Lab: HTTP API

version: 1.01

An introduction to some of the available operations that can be performed using HTTP requests.

OVERVIEW

EventStoreDB provides http endpoints for a number of user and admin operations.

You might use a tool like Postman or http code libraries to access this functionality in production.

Command line tools such as cURL are also commonly used for prototyping applications that use http endpoints.

For this lab we use the httpyac plugin in vscode. The httpyac format is more suited to training classes than Postman which requires registration, and cURL which has a complicated sytax.

This exercise is comprised of the following steps:

1. Install the httpyac plugin for VScode
2. Verify that KurrentDB is running
3. Run a basic GET request against google.com
4. Review of http and format of http request in httpyac
5. Run some examples
6. httpyac format compared to cURL

Step 1. Install httpyac plugin

In the terminal run

```
code --install-extension anweber.vscode-httpyac
```

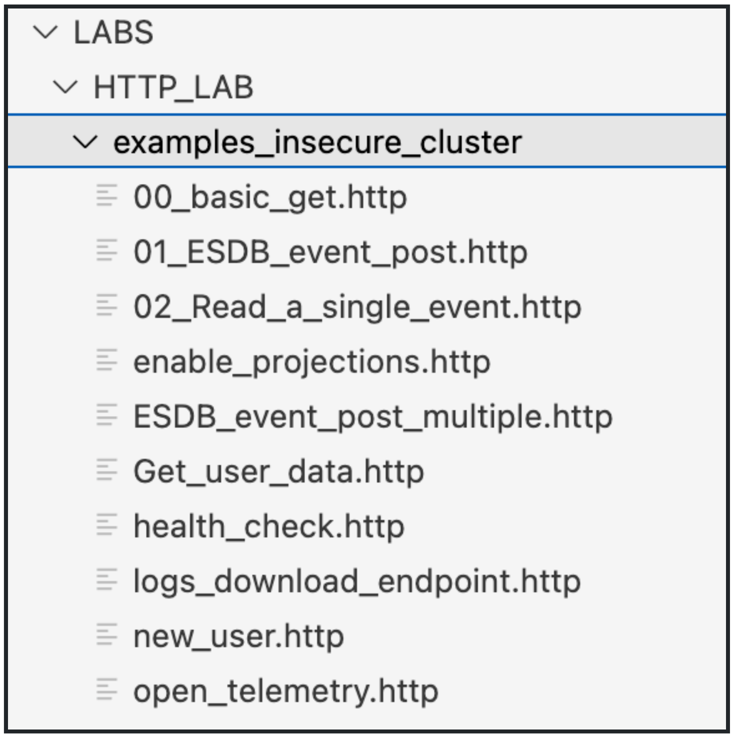
Step 2. Verify that KurrentDB is running

In the terminal run

```
./start_db.sh
```

Step 3. Navigate to the examples folder

Navigate to the `/LABS/HTTP_LAB` folder and open the `examples_insecure_cluster` folder.



```
LABS
  HTTP_LAB
    examples_insecure_cluster
      00_basic_get.http
      01_ESDB_event_post.http
      02_Read_a_single_event.http
      enable_projections.http
      ESDB_event_post_multiple.http
      Get_user_data.http
      health_check.http
      logs_download_endpoint.http
      new_user.http
      open_telemetry.http
```

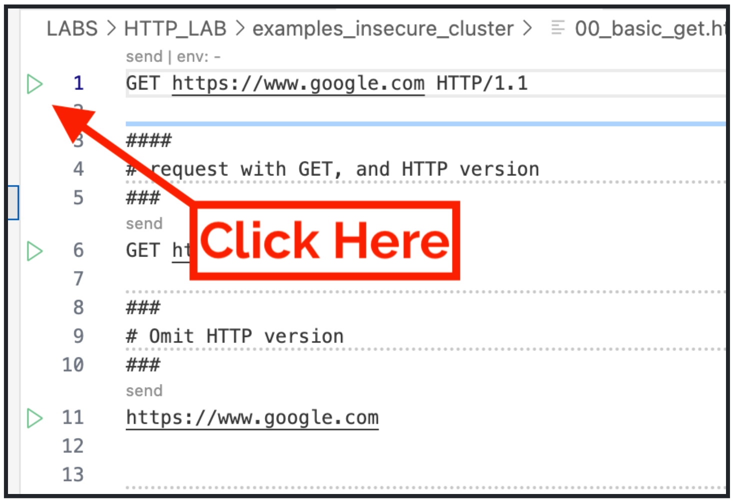
Step 4. Run a simple GET request against google.com

In vscode, open the first file, `00_basic_get.http`

The first line is a GET request to google.com

Click on the arrow to execute.

