lifetime: 5 minutes

Ignore Expired Certificate Validation: ☐

Signing Key: Key_bearer_sso_nginx

Audience:

Scope:

Specify one or more space-separated scope strings (using the ASCII character set) or session variables.

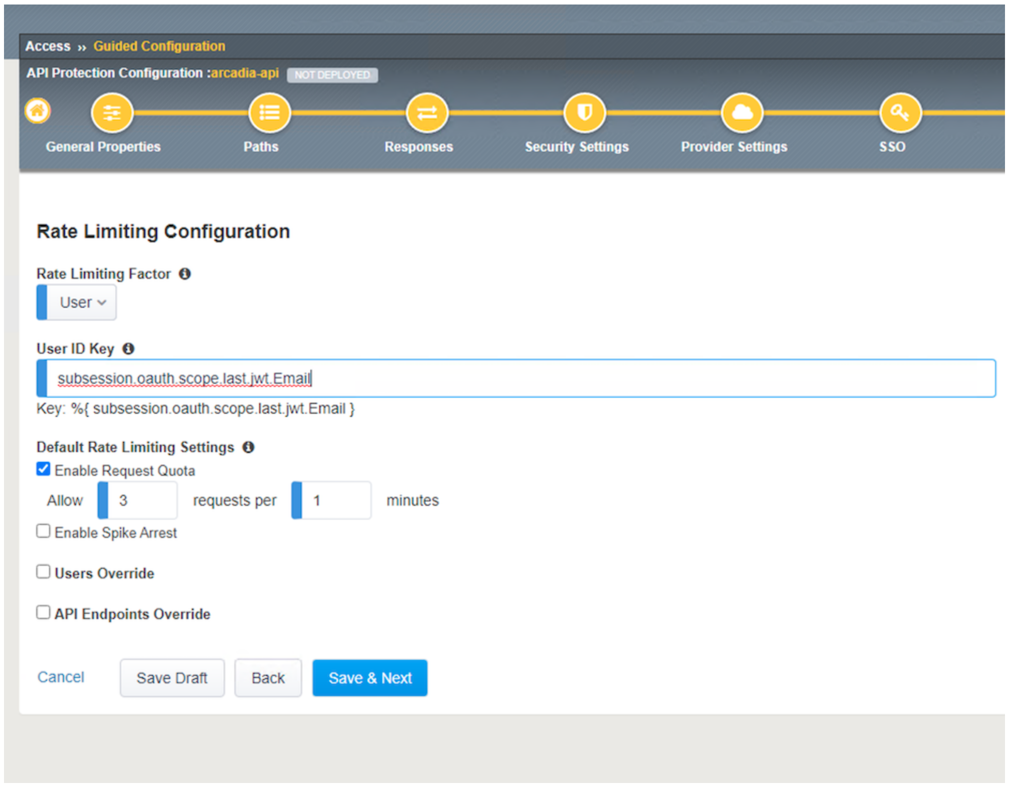
Claim

Available: No available items

Selected:

Note: We will focus on Claims later on

4. Configure `Rate Limiting` as below. We will limit request per user based on their Email address extracted from the JWT token. The value used for the `User ID Key` is `subsession.oauth.scope.last.jwt.Email`


Access » Guided Configuration
API Protection Configuration :arcadia-api NOT DEPLOYED
General Properties
Paths
Responses
Security Settings
Provider Settings
SSO

Rate Limiting Configuration

Rate Limiting Factor: User

User ID Key: subsession.oauth.scope.last.jwt.Email

Key: %{ subsession.oauth.scope.last.jwt.Email }

Default Rate Limiting Settings

[✓] Enable Request Quota

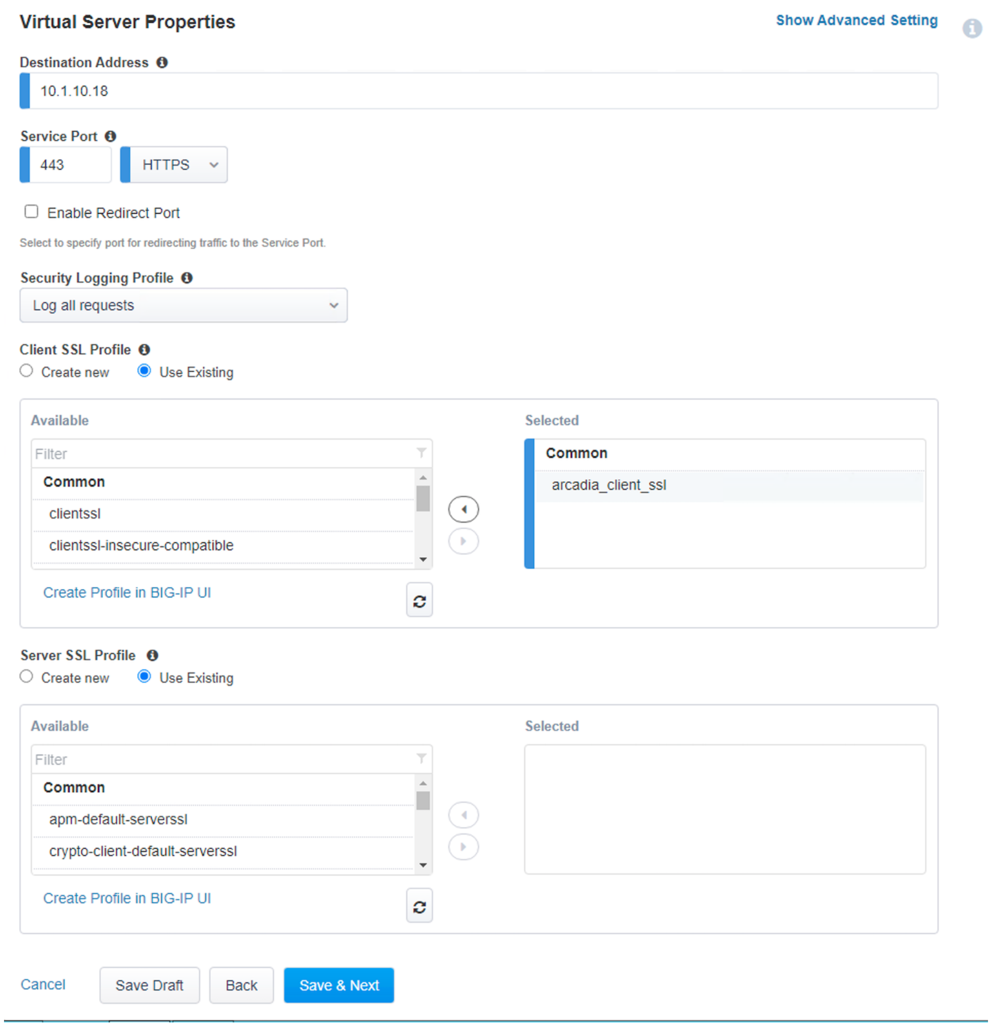
Allow 3 requests per 1 minutes

[ ] Enable Spike Arrest

[ ] Users Override

[ ] API Endpoints Override

1. Configure the `Virtual Server` as below
   - VS : 10.1.10.18
   - Log All Requests
   - Client SSL arcadia_client_ssl



Virtual Server Properties

Show Advanced Setting

Destination Address: 10.1.10.18

Service Port: 443 HTTPS

[ ] Enable Redirect Port

Select to specify port for redirecting traffic to the Service Port.

