Monitoring les serveurs avec zabbix

Département des Mathématiques et Informatique: Réseaux et Cloud Computing

Réalisé par: Sara BEN CHIKH

Safaa HALLI

Encadrant: Noureddine ABGHOUR

Khalid MOUSSAID

Année académique : 2020/2021

Table des matières

Les prérequis pour implémenter zabbix

1- installer VMWare

2- Crée une machine virtuelle nommée “Zabbix-server” avec le système d’exploitation « centos 8» avec l’utilisateur « Zabbix » et le mot de passe « Ok123456 »

3- Choisir l’option «NAT » pour les parametres de la carte reseau dans l’onglet VM ➔setting… ➔Network Adapter

4- Configurer l’adressage de la machine par : « systemctl restart NetworkmManager »
« nmcli connection uo ens33 »

5- Mise a jour le système avec la commande « yum update »

6- Installet l’outil « yum-config-manager » avec la commande « yum install yum-utils »

7- Désactiver selinux
« Setenforce 0 »
« vi /etc/selinux/config
SELINUX=disabled

Installation de zabbix-server

1) Installer zabbix server, agent

dnf install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-sql-scripts zabbix-agent

2) Installer et configurer mariadb

a. Installer la base de donnée «mariadb»

yum install -y httpd mariadb-server php php-cli php-common php-mbstring php-mysqlnd php-xml php-bcmath php-devel php-pear php-gd

b. Démarrer et activer le service

sudo systemctl enable mariadb.service
sudo systemctl start mariadb.service

c. Vérifier le status

sudo systemctl status mariadb.service

d. Executer la commande

« sudo mysql_secure_installation » -> y -> 2 -> password -> re enter password -> y -> y -> y -> y -> y

e. Se connecter a mysql avec la commande

« mysql -u root –p »

f. Créer une base de données

create database zabbix character set utf8 collate utf8_bin;

g. Créer un utlisateur

create user zabbix@localhost identified by 'P@ssw0rdMari@DB';

h. Autorisé des privilèges à l'utilisateur

grant all privileges on zabbix.* to zabbix@localhost;

quit ;

i. Importer le schéma initial et les données

zcat /usr/share/doc/zabbix-sql-scripts/mysql/create.sql.gz | mysql -uzabbix -p zabbix

3) Modifier dans le fichier de configuration de zabbix « /etc/zabbix/zabbix_server.conf »

DBPassword=M@riaDB2021

4) Modifier le fichier de page web

«vi /etc/php.ini»

memory_limit 256M

upload_max_filesize 16M

post_max_size 16M

max_execution_time 300

max_input_time 300

max_input_vars 10000

date.timezone = Maroc/Casabanca

4) Démarrer et activer les services

systemctl restart zabbix-server zabbix-agent httpd php-fpm

systemctl enable zabbix-server zabbix-agent httpd php-fpm

systemctl start httpd

systemctl start mariadb

systemctl enable httpd

systemctl enable mariadb

Installation de zabbix-agent

Sur le client zabbix

Linux

1)Téléchargement de package d'installation depuis le repo

https://repo.zabbix.com/zabbix/5.4/rhel/8/x86_64/

2) Déplacement du fichier vers le répertoire de la machine

Ouvrir winscp -> glisser le package vers le répértoire /install

3) Installation du package

Installer le package avec

sudo rpm -ivh zabbix-agent2-5.4.1-1.el7.x86_64.rpm

4)Modification du fichier de configuration

Ouvrir le fichier de configuration avec la commande:

sudo vi /etc/zabbix/zabbix_agent2.conf

a. Ajouter l'@ serveur dans le fichier de configuration dans le paramètre Serveur=

```
### Option: Server
#       List of comma delimited IP addresses, optionally in CIDR notation, or DN
S names of Zabbix servers and Zabbix proxies.
#       Incoming connections will be accepted only from the hosts listed here.
#       If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.
0.0.1' are treated equally
#       and '::/0' will allow any IPv4 or IPv6 address.
#       '0.0.0.0/0' can be used to allow any IPv4 address.
#       Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.exampl
e.com
#
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:
# Server=

Server=192.168.199.134
```

b. Modifier les options : ServerActive et Hostname

```
### Option: ServerActive
#       List of comma delimited IP:port (or DNS name:port) pairs of Zabbix serve
rs and Zabbix proxies for active checks.
#       If port is not specified, default port is used.
#       IPv6 addresses must be enclosed in square brackets if port for that host
 is specified.
#       If port is not specified, square brackets for IPv6 addresses are optiona
l.
#       If this parameter is not specified, active checks are disabled.
#       Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12f
c::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=192.168.199.134:10051
```

```
### Option: Hostname
#       List of comma delimited unique, case sensitive hostnames.
#       Required for active checks and must match hostnames as configured on the
 server.
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

Hostname=zabbixserver
```

Redémarrer le service zabbix-agent2 avec la commande « systemctl restart zabbix-agent2 » et vérifier si le pare-feu est activé ou non

Windows

1) Télécharger l'agent soit le MSI soit le précompiler à partir du site https://www.zabbix.com/fr/download_agents Et l'installer
2) MSI : suivre les étapes d'installation

3) Précompiler :c:\zabbix\bin\zabbix_agent2.exe –c c:\zabbix\conf\zabbix_agent2.conf -i

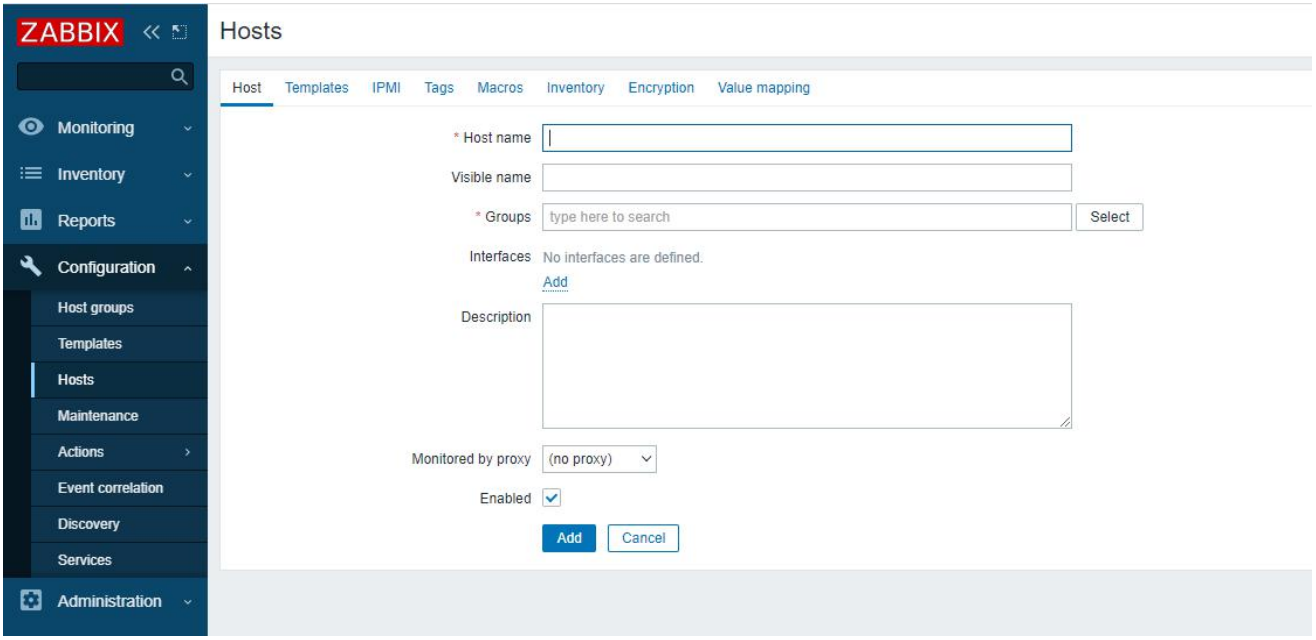
Et On choisit la Template approprié soit « Linux by Zabbix agent » ou « Windows by Zabbix agent » ou « Aix»

