# نصب ANSIBLE

## استفاده از دستور yum (در صورت نیاز)

برای این کار لازم است که سیدی iso نصب لینوکس را mount کنیم و بعد تمامی package های انرا به ادرس /home/rpms کپی کنیم و به ادرس فوق رفته و سپس دستور زیر را اجرا می کنیم.

rpm -ivh createrepo-0.9.9-22.el6.noarch.rpm python-deltarpm-3.5-0.5.20090913git.el6.x86\_64.rpm deltarpm-3.5-0.5.20090913git.el6.x86\_64.rpm

با اجرای دستور بالا پکیج createrpository نصب می شود.

به ادرس /etc/yum.repos.d/ رفته و فایل myrepo.repo را با محتویات زیر میسازیم

[myrepo]

name=My Local Repo

baseurl=file:///home/rpms

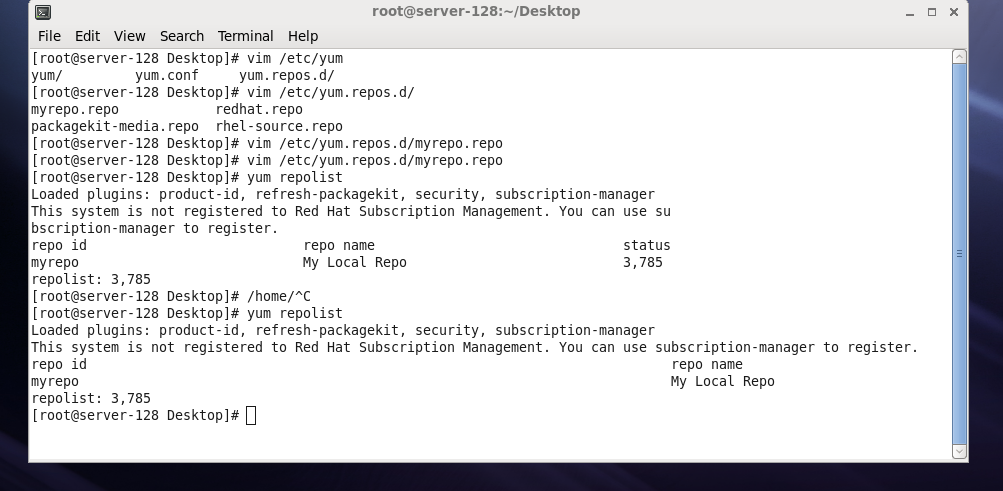
enabled=1

gpgcheck=0

سپس باید دستور زیر را اجرا کنیم

createrepo -p /home/rpms/

سپس با اجرای دستور yum repolist باید خروجی شبیه به این بگیریم:



باید به مسیر زیر رفته و lib های موجود در این فولدر را به مسیر repository خودمان یعنی /home/rpms کپی می کنیم .

\\file.dotin.ir\Teams\Core Operation\GENERAL-OPERATION\ansible\ansible-lib

حالا دستور زیر را اجرا می کنیم:

yum install ansible-2.7.5-1.el6.ans.noarch.rpm

بعد از اتمام مراحل نصب دستور زیر باید خروجی مناسب داشته باشد:

Ansible --version

حالا ansible روی سیستم ما نصب می باشد.

برروی یکی از سرورهایی که ansible قبلا نصب شده است رفته و دستور زیر را اجرا می کنیم و سپس فایل تولید شده که حاوی تنظیمات ansible می باشد را به روی سروری که میخواهیم راه اندازی کنیم منتقل می کنیم.

tar -czvf /etc/ansible-back.tar.gz /etc/ansible/

تنظیمات را به پوشه مد نظر کپی می کنیم.

فقط باید فایل /etc/ansible/hosts را درست کنیم و ای پی های سرورهای مد نظر را وارد کنیم.

برای سروری که میخواهیم validation از روی ان سرور انجام شود script زیر را کپی میکنیم.

