## 

[**CloudSoc Console - Configuring GCP connection 1**](#_hpi6f48j9h3f)

[**CloudSoc Console - Create Public API keys 1**](#_a6k5xnlwjg46)

[**Variables Used 2**](#_ht6oywb3z847)

[**Convert publickey, secret key to base 64 encoding 3**](#_g9c8tdp6mwcv)

[**Generate JWT Token 3**](#_975l6wobu68o)

[**Activating GCP Connection 4**](#_yte1h6ogj5zo)

[**Validate GCP Connection 5**](#_h73yg0h3d42h)

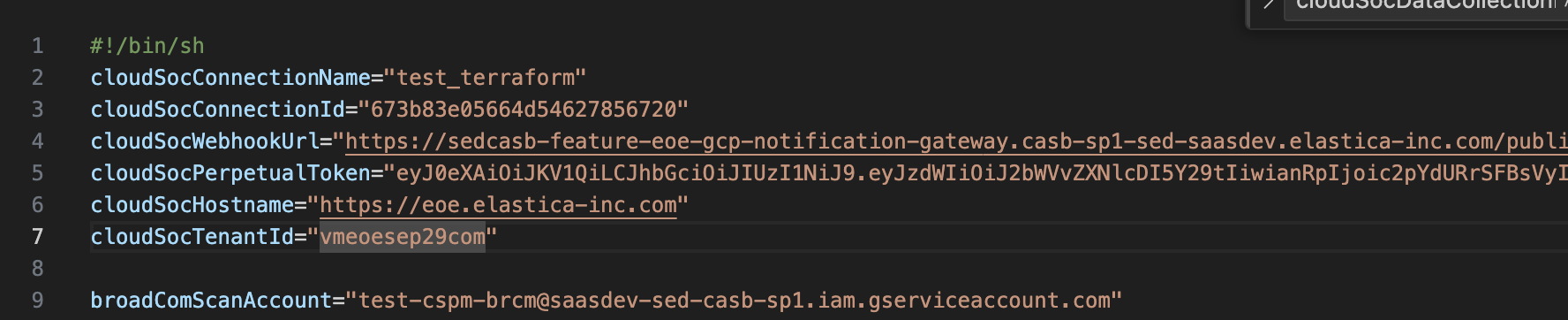
[**Status Code 6**](#_3zsmo7qp58mp)

## **Please Note: All the items marked in red below is needed to run the API**

## CloudSoc Console - Configuring GCP connection

Note: You have already executed this step while running terraform, no need to create another connection, use the downloaded script and copy the variables.

1. Log in to your CloudSoc account
2. Goto Securlet and click on Google Cloud
3. You will be routed to connections page
4. Click on Add Connections and Select GCP
5. You will be directed to GCP Configuration page
6. Give a connection name and save the connection
7. Then press “Download Shell Script”. The script will be downloaded in your default folder.
8. Copy following values from script and keep it handy.You can search for ("cloudSocConnectionName", "cloudSocConnectionId”, “cloudSocHostname” ​​ and cloudSocTenantId) key word in shell script. This will be used later to run public APIs

For example:  
  


## CloudSoc Console - Create Public API keys

1. On the right hand top of CloudSoc console click the gear icon, it will take you to the settings page
2. Under that on left pane click on API keys
3. Under API key, type the name of your key and click on Add new API Key
4. New API key will be generated in the pane below
5. Just on the right of the API keys in the Action column, there is a download key option to download the keys. Click on that and download the file, we will need following information

"tenant": "xxxx",

"api\_server": “xxxx”

"user\_id": “xxxx”

"key\_id": “xxxx”

"key\_secret": “xxxx”

## Variables Used

Below tables list all the variables which are used in public APIs and it also provides a way to get them.

| **Variables** | **Source** | **Remarks** | **Link** |
| --- | --- | --- | --- |
| **publickey:secretkey** | key\_id:key\_secret | You get these values in json when you click on download key | [Here](#_a6k5xnlwjg46) |
| **encoded\_key** | You get this by running command echo -n 'publickey:secretkey' | openssl base64 |  | [Here](#_g9c8tdp6mwcv) |
| **tenant** | Stored in downloaded shell script in cloudSocTenantId variable | This is needed while running public APIs | [Here](#_hpi6f48j9h3f) |
| **hostname** | Stored in downloaded shell script in cloudSocHostname variable | This is needed for calling Public APIs and {hostname} should be replaced by these values depending on the environment that you are in | [Here](#_hpi6f48j9h3f) |
| **valid\_token** | You get this by running /iaas/api/public/v2/token API mentioned below | This is used to authenticate public APIs for valid information | [Here](#_975l6wobu68o) |
| **connection\_id** | Stored in downloaded shell script in cloudSocConnectionId variable | This is used to trigger the public API on right connection | [Here](#_hpi6f48j9h3f) |
| **connection\_name** | Stored in downloaded shell script in cloudSocConnectionName variable | This is used to trigger the public API on right connection | [Here](#_hpi6f48j9h3f) |
| **gcp\_project\_id** | This is the gcp project which you want to onboard for GCP | GCP project that you want to onboard | Login to gcp account and get the project from there. You can get it from the link too by replacing your\_project\_name with you actual project name in the link  <https://console.cloud.google.com/iam-admin/settings?project=your_project_name> |
| **gcp\_service\_account** | This is gcp service account which will be used to perform operations | GCP service account used by cloudsoc to perform actions | Output from Terraform |

