Module 1: Package management

There is a module for most popular package managers, such as DNF and APT, to enable you to install any package on a system. Functionality depends entirely on the package manager, but usually these modules can install, upgrade, downgrade, remove, and list packages. The names of relevant modules are easy to guess. For example, the DNF module is dnf_module, the old YUM module (required for Python 2 compatibility) is yum_module, while the APT module is apt_module, the Slackpkg module is slackpkg_module, and so on.

Example 1:

```
- name: install the latest version of Apache and MariaDB
  dnf:
    name:
     - httpd
      - mariadb-server
    state: latest
```

This installs the Apache web server and the MariaDB SQL database.

Example 2:

```
- name: Install a list of packages
  yum:
    name:
     - nginx
      - postgresql
      - postgresql-server
    state: present
```

This installs the list of packages and helps download multiple packages.

Module 2: Service

After installing a package, you need a module to start it. The service module enables you to start, stop, and reload installed packages; this comes in pretty handy.

Example 1:

```
- name: Start service foo, based on running process
/usr/bin/foo
  service:
    name: foo
    pattern: /usr/bin/foo
    state: started
```

This starts the service **foo**.

Example 2:

```
- name: Restart network service for interface eth0
  service:
    name: network
    state: restarted
    args: eth0
```

This restarts the network service of the interface **eth0**.

Module 3: Copy

The copy module copies a file from the local or remote machine to a location on the remote machine.

Example 1:

```
- name: 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
```

Example 2:

```
- name: Copy file with owner and permission, using
symbolic representation
  copy:
    src: /srv/myfiles/foo.conf
    dest: /etc/foo.conf
    owner: foo
    group: foo
    mode: u=rw,g=r,o=r
```

Module 4: Debug

The debug module prints statements during execution and can be useful for debugging variables or expressions without having to halt the playbook.

Example 1:

```
- name: Display all variables/facts known for a host
  debug:
    var: hostvars[inventory_hostname]
    verbosity: 4
```

This displays all the variable information for a host that is defined in the inventory file.

Example 2:

```
- name: Write some content in a file /tmp/foo.txt
  copy:
    dest: /tmp/foo.txt
    content: |
     Good Morning!
      Awesome sunshine today.
    register: display_file_content
- name: Debug display_file_content
    debug:
      var: display_file_content
      verbosity: 2
```

This registers the content of the copy module output and displays it only when you specify verbosity as 2. For example:

```
ansible-playbook demo.yaml -vv
```

Module 5: File

The file module manages the file and its properties.

- It sets attributes of files, symlinks, or directories.
- It also removes files, symlinks, or directories.

Example 1:

```
- name: Change file ownership, group and permissions
  file:
    path: /etc/foo.conf
    owner: foo
    group: foo
    mode: '0644'
```

This creates a file named **foo.conf** and sets the permission to **0644**.

Example 2:

```
- name: Create a directory if it does not exist
  file:
    path: /etc/some_directory
    state: directory
    mode: '0755'
```

This creates a directory named **some_directory** and sets the permission to **0755**.

Module 6: Lineinfile

The lineinfile module manages lines in a text file.

- It ensures a particular line is in a file or replaces an existing line using a back-referenced regular expression.
- It's primarily useful when you want to change just a single line in a file.

Example 1:

```
- name: Ensure SELinux is set to enforcing mode
  lineinfile:
    path: /etc/selinux/config
    regexp: '^SELINUX='
    line: SELINUX=enforcing
```

This sets the value of **SELINUX=enforcing**.

Example 2:

```
- name: Add a line to a file if the file does not exist,
without passing regexp
  lineinfile:
    path: /etc/resolv.conf
    line: 192.168.1.99 foo.lab.net foo
    create: yes
```

This adds an entry for the IP and hostname in the **resolv.conf** file.

Module 7: Git

The git module manages git checkouts of repositories to deploy files or software.

Example 1:

```
# Example Create git archive from repo
- git:
    repo: https://github.com/ansible/ansible-examples.git
    dest: /src/ansible-examples
    archive: /tmp/ansible-examples.zip
```

Example 2:

```
- git:
    repo: https://github.com/ansible/ansible-examples.git
    dest: /src/ansible-examples
    separate_git_dir: /src/ansible-examples.git
```

This clones a repo with a separate Git directory.

Module 8: Cli_command

The cli_command module, first available in Ansible 2.7, provides a platform-agnostic way of pushing text-based configurations to network devices over the **network_cli connection** plugin.

Example 1:

```
- name: commit with comment
  cli_config:
    config: set system host-name foo
    commit_comment: this is a test
```

This sets the hostname for a switch and exits with a commit message.

Example 2:

```
- name: configurable backup path
  cli_config:
    config: "{{ lookup('template', 'basic/config.j2') }}"
    backup: yes
    backup_options:
      filename: backup.cfg
      dir_path: /home/user
```

This backs up a config to a different destination file.

Module 9: Archive

The archive module creates a compressed archive of one or more files. By default, it assumes the compression source exists on the target.

Example 1:

```
- name: Compress directory /path/to/foo/ into
/path/to/foo.tgz
  archive:
    path: /path/to/foo
    dest: /path/to/foo.tgz
```

Example 2:

```
- name: Create a bz2 archive of multiple files, rooted at
/path
  archive:
    path:
   - /path/to/foo
    - /path/wong/foo
    dest: /path/file.tar.bz2
    format: bz2
```

Module 10: Command

One of the most basic but useful modules, the command module takes the command name followed by a list of space-delimited arguments.

Example 1:

```
- name: return motd to registered var
  command: cat /etc/motd
  register: mymotd
```

Example 2:

```
- name: Change the working directory to somedir/ and run
the command as db_owner if /path/to/database does not
exist.
  command: /usr/bin/make_database.sh db_user db_name
  become: yes
  become_user: db_owner
  args:
    chdir: somedir/
    creates: /path/to/database
```