Gymnásium Jana Keplera

AUTOMATIZOVANÁ INSTALACE LINUXOVÉHO SERVERU POMOCÍ TECHNOLOGIE ANSIBLE

MATURITNÍ PRÁCE

Vypracoval: Jeffrey-Jonathan Klimeš

Vedoucí: František Šimorda

Poděkování

Rád bych poděkoval vedoucímu mé práce Františku Šimordovi za rady a trpělivost. Dále Karlu Jílkovi a všem studentům, kteří mi během dlouhých hodin v “nudli” poradili...

1 Úvod

Počet serverů se neustále zvyšuje a díky virtualizaci můžeme mít na jednom fyzickém serveru mnoho serverů virtuálních. Pro zjednodušení jejich konfigurace existuje mnoho způsobů a zde použitý Ansible je jeden z nich.
Ansible je open-source software, který umožňuje správu, řízení a konfiguraci mnoha počítačů v síti. Poslední dobou se Ansible těší velké popularitě díky své dostupnosti a srozumitelnosti. Do základních operací se může bez problémů pustit kdokoliv bez zkušeností s programováním.
Příkladem může být situace, kdy se chcete ujistit, že služba **Apache** je řádně nainstalovaná a updatovaná. Stačí vytvořit takzvaný *playbook*, což je YAML (.yml) soubor s velice jednoduchým ale důležitým formátováním, do kterého kromě vyžadovaných základních tří řádků kódu napíšeme `yum: name=httpd state=latest`. Jak můžete vidět, stačí jen jednoduše určit název služby *httpd* a požadovaný stav *installed* a Ansible se už o vše postará.
Vývojářem je Red Hat Inc. který stojí také za linuxovou distribucí **CentOS 7**, která je stejně jako Ansible volně dostupná. Centos jsem si já zvolil jako systém běžící na serveru, který budu spravovat.
Zadáním práce je instalace vytvoření nástroje pro automatizovanou instalaci následujících programů:

- **http**
- **nginx**
- **postgresql**
- **mysql**
- **p7zip**
- **wget**
- **wordpress**
- **lm_sensors**
- **htop**
- **ethtool**
- **elinks**
- **Certbot**
- **Wordpress**

A dále program na administraci aplikací v dockerových kontejnerech, což jsem řešil pomocí programu **Portainer**.

Na správný postup a testování jsem potřeboval funkční *controlling machine* - počítač, ze kterého se servery ovládají, a server, nazývaný také *host* nebo *node.*

2 Zprovoznění Controlling Machine

Ansible bohužel nepodporuje Windows jako možnou controlling machine, jen jako možného hosta. Já jsem však jinou možnost než ovládat můj server z Windows neměl, takže jsem použil neoficiální řešení pomocí ***Bash on Ubuntu on Windows***. Komponenty jsou už na Windows předinstalovány, musí se však povolit. To obnáší zapnutí ***developer mode***, povolení služby ***Linux envirovent for Windows*** a dále v příkazové řádce cmd musíme nainstalovat **lxrun**.

```
#  lxrun /install
```

Poté lze již ***Bash on Ubuntu on Windows*** spustit. Je ještě potřeba nainstalovat Ansible:

```
#  sudo apt-get install software-properties-common
#  sudo apt-add-repository ppa:ansible/ansible
#  sudo apt-get update
#  sudo apt-get install ansible
```

Nyní již lze Ansible používat.

2.1 Zprovoznění Serveru

Server k testování jsem zprovoznil v aplikaci VirtualBox od vývojáře Oracle. Do Virtualboxu jsem nainstaloval **Centos 7** s environmentem ***Server with GUI***.
Virtualbox poskytuje mnoho možných síťových nastavení, kterými se dá určit, jak se bude virtuální počítač projevovat v síti, což se ukázalo jako značná překážka. Ze Začátku jsem používal oblíbené a doporučené nastavení ***bridged network***, které však v určitých specifických případech způsobovalo problémy. Nakonec jsem skončil s dvěma zapnutými síťovými adaptéry, a to ***Nat*** a ***Host-only Adapter***.

