Glossary

The following is a list (and re-explanation) of term definitions used elsewhere in the Ansible documentation.

Consult the documentation home page for the full documentation and to see the terms in context, but this should be a good resource to check your knowledge of Ansible's components and understand how they fit together. It's something you might wish to read for review or when a term comes up on the mailing list.

Action

An action is a part of a task that specifies which of the modules to run and the arguments to pass to that module. Each task can have only one action, but it may also have other parameters

also have other parameters.

Ad Hoc

Refers to running Ansible to perform some quick command, using /usr/bin/ansible, rather than the orchestration language, which is /usr/bin/ansible-playbook. An example of an ad-hoc command might be rebooting 50 machines in your infrastructure. Anything you can do ad-hoc can be accomplished by writing a playbook, and playbooks can also glue lots of other operations together.

Async

Refers to a task that is configured to run in the background rather than waiting for completion. If you have a long process that would run longer than the SSH timeout, it would make sense to launch that task in async mode. Async modes can poll for completion every so many seconds, or can be configured to “fire and forget” in which case Ansible will not even check on the task again, it will just kick it off and proceed to future steps. Async modes work with both /usr/bin/ansible and /usr/bin/ansible-playbook.

Callback Plugin

Refers to some user-written code that can intercept results from Ansible and do something with them. Some supplied examples in the GitHub project perform custom logging, send email, or even play sound effects.

Check Mode

Refers to running Ansible with the `--check` option, which does not make any changes on the remote systems, but only outputs the changes that might occur if the command ran without this flag. This is analogous to so-called “dry run” modes

in other systems, though the user should be warned that this does not take into account unexpected command failures or cascade effects (which is true of similar modes in other systems). Use this to get an idea of what might happen, but it is not a substitute for a good staging environment.

Connection Type, Connection Plugin

By default, Ansible talks to remote machines through pluggable libraries. Ansible supports native OpenSSH ('ssh'), or a Python implementation called 'paramiko'. OpenSSH is preferred if you are using a recent version, and also enables some features like Kerberos and jump hosts. This is covered in the getting started section. There are also other connection types like 'accelerate' mode, which must be bootstrapped over one of the SSH-based connection types but is very fast, and local mode, which acts on the local system. Users can also write their own connection plugins.

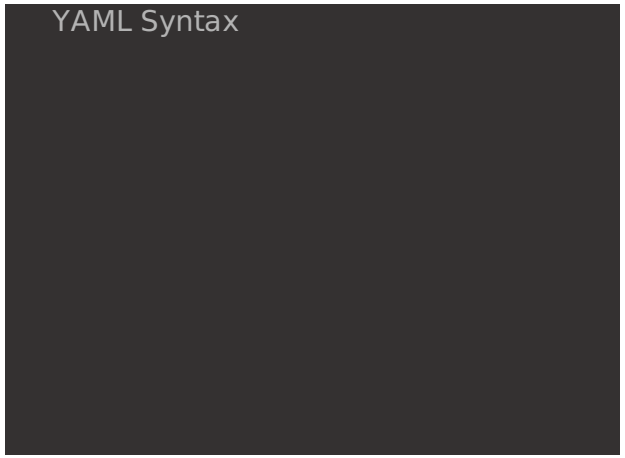
Conditionals

A conditional is an expression that evaluates to true or false that decides whether a given task will be executed on a given machine or not. Ansible's conditionals are powered by the 'when' statement, and are discussed in the playbook documentation.

Diff Mode

A `--diff` flag can be passed to Ansible to show how template files change when they are overwritten, or how they might change when used with `--check` mode. These diffs come out in unified diff format.

Facts



Facts are simply things that are discovered about remote nodes. While they can be used in playbooks and templates just like variables, facts are things that are inferred, rather than set. Facts are automatically discovered by Ansible when running plays by executing the internal ‘setup’ module on the remote nodes. You never have to call the setup module explicitly, it just runs, but it can be disabled to save time if it is not needed. For the convenience of users who are switching from other configuration management systems, the fact module will also pull in facts from the ‘ohai’ and ‘facter’ tools if they are installed, which are fact libraries from Chef and Puppet, respectively.

Filter Plugin

A filter plugin is something that most users will never need to understand. These allow for the creation of new Jinja2 filters, which are more or less only of use to people who know what Jinja2 filters are. If you need them, you can learn how to write them in the API docs section.

Forks

Ansible talks to remote nodes in parallel and the level of parallelism can be set either by passing `--forks`, or editing the default in a configuration file. The default is a very conservative 5 forks, though if you have a lot of RAM, you can easily set this to a value like 50 for increased parallelism.

Gather Facts (Boolean)

Facts are mentioned above. Sometimes when running a multi-play playbook, it is desirable to have some plays that don’t bother with fact computation if they aren’t going to need to utilize any of these values. Setting *gather_facts: False* on a playbook allows this implicit fact gathering to be skipped.

Globbing

Globbing is a way to select lots of hosts based on wildcards, rather than the name of the host specifically, or the name of the group they are in. For instance, it is possible to select “www*” to match all hosts starting with “www”. This concept is pulled directly from Func, one of Michael’s earlier projects. In addition to basic globbing, various set operations are also possible, such as ‘hosts in this group and not in another group’, and so on.

Group

A group consists of several hosts assigned to a pool that can be conveniently targeted together, and also given variables that they share in common.

Group Vars

The “group_vars/” files are files that live in a directory alongside an inventory file, with an optional filename named after each group. This is a convenient place to put variables that will be provided to a given group, especially complex data structures, so that these variables do not have to be embedded in the inventory file or playbook.

Handlers

Handlers are just like regular tasks in an Ansible playbook (see Tasks), but are only run if the Task contains a “notify” directive and also indicates that it changed something. For example, if a config file is changed then the task referencing the config file templating operation may notify a service restart handler. This means services can be bounced only if they need to be restarted. Handlers can be used for things other than service restarts, but service restarts are the most common

usage.

Host

A host is simply a remote machine that Ansible manages. They can have individual variables assigned to them, and can also be organized in groups. All hosts have a name they can be reached at (which is either an IP address or a domain name) and optionally a port number if they are not to be accessed on the default SSH port.

Host Specifier

Each Play in Ansible maps a series of tasks (which define the role, purpose, or orders of a system) to a set of systems.

This “hosts:” directive in each play is often called the hosts specifier.

It may select one system, many systems, one or more groups, or even some hosts that are in one group and explicitly not in another.

Host Vars

Just like “Group Vars”, a directory alongside the inventory file named “host_vars/” can contain a file named after each hostname in the inventory file, in YAML format. This provides a convenient place to assign variables to the host without having to embed them in the inventory file. The Host Vars file can also be used to define complex data structures that can’t be represented in the inventory file.

Lazy Evaluation

In general, Ansible evaluates any variables in playbook content at the last possible

second, which means that if you define a data structure that data structure itself can define variable values within it, and everything “just works” as you would expect. This also means variable strings can include other variables inside of those strings.

Lookup Plugin

A lookup plugin is a way to get data into Ansible from the outside world. These are how such things as “with_items”, a basic looping plugin, are implemented, but there are also lookup plugins like “with_file” which loads data from a file, and even ones for querying environment variables, DNS text records, or key value stores. Lookup plugins can also be accessed in templates, e.g., `{{ lookup('file','/path/to/file') }}`.

Multi-Tier

The concept that IT systems are not managed one system at a time, but by interactions between multiple systems, and groups of systems, in well defined orders. For instance, a web server may need to be updated before a database server, and pieces on the web server may need to be updated after *THAT* database server, and various load balancers and monitoring servers may need to be contacted. Ansible models entire IT topologies and workflows rather than looking at configuration from a “one system at a time” perspective.

Idempotency

The concept that change commands should only be applied when they need to be applied, and that it is better to describe the desired state of a system than the process of how to get to that state. As an analogy, the path from North Carolina in the United States to California involves driving a very long way West, but if I

were instead in Anchorage, Alaska, driving a long way west is no longer the right way to get to California. Ansible's Resources like you to say "put me in California" and then decide how to get there. If you were already in California, nothing needs to happen, and it will let you know it didn't need to change anything.

Includes

The idea that playbook files (which are nothing more than lists of plays) can include other lists of plays, and task lists can externalize lists of tasks in other files, and similarly with handlers. Includes can be parameterized, which means that the loaded file can pass variables. For instance, an included play for setting up a WordPress blog may take a parameter called "user" and that play could be included more than once to create a blog for both "alice" and "bob".

Inventory

A file (by default, Ansible uses a simple INI format) that describes Hosts and Groups in Ansible. Inventory can also be provided via an "Inventory Script" (sometimes called an "External Inventory Script").

Inventory Script

A very simple program (or a complicated one) that looks up hosts, group membership for hosts, and variable information from an external resource – whether that be a SQL database, a CMDB solution, or something like LDAP. This concept was adapted from Puppet (where it is called an "External Nodes Classifier") and works more or less exactly the same way.

Jinja2

Jinja2 is the preferred templating language of Ansible's template module. It is a

very simple Python template language that is generally readable and easy to write.

JSON

Ansible uses JSON for return data from remote modules. This allows modules to be written in any language, not just Python.

Library

A collection of modules made available to /usr/bin/ansible or an Ansible playbook.

Limit Groups

By passing `--limit somegroup` to ansible or ansible-playbook, the commands can be limited to a subset of hosts. For instance, this can be used to run a playbook that normally targets an entire set of servers to one particular server.

Local Connection

By using “connection: local” in a playbook, or passing “-c local” to /usr/bin/ansible, this indicates that we are managing the local host and not a remote machine.

Local Action

A local_action directive in a playbook targeting remote machines means that the given step will actually occur on the local machine, but that the variable ‘{{ ansible_hostname }}’ can be passed in to reference the remote hostname being referred to in that step. This can be used to trigger, for example, an rsync operation.

Loops

Generally, Ansible is not a programming language. It prefers to be more declarative, though various constructs like “with_items” allow a particular task to be repeated for multiple items in a list. Certain modules, like yum and apt, are actually optimized for this, and can install all packages given in those lists within a single transaction, dramatically speeding up total time to configuration.

Modules

Modules are the units of work that Ansible ships out to remote machines. Modules are kicked off by either /usr/bin/ansible or /usr/bin/ansible-playbook (where multiple tasks use lots of different modules in conjunction). Modules can be implemented in any language, including Perl, Bash, or Ruby – but can leverage some useful communal library code if written in Python. Modules just have to return JSON or simple key=value pairs. Once modules are executed on remote machines, they are removed, so no long running daemons are used. Ansible refers to the collection of available modules as a ‘library’.

Notify

The act of a task registering a change event and informing a handler task that another action needs to be run at the end of the play. If a handler is notified by multiple tasks, it will still be run only once. Handlers are run in the order they are listed, not in the order that they are notified.

Orchestration

Many software automation systems use this word to mean different things. Ansible uses it as a conductor would conduct an orchestra. A datacenter or cloud

architecture is full of many systems, playing many parts – web servers, database servers, maybe load balancers, monitoring systems, continuous integration systems, etc. In performing any process, it is necessary to touch systems in particular orders, often to simulate rolling updates or to deploy software correctly. Some system may perform some steps, then others, then previous systems already processed may need to perform more steps. Along the way, emails may need to be sent or web services contacted. Ansible orchestration is all about modeling that kind of process.

paramiko

By default, Ansible manages machines over SSH. The library that Ansible uses by default to do this is a Python-powered library called paramiko. The paramiko library is generally fast and easy to manage, though users desiring Kerberos or Jump Host support may wish to switch to a native SSH binary such as OpenSSH by specifying the connection type in their playbook, or using the “-c ssh” flag.

Playbooks

Playbooks are the language by which Ansible orchestrates, configures, administers, or deploys systems. They are called playbooks partially because it’s a sports analogy, and it’s supposed to be fun using them. They aren’t workbooks :)

Plays

A playbook is a list of plays. A play is minimally a mapping between a set of hosts selected by a host specifier (usually chosen by groups, but sometimes by hostname globs) and the tasks which run on those hosts to define the role that those systems will perform. There can be one or many plays in a playbook.

Pull Mode

By default, Ansible runs in push mode, which allows it very fine-grained control over when it talks to each system. Pull mode is provided for when you would rather have nodes check in every N minutes on a particular schedule. It uses a program called ansible-pull and can also be set up (or reconfigured) using a push-mode playbook. Most Ansible users use push mode, but pull mode is included for variety and the sake of having choices.

ansible-pull works by checking configuration orders out of git on a crontab and then managing the machine locally, using the local connection plugin.

Push Mode

Push mode is the default mode of Ansible. In fact, it’s not really a mode at all – it’s just how Ansible works when you aren’t thinking about it. Push mode allows Ansible to be fine-grained and conduct nodes through complex orchestration processes without waiting for them to check in.

Register Variable

The result of running any task in Ansible can be stored in a variable for use in a template or a conditional statement. The keyword used to define the variable is called ‘register’, taking its name from the idea of registers in assembly programming (though Ansible will never feel like assembly programming). There are an infinite number of variable names you can use for registration.

Resource Model

Ansible modules work in terms of resources. For instance, the file module will select a particular file and ensure that the attributes of that resource match a

particular model. As an example, we might wish to change the owner of /etc/motd to ‘root’ if it is not already set to root, or set its mode to ‘0644’ if it is not already set to ‘0644’. The resource models are ‘idempotent’ meaning change commands are not run unless needed, and Ansible will bring the system back to a desired state regardless of the actual state – rather than you having to tell it how to get to the state.

Roles

Roles are units of organization in Ansible. Assigning a role to a group of hosts (or a set of groups, or host patterns, etc.) implies that they should implement a specific behavior. A role may include applying certain variable values, certain tasks, and certain handlers – or just one or more of these things. Because of the file structure associated with a role, roles become redistributable units that allow you to share behavior among playbooks – or even with other users.

Rolling Update

The act of addressing a number of nodes in a group N at a time to avoid updating them all at once and bringing the system offline. For instance, in a web topology of 500 nodes handling very large volume, it may be reasonable to update 10 or 20 machines at a time, moving on to the next 10 or 20 when done. The “serial:” keyword in an Ansible playbook controls the size of the rolling update pool. The default is to address the batch size all at once, so this is something that you must opt-in to. OS configuration (such as making sure config files are correct) does not typically have to use the rolling update model, but can do so if desired.

Runner

A core software component of Ansible that is the power behind /usr/bin/ansible

directly – and corresponds to the invocation of each task in a playbook. The Runner is something Ansible developers may talk about, but it's not really user land vocabulary.

Serial

See "Rolling Update".

Sudo

Ansible does not require root logins, and since it's daemonless, definitely does not require root level daemons (which can be a security concern in sensitive environments). Ansible can log in and perform many operations wrapped in a sudo command, and can work with both password-less and password-based sudo. Some operations that don't normally work with sudo (like scp file transfer) can be achieved with Ansible's copy, template, and fetch modules while running in sudo mode.

SSH (Native)

Native OpenSSH as an Ansible transport is specified with "-c ssh" (or a config file, or a directive in the playbook) and can be useful if wanting to login via Kerberized SSH or using SSH jump hosts, etc. In 1.2.1, 'ssh' will be used by default if the OpenSSH binary on the control machine is sufficiently new. Previously, Ansible selected 'paramiko' as a default. Using a client that supports ControlMaster and ControlPersist is recommended for maximum performance – if you don't have that and don't need Kerberos, jump hosts, or other features, paramiko is a good choice. Ansible will warn you if it doesn't detect ControlMaster/ControlPersist capability.

Tags

Ansible allows tagging resources in a playbook with arbitrary keywords, and then running only the parts of the playbook that correspond to those keywords. For instance, it is possible to have an entire OS configuration, and have certain steps labeled “ntp”, and then run just the “ntp” steps to reconfigure the time server information on a remote host.

Tasks

Playbooks exist to run tasks. Tasks combine an action (a module and its arguments) with a name and optionally some other keywords (like looping directives). Handlers are also tasks, but they are a special kind of task that do not run unless they are notified by name when a task reports an underlying change on a remote system.

Templates

Ansible can easily transfer files to remote systems, but often it is desirable to substitute variables in other files. Variables may come from the inventory file, Host Vars, Group Vars, or Facts. Templates use the Jinja2 template engine and can also include logical constructs like loops and if statements.

Transport

Ansible uses “Connection Plugins” to define types of available transports. These are simply how Ansible will reach out to managed systems. Transports included are paramiko, SSH (using OpenSSH), and local.

When

An optional conditional statement attached to a task that is used to determine if the task should run or not. If the expression following the “when:” keyword evaluates to false, the task will be ignored.

Van Halen

For no particular reason, other than the fact that Michael really likes them, all Ansible releases are codenamed after Van Halen songs. There is no preference given to David Lee Roth vs. Sammy Lee Hagar-era songs, and instrumentals are also allowed. It is unlikely that there will ever be a Jump release, but a Van Halen III codename release is possible. You never know.

Vars (Variables)

As opposed to Facts, variables are names of values (they can be simple scalar values – integers, booleans, strings) or complex ones (dictionaries/hashes, lists) that can be used in templates and playbooks. They are declared things, not things that are inferred from the remote system’s current state or nature (which is what Facts are).

YAML

Ansible does not want to force people to write programming language code to automate infrastructure, so Ansible uses YAML to define playbook configuration languages and also variable files. YAML is nice because it has a minimum of syntax and is very clean and easy for people to skim. It is a good data format for configuration files and humans, but also machine readable. Ansible’s usage of YAML stemmed from Michael’s first use of it inside of Cobbler around 2006. YAML is fairly popular in the dynamic language community and the format has libraries available for serialization in many languages (Python, Perl, Ruby, etc.).

See also

Frequently Asked Questions
Frequently asked questions

Playbooks
An introduction to playbooks

Best Practices
Best practices advice

User Mailing List
Have a question? Stop by the google group!

irc.freenode.net
#ansible IRC chat channel

Previous Next