PROJECT

Multi-tenant cybersecurity operations centre designed
as an open cloud service with machine learning elements

Linked to work package KPB1 – Design of the virtualization solution

Linked to output
V3 – Description of the selected solution meeting the specified criteria

type – documentation

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Ansible and Zabbix solutions – experiment

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

CONTENTS

List of figures **4**
Introduction **5**
1 Ansible **5**
1.1 Installation of Ansible AWX on the Debian Linux distribution 5
1.1.1 Installation requirements 5
1.1.2 Installation of the Ansible AWX system 5
1.2 Working with Ansible AWX 10
1.2.1 Logging in to Ansible AWX and basic concepts 10
1.2.2 Working with Ansible AWX 12
1.2.3 Running a playbook on a host using Ansible AWX 14
2 Zabbix **21**
2.1 Zabbix monitoring 21
2.2 Integration of Ansible and Zabbix 22
2.2.1 Management using Ansible 22
2.2.2 Zabbix Agent role 23
2.2.3 Community Zabbix 24
2.2.4 Playbooks and annexes 25
2.3 Prometheus 28
References **30**

Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

LIST OF FIGURES

Figure 1 Login window....10
Figure 2 Ansible AWX dashboard....11
Figure 3 Creating a user account....12
Figure 4 Assigning various options to a user account....13
Figure 5 Creating an organization using Ansible AWX 1....14
Figure 6 creating an organization using Ansible AWX 2....14
Figure 7 Creating a project with local storage....15
Figure 8 Creating a Project using GitHub....16
Figure 9 Creating credentials in Ansible AWX....17
Figure 10 Creating an inventory in Ansible AWX....17
Figure 11 Assigning a host to the inventory....17
Figure 12 Running a command on a host in the inventory....18
Figure 13 Running the ping module on a host....18
Figure 14 Executed ping module on a host....19
Figure 15 Creating a template using Ansible AWX....19
Figure 16 Executed template using Ansible AWX....20
Figure 17 Main Zabbix dashboard on the local server....22
Figure 18 Conceptual architecture of Prometheus [22]....29

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

INTRODUCTION

This document summarizes experiments with automation using Ansible, monitoring via Zabbix and their interconnection.

1 ANSIBLE

In the following section, we will present the Ansible component,

1.1 INSTALLATION OF ANSIBLE AWX ON THE DEBIAN LINUX DISTRIBUTION

1.1.1 Requirements for installation

1.1.1.1 System requirements

- At least 4GB of RAM.
- At least two processor cores.
- At least 20GB of disk storage space.
- A working Docker, Openshift or Kubernetes.
- When using an external PostgreSQL database, use at least version 10+.

1.1.1.2 System versions used in the installation

- Ansible AWX – 15.0.1
- Ansible – 2.9.15
- Docker – 19.03.13
- Docker-compose – 1.21.0

1.1.2 Installation of the Ansible AWX system

The installation was carried out on the **Debian 10.6** Linux distribution. Ansible AWX can be installed on any Linux distribution on which an Ansible server can be installed. To install Ansible AWX, we used Docker and docker-compose. It can also be installed using Openshift or Kubernetes. The installation steps for the individual options can be found at https://github.com/ansible/awx.

We will now go through the steps required to install Ansible AWX using Docker and docker-compose.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

1.1.2.1. Installation of the Docker system

- Updating the repositories.

```
apt-get update -y
```

- Installation of the dependencies required to install the Docker system.

```
apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common -y
```

- Downloading and adding the Docker GPG key.

```
curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
```

- Adding the Docker repository.

```
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
```

- Updating the repositories.

```
apt-get update -y
```

- Installation of Docker and docker-compose.

```
apt-get install docker-ce docker-compose -y
```

- Verification of the installation

```
docker –version
docker run hello-world
systemctl status docker.service
```


Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

1.1.2.2. Installation of the Ansible system

- Adding the Ansible repository.

```
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu bionic main" | tee /etc/apt/sources.list.d/ansible.list
```

- Updating the repositories.

```
apt-get update -y
```

- Installation of Ansible.

```
apt-get install ansible -y
```

- Verification of the installation.

```
ansible –version
```

1.1.2.3. Installation of Node.js and the Python library

- Installation of Node.js, NPM and Git.

```
apt-get install nodejs npm git -y
npm install npm --global
```

- Installation of the Python module for Docker and the required dependencies.

```
apt-get install python3-pip pwgen python3-docker -y
pip3 install requests==2.14.2
```

- Next, the Python version needs to be updated.

```
update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
update-alternatives --install /usr/bin/python python /usr/bin/python3 2
```

- Determining the docker-compose version

```
docker-compose version
```


Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

- Installing the docker-compose Python module based on the docker-compose version.

```
pip3 install docker-compose==1.21.0
```

1.1.2.4. Installation of Ansible AWX

- Downloading the latest version of Ansible AWX from GitHub.

```
git clone --depth 50 https://github.com/ansible/awx.git
```

- It is necessary to change to the awx/installer/ folder.

```
cd awx/installer/
```

- Generating a password for the secret key.

```
pwgen -N 1 -s 30 => ************************
```

- Creating a copy of the original inventory file.

```
ocp inventory inventory.old
```

- Configuration of the inventory file. For better clarity, we recommend deleting the contents of the file and adding only the necessary items to it.

```
nano inventory
```


Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

- Contents of the inventory file

```
      localhost    ansible_connection=local    ansible_python_interpreter="/usr/bin/env
python3"
      [all:vars]
      dockerhub_base=ansible
      awx_task_hostname=awx
      awx_web_hostname=awxweb
      postgres_data_dir=/var/lib/pgdocker
      project_data_dir=/var/lib/awx/projects
      host_port=80
      host_port_ssl=443
      docker_compose_dir=/var/lib/awxcompose
      pg_username=awx
      pg_password=awxpass
      pg_database=awx
      pg_port=5432
      admin_user=admin
      admin_password=adminpassword
      create_preload_data=True
      #key that we generated using pwgen
      secret_key=************************
```

- After the installation, problems arose because the awx_task container did not wait for the awx_postgres container to complete initialization and migration. This caused a problem with database synchronization and Ansible AWX could not be started. This problem is described in the report https://github.com/ansible/awx/issues/8300. To prevent this problem, it is necessary to:
  - modify the file:
    - awx/installer/roles/local_docker/templates/docker-compose.yml.j2
  - find the command for starting awx_task in it
    - command: /usr/bin/launch_awx_task.sh
  - and replace it with the command
    - command: /bin/bash -c "sleep 30 && exec /usr/bin/launch_awx_task.sh"
  - this causes the start of awx_task to be put to sleep for the required time.
- Installation of Ansible AWX using a playbook
  - ansible-playbook -i inventory install.yml

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

- After a successful installation, the required containers (awx_task, awx_web, awx_redis and awx_postgres) should be created, which can be verified with the command
  - docker ps
- After the installation, the database migration will take place. This must be completed before the web interface can be accessed. The following command can be used to monitor the progress
  - docker logs -f awx_task

1.2 WORKING WITH ANSIBLE AWX

1.2.1 Logging in to Ansible AWX and basic concepts

Ansible AWX can be accessed via a web browser. It is sufficient to enter the address or domain on which your Ansible server is running. After the page loads, a login window is displayed. The name and password are those you set in the *inventory* configuration file. These are the variables *admin_user* and *admin_password*.

