PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené
ako otvorená cloudová služba s prvkami strojového učenia

Previazané s balíkom KPB1 – Návrh virt. riešenia

Previazané s výstupom
V3 – Popis vybraného riešenia spĺňajúceho stanovené kritéria

typ – dokumentácia

Riešenia Ansible a Zabbix - experiment

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Tento dokument sumarizuje experimenty s automatizáciou s Ansible, monitoringom cez Zabbix a ich prepojením.

1 ANSIBLE

V nasledujúcej časti predstavíme komponent Ansible,

1.1 INŠTALÁCIA ANSIBLE AWX NA LINUXOVEJ DISTRIBÚCII DEBIAN

1.1.1 Požiadavky potrebné na inštaláciu

1.1.1.1 Systémové požiadavky

- Minimálne 4GB operačnej pamäte.
- Minimálne dve jadrá procesora.
- Minimálne 20GB úložného priestoru na disku.
- Funkčný Docker, Openshift alebo Kubernetes.
- Pri použití externej PostgreSQL databázy, použiť minimálne verziu 10+.

1.1.1.2 Verzie systémov použité pri inštalácii

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

1.1.2 Inštalácia systému Ansible AWX

Inštalácia prebiehala na Linuxovej distribúcii **Debian 10.6**. Nainštalovať Ansible AWX je možné na ktorejkoľvek distribúcii Linuxu, na ktorej je možné nainštalovať Ansible server. Na nainštalovanie Ansible AWX sme použili Docker a docker-compose. Dá sa nainštalovať aj pomocou Openshift alebo Kubernetes. Kroky inštalácie k jednotlivým možnostiam sa nachádzajú na https://github.com/ansible/awx.

Teraz si rozoberieme potrebné kroky na nainštalovanie Ansible AWX pomocou Dockeru a docker-compose.

1.1.2.1. Inštalácia systému Docker

- Aktualizovanie repozitárov.

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

- Inštalácia potrebných dependecies na nainštalovanie systému Docker.

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

- Stiahnutie a pridanie Docker GPG kľúča.

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

- Pridanie Docker repozitára.

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

- Aktualizovanie repozitárov.

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

- Inštalácie Docker a docker-compose.

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

- Overenie inštalácie

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

1.1.2.2. Inštalácia systému Ansible

- Pridanie Ansible repozitára.

```
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
```

- Aktualizácia repozitárov.

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

- Inštalácie Ansible.

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

- Overenie inštalácie.

```
ansible –version
```

1.1.2.3. Inštalácia Node.js a Python knižnice

- Inštalácia Node.js, NPM a Git.

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

- Inštalácia python modulu pre Docker a potrebných dependecies.

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

- Ďalej je potrebné aktualizovať verziu pythonu.

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

- Zistenie docker-compose verzie

```
docker-compose version
```

- Nainštalovanie docker-compose python modulu na základe verzie docker-compose.

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

1.1.2.4. Inštalácia Ansible AWX

- Stiahnutie najnovšej verzie Ansible AWX z githubu.

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

- Potrebné je prejsť do zložky awx/installer/.

```
cd awx/installer/
```

- Vygenerovanie hesla pre tajný kľuč.

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

- Vytvorenie kópie pôvodného inventory súboru.

```
ocp inventory inventory.old
```

- Konfigurácia inventory súboru. Odporúčame vymazať obsah súboru a pridať do neho iba potrebné veci, pre lepší prehľad.

```
nano inventory
```

- Obsah súboru inventory

```
      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
      #kľuč, ktorý sme si vygenerovali pomocou pwgen
      secret_key=**********
```

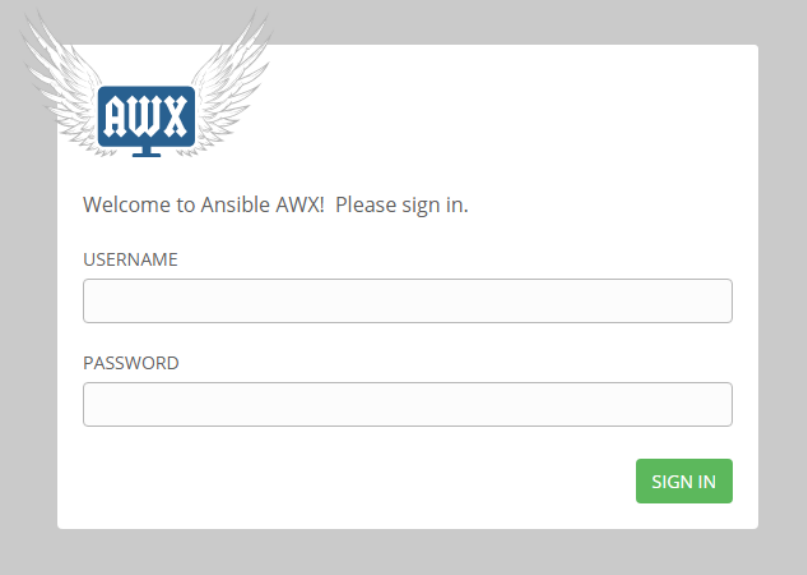
- Po inštalácii vznikali problémy, že awx_task kontajner nepočkal na awx_postgres kontajner na dokončenie inicializácie a migrácie. To spôsobilo problém so synchronizovaním databázy a Ansible AWX nebolo možné rozbehnúť. Tento problém je popísaný v reporte https://github.com/ansible/awx/issues/8300. Aby sa predišlo tomuto problému je potrebné:
  - modifikovať súbor:
    - awx/installer/roles/local_docker/templates/docker-compose.yml.j2
  - v ňom nájsť príkaz na spustenie awx_task
    - command: /usr/bin/launch_awx_task.sh
  - a nahradiť ho príkazom
    - command: /bin/bash -c "sleep 30 && exec /usr/bin/launch_awx_task.sh"
  - to spôsobí to, že uspí spustenie awx_task na potrebný čas.
- Inštalácia Ansible AWX pomocou playbooku
  - ansible-playbook -i inventory install.yml

- Po úspešnej inštalácii by sa mali vytvoriť potrebné kontajnery (awx_task, awx_web, awx_redis a awx_postgres), ktoré sa dokážu overiť príkazom
    - docker ps
- Po inštalácii bude prebiehať migrácia databázy. Toto sa musí dokončiť, aby bolo možné pristúpiť k webovému rozhraniu. Na monitorovanie progresu je možné použiť príkaz
    - docker logs -f awx_task

