Kostenloses eBook

LERNEN ansible

Free unaffiliated eBook created from
Stack Overflow contributors.

#ansible

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ansible

It is an unofficial and free ansible ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ansible.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Ansible

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick darüber, was erreichbar ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen in Ansible erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für ansible neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

Hallo Welt

Erstellen Sie ein Verzeichnis namens `ansible-helloworld-playbook`

```
mkdir ansible-helloworld-playbook
```

Erstellen Sie einen Datei- `hosts` und fügen Sie Remote-Systeme hinzu, die verwaltet werden sollen. Da sich ansible zum Verbinden der Maschinen auf ssh verlässt, sollten Sie sicherstellen, dass diese bereits von Ihrem Computer aus in ssh verfügbar sind.

```
192.168.1.1
192.168.1.2
```

Testen Sie die Verbindung zu Ihren Remote-Systemen mit dem Ansible- Ping- Modul.

```
ansible all -m ping -k
```

Im Erfolgsfall sollte so etwas zurückgegeben werden

```
192.168.1.1| SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.1.2| SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Im Fehlerfall sollte es zurückkehren

```
192.168.1.1| UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh.",
    "unreachable": true
```

```
}
```

Testen Sie den Sudo-Zugriff mit

```
ansible all -m ping -k -b
```

Testen Sie die Verbindung und Konfiguration mit Ping

```
ansible -i hosts -m ping targethost
```

`-i hosts` definieren den Pfad zur Inventardatei
`targethost` ist der Name des Hosts in der `hosts` Datei

Inventar

Inventory ist die Ansible-Methode, um alle Systeme in Ihrer Infrastruktur zu verfolgen. Hier ist eine einfache statische Inventardatei, die ein einzelnes System und die Anmeldeinformationen für Ansible enthält.

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ansible_ssh_pass=PassW0rd
```

Schreiben Sie diese Zeilen beispielsweise in `hosts` Datei und übergeben Sie die Datei mit dem Befehl `-i` / `--inventory-file` Befehl `ansible` oder `ansible-playbook --inventory-file` .

Weitere Informationen finden Sie unter Statisches Inventar und Dynamisches Inventar .

Remote-Maschinen mit Ansible bereitstellen

Wir können Remote-Systeme mit Ansible bereitstellen. Sie sollten über ein SSH-Schlüsselpaar verfügen und Ihren öffentlichen SSH-Schlüssel in die Datei ~ / .ssh / authorized_keys des Rechners bringen. Sie können sich ohne Berechtigung einloggen.

Voraussetzungen:

- Ansible

Sie benötigen eine Inventardatei (z. B .: development.ini), in der Sie den Host bestimmen, was Sie verwenden möchten:

```
[MACHINE_NAME]
MACHINE_NAME hostname=MACHINE_NAME ansible_ssh_host=IP_ADDRESS ansible_port=SSH_PORT
ansible_connection=ssh ansible_user=USER ansible_ssh_extra_args="-o StrictHostKeyChecking=no -
o UserKnownHostsFile=/dev/null"
```

- Hostname - Der Hostname der Remote-Maschine
- ansible_ssh_host - die IP-Adresse oder Domäne des Remote-Hosts
- ansible_port - der Port des Remote-Hosts, der normalerweise 22 ist

- ansible_connection - die Verbindung, bei der wir eine Verbindung herstellen möchten, die wir mit ssh verbinden möchten
- ansible_user - der ssh-Benutzer
- ansible_ssh_extra_args - zusätzliche Argumente, was Sie für die SSH-Verbindung angeben möchten

Erforderliche zusätzliche Argumente für ssh:

- StrictHostKeyChecking - Es kann ein Schlüssel abgefragt werden, um zu prüfen, was auf ein Ja oder Nein wartet. Das Ansible kann diese Frage nicht beantworten und gibt dann einen Fehler aus, der Host ist nicht verfügbar.
- UserKnownHostsFile - Wird für die StrictHostKeyChecking-Option benötigt.

Wenn Sie diese Inventardatei haben, können Sie ein Test-Playbook.yml schreiben:

```
---
- hosts: MACHINE_NAME
  tasks:
    - name: Say hello
      debug:
        msg: 'Hello, World'
```

dann können Sie mit der Bereitstellung beginnen:

```
ansible-playbook -i development.ini playbook.yml
```

ansible.cfg

Dies ist die Standardeinstellung ansible.cfg von Ansible github .

```
# config file for ansible -- http://ansible.com/
# ==============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#remote_tmp     = $HOME/.ansible/tmp
#local_tmp      = $HOME/.ansible/tmp
#forks          = 5
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
#transport      = smart
#remote_port    = 22
```

```
#module_lang    = C
#module_set_locale = False

# plays will gather facts by default, which contain information about
# the remote system.
#
# smart - gather by default, but don't regather if already gathered
# implicit - gather by default, turn off with gather_facts: False
# explicit - do not gather by default, must say gather_facts: True
#gathering = implicit

# by default retrieve all facts subsets
# all - gather all subsets
# network - gather min and network facts
# hardware - gather hardware facts (longest facts to retrieve)
# virtual - gather min and virtual facts
# facter - import facts from facter
# ohai - import facts from ohai
# You can combine them using comma (ex: network,virtual)
# You can negate them using ! (ex: !hardware,!facter,!ohai)
# A minimal set of facts is always gathered.
#gather_subset = all

# some hardware related facts are collected
# with a maximum timeout of 10 seconds. This
# option lets you increase or decrease that
# timeout to something more suitable for the
# environment.
# gather_timeout = 10

# additional paths to search for roles in, colon separated
#roles_path    = /etc/ansible/roles

# uncomment this to disable SSH key host checking
#host_key_checking = False

# change the default callback
#stdout_callback = skippy
# enable additional callbacks
#callback_whitelist = timer, mail

# Determine whether includes in tasks and handlers are "static" by
# default. As of 2.0, includes are dynamic by default. Setting these
# values to True will make includes behave more like they did in the
# 1.x versions.
#task_includes_static = True
#handler_includes_static = True

# change this for alternative sudo implementations
#sudo_exe = sudo

# What flags to pass to sudo
# WARNING: leaving out the defaults might create unexpected behaviours
#sudo_flags = -H -S -n

# SSH timeout
#timeout = 10

# default user to use for playbooks if user is not specified
# (/usr/bin/ansible will use current user as default)
#remote_user = root
```

```
# logging is off by default unless this path is defined
# if so defined, consider logrotate
#log_path = /var/log/ansible.log

# default module name for /usr/bin/ansible
#module_name = command

# use this shell for commands executed under sudo
# you may need to change this to bin/bash in rare instances
# if sudo is constrained
#executable = /bin/sh

# if inventory variables overlap, does the higher precedence one win
# or are hash values merged together?  The default is 'replace' but
# this can also be set to 'merge'.
#hash_behaviour = replace

# by default, variables from roles will be visible in the global variable
# scope. To prevent this, the following option can be enabled, and only
# tasks and handlers within the role will see the variables there
#private_role_vars = yes

# list any Jinja2 extensions to enable here:
#jinja2_extensions = jinja2.ext.do,jinja2.ext.i18n

# if set, always use this private key file for authentication, same as
# if passing --private-key to ansible or ansible-playbook
#private_key_file = /path/to/file

# If set, configures the path to the Vault password file as an alternative to
# specifying --vault-password-file on the command line.
#vault_password_file = /path/to/vault_password_file

# format of string {{ ansible_managed }} available within Jinja2
# templates indicates to users editing templates files will be replaced.
# replacing {file}, {host} and {uid} and strftime codes with proper values.
#ansible_managed = Ansible managed: {file} modified on %Y-%m-%d %H:%M:%S by {uid} on {host}
# This short version is better used in templates as it won't flag the file as changed every
run.
#ansible_managed = Ansible managed: {file} on {host}

# by default, ansible-playbook will display "Skipping [host]" if it determines a task
# should not be run on a host.  Set this to "False" if you don't want to see these "Skipping"
# messages. NOTE: the task header will still be shown regardless of whether or not the
# task is skipped.
#display_skipped_hosts = True

# by default, if a task in a playbook does not include a name: field then
# ansible-playbook will construct a header that includes the task's action but
# not the task's args.  This is a security feature because ansible cannot know
# if the *module* considers an argument to be no_log at the time that the
# header is printed.  If your environment doesn't have a problem securing
# stdout from ansible-playbook (or you have manually specified no_log in your
# playbook on all of the tasks where you have secret information) then you can
# safely set this to True to get more informative messages.
#display_args_to_stdout = False

# by default (as of 1.3), Ansible will raise errors when attempting to dereference
# Jinja2 variables that are not set in templates or action lines. Uncomment this line
# to revert the behavior to pre-1.3.
```

```
#error_on_undefined_vars = False

# by default (as of 1.6), Ansible may display warnings based on the configuration of the
# system running ansible itself. This may include warnings about 3rd party packages or
# other conditions that should be resolved if possible.
# to disable these warnings, set the following value to False:
#system_warnings = True

# by default (as of 1.4), Ansible may display deprecation warnings for language
# features that should no longer be used and will be removed in future versions.
# to disable these warnings, set the following value to False:
#deprecation_warnings = True

# (as of 1.8), Ansible can optionally warn when usage of the shell and
# command module appear to be simplified by using a default Ansible module
# instead.  These warnings can be silenced by adjusting the following
# setting or adding warn=yes or warn=no to the end of the command line
# parameter string.  This will for example suggest using the git module
# instead of shelling out to the git command.
# command_warnings = False


# set plugin path directories here, separate with colons
#action_plugins     = /usr/share/ansible/plugins/action
#cache_plugins      = /usr/share/ansible/plugins/cache
#callback_plugins   = /usr/share/ansible/plugins/callback
#connection_plugins = /usr/share/ansible/plugins/connection
#lookup_plugins     = /usr/share/ansible/plugins/lookup
#inventory_plugins  = /usr/share/ansible/plugins/inventory
#vars_plugins       = /usr/share/ansible/plugins/vars
#filter_plugins     = /usr/share/ansible/plugins/filter
#test_plugins       = /usr/share/ansible/plugins/test
#strategy_plugins   = /usr/share/ansible/plugins/strategy

# by default callbacks are not loaded for /bin/ansible, enable this if you
# want, for example, a notification or logging callback to also apply to
# /bin/ansible runs
#bin_ansible_callbacks = False


# don't like cows?  that's unfortunate.
# set to 1 if you don't want cowsay support or export ANSIBLE_NOCOWS=1
#nocows = 1

# set which cowsay stencil you'd like to use by default. When set to 'random',
# a random stencil will be selected for each task. The selection will be filtered
# against the `cow_whitelist` option below.
#cow_selection = default
#cow_selection = random

# when using the 'random' option for cowsay, stencils will be restricted to this list.
# it should be formatted as a comma-separated list with no spaces between names.
# NOTE: line continuations here are for formatting purposes only, as the INI parser
#       in python does not support them.
#cow_whitelist=bud-frogs,bunny,cheese,daemon,default,dragon,elephant-in-snake,elephant,eyes,\
#              hellokitty,kitty,luke-
koala,meow,milk,moofasa,moose,ren,sheep,small,stegosaurus,\
#              stimpy,supermilker,three-eyes,turkey,turtle,tux,udder,vader-koala,vader,www

# don't like colors either?
# set to 1 if you don't want colors, or export ANSIBLE_NOCOLOR=1
```

```
#nocolor = 1

# if set to a persistent type (not 'memory', for example 'redis') fact values
# from previous runs in Ansible will be stored.  This may be useful when
# wanting to use, for example, IP information from one group of servers
# without having to talk to them in the same playbook run to get their
# current IP information.
#fact_caching = memory


# retry files
# When a playbook fails by default a .retry file will be created in ~/
# You can disable this feature by setting retry_files_enabled to False
# and you can change the location of the files by setting retry_files_save_path

#retry_files_enabled = False
#retry_files_save_path = ~/.ansible-retry

# squash actions
# Ansible can optimise actions that call modules with list parameters
# when looping. Instead of calling the module once per with_ item, the
# module is called once with all items at once. Currently this only works
# under limited circumstances, and only with parameters named 'name'.
#squash_actions = apk,apt,dnf,package,pacman,pkgng,yum,zypper

# prevents logging of task data, off by default
#no_log = False

# prevents logging of tasks, but only on the targets, data is still logged on the
master/controller
#no_target_syslog = False

# controls whether Ansible will raise an error or warning if a task has no
# choice but to create world readable temporary files to execute a module on
# the remote machine.  This option is False by default for security.  Users may
# turn this on to have behaviour more like Ansible prior to 2.1.x.  See
# https://docs.ansible.com/ansible/become.html#becoming-an-unprivileged-user
# for more secure ways to fix this than enabling this option.
#allow_world_readable_tmpfiles = False

# controls the compression level of variables sent to
# worker processes. At the default of 0, no compression
# is used. This value must be an integer from 0 to 9.
#var_compression_level = 9

# controls what compression method is used for new-style ansible modules when
# they are sent to the remote system.  The compression types depend on having
# support compiled into both the controller's python and the client's python.
# The names should match with the python Zipfile compression types:
# * ZIP_STORED (no compression. available everywhere)
# * ZIP_DEFLATED (uses zlib, the default)
# These values may be set per host via the ansible_module_compression inventory
# variable
#module_compression = 'ZIP_DEFLATED'

# This controls the cutoff point (in bytes) on --diff for files
# set to 0 for unlimited (RAM may suffer!).
#max_diff_size = 1048576

[privilege_escalation]
#become=True
```

```
#become_method=sudo
#become_user=root
#become_ask_pass=False

[paramiko_connection]

# uncomment this line to cause the paramiko connection plugin to not record new host
# keys encountered.  Increases performance on new host additions.  Setting works independently
of the
# host key checking setting above.
#record_host_keys=False

# by default, Ansible requests a pseudo-terminal for commands executed under sudo. Uncomment
this
# line to disable this behaviour.
#pty=False

[ssh_connection]

# ssh arguments to use
# Leaving off ControlPersist will result in poor performance, so use
# paramiko on older platforms rather than removing it, -C controls compression use
#ssh_args = -C -o ControlMaster=auto -o ControlPersist=60s

# The path to use for the ControlPath sockets. This defaults to
# "%(directory)s/ansible-ssh-%%h-%%p-%%r", however on some systems with
# very long hostnames or very long path names (caused by long user names or
# deeply nested home directories) this can exceed the character limit on
# file socket names (108 characters for most platforms). In that case, you
# may wish to shorten the string below.
#
# Example:
# control_path = %(directory)s/%%h-%%r
#control_path = %(directory)s/ansible-ssh-%%h-%%p-%%r

# Enabling pipelining reduces the number of SSH operations required to
# execute a module on the remote server. This can result in a significant
# performance improvement when enabled, however when using "sudo:" you must
# first disable 'requiretty' in /etc/sudoers
#
# By default, this option is disabled to preserve compatibility with
# sudoers configurations that have requiretty (the default on many distros).
#
#pipelining = False

# if True, make ansible use scp if the connection type is ssh
# (default is sftp)
#scp_if_ssh = True

# if False, sftp will not use batch mode to transfer files. This may cause some
# types of file transfer failures impossible to catch however, and should
# only be disabled if your sftp version has problems with batch mode
#sftp_batch_mode = False

[accelerate]
#accelerate_port = 5099
#accelerate_timeout = 30
#accelerate_connect_timeout = 5.0

# The daemon timeout is measured in minutes. This time is measured
# from the last activity to the accelerate daemon.
```

```
#accelerate_daemon_timeout = 30

# If set to yes, accelerate_multi_key will allow multiple
# private keys to be uploaded to it, though each user must
# have access to the system via SSH to add a new key. The default
# is "no".
#accelerate_multi_key = yes

[selinux]
# file systems that require special treatment when dealing with security context
# the default behaviour that copies the existing context or uses the user default
# needs to be changed to use the file system dependent context.
#special_context_filesystems=nfs,vboxsf,fuse,ramfs

# Set this to yes to allow libvirt_lxc connections to work without SELinux.
#libvirt_lxc_noseclabel = yes

[colors]
#highlight = white
#verbose = blue
#warn = bright purple
#error = red
#debug = dark gray
#deprecate = purple
#skip = cyan
#unreachable = red
#ok = green
#changed = yellow
#diff_add = green
#diff_remove = red
#diff_lines = cyan
```

