# Gaia API – R81.20

Khalid Al-Shawwaf
Solutions Architect - Americas

Duration: 2 Hours

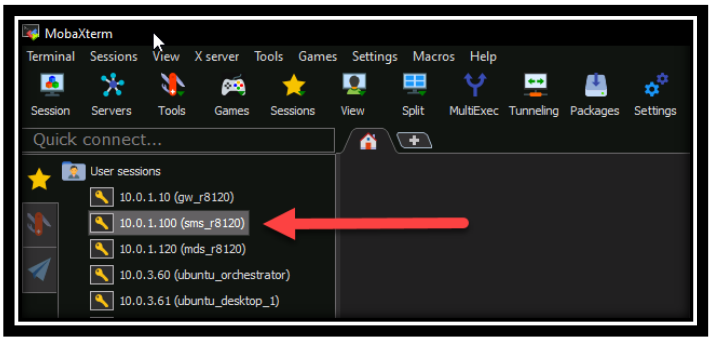
## Introduction

This workshop will teach us how to use the Gaia API to manage the Gaia OS. There are multiple ways to use the API such as:

- Directly on the appliance using the mgmt_cli tool.
- Using Web tools such as Postman.
- Via the management API (reverse proxy).
- Using the Check Point Python SDK.

## Exercise 1: Gaia API Infrastructure

1. Use the pre-installed SSH client ***MobaXterm*** to connect to the bookmarked management server session `10.0.1.100(sms_r8120)` (*double-click to connect*).




2. Use the command `gaia_api` to list all the available options.

```
[Expert@sms_r8120:0]# gaia_api
Usage: gaia_api [OPTION]
stop                                                  Stop server
    [-f|--force <comment>]                            Use 'force all' to terminate running tasks
start                                                 Start server
restart                                               Restart server
    [-f|--force <comment>]                            Use 'force all' to terminate running tasks
status                                                Show server's status
    [-s <comment>]                                    File named debug_pkg_<comment>.tar will be created in user home directory
    [-t|--test <enable|disable>]                      Check status of engine connectivity tests (default false)
login                                                   Login to server with current user
    [-v|--version] {1|1.1|..}
    [-f|--format] {text|json}
access                                                REST API access
    <-u|--user> <username> <-e|--enable> <true|false>  Grant REST API access to specific user
    <-u|--user> unlocal_users <-e|--enable> <true|false> Grant REST API access to unlocal users (Radius/Tacacs)
fingerprint                                           Show server's fingerprint
version                                               Show GAIA API version
```

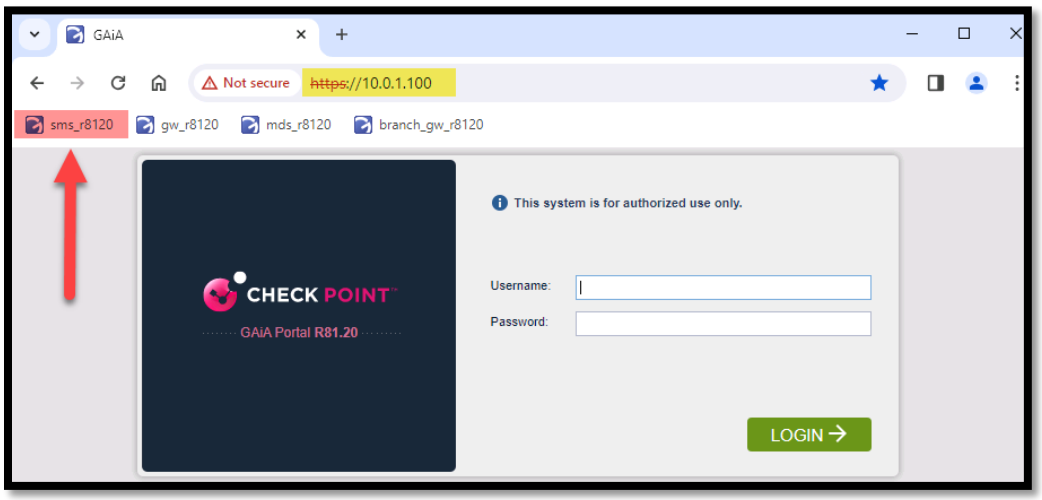
3. Run the command `gaia_api status` and verify the status of the server.

```
[Expert@sms_r8120:0]# gaia_api status

API Status:
--------------------
Build: cp991255113
Uptime: 21:36:07.929022
Current Sessions: 0
Latest Version: 1.7

Processes:

Name            State        PID
--------------------------------
GAIA_API        Started      48468
GAIA_API_DOCS   Started      48466
APACHE          Started      48528
CONFD           Started      15655
CLISHD          Started      71881
CELERY          Started      48465
REDIS           Started      48467

Port Details:
------------------
APACHE Gaia Port:         443

-------------------------------------------
Overall API Status: Started
-------------------------------------------
```

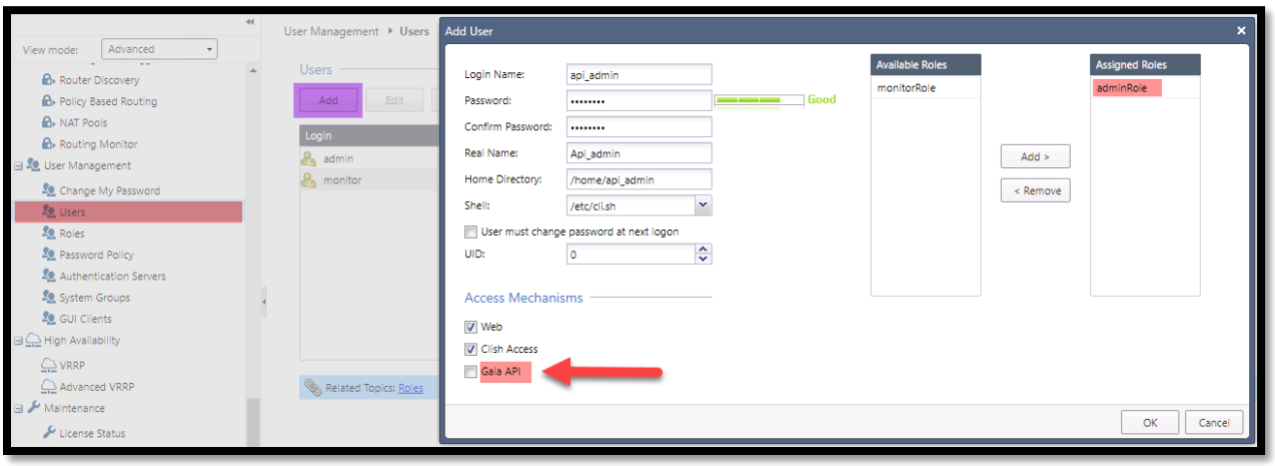
*The Gaia API has a login option. Access to the Gaia API requires a login via a user who is allowed access to the Gaia API infrastructure. Only predefined administrators are allowed by default.*

4. Open a web browser and log in to the Gaia Portal of the management server `https://10.0.1.100`. Login using the credentials below:

   - Username: **admin**
   - Password: **Cpwins!1**




5. Add a new user with the following details:
   - Login Name: **api_admin**
   - Password: **Cpwins!1**




*The Gaia API access permission is unchecked by default, leave it* `unchecked`*.*

6. Use the `mgmt_cli` tool and attempt to log in to the Gaia API server as ***api_admin*** and show the hostname. Use the command below:

```
mgmt_cli -u api_admin -p 'Cpwins!1' --context gaia_api show hostname
```

- We are wrapping the password with single quotes as it contains special characters.
- The `!1` command at the prompt tells the shell to rerun the command on line 1 of the history list.)
- Both `show hostname` and `show-hostname` commands are valid.

```
[Expert@sms_r8120:0]# mgmt_cli -u api_admin -p 'Cpwins!1' --context gaia_api show-hostname
code: "err_login_failed_wrong_username_or_password"
errors: "Login authentication failed"
message: "Login Failed Due to Wrong Username or Password"
```

*Notice that the* ***api_admin*** *was not granted the API permissions, hence the user is unable to authenticate to Gaia API.*

7. Use the command `gaia_api access -u api_admin -e true` to <u>grant the user permission to access the Gaia API</u>.

```
[Expert@sms_r8120:0]# gaia_api access -u api_admin -e true
[Expert@sms_r8120:0]#
```

*Notice that the command will not return any response.*

8. From the Gaia portal, review the `api_admin` permissions and make sure the `Gaia API` is now checked.


9. Run the API command again to retrieve the hostname.

```
mgmt_cli -u api_admin -p 'Cpwins!1' --context gaia_api show hostname
```

```
[Expert@sms_r8120:0]# mgmt_cli -u api_admin -p 'Cpwins!1' --context gaia_api show-hostname
name: sms_r8120
```

*It is also possible to make the changes using the Gaia Portal via the user settings page or via the **CLISH** command `add rba user api_admin access-mechanisms Gaia-API`*

## Exercise 2: Using the Gaia API Documentation

The Gaia API documentation is available online and can be accessed via the link below:
`https://sc1.checkpoint.com/documents/latest/GaiaAPIs/#introduction~v1.7%20`