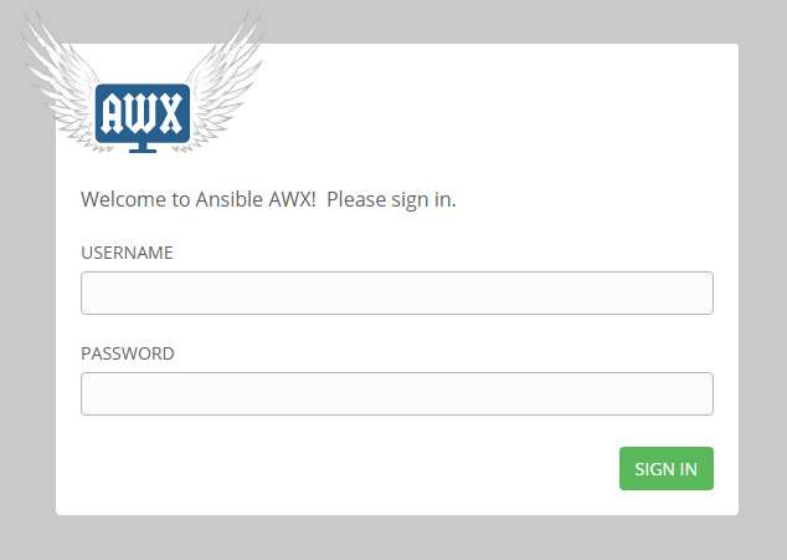


Figure 1 Login window

After a successful login, a dashboard is displayed on which the number of hosts, inventories and projects can be seen. There is also a graph showing the automation tasks (jobs) that have been run.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

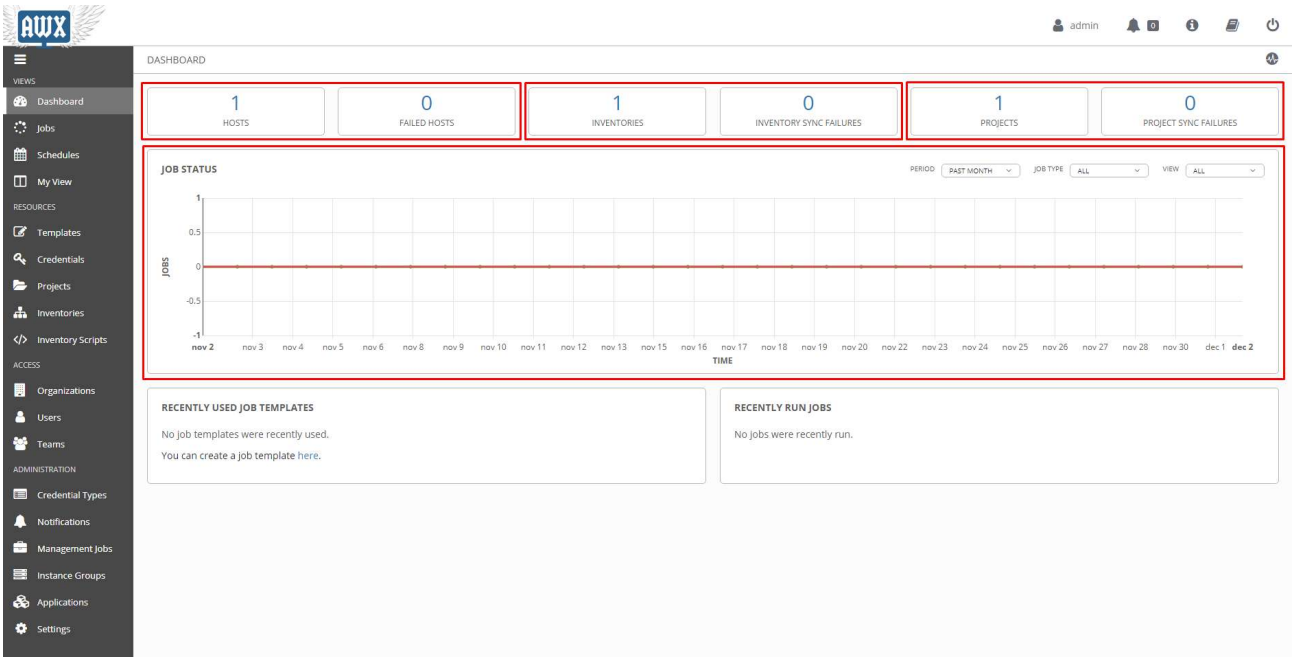


Figure 2 Ansible AWX dashboard

On the left-hand side, you can see various options to choose from. We will now briefly explain them and describe what they contain.

- **Dashboard** – summary of individual hosts, inventories and projects. Graph of successful and failed jobs.
- **Jobs** – contains a list of jobs that have been executed. These jobs can also be run again.
- **Shedules** – Here, the running of individual jobs can be set up based on time.
- **My View** – contains a simple list of various jobs that can be run. This option is intended for those who are not familiar with setting up individual templates, projects and inventories but only need to run some jobs.

- **Templates** – contains the created templates, which relate to individual projects and inventories. They can also be run here to execute the specified tasks.
- **Credentials** – here, various authentication methods to be used by Ansible AWX can be defined. Individual credentials can then be assigned to templates, inventories or projects that require authentication.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

- **Projects** – projects are defined here. Individual projects contain playbooks, which are then used in templates.
- **Inventories** – contains the hosts on which individual tasks are executed.
- **Inventory scripts** – scripts that can automatically add hosts from various environments.

- **Organizations** – a logical grouping of users, teams, projects and inventories.
- **Users** – here, users who can log in to Ansible AWX and manage items in it can be defined.
- **Teams** – a subset of an organization. It also contains users, projects and various credentials and permissions. Credentials can be assigned directly to the whole team rather than to each user separately.

- **Administration –** here there are various options for managing Ansible AWX. For example, the configuration of notifications, various applications and the Ansible AWX settings themselves.

1.2.2 Working with Ansible AWX

1.2.2.1. Creating a user account

A user account can be created in the Users tab. This account can then be used to log in to Ansible AWX. The first name, surname, e-mail, username, password and user type (admin/regular user) can be set.

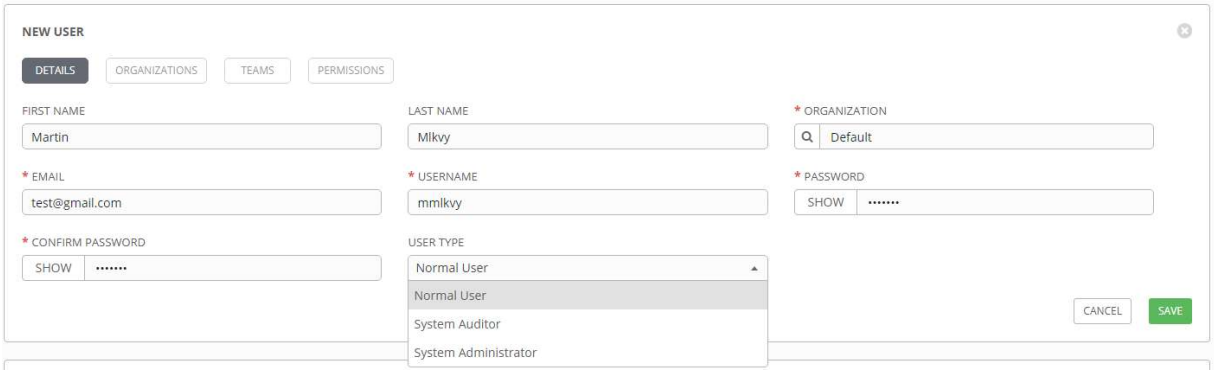


Figure 3 Creating a user account

Once we have created and saved the account, we can add it to an organization or team, or assign various permissions to it.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

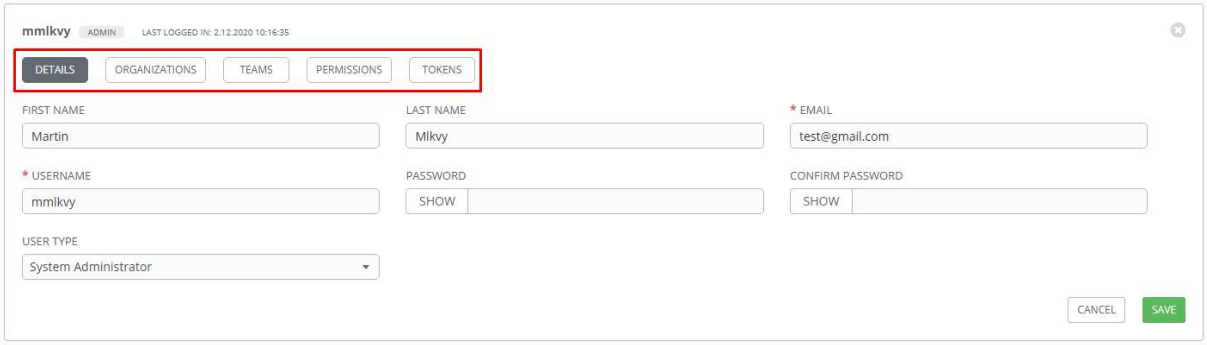


Figure 4 Assigning various options to a user account

Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

1.2.3 Running a playbook on a host using Ansible AWX

1.2.3.1. Creating an SSH connection between the server and the client

First, it is necessary to ensure an SSH connection between the client and the Ansible server. Both must have SSH installed.