Fügen Sie diese Konfiguration in das Stammverzeichnis Ihrer Rollenverzeichnisse ein, um das Verhalten von Ansible bei Verwendung dieser Rolle zu ändern. Sie können beispielsweise festlegen, dass das Erstellen von `playbook.retry` bei fehlgeschlagenen Playbook-Läufen gestoppt wird oder auf geheime Vars verweist, die Sie in Ihrem Git-Repo nicht möchten.

Erste Schritte mit Ansible online lesen: https://riptutorial.com/de/ansible/topic/826/erste-schritte-mit-ansible

Kapitel 2: Ansible Group Vars

Examples

Beispiel group_vars / development und warum

Projektstruktur

```
project/
  group_vars/
     development
  inventory.development
  playbook.yaml
```

Diese Variablen werden aufgrund des Dateinamens auf Hosts in der Entwicklungsgruppe angewendet.

```
---
## Application
app_name: app
app_url: app.io
web_url: cdn.io
app_friendly: New App
env_type: production
app_debug: false

## SSL
ssl: true
ev_ssl: false

## Database
database_host: 127.0.0.1
database_name: app
database_user: sql

## Elasticsearch
elasticsearch_host: 127.0.0.1
```

Ansible Group Vars online lesen: https://riptutorial.com/de/ansible/topic/6226/ansible-group-vars

Kapitel 3: Ansible Gruppenvariablen

Examples

Gruppieren Sie Variablen mit statischem Inventar

Es wird empfohlen, dass Sie Gruppen basierend auf dem Zweck des Hosts (Rollen) sowie der Geografie oder des Datencenters (falls zutreffend) definieren:

Aktenbestand `inventory/production`

```
[rogue-server]
192.168.1.1

[atlanta-webservers]
www-atl-1.example.com
www-atl-2.example.com

[boston-webservers]
www-bos-1.example.com
www-bos-2.example.com

[atlanta-dbservers]
db-atl-1.example.com
db-atl-2.example.com

[boston-dbservers]
db-bos-1.example.com

# webservers in all geos
[webservers:children]
atlanta-webservers
boston-webservers

# dbservers in all geos
[dbservers:children]
atlanta-dbservers
boston-dbservers

# everything in the atlanta geo
[atlanta:children]
atlanta-webservers
atlanta-dbservers

# everything in the boston geo
[boston:children]
boston-webservers
boston-dbservers
```

Datei `group_vars/all`

```
---
apache_port: 80
```

Datei `group_vars/atlanta-webservers`

```
---
apache_port: 1080
```

Datei `group_vars/boston-webservers`

```
---
apache_port: 8080
```

Datei `host_vars/www-bos-2.example.com`

```
---
apache_port: 8111
```

Nach dem Ausführen von `ansible-playbook -i inventory/hosts install-apache.yml` (Hosts im Playbook sind `hosts: all`)

Die Ports wären

Adresse	Hafen
`192.168.1.1`	80
`www-atl-1.example.com`	1080
`www-atl-2.example.com`	1080
`www-bos-1.example.com`	8080
`www-bos-2.example.com`	8111

Ansible Gruppenvariablen online lesen: https://riptutorial.com/de/ansible/topic/6544/ansible-gruppenvariablen

Kapitel 4: Ansible installieren MySQL

Einführung

Wie kann man die MySQL-Binärdatei installieren?

Examples

Wie kann man die MySQL-Binärdatei installieren?

- hosts: MySQL-Aufgaben:
    - Name: Mysql-Benutzer hinzufügen Benutzer: Name: Mysql-Shell: / sbin / nologin
    - name: installiere die neueste Version von libselinux-python yum: name: libselinux-python state: latest
    - name: install perl yum: name: perl zustand: aktuell
    - name: Entfernen Sie das Paket mysql-libs yum: name: mysql-libs state: nicht vorhanden

```
- name: download and unarchive tar
  unarchive:
     src=/tmp/mysql-5.6.35-linux-glibc2.5-x86_64.tar.gz
     dest=/tmp
     copy=yes


- name: Move mysql paceage to specified directory
  command: creates="/usr/local/mysql" mv /tmp/mysql-5.6.35-linux-glibc2.5-x86_64
/usr/local/mysql

- name: chown mysql mysql /usr/local/mysql
  file: path=/usr/local/mysql owner=mysql group=mysql recurse=yes


- name: Add lib to ld.so.conf
  lineinfile: dest=/etc/ld.so.conf line="/usr/local/mysql/lib/"

- name: ldconfig
  command: /sbin/ldconfig

- name: Mkdir mysql_data_dir
  file: path=/data/mysql/3306/{{ item }} state=directory owner=mysql group=mysql
  with_items:
    - data
    - logs
    - tmp

- name: Copy mysql my.cnf
```

```
  copy: src=/etc/my.cnf dest=/etc/my.cnf

- name: Copy mysql my.cnf
  copy: src=/etc/my.cnf dest=/usr/local/mysql/my.cnf


- name: Init mysql db
  command: /usr/local/mysql/scripts/mysql_install_db  \
    --user=mysql  \
    --basedir=/usr/local/mysql \
    --datadir=/data/mysql/3306/data

- name: Add mysql bin to profile
  lineinfile: dest=/etc/profile line="export PATH=$PATH:/usr/local/mysql/bin/"

- name: Source profile
  shell: executable=/bin/bash source /etc/profile

- name: Copy mysqld to init when system start
  command: cp -f /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld

- name: Add mysqld to system start
  command: /sbin/chkconfig --add mysqld

- name: Add mysql to system start when init 345
  command: /sbin/chkconfig --level 345 mysqld on

- name: Retart mysql
  service: name=mysqld state=restarted
```

Ansible installieren MySQL online lesen: https://riptutorial.com/de/ansible/topic/10920/ansible-installieren-mysql

Kapitel 5: Ansible mit OpenStack verwenden

Einführung

OpenStack ist eine Open-Source-Softwareplattform für Cloud Computing. Linux-Instanzen können manuell über die grafische Weboberfläche gestartet / gestoppt oder mithilfe des OpenStack-Cloud-Moduls von ansible automatisiert werden.

Das Konfigurieren von Ansible kann schwierig sein, aber wenn es gut konfiguriert ist, ist es für Tests und Continuous Integration-Umgebungen wirklich einfach und leistungsstark.

Parameter

Parameter	Bemerkungen
Gastgeber: localhost	OpenStack-Befehle werden von unserem localhost aus gestartet
gather_facts: Falsch	Wir brauchen keine Informationen über unseren localhost zu sammeln
auth_url: https://openstack-identity.mycompany.com/v2.0	Verwenden Sie V2.0 URL
Zustand: Gegenwart	'present' / 'abwesend' zum Erstellen / Löschen der Instanz
validate_certs: Falsch	nützlich, wenn https selbstsignierte Zertifikate verwendet
Netzwerk: "{{Netzwerkname}}"	(wahlweise)
auto_ip: ja	(wahlweise)

Bemerkungen

- Wir legen die Authentifizierungs-URL direkt in das Playbook und nicht in eine Variable. In in vars verwendete URL muss mit Escapezeichen versehen werden.
- Seien Sie vorsichtig mit der Authentifizierungs-URL-Version V2.0 anstelle von V3 in https://openstack-identity.mycompany.com/v2.0 .
- Seien Sie in Yml-Dateien sehr vorsichtig, wenn Sie vom Browser kopieren / einfügen. Überprüfen Sie die Leerzeichen doppelt, wenn sie berücksichtigt werden.
- Weitere Informationen finden Sie unter: http://docs.ansible.com/ansible/list_of_cloud_modules.html#openstack

Examples

Überprüfen Sie Ihre Ansible-Version

Überprüfen Sie, ob die richtigen Softwareversionen installiert sind:

- Ansible> = 2,0
- Python> = 2,6
- Farbmodul für Python

```
$ansible --version
ansible 2.2.0.0

$python --version
Python 2.7.5
```

Installieren Sie die Python-Komponente, mit der Openstack gesteuert wird.

```
$pip install shade
```

Hinweis: Wenn Sie einen Firmen-Proxy verwenden, ist es immer nützlich, den richtigen Pip-Syntax zu kennen

```
$pip install --proxy proxy_ip:proxy_port shade
```

Sammeln Sie Informationen aus der OpenStack-GUI, um Ansible zu konfigurieren

---

Unser Openstack-Mieter ist bereits eingestellt:

- Ein virtuelles LAN gibt Instanzen private IP
- Ein virtueller Router ordnet die öffentliche IP der privaten IP zu
- Ein Sicherheitsschlüssel wurde generiert
- Wir haben eine Standard-Firewall-Konfiguration für SSH und Port 80
- Dank der OpenStack-Weboberfläche können wir eine Instanz starten

Lassen Sie sich alle erforderlichen Informationen über dieses Webinterface sammeln.

