www.tricorind.com

TRI-COR INDUSTRIES

571-458-3824

WHAT IS ANSIBLE® AND HOW CAN IT HELP ME?

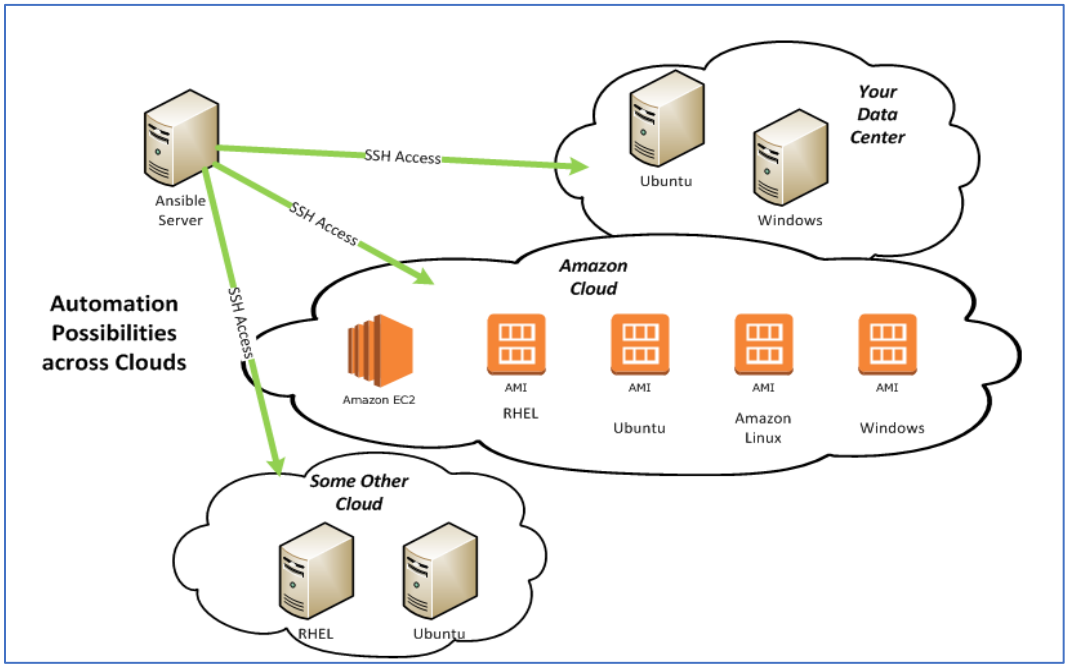
Ansible® is an industry-leading automation tool that can centrally govern and monitor disparate systems and workloads and transform and modernize enterprise IT services through the rapid deployment of applications that can scale at pace with demand. Here at TRI-COR Industries (TCI), we are focused on leveraging next-generation tools and technologies to deliver optimal operational outcomes for our customers. The following describes why we have identified Ansible® as a best-of-breed solution when performing Linux system administration tasks across multiple cloud environments.

Compelling Reasons to Use Ansible® in a DevOps Environment

- No software client needs to be installed: It is an *agentless deployment* for Linux. Ansible® uses Secure Shell (SSH) on Linux Operating Systems (OS) and Windows Remote Management (WinRM) for Windows OS connectivity.
- It is *easy to use* for someone with a systems administration or Bash scripting background.
- It can be used to automate *across multiple cloud environments* like Amazon Web Services (AWS) and/or in custom data centers due to its portability and light footprint.

Using Ansible® Across Clouds

Below is an illustration of how Ansible® can be used seamlessly across clouds. If you can setup SSH access to Linux, it is going to work for you in most cases. Note that Windows is slightly different with the reliance on WinRM. WinRM uses a different port than SSH, but the same basic principles apply. This gives DevOps teams the ability to truly manage systems across diverse cloud environments and data centers with consistency and repeatability. The principles can be used to manage network gear across clouds or data centers.