First, we generate an SSH key on the server. We do this using the **ssh-keygen** command.

Once we have generated the key, we send the server's public key to the host that we want to manage using the Ansible server. We do this using the command **ssh-copy-id root@ 192.168.1.101**.

1.2.3.2. Creating an organization in Ansible AWX

Log in to Ansible AWX. In the side panel, select the Organizations tab. There, we create a new organization.

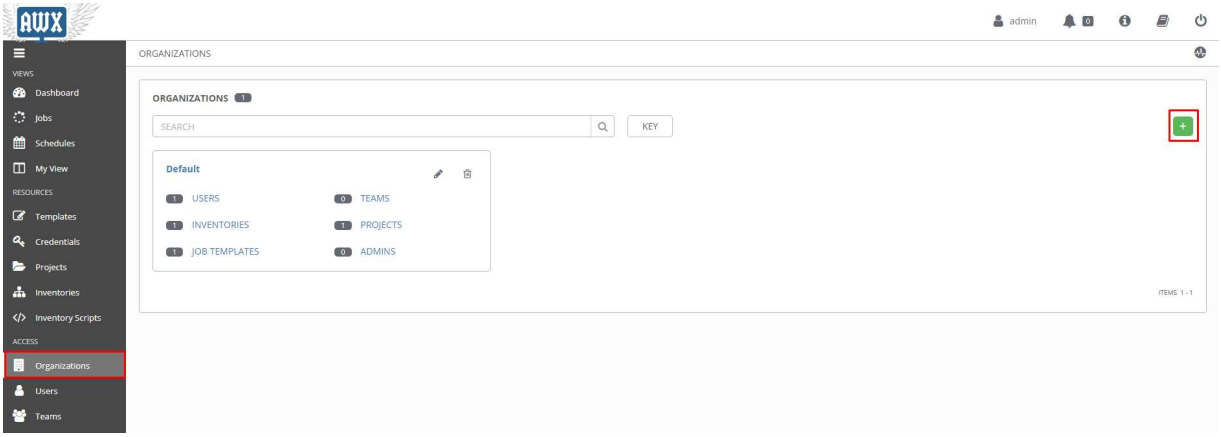


Figure 5 Creating an organization using Ansible AWX 1

Enter the name of the organization and save it.

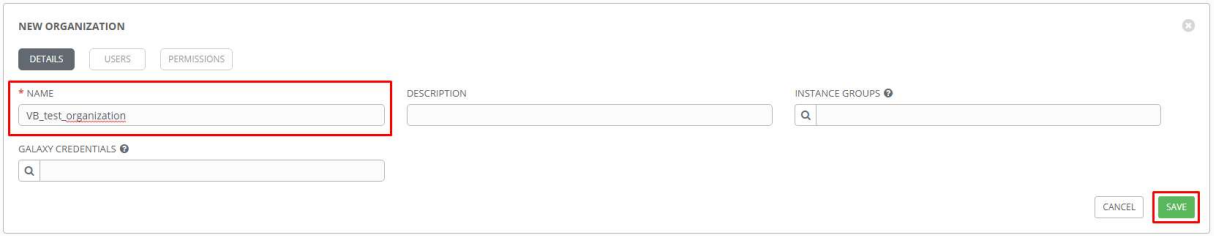


Figure 6 creating an organization using Ansible AWX 2

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

1.2.3.3. Creating a project and assigning an organization to it

Ansible AWX projects contain individual playbooks. We will describe two ways of uploading a playbook to Ansible AWX (from a local folder or from a repository on GitHub).

1. Using a local folder.

First, it is necessary to create a folder on the Ansible server in the location where projects have their working directory. This was defined during the installation of Ansible AWX in the *inventory* file. Specifically, it was the variable *project_data_dir.*

mkdir /var/lib/awx/projects/test_konektivity_p

Once we have created the folder, we can add a file in .yaml (.yml) format to it.

nano /var/lib/awx/projects/test_konektivity_p/testKonektivity.yml

```
- name: Testing conections
  hosts: all
  remote_user: root

  tasks:
  - name: Ping all
    ping:
```

Now we can create a project using Ansible AWX. We do this by clicking on the Projects tab in the left-hand bar and creating a new project there. There, we enter its name, select the organization, set the SCM type to manual and select the folder where we created our playbook.

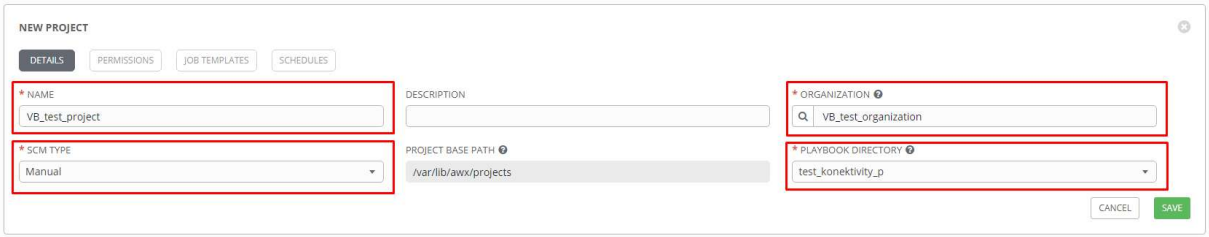


Figure 7 Creating a project with local storage

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2. Using a GitHub repository.

Ansible AWX can download playbooks from GitHub repositories. For it to be able to access the repository, the repository must be public. Then, when creating the project, Git must be selected as the SCM Type and the URL of the repository containing the playbooks must be entered.

