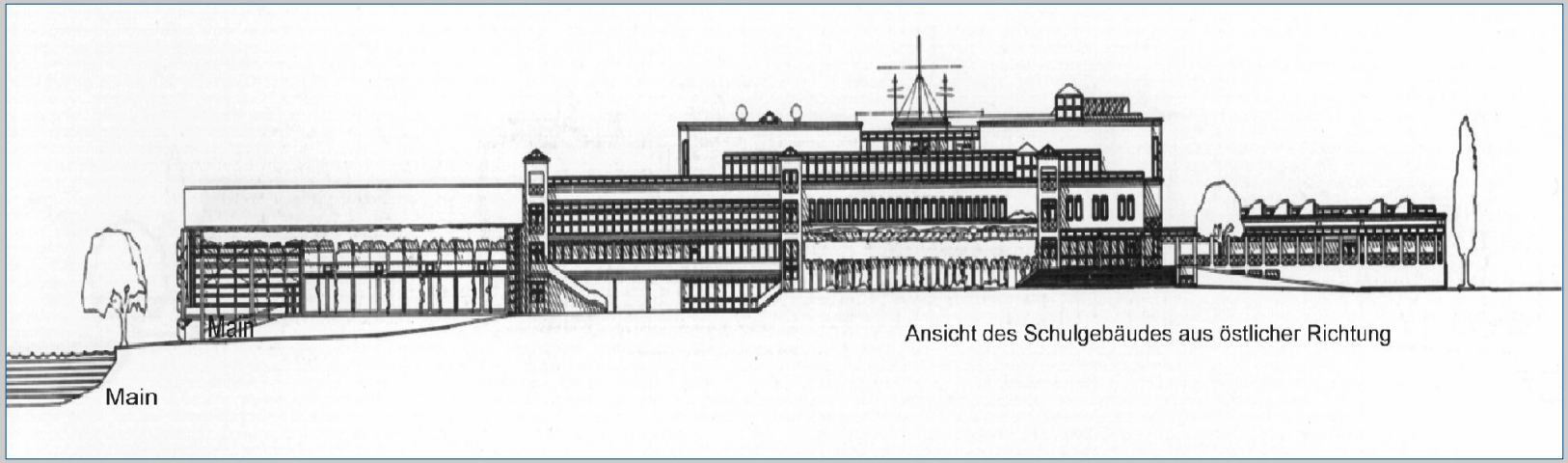
Werner-von-Siemens-Schule Frankfurt am Main

