Personal S3 Stoarage

Jul 23, 2019

Table of Contents

For hosting a private S3 Object storage we use min.io at hetzner.cloud, created with Terraform and configured with Ansible.

Precondition

- For full Configuration you need the Base Scripts from nolte/ansible_playbook-baseline-online-server.
- The Terraform Infrastructuon part used Modules from nolte/terraform-infrastructure-modules.

CHAPTER 1

Usage

For Interaction with the Hetzner API, you must be define a environment variable with the name `HCLOUD_TOKEN`. This Variable will be used from the Terraform Hetzner Cloud Provider, and the hcloud Dynamic Ansible Inventory plugin.

```
export HCLOUD_TOKEN=$(pass internet/hetzner.com/projects/personal_storage/token)
```

For the Dependency Management it is recommedet to use a seperated virtual env like:

```
virtualenv -p python3 ~/venvs/ansible-vagrant
source ~/venvs/ansible-vagrant/bin/activate
pip install -r requirements.txt
pre-commit install
```

1.1 Infrastructure

The Terraform Source at the `./infrastructure` folder, is splitted into two different Steps. Firstly `./infrastructure/longterm_elements` for manage the Hetzner Project and the Storage Volume, so be carefull when you call `terraform destroy`, **you lost all your Stored Data!** The second part are located at `./infrastructure/minio_env`, here we attach the Storage volume and create the computing instance. `terraform destory` only delete the Computing Instance! The Storage Volume are not removed, so all your data are safe! Both parts used self written Terraform Modules from nolte/terraform-infrastructure-modules as wrapper for the Terraform hcloud provider.

1.2 Maintenance

For Installation and Maintenance, we use Ansible with a Dynamic Inventory. We splitted the production used inventory from the playbook Repository. For define the Inventory Location you can use a environment variable `export ANSIBLE_INVENTORY=$(pwd)/inventory/prod/`, or the `-i` parameter. At this Git Repository, you will

only find MinIO Specific Configuration steps. For the base configutation we use the nolte/ansible_playbook-baseline-online-server scripts, like base firewall configruations or install Docker.

For quick usage you can use the gilt - A GIT layering tool by:

```
gilt overlay
```

now you have all required dependencies at the `./ext_debs` working directory, and configure the basement with:

```
ansible-playbook ./ext_debs/ansible_playbook-baseline-online-server/master-configure-
↪system.yml
```

1.2.1 Storage Box Installation

```
ansible-playbook maintenance/master-configure-system.yml
```

1.3 Development

Future Read:

- how-to-secure-access-to-minio-server-with-tls
- minio-multi-user-quickstart-guide
- minio-client-quickstart-guide
- resizing-hetzner-cloud-block-storage-volumes-on-the-fly

1.3.1 Usermanagement

For Administration Tasks you can use the MinIO Admin Tool.

Listing 1: configure mc admin tool

```
export HCLOUD_TOKEN=$(pass internet/hetzner.com/projects/personal_storage/token) && \
export STORAGE_NODE_ENDPOINT=$(curl -s -H "Authorization: Bearer $HCLOUD_TOKEN"
↪'https://api.hetzner.cloud/v1/servers?name=storagenode' | jq -r '.servers[0].public_
↪net.ipv4.dns_ptr') && \
    mc config host add mystoragebox \
    https://$STORAGE_NODE_ENDPOINT \
    $(pass internet/project/mystoragebox/minio_access_key) \
    $(pass internet/project/mystoragebox/minio_secret_key)
```

Listing 2: check mc admin tool

```
mc admin info mystoragebox
```

Bucket Policy

The MinIO Bucket Policies ar AWS Compatible.

