

App Delivery Manager

Lab Guide

Version 1.6

7/17/2023 12:24 A.cM

UDF Lab

Deploy the following UDF Blueprint in the region closest to you:

1. Click on Components
2. Scroll to Ubuntu Jumpbox, click Access, and click XRDP
3. username is: ubuntu password is: UBUNTU123!@#

NMS ADM Lab TechXchange 2023



Maintained by Brett Wolmarans

F5, Sr. Product Management Engineer – NGINX Management Suite Modules Ecosystem



Solution



UDF

\$0.53

per hour

11

Deploys

 **DEPLOY**

 **COLLABORATORS (2)**

 **UNPUBLISH**

 Documentation

 Internal

 Components

 Revisions

 C

Summary

Welcome to the ADM Lab!

You are going to do all your work from the RDP Jumpbox.
Please have RDP client ready to go on your Windows or Mac.

1. Click on Components
2. Scroll to Ubuntu Jumpbox, click Access, and click XRDP
3. username is: ubuntu password is: UBUNTU123!@#

Please ctrl-Click here for the Lab Guide: [ADM_UDF_Lab.pdf](#)

Please ctrl-Click here for the ADM Lab Presentation slides: [ADM_UDF_Lab_Presentation.pdf](#)

Presentation Slides

The Lab Presentation Slides link can be found in the UDF deployment documentation.

Module 1 – Simple HTTP Deployment

App Delivery Manager

Learn how to create a simple HTTP deployment that securely load balances between several apps including microservices apps

High Level Business Objective

- You work in the information technology department for a global drinks company, and you have been tasked with delivering the following critical business applications reliably, securely, and with high performance:
 - **Brewz**
 - **Juiceshop**

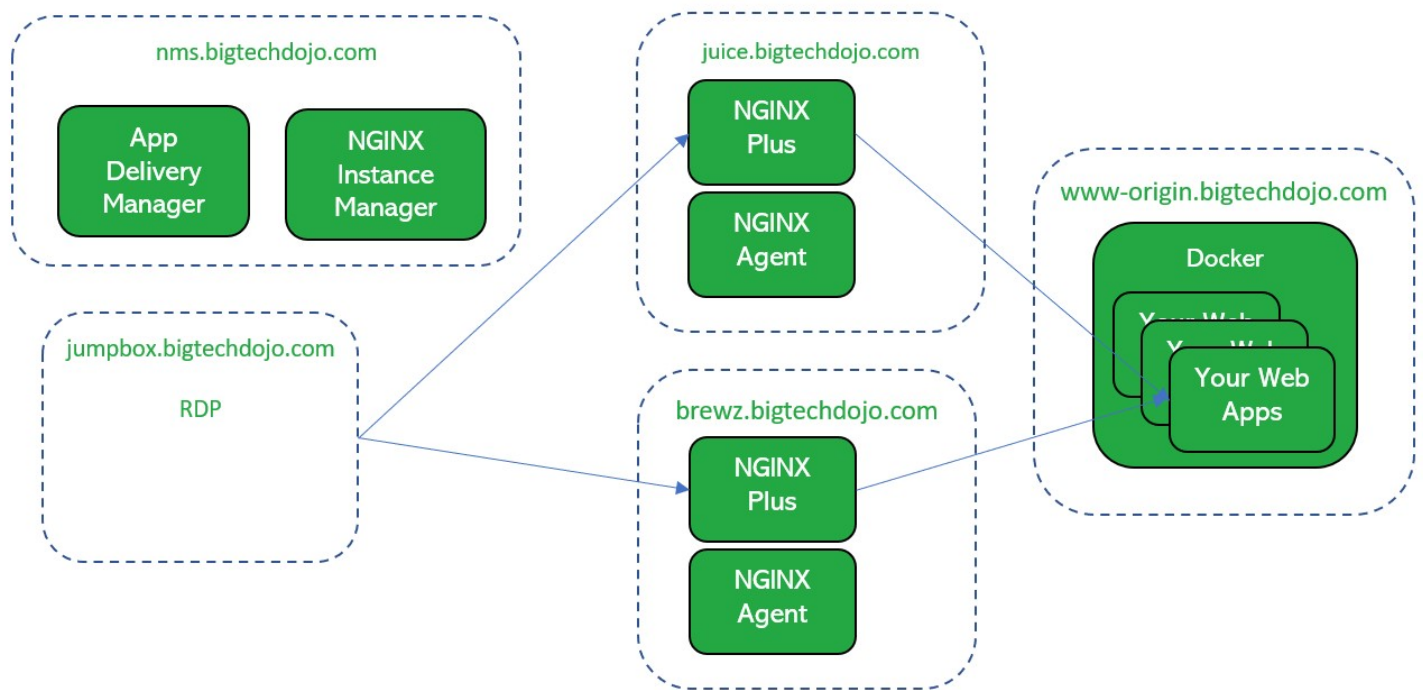
Brewz lives on **www.bigtechdojo.com** server, and consists of multiple microservices as shown here:

IMAGE	PORTS
spa-demo-app_recommendations	0.0.0.0:8001->8001/tcp
spa-demo-app_spa	0.0.0.0:8081->80/tcp,
spa-demo-app_inventory	0.0.0.0:8002->8002/tcp
spa-demo-app_api	0.0.0.0:8000->8000/tcp
spa-demo-app_checkout	0.0.0.0:8003->8003/tcp

Juiceshop also lives on **www.bigtechdojo.com**, is containerized, but is monolithic, deployed on a single port, but there are two instances of Juiceshop on ports **3000** and **3001** as shown here:

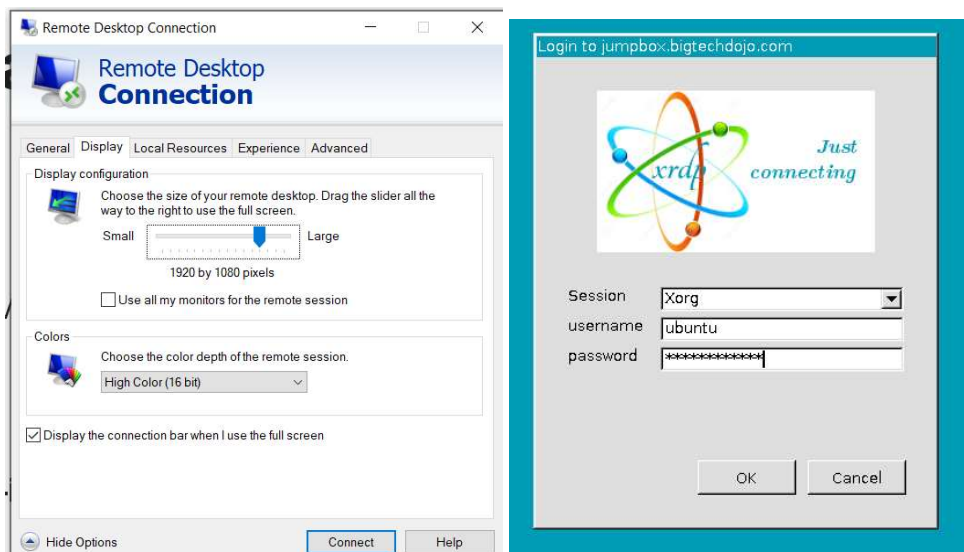
bkimminich/juice-shop	0.0.0.0:3001->3000/tcp
bkimminich/juice-shop	0.0.0.0:3000->3000/tcp

Diagram



List of Components

1. NMS - this is your NMS and ADM management platform. Creds are shown in the UDF deployment->NMS component->details
2. JUICE - this is one of your NGINX+ and NGINX-AGENT instances.
3. BREWZ - this is another
4. WWW - this is a box of web servers, with one listening on port 80
5. UBUNTU - this is simply a jumpbox. creds are shown in the UDF deployment->Ubuntu component->details.
 1. Access via RDP, and then do all your work from the jumpbox.
 2. This box is faster, smoother, more fun and easier than a Windows Jumpbox. Don't worry it has Firefox on it.
 3. Pro Tip: When you download the RDP file, right-click and edit the file and set color depth to 16-bit, set the screen resolution to slightly less than your monitor, set the username to ubuntu, and enable saving username. Use Ctrl-Shift-V to paste.



Detailed Requirements & Information

- The platform infrastructure team has already configured the software and hardware for you, including NGINX Management Suite, App Delivery Manager, the NGINX instances, and the application servers.

You are the application delivery team, and in that role, you will consume the platform to deliver the application. The business wants the Juice application deployed in the West, and the Brewz application deployed in the East.

Apps need to be accessed securely over the Internet at the domains as shown in the diagram.

You will know you have succeeded when you can browse securely to the Juice Gateway and use the Brewz app, and you can browse securely to the Brewz Gateway and use the Juice app.

You will do everything from the Ubuntu RDP jumpbox



Spoiler Alert – Do not proceed unless you want to see the step-by-step solution

Solution

In the NGINX Management Suite web interface, you access the App Delivery Manager (ADM) features by performing the following operations. **You will do everything from the Ubuntu RDP jumpbox.**

1. Log into the Ubuntu jumpbox via RDP: credentials can be found in UDF under “Ubuntu JumpBox RDP”, “Details” and then “Documentation” tab
2. Start Firefox on the Ubuntu jumpbox
3. Click on the “Juice Gateway” book mark link in Firefox, and notice nothing is there. This is because Juice Gateway is a totally unconfigured nginx instance.
4. Click on the NMS bookmark in the bookmark bar and login. Credentials can be found in UDF under “NMS-ADM”, “Details” and then “Documentation” tab
5. From the Launchpad, select the **NGINX Instance Manager** card to see your instances that have already been deployed for you.
 1. These have been put into two Instance Groups, region-1 and region-2.
 2. Click on one of the instance groups and click “edit config” to see the “blank” configuration.
6. Now, go back to the Launchpad and select the **App Delivery Manager** card

Create an Environment

The first resource you need to create, if one doesn’t already exist, is an Environment resource. This can be accomplished by taking the following steps:

1. Select **Environments** from the **App Delivery Manager** list in the left-hand sidebar. The list of existing environments will then display.
2. Select **Create Environment** on the right-hand side of the list. A panel will appear that allows you to configure the environment.
3. Enter the value **Production** for the **Name** field. This logical environment is not just for one app, but for all your internet-facing websites. You can take the defaults for all the other fields (this exercise does not require customized templates).
4. Select **Submit** to finish creating the environment.

Create a Gateway for Juice Shop

The gateway controls how traffic will route through an NGINX instance to get to the app workloads.

1. Select **Gateways** from the **App Delivery Manager** list in the left-hand sidebar. The list of existing gateways will then display.
2. Select **Create Gateway** on the right-hand side of the list. A panel will appear that allows you to configure the gateway.
3. From the Configuration page of Create Gateway, enter the gateway name as **Juice Gateway**. You can accept defaults for the next two fields.
4. For the environment field, select the environment **Production** that you previously created.
5. Select **Next** to get to the **Placements** page.
6. The platform team should have created an instance group **region-1**. Select **Add Placement** and from the Instance Group **Refs** dropdown, select **region-1**. Then click **Done**.
7. Select **Next** to get to the **Hostnames** page.
8. Select **Add Hostname** then enter **https://juice.bigtechdojo.com** for the Hostname.