If succesful you should see the contents of the google webpage as plain text in a response window. Close the response windows after each command to prevent having a very crowded collection of windows.



LABS > HTTP_LAB > examples_insecure_cluster > 00_basic_get.ht

```
send | env: -
GET https://www.google.com HTTP/1.1

####
# request with GET, and HTTP version
###
send
GET ht

###
# Omit HTTP version
###
send
https://www.google.com

```

Step 5. Post an event to KurrentDB

Open the file `01_ESDB_event_post.http` and make the http POST request to KurrentDB. The endpoint is `/streams/newstream` and the POST content is a json string containing the event to be appended to the stream `newstream`

The response should be `HTTP/1.1 201 - Created`

Step 6. Verify the event using the webui

Go to the stream browser tab in the webui, and verify that the stream `newstream` has the event.

Step 7. Read an Event

Open the file `02_read_a_single_event.http` and submit the request.

You should see the response window with a first line of

```
HTTP/1.1 200 - OK
```

And the following lines will contain the JSON formatted event.

Note this requires that you had posted the event in the previous step.

Congratulations !!

You have used the http api to post an event to Kurrent DB

Step 7 was your last step, the following is additional information that you might be interested in.

Overview of HTTP Request details

An HTTP request has:

- a request line
- optional header fields
- optional message body.

Request-Line

A request line consists of a request method, target, and the HTTP protocol version. If the request method is omitted, 'GET' will be used as a default. The HTTP protocol version can also be omitted.

See line 1, of `00_basic_get.http` in the `EventstoreDB_examples/examples_insecure_cluster` folder.

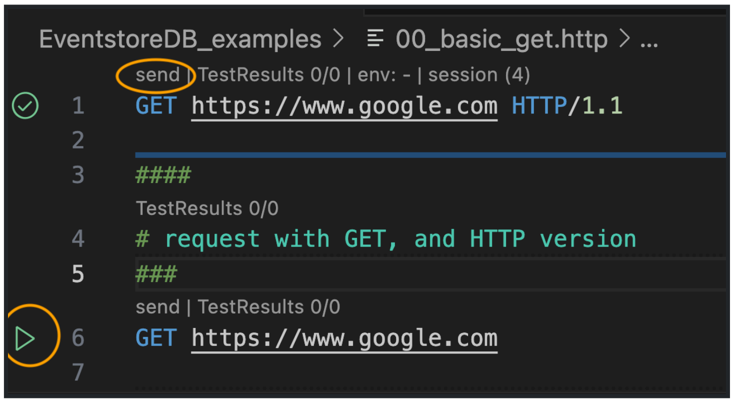
This is a line with the request type GET included, to the host `www.google.com,` using HTTP version 1.1

```
GET https://www.google.com HTTP/1.1
```

Note that if the request type was not included GET would be assumed.

If the protocol is ommitted it would default to HTTP/1.1

The file contains examples of various formats for what resolves to the same request. In vscode, with the httpyac plugin installed, you can run each line by clickin on `send` above the line, or clicking on the green arrow next to the line.


EventstoreDB_examples > 00_basic_get.http > ...
send | TestResults 0/0 | env: - | session (4)
TestResults 0/0
send | TestResults 0/0

```
GET https://www.google.com HTTP/1.1

####
# request with GET, and HTTP version
###
GET https://www.google.com
```

Query-String

Query string can be part of the request line.

```
GET https://google.com/?q=EventStore
```

Or the query string can be on the next line.

```
GET https://google.com/
    ?q=EventStore
```

Headers

Headers are entered one per line following the request.