Herzlich Willkommen zum Quantensprung
Orchestrierung von VMs mit Ansible



Richard Wegers

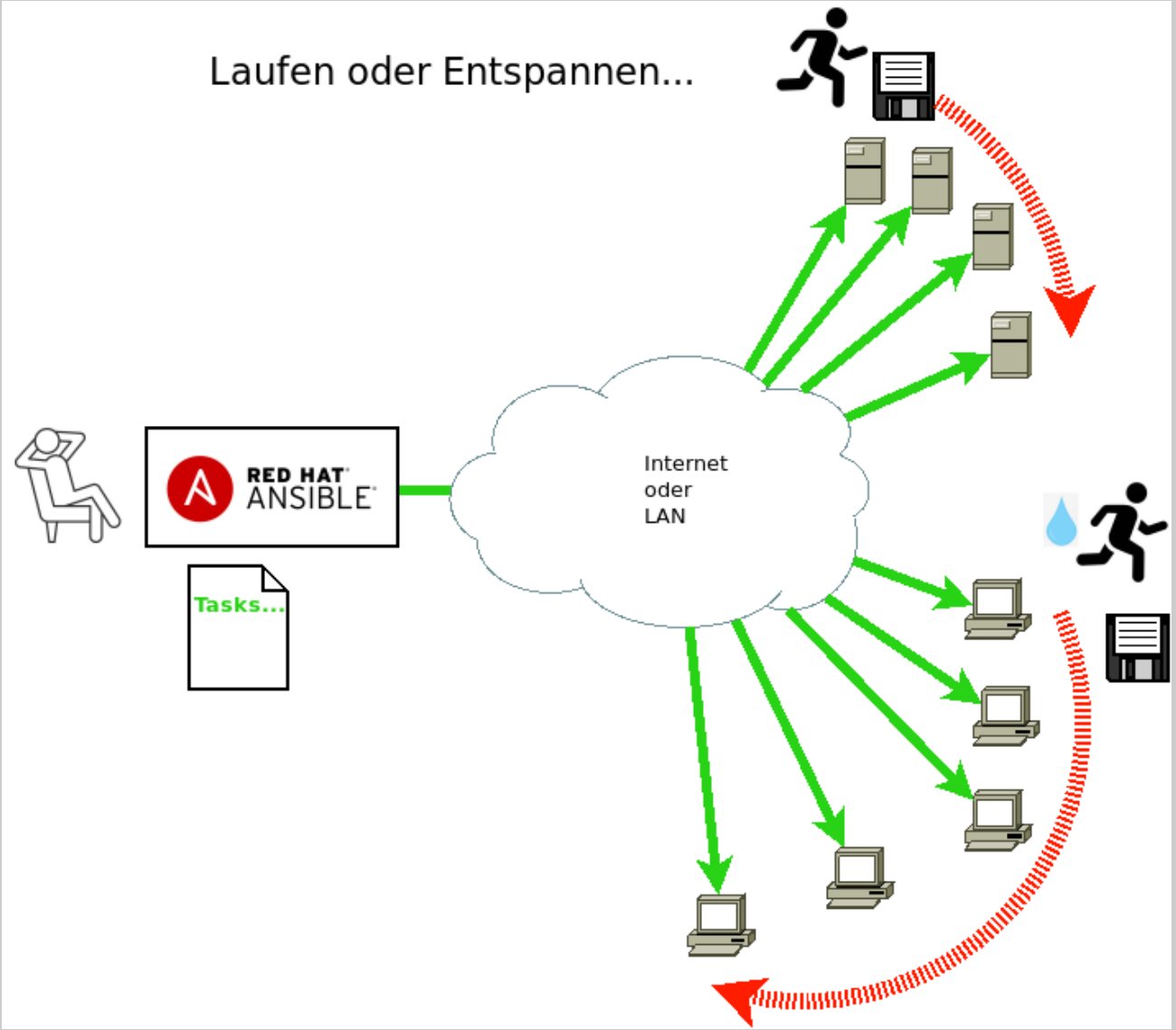
Abteilungsleiter an der Werner-von-Siemens-Schule

Frankfurt am Main

Überblick

- Aufgabenstellung
- Orchestrierung
- Ansible Playbooks
- Ansible Roles
- Hetzner Cloud mit Ansible

Aufgabenstellung



Aufgabenstellung

- viele Server/VMs/PC sind zu administrieren
- Ergebnisse bei manueller Administration
  ggf. fehlerhaft, da Schritte vergessen werden
  → geringe Qualität der Konfiguration
  → keine/aufwendige Dokumentation der tatsächlich ausgeführten Schritte
  → zeitintensiv, da jeder Schritt manuell angestoßen werden muss
- **Ziel:**
  Automatisierung durch Orchestrierung