9. Your platform team working with your security team, added Certificates already.

10. Switch to NIM and view the certificates:

Certificates and Keys

Overview

2 2 0 0 0 2

Filter Refresh Export Add

Name	Domain	Management	Status	Actions
juice.bigtechdojo.com	juice.bigtechdojo.com	Managed	Healthy Until 3 Oct 2023	...
brewz.bigtechdojo.com	brewz.bigtechdojo.com	Managed	Healthy Until 3 Oct 2023	...

11. Click on the [juice.bigtechdojo.com](#) and you will see details of the Cert and Key, and you will see it is deployed to the **region-1** instance group.

Certificates and Keys > Certificates Detail

[juice.bigtechdojo.com](#) | [juice.bigtechdojo.com](#)

Update Cert

Certificate

Display Name	juice.bigtechdojo.com	Status	Healthy
Domain	juice.bigtechdojo.com	Expiry Date	3 Oct 2023
Management	Managed	Serial Number	410876544244367085140332931434693804432158
NMS UUID	a4acd410-ead8-4438-b5d8-d0211d66c27f		

Key

Encryption Type	SHA256-RSA
Last Updated	5 Jul 2023

Filter

Name	Association T...	Certificate Paths	Key Paths
juice.bigtechdojo.com	Instance	/etc/nginx/aux/juice.bigtechdojo.com.crt	/etc/nginx/aux/juice.bigtechdojo.com.key

Instance Type	NGINX Plus - 1.23.4 (nginx-plus-r29)	Instance Status	Online	Instance Group	region-1
Config Path	/etc/nginx/nginx.conf	Process Path	/usr/sbin/nginx	Registration Time	6/28/2023, 4:09:22 PM
Start Time	7/5/2023, 5:45:06 PM	Last Seen	half a minute ago	Process ID	722

12. Switch back to ADM.

13. In the Shared TLS Settings section, select the **juice.bigtechdojo.com** certificate.

14. Click Submit

Create your Application for Juice Shop

Follow these steps to create the applications:

1. Select **Apps** from the **App Delivery Manager** list in the left-hand sidebar. The list of existing apps will then display.
2. Select **Create App** on the right-hand side of the list. A panel will appear that allows you to configure the app.
3. Enter the value **Juice** for the Name field. Select **Production** for the **Environment** field. You can take the defaults for all the other fields.
4. Select **Submit** to finish creating the app.
5. Status will be displayed as **Configured**

Create the Juice Shop Production Web Component

The app we just created is a wrapper that can be composed of multiple components, each potentially referencing a unique service or microservice. To create the production component, perform these steps:

1. You should be on the **Apps Overview** page at this point. Select the app that was just created in the list by clicking the app name.
2. The main display will now show basic metrics for the app. We are not, at this point, interested in the metrics, but from this page we can create a component. At the top of the page, select Web Components.
3. The list of web components will appear, but should be empty.
4. Select **Create Web Component** on the top right-hand side of the display. A panel will appear that allows you to configure the component. There will be several pages of configuration that will need to be performed.
5. On the first page (Configuration), enter the value **Juice** for the Name field.
6. The only other field that needs to be set on this page is the **Gateway Refs** field. Under this field, select **Juice Gateway**.
7. Click **Next** to advance to the URIs page.
8. Enter **/** for the URI (if you are not able to enter a value, click the pencil icon to edit the URI).

9. Click **Next** to proceed to the Workload Groups page and select **Add Workload Group**
 - a. In the **Workload Group Name** field, enter **Juice Servers**
 - b. In the Backend Workload URIs section, enter for the URI field and click Done:
http://www.bigtechdojo.com:3000
 - c. Select Add Backend Workload URI to add another workload, and enter:
http://www.bigtechdojo.com:3001
 - d. Click **Done**, then Click **Done** for the overall Workload Groups page.

Create Web Component

BASIC

- Configuration
- URIs
- Workload Groups**

Workload Groups

Workload Group Name *

juice-origin

Backend Workload URIs

http://www-origin.bigtechdojo.com:3001

http://www-origin.bigtechdojo.com:3000

Additional Settings

- > Load Balancing Method
- > Site References

10. Select the **Submit** button to complete the component configuration.
11. When the configuration is applied **Status** will be shown as **Configured**

Testing

1. Select the **Juice** bookmark, or refresh the tab if you had one open, and you will see the Juiceshop application loading through your NGINX+ instance.
2. In NMS, navigate to Instance Manager, go to Instance group **region-1**, click on “Edit Config” and you will see the configuration that ADM has created.

```
# Created by Gateway: Juice(52d03640-ee57-4711-a8c3-3dae9be3e03e)
server {
    server_name juice.bigtechdojo.com;
    listen 443 ssl reuseport;
    ssl_certificate /etc/nginx/aux/juice.bigtechdojo.com.crt;
    ssl_certificate_key /etc/nginx/aux/juice.bigtechdojo.com.key;
    status_zone 52d03640-ee57-4711-a8c3-3dae9be3e03e;
    f5_metrics_marker environment b94840ed-bb96-43ab-b8a9-385e274e316e;
    f5_metrics_marker gateway 52d03640-ee57-4711-a8c3-3dae9be3e03e;
    # Generated by web component juice(92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7)
    location / {
        status_zone 92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_set_header Connection "";
        proxy_http_version 1.1;
        proxy_pass http://92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7;
        # metrics
        f5_metrics_marker app d7a55a20-f0c5-41b8-b583-453ad04184b6;
        f5_metrics_marker component 92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7;
    }
}
upstream 92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7 {
    zone 92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7 1280K;
    server www-origin.bigtechdojo.com:3000 max_fails=1 weight=1;
    server www-origin.bigtechdojo.com:3001 max_fails=1 weight=1;
}
# end publication:region-1, generated for ADM-3e94528f-6f9f-44de-8cdf-fd73fe5fdccb
```