/appserver/crontabjobs/sync-source/validation./validate-source.sh

و برروی همه کلاینتها اسکریپت زیر را کپی می کنیم:

/appserver/crontabjobs/sync-source/create-source-md5sum.sh

برای rotate شدن لاگهای انسیبل دو فایل زیر را با محتویات زیر درست می کنیم.

1. vim /etc/logrotate.d/ansible

/var/log/ansible/\*.log

{

create 775 root root

extension log

rotate 1000

size 400M

missingok

notifempty

compress

delaycompress

sharedscripts

dateext

}

1. vim /etc/logrotate.d/source-validation

/appserver/crontabjobs/sync-source/validate/result/\*.log

{

create 775 root root

extension log

rotate 1000

size 400M

missingok

notifempty

compress

delaycompress

sharedscripts

dateext

}

برای تست کردن اینکه logrotate ایجاد شده درست کار می کند یا خیر دستور زیر را جرا کنید اگر خطا نداد مشکلی نیست.

logrotate -f /etc/logrotate.d/ansible

برای پاک کردن یک rpm نصب شده ابتدا انرا با دستور زیر پیدا می کنم.

rpm -qa | grep -i webmin

و بعد ان را با دستور زیر پاک می کنیم.

rpm -e <package name>

براي upgrade کردن يک package :

rpm -Uvh package\_file.rpm

vim /etc/xinetd.d/rsync

service xinetd restart

# logging is off by default unless this path is defined

# if so defined, consider logrotate

log\_path = /var/log/ansible.log

# Ansible چیست ؟

Ansible یک ابزار متن باز قدرتمند و کارآمد برای مدیریت و پیکربندی سیستم ها ، گسترش برنامه ها ، اتوماتیک سازی و هماهنگ سازی وظایف و برخی موارد دیگر می باشد.به عبارت دیگر Ansible یک Remote Administration بسیار قوی با امکانات بسیارکارآمد می باشد و از پروتکل SSH برای برقراری ارتباط و مدیریت سیستم ها و دستگاه ها استفاده می نماید.در حال حاضر مالکیت Ansible در اختیار کمپانی Redhat هست و توسط آن اداره میشود .Ansible بر روی سیستم عامل های Red Hat Enterprise Linux (TM), CentOS, Fedora, Debian, Ubuntu قابل نصب و استفاده هست.

اهداف طراحی Ansible :

1. حداقل میزان مصرف منابع
2. امنیت
3. اعتماد پذیری بالا
4. آسان بودن نصب ، راه اندازی و کار با آن

## چرا از Ansible استفاده کنیم ؟

ابزارهای دیگری هم همانند Ansible به عنوان مثال Puppet و Chef وجود دارند.

دلایل استفاده از Ansible :

Agentless بوده و نصب به نیاز Agent بر روی سیستم و یا دیوایس نیست و تنها از پروتکل SSH برای ارتباط و مدیریت آنها استفاده می نماید اما در Puppet و Chef نیاز به نصب Agent هست.بر پایه زبان پایتون ساخته شده و از فرمت YAML که یک زبان پیکربندی ساده هست استفاده و یادگیری آن را آسان میکند اما در puppet و Chef از زبان Ruby که یادگیری آن سخت است استفاده شده است.برای مدیریت لینوکس نیاز به python و برای مدیریت ویندوز هم نیاز به powershell دارد و برای ارتباط و مدیریت هم از ssh استفاده می نماید.

## Ansible چگونه کار میکند ؟

Ansibleاز دو بخش تشکیل شده است :

Control Machine

* Node

تمامی Node ها توسط یک Control Machine از طریق SSH مدیریت خواهد شد.Ansible یک ابزار ماژولار هست و ماژول ها هم یک واحد کاری در Ansible به حساب می آیند.ماژول ها بصورت standalone اجرا میشوند و همچنین میتوان با زبان های bash ، Perl ، Python و یا ruby آنها را ایجاد کرد.تا کنون برای Ansible بیش از 700 ماژول تولید شده است.با منتشر شدن ورژن جدیدی از Ansible حداقل 20 ماژول و یا بیشتر هم منتشرخواهد شد.برای تنظیم و مدیریت Node ها Ansible ماژول ها را از طریق SSH به Node ها انتقال میدهد.ماژول ها بصورت موقت در Node ها ذخیره شده و از طریق پروتکل JSON با Control Machine با استفاده از خروجی استاندارد ارتباط برقرار میکند.زمانی که Ansible در حال مدیریت Node ها نباشد هیچ برنامه و یا پردازشی بصورت Background اجرا نشده و به همین دلیل منابعی نیز مصرف نخواهد شد.