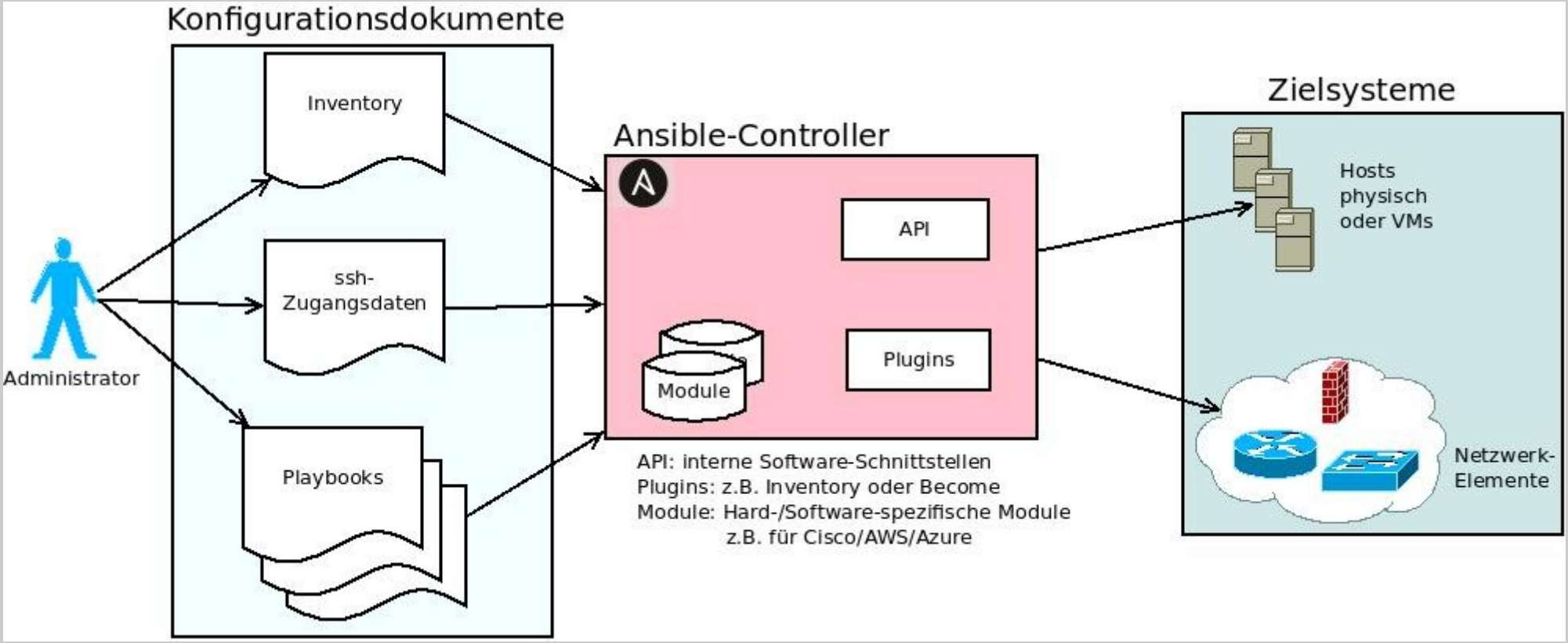
Orchestrierung

- Einsatz von Orchestrierungsframeworks
- Beispiele: puppet, chef, teraform, ansible
- Wesentlicher Unterschied der Systeme

  - Agent basiert
    → nur Funktionen möglich, die Agent unterstützt

  - ohne Agent (nur sichere ssh-Verbindung)
    → beliebige Funktionen nutzbar (theoretisch)

Orchestrierung mit Ansible



Ansible Playbooks

- **Inventory**: Enthält alle Informationen/Parameter zu den Zielsystemen
  Beispiel: IP, Hostname, etc.
- **Playbook**:
  Enthält Parameter für den Verbindungsaufbau, sowie die eigentlichen Aufgaben (tasks)
- **Tasks**:
  abzuarbeitende Administrationsschritte

Aufbau: Inventory

inventory.yml

```
all:
  hosts:
    webserver01:
      servername: "webserver01"
      ansible_host: 168.119.162.125
      ansible_user: root
      ansible_become_password: "{{ web01_password }}"
      domain: "test01.local"
  ...

  children:
    lbalancer:
      hosts:
        loadbalancer00:
    wserver:
      hosts:
        webserver01:
        webserver02:
```

Variablen, die pro Maschine gesetzt werden.

Gruppe von Maschinen, die identisch verwaltet werden sollen.

