

Cloud provider  
→ Working with [Terraform](#)

What's it all about?

**Quote:**

“Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently.”

# Terraform resources

- Why Terraform?
- Install Terraform.

# Hetzner API token

- Access your cloud project using the Hetzner **GUI** interface.
- Go to Security --> API Tokens --> Generate API token
- Provide a name and hit Generate API token.
- Copy the generated token's value and store it in a secure location e.g. a password manager.

## Caution

The Hetzner GUI blocks future access to the token.

# Minimal Terraform configuration

```
# Define Hetzner cloud provider
terraform {
  required_providers {
    hcloud = {
      source = "hetznercloud/hcloud"
    }
  }
  required_version = ">= 0.13"
}

# Configure the Hetzner Cloud API token
provider "hcloud" {
  token = "your_api_token_goes_here"
}

# Create a server
resource "hcloud_server" "helloServer" {
  name          = "hello"
  image         = "debian-12"
  server_type   = "cx22"
}
```

# Terraform init

```
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hetznercloud/hcloud...
- Installing hetznercloud/hcloud v1.46.1...
- Installed hetznercloud/hcloud v1.46.1 (signed by a HashiCorp partner, key ID 5219EACB3A77198B)

...
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. ...
```

# Terraform plan

```
$ terraform plan
```

```
Terraform used the selected providers to generate the following execution plan. Resource actions ...
+ create
```

```
Terraform will perform the following actions:
```

```
# hcloud_server.helloServer will be created
+ resource "hcloud_server" "helloServer" {
    + allow_deprecated_images      = false
    + backup_window                = (known after apply)
...
}
```

```
Plan: 1 to add, 0 to change, 0 to destroy.
```

# Terraform apply

```
$ terraform apply
...
Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: yes

hcloud_server.helloServer: Creating...
hcloud_server.helloServer: Still creating... [10s elapsed]
hcloud_server.helloServer: Creation complete after 14s [id=45822789]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

# A word on storing secrets

## Guide to Managing Terraform secrets.

variables.tf	env	providers.tf
<pre>variable "hcloud_token" {     nullable = false     sensitive = true }</pre>	<pre>export TF_VAR_hcloud_token=mBSxD...</pre>	<pre>provider "hcloud" {     token = var.hcloud_token }</pre>
<pre>. env # sourcing environment terraform apply</pre>		

# Credentials by E-Mail

Your server "hello" was created!

You can access your server with the following credentials:

IPv4      **128.140.108.60**

IPv6      2a01:4f8:1c1c:8e3a::/64

User      root

Password      rJ3pNvJXbqMp3XNTvFdq

You will be prompted to change your password on your first login.

To improve security, we recommend that you add an SSH key when creating a server.

Problems: 😞

- Firewall blocks **ssh** server access:

```
$ ssh root@128.140.108.60
ssh: connect to host 128.140.108.60 port 22: Connection refused
```

Access by Vnc console login only

- IP and (initial) credentials by email 😱

Solution:

1. Add firewall inbound **ssh** access rule.
2. Configure **ssh** public key login.

## ssh access, firewall

```
resource "hcloud_firewall" "sshFw" {
  name = "ssh-firewall"
  rule {
    direction = "in"
    protocol  = "tcp"
    port      = "22"
    source_ips = ["0.0.0.0/0", "::/0"]
  }
}

resource "hcloud_server" "helloServer" {
  ...
  firewall_ids = [hcloud_firewall.sshFw.id]
}
```

## ssh access, public key

```
resource "hcloud_ssh_key" "loginUser" {
  name      = "goik@hdm-stuttgart.de"
  public_key = file("~/ssh/id_ed25519.pub")
}

...
resource "hcloud_server" "helloServer" {
  ...
  ssh_keys    = [hcloud_ssh_key.loginUser.id]
}
```

Note: Use the Hetzner Web GUI for removing any conflicting manually installed ssh keys beforehand.

# Apply ssh key access

```
$ terraform apply

# hcloud_firewall.sshFw will be created
+ resource "hcloud_firewall" "sshFw" {
  ...
}

# hcloud_server.helloServer will be created
+ resource "hcloud_server" "helloServer" {
  ...
}

# hcloud_ssh_key.goik will be created
+ resource "hcloud_ssh_key" "loginUser" {
  ...
}

Plan: 3 to add, 0 to change, 0 to destroy.
...
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

# Output data details #1/2

See [terraform output documentation](#):

## File outputs.tf

```
output "hello_ip_addr" {
  value      = hcloud_server.helloServer.ipv4_address
  description = "The server's IPv4 address"
}

output "hello_datacenter" {
  value      = hcloud_server.helloServer.datacenter
  description = "The server's datacenter"
}
```

## Result

```
$ terraform output
hello_datacenter = "nbg1-dc3"
hello_ip_addr = "159.69.152.37"
```

```
$ terraform output hello_ip_addr
"159.69.152.37"
```

# Output data details #2/2

## File outputs.tf

```
output "hello_ip_addr" {
  value      = hcloud_server.helloServer.ipv4_address
  description = "The server's IPv4 address"
}

output "hello_datacenter" {
  value      = hcloud_server.helloServer.datacenter
  description = "The server's datacenter"
}
```

## terraform output - json

```
{
  "hello_datacenter": {
    "sensitive": false,
    "type": "string",
    "value": "nbg1-dc3"
  },
  "hello_ip_addr": {
    "sensitive": false,
    "type": "string",
    "value": "159.69.152.37"
  }
}
```

