We observed that our inhouse servers, running splunk were not working properly. Some performance issues and throttling were observed. To understand why this was happening we onboarded different types of logs and monitored various parameters. Maintaining proper temperature and humidity levels of server room is also an important factor that can influence its performance. This post will guide you through on how to get actuator’s Temperature and Humidity readings and **HEC** it into the **Splunk**.

**Hardware Requirements:**

1. DHT11 Module: Temperature and Humidity sensor
2. NodeMCU: WIFI module devkit
3. Jumper wires

**Technology Requirements:**

1. Splunk
2. Arduino IDE

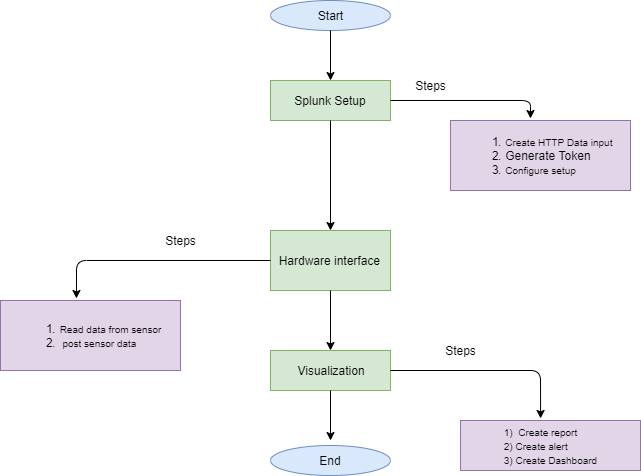
**What is HTTP POST request method?**

POST is a request method supported by HTTP, used by the World Wide Web. By design, the POST method requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it. It is often used when uploading a file or when submitting a completed web form. The header field in the POST request usually indicates the message body's Internet media type and other parameters used like authentication.

**What is HEC?**

HTTP Event Collector or HEC is a built-in feature of Splunk which is integrated keeping developers in mind. HEC is a token-based HTTP server where developers can POST the data/logs to be indexed.