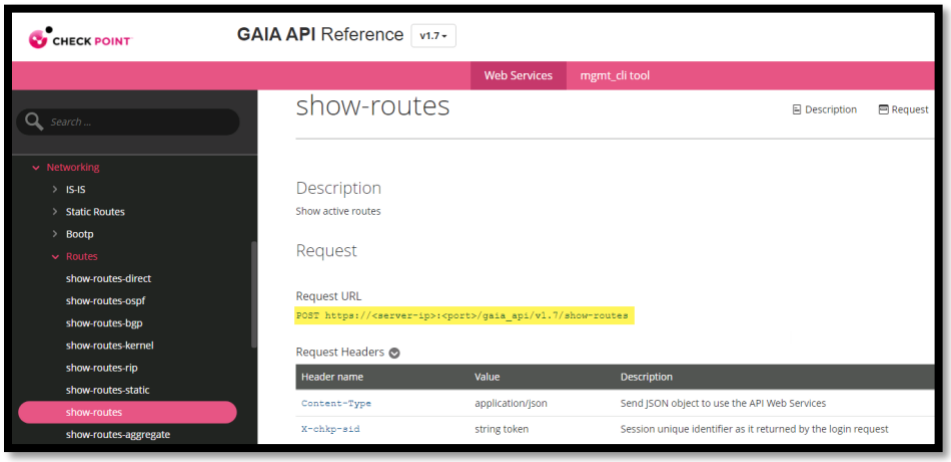
The documentation service is also a part of the local Gaia API server. We will use the API reference to find the API commands for the `mgmt_cli` tool and for the `Web Services`. All versions are accessible up to the latest version are installed on the server.

1. Navigate to the Gaia documentations portal on the management server via the link: `https://10.0.1.100/gaia_docs/`.

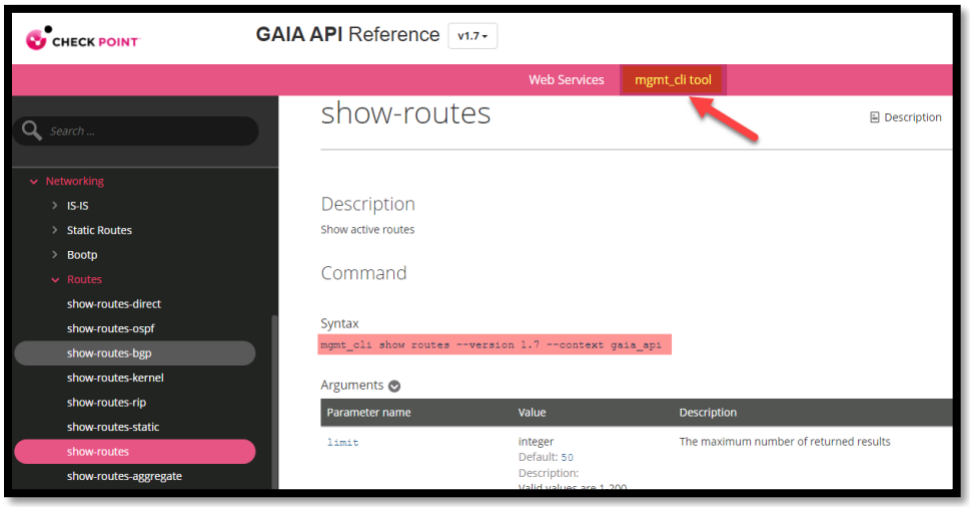
*The forward slash at the end is necessary.*

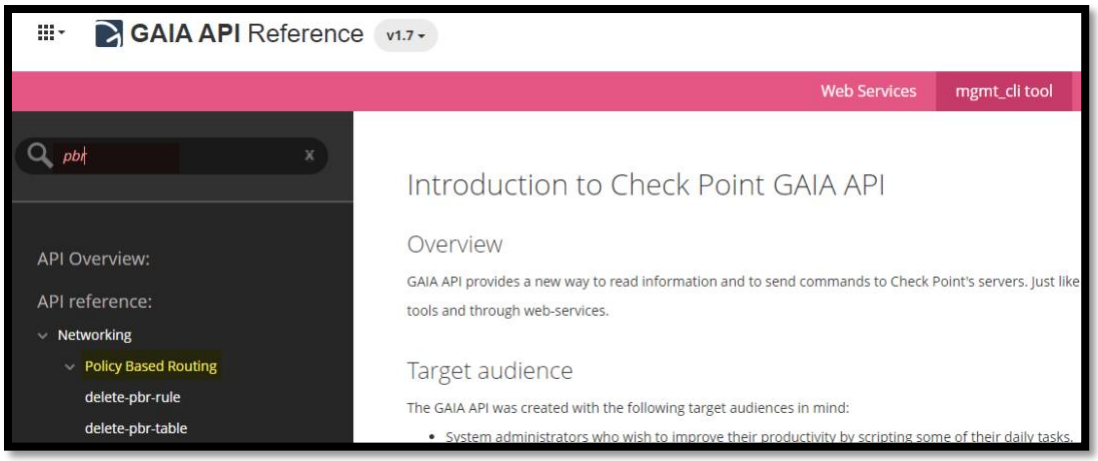

2. From the menu on the left side, navigate to the routes section and select the show-routes command documentation `Networking -> Routes -> show-routes`.




3. The API reference defaults to the Web Service page, switch to the *`mgmt_cli tool`* section from the top menu and check the command format when using `mgmt_cli`.

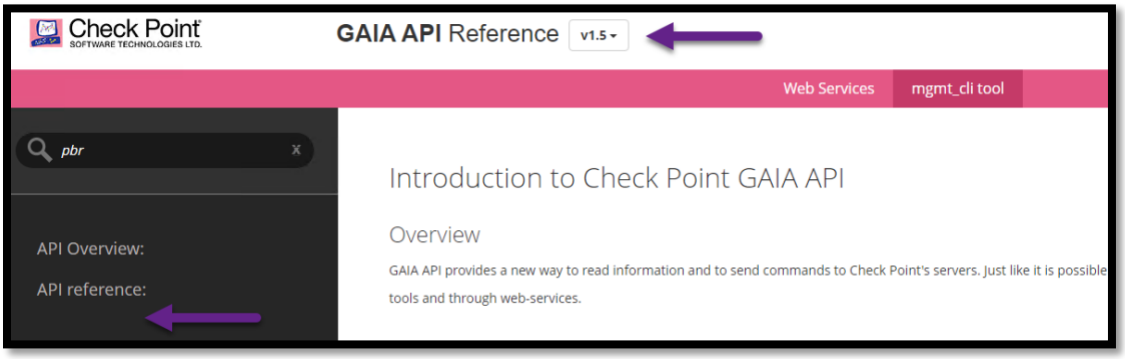


4. Use the search tab to search for policy-based routing `PBR` API calls.




*Note that you need to search using the words expected in the API call. The search mechanism does not look for titles or section names. If you search for `Policy Based Routing`, you will get no results.*

5. Change the reference version from **v1.7** to **v1.5**. Notice that the results disappeared.




- The policy-based routing API call was added in **v1.7**.
- Switching the reference version will only show documentation for **v1.5**, which does not include the Policy Based Routing API commands.
- The Changelogs for the Gaia API are documented under the Gaia API solution sk143612.
- The version can be used in the API command using the argument (--version). For example:
  `mgmt_cli show version` **`--version 1.5`** `--context gaia_api`

```
[Expert@sms_r8120:0]# mgmt_cli -u api_admin -p 'Cpwins!1' show version --version 1.5 --context gaia_api
os-build: '627'
os-edition: 64-bit
os-kernel-version: 3.10.0-1160.15.2cpx86_64
product-version: Check Point Gaia R81.20
```

6. Switch the documentation version back to **v1.7**.

## Exercise 3: Using mgmt_cli tool.

The mgmt_cli tool is installed as part of the Check Point Gaia OS on all R80 machines. and can be used in scripts running in expert mode.

It is also installed as part of the R81 SmartConsole installation typically under: *C:\Program Files(x86)\CheckPoint\SmartConsole\R81\PROGRAM* and can be copied to run on any Windows machine. This exercise will use the tool to perform Gaia API calls.

