Example:

*# Example from Ansible Playbooks*

- copy:

src: /srv/myfiles/foo.conf

dest: /etc/foo.conf

owner: foo

group: foo

mode: 0644

*# The same example as above, but using a symbolic mode equivalent to 0644*

- copy:

src: /srv/myfiles/foo.conf

dest: /etc/foo.conf

owner: foo

group: foo

mode: u=rw,g=r,o=r

*# Another symbolic mode example, adding some permissions and removing others*

- copy:

src: /srv/myfiles/foo.conf

dest: /etc/foo.conf

owner: foo

group: foo

mode: u+rw,g-wx,o-rwx

*# Copy a new "ntp.conf file into place, backing up the original if it differs from the copied version*

- copy:

src: /mine/ntp.conf

dest: /etc/ntp.conf

owner: root

group: root

mode: 0644

backup: yes

*# Copy a new "sudoers" file into place, after passing validation with visudo*

- copy:

src: /mine/sudoers

dest: /etc/sudoers

validate: /usr/sbin/visudo -cf %s

*# Copy a "sudoers" file on the remote machine for editing*

- copy:

src: /etc/sudoers

dest: /etc/sudoers.edit

remote\_src: yes

validate: /usr/sbin/visudo -cf %s

*# Create a CSV file from your complete inventory using an inline template*

- hosts: all

tasks:

- copy:

content: |

HOSTNAME;IPADDRESS;FQDN;OSNAME;OSVERSION;PROCESSOR;ARCHITECTURE;MEMORY;

**{%** **for** host **in** hostvars **%}**

**{%** **set** vars **=** hostvars**[**host**|string]** **%}**

**{{** vars.ansible\_hostname **}}**;**{{** vars.remote\_host **}}**;**{{** vars.ansible\_fqdn **}}**;**{{** vars.ansible\_distribution **}}**;**{{** vars.ansible\_distribution\_version **}}**;**{{** vars.ansible\_processor**[**1**]** **}}**;**{{** vars.ansible\_architecture **}}**;**{{** **(**vars.ansible\_memtotal\_mb**/**1024**)|round|int** **}}**; # NOQA

**{%** **endfor** **%}**

dest: /some/path/systems.csv

backup: yes

run\_once: yes

delegate\_to: localhost

Copying files from a local machine to the remote server

By default, the copy module will check the file set in the src parameter, on the local machine. And then it will copy the file to the remote machine path specified in the destpath. The example below will copy the sample.txt file in the home directory of the current user( on the local machine), to the /tmp directory on the remote server. Since we are not specifying any permission for the file, the default permission for the remote file is set as -rw-rw-r–(0664).

- hosts: blocks

tasks:

- name: Ansible copy file to remote server

copy:

src: ~/sample.txt

dest: /tmp

Note 1: If the file is already present on the remote server and if the source file’s content is different, then on running the task, the destination file will be modified. You can control this by setting the force parameter. The default is set to yes’. So it modifies the file by default. If you don’t want the file to be modified if the source file is different, then you can set it to ‘No’. The following task will only copy the file if the file does not exist on the remote server.

- hosts: blocks

tasks:

- name: Ansible copy file force

copy:

src: ~/sample.txt

dest: /tmp

force: no

Note 2: If the file couldn’t be found on the local machine, then Ansible will throw an error similar to below.  
fatal: [remote-machine-1]: FAILED! => {“changed”: false, “failed”: true, “msg”: “Unable to find ‘~/sample.txt’ in expected paths.”}

Copying a directory from a local machine to the remote server

You can also copy folders/directories using Ansible copy module. If the ‘src’ path is a directory, it will be copied recursively. That means the entire directory is copied.

Now, there are two different variations for this. Depending on whether you have the ‘/’ character at the end of the ‘src’ path or not.

The first method will **create a directory on the remote server**, with the name set in the src parameter. And then it will copy and paste the contents of the source folder to that directory. **If you want this behaviour, then don**‘**t give the ‘/’ after the path in the src parameter.**

The below Ansible copy directory example will first create a directory named  copy\_dir\_ex in the /tmp of the remote server. See how there is a ‘copy\_dir\_ex’ folder inside the ‘tmp’ folder.