Security Logging Profile: Log all requests

Client SSL Profile: ( ) Create new [✓ Use Existing]

Available: Common — clientssl, clientssl-insecure-compatible

Selected: Common — arcadia_client_ssl

Create Profile in BIG-IP UI

Server SSL Profile: ( ) Create new [✓ Use Existing]

Available: Common — apm-default-serverssl, crypto-client-default-serverssl

Selected:

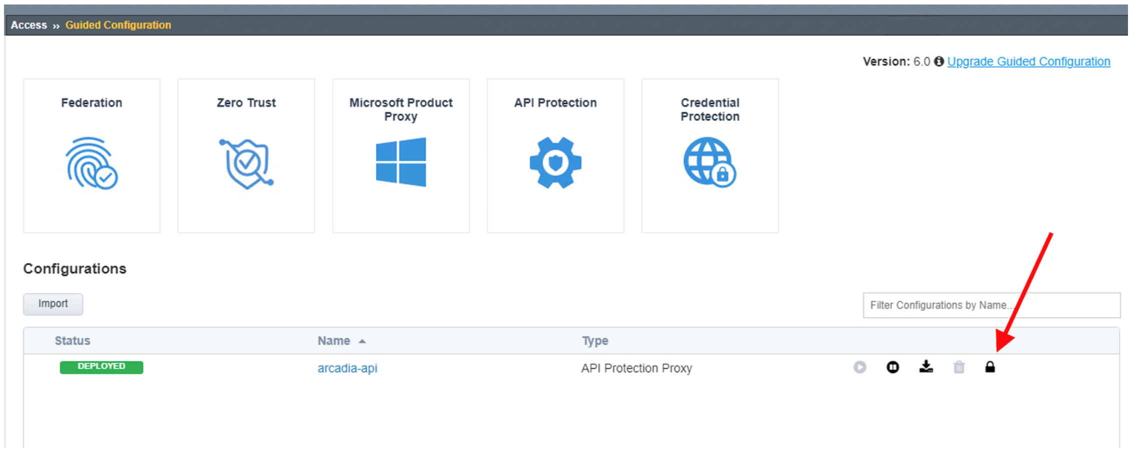
Create Profile in BIG-IP UI

Cancel | Save Draft | Back | Save & Next

2. Click `Deploy`

7. Now we have to add manually the 2 more providers in the APM configuration (due to the BUGID in AGC 6.0)

   1. `Unstrict` the configuration in AGC, by clicking on the `lock` icon and approve the change.

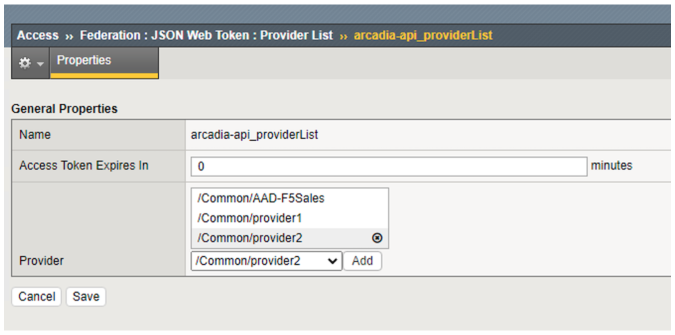


Access » Guided Configuration
Version: 6.0 Upgrade Guided Configuration
Federation
Zero Trust
Microsoft Product Proxy
API Protection
Credential Protection
Configurations
Import
Filter Configurations by Name...

Status	Name	Type
DEPLOYED	arcadia-api	API Protection Proxy

   2. Click `Access` > `Federation` > `JSON Web Token` > `Provider List` and `edit` the existing profile

3. Add `provider1` and `provider2` into the list



Access » Federation : JSON Web Token : Provider List » arcadia-api_providerList

Properties

General Properties

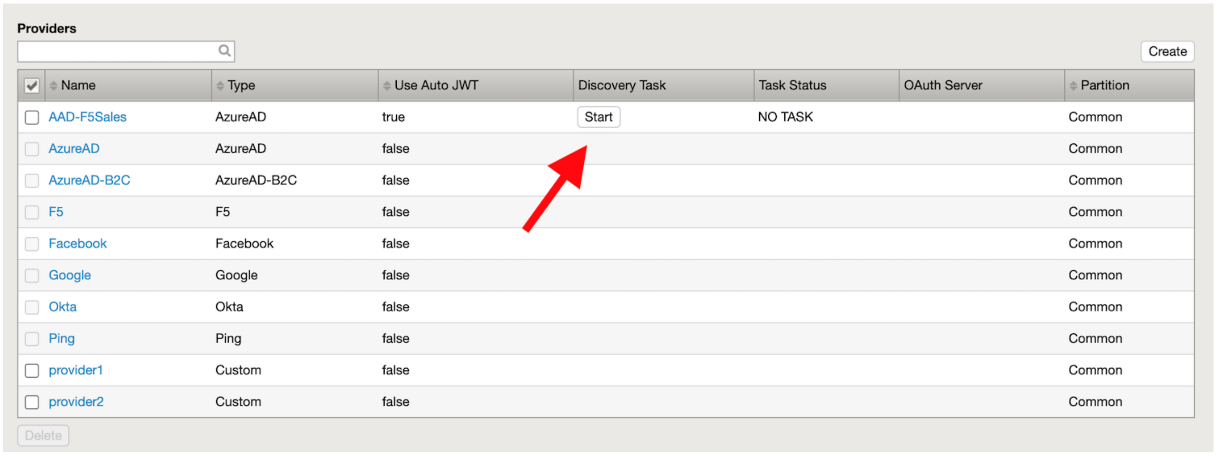
Name	arcadia-api_providerList
Access Token Expires In	0 minutes
Provider	/Common/AAD-F5Sales /Common/provider1 /Common/provider2 /Common/provider2 Add

Cancel Save

Note: Congratulation, Arcadia API is protected by an Advanced WAF (you can check the policy) and APM in order to authenticate requests from 3 providers.

Note: I invite you to check the Access > API Protection configuration

Warning: In order to use Oauth with Azure AD, you have to force an update of the Azure JWT keys. In `Federation > Oauth Client / Resource Server > Provider`, click on `Start` button to force APM to download the new keys.