2.2 Komunikace mezi počítači

Ansible provádí operace pomocí **SSH**, takže je k bezproblémovému přístupu k serveru vhodné zkopírovat ssh klíč:

```
#  ssh-keygen -t rsa
#  ssh-copy-id
```

Dále je nutné zadat Ansible, na kterých počítačích má operace provádět. To lze dvěma způsoby, první a spolehlivější je konfigurace ***hosts*** souboru, umístěného v ***/etc/ansible***. Druhý

způsob jsou inventáře, což je jednoduchý soubor s formátováním shodným s ***hosts*** *souborem,* nevolá se však automaticky ale pomocí přepínače "`-i název_inventáře`".
Zápis našich serverů do ***hosts*** nebo **inventářů** je jednoduchý. Můžeme zapisovat jak jednoduché adresy serverů, tak skupiny, pokud chceme servery filtrovat. Skupina serverů se píše sama na řádek jako [skupina] a servery jednoduše na řádky pod tím jako server1.cz. V mém případě jsem pro bezproblémovou instalaci připsal na řádek za serverem:
`ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=heslo_na_roota`
Příklad souboru hosts:

```
[Testing]
192.168.56.101 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=root
```

3 Playbooks

Nyní již můžeme na serveru spouštět playbooky, buď pomocí ***hosts*** souboru:

```
ansible-playbook testplaybook.yml
```

nebo pomocí inventáře, který jsme si pojmenovali ***inventory****:*

```
ansible-playbook -i inventory testplaybook.yml
```

Playbooky mají jednoduchý formát, ale je velice důležité jej dodržovat. Každý .yml playbook začíná třemi pomlčkami ---, pod nimi většinou stojí hosts:, kde můžeme určit kde se playbook spustí, ale vždy jsem tu psal hosts: all, protože je jednodušší filtrovat servery pomocí [skupin].
Pod tímto již stojí jen tasks: a následují různé úkoly. Ty se ještě pomocí name: dají shlukovat do určitých skupin. Příkladem je tento playbook, který jsem používal k testování připojení (jestli Ansible opravdu má přístup k serveru):

```
---
- name: testing ansible
  hosts: all


  tasks:
        - name: echo a message
        debug: msg="this is working"
```

3.1 Playbooks yum instalace

Základním úkolem mého playbooku byla instalace programů. Centos používá ***yum***.
Instalace pomocí playbooku vypadá takto:

```
---
- name: Tohle nainstaluje apache
  hosts: all
  tasks:
  - name: Installace apache
        yum:
        name: httpd
        state: installed
```

Za `state:` se dají použít i jiné přepínače, např: absent, latest, present, removed, none...

3.1 Playbooks service

Dalším častým úkolem pro playbooky byla správa služeb, například aby výše nainstalovaný Apache fungoval, je ještě potřeba ho spustit a povolit. K tomu se využívá `service:`
Playbook, který Apache nastaví, vypadá takto:

```
---
- name: Tohle nastaví apache
  hosts: all
  tasks:
  - name: ujistime se ze httpd bezi
        service:
        name: httpd
        state: started
  - name: otevreni portu 80 pro http
        firewalld:
        service: http
        permanent: true
        state: enabled
  - name: restartovani firewalld kvuli nacteni novych nastaveni
        service:
        name: firewalld
        state: restarted
```

4 main.yml Hlavni playbook

Instalace většiny programů je soustředěna do jednoho playbooku ***main.yml*** v domovském adresáři projektu. Playbook má za sebou nespočet verzí, ale celkově se nejedná o nijak složitou konstrukci. Za zmínku stojí způsob instalace vícero balíčků zároveň, použitím jednoduchého:

```
yum:
        name:
        - httpd
        - htop
        - nginx
```

Namísto zastaralého, avšak hojně používaného:

```
yum: pkg={{ item }} state=present
  with_items:
    - httpd
    - htop
- nginx
```

Celý kód pak vypadá takto:

```
---
- name: naistaluje vsechny potrebne programy na node a take je nastavi
  hosts: all
  tasks:

  - name: instalace dalsich repozitaru pro htop
        yum:
        name: epel-release
        state: installed
        changed_when: false
#  - name: instalace dalsich repozitaru pro mysql
#        get_url:
#        url: http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
#        dest: /opt
#  - name: instalace dalsich repozitaru pro mysql
#        yum:
#        name: /opt/mysql-community-release-el7-5.noarch.rpm
#        state: present
  - name: update yumu
        yum:
        state: latest

  - name: instalace potrebnych programu
        yum:
        name:
        - httpd
```

```
        - htop
        - nginx
        - postgresql-server
        - postgresql-contrib
        - mariadb-server
        - mariadb
        #       - mysql-server
        #       - mysql-client
        #       - python-mysqldb
        - p7zip
        - wget
        - lm_sensors
        - ethtool
        - certbot
        state: installed
        changed_when: false
  - name: APACHE ujistime se ze bezi
        service:
        name: httpd
        state: started
  - name: APACHE otevreni portu pro http
        firewalld:
        service: http
        permanent: true
        state: enabled
  - name: APACHE restartovani firewalld kvuli nacteni novych nastaveni
        service:
        name: firewalld
        state: restarted
# Pouzit misto Apache pokud chceme, avsak neda se spustit soubezne.
#  - name: NGINX start
#       service:
#       name: nginx
#       state: started

  # Da se spustit pouze jednou, pri opetovnem spusteni playbooku zakomentovat
  - name: POSTGRESQL pomuze to nahodit
        command: postgresql-setup initdb

  - name: POSTGRESQL startuje
        service:
        name: postgresql
        state: started
```