Aufbau: Playbook

playbook.yml

```
---
- name: Webserver Anwendung installieren
  hosts: wserver
  become: true
  become_user: "{{ ansible_user }}"
  vars:
    www_user: "root"
  tasks:
     - name: Installation von nginx
       # Repository aktualisieren
         - name: Update apt packages
           apt:
             update_cache: yes
             cache_valid_time: 3600

         # Webserver installieren
         - name: "install nginx"
           apt:
             name: ['nginx']
             state: latest
```

Variablen aus inventory

Variablen, innerhalb von Playbook

playbook.yml

```
(...FORTSETZUNG)

# vhosts einrichten
- name: "create www directory"
  file:
    path: /var/www/{{ domain }}
    state: directory
    mode: '0775'
    owner: "{{ www_user }}"
    group: "{{ www_user }}"
  notify: restart nginx

handlers:
# handlers für nginx-Installation
- name: restart nginx
  ansible.builtin.service:
    name: nginx
    state: restarted
```

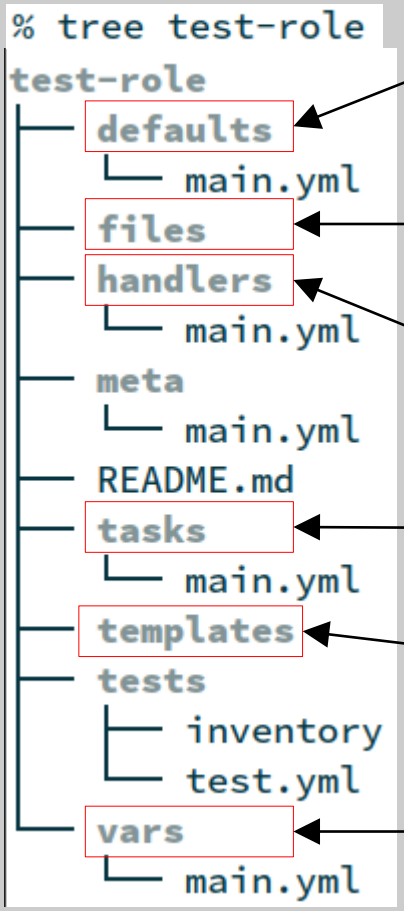
Variablen, aus inventory

Neustart des Servers per Handler

Aufruf:

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

Ansible Roles: test-role



defaults: Standardwerte für Variabeln

files: wie `templates`, aber statisch

handlers: z.B. für restart eines Dienstes nach Konfigänderung

tasks: eigentliche Aufgaben

templates: Vorlagendateien „*.j2“ z.B. für Konfigfiles; können Variabeln enthalten (dynamisch)

vars: Variabeln für gesamte Rolle

Die Ordnerstruktur (inkl. „main.yml“) kann mit folgendem angelegt werden.

Aufruf:

```
% ansible-galaxy init test-role
```

Ansible Rollen aufrufen

init_project.yml

```
---
 - name: Projekt Loadbalancer Start -
Webserver
   hosts: wserver
   remote_user: root
   gather_facts: no
   roles:
     - wserver
     - nginx
     - docker
     - nextcloud

 - name: Projekt Loadbalancer Start -
Loadbalancer
   hosts: lbalancer
   remote_user: root
   gather_facts: no
   roles:
     - loadbalancer
```