Providers

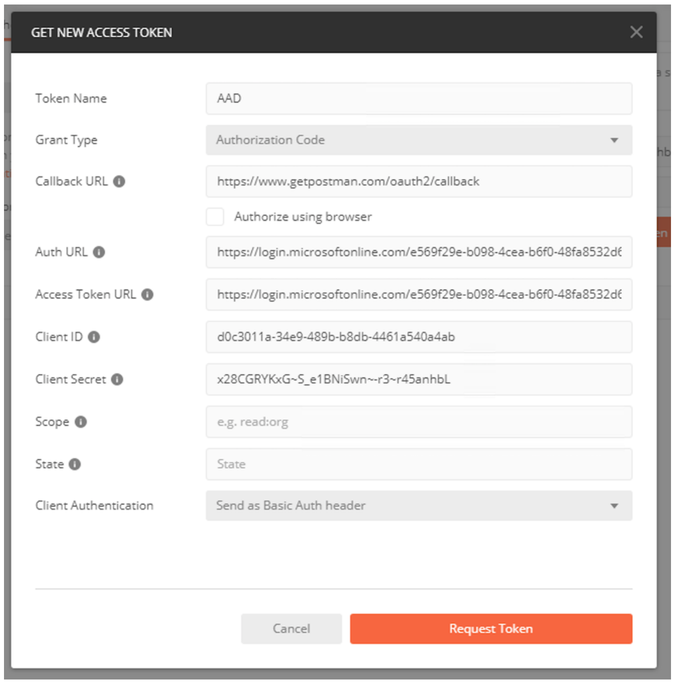
Create

Name	Type	Use Auto JWT	Discovery Task	Task Status	OAuth Server	Partition
AAD-F5Sales	AzureAD	true	Start	NO TASK		Common
AzureAD	AzureAD	false				Common
AzureAD-B2C	AzureAD-B2C	false				Common
F5	F5	false				Common
Facebook	Facebook	false				Common
Google	Google	false				Common
Okta	Okta	false				Common
Ping	Ping	false				Common
provider1	Custom	false				Common
provider2	Custom	false				Common

Delete

Test your protected API with Authentication, WAF and Rate Limiting

1. Open `Postman` and select the `Arcadia API` collection
2. Select one call, the one you want.
3. `F5ers only - For F5 partners and customers, please jump to the next bullet point.` In `authentication` select `Oauth 2.0`. We will start with an Azure AD provider - similating a partner having an AAD subcription and wanting to use it.
   1. Click `Get New Token`
   2. I have already set the values for the Oauth Client. As a reminder, here, Postman is the Oauth Agent - it is requesting the Access Token



GET NEW ACCESS TOKEN

Token Name	AAD
Grant Type	Authorization Code
Callback URL	https://www.getpostman.com/oauth2/callback
	Authorize using browser
Auth URL	https://login.microsoftonline.com/e569f29e-b098-4cea-b6f0-48fa8532d6
Access Token URL	https://login.microsoftonline.com/e569f29e-b098-4cea-b6f0-48fa8532d6
Client ID	d0c3011a-34e9-489b-b8db-4461a540a4ab
Client Secret	x28CGRYKxG~S_e1BNiSwn~-r3~r45anhbL
Scope	e.g. read:org
State	State
Client Authentication	Send as Basic Auth header

   3. Authenticate with your Corporate F5 account. If it fails, it means you are not part of the F5 Sales Azure tenant (Open an IT Ticket)
   4. When done, click `Use token` and send your request.

Note: It passes. Token is approved by APM, and a new token is generated by APM and sent to the NGINX API GW (Bearer SSO)

4. `Available for F5ers, partners and customers`. Now, try with the 2 other providers (partner1 and partner2)
   1. You can find the tokens on the desktop in the file `JWT tokens.txt`
   2. Don't use `Oauth 2.0`, as we already have the tokens. But use `Bearer Token` instead. I generated these tokens from the website http://jwtbuilder.jamiekurtz.com/

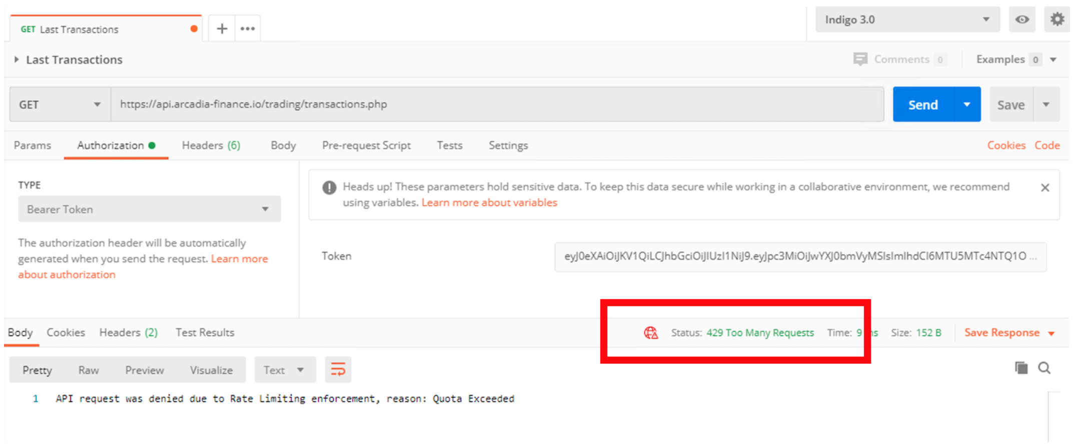
```
Partner 1:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJpc3MiOiJwYXJ0bmVyMSIsImlhdCI6MTU5MzQ1NTk4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGkuYXJjYW
↪JRboDfKWvSLVU3md6OULGifoVxJ-ryx7y-0DKrOlPOM
```

(continues on next page)

(continued from previous page)

```
Partner 2:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJpc3MiOiJwYXJ0bmVyMiIsImlhdCI6MTU5MzQ1NTk4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGkuYXJjYW
↪aqTxd6X4z7EFijJsyiuq8mZAKMLG519Bmjz1ra24L-s
```

5. Test the **Rate Limiting** by sending 4 calls with the same token. The 4th will be block. You can notice the reponse code `429 Too Many Requests`



GET Last Transactions

Last Transactions

GET https://api.arcadia-finance.io/trading/transactions.php

Authorization: TYPE Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJwYXJ0bmVyMSIsImlhdCI6MTU5MTc4NTQ1O ...

Status: 429 Too Many Requests Time: 9 s Size: 152 B

```
API request was denied due to Rate Limiting enforcement, reason: Quota Exceeded
```

6. Send an **attack**

   1. Select the call `POST Buy Stocks XSS attack`
   2. Send the request and notice the `200 OK` response. It means the WAF didn't block the request
   3. Check why and change your policy accordingly.

Note: Tip : attack signatures are in Staging mode

1.3.4 Module 4 - Fine grained access with NGINX Controller APIm module

In this lab, we will allow access to the Arcadia API, only for Manager Role. To do so, we will first check the JWT token claims and understand how to forward a claim from a provider into the Bearer SSO.