Informationen zur Authentifizierung finden Sie in der Datei openstack.rc. Diese Datei kann über das OpenStack-Webinterface in [Zugriff und Sicherheit / API-Zugriff] heruntergeladen werden.

```
$cat openstack.rc
#!/bin/bash

# To use an OpenStack cloud you need to authenticate against the Identity
# service named keystone, which returns a **Token** and **Service Catalog**.
# The catalog contains the endpoints for all services the user/tenant has
# access to - such as Compute, Image Service, Identity, Object Storage, Block
```

```
# Storage, and Networking (code-named nova, glance, keystone, swift,
# cinder, and neutron).
#
# *NOTE*: Using the 2.0 *Identity API* does not necessarily mean any other
# OpenStack API is version 2.0. For example, your cloud provider may implement
# Image API v1.1, Block Storage API v2, and Compute API v2.0. OS_AUTH_URL is
# only for the Identity API served through keystone.
export OS_AUTH_URL=https://openstack-identity.mycompany.com/v3

# With the addition of Keystone we have standardized on the term **tenant**
# as the entity that owns the resources.
export OS_TENANT_ID=1ac99fef77ee40148d7d5ba3e070caae
export OS_TENANT_NAME="TrainingIC"
export OS_PROJECT_NAME="TrainingIC"

# In addition to the owning entity (tenant), OpenStack stores the entity
# performing the action as the **user**.
export OS_USERNAME="UserTrainingIC"

# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT

# If your configuration has multiple regions, we set that information here.
# OS_REGION_NAME is optional and only valid in certain environments.
export OS_REGION_NAME="fr"
# Don't leave a blank variable, unset it if it was empty
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
```

Wir erhalten OS_AUTH_URL, OS_TENANT_NAME, OS_USERNAME.

Version der Authentifizierungs-API: OS_AUTH_URL

Vorsicht vor Authentifizierungs-API-Version. Standardmäßig ist Version 3 aktiviert, aber für Ansible ist Version 2.0 erforderlich. Wir erhalten die URL und setzen V2.0 anstelle von V3: https://openstack-identity.mycompany.com/v2.0

VM-Informationen

Erstellen Sie mithilfe der OpenStack-Weboberfläche eine Instanz, und rufen Sie den Namen für Image, Flavour, Schlüssel, Netzwerk und Sicherheitsgruppe ab.

Erstellen Sie eine ./group_vars/all -Datei mit allen erforderlichen Informationen.

```
$vi ./group_vars/all
# Authentication
AuthUserName: UserTrainingIC
AuthPassword: PasswordTrainingIC
TenantName: TrainingIC

# VM infos
ImageName: CentOS-7-x86_64-GenericCloud-1607
FlavorName: m1.1cpu.1gb
InfraKey: KeyTrainingIC
NetworkName: NetPrivateTrainingIC
SecurityGroup: default
```

Schreiben Sie das ansible Playbook, um die Instanz zu erstellen

Verwenden Sie den Befehl 'os_server' vom Modul 'Cloud' [http://docs.ansible.com/ansible/os_server_module.html] . Variablen sind in ./group_vars/all definiert.

```
$vi launch_compute.yml
- name: launch a compute instance
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Create and launch the VM
    os_server:
      auth:
        auth_url: https://openstack-identity.mycompany.com/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ TenantName }}"
      state: present
      validate_certs: False
      name: "MyOwnPersonalInstance"
      image: "{{ ImageName }}"
      key_name: "{{ InfraKey }}"
      timeout: 200
      flavor:   "{{ FlavorName }}"
      security_groups: "{{ SecurityGroup }}"
      network: "{{ NetworkName }}"
      auto_ip: yes
```

```
$ ansible-playbook  -s launch_compute.yml
[WARNING]: provided hosts list is empty, only localhost is available
PLAY [launch a compute instance] ***********************************************
TASK [Create and launch the VM] ************************************************
changed: [localhost]
PLAY RECAP *********************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0
```

Informieren Sie sich über unsere neue Instanz

Verwenden Sie den Befehl 'os_server_facts' vom Modul 'Cloud' [http://docs.ansible.com/ansible/os_server_module.html] . Variablen sind in ./group_vars/all definiert und der Instanzname befindet sich auf dem Server: "MyOwnPersonalInstance".

```
$vi get_compute_info.yml
- name: Get and print instance IP
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Get VM infos
    os_server_facts:
      auth:
        auth_url: https://openstack-identity.mygroup/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ TenantName }}"
```

```
      validate_certs: False
      server: "MyOwnPersonalInstance"

  - name: Dump all
    debug:
      var: openstack_servers
```

```
$ansible-playbook  -s get_compute_info.yml
[WARNING]: provided hosts list is empty, only localhost is available
PLAY [Get and print instance IP] ***********************************************
TASK [Get VM IP] ***************************************************************
ok: [localhost]
TASK [Affichage] ***************************************************************
ok: [localhost] => {
    "openstack_servers": [
        {
            "OS-DCF:diskConfig": "MANUAL",
            "OS-EXT-AZ:availability_zone": "fr",
            "OS-EXT-STS:power_state": 1,
            "OS-EXT-STS:task_state": null,
[...]
            "volumes": []
        }
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0
```

Das ist sehr wortreich. Es werden viele Informationen angezeigt. Normalerweise wird nur die IP-Adresse benötigt, um über SSH auf die neue Instanz zuzugreifen.

Erhalten Sie Ihre neue öffentliche IP-Instanz

Anstatt alle Informationen auszudrucken, drucken wir nur die IP-Adresse der ersten Instanz mit dem Namen "MyOwnPersonalInstance". Es ist normalerweise alles was wir brauchen.

```
$vi get_compute_ip.yml
- name: Get and print instance IP
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Get VM infos
    os_server_facts:
      auth:
        auth_url: https://openstack-identity.mycompany.com/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ TenantName }}"
      validate_certs: False
      server: "MyOwnPersonalInstance"

  - name: Dump IP
    debug:
      var: openstack_servers[0].interface_ip
```

Löschen Sie unsere Instanz

Um unsere Instanz zu löschen, verwenden Sie den Befehl os_server erneut mit allen Authentifizierungsinformationen und ersetzen Sie einfach 'state: present' durch 'state: abwesend'.

```
$vi stop_compute.yml
- name: launch a compute instance
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Create and launch the VM
    os_server:
      auth:
        auth_url: https://openstack-identity.mygroup/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ ProjectName }}"
      state: absent
      validate_certs: False
      name: "{{ TPUser }}"
      timeout: 200
```

Ansible mit OpenStack verwenden online lesen:
https://riptutorial.com/de/ansible/topic/8712/ansible-mit-openstack-verwenden

Kapitel 6: Ansible: Looping

Examples

with_items - einfache Liste

Eine `with_items` Schleife in ansible kann verwendet werden, um Werte leicht zu überschreiben.

```
- name: Add lines to this file
  lineinfile: dest=/etc/file line={{ item }} state=present
  with_items:
    - Line 1
    - Line 2
    - Line 3
```

with_items - vordefinierte Liste

Sie können auch eine Variablenliste durchlaufen.

Von vars:

```
favorite_snacks:
  - hotdog
  - ice cream
  - chips
```

und dann die Schleife:

```
- name: create directories for storing my snacks
  file: path=/etc/snacks/{{ item }} state=directory
  with_items: '{{ favorite_snacks }}'
```

Wenn Sie Ansible 2.0+ verwenden, müssen Sie den Aufruf der Variablen in Anführungszeichen setzen.

with_items - vordefiniertes Wörterbuch

Es ist möglich, komplexere Schleifen mit Wörterbüchern zu erstellen.

Von vars:

```
packages:
  - present: tree
  - present: nmap
  - absent: apache2
```

dann die Schleife:

```
- name: manage packages
  package: name={{ item.value }} state={{ item.key }}
  with_items: '{{ packages }}'
```

Oder, wenn Sie den Schlüsselwert nicht verwenden möchten:

Vars:

```
packages:
  - name: tree
    state: present
  - name: nmap
    state: present
  - name: apache2
    state: absent
```

dann die Schleife:

```
- name: manage packages
  package: name={{ item.name }} state={{ item.state }}
  with_items: '{{ packages }}'
```

with_items - Wörterbuch

Sie können ein Wörterbuch für eine etwas komplexere Schleife verwenden.

```
- name: manage packages
  package: name={{ item.name }} state={{ item.state }}
  with_items:
    - { name: tree, state: present }
    - { name: nmap, state: present }
    - { name: apache2, state: absent }
```

Verschachtelte Schleifen

Sie können verschachtelte Schleifen mit `with_nested` .

von vars:

```
keys:
  - key1
  - key2
  - key3
  - key4
```

dann die Schleife:

```
- name: Distribute SSH keys among multiple users
  lineinfile: dest=/home/{{ item[0] }}/.ssh/authorized_keys line={{ item[1] }} state=present
  with_nested:
    - [ 'calvin', 'josh', 'alice' ]
    - '{{ keys }}'
```

Diese Task durchläuft die einzelnen Benutzer und füllt ihre `authorized_keys` mit den 4 in der Liste definierten Schlüsseln.

Ansible: Looping online lesen: https://riptutorial.com/de/ansible/topic/6414/ansible--looping

Kapitel 7: Ansible: Loops und Bedingungen

Bemerkungen

Offizielle Dokumente erklären die Bedingungen für das Playbook.

- http://docs.ansible.com/ansible/playbooks_conditionals.html

Ansible (Github)

- https://github.com/marxwang/ansible-learn-resources

Examples

Welche Arten von Bedingungen sollten verwendet werden?

Verwenden Sie Conditionals via (Syntax steht in `[brackets]`):

- wann [**wann:**]

```
Task:
   - name: run if operating system is debian
     command: echo "I am a Debian Computer"
     when: ansible_os_family == "Debian"
```

- Schleifen [**with_items:**]
- Schleifen [**with_dicts:**]
- Benutzerdefinierte Fakten [**when:** my_custom_facts == '1234']
- Bedingte Einfuhren
- Wählen Sie Dateien und Vorlagen basierend auf Variablen aus

[When] Bedingung: `ansible_os_family` Listen

Allgemeiner Gebrauch

- wann: ansible_os_family == "CentOS"
- wann: ansible_os_family == "Redhat"
- wann: ansible_os_family == "Darwin"
- wann: ansible_os_family == "Debian"
- wann: ansible_os_family == "Windows"

Alle Listen

basierend auf hier diskutieren http://comments.gmane.org/gmane.comp.sysutils.ansible/4685

```
OS_FAMILY = dict(
            RedHat = 'RedHat',
            Fedora = 'RedHat',
            CentOS = 'RedHat',
            Scientific = 'RedHat',
            SLC = 'RedHat',
            Ascendos = 'RedHat',
            CloudLinux = 'RedHat',
            PSBM = 'RedHat',
            OracleLinux = 'RedHat',
            OVS = 'RedHat',
            OEL = 'RedHat',
            Amazon = 'RedHat',
            XenServer = 'RedHat',
            Ubuntu = 'Debian',
            Debian = 'Debian',
            SLES = 'Suse',
            SLED = 'Suse',
            OpenSuSE = 'Suse',
            SuSE = 'Suse',
            Gentoo = 'Gentoo',
            Archlinux = 'Archlinux',
            Mandriva = 'Mandrake',
            Mandrake = 'Mandrake',
            Solaris = 'Solaris',
            Nexenta = 'Solaris',
            OmniOS = 'Solaris',
            OpenIndiana = 'Solaris',
            SmartOS = 'Solaris',
            AIX = 'AIX',
            Alpine = 'Alpine',
            MacOSX = 'Darwin',
            FreeBSD = 'FreeBSD',
            HPUX = 'HP-UX'
        )
```

Wenn Bedingung

Grundlegende Verwendung

Mit der when-Bedingung können Sie steuern, ob eine Aufgabe oder Rolle ausgeführt wird oder übersprungen wird. Dies wird normalerweise verwendet, um das Spielverhalten basierend auf Fakten des Zielsystems zu ändern. Betrachten Sie dieses Spielbuch:

```
- hosts: all
  tasks:
    - include: Ubuntu.yml
      when: ansible_os_family == "Ubuntu"
```

```
    - include: RHEL.yml
      when: ansible_os_family == "RedHat"
```

Wo `Ubuntu.yml` und `RHEL.yml` eine verteilungsspezifische Logik enthalten.