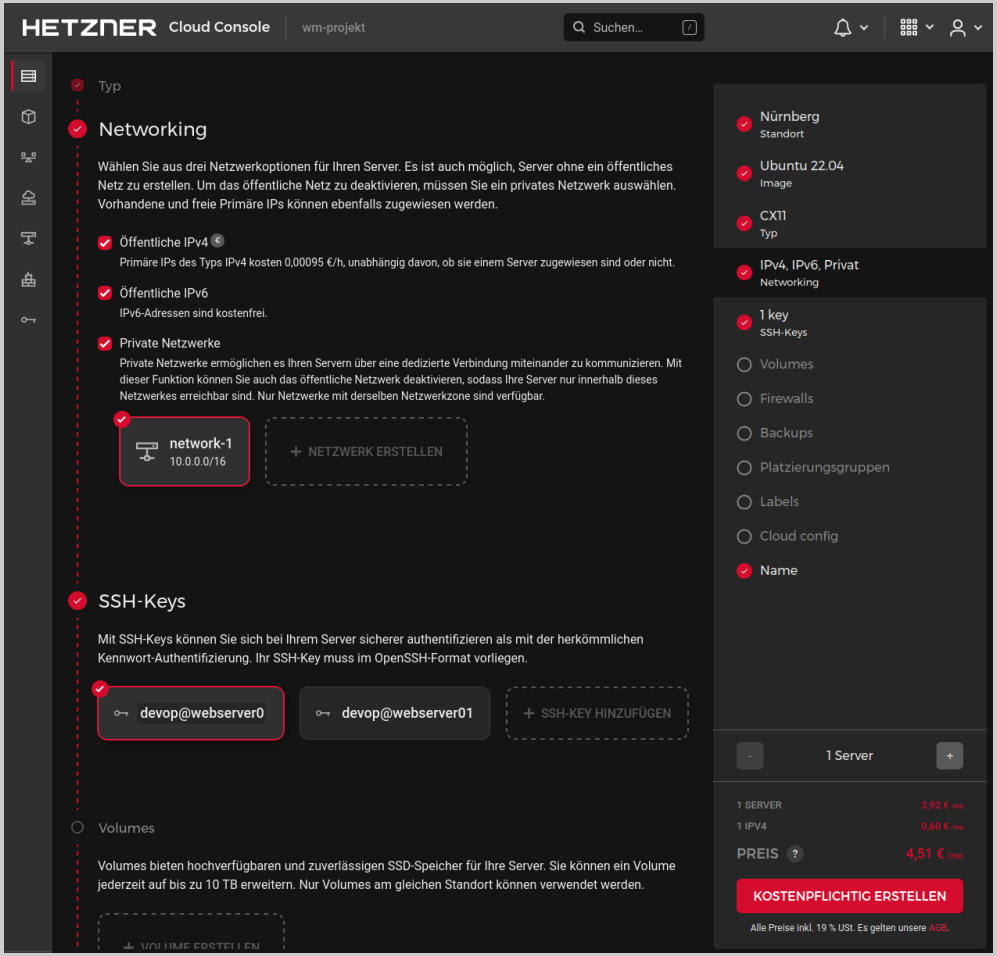
Rollen, die für die Webserver ausgeführt werden sollen.

Aufruf:

```
% ansible-playbook -i inventory.yml init_project.yml
```

Hetzner Cloud

- Live-Demo im Webfront



Hetzner Cloud mit Ansible

inventory.yml

```
all:
  hosts:
    webserver01:
      servername: "webserver01"
      ansible_host: 168.119.xx.xx
      ansible_user: root
      ansible_become_password: "{{ web01_password }}"
      domain: "test01.local"

    webserver02:
      servername: "webserver02"
      ansible_host: 167.235.xx.xx
      ansible_user: root
      ansible_become_password: "{{ web02_password }}"
      domain: "test02.local"

  children:
    wserver:
      hosts:
        webserver01:
        webserver02:
```

playbook.yml

```
---
- name: Einrichten der Server
  hosts: wserver
  connection: local
  gather_facts: no
  user: root
  tasks:
    - name: Erstellen eines Ubuntu-Server
      hcloud_server:
          api_token: "{{ hcloud_token }}"
          name: "{{ servername }}"
          server_type: cx11
          location: nbg1
          image: ubuntu-22.04
          enable_ipv4: yes
          enable_ipv6: yes
          ssh_keys:
            - devop@webserver
          state: present
      register: server

    - name: IP-Adresse ausgeben
      debug:
        msg: "{{ server.hcloud_server.ipv4_address }}"
```

s. Hetzer-Weboberfläche alle Werte könnten per Variable gesetzt werden könnten z.B. als **`default`** definiert werden.

