DCS IP SEGMENT MANAGER

Developed by:
GERALDEZ, Maureen
CALDEJON, Francine

Supervised by:
LIM, Genesis
GONZALES, Michael
GONZALES, Michael
SARITA, Arturo
GARCIA, Jerry

BACKGROUND

To solve cases or tickets regarding network issues, the DCS Team relies on the DCS IP Segments found in Confluence. They manually look for the IP Segment to know who the owner of the IP is or vice versa. Automation of this process will make the work of DCS Team faster and easier.

SCOPE AND LIMITATIONS

- Prior to the start of this project, the developers were assisting in another project because at that time, there was still no official project assigned to the interns. Afterwards, the project specifications were given to the interns.
- The developers started on the project around the beginning of July and were also given only until the end of July to finish the project which slightly affected the output.
- The project's scope includes:
    - Giving details from IP Segments page from Confluence
    - Categorization of the different Segments
    - Adding and Deleting IP Records

SYSTEM OVERVIEW

The DCS IP Segment Manager is a project made by the interns of Data Center Services Department of Trend Micro, June-July 2017. This system allows the DCS Team to look for IPs that are used inside Trend Micro. This system may search for IP addresses and will be able to determine the machine that uses that IP and which data center it is located. The IP Segments used are only the IP ranges currently in use. The unused IP ranges (DCS-managed) are not included in the system.

Aside from searching, the system also has a display function wherein it can either display all the IP segments, or display the segments for each category only. The categories are as follows:

- ELASTIC CLOUD DC
- SDI DC/Service POD
- OPENCLOUD AVAILABILITY ZONE
- Management IPs (includes DRAC, IPMI, and PDU IP)
- Global IP List - under the different network teams in Corp IT, DCS, and other local IT in different BU. This might be useful for us to trace the respective IP owners and avoid confusion.

SIGNIFICANCE OF THE PROJECT

Before, to check which datacenter the IP address belongs to, the DCS team would need to do a manual checking on the IP Segments page of Confluence. This can be a tedious task as there are many IP segments and there are cases wherein the IP segment is listed as a range (e.g. 150.70.68-71.x) which causes further checking if the IP is within that range. Once an IP segment match has been found, there is also the need to verify if the IP address can be found in the subnet (in the cases wherein the subnet of the IP segment is indicated).

With this project, there will be an automation of the search for IP addresses. The user would simply input the IP address, and the output would be on which IP segment it is a part of and in which datacenter it is located.

PROJECT RESOURCES

- **DATABASE:** PhpMyAdmin, MySQL Workbench 6.3 CE
- **SERVER:** Wamp Server
- **CODE EDITOR:** Sublime Text, Brackets
- **LANGUAGE:** PHP
  The primary language used in coding the project is PHP. However, Python was also used in developing the algorithms for easier coding and checking, and was then later converted to PHP. HTML and CSS was also used in developing the GUI of the system.
- **FRAMEWORK:** Bootstrap CSS
  Majority of the project design is based on Bootstrap. The design is based on the **dcsportal** page.

SCREENSHOTS

The first screen a user will encounter is the Home Page (Fig 1.0).

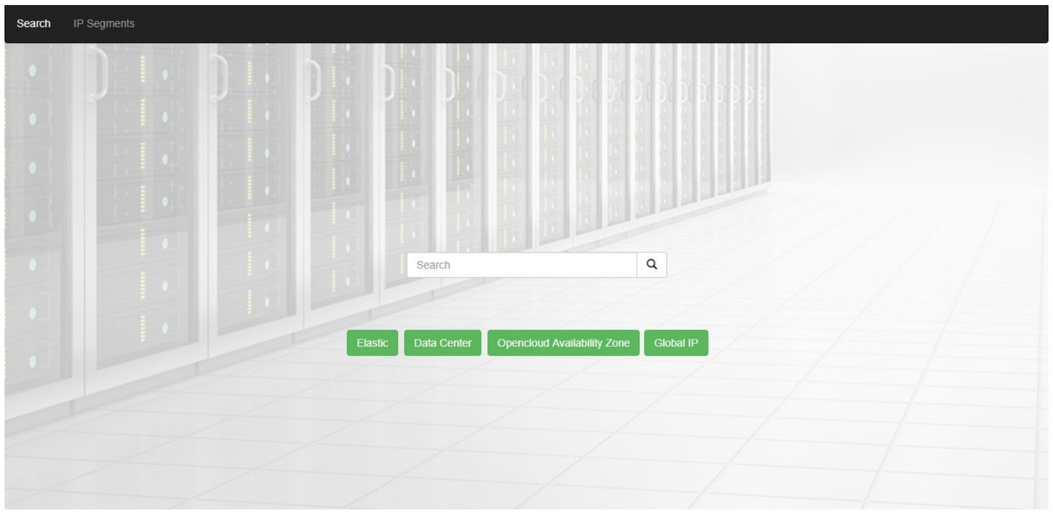


Figure 1.0 Home Page

Using the Navigation Bar located at the top of the page (Fig. 1.1), a user can go to either the IP Segments page or back to the Search Page.

