Practical Exercise: Getting Started with WSO2 API Manager

Training Objective

Verify that the products required for running this tutorial are installed and configured. and deploy and test the data required to work with the sample.

Note: The participants are expected to be connected to the internet throughout in order to successfully complete the lab exercises.

Business Scenario

PizzaShack Limited wants to extend their website for placing and managing online orders as a part of their effort in becoming the #1 online pizza shop. They have also found it increasingly useful to build an application for smartphones. The application is a Web application allowing you to choose and buy a pizza online. They have subcontracted the development of the smartphone application to FunkyApps LLC. John Doe, Chief Architect of FunkyApps had some interesting feedback for PizzaShack. He suggested that the company considers monitoring of consumer statistics and probably looking into complex event processing in the future. John, also suggested that they make use of an API Store backed by a modern API Gateway providing security features such as OAuth 2.0 access tokens.

In order to achieve this, PizzaShack will be using WSO2 API Manager and a number of other WSO2 products for monitoring statistics, single sign-on, and so on.

The application leverages an API with 3 resources, which are exposed via the API Manager. Corresponding services are hosted in the WSO2 API Manager. WSO2 API Manager offers Analytics as a Cloud Service. This will be used for monitoring.

High-Level Steps

- Install WSO2 API Manager
- Configure API Manager Analytics
- Other installations
- Overview of the key directories in WSO2 API Manager
- Key configuration files
- Configure port offsets

Detailed Instructions Install WSO2 API Manager

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentation [1] for detailed instructions. If prerequisites are fulfilled, instructions on installing the product can be found for Linux, OS X or Windows at [2].

[1] **Installation Prerequisites**

WSO2 API Manager contains multiple runtimes. Before installing the runtimes, be sure that the appropriate prerequisites are fulfilled.

Environment compatibility

Listed below are the OS and database requirements.

Operating systems\ Databases

- Install a JDK version that is compatible with this product version.

  Tested JDKs

  The **WSO2 API-M** runtime has been tested with the following JDKs:

JDKS	Versions
CorrettoJDK	11, 17
AdoptOpenJDK	11, 17
OpenJDK	11, 17
Oracle JDK	11, 17
Temurin OpenJDK	11, 17

- All WSO2 Carbon-based products are generally compatible with most common DBMSs. The embedded H2 database is suitable for development, testing, and some production environments. For most enterprise production environments. However, WSO2 recommends that you use an industry-standard RDBMS such as Oracle, PostgreSQL, MySQL, MS SQL, etc. Additionally, WSO2 does not recommend the H2 database as a user store.
- It is **not recommended to use Apache DS** in a production environment due to scalability issues. Instead, use an LDAP like OpenLDAP for user management.
- On a production deployment, it is recommended that WSO2 products are installed on the latest releases of RedHat Enterprise Linux or Ubuntu Server LTS.
- Environments that WSO2 products are tested with.

Tested Operating Systems

As WSO2 API Manager is a Java application, you can generally run it on most operating systems. Listed below are the operating systems that have been tested with the API-M 4.2.0 runtime.

Operating System	Versions
Windows	2016
Ubuntu	18.04, 20.04
Red Hat Enterprise Linux	7.0
CentOS	7.4, 7.5

System requirements

Check the following system requirements for the API-M.

API-M runtime

Type	Requirement
Physical	• 3 GHz Dual-core Xeon/Opteron (or latest) • 4 GB RAM (2 GB for JVM and 2 GB for the operating system) • 10 GB free disk space • ~ Recommended minimum - 2 Cores. For high concurrences and better performances - 4 Cores. Disk space is based on the expected storage requirements that are calculated by considering the file uploads and the backup policies. For example, if three WSO2 product instances are running in a single machine, it requires a 4 GHz CPU, 8 GB RAM (2 GB for the operating system and 6 GB (2 GB for each WSO2 product instance)) and 30 GB of free space.
Virtual Machine (VM)	• 2 compute units minimum (each unit having 1.0-1.2 GHz Opteron/Xeon processor) • 4 GB RAM • 10 GB free disk space • One CPU unit for the operating system and one for JVM. Three WSO2 product instances running would require VM of 4 compute units, 8 GB RAM, and 30 GB free space.