- hosts: blocks

tasks:

- name: Ansible copy directory to the remote server

copy:

src:/Users/mdtutorials2/Documents/Ansible/copy\_dir\_ex

dest:/Users/mdtutorials2/Documents/Ansible/tmp

output

------

Ansible-Pro:Ansible mdtutorials2$ tree tmp

tmp

└── copy\_dir\_ex

├── file1

├── file2

├── file3

└── tmp2

├── file4

└── file5

The second method will copy only the files from the source directory to the remote server. It will ***not* create a directory** on the remote server. **If you want this behaviour, then give the ‘/’ after the path in the src parameter.**

In the below example the files inside the copy\_dir\_ex will be copied to the /tmp folder of the remote server. As you can see the src directory is not created in the destination. Only the contents of the directory are copied.

- hosts: blocks

tasks:

- name: Ansible copy files from a directory to remote server

copy:

src:/Users/mdtutorials2/Documents/Ansible/copy\_dir\_ex/

dest:/Users/mdtutorials2/Documents/Ansible/tmp

output

------

tmp/

├── file1

├── file2

├── file3

└── tmp2

├── file4

└── file5

Note 1: If we need to set the permissions of the remote directory we can do it using the **directory\_mode** parameter. The permissions will be set only if the directory does not exist on the remote machine.

Note 2: You can also set the group and owner of the directory. You should set the respective names to the ‘group’ and ‘owner’ parameters.

Copying files between different folders on the same remote machine

You can also copy files between the various locations on the remote servers. You have to set the remote\_src parameter to yes.

The following example copies the hello6 file in the /tmp directory of the remote server and pastes it in the /etc/ directory.

- hosts: blocks

tasks:

- name: Ansible copy files remote to remote

copy:

src: /tmp/hello6

dest: /etc

remote\_src: yes

Note 1: As of Ansible 2.2.1.0, copying directories in the remote servers are not supported. If you try, you will get the following error.

fatal: [remote-machine-1]: FAILED! => {"changed": false, "failed": true, "msg": "Remote copy does not support recursive copy of directory: /tmp/copy\_dir\_ex"}

Copying multiple files/directories using with\_items

If you have multiple files to be copied, then you can iterate over them using with\_items.

The following example will copy multiple files given as a list of the home directory.

- hosts: blocks

tasks:

- name: Ansible copy multiple files with\_items

copy:

src: ~/{{item}}

dest: /tmp

mode: 0774

with\_items:

['hello1','hello2','hello3','sub\_folder/hello4']

Copying multiple files with different permission/destination settings

In the above task, we are copying multiple files, but all the files have the same permissions and same destination. But sometimes we would want to set permissions for different files or maybe the destination folder will be different for each file. This is possible by using with\_items along with dictionary structure.

In the following task, I am trying to copy 3 files to 2 different folders. Also, the file permissions are different for each of them. I am giving a dictionary structure mentioning the different setting for each file.

As you can see from the output the files are copied to the given folders and the permissions are set correctly.

- hosts: all

tasks:

- name: Copy multiple files in Ansible with different permissions

copy:

src: "{{ item.src }}"

dest: "{{ item.dest }}"

mode: "{{item.mode}}"

with\_items:

- { src: '/home/mdtutorials2/test1',dest: '/tmp/devops\_system1', mode: '0777'}

- { src: '/home/mdtutorials2/test2',dest: '/tmp/devops\_system2', mode: '0707'}

- { src: '/home/mdtutorials2/test3',dest: '/tmp2/devops\_system3', mode: '0575'}

output

======

mdtutorials2@system01:~$ ls -lrt /tmp

drwxrwxrwx 2 root root 4096 Oct 9 14:28 devops\_system1

drwx---rwx 2 root root 4096 Oct 9 14:28 devops\_system2

mdtutorials2@system01:~$ ls -lrt /tmp2

-r-xrwxr-x 1 root root 0 Oct 9 14:33 devops\_system3

Copying all the files inside a folder that matches a pattern(wildcard)

If you need to copy all the files in a directory that matches a wildcard character, then you can use with\_fileglob.

In the following example, all files that start with ‘hello’ in the /tmp directory of the local machine is copied to the remote server.