Figure 1.1 NavBar

The main component of the system, which is the IP segment search engine, is located at the center of the homepage (Fig 1.2). Here, a user can either input an IP address to find out what IP segment is said IP address included, or a user can simply click on a button of a corresponding category to see all the IP segments associated with that category. If a user will click the Elastic button, all the IP segments of Elastic will be displayed on a new page (Fig 1.3).

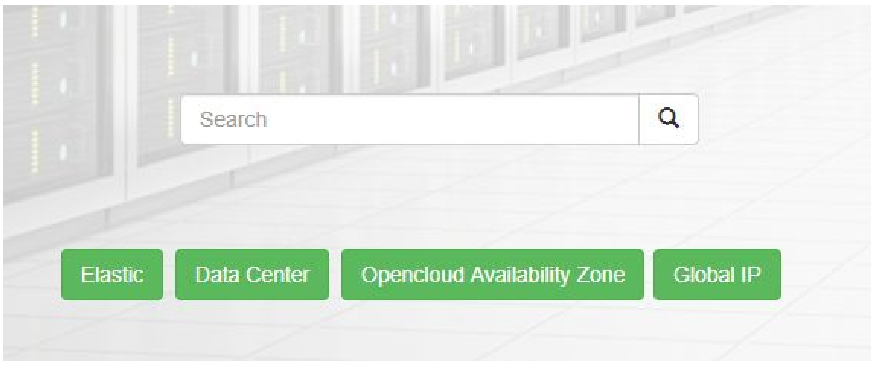
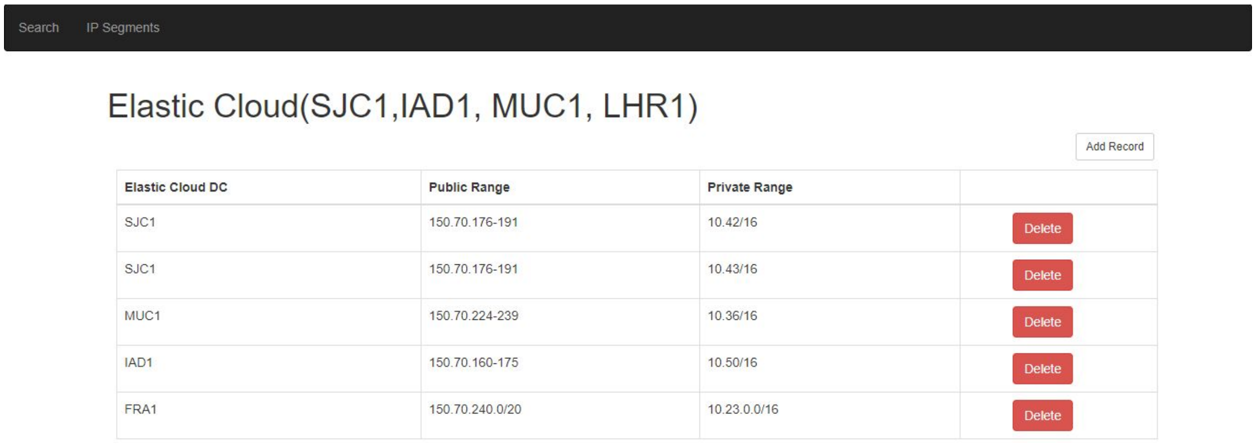


