“Network Monitoring Engineer” Test Solution

Name: Muhammad Awais Ikram

Task1:

For the task 1, I have done it with 3 different approaches.
There are three code named as logic-1.sh, logic-2.sh, logic-3.sh available on GitHub. Link is here:

https://github.com/awaisikram/Task.git

Build Instructions:

To re-implement all logics user must use

- Linux environment (Ubuntu, Centos)
- Windows Bash
- Draw.io (to see flow diagram)

Usage:

User must save file on Linux environment and make the file executable using following command

```
# chmod +x “filename”
```

Logic 1:

Logic Explanation:

- Find a way to generate random numbers
- Find a way to generate random number in a specific range
- Try to store random numbers in an array
- Try to remove duplicate value from the array

Flow Diagram:

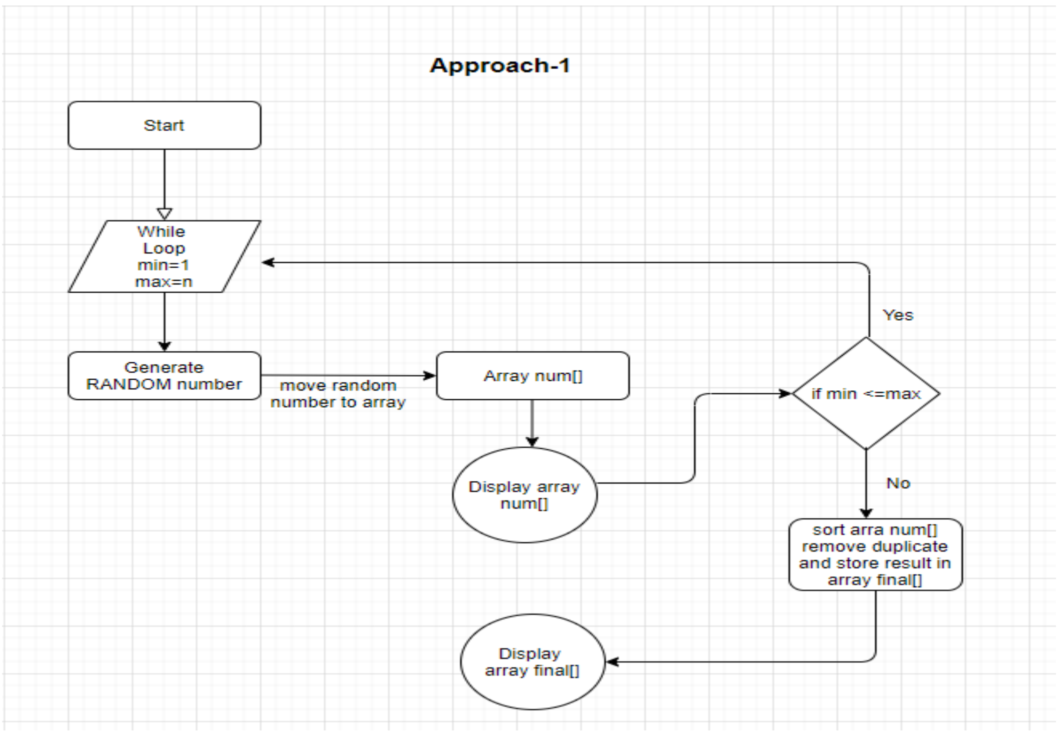


Figure 1: Flow chart approach 1

“Network Monitoring Engineer” Test Solution

Description:

The script **(logic-1.sh)** generate a random number within range of 1-10 in a loop and store that result of random generator to an array “num[]”. Range of loop is from [1-n], where “n” can be any integer value >=10. Loop will continue run until range “n” reached. At this point code will sort the array num[] and will remove duplicate from array num[] and final result will be stored in another array final[] to display.

Note: Run script multiple times to see different results.