Eine weitere häufige Verwendung ist das Begrenzen der Ergebnisse auf bestimmte Ansible-Bestandsgruppen. Betrachten Sie diese Bestandsdatei:

```
[dbs]
mydb01

[webservers]
myweb01
```

Und dieses Spielbuch:

```
- hosts: all
  tasks:
    - name: Restart Apache on webservers
      become: yes
      service:
        name: apache2
        state: restarted
      when: webservers in group_names
```

Hier wird die magische Variable `group_names` .

Bedingte Syntax und Logik

Einzelbedingung

Syntax

```
when: (condition)
```

Beispiel

- `when: ansible_os_family == "Debian"`
- `when: ansible_pkg_mgr == "apt"`
- `when: myvariablename is defined`

Boolescher Filter

Beispiel

```
when: result|failed
```

Mehrere Bedingungen

Syntax

```
When: condition1 and/or condition2
```

Beispiel (einfach)

```
when: ansible_os_family == "Debian" and ansible_pkg_mgr == "apt"
```

Beispiel (komplex)

Verwenden Sie zur Klarheit oder zur Kontrolle der Rangfolge Klammern. "UND" hat eine höhere Priorität als "ODER".

Klauseln können Zeilen umfassen:

```
when:
  ansible_distribution in ['RedHat', 'CentOS', 'ScientificLinux'] and
  (ansible_distribution_version|version_compare('7', '<') or
  ansible_distribution_version|version_compare('8', '>='))
  or
  ansible_distribution == 'Fedora'
  or
  ansible_distribution == 'Ubuntu' and
  ansible_distribution_version|version_compare('15.04', '>=')
```

Beachten Sie die Verwendung von Klammern, um das "oder" bei der ersten Verteilungsprüfung zu gruppieren.

Holen Sie sich `ansible_os_family` und` ansible_pkg_mgr` mit dem Setup

Wir können Fakten (`ansible_os_family` , `ansible_pkg_mgr`) mit dem Ad-Hoc-Befehl des Setup-Moduls und des Filters erhalten.

- ansible_os_family:

```
$ ansible all -m setup -a 'filter=ansible_os_family'
ra.local | SUCCESS => {
    "ansible_facts": {
        "ansible_os_family": "Debian"
    },
    "changed": false
}
```

- ansible_pkg_mgr:

```
$ ansible all -m setup -a 'filter=ansible_pkg_mgr'
debian.local | SUCCESS => {
    "ansible_facts": {
        "ansible_pkg_mgr": "apt"
    },
    "changed": false
}
```

Einfache "Wann" -Beispiele

Gegeben:

```
---
variable_name: True
```

Dann laufen diese Aufgaben immer mit.

```
- name: This is a conditional task
  module: src=/example/ dest=/example
  when: variable_name

- name: This is a conditional task
  module: src=/example/ dest=/example
  when: True
```

Diese Aufgabe wird niemals ausgeführt.

```
- name: This is a conditional task
  module: src=/example/ dest=/example
  when: False
```

Verwenden Sie bis, um eine erneute Wiederholung des Live-Checks zu versuchen

Dies ist ein Beispiel für die Verwendung einer Option / retries / delay, um eine Live-Überprüfung für eine startende Webanwendung zu implementieren. Es wird davon ausgegangen, dass es einige Zeit (bis zu 3 Minuten) gibt, in der die Webapp Socket-Verbindungen ablehnt. Danach wird die / Alive-Seite nach dem Wort "OK" durchsucht. Es delegiert auch den Abruf der URL an den localhost, der ausgeführt wird. Dies ist als letzte Aufgabe in einem Bereitstellungsspielbuch sinnvoll.

```
---
- hosts: my-hosts
  tasks:
  - action: uri url=http://{{ ansible_all_ipv4_addresses }}:8080/alive return_content=yes
    delegate_to: localhost
    register: result
    until: "'failed' not in result and result.content.find('OK') != -1"
    retries: 18
    delay: 10
```

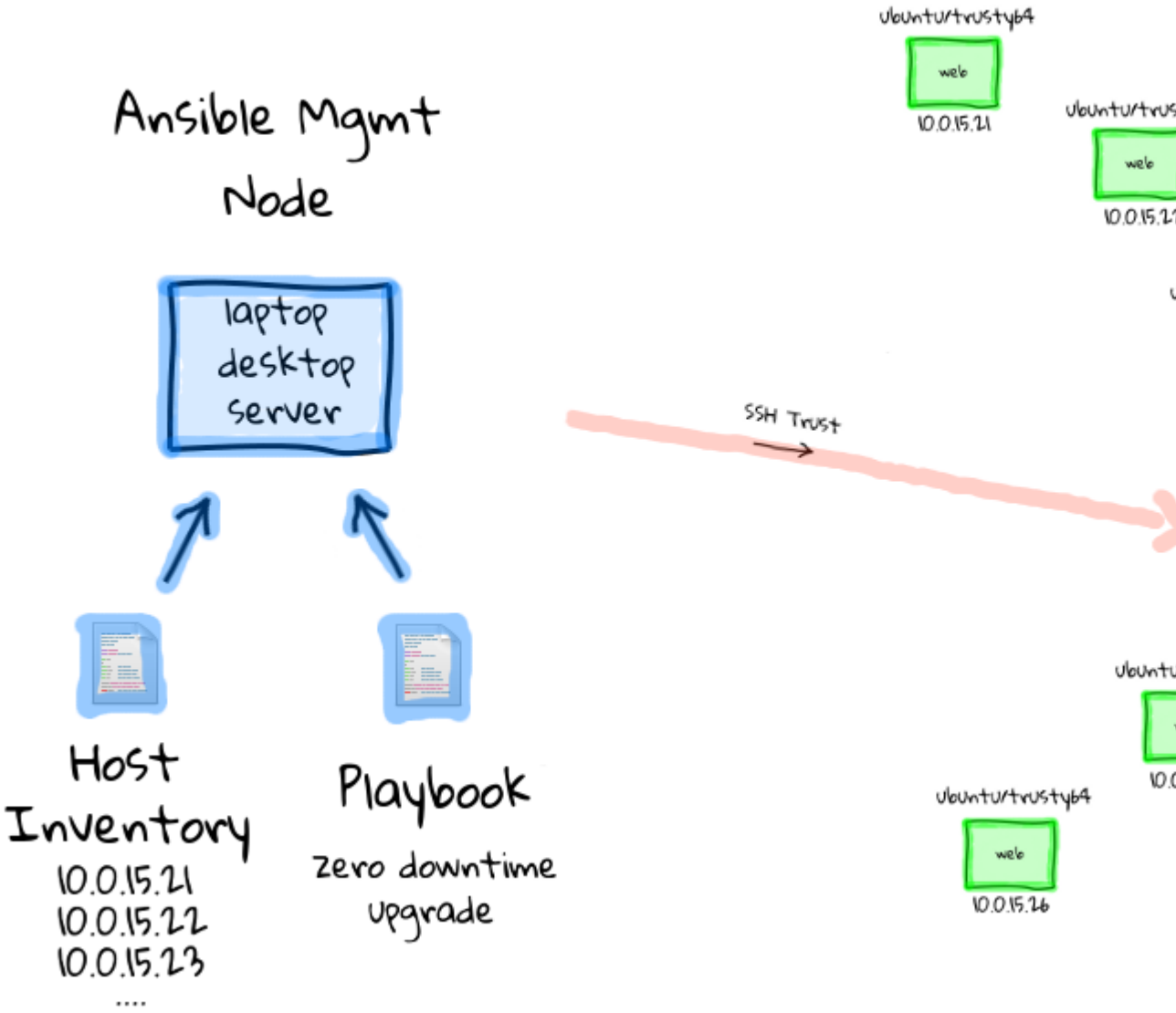
Das Muster für das Wiederholungsversuch kann mit jeder Aktion verwendet werden. Die Ansible-Dokumentation enthält ein Beispiel für das Warten, bis ein bestimmter Shell-Befehl ein gewünschtes Ergebnis zurückgibt: http://docs.ansible.com/ansible/playbooks_loops.html#do-until-loops .

Ansible: Loops und Bedingungen online lesen:
https://riptutorial.com/de/ansible/topic/3555/ansible--loops-und-bedingungen

Kapitel 8: Ansible-Architektur

Examples

Ansible-Architektur verstehen



Die Idee ist, eine oder mehrere Steuerungsmaschinen zu haben, von denen aus Sie Ad-hoc-Befehle an entfernte Maschinen `ansible` können (über ein `ansible` Tool) oder einen sequenzierten Befehlssatz über Playbooks `ansible-playbook` (über ein `ansible-playbook` Tool).

Grundsätzlich verwenden wir Ansible Control Machine, dies ist in der Regel Ihr Desktop, Laptop oder Server. Von dort aus verwenden Sie Ansible, um Konfigurationsänderungen per ssh herauszuschieben.

Die Host-Inventardatei bestimmt die Zielcomputer, auf denen diese Spiele ausgeführt werden. Die Ansible-Konfigurationsdatei kann an die Einstellungen in Ihrer Umgebung angepasst werden.

Ansible-Architektur online lesen: https://riptutorial.com/de/ansible/topic/7659/ansible-architektur

Kapitel 9: Dynamisches Inventar

Bemerkungen

Umgebungsvariablen im dynamischen Inventar funktionieren nicht, z

```
"ansible_ssh_private_key_file": $HOME/.ssh/key.pem"
```

Wenn der Dynamic-Inventory-Server beispielsweise `$HOME` durchläuft, ersetzen Sie die Variable im Clientcode (Python):

```
json_input.replace("$HOME", os.environ.get("HOME"))
```

Examples

Dynamisches Inventar mit Anmeldeinformationen

Dynamisches Inventar an `ansible-playbook` :

```
ansible-playbook -i inventory/dyn.py -l targethost my_playbook.yml
```

`python inventory/dyn.py` sollte so etwas drucken:

```
{
  "_meta": {
    "hostvars": {
      "10.1.0.10": {
        "ansible_user": "vagrant",
        "ansible_ssh_private_key_file": "/home/mrtuovinen/.ssh/id_rsa",
        "ansible_port": 22
      },
      "10.1.0.11": {
        "ansible_user": "ubuntu",
        "ansible_ssh_private_key_file": "/home/mrtuovinen/.ssh/id_rsa",
        "ansible_port": 22
      },
      "10.1.0.12": {
        "ansible_user": "steve",
        "ansible_ssh_private_key_file": "/home/mrtuovinen/.ssh/key.pem",
        "ansible_port": 2222
      }
    }
  },
  "vagrantbox": [
    "10.1.0.10"
  ],
  "ubuntubox": [
    "10.1.0.11"
  ],
  "osxbox": [
```

```
    "10.1.0.12"
  ]
}
```

Dynamisches Inventar online lesen: https://riptutorial.com/de/ansible/topic/1758/dynamisches-inventar