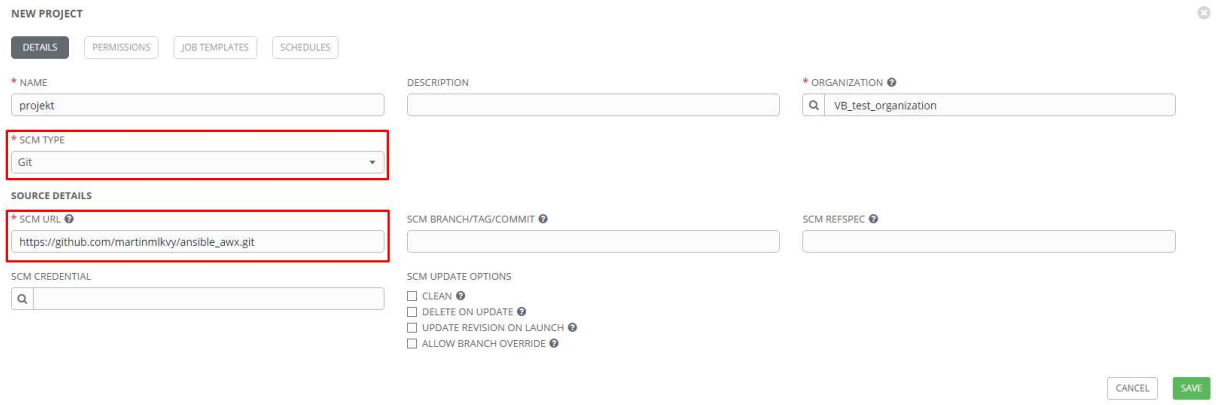


Figure 8 Creating a Project using GitHub

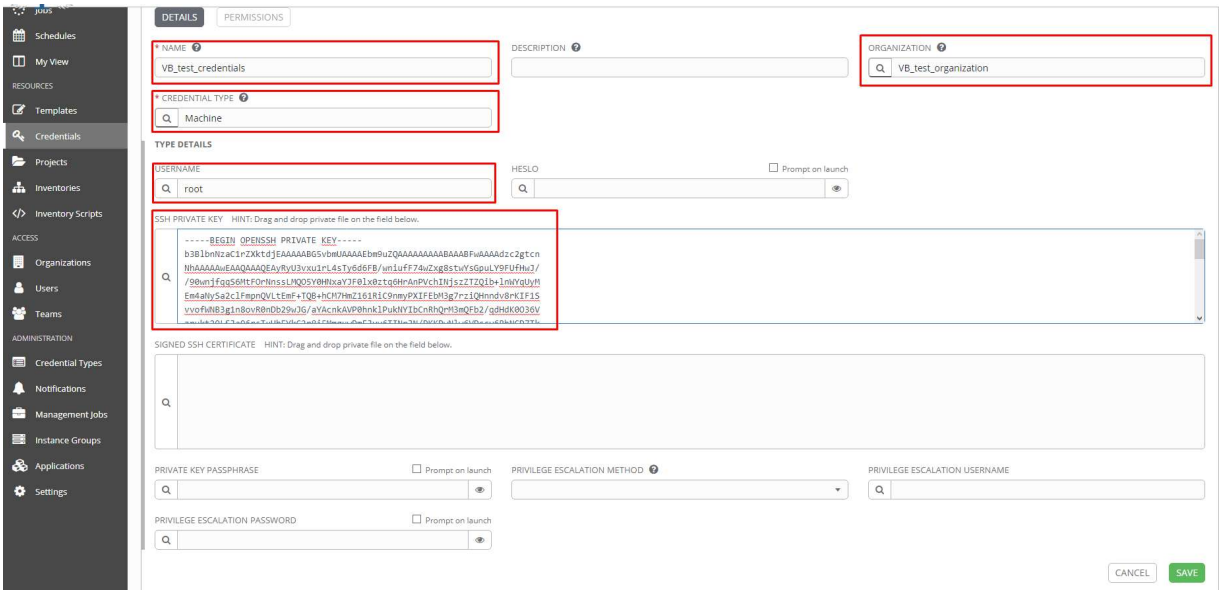
1.2.3.4. Creating credentials

These credentials will be used when running playbooks on the Ansible host.

First, we need to obtain the private key of the Ansible server. Copy it; we will import it into Ansible AWX later.

cat /root/.ssh/id_rsa

Now, in Ansible AWX, we select the Credentials tab in the left-hand bar and create a new instance there. We select machine as the Credential type, because it will only concern our host. We enter the username with which we will log in to the host (root). We attach the private key of our Ansible server.



Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Figure 9 Creating credentials in Ansible AWX

1.2.3.5. Creating an inventory and assigning a host

In Ansible AWX, we click on the Inventory tab, create a new inventory there and save it.

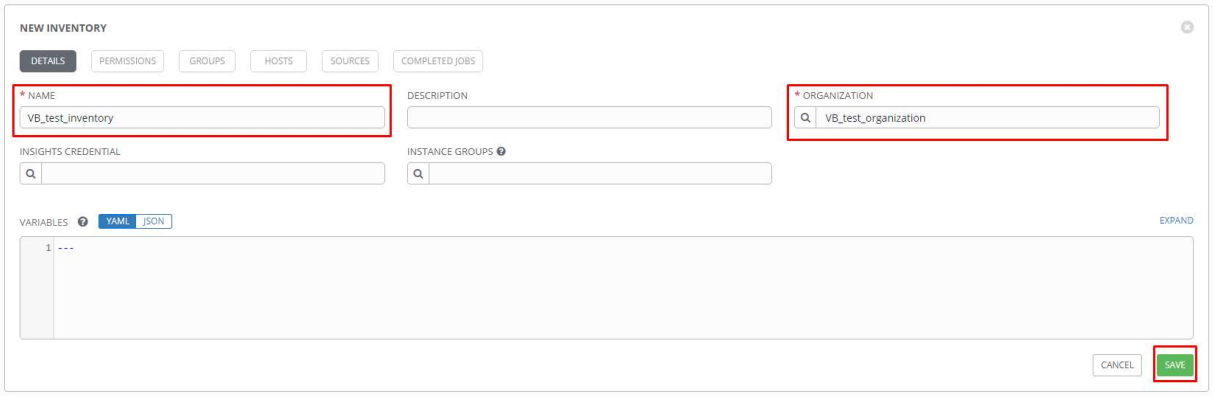


Figure 10 Creating an inventory in Ansible AWX

After saving the created inventory, we go to the hosts tab in the inventory, create a new host and save it.

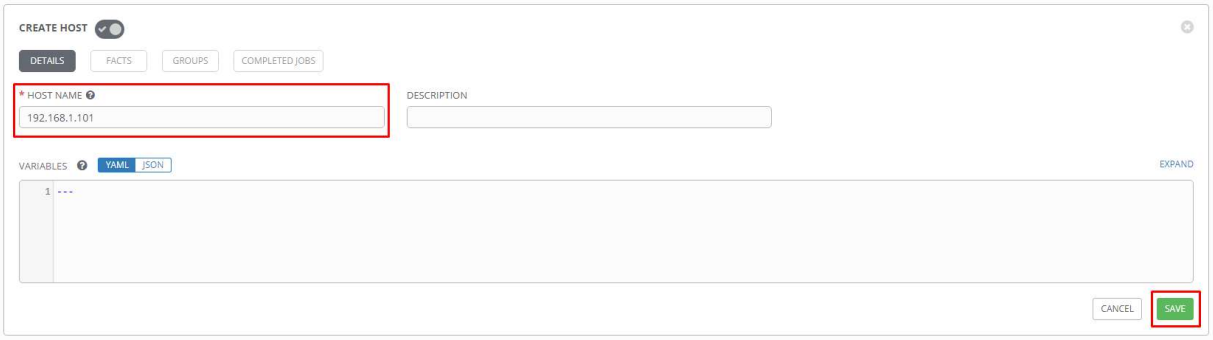


Figure 11 Assigning a host to the inventory

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Now we will try running the ping module on the client to test the connectivity and the SSH connection between Ansible AWX and the Ansible host. Go to Inventories, open the newly created inventory and go to the hosts tab. There, select the host that we created and press run command.