Create a Gateway for Brewz

Follow the steps as shown above in the “Create a gateway for Juice Shop” section, with the following changes:

1. Configuration: name the gateway **Brewz Gateway**
2. Placements: Place this gateway in **region-2** of the country, for low latency.
3. Hostnames: Enter <https://brewz.bigtechdojo.com> for the **Hostname**
4. Select the **brewz.bigtechdojo.com** certificate.
5. Click **Submit**

Create your Application for Brewz

Now that you have the hang of a simple monolithic application, you are going to deploy the microservices application Brewz. Brewz lives on the following ports:

IMAGE	PORTS
spa-demo-app_recommendations	0.0.0.0:8001->8001/tcp
spa-demo-app_spa	0.0.0.0:8081->80/tcp,
spa-demo-app_inventory	0.0.0.0:8002->8002/tcp
spa-demo-app_api	0.0.0.0:8000->8000/tcp
spa-demo-app_checkout	0.0.0.0:8003->8003/tcp

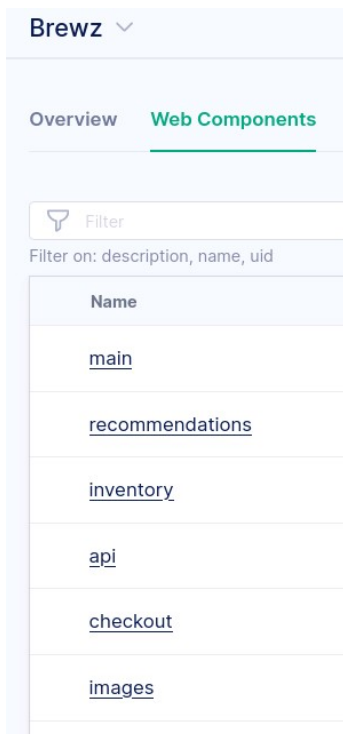
1. Create a new application
2. Enter **Brewz** for the **Name** field and select **Production** in the **Environment** field

The front-end of the Single-Page Application directs traffic to each of these containers based on the following routes:

URI	Port	Web Component	Workload URI
/checkout	8003	checkout	http://www.bigtechdojo.com:8003
/	8081	main	http://www.bigtechdojo.com:8081
/recommendations	8001	recommendations	http://www.bigtechdojo.com:8001
/inventory	8002	inventory	http://www.bigtechdojo.com:8002
/api	8000	api	http://www.bigtechdojo.com:8000
/images	8000	Images	http://www.bigtechdojo.com:8000

1. Create each of the 6 Brewz Web Components

You are going to create a web component for each Brewz microservice so that you end up with the following 6 web components:



*Note: You could also create a single Web Component, and add multiple URI's to that single Web Component. The advantage of adding each URI as a separate Web Component is that if needed, each URI path could be mapped to a different NGINX instance, if needed.

Detailed steps are provided here for the **/checkout** route, you are going to learn from these steps and create the other routes.

Follow the steps you already completed above for the Juice app, with the following changes:

- On the first page (Configuration), enter the value **checkout** for the **Name** field.
- The only other field that needs to be set on this page is the **Gateway Refs** field. Under this field, select **Brewz Gateway**. Then click **Next** to advance to the URIs page.

14. Enter **/checkout** for the URI (click the pencil icon to edit the URI).
15. Click **Done** to save the URI
16. Click **Next** to proceed to the Workload Groups page.
 - a. In the **Workload Group Name** field, enter **Brewz Checkout**
 - b. In the Backend Workload URIs section, **copy from the table above** and enter for the URI field:

<http://www.bigtechdojo.com:8003>

- c. Click **Done**, then click **Done** for the overall Workload Groups page.
17. Select the **Submit** button to complete the component configuration.
18. Create the **remaining five routes to Brewz microservices as per the table above.**
Make sure the Backend Workload URI port is configured based on the table.

Testing

1. Select the **Juice** bookmark, or refresh the tab if you had one open, and you will see the application.
2. In NMS, navigate to Instance Manager, go to Instance group **region-1**, click on “Edit Config” and you will see the configuration that ADM has created.

View Paths in Network Inspector

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application		
Filter URLs		
Status	Method	URL
200	GET	https://brewz.bigtechdojo.com/products
200	GET	https://brewz.bigtechdojo.com/assets/index.6a1c8b32.js
200	GET	https://brewz.bigtechdojo.com/assets/index.bf312d07.css
200	GET	https://brewz.bigtechdojo.com/api/products
200	GET	https://brewz.bigtechdojo.com/favicon.ico
200	GET	https://brewz.bigtechdojo.com/images/tonys-coffee-upland-blend-medium-roast.png
200	GET	https://brewz.bigtechdojo.com/images/puroast-bourbon-pecan-torte.png
200	GET	https://brewz.bigtechdojo.com/images/peace-coffee-morning-glory.png
200	GET	https://brewz.bigtechdojo.com/images/tim-hortons-original-blend-medium-roast.png
200	GET	https://brewz.bigtechdojo.com/images/groundwork-organic-ethiopia-light-roast.png
200	GET	https://brewz.bigtechdojo.com/images/first-colony-columbian-santa-marta.png
200	GET	https://brewz.bigtechdojo.com/images/cafe-altura-organic-italian-style-dark-roast.png
200	GET	https://brewz.bigtechdojo.com/images/equal-exchange-organic-breakfast-blend.png
200	GET	https://brewz.bigtechdojo.com/images/illy-caffe-medium-roast.png
200	GET	https://brewz.bigtechdojo.com/images/seattles-best-dark-intense.png
200	GET	https://brewz.bigtechdojo.com/images/stumptown-coffee-organic-holler-mountain.png
200	GET	https://brewz.bigtechdojo.com/images/community-coffee-breakfast-blend-medium-roast.png