## کامپوننت های Ansible :

Ansible از کامپوننت های زیر تشکیل شده است :

Inventories

Modules

Variables

Facts

Playbooks and plays

Configuration Files

Templates

Handlers

Roles

Ansible Vault

### Inventories :

توسط Inventory فهرستی از سیستم ها و دستگاه هایی که قرار هستند بوسیله Ansible مدیریت شوند را مشخص می نماییم .این سیستم ها و دستگاه ها با عنوان هاست در Inventory شناخته میشوند.Inventory را میتوان به دو صورت Static or Local و Dynamic مشخص نمود.

Static or Local :

در این حالت بصورت پیش فرض هاست ها درون فایل etc//ansible//hosts// مشخص میشوند.همچنین میتوان مسیر دیگری برای آن در نظر گرفت.

Dynamic :

در این حالت میتوان اسکریپتی با زبان پایتون نوشت که به عنوان مثال به سرویس ابری آمازون متصل شده و سیستم ها و دستگاه ها را فراخوانی کرده و بوسیله Ansible مدیریت نماییم.همچنین در این روش با استفاده از اسکریپت مذکور میتوان سیستم جدیدی در این سرویس ابری ایجاد و آن را فراخوانی و توسط Ansible مدیریت نماییم.

### Modules :

Ansible یک ابزار ماژولار هست و ماژول ها هم یک واحد کاری در Ansible به حساب می آیند.ماژول ها بصورت standalone اجرا میشوند و همچنین میتوان با زبان های bash ، Perl ، Python و یا ruby آنها را ایجاد کرد.ماژول ها هم بصورت Local و هم بصورت Remote اجرا میشوند.به عنوان مثال ماژولی با نام yum هست که توسط آن میتوان پکیجی را در توزیع های rpm base نصب ، حذف و یا به روز رسانی نمود.تا کنون برای Ansible بیش از 700 ماژول تولید شده است.با منتشر شدن ورژن جدیدی از Ansible حداقل 20 ماژول و یا بیشتر هم منتشرخواهد شد.

### Variables :

با تعریف متغییر در Ansible میتوان برخی روندها از جمله روند اجرا را سفارشی نمود.

### Facts :

توسط Ansible Facts میتوان اطلاعاتی اعم از نوع سیستم عامل ، IP Address ، مشخصات CPU ، میزان RAM و غیره را از هاست ها دریافت نمود.

### Playbooks and plays :

Playbooks دستورالعمل هایی است که توسط آن میتوان نحوه مدیریت هاست ها را مشخص نمود.

Playbooks از زبان YAML که یک زبان پیکربندی ساده هست برای ساختار دستورالعمل های خود استفاده می نماید.

یک Play یک Task در Playbooks هست که بصورت Standalone توسط playbooks اجرا میشوند.

مثالی از playbooks و ساختار زبان YAML :

---

- name: This is a test

hosts: centos

remote\_user: ansible

become: yes

become\_method: sudo

connection: ssh

tasks:

- name: install apache on centos

yum:

name: httpd

state: latest

…

### Configuration Files :

مسیر پیش فرض فایل کانفیگ Ansible در مسیر etc//ansible//ansible.cfg// هست که میتوان مسیر دیگری نیز مشخص نمود.Playbooks در زمان اجرای کانفیگ های Ansible را از این فایل دریافت می نماید.

### Templates :

Template یک ماژول در Ansible می باشد که به شما اجازه میدهد فایلی با متغییرهای مورد نیاز با پسوند .j2 ایجاد و در زمان اجرای این ماژول آن را جایگزین فایل کانفیگ سرویسی و یا فایل مورد نظر خود نمایید.فایل با پسوند .j2 یک فایل text ساده هست که محتوای آن میتواند کانفیگ سرویسی و یا محتوایی که ما مشخص می نماییم باشد.تمپلیت ها توسط زبان تمپلیتی jinja2 پردازش میشوند.

