Base Service Configuration

unknown

Mar 31, 2022

TABLE OF CONTENTS

CHAPTER

ONE

BASE SERVICE CONFIGURATION

This Repository will be handle the base configuration of Public Services, like Minecraft Server or Personal S3 Storage, a try dont get any Snowflake Server. Mostly hosted at hetzner.cloud, and created with Terraform.

1.1 Features

- harder `sshd`
  - configure `fail2ban`
- install `restic`
- base `logrotate` configuration
- install `python3`
- configure system `ntp` for time handling
- configure `docker` (optional)
- Security Scans
  - execute `open-scap-scan`
  - root kit analyse with `rkhunter`
  - configure `aide` (planed)

1.2 Supported Distributions

1.2.1 Out of Scope

- Provide any Infrastructure (see nolte/terraform-infrastructure-modules)
- Install any Services, like Minecraft (nolte/minecraft-infrastructure), or MinIO (nolte/personal-storage-infrastructure).

1.3 Usage

1.3.1 Prepare Python Env

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

1.3.2 Start SSH Agent

```
pass private/keyfiles/ssh/ansible_rollout/passphrase -c
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/ansible_id_ed25519
```

1.3.3 Playbooks

playbook	ansible_group	description
`master-configure-system.yml`		master playbook for combine the base and docker playbook.
`playbook-base-configureation.yml`	*all*	Configure the base system.
`playbook-docker-configureation.yml`	*dockerbased*	Prepare the System for docker Usage

```
export ANSIBLE_INVENTORY=$(pwd)/prod
export HCLOUD_TOKEN=$(pass .../token)

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

1.4 Development

For Locally automatical testing we use a Combination of Molecule and Vagrant

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

1.4.1 Testing

running the tests:

```
molecule test
```

Infrastructure Tests

```
pytest --connection=ansible --hosts=all test/*
```

1.4.2 Reusing and Sharing

For reusing and sharing you can create own vagrant box with the Packer `/packer`.

First Build the CentOS Base (WIP)

1.4.3 Releasing

Must be executed from the `develop` branch.

```
pre-commit uninstall \
    && bump2version --tag release --commit \
    && git checkout master && git merge develop && git checkout develop \
    && bump2version --no-tag patch --commit \
    && git push origin master --tags \
    && git push origin develop \
    && pre-commit install
```

1.4.4 Setup Local Env

```
asdf plugin-add packer https://github.com/Banno/asdf-hashicorp.git
asdf plugin-add terraform https://github.com/Banno/asdf-hashicorp.git
asdf plugin-add python
asdf plugin-test act https://github.com/grimoh/asdf-act --asdf-tool-version latest
```

```
python -m venv env
source env/bin/activate
pip install -r requirements.txt
```

CHAPTER

TWO

FEATURE DETAILS

2.1 Fail2Ban

fail2ban is a good way to keep Brute force attack's away from our System.
The Base configuration will be done from the robertdebock.fail2ban Ansible Role. For Client required configuration take a look at *Client Side SSH Configration*.

2.1.1 Usefull Commands

Show Current Jails

Listing 1: list all jails

```
sudo su
fail2ban-client status | sed -n 's/,//g;s/.*Jail list://p' | xargs -n1 fail2ban-client␣
↪status
```

```
Status for the jail: nginx-req-limit
|- Filter
|  |- Currently failed:     0
|  |- Total failed: 0
|  `- File list:    /var/log/nginx/error.log
`- Actions
   |- Currently banned:     0
   |- Total banned: 0
   `- Banned IP list:
Status for the jail: sshd
|- Filter
|  |- Currently failed:     16
|  |- Total failed: 108
|  `- Journal matches:      _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned:     0
   |- Total banned: 3
   `- Banned IP list:
```

2.1.2 Additional Links

some untested Prometheus Exporters

- jangrewe/prometheus-fail2ban-exporter
- Kylapaallikko/fail2ban_exporter

2.2 Harder SSHd

You must set your used `private_key_file` and some SSH Extra args `-o IdentitiesOnly=yes`, otherwise the *fail2ban* process will block your IP!.

2.2.1 Client Side SSH Configration

One of the important config changes is `StrictModes yes`, this required some changes at the Client Side.

Listing 2: example ~/.ansible.cfg

```
[defaults]
validate_certs=False
private_key_file=~/.ssh/ansible_id_ed25519

[ssh_connection]
ssh_extra_args=-o IdentitiesOnly=yes
```

Listing 3: example ~/.ssh/config

```
Host *
  IdentitiesOnly yes
  IdentityFile ~/.ssh/ansible_id_ed25519
```

2.3 Security Checks

At the moment, execute the security checks is a Manual Step. The Future Plan is, to execute and report the scans by a Scheduler System.

2.3.1 open-scap-scan

The *OpenSCAP* Project make it easy to scan your Systems of existing Security vulnerable or configuration missmatch.

Listing 4: execute checks

```
ansible-playbook playbook-execute-security-open-scap-scan.yml
```

The Generated Report will be stored at `./generated_reports/oscap-reports/{{ inventory_hostname }}-{{ oscap_policy }}.html`

2.3.2 Root Kit Analyse

For Root kit analyse we use the *rkhunter* Tool.

Listing 5: execute root kit checks

```
ansible-playbook playbook-execute-security-rootkit-scan.yml
```

The Generated Report will be stored at `./generated_reports/rootkitscan/{{ inventory_hostname }}.txt`

CHAPTER

THREE

GLOSSARY

Terraform With Terraform we Create the Infrastructure like Volumes, FloatingIP and Virtual Machines. for the Hetzner Intergration wie use the ref-env-provider-hetzner-integration-terraform

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 (Implementation details at: *Fail2Ban*).

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

Open 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.

rkhunter rkhunter hunter for Rootkits.

chkrootkit chkrootkit locally checks for signs of a rootkit *(planned)*.

CHAPTER

FOUR

EXTERNAL DEPENDENCIES

For Configure the Base we use a set of external ansible roles, hosted at Ansible Galaxy and listed in `requirements.yml`

Your can use the `ansible-galaxy` command for installation,

```
ansible-galaxy install -r requirements.yml
```

INDEX