~ 512 MB heap size. This is generally sufficient to process typical SOAP messages but the requirements vary with larger message sizes and the number of messages processed concurrently.

- 1 c5.large instance to run one WSO2 product instance.

EC2

Three WSO2 product instances can be run in 1 EC2 Extra-Large instance. Based on the I/O performance of the c5.large instance, it is recommended to run multiple instances in a larger instance (c5.xlarge).

[2] Installing the API Manager Runtime

Follow the steps given below to install the WSO2 API Manager runtime.

Installing the API Manager

1. Go to the WSO2 API Manager website, click **TRY IT NOW**, and then click **Zip Archive** to download the API Manager distribution as a ZIP file.
2. Extract the archive file to a dedicated directory for the API Manager, which will hereafter be referred to as `<API-M_HOME>`.

Setting up JAVA_HOME

You must set your `JAVA_HOME` environment variable to point to the directory where the Java Development Kit (JDK) is installed on the computer.

Info

Environment variables are global system variables accessible by all the processes running under the operating system.

On Linux/OS X

1. In your home directory, open the BASHRC file (.bash_profile file on Mac) using editors such as vi, emacs, pico, or mcedit.
2. Assuming you have JDK 11 in your system, add the following two lines at the bottom of the file, replacing `/usr/java/jdk-11.0.x` with the actual directory where the JDK is installed.

```
On Linux:
 export JAVA_HOME=/usr/java/jdk-11.0.x
 export PATH=${JAVA_HOME}/bin:${PATH}

On OS X:
 export
 JAVA_HOME=/System/Library/Java/JavaVirtualMachines/11.0.x.jdk/Contents/
 Home
```

3. Save the file.

   Info

   If you do not know how to work with text editors in a Linux SSH session, run the following command: `cat >> .bashrc.` Paste the string from the clipboard and press "Ctrl+D."

   To verify that the `JAVA_HOME` variable is set correctly, execute the following command.

   ```
   On Linux:
   echo $JAVA_HOME

   On OS X:
   which java

   If the above command gives you a path like /usr/bin/java, then it is a
   symbolic link to the real location. To get the real location, run the
   following:
   ls -l `which java`
   ```

   The system returns the JDK installation path.

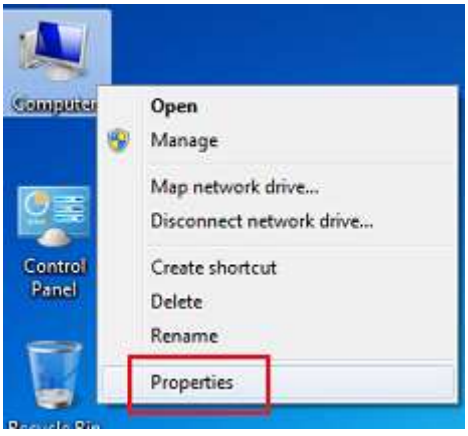
On Windows

Typically, the JDK is installed in a directory under `C:/Program Files/Java` , such as `C:/Program Files/Java/jdk-11.0.x`. If you have multiple versions installed, choose the latest one, which you can find by sorting by date.