1.2 PRÁCA S ANSIBLE AWX

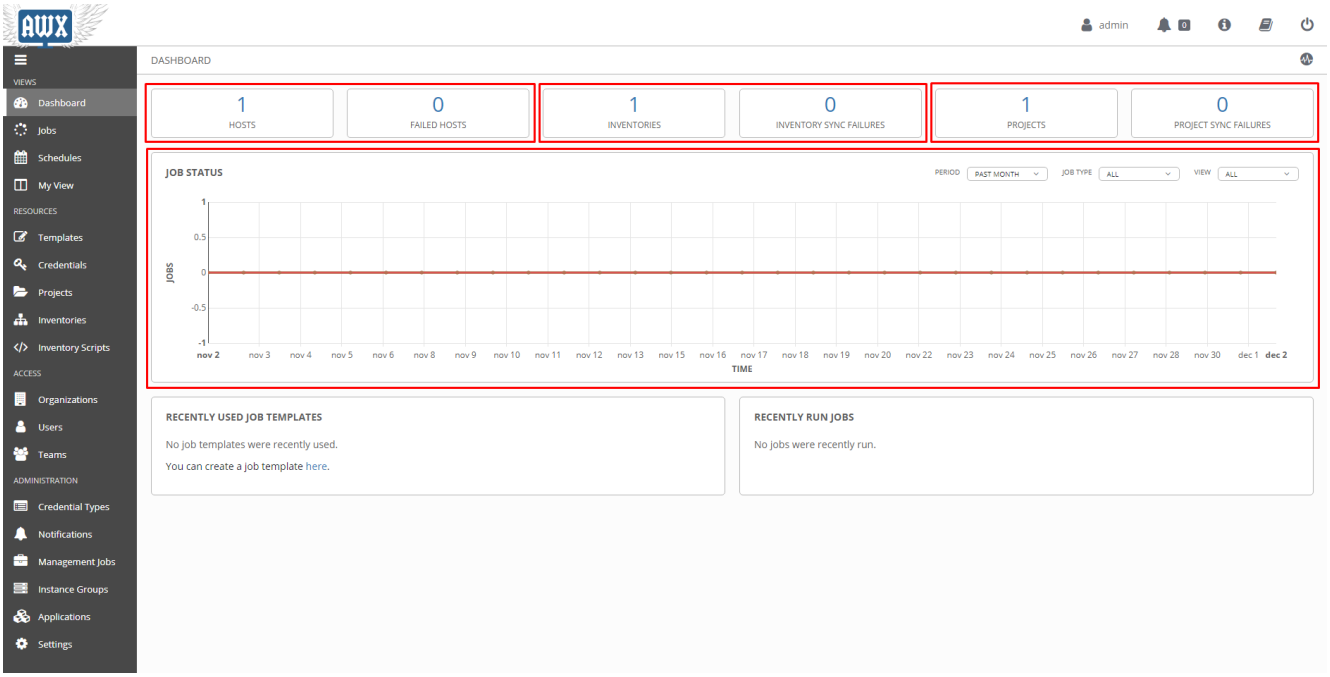
1.2.1 Prihlásenie do Ansible AWX a základné pojmy

Do Ansible AWX sa dá prihlásiť prostredníctvom webového prehliadača. Stačí zadať adresu alebo doménu na, ktorej vám beží Ansible server. Po načítaní stránky sa zobrazí prihlasovacie okno. Meno a heslo sú podľa toho, aké ste nastavili v konfiguračnom súbore *inventory*. Jedná sa o premenné *admin_user* a *admin_password*.



Obrázok 1 Prihlasovacie okno

Po úspešnom prihlásení sa zobrazí dashboard na, ktorom je vidieť počet hostov, invetories a projects. Taktiež, sa tu nachádza aj graf, ktorý ukazuje spustené automatizačné úlohy (jobs).



Obrázok 2 Ansible AWX dashboard

Na ľavej strane môžete vidieť rôzne možnosti na výber. Teraz si ich v skratke vysvetlíme a povieme čo sa v nich nachádza.

- **Dashboard** – zhrnutie jednotlivých hostov, inventáre a projekty. Graf úspešných a neúspešných úloh.
- **Jobs** – obsahuje list úloh, ktoré boli vykonané. Taktiež sa tieto úlohy dajú znovu spustiť.
- **Shedules** – Tu sa dá nastaviť spúšťanie jednotlivých úloh na základe času.
- **My View** – obsahuje jednoduchý zoznam rôznych úloh, ktoré je možné spustiť. Táto možnosť je určené pre tých, ktorý sa nevyznajú v nastavovaní jednotlivých templates, projektov a inventárov ale potrebujú spustiť iba nejaké úlohy.

- **Templates** – obsahuje vytvorené templates, ktoré sa vzťahujú na jednotlivé projekty a inventáre. Taktiež sa tu dajú spustiť, aby vykonali zadané úlohy.
- **Credentials** – tu sa dajú definovať rôzne spôsoby autentifikácie, ktoré má používať Ansible AWX. Jednotlivé credentials sa dajú potom priradiť k templatom, inventárom alebo projektom, ktoré potrebujú autentifikáciu.
- **Projects** – tu sa definujú projekty. Jednotlivé projekty obsahujú playbooky, ktoré sa potom používajú v templatoch.

- **Inventories** – obsahuje hostov, na ktorých sa vykonávajú jednotlivé úlohy.
- **Inventory scripts** – scripty, ktoré automaticky vedia pridávať hostov z rôznych prostredí.

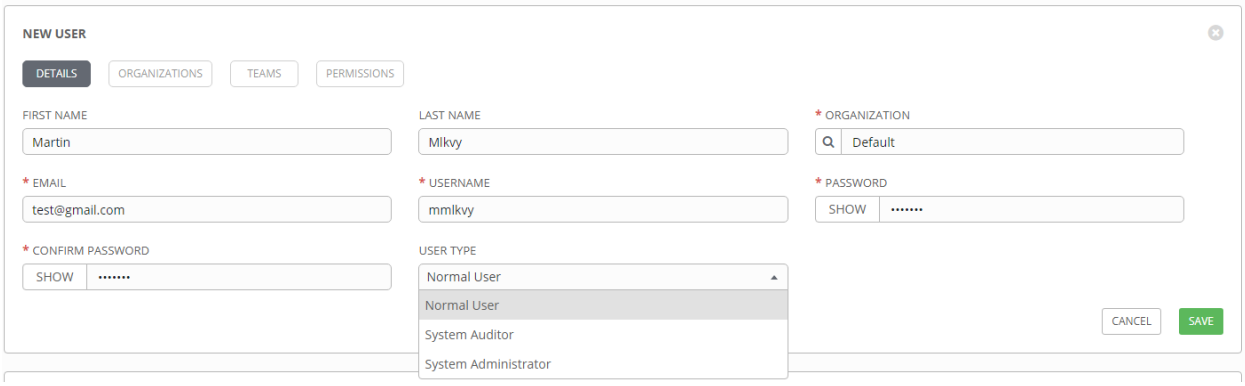
- **Organizations** – logické zoskupenie používateľov, tímov, projektov a inventárov.
- **Users** – tu sa dajú definovať používateľa, ktorý sa dokážu prihlásiť do Ansible AWX a spravovať veci v ňom.
- **Teams** – je to podmnožina organizácie. Taktiež obsahuje používateľov, projekty a rôzne credentials a permisions. Credentials je možné prideliť rovno celému tímu a nie každému používateľovi samostatne.

- **Administration –** tu sa nachádzajú rôzne možnosti, pre správu Ansible AWX. Napríklad nastavenie notifikácií, rôznych aplikácii a samotné nastavenia Ansible AWX.