Kapitel 10: Einführung in Playbooks

Examples

Überblick

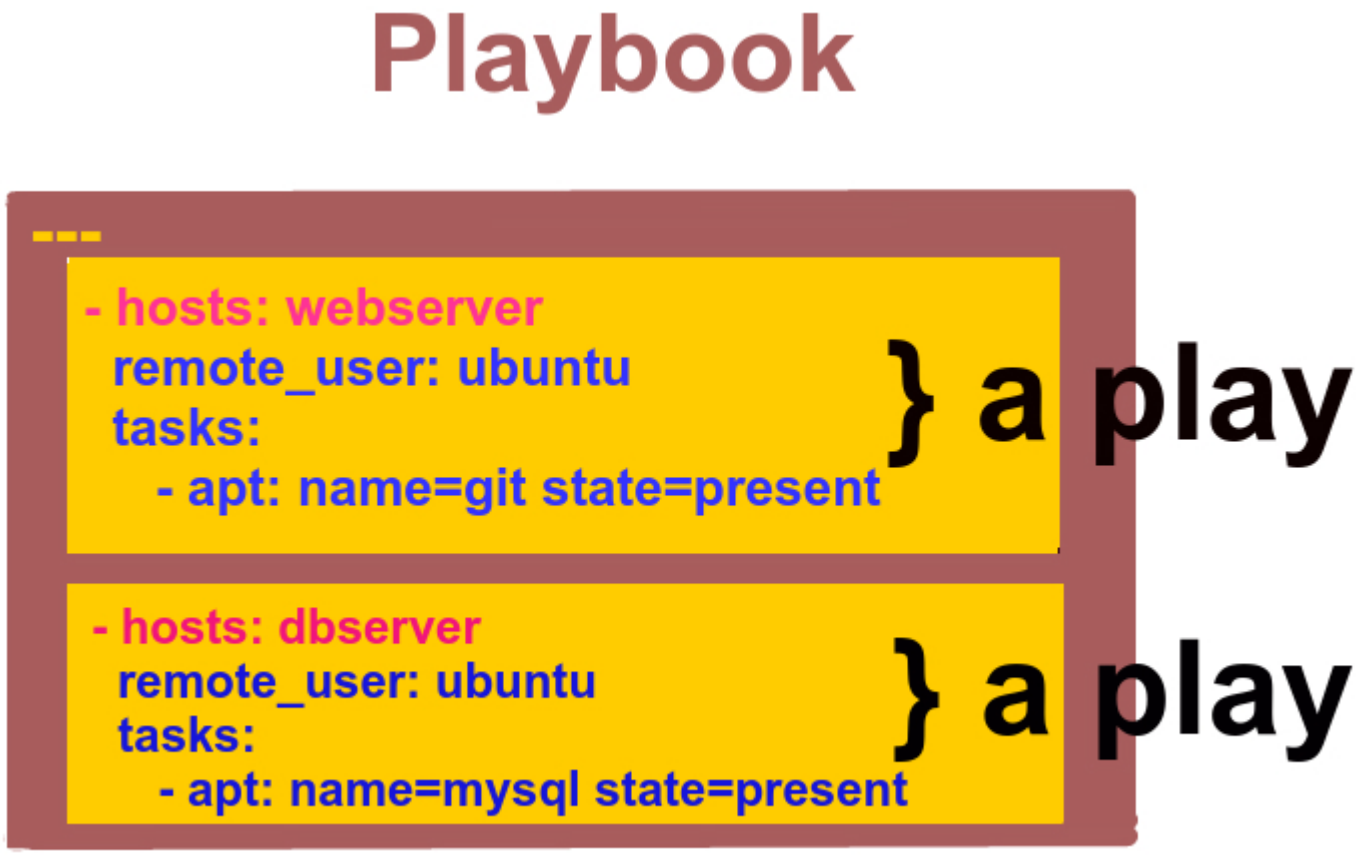
In Ansible ist ein Playbook eine YAML-Datei mit der Definition, wie ein Server aussehen soll. In einem Playbook legen Sie fest, welche Maßnahmen Ansible ergreifen soll, um den Server in den gewünschten Zustand zu bringen. Nur das, was Sie definieren, wird erledigt.

Dies ist ein grundlegendes ansible Textbuch , das git auf jedem Host gehört zur installiert `web` - Gruppe:

```
---
- name: Git installation
  hosts: web
  remote_user: root
  tasks:
    - name: Install Git
      apt: name=git state=present
```

Struktur des Spielbuchs

Das Format eines Spielbuchs ist recht unkompliziert, jedoch in Bezug auf Abstand und Layout streng. Ein Spielbuch besteht aus Spielen. Ein Spiel ist eine Kombination aus Ziel-Hosts und den Aufgaben, die auf diesen Hosts ausgeführt werden sollen. Eine Zeichnung eines Spielbuchs lautet also:



Um dieses Playbook auszuführen, führen wir einfach Folgendes aus:

```
ansible-playbook -i hosts my_playbook.yml
```

Spielstruktur

Hier ist ein einfaches Spiel:

```
- name: Configure webserver with git
  hosts: webserver
  become: true
  vars:
    package: git
  tasks:
    - name: install git
      apt: name={{ package }} state=present
```

Wie bereits erwähnt, muss jedes Spiel Folgendes enthalten:

- Eine Gruppe von zu konfigurierenden Hosts
- Eine Liste von Aufgaben, die auf diesen Hosts ausgeführt werden sollen

Stellen Sie sich ein Stück als das Ding vor, das Hosts mit Aufgaben verbindet. Neben der Angabe von Hosts und Aufgaben unterstützen die Spiele auch eine Reihe optionaler Einstellungen. Zwei übliche sind:

- `name` : Ein Kommentar, der beschreibt, worum es in dem Stück geht. Ansible druckt dies aus, sobald das Spiel beginnt
- `vars` : eine Liste von Variablen und Werten

Stichworte

Play enthält mehrere Aufgaben, die markiert werden können:

```
- name: Install applications
  hosts: all
  become: true
  tasks:
    - name: Install vim
      apt: name=vim state=present
      tags:
        - vim
    - name: Install screen
      apt: name=screen state=present
      tags:
        - screen
```

Task mit dem Tag 'vim' wird ausgeführt, wenn in den Tags 'vim' angegeben ist. Sie können beliebig viele Tags angeben. Es ist nützlich, Tags wie 'install' oder 'config' zu verwenden. Dann können Sie ein Playbook mit Angabe von Tags oder Skip-Tags ausführen. Zum

```
ansible-playbook my_playbook.yml --tags "tag1,tag2"
ansible-playbook my_playbook.yml --tags "tag2"
ansible-playbook my_playbook.yml --skip-tags "tag1"
```

Standardmäßig werden alle Tags von Ansible ausgeführt

Einführung in Playbooks online lesen: https://riptutorial.com/de/ansible/topic/3343/einfuhrung-in-playbooks

Kapitel 11: Galaxis

Examples

Rollen teilen mit Ansible Galaxy

Es ist auch möglich, problemlos Rollen mit der Community zu teilen oder Rollen, die von anderen Mitgliedern der Community mit Ansible Galaxy erstellt wurden, herunterzuladen.

Ansible wird mit einem Befehlszeilentool namens `ansible-galaxy` , mit dem Rollen in dem in der Datei `ansible.cfg` definierten `ansible.cfg` werden können:

```
ansible-galaxy install username.rolename
```

Sie können das Ansible Galaxy-Tool auch verwenden, um Rollen von anderen Speicherorten wie GitHub herunterzuladen, indem Sie eine Textdatei mit dem als `src` definierten Speicherort erstellen:

```
- src: https://github.com/username/rolename
```

Und dann installieren Sie die Rollen wie folgt in der Textdatei:

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

Sie können auch das `ansible-galaxy` Tool verwenden, um eine Rolle "Gerüst" zu erstellen:

```
ansible-galaxy init rolename
```

Wenn Sie eine Rolle erstellt und in GitHub hochgeladen haben, können Sie sie auf Ansible Galaxy freigeben, indem Sie nach der Anmeldung eine Verknüpfung zu Ihrem GitHub-Repo in Ansible Galaxy herstellen.

Weitere Beispiele unter Galaxy-Thema .

Galaxis online lesen: https://riptutorial.com/de/ansible/topic/6599/galaxis

Kapitel 12: Galaxis

Examples

Grundlegende Befehle

Suchfunktion in Ansible Galaxy

```
ansible-galaxy search role_name
```

Installieren Sie die Rolle von Ansible Galaxy

```
ansible-galaxy install role_name
```

Mehr Hilfe

```
ansible-galaxy --help
```

Galaxis online lesen: https://riptutorial.com/de/ansible/topic/6656/galaxis

Kapitel 13: Geheime Verschlüsselung

Bemerkungen

Ansible bietet Vault (nicht zu verwechseln mit HashiCorp Vault !), Um sensible Datenverschlüsselung zu handhaben. Vault dient hauptsächlich dazu, strukturierte Daten wie Variablen, Aufgaben und Handler zu verschlüsseln.

Examples

Verschlüsselte vertrauliche strukturierte Daten

Erstellen Sie zunächst eine Schlüsseldatei, z. B. `vault_pass_file` , die idealerweise eine lange Folge von zufälligen Zeichen enthält. In Linux-Systemen können Sie `pwgen` , um eine zufällige Kennwortdatei zu erstellen:

```
pwgen 256 1 > vault_pass_file
```

Verwenden Sie dann diese Datei, um vertrauliche Daten zu verschlüsseln, z. B. `groups_vars/group.yml` :

```
ANSIBLE_VAULT_PASSWORD_FILE=vault_pass_file ansible-vault encrypt group_vars/group.yml
```

Um ein Playbook zu starten, benötigen Sie ab `vault_pass_file` die `vault_pass_file` :

```
ANSIBLE_VAULT_PASSWORD_FILE=vault_pass_file ansible-playbook -i inventories/nodes my-
playbook.yml
```

Beachten Sie, dass Sie anstelle der Umgebungsvariablen `ANSIBLE_VAULT_PASSWORD_FILE` auch das Flag - `--vault-password-file vault_pass_file` verwenden können.

Um das Geheimnis auf dem Datenträger zu bearbeiten oder zu entschlüsseln, können Sie das `ansible-vault edit` mit `ansible-vault decrypt` verwenden.

Verwenden von Lookup-Pipes zum Entschlüsseln nicht strukturierter, mit einem Tresor verschlüsselter Daten

Mit Vault können Sie auch nicht strukturierte Daten, z. B. private Schlüsseldateien, verschlüsseln und sie dennoch mit dem `lookup` Modul im Spiel entschlüsseln.

```
---

- name: Copy private key to destination
  copy:
    dest=/home/user/.ssh/id_rsa
    mode=0600
```

```
    content=lookup('pipe', 'ANSIBLE_VAULT_PASSWORD_FILE=vault_pass_file ansible-vault view
keys/private_key.enc')
```

Verwenden von local_action zum Entschlüsseln von mit Vault verschlüsselten Vorlagen

Mit dem `local_action` Modul können Sie ein Spiel ausführen, das mit Tresor-verschlüsselten Vorlagen `local_action` .

```
---

- name: Decrypt template
  local_action: "shell {{ view_encrypted_file_cmd }} {{ role_path }}/templates/template.enc >
{{ role_path }}/templates/template"
  changed_when: False

- name: Deploy template
  template:
    src=templates/template
    dest=/home/user/file

- name: Remove decrypted template
  local_action: "file path={{ role_path }}/templates/template state=absent"
  changed_when: False
```

Bitte beachten Sie das `changed_when: False` . Dies ist wichtig, wenn Sie Idempotenztests mit Ihren Rollen ausführen, ansonsten wird jedes Mal, wenn Sie das Playbook ausführen, eine Änderung angezeigt. In `group_vars/all.yml` Sie einen globalen Entschlüsselungsbefehl zur Wiederverwendung `view_encrypted_file_cmd` , z. B. als `view_encrypted_file_cmd` .

group_vars / all.yml

```
---

view_encrypted_file_cmd: "ansible-vault --vault-password-file {{ lookup('env',
'ANSIBLE_VAULT_PASSWORD_FILE') }} view"
```

Wenn Sie nun ein Spiel `ANSIBLE_VAULT_PASSWORD_FILE` müssen Sie die Umgebungsvariable `ANSIBLE_VAULT_PASSWORD_FILE` , dass sie auf Ihre `ANSIBLE_VAULT_PASSWORD_FILE` (idealerweise mit einem absoluten Pfad).

Geheime Verschlüsselung online lesen: https://riptutorial.com/de/ansible/topic/3355/geheime-verschlusselung

Kapitel 14: Installation

Einführung

Installieren von Ansible in einem beliebigen Betriebssystem, einschließlich Windows mit Virtual Box und Vagrant. Eine alternative Lösung ist auch verfügbar, wenn Sie nur anpassbare Ad-hoc-Befehle und Playbooks üben möchten und keine lokale Umgebung einrichten möchten.

Examples

Ansible auf Ubuntu installieren

Ansible verwaltet ein PPA-Repository, mit dem die Ansible-Binärdateien installiert werden können:

```
sudo apt-add-repository ppa:ansible/ansible -y
sudo apt-get update && sudo apt-get install ansible -y
```

Verwenden Sie `pip` um eine bestimmte Version zu installieren. Die PPA ist möglicherweise nicht mehr aktuell.

Ansible unter MacOS installieren

Es gibt zwei Möglichkeiten, Ansible unter OS X zu installieren, entweder mit dem Homebrew- oder dem Pip-Paket-Manager.

Wenn Sie über ein Homebrew verfügen, kann das neueste Ansible mit dem folgenden Befehl installiert werden:

```
brew install ansible
```

Um den Ansible 1.9.X-Zweig zu installieren, verwenden Sie den folgenden Befehl:

```
brew install homebrew/versions/ansible19
```

Um den Ansible 2.0.X-Zweig zu installieren, verwenden Sie den folgenden Befehl:

```
brew install homebrew/versions/ansible20
```

Verwenden Sie zum Installieren mit pip den folgenden Befehl: `pip install ansible` .