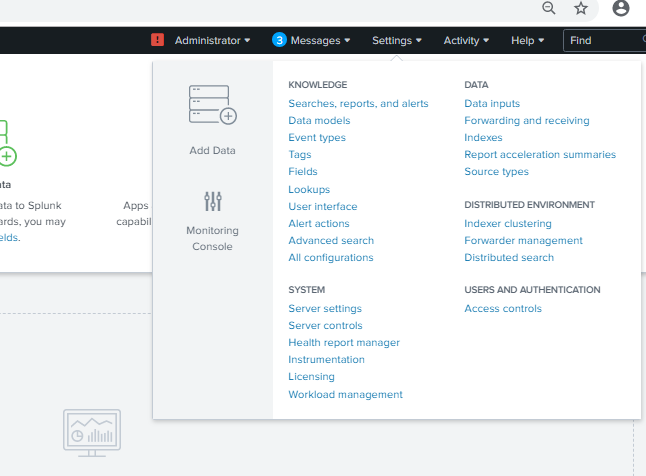
**Flow Diagram:**



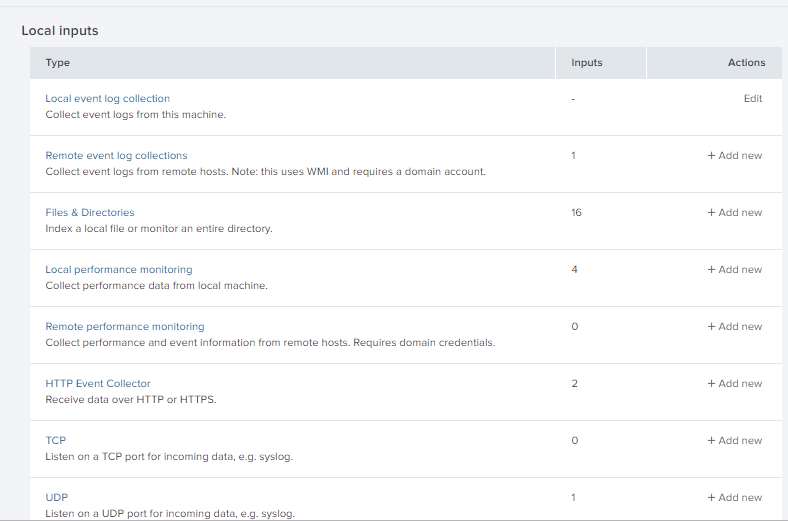
# Part 1 **HEC setup on Splunk:**

This part contains steps to set-up HEC in Splunk.

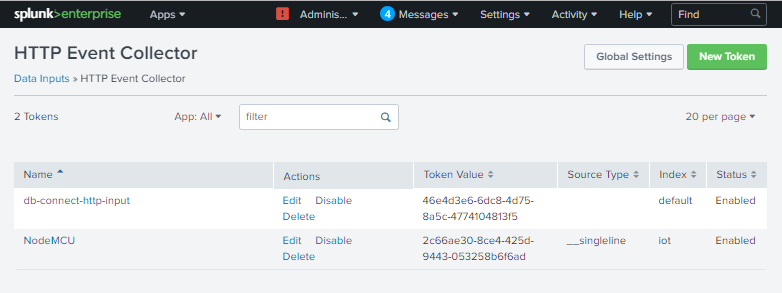
1. Click on settings and select data inputs



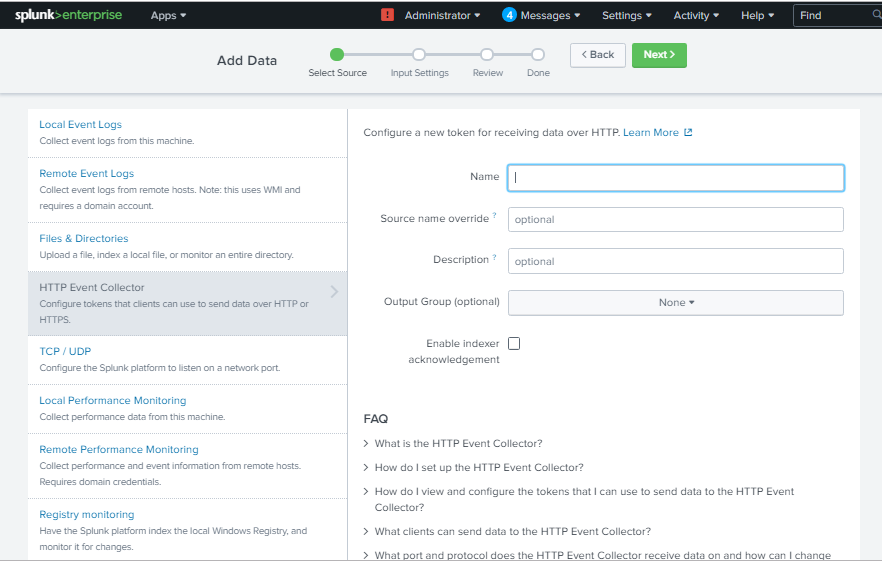
1. Select **HTTP Event Collector** option in local inputs