Sur le serveur Zabbix

Création d'hôte

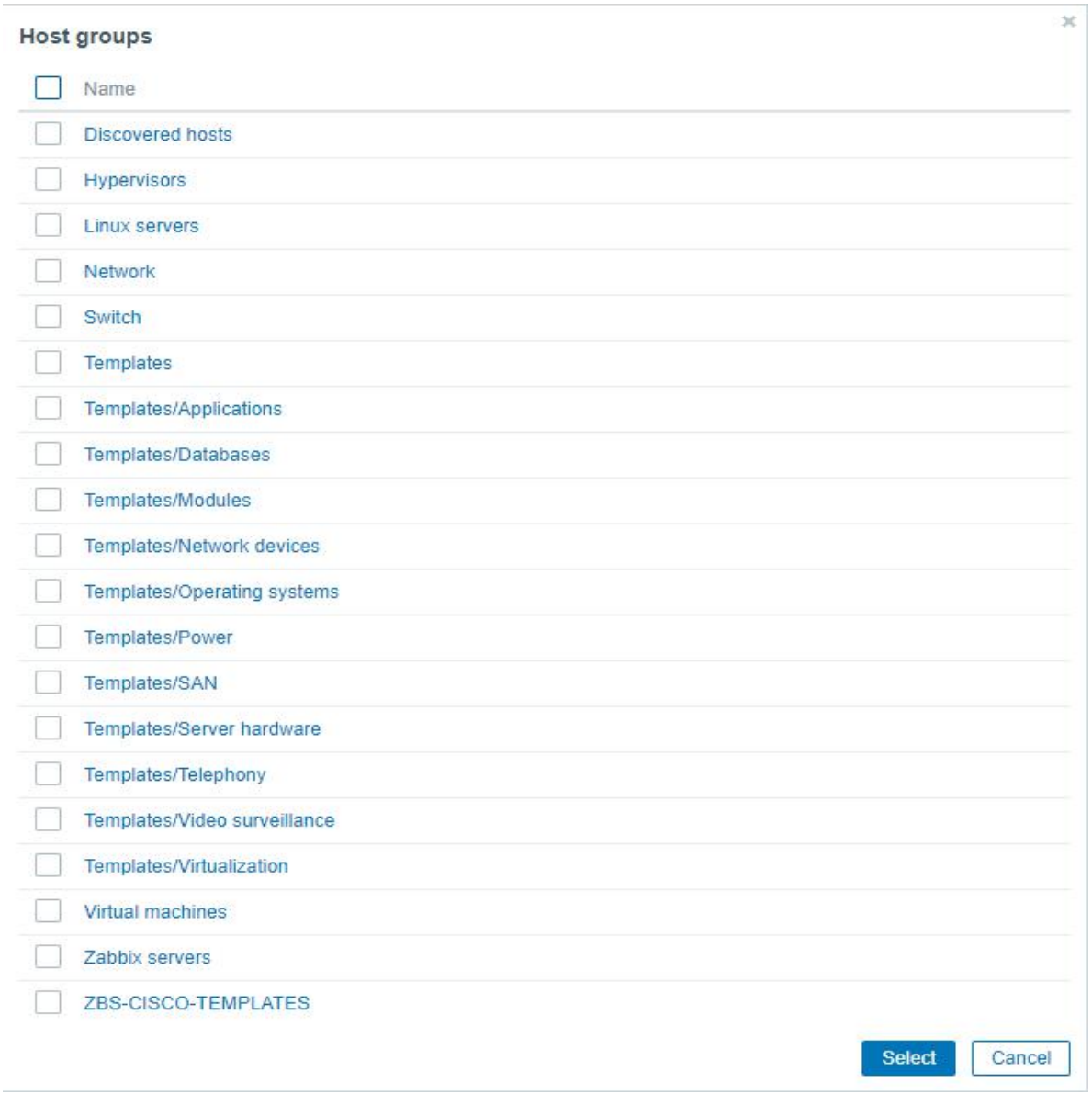
Dans l'onglet « Configuration » -> « Hosts » appuyer sur « create host » en haut à droite de la page

Dans cette fenêtre :

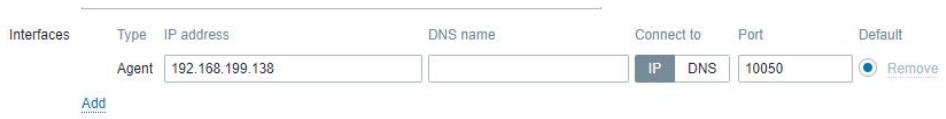


On ajoute le nom de la machine, le groupe approprié

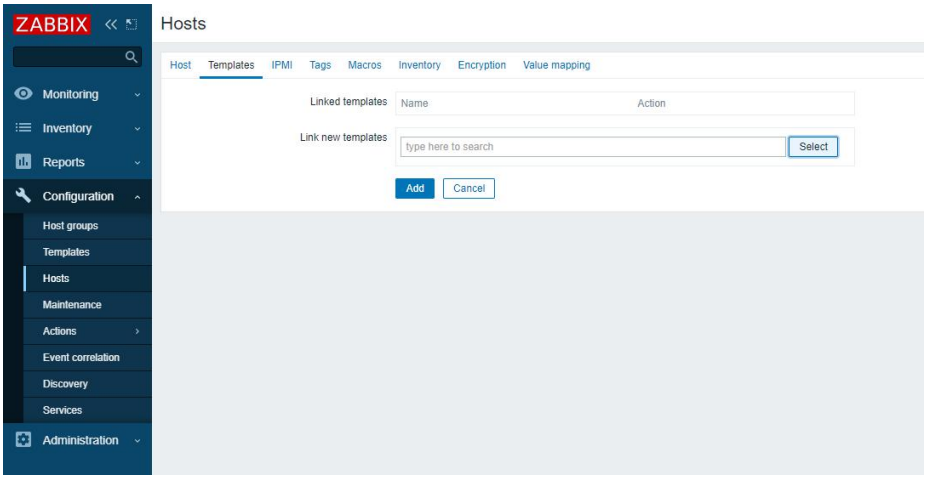
Dans notre cas c'est « templates/Operating systems »