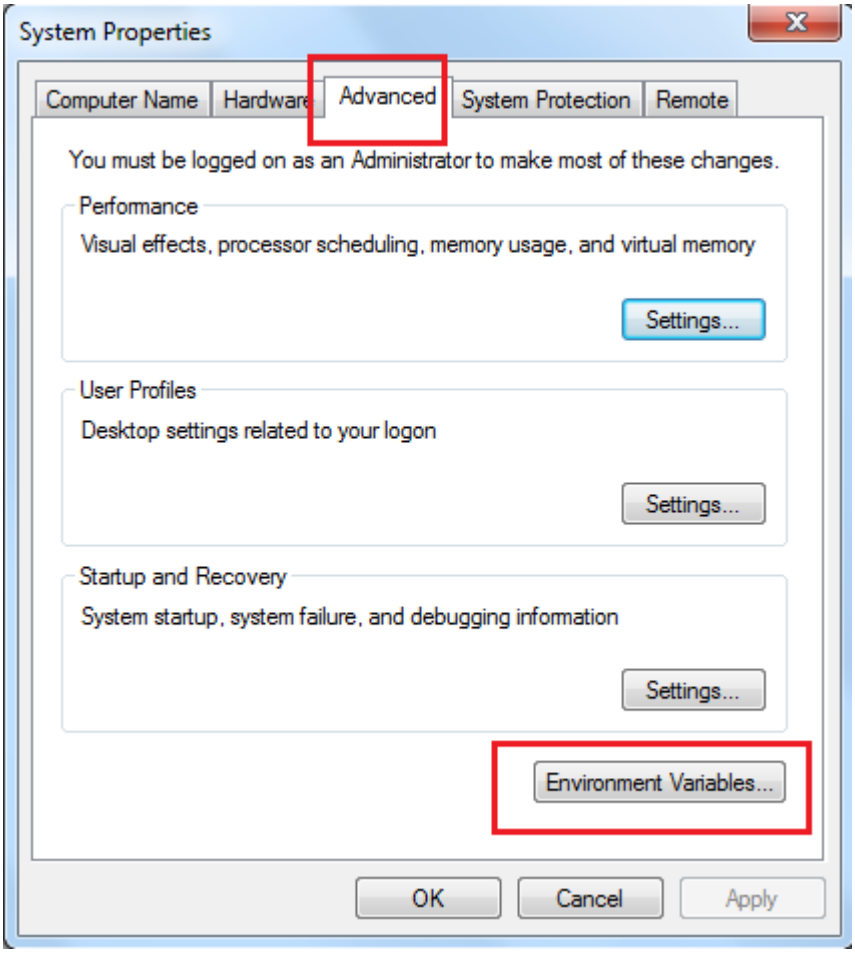
You set up `JAVA_HOME` using the System Properties, as described below. Alternatively, if you just want to set `JAVA_HOME` temporarily for the current command prompt window, set it at the command prompt.

Setting up JAVA_HOME using the system properties

1. Right-click the **My Computer** icon on the desktop and click **Properties.**



2. In the System Properties window, click the **Advanced** tab, and then click **Environment Variables**.



3. Click **New** under **System variables** (for all users) or under **User variables** (just for the user who is currently logged in).

4. Enter the following information:

   - In the **Variable name** field, enter: `JAVA_HOME`
   - In the **Variable value** field, enter the installation path of the Java Development Kit, such as: `c:/Program Files/Java/jdk-11.0.x`

The `JAVA_HOME` variable is now set and will apply to any subsequent command prompt windows you open. If you have existing command prompt windows running, you must close and reopen them for the `JAVA_HOME` variable to take effect, or manually set the `JAVA_HOME` variable in those command prompt windows as described in the next section. To verify that the `JAVA_HOME` variable is set correctly, open a command window (from the **Start** menu, click **Run**, and then type `CMD` and click **Enter**) and execute the following command:

```
set JAVA_HOME
```

The system returns the JDK installation path.

Setting system properties

If you need to set additional system properties when the server starts, you can take the following approaches:

- **Set the properties from a script**: Setting your system properties in the startup script is ideal because it ensures that you set the properties every time you start the server. To avoid having to modify the script each time you upgrade, the best approach is to create your startup script that wraps the WSO2 startup script and adds the properties you want to set, rather than editing the WSO2 startup script directly.
- **Set the properties from an external registry**: If you want to access properties from an external registry, you could create Java code that reads the properties at runtime from that registry. Be sure to store sensitive data such as username and password to connect to the registry in a property file instead of in the Java code and secure the properties file with the secure vault.

Info

When using SUSE Linux, it ignores `/etc/resolv.conf` and only looks at the `/etc/hosts` file. This means that the server will throw an exception on startup if you have not specified anything besides localhost. To avoid this error, add the following line above `127.0.0.1 localhost` in the `/etc/hosts` file: `<ip_address><machine_name> localhost`.