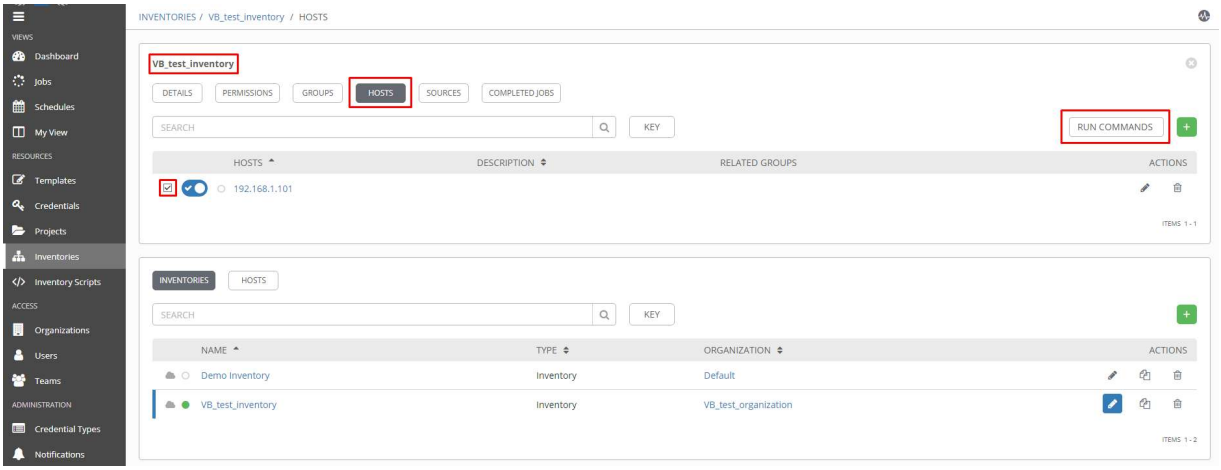


Figure 12 Running a command on a host in the inventory

A window will be displayed in which we select the ping module and the machine credentials. You can also change the verbosity of the output. Then the job must be run using the launch button.

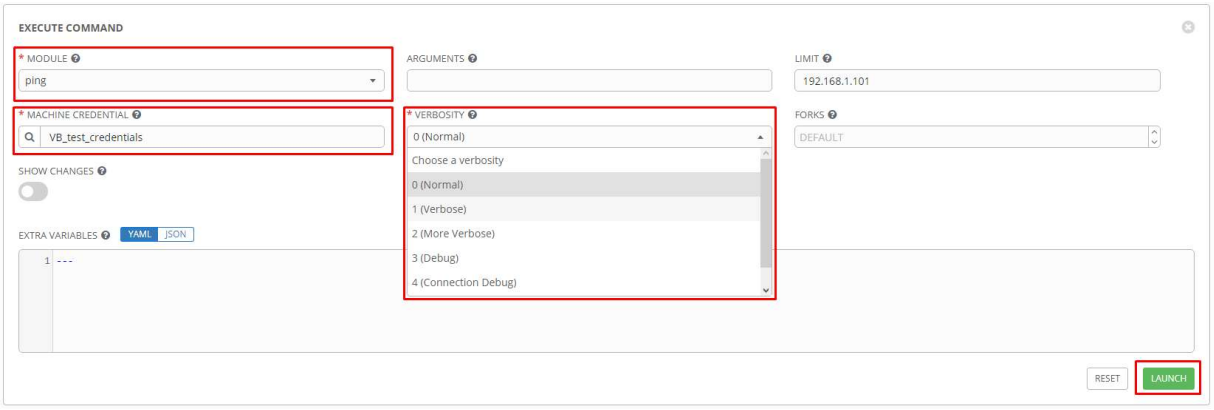


Figure 13 Running the ping module on a host

Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

A window is then displayed in which you can follow the progress of the job execution. After successful execution, it should look like this.

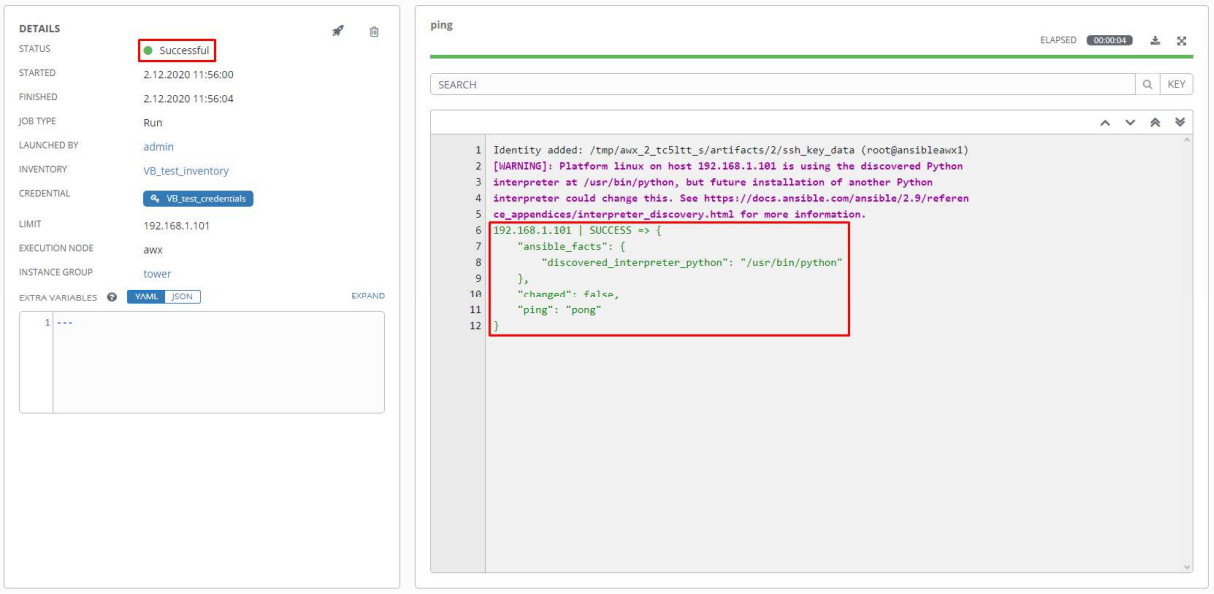


Figure 14 Executed ping module on a host

1.2.3.6. Creating a template using Ansible AWX

In Ansible AWX, we select the Templates tab and create a new job template there. The required fields must be filled in as you can see in the figure. The verbosity mode can also be selected. Once you have everything set up, the template must be saved. After it has been saved, we can run it using Launch

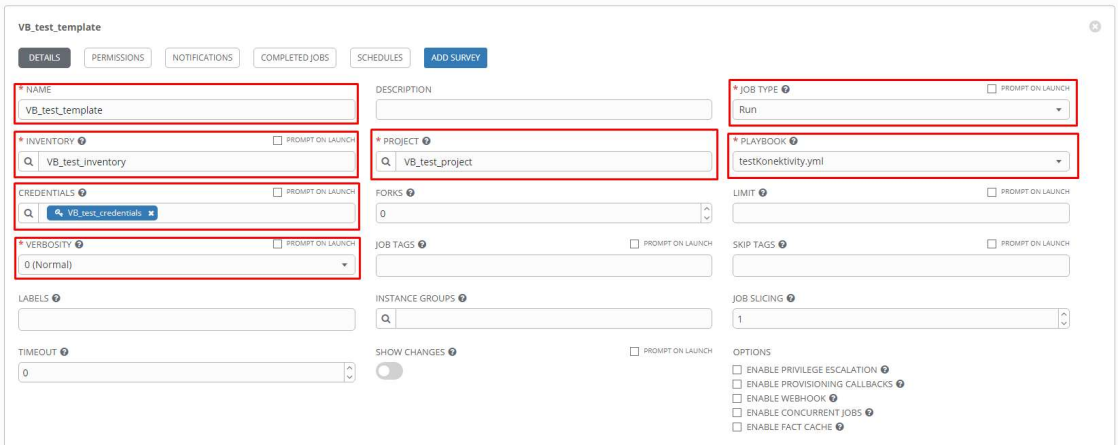


Figure 15 Creating a template using Ansible AWX

After you run the template, a window is displayed where you can follow the progress of the job being executed. When it finishes, it should look something like this.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

