**Intro to Lookups: Getting File Contents**

The file lookup is the most basic lookup type.

Contents can be read off the filesystem as follows:

---

- hosts: all

vars:

contents: "**{{** lookup**(**'file'**,** '/etc/file.txt'**)** **}}**"

tasks:

- debug: msg="the value of file.txt is **{{** contents **}}**"

## More Lookups

Various lookup plugins allow additional ways to iterate over data. In Loops you will learn how to use them to walk over collections of numerous types. However, they can also be used to pull in data from remote sources, such as shell commands or even key value stores. This section will cover lookup plugins in this capacity.

Here are some examples:

---

- hosts: all

tasks:

- debug: msg="**{{** lookup**(**'env'**,**'HOME'**)** **}}** is an environment variable"

- name: lines will iterate over each line from stdout of a command

debug: msg="**{{** item **}}** is a line from the result of this command"

with\_lines: cat /etc/motd

- debug: msg="**{{** lookup**(**'pipe'**,**'date'**)** **}}** is the raw result of running this command"

- name: Always use quote filter to make sure your variables are safe to use with shell

debug: msg="**{{** lookup**(**'pipe'**,**'getent ' **+** myuser**|quote** **)** **}}**"

- name: Quote variables with\_lines also as it executes shell

debug: msg="**{{** item **}}** is a line from myfile"

with\_lines: "cat **{{**myfile**|quote}}**"

- name: redis\_kv lookup requires the Python redis package

debug: msg="**{{** lookup**(**'redis\_kv'**,** 'redis://localhost:6379,somekey'**)** **}}** is value in Redis for somekey"

- name: dnstxt lookup requires the Python dnspython package

debug: msg="**{{** lookup**(**'dnstxt'**,** 'example.com'**)** **}}** is a DNS TXT record for example.com"

- debug: msg="**{{** lookup**(**'template'**,** './some\_template.j2'**)** **}}** is a value from evaluation of this template"

*# Since 2.4, you can pass in variables during evaluation*

- debug: msg="**{{** lookup**(**'template'**,** './some\_template.j2'**,** template\_vars**=**dict**(**x**=**42**))** **}}** is evaluated with x=42"

- name: loading a json file from a template as a string

debug: msg="**{{** lookup**(**'template'**,** './some\_json.json.j2'**,** convert\_data**=False)** **}}** is a value from evaluation of this template"

- debug: msg="**{{** lookup**(**'etcd'**,** 'foo'**)** **}}** is a value from a locally running etcd"

*# shelvefile lookup retrieves a string value corresponding to a key inside a Python shelve file*

- debug: msg="**{{** lookup**(**'shelvefile'**,** 'file=path\_to\_some\_shelve\_file.db key=key\_to\_retrieve'**)** **}}**

*# The following lookups were added in 1.9*

*# url lookup splits lines by default, an option to disable this was added in 2.4*

- debug: msg="**{{**item**}}**"

with\_url:

- 'https://github.com/gremlin.keys'

*# outputs the cartesian product of the supplied lists*

- debug: msg="**{{**item**}}**"

with\_cartesian:

- "**{{**list1**}}**"

- "**{{**list2**}}**"

- [1,2,3,4,5,6]

- name: Added in 2.3 allows using the system's keyring

debug: msg=**{{**lookup**(**'keyring'**,**'myservice myuser'**)}}**

As an alternative, you can also assign lookup plugins to variables or use them elsewhere. These macros are evaluated each time they are used in a task (or template):

vars:

motd\_value: "**{{** lookup**(**'file'**,** '/etc/motd'**)** **}}**"

tasks:

- debug: msg="motd value is **{{** motd\_value **}}**"

## User Create

## Examples

*# Add the user 'johnd' with a specific uid and a primary group of 'admin'*

- user:

name: johnd

comment: "John Doe"

uid: 1040

group: admin

*# Add the user 'james' with a bash shell, appending the group 'admins' and 'developers' to the user's groups*

- user:

name: james

shell: /bin/bash

groups: admins,developers

append: yes

*# Remove the user 'johnd'*

- user:

name: johnd

state: absent

remove: yes

*# Create a 2048-bit SSH key for user jsmith in ~jsmith/.ssh/id\_rsa*

- user:

name: jsmith

generate\_ssh\_key: yes

ssh\_key\_bits: 2048

ssh\_key\_file: .ssh/id\_rsa

*# added a consultant whose account you want to expire*

- user:

name: james18

shell: /bin/zsh

groups: developers

expires: 1422403387

## Users.yml

|  |
| --- |
| --- |
|  | users: |
|  | - username: maruna |
|  | use\_sudo: no |
|  |  |
|  | - username: jaroslav |
|  | use\_sudo: yes |

## CreateUser.yml

|  |
| --- |
| --- |
|  | - hosts: all |
|  | user: root |
|  |  |
|  | # vars: |
|  | # users: |
|  | # - username: galya |
|  | # use\_sudo: no |
|  |  |
|  | # - username: kolya |
|  | # use\_sudo: yes |
|  |  |
|  | tasks: |
|  | - include\_vars: users.yml |
|  |  |
|  | - name: Add users | create users, shell, home dirs |
|  | user: name={{ item.username }} shell=/bin/bash createhome=yes comment='create with ansible' |
|  | with\_items: '{{users}}' |
|  |  |
|  | - name: Setup | authorized key upload |
|  | authorized\_key: user={{ item.username }} |
|  | key="{{ lookup('file', 'pub\_keys/{{ item.username }}.pub') }}" |
|  | # path='/home/{{ item.username }}/.ssh/authorized\_keys' |
|  | # manage\_dir=no |
|  | with\_items: '{{users}}' |
|  |  |
|  | - name: Sudoers | update sudoers file and validate |
|  | lineinfile: "dest=/etc/sudoers |
|  | insertafter=EOF |
|  | line='{{ item.username }} ALL=(ALL) NOPASSWD: ALL' |
|  | regexp='^{{ item.username }} .\*' |
|  | state=present" |
|  | when: '{{ item.use\_sudo }} == True' |
|  | with\_items: '{{users}}' |