## Problem 2: VCS and visible provider API token 😱

### Versioned file main.tf:

```
...  
provider "hcloud" { token = "xdaGfz9Lmw08Swkg ... "}  
...
```

### Solution:

- Declare a variable `hcloud_token` in a `variables.tf` file
- Add a non-versioned file `secrets.auto.tfvars`.
- Optional: Provide a versioned `secrets.auto.tfvars.template` documenting file

# Solution by variable

Declaring **hcloud\_token** in variables.tf:

```
variable "hcloud_token" { # See secret.auto.tfvars  
  nullable = false  
  sensitive = true  
}
```

Defining **hcloud\_token**'s value in secrets.auto.tfvars:

```
hcloud_token="xdaGfz9Lmw08Swkg ... "
```

Using **hcloud\_token** in main.tf:

```
provider "hcloud" { token = var.hcloud_token }
```

Template file  
secrets.auto.tfvars.template:

```
hcloud_token="your_api_token_goes_here"
```

# Solution by file

```
# Configure the Hetzner Cloud API token
provider "hcloud" {
    token = file("../provider.token.key"
}
```

## Content of file provider.token.key:

```
xdaGfz9Lmw08SWkg ...
```

## Content of file provider.token.key.template:

```
your_api_token_goes_here
```

## Followup exercise

332. Incrementally creating a base system

# Overview

Cloud provider  
→ Cloud-init

# Introduction and reference

- Cloud Stack [Conference talk](#).
- [Cloud-init documentation](#)
- Validation
- Debugging

## In a nutshell

- Distribution image containing pre-installed Cloud Init
- Script configurable installation options

# Configuration options

- Individual CRUD file operations
- Supplying ssh user and host keys.
- Adding users
- ...
- Installing packages
- System Upgrade + reboot
- Arbitrary command execution

# Bash DIY

## main.tf

```
resource "hcloud_server" "nginxServer" {  
    name          = "hello"  
    image         = "debian-12"  
    server_type   = "cx22"  
    user_data     = file("myinit.sh")  
}
```

## myinit.sh

```
#!/bin/sh  
  
aptitude update && aptitude -y upgrade
```

Followup exercise

### 333. Automatic Nginx installation

# Terraform interface to Cloud Init

```
resource "hcloud_server" "web" {
  name          = var.server_name
  ...
  user_data = file("userData.yml")
}
```

```
#cloud-config
packages:
  - nginx
runcmd:
  - systemctl enable nginx
  - rm /var/www/html/*
  - >
    echo "I'm Nginx @ $(dig -4 TXT +short o-o.myaddr.l.google.com @ns1.google.com)
created $(date -u)" >> /var/www/html/index.html
```

# Using template files

```
# main.tf
resource "local_file" "user_data" {
  content      = templatefile("tpl/userData.yml", {
    loginUser      = "devops"
  })
  filename      = "gen/userData.yml"
}

resource "hcloud_server" "helloServer" {
  ...
  # Will become gen/userData.yml
  user_data      = local_file.user_data.content
}
```

```
# tpl/userData.yml
...
users:
  - name: ${loginUser}
  ...
```

```
# tpl/userData.yml
...
users:
  - name: devops
  ...
```

# cloud-init file validation

- **# cloud-init schema --system --annotate**

```
...
apt:          # E1
- debconf_selections: openssh-server  openssh-server/password-authentication  boolean
              false openssh-server  openssh-server/permit-root-login      boolean false
...
# Errors: -----
# E1: [ {'debconf_selections': 'openssh-server ....boolean false'} ] is not of type 'object'
```

- **# cloud-init schema --config-file /var/lib/cloud/instance/user-data.txt**

```
Valid cloud-config: /var/lib/cloud/instance/user-data.txt
```

# Yaml syntax quirk 1: Missing indentation

```
resource "tls_private_key" "host" {
    algorithm    = "ED25519"
}
...
resource "local_file" "user_data" {
    content      = templatefile("tpl/userData.yml", {
        host_ed25519_private =
            tls_private_key.host.private_key_openssh
        host_ed25519_public   = ...
    })
    filename     = "gen/user_data.yml"
}
```

```
...
ssh_keys:
    ed25519_private: |
        -----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAAABG5vbmUA...
-----END OPENSSH PRIVATE KEY-----
```

---

```
# E1: File gen/user_data.yml is not valid YAML.
in "<unicode string>", line 19, column 1:
b3B1bnNzaC1rZXktdjEAAAAABG5vbmUA ...
```

# Yaml missing indentation resolution

```
# main.tf
resource "local_file" "user_data" {
  content = templatefile("tpl/userData.yml", {
    host_ed25519_private =
      indent(4, tls....host.private_key_openssh)
    ...})
  filename = "gen/user_data.yml"
}
```

```
# gen/usr_data.yml
ssh_keys:
  ed25519_private: |
    -----BEGIN OPENSSH PRIVATE KEY-----
    b3BlbnNzaC1 ...
    ...
    ...
    -----END OPENSSH PRIVATE KEY-----
```

## Yaml syntax quirk 2: Wrong indentation

```
# gen/usr_data.yml

ssh_keys:
  ed25519_private: |
    -----BEGIN OPENSSH PRIVATE KEY-----
    b3B1bnNzaC1 ...
    ...
    ...
    -----END OPENSSH PRIVATE KEY-----
```

```
# /etc/ssh/ssh_host_ed25519_key
...
ssh_keys:
  ed25519_private: |
    -----BEGIN OPENSSH PRIVATE KEY-----
    b3B1bnNzaC1 ...
    ...
    ...
    -----END OPENSSH PRIVATE KEY-----
```