Figure 1.2 Search Bar and Buttons

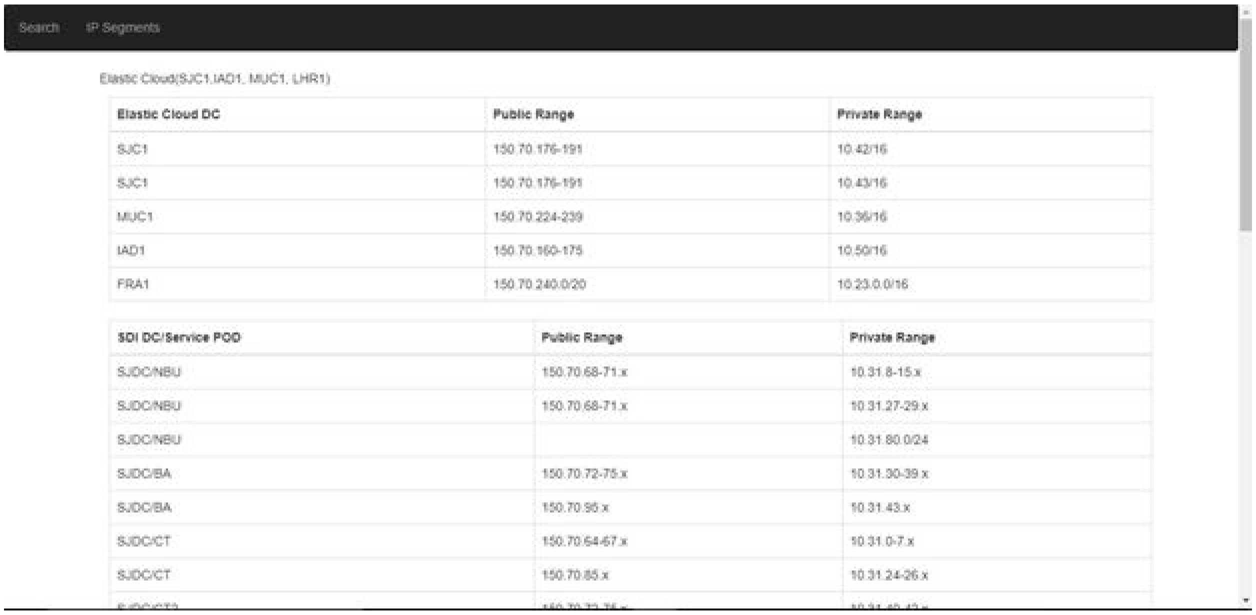


Elastic Cloud(SJC1,IAD1, MUC1, LHR1)

Elastic Cloud DC	Public Range	Private Range	
SJC1	150.70.176-191	10.42/16	Delete
SJC1	150.70.176-191	10.43/16	Delete
MUC1	150.70.224-239	10.36/16	Delete
IAD1	150.70.160-175	10.50/16	Delete
FRA1	150.70.240.0/20	10.23.0.0/16	Delete

Figure 1.3 Elastic IP Segments Page

Clicking 'IP Segments' on the Navigation Bar will redirect the user to the IP Segments page (Fig 1.4) where all of the IP Segments in all categories are displayed.



Elastic Cloud(SJC1,IAD1, MUC1, LHR1)

Elastic Cloud DC	Public Range	Private Range
SJC1	150.70.176-191	10.42/16
SJC1	150.70.176-191	10.43/16
MUC1	150.70.224-239	10.36/16
IAD1	150.70.160-175	10.50/16
FRA1	150.70.240.0/20	10.23.0.0/16

SDI DC/Service POD	Public Range	Private Range
SJDC/NBU	150.70.68-71.x	10.31.8-15.x
SJDC/NBU	150.70.68-71.x	10.31.27-29.x
SJDC/NBU		10.31.80.0/24
SJDC/BA	150.70.72-75.x	10.31.30-39.x
SJDC/BA	150.70.95.x	10.31.43.x
SJDC/CT	150.70.64-67.x	10.31.0-7.x
SJDC/CT	150.70.85.x	10.31.24-26.x

Figure 1.4 IP Segments Page

FUTURE WORK

Here are recommendations on how to continue and further improve the project:

- Include an 'Edit' function to the system (as the current system only has add and delete capabilities)
- Include an automated *nslookup* function to the system, wherein a user can input an ip address and the system will issue an *nslookup* command on the command line for said ip address and display the results on the screen
- Refinement/Improvement on the computation algorithm for the search function
- Indicate subnet masks for all IP segments
- Add a subnet mask input field for the search (recommended to do this when all the IP segments in the list has their subnet mask indicated)
- Separate the IP ranges (at least, on the Database side) to each individual IP address/Network address (i.e. expand 150.70.176-191 into 150.70.176, 150.70.177, 150.70.178, ... 150.70.191)
- Have error checking for user input
    - Check if valid IP address
        - Complete IP address (with 4 octets (dotted decimal form))
        - No value in each octet greater than 255
        - No value in each octet less than 0
        - No letters/special characters in each octet
        - Correct placement of "." (in between two octets), i.e. no consecutive "."
    - Check if valid subnet mask (recommended to do this when all there is an implemented subnet mask field in the search function)
        - Either dotted-decimal form (e.g. 255.255.255.0) or slash notation (/24) can be accepted
        - If dotted-decimal: make sure that the left octet is always greater than the right octet (e.g. 255.255.248.0 and NOT 255.248.0.255)
        - If slash notation: no value greater than /32
        - No value in octet greater than 255
        - No value in octet less than 0
        - No letters/special characters in each octet
        - Correct placement of "." (in between two octets), i.e. no consecutive "."
- Add more descriptions to the IP segments
    - Have a brief description about the different categories or the different IP Segments
    - Complete the IP segments list (i.e. some fields are empty and "(need to check)")
    - Have more clarification as some IP segments are ambiguous
        - E.g. 150.70.176-191
        - 150.70.176-191.**0**
        - Have clarification on how subnetting was done
            - 150.70.176.0/a, 150.70.177.0/b, 150.70.178.0/c, ..., 150.70.191.0/d
            - 150.70.176.0/20 (subnet that contains 150.70.176.0 - 150.70.191.254)