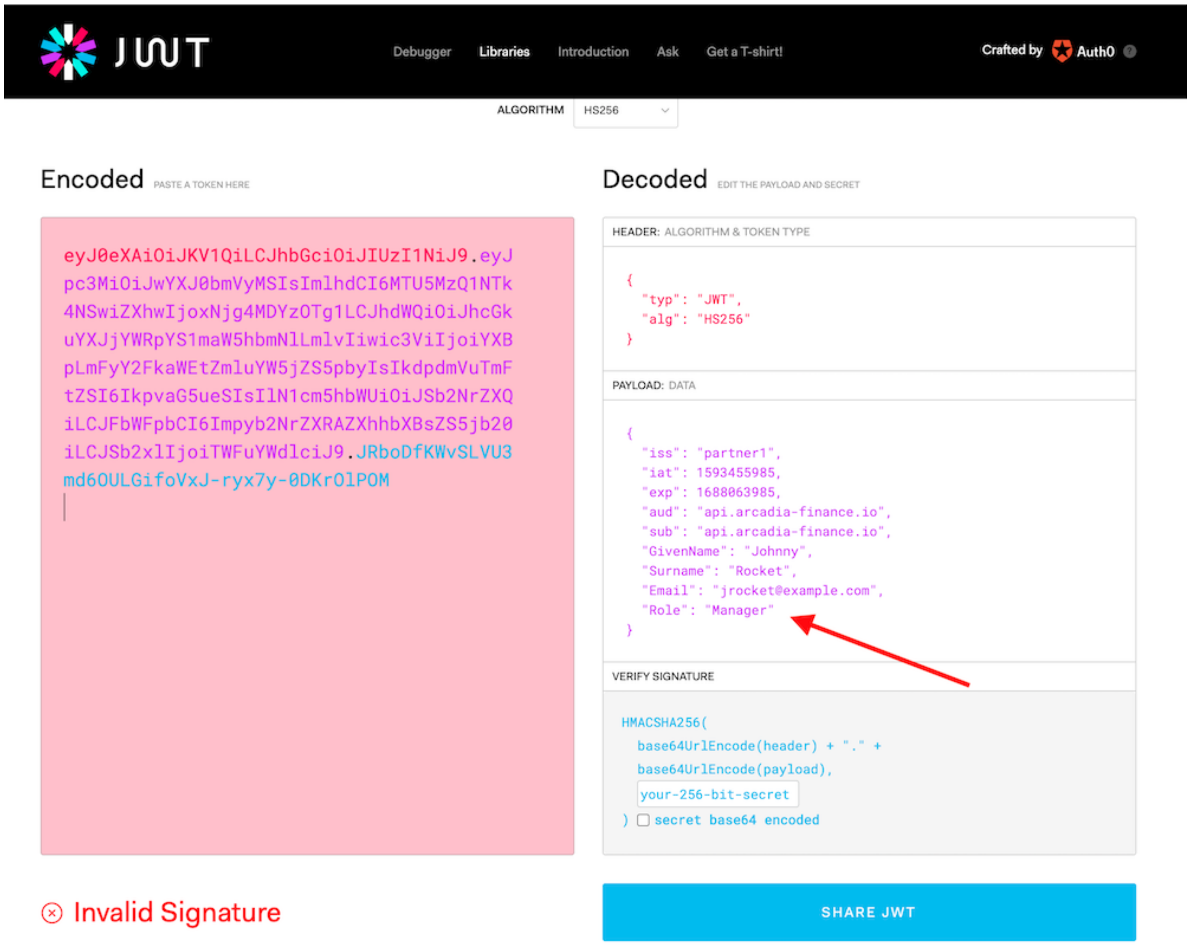
Step 1 - Understand the JWT token claims

In the previous lab, we used 2 tokens:

```
Partner 1:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJpc3MiOiJwYXJ0bmVyMSIsImlhdCI6MTU5MzQ1NTk4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGkuYXJjYWRpYS
↪JRboDfKWvSLVU3md6OULGifoVxJ-ryx7y-0DKrOlPOM
```

```
Partner 2:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJpc3MiOiJwYXJ0bmVyMiIsImlhdCI6MTU5MzQ1NTk4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGkuYXJjYWRpYS
↪aqTxd6X4z7EFijJsyiuq8mZAKMLG519Bmjz1ra24L-s
```

1. Navigate to https://jwt.io/ and paste `Partner1` JWT token into the website. And check the claim `Role`. `Partner 1` is a manager.