### Handlers :

Handlers یک Task درون Playbooks می باشد که توسط آن میتوانیم Error ها را Handle نماییم.

### Roles :

یک Roles یک Playbooks هست که از یک یا چندین Playbooks تشکیل شده است و توسط Playbooks های یک role میتوانیم عملیاتی را بر روی هاست ها انجام دهیم.به عنوان مثال میتوانیم role را برای نصب آپاچی ایجاد نماییم و هر زمان که نیاز به نصب آپاچی داریم این role را در Playbooks فراخوانی کنیم تا سرویس آپاچی برای ما نصب شود.ما میتوانیم Role ها را بصورت دستی و یا توسط مخزن Ansible Galaxy نصب و ایجاد نماییم.

### Ansible Vault :

توسط Ansible Vault میتوانیم فایل های حساس به عنوان مثال فایل های حاوی پسورد را رمزگذاری و هر زمان هم که نیاز بود آنها را رمزگشایی نماییم.در کل Ansible Vault ابزاری برای رمزگذاری و نگهداری از فایل های حساس در سیستم می باشد.

Ansible به دو روش اجرا میشود :

* Ad-hoc command
* Playbooks

Ad-Hoc command :

در این روش میتوان از دستور ansible به همراه ماژول های آن استفاده نمود به عنوان مثال :

Ansible all -s –m yum –a “name=httpd state=latest"

در مثال بالا با استفاده از دستور Ansible و ماژول yum پکیج آپاچی را بر روی تمامی هاست های مشخص شده در فایل hosts نصب خواهد کرد.

Playbooks :

در این روش میبایست فایلی را با پسوند yaml. و یا .yml ایجاد و درون آن هم از ساختار زبان YAML که یک زبان پیکربندی ساده هست استفاده نمود.همچنین درون این فایل میتوان به هر تعداد ماژول که نیاز هست استفاده نمود.در روش Ad-Hoc در هر بار اجرا میتوان یک ماژول رو اجرا نمود اما در روش playbooks همانطور که عنوان شد میتوان به هر تعداد ماژول که نیاز هست استفاده نمود.در روش playbooks همچنین میتوان مدیریت و کنترلی را بر روی اجرای ماژول ها داشته باشیم.از دستور زیر هم برای اجرای playbook استفاده می کنیم :

Ansible-playbook test.yaml

ansible all --list-hosts

با دستور –i hosts مي توانيم ليست host ها را با فايل hosts که در مسير جاري مي باشد مقداري دهي کنيم.

اکنون زمان بررسی 3 سرور با استفاده از ping از روی لوکال هاست مان میباشد. برای اجرای این منظور ما نیاز داریم که از دستور ansible به همراه آپشن -m (ماژول) و -all (گروه سرورها) استفاده کنیم.

# ansible -m ping web-servers  
  
OR  
  
# ansible -m ping -all

 ansible -m command -a "df -h" web-servers

با -a پارامترهای دستور را مشخص می کنیم.

 ansible -m shell -a "hostname" web-servers

 mkdir /etc/ansible/playbooks

نکته : در سرورهای client باید ssh , python نصب باشد.

Ansible –b

-b یعنی با روت به کلاینت وصل شو.

برای اینکه بتوانیم به یک سرور که در یک گروه خاص قرار داد اشاره کنیم باید یک نام alias برای ان بصورت زیر تعریف کنیم.

[webservers]

Webserver1 ansible\_host=10.20.11.46

برای چک کردن سرویس از دستور زیر استفاده می کنیم:

Ansible –m service –a “name=apache2 state=started” webservers

Apache2 در واقع نام سرویس در سرور مقصد می باشد.