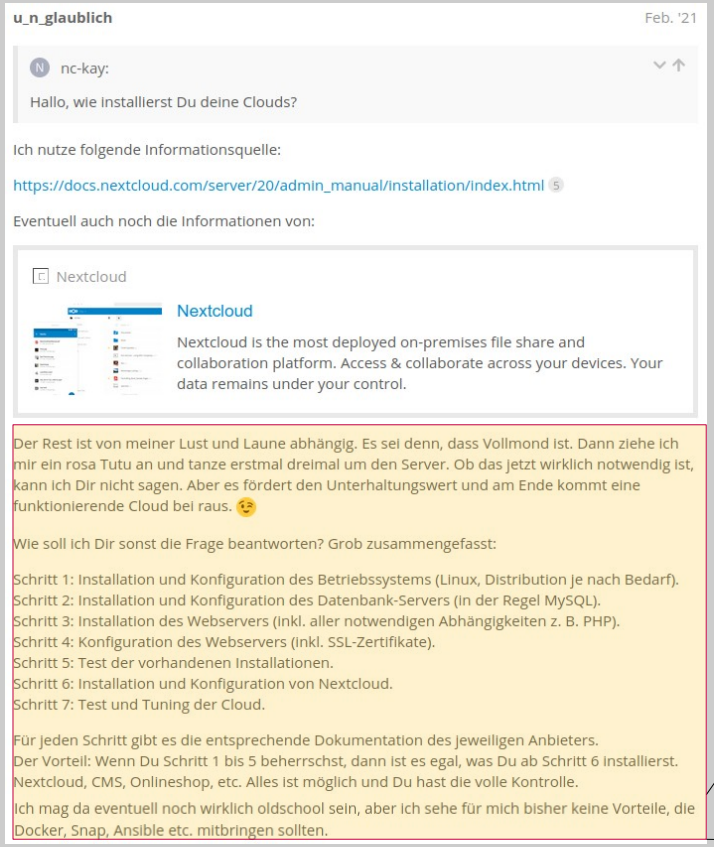
Weiterführende Links

- Ansible Dokumentation (hier Hetzner)
  Ausführliche Dokumentation der verfügbaren Ansible-Module/Plugins
  https://docs.ansible.com/ansible/latest/collections/hetzner/hcloud/index.html

- Ansible Rollen Repository
  Einfache Installation existierender Rollen:
  `ansible-galaxy install nginxinc.nginx_controller_install`
  https://galaxy.ansible.com/

- Hetzer-Cloud Dokumentation
  Die Hetzner-Cloud kann auch direkt über REST-API angesprochen werden.
  https://docs.hetzner.cloud/

Schlusswort



u_n_glaublich — Feb. '21

nc-kay:

Hallo, wie installierst Du deine Clouds?

Ich nutze folgende Informationsquelle:

https://docs.nextcloud.com/server/20/admin_manual/installation/index.html

Eventuell auch noch die Informationen von:

Nextcloud

Nextcloud

Nextcloud is the most deployed on-premises file share and collaboration platform. Access & collaborate across your devices. Your data remains under your control.

Der Rest ist von meiner Lust und Laune abhängig. Es sei denn, dass Vollmond ist. Dann ziehe ich mir ein rosa Tutu an und tanze erstmal dreimal um den Server. Ob das jetzt wirklich notwendig ist, kann ich Dir nicht sagen. Aber es fördert den Unterhaltungswert und am Ende kommt eine funktionierende Cloud bei raus. 😉