1. Connect to the CLI of the management using **MobaXterm** and issue the command:
   `mgmt_cli show-routes --context gaia_api`

*Note that the* `mgmt_cli` *tool will prompt you for the admin credentials, as it was not provided in advance with the command above.*

```
[Expert@sms_r8120:0]# mgmt_cli show-routes --context gaia_api
Username: api_admin
Password:
from: 1
objects:
- address: 0.0.0.0
  age: 105591
  mask-length: 0
  next-hop:
    gateways:
    - address: 10.0.1.10
      interface: eth0
  protocol: Static
- address: 10.0.0.1
  mask-length: 32
  next-hop:
    interface: loop00
  protocol: Connected
- address: 10.0.1.0
  mask-length: 24
  next-hop:
    interface: eth0
  protocol: Connected
- address: 127.0.0.0
  mask-length: 8
  next-hop:
    interface: lo
  protocol: Connected
to: 4
total: 4
```

2. Run the command again and this time provide the credentials using the flag -u for the username and -p for the password.

```
mgmt_cli show-routes --context gaia_api -u api_admin -p 'Cpwins!1'
```

```
[Expert@sms_r8120:0]# mgmt_cli show-routes --context gaia_api -u api_admin -p 'Cpwins!1'
from: 1
objects:
- address: 0.0.0.0
  age: 105704
  mask-length: 0
  next-hop:
    gateways:
    - address: 10.0.1.10
      interface: eth0
  protocol: Static
- address: 10.0.0.1
  mask-length: 32
  next-hop:
    interface: loop00
  protocol: Connected
- address: 10.0.1.0
  mask-length: 24
  next-hop:
    interface: eth0
  protocol: Connected
- address: 127.0.0.0
  mask-length: 8
  next-hop:
    interface: lo
  protocol: Connected
to: 4
total: 4
```

*Note that it is possible to use* `environment variables` *with the mgmt_cli tool. The table below shows the variables that can be exported instead of providing the username and password on the terminal.*

| Parameter name | Short name | Environment variable |
|---|---|---|
| User name | -u | MGMT_CLI_USER |
| Password | -p | MGMT_CLI_PASSWORD |
| Check Point server address | -m | MGMT_CLI_MANAGEMENT |

3. Log in to the Gaia API using the command below. Notice that we received a session ID `sid` back from the server.

```
[Expert@sms_r8120:0]# mgmt_cli login -u api_admin -p 'Cpwins!1' --context gaia_api
api-server-version: "1.7"
last-login-was-at:
  iso-8601: "2023-11-16T10:05+16.00.0"
  posix: "1700157950128"
read-only: "false"
session-timeout: "600"
sid: "98399212057579212057499212048536992120102551059212049529212010299 39"
url: "https://127.0.0.1:443/gaia_api"
```

4. Run a new API call to get the hostname of the Gaia server using the session ID we received in the previous step.

```
[Expert@sms_r8120:0]# mgmt_cli show-hostname --context gaia_api --session-id 98399212057579212057499212048536992120102551059212049529212010299 39
name: sms_r8120
```

*Note that it is inconvenient to provide the `session-id` manually to each command. The proper way is to **save the reply from the server to a text file** where mgmt_cli can read the `SID`.*

5. Use the command below to log in to the Gaia API server and save the results in a text file called id.txt.

```
mgmt_cli login -u api_admin -p 'Cpwins!1' --context gaia_api --format
json > id.txt
```

```
[Expert@sms_r8120:0]# mgmt_cli login -u api_admin -p 'Cpwins!1' --context gaia_api --format json > id.txt
[Expert@sms_r8120:0]#
[Expert@sms_r8120:0]# cat id.txt
{
  "api-server-version": "1.7",
  "last-login-was-at": {
    "iso-8601": "2023-11-16T10:17+16.00.0",
    "posix": 1700158677574
  },
  "read-only": false,
  "session-timeout": 600,
  "sid": "98399357921201015692120494892120974892120999887117 39",
  "url": "https://127.0.0.1:443/gaia_api"
}
```

*Note that we're using the argument `--format` to request a response in JSON format. you can also use `-f` to request the server to reply using JSON format*

6. Use the `mgmt_cli` to show the hostname while reading the session ID from the file we saved in the previous step. Use the `--format json` to receive the reply in a JSON format.

```
mgmt_cli show hostname -s id.txt -f json
```

```
[Expert@sms_r8120:0]# mgmt_cli show hostname -s id.txt -f json
{
  "name": "sms_r8120"
}
```

*The context **gaia_api** was saved in the `id.txt` file (part of the url). Hence, it's no longer required to be provided with the commands.*

7. Finally, log out using the command below. This renders the session ID invalid, and a login is required before issuing any new command.

```
mgmt_cli logout -s id.txt -f json
```

```
[Expert@sms_r8120:0]# mgmt_cli logout -s id.txt -f json
{
  "message": "OK"
}
```

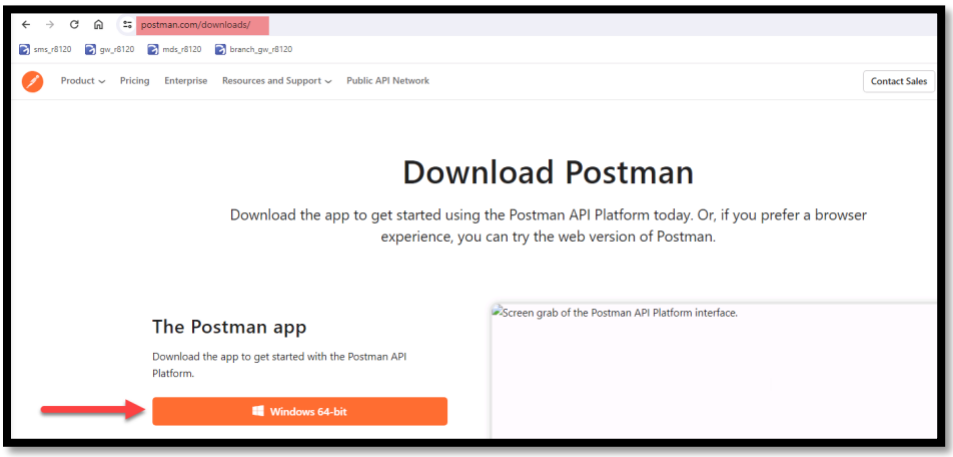
 

## Exercise 4: Working with Postman

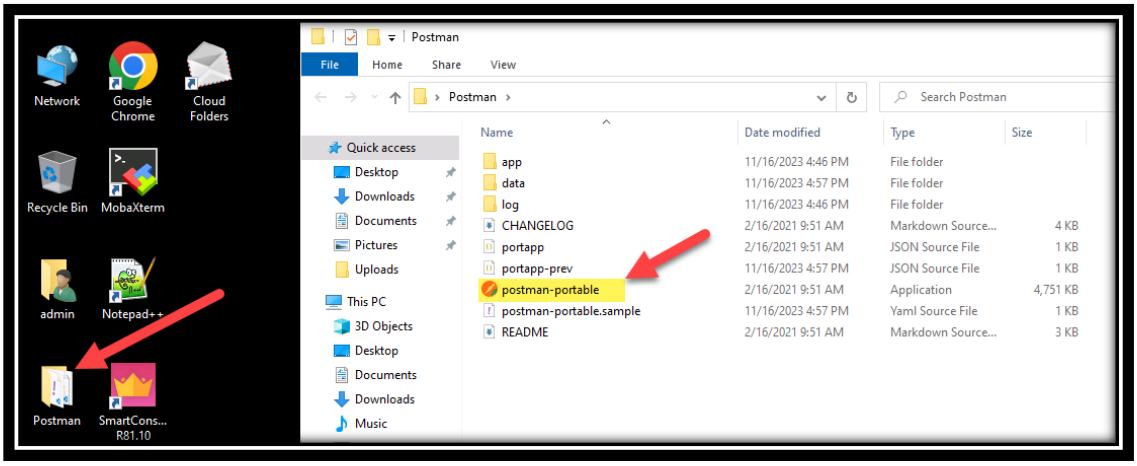
Up to this point, we used the `mgmt_cli` tool to run the Gaia API commands. In this exercise, we will use Postman to access the Gaia API.

New Versions of Postman require an account. **If** you have an account **or** if you are willing to sign up, follow step 1 to download and install Postman. Otherwise, use the Portable version as described below.

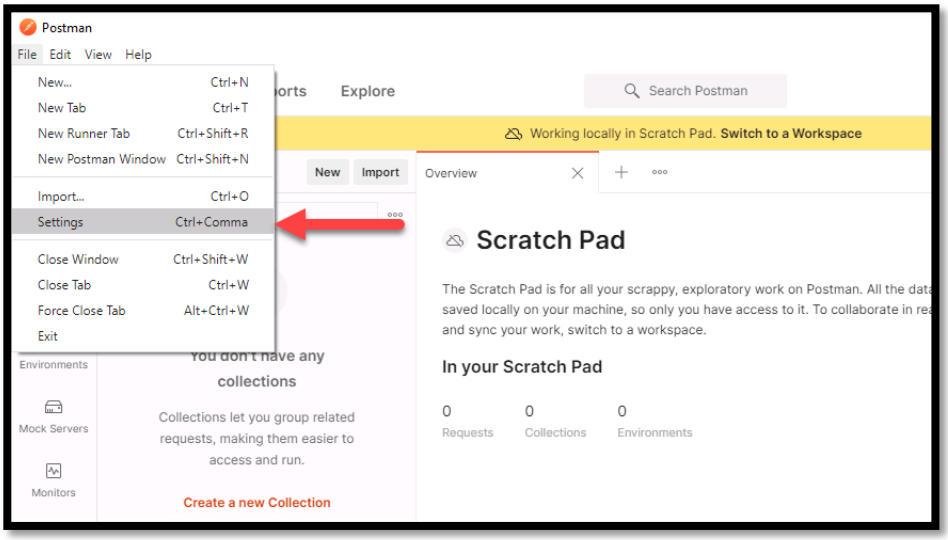
- To use the latest version of Postman (account is required), from the win_10 Jump box, open a web browser, and navigate to `https://postman.com/downloads/` and `download and install` the latest postman version.