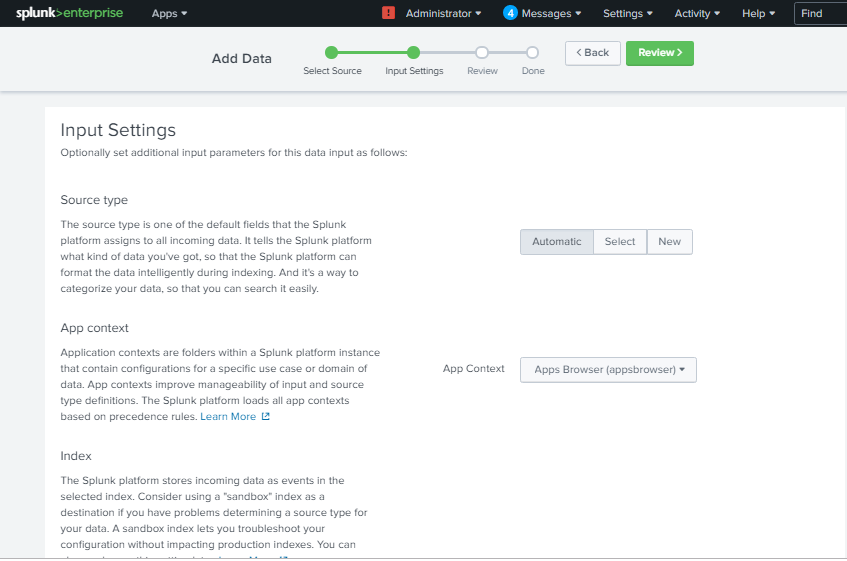


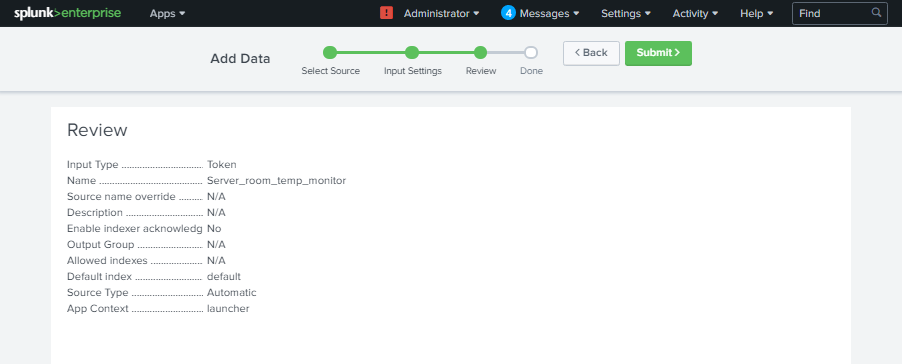
1. generate **New Token** by clicking on new token button



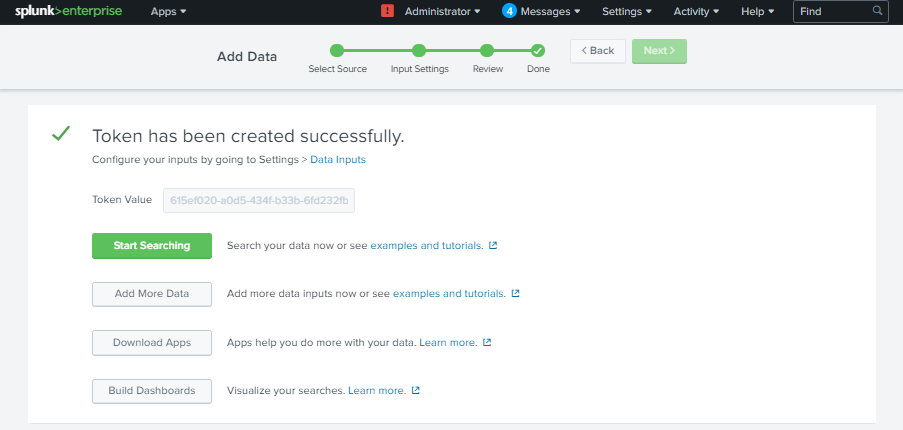
1. Fill the desired options and click on next