Configure API Manager Analytics

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentations [1]

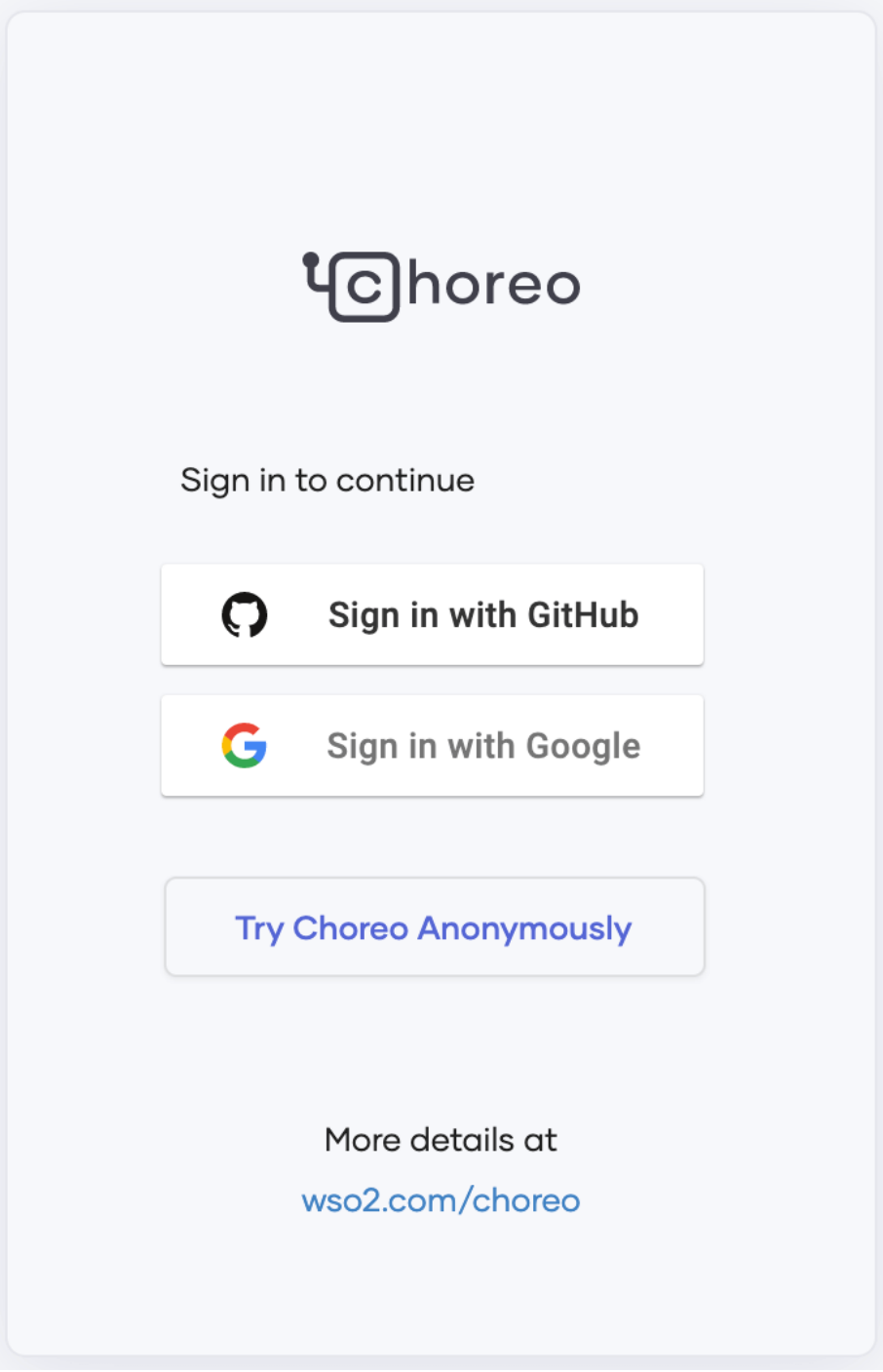
[1] **Choreo Based Analytics**

API Manager offers analytics as a cloud service. Therefore, you need to register with the analytics cloud in order to use API Manager Analytics. Follow the instructions below to get started with analytics:

Step 1 - Sign in to Choreo

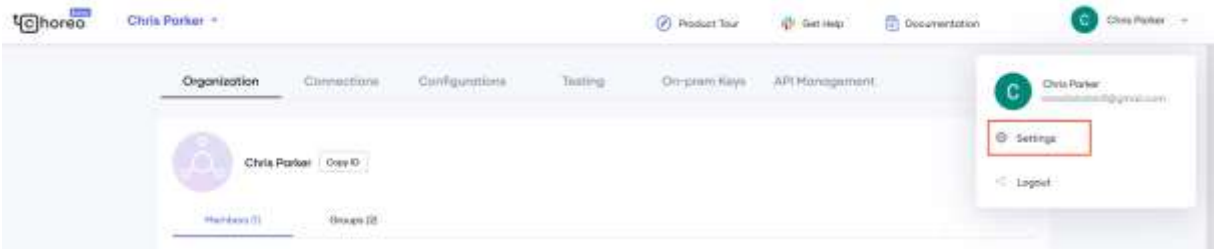
Follow the instructions below to sign in to Choreo.

1. Navigate to Choreo using the following URL. https://console.choreo.dev.
2. Sign-in to Choreo.

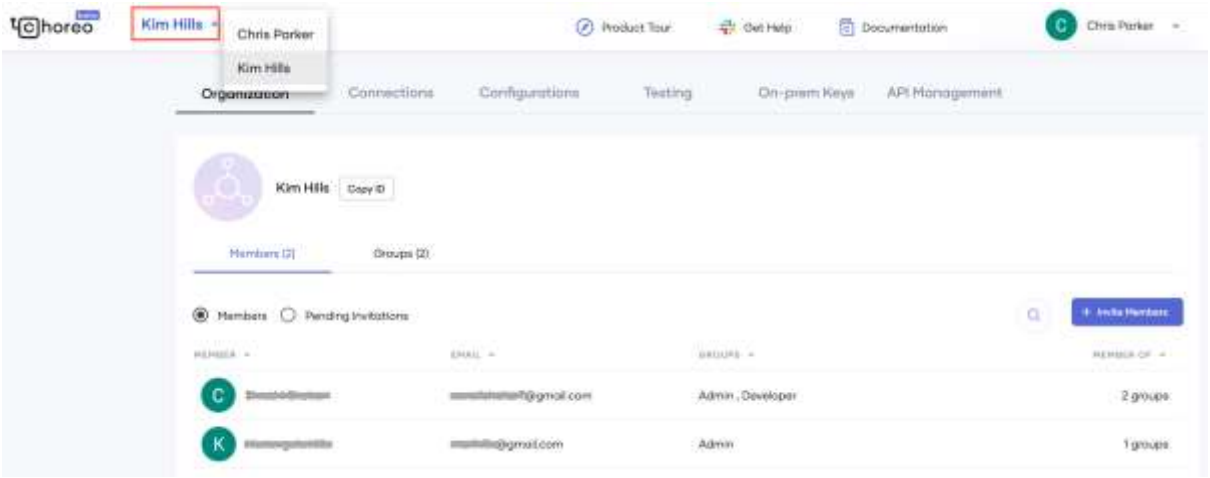


Follow the instructions below to register your on-premise environment:

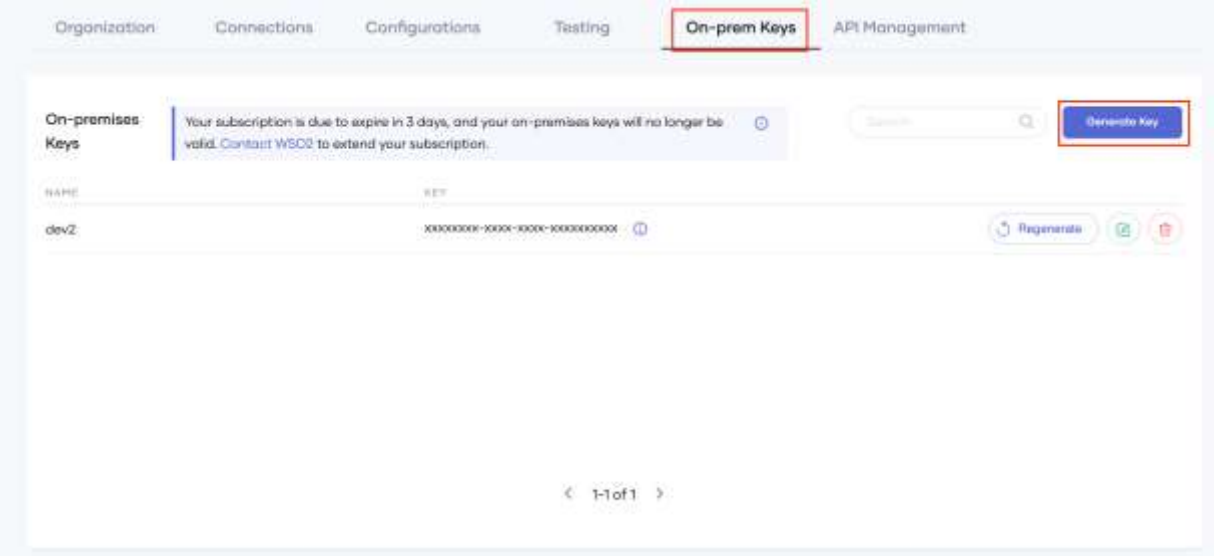
1. Click on the user profile in the top right corner of the screen and select **Settings**.