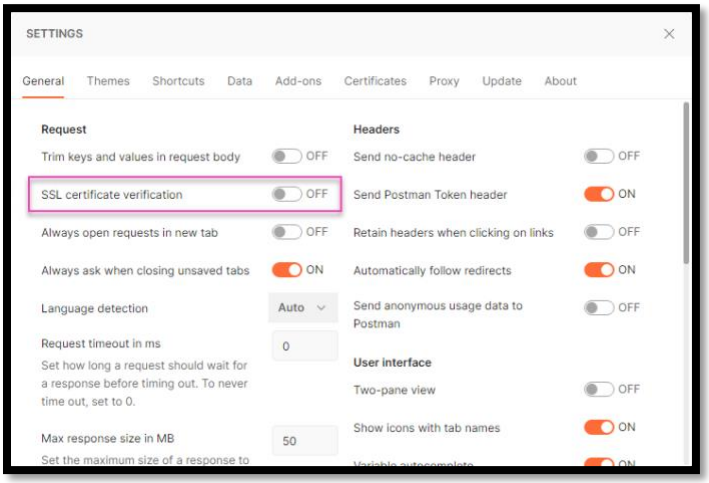

1. Locate the Postman folder on the desktop and launch the postman-portable application.



2. Navigate to the settings windows in Postman.

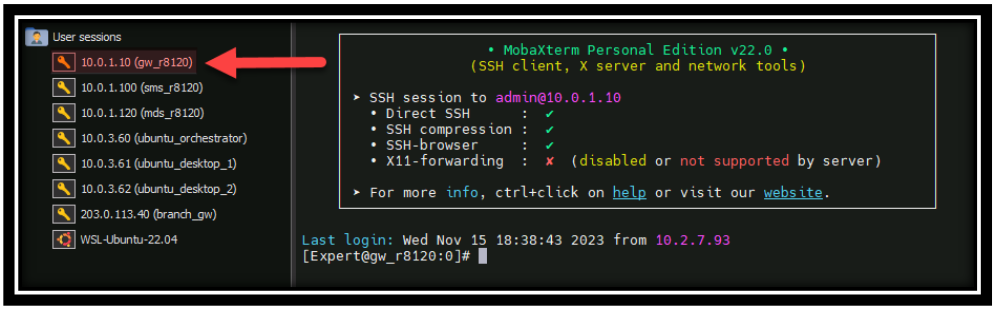



3. Disable the `SSL certificate verification`.



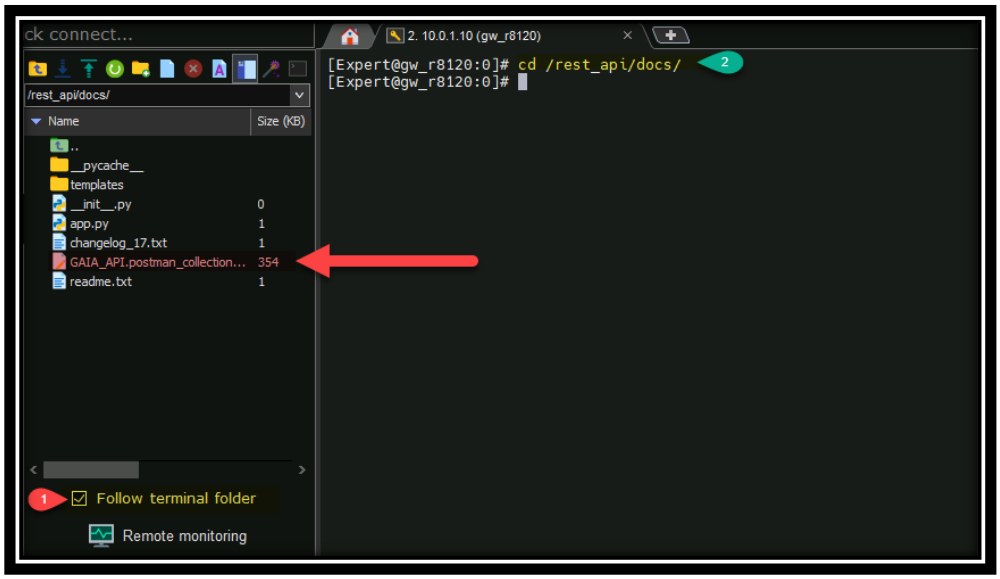

*This step is necessary as we have a self-signed certificate on our Gaia servers.*

4. Use `MobaXterm` to connect to the Gateway `10.0.1.10(gw_r8120)` over SSH.

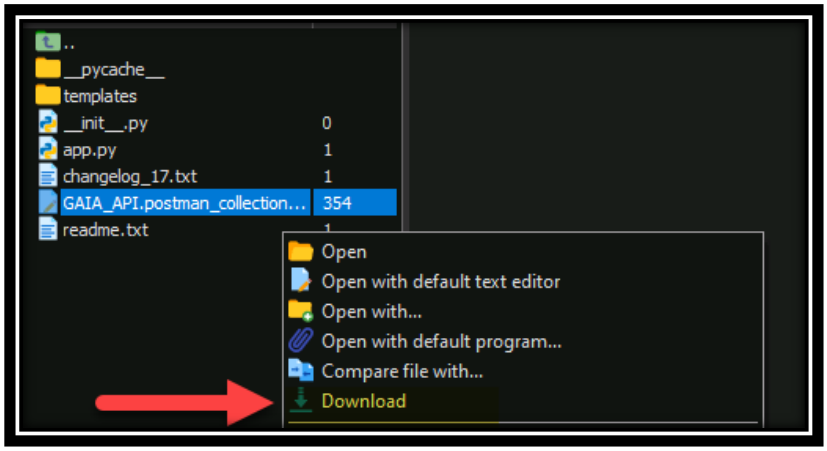


5. Make sure to check the option Follow terminal folder, then run the command:
   `cd /rest_api/docs`

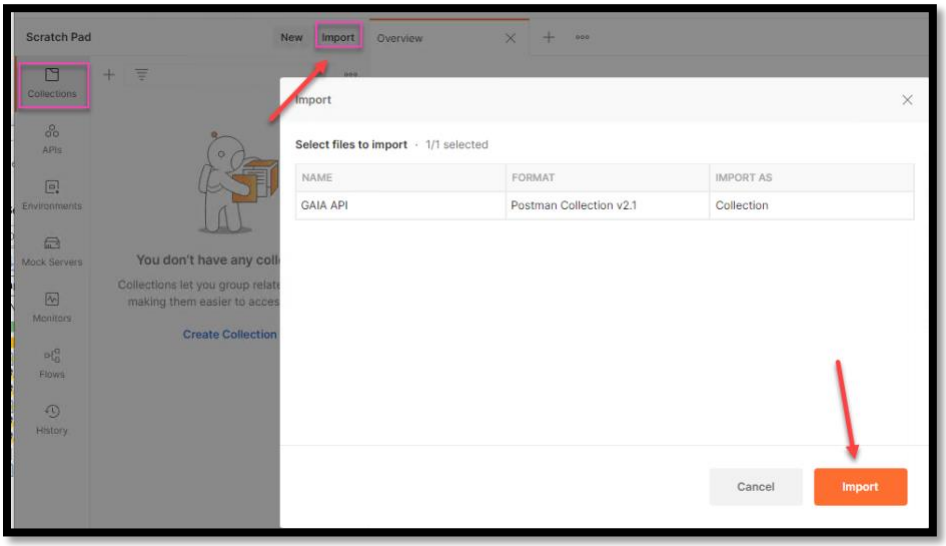
*This is the location where the Gaia API collection for postman is located by default.*