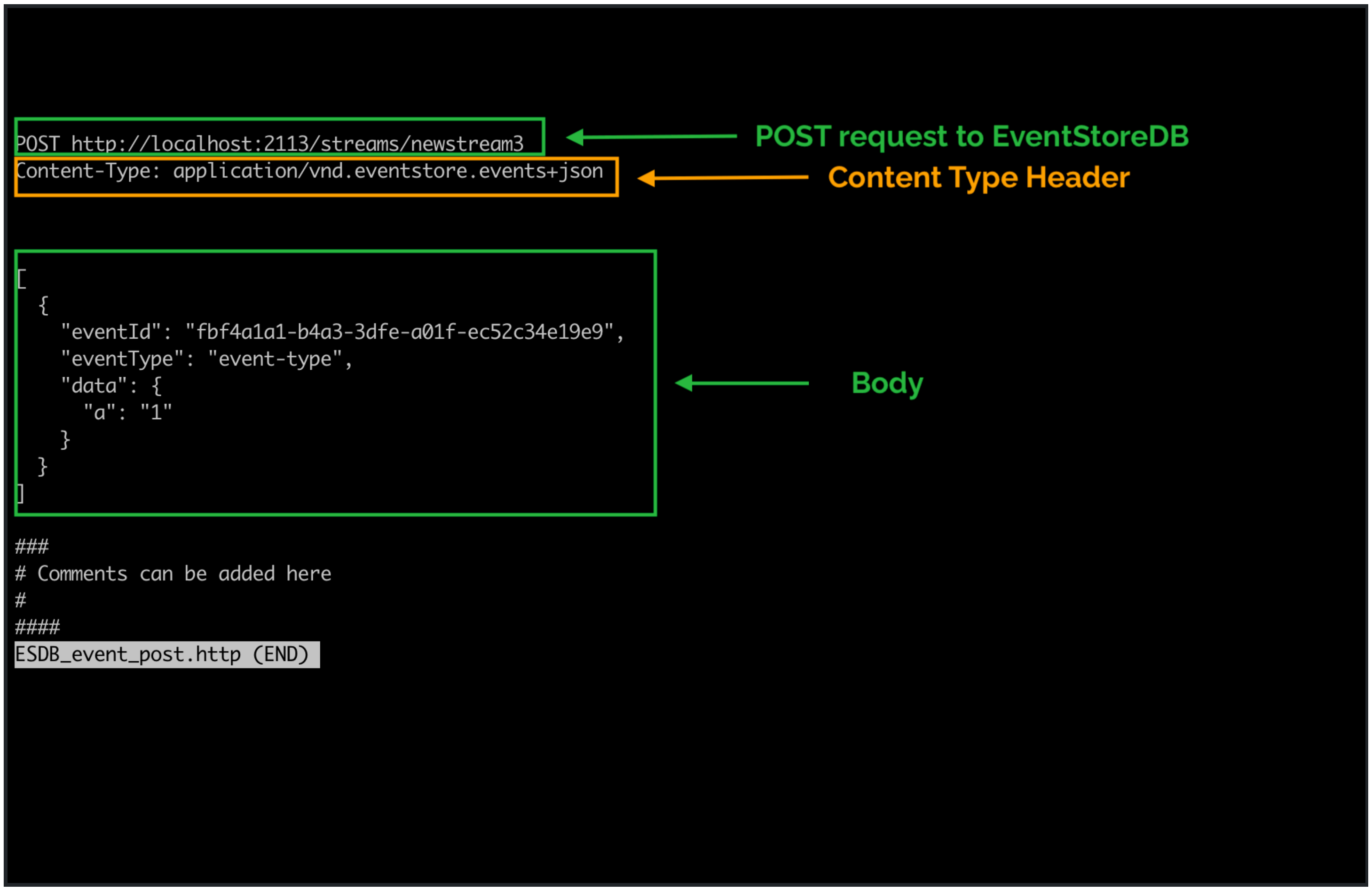
Example:

```
GET https://httpbin.org/anything
Content-Type: text/html
Authorization: Bearer token
```

Request Body

The request body is entered as lines below the Request Headers. Any line that can not be interpreted as a Header is assumed to be the Request Body.

See the image below.



```
POST http://localhost:2113/streams/newstream3
Content-Type: application/vnd.eventstore.events+json


[
  {
    "eventId": "fbf4a1a1-b4a3-3dfe-a01f-ec52c34e19e9",
    "eventType": "event-type",
    "data": {
      "a": "1"
    }
  }
]

###
# Comments can be added here
#
####
ESDB_event_post.http (END)
```

Here is the text of the file.

```
POST http://localhost:2113/streams/newstream
Content-Type: application/vnd.eventstore.events+json




[
  {
    "eventId": "fbf4a1a3-b4a3-3dfe-a01f-ec52c34e19e9",
    "eventType": "event-type",
    "data": {
      "a": "1"
    }
  }
]
```

Comparison of httpyac format to curl

If you prefer cURL commands, here is a quick comparison.

This httpyac format to post an event to a stream

```
POST http://localhost:2113/streams/newstream
Content-Type: application/vnd.eventstore.events+json



[
  {
    "eventId": "fbf4a1a3-b4a3-3dfe-a01f-ec52c34e19e9",
    "eventType": "event-type",
    "data": {
      "a": "1"
    }
  }
]
```

Would be the equivalent to the following cURL command.

```
curl \-i \-d "@event.json" "http://127.0.0.1:2113/streams/newstream3" \\
    \-H "Content-Type:application/vnd.eventstore.events+json"
```

With a data file event.json containing.

```json

[
  {
    "eventId": "fbf4a1a3-b4a3-3dfe-a01f-ec52c34e19e9",
    "eventType": "event-type",
    "data": {
      "a": "1"
    }
  }
]
```