1.2.2 Práca s Ansible AWX

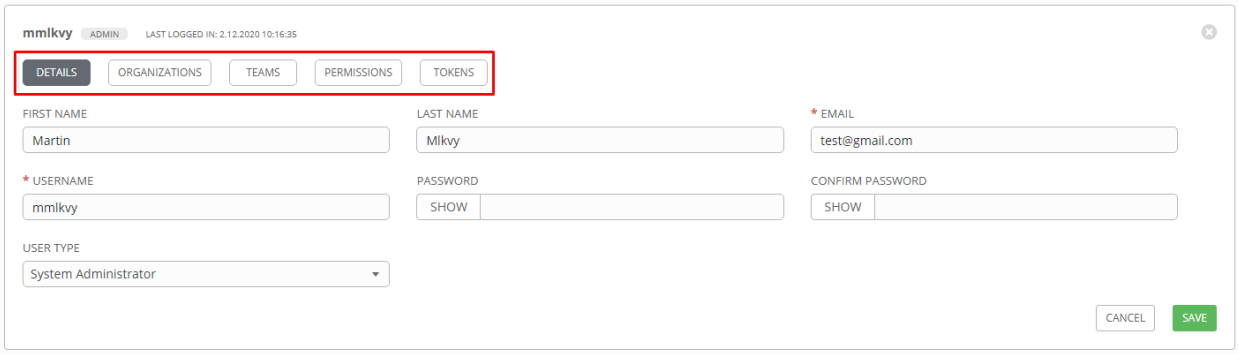
1.2.2.1. Vytvorenie používateľského konta

Používateľské konto sa dá vytvoriť v záložke Users. Pomocou tohto konta sa dá potom prihlásiť do Ansible AWX. Dá sa nastaviť meno, priezvisko, email, prihlasovacie meno, heslo a typ používateľa (admin/obyčajný používateľ).



Obrázok 3 Vytvorenie používateľského konta

Potom ako účet vytvoríme a uložíme, tak ho vieme pridať do organizácie, tímu alebo mu priradiť rôzne povolenia.



Obrázok 4 Priradenie rôznych možností ku používateľskému kontu

1.2.3 Spustenie playbooku nad hostom pomocou Ansible AWX

1.2.3.1. Vytvorenie ssh pripojenia medzi serverom a klientom

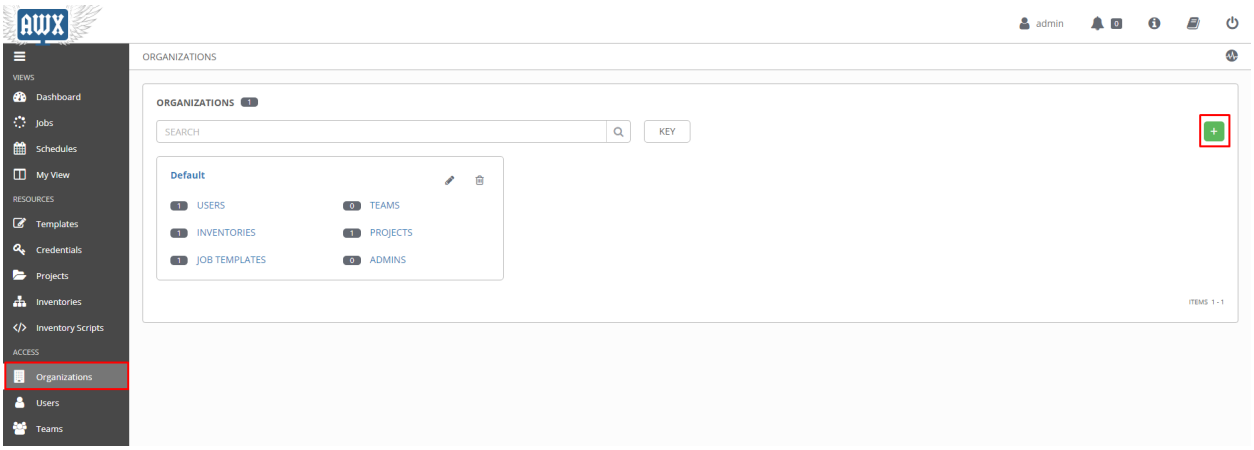
Ako prvé je potrebné zabezpečiť ssh spojenie medzi klientom a Ansible serverom. Obaja musia mať nainštalované ssh.

Najskôr si vygenerujeme ssh kľuč na servery. Urobíme to pomocou príkazu **ssh-keygen**.

Potom ako sme si vygenerovali kľúč, tak pošleme verejný kľúč servera na hosta, ktorého chceme spracovávať pomocou Ansible servera. To spravíme pomocou príkazu **ssh-copy-id root@ 192.168.1.101**.

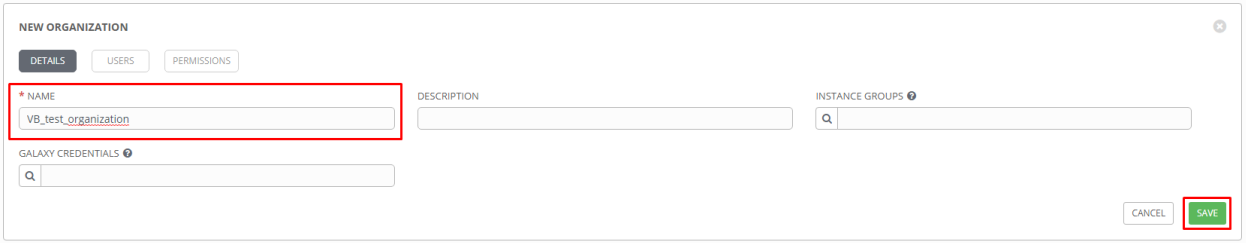
1.2.3.2. Vytvorenie organizácie v Ansible AWX

Prihláste sa do Ansible AWX. V bočnom panely je potrebné vybrať záložku Organizations. Tam vytvoríme novú organizáciu.



Obrázok 5 Vytvorenie organizácie pomocou Ansible AWX 1

Treba zadať meno organizácie a uložiť ju.



Obrázok 6 vytvorenie organizácie pomocou Ansible AWX 2

1.2.3.3. Vytvorenie projektu a priradenie organizácie k nemu

V projektoch Ansible AWX sa nachádzajú jednotlivé playbooky. Popíšeme si dva spôsoby ako nahrať playbook na Ansible AWX (z lokálneho priečinka alebo z repozitára na githube).

1. Použitie lokálneho priečinka.

Ako prvé je potrebné vytvoriť priečinok na Ansible servery tam kde majú projekty working directory. To sa definovalo pri inštalácii Ansible AWX v súbore *inventory*. Konkrétne sa jednalo o premennú *project_data_dir.*

mkdir /var/lib/awx/projects/test_konektivity_p

Akonáhle máme vytvorený priečinok, tak do neho môžeme pridať súbor formátu .yaml (.yml).