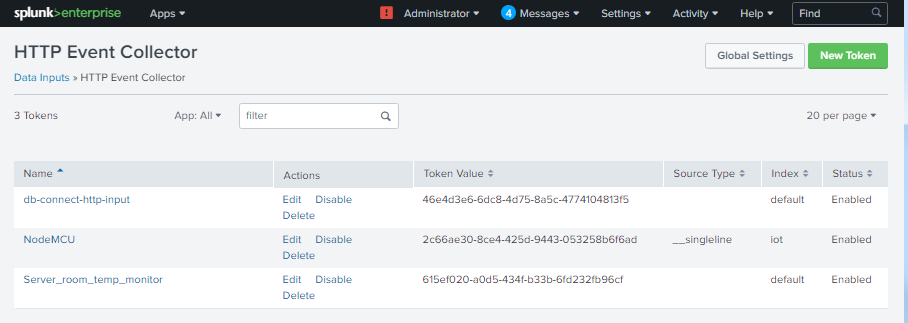
1. Configure input settings
2. Review all configuration



1. Congratulations! Your **HEC** is configured and token is generated successfully!



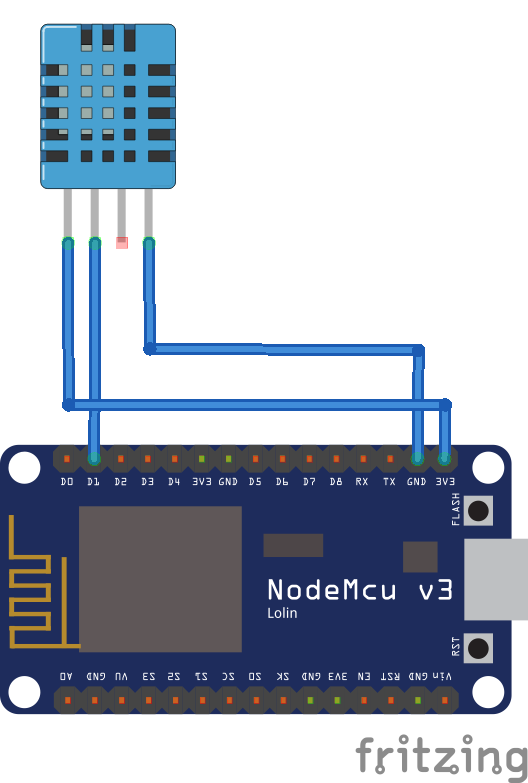
1. You can get the list of HECs in **Settings** **Data Inputs** **HTTP Event Collector**



# PART 2 **Working with IC and Sensors:**

This part contains the steps to interface the sensor and NodeMCU and POST data in Splunk.

**Hardware Interfacing:**



Using Jumper wires connect:

|  |  |
| --- | --- |
| **NODEMCU** | **DHT11** |
| Vcc | Vcc |
| GND | GND |
| D1 | Data Pin |

**Code:**

|  |
| --- |
| #include <ESP8266WiFi.h>  #include <ESP8266HTTPClient.h>  #include "DHT.h"  #define DHTPIN 5  #define DHTTYPE DHT11  float t,h;  DHT dht(DHTPIN, DHTTYPE);  void setup() {  Serial.begin(9600); //Serial connection  dht.begin();  WiFi.begin("xxxx", "xxxx"); //WiFi connection  while (WiFi.status() != WL\_CONNECTED) { //Wait for the WiFI connection completion  delay(500);  Serial.println("Waiting for connection");  }  }    void loop() {  h = dht.readHumidity();  t = dht.readTemperature();  if(WiFi.status()== WL\_CONNECTED){ //Check WiFi connection status  HEC();  }else{  Serial.println("Error in WiFi connection");  }    delay(60000); //Send a request every 1 mins  }  void HEC(){  HTTPClient http; //Declare object of class HTTPClient  temp=DHT  http.begin("http://**<YourIPAddress/URL>**:8088/services/collector"); //Specify request destination  http.addHeader("Content-Type", "text/plain"); //Specify content-type header  http.addHeader("Authorization", "Splunk xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx");  String post="{\"event\":{\"Temperature\":"+String(t)+",\"Humidity\":"+String(h)+"}}";  Serial.println(post);  int httpCode = http.POST(post); //Send the request  String payload = http.getString(); //Get the response payload  Serial.println(httpCode); //Print HTTP return code  Serial.println(payload); //Print request response payload  http.end(); //Close connection  } |

**How the Code Works:**

Take a quick look at the code and see how it works.