Verwenden Sie `pip install ansible=<required version>` um eine bestimmte Version zu `pip install ansible=<required version>` .

Installation auf Red Hat-basierten Systemen

Ansible kann auf CentOS oder anderen Red Hat-basierten Systemen installiert werden. Zunächst sollten Sie die Voraussetzungen installieren:

```
sudo yum -y update
sudo yum -y install gcc libffi-devel openssl-devel python-pip python-devel
```

dann installiere Ansible mit pip:

```
sudo pip install ansible
```

Ich kann Ihnen empfehlen, die setuptools nach der Installation zu aktualisieren:

```
sudo pip install --upgrade setuptools
```

Sie können auch den lokalen Package Manager verwenden:

```
yum install ansible
```

Installation von der Quelle

Ansible wird am **besten** an einer Kasse verwendet.

Es läuft wie Sie (nicht root) und hat minimale Python-Abhängigkeiten.

Installation der Python-Pip-Abhängigkeit mit Pip:

```
sudo pip install paramiko PyYAML Jinja2 httplib2 six
```

Als nächstes klonen Sie das Ansible-Repo aus GitHub:

```
cd ~/Documents
git clone git://github.com/ansible/ansible.git --recursive
cd ansible
```

Fügen Sie abschließend die Ansible-Initialisierungsskriptzeile zu ~ / .bashrc oder ~ / .zshrc hinzu:

```
source ~/Documents/ansible/hacking/env-setup
```

Starten Sie Ihre Terminalsitzung neu und testen Sie mit

```
ansible --version
```

Installation unter Amazon Linux von git repo

Amazon Linux ist eine RHEL-Variante, daher sollten die Red Hat-Anweisungen größtenteils funktionieren. Es gibt jedoch mindestens eine Diskrepanz.

Es gab einen Fall, in dem **das** Paket **python27-devel** im Gegensatz zu **python-devel** ausdrücklich erforderlich war.

Hier werden wir aus dem Quellcode installieren.

```
sudo yum -y update
sudo yum -y install python27 python27-devel openssl-devel libffi-devel gcc git

git clone https://github.com/ansible/ansible/<search the github for a preferable branch>

cd ansible
sudo python setup.py build
sudo python setup.py install
```

Installieren von Ansible auf einem beliebigen Betriebssystem (Windows) mit Virtual Box + Vagrant

Mein Laptop hat Windows 10. Hier erkläre ich Schritte, die Sie zum Testen und Lernen von Ansible befolgen können.

Einige Theorie

Für Ansible benötigen Sie eine Steuerungsmaschine und einen Host (oder Hosts), um das Playbook auszuführen.

- **Control Machine** sollte auf Linux oder MacOS (Windows nicht zulässig) basieren und Python (Version 2.6 oder höher) benötigen. Hier wird Ansible installiert.
- **Zielmaschine** (Host / Knoten) kann Linux / MacOS / Windows sein. Hierfür muss nur Python installiert werden. Keine Agentensoftware erforderlich.

KONFIGURATION

Schritt 1: **Installieren Sie Virtual Box**

Virtual Box ist eine Software zum Erstellen von virtuellen Computern mit unterschiedlichen Betriebssystemen. Es ist, als hätten Sie mehrere Computer oder unterschiedliche Betriebssysteme und unterschiedliche Versionen.

Laden Sie die Virtual Box entsprechend dem Betriebssystem in Ihrem System herunter und installieren Sie sie.

Schritt 2: **Installieren Sie Vagrant**

Vagrant ist eine Befehlszeilenschnittstelle, um virtuelle Maschinen in einer virtuellen Box zu erstellen. Das macht die Sache einfach. Sie müssen grundlegende Vagrant-Befehle lernen.

Schritt 3: **Erstellen Sie einen Ordner, in dem Sie Ihre virtuelle Maschine haben möchten**

Schritt 4: **Erstellen Sie eine virtuelle Maschine mit Vagrant**

Öffnen Sie das Terminal, gehen Sie zu dem Pfad, in dem Sie den Ordner erstellt haben, und führen Sie die folgenden beiden Befehle aus.

Sie müssen **Virtual Box** auswählen. Ich installiere zum Beispiel Ubuntu. Sie können alles aus der Liste auswählen. Sie müssen diese beiden Befehle in der Kategorie " **virtuelle Box** " ausführen: `vagrant init ubuntu/trusty64` und `vagrant up --provider virtualbox` . Andere Kategorien können sein: hyperv, vmware_desktop usw. (dies wird einige Zeit dauern, da die erforderlichen Dateien heruntergeladen werden).

Schritt 4: Installieren Sie Ansible

Für UbuntuOS: `sudo apt-get install ansible`

Alternative Lösung :

Sie können **Katacoda** verwenden, um **Ansible** zu üben. Sie müssen nichts installieren oder einrichten. Führen Sie zwei in Schritt 2 gegebene Befehle aus. Danach können Sie loslegen.

Installation online lesen: https://riptutorial.com/de/ansible/topic/4906/installation

Kapitel 15: Inventar

Parameter

Parameter	Erläuterung
ansible_connection	Verbindungstyp zum Host Dies kann der Name eines der Verbindungs-Plugins von ansible sein. SSH Protokolltypen sind `smart` , `ssh` oder `paramiko` . Der Standard ist smart. Nicht-SSH-basierte Typen werden im nächsten Abschnitt beschrieben.
ansible_host	Der Name des Hosts, zu dem eine Verbindung hergestellt werden soll, falls er sich von dem Alias unterscheidet, den Sie ihm geben möchten.
ansible_port	Die SSH-Portnummer, wenn nicht 22
ansible_user	Der standardmäßige ssh-Benutzername, der verwendet werden soll.
ansible_ssh_pass	Das zu verwendende ssh-Passwort (dies ist unsicher, wir empfehlen dringend die Verwendung von `--ask-pass` oder SSH-Schlüsseln)
ansible_ssh_private_key_file	Von ssh verwendete private Schlüsseldatei. Nützlich, wenn mehrere Schlüssel verwendet werden und Sie keinen SSH-Agenten verwenden möchten.
ansible_ssh_common_args	Diese Einstellung wird immer an die Standardbefehlszeile für **sftp** , **scp** und **ssh** angehängt. Nützlich zum Konfigurieren eines `ProxyCommand` für einen bestimmten Host (oder eine bestimmte Gruppe).
ansible_sftp_extra_args	Diese Einstellung wird immer an die Standard- **Sftp-** Befehlszeile angehängt.
ansible_scp_extra_args	Diese Einstellung wird immer an die Standard-Befehlszeile von **scp** angehängt.
ansible_ssh_extra_args	Diese Einstellung wird immer an die standardmäßige **ssh-** Befehlszeile angehängt.
ansible_ssh_pipelining	Bestimmt, ob SSH-Pipelining verwendet werden soll oder nicht. Dies kann die `pipelining` Einstellung in `ansible.cfg` .
ansible_become	Äquivalent zu `ansible_sudo` oder `ansible_su` , um die

Parameter	Erläuterung
	Privilegieneskalation zu erzwingen
ansible_become_method	Ermöglicht das Festlegen der Berechtigungseskalationsmethode
ansible_become_user	Äquivalent zu `ansible_sudo_user` oder `ansible_su_user` : Ermöglicht die Einstellung des Benutzers, zu dem Sie eine Privilegieneskalation erhalten
ansible_become_pass	Entspricht `ansible_sudo_pass` oder `ansible_su_pass` . Hier können Sie das Kennwort für die Berechtigungseskalation festlegen
ansible_shell_type	Der Shell-Typ des Zielsystems. Sie sollten diese Einstellung nur verwenden, wenn Sie die `ansible_shell_executable` auf eine nicht mit Bourne (sh) kompatible Shell festgelegt haben. Standardmäßig werden Befehle mit der `sh` -Style-Syntax formatiert. Wenn Sie diese Einstellung auf `csh` oder `fish` , folgen die auf Zielsystemen ausgeführten Befehle stattdessen der Syntax dieser Shell.
ansible_python_interpreter	Der Python-Pfad des Zielhosts. Dies ist nützlich für Systeme mit mehr als einem Python oder nicht unter **/ usr / bin / python** wie * BSD oder wo **/ usr / bin / python** kein Python der 2.X-Serie ist. Wir verwenden den Mechanismus **/ usr / bin / env nicht** , da hierfür der Pfad des Remote-Benutzers richtig gesetzt werden muss. Außerdem wird davon **ausgegangen** , dass die ausführbare **Python-** Datei python genannt wird, wobei die ausführbare Datei etwa **python2.6 heißen könnte** .
ansible _ * _ dolmetscher	Funktioniert für alles wie Ruby oder Perl und funktioniert genauso wie `ansible_python_interpreter` . Dies ersetzt Shebang von Modulen, die auf diesem Host ausgeführt werden.
ansible_shell_executable	Dadurch wird die Shell festgelegt, die der anpassbare Controller auf dem Zielcomputer verwendet. Er überschreibt die `executable` in `ansible.cfg` die standardmäßig auf **/ bin / sh eingestellt ist** . Sie sollten es eigentlich nur ändern, wenn **/ bin / sh nicht verwendet werden kann** (dh **/ bin / sh** ist nicht auf dem Zielcomputer installiert oder kann nicht von sudo ausgeführt werden.). Neu in Version 2.1.

Examples

Inventar mit Benutzername und Passwort

Inventory ist die Ansible-Methode, um alle Systeme in Ihrer Infrastruktur zu verfolgen. Hier ist eine einfache Inventardatei, die ein einzelnes System und die Anmeldeinformationen für Ansible enthält.

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ansible_ssh_pass=PassW0rd
```

Inventar mit benutzerdefiniertem privaten Schlüssel

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ssh_private_key_file=~/.ssh/custom_key
```

Inventar mit benutzerdefiniertem SSH-Port

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ansible_port=2222
```

Statisches Inventar an das Ansible-Playbook übergeben

```
ansible-playbook -i path/to/static-inventory-file -l myhost myplaybook.yml
```

Übergeben Sie das dynamische Inventar an das Ansible-Playbook

```
ansible-playbook -i path/to/dynamic-inventory-script.py -l myhost myplaybook.yml
```

Weitere Informationen finden Sie unter Dynamisches Inventar .

Inventar, Gruppen-Vars und Sie

Projektstruktur (Ansible Best Practice).

```
project/
  group_vars/
     development
  inventory.development
  playbook.yaml
```

Alles beginnt mit der Entwicklung von Lagerbeständen

```
[development]
dev.fakename.io

[development:vars]
ansible_host: 192.168.0.1
ansible_user: dev
ansible_pass: pass
```

```
ansible_port: 2232

[api:children]
development
```

Damit können Sie auf group_vars verlinken. Halten Sie die Daten für diese Umgebung "spezifisch" ...