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

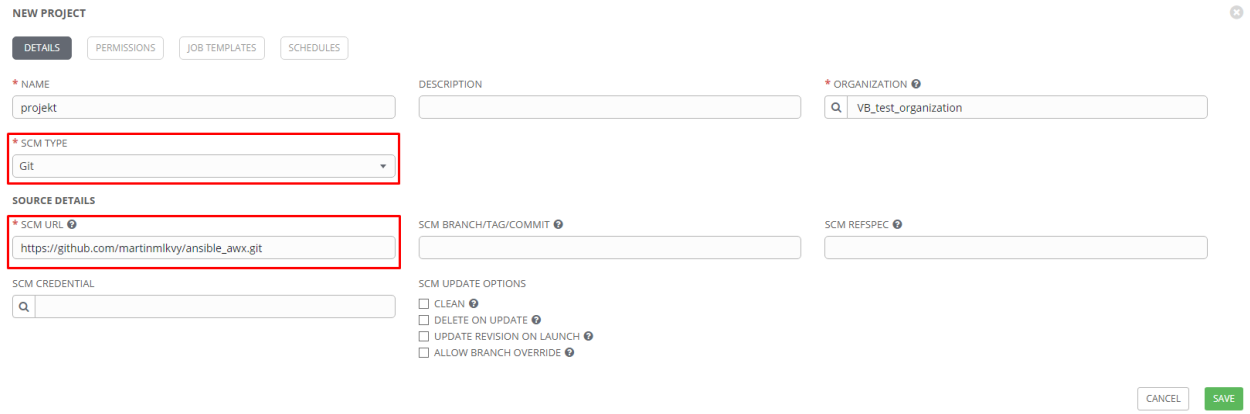
```
- name: Testing conections
  hosts: all
  remote_user: root


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

Teraz môžeme pomocou Ansible AWX vytvoriť projekt. Urobíme to tak, že klikneme na ľavej lište na záložku Projects a tam vytvoríme nový projekt. Tam mu zadáme názov, vyberieme organizáciu, SCM type nastavíme na manual a vyberieme priečinok kde sme vytvorili náš playbook.

Obrázok 7 Vytvorenie projektu s lokálnym úložiskom

2. Použitie GitHub repozitára.

Ansible AWX si vie stiahnuť playbooky z GitHub repozitárov. Na to aby sa vedel dostať do repozitára, tak daný repozitár musí byť verejný. Potom pri vytváraní projektu treba zvoliť ako SCM Type Git a vložiť url adresu repozitára, v ktorom sa nachádzajú dané playbooky.



Obrázok 8 Vytvorenie Projektu pomocou GitHubu

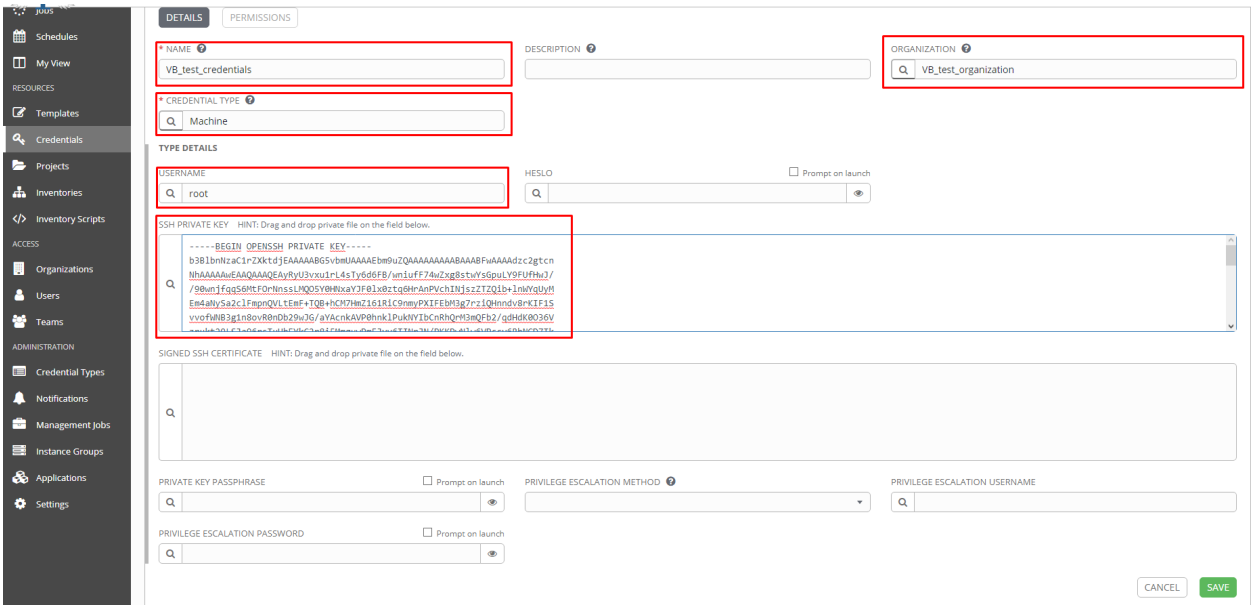
1.2.3.4. Vytvorenie credentials

Tieto credentials sa použijú pri spúšťaní playbookov na ansible hostovi.

Najskôr musíme získať privátny kľúč ansible servera. O kopírujte si ho, neskôr ho budeme importovať do Ansible AWX.

cat /root/.ssh/id_rsa

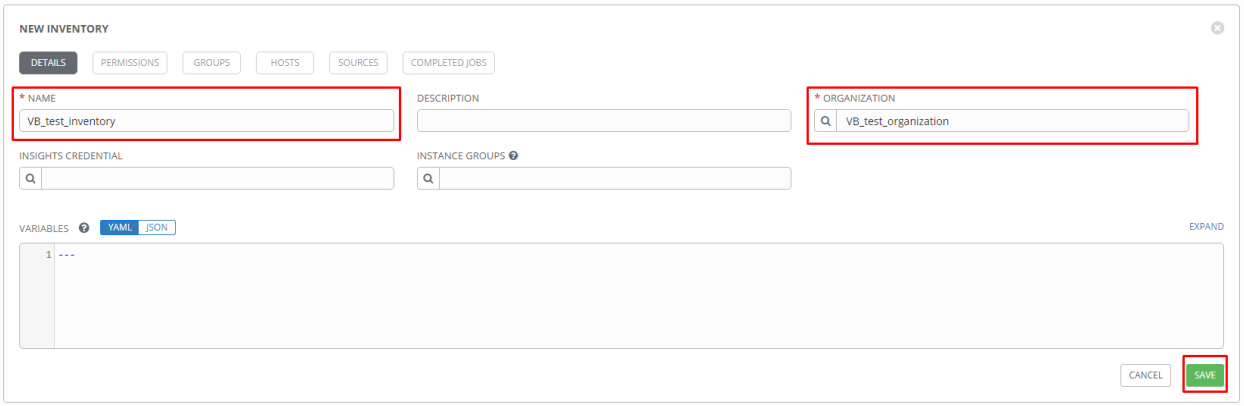
Teraz v Ansible AWX v ľavej lište zvolíme záložku Credentials a tam vytvoríme novú inštanciu. Credential type zvolíme machine, lebo sa bude týkať iba nášho hosta. Zadáme prihlasovacie meno, akým sa budeme prihlasovať na hosta (root). Priložíme privátny kľúč nášho Ansible servera.