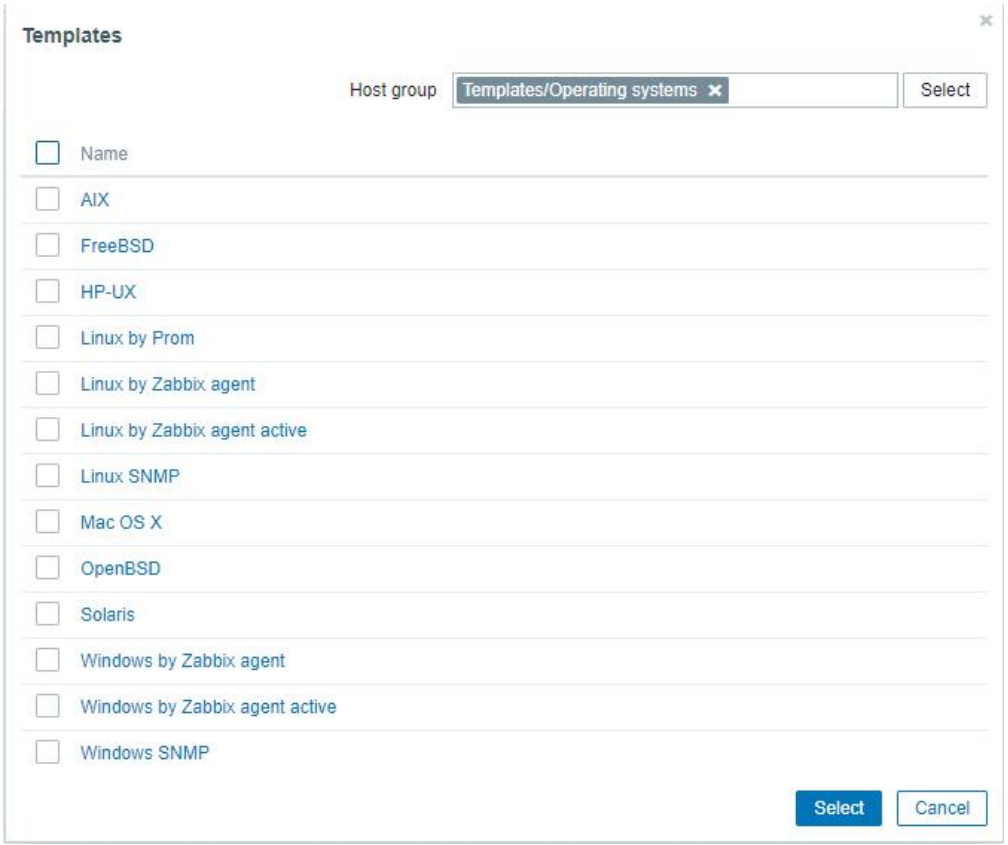
Et l@ de la machine dans interfaces



Après dans l'onglet « templates » on appuie sur select



Et On choisit la template approprié dans notre cas c'est « Linux by Zabbix agent »



On appuie sur « select » puis « add »

Configuration de l'agent zabbix active :

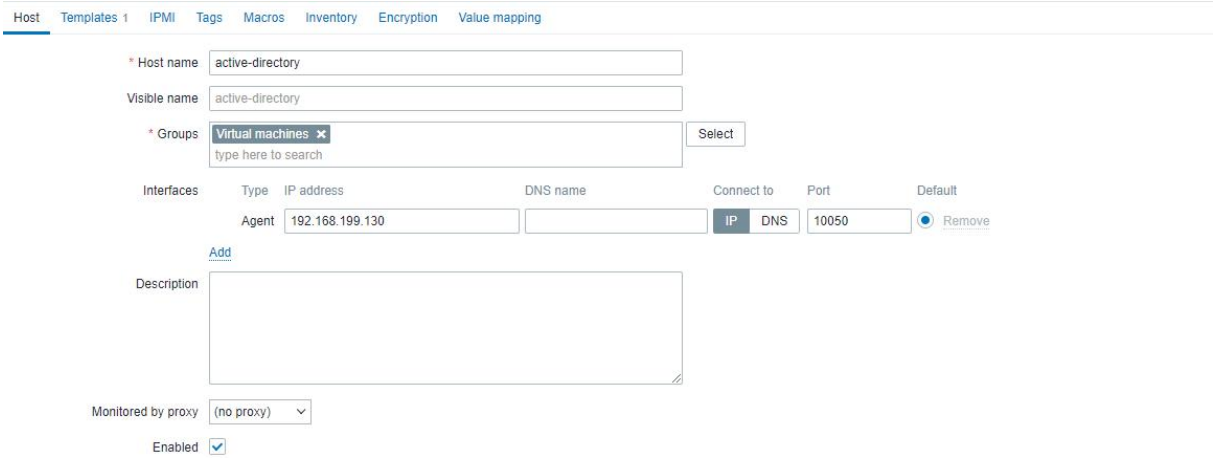
Modifier dans fichier de configuration zabbix-agent2.conf

```
ListenPort=10050
ServerActive=Ip-adresse-du-serveur:10051
Hostname=nom exacte de la machine
```

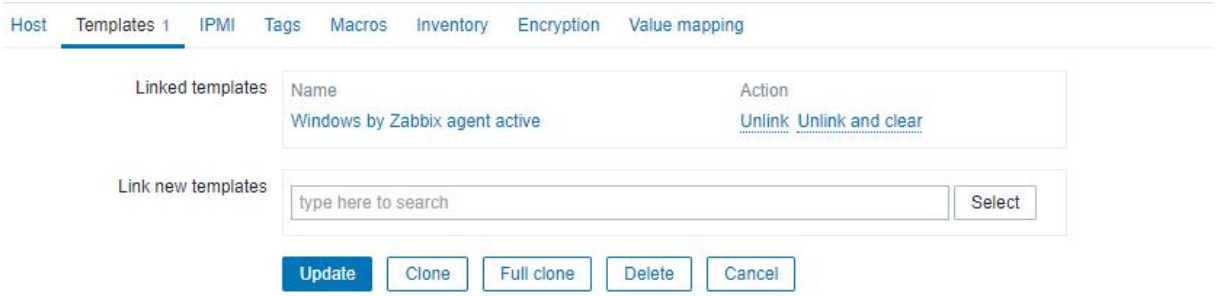
Puis on ajoute la machine dans l'interface graphique de server par:

Dans l'onglet « Configuration » -> « Hosts » appuyer sur « create host » en haut à droite de la page

Dans cette fenêtre :



Et ajouter la template «windows by zabbix agent active»



Puis appuyer sur «add»

Monitoring Tomcat

Sur le serveur Zabbix :

1) l'installation du package

on installe le package zabbix-java-gateway avec la commande :

yum install zabbix-java-gateway »

2) modification des fichiers de configuration

on modifie dans le fichier de configuration

vim /etc/zabbix/zabbix_server.conf

On ajoute l'@ de bouclage « 127.0.0.1 » dans l'option « JavaGateway »

```
### Option: JavaGateway
#       IP address (or hostname) of Zabbix Java gateway.
#       Only required if Java pollers are started.
#
# Mandatory: no
# Default:
JavaGateway=127.0.0.1
```

On ajoute le port « 10052 » dans l'option « JavaGatewayPort »

```
### Option: JavaGatewayPort
#       Port that Zabbix Java gateway listens on.
#
# Mandatory: no
# Range: 1024-32767
# Default:
JavaGatewayPort=10052
```

On ajoute la valeur « 5 » dans l'option « JavaPollers »

```
### Option: StartJavaPollers
#       Number of pre-forked instances of Java pollers.
#
# Mandatory: no
# Range: 0-1000
# Default:
StartJavaPollers=5
```

Après on ajoute dans le fichier « zabbix_java_gateway.conf »

L'option suivante :

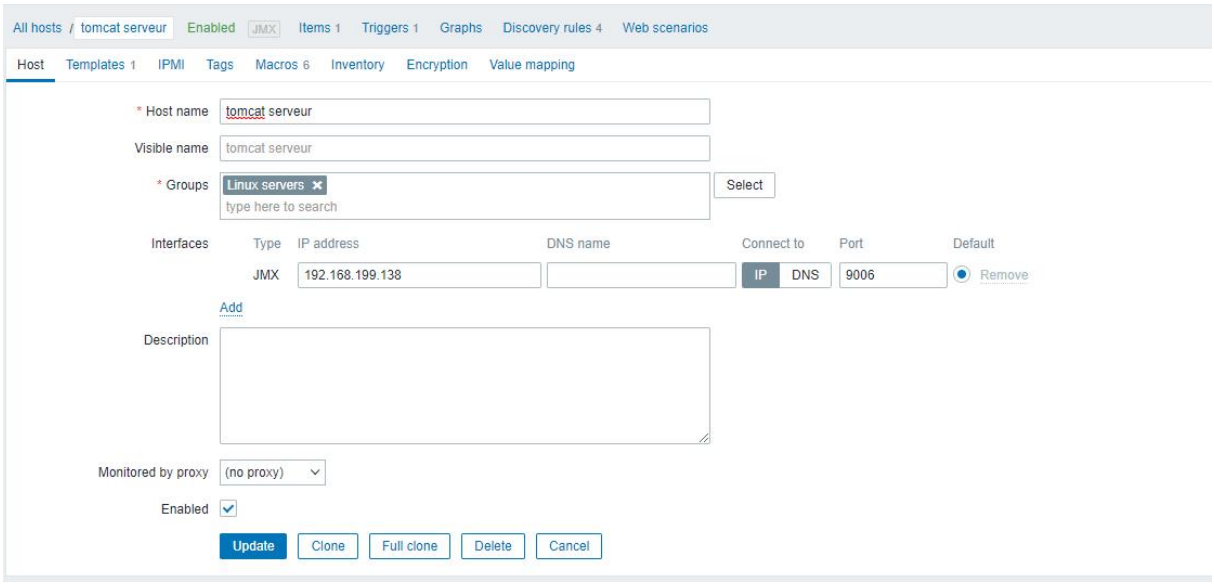
```
JAVA_OPTIONS="$JAVA_OPTIONS
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9006
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.registry.ssl=false"
-Dcom.sun.management.jmxremote.rmi.port=9006
-Djava.rmi.server.hostname=127.0.0.1
-Dcom.sun.management.jmxremote=9006
```

```
JAVA_OPTIONS="$JAVA_OPTIONS
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9006
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.registry.ssl=false"
-Dcom.sun.management.jmxremote.rmi.port=9006
-Djava.rmi.server.hostname=127.0.0.1
-Dcom.sun.management.jmxremote=9006
```