2. If you are a member of multiple organizations, select the appropriate organization from the top left-hand corner.



3. Select the **On-prem Keys** tab and click **Generate keys**.



4. Enter a suitable name for your environment (e.g., customer1-dev).
5. Click **Generate**.

   Info

   The validity period of the key and the number of keys that you are allowed to generate vary based on the type of user as explained below:

User Type	Descrption	Validity Period	Number of Keys Allowed
Free user	Refers to users who do not have a valid subscription for Choreo.	2 weeks	3 keys
Subscription customers	Refers to users who have a valid subscription for Choreo.	1 year	unlimited

6. Copy the key that was generated before closing the dialog box.

Step 3 - Configure the Gateway

API Analytics is delivered via the API Analytics Cloud. Therefore, the API Manager Gateway needs to be configured to publish analytics data into the cloud.

The Gateway configuration process varies based on the Gateway that you are using.

If your system connects to the service through a proxy server/firewall, you need to grant access to the following endpoints to access the Choreo Analytics Cloud service to publish data.

Host	Port	Protocol
`analytics-prod-incoming.servicebus.windows.net`	5671	AMQP
`analytics-prod-incoming.servicebus.windows.net`	5672	AMQP
`analytics-event-auth.choreo.dev`	443	HTTPS

Basic configurations

Follow the instructions below to publish analytics data to the analytics cloud via the API Gateway:

1. Open the `<API-M_HOME>/repository/conf/deployment.toml` file and update the `[apim.analytics]` config segment as follows:

```
[apim.analytics]
enable = true
config_endpoint = "https://analytics-event-auth.choreo.dev/auth/v1"
auth_token = "<use token that you generate>"
```

Note

- This is the basic configuration that you need to publish analytics data to the analytics cloud.

Parameter	Default Value
`worker.thread.count`	1 thread
`queue.size`	20000 requests
`client.flushing.delay`	10 seconds

2. Enter the on-premise token, which you obtained via the Choreo Portal in the **Register your environment** step, as the Auth token field.
3. Restart the API Gateway.

Advanced configurations

This section explains the additional configurations that you can carry out to fine-tune the analytics data publishing process. The following configurations are set to default values derived through testing by default. However, based on other factors, you may need to fine-tune these parameters.

Worker Thread Count

This property defines the number of threads that publish analytics data into the Analytics Cloud. The default value is one thread. One thread can serve up to 3200 requests per second with unrestricted internet bandwidth. In case a single thread is not enough to meet the load handled by your Gateway, you will encounter the following error message in Gateway logs.

```
Event queue is full. Starting to drop analytics events.
```

If you get the above-mentioned error during an API invocation spike, then increase the `Queue Size` as explained in next section. However, if you are getting this error repeatedly, then you should increase the worker thread count as shown below.

```
[apim.analytics]
enable = true
config_endpoint = "https://analytics-event-auth.choreo.dev/auth/v1"
auth_token = "<use token that you generate>"
properties.'worker.thread.count' = 2
```

