EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Using Terraform to define infrastructure as code

Maximilian Hanussek
High Performance and Cloud Computing Group
Zentrum für Datenverarbeitung

Applied Bioinformatics Group
Eberhard Karls University Tübingen

12.10.20
de.NBI cloud user meeting[3]
Virtual

Workshop Goals

- Get a basic understanding of Terraform
- Get an idea how the syntax is working
- Learn what Terraform can be used for
- Create a basic cluster setup (master and compute)

Outline

- Introduction
  - Terraform
  - (Virtual-)Cluster
  - Software Stack
- Hands-On
  - Part1: Terraform Basics
  - Part2: Build more complex infrastructures
  - Part3: Basic virtual cluster deployment
  - Part4: VALET

What is Terraform?

- (Open source) Tool from HashiCorp[1]
- Allows you to maintain infrastructure as Code
- Write configurations as functions and modules
- With all benefits of it (versioning, sharing)
- Provides reproducibility

[1]https://www.terraform.io

Why Terraform?

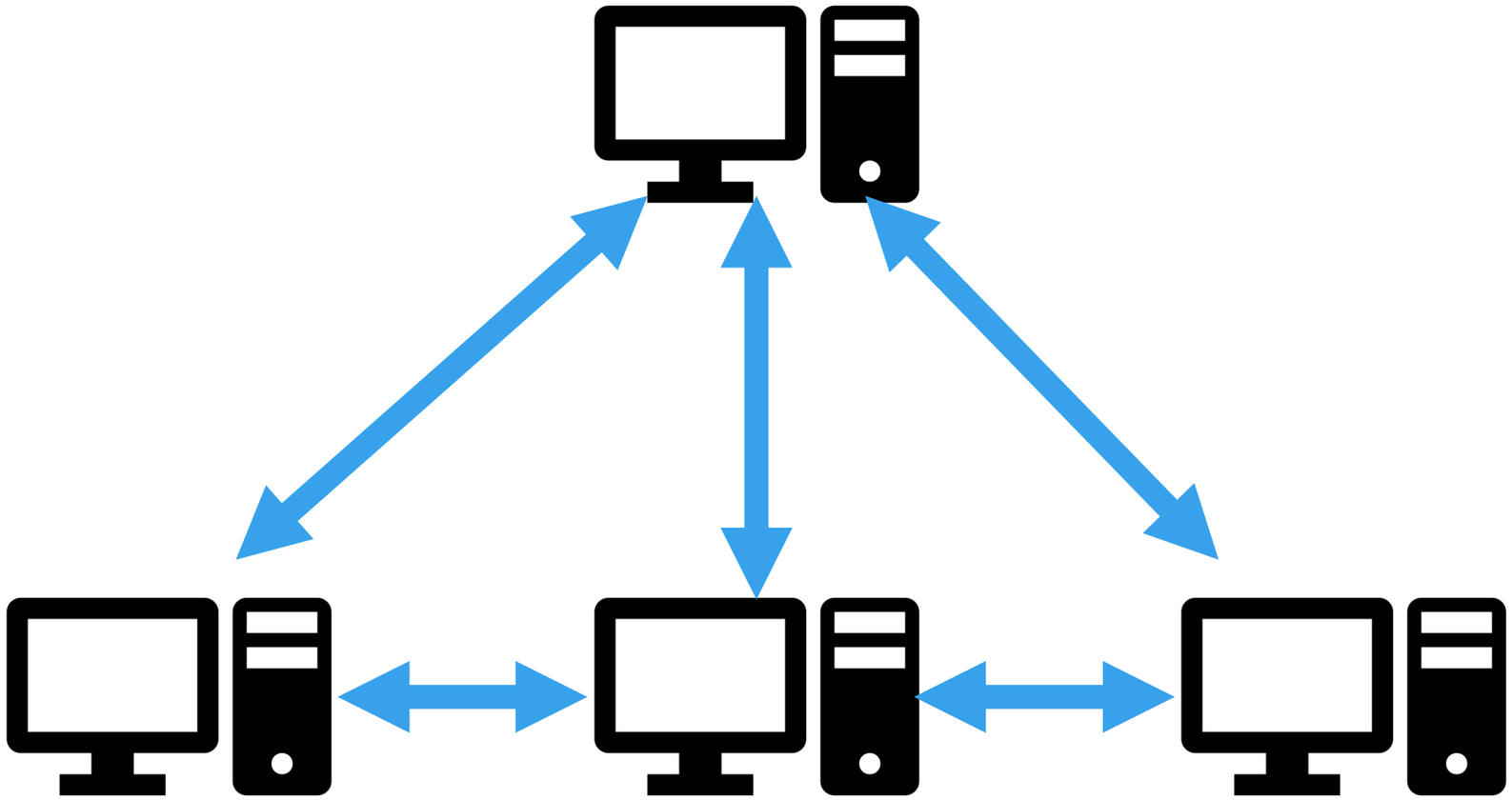
- Easy to install
- Light weighted
- Many tutorials out there
- Comes with a variety of interfaces
- Documentation
- Makes your and our life easier
- Lots of valuable features

Introduction - Cluster

A **computer cluster** is a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system.[2]

[2]https://en.wikipedia.org/wiki/Computer_cluster

Introduction - Cluster

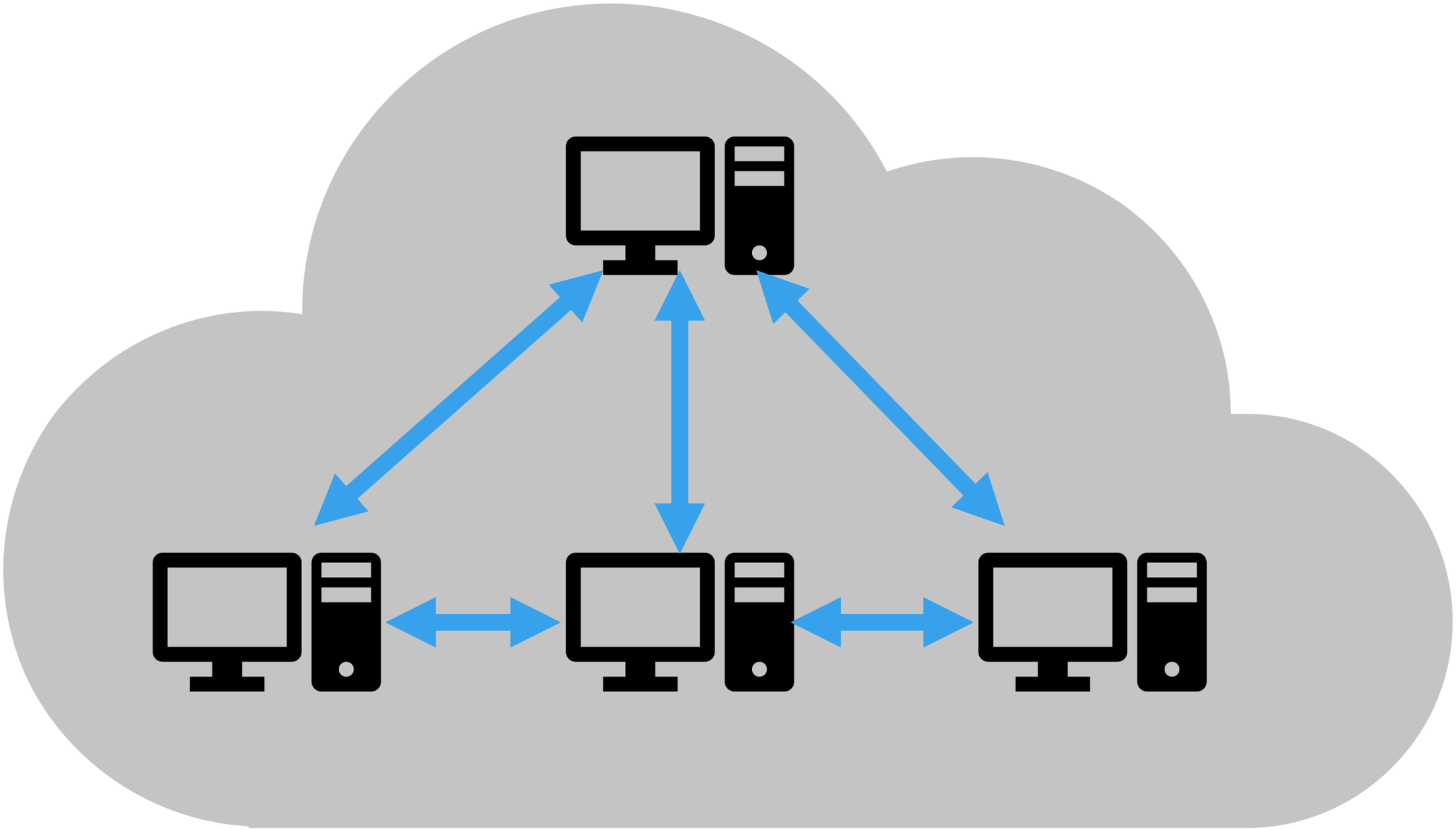


Introduction - Cluster

Use cases:

- Need multiple resources of a single VM
- Your workload can be parallelized
- No need to handle different VMs, just a single cluster
- Run whole pipelines, schedule jobs differently

Introduction - Cluster



Introduction – Softwarestack

- Master node(s) (VMs)
- Compute node(s) (VMs)
- Network connection
- (Shared Filesystem)
- (Batch System/Scheduler)
- (Middleware)

Introduction – Terraform

Classical HPC cluster:

- Built up machines
- Configure networks
- Set up shared filesystem
- Configure Scheduler

Virtual cluster:

- Spawn virtual machines
- Configure virtual networks
- Set up shared filesystem
- Configure Scheduler
- ➢ **Use Infrastructure as a Code**

How to use Terraform?

Prerequisites

- Internet connection
- Download and install Terraform (0.13.2)
  - https://releases.hashicorp.com/terraform/

- Have a key pair available in the OpenSSH format, if not generate one

```
ssh-keygen -t rsa -b 4096
```

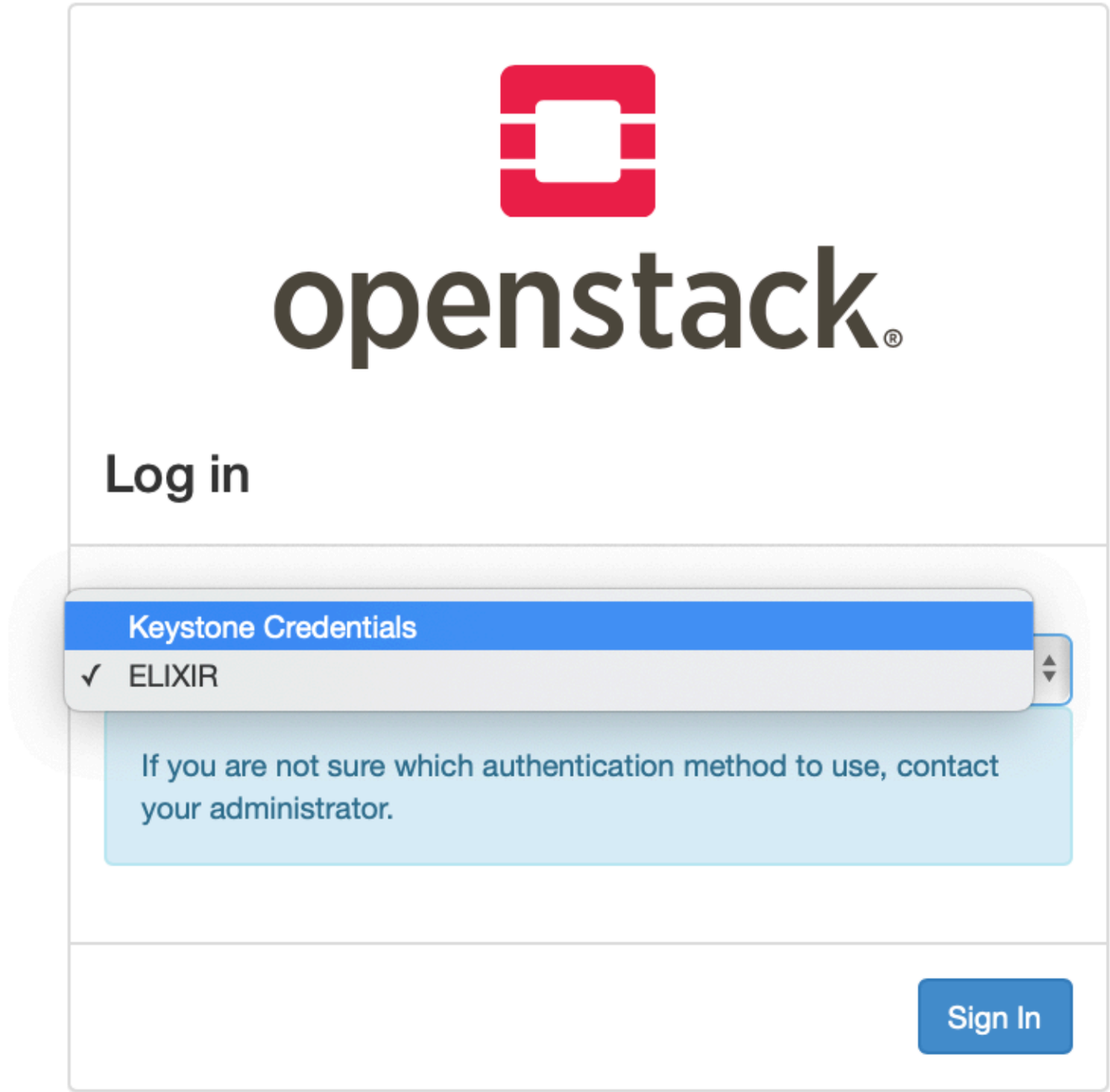
- Download/clone GitHub repo (git clone, wget, curl, via browser)

https://github.com/MaximilianHanussek/deNBI_cloud_terraform_2020.git

Prerequisites - Login

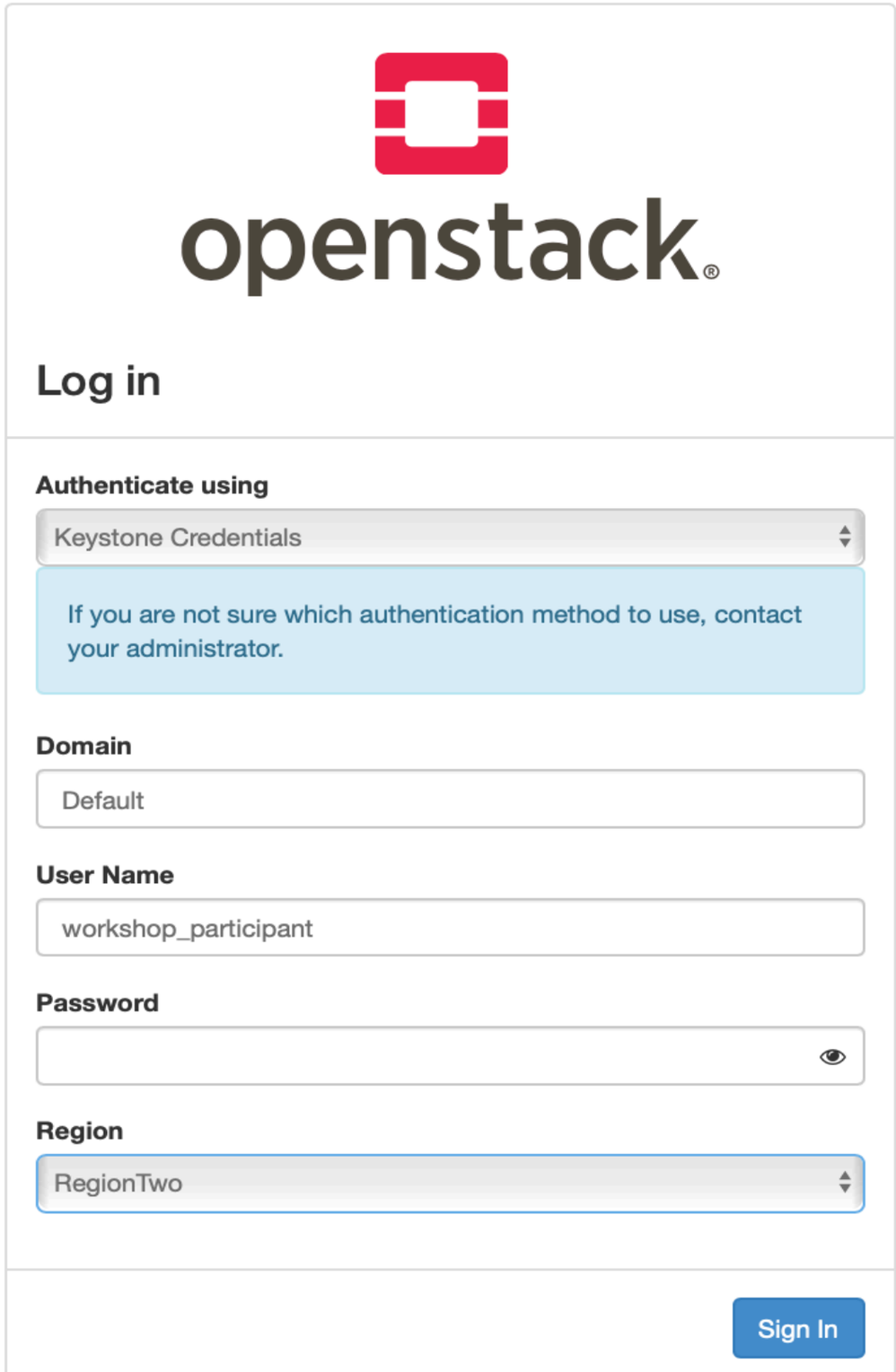
- Dashboard Login
  - URL: https://denbi.uni-tuebingen.de/

Prerequisites - Login



https://denbi.uni-tuebingen.de/

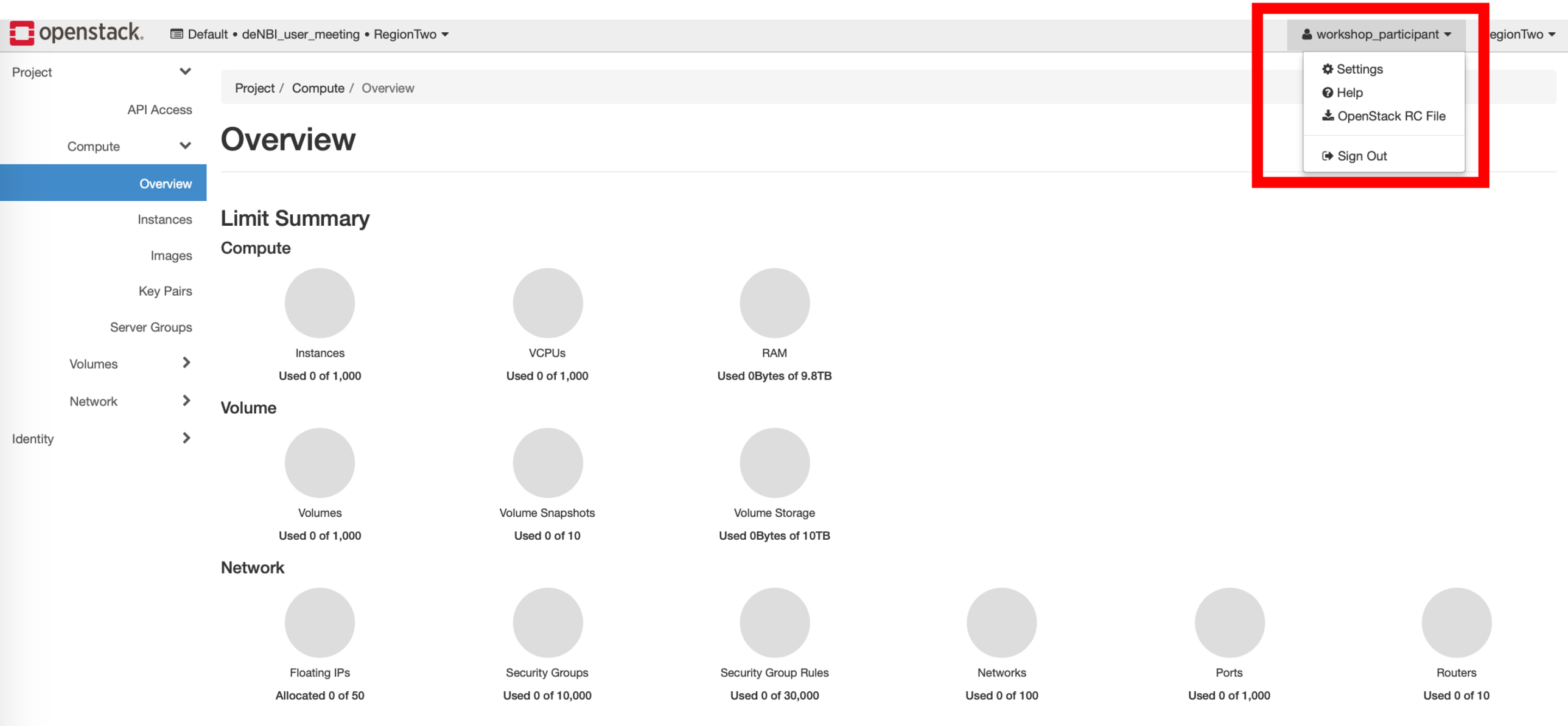
Prerequisites - Login



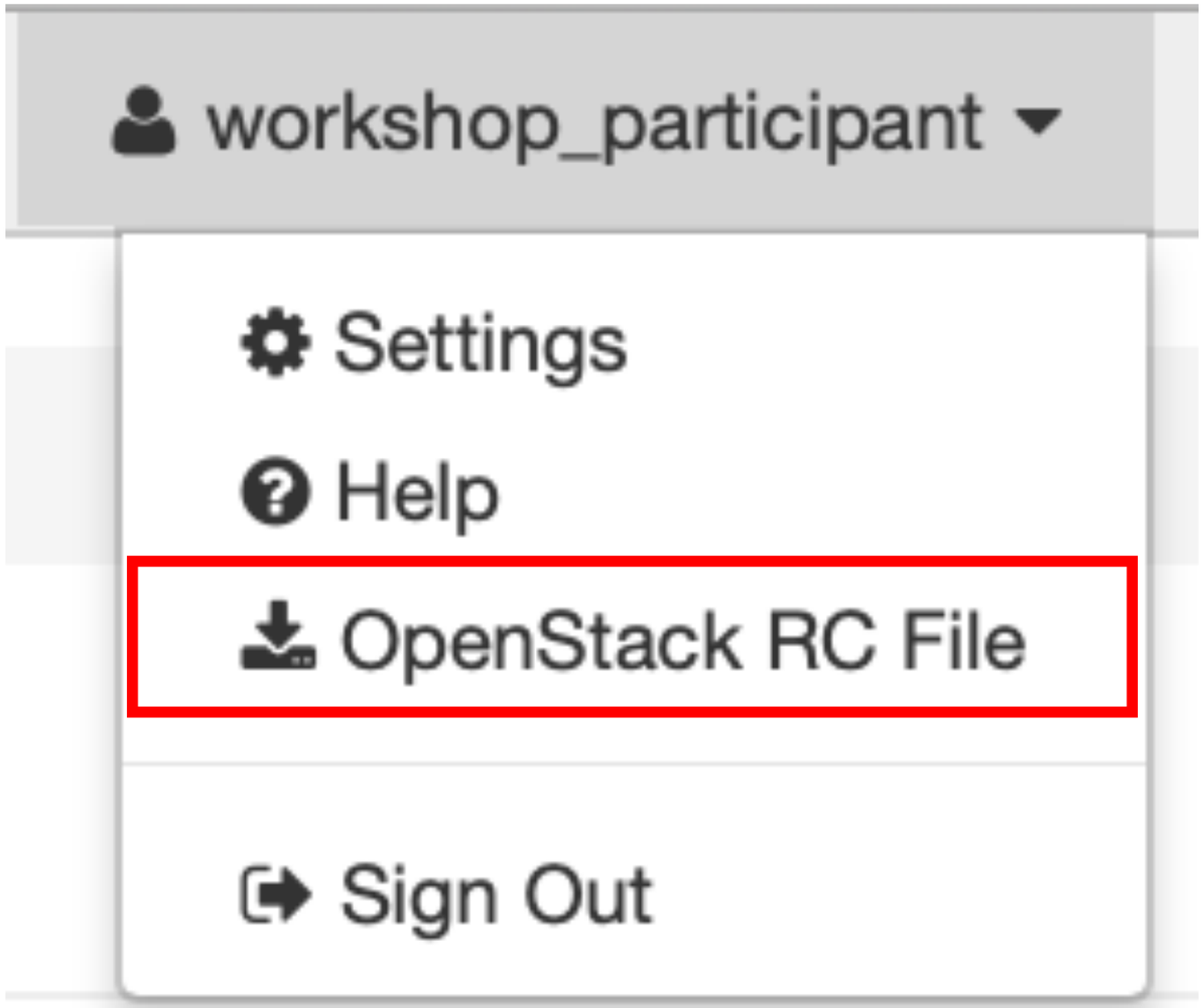
Prerequisites - Login

Domain: Default
User name: workshop_participant
Password: unicorn42
Region: RegionTwo

Prerequisites– RC File



Prerequisites – RC File



Hands-On Part 1

- **Goal:** Understand TF structure and start VM
- Change into:

```
deNBI_cloud_terraform_workshop/terraform_workshop_part_1
```

- See a bunch of files
  1. providers.tf
  2. key_pair.tf
  3. security_group.tf
  4. vars.tf
  5. main.tf
  6. versions.tf

providers.tf

```
provider "openstack" {}
```

versions.tf

```
terraform {
  required_providers {
    openstack = {
      source = "terraform-providers/openstack"
    }
  }
  required_version = ">= 0.13"
}
```

Hands-On Part 1

key_pair.tf

```
resource "openstack_compute_keypair_v2" "workshop_keypair" {
  name       = var.workshop-key-name
  public_key = var.public-key
}
```

terraform key words

user chosen name

security_group.tf

```
resource "openstack_networking_secgroup_v2"
"terraform_workshop_sec_group" {

  name                  = var.security-groups[0]
  description           = "Allow SSH access and ping request“

  delete_default_rules = true

}
```

terraform key words

user chosen name

key word values

security_group.tf (outgoing connections)

```
resource "openstack_networking_secgroup_rule_v2" "egress_public_4" {
  direction         = "egress"
  ethertype         = "IPv4"
  security_group_id =
openstack_networking_secgroup_v2.terraform_workshop_sec_group.id
}

resource "openstack_networking_secgroup_rule_v2" "egress_public_6" {
  direction         = "egress"
  ethertype         = "IPv6"
  security_group_id =
openstack_networking_secgroup_v2.terraform_workshop_sec_group.id
}
```

security_group.tf (SSH)

```
resource "openstack_networking_secgroup_rule_v2"
"ingress_public_4_ssh" {

  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
  port_range_min    = 22
  port_range_max    = 22
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id =
openstack_networking_secgroup_v2.terraform_workshop_sec_gr
oup.id
}
```

security_group.tf (ICMP)

```
resource "openstack_networking_secgroup_rule_v2"
"ingress_public_4" {

  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "icmp"
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id =
openstack_networking_secgroup_v2.terraform_workshop_sec_gr
oup.id
}
```

vars.tf (volume)

```
variable "cinder-disc-size" {
  default = 10
}

variable "cinder-storage-backend" {
  default = "quobyte_hdd"
}

variable "volume-name" {
  default = "maxhanussek-workshop-volume"
}
```

Needs to be changed

Cloud site specific

vars.tf (flavor, name, image)

```
variable "flavors" {
  type = map
  default = {
    "workshop-vm" = “de.NBI default"
  }
}

variable "vm-name" {
  default = "maxhanussek-workshop-vm"
}

variable "workshop-image" {
  default = "CentOS 7.7 2020-07-07"
}
```

vars.tf (keys)

```
variable "workshop-key-name" {
  default = "maxhanussek-keypair"
}

variable "public-key" {
  default = ""
}
```

vars.tf (security-group, network)

```
variable "security-groups" {
  default = [
    "maxhanussek-sec-group"
  ]
}


variable "network" {
  default = "denbi_uni_tuebingen_external2"

}
```

main.tf (image, volume)

```
data "openstack_images_image_v2" "workshop_image" {
  name        = var.workshop-image
  most_recent = true
}

resource "openstack_blockstorage_volume_v2" "cinder_volume" {
  name        = var.volume-name
  size        = var.cinder-disc-size
  volume_type = var.cinder-storage-backend
}
```

main.tf (instance)

```
resource "openstack_compute_instance_v2" "workshop_vm" {

name            = var.vm-name
flavor_name     = var.flavors["workshop-vm"]
image_id = data.openstack_images_image_v2.workshop_image.id
key_pair = openstack_compute_keypair_v2.workshop_keypair.name
security_groups = var.security-groups

  network {
    name = var.network
  }
```

main.tf (instance)

```
block_device {
    uuid                  =
data.openstack_images_image_v2.workshop_image.id
    source_type           = "image"
    destination_type      = "local"
    boot_index            = 0
    delete_on_termination = true
  }

block_device {
    uuid                  =
openstack_blockstorage_volume_v2.cinder_volume.id
    source_type           = "volume"
    destination_type      = "volume"
    boot_index            = -1
    delete_on_termination = true
  }
}
```

Use Terraform

- Initialize Terraform in Terraform directory (`deNBI_cloud_terraform_2020/terraform_workshop_part_1`)

```
terraform init
```

Hands-On Part 1

Initializing the backend...

Initializing provider plugins...

...

* provider.openstack: version = "~> 1.31.0"

Terraform has been successfully initialized!

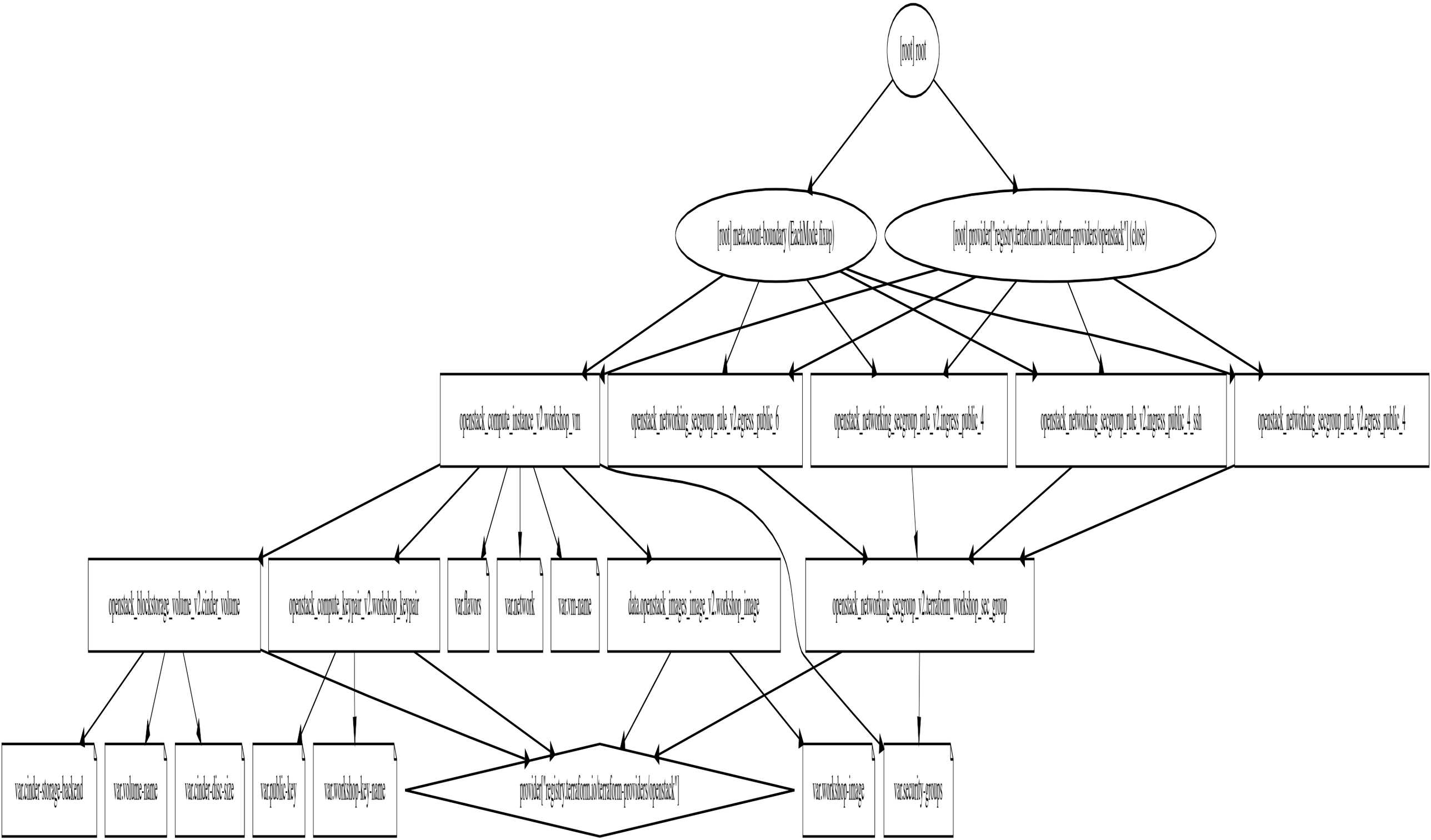
...

Use Terraform

- Run Terraform graph
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_1`)

```
terraform graph | dot -Tsvg > graph.svg
```

Hands-On Part 1



Use Terraform

- Run Terraform plan
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_1`)

```
terraform plan
```

Hands-On Part 1

```
Error: One of 'auth_url' or 'cloud' must be
specified


on providers.tf line 1, in provider "openstack":
 1: provider "openstack" {}
```

Use Terraform

- Source OpenStack credentials
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_1`)

```
source deNBI_user_meeting-openrc.sh
```

Password:

```
terraform plan
```

Hands-On Part 1

```
# openstack_blockstorage_volume_v2.cinder_volume will be
created
  + resource "openstack_blockstorage_volume_v2"
"cinder_volume" {
      + attachment        = (known after apply)
      + availability_zone = (known after apply)
      + id                = (known after apply)
      + metadata          = (known after apply)
      + name              = "maxhanussek-workshop-volume"
      + region            = (known after apply)
      + size              = 10
      + volume_type       = "quobyte_hdd"
    }
```

Hands-On Part 1

```
Plan: 8 to add, 0 to change, 0 to destroy.
```

Use Terraform

- Run Terraform apply (confirm with ‘yes’)
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_1`)

```
terraform apply
```

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
```

Check the Dashboard for the created resources

https://denbi.uni-tuebingen.de

Use Terraform

- Run Terraform destroy (confirm with ‘yes’)
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_1`)

```
terraform destroy
```

```
Destroy complete! Resources: 8 destroyed.
```

Check the Dashboard for the destroyed resources

https://denbi.uni-tuebingen.de

Use Terraform

- Change to: `terraform_workshop_part_2`

- Initialize Terraform in Terraform directory (`deNBI_cloud_terraform_2020/terraform_workshop_part_2`)

```
terraform init
```

Use Terraform

- Goal: Start multiple (similar) VMs
- Add new variable to end of vars.tf

```
variable "node-count" {
  default = 3
}
```

Use Terraform

- Change parts of main.tf (Volume resource)

```
resource "openstack_blockstorage_volume_v2"
"cinder_volume" {
  count = var.node-count
  name = "${var.volume-name}-${count.index}"
  size = var.cinder-disc-size
  volume_type = var.cinder-storage-backend
}
```

Use Terraform

- Change parts of main.tf (Instance resource)

```
resource "openstack_compute_instance_v2" "workshop_vm" {
  count          = var.node-count
  name           = "${var.vm-name}-${count.index}"
  flavor_name = var.flavors["workshop-vm"]
  image_id       = data.openstack_images_image_v2.workshop_image.id
  key_pair       = openstack_compute_keypair_v2.workshop_keypair.name
  security_groups = var.security-groups

  network {
    name = var.network
  }
```

Use Terraform

- Change parts of main.tf (Block device)

```
block_device {
  uuid =
element(openstack_blockstorage_volume_v2.cinder_volume.*.id
, count.index)
  source_type                = "volume"
  destination_type           = "volume"
  boot_index                 = -1
  delete_on_termination = true
 }
}
```

Use Terraform

- Change to: `terraform_workshop_part_2`

- Run terraform plan & terraform apply
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_2`)

```
terraform plan

terraform apply
```

Use Terraform

- Changed network variable in vars.tf

```
variable "network" {
  default = "denbi_uni_tuebingen_internal"
}
```

terraform.tfstate

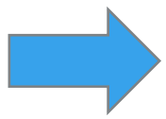
- Have a look at terraform.tfstate file
  - Holds all information about the infrastructure
  - !!! Holds also your credentials
  - !!! Holds also keys

➢ Be careful if you share these file (Git, …)

Use Terraform

- Change the infrastructure
- Open `vars.tf` file

```
variable "node-count" {
  default = 3
}
```



```
variable "node-count" {
  default = 4
}
```

```
terraform plan
```

```
Plan: 2 to add, 0 to change, 0 to destroy.
```

```
terraform apply
```

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Use Terraform

- Change the infrastructure again
- Open `vars.tf` file

```
variable "node-count" {
  default = 4
}
```

```
variable "node-count" {
  default = 2
}
```

Hands-On Part 2

```
terraform plan
```

```
Plan: 0 to add, 0 to change, 4 to destroy.
```

```
terraform apply
```

```
Apply complete! Resources: 0 added, 0 changed, 4 destroyed.
```

Use Terraform

- Change the infrastructure by accident
- Go to the dashboard and delete a VM
- (Please take one of your own ones!)

```
terraform plan

terraform apply
```

```
terraform plan
```

```
Plan: 2 to add, 0 to change, 0 to destroy.
```

```
terraform apply
```

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Use Terraform

- Changing the infrastructure through
- Variable changes (remove or add)
- Handling of deletions from outside

Terraform makes it easy to recover from unwanted changes or scaling up and down the infrastructure

Use Terraform

- Run terraform destroy

```
terraform destroy
```

- Check terraform.tfstate again


```
{
  "version": 4,
  "terraform_version": "0.13.2",
  "serial": 183,
  "lineage": "8c08c491-29cd-9f7a-b79c-5f607a59374d",
  "outputs": {},
  "resources": []
}
```

Use Terraform

- Change to: `terraform_workshop_part_3`

- Initialize Terraform in Terraform directory (`deNBI_cloud_terraform_2020/terraform_workshop_part_3`)

```
terraform init
```

Use Terraform

- Goal: Build a cluster structure
- Cluster setup:
  - 3 VMs (1 Master node, 2 Compute nodes)
  - 3 Volumes
  - 2 different networks
  - 2 different flavors
  - 2 different images

Hands-On Part 3

vars.tf

```
variable "volume-name" {
  type = map
  default = {
    "master" = "maxhanussek-workshop-volume-master"
    "compute"= "maxhanussek-workshop-volume-compute"
  }
}

variable "vm-name" {
  type = map
  default = {
    "master" = "maxhanussek-workshop-vm-master"
    "compute"= "maxhanussek-workshop-vm-compute"
  }
}
```

vars.tf

```
variable “flavors" {
  type = map
  default = {
    "master" = “de.NBI small"
    "compute"= “de.NBI default"
  }
}

variable “workshop-image" {
  type = map
  default = {
    "master" = " CentOS 7.7 2020-07-07"
    "compute"= " CentOS 7.8 2020-07-07"
  }
}
```

Hands-On Part 3

vars.tf

```
variable “network" {
  type = map
  default = {
    "master" = “denbi_uni_tuebingen_external2"
    "compute"= “denbi_uni_tuebingen_internal"
  }
}

variable “private-key-path" {
  default = “/path/to/private/key“
}

variable "internal-key-name" {
  default = "maxhanussek-internal-key"
}
```

Hands-On Part 3

key_pair.tf

```
resource "tls_private_key" "internal_connection_key" {
  algorithm  = "RSA"
  rsa_bits   = 4096
}

resource "openstack_compute_keypair_v2"
"internal_connection_key_openstack" {
  name       = var.internal-key-name
  public_key =
tls_private_key.internal_connection_key.public_key_openssh
}
```

Hands-On Part 3

main.tf

```
data "openstack_images_image_v2" "workshop_image_master" {
  name        = var.workshop-image["master"]
  most_recent = true
}

data "openstack_images_image_v2" "workshop_image_compute"
{
  name        = var.workshop-image["compute"]
  most_recent = true
}
```

main.tf

```
resource "openstack_compute_instance_v2" "workshop_vm_master" {
  name            = var.vm-name["master"]
  flavor_name     = var.flavors["master"]
  image_id        =
data.openstack_images_image_v2.workshop_image_master.id
  key_pair        =
openstack_compute_keypair_v2.workshop_keypair.name
  security_groups = var.security-groups

  network {
    name = var.network["compute"]
  }

  network {
    name = var.network["master"]
    access_network = true
  }
```

main.tf (provisioner)

```
provisioner "file" {
    content = tls_private_key.internal_connection_key.private_key_pem
    destination = "~/.ssh/connection_key.pem"

    connection {
      type        = "ssh"
      private_key = file(var.private-key-path)
      user        = "centos"
      timeout     = "5m"
      host        = self.access_ip_v4
    }
}
```

main.tf (provisioner)

```
provisioner "file" {
    source      = "hello_world.txt"
    destination = "/home/centos/hello_world.txt"

  connection {
    type        = "ssh"
    private_key = file(var.private-key-path)
    user        = "centos"
    timeout     = "5m"
    host        = self.access_ip_v4
  }
}
```

main.tf (provisioner)

```
provisioner "remote-exec" {
    script = "set_internal_private_key_permissions.sh"


  connection {
    type        = "ssh"
    private_key = file(var.private-key-path)
    user        = "centos"
    timeout     = "5m"
    host        = self.access_ip_v4
  }
}
```

Hands-On Part 3

main.tf

```
provisioner "remote-exec" {
    script = "mount_cinder_volumes.sh"

  connection {
    type        = "ssh"
    private_key = file(var.private-key-path)
    user        = "centos"
    timeout     = "5m"
    host        = self.access_ip_v4
  }
}
```

Use Terraform

- Change to: `terraform_workshop_part_3`

- Run terraform plan and terraform apply
  (`deNBI_cloud_terraform_2020/terraform_workshop_part_3`)

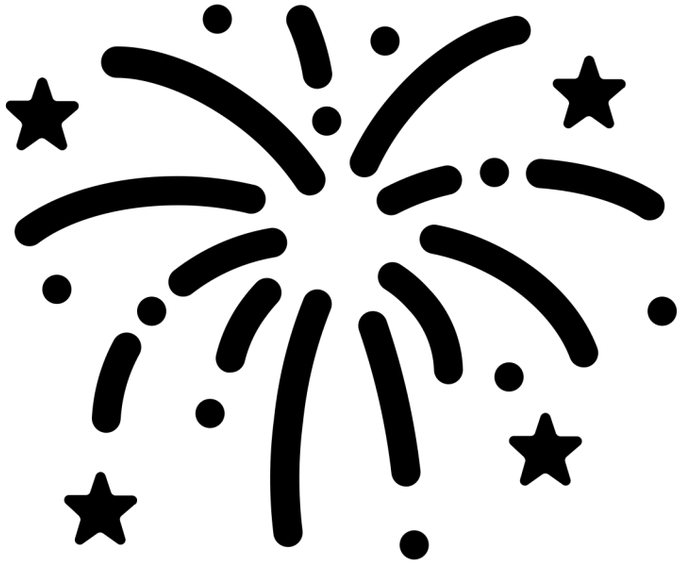
```
terraform plan

terraform apply
```

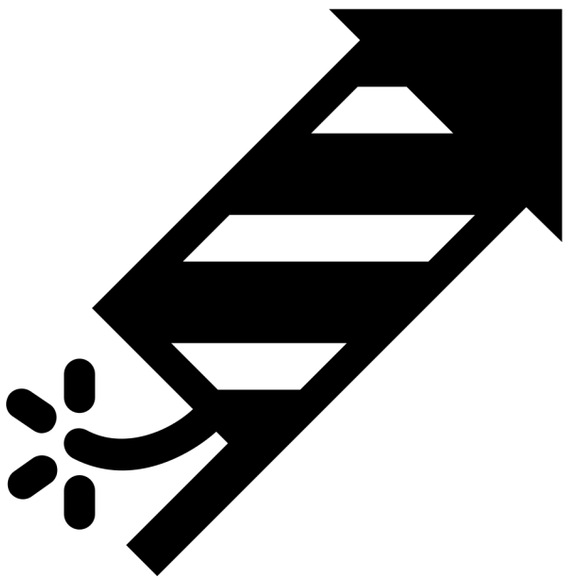
Use Terraform

- Make use of the map module
- Let TF create “one time” keys for you
- Provisioner module can be used for post mods

Terraform makes it more convenient to deploy more complex infrastructures in an automated way and also be able to handle it.



Congratulations we have deployed a virtual cluster infrastructure



Hands-On Part 4

Use Terraform

- Goal: Build an advanced virtual cluster (VALET)
- Features:
  - Batch System (PBS TORQUE)
  - Middleware UNICORE (incl. Workflow engine)
  - Monitoring system (Zabbix)
  - Add and remove nodes without downtime
  - Automated load based scaling available

https://github.com/MaximilianHanussek/VALET