### 

## Convert publickey, secret key to base 64 encoding

You can generate this on MAC by using the following command.

echo -n 'publickey:secretkey' | openssl base64

You will get the 'publickey’(Key\_id), ‘secretkey’(key\_secret). Copy the output from the above command and keep it for next steps. This will generate the encoded\_key which is needed in subsequent APIs

## Generate JWT Token

This will be used in subsequent APIs. This is used to authenticate the public APIs.

URL: {hostname}/iaas/api/public/v2/token

Method: POST

Headers:

X-Elastica-Dbname-Resolved:true

Content-Type:application/json

Authorization:Basic <encoded\_key>

tenant:<tenant>

You will get the <encoded\_key> from above step and <tenant> (tenant) is present here

Body: N/A

Response: Please note we will be using this

{"token": "<valid\_token>"}

**Curl Request:**

| curl --location --request POST '{hostname}/iaas/api/public/v2/token' \  --header 'authorization: Basic <encoded\_key>' \  --header 'cache-control: no-cache' \  --header 'content-type: application/json' \  --header 'tenant:<tenant>' \  --header 'x-elastica-dbname-resolved: true' |
| --- |

## Activating GCP Connection

This will be used to onboard the GCP connection. If successful this API will make the connection as active in the cloudsoc console.

URL: {hostname}/iaas/api/public/v2/gcp\_connection

Method: POST

Headers:

X-Elastica-Dbname-Resolved:true

Content-Type:application/json

Authorization:Basic <valid\_token>

tenant:<tenant>

Body:

{

"gcp\_mode": "org",

"gcp\_target": "GCPOrg",

"gcp\_target\_id": "<gcp\_project\_id>",

"gcp\_target\_principal": "<gcp\_service\_account>",

"id": "<connection\_id>",

"mode": "multiple",

"name": "<connection\_name>",

"type": "gcp"

}

**Curl Request:**

| curl --location '{hostname}/iaas/api/public/v2/gcp\_connection' \  --header 'Authorization: Bearer {"<valid\_token>”}' \  --header 'X-Elastica-Dbname-Resolved: True' \  --header 'tenant:<tenant>'\  --header 'Content-Type: application/json' \  --data-raw '{  "gcp\_mode": "org",  "gcp\_target": "GCPOrg",  "gcp\_target\_id": "<gcp\_project\_id>",  "gcp\_target\_principal": "<gcp\_service\_account>",  "id": "<connection\_id>",  "mode": "multiple",  "name": "<connection\_name>",  "type": "gcp"  }' |
| --- |

## Validate GCP Connection

This is just a pre-validation before creating GCP resources.

URL: {hostname}/iaas/api/public/v2/connections/{connection\_id}/validate

Method: POST

Headers:

X-Elastica-Dbname-Resolved:true

Content-Type:application/json

Authorization:Basic <valid\_token>

tenant:tenant:<tenant>

Body:

{

"gcp\_mode": "org",

"gcp\_target": "GCPOrg",

"gcp\_target\_id": "<gcp\_project\_id>",

"gcp\_target\_principal": "<gcp\_service\_account>",

"id": "<connection\_id>",

"mode": "multiple",

"name": "<connection\_name>",

"type": "gcp"

}

**Curl Request:**

| curl --location '{hostname}}/iaas/api/public/v2/connections/{connection\_id}/validate' \  --header 'Authorization: Bearer {"<valid\_token>”}' \  --header 'X-Elastica-Dbname-Resolved: True' \  --header 'tenant:<tenant>'\  --header 'Content-Type: application/json' \  --data-raw '{  "gcp\_mode": "org",  "gcp\_target": "GCPOrg",  "gcp\_target\_id": "<gcp\_project\_id>",  "gcp\_target\_principal": "<gcp\_service\_account>",  "id": "<connection\_id>",  "mode": "multiple",  "name": "<connection\_name>",  "type": "gcp"  }' |
| --- |

## Status Code

| 200 / 202 | Successful operation. |
| --- | --- |
| 400 | Bad request. The request body is incorrect. |
| 401 | Unauthorized access. The token is either invalid or expired. |
| 403 | Forbidden.  The request was valid and the credentials were successfully authenticated. However, the credentials do not grant permission to access the resource. |
| 404 | URL is incorrect. |
| 429 | Too many requests. |
| 500 | Internal server error for any other issues |

### 