Ansible® Playbooks

Below are examples of Ansible® playbooks. The playbooks illustrate the simplicity and power of Ansible® for automating an installation, copying files, restarting a service, and error control. In the examples below, we are going to install Apache to a Red Hat Enterprise Linux (RHEL), Ubuntu, and Amazon Machine Image (AMI) of Linux. These are all different flavors of Linux with different software package managers, different commands, and different usernames. Ansible® can do it with ease.

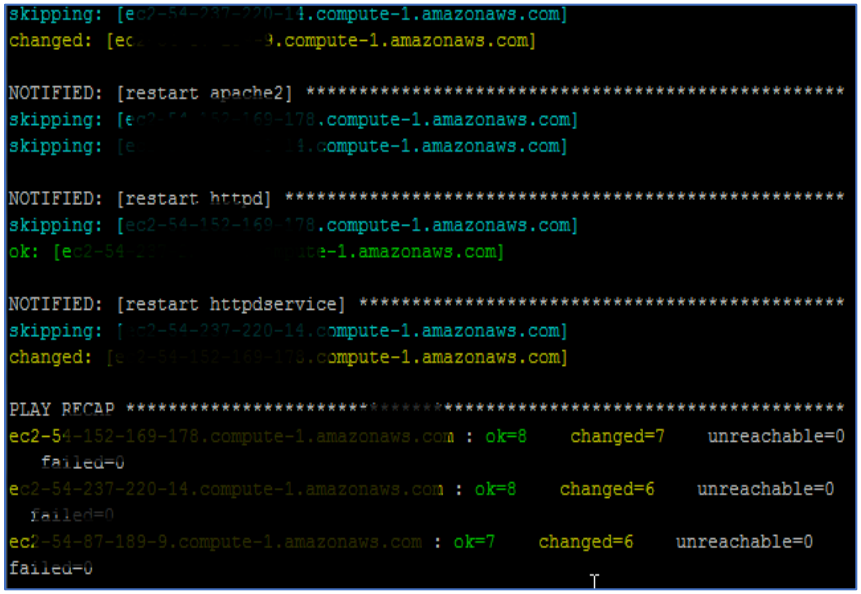
Refer to the Playbook 1 and Playbook 2 examples to see how the Linux operating system is called with Ansible® and regular shell commands. Imagine whatever task you need to perform across hundreds of servers. Next, consider how this type of activity was performed in the past *without an automation tool*. You would have to complete this activity with Bash scripts with error controls, complex checks, SCP commands, etc.

To begin, simply run the playbook on your Ansible® server with the command syntax below. This example assumes that the user has a functioning Ansible® environment and that Ubuntu, RHEL, and AWS Linux AMI have been correctly configured.

ansible-playbook <playbook name.yml>

As identified in this screenshot to the right, it will depict whether the steps are executing successfully or if changes have occurred (the final summary shows the number of changes).

Note that the dynamic IP of the instances has been blacked out. Also note that the different OS have different names for the Apache service that Ansible® correctly used.



If using AWS, the Ansible® host file will include the proper AWS username by OS for SSH:

ec2-x-x-x-x.compute-1.amazonaws.com ansible_ssh_user=ubuntu
ec2-x-x-x-x.compute-1.amazonaws.com ansible_ssh_user=ec2-user

Playbook 1 is shown below. Refer to the sections with 'when' statements for the OS type. Playbook 1 is in YAML Ain't Markup Language (YAML) format, so *indents are critical to successful execution*.