# Solving Yaml quirks using yamlencode / JSON

```
resource "local_file" "user_data" {
  content = templatefile("tpl/userData.yml", {
    host_ed25519_private = tls_private_key.host.private_key_openssh
    host_ed25519_public   = tls_private_key.host.public_key_openssh
    devopsUsername         = hcloud_ssh_key.loginUser.name
    ...
  })
}
```

```
#cloud-config

${yamlencode(
{
  ...
  ssh_keys = {
    ed25519_private = host_ed25519_private
    ed25519_public  = host_ed25519_public
  } ...
}
)}

users:
- name: ${devopsUsername} ...
```

# Watch out for your enemies!

```
root@hello:~# journalctl -f
May 06 04:41:20 hello cloud-init[898]: Cloud-init v. 22.4.2 finished at Mon, 06 May 2024 04:41:20 +0000. Datasource DataS...
...
May 06 04:46:16 hello sshd[927]: Invalid user abc from 43.163.218.130 port 33408
May 06 04:46:17 hello sshd[927]: Received disconnect from 43.163.218.130 port 33408:11: Bye Bye [preauth]
May 06 04:46:17 hello sshd[927]: Disconnected from invalid user abc 43.163.218.130 port 33408 [preauth]
...
May 06 04:50:54 hello sshd[930]: fatal: Timeout before authentication for 27.128.243.225 port 59866
...
May 06 04:52:45 hello sshd[933]: Invalid user cos from 43.163.218.130 port 59776
...
May 06 04:53:04 hello sshd[935]: Invalid user admin from 194.169.175.35 port 51128
May 06 04:53:49 hello sshd[937]: User root from 43.163.218.130 not allowed because not listed in AllowUsers
May 06 04:53:49 hello sshd[937]: Disconnected from invalid user root 43.163.218.130 port 50592 [preauth]
```

Followup exercise

### 334. Working on Cloud-init

Problem: Duplicate known\_hosts entry on re-creating server

### Problem of repeated **terraform apply**:

```
$ ssh root@128.140.108.60
@@@@@@@WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@ IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
```

# Solution: Generating known\_hosts ...

```
resource "local_file" "known_hosts" {
  content      = join(" "
    , [ hcloud_server.helloServer.ipv4_address
        , tls_private_key.host.public_key_openssh ]
    )
  filename     = "gen/known_hosts"
  file_permission = "644"
}
```

157.180.68.60 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5...XXgAC3eaeHMQ8ePtbnw

... and ssh wrapper

## main.tf

```
resource "local_file" "ssh_script" {
  content = templatefile("tpl/ssh.sh", {
    ip=hcloud_server.hello.ipv4_address
  })
  filename      = "bin/ssh"
  file_permission = "755"
  depends_on     = [local_file.known_hosts]
}
```

## tpl/ssh.sh

```
#!/usr/bin/env bash

GEN_DIR=$(dirname "$0")/../.gen

ssh -o UserKnownHostsFile= \
"$GEN_DIR/known_hosts" devops@${ip} "$@"
```

Followup exercise

### 335. Solving the `~/.ssh/known_hosts` quirk

# Failsafe console login

```
#cloud-config

chpasswd:
  list: |
    root:bingo9wingo
  expire: False
```

# Avoiding Yaml parsing issues

```
#cloud-config

write_files:
  - content:
      ${base64encode(private_key_pem)}
    encoding: base64
    path: /etc/nginx/snippets/cert/private.pem
```