ALGORITHM HS256

Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ
pc3MiOiJwYXJ0bmVyMSIsImlhdCI6MTU5MzQ1NTk
4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGk
uYXJjYWRpYS1maW5hbmNlLmlvIiwic3ViIjoiYXB
pLmFyY2FkaWEtZmluYW5jZS5pbyIsIkdpdmVuTmF
tZSI6IkpvaG5ueSIsIlN1cm5hbWUiOiJSb2NrZXQ
iLCJFbWFpbCI6Impyb2NrZXRAZXhhbXBsZS5jb20
iLCJSb2xlIjoiTWFuYWdlciJ9.JRboDfKWvSLVU3
md6OULGifoVxJ-ryx7y-0DKrOlPOM
```

Invalid Signature

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

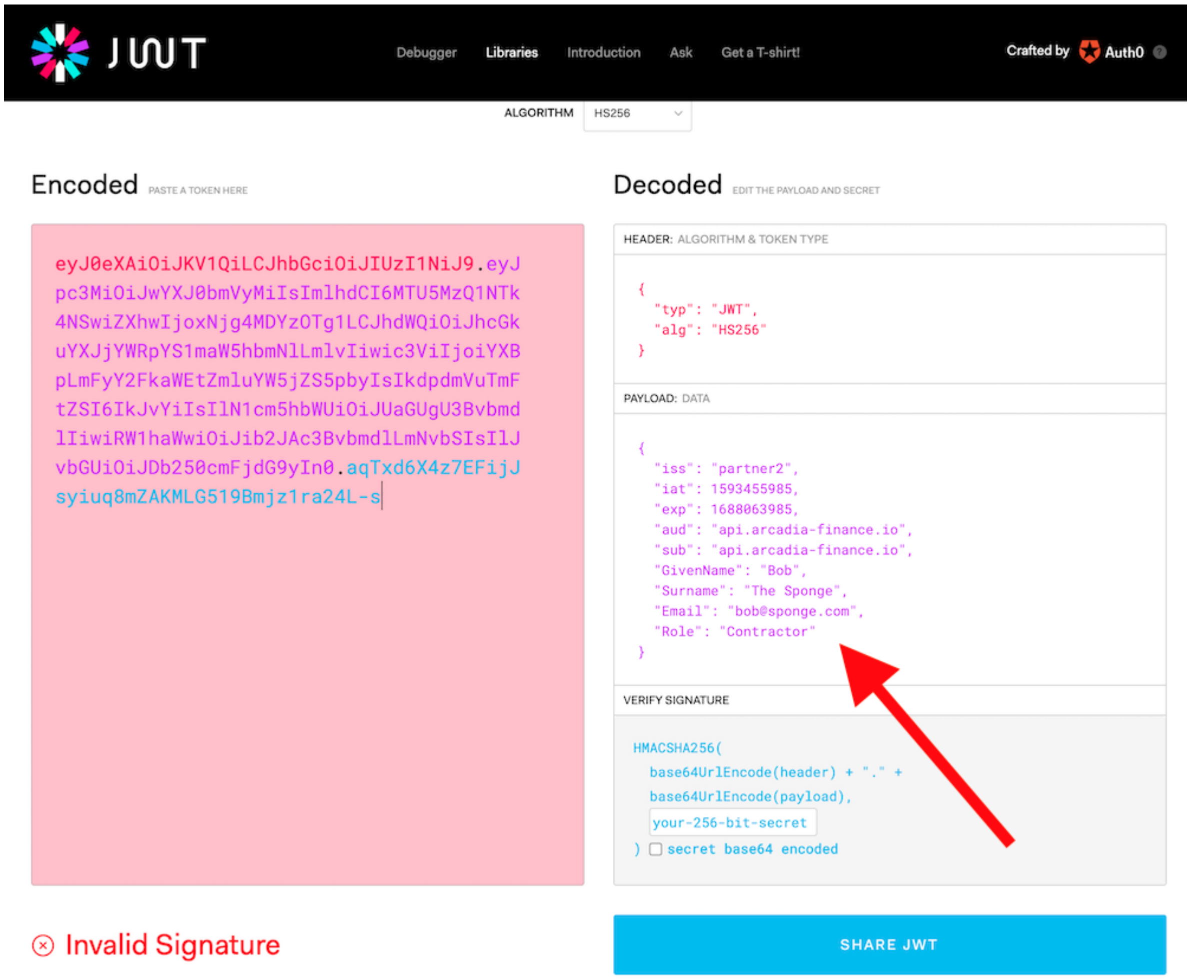
```
{
  "iss": "partner1",
  "iat": 1593455985,
  "exp": 1688063985,
  "aud": "api.arcadia-finance.io",
  "sub": "api.arcadia-finance.io",
  "GivenName": "Johnny",
  "Surname": "Rocket",
  "Email": "jrocket@example.com",
  "Role": "Manager"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

SHARE JWT

2. Do the same with Partner2 JWT token. `Partner 2` is a contractor



```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ
pc3MiOiJwYXJ0bmVyMiIsImlhdCI6MTU5MzQ1NTk
4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGk
uYXJjYWRpYS1maW5hbmNlLmlvIiwic3ViIjoiYXB
pLmFyY2FkaWEtZmluYW5jZS5pbyIsIkdpdmVuTmF
tZSI6IkJvYiIsIlN1cm5hbWUiOiJUaGUgU3Bvbmd
lIiwiRW1haWwiOiJib2JAc3BvbmdlLmNvbSIsIlJ
vbGUiOiJDb250cmFjdG9yIn0.aqTxd6X4z7EFijJ
syiuq8mZAKMLG519Bmjz1ra24L-s
```