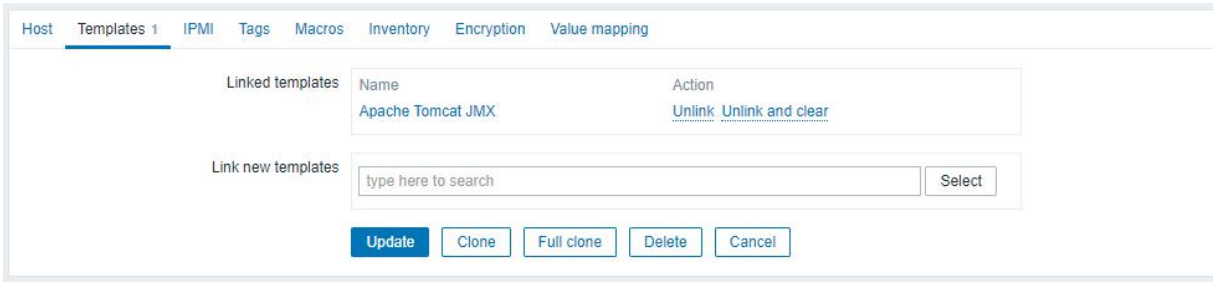
3) l'ajoute de serveur Tomcat :

Sur la console web :

Dans l'onglet « Configuration » -> « Hosts » on sélectionne le serveur dont Tomcat est installer et on ajoute une interface jmx avec @ du serveur et le numéro de port dans notre cas c'est 9006



Après dans l'onglet Templates on ajoute la Template « Template App Apache Tomcat jmx »



Et en appuie sur Update

Si la Template n'existe pas, on importe le fichier nommé « Template App Apache Tomcat jmx.yaml » qui se trouve en pièce jointe

Monitoring IBM STORWIZE

1) Importer les deux Templates
   « Storwize_HEALTH_STATUS.xml » et « Template IBM-Storwize-3700 CLI v2.xml »

Lier les Templates avec l'hote avec l'ajout d'une interface snmp .

2) Remplir les champs des macros
3) créer le script

nommer le script « storwize_get_state.py » et l'enregister dans /usr/lib/zabbix/externalscripts

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-

import os
import time
import argparse
import sys
import json
import subprocess
import paramiko
import logging
import logging.handlers
import csv
import re


# Create log-object
```

```
LOG_FILENAME = "/tmp/storwize_state.log"
# sys.argv[5] contain this string "--
storage_name=<storage_name_in_zabbix>". List slicing
delete this part "--storage
_name="
STORAGE_NAME = sys.argv[5][15:]
storwize_logger = logging.getLogger("storwize_logger")
storwize_logger.setLevel(logging.INFO)

# Set handler
storwize_handler = logging.handlers.RotatingFileHandler(
    LOG_FILENAME, maxBytes=(1024**2)*10,
backupCount=5)
storwize_formatter = logging.Formatter(
    '{0} - %(asctime)s - %(name)s - %(levelname)s -
%(message)s'.format(STORAGE_NAME))

# Set formatter for handler
storwize_handler.setFormatter(storwize_formatter)

# Add handler to log-object
storwize_logger.addHandler(storwize_handler)


def storwize_connect(storwize_user, storwize_password,
storwize_ip, storwize_port):
    try:
        ssh_client = paramiko.SSHClient()

ssh_client.set_missing_host_key_policy(paramiko.AutoAddP
olicy())
        ssh_client.connect(hostname=storwize_ip,
```

```
username=storwize_user,
                        password=storwize_password, port=22)
        storwize_logger.info("Connection Established
Successfully")
        return ssh_client
    except Exception as oops:
        storwize_logger.info("Connection Close Error Occurs:
{0}".format(oops))
        sys.exit("1000")


def storwize_logout(ssh_client):
    try:
        ssh_client.close()
        storwize_logger.info("Connection Closed Successfully")
    except Exception as oops:
        storwize_logger.info(
            "Connection Close  Error Occurs: {0}".format(oops))
        sys.exit("1000")


def convert_to_zabbix_json(data):
    output = json.dumps({"data": data}, indent=None,
separators=(',', ': '))
    return output


def convert_text_to_numeric(value):
    if value == 'online':
        numericValue = 0
    elif value == 'offline':
        numericValue = 1
```

```
    elif value == 'degraded':
        numericValue = 2
    elif value == 'active':
        numericValue = 3
    elif value == 'inactive_configured':
        numericValue = 4
    elif value == 'inactive_unconfigured':
        numericValue = 5
    elif value == 'offline_unconfigured':
        numericValue = 6
    elif value == 'excluded':
        numericValue = 7
    elif value == 'on':
        numericValue = 8
    elif value == 'off':
        numericValue = 9
    elif value == 'slow_flashing':
        numericValue = 10
    elif value == 'degraded_paths':
        numericValue = 11
    elif value == 'degraded_ports':
        numericValue = 12
    else:
        numericValue = 100

    return numericValue


def advanced_info_of_resource(resource, needed_attributes,
storwize_connection, *id_of_resource):
    """ needed_attributes - list of parameters, that we wont to
get
```

```
        id_of_resource - list of additional parameters, that
uniquely determine resource.
        Example: for PSU - first element of list is enclosure_id,
secondary element of list is PSU_id"""

    if resource == 'lsenclosure':
        stdin, stdout, stderr =
storwize_connection.exec_command(
            'svcinfo {0} {1}'.format(resource, id_of_resource[0]))
    elif resource == 'lsenclosurepsu':
        stdin, stdout, stderr =
storwize_connection.exec_command(
            'svcinfo {0} -psu {1} {2}'.format(resource,
id_of_resource[1], id_of_resource[0]))

    if len(stderr.read()) > 0:
        storwize_logger.info(
            "Error Occurs in advanced info of enclosure -
{0}".format(stderr.read()))
        storwize_logout(storwize_connection)
        sys.exit("1100")
    else:
        # Получили расширенные атрибуты в виде строки
(variable contain advanced attributes in string)
        attributes_of_resource = stdout.read()
        # Здесь будут храниться расширенные атрибуты
ресурса в формате ключ-значение (will contain advanced
attribute
s in key-value)
        dict_of_attributes = {}
        try:
            # Разделил строку и получили список из
```

```
расшренные атрибутов
            for attribute in attributes_of_resource.split('\n'):
                if len(attribute) > 0:
                    temp = attribute.split(' ')
                    dict_of_attributes[temp[0]] = temp[1]
        except Exception as oops:
            storwize_logger.error(
                "Error occures in function
advanced_info_of_resource - {0}".format(oops))
            storwize_logout(storwize_connection)
            sys.exit("1100")

