"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe" startvm "DevOps\_Server\_7" --type headless

"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe" startvm "Centos68" --type headless

"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe" startvm "DevOps\_client" --type headless

"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe" list runningvms

"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe" startvm "Ubuntu" --type headless

"c:\program files\oracle\virtualbox\vboxmanage.exe" controlvm "DevOps\_Server\_7" poweroff

"c:\program files\oracle\virtualbox\vboxmanage.exe" controlvm "Centos68" poweroff

"c:\program files\oracle\virtualbox\vboxmanage.exe" controlvm "Ubuntu" poweroff

"c:\program files\oracle\virtualbox\vboxmanage.exe" controlvm "DevOps\_client" poweroff

"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe" list vms

Web Directories Listable Vulnerability ...Disable directory browsing or listing for all directories.

centos 7.4 ---server (controlserver) ---192.168.56.101

centos 7.4 ---lb01 ---192.168.56.110

centos 6.8 ---app01 ---192.168.56.120

ubuntu 16.04 ---db01 ---192.168.56.102

192.168.56.101 controlserver

192.168.56.110 lb01

192.168.56.120 app01

192.168.56.102 db01

[nodes]

host1 ansible\_user=root

host2 ansible\_user=root

...

ansible-playbook -i hosts site.yml --user <user> --ask-pass -vvvv

sudo ifconfig enp0s8 down

sudo ifconfig enp0s3 down

vagrant.exe up control;vagrant.exe up lb01;vagrant.exe up app01;vagrant.exe up app02; vagrant.exe up db01;vagrant status

/etc/ansible/hosts

/etc/ansible/ansible.cfg--global

ansible --list-hosts all (target)

ansible -i dev --list-hosts

[controlserver]

control ansible\_connection=local

ansible all -a "free -m" -u ansible

ansible all -a "hostname"

ansible all -m ping

export ANSIBLE\_- HOSTS=/etc/ansible/hosts

ansible all -s -m yum -a "name=ntp state=installed"

ansible all -s -m service -a "name=ntpd state=started enabled=yes"

ansible all -s -a "ls"

ansible all -s -a "service ntpd stop"

-s option (alias for --sudo) -k (alias for --ask-sudo-pass)

ansible all -s -a "tail /var/log/messages"

As stated in the caveats, if you want to filter the messages log with something like grep, you can’t use Ansible’s default command module, but instead, shell:

ansible all -s -m shell -a "tail /var/log/messages | \ grep ansible-command | wc -l"

ansible all -a "df -h"

ansible all -a "date"

ansible all -s -m yum -a "name=ntp state=installed"

ansible all -s -m service -a "name=ntpd state=started enabled=yes"

ansible all -s -a "service ntpd stop"

ansible all -s -a "iptables -F"

ansible all -s -a "iptables -A INPUT -s 192.168.60.0/24 -p tcp -m tcp --dport 3306 -j ACCEPT"

ansible all -s -m group -a "name=admin state=present"

ansible all -s -m user -a "name=johndoe group=admin createhome=yes"

ansible all -s -m user -a "name=johndoe state=absent remove=yes"

ansible all -m copy -a "src=/etc/hosts dest=/tmp/hosts

ansible all -s -m fetch -a "src=/etc/hosts dest=/tmp"

ansible all -m file -a "dest=/tmp/test mode=644 state=directory" ---create file

ansible all -m file -a "dest=/tmp/test state=absent"

## ansible all -s -B 3600 -a "yum -y update" -🡪 Run operations in the background

## ansible all -s -B 3600 -a "ps"

* + -B <seconds>: the maximum amount of time (in seconds) to let the job run.
  + -P <seconds>: the amount of time (in seconds) to wait between polling the servers for an updated job status.

If we leave out -P, Ansible defaults to polling every 10 seconds:

* + - 1. | success >> { "ansible\_job\_id": "763350539037",

"results\_file": "/root/.ansible\_async/763350539037", "started": 1

}

$ ansible all -m async\_status -a "jid=763350539037"

#### Fire-and-forget tasks

You may also need to run occasional long-running maintenance scripts, or other tasks that take many minutes or hours to complete, and you’d rather not babysit the task. In these cases, you can **set the -B value as high as you want (be generous, so your task will complete before Ansible kills it!), and set -P to ‘0’, so Ansible fires off the command then forgets about it:**

$ ansible multi -B 3600 -P 0 -a "/path/to/fire-and-forget-script.sh" background launch...