Listing 3: Simple Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::backup"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::backup/*"
        }
    ]
}
```

Additional Links:

- Bucket Policy
- AWS Bucket Doku
- AWS example bucket policies

Listing 4: create a policy

```
mc admin policy add mystoragebox backup_policy test.json
```

Access Keys

Listing 5: Create A User

```
new_user=backupuser \
    && pass generate -n internet/project/mystoragebox/users/${new_user}/minio_access_
↪key 25 \
    && pass generate internet/project/mystoragebox/users/${new_user}/minio_secret_key␣
↪45 \
    && mc admin user add mystoragebox \
        $(pass internet/project/mystoragebox/users/${new_user}/minio_access_key) \
        $(pass internet/project/mystoragebox/users/${new_user}/minio_secret_key) \
        backup_policy
```

Listing 6: Remove existing User

```
mc admin user remove mystoragebox $(pass internet/project/mystoragebox/users/${new_
↪user}/minio_access_key)
```

1.3.2 Integrate

A S3 Object Storage can be used for different Use Cases, like Archive Backups or share the Terraform State File.

Terraform State File

For a remote State file you can use the S3 backend Type.

Listing 7: Export Required Envs

```
export HCLOUD_TOKEN=$(pass internet/hetzner.com/projects/personal_storage/token) && \
  export AWS_ACCESS_KEY_ID=$(pass internet/project/mystoragebox/minio_access_key) && \
  export AWS_SECRET_ACCESS_KEY=$(pass internet/project/mystoragebox/minio_secret_key)␣
↪&& \
  export AWS_S3_ENDPOINT=https://$(curl -s -H "Authorization: Bearer $HCLOUD_TOKEN"
↪'https://api.hetzner.cloud/v1/servers?name=storagenode' | jq -r '.servers[0].public_
↪net.ipv4.dns_ptr')
```

Listing 8: Terraform State File Provider

```
terraform {
  backend "s3" {
    key                         = "minecraft/productuion/project"
    region                      = "main"
    bucket                      = "terraform-states"
    skip_requesting_account_id  = true
    skip_credentials_validation = true
    skip_get_ec2_platforms      = true
    skip_metadata_api_check     = true
    skip_region_validation      = true
    force_path_style            = true
  }
}
```

Restic Backups

For Backups with *restic* can use a s3 repository, so you have a central storage for restoring and archiving.

Listing 9: list existing snapshots

```
export HCLOUD_TOKEN=$(pass internet/hetzner.com/projects/personal_storage/token) && \
  export AWS_ACCESS_KEY_ID=$(pass internet/project/mystoragebox/minio_access_key) && \
  export AWS_SECRET_ACCESS_KEY=$(pass internet/project/mystoragebox/minio_secret_key)␣
↪&& \
  export RESTIC_PASSWORD=$(pass internet/project/minecraft/backups/restic_password) &&
↪ \
  export RESTIC_REPOSITORY=s3:https://$(curl -s -H "Authorization: Bearer $HCLOUD_
↪TOKEN" 'https://api.hetzner.cloud/v1/servers?name=storagenode' | jq -r '.servers[0].
↪public_net.ipv4.dns_ptr')/backup/minecraft-production/restic/gamedata && \
  restic snapshots
```

1.3.3 Glossary

Terraform With Terraform we Create the Infrastructure like Volumes, FloatingIP and Virtual Machines. For the Hetzner Intergration wie use the hcloud provider

Ansible Ansible is used for System configuration.

restic restic is a backup tool.

Vagrant Vagrant, is used for the local Environment.

logrotate Remove old, and rotate the logs with logrotate.

fail2ban Usig fail2ban for block brute force attacks.

Extra Packages for Enterprise Linux The EPEL repository is used for install extra packages like *restic*.

Open JDK Java JDK

pass The Commandline based passwordstore, can integrated to *Ansible* and *Terraform*,

pass ansible plugin Used for Secrets lookups passwordstore plugin

pass Terraform Provider For combinate *Terraform* and *pass* we use the custom provider camptocamp/terraform-provider-pass.

Ansile Master Playbooks importing-playbooks

Hetzner Cloud Hetzner Cloud

firewall hier wird der klassiker FirewallD verwendet.

Advanced Intrusion Detection Environment (aide) Store file see install-aide-centos-7. *(umsetzung offen)*

OpenSCAP System vulnerability scans, see (open-scap)

Sphinx Sphinx, is a tool that makes it easy to create documentation

reStructuredText reStructuredText Markdown alternative.

Molecule Molecule used for automatical Ansible Tests.

Testinfra Testinfra Testing infrastructure with Ansible and Pytest.

Virtualenv Virtualenv create isolated Python environments.

Index