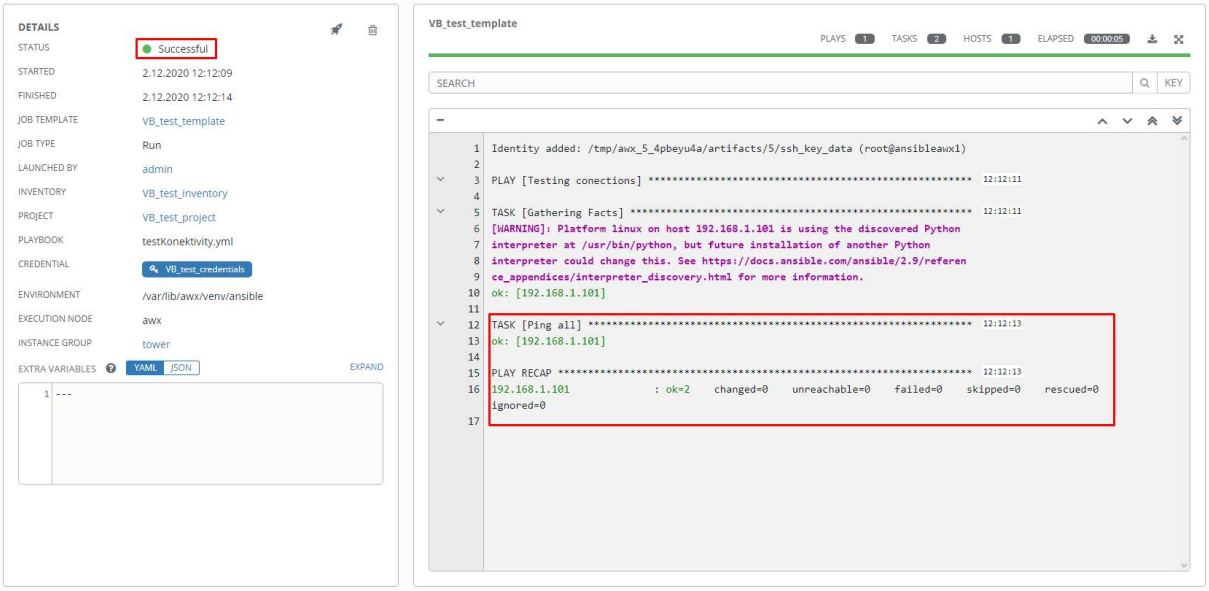


Figure 16 Executed template using Ansible AWX

Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2 ZABBIX

2.1 ZABBIX MONITORING

As part of the work with the Zabbix monitoring software, I deployed a Zabbix server on my local computer, which I used as a test environment for monitoring hosts and testing templates. I used the **Zabbix appliance**. It is the simplest way of deploying a Zabbix server and it contains all the necessary components (server, agent, database, gateway). The appliance runs on the CentOS Linux distribution. For the local deployment, I downloaded the .iso installation image from the official Zabbix website. I used this image to create a virtual machine in VirtualBox. The simplest way of monitoring hosts is by means of Zabbix agents, which collect metrics from end hosts and send them to the server. I tested this method on several Linux distributions. The deployment differed only in the use of different package managers. In the configuration, it is important to set the hostname and the location (IP address, domain) of the Zabbix server.

```
      !Example of installing the zabbix agent on the CentOS Stream distribution
      [root@CentOS_Stream-VM               ~]#               rpm               -Uvh
https://repo.zabbix.com/zabbix/5.2/rhel/8/x86_64/
      zabbix-release-5.2-1.el8.noarch.rpm
      [root@CentOS_Stream-VM ~]# dnf clean all
      [root@CentOS_Stream-VM ~]# dnf -y install zabbix-agent
      [root@CentOS_Stream-VM ~]# systemctl enable --now zabbix-agent

      !Configuration of the zabbix agent → /etc/zabbix/zabbix_agentd.conf
      ##### Passive checks related
      ### Option: Server
      Server=localhost

      ##### Active checks related
      ### Option: ServerActive
      ServerActive=localhost

      Hostname=<Zabbix_server_hostname>
```

In the web interface of the Zabbix server, the end hosts must be added to the configuration. Templates are used for this purpose. Various official and community templates can be found on the Zabbix website [5,6]. We also used some of the templates on the department's Ares KIS Zabbix server. Another useful feature is the web check, which verifies the availability of a website on the given end host.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

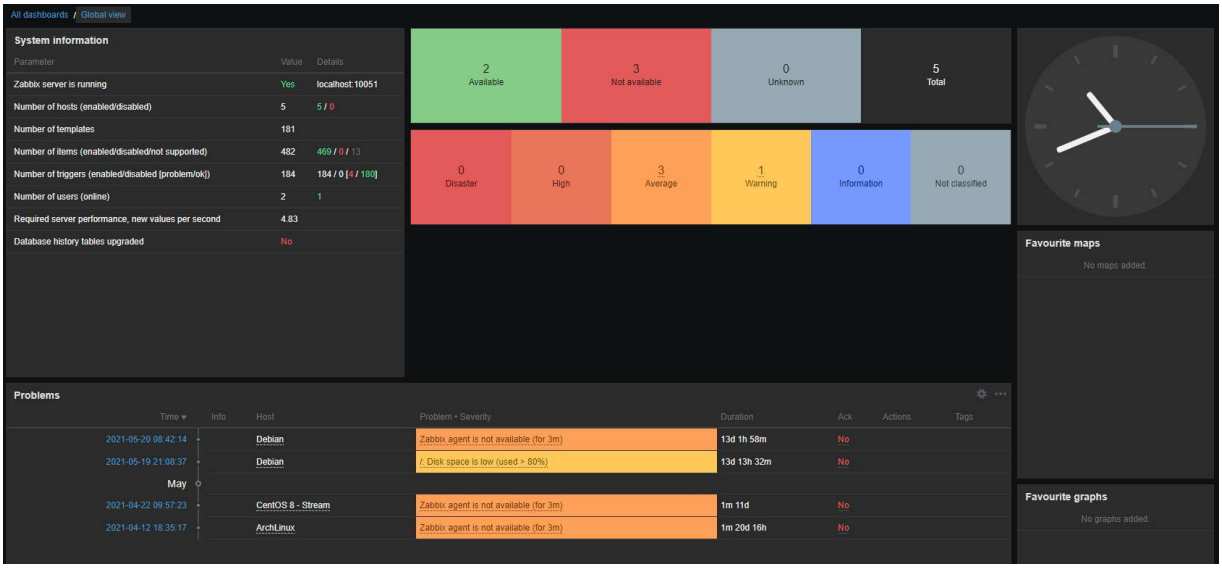


Figure 17 Main Zabbix dashboard on the local server

2.2 INTEGRATION OF ANSIBLE AND ZABBIX

In this chapter, we will discuss the possibility of using Ansible to manage the Zabbix server and the hosts managed by the Zabbix server.

2.2.1 Management using Ansible

The official Zabbix website (https://www.zabbix.com/integrations/ansible) contains links to roles that can be used to manage Zabbix solutions. Specifically, they can be used to manage:

- Zabbix Agent
- Zabbix Server
- Zabbix Proxy
- Zabbix Web

We mainly tested the management of Zabbix agents. For these, we tried to automate their deployment, as it is a recurring activity.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2.2.2 Zabbix Agent role

The role for managing Zabbix agents works by installing the Zabbix agent on the given host, subsequently configuring the Zabbix agent and adding the host to the Zabbix server by means of API calls. The role is available on GitHub and is regularly updated (https://github.com/dj-wasabi/ansible-zabbix-agent). We worked with version 2.2.0. A detailed description of the functionality of this role can also be found there. The playbook configuration for running the role can be found in Annex D.

2.2.2.1. Operating systems

The role can be run on the following operating systems. Specifically, we tested the Debian 10.6 Linux distribution.

- Red Hat
- Fedora
- Debian
- Ubuntu
- opensuse
- Windows (Best effort)
- macOS

2.2.2.2. Prerequisites

For the role to work, pip must be installed on the host. This is because python-netaddr is used for the correct handling of IP addresses. The page where the role is located also lists the versions with which the role is compatible. This concerns both the Zabbix version and the operating system version.