Obrázok 9 Vytvorenie credentials v Ansible AWX

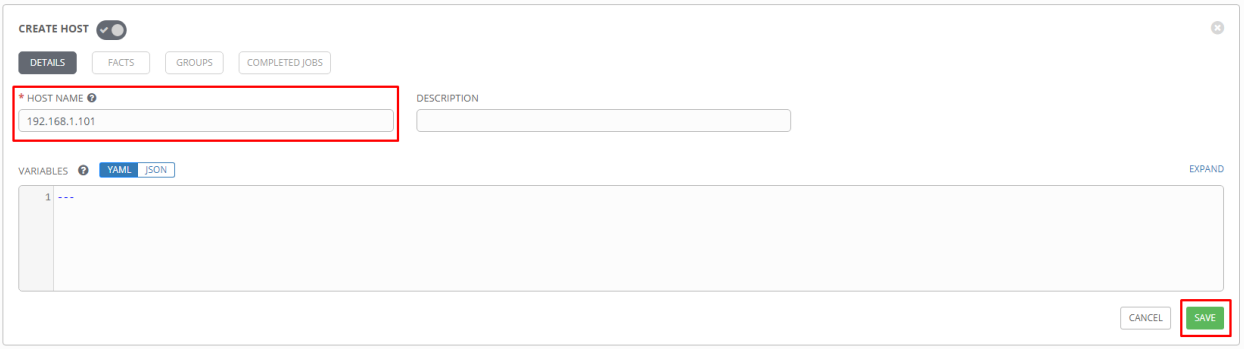
1.2.3.5. Vytvorenie inventára a priradenie hosta

V Ansible AWX klikneme na záložku Inventory a ta vytvoríme nový inventár a uložíme ho.



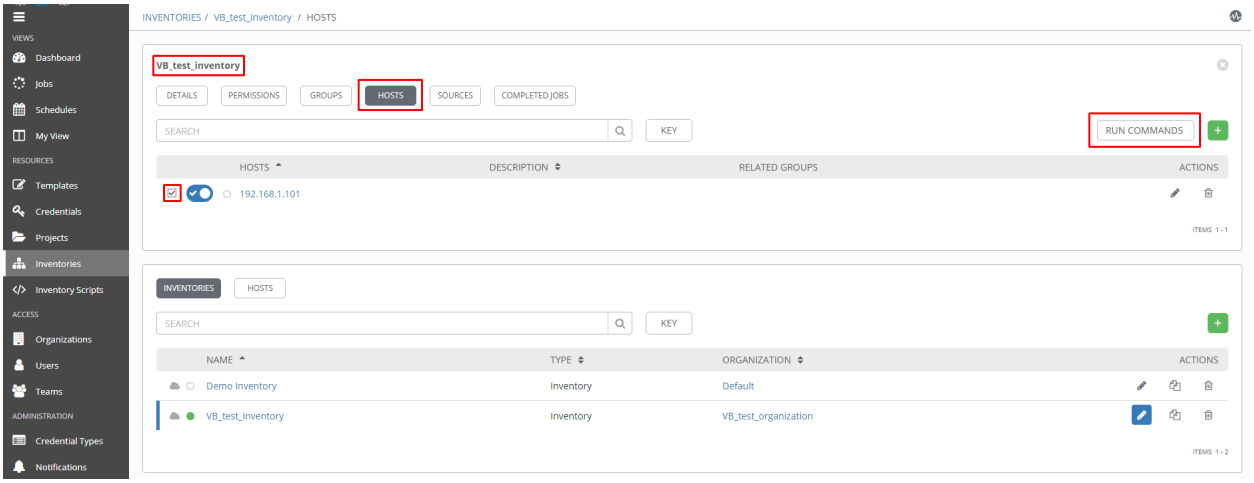
Obrázok 10 Vytvorenie inventára v Ansible AWX

Potom ako sme uložili vytvorený inventár tak prejdeme do záložky hosts v inventári a vytvoríme nového hosta a uložíme ho.



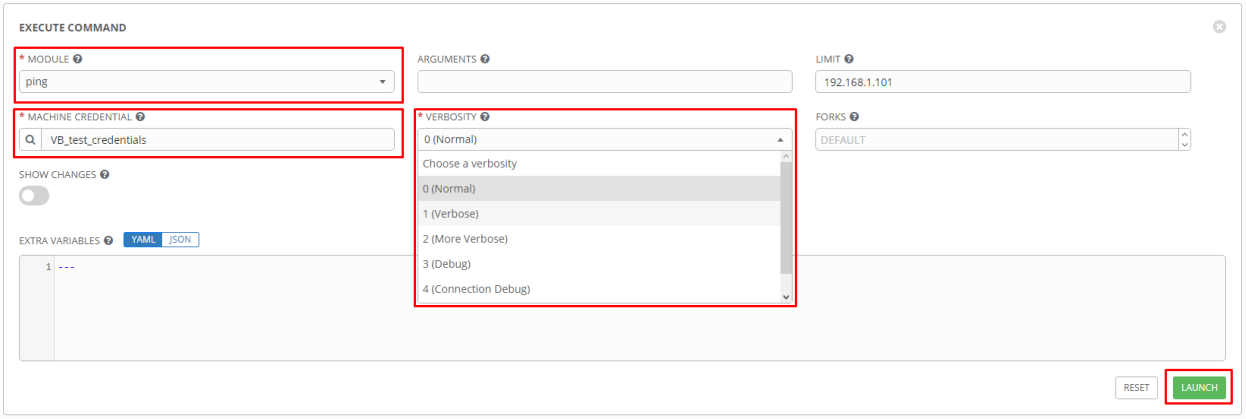
Obrázok 11 Priradenie hosta do inventára

Teraz vyskúšame spustiť nad klientom ping modul aby sme otestovali konektivitu a spojenie pomocou ssh medzi Ansible AWX a Ansible hostom. Prejdite na Inventories, otvorte novo vytvorený inventár a prejdite do záložky hosts. Tam vyberte hosta, ktorého sme vytvorili a stlačte run command.



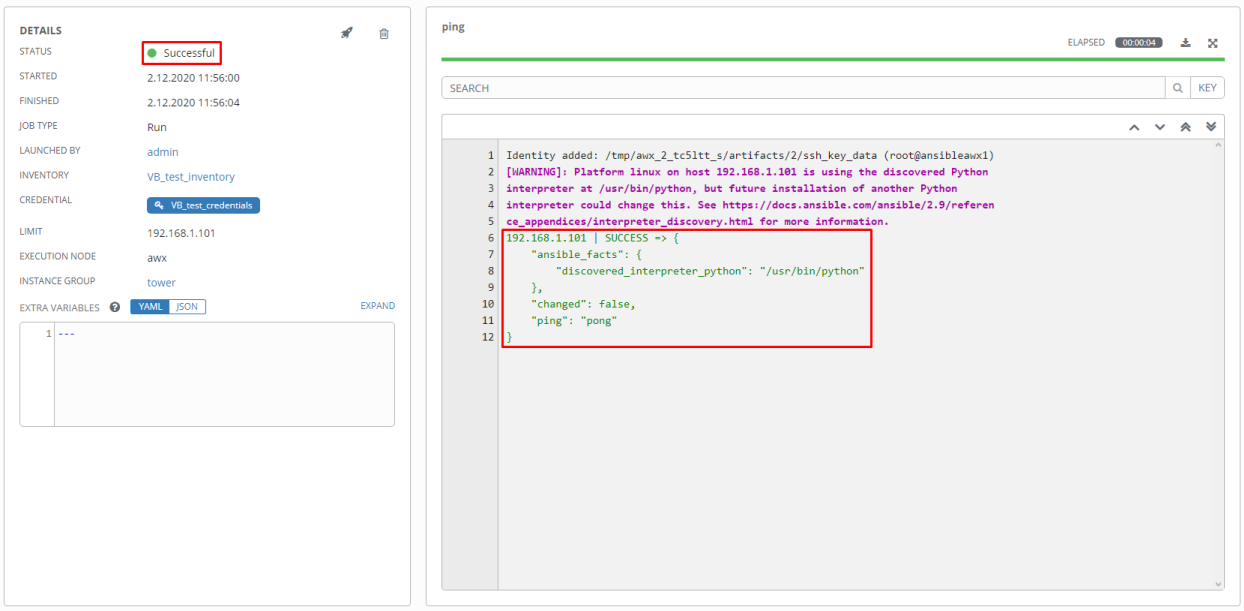
Obrázok 12 Spustenie príkazu nad hostom v inventári

Zobrazí sa vám okno, v ktorom vyberieme modul ping a machine credentials. Môžete taktiež zmeniť verbosity výpisu. Potom je potrebné spustiť úlohu pomocou tlačidla launch.



Obrázok 13 Spustenie ping modulu nad hostom

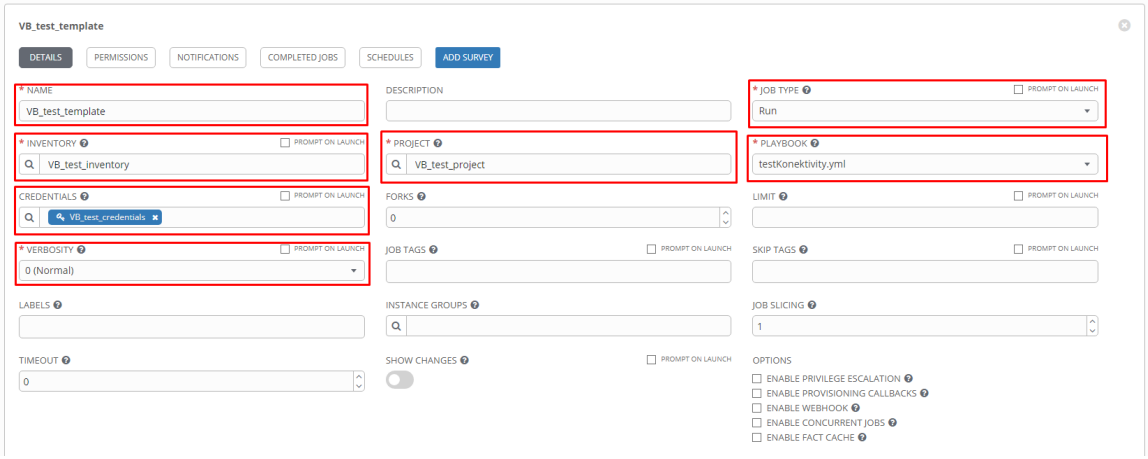
Potom sa zobrazí okno, v ktorom môžete sledovať priebeh vykonávania úlohy. Po úspešnom vykonaní by to malo vyzerať takto.



Obrázok 14 Vykonaný ping modul nad hostom

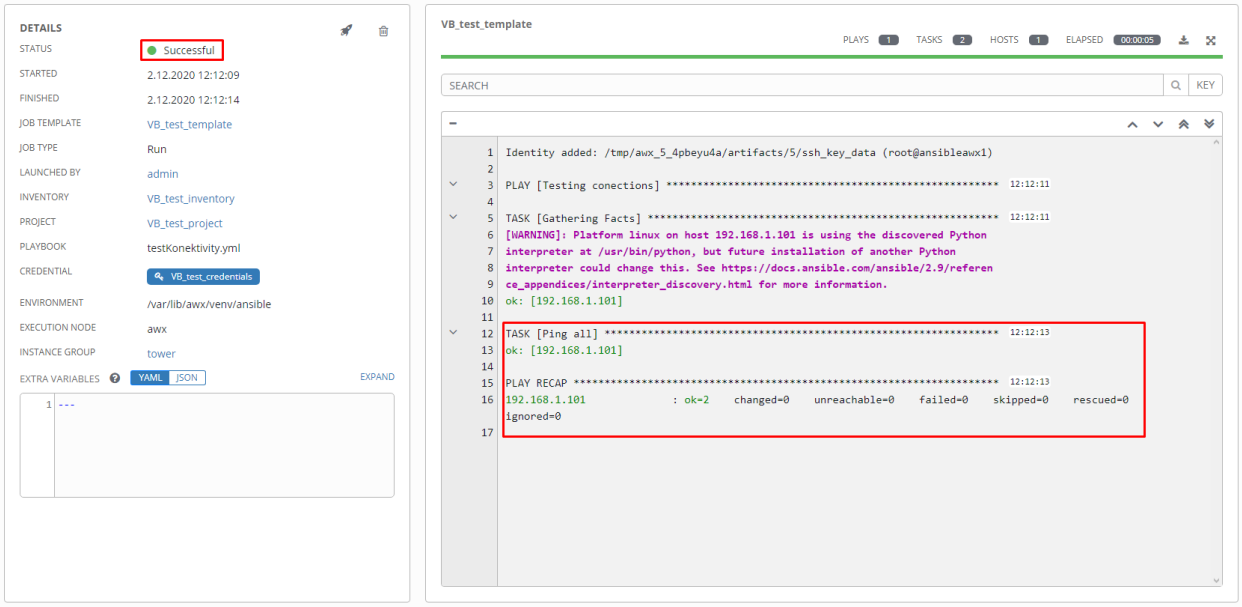
1.2.3.6. Vytvorenie template pomocou Ansible AWX