    # Создаем словарь из необходимых нам свойств ресурса
(create dictionary that contain properies of resource)
    result = {}
    for each_value in needed_attributes:
        result[each_value] = dict_of_attributes[each_value]

    return result


def convert_capacity_to_bytes(capacity_in_string):
    """ Конвертирует значение, которое отдает СХД в виде
строки, в байты
    Convert value, from string to byte, that get from storage
device
"""

    convert_to_bytes = {'TB': 1024**4,
                        'GB': 1024**3, 'MB': 1024**2, 'KB': 1024}
    try:
        # Ищем по регулярному выражению и находим две
```

```
группы совпадения
        list_of_capacity = re.search('([\d\.]+)([\D]+)',
capacity_in_string)
        converted_capacity = float(list_of_capacity.group(
            1)) * convert_to_bytes[list_of_capacity.group(2)]
        # Конвертация в целые числа, потому что для float в
заббиксе есть ограничение (convert to type ineger)
        return int(converted_capacity)
    except Exception as oops:
        storwize_logger.error(
            "Error occurs in converting capactity_in_string to
capactiy_in_bytes".format(oops))


def send_data_to_zabbix(zabbix_data, storage_name):
    sender_command = "/usr/bin/zabbix_sender"
    config_path = "/etc/zabbix/zabbix_agentd.conf"
    time_of_create_file = int(time.time())
    temp_file = "/tmp/{0}_{1}.tmp".format(storage_name,
time_of_create_file)

    with open(temp_file, "w") as f:
        f.write("")
        f.write("\n".join(zabbix_data))

    send_code = subprocess.call([sender_command, "-vv", "-c",
config_path, "-s", storage_name,
                                 "-T", "-i", temp_file],
stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    os.remove(temp_file)
    return send_code
```

```
def discovering_resources(storwize_user, storwize_password,
storwize_ip, storwize_port, storage_name,
list_resources)
                                    :
    storwize_connection = storwize_connect(
        storwize_user, storwize_password, storwize_ip,
storwize_port)

    xer = []
    try:
        for resource in list_resources:
            stdin, stdout, stderr =
storwize_connection.exec_command(
                'svcinfo {0} -delim :'.format(resource))

            if len(stderr.read()) > 0:  # Если случились ошибки,
запиши их в лог и выйди из скрипта (If errors occur,
than write them to log and correctyl end of ssh-session)
                storwize_logger.info(
                    "Error Occurs in SSH Command -
{0}".format(stderr.read()))
                storwize_logout(storwize_connection)
                sys.exit("1100")
            else:
                resource_in_csv = csv.DictReader(
                    stdout, delimiter=':')  # Create CSV

                discovered_resource = []
                storwize_logger.info(
                    "Starting discovering resource -
{0}".format(resource))
```

```
            for one_object in resource_in_csv:
                if ['lsvdisk', 'lsmdisk',
'lsmdiskgrp'].count(resource) == 1:
                    one_object_list = {}
                    one_object_list["{#ID}"] = one_object["id"]
                    one_object_list["{#NAME}"] =
one_object["name"]
                    discovered_resource.append(one_object_list)
                elif ['lsenclosurebattery', 'lsenclosurepsu',
'lsenclosurecanister'].count(resource) == 1:
                    one_object_list = {}
                    one_object_list["{#ENCLOSURE_ID}"] =
one_object["enclosure_id"]
                    if resource == 'lsenclosurebattery':
                        one_object_list["{#BATTERY_ID}"] =
one_object["battery_id"]
                    if resource == 'lsenclosurepsu':
                        one_object_list["{#PSU_ID}"] =
one_object["PSU_id"]
                    if resource == 'lsenclosurecanister':
                        one_object_list["{#CANISTER_ID}"] =
one_object["canister_id"]
                    discovered_resource.append(one_object_list)
                elif ['lsportfc', 'lsportsas'].count(resource) == 1:
                    one_object_list = {}
                    one_object_list["{#PORT_ID}"] =
one_object["port_id"]
                    one_object_list["{#NODE_NAME}"] =
one_object["node_name"]
                    discovered_resource.append(one_object_list)
                elif ['lsenclosure'].count(resource) == 1:
                    one_object_list = {}
```

```
                    one_object_list["{#ID}"] = one_object["id"]
                    one_object_list["{#MTM}"] =
one_object["product_MTM"]
                    one_object_list["{#SERIAL_NUMBER}"] =
one_object["serial_number"]
                    discovered_resource.append(one_object_list)
                elif ['lsdrive'].count(resource) == 1:
                    one_object_list = {}
                    one_object_list["{#ENCLOSURE_ID}"] =
one_object["enclosure_id"]
                    one_object_list["{#SLOT_ID}"] =
one_object["slot_id"]
                    discovered_resource.append(one_object_list)
                else:
                    one_object_list = {}
                    one_object_list["{#ID}"] = one_object["id"]
                    one_object_list["{#ENCLOSURE_ID}"] =
one_object["enclosure_id"]
                    discovered_resource.append(one_object_list)

            storwize_logger.info(
                "Succes get resource - {0}".format(resource))

            converted_resource = convert_to_zabbix_json(
                discovered_resource)
            timestampnow = int(time.time())
            xer.append("%s %s %s %s" % (storage_name,
resource,
                       timestampnow, converted_resource))
    except Exception as oops:
        storwize_logger.error("Error occurs in discovering -
{0}".format(oops))
```

```
        storwize_logout(storwize_connection)
        sys.exit("1100")

    storwize_logout(storwize_connection)
    return send_data_to_zabbix(xer, storage_name)


def get_status_resources(storwize_user, storwize_password,
storwize_ip, storwize_port, storage_name, list_resources):
    storwize_connection = storwize_connect(
        storwize_user, storwize_password, storwize_ip,
storwize_port)

    # В этот список будут складываться состояние каждого
ресурса (диск, блок питания, ...) в формате zabbix (This lis
t will contain state of every resource (disk, psu, ...) on zabbix
format)
    state_resources = []
    is_there_expansion_enclosure = 0

    try:
        for resource in list_resources:
            stdin, stdout, stderr =
storwize_connection.exec_command(
                'svcinfo {0} -delim :'.format(resource))

            if len(stderr.read()) > 0:  # Если случились ошибки,
запиши их в лог и выйди из скрипта (If errors occur,
then write them to log-file and correctyly end of ssh-session)
                storwize_logger.error(
                    "Error Occurs in SSH Command -
{0}".format(stderr.read()))
```

```
            storwize_logout(storwize_connection)
            sys.exit("1100")
        else:
            resource_in_csv = csv.DictReader(
                stdout, delimiter=':')  # Create CSV
            timestampnow = int(time.time())
            storwize_logger.info(
                "Starting collecting status of resource -
{0}".format(resource))

            for one_object in resource_in_csv:
                if ['lsmdiskgrp'].count(resource) == 1:
                    key_health = "health.{0}.[{1}]".format(
                        resource, one_object["name"])
                    key_overallocation =
"overallocation.{0}.[{1}]".format(
                        resource, one_object["name"])
                    key_used = "used.{0}.[{1}]".format(
                        resource, one_object["name"])
                    key_virtual = "virtual.{0}.[{1}]".format(
                        resource, one_object["name"])
                    key_real = "real.{0}.[{1}]".format(
                        resource, one_object["name"])
                    key_free = "free.{0}.[{1}]".format(
                        resource, one_object["name"])
                    key_total = "total.{0}.[{1}]".format(
                        resource, one_object["name"])

                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))
                    state_resources.append("%s %s %s %s" % (
```

```
                    storage_name, key_overallocation,
timestampnow, one_object["overallocation"]))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_used, timestampnow,
convert_capacity_to_bytes(one_object["used_capacity
"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_virtual, timestampnow,
convert_capacity_to_bytes(one_object["virtual_ca
pacity"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_real, timestampnow,
convert_capacity_to_bytes(one_object["real_capacity
"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_free, timestampnow,
convert_capacity_to_bytes(one_object["free_capacity
"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_total, timestampnow,
convert_capacity_to_bytes(one_object["capacity"]))