Critical Analytics

In the ADM module select **Apps** then **Juice** and open the **Critical Analytics** tab. Select Breakout By: **HTTP URI** and select **Last 30 minutes**



Module 2 – NAP WAF

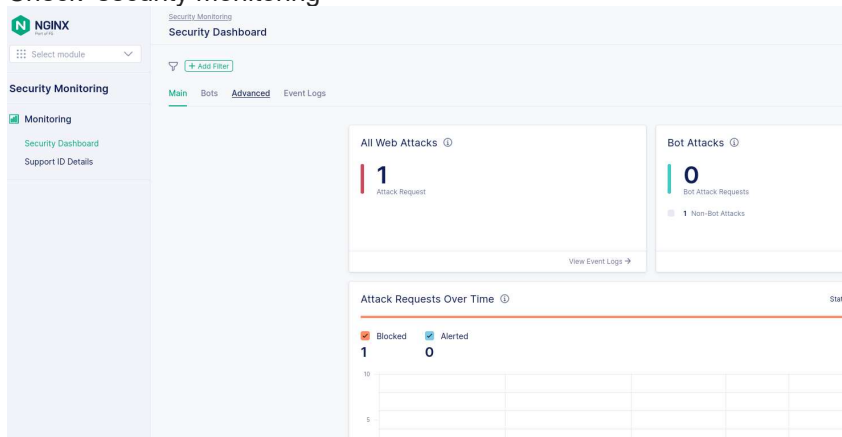


By selecting the NAP Policy in the Template, the pre-compiled policy TGZ is automatically pushed out from the ADM host to the NGINX Dataplane host. Furthermore, if the app developer clears any policy from the WAF Template, the agent removes the TGZ from the dataplane host.

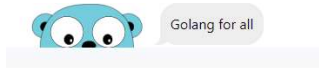
1. Edit your **Production** environment (click on the three dots) and select **Use Cases** templates and select the following templates: **builtin-waf-v2**
2. Click on your Juice App (underlined)
3. Edit your Juice Web Component
4. Custom extensions, select **NginxDefaultPolicy**, set logging to **secops_dashboard**, and logging location to **syslog:server=127.0.0.1:514**

A screenshot of the 'App Protect' configuration page. The page has a light blue header with the title 'App Protect' and a subtitle 'App Protect policies to be used by components. Allows setting secu'. Below the header, there are three sections: 'Security Policy *' with a dropdown menu showing 'NginxDefaultPolicy', 'Logging Policy' with a dropdown menu showing 'secops_dashboard', and 'Logging Location' with a text input field containing 'syslog:server=127.0.0.1:514'. Each section has a small description below it.

5. Submit
6. Review the resulting config in Instance Manager on the region-1 instance group.
7. Go to your juice gateway, click login, and enter **or 1=1;-** for the login name and **ddd** for the password
8. Check security monitoring



Module 3 – More Templates



GO templates

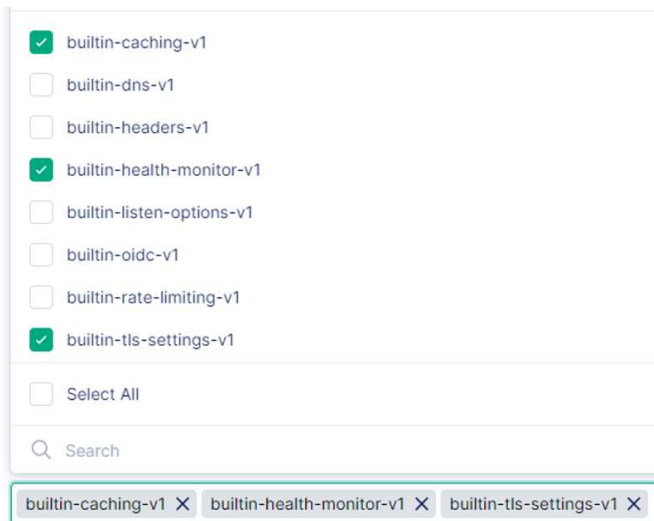
Templates allow a customer to extend ADM.

In this module, you are going to enable existing templates. After that, you will install new templates into ADM that professional services has created to enable remote syslog logging to a rsyslog server.

Enabling Templates

You are going to enable the Caching, Health Monitoring, and TLS templates.

1. Edit your **Production** environment (click on the three dots) and select **Use Cases** templates and select the following templates:
 - builtin-caching-v1
 - builtin-health-monitor-v1
 - builtin-tls-settings-v1



2. From the **Gateways** menu **Edit** your **Juice Gateway** and select **Custom Extensions** and enable the following Custom settings:
 1. TLS Options
 1. Proxy SSL Protocols: **TLSv1.1 TLSv1.2**
 2. Proxy SSL Ciphers : **HIGH:!aNULL:!MD5**
3. Click **Submit**

4. From the **Apps** menu select **Juice** and the **Web components** tab. Edit your **Juice Shop** web component by clicking on the three dots and select **Custom Extension**
5. Find the **Health check configuration** and configure the Health Check **Interval** to 10 **Jitter** to 1

Workload Groups

Custom Extensions

Health check configuration

Health check settings for components applicable to web client requests. Adds health_check directive - https://nginx.org/en/docs/http/nginx_http_upstream_hc_module.html#health_check ⊗ Clear section ^

Interval Optional

10 ⊗

Sets the interval between two consecutive health checks, by default, 5 seconds.

