Open Source Automation Days | OSAD 2021 | virtuelles München | 5. Oktober 2021

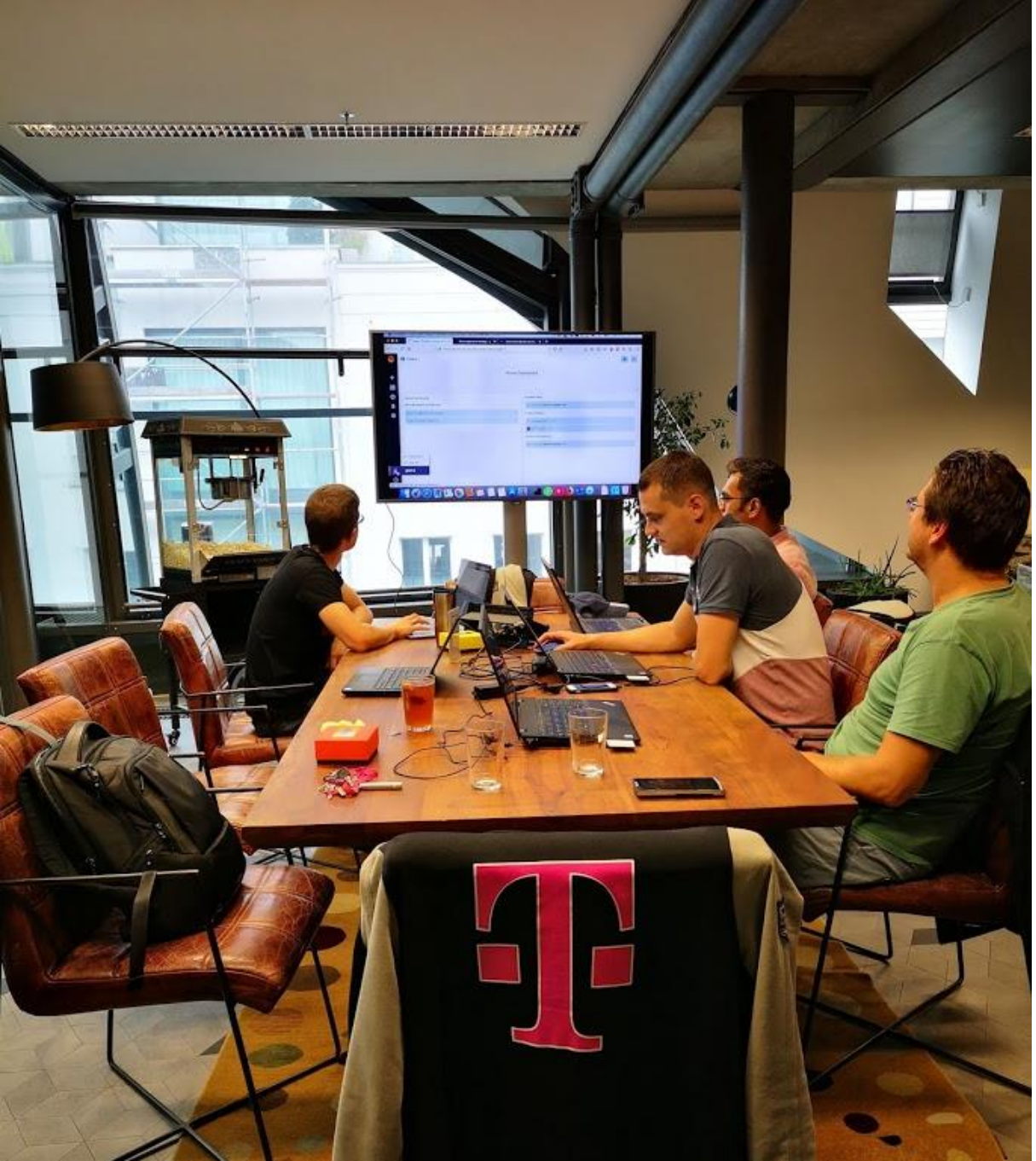
Automatisierungswerkzeuge für OpenStack-basierte Clouds

Nils Magnus (nils.magnus@t-systems.com)
Senior Cloud Architect
Open Telekom Cloud

T · · Systems · Let's power higher performance

NILS MAGNUS
Cloud Architect und Community Reach-out Manager

- Architekt bei der Open Telekom Cloud für T-Systems
- Kontakthalten mit technischen Communitys
- Werkzeuge für Cloudanwender anbieten
- Viele Jahre Security-Consulting
- Journalist, Redakteur und Autor bei Tech-Magazinen
- System Engineering und Betrieb
- Nutzer und Verfechter von Open Source
- Co-founder LinuxTag
- Vorstand der German Unix User Group



Ecosystem Squad der Open Telekom Cloud

- Softwareentwicklung und -pflege
- QA für diverse bestehenden Werkzeuge
- Zusammenarbeit mit diversen Open-Source-Communitys

- Agile, aber undogmatische Zusammenarbeit
- Work from home und nicht erst seit Corona
- Wöchentliche Team-Meetings
- Immer wieder Topic Workshops

WE HIRE!

Wir arbeiten unmittelbar an vielen wichtigen Cloud-Werkzeugen mit

CLI, SDK,
OTCExtensions

kubernetes

ANSIBLE

openstack®

RANCHER®

ZUUL

Terraform

Open Infrastructure
FOUNDATION
GOLD MEMBER

Eine kleine Evolution des IT-Betriebs.

Von der Turnschuh-Administration ...

... über die Maus-Administration ...

Elastic Cloud Server

Create ECS | Create HANA ECS

You can create 89 more ECSs. The ECSs can use up to 788 vCPUs and 1,561 GB of memory. Quota details

Start | Stop | Restart | Delete

All statuses | Name | Search by Tag

Name/ID	AZ	Status	Specifications/Image	IP Address	Tag	Operation
awx-controller d2fcd773-98f7-4275…	eu-de-03	Running	1 vCPUs | 1 GB | s3…. Standard_Ubuntu_2…	80.158.49.14… 10.0.0.70 (Pr…	--	Remote Login | More
tut1 0775e669-849d-43…	eu-de-03	Running	1 vCPUs | 8 GB | s2…. Standard_CentOS_7…	80.158.51.76… 10.0.0.240 (…	--	Remote Login | More
ecs-freddy 4b40f512-75d1-442…	eu-de-01	Running	2 vCPUs | 8 GB | s2…. Standard_Ubuntu_1…	80.158.47.18… 10.0.0.103 (…	--	Remote Login | More
bast f50f4e23-ded7-4f04…	eu-de-03	Running	1 vCPUs | 1 GB | s2…. Standard_Ubuntu_1…	80.158.45.19… 192.168.1.18…	--	Remote Login | More
Matt-Q f451f0a8-5b2e-408…	eu-de-01	Stopped	1 vCPUs | 1 GB | s2…. Standard_Ubuntu_1…	80.158.23.68… 192.168.0.16…	--	Remote Login | More
webserver-test 62f9cc89-4a5c-453…	eu-de-01	Running	1 vCPUs | 1 GB | s2…. Standard_Ubuntu_1…	80.158.43.23… 192.168.0.24…	--	Remote Login | More
AutoMatt	eu-de-01	Stopped	1 vCPUs | 8 GB | s2….	80.158.16.22…		Remote Login | More

... zur Krone der Schöpfung.

Der Kommandozeile!

```
openstack $ 
```

Erste Begegnung:
Die browserbasierte Console

Elastic Cloud Server

Create ECS | Create HANA ECS

You can create 89 more ECSs. The ECSs can use up to 788 vCPUs and 1,561 GB of memory. Quota details

Start | Stop | Restart | Delete

All statuses | Name | Search by Tag

Name/ID	AZ	Status	Specifications/Image	IP Address	Tag	Operation
awx-controller d2fcd773-98f7-4275…	eu-de-03	Running	1 vCPUs | 1 GB | s3…. Standard_Ubuntu_2…	80.158.49.14… 10.0.0.70 (Pr…	--	Remote Login | More
tut1 0775e669-849d-43…	eu-de-03	Running	1 vCPUs | 8 GB | s2…. Standard_CentOS_7…	80.158.51.76… 10.0.0.240 (…	--	Remote Login | More
ecs-freddy 4b40f512-75d1-442…	eu-de-01	Running	2 vCPUs | 8 GB | s2…. Standard_Ubuntu_1…	80.158.47.18… 10.0.0.103 (…	--	Remote Login | More
bast f50f4e23-ded7-4f04…	eu-de-03	Running	1 vCPUs | 1 GB | s2…. Standard_Ubuntu_1…	80.158.45.19… 192.168.1.18…	--	Remote Login | More
Matt-Q f451f0a8-5b2e-408…	eu-de-01	Stopped	1 vCPUs | 1 GB | s2…. Standard_Ubuntu_1…	80.158.23.68… 192.168.0.16…	--	Remote Login | More
webserver-test 62f9cc89-4a5c-453…	eu-de-01	Running	1 vCPUs | 1 GB | s2…. Standard_Ubuntu_1…	80.158.43.23… 192.168.0.24…	--	Remote Login | More
AutoMatt	eu-de-01	Stopped	1 vCPUs | 8 GB | s2….	80.158.16.22…		Remote Login | More

Automatisierung: Fehlanzeige

Wiederholung immer gleicher Abläufe

Ein Cluster = viele Systeme

manuelle Abläufe sind fehleranfällig

Browser lassen sich schlecht skripten

Das API (Application Programmable Interface)

Universell, aber mühsam

```
$ curl -giX GET https://ims.eu-de.otc.t-systems.com/v2/images \
    -H "Accept: application/json" \
    -H "User-Agent: openstacksdk/0.52.0 keystoneauth1/4.3.0"

{ "schema": "/v2/schemas/images",
  "images": [{
    "min_disk": 4,
    "created_at": "2021-08-20T09:27:46Z",
    "file": "/v2/images/9b8452c7-4267-ab61-6047dfdc9d1d/file",
    "checksum": "a1733c9887975ed17d6e4a3131f89ab8",
    "id": "9b8452c7-7504-4267-ab61-6047dfdc9d1d",
    "__description": "OpenSUSE 42.3 64bit"
    "__os_type": "Linux",
    "__image_size": "389111808",

    [ … ]
```

Das SDK (Software Development Kit)

Für Entwickler, aber keine Komplettlösung

SDK: Mit Python auf alle Cloud-Ressourcen zugreifen

Python ist die vorrangige Programmiersprache von OpenStack

Weitere SDKs sind beispielsweise für Go oder Javascript vorhanden

Objektorientierte CRUD-Operationen:

- **C**reate
- **U**pdate
- **R**etrieve
- **D**elete

SDK

```
import openstacksdk
conn = openstack.connect('otc')

image   = conn.compute.find_image(IMAGE_NAME)
flavor  = conn.compute.find_flavor(FLAVOR_NAME)
network = conn.network.find_network(NETWORK_NAME)
keypair = create_keypair(conn)

server = conn.compute.create_server(
    name=SERVER_NAME,
    image_id=image.id,
    flavor_id=flavor.id,
    networks=[{"uuid": network.id}],
    key_name=keypair.name)

server = conn.compute.wait_for_server(server)

print("ssh -i {key} linux@{ip}".format(
    key=PRIVATE_KEYPAIR_FILE,
    ip=server.access_ipv4))
```

Das CLI (Command Line Interface)

Grün auf schwarz: Die Kommandozeile

Kommandozeile: OpenStack Client

Interaktive Verwaltung der Cloud

Integration in Shellskripte

Muster: openstack *Nomen + Verb*

Hunderte von Ressourcen

Diverse Formate: Tabelle, JSON, plain

OSC

```
$ openstack --os-cloud otc server create \
  --flavor s2.medium.1 \
  --image Standard_Ubuntu_20.04_latest \
  --network test-network test-server

$ openstack server list --debug
$ openstack server list --format json
$ openstack server list -c ID -c Status -c Networks
$ openstack server list --long --fit-width
$ openstack --help
```

Installation von OpenStack Client (OSC) and Software Development Kit (SDK)

Mehrere Wege der Installation

- Betriebssystempakete
- Globale Python-Installtion
- Lokale Python-Installation
- Lokale Installation in Virtualenv
- Direkt aus den GitHub-Quellen

OSC INSTALLATION

- **Distribution based installation**

```
dnf install python-openstackclient
apt install python-openstackclient
```

- **PyPI based packages**

```
pip install python-openstackclient
```

- **The coder way**

```
git clone https://opendev.org/openstack/python-openstackclient.git
tox -e py37
source python-openstackclient/.tox/py37/bin/activate
```

SDK INSTALLATION

- **Distribution based installation**

```
dnf install python3-openstacksdk
apt install python-openstacksdk
```

- **PyPI based packages**

```
pip install openstacksdk
```

- **The coder way**

```
git clone https://opendev.org/openstack/openstacksdk.git
tox -e py37
source python-openstackclient/.tox/py37/bin/activate
```

Credentials für die Authentifikation gegenüber der Cloud

.ostackrc

```
export OS_USERNAME=username
export OS_PASSWORD=password
export OS_TENANT_NAME=eu-de
export OS_PROJECT_NAME=eu-de
export OS_AUTH_URL=\
        https://iam.eu-de.otc.t-systems.com/v3
export OS_USER_DOMAIN_NAME="OTC0000000000XXXXXXXXX"
export OS_IDENTITY_API_VERSION=3
export OS_VOLUME_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export OS_ENDPOINT_TYPE=publicURL
```

clouds.yaml und secure.yaml

```
clouds:
  my-cloud:
    profile: otc
    auth:
      project_name: eu-de_my-proj
      user_domain_name: OTC00000000001000000XXX
      username: username
      password: password
```

in ".", "~ /.config/openstack"
oder "/etc/openstack"

IaC-Frameworks (Infrastructure as Code)

Ansible & Terraform

Eine Slide über Infrastructure as Code (IaC)

Erster Ansatz zur Automatisierung: Runbooks

Detaillierte Dokumentation, was in welcher Reihenfolge zu tun ist

Prosa ist oft ungenau

Die beste Dokumentation ist ausführbarer Code

imperative (wie?) vs. declarative (was?)

standardisiert (YAML, JSON) vs. domain-spezifisch (HCL)

Eine Slide über System Provisioning und Configuration Management

Übergeordneter Process: System bereitstellen

Schritt 1: Provisionierung. Infrastruktur einrichten

Schritt 2: Konfiguration. Anwendungen anpassen

Bekannt für's Provisionieren: Terraform

Bekannt für Konfigurationen: ANSIBLE

```
resource "opentelekomcloud_vpc_v1" "vpc" {
  name = "my-vpc"
  cidr = "192.168.0.0/16"
}

resource "opentelekomcloud_vpc_subnet_v1" "subnet" {
  name       = "my-subnet"
  cidr       = cidrsubnet(opentelekomcloud_vpc_v1.vpc.cidr, 8, 0)
  gateway_ip = cidrhost(opentelekomcloud_vpc_v1.vpc.cidr, 1)

  vpc_id     = opentelekomcloud_vpc_v1.vpc.id
}
```

```
$ terraform apply
```

Applies configuration. Only difference with currently stored state with be applied. "apply" is expected to be idempotent.

```
$ terraform destroy
```

Destroys whole managed infrastructure.

Ansible Playbook 1/7: Konfiguration festlegen

```
---
- name: Mein komplettes Playbook
  hosts: "localhost"
  vars:
    router_name:        "my-router"
    network_name:       "my-network"
    subnet_name:        "my-subnet"
    subnet_cidr:        "192.168.1.0/24"
    subnet_dns_servers: "['100.125.4.25', '8.8.4.4']"
    enable_snat:        false
    server_name:        "my-server"
    region:             "eu-de"
    az:                 "eu-de-02"
    server_sg:          "sg-vc-web" # TODO: create
    image:              "Standard_Ubuntu_20.04_latest"
    pubkey_name:        "nils-magnus-otc-4"
    pubkey:             "ssh-rsa AAAAB3NzaCE […] AAAADAQABw== Nils Magnus (nils.magnus@t-systems.com)"
    flavor:             "s3.medium.4"
    cloud:              "vaxxchain" # for credentials
```

Ansible Playbook 2/7: Infrastruktur anlegen

```
roles:
   - { role: opentelekomcloud.vpc, state: 'present' }
 tasks:
   - name: 1 | Upload SSH public key
     openstack.cloud.keypair:
       name: "{{ pubkey_name }}"
       public_key: "{{ pubkey }}"

   - name: 2 | Create bastion server
     openstack.cloud.server:
       name: "{{ server_name }}", wait: "yes"
       region_name: "{{ region }}"
       network: "{{ network_name }}"
       security_groups: "{{ server_sg }}"
       availability_zone: "{{ az }}"
       image: "{{ image }}"
       key_name: "{{ pubkey_name }}"
       cloud: "{{ cloud }}"
       flavor: "{{ flavor }}"
     register: returned_server
```

Ansible Playbook 3/7:
Sicheres Einloggen auf neuem Host per SSH

```
- name: 3 | Display my new IP address
  debug:
    msg: "Die IP lautet {{ returned_server.server.accessIPv4 }}"

- name: 4 | Lokalen, alten Hostkey löschen
  remote_user: magnus
  delegate_to: localhost
  command: "ssh-keygen -f /home/magnus/.ssh/known_hosts -R {{ returned_server.server.accessIPv4 }}"

- name: 5 | Wait for SSH port become ready
  delegate_to: "{{ returned_server.server.accessIPv4 }}"
  remote_user: ubuntu
  wait_for:
    port: 22
```

Ansible Playbook 4/7: Softwareaktualisierung

```
- name: 6 | Alle bestehenden Pakete aktualisieren
  remote_user: ubuntu
  become: yes
  delegate_to: "{{ returned_server.server.accessIPv4 }}"
  apt:
    upgrade: dist
    update_cache: yes

- name: 7 | Zusätzliche Pakete installieren
  remote_user: ubuntu
  become: yes
  delegate_to: "{{ returned_server.server.accessIPv4 }}"
  apt:
    name:
      - python3-venv
      - emacs
```

Ansible Playbook 5/7: Großreinemachen

```
- name: 8 | Komische Dateien löschen
      remote_user: ubuntu
      become: yes
      delegate_to: "{{ returned_server.server.accessIPv4 }}"
      file:
        path: "{{ item }}"
        state: absent
      with_items:
        - /etc/update-motd.d/10-help-text
        - /etc/update-motd.d/50-motd-news
        - /etc/update-motd.d/99-otc-motd
        - /home/magnus/.bash_logout
        - /home/magnus/.ostackrc
        - /home/magnus/.s3rc
```

Ansible Playbook 6/7: Konfiguration festlegen

```
- name: 9 | Gruppen anlegen
  remote_user: ubuntu
  become: yes
  delegate_to: "{{ returned_server.server.accessIPv4 }}"
  group:
    name: docker

- name: 10 | User anlegen
  remote_user: ubuntu
  become: yes
  delegate_to: "{{ returned_server.server.accessIPv4 }}"
  user:
    name: magnus
    comment: Nils Magnus
    password: $6$Ge3PyA […] 7VLCa7z4w1.Bh/LgA9rw3KPiQr1
    shell: /bin/bash
    groups: admin, docker
    append: yes
```

Ansible Playbook 7/7: Konfiguration festlegen

```
- name: 11 | Installiere meinen GitHub SSH-Key in authorized_keys
  remote_user: ubuntu
  become: yes
  delegate_to: "{{ returned_server.server.accessIPv4 }}"
  authorized_key:
    user: magnus
    key: https://github.com/Nils-Magnus.keys
```

```
controller$ ansible-playbook allinone.yaml
```

Ansible Collections for Open Telekom Cloud

Key Facts

Ansible facilitates **modules**, which communicate with the OTC API. Modules are shipped as **collections**.

Addition to native OpenStack modules, which provide extended functionality.

Open source project

cloud - collection containing additional Ansible modules for managing OTC specific resources (or modified OpenStack modules in case of behavioral differences).

common – collection with most frequently used functions.

Installation

with Ansible-Galaxy:

```
$ ansible-galaxy collection \
    install opentelekomcloud.cloud
```

It's also possible to run in a Python virtual environment.

Principle

Write playbook on local controller.

Run playbook facilitating collections.

Documentation

https://galaxy.ansible.com/opentelekomcloud

https://docs.ansible.com/ansible/latest/galaxy/user_guide.html

https://docs.ansible.com/ansible/latest/user_guide/collections_using.html#collections

Zugang zur Cloud

Einmalige Vorgänge, Troubleshooting → Console

Direkter Zugang, Tooling → API

Entwickler, Einbinden in Bibliotheken → SDK

Troubleshooting, nur mächtiger → CLI

Automatisierung durch Beschreibung → IaC

Ressourcen

Bug Reports, Contributions und Zusammenarbeit

Fehler gefunden?

Trage einen Issue in unserem GitHub-Repo ein:
https://github.com/optentelekomcloud/

→ Projekt auswählen → "Issues"

Dokumentation

- OpenStack SDK und CLI:
  https://docs.openstack.org/openstacksdk/latest/
- OTC-Extensions:
  https://python-otcextensions.readthedocs.io/en/latest/
- Weitere Projekte in unserer GtHub-Organisation

Kontakt

E-Mail: DL-PBCOTCDELECO@t-systems.com

We hire System Engineers and Cloud Architects!

Vielen Dank!

Mehr Informationen über unsere Projekte, detaillierte Beispiele und vollständige Konfigurationen findet Ihr in unserem GitHub-Repository:

https://github.com/opentelekomcloud/

Antworten auf Fragen und Suchergebnisse nach Tutorials liefert unser Community Portal:

https://community.open-telekom-cloud.com/