Queue Size

This property denotes the number of analytics events that the Gateway keeps in-memory and uses to handle request bursts. The default value is set to 20000. As explained in the previous section, if you get the following error message in the Gateway logs during API invocation spikes, you should consider increasing queue size.

```
Event queue is full. Starting to drop analytics events.
```

However, another factor that you should consider when increasing the queue size is the memory footprint. A single analytics publishing event is around 1 KB. Therefore, you should plan the capacity to not hinder the JVM heap. To tweak the property, add the configuration as shown below.

```
[apim.analytics]
enable = true
config_endpoint = "https://analytics-event-auth.choreo.dev/auth/v1"
auth_token = "<use token that you generate>"
properties.'queue.size' = 10000
```

Client Flushing Delay

This property denotes the guaranteed frequency (in milliseconds) that the analytics events will be published to the cloud. Currently, analytics events are batched before being sent. Once a given batch is full, it is published into the Analytics Cloud. However, under low throughput, there might be a slight delay for a batch to be filled. The **Client Flushing Delay** property is used when such delays happen. A separate publisher will publish the analytics events after every Client Flushing Delay if the Event Queue, mentioned above,

is empty and also if none of the worker threads are currently publishing. By default, this is set to 10 seconds. To tweak the property, add the configuration as shown below.

```
[apim.analytics]
enable = true
config_endpoint = "https://analytics-event-auth.choreo.dev/auth/v1"
auth_token = "<use token that you generate>"
properties.'client.flushing.delay' = 15000
```

Step 4 - View the Analytics Dashboards

Invoke APIs and open Choreo Insights to view the dashboards.

Other Installations

In order to complete the use case described in this lab kit the following products must be installed: A REST API client or cURL [1], CLI tool [2] (Dev-Ops Tooling)

[1] https://www.getpostman.com/apps
[2] https://wso2.com/api-management/tooling/

For MAC OS X
Installing brew[1], JDBC driver for MySQL[2] and cURL[3]

[1] Install brew from https://brew.sh/
[2] In the terminal run

- `brew tap gbeine/homebrew-java`
- `brew install mysql-connector-java`

[3] In terminal run `'brew install curl'`

For Windows
Installing JDBC Drivers for SQL[1]

[1] Install Drivers from this location
[2] Install cURL from this location

Overview of the Key Directories in WSO2 API Manager

The structure of the <APIM_HOME> folder is as follows.

bin - This folder contains all the executable files including those scripts that are used to start/stop the application on Linux and Windows environments e.g., api-manager.sh andapi-manager.bat.

dbscripts – A collection of database scripts required to create the Carbon database and the API Manager specific database on a variety of database management systems.

lib –The lib directory houses all the jar files that will be converted to OSGi bundles at startup and copied to the dropins directory.

repository – The main repository for all kinds of deployments and configurations in Carbon. This includes all default services, created APIs, Carbon configurations etc.

res urces that may be used by the API-M.

samples - Sample APIs that can be used to explore the WSO2 API Manager functionality.

tmp – Will contain temporary files that are created when a product is run. These files will be cleared from time to time based on housekeeping tasks.

Key Configuration Files

File	Description
<PRODUCT_HOME>/repository/conf/deployment.toml	The main configuration file.
<PRODUCT_HOME>/repository/conf/log4j2.properties	The log4j2 configuration file used by WSO2Carbon.

Configure Port Offset

When you run multiple WSO2 products, multiple instances of the same product, or multiple WSO2 product clusters on the same server or virtual machines (VMs), you must change their default ports with an offset value to avoid port conflicts. The default HTTP and HTTPS ports (without offset) of a WSO2 product are 9763 and 9443 respectively. Port offset defines the number by which all ports defined in the runtime such as the HTTP/S ports will be changed. For example, if the default HTTP port is 9763 and the port offset is 1, the effective HTTP port will change to 9764. For each additional WSO2 product instance, you set the port offset to a uniquevalue. The default port offset is 0.

There are two ways to set an offset to a port:

- Pass the port offset to the server during startup. The following command starts theserver with the default port incremented by 3 : **/api-manager.sh -DportOffset=3**
- Set the Ports section of **<PRODUCT_HOME>/repository/conf/deployment.toml** asfollows:

```
[server]
offset=3
```

We will be using API-M and APIM Analytics for these exercises. APIM will run locally and API Manager Analytics will be delivered via API Manager Analytics cloud.