Jitter Optional

1 ⊗

6. Scroll down until you find **Caching Time** and click **Add Item**

Caching time

Defines the caching time for different response codes ^

+ Add item

7. Configure 5s of caching time for 200 response codes:

Caching time

Defines the caching time for different response codes ⊗ Clear section ^

ITEM 1 Remove item

Response codes Optional

200 ⊗

Defines the response codes whose data needs to be cached. NGINX defaults this to 200, 301, 302 response codes if not set.

Time Optional

5s ⊗

Defines the caching time for the above response codes.

8. Scroll down until you find **Cache Path** and click **Add Item**:

Cache path configurations*

Sets the cache paths and configuration parameters. ^

+ Add item

9. Set the storage path to **/tmp** and the Memory Zone size to 10m:

Cache path configurations*

Sets the cache paths and configuration parameters. ^

ITEM 1 Remove item

Storage path* Required

/tmp ⊗

Defines what the path is for this storage.

Memory zone size* Required

10m ⊗

Defines the size of the memory zone where information about the data is stored.

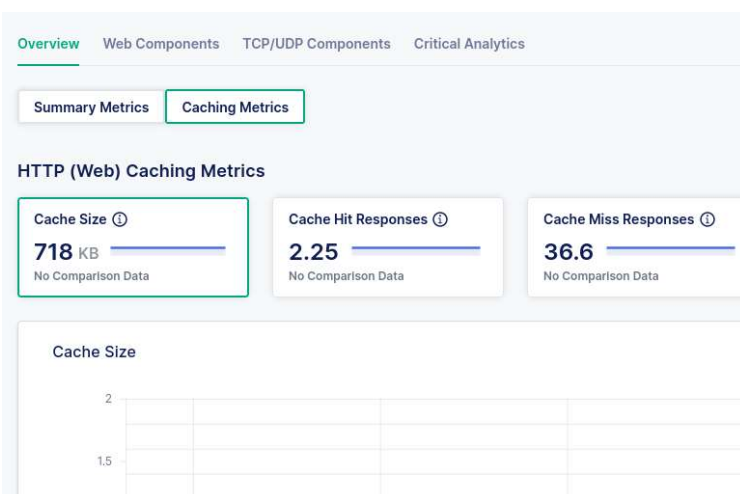
10. View the resulting configuration in Instance Manager by selecting the **Instance Groups** section and opening **region-1**
11. Look for **location = /_health_check** to see configuration lines added by the template.

```
location = /_health_check_92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7 {  
    internal;  
    health_check jitter=1 interval=10 fails=1 passes=1 uri=/  
    proxy_pass http://92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7;  
}  
# Generated by web component juice(92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7)
```

Look at the Caching configuration for the / Web Component:

```
location / {  
    # -- caching shared memory zone --  
    proxy_cache 92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7|d42b9c57d24cf5db3bd8d332dc35437f;  
    proxy_cache_valid 200 5s;  
  
    # -- cache paths for web-component 92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7 --  
    proxy_cache_path /tmp/92548ecf-ee8c-4ff4-9831-b19b5fdb9cd7 keys_zone=92548ecf-ee8c-4ff4-9831
```

View Caching Stats for the Juice App



Module 4 – OIDC

NGINX Dataplane Setup

NOTE: The following steps **have already been done for you** by the Platform team (aka the Server team). **You do not have to do these steps**, but these are provided for reference/learning:

--- snip ---

- SSH to Juice NGINX Host, which is an Ubuntu OS Host
- Install the njs module where the nginx instances are by running: For Debian/Ubuntu

```
$ sudo systemctl stop nginx
$ sudo systemctl stop nginx-agent
$ sudo apt-get update
$ sudo install nginx-plus-module-njs
$ sudo systemctl restart nginx
$ sudo systemctl restart nginx-agent
```

- SSH to the NMS System

```
$ cd /etc/nms/modules/adm/templates/usecases/builtin-oidc-v1/files
```

- Copy the `openid_connect.js` file to `/etc/nginx/conf.d` on your Juiceshop NGINX Host

```
ubuntu@nms:/etc/nms/modules/adm/templates/usecases/builtin-oidc-v1$ ls -altr
total 52
-rw-r----- 1 nms nms 1412 Jul 10 03:28 web-component.json
-rw-r----- 1 nms nms 5304 Jul 10 03:28 server-gateway.tmpl
-rw-r----- 1 nms nms  635 Jul 10 03:28 main-instance-group.tmpl
-rw-r----- 1 nms nms 1129 Jul 10 03:28 location-web-component-locations.tmpl
-rw-r----- 1 nms nms 3243 Jul 10 03:28 http-instance-group.tmpl
-rw-r----- 1 nms nms 4351 Jul 10 03:28 gateway.json
-rw-r----- 1 nms nms 4745 Jul 10 03:28 README.md
drwxr-xr-x 16 nms nms 4096 Jul 10 22:30 ..
drwxr-xr-x  2 nms nms 4096 Jul 10 22:30 files
drwxr-xr-x  3 nms nms 4096 Jul 10 22:30 .
ubuntu@nms:/etc/nms/modules/adm/templates/usecases/builtin-oidc-v1$ cd files
ubuntu@nms:/etc/nms/modules/adm/templates/usecases/builtin-oidc-v1/files$ ls
openid_connect.js
```

--- snip ---

KeyCloak Setup

NOTE: The following steps **have already been done for you** by the Platform team (aka the Server team). **You do not have to do these steps**, but these are provided for reference/learning:

--- snip ---

KeyCloak is configured at <http://10.1.1.9:8080> and admin/admin. This is run easily as a docker container:

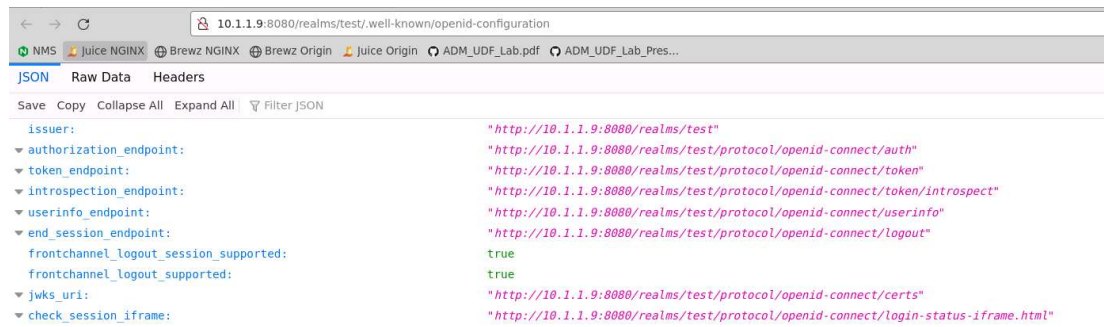
```
docker run -d --restart unless-stopped -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:22.0.0 start-dev
```

Create a new realm named **Test**. Create a new client and configure the Client ID to be: **juice**, set Client authentication to **On**, and set the valid redirect URL to **https://juice.bigtechdojo.com:443/_codexch**

The screenshot shows the Keycloak 'Client details' page for a client named 'juice'. The 'Credentials' tab is selected, showing the 'Client Id and Secret' section. The 'Client Id' is 'juice' and the 'Client Secret' is 'ZSbAMvg9TnFgrET3U4wD014WChILbWtT'. A 'Save' button is visible. Below this, the 'General Settings' section is visible, showing the 'Client ID' as 'juice', 'Name' as an empty field, 'Description' as an empty field, and 'Always display in UI' as 'Off'. The 'Access settings' section shows 'Root URL' as an empty field, 'Home URL' as an empty field, and 'Valid redirect URIs' as 'https://juice.bigtechdojo.com:443/_codexch'.

Go to users, add user **juice-developer**, click Credentials, set password to **juice-developer** and turn off Temporary or else the user will have to change it on first login.

Browse to <http://10.1.1.9:8080/realms/test/.well-known/openid-configuration>



The screenshot shows a web browser window with the address bar displaying `10.1.1.9:8080/realms/test/.well-known/openid-configuration`. The browser's developer tools are open to the 'JSON' tab, showing the following JSON response:

```
{  "issuer": "http://10.1.1.9:8080/realms/test",  "authorization_endpoint": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/auth",  "token_endpoint": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/token",  "introspection_endpoint": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/token/introspect",  "userinfo_endpoint": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/userinfo",  "end_session_endpoint": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/logout",  "frontchannel_logout_session_supported": true,  "frontchannel_logout_supported": true,  "jwks_uri": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/certs",  "check_session_iframe": "http://10.1.1.9:8080/realms/test/protocol/openid-connect/login-status-iframe.html"}
```

--- snip ---

Environment

Enable the OIDC use case template.

OIDC Configuration

On the Juice Gateway, configure the OIDC template as follows:

OIDC Setting	Value
OIDC Authorize Endpoint	http://10.1.1.9:8080/realms/test/protocol/openid-connect/auth
OIDC Token Endpoint	http://10.1.1.9:8080/realms/test/protocol/openid-connect/token
OIDC JWKS URI	http://10.1.1.9:8080/realms/test/protocol/openid-connect/certs
Client ID	juice
Client Secret	WoWs70m1DXKLRbBxL4CSqi2KmWpaAQ0L

Form

API Spec

Reset All Fields

Edit Gateway

BASIC

Configuration

Placements

Hostnames

Custom Extensions

Custom Extensions

OIDC settings

Adds OIDC settings to server blocks generated by all URIs. Use OIDC to enable single sign-on (SSO) to authenticate to your app. The full specification for OIDC is available at https://openid.net/specs/openid-connect-core-1_0.html.

Clear section

OIDC Authorize endpoint *

Required

http://10.1.1.9:8080/realms/test/protocol/openid-connect/auth

The endpoint performs authentication of the end-user using request parameters defined by OAuth 2.0 and additional parameters and parameter values defined by OpenID Connect

OIDC Token endpoint *

Required

http://10.1.1.9:8080/realms/test/protocol/openid-connect/token

To obtain an Access Token, an ID Token, and optionally a Refresh Token, the Client sends a Token Request to the Token Endpoint to obtain a Token Response

OIDC JWKS URI *

Required

http://10.1.1.9:8080/realms/test/protocol/openid-connect/certs

The endpoint returns a JWKS containing public keys that enable clients to validate a JSON Web Token (JWT) issued by this OpenID Connect Provider.

Client ID *

Required

juice

Unique client identifier from IdP

Client secret *

Required

ZSbAMvg9TnFgrET3U4wD014WCh1LbWtT

Client secret must be entered with the client form IDP

Configure the Juice Web Component by setting OIDC authentication to **TRUE**

The screenshot shows a web interface for editing a web component. At the top, there are tabs for 'Form' (selected) and 'API Spec', and a 'Reset All Fields' button. On the left, a sidebar titled 'Edit Web Component' contains a 'BASIC' section with four items: 'Configuration', 'URIs', 'Workload Groups', and 'Custom Extensions' (which is highlighted with a green bar). The main area is titled 'Custom Extensions' and contains an 'OIDC Setting' section. This section has a descriptive paragraph about adding OIDC settings and a 'Clear section' button. Below this is a form field labeled 'OIDC authentication enabled' with a red asterisk and a 'Required' label. The field is a dropdown menu currently showing 'TRUE'. A note below the field explains that this value should be true if the location can be accessed only after authentication by the OIDC provider.

