[Skip to content](https://gist.github.com/marktheunissen/2979474" \l "start-of-content)

Top of Form



Bottom of Form

* [All gists](https://gist.github.com/discover)
* [GitHub](https://github.com/)
* [New gist](https://gist.github.com/)
* @pradeep1947

Top of Form

 Star[**1,564**](https://gist.github.com/marktheunissen/2979474/stargazers)

Bottom of Form

Top of Form

 Fork[**350**](https://gist.github.com/marktheunissen/2979474/forks)

Bottom of Form

@marktheunissen[marktheunissen](https://gist.github.com/marktheunissen)/**[pedantically\_commented\_playbook.yml](https://gist.github.com/marktheunissen/2979474)**forked from [phred/pedantically\_commented\_playbook.yml](https://gist.github.com/phred/2897937)

Created 7 years ago •

Top of Form



Bottom of Form

Embed



[**Download ZIP**](https://gist.github.com/marktheunissen/2979474/archive/bb38ddae2ed60f11dbc0af481927912e8b4433f3.zip)

[Code](https://gist.github.com/marktheunissen/2979474)[Revisions **5**](https://gist.github.com/marktheunissen/2979474/revisions)[Stars **1564**](https://gist.github.com/marktheunissen/2979474/stargazers)[Forks **350**](https://gist.github.com/marktheunissen/2979474/forks)

Insanely complete Ansible playbook, showing off all the options

[**Raw**](https://gist.github.com/marktheunissen/2979474/raw/bb38ddae2ed60f11dbc0af481927912e8b4433f3/pedantically_commented_playbook.yml)

**[pedantically\_commented\_playbook.yml](https://gist.github.com/marktheunissen/2979474" \l "file-pedantically_commented_playbook-yml)**

|  |  |
| --- | --- |
|  | --- |
|  | # ^^^ YAML documents must begin with the document separator "---" |
|  | # |
|  | #### Example docblock, I like to put a descriptive comment at the top of my |
|  | #### playbooks. |
|  | # |
|  | # Overview: Playbook to bootstrap a new host for configuration management. |
|  | # Applies to: production |
|  | # Description: |
|  | # Ensures that a host is configured for management with Ansible. |
|  | # |
|  | ########### |
|  | # |
|  | # |
|  | # Note: |
|  | # YAML, like Python, cares about whitespace. Indent consistently throughout. |
|  | # Be aware! Unlike Python, YAML refuses to allow the tab character for |
|  | # indentation, so always use spaces. |
|  | # |
|  | # Two-space indents feel comfortable to me, but do whatever you like. |
|  | # vim:ff=unix ts=2 sw=2 ai expandtab |
|  | # |
|  | # If you're new to YAML, keep in mind that YAML documents, like XML |
|  | # documents, represent a tree-like structure of nodes and text. More |
|  | # familiar with JSON? Think of YAML as a strict and more flexible JSON |
|  | # with fewer significant characters (e.g., :, "", {}, []) |
|  | # |
|  | # The curious may read more about YAML at: |
|  | # http://www.yaml.org/spec/1.2/spec.html |
|  | # |
|  |  |
|  |  |
|  | ### |
|  | # Notice the minus on the line below -- this starts the playbook's record |
|  | # in the YAML document. Only one playbook is allowed per YAML file. Indent |
|  | # the body of the playbook. |
|  | - |
|  |  |
|  | hosts: all |
|  | ########### |
|  | # Playbook attribute: hosts |
|  | # Required: yes |
|  | # Description: |
|  | # The name of a host or group of hosts that this playbook should apply to. |
|  | # |
|  | ## Example values: |
|  | # hosts: all -- applies to all hosts |
|  | # hosts: hostname -- apply ONLY to the host 'hostname' |
|  | # hosts: groupname -- apply to all hosts in groupname |
|  | # hosts: group1,group2 -- apply to hosts in group1 & group2 |
|  | # hosts: group1,host1 -- mix and match hosts |
|  | # hosts: \*.mars.nasa.gov wildcard matches work as expected |
|  | # |
|  | ## Using a variable value for 'hosts' |
|  | # |
|  | # You can, in fact, set hosts to a variable, for example: |
|  | # |
|  | # hosts: $groups -- apply to all hosts specified in the variable $groups |
|  | # |
|  | # This is handy for testing playbooks, running the same playbook against a |
|  | # staging environment before running it against production, occasional |
|  | # maintenance tasks, and other cases where you want to run the playbook |
|  | # against just a few systems rather than a whole group. |
|  | # |
|  | # If you set hosts as shown above, then you can specify which hosts to |
|  | # apply the playbook to on each run as so: |
|  | # |
|  | # ansible-playbook playbook.yml --extra-vars="groups=staging" |
|  | # |
|  | # Use --extra-vars to set $groups to any combination of groups, hostnames, |
|  | # or wildcards just like the examples in the previous section. |
|  | # |
|  |  |
|  | sudo: True |
|  | ########### |
|  | # Playbook attribute: sudo |
|  | # Default: False |
|  | # Required: no |
|  | # Description: |
|  | # If True, always use sudo to run this playbook, just like passing the |
|  | # --sudo (or -s) flag to ansible or ansible-playbook. |
|  |  |
|  | user: remoteuser |
|  | ########### |
|  | # Playbook attribute: user |
|  | # Default: "root' |
|  | # Required: no |
|  | # Description |
|  | # Remote user to execute the playbook as |
|  |  |
|  | ########### |
|  | # Playbook attribute: vars |
|  | # Default: none |
|  | # Required: no |
|  | # Description: |
|  | # Set configuration variables passed to templates & included playbooks |
|  | # and handlers. See below for examples. |
|  | vars: |
|  | color: brown |
|  |  |
|  | web: |
|  | memcache: 192.168.1.2 |
|  | httpd: apache |
|  | # Tree-like structures work as expected, but be careful to surround |
|  | # the variable name with ${} when using. |
|  | # |
|  | # For this example, ${web.memcache} and ${web.apache} are both usable |
|  | # variables. |
|  |  |
|  | ######## |
|  | # The following works in Ansible 0.5 and later, and will set $config\_path |
|  | # "/etc/ntpd.conf" as expected. |
|  | # |
|  | # In older versions, $config\_path will be set to the string "/etc/$config" |
|  | # |
|  | config: ntpd.conf |
|  | config\_path: /etc/$config |
|  |  |
|  | ######## |
|  | # Variables can be set conditionally. This is actually a tiny snippet |
|  | # of Python that will get filled in and evaluated during playbook execution. |
|  | # This expressioun should always evaluate to True or False. |
|  | # |
|  | # In this playbook, this will always evaluate to False, because 'color' |
|  | # is set to 'brown' above. |
|  | # |
|  | # When ansible interprets the following, it will first expand $color to |
|  | # 'brown' and then evaluate 'brown' == 'blue' as a Python expression. |
|  | is\_color\_blue: "'$color' == 'blue'" |
|  |  |
|  | ##### |
|  | # Builtin Variables |
|  | # |
|  | # Everything that the 'setup' module provides can be used in the |
|  | # vars section. Ansible native, Facter, and Ohai facts can all be |
|  | # used. |
|  | # |
|  | # Run the setup module to see what else you can use: |
|  | # ansible -m setup -i /path/to/hosts.ini host1 |
|  | main\_vhost: ${ansible\_fqdn} |
|  | public\_ip: ${ansible\_eth0.ipv4.address} |
|  |  |
|  | # vars\_files is better suited for distro-specific settings, however... |
|  | is\_ubuntu: "'${ansible\_distribution}' == 'ubuntu'" |
|  |  |
|  |  |
|  | ########## |
|  | # Playbook attribute: vars\_files |
|  | # Required: no |
|  | # Description: |
|  | # Specifies a list of YAML files to load variables from. |
|  | # |
|  | # Always evaluated after the 'vars' section, no matter which section |
|  | # occurs first in the playbook. Examples are below. |
|  | # |
|  | # Example YAML for a file to be included by vars\_files: |
|  | # --- |
|  | # monitored\_by: phobos.mars.nasa.gov |
|  | # fish\_sticks: "good with custard" |
|  | # # (END OF DOCUMENT) |
|  | # |
|  | # A 'vars' YAML file represents a list of variables. Don't use playbook |
|  | # YAML for a 'vars' file. |
|  | # |
|  | # Remove the indentation & comments of course, the '---' should be at |
|  | # the left margin in the variables file. |
|  | # |
|  | vars\_files: |
|  | # Include a file from this absolute path |
|  | - /srv/ansible/vars/vars\_file.yml |
|  |  |
|  | # Include a file from a path relative to this playbook |
|  | - vars/vars\_file.yml |
|  |  |
|  | # By the way, variables set in 'vars' are available here. |
|  | - vars/$hostname.yml |
|  |  |
|  | # It's also possible to pass an array of files, in which case |
|  | # Ansible will loop over the array and include the first file that |
|  | # exists. If none exist, ansible-playbook will halt with an error. |
|  | # |
|  | # An excellent way to handle platform-specific differences. |
|  | - [ vars/$platform.yml, vars/default.yml ] |
|  |  |
|  | # Files in vars\_files process in order, so later files can |
|  | # provide more specific configuration: |
|  | - [ vars/$host.yml ] |
|  |  |
|  | # Hey, but if you're doing host-specific variable files, you might |
|  | # consider setting the variable for a group in your hosts.ini and |
|  | # adding your host to that group. Just a thought. |
|  |  |
|  |  |
|  | ########## |
|  | # Playbook attribute: vars\_prompt |
|  | # Required: no |
|  | # Description: |
|  | # A list of variables that must be manually input each time this playbook |
|  | # runs. Used for sensitive data and also things like release numbers that |
|  | # vary on each deployment. Ansible always prompts for this value, even |
|  | # if it's passed in through the inventory or --extra-vars. |
|  | # |
|  | # The input won't be echoed back to the terminal. Ansible will always |
|  | # prompt for the variables in vars\_prompt, even if they're passed in via |
|  | # --extra-vars or group variables. |
|  | # |
|  | # TODO: I think that the value is supposed to show as a prompt but this |
|  | # doesn't work in the latest devel |
|  | # |
|  | vars\_prompt: |
|  | passphrase: "Please enter the passphrase for the SSL certificate" |
|  |  |
|  | # Not sensitive, but something that should vary on each playbook run. |
|  | release\_version: "Please enter a release tag" |
|  |  |
|  | ########## |
|  | # Playbook attribute: tasks |
|  | # Required: yes |
|  | # Description: |
|  | # A list of tasks to perform in this playbook. |
|  | tasks: |
|  | ########## |
|  | # The simplest task |
|  | # Each task must have a name & action. |
|  | - name: Check that the server's alive |
|  | action: ping |
|  |  |
|  | ########## |
|  | # Ansible modules do the work! |
|  | - name: Enforce permissions on /tmp/secret |
|  | action: file path=/tmp/secret mode=0600 owner=root group=root |
|  | # |
|  | # Format 'action' like above: |
|  | # <modulename> <module parameters> |
|  | # |
|  | # Test your parameters using: |
|  | # ansible -m <module> -a "<module parameters>" |
|  | # |
|  | # Documentation for the stock modules: |
|  | # http://ansible.github.com/modules.html |
|  |  |
|  | ########## |
|  | # Use variables in the task! |
|  | # |
|  | # Variables expand in both name and action |
|  | - name: Paint the server $color |
|  | action: command echo $color |
|  |  |
|  |  |
|  | ########## |
|  | # Trigger handlers when things change! |
|  | # |
|  | # Ansible detects when an action changes something. For example, the |
|  | # file permissions change, a file's content changed, a package was |
|  | # just installed (or removed), a user was created (or removed). When |
|  | # a change is detected, Ansible can optionally notify one or more |
|  | # Handlers. Handlers can take any action that a Task can. Most |
|  | # commonly they are used to restart a service when its configuration |
|  | # changes. See "Handlers" below for more about handlers. |
|  | # |
|  | # Handlers are called by their name, which is very human friendly. |
|  |  |
|  | # This will call the "Restart Apache" handler whenever 'copy' alters |
|  | # the remote httpd.conf. |
|  | - name: Update the Apache config |
|  | action: copy src=httpd.conf dest=/etc/httpd/httpd.conf |
|  | notify: Restart Apache |
|  |  |
|  | # Here's how to specify more than one handler |
|  | - name: Update our app's configuration |
|  | action: copy src=myapp.conf dest=/etc/myapp/production.conf |
|  | notify: |
|  | - Restart Apache |
|  | - Restart Redis |
|  |  |
|  | ########## |
|  | # Include tasks from another file! |
|  | # |
|  | # Ansible can include a list of tasks from another file. The included file |
|  | # must represent a list of tasks, which is different than a playbook. |
|  | # |
|  | # Task list format: |
|  | # --- |
|  | # - name: create user |
|  | # action: user name=$user color=$color |
|  | # |
|  | # - name: add user to group |
|  | # action: user name=$user groups=$group append=true |
|  | # # (END OF DOCUMENT) |
|  | # |
|  | # A 'tasks' YAML file represents a list of tasks. Don't use playbook |
|  | # YAML for a 'tasks' file. |
|  | # |
|  | # Remove the indentation & comments of course, the '---' should be at |
|  | # the left margin in the variables file. |
|  |  |
|  | # In this example $user will be 'sklar' |
|  | # and $color will be 'red' inside new\_user.yml |
|  | - include: tasks/new\_user.yml user=sklar color=red |
|  |  |
|  | # In this example $user will be 'mosh' |
|  | # and $color will be 'mauve' inside new\_user.yml |
|  | - include: tasks/new\_user.yml user=mosh color=mauve |
|  |  |
|  | # Variables expand before the include is evaluated: |
|  | - include: tasks/new\_user.yml user=chris color=$color |
|  |  |
|  |  |
|  | ########## |
|  | # Run a task on each thing in a list! |
|  | # |
|  | # Ansible provides a simple loop facility. If 'with\_items' is provided for |
|  | # a task, then the task will be run once for each item in the 'with\_items' |
|  | # list. $item changes each time through the loop. |
|  | - name: Create a file named $item in /tmp |
|  | action: command touch /tmp/$item |
|  | with\_items: |
|  | - tangerine |
|  | - lemon |
|  |  |
|  | ########## |
|  | # Choose between files or templates! |
|  | # |
|  | # Sometimes you want to choose between local files depending on the |
|  | # value of the variable. first\_available\_file checks for each file |
|  | # and, if the file exists calls the action with $item={filename}. |
|  | # |
|  | # Mostly useful for 'template' and 'copy' actions. Only examines local |
|  | # files. |
|  | # |
|  | - name: Template a file |
|  | action: template src=$item dest=/etc/myapp/foo.conf |
|  | first\_available\_file: |
|  | # ansible\_distribution will be "ubuntu", "debian", "rhel5", etc. |
|  | - templates/myapp/${ansible\_distribution}.conf |
|  |  |
|  | # If we couldn't find a distribution-specific file, use default.conf: |
|  | - templates/myapp/default.conf |
|  |  |
|  | ########## |
|  | # Conditionally execute tasks! |
|  | # |
|  | # Sometimes you only want to run an action when a under certain conditions. |
|  | # Ansible will 'only\_if' as a Python expression and will only run the |
|  | # action when the expression evaluates to True. |
|  | # |
|  | # If you're trying to run an task only when a value changes, |
|  | # consider rewriting the task as a handler and using 'notify' (see below). |
|  | # |
|  | - name: "shutdown all ubuntu" |
|  | action: command /sbin/shutdown -t now |
|  | only\_if: "$is\_ubuntu" |
|  |  |
|  | - name: "shutdown the government" |
|  | action: command /sbin/shutdown -t now |
|  | only\_if: "'$ansible\_hostname' == 'the\_government'" |
|  |  |
|  | ########## |
|  | # Notify handlers when things change! |
|  | # |
|  | # Each task can optionally have one or more handlers that get called |
|  | # when the task changes something -- creates a user, updates a file, |
|  | # etc. |
|  | # |
|  | # Handlers have human-readable names and are defined in the 'handlers' |
|  | # section of a playbook. See below for the definitions of 'Restart nginx' |
|  | # and 'Restart application' |
|  | - name: update nginx config |
|  | action: file src=nginx.conf dest=/etc/nginx/nginx.conf |
|  | notify: Restart nginx |
|  |  |
|  | - name: roll out new code |
|  | action: git repo=git://codeserver/myapp.git dest=/srv/myapp version=HEAD branch=release |
|  | notify: |
|  | - Restart nginx |
|  | - Restart application |
|  |  |
|  |  |
|  | ########## |
|  | # Run things as other users! |
|  | # |
|  | # Each task has an optional 'user' and 'sudo' flag to indicate which |
|  | # user a task should run as and whether or not to use 'sudo' to switch |
|  | # to that user. |
|  | - name: dump all postgres databases |
|  | action: pg\_dumpall -w -f /tmp/backup.psql |
|  | user: postgres |
|  | sudo: False |
|  |  |
|  | ########## |
|  | # Run things locally! |
|  | # |
|  | # Each task also has a 'connection' setting to control whether a local |
|  | # or remote connection is used. The only valid options now are 'local' |
|  | # or 'paramiko'. 'paramiko' is assumed by the command line tools. |
|  | # |
|  | # This can also be set at the top level of the playbook. |
|  | - name: create tempfile |
|  | action: dd if=/dev/urandom of=/tmp/random.txt count=100 |
|  | connection: local |
|  |  |
|  | ########## |
|  | # Playbook attribute: handlers |
|  | # Required: no |
|  | # Description: |
|  | # Handlers are tasks that run when another task has changed something. |
|  | # See above for examples. The format is exactly the same as for tasks. |
|  | # Note that if multiple tasks notify the same handler in a playbook run |
|  | # that handler will only run once. |
|  | # |
|  | # Handlers are referred to by name. They will be run in the order declared |
|  | # in the playbook. For example: if a task were to notify the |
|  | # handlers in reverse order like so: |
|  | # |
|  | # - task: touch a file |
|  | # action: file name=/tmp/lock.txt |
|  | # notify: |
|  | # - Restart application |
|  | # - Restart nginx |
|  | # |
|  | # The "Restart nginx" handler will still run before the "Restart |
|  | # application" handler because it is declared first in this playbook. |
|  | handlers: |
|  | - name: Restart nginx |
|  | action: service name=nginx state=restarted |
|  |  |
|  | # Any module can be used for the handler action |
|  | - name: Restart application |
|  | action: command /srv/myapp/restart.sh |
|  |  |
|  | # It's also possible to include handlers from another file. Structure is |
|  | # the same as a tasks file, see the tasks section above for an example. |
|  | - include: handlers/site.yml |