JumpCloud Password Hashing Application Assignment

Background Test Plan Choice:

This Test Suite outlines the Major and Critical Test Cases, that is used to implement and confirm the user acceptance testing as per the Acceptance Criteria, provided by the Business/Product. They should be used as the 'Checklist' during Product-Hand Over from the Development team to the Business/Product, usually done during End of Sprint Demos.

All the Acceptance Criteria are to be tested, for both Positive and Negative outcomes, to ensure and emphasize the SDLC Best Practices, hence a cutting edge, top notch and next-gen product in the market.

For the functional testing, the test cases cover 'conflict-testing', where the application is tested on the capability to process requests that are functionally not required. For instance, sending a GET request to a POST endpoint. The test cases also test the ability to detect invalid endpoints and interact with the users accordingly.

Also, covered in the testcases, is the ability to restrict and adhere to the specified connection type, as per the requirements. In this case, the application is required to use http, so, testing the behavior when a connection is attempted from https, is necessary.

The test cases are written in plain English, in order to make sure the non-technical (Business/Product) team members are able to understand the testing scope for the application. This will enable a thorough grooming/refinement for the test cases, to ensure the end-users' expectations are met/covered.

Since, I do not have a thoroughly deep and complete understanding of the application, the test cases are derived from the provided Acceptance Criteria, along with the general and deep knowledge of the SDLC best practices. My assumption is that the development code coverage meets the company's code coverage threshold.

Overall, the test plan flow includes the following columns;

1. TestCase_ID – The unique identifier for each test case. This is used for mapping purposes, across the development management tools.
2. Test Scenario – This outlines the scenario at test. It is written in plain English for easier understanding amongst the scrum team members and even to the management and/or client's level
3. Test Steps – These are steps following in order to test the corresponding test case/scenario
4. Test Data – These are data used/implemented in the test steps. They are handy for faster testing and also reproduction purposes, in case of any failures.
5. Expected Results – These are the outlined expected results for the corresponding test cases/scenarios, as per the requirements or functional expectations

6. Actual Results – These are the actual outcome for the corresponding test cases, scenarios and steps. It determines whether the expectations are met/achieved or not. If the expectation is met/achieved, it is a PASS, otherwise, it is a FAIL.
7. Notes – This column is used to note down any concerns, recommendations, questions, clarifications etc for the corresponding test case, scenario, steps or test data.

Bugs:

To avoid duplication of efforts, depending on the project management, defect management and also Testcase management tools used, we can integrate the three, so that, whenever there is any failure in the test runs, it creates a defect in the defect board. This does not only save time, but also, increases the visibility within the team.

Looking at the test plan, it has TestCase_ID, which is used as the primary key (identifier), when creating the defects. Also, in the PASS/FAIL column, there is a filter functionality that can be used to filter only failed test cases.

Therefore, instead of creating another list of bugs here, I prefer using the Test Plan document, which contains the steps and test data for reproduction purposes.

Recommendations:

1. **Avoid Predictability:**
   The /hash endpoint seems to be hashing the given passwords and returning the response in accordance to the record number (job identifier), which is predictably incremental. For the application's and users' data safety, I recommend the use of a random record number. This can easily be done by either generating a random alphanumerical string or implementing the use of timestamp to generate this key. Converting the timestamp to microseconds would be great as it will ensure not duplicate possibility and also longer record number.
2. **Password Policy Implementation**:
   The application does not implement any of the best password policy standards.

a. It accepts any characters as a password, including empty strings. Restricting the character count and combination would ensure any hacker is highly stopped.
b. The password length is not restricted. It accepts and hashes passwords of any length of characters, including empty strings.
c. Implement wildcards and spaces for stronger and longer passwords. This will not only ensure longer passwords implementation, but also allow users to apply easy to remember but passwords. For example; you could let users use '1@m g0!ng 2b gr8 1n th1$', translating to 'I am going to be great in this'. Such combinations are pretty tough for hackers to decode/crack.
d. Implement a password guidance/graphical console for users to understand what exactly is required of them. Remember, not every user understands what a Strong or Weak password means.

e. Restrict the use of predictable passwords; Password, Mylogin, Letmein, pass123 are just but a few of the easily crack-able passwords, that should not be allowed.

Depending on the mission/vision of the vendor and clients, the password policies would vary, to ensure the respective requirements are achieved. Above, are just but a few examples of the standard Password Policies, recommended by the Identity IQ best practices.

3. **Data Protection/Security:**
   a. The /hash/{recordNumber} endpoint, that seems to retrieve the hashed passwords, is not secured. There needs to an implementation for auth, as part of the header, for the request. Currently, it seems any user who has a record number is able to easily retrieve the hashed password, hence risking the users' data.
   b. Since the application is dealing with very sensitive users' data, I would recommend the implementation of auth, for all requests that read, write and modify these data.
   c. Implement MFA. The Multi-Factor Authentication would add another cushion of the data accessibility, hence ensuring the right access to the right user.
   d. Implement IP address whitelisting for the admin users, who have the privileges of manipulating the data at rest (in database).

4. **Restriction of Attempts:**
   a. Implement a mechanism that locks the users out after a specified number of attempts to retrieve the hashed password. This will ensure data security against exhortation method of hacking is minimized.
   b. Restrict and Time the number and periods for 'Replays'. Implement a mechanism where, hashing/retrieval is restricted and limited to a specific time period. This will minimize the possibility of the 'Replay' attack to the system.

5. **Over-shoulder Attack Prevention:**
   a. There is need to ensure the password is covered/white-texted while typing in. This would ensure that anyone tying to look over the user's shoulder does not get the real password, easily. Currently, the application allows the user to type in the password and prints it out as it is entered.

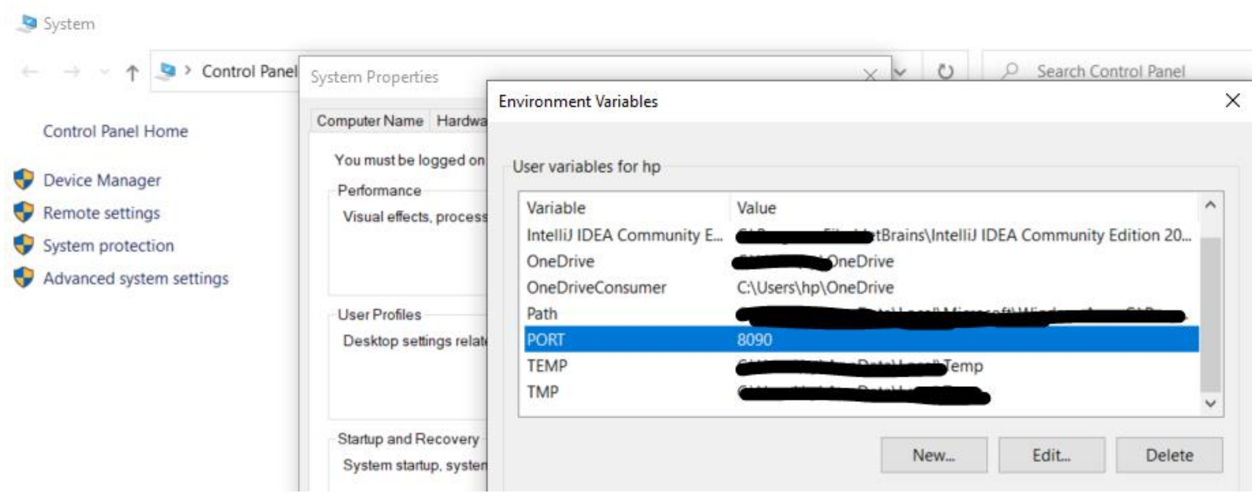
6. **Application Integration:**
   a. Ensure the application is able to integrate with newer technologies, such as fingerprints, facial recognition, voice recognition, inbuilt and third-party applications (eg: siri/alexa), used by users to access respective applications.

Screenshots From Testing:

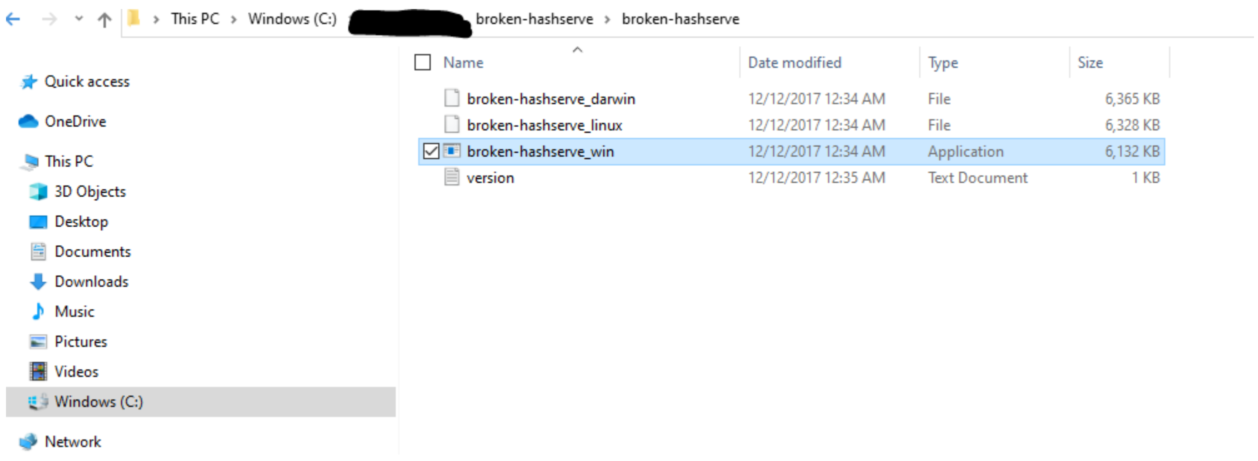
For easier understanding of the test results, screenshots play a major role in relating the PASSes and FAILs to the test cases/scenarios.

The screenshots are named in accordance to the TestCase_ID.