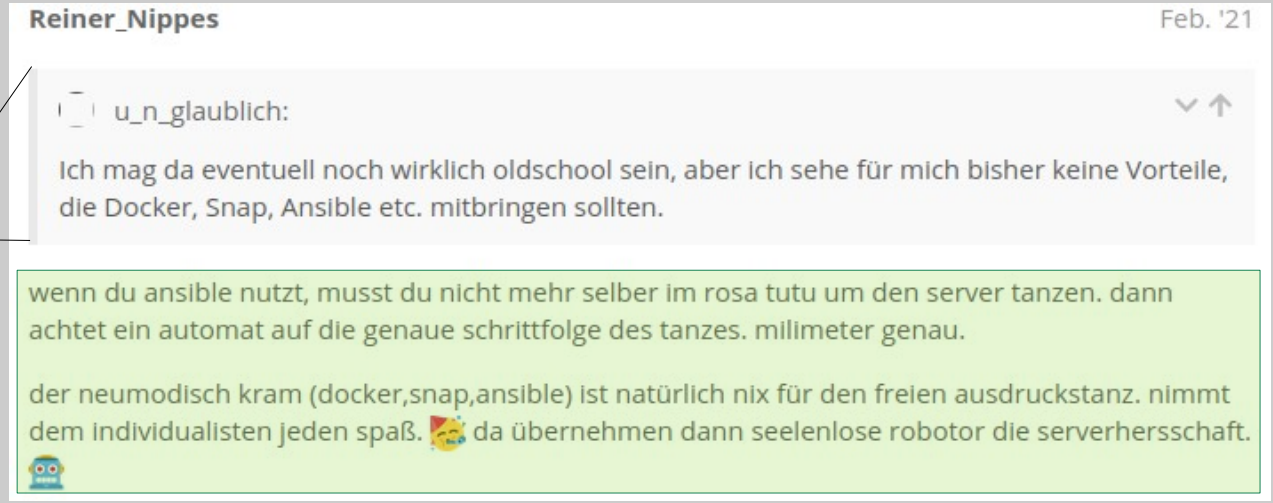
Wie soll ich Dir sonst die Frage beantworten? Grob zusammengefasst:

Schritt 1: Installation und Konfiguration des Betriebssystems (Linux, Distribution je nach Bedarf).
Schritt 2: Installation und Konfiguration des Datenbank-Servers (in der Regel MySQL).
Schritt 3: Installation des Webservers (inkl. aller notwendigen Abhängigkeiten z. B. PHP).
Schritt 4: Konfiguration des Webservers (inkl. SSL-Zertifikate).
Schritt 5: Test der vorhandenen Installationen.
Schritt 6: Installation und Konfiguration von Nextcloud.
Schritt 7: Test und Tuning der Cloud.

Für jeden Schritt gibt es die entsprechende Dokumentation des jeweiligen Anbieters.
Der Vorteil: Wenn Du Schritt 1 bis 5 beherrschst, dann ist es egal, was Du ab Schritt 6 installierst. Nextcloud, CMS, Onlineshop, etc. Alles ist möglich und Du hast die volle Kontrolle.

Ich mag da eventuell noch wirklich oldschool sein, aber ich sehe für mich bisher keine Vorteile, die Docker, Snap, Ansible etc. mitbringen sollten.

Zieht das rosa Tutu aus
und fangt an zu administrieren… ;-)



Reiner_Nippes — Feb. '21

u_n_glaublich:

Ich mag da eventuell noch wirklich oldschool sein, aber ich sehe für mich bisher keine Vorteile, die Docker, Snap, Ansible etc. mitbringen sollten.

wenn du ansible nutzt, musst du nicht mehr selber im rosa tutu um den server tanzen. dann achtet ein automat auf die genaue schrittfolge des tanzes. milimeter genau.

der neumodisch kram (docker,snap,ansible) ist natürlich nix für den freien ausdruckstanz. nimmt dem individualisten jeden spaß. 🥳 da übernehmen dann seelenlose robotor die serverhersschaft. 🤖

Quelle:
https://help.nextcloud.com/t/kein-docker-nextcloud-installation-mit-ansible-playbooks-nach-reiner-nippes/90101/45