File	Port Offset
<API-M_HOME>/repository/conf/deployment.toml	0

Running the API Manager Runtime

Follow the steps given below to run the WSO2 API Manager runtime and access its web portals: **Management Console**, **API Publisher**, and the **Developer Portal**.

Starting the API-M server

Follow the steps given below to start the server.

1. Open a command prompt as explained below.

On **Linux/Mac** Establish an SSH connection to the server, log on to the text Linux console, or open a

OS	terminal window.
On **Windows**	Click **Start >Run**, type **cmd** at the prompt, and then press **Enter**.

2. Navigate to the `<API-M_HOME>/bin` folder from your command line.
3. Execute one of the commands given below.
    - To start the server:

      On MacOS/Linux

      ```
      sh api-manager.sh
      ```

      On Windows

      ```
      api-manager.bat –run
      ```

    - To start the server in background mode:

      On macOS/Linux
      ```
      sh api-manager.sh start
      ```

      On Windows

      ```
      api-manager.bat –start
      ```

When the server starts successfully, the following log is printed: `"WSO2 Carbon started in 'n' seconds"`

Accessing the Web Portals

When you start the API-M runtime, all of its web portals are started. You will see the URLs of each portal printed in the server-startup log as shown below.

Note that the server is running on `localhost` by default. You can use these URLs to access the web portals on your computer from any other computer connected to the LAN. When accessing the portals from the same computer where it is installed, you can use `localhost` instead of the IP address.

To sign in to each web portal:

1. Copy the web portal's URL to your browser.

   Tip

   The web browser typically displays an "insecure connection" message, which requires your confirmation before you can continue.

2. Enter your username and password to sign in.

   Tip

   The default username and password is `admin`.

Stopping the API-M server

- To stop the API-M server standalone application, go to the terminal and press *Ctrl*+*C*.
- To stop the API-M server in background mode:

  On macOS/Linux
  ```
  sh api-manager.sh stop
  ```

  On Windows

  ```
  api-manager.bat --stop
  ```

Tip

For additional options, you can use with the startup commands, add `-help` after the start-up command as shown below.

```
sh api-manager.sh -help
```

Troubleshooting server startup errors

- If you are in a Windows environment, the HTTPS listener would have started on a host address of 0:0:0:0:0:0:0:0. You can verify that from the Carbon logs. In that case, you need to define 0:0:0:0:0:0:0:0 as the bind-address in `<API-M_HOME>/repository/resources/security/listenerprofiles.xml` to avoid errors during SSL reloads.
- If you are on a Mac OS, you may encounter the following startup error with similar logs.
- `[2021-04-16 08:48:27,655] ERROR - InboundEndpoint Error initializing inbound endpoint SecureWebhookServer`

- `[2021-04-16 08:48:27,655] ERROR - InboundEndpointDeployer Inbound Endpoint deployment from the file : /Users/sanjeewa/Downloads/wso2am-4.0.0/repository/deployment/server/synapse-configs/default/inbound-endpoints/SecureWebhookServer.xml : Failed. org.apache.synapse.SynapseException: Error initializing inbound endpoint SecureWebhookServer at org.apache.synapse.inbound.InboundEndpoint.init(InboundEndpoint.java:83) ~[synapse-core_2.1.7.wso2v227.jar:2.1.7-wso2v227]`

  This may occur due to a native `launchd` service `com.apple.ftp-proxy.plist` living at `/System/Library/LaunchDaemons/com.apple.ftp-proxy.plist` that fires `/usr/libexec/ftp-proxy`. To fix this issue, change the default port that the webhooks HTTPS inbound endpoint is listening in all the Gateway nodes in `<APIM_HOME>/repository/conf/deployment.toml`.

  ```
  [apim.webhooks.https]
  port=8021
  ```

  For the Control Plane nodes (with the Publisher), change the `deployment.toml` file as follows:

  ```
  [[apim.gateway.environment]]
  ######## other properties ########
  websub_event_receiver_https_endpoint = "https://localhost:8021"
  ```

Expected Outcome

As the API-M was not given an offset, it will run on the default port.