Note

Congrats to paulyng

# Overview

Cloud provider  
→ Volumes

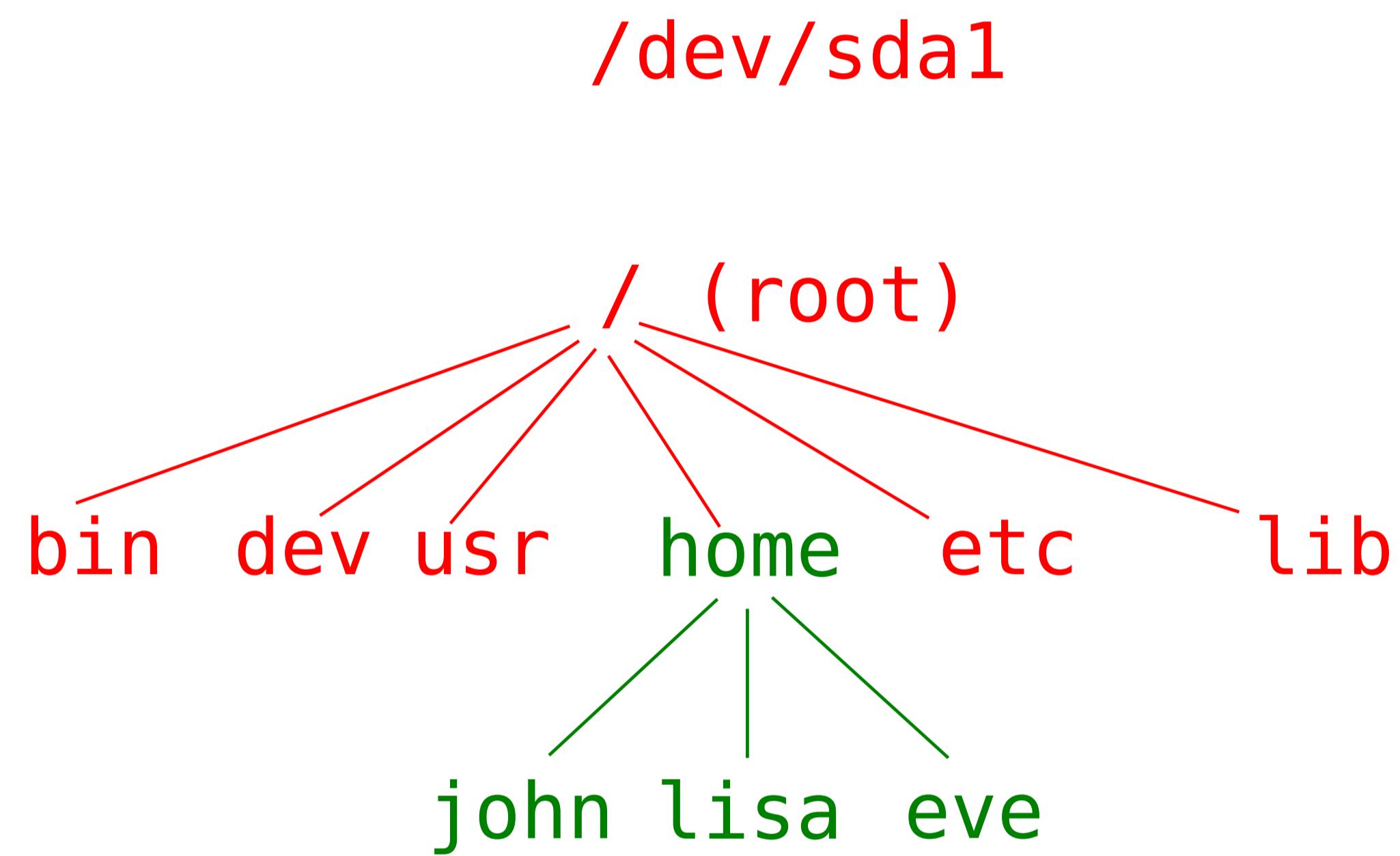
# A volume: The easy way

```
resource "hcloud_server" "helloServer" {
    server_type  = "cx22"
    ...
}

resource "hcloud_volume" "volume01" {
    name        = "volume1"
    size        = 10
    server_id   = hcloud_server.helloServer.id
    automount   = true
    format      = "xfs"
}
```

```
df
...
/mnt/HC_Volume_100723816
```

# Unix mount



/dev/sda2

Followup exercise

### 336. Partitions and mounting

# Volume details

```
output "volume_device" {  
  value = hcloud_volume.volume01.linux_device  
  description = "The volume's device"  
}
```

```
terraform apply  
...  
volume_device = "/dev/disk/by-id/scsi-0HC_Volume_102626366"
```

Desired /etc/fstab entry:

```
/dev/disk/by-id/scsi-0HC_Volume_102626366 /volume01 xfs rw,defaults 0 2
```

# Providing a mount point's name

```
resource "hcloud_server" "helloServer" {
    ...
    user_data = templatefile("tpl/userData.yml", {
        ...
        volume01Id = hcloud_volume.volume01.id
    })
}
resource "hcloud_volume" "volume01" {
    server_id = hcloud_server.helloServer.id
    ...
}
```

## Problem: Cyclic dependency

```
helloServer <--> volume01
```

# De-coupling server and volume creation

## main.tf

```
resource "hcloud_volume" "vol01" {
  name = "volume1" ...
}
resource "hcloud_server" "hello" {
  user_data = templatefile("userData.yml.tpl", {
    volumename = hcloud_volume.vol01.name
    device = hcloud_volume.vol01.linux_device
  }) ...
}
resource "hcloud_volume_attachment" "main" {
  volume_id=hcloud_volume.vol01.id
  server_id=hcloud_server.hello.id ... }
```

```
# userData.yml.tpl
runcmd:
  - mkdir ${volumename}
  - echo ${device} ${volumename} xfs rw,defaults 0 2 >> /etc/fstab
```

## Followup exercise

337. Mount point's name specification

Cloud provider  
→ Terraform modules

# Terraform module Documentation

- Reuse Configuration with Modules
- What are Terraform Modules and How to Use Them?

# Example: Creating a JSON host meta data file

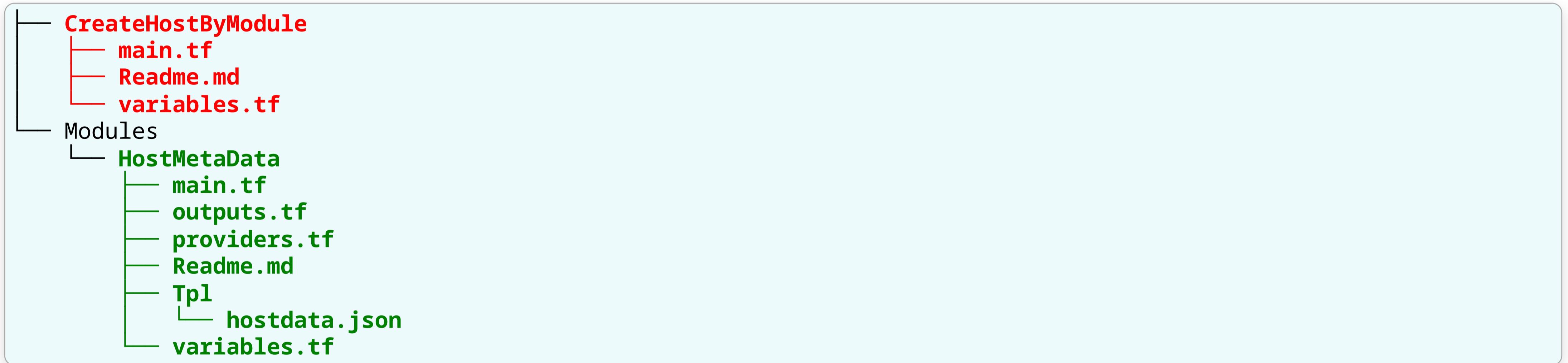
```
# main.tf
resource "hcloud_server" "myServer" { ... }

resource "local_file" "hostdata" {
  content = templatefile("Tpl/hostdata.json", {
    ip4      = hcloud_server.helloServer.ipv4_address
    ip6      = hcloud_server.helloServer.ipv6_address
    location = hcloud_server.helloServer.location
  })
  filename  = "Gen/hostdata.json"
}
```

```
{ "network": {
    "ipv4": "${ip4}", "ipv6": "${ip6}" ,
    "location": "${location}"}
```

```
{
  "network": {
    "ipv4": "157.180.78.16",
    "ipv6": "2a01:4f9:c013:be69::1"
  },
  "location": "hel1"
}
```