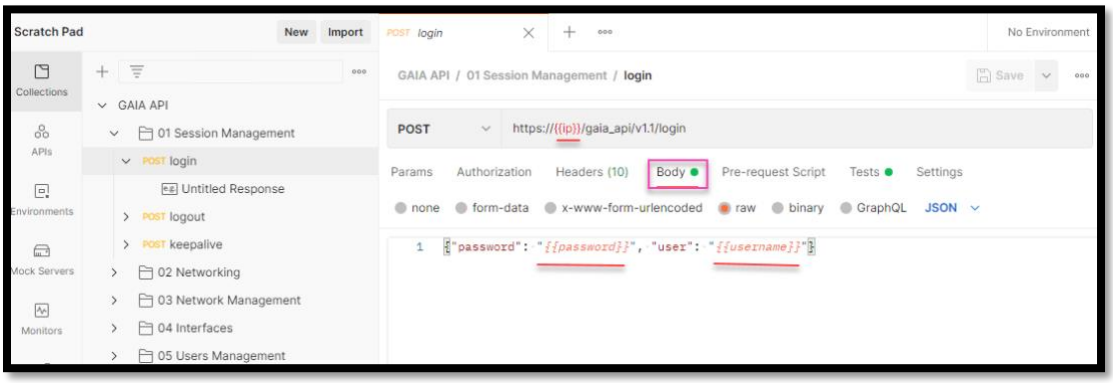

6. Drag and drop the file to your desktop. You can also right-click and select download to download the file.




7. Import the collection from the file, as shown in the screenshot below.

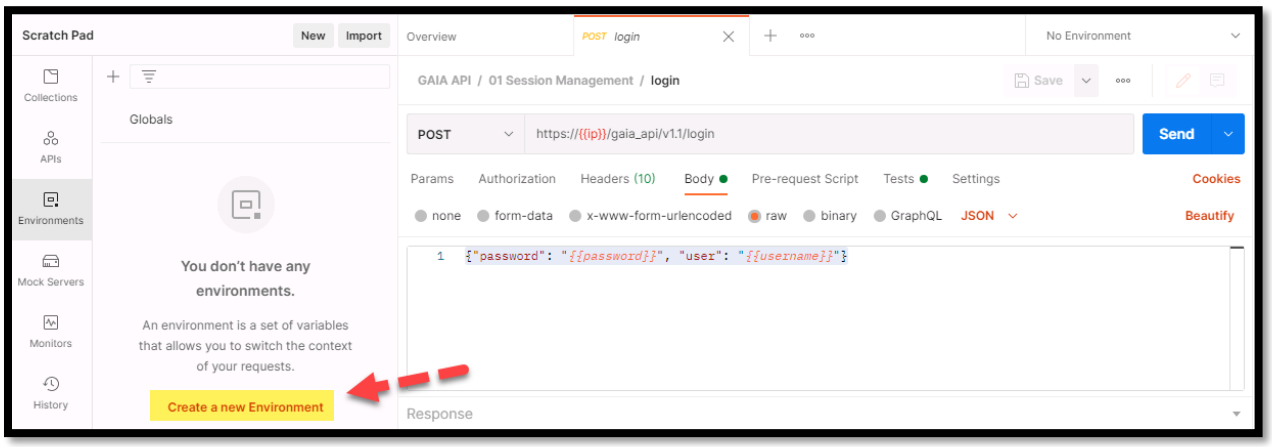



8. Expand the Gaia API list and under 01 Session Management select POST Login and switch to the Body tab.

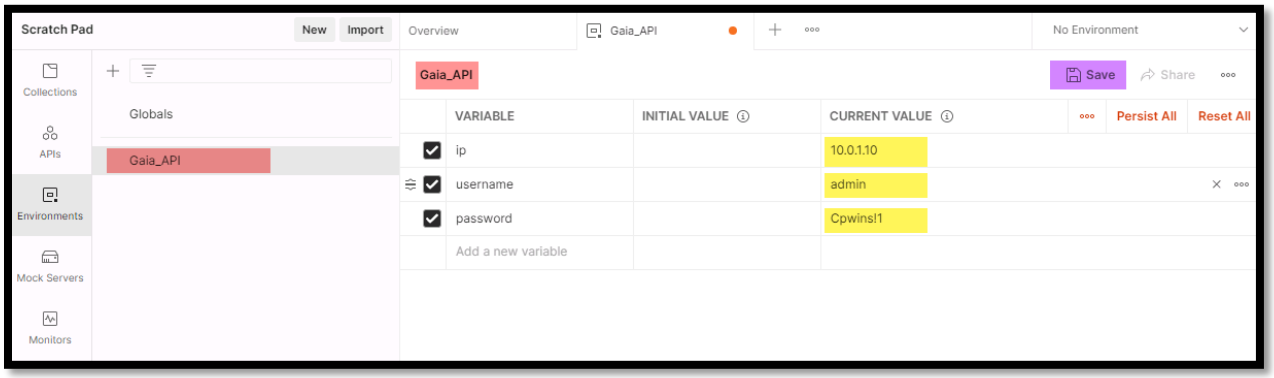



- *Note that the IP, user, and password are created as variables.*
- *You can provide the details manually, but the proper way is to create an environment that contains the values of such variables.*
- *Each API call has the same {{ip}} variable and manually replacing it with your address is not logical.*

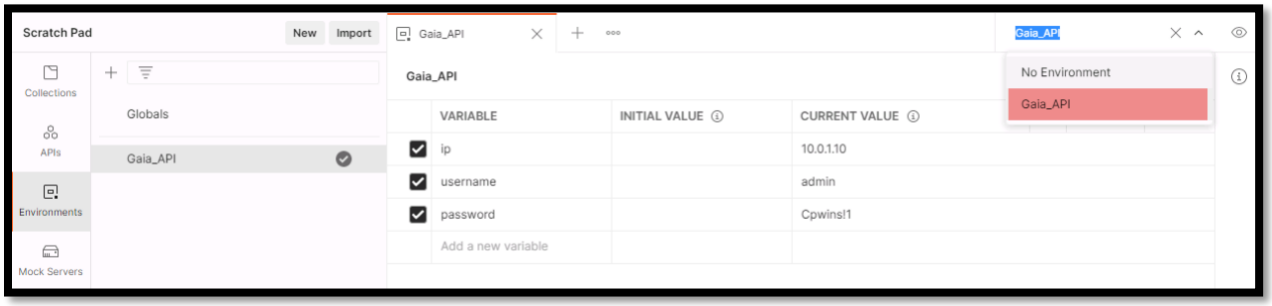
9. Switch to the Environments view and select Create a new Environment




10. Fill in the details to provide values for those variables and click Save.

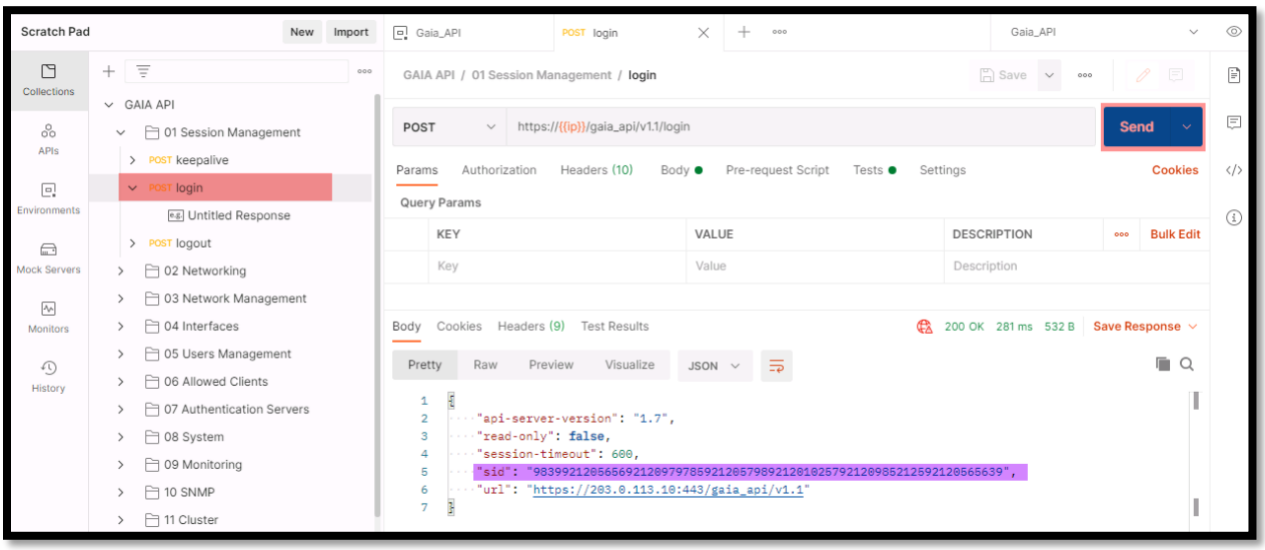



11. Select the environment we just created.

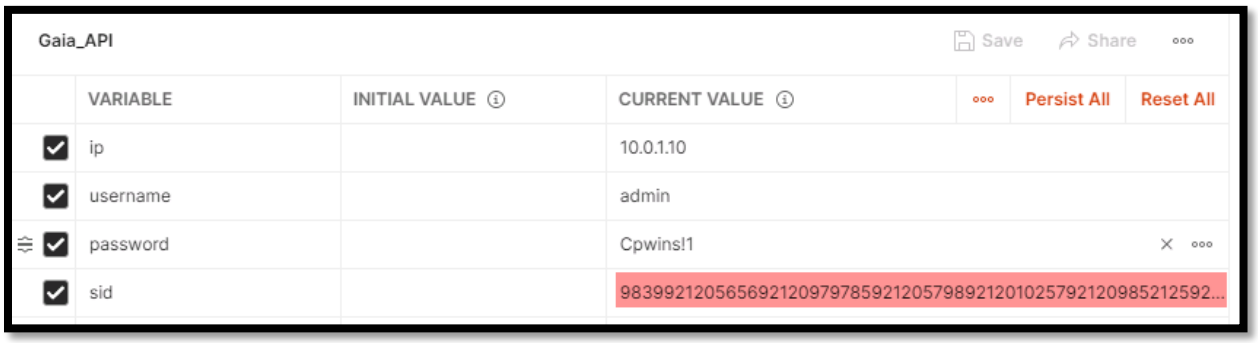



*Tip: Hover the mouse over the {{ip}} variable and make sure you can see the real address.*

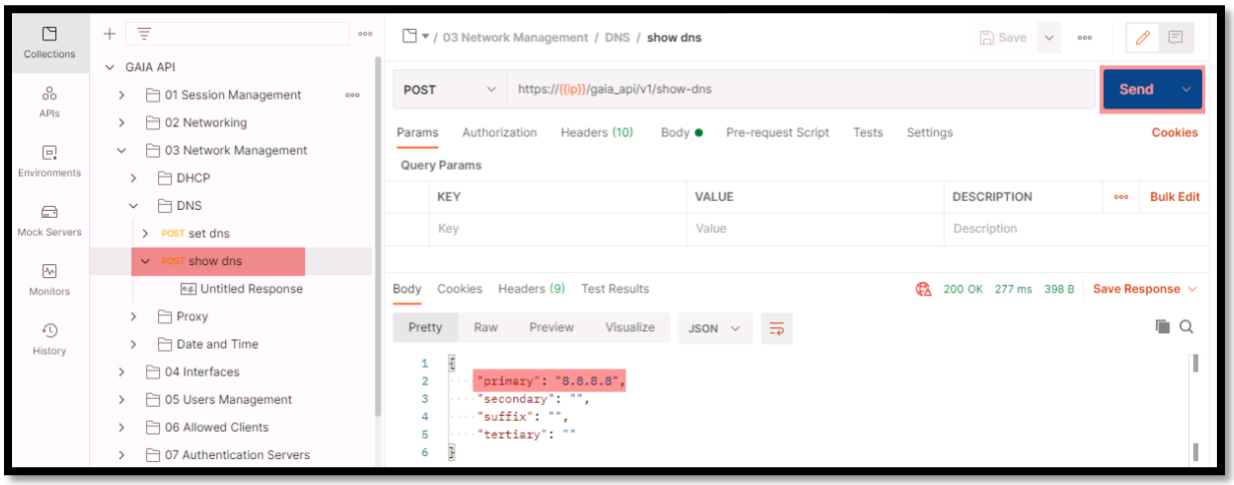
12. Run the login command by clicking Send and review the results.