TC1.a

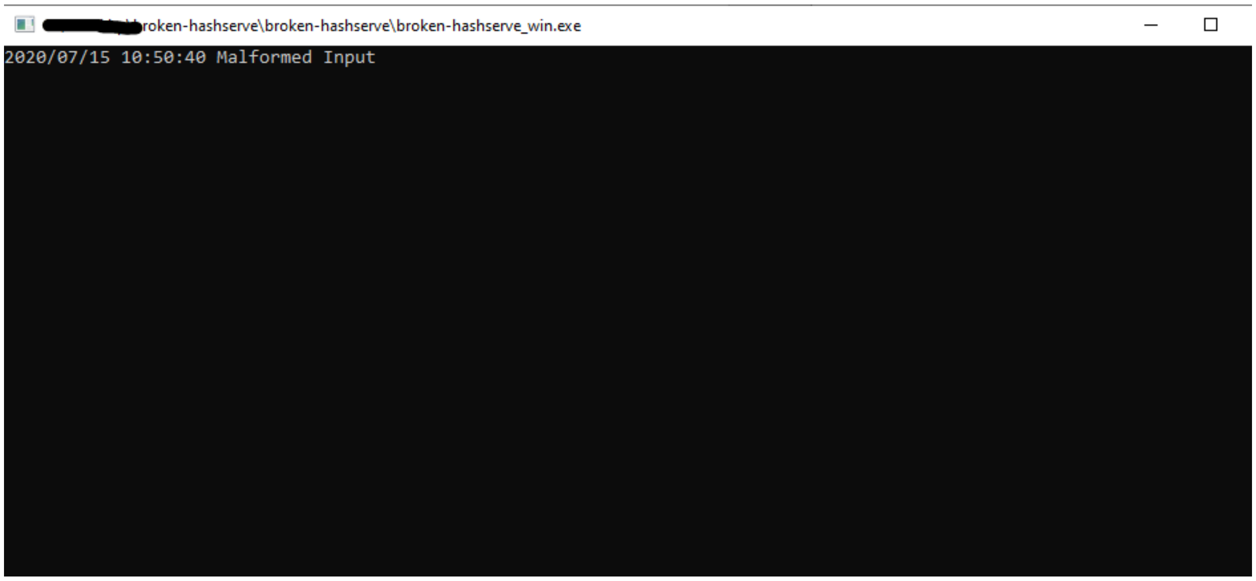


Variable	Value
IntelliJ IDEA Community E...	[redacted]tBrains\IntelliJ IDEA Community Edition 20...
OneDrive	[redacted]OneDrive
OneDriveConsumer	C:\Users\hp\OneDrive
Path	[redacted]
PORT	8090
TEMP	[redacted]Temp
TMP	[redacted]



Name	Date modified	Type	Size
broken-hashserve_darwin	12/12/2017 12:34 AM	File	6,365 KB
broken-hashserve_linux	12/12/2017 12:34 AM	File	6,328 KB
broken-hashserve_win	12/12/2017 12:34 AM	Application	6,132 KB
version	12/12/2017 12:35 AM	Text Document	1 KB

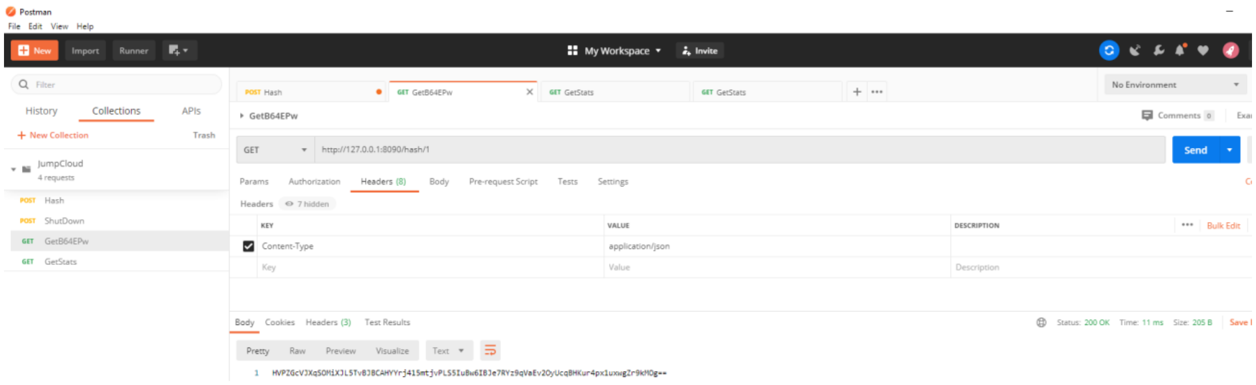
TC1.a/TC1.b



[redacted]roken-hashserve\broken-hashserve\broken-hashserve_win.exe

```
2020/07/15 10:50:40 Malformed Input
```

TC1.c



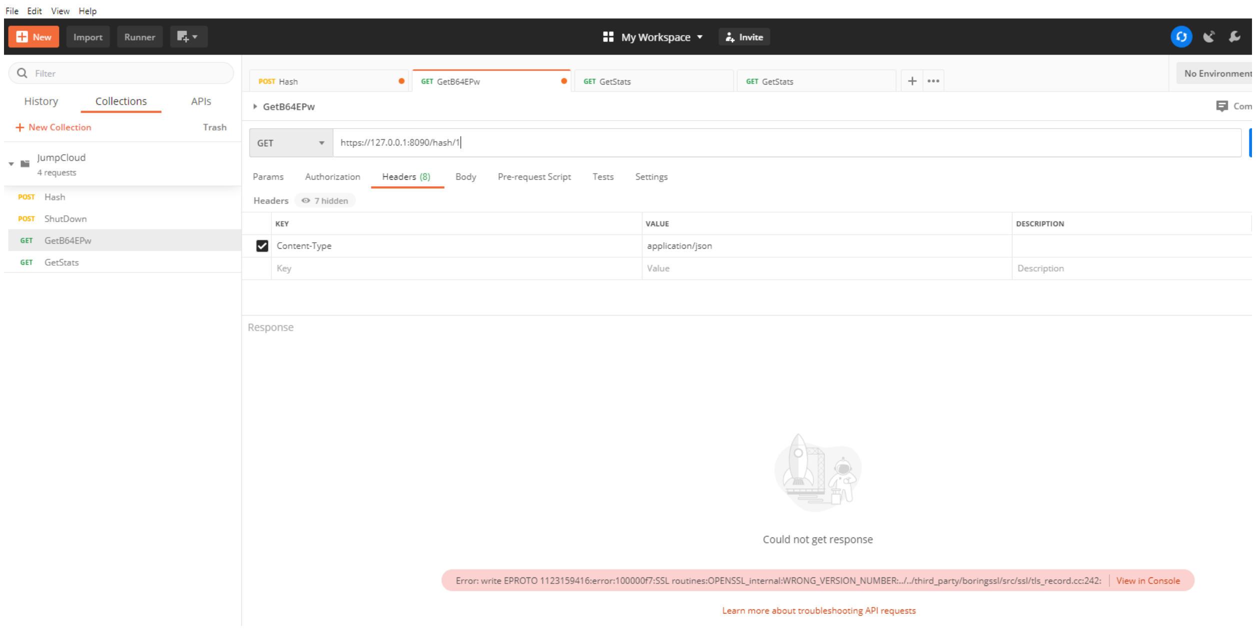
GET http://127.0.0.1:8090/hash/1

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	

Status: 200 OK Time: 11 ms Size: 205 B

```
HVPZGcVJXqSOMiXJL5TvBJBCAHYYrj415mtjvPLS5IuBw6IBJe7RYz9qVaEv2OyUcqBHKur4px1uxwgZr9kMOg==
```

TC1.d



GetB64EPw

GET https://127.0.0.1:8090/hash/1

Headers (8)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

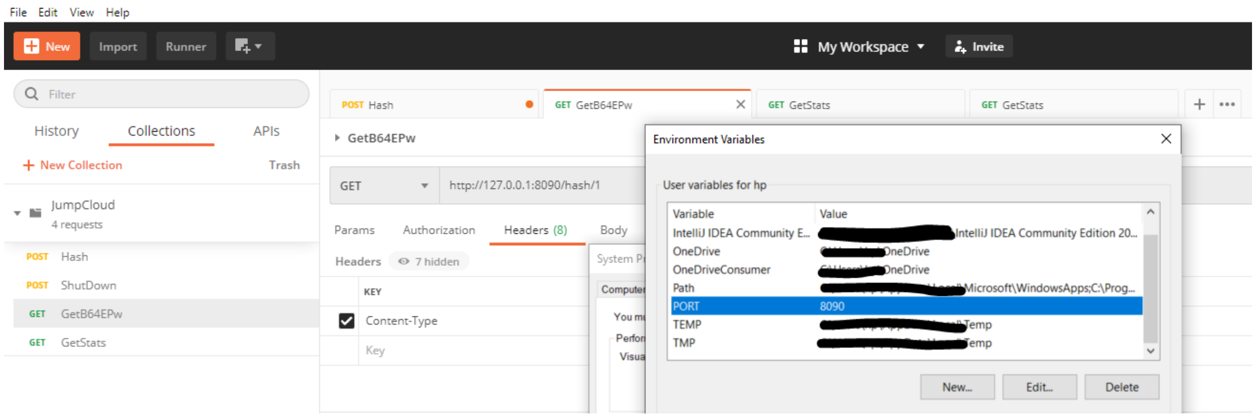
Response

Could not get response

```
Error: write EPROTO 1123159416:error:100000f7:SSL routines:OPENSSL_internal:WRONG_VERSION_NUMBER:../../third_party/boringssl/src/ssl/tls_record.cc:242:
```