5 Wordpress

Nejsložitější operací bylo zprovoznění Wordpressu. Přibližně jsem se snažil postupovat podle tohoto tutorialu:
https://www.tecmint.com/automate-wordpress-installations-in-multiple-linux-servers-using-ansible/

Ukázalo se však, že celá operace není ani zdaleka tak jednoduchá. Celkově jsem rozhodně strávil více času sprovozňováním wordpressu než čímkoliv jiným, počítaje i instalaci virtualboxu atp.
První proces bylo rozchodit speciální strukturu složek, která se s Ansible využívá při složitějších operacích. Tato struktura složek se dá vygenerovat pomocí Ansiblové funkce ***ansible-galaxy***.
V mém případě jsem generoval dvě složky, ***wp-dependecies*** a ***wp-install-config***. Příkaz vypadá takto:

```
#  ansible-galaxy init wp-dependencies
#  ansible-galaxy init wp-install-config
```

Ansible Galaxy takto vytvořil v aktuálním adresáři tyto dvě složky. V každé složce ještě složky defaults, files, handlers, meta, tasks, templates, vars. V každé v této složce, kromě files a templates, vytvořil YAML soubor ***main.yml***.
Zezačátku to vypadalo, že můžu příkazem ansible-galaxy init vytvořit složky kdekoliv, ale není to tak, vygenerované složky musí být v určitých umístěních, jako třeba /etc/ansible/playbooks/ a nebo se musí vložit do složky ***roles***.
Hierarchie poté vypadá přibližně tak, že máme hlavní YAML soubor, v mém případě *wordpress.yml* a ve stejném adresáři složku *roles,* v *roles* poté *wp-dependecies* a *wp-install-config.*

5.1 Playbooks

Podle výše uvedeného tutorialu jsem postupoval dále, ale hlavně nejdůležitější playbook ***/roles/wp-dependencies/tasks/main.yml*** si vyžádal ohromné množství práce. Celkově jsem vystřídal 13 různých chybových hlášek, dvě z nich pomohla rozklíčovat až komunita na stackoverflow.com. Ve finále vypadá takto:

```
---
# tasks file for wp-dependencies
- name: Update packages (this is equivalent to yum update -y)
  yum: name=* state=latest

- name: Install dependencies for WordPress
  yum:
        name:
        - php
```

```
        - php-mysql
        - MySQL-python
        state: present

- name: Ensure MariaDB is running (and enable it at boot)
  service: name=mariadb state=started enabled=yes

- name: Copy ~/.my.cnf to nodes
  copy: src=.my.cnf dest=/root/.my.cnf
- name: nastaveni hesla pro root usera
  mysql_user:
        name=root
        password=root

- name: Create MariaDB database
  mysql_db: name={{ wp_mysql_db }} state=present login_user=root login_password=root

- name: Create MariaDB username and password
  mysql_user: login_user=root login_password=root name={{ wp_mysql_user }} password={{
wp_mysql_password }}
        priv="*.*:ALL"
```

Tento playbook je doplněný o ***/roles/wp-dependencies/defaults/main.yml***, který vypadá takto:

```
---
# defaults file for wp-dependencies
  wp_mysql_db: MyWP
  wp_mysql_user: wpUser
  wp_mysql_password: wpP4ss
```

