Inhaltsverzeichnis

Automatische Installationen mit Ansible

setup venv

```
python -m venv .venv
source .venv/bin/activate
```

install Ansible

```
pip install ansible
```

wir basteln und eine `inventory`- Datei

inventory.yaml

```
all:
  hosts:
    docker_test: 192.168.56.103
```

Dann brauchen wir unseren public SSH key:

erstmal checken, das wir nicht schon einen haben

```
ls -l ~/.ssh
```

dann erzeugen

```
ssh-keygen -b 4096
```

und an den anderen Rechner übertragen- auch hier wieder Vorsicht, dass man dort nicht bestehende Keys überschreibt:

```
scp /home/steffen/.ssh/id_rsa.pub
dockerapps@192.168.56.103:/home/dockerapps/.ssh/authorized_keys
```

speichern des github- Tokens:

```
export EDITOR=nano
ansible-vault create secrets.yml
```

```
gituser: stko
gitpass: <yourgittoken>
```

Git credentials als Parameter einbauen

Wenn das Playbook Daten aus privaten Repositories ziehen soll, wird's tricky [1]:

- Zuerst einmal muss man sich im privaten Repository ein Token mit `Pull Repository` Rechten anlegen (<TOKEN>)
- dann baut man in den playbook- git Befehl in den Repository- URL die Parameter ein:

```
   repo: "https://git:{{ githubpassword | urlencode | replace ('/', '%2f') }}@git.s26314....
```

und übergibt den Parameter dann beim Start von ansible-playbook

```
  ansible-playbook -i hosts github.yml -e "githubpassword=<TOKEN>"
```