# Parent module / sub module layout



# Parent module implementation

```
# variable.tf
variable "hcloud_token" { sensitive = true }
```

```
# main.tf
module "createHostAmongMetaData" {
  source      = "../Modules/HostMetaData"
  name        = "myServer"
  hcloud_token = var.hcloud_token
}
```

# Sub module implementation

```
# variable.tf
variable "name" {
  type = string
  nullable = false
}
```

```
variable "hcloud_token" {
  type = string
  nullable = false
}
```

```
# main.tf
resource "hcloud_server" "newServer" {
  name          = var.name
  image         = "debian-12"
  server_type   = "cx22" }

resource "local_file" "hostdata" {
  content = templatefile("${path.module}/Tpl/hostdata.json", {
    ip4    = hcloud_server.newServer.ipv4_address
    ip6    = hcloud_server.newServer.ipv6_address
    location = hcloud_server.newServer.location
  })
  filename      = "Gen/${var.name}.json" }
```

# Sub module, JSON template file Tpl/hostdata.json and result

```
{  
  "network": {  
    "ipv4": "${ip4}",  
    "ipv6": "${ip6}"  
  },  
  "location": "${location}"  
}
```

```
{  
  "network": {  
    "ipv4": "157.180.78.16",  
    "ipv6": "2a01:4f9:c013:be69::1"  
  },  
  "location": "hell1"  
}
```

# Parent module vs. sub module context

```
resource "local_file" "hostdata" {
  content = templatefile(
    "${path.module}/Tpl/hostdata.json" ...
  )
```

```
└── CreateHostByModule
    ├── main.tf
    ├── Readme.md
    └── variables.tf
    └── Tpl
        └── hostdata.json
└── Modules
    └── HostMetaData
        ├── main.tf
        ├── outputs.tf
        ├── providers.tf
        ├── Readme.md
        └── Tpl
            └── hostdata.json
    └── variables.tf
```

## Followup exercise

338. A module for **ssh** host key handling

Cloud provider  
→ Exercises related [DNS server](#)

## Subdomain per group

- Dedicated course related **DNS** server ns1.hdm-stuttgart.cloud.
- One subdomain per group e.g. **g3.sdi.hdm-stuttgart.cloud** corresponding to Group 3.
- Zone edits require a subdomain specific hmac secret key being provided as **dnsupdate.sec** file in your personal group entry below the **SDI course**:

```
hmac-sha512:g3.key:I5sDDS3L1BU...
```

### Note

The per zone secrets have been created using **tsig-keygen**.

- Edits become globally visible. Mind the TTL setting: A higher value means you'll have to wait longer until updates become visible.

# Key file location

Key file available in your working group below 113475 Software defined Infrastructure.

# Querying DNS by zone transfer

```
$ export HMAC=hmac-sha512:g3.key:YXWSeh31...
$ dig @ns1.hdm-stuttgart.cloud -y $HMAC -t AXFR g3.sdi.hdm-stuttgart.cloud
...
g3.sdi.hdm-stuttgart.cloud. 600 IN      SOA      ns1.hdm-stuttgart.cloud. ...
g3.sdi.hdm-stuttgart.cloud. 600 IN      NS       ns1.hdm-stuttgart.cloud.
g3.sdi.hdm-stuttgart.cloud. 600 IN      SOA      ns1.hdm-stuttgart.cloud. ...
g3.key.                      0        ANY     TSIG     hmac-sha512. 1746433052 300 64 bak...
...
```

See [AXFR](#) for details.

# Creating an "A" record with TTL=10

```
export HMAC=hmac-sha512:g3.key:YXWSeh3l...
$ nsupdate -y $HMAC
> server ns1.hdm-stuttgart.cloud
> update add www.g3.sdi.hdm-stuttgart.cloud 10 A 141.62.75.114
> send
> quit
$ dig +noall +answer @ns1.hdm-stuttgart.cloud www.g3.sdi.hdm-stuttgart.cloud
www.g3.sdi.hdm-stuttgart.cloud. 9 IN A 141.62.75.114
$ dig +noall +answer @8.8.8.8 www.g3.sdi.hdm-stuttgart.cloud
www.g3.sdi.hdm-stuttgart.cloud. 3 IN A 141.62.75.114
```

# Modify by delete/create

```
$ nsupdate -y $HMAC
> server ns1.hdm-stuttgart.cloud
> update delete www.g3.sdi.hdm-stuttgart.cloud. 10 IN A      141.62.75.114
> send
> quit
>
$ dig +noall +answer @8.8.8.8 www.g3.sdi.hdm-stuttgart.cloud
$
```

## Note

Examples at DNS Updates with nsupdate

# Mind record caching

```
goik>dig +noall +answer www.g3.sdi.hdm-stuttgart.cloud
www.g3.sdi.hdm-stuttgart.cloud. 9 IN A 141.62.75.114
```

- DNS record caching.
- Another **9 seconds** to go before DNS cache invalidation.

Followup exercise

339. Enhancing your web server.

# Overview

Cloud provider  
➡ Terraform loops

# Loop documentation

- Terraform tips & tricks: loops, if-statements, and gotchas
- Strings and Templates
- The count Meta-Argument

# Using count

## Defining 10 server

```
resource "hcloud_server" "server" {  
  count      = 10  
  name       = "www-${count.index}"  
  user_data = local_file.user_data[count.index].content  
  ...  
}
```

## 10 corresponding A-records

```
resource "dns_a_record_set" "dnsRecordSet" {  
  count      = 10  
  zone       = "g03.sdi.hdm-stuttgart.cloud."  
  name       = hcloud_server.server[count.index].name  
  addresses = [hcloud_server.server[count.index].ipv4_address]  
}
```

# Overview

Cloud provider  
→ Terraform and DNS

# Bind server ns1.sdi.hdm-stuttgart.cloud

- Providing **DNS** info for sdi.hdm-stuttgart.cloud and sub-zones:
  - g01.sdi.hdm-stuttgart.cloud
  - g02.sdi.hdm-stuttgart.cloud
  - ...
- Remote API for per-zone editing

# DNS provider

```
terraform {  
  required_providers {  
    dns = {  
      source = "hashicorp/dns"  
    }  
  }  
}
```

This requires:

- A corresponding variable "**dns\_secret**" [{ . . . }](#) declaration.
- An e.g., `export TF_VAR_dns_secret = "sVfw2a...vAUqw=="` (non - versioned!) environment variable.

# DNS provider configuration

```
provider "dns" {  
  update {  
    server      = "ns1.sdi.hdm-stuttgart.cloud"  
    key_name    = "g12.key." # Corresponding to your group e.g., Group 12  
    key_algorithm = "hmac-sha512"  
    key_secret   = var.dns_secret  
  }  
}
```

This requires:

- A corresponding variable "**dns\_secret**" `{ . . . }` declaration.
- An e.g., `export TF_VAR_dns_secret = "sVfw2a...vAUqw=="` (non - versioned!) environment variable.

# Defining an "A" record

```
resource "dns_a_record_set" "helloRecord" {  
    zone = "${var.dnsSubnetName}." # The dot matters!  
    name = hcloud_server.helloServer.name  
    addresses = [hcloud_server.helloServer.ipv4_address]  
    ttl = 10  
}
```

## Followup exercises

- 340. Creating DNS records
- 341. Creating a host with corresponding DNS entries
- 342. Creating a fixed number of servers

Cloud provider  
→ Generating web SSL certificates

# Understanding web certificates

[https://www.youtube.com/watch?v=T4Df5\\_cojAs](https://www.youtube.com/watch?v=T4Df5_cojAs)

# Certificate trust level

## **Domain Validated**

Fully automated process solely based on **DNS** / infrastructure challenges.

## **Organization Validated**

Checking organization in question.

## **Extended Validation**

Additional checks i.e. telephone based verification.

# Certificates by Terraform

```
provider "acme" {
  server_url = "https://acme-staging-v02.api.letsencrypt.org/directory"

  # Production:
  # server_url = "https://acme-v02.api.letsencrypt.org/directory"
}
resource "tls_private_key" "private_key" { algorithm = "RSA" }
resource "acme_registration" "reg" {
  account_key_pem = tls_private_key.private_key.private_key_pem
  email_address   = "nobody@example.com"
}
resource "acme_certificate" "certificate" {
  ...
  dns_challenge { ... }
}
```

# dns\_challenge provider

```
resource "acme_certificate" "certificate" {  
    ...  
    dns_challenge {  
        provider = "route53"  
        ...  
    }  
}
```

## acme DNS provider list:

- acme-dns
- alidns
- ...
- rfc2136
- ...
- zonomi

# rfc2136 provider configuration

```
dns_challenge {
  provider = "rfc2136"

  config = {
    RFC2136_NAMESERVER      = "ns1.sdi.hdm-stuttgart.cloud"
    RFC2136_TSIG_ALGORITHM  = "hmac-sha512"
    RFC2136_TSIG_KEY         = "gxy.key."
    RFC2136_TSIG_SECRET      = var.dns_secret # Relates to environment variable TF_VAR_dns_secret
  }
}
```

# Bind server log file

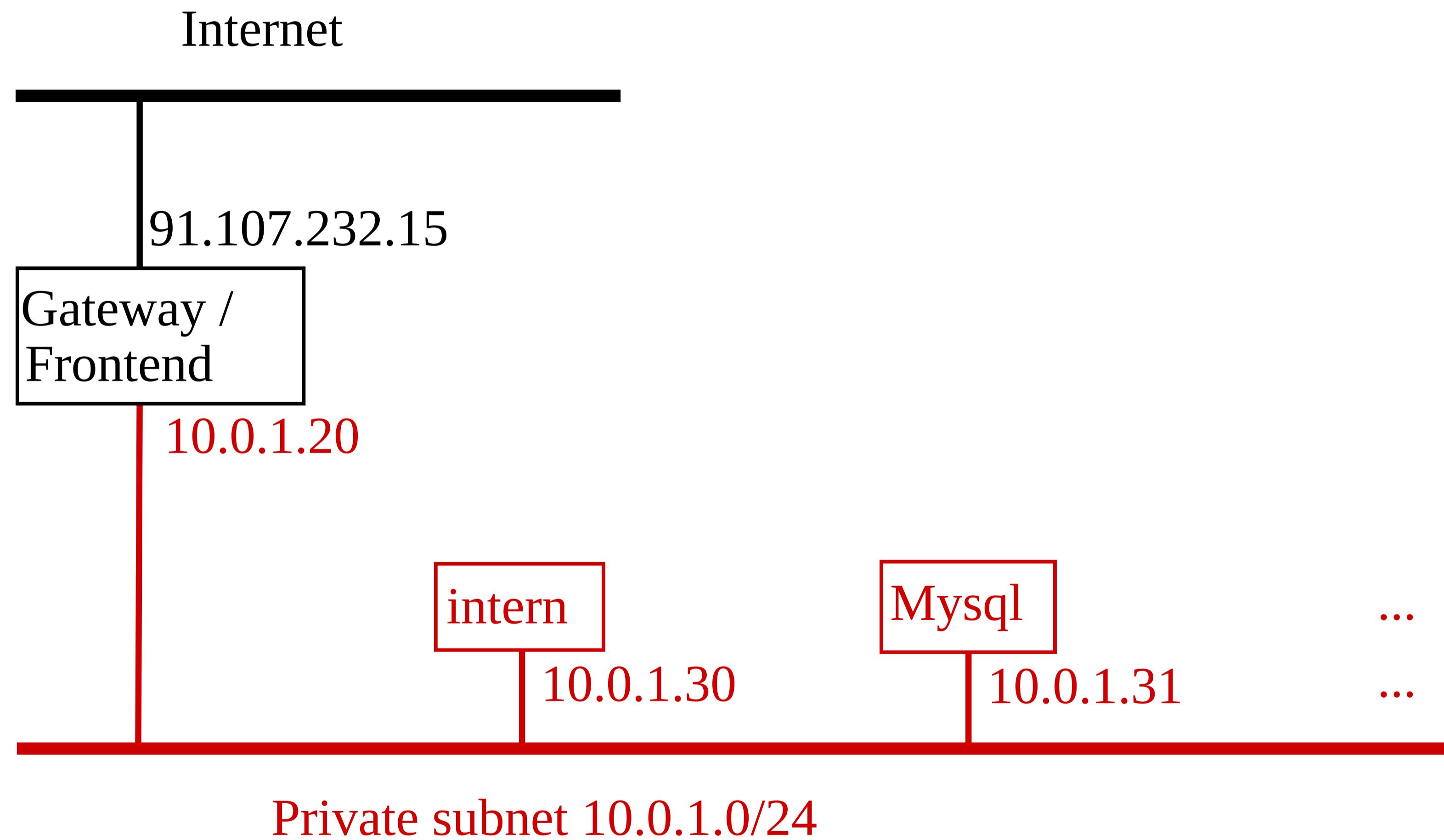
```
... updating zone 'goik.sdi.hdm-stuttgart.cloud/IN':  
  deleting rrset at '_acme-challenge.goik.sdi.hdm-stuttgart.cloud' TXT  
... updating zone 'goik.sdi.hdm-stuttgart.cloud/IN':  
  adding an RR at '_acme-challenge.goik.sdi.hdm-stuttgart.cloud' TXT  
    "GtJZJZjCZLWoGsQDODCFnY37TmMjRiy8Hw9M1eDGhkQ"  
... deleting rrset at ... TXT  
... adding an RR ... TXT "eJckWl2F43nsf27bzV0jcrTGp_VFeCj2qTVM5Uodg-4"  
... deleting an RR at _acme-challenge.goik.sdi.hdm-stuttgart.cloud TXT  
... updating zone ... deleting an RR ... TXT
```

## Followup exercises

- 343. [Creating a web certificate](#)
- 344. [Testing your web certificate](#)
- 345. [Combining certificate generation and server creation](#)

Cloud provider  
→ Private networks

# Private subnet overview



# Terraform network and subnetwork creation

```
resource "hcloud_network" "privateNet" {
    name      = "Private network"
    ip_range = "10.0.0.0/8"
}
resource "hcloud_network_subnet" "privateSubnet" {
    network_id    = hcloud_network.privateNet.id
    type          = "cloud"
    network_zone = "eu-central"
    ip_range     = "10.0.1.0/24"
}
resource "hcloud_network_route" "gateway"{
    network_id = hcloud_network.privateNet.id
    destination = "0.0.0.0/0"
    gateway = "10.0.1.20"
}
```

# Gateway: External and internal interface

```
resource "hcloud_server" "gateway" {
    ...
    public_net {
        ipv4 = hcloud_primary_ip.gatewayIp.id
    }
    network {
        network_id = hcloud_network.privateNet.id
        ip         = "10.0.1.20"
    }
}
```

# intern host

```
resource "hcloud_server" "intern" {
    ...
    public_net {
        ipv4_enabled = false
        ipv6_enabled = false
    }
    network {
        network_id = hcloud_network.privateNet.id
        ip          = "10.0.1.30"
    }
}
```