V Ansible AWX zvolíme záložku Templates a tam vytvoríme novú job template. Treba vyplniť potrebné kolónky ako vidíte na obrázku. Taktiež sa dá vybrať spôsob verbosity. Akonáhle máte všetko nastavené tak treba template uložiť. Potom ako bude uložená tak ju môžeme spustiť pomocou Launch



Obrázok 15 Vytvorenie template pomocou Ansible AWX

Potom ako spustíte template sa zobrazí okno kde môžete sledovať priebeh vykonávanej úlohy. Keď dobehne tak by to malo vyzerať nejak takto.



Obrázok 16 Vykonaný template pomocou Ansible AWX

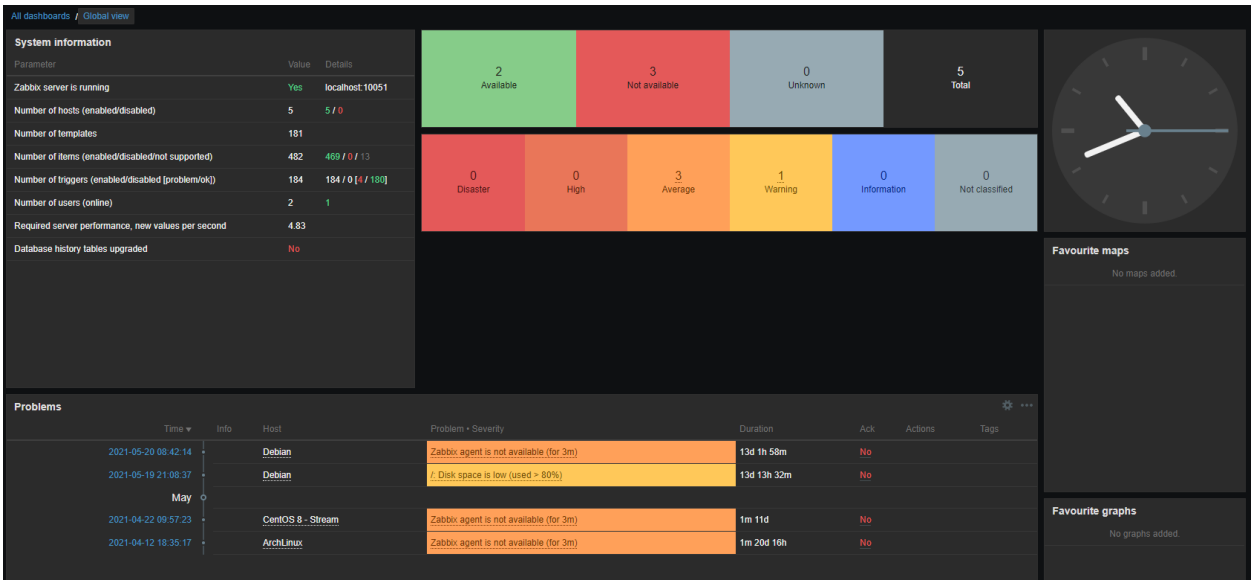
2 ZABBIX

2.1 MONITORING ZABBIX

V rámci práce s monitorovacím softvérom Zabbix som nasadil Zabbix server na svojom lokálnom počítači, ktorý som používal ako testovacie prostredie na monitorovanie hostov a testovanie šablón (templates). Použil som **Zabbix appliance**. Je to najjednoduchší spôsob nasadenia Zabbix servera a obsahuje všetky potrebné komponenty (server, agent, databáza, gateway). Appliance beží na Linuxovej distribúcií CentOS. Pre lokálne nasadenie som si stiahol .iso inštalačný obraz (image) z oficiálnej Zabbix stránky. Tento obraz som použil na vytvorenie virtuálneho stroja vo Virtualboxe. Najjednoduchší spôsob monitorovania hostov je pomocou Zabbix agentov, ktorí zbierajú metriky z koncových hostov a posielajú ich na server. Tento spôsob som otestoval na viacerých Linux distribúciách. Nasadenie sa líšilo iba v použití rôznych package manažérov. V konfigurácií je dôležité nastaviť hostname a lokalitu (IP adresa, doména) Zabbix servera.

```
      !Príklad inštalácie zabbix agenta na CentOS Stream distribúcií
      [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

      !Konfigurácia zabbix agenta → /etc/zabbix/zabbix_agentd.conf
      ##### Passive checks related
      ### Option: Server
      Server=localhost

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

      Hostname=<Zabbix_server_hostname>
```

Vo webovom rozhraní Zabbix servera sa musia koncové hosty pridať do konfigurácie. Na to sa využívajú šablóny. Na Zabbix stránkach sa dajú nájsť rôzne oficiálne a komunitné šablóny [5,6]. Niektoré z šablón sme použili aj na katedrovom Ares KIS Zabbix serveri. Užitočnou funkciou je aj web check, ktorý overuje dostupnosť webovej stránky na danom koncovom hostovi.



Obrázok 17 Hlavný Zabbix dashboard na lokálnom serveri

2.2 INTEGRÁCIA ANSIBLE A ZABBIX

V tejto kapitole si povieme o možnosti správy Zabbix servera a hostov, ktorých má na správu Zabbix server pomocou Ansible.

2.2.1 Správa pomocou Ansible

Na oficiálnej stránke Zabbix-u(https://www.zabbix.com/integrations/ansible) sa nachádzajú odkazy na role, ktoré sa dajú použiť na správu Zabbix-ových riešení. Konkrétne sa dajú použiť na správu:

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

My sme testovali hlavne správu Zabbix agentov. Pri nich sme sa snažili zautomatizovať ich nasadenie, keďže sa jedná o opakujúcu sa činnosť.

2.2.2 Rola Zabbix Agent

Rola na správu Zabbix agentov funguje na spôsobe nainštalovania Zabbix agenta na danom hostovy, následnej konfigurácie Zabbix agenta a pridaniu hosta na Zabbix server pomocou API volaní. Rola sa nachádza na GitHube a je pravidelne aktualizovaná(https://github.com/dj-wasabi/ansible-zabbix-agent). My sme pracovali s verziou 2.2.0. Taktiež sa tu nachádza aj detailný popis funkcionality tejto role. Nastavenie playbooku na spustenie role sa nachádza v prílohe D.

2.2.2.1. Operačné systémy

Daná rola sa dá spustiť na týchto operačných systémoch. My sme konkrétne testovali linuxovú distribúciu Debian 10.6.

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

2.2.2.2. Podmienky

Nato aby rola mohla fungovať tak je potrebné aby na hostovy bol nainštalovaný pip. To z toho dôvodu, že sa používa python-netaddr pre správnu správu IP adries. Na stránke kde sa nachádza rola, sú aj vypísané verzie s ktorými je daná rola kompatibilná. To sa týka verzie Zabbix a aj verzie operačného systému.

2.2.2.3. Inštalácia

Inštalácia je veľmi jednoduchá. Robí sa to pomocou príkazu ansible-galaxy install dj-wasabi.zabbix-agent. Tento príkaz nainštaluje rolu do predvoleného priečinka kde sa inštalujú role. V konfiguračnom súbore Ansible servera sa dá nastaviť aby sa role nachádzali v priečinku /etc/ansible/roles. Ak toto nie je povolené tak sa automaticky nainštaluje do priečinka /root/.ansible/roles. Potom stačí iba v playbooku danú rolu zavolať.

2.2.2.4. Premenné

Rola funguje na základe definovania premenných. Ich popis sa nachádza na GitHube. Všetky sa nachádzajú v priečinku **defaults/main.yml**. My si vysvetlíme iba tie čo sme používali. Parametre sa nastavujú pri volaní role v playbooku.

- zabbix_agent_server – IP adresa Zabbix servera alebo Zabbix proxy. Host na ktorom sa bude rola používať musí mať prístup na Zabbix server cez port 10051.
- zabbix_agent_serveractive – IP adresa Zabbix serveru alebo Zabbix proxy pre aktívnu kontrolu.
- zabbix_url – URL na ktorej je dostupný Zabbix web.
- zabbix_api_user – prihlasovacie meno na API rozhranie Zabbix serveru.
- zabbix_api_pass – heslo na prihlásene do API Zabbix serveru.
- zabbix_create_host – present ak chcem vytvoriť hosta alebo absent ak ho chcem odstrániť.
- zabbix_api_create_hosts – ak chcem povoliť API volania na Zabbix server.
- zabbix_host_groups – skupina do ktorej chcem priradiť hosta na Zabbix server (napr. Linux Servers).
- zabbix_link_templates – template, ktorý sa použije pri danom hostovi.
- zabbix_agent_interfaces – rozhranie, ktoré sa nastaví ako monitorovacie rozhranie na Zabbix servery. Dajú sa nastaviť 4 typy (Zabbix Agent, SNMP, IPMI, JMX).

2.2.3 Community Zabbix

Na správu Zabbix-ových riešení sa používajú moduly, ktoré sú zdokumentované v dokumentácii Ansible. Nachádza sa tu popis jednotlivých modulov, ale aj príklady použitia (https://docs.ansible.com/ansible/latest/collections/community/zabbix/index.html).

2.2.4 Playbooky a prílohy

2.2.4.1. Playbook použitý na migráciu Elastic databázy

```
- name: Migracia elasticsearch databazy
  hosts: elastic-new
  remote_user: mlkvy
  vars:
    host_get_index: 158.193.153.95
    host_post_index: 158.193.153.66

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

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

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

    - name: Migrovanie indexov
      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 }}"
```

2.2.4.2. Playbook použitý na značkovanie pomocou tabuľky

```
      - name: Tagovanie na základe tabulky
        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 }}"
```

2.2.4.3. Playbook použitý na značkovanie pomocou signatúr zo Suricaty

```
      - name: Tagovanie na základe signatur zo suricaty
        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 }}"
              method: POST
              timeout: 360
```

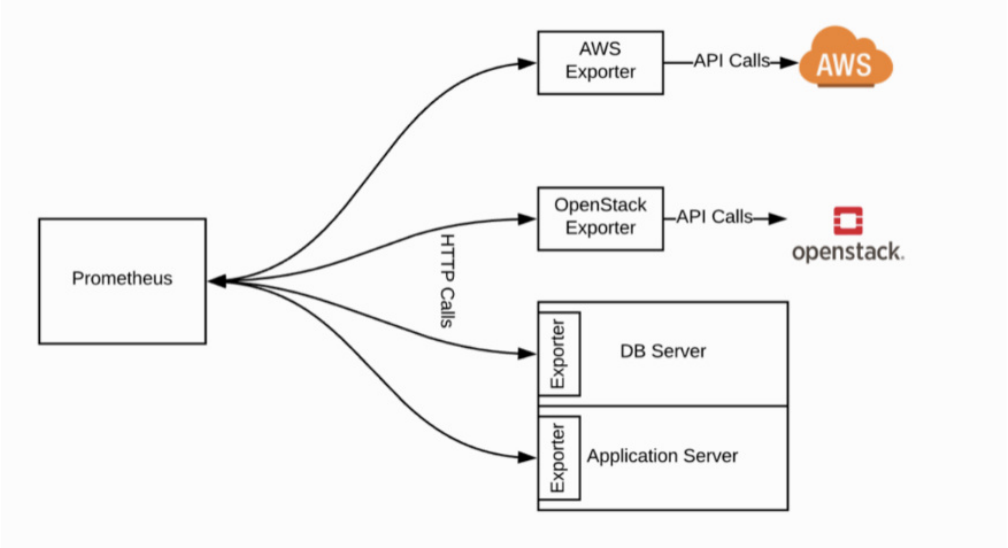
```
            body_format: json
            body:
              tags: "{{ item.6 }}"
            headers:
              Content-Type: "application/json"
          with_together:
            - "{{ src_ip }}"
            - "{{ src_port }}"
            - "{{ dest_ip }}"
            - "{{ dest_port }}"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

2.2.4.4. Rola na správu Zabbix

```
    - 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 predstavuje jeden z najlepších open-source monitorovacích nástrojov súčasnej doby. Nástroj podporuje viacero cloud backend riešení a dimenzionálny dátový model. V jadre, Prometheus zbiera metriky z HTTP endpointov, buď priamo alebo špeciálnych nástrojov, ktoré metriky z monitorovaných zariadení zbierajú ako *Node-exporter*. Všetky zozbierané dáta ukladá a porovnáva ich so zadefinovanými prahovými hodnotami pre vygenerovanie varovaní. Ak sú prahové hodnoty prekročené, na posielanie varovaní využíva nastroj *Alertmanager*. Nástroj neposkytuje vizualizáciu dát alebo dashboard, ale využíva na to ďalší nástroj Grafana. Tento systém sa nazýva aj Grafana-Alertmanager-Prometheus (GAP) a je často využívaný v cloudových riešeniach [22].



Obrázok 18 Koncepčná architektúra Prometheus [22]

Na obrázku vyššie môžeme vidieť ako dokáže byť implementovaný Prometheus na hybridný cloud. Pre každý väčší cloud existuje takzvaný exporter, ktorý je potrebné nainštalovať na hosta. Exporter predstavuje niečo ako agenta pre Prometheus, pretože sa stáva koncovým bodom HTTP protokolu, vďaka ktorému Prometheus zozbiera metriky. Vďaka tomu je Prometheus mimoriadne škálovateľný, robustný a bezstavový.

Obrázok 95 Infraštruktúra s použitím VXLAN sa dá použiť ako dobrý príklad pre návrh nasadenia monitorovania pomocou nástroja Prometheus. Sieťový administrátor bude v tomto prípade chcieť monitorovať fyzické servery (uzly) na ktorých beží niektorá z OpenStack služieb. V ideálnom prípade je vhodné nasadiť GAP na separátny fyzický server. Najjednoduchším spôsobom nasadenia nástroja je pomocou docker-compose ako docker kontajner. V rámci diplomovej práce som pripravil konfiguračné súbory, ktoré dovoľujú nasadenie monitorovacieho riešenia. Toto riešenie pozostáva zo samotného Prometheus nástroja, Grafana nástroja pre vykresľovanie grafov a vytváranie dashboardov, Alertmanager nástroja pre posielanie upozornení a Node-exporter pre zbieranie dát z monitorovaných serverov. Konfigurácia Alertmanager je pre komunikačnú platformu Slack, ale existuje podpora je aj pre Teams, E-mail alebo SMS. Nasadiť node-exporter bude potrebné okrem Prometheus servera aj na každý monitorovaný uzol. Celé riešenie je automatizované. Server na ktorom bude GAP bežať a tak isto aj monitorovaný server, musia mať nainštalovaný docker a docker-compose.

Monitorovanie virtuálnych strojov vytvorených na cloudoch prebieha o niečo komplikovanejšie. Rovnako ako v predošlom prípade by sme mali jeden fyzický monitor server obsahujúci GAP. Ak predpokladáme, že VMs budú na privátnych podsieťach, riešením by bolo na každom cloude vytvoriť monitorovaciu VM na ktorej by sa nachádzal Prometheus server. Táto monitorovacia VM by mala pridelenú floating ip (v prípade OpenStacku) a tým pádom by ponúkala verený endpoint pre Grafanu, nachádzajúcu sa na fyzickom monitor serveri. V Grafane sa každý Prometheus server nataví ako *datasoruce*. Alertmanager podporuje viacej Prometheus inštancií. Bude čakať na hlásenia o prekročení prahových hodnôt a posielať upozornenia na komunikačné kanály.

Alternatívnym nástrojom k riešeniu Prometheus môže byť aj monitorovací systém Zabbix alebo iný systém s ekvivalentnou funkcionalitou.

ZOZNAM POUŽITEJ LITERATÚRY

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

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

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

[4] **juju.is**. 2022. Building and deploying a mini charm. [online]. 2022. Dostupné na internete: 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. Dostupné na internete: https://bugs.launchpad.net/juju/+bug/1942864

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

[7] **docs.openstack.org**. 2022. Install OpenStack. [online]. 2022. Dostupné na internete: 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. Dostupné na internete: https://opendev.org/openstack/skyline-apiserver#user-content-install--run

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

[10] S. M. B. a. 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. a. 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. a. 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]. Dostupné na internete: https://docs.openstack.org/arch-design/design-networking.html. [Cit. 2021].

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

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

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

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

[21] ManageIQ, „ManageIQ,“ 2021. [Online]. Available: https://www.manageiq.org/docs/get-started/. [Cit. 2021].

[22] Prometheus Authors, „https://prometheus.io/,“ 2021. [Online]. Available: Prometheus. [Cit. 2021].