## Install Apache

---

- hosts: apache

sudo: yes

tasks:

- name: install apache2

apt: name=apache2 update\_cache=yes state=latest

- name: enabled mod\_rewrite

apache2\_module: name=rewrite state=present

notify:

- restart apache2

handlers:

- name: restart apache2

service: name=apache2 state=restarted

## Install nginx

|  |
| --- |
| --- |
|  |  |
|  | ######################################## |
|  | ## Ansible Playbook for installing Nginx |
|  | ######################################## |
|  |  |
|  | - name: check registered the repository of nginx-release |
|  | shell: rpm -qa | grep nginx-release |
|  | register: result |
|  | ignore\_errors: True |
|  | always\_run: yes |
|  | changed\_when: no |
|  | - name: add repository nginx-release (CentOS6/CentOS7) |
|  | yum: name="http://nginx.org/packages/centos/{{ansible\_distribution\_major\_version}}/noarch/RPMS/nginx-release-centos-{{ansible\_distribution\_major\_version}}-0.el{{ansible\_distribution\_major\_version}}.ngx.noarch.rpm" |
|  | when: result|failed |
|  |  |
|  | - name: disable the repository (pls set --enablerepo=nginx if you use it) |
|  | replace: dest=/etc/yum.repos.d/nginx.repo regexp="enabled \*= \*1" replace="enabled=0" |
|  | ignore\_errors: True |
|  |  |
|  | - name: install nginx |
|  | yum: name=nginx state=present enablerepo=nginx |
|  |  |
|  | - name: Start Nginx |
|  | service: name=nginx enabled=yes state=started |

## The Password Lookup

A special case is using /dev/null as a path. The password lookup will generate a new random password each time, but will not write it to /dev/null. This can be used when you need a password without storing it on the controller.

Generated passwords contain a random mix of upper and lowercase ASCII letters, the numbers 0-9 and punctuation (”. , : - \_”). The default length of a generated password is 20 characters. This length can be changed by passing an extra parameter:

---

- hosts: all

tasks:

- name: create a mysql user with a random password

mysql\_user:

name: "**{{** client **}}**"

password: "**{{** lookup**(**'password'**,** 'credentials/' **+** client **+** '/' **+** tier **+** '/' **+** role **+** '/mysqlpassword length=15'**)** **}}**"

priv: "**{{** client **}}**\_**{{** tier **}}**\_**{{** role **}}**.\*:ALL"

*# (...)*

Starting in version 1.4, password accepts a “chars” parameter to allow defining a custom character set in the generated passwords. It accepts comma separated list of names that are either string module attributes (ascii\_letters,digits, etc) or are used literally:

---

- hosts: all

tasks:

- name: create a mysql user with a random password using only ascii letters

mysql\_user: name=**{{** client **}}** password="**{{** lookup**(**'password'**,** '/tmp/passwordfile chars=ascii\_letters'**)** **}}**" priv=**{{** client **}}**\_**{{** tier **}}**\_**{{** role **}}**.\*:ALL

- name: create a mysql user with a random password using only digits

mysql\_user:

name: "**{{** client **}}**"

password: "**{{** lookup**(**'password'**,** '/tmp/passwordfile chars=digits'**)** **}}**"

priv: "**{{** client **}}**\_**{{** tier **}}**\_**{{** role **}}**.\*:ALL"

- name: create a mysql user with a random password using many different char sets

mysql\_user:

name: "**{{** client **}}**"

password" "**{{** lookup**(**'password'**,** '/tmp/passwordfile chars=ascii\_letters,digits,hexdigits,punctuation'**)** **}}**"

priv: "**{{** client **}}**\_**{{** tier **}}**\_**{{** role **}}**.\*:ALL"

*# (...)*

## The Passwordstore Lookup

New in version 2.3.

The passwordstore lookup enables Ansible to retrieve, create or update passwords from the passwordstore.org pass utility. It also retrieves YAML style keys stored as multilines in the passwordfile.

# Examples

Basic lookup. Fails if example/test doesn’t exist:

password="**{{** lookup**(**'passwordstore'**,** 'example/test'**)}}**"

Create pass with random 16 character password. If password exists just give the password:

password="**{{** lookup**(**'passwordstore'**,** 'example/test create=true'**)}}**"

Different size password:

password="**{{** lookup**(**'passwordstore'**,** 'example/test create=true length=42'**)}}**"

Create password and overwrite the password if it exists. As a bonus, this module includes the old password inside the pass file:

password="**{{** lookup**(**'passwordstore'**,** 'example/test create=true overwrite=true'**)}}**"

Return the value for user in the KV pair user: username:

password="**{{** lookup**(**'passwordstore'**,** 'example/test subkey=user'**)}}**"

Return the entire password file content:

password="**{{** lookup**(**'passwordstore'**,** 'example/test returnall=true'**)}}**"

**The location of the password-store directory can be specified in the following ways:**

* Default is ~/.password-store
* Can be overruled by PASSWORD\_STORE\_DIR environment variable
* Can be overruled by ‘passwordstore: path/to/.password-store’ ansible setting
* Can be overruled by ‘directory=path’ argument in the lookup call