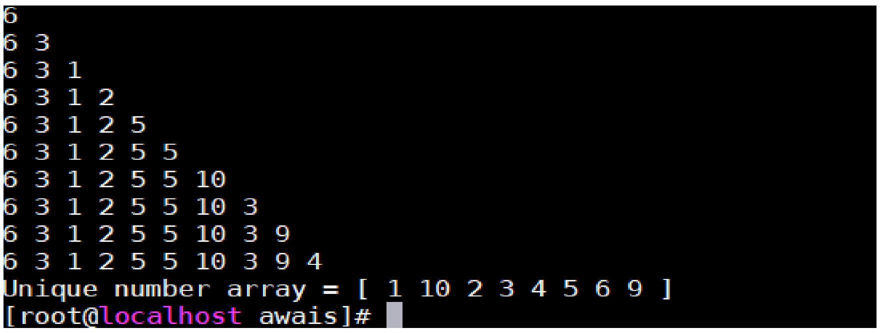
Limitations:

If we increase the upper limit of loop, there is more probability to have complete unique numbers from 1-10.

For example:

When loop range is 1-10, final array has 8 unique numbers (can vary) in between 1-10

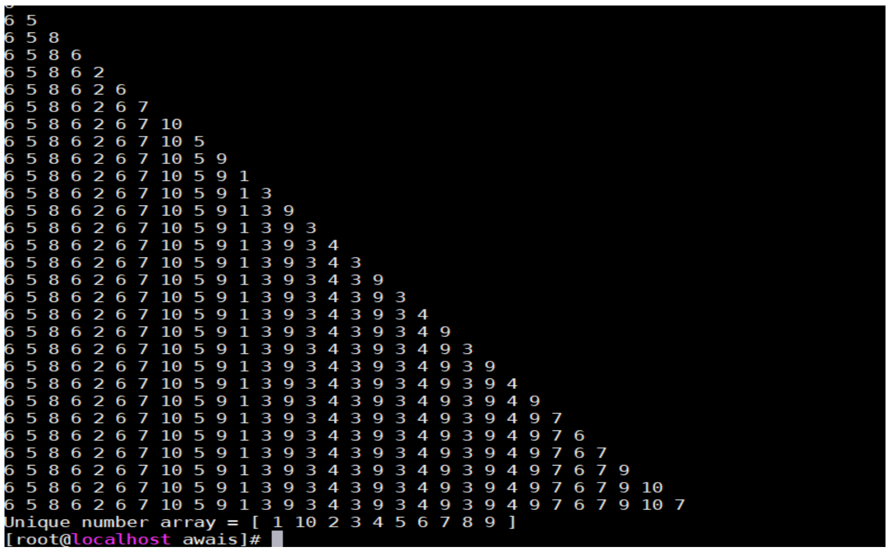
Results are shown here in Figure 2.



```
6
6 3
6 3 1
6 3 1 2
6 3 1 2 5
6 3 1 2 5 5
6 3 1 2 5 5 10
6 3 1 2 5 5 10 3
6 3 1 2 5 5 10 3 9
6 3 1 2 5 5 10 3 9 4
Unique number array = [ 1 10 2 3 4 5 6 9 ]
[root@localhost awais]#
```

Figure 2: Loop range 1-10

When loop range is 1-30, final array has 10 unique numbers in between 1-10(can vary) as shown in Figure 3.



```
6 5
6 5 8
6 5 8 6
6 5 8 6 2
6 5 8 6 2 6
6 5 8 6 2 6 7
6 5 8 6 2 6 7 10
6 5 8 6 2 6 7 10 5
6 5 8 6 2 6 7 10 5 9
6 5 8 6 2 6 7 10 5 9 1
6 5 8 6 2 6 7 10 5 9 1 3
6 5 8 6 2 6 7 10 5 9 1 3 9
6 5 8 6 2 6 7 10 5 9 1 3 9 3
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9 7
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9 7 6
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9 7 6 7
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9 7 6 7 9
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9 7 6 7 9 10
6 5 8 6 2 6 7 10 5 9 1 3 9 3 4 3 9 3 4 9 3 9 4 9 7 6 7 9 10 7
Unique number array = [ 1 10 2 3 4 5 6 7 8 9 ]
[root@localhost awais]#
```

Figure 3: Loop range 1-30

Logic 2:

Logic Explanation:

- Find a way to generate random number in a specific range
- Storing random numbers in an array
- If new random number is already present in array, discard it start next loop
- If new random number is not present in array, store that number to array
- Continue until array full of 1-10 unique numbers

Flow diagram:

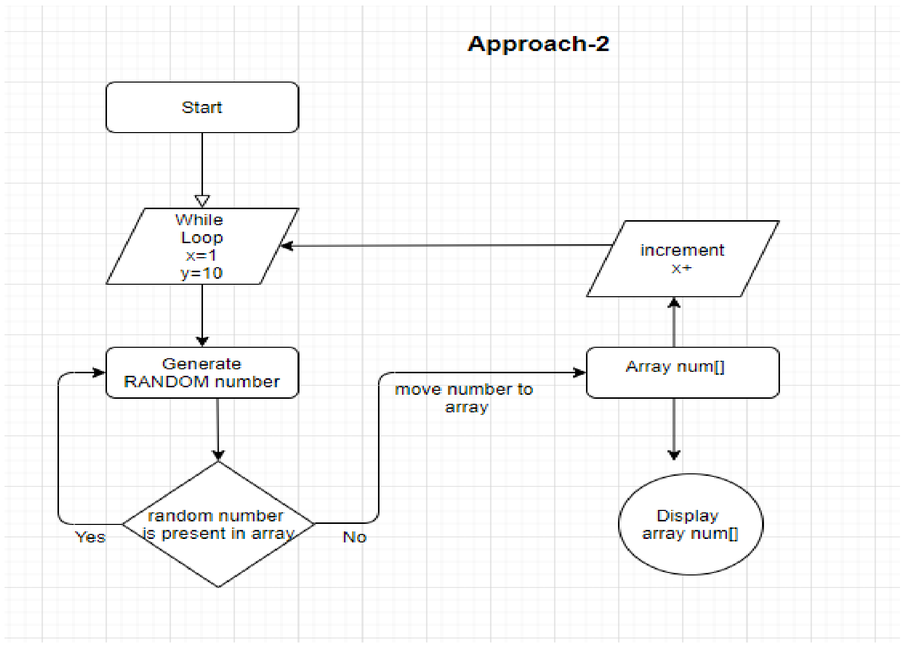


Figure 4: Flow chart approach 2

Description:

The script **(logic-2.sh)** will generate a random number inside of a loop. The range of loop is [1-10]. Conditions check, if generated random number is already present in the array. If it is already present, discard the number and re generate a new random number unless the condition become false and generated random number is not present in the array. Now increment the loop value and start the whole process again.

Note: Script will continue running, until the array has 10 unique numbers.

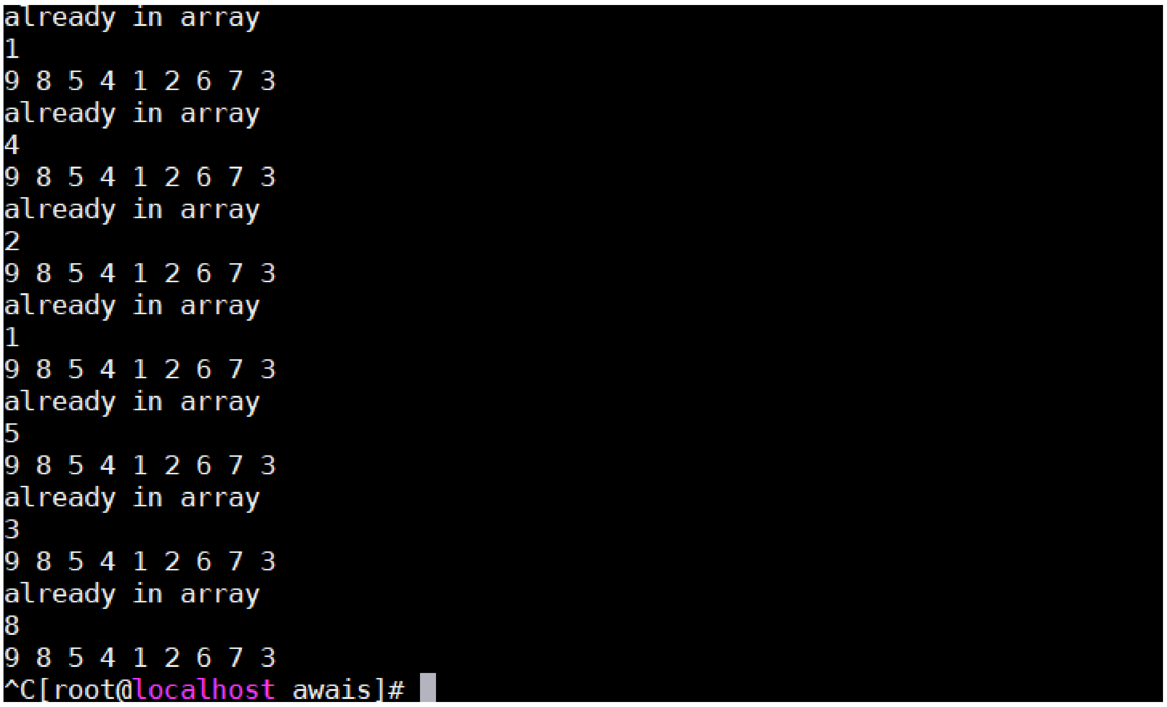
Limitation:

This script takes time to fill array with 10 unique numbers. This is because computer takes random number each time and if it that number is already present in array it will go back to generate new number until it finds a unique number which is not present in the array.

It is obvious that some time it can take very long time to fill whole array with unique numbers.

For example:

Here in the Figure 5, it is taking time to fill whole array with unique number



```
already in array
1
9 8 5 4 1 2 6 7 3
already in array
4
9 8 5 4 1 2 6 7 3
already in array
2
9 8 5 4 1 2 6 7 3
already in array
1
9 8 5 4 1 2 6 7 3
already in array
5
9 8 5 4 1 2 6 7 3
already in array
3
9 8 5 4 1 2 6 7 3
already in array
8
9 8 5 4 1 2 6 7 3
^C[root@localhost awais]#
```

Figure 5: approach 2 results

Logic 3:

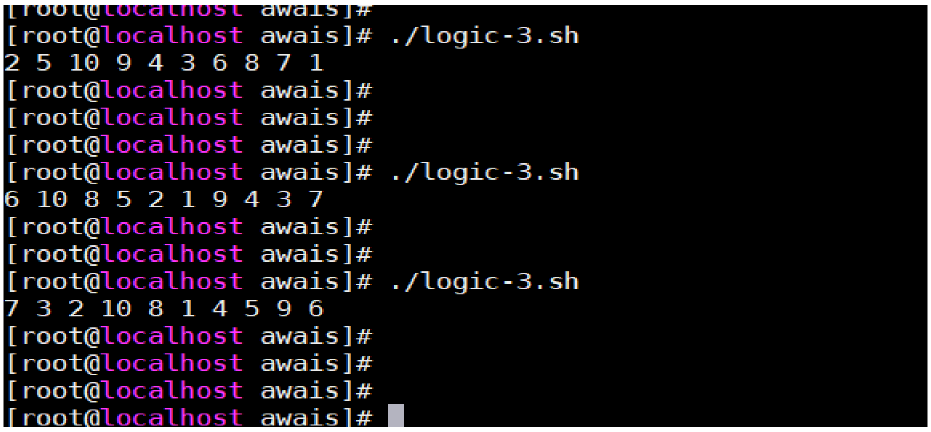
Description:

Another simple approach to generate random numbers is a specific range is also available in Linux.

“shuf” or shuffle is a built-in command in Linux environment which can create random numbers and can print unique numbers only.

Bash script **(logic-3.sh)** defines the range in which user want to generate random number and define the maximum number to display on the screen. The very interesting part is that if the range is let say 1-10 and user want to display only 5 numbers, then “shuf” command will only display 5 unique numbers in random order.