برای stop کردن سرور ها از دستور زیر:

Ansible –m service –a “name=apache2 state=sttopped” webservers

Ansible –m service –a “name=apache2 state=started enabled=true” webservers

enabled=true یعنی در ریستارت سرور این سرویس بالا باشد.معادل chkconfig on

ماژول کپی:

Ansible –b –m copy –a “src=index.html dest=/var/www/html/” webservers

ماژول file :

Ansible –b –m file –a “path=/var/www/html/index.html state=touch” webservers

یک فایل با نام index.html می سازد.

اگر state ها را ندانستیم می توانیم دستور را خالی بدون مقدار state اجرا کنیم تا مقادیری که میپذیرد را به ما نشان دهد.

State=absent فایل را پاک کن.

Playbook :

دقت شود که فایلهای yaml به فاصله حساس هستند.

# mkdir /etc/ansible/playbooks

---

* Hosts: webserver

Tasks:

- name: comment for task

shell: uname –a > /home/name.txt

- name: comment for task

shell: whoami > /home/name.txt

- name: install apache server on servers

Apt: name=apache2 state=present update\_cache=yes

become: yes

...

Tasks ها لیستی از ماژولهای قابل اجرا می باشند: در مثلا بالا shell و apt ماژول می باشند و پارامترهای انها در مقابل انها امده است.

اجرای yaml ها:

Ansible-playbook myfile.yaml

Become: yes معادل همان –b می باشد که میگوید root شو.

--check براي dry run . يعني فقط نتيجه را نمايش مي دهد و هيچ چيز را اعمال نميکند.

Ansible-playbook myfile.yaml --check

می توانیم پارامترهای در یک سطر بزنیم که در این صورت به شکل زیر عمل می کنیم:

- service: name=comment shell=uname –a > /home/name.txt

نکته: در ansible دستورات به ترتیب اجرا می شوند ولی ترتیب سرورها همان ترتیب فایل host یا به ترتیبی که در درون دستور داده می شود نیست( زیرا که ممکن است ssh به یک سرور طول بکشد و ansible منتظر نمیماند که اتصال برقرار شود و بقیه را انجام می دهد.)

Handlers ها مانند task ها هستند ولی فقط زمانی ه فراخوانی شوند اجرا می شوند.

- name: write the apache config file

template:

src: /srv/httpd.j2

dest: /etc/httpd.conf

notify:

- restart apache

- name: ensure apache is running

service:

name: httpd

state: started

handlers:

- name: restart apache

service:

name: httpd

state: restarted

تعریف متغیر ها برای host ها:

[atlanta]

host1

host2

[atlanta:vars]

ntp\_server=ntp.atlanta.example.com

proxy=proxy.atlanta.example.com

محدود کردن اجرای یک playbook :

Ansible-playbook myfile.yaml -l Atlanta

Ansible-playbook myfile.yaml --limit Atlanta

فقط انهایی را اجرا می کند که host انها Atlanta باشد.

می توانیم همین کار را با tag ها نیز انجام دهیم( به شرطی که در تعریف task ها برای انها tag زده باشیم.)

Ansible-playbook myfile.yaml --tags Atlanta

Role ها:

رفته رفته playbook ها بزرگ می شوند و نگهداری انها سخت می شود. برای اینکار انها را به ماژولها میشکنیم تا بتوانیم انها را مدیریت کنیم.

Role ها اطلاعات مهمی را تبدیل می کنند: task ها , files ها و variables ها

Role شامل فولدر های زیر می باشد.

* tasks - contains the main list of tasks to be executed by the role.
* handlers - contains handlers, which may be used by this role or even anywhere outside this role.
* defaults - default variables for the role (see [Using Variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html) for more information).
* vars - other variables for the role (see [Using Variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html) for more information).
* files - contains files which can be deployed via this role.
* templates - contains templates which can be deployed via this role.
* meta - defines some meta data for this role. See below for more details.