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

```
{
  "iss": "partner2",
  "iat": 1593455985,
  "exp": 1688063985,
  "aud": "api.arcadia-finance.io",
  "sub": "api.arcadia-finance.io",
  "GivenName": "Bob",
  "Surname": "The Sponge",
  "Email": "bob@sponge.com",
  "Role": "Contractor"
}
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Invalid Signature

Step 2 - Create a new Claim in APM in order to forward this claim into the Bearer SSO

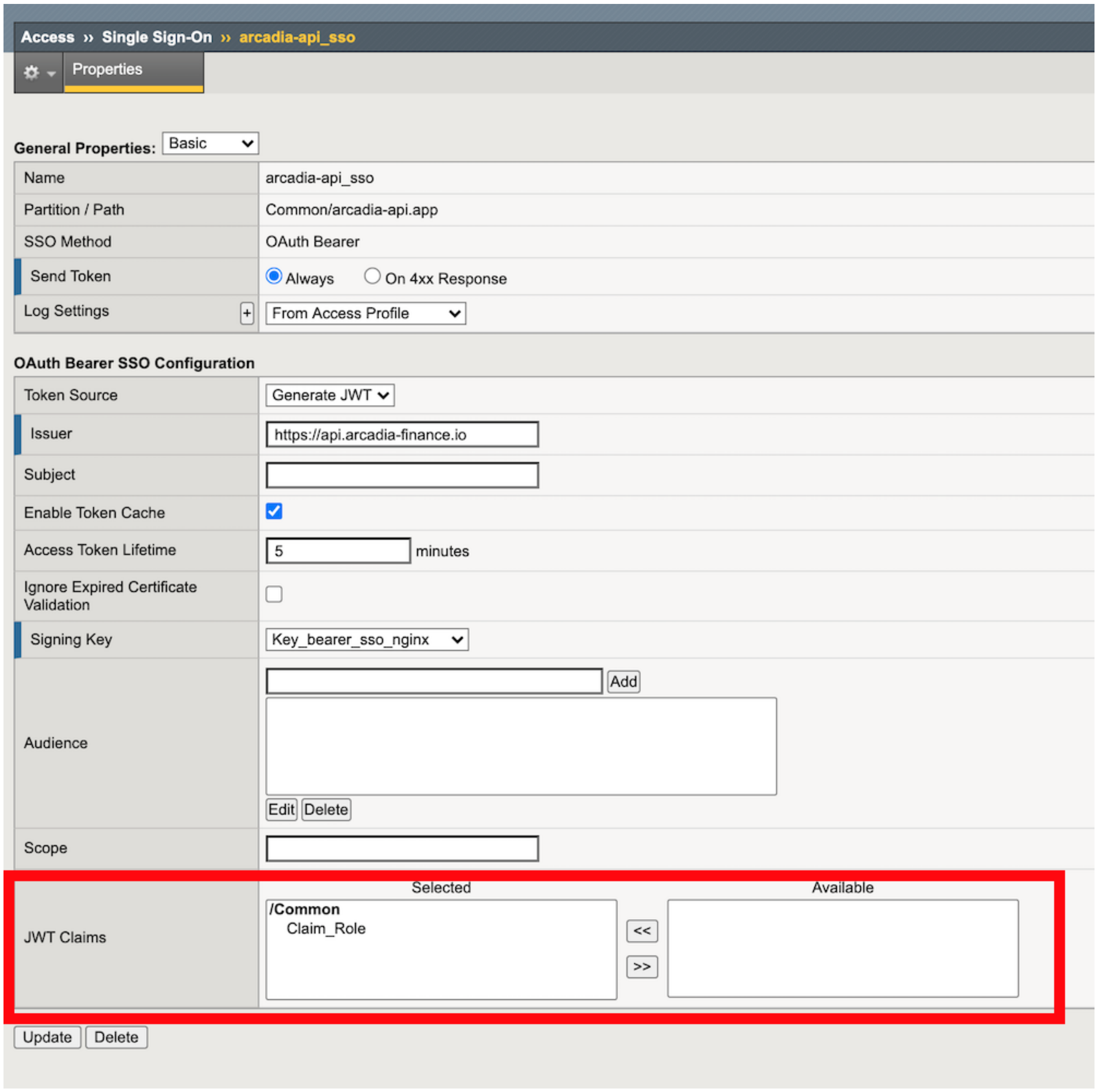
Note: The providers will inject a Claim into the JWT. This claim is `Role`. We need to re-inject this claim into the Bearer SSO token so that NGINX GW only accept requests from users belonging to Manager Role.

1. In the `BIG-IP` > `Federation` > `Oauth Authorization Server` > `Claim`
2. Click `Create`
3. Create this `Claim`
    1. Name `Claim_Role`
    2. Claim Type `String`
    3. Claim Name `Role`
    4. Claim Value `%{subsession.oauth.scope.last.jwt.Role}`
4. Click `Save`

Step 3 - Modify the Bearer SSO in order to inject this new Claim

Note: Now, it is time to tell to the Bearer SSO profile to inject this claim in the JWT SSO token

1. In `Single Sign-on` > `Oauth Bearer` > `arcadia-api-sso`
2. Add the previous created `Claim` into the `Selected` list
3. Click `Update`



General Properties: Basic

Name	arcadia-api_sso
Partition / Path	Common/arcadia-api.app
SSO Method	OAuth Bearer
Send Token	[✓ Always] On 4xx Response
Log Settings	From Access Profile

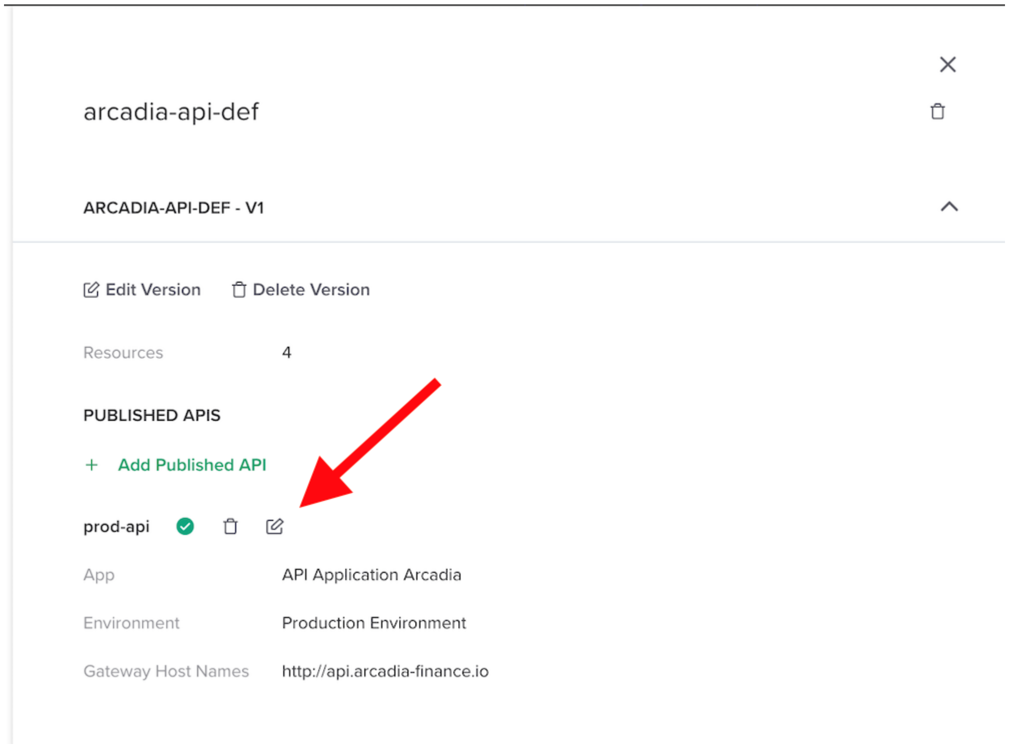
OAuth Bearer SSO Configuration

Token Source	Generate JWT
Issuer	https://api.arcadia-finance.io
Subject	
Enable Token Cache	✓
Access Token Lifetime	5 minutes
Ignore Expired Certificate Validation	☐