13. Under `Environments`, select the `Gaia_API` environment we created in the previous steps and notice that we now have the session ID `sid` saved in the environment.

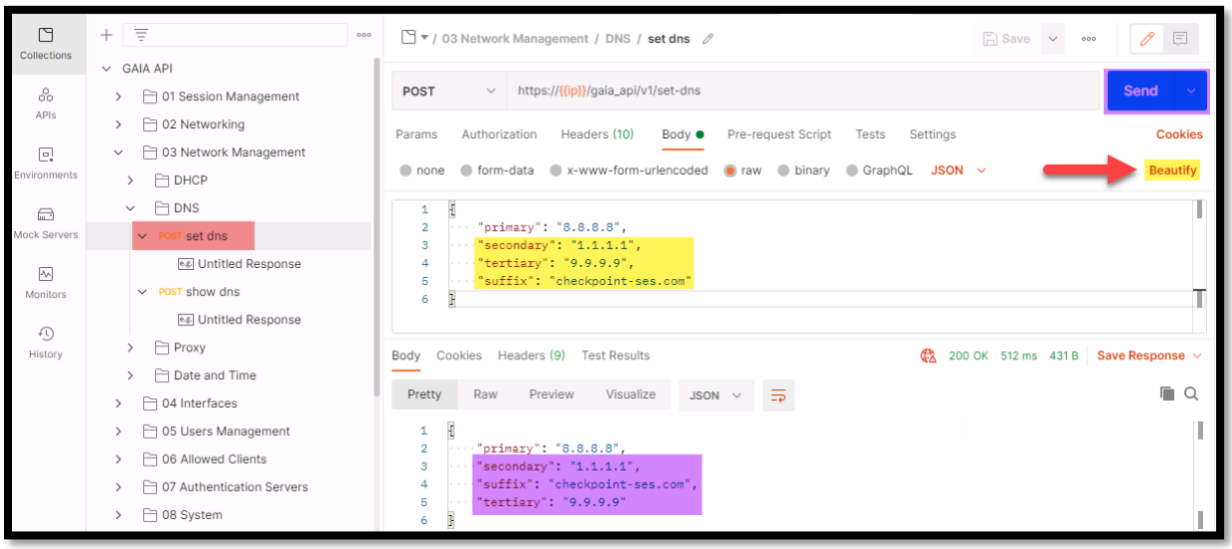



14. Now that we are logged in and have a `Session ID`, we should be able to use any API requests. Run the API request to show the existing DNS server.



15. Use the saved call `set DNS` to add a second and third DNS server. Use the `Beautify` feature below to show the request in proper JSON format.




16. To Practice, and test multiple API requests to get familiar with the required fields, header, and body for different requests.

## Exercise 5: Using the Check Point Python SDK

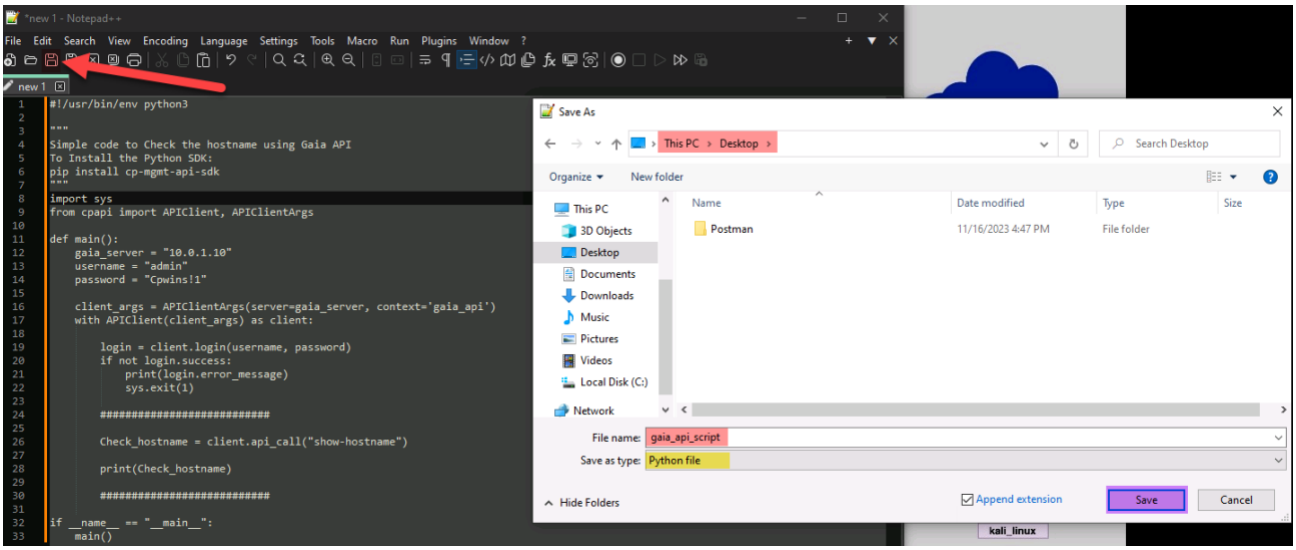
This exercise will use the Check Point Python SDK to write a simple script to run the Gaia API command and manage our Gaia servers. The SDK and its documentation are available on GitHub `https://github.com/CheckPointSW/cp_mgmt_api_python_sdk`.

1. Open `Notepad++` and paste the code template below to the file or type it if you prefer or download it from `https://github.com/alshawwaf/Gaia_API_Workshop`

```
#!/usr/bin/env python3

"""
Simple code to Check the hostname using Gaia API
To Install the Python SDK:
pip install cp-mgmt-api-sdk
"""
import sys
from cpapi import APIClient, APIClientArgs

def main():
    gaia_server = "10.0.1.10"
```

```python
    username = "admin"
    password = "Cpwins!1"

    client_args = APIClientArgs(server=gaia_server, context='gaia_api')
    with APIClient(client_args) as client:

        login = client.login(username, password)
        if not login.success:
            print(login.error_message)
            sys.exit(1)

        ###########################

        Check_hostname = client.api_call("show-hostname")

        print(Check_hostname)

        ###########################

if __name__ == "__main__":
    main()
```

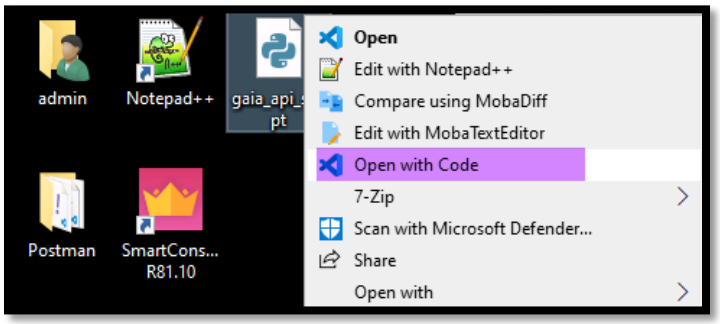
2. Save the file to your `Desktop` and make sure the file type is set to `Python`.