```
---
app_name: NewApp_Dev
app_url: https://dev.fakename.io
app_key: f2390f23f01233f23f
```

Damit kann man das folgende Playbook gegen die Inventardatei ausführen:

```
---
- name: Install api.
  hosts: api
  gather_facts: true
  sudo: true
  tags:
    - api
  roles:
    - { role: api,         tags: ["api"]         }
```

mit der folgenden runline:

```
ansible-playbook playbook.yaml -i  inventory.development
```

Hosts-Datei

Die Hostdatei wird zum Speichern von Verbindungen für Anisble-Playbooks verwendet. Es gibt Optionen zum Definieren von Verbindungsparametern:

`ansible_host` ist der Hostname oder die IP-Adresse

`ansible_port` ist der Port, den die Maschine für SSH verwendet

`ansible_user` ist der entfernte Benutzer, zu dem eine Verbindung hergestellt werden soll

`ansible_ssh_pass` wenn ein Kennwort für SSH verwendet wird

`ansible_ssh_private_key_file` wenn Sie mehrere für Hosts spezifische Schlüssel verwenden müssen

Dies sind die am häufigsten verwendeten Optionen. Weitere Informationen finden Sie in der offiziellen Dokumentation von Ansible .

Hier ist eine Beispiel- `hosts` Datei:

```
# Consolidation of all groups
```

```
[hosts:children]
web-servers
offsite
onsite
backup-servers

[web-servers]
server1 ansible_host=192.168.0.1 ansible_port=1600
server2 ansible_host=192.168.0.2 ansible_port=1800

[offsite]
server3 ansible_host=10.160.40.1 ansible_port=22 ansible_user=root
server4 ansible_host=10.160.40.2 ansible_port=4300 ansible_user=root

# You can make groups of groups
[offsite:children]
backup-servers

[onsite]
server5 ansible_host=10.150.70.1 ansible_ssh_pass=password

[backup-servers]
server6 ansible_host=10.160.40.3 ansible_port=77
```

Inventar online lesen: https://riptutorial.com/de/ansible/topic/1764/inventar

Kapitel 16: Rollen

Examples

Rollen verwenden

Ansible nutzt das Konzept der Rollen modularen Code besser zu ermöglichen und zu vermeiden , sich zu wiederholen.

Eine Rolle ist einfach eine Ordnerstruktur, aus der Ansible weiß, woher VAR-Dateien, Aufgaben und Handler geladen werden. Ein Beispiel könnte ungefähr so aussehen:

```
apache/
├── defaults
│   └── main.yml
├── files
│   ├── mod-pagespeed-stable_current_i386.deb
│   ├── mod-pagespeed-stable_current_i386.rpm
│   ├── mod-pagespeed-stable_current_amd64.deb
│   └── mod-pagespeed-stable_current_x86_64.rpm
├── tasks
│   ├── debian.yml
│   ├── main.yml
│   └── redhat.yml
├── templates
│   ├── httpd.conf.j2
│   └── sites-available
│       └── virthualhost.conf.j2
└── vars
    ├── debian
    └── redhat
```

Sie können die Rolle dann mit einem einfachen Spielbuch verwenden, das wie folgt aussieht:

```
- hosts: webservers
  roles:
     - apache
```

Wenn Sie Ansible für dieses Playbook ausführen, werden alle Hosts in der `webservers` und die oben definierte `apache` Rolle ausgeführt. Dabei werden automatisch alle Standardvariablen für die Rolle `tasks/main.yml` und alle in `tasks/main.yml` enthaltenen `tasks/main.yml` . Ansible kann auch nach bestimmten Dateitypen in rollenfreundlichen Speicherorten suchen:

- Wenn die Rollen / x / Aufgaben / main.yml vorhanden sind, werden die darin aufgeführten Aufgaben zum Spiel hinzugefügt
- Wenn die Rollen / x / handlers / main.yml vorhanden sind, werden die darin aufgeführten Handler zum Spiel hinzugefügt
- Wenn die Rollen / x / vars / main.yml vorhanden sind, werden die darin aufgelisteten

Variablen zum Spiel hinzugefügt

- Wenn Rollen / x / meta / main.yml vorhanden sind, werden alle darin aufgeführten Rollenabhängigkeiten der Liste der Rollen hinzugefügt (1.3 und höher).
- Jede Kopie, jedes Skript, jede Vorlage oder Include-Aufgaben (in der Rolle) können auf Dateien in den Rollen / x / {Dateien, Vorlagen, Aufgaben} / (Verzeichnis ist abhängig von der Aufgabe) verweisen, ohne sie relativ oder absolut zu pflegen

Rollenabhängigkeiten

`meta/main.yml` von Rollen können Sie auch andere Rollen als Abhängigkeit definieren, indem Sie eine `meta/main.yml` Datei mit einem `dependencies` `meta/main.yml` :

```
dependencies:
  - role: common
```

Es ist auch möglich, einen Wert an einen Parameter / eine Variable in der abhängigen Rolle zu übergeben:

```
dependencies:
  - { role: common, some_parameter: 3 }
```

Oder führen Sie die abhängige Rolle sogar bedingt aus:

```
dependencies:
  - { role: common, some_parameter: 3 }
  - { role: sshd, enable_sshd: false,
                        when: environment == 'production' }
```

Abhängige Rollen werden immer vor den von ihnen abhängigen Rollen ausgeführt. Sie werden auch nur einmal ausgeführt. Wenn zwei Rollen dieselbe als Abhängigkeit angeben, wird sie nur beim ersten Mal ausgeführt.

Stellen Sie sich die Rollen role1, role2 und role3 mit den folgenden `meta/main.yml` 's vor:
role1 / meta / main.yml:

```
dependencies:
    - role: role3
```

role2 / meta / main.yml:

```
dependencies:
    - role: role3
```

Bei der Ausführung von Rolle1 und Rolle2 im gleichen Spielbuch (wobei Rolle1 vor Rolle2 aufgerufen wurde), lautet die Ausführungsreihenfolge wie folgt:

```
role3 -> role1 -> role2
```

Sie können dieses Verhalten überschreiben, indem Sie `allow_duplicates: yes` in `meta/main.yml` von role1 und role2. Die resultierende Ausführungsreihenfolge wäre:

```
role3 -> role1 -> role3 -> role2
```

Trennen von verteilungsspezifischen Aufgaben und Variablen innerhalb einer Rolle

Verteilungsspezifische Aufgaben und Variablen lassen sich problemlos in verschiedene dedizierte .yml-Dateien aufteilen. Ansible hilft uns dabei, die Verteilung der Ziel-Hosts über `{{ ansible_distribution }}` und `{{ ansible_distribution_version }}` automatisch zu identifizieren. `{{ ansible_distribution_version }}` wir nur die .yml-Dateien der Distribution entsprechend benennen.

Für Ubuntu Xenial würde die grundlegende Rolle dir tree dann ungefähr so aussehen:

```
role
├── tasks
│   ├── main.yml
│   └── Ubuntu16.04.yml
└── vars
    └── Ubuntu16.04.yml
```

In die `tasks/main.yml` können jetzt automatisch die richtigen Variablen und Aufgaben für die Verteilung der Ziel-Hosts `tasks/main.yml` .

Aufgaben / main.yml

```
---

- name: include distribution specific vars
  include_vars: "{{ ansible_distribution }}{{ ansible_distribution_version }}.yml"

- name: include distribution specific install
  include: "{{ ansible_distribution }}{{ ansible_distribution_version }}.yml"
```

In den `tasks/Ubuntu16.06.yml` und `vars/Ubuntu16.04.yml` können jetzt Aufgaben und Variablen für Ubuntu Xenial definiert werden.

Rollen online lesen: https://riptutorial.com/de/ansible/topic/3396/rollen

Kapitel 17: Schleifen

Examples

Kopieren Sie mehrere Dateien in einer einzigen Aufgabe

```
- name: copy ssl key/cert/ssl_include files
  copy: src=files/ssl/{{ item }} dest=/etc/apache2/ssl/
  with_items:
    - g_chain.crt
    - server.crt
    - server.key
    - ssl_vhost.inc
```

Installieren Sie mehrere Pakete in einer einzigen Task

```
- name: Installing Oracle Java and support libs
  apt: pkg={{ item }}
  with_items:
    - python-software-properties
    - oracle-java8-installer
    - oracle-java8-set-default
    - libjna-java
```

Schleifen online lesen: https://riptutorial.com/de/ansible/topic/6095/schleifen

Kapitel 18: So erstellen Sie einen DreamHost Cloud Server aus einem Ansible Playbook

Examples

Installieren Sie die Shade-Bibliothek

Shade ist eine von OpenStack entwickelte Bibliothek, die die Interaktion mit OpenStack-Clouds wie DreamHost vereinfacht.

$ pip Schatten installieren

Schreibe ein Playbook, um einen Server zu starten

Erstellen Sie eine Datei namens `launch-server.yaml` , die unser Playbook sein wird.

Der erste Teil des Playbooks enthält eine Liste von Hosts, auf denen Ihr Playbook ausgeführt wird. Wir haben nur einen lokalen Host.

```
- hosts: localhost
```

Dann müssen wir eine Liste von Aufgaben definieren, die in diesem Spielbuch ausgeführt werden sollen. Es wird nur einen Server geben, der einen Ubuntu Xenial-Server auf DreamCompute startet.

```
tasks:
  - name: launch an Ubuntu server
```

Der nächste Teil des Playbooks verwendet das Modul `os_server` (OpenStack Server). Dies definiert, wie der Server in DreamCompute aussehen muss.

```
os_server:
```

Der erste Schritt ist die Authentifizierung bei DreamCompute. Ersetzen Sie `{username}` durch Ihren DreamCompute-Benutzernamen, `{password}` durch Ihr DreamCompute-Kennwort und `{project}` durch Ihr DreamCompute-Projekt. Sie finden diese in der OpenStack RC- Datei.

```
  auth:
    auth_url: https://iad2.dream.io:5000
    username: {username}
    password: {password}
    project_name: {project}
```

Nächste Zeilen definieren einige Elemente des neuen Servers.

```
  state: present
  name: ansible-vm1
  image: Ubuntu-16.04
  key_name: {keyname}
  flavor: 50
  network: public
  wait: yes
```

Lasst uns die letzten paar Zeilen aufbrechen:

- `state` ist der Status des Servers. Mögliche Werte sind `present` oder nicht `absent`
- `name` ist der Name des Servers, der erstellt werden soll. kann einen beliebigen Wert haben
- `image` ist das Image, von dem der Server gestartet werden soll. Mögliche Werte sind im DreamHost Cloud-Webpanel sichtbar. Die Variable akzeptiert entweder den Bildnamen oder die UUID
- `key_name` ist der Name des öffentlichen Schlüssels, der dem Server nach seiner `key_name` soll. Dies kann ein beliebiger Schlüssel sein, der bereits DreamCompute hinzugefügt wurde.
- `flavor` ist die Flavour des Servers, der gestartet werden soll. Dies definiert, wie viel RAM und CPU Ihr Server haben wird. Die Variable akzeptiert entweder den Namen eines Flavours (gp1.semisonic) oder die ID (50, 100, 200 usw.).
- `network` ist das Netzwerk, in dem der Server installiert werden kann. Im Fall der DreamHost Cloud ist dies das `public` Netzwerk.
- `wait` auf `wait` eingestellt ist, muss das Playbook warten, bis der Server erstellt wurde, bevor es fortfährt.

Das Playbook ausführen

Führen Sie das Ansible-Playbook aus:

```
$ ansible-playbook launch-server.yaml
```

Sie sollten Ausgabe wie sehen

```
PLAY [localhost]
*****************************************************************

TASK [setup]
*****************************************************************
ok: [localhost]

TASK [launch an Ubuntu server]
*********************************************
changed: [localhost]

PLAY RECAP
*****************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

Wenn Sie nun das DreamHost Cloud-Dashboard überprüfen, sollte eine neue Instanz mit dem Namen "ansible-vm1" angezeigt werden.

So erstellen Sie einen DreamHost Cloud Server aus einem Ansible Playbook online lesen: https://riptutorial.com/de/ansible/topic/4689/so-erstellen-sie-einen-dreamhost-cloud-server-aus-einem-ansible-playbook

Kapitel 19: Verwendung von Ansible mit Amazon Web Services

Bemerkungen

Beispiel-2: Dies ist ein Beispiel, kopieren Sie es also nicht. Stattdessen sollten Sie die Variablen an Ihre Bedürfnisse anpassen. ansible_key, Sicherheitsgruppenregeln usw.

Beispiel-1: Um die strikte Überprüfung des SSH-Host-Schlüssels zu deaktivieren, ein Verhalten, das wir beim Automatisieren von Aufgaben nicht wünschen, setzen wir es in der Datei `ansible.cfg` auf `no` . dh: `StrictHostKeyChecking=no`

Die Datei `ec2.py` ist ein Python-Skript, das Ihre AWS-Ressourcen basierend auf der Datei `ec2.ini` ausführt und zurückgibt. `ec2.ini` Konfigurationsdatei müssen Sie anpassen, wenn Sie den Umfang Ihres Projekts auf bestimmte Regionen, bestimmte Tags usw. beschränken möchten ...

Examples

So starten Sie die EC2-Instanz von offiziellen Amazon AMIs, ändern Sie sie und speichern Sie sie als neues AMI

Dies ist ein sehr üblicher Workflow, wenn Ansible zum Bereitstellen einer AWS EC2-Instanz verwendet wird. Dieser Beitrag setzt ein grundlegendes Verständnis von Ansible voraus und setzt am wichtigsten voraus, dass Sie es ordnungsgemäß konfiguriert haben, um eine Verbindung zu AWS herzustellen.

Als offizielle Dokumentation von Ansible werden wir vier Rollen verwenden:

1- **ami_find** , um die Ami-ID zu erhalten, auf deren Basis wir unsere EC2-Instanz starten werden.

2- **ec2_ami_creation** , um die EC2-Instanz effektiv zu starten.

3- **code_deploy** zum Ändern der Instanz; Dies kann alles sein, so dass wir einfach eine Datei auf den Zielcomputer übertragen.