Signing Key	Key_bearer_sso_nginx
Audience	
Scope	
JWT Claims	Selected: /Common Claim_Role; Available:

Step 4 - Update the authorization component setting in the controller

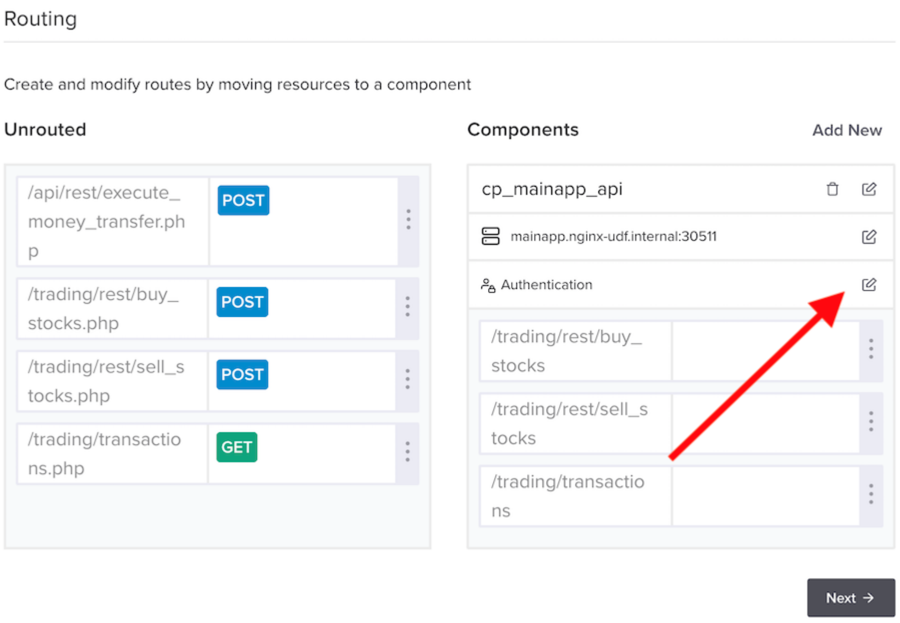
Note: Now, the BIG-IP APM is injecting the Claim in the JWT Bearer SSO token. It is time to tell to the NGINX GW, to only grant access if the `Role` contains `Manager`

1. In the controller GUI
2. In `APIs` > `arcadia-api-def`, edit the `published API` `prod-api`



Resources	4
App	API Application Arcadia
Environment	Production Environment
Gateway Host Names	http://api.arcadia-finance.io

3. Edit `Authentication`


Routing

Create and modify routes by moving resources to a component

Unrouted	
/api/rest/execute_money_transfer.php	POST
/trading/rest/buy_stocks.php	POST
/trading/rest/sell_stocks.php	POST
/trading/transactions.php	GET

Components — Add New

cp_mainapp_api

mainapp.nginx-udf.internal:30511

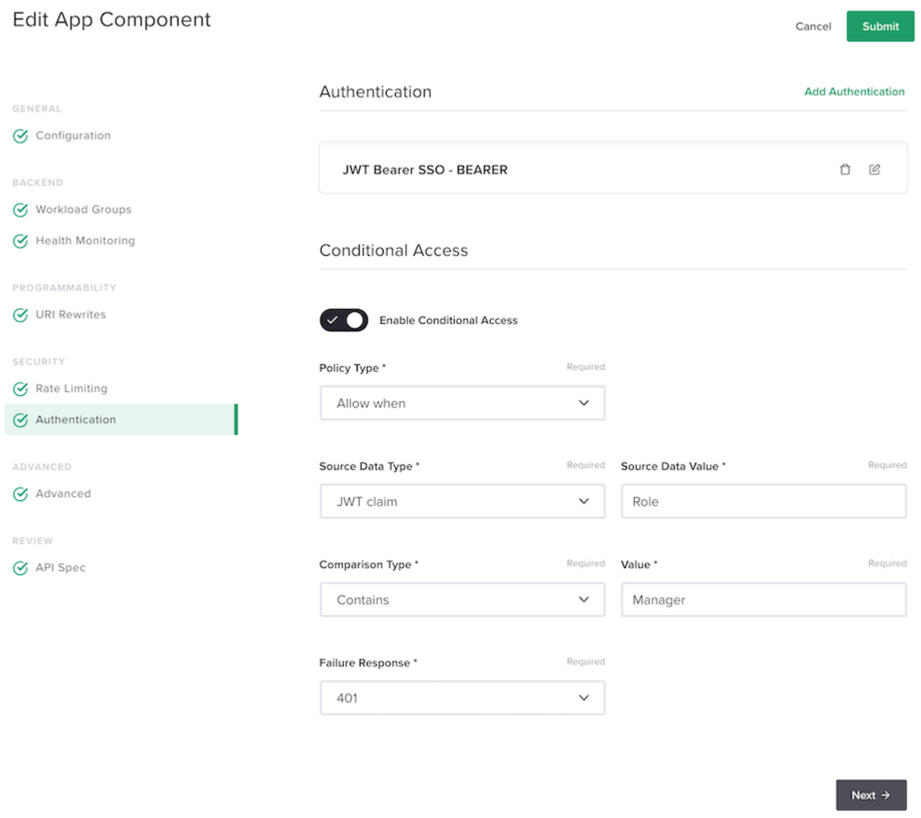
Authentication

/trading/rest/buy_stocks

/trading/rest/sell_stocks

/trading/transactions

4. Click on `Enable Conditional Access`
5. And enter the values below. This will allow access only if the `Claim` contains `Manager`. Select `401` as Failure Reposonse as `403` is not allowed as reponse type by Adv. Waf (by default)


Edit App Component

Authentication — Add Authentication

JWT Bearer SSO - BEARER

Conditional Access

Enable Conditional Access: enabled

Policy Type *	Allow when
Source Data Type *	JWT claim
Source Data Value *	Role
Comparison Type *	Contains
Value *	Manager
Failure Response *	401

6. Click `Submit` and `Submit`

Step 5 - Make a test

1. In `Postman`, send a request with `Partner1` JWT token. As a reminder, he is `Manager`. Request passes.