- hosts: blocks

tasks:

- name: Ansible copy multiple files with wildcard matching.

copy:

src: "{{ item }}"

dest: /etc

with\_fileglob:

- /tmp/hello\*

Creating backup of a file in the remote servers before copy

While copying files, mistakes can happen. You may copy the wrong file, write wrong contents etc. This is going to create a lot of headaches. So it would be helpful if a backup of the remote file is created on the remote server.

Ansible copy module provides a ‘backup’ parameter just for that. If the remote file exists and if it is different from the file which is copied, then a new file will be created. The new file will be named by appending the timestamp and the original file name. The default value is ‘no’ for the backup parameter.

For example, the following example will create a backup of the helloworld.txt in the /tmp directory of the remote server. It will be named something like ‘helloworld.txt.8925.2017-04-05@16:53:13’.

- hosts: blocks

tasks:

- name: ansible copy file backup example

copy:

src: ~/helloworld.txt

dest: /tmp

backup: yes

Copying files using the Ad-hoc method

Most of the above tasks can be done in an Adhoc way also.

ansible blocks -m copy -a "src=~/sample.txt dest=/tmp" -s -i inventory.ini

ansible blocks -m copy -a "src=~/copy\_dir\_ex dest=/tmp" -s -i inventory.ini

ansible blocks -m copy -a "src=/tmp/hello6 dest=/tmp/hello7 remote\_src=yes" -s -i inventory.ini

Copying files from remote machine to the local machine

You can also copy the files from remote servers to the local machine. This can be done using the Ansible fetch module. This is useful when you want to copy some log files from the remote servers to your local machine.

By default, a directory named after each host you are connecting will be created in your destination directory(local machine). The fetched files will be copied there. If the file doesn’t exist on the remote server, no error will be thrown by default.

In the following example, I am running the task on remote-server-1. The file will be copied into **/etc/remote-server-1/tmp** directory of the local machine.

- hosts: blocks

tasks:

- name: Ansible fetch files from remote server to the local machine using Ansible fetch module

fetch:

src: /tmp/hello2

dest: /etc

mode: 0774

If you don’t want this behavior and you need the file to be copied directly to the destination directory, then you should use the ‘flat’ parameter.

- hosts: blocks

tasks:

- name: Ansible fetch directory example with flat parameter set

fetch:

src: /tmp/hello2

dest: /tmp/

mode: 0774

flat: yes

Note 1: If you use flat parameter and the file name is not unique the existing file will be replaced every time the file is fetched.

Note 2: If you want an error to be thrown if the source file is missing, then set the ‘fail\_on\_missing’ parameter to yes. The following example will throw an error if the remote file does not exist.

- hosts: blocks

tasks:

- name: Ansible fetch example with fail\_on\_missing set

fetch:

src: /tmp/fetch.txt

dest: /tmp/

mode: 0774

fail\_on\_missing: yes

Note 1: If you are trying to set the destination path to a directory, put a ‘\’ at the end of the path. Else Ansible will run the task as if the destination path is a file and try to replace it. You might get the following error.

fatal: [remote-machine-1]: FAILED! => {“failed”: true, “msg”: “Failed to fetch the file: [Errno 21] Is a directory: ‘/tmp'”}

Writing to a file using the copy module

You can also write to a file using the **contents** parameter in Ansible copy module. The following example will write the values given to the content parameter to the check4.txt file.

- hosts: all

tasks:

- name: Ansible write to a file example

- copy:

content: |

Content parameter example.

Check4.txt will be created after this task is executed.

dest: /Users/mdtutorials2/Documents/Ansible/check4.txt

backup: yes

Return values of copy module

The copy module returns some values for each task. The full list is available in the Ansible docs.

Example:

"changed": true,

"checksum": "98d8fb24e8b2c2cec9c5ae963bd65c3657f50b16",

"dest": "/tmp/sample.txt",

"gid": 0,

"group": "root",

"md5sum": "ce83d23d6eb6bf079e1fc5c448ea9a9f",

"mode": "0644",

"owner": "root",

"size": 13,

"src": "/home/mdtutorials2/.ansible/tmp/ansible-tmp-1489974916.02-178756727263160/source",

"state": "file",

"uid": 0