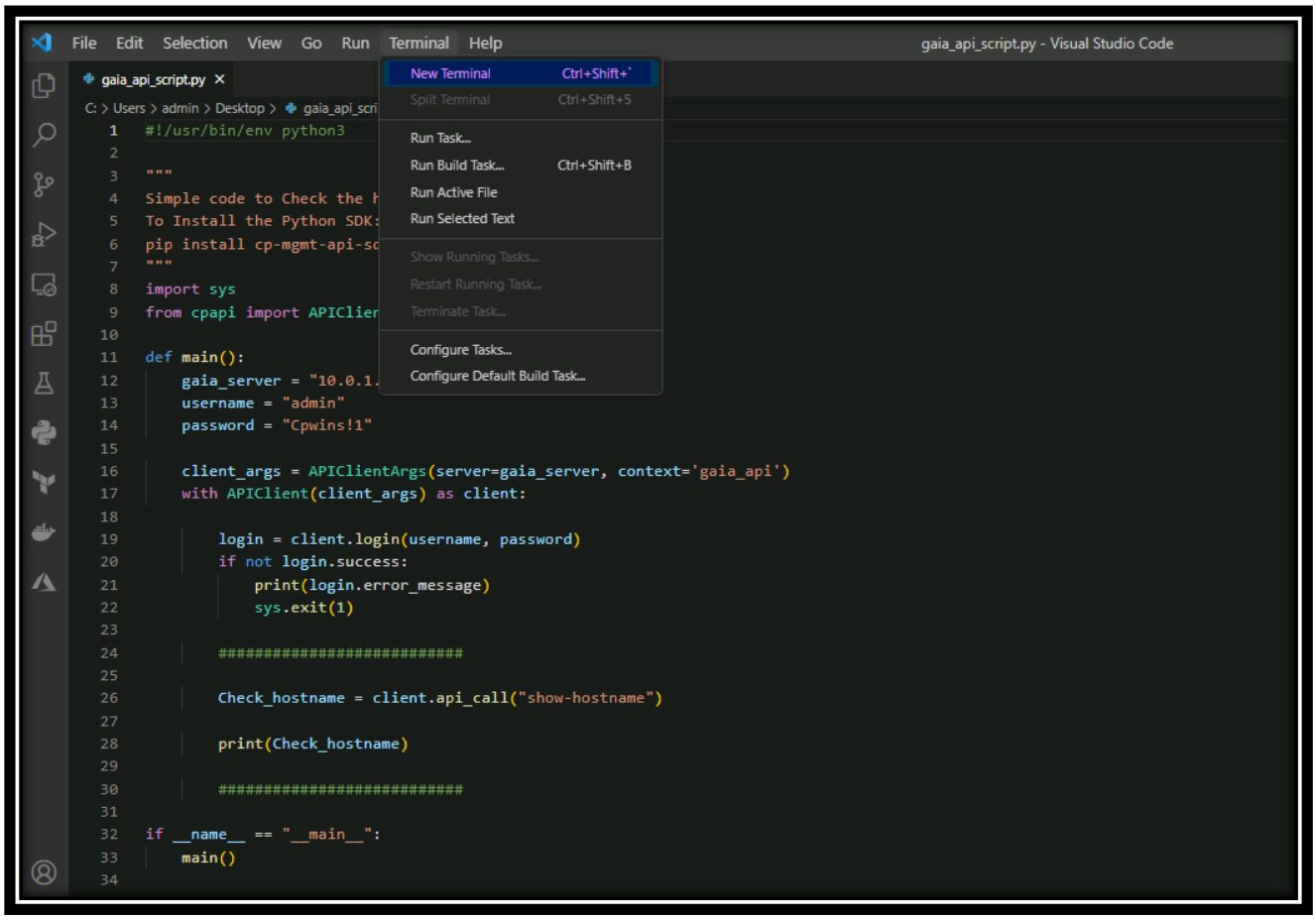

Notice that Notepad++ is now showing the python script in a proper format as you can see below.

```python
gaia_api_script.py
#!/usr/bin/env python3

"""
Simple code to Check the hostname using Gaia API
To Install the Python SDK:
pip install cp-mgmt-api-sdk
"""
import sys
from cpapi import APIClient, APIClientArgs

def main():
    gaia_server = "10.0.1.10"
    username = "admin"
    password = "Cpwins!1"

    client_args = APIClientArgs(server=gaia_server, context='gaia_api')
    with APIClient(client_args) as client:

        login = client.login(username, password)
        if not login.success:
            print(login.error_message)
            sys.exit(1)

        ##########################

        Check_hostname = client.api_call("show-hostname")

        print(Check_hostname)

        ##########################

if __name__ == "__main__":
    main()
```

3. It is preferred to use a more advanced IDE such as Visual Studio Code. Close your `Notepad++`. Right-click on the file from the Desktop and select `Open with Code` to open the file using Visual Studio Code.

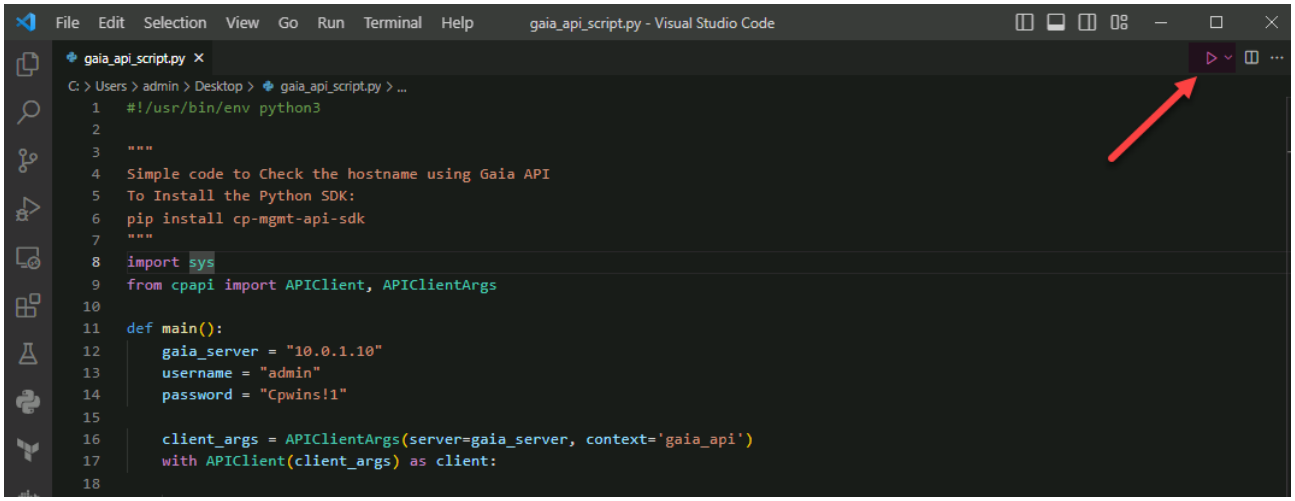



4. Visual Studio Code provides the ability to run scripts using integrated terminals. From the View tab, select `Terminal -> New Terminal` to open the default terminal or use `Ctrl+`.`




5. Use the terminal to install the `Check Point Python SDK` using the command below: `pip install cp-mgmt-api-sdk`


6. Run the scrip as shown below. You can also start the script using the command: `python.exe c:/Users/admin/Desktop/gaia_api_script.py`




7. Approve the server's fingerprint as this is the first time we are connecting to this Gateway.

```
$ C:/Users/admin/.virtualenvs/admin--vUoXxCa/Scripts/python.exe c:/Users/admin/Desktop/gaia_api_script.py
You currently do not have a record of this server's fingerprint.
Server's fingerprint: 2BC52799CF868CBCFD2E59B5903622BB544F496B
Do you accept this fingerprint? [y/n] y
```

8. Review the API response below and notice that we got a full response with more details that we need.

```
APIResponse({
    "data": {
        "name": "gw_r8120"
    },
    "res_obj": {
        "data": {
            "name": "gw_r8120"
        },
        "status_code": 200
    },
    "status_code": 200,
    "success": true
})
```

9. Change the check_hostname call to add `.data` at the end `Check_hostname = client.api_call("show-hostname").data` This will only show fields inside the data object.

```
26        Check_hostname = client.api_call("show-hostname").data

27

28        print(Check_hostname)

29

30        ###########################

31

32

33    if __name__ == "__main__":
34        main()
```

```
OUTPUT    DEBUG CONSOLE    TERMINAL    AZURE

            "name": "gw_r8120"
        },
        "status_code": 200
    },
    "status_code": 200,
    "success": true
})
(admin)
admin@Admin-PC MINGW64 ~
$ C:/Users/admin/.virtualenvs/admin--vUoXxCa/Scripts/python.exe c:/Users/admin/Desktop/gaia_api_script.py
{'name': 'gw_r8120'}
```

10. We can also filter using the key of the `JSON` reply to show only the value.

```
26        Check_hostname = client.api_call("show-hostname").data["name"]

27

28        print(Check_hostname)

29

30        ###########################

31
```

```
OUTPUT    DEBUG CONSOLE    TERMINAL    AZURE

    },
    "status_code": 200,
    "success": true
})
(admin)
admin@Admin-PC MINGW64 ~
$ C:/Users/admin/.virtualenvs/admin--vUoXxCa/Scripts/python.exe c:/Users/admin/Desktop/gaia_api_script.py
{'name': 'gw_r8120'}
(admin)
admin@Admin-PC MINGW64 ~
$ C:/Users/admin/.virtualenvs/admin--vUoXxCa/Scripts/python.exe c:/Users/admin/Desktop/gaia_api_script.py
gw_r8120
```

11. Change your script to edit the interface `eth0` and add a comment to this interface. Consult with the Gaia API documentation to find the required fields.

```
edit_interface = client.api_call("set-physical-interface",payload={"enabled": "true","name":
"eth0","comments": "Added via Python Script",})
```

Notice that we are providing all the required fields as payload.

```
26        edit_interface = client.api_call(
27            "set-physical-interface",
28            payload={
29                "enabled": "true",
30                "name": "eth0",
31                "comments": "Added via Python Script",
32            },
33        )
34
35        print(edit_interface)
36
37        ###########################
38
```

OUTPUT DEBUG CONSOLE TERMINAL AZURE

```
$ C:/Users/admin/.virtualenvs/admin--vUoXxCa/Scripts/python.exe c:/Users/admin/Desktop/gaia_api_script.py
APIResponse({
    "data": {
        "auto-negotiation": true,
        "comments": "{Added via Python Script}",
        "dhcp": {
            "enabled": false
        },
```