```
Partner 1:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJpc3MiOiJwYXJ0bmVyMSIsImlhdCI6MTU5MzQ1NTk4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGkuYXJjYWRpYS
↪JRboDfKWvSLVU3md6OULGifoVxJ-ryx7y-0DKrOlPOM
```

2. Then, send the same request with `Partner2` JWT token. As a reminder, he is `Contractor`. Request fails.

```
Partner 2:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJpc3MiOiJwYXJ0bmVyMiIsImlhdCI6MTU5MzQ1NTk4NSwiZXhwIjoxNjg4MDYzOTg1LCJhdWQiOiJhcGkuYXJjYWRpYS
↪aqTxd6X4z7EFijJsyiuq8mZAKMLG519Bmjz1ra24L-s
```

Note: As you can notice, APM is collecting the different claims and only forward the relevant claims to the internal API GW. Then, API GWs grant access based on the claim values.

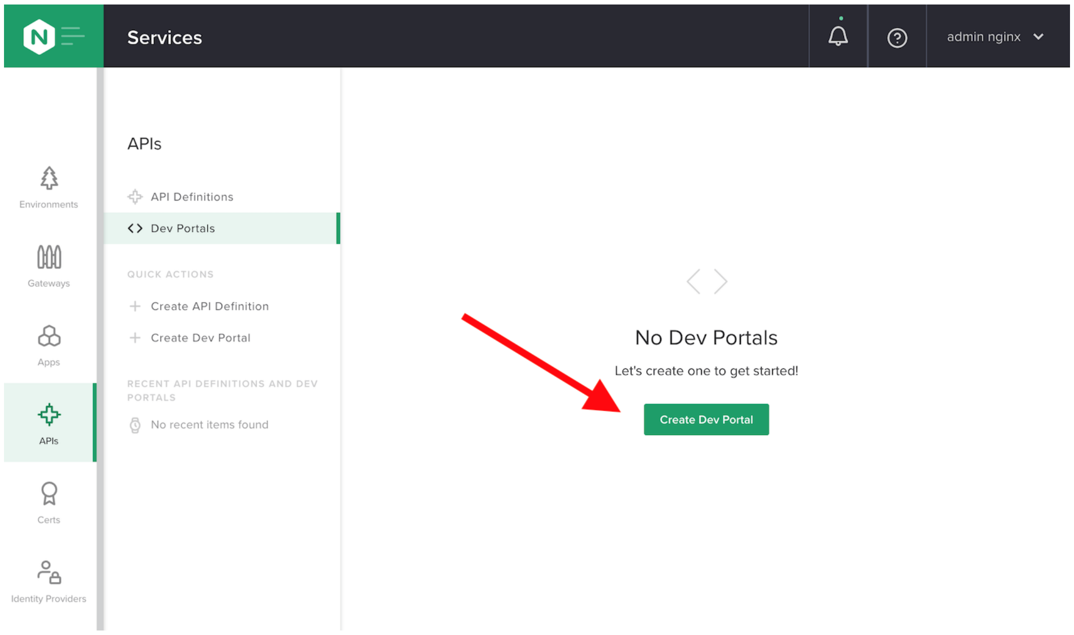
1.3.5 Module 5 - Developer Portal

Note: If you remember, we deployed 3 instances. One for the WebApp, one for the APIs and another one for the DevPortal. We will use the latest in this lab.

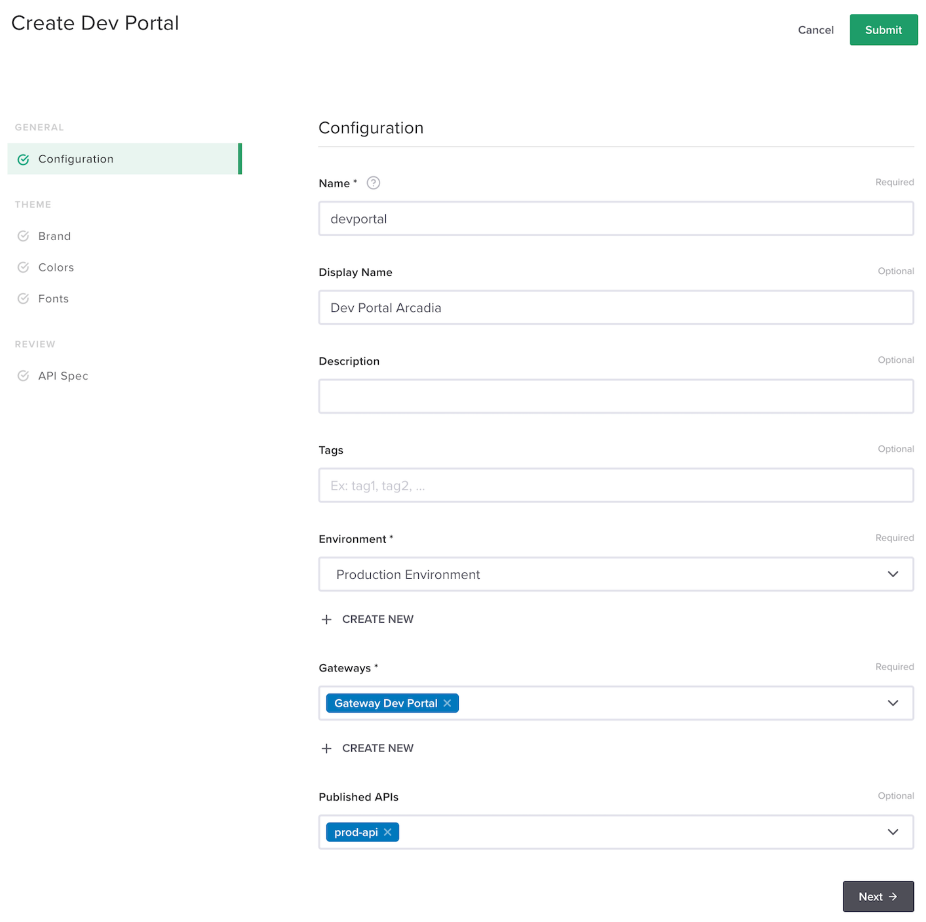
When we uploaded the OAS3 file, this file included the API documentation as well. There is only one step to publish the documentation into the DevPortal instance.

Step 1 - Create a Dev Portal object

1. In `APIs`, then `Dev Portals` create a new Dev Portal object



2. **Configure the object as below**
    1. Name: `devportal`
    2. Display Name: `Dev Portal Arcadia`
    3. Environment: `Production Environment`
    4. Gateway: `Gateway Dev Portal`
    5. Published API: `prod-api`


Create Dev Portal

Configuration

Name * (Required): devportal

Display Name (Optional): Dev Portal Arcadia

Description (Optional):

Tags (Optional): Ex: tag1, tag2, ...

Environment * (Required): Production Environment

Gateways * (Required): Gateway Dev Portal

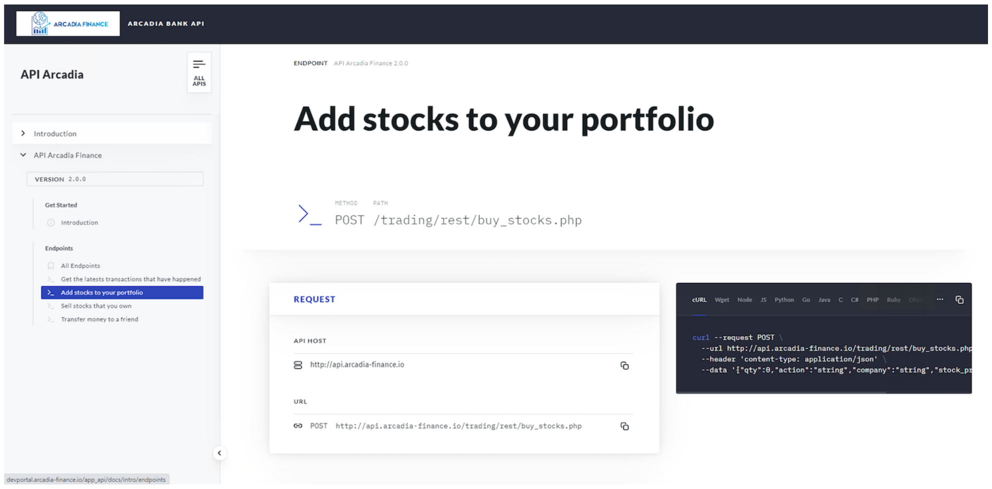
Published APIs (Optional): prod-api

6. Click `Next`

3. Give a `Brand name` like `API for Arcadia Application`, and upload any logo if you want
4. Click `Next` and `Submit`

Step 2 - Navigate to the Developer Portal

1. RDP to `Jumphost` as `user/user`
2. **Open `Chrome` and click on bookmark `Dev Portal APIm`**
    1. Click on `Explore API` and `Get Started`



Add stocks to your portfolio

```
METHOD  PATH
POST /trading/rest/buy_stocks.php
```

API HOST: http://api.arcadia-finance.io

URL: POST http://api.arcadia-finance.io/trading/rest/buy_stocks.php

```
curl --request POST \
  --url http://api.arcadia-finance.io/trading/rest/buy_stocks.php
  --header 'content-type: application/json' \
  --data '{"qty":0,"action":"string","company":"string","stock_pr
```

Note: Navigate in the Developer Portal. As you can notice, this has been populated automatically thanks to the OAS file. As a reminder, the OAS file looks like that (this is an extract for the `buy stock` API).

```
/trading/rest/buy_stocks.php:
  post:
    summary: Add stocks to your portfolio
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/buy'
          example:
            trans_value: '312'
            qty: '16'
            company: MSFT
            action: buy
            stock_price: '198'
    responses:
      '200':
        description: 200 response
        content:
          application/json:
            example:
              status: success
              name: Microsoft
              qty: '16'
              amount: '312'
              transid: '855415223'
```