Learn more about troubleshooting API requests

TC2.a



GetB64EPw

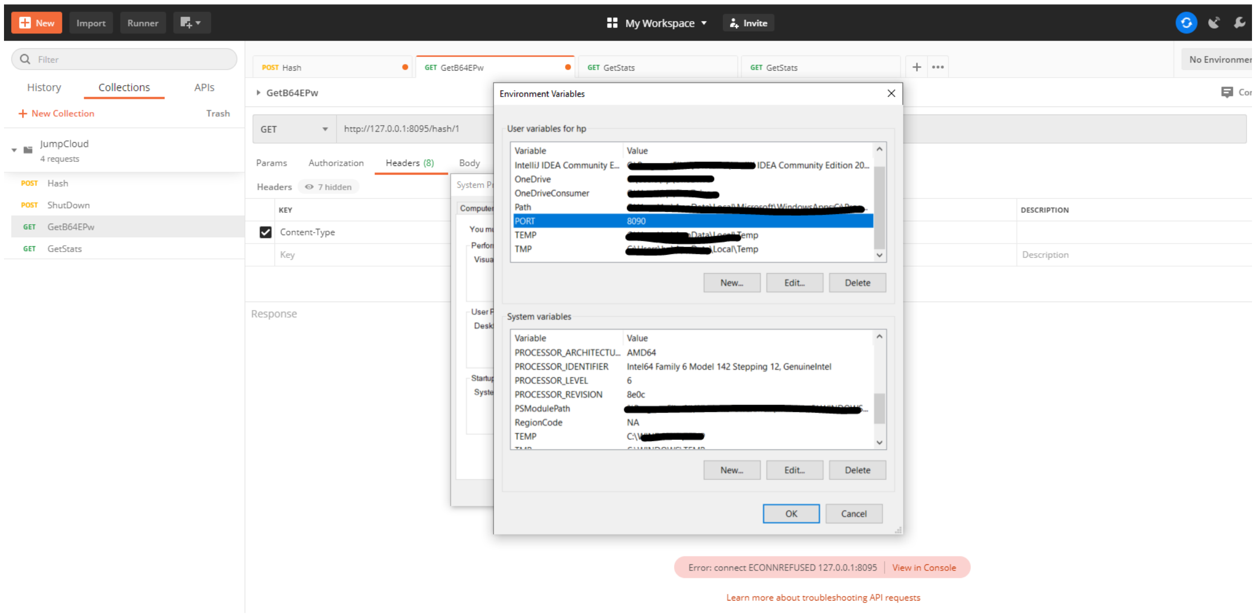
GET http://127.0.0.1:8090/hash/1

Environment Variables

User variables for hp

Variable	Value
IntelliJ IDEA Community E...	[redacted]IntelliJ IDEA Community Edition 20...
OneDrive	[redacted]OneDrive
OneDriveConsumer	[redacted]OneDrive
Path	[redacted]Microsoft\WindowsApps;C:\Prog...
PORT	8090
TEMP	[redacted]Temp
TMP	[redacted]Temp

TC2.b


GET GetB64EPw — http://127.0.0.1:8095/hash/1

Environment Variables

User variables for hp

Variable	Value
IntelliJ IDEA Community E...	[redacted] IDEA Community Edition 20...
OneDrive	[redacted]
OneDriveConsumer	[redacted]
Path	[redacted]
PORT	8090
TEMP	[redacted] Temp
TMP	[redacted] \Local\Temp

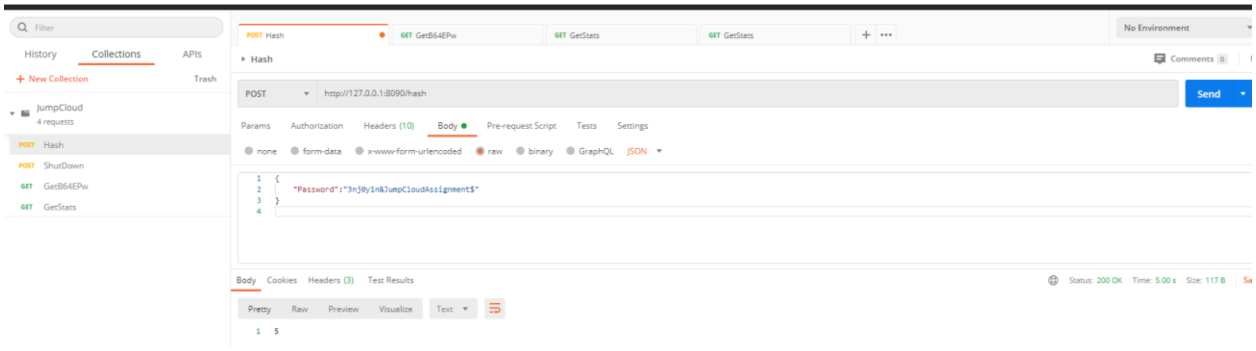
System variables

Variable	Value
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e0c
PSModulePath	[redacted]
RegionCode	NA
TEMP	C:\ [redacted]

Error: connect ECONNREFUSED 127.0.0.1:8095 | View in Console

Learn more about troubleshooting API requests

TC3.a


POST Hash — http://127.0.0.1:8090/hash

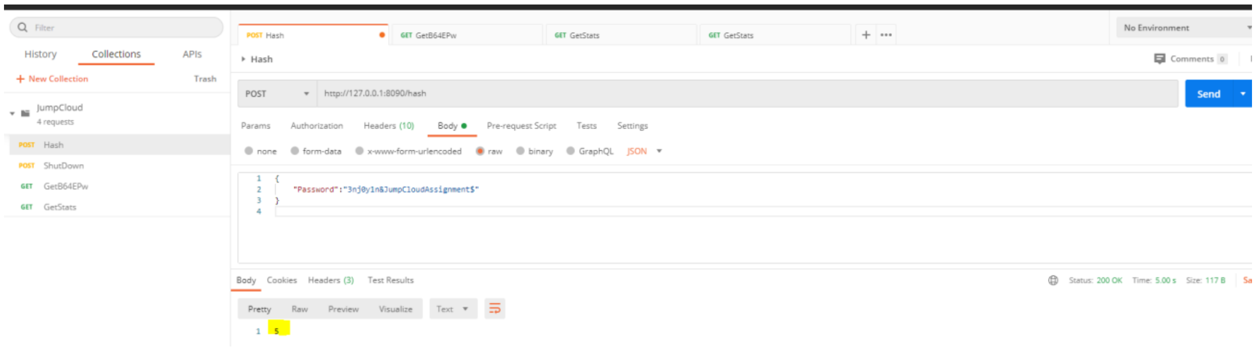
Body: raw, JSON

```
{
    "Password":"3nj0y1n&JumpCloudAssignment$"
}
```

Status: 200 OK Time: 5.00 s Size: 117 B

```
5
```

TC3.b


POST Hash — http://127.0.0.1:8090/hash

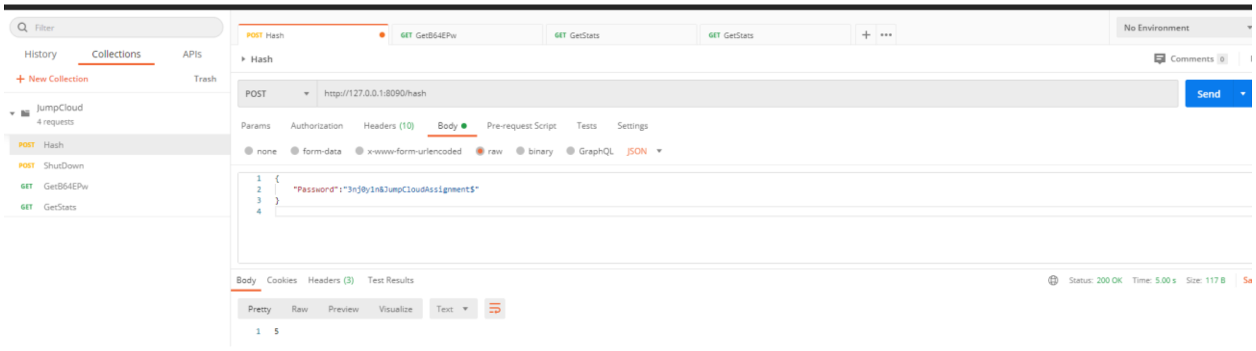
Body: raw, JSON

```
{
    "Password":"3nj0y1n&JumpCloudAssignment$"
}
```

Status: 200 OK Time: 5.00 s Size: 117 B

```
5
```

TC3.c



POST http://127.0.0.1:8090/hash

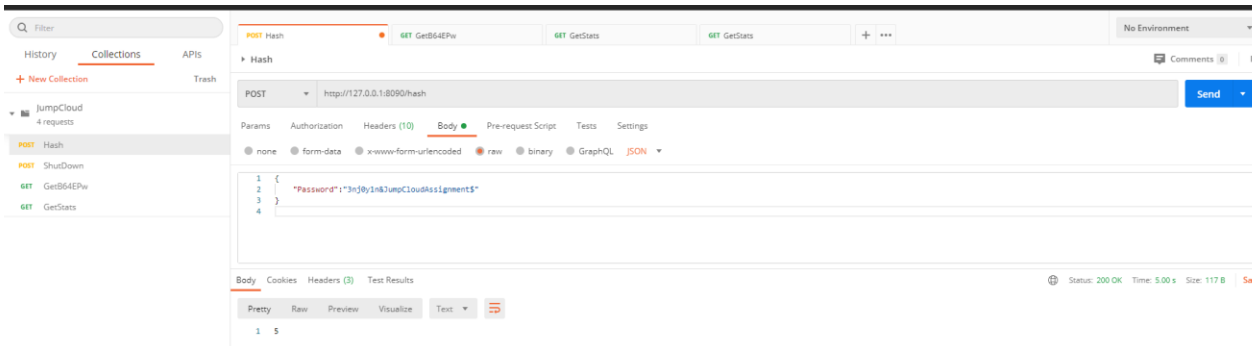
Body: raw JSON

```
{
    "Password":"3nj0y1n&JumpCloudAssignment$"
}
```