12. Review the interface settings via the Gaia Portal and make sure you can see the comment.

Interfaces

Add ▾ Edit Delete Refresh

| Name | Type | IPv4 Address | Subnet Mask | IPv6 Address | IPv6 Mask Length | Link Status | SD-WAN | Comment |
|---|---|---|---|---|---|---|---|---|
| eth0 | Ethernet | 203.0.113.10 | 255.255.255.0 | - | - | Up | — | Added via Python Script |
| eth1 | Ethernet | 10.0.1.10 | 255.255.255.0 | - | - | Up | — | |
| eth2 | Ethernet | 10.0.2.10 | 255.255.255.0 | - | - | Up | — | |
| eth3 | Ethernet | 10.0.3.10 | 255.255.255.0 | - | - | Up | — | |
| lo | Loopback | 127.0.0.1 | 255.0.0.0 | - | - | Up | — | |
| loop00 | Loopback | 10.0.0.1 | 255.255.255.255 | - | - | Up | — | |

13. Exit the `Visual Studio Code`.

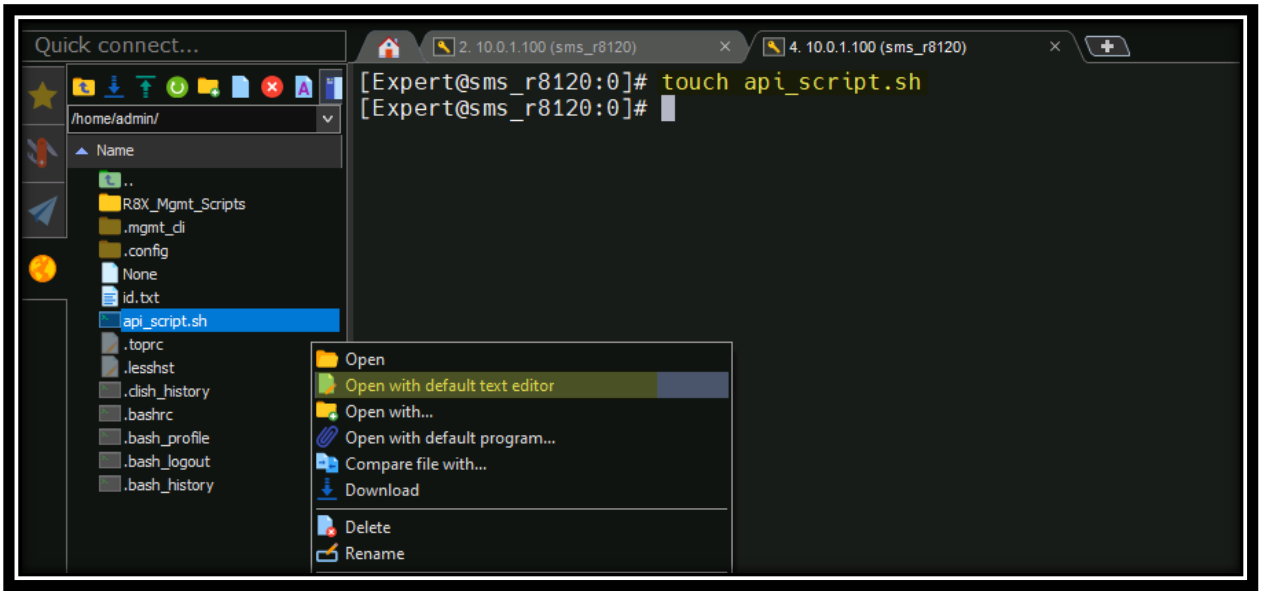
## Exercise 6: Writing bash script with mgmt_cli

In this exercise, we will write simple bash scripts to utilize the `mgmt_cli` and perform tasks against the Gaia servers.

1. Connect to the CLI of the management server using `MobaXterm` and create an empty file.

```
[Expert@Mgmt:0]# touch api_script.sh
[Expert@Mgmt:0]#
```

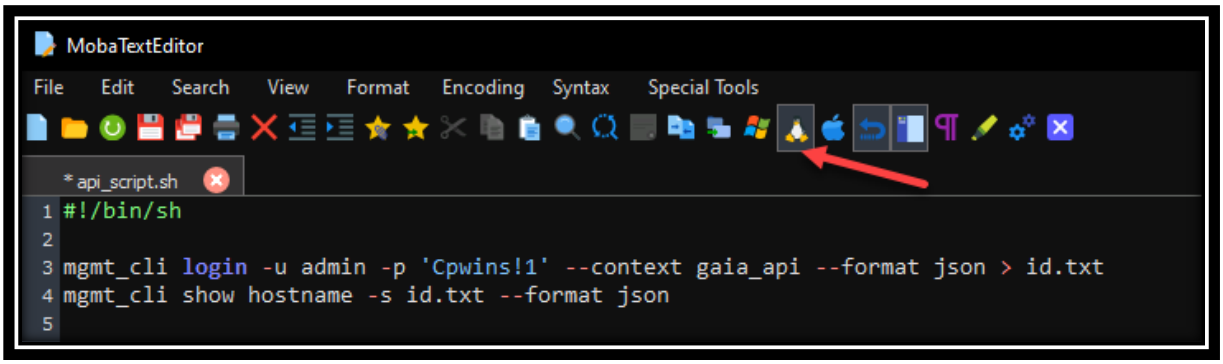
2. Use any text editor to edit the script file such as `vi`. Alternatively, you can open it using the default text editor of `MobaXterm`




3. Copy the script lines below or type them if you prefer in the file we just created.

```
#!/bin/sh

mgmt_cli login -u admin -p ‘Cpwins!1’ --context gaia_api --format json > id.txt
mgmt_cli show hostname -s id.txt --format json
```

4. Note that if you use the built-in editor, you must select `UNIX format` as shown below.



5. Run your script from the command line using the command `bash api_script.sh`

```
[Expert@sms_r8120:0]# bash api_script.sh
{
  "name": "sms_r8120"
}
```

6. Open the file using `Visual Studio Code` and change the script to perform a call to get the version details of the Gaia server.

```
MobaTextEditor
File  Edit  Search  View  Format  Encoding  Syntax  Special Tools
api_script.sh
1 #!/bin/sh
2
3 mgmt_cli login -u admin -p 'Cpwins!1' --context gaia_api --format json > id.txt
4 mgmt_cli show-version -s id.txt --format json
5
6
```

7. Run the script and review the output details.

```
[Expert@sms_r8120:0]# bash api_script.sh
{
  "os-build": "627",
  "os-edition": "64-bit",
  "os-kernel-version": "3.10.0-1160.15.2cpx86_64",
  "product-version": "Check Point Gaia R81.20"
}
```

8. Check Point Gaia has the JQ tool installed on all supported versions `R8x.xx`. This tool can help us filter the JSON response and have the exact field. Change the script and use `jq` to filter the response to only show the `product-version`

```
MobaTextEditor
File  Edit  Search  View  Format  Encoding  Syntax  Special Tools
api_script.sh
1 #!/bin/sh
2
3 mgmt_cli login -u admin -p 'Cpwins!1' --context gaia_api --format json > id.txt
4 mgmt_cli show-version -s id.txt --format json | jq '."product-version"'
5
```

9. Run the script and notice that only the product version was returned.

```
[Expert@sms_r8120:0]# bash api_script.sh
"Check Point Gaia R81.20"
```

10. Use the flag “-r” with JQ to get rid of the quotes in the response.

```
MobaTextEditor
File  Edit  Search  View  Format  Encoding  Syntax  Special Tools
* api_script.sh
1 #!/bin/sh
2
3 mgmt_cli login -u admin -p 'Cpwins!1' --context gaia_api --format json > id.txt
4 mgmt_cli show-version -s id.txt --format json | jq -r '."product-version"'
5
6
```

```
[Expert@sms_r8120:0]# bash api_script.sh
Check Point Gaia R81.20
```

## Exercise 6: Gaia API via Management (Proxy)

In the previous exercise, we used `mgmt_cli` to access the Gaia API locally. The management Web API also uses mgmt_cli. This exercise will teach us how to use the `management API as a proxy to run GAIA API commands` against any target gateway.

1. From the cli of the management station, run the following command and notice the error message.

```
mgmt_cli -u api_admin -p 'Cpwins!1' gaia-api/show-hostname target 10.0.1.10 -f json
```

```
[Expert@sms_r8120:0]# mgmt_cli -u api_admin -p 'Cpwins!1' gaia-api/show-hostname target 10.0.1.10 -f json
code: "err_login_failed"
message: "Authentication to server failed."
```

We are using the Management API as a proxy to issue Gaia API command to the managed Gateways. Hence, we must login using a user with access to the management API.

2. Change the credentials and use the default admin of the Management server and run the command:

**Username:** `admin`
**Password:** `Cpwins!1`

```
mgmt_cli -u admin -p 'Cpwins!1' gaia-api/show-hostname target 10.0.1.10 -f json
```


- The credentials provided we used is the management administrator (not the `api_admin` we created for the Gaia API).
- Instead of using the parameter `--context` we used the format `gaia-api/<command>.`
- Using `--context`, indicates to the `mgmt_cli` that we are trying to use Gaia API.
- You can use `-m <address>` with mgmt_cli to log in to a remote management server.
- We provided a required argument called `target` to specify the target gateway on which the API command will be executed.
- More details in the Management API reference:
  `https://sc1.checkpoint.com/documents/latest/APIs/index.html#cli/gaia-api~v1.9%20`

Thank You