In the Figure 6, each time the script was executed, it has generated 10 unique numbers randomly.



```
[root@localhost awais]#
[root@localhost awais]# ./logic-3.sh
2 5 10 9 4 3 6 8 7 1
[root@localhost awais]#
[root@localhost awais]#
[root@localhost awais]#
[root@localhost awais]# ./logic-3.sh
6 10 8 5 2 1 9 4 3 7
[root@localhost awais]#
[root@localhost awais]#
[root@localhost awais]# ./logic-3.sh
7 3 2 10 8 1 4 5 9 6
[root@localhost awais]#
[root@localhost awais]#
[root@localhost awais]#
[root@localhost awais]#
```

Figure 6: approach 3 results

Task 2:

According to the task description, it is clear initially, that server is handling SSL offloading and it is handling 25000 transactions/requests per second.

I will try to explain my analyses regarding current situation.

Metric priority is mentioned with Number 1-4 defines which metric is at priority 1st to monitor according to the current scenario and so on. 1 = 1st priority 2=2nd priority 3= 3rd Priority 4=4th Priority.

CPU performance: (1)

As user need secure communication between client and the application server. For this secure communication, SSL is important to convert HTTP traffic to the HTTPS. But the process of sending secure data is not an easy as it looks. It requires lot of encryption and decryption of data on both communication side.

As in our task, server needs to perform SSL offloading and handle 25000 requests per second that means it must perform

- lot of encryption and decryption of data for large number of clients
- process in-coming request very fast for large number of clients

So according to this analysis, server should have a good CPU performance. If CPU performance of the server reaches to a critical point for example: above 80% then Network monitoring engineer must inform System administrator to react immediately.

RAM: (2)

For high processing tasks, server also need high RAM. In our case server is handling large number of requests per second and encryption/decryption of data, there should be enough RAM available to speed up the processing of in-coming request.

If server RAM is going to exceed a certain threshold for example above 85% or 90% RAM is utilised, then Network monitoring engineer must inform System administrator to perform necessary actions.

Network traffic: (3)

As mentioned in the task that server is handling proxies. That means there are large number of clients which are communicating with proxies and there is large network traffic between clients and proxy.

Server has 2 10Gbps network interface cards. There can be 2 cases to analyse this situation.

- If network traffic on any interface card is approaching to its maximum bandwidth
- There is no traffic on network

In both cases Network monitoring engineer must be informed System administrators to take immediate actions.

Hard disk: (4)

Large number of requests are incoming to the server that means we need to make logs of each incoming request if it is required or if there are other files need to be stored on server. To store logs and files we need Hard disk. If hard disk is approaching to its full storage capacity, Network monitoring engineer must inform System administrator to take necessary actions.

ZABBIX Monitoring:

To monitor all above metrics, ZABBIX monitoring solution is a good approach. Figure 7 shows a general overview of setup to monitor our application server using ZABBIX.