Playbook 1 = apacheinstall.yml

```
---
- hosts: lab
  vars:
    http_port: 80
    max_clients: 200
```

```
  sudo: yes
  sudo_user: root
  tasks:

  - name: ensure apache2-ubuntu is at the latest version
    apt: pkg=apache2 update_cache=yes  state=latest
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'

  - name: ensure httpd-redhat is at the latest version
    yum: pkg=httpd state=latest
    when: ansible_distribution == 'Amazon' or ansible_distribution == 'RedHat'

# make directories
  - command: sudo /bin/mkdir /var/www
    ignore_errors: yes

  - name: "Make sure index.html is present for the default virtual host"
    copy: src=files/index.html dest=/var/www/html/index.html

  - command:  sudo /usr/sbin/useradd -M -r -U www-data
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'


  - name: "Make sure Ubuntu index.html is owned by apache user = www-data"
    file: path=/var/www/html/index.html owner=www-data group=www-data
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'


  - name: "Make sure RedHat index.html is owned by apache user = apache"
    file: path=/var/www/html/index.html owner=apache group=apache
    when: ansible_distribution == 'Amazon' or ansible_distribution == 'RedHat'


    notify:
    - restart apache2
    - restart httpd
    - restart httpdservice

  - name: ensure httpd is running (and enable it at boot)
    service: name=httpd enabled=yes
    when: ansible_distribution == 'Amazon' or ansible_distribution == 'RedHat'

  - name: ensure httpd.service is running
    service: name=httpd state=started
    when: ansible_distribution == 'Amazon' or ansible_distribution == 'RedHat'

  - name: ensure apache is running (and enable it at boot)
    service: name=apache2 state=started enabled=yes
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'


  handlers:
    - name: restart apache2
      service: name=apache2 state=started
```

```
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'

  - name: restart httpd
    service: name=httpd state=started
    when: ansible_distribution == 'Amazon'


  - name: restart httpdservice
    service: name=httpd.service state=restarted
    when: ansible_distribution == 'RedHat'
```

Note that if you want to practice running Playbook 1 several times to see the process, you should have another playbook to "reset" your servers to remove Apache installations. Playbook 2 is a remove step and is shown below. This playbook is in YAML format as well, so indents are critical to successful execution. Refer to the sections with 'when' statements for the OS type.

Playbook 2 = apacheremove.yml

```
---
- hosts: lab
  vars:
    http_port: 80
    max_clients: 200
  sudo: yes
#  remote_user: root
  tasks:

  - name: ensure apache is at the absent version
    apt: pkg=apache2 update_cache=yes  state=absent purge=yes
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'

  - name: ensure apache is at the absent version
    yum: pkg=httpd state=absent
    when: ansible_distribution == 'Amazon' or ansible_distribution == 'RedHat'

# remove the file directories
  - command: /bin/rm -rf /var/www
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu' or ansible_distribution
== 'Amazon'
    ignore_errors: yes

  - command: /usr/bin/rm -rf /var/www
    when: ansible_distribution == 'Amazon' or ansible_distribution == 'RedHat'
    ignore_errors: yes

# remove the users
  - command: /sbin/userdel apache
    when: ansible_distribution == 'Redhat'
    ignore_errors: yes

  - command: /sbin/groupdel apache
    when: ansible_distribution == 'Redhat'
    ignore_errors: yes

  - command: /usr/sbin/userdel apache
    when: ansible_distribution == 'Amazon'
    ignore_errors: yes
```

```
# remove the groups
  - command: /usr/sbin/groupdel apache
    when: ansible_distribution == 'Amazon'
    ignore_errors: yes

  - command: /usr/sbin/userdel www-data
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'
    ignore_errors: yes

  - command: /usr/sbin/groupdel www-data
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'
    ignore_errors: yes


 handlers:
   - name: restart apache
     service: name=apache2 state=restarted
     when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'

   - name: restart httpd
     service: name=httpd state=restarted
     when: ansible_distribution == 'Amazon' or ansible_distribution == 'Redhat'
```

Summary

The examples above are *very* basic examples of Ansible® usage. Your playbooks will probably be longer, possibly more complex, and more appropriate for your environment. Keep in mind that you can also run ad-hoc commands without playbooks. Ansible® can be scheduled with Cron or Ansible Tower® to do your bidding as needed.

Ansible® offers comprehensive user documentation and it is recommended that it is reviewed. You can start your research into Ansible® here: https://www.ansible.com/get-started.

About the Author

Kevin Cox is a Solutions Architect for TCI. His role at TCI is to help clients leverage skills and technology for the best delivery of IT services. He offers a wide-range of diverse experience and expertise for clients. Kevin has worked in IT as a Sysadmin, DBA, Architect, and in various leadership roles.