Status: 200 OK Time: 5.00 s Size: 117 B

```
5
```

TC3.d



POST http://127.0.0.1:8090/hash

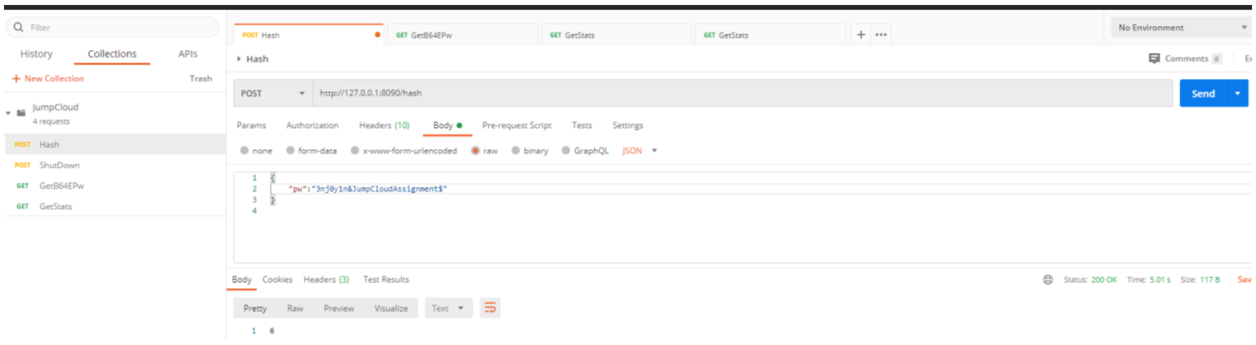
Body: raw JSON

```
{
    "Password":"3nj0y1n&JumpCloudAssignment$"
}
```

Status: 200 OK Time: 5.00 s Size: 117 B

```
5
```

TC3.e



POST http://127.0.0.1:8090/hash

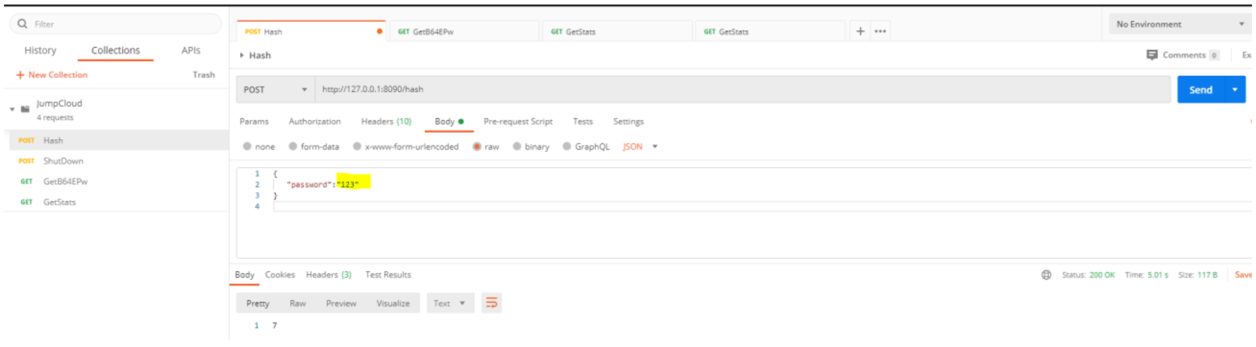
Body: raw JSON

```
{
    "pw":"3nj0y1n&JumpCloudAssignment$"
}
```

Status: 200 OK Time: 5.01 s Size: 117 B

```
6
```

TC3.f



POST http://127.0.0.1:8090/hash

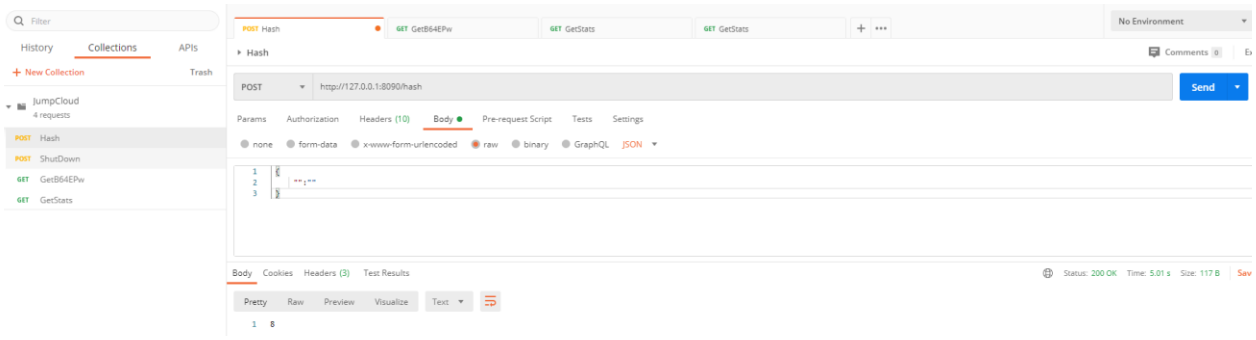
Body: raw JSON

```
{
"password":"123"
}
```

Status: 200 OK Time: 5.01 s Size: 117 B

```
7
```

TC3.g



POST http://127.0.0.1:8090/hash

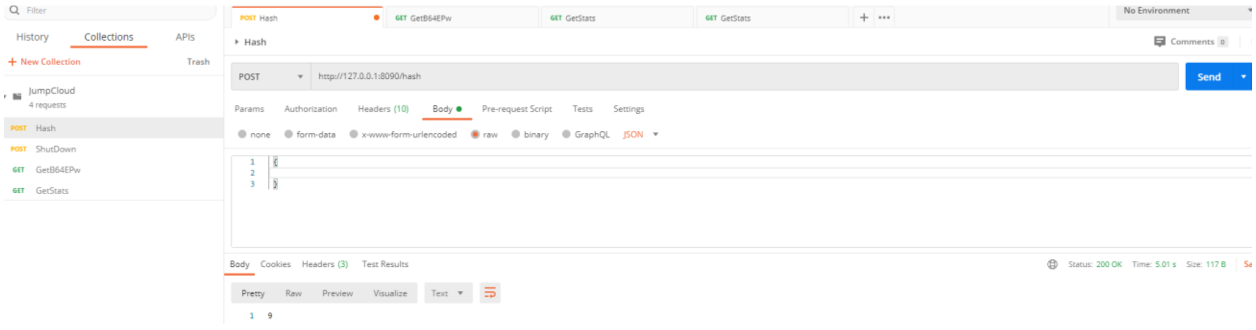
Body: raw JSON

```
{
    "":""
}
```

Status: 200 OK Time: 5.01 s Size: 117 B

```
8
```

TC3.h



POST http://127.0.0.1:8090/hash

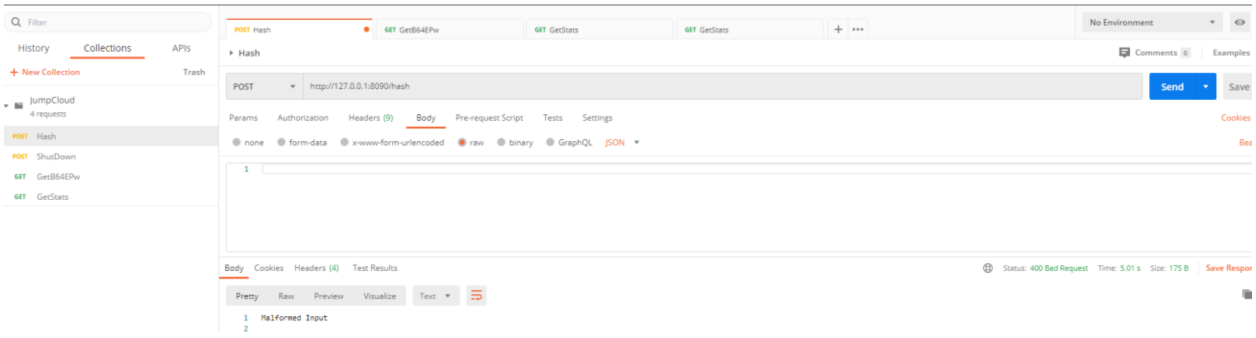
Body: raw JSON

```
{

}
```

Status: 200 OK Time: 5.01 s Size: 117 B

```
9
```

TC3.i



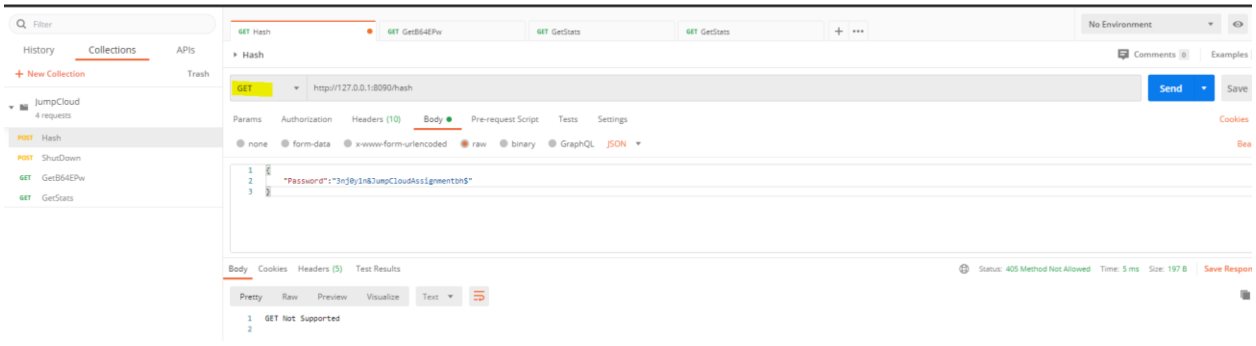
POST http://127.0.0.1:8090/hash

Body: raw JSON

Status: 400 Bad Request Time: 5.01 s Size: 175 B

```
Malformed Input
```

TC3.j


GET http://127.0.0.1:8090/hash

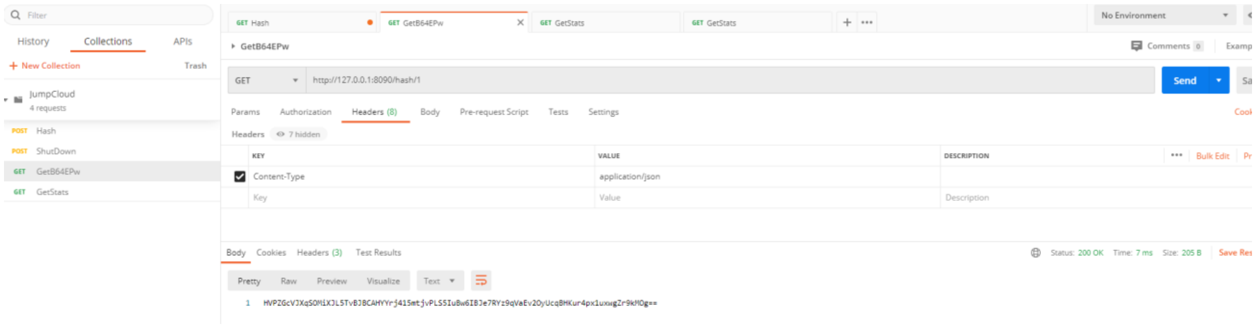
Body (raw, JSON):

```
{
    "Password":"3nj0y1n&JumpCloudAssignmentbh$"
}
```

Status: 405 Method Not Allowed Time: 5 ms Size: 197 B

```
GET Not Supported
```

TC4.a

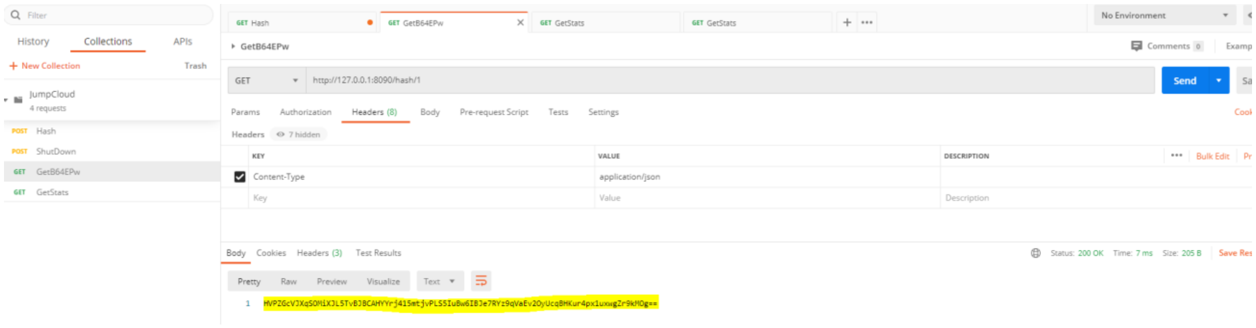

GET http://127.0.0.1:8090/hash/1

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	

Status: 200 OK Time: 7 ms Size: 205 B

```
HVPZGcVJXqSOMiXJL5TvBJBCAHYYrj415mtjvPLS5IuBw6IBJe7RYz9qVaEv2OyUcqBHKur4px1uxwgZr9kMOg==
```

TC4.b

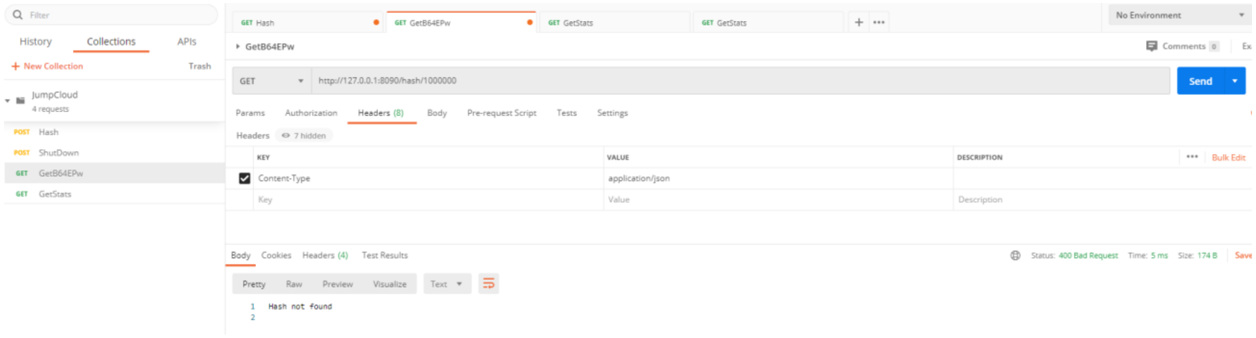

GET http://127.0.0.1:8090/hash/1

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	

Status: 200 OK Time: 7 ms Size: 205 B

```
HVPZGcVJXqSOMiXJL5TvBJBCAHYYrj415mtjvPLS5IuBw6IBJe7RYz9qVaEv2OyUcqBHKur4px1uxwgZr9kMOg==
```

TC4.c

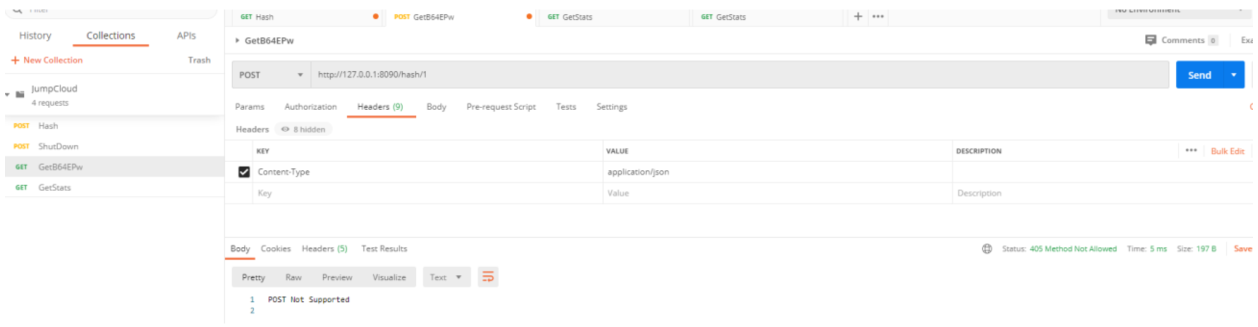

GET http://127.0.0.1:8090/hash/1000000

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	

Status: 400 Bad Request Time: 5 ms Size: 174 B

```
Hash not found
```

TC4.d



POST http://127.0.0.1:8090/hash/1

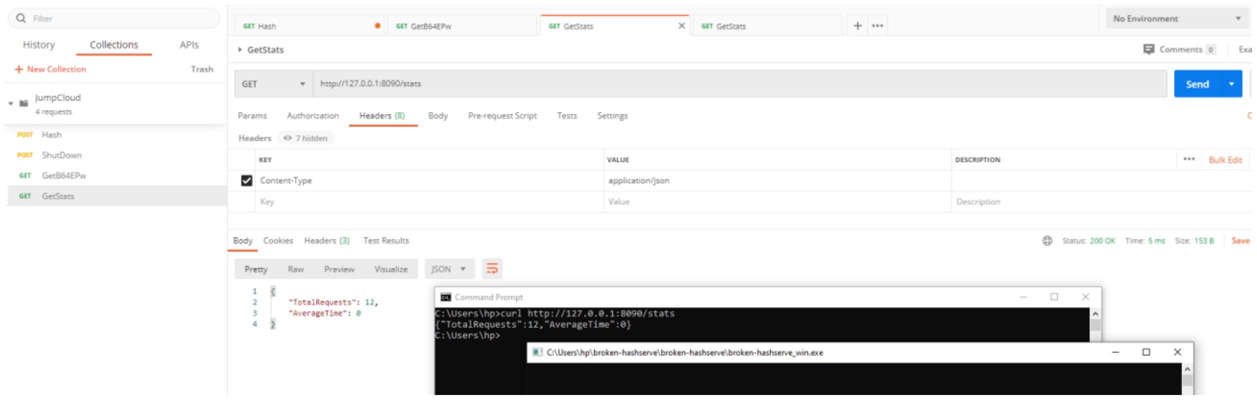
Headers (9)

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	

Status: 405 Method Not Allowed Time: 5 ms Size: 197 B

```
POST Not Supported
```

TC5.a



GET http://127.0.0.1:8090/stats

Headers (8)

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	

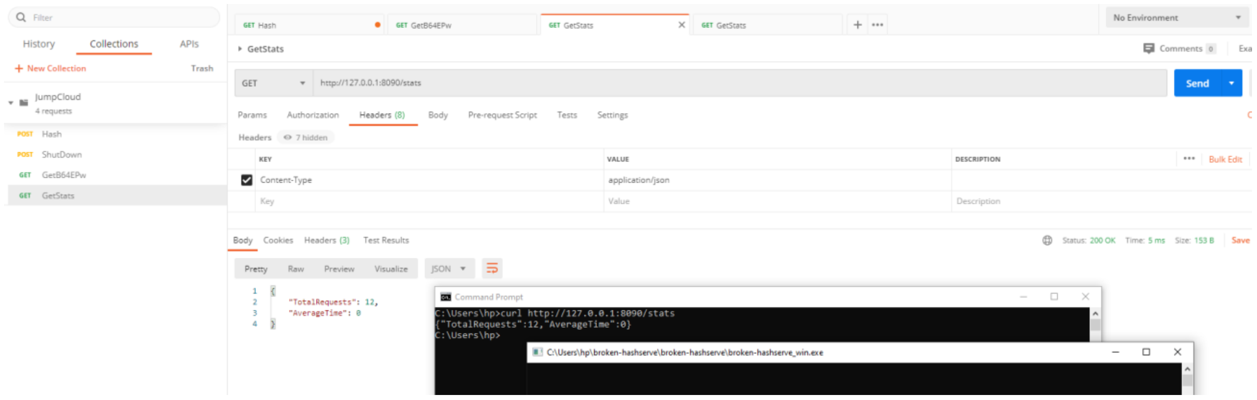
Status: 200 OK Time: 5 ms Size: 153 B

```
{
    "TotalRequests": 12,
    "AverageTime": 0
}
```

```
C:\Users\hp>curl http://127.0.0.1:8090/stats
{"TotalRequests":12,"AverageTime":0}
C:\Users\hp>
```

C:\Users\hp\broken-hashserve\broken-hashserve\broken-hashserve_win.exe

TC5.b



GET http://127.0.0.1:8090/stats

Headers (8)

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	

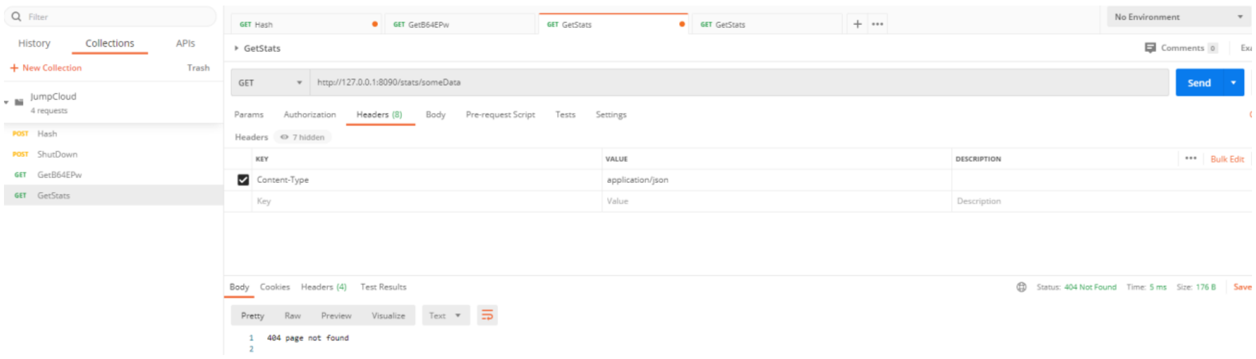
Status: 200 OK Time: 5 ms Size: 153 B

```
{
    "TotalRequests": 12,
    "AverageTime": 0
}
```

```
C:\Users\hp>curl http://127.0.0.1:8090/stats
{"TotalRequests":12,"AverageTime":0}
C:\Users\hp>
```

C:\Users\hp\broken-hashserve\broken-hashserve\broken-hashserve_win.exe

TC5.c



GetStats

GET http://127.0.0.1:8090/stats/someData

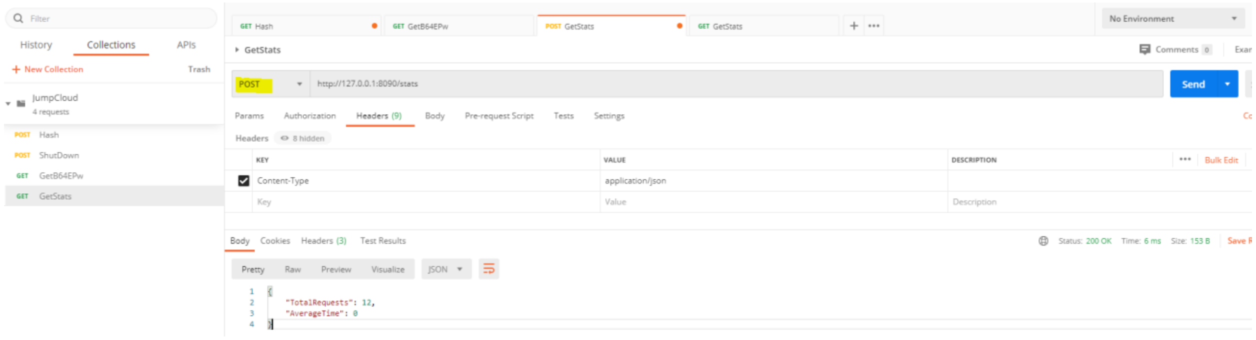
Headers (8) — 7 hidden

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

Status: 404 Not Found Time: 5 ms Size: 176 B

```
404 page not found
```

TC5.d



GetStats

POST http://127.0.0.1:8090/stats

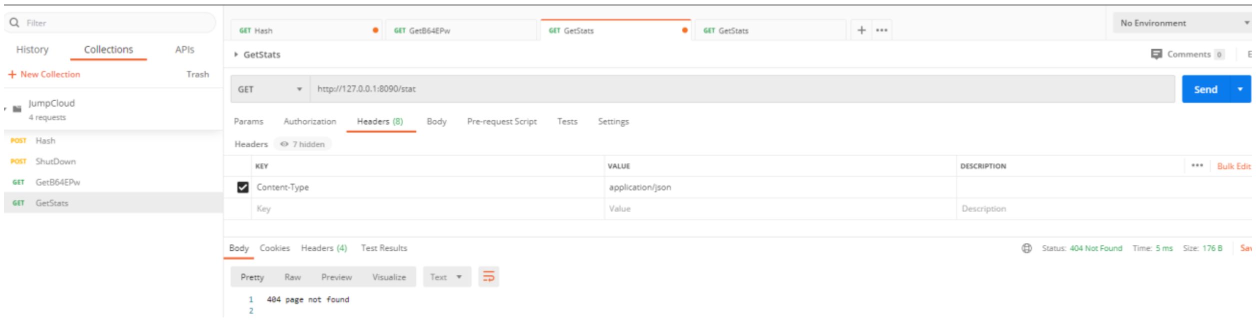
Headers (9) — 8 hidden

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

Status: 200 OK Time: 6 ms Size: 153 B

```
{
    "TotalRequests": 12,
    "AverageTime": 0
}
```

TC5.e



GetStats

GET http://127.0.0.1:8090/stat

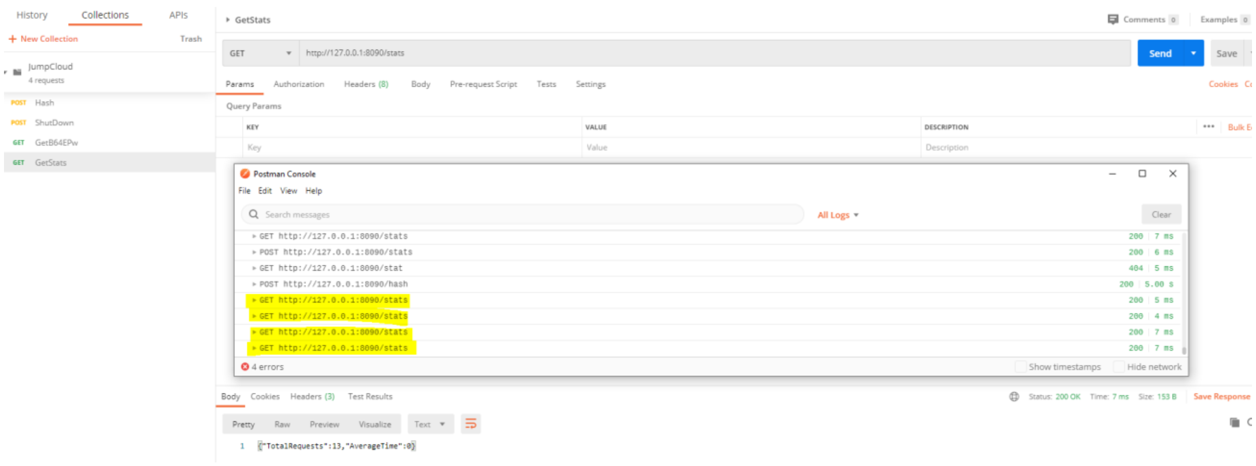
Headers (8) — 7 hidden

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

Status: 404 Not Found Time: 5 ms Size: 176 B

```
404 page not found
```

TC6.a



GetStats

GET http://127.0.0.1:8090/stats

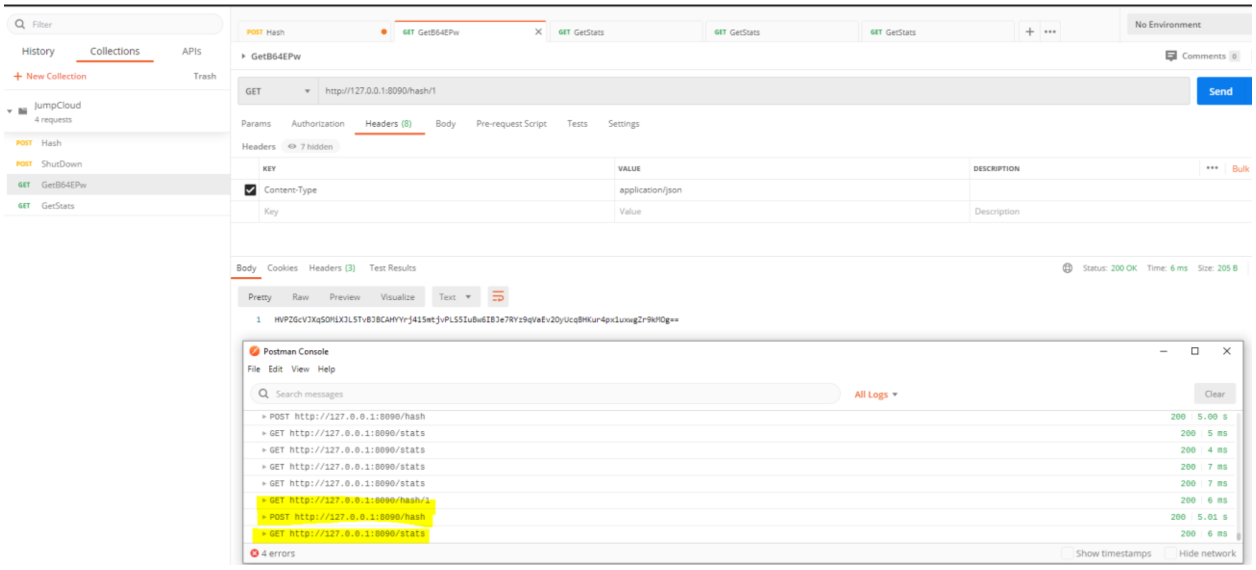
```
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
▸ POST http://127.0.0.1:8090/stats               200 | 6 ms
▸ GET http://127.0.0.1:8090/stat                 404 | 5 ms
▸ POST http://127.0.0.1:8090/hash                200 | 5.00 s
▸ GET http://127.0.0.1:8090/stats                200 | 5 ms
▸ GET http://127.0.0.1:8090/stats                200 | 4 ms
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
```

4 errors

Status: 200 OK Time: 7 ms Size: 153 B

```
{"TotalRequests":13,"AverageTime":0}
```

TC6.b



GetB64EPw

GET http://127.0.0.1:8090/hash/1

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

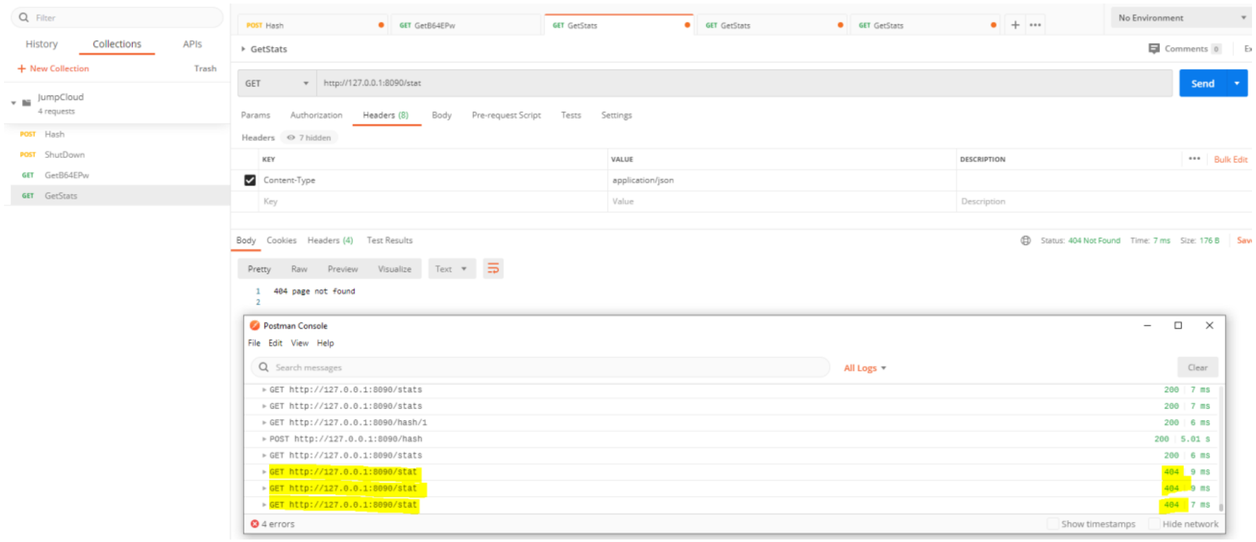
Status: 200 OK Time: 6 ms Size: 205 B

```
HVPZGcVJXqSOMiXJL5TvBJBCAHYYrj415mtjvPLS5IuBw6IBJe7RYz9qVaEv2OyUcqBHKur4px1uxwgZr9kMOg==
```

```
▸ POST http://127.0.0.1:8090/hash                200 | 5.00 s
▸ GET http://127.0.0.1:8090/stats                200 | 5 ms
▸ GET http://127.0.0.1:8090/stats                200 | 4 ms
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
▸ GET http://127.0.0.1:8090/hash/1               200 | 6 ms
▸ POST http://127.0.0.1:8090/hash                200 | 5.01 s
▸ GET http://127.0.0.1:8090/stats                200 | 6 ms
```

4 errors

TC6.c



GetStats

GET http://127.0.0.1:8090/stat

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

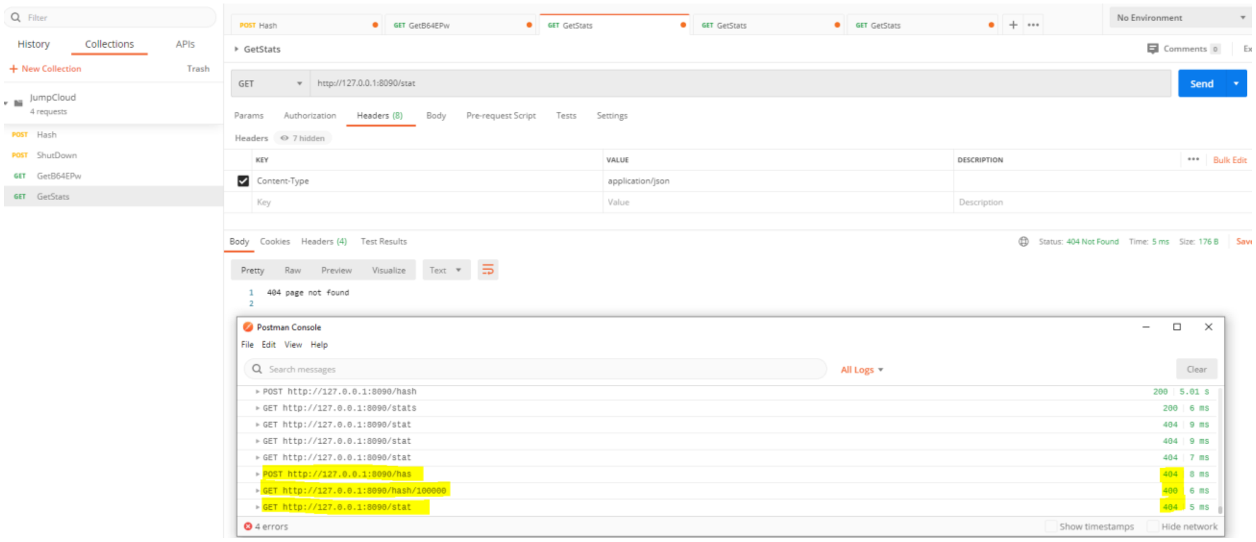
Status: 404 Not Found Time: 7 ms Size: 176 B

```
404 page not found
```

```
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
▸ GET http://127.0.0.1:8090/stats                200 | 7 ms
▸ GET http://127.0.0.1:8090/hash/1               200 | 6 ms
▸ POST http://127.0.0.1:8090/hash                200 | 5.01 s
▸ GET http://127.0.0.1:8090/stats                200 | 6 ms
▸ GET http://127.0.0.1:8090/stat                 404 | 9 ms
▸ GET http://127.0.0.1:8090/stat                 404 | 9 ms
▸ GET http://127.0.0.1:8090/stat                 404 | 7 ms
```

4 errors

TC6.d


GetStats

GET http://127.0.0.1:8090/stat

Headers (8)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

Status: 404 Not Found Time: 5 ms Size: 176 B

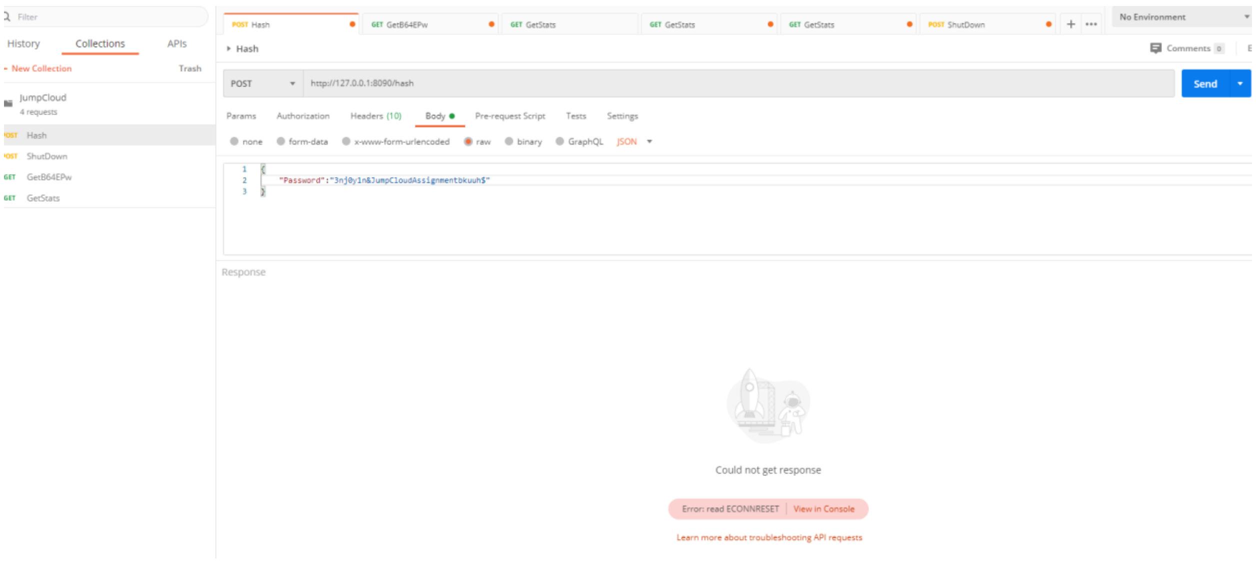
```
404 page not found
```

Postman Console

```
▸ POST http://127.0.0.1:8090/hash                 200   5.01 s
▸ GET http://127.0.0.1:8090/stats                 200   6 ms
▸ GET http://127.0.0.1:8090/stat                  404   9 ms
▸ GET http://127.0.0.1:8090/stat                  404   9 ms
▸ GET http://127.0.0.1:8090/stat                  404   7 ms
▸ POST http://127.0.0.1:8090/has                  404   8 ms
▸ GET http://127.0.0.1:8090/hash/100000           400   6 ms
▸ GET http://127.0.0.1:8090/stat                  404   5 ms
```

4 errors

TC7.a


Hash

POST http://127.0.0.1:8090/hash

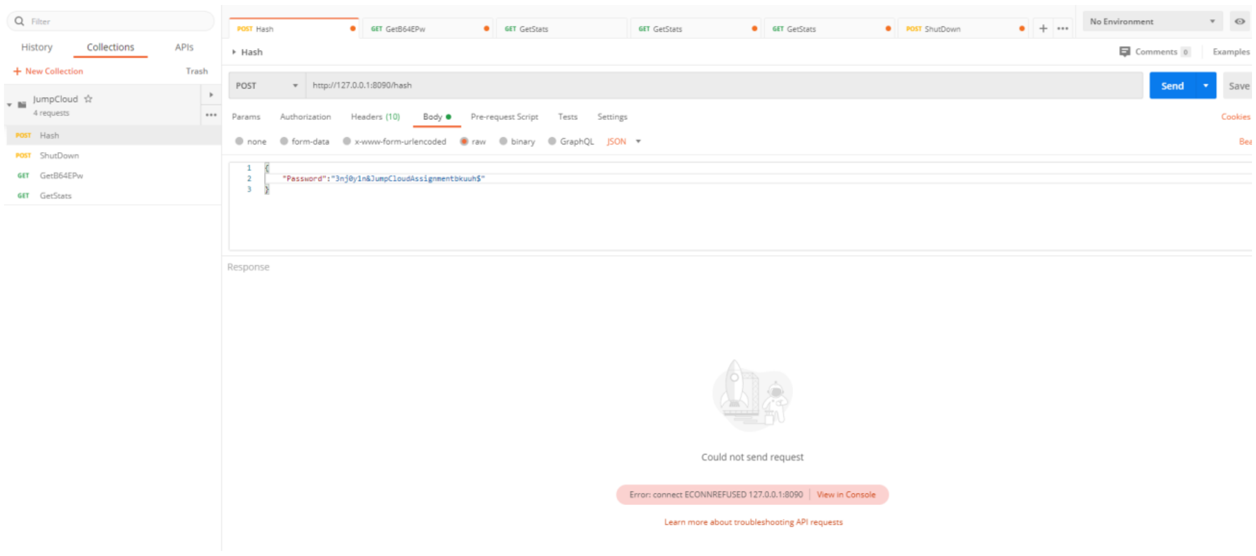
```
{
    "Password":"3nj0y1n&JumpCloudAssignmentbkuuh$"
}
```

Response

Could not get response

Error: read ECONNRESET | View in Console

Learn more about troubleshooting API requests



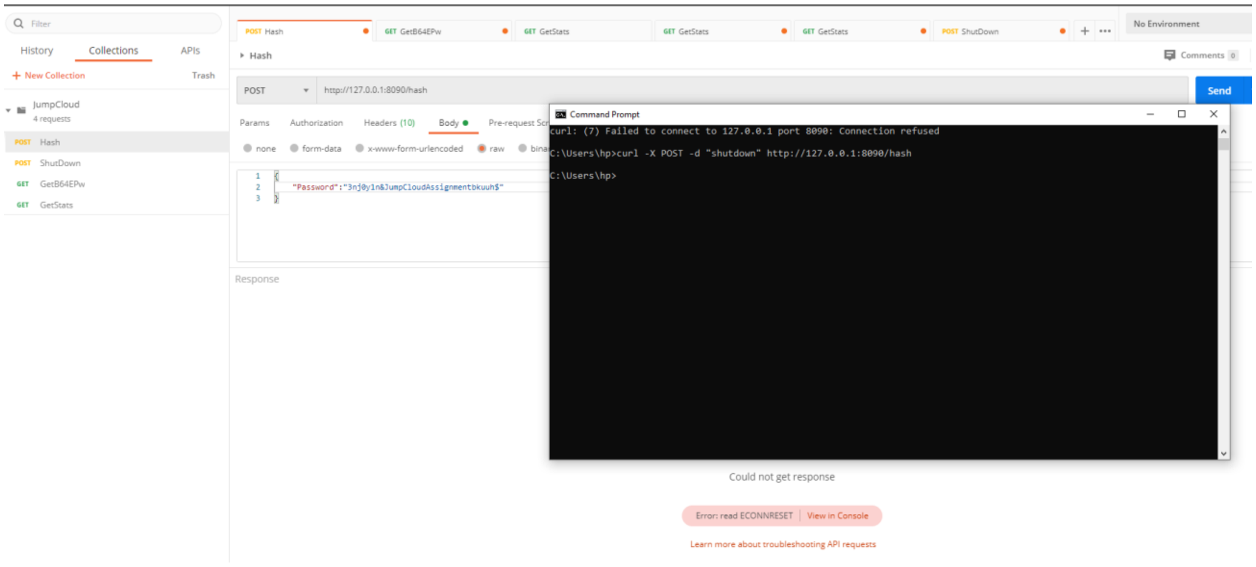
POST http://127.0.0.1:8090/hash

```
{
    "Password":"3nj0y1n&JumpCloudAssignmentbkuuh$"
}
```

Could not send request

Error: connect ECONNREFUSED 127.0.0.1:8090 | View in Console

Learn more about troubleshooting API requests



POST http://127.0.0.1:8090/hash

```
{
    "Password":"3nj0y1n&JumpCloudAssignmentbkuuh$"
}
```

Command Prompt

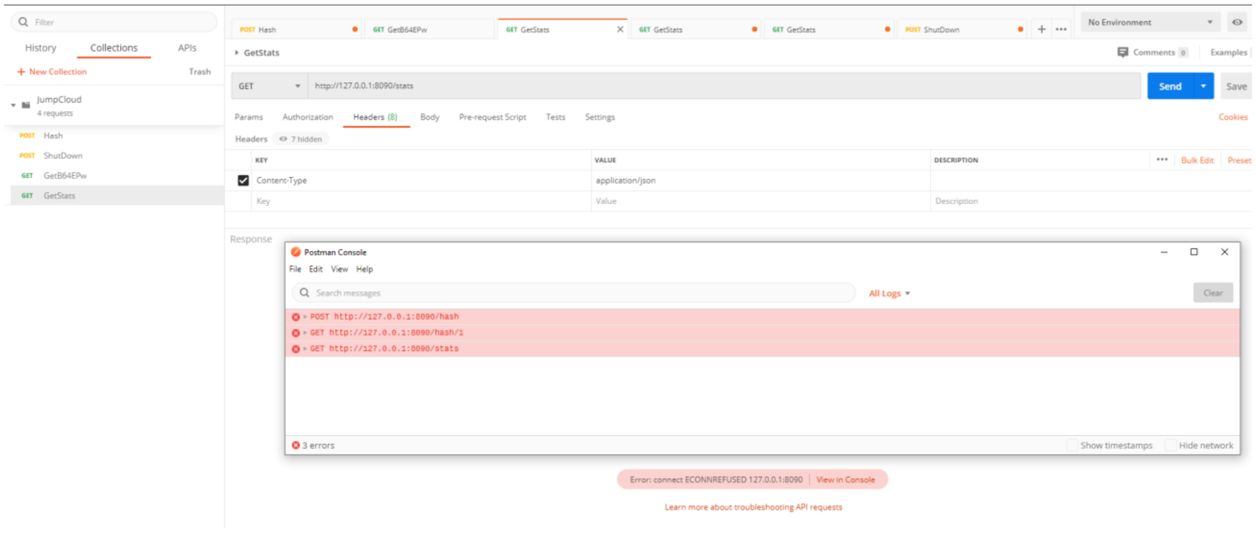
```
curl: (7) Failed to connect to 127.0.0.1 port 8090: Connection refused

C:\Users\hp>curl -X POST -d "shutdown" http://127.0.0.1:8090/hash

C:\Users\hp>
```

Could not get response

Error: read ECONNRESET | View in Console

Learn more about troubleshooting API requests

TC7.b



GetStats

GET http://127.0.0.1:8090/stats

Headers (8)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

Postman Console

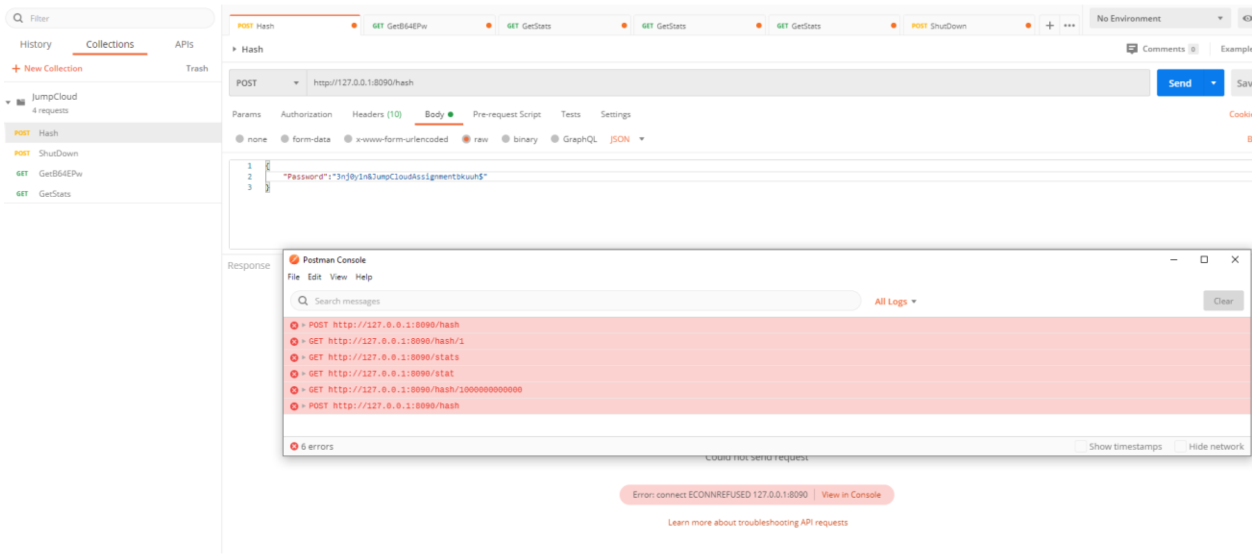
```
POST http://127.0.0.1:8090/hash
GET http://127.0.0.1:8090/hash/1
GET http://127.0.0.1:8090/stats
```

3 errors

Error: connect ECONNREFUSED 127.0.0.1:8090 | View in Console

Learn more about troubleshooting API requests

TC7.c



Hash

POST http://127.0.0.1:8090/hash

```
{
    "Password":"3nj0y1n&JumpCloudAssignmentbkuuh$"
}
```

Postman Console

```
POST http://127.0.0.1:8090/hash
GET http://127.0.0.1:8090/hash/1
GET http://127.0.0.1:8090/stats
GET http://127.0.0.1:8090/stat
GET http://127.0.0.1:8090/hash/1000000000000
POST http://127.0.0.1:8090/hash
```

6 errors

Could not send request

Error: connect ECONNREFUSED 127.0.0.1:8090 | View in Console

Learn more about troubleshooting API requests