Ein Beispiel- Playbook, verallgemeinert aus einer Arbeit einer Kollegin:

```
---
- name: Deploy YOURPROGRAM on Raspberry Pi
  hosts: yourprogram_devices
  become: yes

  vars:
    user: pi
    home_dir: "/home/pi"
    module_dir: "{{ home_dir }}/installdir"
    yourprogram_dir: "{{ module_dir }}/yourprogram"
    docker_compose_version: "v2.23.3"
    ansible_python_interpreter: /usr/bin/python3


  tasks:

    - name: Update & upgrade packages
      apt:
        update_cache: yes
        upgrade: dist
        cache_valid_time: 3600

    - name: Install required packages
      apt:
        name:
          - git
          - python3-venv
          - python3-pip
          - python3-virtualenv

        state: present

    # in case of re-deployment, remove existing dir
    - name: Remove existing dir if exists
      ansible.builtin.file:
        state: absent
```

```
        path: "{{ module_dir }}"


    - name: ensures {{ module_dir }} dir exists
      file:
        path: "{{ module_dir }}"
        state: directory


    - name: Clone YOURPROGRAM repositories
      git:
        repo: "{{ item.repo }}"
        dest: "{{ module_dir }}/{{ item.name }}"
      loop:
        - { name: labdash, repo: "https://git:{{ githubpassword | urlencode
| replace ('/', '%2f') }}@github.com/stko/labdash.git" }
        - { name: labdash_internal, repo: "https://git:{{ githubpassword |
urlencode | replace ('/', '%2f')
}}@git.s26314.creoline.cloud/toplevel/inhouse/labdash_internal.git" }
        - { name: yourprogram, repo: "https://git:{{ githubpassword |
urlencode | replace ('/', '%2f')
}}@git.s26314.creoline.cloud/toplevel/inhouse/yourprogram.git" }

    # - name: Create Python virtual environment
    #   ansible.builtin.command: python3 -m venv .venv
    #   args:
    #     chdir: "{{ yourprogram_dir }}"
    #     creates: "{{ yourprogram_dir }}/.venv"

    # - name: update pip in virtual environment
    #   ansible.builtin.command: python3 -m pip install --upgrade pip
    #   args:
    #     chdir: "{{ yourprogram_dir }}"
    #     creates: "{{ yourprogram_dir }}/.venv/bin/pip"

    # - name: install setuptools in virtual environment
    #   ansible.builtin.command: python3 -m pip install --upgrade setuptools
packaging
    #   args:
    #     chdir: "{{ yourprogram_dir }}"
    #     #creates: "{{ yourprogram_dir }}/.venv/bin/pip"


   # - name: Ensure packaging is installed system-wide
   #   ansible.builtin.raw: '{{ ansible_python_interpreter }} -m pip install
--quiet packaging'

    - name: Install Python dependencies
      ansible.builtin.pip:
        requirements: "{{ yourprogram_dir }}/requirements.txt"
        virtualenv: "{{ yourprogram_dir }}/.venv"
```

```
        virtualenv_python: python3
        virtualenv_site_packages: yes
      register: pip_result

    # - name: Install editable dependencies
    #   ansible.builtin.pip:
    #     name: "{{ item }}"
    #     virtualenv: "{{ yourprogram_dir }}/.venv"
    #     virtualenv_python: python3
    #     virtualenv_site_packages: yes
    #     editable: yes
    #   loop:
    #     - "{{ module_dir }}/cnclib"
    #     - "{{ module_dir }}/tcpc"
    #     - "{{ module_dir }}/MTDS_modify"
    #     - "{{ module_dir }}/mtqr/python"
    #     - "{{ module_dir }}/labdash"


    - name: Add a line to a file if the file does not exist, without passing
regexp
      ansible.builtin.lineinfile:
        path: /etc/sysctl.d/98-rpi.conf
        line: net.ipv4.ip_forward = 1
        create: yes

    # - name: Install Docker
    #   ansible.builtin.shell: curl -fsSL https://get.docker.com | sh
    #   args:
    #     creates: /usr/bin/docker

    # - name: Add user to Docker group
    #   user:
    #     name: "{{ user }}"
    #     groups: docker
    #     append: yes

    # - name: ensures docker plugin dir exists
    #   file:
    #     path: "{{ home_dir }}/.docker/cli-plugins/"
    #     state: directory
    #     mode: '0755'


    # - name: Install Docker Compose v2
    #   get_url:
    #     url: "https://github.com/docker/compose/releases/download/{{
docker_compose_version }}/docker-compose-linux-aarch64"
    #     dest: "{{ home_dir }}/.docker/cli-plugins/docker-compose"
    #     mode: '0755'
```

```
  # Docker installation and setup
https://github.com/docker/for-linux/issues/1105#issuecomment-913964377

    # - name: prepare network for docker installation
    #   ansible.builtin.shell: sudo update-alternatives --set iptables
/usr/sbin/iptables-legacy

##################
https://github.com/linuxserver/docker-wireguard/issues/138#issuecomment-1003
173623 ##############


    # - name: Make sure docker is running
    #   ansible.builtin.systemd_service:
    #     state: started
    #     name: docker

    # - name: Deploy Docker services
    #   ansible.builtin.shell: docker compose up --build -d
    #   args:
    #     chdir: "{{ yourprogram_dir }}/test/rabbitmq"



    - name: Configure YOURPROGRAM systemd service
      copy:
        dest: /etc/systemd/system/yourprogram.service
        content: |
          [Unit]
          Description=YOURPROGRAM Description
          After=network.target

          [Service]
          ExecStart={{ yourprogram_dir }}/.venv/bin/python yourprogram.py
          WorkingDirectory={{ yourprogram_dir }}
          Restart=always
          User={{ user }}

          [Install]
          WantedBy=multi-user.target

    - name: Enable YOURPROGRAM service
      systemd:
        name: yourprogram.service
        enabled: yes
        state: started
```

Python Module installieren mit Pip

Scheinbar gibt es (mindestens beim Raspi) Probleme, dass der Pip- Routine schlichtweg derArbeitsspeicher ausgeht.. ?!? Ein Mittel dagegen ist, wenigstens den Speicherhunger von Pip etwas mit `extra_args: –no-cache-dir` einzudämmen:

```
  - name: Install Python dependencies
      ansible.builtin.pip:
        requirements: "{{ zuulac_dir }}/requirements_raw.txt"
        virtualenv: "{{ zuulac_dir }}/.venv"
        virtualenv_python: python3
        virtualenv_site_packages: yes
```

1)

https://stackoverflow.com/a/37851105