Now ping all your nodes:

$ ansible all -m ping

Ansible will attempt to remote connect to the machines using your current user name, just like SSH would. To override the remote user name, just use the ‘-u’ parameter.

If you would like to access sudo mode, there are also flags to do that:

*# as bruce*

$ ansible all -m ping -u bruce

*# as bruce, sudoing to root*

$ ansible all -m ping -u bruce --sudo

*# as bruce, sudoing to batman*

$ ansible all -m ping -u bruce --sudo --sudo-user batman

*# With latest version of ansible `sudo` is deprecated so use become*

*# as bruce, sudoing to root*

$ ansible all -m ping -u bruce -b

*# as bruce, sudoing to batman*

$ ansible all -m ping -u bruce -b --become-user batman

(The sudo implementation is changeable in Ansible’s configuration file if you happen to want to use a sudo replacement. Flags passed to sudo (like -H) can also be set there.)

Now run a live command on all of your nodes:

$ ansible all -a "/bin/echo hello"

# [Check Mode](http://docs.ansible.com/ansible/latest/playbooks_checkmode.html#id1)

Check mode is just a simulation, and if you have steps that use conditionals that depend on the results of prior commands, it may be less useful for you. However it is great for one-node-at-time basic configuration management use cases.

Example:

ansible-playbook foo.yml –check

### Testing

Use the --syntax-check and -list-tasks options, plus the dummy inventory file to do a full syntax check, including all includes roles and task files:

ansible-playbook --syntax-check --list-tasks -i tests/ansible\_hosts ./example-playbook.yml

## [Enabling or disabling check mode for tasks](http://docs.ansible.com/ansible/latest/playbooks_checkmode.html#id2)

New in version 2.2.

Sometimes you may want to modify the check mode behavior of individual tasks. This is done via the check\_mode option, which can be added to tasks.

There are two options:

1. Force a task to **run in check mode**, even when the playbook is called **without** --check. This is called check\_mode: yes.
2. Force a task to **run in normal mode** and make changes to the system, even when the playbook is called **with** --check. This is called check\_mode: no.
3. Instead of yes/no you can use a Jinja2 expression, just like the when clause.
4. Example:
5. tasks:
6. - name: this task will make changes to the system even in check mode
7. command: /something/to/run --even-in-check-mode
8. check\_mode: no
9. - name: this task will always run under checkmode and not change the system
10. lineinfile: line="important config" dest=/path/to/myconfig.conf state=present
11. check\_mode: yes

Run Playbook

With ansible, you can use sudo or su:

- hosts: example.com

gather\_facts: False

su: yes

su\_user: root

tasks:

- shell: whoami

You could also do that from the command line:

ansible-playbook --su --su-user=root --ask-su-pass playbook.yml

[**Your first commands**](http://docs.ansible.com/ansible/latest/intro_getting_started.html#id6)

Now that you’ve installed Ansible, it’s time to get started with some basics.

Edit (or create) /etc/ansible/hosts and put one or more remote systems in it. Your public SSH key should be located in authorized\_keys on those systems:

192.0.2.50

aserver.example.org

bserver.example.org

This is an inventory file, which is also explained in greater depth here: [Inventory](http://docs.ansible.com/ansible/latest/intro_inventory.html).

We’ll assume you are using SSH keys for authentication. To set up SSH agent to avoid retyping passwords, you can do:

$ ssh-agent bash

$ ssh-add ~/.ssh/id\_rsa

(Depending on your setup, you may wish to use Ansible’s --private-key option to specify a pem file instead)

Tips

When running commands, you can specify the local server by using “localhost” or “127.0.0.1” for the server name.

Example:

$ ansible localhost -m ping -e 'ansible\_python\_interpreter="/usr/bin/env python"'

You can specify localhost explicitly by adding this to your inventory file:

localhost ansible\_connection=local ansible\_python\_interpreter="/usr/bin/env python"

[**Host Key Checking**](http://docs.ansible.com/ansible/latest/intro_getting_started.html#id7)

Ansible 1.2.1 and later have host key checking enabled by default.

If a host is reinstalled and has a different key in ‘known\_hosts’, this will result in an error message until corrected. If a host is not initially in ‘known\_hosts’ this will result in prompting for confirmation of the key, which results in an interactive experience if using Ansible, from say, cron. You might not want this.

If you understand the implications and wish to disable this behavior, you can do so by editing /etc/ansible/ansible.cfg or ~/.ansible.cfg:

[defaults]

host\_key\_checking = False

Alternatively this can be set by the [ANSIBLE\_HOST\_KEY\_CHECKING](http://docs.ansible.com/ansible/latest/config.html#envvar-ANSIBLE_HOST_KEY_CHECKING) environment variable:

$ export ANSIBLE\_HOST\_KEY\_CHECKING**=**False