4- **build_ami** , um unser neues Image basierend auf der laufenden ec2-Instanz zu erstellen. Dieser Beitrag setzt voraus, dass Sie sich auf der obersten Ebene Ihres Ansible-Projekts befinden: `my_ansible_project`

Die erste Rolle: **ami_find**

```
cd my_ansible_project/roles && ansible-galaxy init ami_find
```

In dieser Rolle verwenden wir das Modul ec2_ami_find. Als Beispiel suchen wir nach einem Ubuntu-Computer und erhalten dessen **ami_id** (ami-xxxxxxxx). Bearbeiten

`my_ansible_project/roles/ami_find/tasks/main.yml` Datei `my_ansible_project/roles/ami_find/tasks/main.yml` :

```
---
- ec2_ami_find:
    name: "ubuntu/images/hvm-ssd/ubuntu-trusty-14.04-amd64-server-*"
    sort: name
    sort_order: descending
    sort_end: 1
    region: "{{ aws_region }}"
  register: ami_find
- set_fact: ami_ubuntu="{{ ami_find.results[0].ami_id }}"
```

Die zweite Rolle: **ec2_ami_creation**

Hier werden wir die `ami_id` verwenden, die wir von der ersten Rolle bekommen haben, und dann unsere neue Instanz basierend darauf starten:

```
cd my_ansible_project/roles && ansible-galaxy init ec2_ami_creation
```

In dieser Rolle verwenden wir vor allem das ec2_module , um unsere Instanz zu starten. Bearbeiten `my_ansible_project/roles/ec2_ami_creation/tasks/main.yml` Datei `my_ansible_project/roles/ec2_ami_creation/tasks/main.yml` :

```
---
- ec2_vpc_subnet_facts:
    region: "{{aws_region}}"
  register: vpc
- name: creation of security group of the ec2 instance
  ec2_group:
    name: example
    description: an example EC2 group
    region: "{{ aws_region }}"
    rules:
      - proto: tcp
        from_port: 22
        to_port: 22
        cidr_ip: 0.0.0.0/0
    state: present
  register: ec2_sg

- name: create instance using Ansible
  ec2:
    key_name: "{{ ansible_key }}"
    group: example
    vpc_subnet_id: "{{vpc.subnets[0].id}}"
    instance_type: "{{ instance_type }}"
    ec2_region: "{{ aws_region }}"
    image: "{{ base_image }}"
    assign_public_ip: yes
    wait: yes
  register: ec2

- set_fact: id={{ec2.instances[0].id}}

- name: adding the newly created instance to a temporary group in order to access it later
```

```
from another play
  add_host: name={{ item.public_ip }} groups=just_created
  with_items: ec2.instances

- name: Wait for SSH to come up
  wait_for: host={{ item.public_dns_name }} port=22 delay=10 timeout=640 state=started
  with_items: ec2.instances
```

Die dritte Rolle: **code_deploy**

Hier stellen wir diese Instanz `just_created` , die zu einer Gruppe mit dem Namen `just_created`

```
cd my_ansible_project/roles && ansible-galaxy init code_deploy
```

In dieser Rolle verwenden wir das template_module , um eine Datei zu übertragen und den Hostnamen des Rechners darin zu schreiben. Bearbeiten `my_ansible_project/roles/code_deploy/tasks/main.yml` Datei `my_ansible_project/roles/code_deploy/tasks/main.yml` :

```
---
- template: src=my_file.txt.j2 dest=/etc/my_file.txt
```

Wechseln Sie dann in den Vorlagenordner in Ihrer Rolle:

`cd my_ansible_project/roles/templates` und fügen Sie eine Datei namens `my_file.txt.j2` die `my_file.txt.j2` enthält:

```
my name is {{ ansible_hostname }}`
```

Die vierte Rolle: `build_ami`

Wir erstellen jetzt ein Image der laufenden Instanz mit dem Modul ec2_ami . Gehen Sie in Ihren Projektordner und:

```
 cd my_ansible_project/roles && ansible-galaxy init build_ami
```

Bearbeiten `my_ansible_project/roles/build_ami/tasks/main.yml` Datei `my_ansible_project/roles/build_ami/tasks/main.yml` :

```
---
- ec2_ami:
    instance_id: "{{ instance_id }}"
    wait: yes
    name: Base_Image
```

Ich denke, Sie haben sich gefragt, wie Sie all diese Rollen zusammenstellen sollen. Habe ich recht? Wenn ja, lesen Sie weiter.

Wir werden ein Spielbuch schreiben, das aus drei Spielen besteht: Das erste Spiel, das auf `localhost` anwendbar ist, nennt unsere ersten beiden Rollen, das zweite Spiel gilt für unsere

Gruppe **just_created** . Die letzte Rolle wird auf `localhost` . Warum `localhost` ? Wenn wir einige AWS-Ressourcen **verwalten** möchten, verwenden wir so einfach Ihren lokalen Computer. Als Nächstes verwenden wir eine vars-Datei, in die wir unsere Variablen `ansible_key` werden: `ansible_key` , `aws_region` usw.

Erstellen Sie einen Infrastrukturordner am oberen `aws.yml` Ihres Projekts und fügen Sie eine Datei namens `aws.yml` :

```
---
aws_region: ap-southeast-2
ansible_key: ansible
instance_type: t2.small
```

Erstellen `build_base_image.yml` am Anfang Ihres Projekts `build_base_image.yml` und fügen Sie `build_base_image.yml` hinzu:

```
---
   - hosts: localhost
     connection: local
     gather_facts: False
     vars_files:
       - infrastructure/aws.yml
     roles:
       - ami_find
       - { role: ec2_creation, base_image: "{{ ami_ubuntu }}"}

   - hosts: just_created
     connection: ssh
     gather_facts: True
     become: yes
     become_method: sudo
     roles:
       - code_deploy

   - hosts: localhost
     connection: local
     gather_facts: False
     vars_files:
       - infrastructure/aws.yml
     roles:
       - { role: new_image, instance_id: "{{ id }}"}
```

Vergessen Sie nicht, Ihre Ressourcen nach dem Testen zu löschen, oder erstellen Sie eine Rolle, um die laufende Instanz zu löschen :-)

So konfigurieren Sie Ansible ordnungsgemäß für die Verbindung zu Amazon Web Services

Das Verwalten von AWS-Ressourcen, die nach oben und nach unten skaliert werden, stößt an die Grenzen der statischen Inventardatei, weshalb wir etwas Dynamisches benötigen. Dafür sind die dynamischen Lagerbestände gedacht. Lasst uns beginnen:

Laden Sie diese Dateien ec2.ini und ec2.py in Ihren Projektordner herunter:

```
cd my_ansible_project
wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.py
wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.ini
```

Machen Sie danach die Datei `ec2.py` ausführbar:

```
chmod +x ec2.py
```

Exportieren Sie jetzt Ihren AWS Secret- und Access-Schlüssel als Umgebungsvariablen:

```
export AWS_ACCESS_KEY_ID='ABCDEFGHIJKLM'
export AWS_SECRET_ACCESS_KEY='NOPQRSTUVWXYZ'
```

Um das `ec2.py` Skript verwenden zu können, benötigen Sie das Python AWS SDK, `boto` damit Sie es installieren können:

```
sudo pip install boto
```

Um zu testen, ob alles in Ordnung ist, führen Sie die `ec2.py` indem Sie Ihre Ressourcen auflisten:

```
./ec2.py --list
```

Sie sollten etwas ähnliches sehen:

```
{
  "_meta": {
    "hostvars": {}
  }
}
```

Jetzt möchten wir das dynamische Inventar zusammen mit unserer statischen Hosts-Datei verwenden. Erstellen Sie zuerst einen Ordner mit dem Namen " `inventory` , fügen Sie `ec2.py` , `ec2.ini` und unsere `hosts` Datei hinzu, und `ec2.ini` Sie Ansible an, diesen Ordner als Inventardatei zu verwenden:

```
mkdir inventory
mv ec2.py inventory/ec2.py
mv ec2.ini inventory/ec2.ini
mv hosts inventory/hosts
```

Als Nächstes sollten Sie die Konfiguration auf Projektebene für Ansible definieren, indem Sie in Ihrem Projektordner eine Ansible-Konfigurationsdatei namens `ansible.cfg` und `ansible.cfg` hinzufügen:

```
[defaults]
hostfile = inventory
[ssh_connection]
pipelining = False
ssh_args = -o ControlMaster=auto -o ControlPersist=30m -o StrictHostKeyChecking=no
```

Als Nächstes müssen wir Ansible so konfigurieren, dass ein SSH-Schlüssel verwendet wird, um den Zugriff auf unsere EC2-Instanzen zu authentifizieren. Die Verwendung eines SSH-Agenten ist der beste Weg, um sich bei Ressourcen zu authentifizieren, da dies die Verwaltung von Schlüsseln vereinfacht:

```
ssh-agent bash
ssh-add ~/.ssh/keypair.pem
```

Das ist es! Wenn Sie dies befolgt haben, können Sie es mithilfe des `ping` Moduls testen. Anschließend werden Ihre laufenden Instanzen angezeigt, die für die Verwendung Ihres Schlüssels konfiguriert wurden, der mit pong reagiert:

```
ansible -m ping all
11.22.33.44 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Verwendung von Ansible mit Amazon Web Services online lesen:
https://riptutorial.com/de/ansible/topic/3302/verwendung-von-ansible-mit-amazon-web-services

Kapitel 20: Werden (Privileg-Eskalation)

Einführung

Häufig müssen Sie Befehle unter einem anderen Benutzer ausführen oder *Root*- Rechte erhalten. Mit diesen Optionen können Sie **ein** anderer Benutzer im Gastsystem werden.

Syntax

- `become` : kann auf true oder yes gesetzt werden und löst die Einstellungen der Benutzereskalation aus.
- `become_user` : Auf den gewünschten Benutzer im Remote-Host setzen.
- `become_method` : `become_method` den Befehl an, der zum Anmelden und Ändern des Benutzers verwendet wird.
- `become_flags` : Login-Parameter ändern. Wird meistens verwendet, wenn Sie zu einem Systembenutzer ohne Shell-Berechtigungen wechseln möchten.

Examples

Nur in einer Aufgabe

```
- name: Run script as foo user
  command: bash.sh
  become: true
  become_user: foo
```

Führen Sie alle Rollenaufgaben als root aus

```
- hosts: all
  become: true

- name: Start apache
  service: apache2
  state: started
```

Führen Sie eine Rolle als root aus

```
- hosts: all
  roles:
    - { role: myrole, become: yes }
    - myrole2
```

Werden (Privileg-Eskalation) online lesen: https://riptutorial.com/de/ansible/topic/8328/werden--privileg-eskalation-

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Ansible	activatedgeek, Alex, baptistemm, calvinmclean, Community, Jake Amey, jasonz, jscott, Michael Duffy, mrtuovinen, Pants, PumpkinSeed, tedder42, thisguy123, ydaetskcoR
2	Ansible Group Vars	Nick, Peter Mortensen
3	Ansible Gruppenvariablen	mrtuovinen
4	Ansible installieren MySQL	Fernando
5	Ansible mit OpenStack verwenden	BANANENMANNFRAU, Sebastien Josset
6	Ansible: Looping	calvinmclean
7	Ansible: Loops und Bedingungen	A K, Chu-Siang Lai, Jordan Anderson, marx, Mike, mrtuovinen, Nick, Rob H, wolfaviators
8	Ansible-Architektur	Jordan Anderson, Yogesh Darji
9	Dynamisches Inventar	mrtuovinen
10	Einführung in Playbooks	32cupo, Abdelaziz Dabebi, ydaetskcoR
11	Galaxis	mrtuovinen, ydaetskcoR
12	Geheime Verschlüsselung	fishi
13	Installation	ca2longoria, Jake Amey, Michael Duffy, mrtuovinen, Nick, PumpkinSeed, Raj, tedder42, ydaetskcoR
14	Inventar	calvinmclean, mrtuovinen, Nick
15	Rollen	Chu-Siang Lai, fishi, mrtuovinen, winston, ydaetskcoR
16	Schleifen	marx, mrtuovinen

17	So erstellen Sie einen DreamHost Cloud Server aus einem Ansible Playbook	Stefano Maffulli
18	Verwendung von Ansible mit Amazon Web Services	Abdelaziz Dabebi, another geek, ydaetskcoR
19	Werden (Privileg-Eskalation)	Jordan Anderson, Willian Paixao