Followup exercise

### 346. Creating a subnet

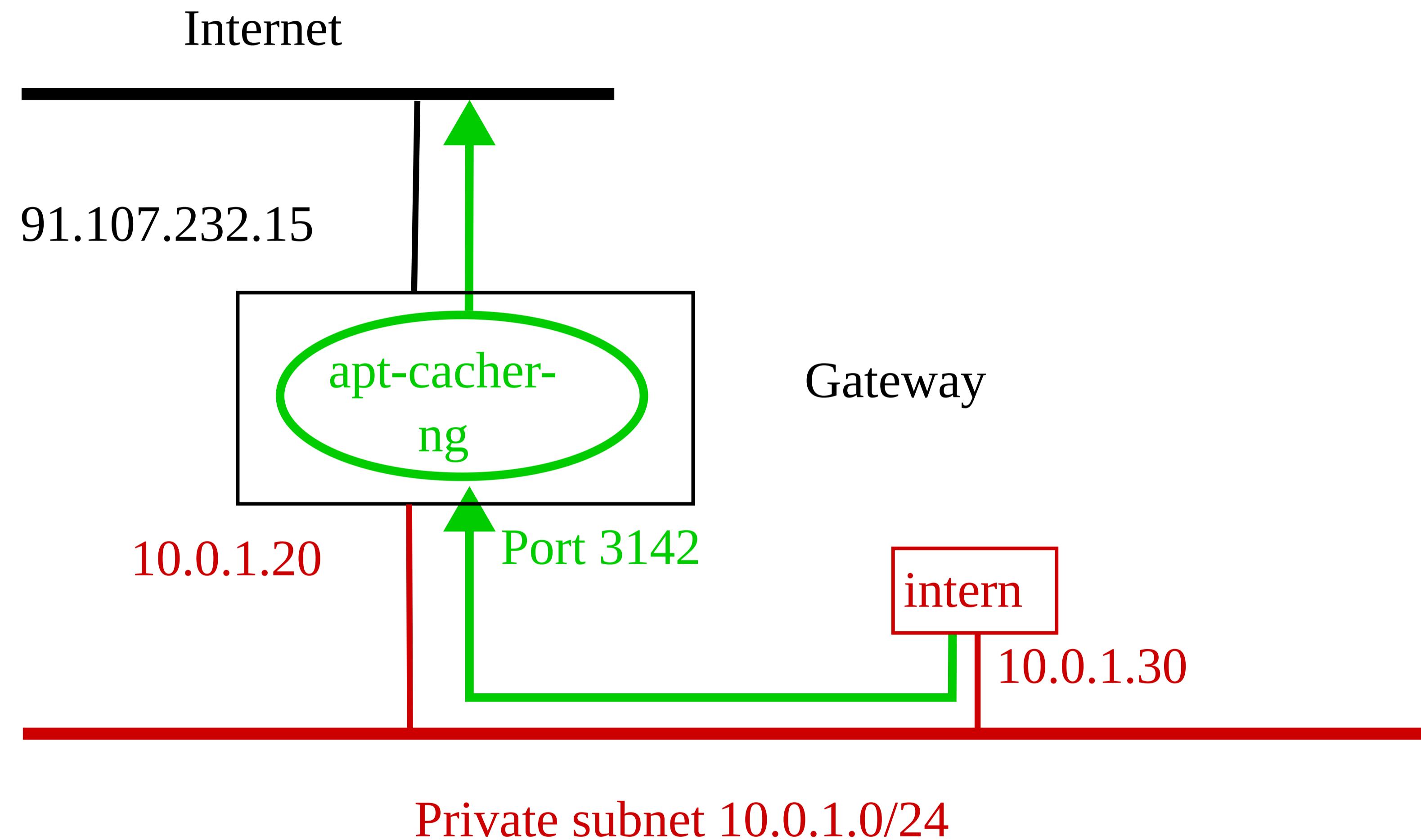
## Lack of internet access

- Host “intern” does not have Internet access.
- Consequences:
  1. No package updates.
  2. No package installs
  3. ...

# Possible solutions

1.
  - Allow IP forwarding on gateway host
  - Configure NAT enabling gateway host as router
2. Use an application level gateway:
  1. [apt-cacher-ng](#)
  2. [Squid](#)

# http proxy apt-cacher-ng



## Cloud-init problem

- Problem: `apt-cacher-ng` installation requires time for service to become available.
- Consequence: Package installs on host “**intern**” must be deferred.
- Problem: No standard Terraform “service ready” dependency management hook.

# Service ready query script

```
#!/bin/bash
echo "Waiting for apt-cacher-ng to launch on port 3142 ..."
while ! nc -z ${frontendPrivatenetIp} 3142;
do
  sleep 8 # wait for 8 second before polling again
  echo apt-cacher-ng not yet ready ...
done
echo "apt-cacher-ng service ready"
```

# Terraform "service ready" dependency hook

```
resource "null_resource" "waitForProxy" {
  connection {
    type     = "ssh"
    user     = "devops"
    host_key = ...public_key_openssh
    agent    = "true"
    host     = ...web.ipv4_address
  }
  provisioner "remote-exec" {
    inline=["/usr/bin/waitForAptProxy"]
  }
}
```

```
resource "hcloud_server" "intern" {
  ...
  depends_on = [
    hcloud_network_subnet.pSubnet
    ,null_resource.waitForProxy
  ]
}
```

Followup exercise

347. Adding an application level gateway