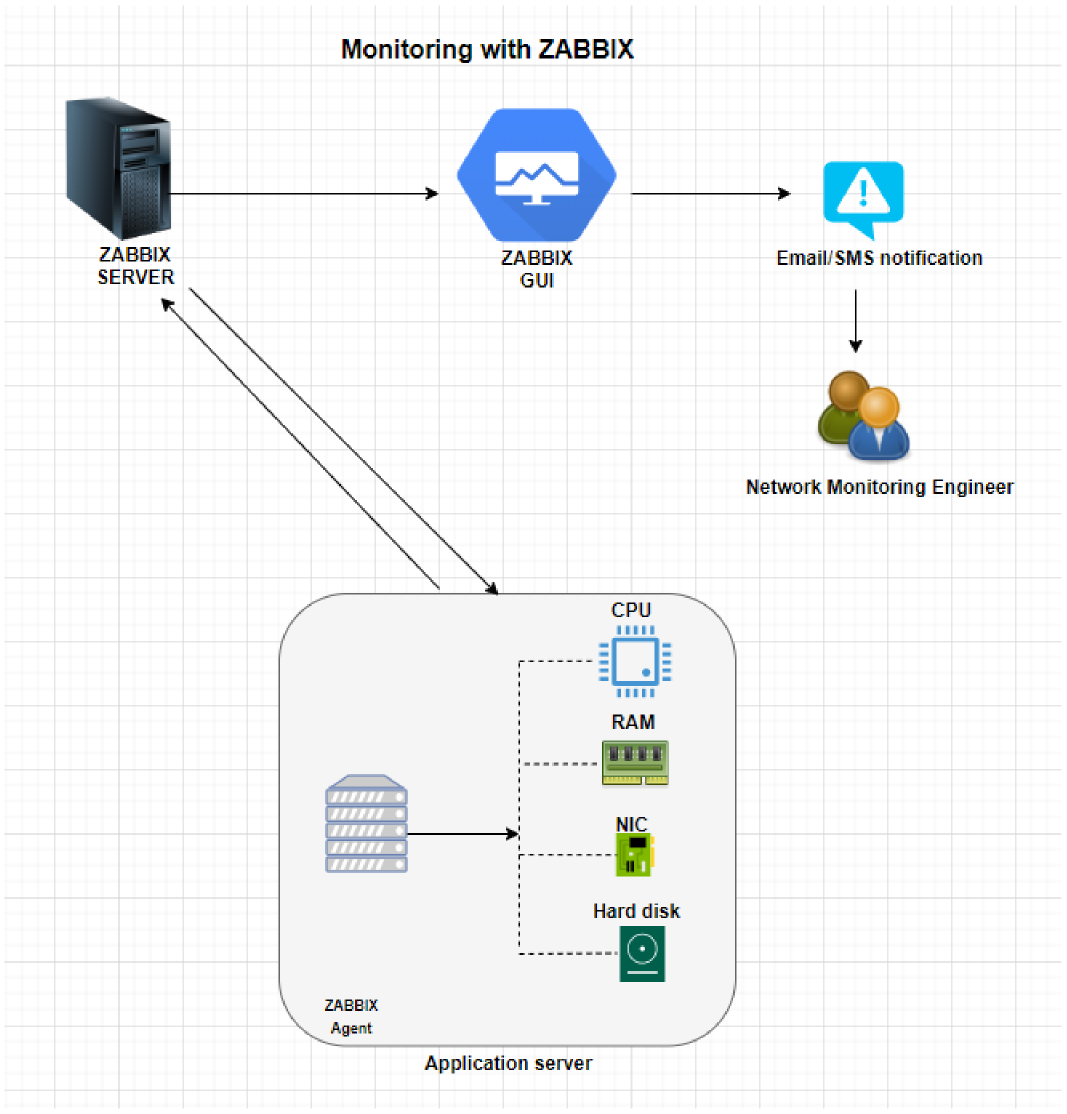


Figure 7: Application server Monitoring

Zabbix solution has two major components.

- Server
- Agent

User must install Zabbix agent into the application server which needs to be monitored. Zabbix agent will send monitoring metrics to the Zabbix server. In Zabbix GUI (graphical user interface), user must configure a host which will be associated with application server.

Items:

Every host has many items. Each item is associated with a specific kind of metric and get information related to that specific metric. For example. Network monitoring engineer require four items each for CPU, RAM, network traffic and hard disk space.

Triggers:

Network monitoring engineer must create triggers for each item explained above. Triggers allows network monitoring engineer to define critical limits of each metric. If any metric approaches to that critical limit, Zabbix will create an alert according to the severity of the incident.

Notifications:

Zabbix has a capability to send notification of alerts to System administrators or Network engineers. In case of an alert, Network monitoring engineer will get notification via Email/SMS according to which notification system is integrated with Zabbix.

Monitoring Challenges:

CPU, RAM, network traffic and hard disk are important metrics which must be monitored for every kind of server.

There are some challenges which should also consider.

- In current situation, Zabbix is handling only one server. What if there are thousands of servers and Zabbix needs to monitor all of them. In that case Zabbix server itself performance will affect because of lot of metric values needs to be processed by Zabbix server

- Zabbix solution support the notification system. If any host (server) has an issue with any metric it can immediately generate an alert and notify Network monitoring engineer, but what if Zabbix server itself is down or has a problem. In that case whole notification system will be stopped and Network Monitoring engineers will not have any information of Zabbix server itself problem.