  )

            elif ['lsenclosurecanister'].count(resource) == 1:
                key_health = "health.{0}.[{1}.{2}]".format(
                    resource, one_object["enclosure_id"],
one_object["canister_id"])
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))
            elif ['lsenclosurebattery'].count(resource) == 1:
```

```
                    key_health = "health.{0}.[{1}.{2}]".format(
                        resource, one_object["enclosure_id"],
one_object["battery_id"])
                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))
                elif ['lsdrive'].count(resource) == 1:
                    key_health = "health.{0}.[{1}.{2}]".format(
                        resource, one_object["enclosure_id"],
one_object["slot_id"])
                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))
                elif ['lsenclosurepsu'].count(resource) == 1:
                    needed_attributes = [
                        'input_failed', 'output_failed', 'fan_failed']
                    enclosure_id = one_object["enclosure_id"]
                    psu_id = one_object["PSU_id"]
                    advanced_info = advanced_info_of_resource(
                        resource, needed_attributes,
storwize_connection, enclosure_id, psu_id)

                    key_input_failed =
"inFailed.{0}.[{1}.{2}]".format(
                        resource, one_object["enclosure_id"],
one_object["PSU_id"])
                    key_output_failed =
"outFailed.{0}.[{1}.{2}]".format(
                        resource, one_object["enclosure_id"],
one_object["PSU_id"])
                    key_fan_failed =
"fanFailed.{0}.[{1}.{2}]".format(
```

```
                    resource, one_object["enclosure_id"],
one_object["PSU_id"])
                key_health = "health.{0}.[{1}.{2}]".format(
                    resource, one_object["enclosure_id"],
one_object["PSU_id"])
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_input_failed,
timestampnow, convert_text_to_numeric(advanced_info["inpu
t_failed"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_output_failed,
timestampnow, convert_text_to_numeric(advanced_info["out
put_failed"])))
                state_resources.append("%s %s %s %s" % (
                    storage_name, key_fan_failed,
timestampnow,
convert_text_to_numeric(advanced_info["fan_fa
iled"])))
            elif ['lsenclosure'].count(resource) == 1:
                needed_attributes = ['fault_LED']
                enclosure_id = one_object["id"]
                advanced_info = advanced_info_of_resource(
                    resource, needed_attributes,
storwize_connection, enclosure_id)

                key_fault_led =
"faultLED.{0}.[{1}.{2}]".format(
                    resource, one_object["id"],
one_object["serial_number"])
```

```
                    key_health = "health.{0}.[{1}.{2}]".format(
                        resource, one_object["id"],
one_object["serial_number"])
                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))
                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_fault_led, timestampnow,
convert_text_to_numeric(advanced_info["fault_L
ED"])))

                    if one_object["type"] == "expansion":
                        is_there_expansion_enclosure += 1

                elif ['lsportfc', 'lsportsas'].count(resource) == 1:
                    key_running = "running.{0}.[{1}.{2}]".format(
                        resource, one_object["port_id"],
one_object["node_name"])
                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_running, timestampnow,
convert_text_to_numeric(one_object["status"])))
                elif ['lsvdisk', 'lsmdisk'].count(resource) == 1:
                    key_health = "health.{0}.[{1}]".format(
                        resource, one_object["name"])
                    state_resources.append("%s %s %s %s" % (
                        storage_name, key_health, timestampnow,
convert_text_to_numeric(one_object["status"])))

            state_resources.append("%s %s %s %s" % (
                storage_name, "is_there_expansion_enclosure",
timestampnow, is_there_expansion_enclosure))
    except Exception as pizdec:
```

```
        storwize_logger.error(
            "Error occurs in collecting status - {}".format(pizdec))
        # Если возникло исключение, нужно корректно
заверешить ssh-сессию (If exception occur, than correctly
end of
ssh-session)
        storwize_logout(storwize_connection)
        sys.exit("1100")

    # Завершаем ssh-сессию при успешном выполнении
сбора метрик (Correctly end of session after get metrics)
    storwize_logout(storwize_connection)
    return send_data_to_zabbix(state_resources, storage_name)


def main():

    storwize_parser = argparse.ArgumentParser()
    storwize_parser.add_argument(
        '--storwize_ip', action="store", help="Where to connect",
required=True)
    storwize_parser.add_argument(
        '--storwize_port', action="store", required=True)
    storwize_parser.add_argument(
        '--storwize_user', action="store", required=True)
    storwize_parser.add_argument(
        '--storwize_password', action="store", required=True)
    storwize_parser.add_argument(
        '--storage_name', action="store", required=True)

    group =
storwize_parser.add_mutually_exclusive_group(required=Tru
```

```
e)
    group.add_argument('--discovery', action='store_true')
    group.add_argument('--status', action='store_true')
    arguments = storwize_parser.parse_args()

    list_resources = ['lsvdisk', 'lsmdisk', 'lsmdiskgrp',
'lsenclosure', 'lsenclosurebattery',
                      'lsenclosurepsu', 'lsenclosurecanister', 'lsdrive',
'lsportfc', 'lsportsas']

    if arguments.discovery:
        storwize_logger.info(
            "********************************** Starting
Discovering **********************************")
        result_discovery =
discovering_resources(arguments.storwize_user,
arguments.storwize_password,
                                            arguments.storwize_ip,
arguments.storwize_port, arguments.storage_na
me, list_resources)
        print (result_discovery)
    elif arguments.status:
        storwize_logger.info(
            "********************************** Starting Get
Status **********************************")
        result_status =
get_status_resources(arguments.storwize_user,
arguments.storwize_password,
                                          arguments.storwize_ip,
arguments.storwize_port, arguments.storage_name,
list_resources)
        print (result_status)
```

```
if __name__ == "__main__":
    main()
```

4) Attribuer les droits au fichier

```
Chmod 777 storwize_get_state.py
```

5) Installer ces dépendances

```
sudo yum groupinstall 'development tools' -y
sudo yum install openssl-devel libffi-devel bzip2-devel -y
```

6) Installer python2.7

```
curl -O https://www.python.org/ftp/python/2.7/Python-2.7.tgz
tar -xzf Python-2.7.tgz
cd Python-2.7/
./configure --enable-optimizations
make altinstall
pip2.7 install paramiko
pip2.7 install cryptography==2.8
```

7) exécuter les scripts est le résultat doit donner 0

Monitoring MSSQL Server

1) Ajouter la Template MSSQL by ODBC pour l'hôte

2) modifier ces macros comme suite :

{$MSSQL.DSN}=

{$MSSQL.INSTANCE}=

{$MSSQL.PASSWORD}=

{$MSSQL.USER}=

3) Installer l'ODBC sur le serveur zabbix :

```
curl https://packages.microsoft.com/config/rhel/8/prod.repo > /etc/yum.repos.d/mssql-release.repo
sudo yum remove unixODBC-utf16 unixODBC-utf16-devel
sudo ACCEPT_EULA=Y yum install -y msodbcsql17
sudo ACCEPT_EULA=Y yum install -y mssql-tools echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc source ~/.bashrc
sudo yum install -y unixODBC-devel
```

4) Configurer le log de l'odbc

dans le ficier /etc/ odbcinst.ini

a. Vérifier que ces données existent

```
[ODBC Driver 17 for SQL Server]
Description=Microsoft ODBC Driver 17 for SQL Server
Driver=/opt/microsoft/msodbcsql17/lib64/libmsodbcsql-17.7.so.2.1
UsageCount=1
```

*si vous voulez le log de l'odbc

b. Ajouter ces paramètres sur ce fichier

```
[ODBC]
Trace=Yes
TraceFile=/tmp/unixodbc.log
ForceTrace=Yes
Pooling=No
```

5) Dans le fichier /etc/odbc.ini

On ajoute

```
[MYMSSQL]
Driver=ODBC Driver 17 for SQL Server
Server=192.168.199.135
PORT=1433
UID=zabbix
PWD=P@ssw0rd
TDS_Version=8.0
```

6) Créer un utilisateur de serveur de base de données et lui attribuer ces permissions :