2.2.2.3. Installation

The installation is very simple. It is done using the command ansible-galaxy install dj-wasabi.zabbix-agent. This command installs the role into the default folder where roles are installed. In the Ansible server configuration file, it can be set so that roles are located in the /etc/ansible/roles folder. If this is not enabled, the role is automatically installed into the /root/.ansible/roles folder. Then it is sufficient simply to call the role in the playbook.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2.2.2.4. Variables

The role works on the basis of defining variables. Their description can be found on GitHub. All of them are located in **defaults/main.yml**. We will explain only those that we used. The parameters are set when the role is called in the playbook.

- zabbix_agent_server – IP address of the Zabbix server or Zabbix proxy. The host on which the role will be used must have access to the Zabbix server via port 10051.
- zabbix_agent_serveractive – IP address of the Zabbix server or Zabbix proxy for active checks.
- zabbix_url – URL at which the Zabbix web is available.
- zabbix_api_user – username for the API interface of the Zabbix server.
- zabbix_api_pass – password for logging in to the Zabbix server API.
- zabbix_create_host – present if I want to create the host or absent if I want to remove it.
- zabbix_api_create_hosts – if I want to enable API calls to the Zabbix server.
- zabbix_host_groups – the group to which I want to assign the host on the Zabbix server (e.g. Linux Servers).
- zabbix_link_templates – the template to be used for the given host.
- zabbix_agent_interfaces – the interface to be set as the monitoring interface on the Zabbix server. 4 types can be set (Zabbix Agent, SNMP, IPMI, JMX).

2.2.3 Community Zabbix

Modules documented in the Ansible documentation are used to manage Zabbix solutions. It contains a description of the individual modules as well as usage examples (https://docs.ansible.com/ansible/latest/collections/community/zabbix/index.html).

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2.2.4 Playbooks and annexes

2.2.4.1. Playbook used to migrate the Elastic database

```
- name: Migration of elasticsearch database
  hosts: elastic-new
  remote_user: mlkvy
  vars:
    host_get_index: 158.193.153.95
    host_post_index: 158.193.153.66

  tasks:
    - name: Load indices for migration
      uri:
        url: http://{{ host_get_index }}:9200/_cat/indices?h=idx
        return_content: yes
        method: GET
        timeout: 360
        body_format: json
      register: output

    - name: Set variable indexy_source
      set_fact:
        indexy_source: "{{ output.content.split('\n')}}"

    - name: Set variable indexy_dest
      set_fact:
        indexy_dest: "{{ indexy_source | replace('reindexed-v6-','')}}"

    - name: Migrating indices
      uri:
        url: http://{{ host_post_index }}:9200/_reindex?pretty
        method: POST
        timeout: 100000
        body:
          source:
            remote:
              host: "http://{{ host_get_index }}:9200"
            index: "{{ item.0 }}"
            size: 250
          dest:
            index: "{{ item.1 }}"
        body_format: json
        headers:
          Content-Type: "application/json"
      when: item.0 | regex_search("\w*")
      with_together:
        - "{{ indexy_source }}"
        - "{{ indexy_dest }}"
```


Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2.2.4.2. Playbook used for tagging based on a table

```
      - name: Tagging based on a table
        hosts: test-tag
        remote_user: mlkvy
        vars:
          url_user: admin
          url_pass: nbusr123
          url_host: 158.193.153.129
          path_csv: /home/mlkvy/files/KIS_2019_utoky.csv

        tasks:
          - name: Read users from CSV file and return a dictionary
            read_csv:
              path: "{{ path_csv }}"
              delimiter: ','
            register: input_csv

          - name: uri
            uri:
              url: http://{{ url_host }}:8005/addTags?expression=ip.src={{ item.Utocnik
}}&ip.dst={{  item.Obet  }}&stopTime={{  (  item.Koniec  |to_datetime).strftime('%s')
}}&startTime={{ ( item.Zaciatok |to_datetime).strftime('%s') }}&length={{ item.4 }}
              user: "{{ url_user }}"
              password: "{{ url_pass }}"
              method: POST
              timeout: 360
              body_format: json
              body:
                tags: "atack,{{ item.Utok }}"
              headers:
                Content-Type: "application/json"
            loop: "{{ input_csv.list }}"
```


Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2.2.4.3. Playbook used for tagging based on Suricata signatures

```
      - name: Tagging based on Suricata signatures
        hosts: test-tag
        remote_user: mlkvy
        vars:
          url_host: 158.193.153.91
          url_user: admin
          url_pass: nbusr123
          json_path: /home/mlkvy/files/eve.json

        tasks:
          - name: Load eve.json into register
            slurp:
              src: "{{ json_path }}"
            register: output
          - name: Set start_time variable
            set_fact:
              start_time:      "{{      output['content']      |      b64decode      |
regex_findall('(?<=\"timestamp\":\")[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-
9]{2}') }}"
              - name: Modify start_time variable
            set_fact:
              start_time: "{{ start_time | regex_replace('T', ' ')  }}"
          - name: Set stop_time variable
            set_fact:
              stop_time: "{{ ansible_date_time.date }} {{ ansible_date_time.time }}"
          - name: Set src_ip variable
            set_fact:
              src_ip:       "{{       output['content']       |       b64decode       |
regex_findall('(?<=\"src_ip\":\")[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}') }}"
              - name: Set src_port variable
            set_fact:
              src_port:       "{{       output['content']       |       b64decode       |
regex_findall('(?<=\"src_port\":)[0-9]{1,5}') }}"
              - name: Set dest_ip variable
            set_fact:
              dest_ip:       "{{       output['content']       |       b64decode       |
regex_findall('(?<=\"dest_ip\":\")[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}') }}"
          - name: Set dest_port variable
            set_fact:
              dest_port:      "{{      output['content']      |      b64decode      |
regex_findall('(?<=\"dest_port\":)[0-9]{1,5}') }}"
          - name: Set protocol variable
            set_fact:
              protocol:       "{{       output['content']       |       b64decode       |
regex_findall('(?<=\"proto\":\")\\w{3}') }}"

          - name: Set name variable
            set_fact:
              name:       "{{       output['content']       |       b64decode       |
regex_findall('(?<=\"signature\":\")[a-zA-Z0-9 \\-\\(\\)\\_/]*') }}"
          - name: uri
            uri:
              url:  http://{{  url_host  }}:8005/addTags?start=0&expression=ip.src={{
item.0  }}+%26%26+port.src={{  item.1  }}+%26%26+ip.dst={{  item.2  }}+%26%26+port.dst={{
item.3      }}+%26%26+ip.protocol={{      item.4      }}&startTime={{      (      item.5
|to_datetime).strftime('%s')  }}&stopTime={{  (  stop_time  |to_datetime).strftime('%s')
}}&length=50
              user: "{{ url_user }}"
              password: "{{ url_pass }}"
```


Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

```
            method: POST
            timeout: 360
            body_format: json
            body:
              tags: "{{ item.6 }}"
            headers:
              Content-Type: "application/json"
          with_together:
            - "{{ src_ip }}"
            - "{{ src_port }}"
            - "{{ dest_ip }}"
            - "{{ dest_port }}"
            - "{{ protocol }}"
            - "{{ start_time }}"
            - "{{ name }}"
```

2.2.4.4. Role for Zabbix management

```
    - hosts: host-local
      become: yes
      roles:
         - role: dj-wasabi.zabbix-agent
           zabbix_agent_server: 192.168.1.150
           zabbix_agent_serveractive: 192.168.1.150
           zabbix_url: http://192.168.1.150
           zabbix_api_use: true
           zabbix_api_user: Admin
           zabbix_api_pass: zabbix
           zabbix_api_create_hosts: true
           zabbix_create_host: present
           zabbix_agent_hostname: "{{ ansible_fqdn }}"
           zabbix_host_groups:
             - Linux servers
           zabbix_link_templates:
             - Linux by Zabbix agent
```

2.3 PROMETHEUS

Prometheus is one of the best open-source monitoring tools of the present day. The tool supports several cloud backend solutions and a dimensional data model. At its core, Prometheus collects metrics from HTTP endpoints, either directly or from special tools that collect metrics from the monitored devices, such as *Node-exporter*. It stores all the collected data and compares them with defined threshold values in order to generate alerts. If the threshold values are exceeded, it uses the *Alertmanager* tool to send alerts. The tool does not provide data visualization or a dashboard; instead, it uses another tool, Grafana, for this purpose. This system is also called Grafana-Alertmanager-Prometheus (GAP) and is often used in cloud solutions [22].