* + - 1. | success >> { "ansible\_job\_id": "204960925196",

"results\_file": "/root/.ansible\_async/204960925196", "started": 1

}

ansible multi -s -a "tail /var/log/messages"

As stated in the caveats, if you want to filter the messages log with something like grep, you **can’t use Ansible’s default command module, but instead, shell:**

$ ansible multi -s -m shell -a "tail /var/log/messages | \ grep ansible-command | wc -l"

🡪Run shell script on all the servers every day at 4 a.m., add the cron job with:

ansible all -s -m cron -a "name='daily-cron-all-servers' \ hour=4 job='/path/to/daily-script.sh'"

Ansible will assume \* for all values you don’t specify (valid values are day, hour, minute, month, and weekday). You could also specify special time values like reboot, yearly, or monthly using special\_time=[value].

You can also set the user the job will run under via user=[user], and create a backup of the current crontab by passing backup=yes.

ansible all -s -m cron -a "name='daily-cron-all-servers' state=**absent**"

## -🡪 Deploy a version-controlled application

First, update the git checkout to the application’s new version branch, 1.2.4, on all the app servers:

$ ansible app -s -m git -a "repo=git://example.com/path/to/repo.git \ dest=/opt/myapp update=yes version=1.2.4"

Ansible’s git module lets you specify a branch, tag, or even a specific commit with the version parameter (in this case, we chose to checkout tag 1.2.4, but if you run the command again with a branch name, like prod, Ansible will happily do that instead). To force Ansible to update the checked-out copy, we passed in update=yes. The repo and dest options should be self-explanatory.

Then, run the application’s update.sh shell script:

$ ansible app -s -a "/opt/myapp/update.sh"

#### Accelerated Mode

**Ansible’s Accelerated mode achieves greater performance for playbooks. Instead of connecting repeatedly via SSH, Ansible connects via SSH initially, then uses the AES key used in the initial connection to communicate further commands and transfers via a separate port (5099 by default, but this is configurable).**

The only extra package required to use accelerated mode is **python-keyczar,** and almost everything in normal OpenSSH mode works in Accelerated mode, with two exceptions when using sudo:

* + - * + Your sudoers file needs to have requiretty disabled (comment out the line with it, or set it per user by changing the line to **Defaults:username !requiretty).**
        + You must disable sudo passwords by setting NOPASSWD in the sudoers file.

**Accelerated mode can offer 2-4 times faster performance** (especially for things like file transfers) compared to OpenSSH, and you can enable it for a playbook by adding the option accelerate: true to your playbook, like so:

---

* hosts: all accelerate: true

...

It goes without saying, if you use accelerated mode, you need to have the port through which it communicates open in your firewall (port 5099 by default, or whatever port you set with the accelerate\_port option after accelerate).

Accelerate mode is a spiritual descendant of the now-deprecated ‘Fireball’ mode, which used a similar method for accelerating Ansible communications, **but required ZeroMQ to be installed on the controlled server (which is at odds with Ansible’s simple no-dependency, no-daemon philosophy), and didn’t work with sudo commands at all.**

#### Faster OpenSSH in Ansible 1.5+

Ansible 1.5 and later contains a very nice improvement to Ansible’s default OpenSSH implementa- tion.

**Instead of copying files, running them on the remote server, then removing them, the new method of OpenSSH transfer will send and execute commands for most Ansible modules directly over the SSH connection.**

This method of connection is only available in Ansible 1.5+, and it can be enabled by adding pipelining=True under the [ssh\_connection] section of the Ansible configuration file (ansible.cfg, which will be covered in more detail later).

The pipelining=True configuration option won’t help much unless you have removed or commented the Defaults requiretty option in /etc/sudoers. This is commented out in the default configuration for most OSes, but you might want to double-check this setting to make sure you’re getting the fastest connection possible!

**Since a stable, fast, and secure SSH connection is the heart of Ansible’s communication abilities, Ansible’s implementation of SSH has continually improved throughout the past few years—and is still improving today.**

**One thing that is universal to all of Ansible’s SSH connection methods is that Ansible uses the connection to transfer one or a few files defining a play or command to the remote server, then runs the play/command, then deletes the transferred file(s), and reports back the results.**