```
USE msdb;
GRANT SELECT ON OBJECT::msdb.dbo.sysjobs TO zbx_monitor
GRANT SELECT ON OBJECT::msdb.dbo.sysjobservers TO zbx_monitor
GRANT SELECT ON OBJECT::msdb.dbo.sysjobactivity TO zbx_monitor
GRANT EXECUTE ON OBJECT::msdb.dbo.agent_datetime TO zbx_monitor
```

Et aussi

View server state , view any definition , view any database

Monitoring nodejs

Sur le serveur zabbix

1) Ajouter la Template pm2-zabbix
2) Installer zabbix-get

```
rpm –ivh zabbix-get
```

(Continuer l'étape 3 après avoir configurer le serveur Nodejs)

3) Tester avec la commande si l'application apparait

```
zabbix_get -s IP-serveur-Nodejs -k pm2.processes
```

Le Résulat est comme suite :

```
{
    "data": [
        {
            "{#PROCESS_ID}": "app-0",
            "{#PROCESS_NAME}": "app"
        }
```

Sur le serveur Nodjs

N.B: L'agent zabbix doit être configurer comme active

1) installer git

```
sudo yum install git
```

2) Installer Pm2

```
sudo npm install -g pm2
```

3) Installer Pm2-zabbix en tant que root

```
sudo npm install -g pm2-zabbix
```

4) Installer zabbix-sender

rpm –ivh zabbix-sender

5) Si , zabbix-agent est installé passer directement à l'étape suivante.

Si , Zabbix-agent2 est installe veiller suivre cette étape.

On modifie le chemin de l'agent-zabbix2 dans le fichier de configuration de zabbix-sender

sudo vi /usr/lib/node_modules/pm2-zabbix/node_modules/zabbix-sender/lib/ZabbixSender.js

modifier

config : options.config || '/etc/zabbix/zabbix_agent.conf'

par

config : options.config || '/etc/zabbix/zabbix_agent2.conf',

6) démarrer le monitoring de l'application avec :

Pm2 start app.js

7) Vérifier que le pm2-zabbix à découvert l'application :

pm2-zabbix –discover

Le Résultat :


```
{
	"data": [
		{
			"{#PROCESS_ID}": "app-0",
			"{#PROCESS_NAME}": "app"
		}
	]
}
```

8) on modifie dans le fichier sudoers comme suite :

Root ALL=(ALL) ALL

zabbix ALL=(ALL)ALL

Defaults:root !requiretty

Defaults:zabbix !requiretty

Defaults!/usr/bin/zabbix_sender !requiretty

9) Attribuer le droit de l'écriture à tout le monde pour le répértoire /var/lib

```
chmod o+w /var/lib
```

10) Installer le package openssh-askpass

```
sudo yum install openssh-askpass
```

11) Excuter la commande

```
sudo setsid ssh ro@nom-du-serveur
```

Si la commande donne une erreur c'est à cause de l'absence de l'interface graphique.

12) On ajoute le paramètre pm2.processes dans le fichier de configuration de zabbix-agent2

```
UserParameter=pm2.processes,sudo -u root pm2-zabbix –discover
```

13) On redémarre l'agent zabbix2

```
systemctl restart zabbix-agent2
```

Monitoring TSM

1) installer zabbix-sender
2) installer

gskcrypt64-8.0.55.2.linux.x86_64.rpm
TIVsm-API64.x86_64.rpm
gskssl64-8.0.55.2.linux.x86_64.rpm
TIVsm-BA.x86_64.rpm

3) Importer la Template

Template_Tivoli_Storage_Manager puis ajouter les items qui ne se trouve pas (regarder la capture d'écran « tsm items »)

4) Créer un script dans /usr/lib/zabbix/externalscripts/

Nommer «tsm.sh» dont le contenu est :

```
#!/bin/bash
# Title: tsm.sh
# Description: Script to gather data from TSM and
report back to zabbix.
```

```
# Original Author: Chris S. / [wings4marie @ gmail
DOT com] / [IRC: Parabola@Freenode]
# Author: Chris Jones / [rollercow @ sucs.org] - Mostly
tidying
#########################################################
#########################################
#
# Tested with TSM 6.2 and zabbix 2.0.11
#
####################
# CONFIGURATION #
####################
zabbix_sender="/usr/bin/zabbix_sender"
zabbix_config="/etc/zabbix/zabbix_agentd.conf"
zabbix_log="/dev/null"
tsm_binary="/usr/bin/dsmadmc" # Path to the admin
CLI binary tool
tsm_user="admin" # TSM username
tsm_pass="Admin1234" # TSM Password

####################
#  FUNCTIONS    #
####################
function send_value {
        "$zabbix_sender" -c $zabbix_config -k $1 -o $2 >
$zabbix_log
}

function tsm_cmd {
```

```
        "$tsm_binary" -id=$tsm_user -pa=$tsm_pass -
dataonly=yes "$1" | grep -v ANS0102W # shuts up
persistent warning
}

####################
#  TAPE STATS   #
####################

function tsm_scratchvols { # Number of scratch
volumes
        scratchvols=$(tsm_cmd "select count(*)
Scratch_Vols from libvolumes where status='Scratch'")
        send_value tsm.tapes.scratchvols "$scratchvols"
}
function tsm_totalvols { # Total number of volumes
        totalvols=$(tsm_cmd "select count(*) Total_Vols
from libvolumes" )
        send_value tsm.tapes.totalvols "$totalvols"
}
function tsm_consolidate_num { # Number of tapes
marked for consolidation
        volCount=$(tsm_cmd "SELECT
count(volume_name) FROM volumes WHERE
status='FULL' AND pct_utilized < 30")
        send_value tsm.tapes.consolidate.count
"$volCount"
}
function tsm_tapes_errors { # Number of tapes with an
```

```
error status
        tapeErrors=$(tsm_cmd "SELECT COUNT(*)
FROM volumes WHERE error_state='YES'")
        send_value tsm.tapes.errors.status "$tapeErrors"
}
####################
#  DRIVE STATS  #
####################

function tsm_drives_offline { # Number of drives
marked as offline
        offlineDrives=$(tsm_cmd "SELECT COUNT(*)
FROM drives WHERE NOT online='YES'")
        send_value tsm.drives.offline.count
"$offlineDrives"
}

function tsm_drives_loaded { # Number of drives with a
tape (loaded)
        loadedDrives=$(tsm_cmd "SELECT COUNT(*)
FROM drives WHERE drive_state='LOADED'")
        send_value tsm.drives.loaded.count
"$loadedDrives"
}

function tsm_drives_empty { # Number of "empty"
Drives within your library
        emptyDrives=$(tsm_cmd "SELECT COUNT(*)
FROM drives WHERE drive_state='EMPTY'")
```

```
        send_value tsm.drives.empty.count
"$emptyDrives"
}

####################
#  POOL  STATS  #
####################

function tsm_DBPOOL_usage { # See NOTES
        DBPOOL=$(tsm_cmd "SELECT pct_utilized
FROM stgpools WHERE stgpool_name='DBPOOL'
ORDER BY stgpool_name DESC")
        send_value tsm.pools.DBPOOL "$DBPOOL"

}
function tsm_DEVPOOL_usage { # See NOTES
        DEVPOOL=$(tsm_cmd "SELECT pct_utilized
FROM stgpools WHERE stgpool_name='DEVPOOL'
ORDER BY stgpool_name DESC")
        send_value tsm.pools.DEVPOOL "$DEVPOOL"

}
function tsm_FSPOOL_usage { # See NOTES
        FSPOOL=$(tsm_cmd "SELECT pct_utilized
FROM stgpools WHERE stgpool_name='FSPOOL'
ORDER BY stgpool_name DESC")
        send_value tsm.pools.FSPOOL "$FSPOOL"

}
```

```
function tsm_COPYPOOL_usage { # See NOTES
        COPYPOOL=$(tsm_cmd "SELECT pct_utilized
FROM stgpools WHERE stgpool_name='COPYPOOL'
ORDER BY stgpool_name DESC")
        send_value tsm.pools.COPYPOOL
"$COPYPOOL"

}
function tsm_ARCH_FSPOOL_usage { # See NOTES
        ARCH_FSPOOL=$(tsm_cmd "SELECT
pct_utilized FROM stgpools WHERE
stgpool_name='ARCH_FSPOOL' ORDER BY
stgpool_name DESC")
        send_value tsm.pools.ARCH_FSPOOL
"$ARCH_FSPOOL"

}




####################
#   TSM STATS   #
####################

function tsm_nodes_count { #Total number of nodes in
your TSM environment
        nodeCount=$(tsm_cmd "SELECT COUNT(*)
```

```
FROM nodes")
        send_value tsm.nodes.count "$nodeCount"
}

function tsm_nodes_locked { # number of nodes
marked as locked
        lockedNodes=$(tsm_cmd "SELECT
count(node_name) FROM nodes WHERE
locked='YES'")
        send_value tsm.nodes.locked.count
"$lockedNodes"
}

function tsm_nodes_sessioncount {
        sessCount=$(tsm_cmd "SELECT COUNT(*)
FROM sessions WHERE session_type='Node'")
        send_value tsm.nodes.sessions.count "$sessCount"
}

function tsm_failedjobs { # Number of jobs marked as
"Failed"
        failedInt=$(tsm_cmd "query event * *
begind=today-1 begint=00:00:00 endd=today-1
endt=23:59:59 exceptionsonly=yes" | grep Failed | wc -
l)
        send_value tsm.jobs.failed "$failedInt"
}

function tsm_missedjobs { # Number of jobs marked as
```

```
"Missed"
        missedInt=$(tsm_cmd "query event * *
begind=today-1 begint=00:00:00 endd=today-1
endt=23:59:59 exceptionsonly=yes" | grep Missed | wc -
l)
        send_value tsm.jobs.missed "$missedInt"
}

function tsm_summary_24hrs { #Data in B by activity
        summary=$(tsm_cmd "SELECT
activity,sum(bytes) FROM summary where
end_time>current_timestamp-24 hours GROUP BY
activity" | sed 's/ //')
        for jobtype in archive backup full_dbbackup
reclamation stgpoolbackup POOLCOPYUTILGB
POOLPRIMUTILGB PROCESS_END REPLICATION
REPL_BYTESINGESTED
REPL_BYTESREPLICATED REPL_TREND
REPL_WORKLOAD "TAPEMOUNT"
        do
                echo "$summary" | grep -i $jobtype >
/dev/null
                if [ $? = 0 ]
                then
                    send_value tsm.summary.daily.$jobtype
$(echo "$summary" | grep -i $jobtype | awk {'print $2'})
                fi
        done
}
```

```
function tsm_summary_total_stored { # Total data
stored in B
        totalStored=$(tsm_cmd "SELECT
cast(SUM(logical_mb)*1024*1024 as bigint) FROM
occupancy")
        send_value tsm.summary.total.stored
"$totalStored"
}

####################
#  SCHEDULING   #
####################
##########################################################
#######################################
# Place functions within each Category for execution
#
# INFO: (Use Cron)
#
# daily - Scheduled for 8am, 7 days a week
#
# hourly - Run every 60 minutes
#
# Below are my Cron entries
#
#
#
# 0 8 * * * /bin/bash "/etc/zabbix/externalscripts/tsm.sh"
daily             #
```

```
# 0,60 * * * * /bin/bash
"/etc/zabbix/externalscripts/tsm.sh" hourly          #
#
#
##################################################################
#####################################
function daily {
    tsm_missedjobs
    tsm_failedjobs
    tsm_summary_24hrs
}

function hourly {
    tsm_scratchvols
    tsm_totalvols
    tsm_consolidate_num
    tsm_tapes_errors
    tsm_drives_offline
    tsm_drives_loaded
    tsm_drives_empty
    tsm_nodes_count
    tsm_nodes_locked
    tsm_nodes_sessioncount
    tsm_summary_total_stored
    tsm_DBPOOL_usage
    tsm_DEVPOOL_usage
    tsm_FSPOOL_usage
    tsm_COPYPOOL_usage
    tsm_ARCH_FSPOOL_usage
```

```
    #tsm_logpool_usage
}

if [[ "$1" == *hourly* ]]; then
        hourly
elif [[ "$1" == *daily* ]]; then
        daily
else echo "Useage: tsm.sh [hourly|daily]"
fi
```

Monitoring Nutanix Avec Zabbix

La template est paramétrée pour utiliser SNMPv3 en mode authPriv avec Auth SHA et Privacy AES.

Prism Central

La configuration de prism se fait de façon assez simple via l'interface de paramétrage SNMP.

Assurez-vous que le transport soit bien configuré en UDP, sur le port de votre choix (161 par défaut)

Se rendre ensuite dans l'onglet Users et ajouter un nouvel utilisateur pour que zabbix puisse se connecter

Profitez-en pour télécharger la MIB via le bouton Download MIB, nous allons en avoir besoin.

Cliquez sur Save. La configuration de prism est terminée

MIB Nutanix

Une fois la MIB Nutanix récupérée, il faut la placer sur le serveur Zabbix. Sur un serveur CentOS, il faut placer le fichier dans /usr/share/snmp/mibs

```
# ls /usr/share/snmp/mibs/NUTANIX*

/usr/share/snmp/mibs/NUTANIX-MIB
```

Afin que Zabbix prenne ce nouveau fichier en considération, il est nécessaire de redémarrer le service

```
# systemctl restart zabbix-server
```

Pour tester le bon fonctionnement de la MIB et de la configuration de prism, utilisons la commande snmpwalk qui va, dans cet exemple, lister les noms des containers Nutanix.

Commande :

snmpwalk -v 3 -a SHA -A VOTREMOTDEPASSESHA -u zabbix -x AES -X VOTREMOTDEPASSEAES IP.DE.PRISM.CENTRAL -l AuthPriv citContainerName

```
NUTANIX-MIB::citContainerName.1 = STRING: container00.
NUTANIX-MIB::citContainerName.2 = STRING: containerA.
NUTANIX-MIB::citContainerName.3 = STRING: containerB.
NUTANIX-MIB::citContainerName.4 = STRING: containerC.
NUTANIX-MIB::citContainerName.5 = STRING: RESERVED_SPACE.
NUTANIX-MIB::citContainerName.6 = STRING: containerD.
```

La configuration du serveur est terminée. On peut passer à la configuration de Zabbix.

Zabbix

Si ce n'est pas déjà fait, créez un nouvel hôte pour prism central dans Zabbix. Cet hôte à besoin de 3 macros pour que le template Nutanix fonctionne. Ces 3 macros sont l'utilisateurs SNMP, le mot de passe SHA et le mot de passe AES.

Ces macros sont :

{$NTX_SNMP_USER}
{$NTX_SNMP_AUTH}
{$NTX_SNMP_PRIV}

Importez ensuite ce Template dans Zabbix et appliquez le sur votre hôte Prism Central.

Par défaut, la découverte des VM est désactivée car il n'est pas possible de créer proprement des hôtes depuis ce processus LLD.

Si vous avez un cluster Nutanix volumineux et que Zabbix ne peux remonter les informations à cause d'une erreur de timeout, pensez à augmenter ce timeout dans la configuration de Zabbix

Timeout=10

N.B :
Les bugs rencontrés actuellement sont les suivants :

Jusqu'à la version 4.6 de NOS, le Discovery n'arrive pas à remonter tous les hyperviseurs. Le comportement est le même avec snmpwalk. Il semble que ce soit un problème du coté de Nutanix. Le problème est corrigé en 4.6.
Le nombre de VM remontées est limité à 250. Sans doute une limitation coté Nutanix également.