First include all the necessary libraries. You include different libraries depending on the board you’re using, Functionalities and actuators interfacing.

|  |
| --- |
| #include <ESP8266WiFi.h>  #include <ESP8266HTTPClient.h>  #include "DHT.h" |

The following is used to communicate with the actuator DHT11 on GPIO5 i.e. D1.

|  |
| --- |
| #define DHTPIN 5  #define DHTTYPE DHT11  float t,h;  DHT dht(DHTPIN, DHTTYPE); |

Connect to WiFi.

|  |
| --- |
| WiFi.begin("**xxxx** ", " **xxxx** "); //WiFi connection  while (WiFi.status() != WL\_CONNECTED) { //Wait for the WiFI connection completion  delay(500);  Serial.println("Waiting for connection");  } |

Read Sensor Value and check Wifi connection status

|  |
| --- |
| h = dht.readHumidity();  t = dht.readTemperature();  if(WiFi.status()== WL\_CONNECTED){ //Check WiFi connection status  **<HEC is done here>**  } |

**HTTP post request:**

**Destination URL:** This is where you’ll need to send the post request to. This is the URL/IP address where your splunk is hosted with default port 8088 and followed by /services/collector. It should look something like this: **http://<Your splunk URL/IP>:8088/services/collector**

**Authorization code:**

Authorization code is the token that we got from Splunk after HEC setup. This needs to be placed in header.

**Content type:** It is the type of the content/data/body that we’re sending.

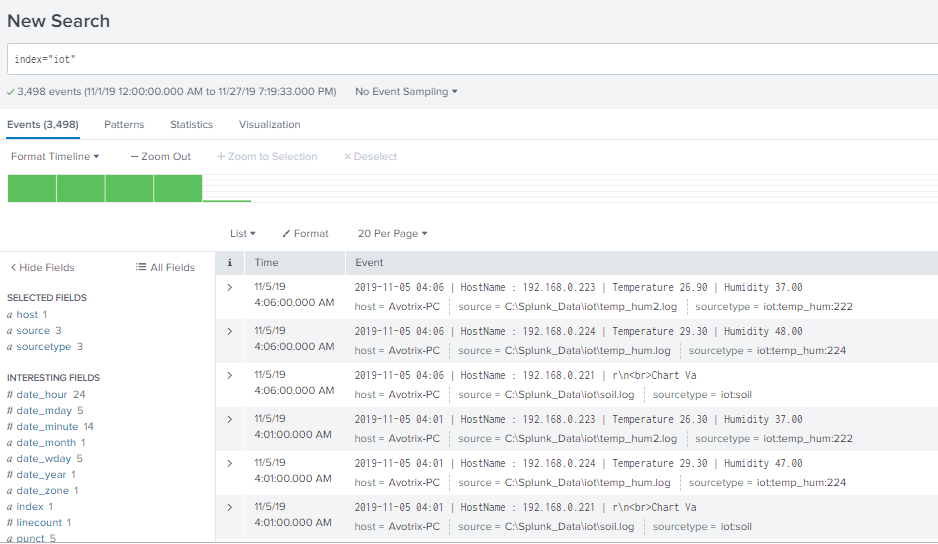
**Data:** Here the data is given in post variable. This is the actual data that is to be indexed in splunk. For JSON setup (in Splunk) the data should be in **event JSON object** like:

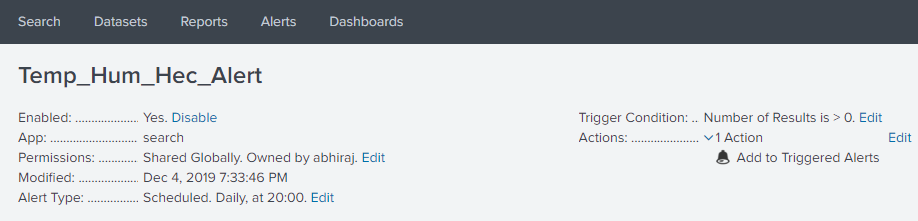
{“event”:{“Key”:Value}}

Here the red colored part is your data.

|  |
| --- |
| void HEC(){  HTTPClient http; //Declare object of class HTTPClient  temp=DHT  http.begin("http://**<YourIPAddress/URL >**:8088/services/collector"); //Specify request destination  http.addHeader("Content-Type", "text/plain"); //Specify content-type header  http.addHeader("Authorization", "Splunk xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx");  String post="{\"event\":{\"Temperature\":"+String(t)+",\"Humidity\":"+String(h)+"}}";  Serial.println(post);  int httpCode = http.POST(post); //Send the request  String payload = http.getString(); //Get the response payload  Serial.println(httpCode); //Print HTTP return code  Serial.println(payload); //Print request response payload  http.end(); //Close connection  } |

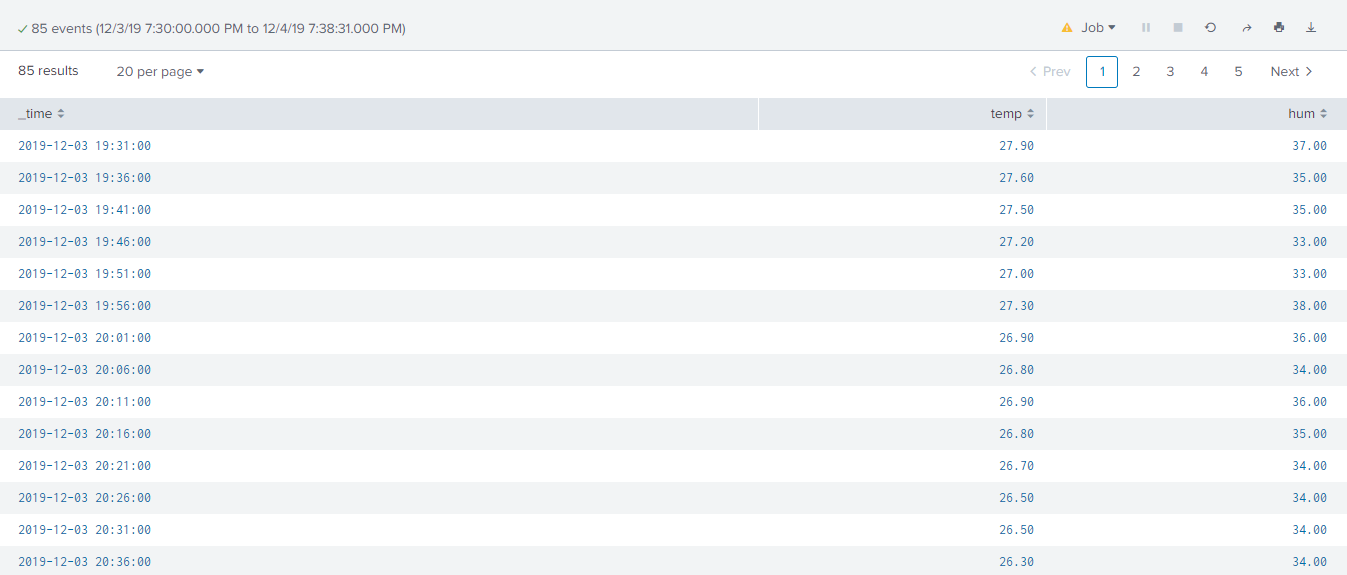
# Part 3 **Visualization**



**Alert** 

**Report**





**Dashboard**