Pak zde ještě je ***/roles/wp-install-config/tasks/main.yml***:

```
---
# tasks file for wp-install-config
- name: Create directory to download WordPress
  command: mkdir -p /opt/source/wordpress
  args:
        warn: false
- name: Download WordPress
  get_url: url=https://www.wordpress.org/latest.tar.gz
dest=/opt/source/wordpress/wordpress.tar.gz validate_certs=no
```

```
- name: Extract WordPress
  command: "tar xzf /opt/source/wordpress/wordpress.tar.gz -C /var/www/html
--strip-components 1"
  args:
        warn: false
- name: Send config file
  copy: src=wp-config.php dest=/var/www/html/wp-config.php mode=0644
- name: kopirovani localhost.conf
  copy: src=localhost.conf dest=/etc/httpd/conf.d mode=0644
- name: mazani wp-config.php
  file:
        path: /usr/share/wordpress/wp-config.php
        state: absent
- name: kopirovani wp-config.php
  copy: src=wp-config.php dest=/usr/share/wordpress
- name: mazani welcome.conf
  file:
        path: /etc/httpd/conf.d/welcome.conf
        state: absent
- name: restartovani httpd
  service:
        name: httpd
        state: restarted
```

5.2 wp-config.php

Dalším souborem který najdeme v domovském adresáři mé práce je ***wp-config.php***, který obsahuje mnoho řádků kódu, kterého jsem se nikdy nedotkl, ale obsahuje i následující pasáž:

```
/** The name of the database for WordPress */
define('DB_NAME', 'MyWP');

/** MySQL database username */
define('DB_USER', 'wpUser');

/** MySQL database password */
define('DB_PASSWORD', 'wpP4ss');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```

Zde jsem několikrát proměnné změnil, ale nakonec jsem skončil takto. ***wp-config.php*** se již výše ukázaným playbookem kopíruje do **/usr/share/wordpress** a nahrazuje místní *wp-config.php,* který playbook napřed smaže. Poté se kopíruje i do **/var/www/html**.

5.3 .my.cnf

Dalším souborem v domovském adresáři mé práce je ***.my.cnf****,* který se kopíruje do **/root** serveru.
Obsahuje tento kód:

```
[client]
user=root
password=root
```

5.3 localhost.conf

Dalším souborem je **localhost.con**f, který se kopíruje do **/etc/htpd/conf.d** a nastaví virtualhosta.
Vypadá takto:

```
<VirtualHost *:80>
        ServerAdmin root@localhost
        ServerName localhost
        #  ServerAlias www.localhost.com
        DocumentRoot /var/www/html/

        <Directory /var/www/html/>
                Options -Indexes
        <IfModule mod_rewrite.c>
                RewriteEngine On
                RewriteBase /
                RewriteCond %{REQUEST_FILENAME} !-f
                RewriteCond %{REQUEST_FILENAME} !-d
                RewriteRule . /index.php [L]
                </IfModule>
        </Directory>
</VirtualHost>
```

6 Portainer

Poslední úlohou byla instalace programu na správů programů v dockerových kontejnerech. Na konzultaci s vedoucím jsme se dohodli na programu ***Potrainer***. Jeho instalační playbook vypadá takto:

```
---
- name: instalace portaineru
  hosts: all
  tasks:
  - name: instalace dockeru
        yum:
        name: docker
        state: installed
  - name: startovaní dockeru
        service:
        name: docker
        state: started
  - name: povolení dockeru
        command: systemctl enable docker
  - name: pull
        command: docker pull portainer/portainer
  - name: spusteni portaineru
        command: docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer
```

6 full.yml - kompletní instalace

Pro kompletní instalaci celého zadání slouží soubor **full.yml**, který jen postupně stouští **main.yml**, **wordpress.yml** a **portainer.yml**, v tomto pořadí.
Instalace by se sice dala jednoduše složit do jednoho velikého souboru, ale kvůli přehlednosti a rychlejšímu troubleshootingu jsem zvolil oddělenou instalaci. Sluší se ještě podotknout, že jednotlivé **main.yml** v složce ***roles***, sloužící k instalaci wordpressu, se jednotlivě spouštět nedají.

full.yml:

```
---
- import_playbook: main.yml
- import_playbook: wordpress.yml
- import_playbook: portainer.yml
```

7 zdroje

cs.wikipedia.org
www.root.cz
cloudacademy.com
www.tecmint.com
www.askubuntu.com
www.stackoverflow.com
www.docs.ansible.com
www.linode.com
www.gist.github.com
www.github.com
www.howtoforge.com
www.linux4one.com
www.code-maven.com
www.codingbee.net
www.techotopia.com
www.unix.stackexchange.com
www.upcloud.com
www.scaleway.com
www.linuxhint.com
www.ssh.com
www.superuser.com
www.mariadb.com
www.sesan.gob.gt
www.linuxhelp.com
www.serverfault.com