Form API Spec Reset All Fields

Edit Web Component

BASIC

- Configuration
- URIs
- Workload Groups
- Custom Extensions**

Custom Extensions

OIDC Setting

Adds OIDC settings to web component location blocks. Use OIDC to enable single sign-on (SSO) to authenticate to your app. The full specification for OIDC is available on the OpenID Foundation's website at https://openid.net/specs/openid-connect-core-1_0.html. Clear section

OIDC authentication enabled * Required

TRUE

Set this value to be true, if this location can be accessed only after authentication by the OIDC provider configured at the gateway. If no authentication is needed set this value to be false.

Cancel Submit Next

Test OIDC

1. Reload the browser tab with <https://juice.bigtechdojo.com>
2. You will be re-directed to Keycloak to complete a Login authorization flow
3. Login as : **juice-developer / juice-developer**
4. Keycloak will re-direct you to the application

Module 5 - Installing Custom Templates

You can also write your own templates, or have somebody else write them for you. In this case, you are going to use templates that are community supported and housed at on F5 Devcentral's Github.

So you be changing hats for this part. **Take off your App Delivery Team hat, and put on your Platform team hat.**

You will now be **helping the Platform Team**, so you are going to touch the NMS at the Operating System level.

1. Open a terminal window on the jumpbox
2. In this terminal window, SSH to the NMS system (ssh keys are already set up)

```
ssh nms.bigtechdojo.com
```

3. On the nms box (Hint: on a Windows system, use CTRL-SHIFT-V to paste into the RDP terminal window)

```
git clone https://github.com/f5devcentral/nms-community-templates
cd nms-community-templates
cd adm-lab
sudo cp -r bigtechdojo-logging-v1/ /etc/nms/modules/adm/templates/usecases/
```

4. In your jumpbox Terminal window, fully exit out of your SSH session to NMS by typing exit so you are back to your jumpbox prompt.
5. In your jumpbox terminal window, type

```
tail -f /var/log/syslog
```

6. **That is the end of the work for the Platform team. You can take off your Platform team hat, and put your App Delivery Team hat back on.**

7. Send web traffic to your gateway by refreshing the browser, and notice you do not yet see HTTP requests in your syslog on the jumpbox
8. Now, go back to your **Browser** window. In the ADM section go to your **Production** environment, select **Edit** from the “...” link on the right side of the screen. Click on the **Templates** section and select your **bigtechdojo-logging-v1** template in the **Use Cases** section, then **Submit** all changes
9. Edit your **Juice Gateway**
10. Select the **edit** button to modify the **Hostname**.
11. You will see new **Global block access_log** and **Global block error_log** fields, from the template.
12. Enter the following for the **Access Log** field:

`syslog:server=jumpbox.bigtechdojo.com:514,facility=local7,tag=nginx,severity=info`

13. Enter the following for the **Error Log** field:

`syslog:server=jumpbox.bigtechdojo.com:514,facility=local7,tag=nginx,severity=error`

The screenshot shows the 'Edit Gateway' configuration page in the Juice Gateway interface. The left sidebar has a 'Hostnames' tab selected. The main content area is divided into two sections: 'Server block access_log directive' and 'Server block error_log directive'. Each section has a 'Local Access Log arguments' or 'Local Error Log arguments' field containing the provided syslog configurations. There are also 'Clear section' and 'Cancel' buttons for each section. The bottom of the page has 'Cancel and Exit', 'Submit', and 'Next' buttons.

14. Click **Submit**

15. View the resulting configuration in Instance Manager, you will see your **access_log** and **error_log** directives created under the Server block.

16. On your jumpbox, tail -f /var/log/syslog, send web traffic to your gateway by refreshing the browser, and make sure you see nginx access log entries in your syslog

```
ubuntu@jumpbox: ~  
File Edit View Search Terminal Help  
.bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:25:01 jumpbox CRON[3059]: (ubuntu) CMD (~/.ddns/ddns.sh >/dev/null 2>&1)  
May  4 18:25:01 west-gw.bigtechdojo.com CRON[2895]: (ubuntu) CMD (~/.ddns/ddns.sh >/dev/null 2>&1)  
May  4 18:25:09 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:25:09 +0000] "GET /socket.io/?EI0=4&transp  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:25:09 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:25:09 +0000] "POST /socket.io/?EI0=4&trans  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:25:34 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:25:34 +0000] "GET /socket.io/?EI0=4&transp  
w.bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:25:34 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:25:34 +0000] "POST /socket.io/?EI0=4&trans  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:25:59 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:25:59 +0000] "GET /socket.io/?EI0=4&transp  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:25:59 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:25:59 +0000] "POST /socket.io/?EI0=4&trans  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:26:01 jumpbox CRON[3065]: (ubuntu) CMD (~/.ddns/ddns.sh >/dev/null 2>&1)  
May  4 18:26:01 west-gw.bigtechdojo.com CRON[2917]: (ubuntu) CMD (~/.ddns/ddns.sh >/dev/null 2>&1)  
May  4 18:26:24 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:26:24 +0000] "GET /socket.io/?EI0=4&transp  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:26:24 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:26:24 +0000] "POST /socket.io/?EI0=4&trans  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:26:49 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:26:49 +0000] "GET /socket.io/?EI0=4&transp  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"  
May  4 18:26:49 west-gw.bigtechdojo.com nginx: 10.1.1.4 - - [04/May/2023:18:26:49 +0000] "POST /socket.io/?EI0=4&trans  
bigtechdojo.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
```