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

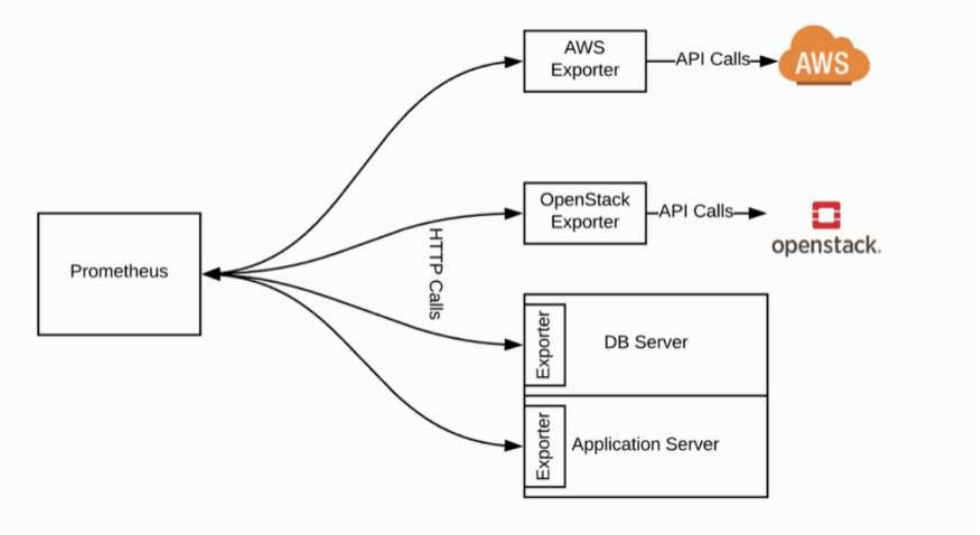


Figure 18 Conceptual architecture of Prometheus [22]

In the figure above, we can see how Prometheus can be implemented on a hybrid cloud. For each major cloud there is a so-called exporter, which needs to be installed on the host. The exporter is something like an agent for Prometheus, because it becomes an HTTP endpoint through which Prometheus collects the metrics. Thanks to this, Prometheus is extremely scalable, robust and stateless.

Figure 95 Infrastructure using VXLAN can serve as a good example for designing a monitoring deployment using the Prometheus tool. In this case, the network administrator will want to monitor the physical servers (nodes) on which one of the OpenStack services is running. Ideally, it is advisable to deploy GAP on a separate physical server. The simplest way of deploying the tool is by using docker-compose as a docker container. As part of my diploma thesis, I prepared configuration files that allow the deployment of the monitoring solution. This solution consists of the Prometheus tool itself, the Grafana tool for plotting graphs and creating dashboards, the Alertmanager tool for sending alerts and Node-exporter for collecting data from the monitored servers. The Alertmanager configuration is for the Slack communication platform, but there is also support for Teams, E-mail or SMS. In addition to the Prometheus server, node-exporter will need to be deployed on each monitored node. The whole solution is automated. The server on which GAP will run, as well as the monitored server, must have docker and docker-compose installed.

Monitoring virtual machines created on clouds is somewhat more complicated. As in the previous case, we would have one physical monitor server containing GAP. If we assume that the VMs will be on private subnets, the solution would be to create a monitoring VM on each cloud on which a Prometheus server would be located. This monitoring VM would be assigned a floating IP (in the case of OpenStack) and would thus offer a public endpoint for Grafana, located on the physical monitor server. In Grafana, each Prometheus server is set up as a *datasoruce*. Alertmanager supports multiple Prometheus instances. It will wait for reports of threshold values being exceeded and send alerts to the communication channels.

An alternative tool to the Prometheus solution may also be the Zabbix monitoring system or another system with equivalent functionality.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

REFERENCES

[1] **juju.is**. 2022. How to work with multiple users. [online]. 2022. Available at: https://juju.is/docs/olm/working-with-multiple-users

[2] **Community forum**. Multi-user basic setup. [online]. 2022. Available at: https://discourse.charmhub.io/t/multi-user-basic-setup/1195

[3] **openstack-bundles.** openstack-bundles/stable/openstack-base/. [online]. 2022. Available at: https://github.com/openstack-charmers/openstack-bundles/tree/master/stable/openstack-base

[4] **juju.is**. 2022. Building and deploying a mini charm. [online]. 2022. Available at: https://juju.is/tutorials/build-and-deploy-mini-charm#1-introduction

[5] **launchpad.net**. 2022. juju fails to provision lxd containers with lxd 4.18. [online]. 2022. Available at: https://bugs.launchpad.net/juju/+bug/1942864

[6] **launchpad.net**. 2022. charm no longer works with latest mysql-router version. [online]. 2022. Available at: https://bugs.launchpad.net/charm-mysql-router/+bug/1971565

[7] **docs.openstack.org**. 2022. Install OpenStack. [online]. 2022. Available at: https://docs.openstack.org/project-deploy-guide/charm-deployment-guide/yoga/install-openstack.html

[8] **opendev.org**. 2022. A modern dashboard for OpenStack - API server. [online]. 2022. Available at: https://opendev.org/openstack/skyline-apiserver#user-content-install--run

[9] **satishdotpatel.github.io**. 2022. Openstack Skyline Dashboard. [online]. 2022. Available at: https://satishdotpatel.github.io/openstack-skyline-dashborad/

[10] S. M. B. and M. P. Dhore, "Hybrid Cloud: A Solution to Cloud Interoperability," rev. International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018.

[11] J. C. L. E. O. and P. T. M. Denis, "Identity federation in OpenStack - an introduction to hybrid clouds," Journal of Physics: Conference Series, 2015.

[12] D. G. Dutt, Cloud Native Data Center Networking, Sebastopol: O'Reilly Media, 2020.

[13] L. Chao, Cloud Computing Networking: Theory, Practice, and Development, Boca Raton: CRC Press, 2016.

[14] J. C. L. E. O. and P. T. M. Denis, "Identity federation in OpenStack - an introduction to hybrid clouds," Journal of Physics: Conference Series, 2015.

[15] OpenStack, "Openstack Network architecture," 2021. [Online]. Available at: https://docs.openstack.org/arch-design/design-networking.html. [Cited 2021].

[16] J. Denton, Learning OpenStack Networking (Neutron) Second Edition, Birmingham: Packt Publishing Ltd., 2015.

[17] O. Khedher and C. D. Chowdhury, Mastering OpenStack, Birmingham: Packt Publishing, 2017.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

[18] OpenStack, "Virtual Private Network-as-a-Service (VPNaaS) scenario," Openstack, 2021. [Online]. Available: https://docs.openstack.org/neutron/rocky/admin/vpnaasscenario.html. [Cited 2021].

[19] Amazon Web Services, Inc., "AWS Site-to-Site VPN," 2021. [Online]. Available: https://docs.aws.amazon.com/vpn/latest/s2svpn/VPC_VPN.html. [Cited 2021].

[20] RDO, "Packstack: Create a proof of concept cloud," 2021. [Online]. Available: https://www.rdoproject.org/install/packstack/. [Cited 2021].

[21] ManageIQ, "ManageIQ," 2021. [Online]. Available: https://www.manageiq.org/docs/get-started/. [Cited 2021].

[22] Prometheus Authors, "https://prometheus.io/," 2021. [Online]. Available: Prometheus. [Cited 2021].

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC