**Directives Glossary**

Here we list the common playbook objects and their directives. Note that not all directives affect the object itself and might just be there to be inherited by other contained objects. Aliases for the directives are not reflected here, nor are mutable ones, for example *action* in task can be substituted by the name of any module plugin.

Be aware that this reflects the ‘current development branch’ and that the keywords do not have ‘version\_added’ information.

* Play
* Role
* Block
* Task

**Play**

**accelerate**

DEPRECATED, set to True to use accelerate connection plugin.

**accelerate\_ipv6**

DEPRECATED, set to True to force accelerate plugin to use ipv6 for it’s connection.

**accelerate\_port**

DEPRECATED, set to override default port use for accelerate connection.

**always\_run**

DEPRECATED, forces a task to run even in check mode, use check\_mode directive instead.

**any\_errors\_fatal**

Force any un-handled task errors on any host to propagate to all hosts and end the play.

**become**

Boolean that controls if privilege escalation is used or not on Task execution.

**become\_flags**

A string of flag(s) to pass to the privilege escalation program when become is True.

**become\_method**

Which method of privilege escalation to use. i.e. sudo/su/etc.

**become\_user**

User that you ‘become’ after using privilege escalation, the remote/login user must have permissions to become this user.

**check\_mode**

A boolean that controls if a task is executed in ‘check’ mode

**connection**

Allows you to change the connection plugin used for tasks to execute on the target.

**diff**

Toggle to make tasks return ‘diff’ information or not.

**environment**

A dictionary that gets converted into environment vars to be provided for the task upon execution.

**fact\_path**

Set the fact path option for the fact gathering plugin controlled by gather\_facts.

**force\_handlers**

Will force notified handler execution for hosts even if they failed during the play, it will not trigger if the play itself fails.

**gather\_facts**

A boolean that controls if the play will automatically run the ‘setup’ task to gather facts for the hosts.

**gather\_subset**

Allows you to pass subset options to the fact gathering plugin controlled by gather\_facts.

**gather\_timeout**

Allows you to set the timeout for the fact gathering plugin controlled by gather\_facts.

**handlers**

A section with tasks that are treated as handlers, these won’t get executed normally, only when notified. After each section of tasks is complete.

**hosts**

A list of groups, hosts or host pattern that translates into a list of hosts that are the play’s target.

**ignore\_errors**

Boolean that allows you to ignore task failures and continue with play. It does not affect connection errors.

**max\_fail\_percentage**

can be used to abort the run after a given percentage of hosts in the current batch has failed.

**name**

It’s a name, works mostly for documentation, in the case of tasks/handlers it can be an identifier.

**no\_log**

Boolean that controls information disclosure.

**order**

Controls the sorting of hosts as they are used for executing the play. Possible values are inventory (default), sorted, reverse\_sorted, reverse\_inventory and shuffle.

**port**

Used to override the default port used in a connection.

**post\_tasks**

A list of tasks to execute after the tasks section.

**pre\_tasks**

A list of tasks to execute before roles.

**remote\_user**

User used to log into the target via the connection plugin. AKA login user.

**roles**

List of roles to be imported into the play

**run\_once**

Boolean that will bypass the host loop, forcing the task to execute on the first host available and will also apply any facts to all active hosts.

**serial**

Defines the ‘batch’ of hosts to execute the current play until the end.

**strategy**

Allows you to choose the connection plugin to use for the play.

**tags**

Tags applied to the task or included tasks, this allows selecting subsets of tasks from the command line.

**tasks**

Main list of tasks to execute in the play, they run after roles and before post\_tasks.

**vars**

Dictionary/map of variables

**vars\_files**

List of files that contain vars to include in the play.

**vars\_prompt**

list of variables to prompt for.

**vault\_password**

Secret used to decrypt vaulted files or variables.

**Role**

**always\_run**

DEPRECATED, forces a task to run even in check mode, use check\_mode directive instead.

**any\_errors\_fatal**

Force any un-handled task errors on any host to propagate to all hosts and end the play.

**become**

Boolean that controls if privilege escalation is used or not on Task execution.

**become\_flags**

A string of flag(s) to pass to the privilege escalation program when become is True.

**become\_method**

Which method of privilege escalation to use. i.e. sudo/su/etc.

**become\_user**

User that you ‘become’ after using privilege escalation, the remote/login user must have permissions to become this user.

**check\_mode**

A boolean that controls if a task is executed in ‘check’ mode

**connection**

Allows you to change the connection plugin used for tasks to execute on the target.

**delegate\_facts**

Boolean that allows you to apply facts to delegated host instead of inventory\_hostname.

**delegate\_to**

Host to execute task instead of the target (inventory\_hostname), connection vars from the delegated host will also be used for the task.

**diff**

Toggle to make tasks return ‘diff’ information or not.

**environment**

A dictionary that gets converted into environment vars to be provided for the task upon execution.

**ignore\_errors**

Boolean that allows you to ignore task failures and continue with play. It does not affect connection errors.

**no\_log**

Boolean that controls information disclosure.

**port**

Used to override the default port used in a connection.

**remote\_user**

User used to log into the target via the connection plugin. AKA login user.

**run\_once**

Boolean that will bypass the host loop, forcing the task to execute on the first host available and will also apply any facts to all active hosts.

**tags**

Tags applied to the task or included tasks, this allows selecting subsets of tasks from the command line.

**vars**

Dictionary/map of variables

**when**

Conditional expression, determines if an iteration of a task is run or not.

**Block**

**always**

List of tasks, in a block, that execute no matter if there is an error in the block or not.

**always\_run**

DEPRECATED, forces a task to run even in check mode, use check\_mode directive instead.

**any\_errors\_fatal**

Force any un-handled task errors on any host to propagate to all hosts and end the play.

**become**

Boolean that controls if privilege escalation is used or not on Task execution.

**become\_flags**

A string of flag(s) to pass to the privilege escalation program when become is True.

**become\_method**

Which method of privilege escalation to use. i.e. sudo/su/etc.

**become\_user**

User that you ‘become’ after using privilege escalation, the remote/login user must have permissions to become this user.

**block**

List of tasks in a block.

**check\_mode**

A boolean that controls if a task is executed in ‘check’ mode

**connection**

Allows you to change the connection plugin used for tasks to execute on the target.

**delegate\_facts**

Boolean that allows you to apply facts to delegated host instead of inventory\_hostname.

**delegate\_to**

Host to execute task instead of the target (inventory\_hostname), connection vars from the delegated host will also be used for the task.

**diff**

Toggle to make tasks return ‘diff’ information or not.

**environment**

A dictionary that gets converted into environment vars to be provided for the task upon execution.

**ignore\_errors**

Boolean that allows you to ignore task failures and continue with play. It does not affect connection errors.

**name**

It’s a name, works mostly for documentation, in the case of tasks/handlers it can be an identifier.

**no\_log**

Boolean that controls information disclosure.

**port**

Used to override the default port used in a connection.

**remote\_user**

User used to log into the target via the connection plugin. AKA login user.

**rescue**

List of tasks in a block that run if there is a task error in the main block list.

**run\_once**

Boolean that will bypass the host loop, forcing the task to execute on the first host available and will also apply any facts to all active hosts.

**tags**

Tags applied to the task or included tasks, this allows selecting subsets of tasks from the command line.

**vars**

Dictionary/map of variables

**when**

Conditional expression, determines if an iteration of a task is run or not.

**Task**

**action**

The ‘action’ to execute for a task, it normally translates into a C(module) or action plugin.

**always\_run**

DEPRECATED, forces a task to run even in check mode, use check\_mode directive instead.

**any\_errors\_fatal**

Force any un-handled task errors on any host to propagate to all hosts and end the play.

**args**

DEPRECATED, A secondary way to add arguments into a task, it takes a dictionary in which keys map to options and values .. well you get it.

**async**

Run a task asyncronouslly if the C(action) supports this.

**become**

Boolean that controls if privilege escalation is used or not on Task execution.

**become\_flags**

A string of flag(s) to pass to the privilege escalation program when become is True.

**become\_method**

Which method of privilege escalation to use. i.e. sudo/su/etc.

**become\_user**

User that you ‘become’ after using privilege escalation, the remote/login user must have permissions to become this user.

**changed\_when**

Conditional expression that overrides the task’s normal ‘changed’ status.

**check\_mode**

A boolean that controls if a task is executed in ‘check’ mode

**connection**

Allows you to change the connection plugin used for tasks to execute on the target.

**delay**

Number of seconds to delay between retries, this setting is only used in combination with until.

**delegate\_facts**

Boolean that allows you to apply facts to delegated host instead of inventory\_hostname.

**delegate\_to**

Host to execute task instead of the target (inventory\_hostname), connection vars from the delegated host will also be used for the task.

**diff**

Toggle to make tasks return ‘diff’ information or not.

**environment**

A dictionary that gets converted into environment vars to be provided for the task upon execution.

**failed\_when**

Conditional expression that overrides the task’s normal ‘failed’ status.

**ignore\_errors**

Boolean that allows you to ignore task failures and continue with play. It does not affect connection errors.

**local\_action**

Same as action but also implies delegate\_to: localhost

**loop\_control**

Several keys here allow you to modify/set loop behaviour in a task see http://docs.ansible.com/ansible/latest/playbooks\_loops.html#loop-controlfor details.

**name**

It’s a name, works mostly for documentation, in the case of tasks/handlers it can be an identifier.

**no\_log**

Boolean that controls information disclosure.

**notify**

list of handlers to notify when the task returns a ‘changed=True’ status.

**poll**

Sets the polling interval in seconds for async tasks (default 10s).

**port**

Used to override the default port used in a connection.

**register**

Name of variable that will contain task status and module return data.

**remote\_user**

User used to log into the target via the connection plugin. AKA login user.

**retries**

Number of retries before giving up in a until loop. This setting is only used in combination with until.

**run\_once**

Boolean that will bypass the host loop, forcing the task to execute on the first host available and will also apply any facts to all active hosts.

**tags**

Tags applied to the task or included tasks, this allows selecting subsets of tasks from the command line.

**until**

This keyword implies a ‘retries loop’ that will go on until the condition supplied here is met or we hit the retries limit.

**vars**

Dictionary/map of variables

**when**

Conditional expression, determines if an iteration of a task is run or not.

**with\_<lookup\_plugin>**

with\_ is how loops are defined, it can use any available lookup plugin to generate the item list