برای ساختن یک role با ساختار فایلی بالا از دستور زیر استفاده کنید.

ansible-galaxy init --force --offline testrole

## [Using Roles](https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html#id6)

The classic (original) way to use roles is via the roles: option for a given play:

---

- hosts: webservers

roles:

- common

- webservers

This designates the following behaviors, for each role ‘x’:

* If roles/x/tasks/main.yml exists, tasks listed therein will be added to the play.
* If roles/x/handlers/main.yml exists, handlers listed therein will be added to the play.
* If roles/x/vars/main.yml exists, variables listed therein will be added to the play.
* If roles/x/defaults/main.yml exists, variables listed therein will be added to the play.
* If roles/x/meta/main.yml exists, any role dependencies listed therein will be added to the list of roles (1.3 and later).
* Any copy, script, template or include tasks (in the role) can reference files in roles/x/{files,templates,tasks}/ (dir depends on task) without having to path them relatively or absolutely.

When used in this manner, the order of execution for your playbook is as follows:

* Any pre\_tasks defined in the play.
* Any handlers triggered so far will be run.
* Each role listed in roles will execute in turn. Any role dependencies defined in the roles meta/main.yml will be run first, subject to tag filtering and conditionals.
* Any tasks defined in the play.
* Any handlers triggered so far will be run.
* Any post\_tasks defined in the play.
* Any handlers triggered so far will be run.

Role ها می توانند به هم وابستگی داشته باشند. یعنی اجرای منوط به اجرای یک role دیگر باشد که در این صورت باید در زیر فولدر meta در main.yaml تعریف شوند.

Template :

اینها فایلهای متنی هستند که می توانند با متغیرهای محلی پر شوند و نهایتا یک فایل متنی شوند و به سرور محل مقصد کپی شوند.(مانند velocity )

Template ها به زبان jinja2 هستند.

Variables :

متغيرهاي هم نام به ترتيب اولويت زير فراخواني خواهند شد.

* 1. ansible-playbook -v /etc/ansible/playbooks/varsPlay.yaml -e 'myvar="inline message"' (inline mode)
  2. Host variable (in host\_vars folder in ansible with same name with hostname in hosts file)
  3. Group variable( in group\_vars folder in ansible
  4. /etc/ansible/roles/testrole/vars/main.yml
  5. /etc/ansible/roles/testrole/defaults/main.yml
  6. /etc/ansible/group\_vars/all.yaml

مي توانيم متغيرها را گروه بندي کنيم بصورتي که فقط براي گروه هاستهاي انتخاب اعمال شوند.

اول بايد در host ها گروه بندي کنيم:

[group1]

10.100.0.213

و بعد در فولدر متغير ها (group\_vars)يک فايل با نام group1.yaml و متغيي را در ان تعريف مي کنم.

حالا اگر قرار باشد که براي host که در group قرار دارد دستورات انجام شود متغيري که در فايل group1 تعريف شده است اعمال خواهد شد.

اگر در فولدر host\_vars متغيري با نام همان host که در فايل hosts هست تعريف کنيم اين متغير اولويت بيشتري نسبت به group ها دارد و اين متغير اعمال خواهد شد.

براي ذخيره سازي پاسخ يک ماژول در يک متغيير از کلمه register استفاده ميکنم.

- name: check volume

Stat:

Path: “ {{ volume\_path }}”

Register: vp

-debug: msg=”{{ vp }}”

از اين متغير ميتوانيم در ماژولهاي بعدي نيز استفاده کنيم.

ميتوانيم متغيرها بصورت dictionary تعريف کنيم.

Facts :

Fact ها متغييرهايي هستند که ansible بصورت اتومات در زماني که سيستمهاي دور را کنترل مي کند جمع اوري ميکند. در واقع اطلاعاتي هستند که براي سيستمهاي دور مي باشند.

ansible all –m setup

ansible <hostname> -m setup -a "filter=ansible\_local"

براي اجراي ماژولها بصورت شرطي از دستور when استفاده ميکنيم.

When: (ansible\_os\_family == ‘Redhat’ and ansible\_os\_family == ‘Redhat’) or (ansible\_os\_family == ‘Redhat’ and ansible\_os\_family == ‘Redhat’)

يا

When:

-ansible\_os\_family == ‘Redhat’

-ansible\_os\_family == ‘Redhat’

Ansible vault

اهداف vault:

. ذخيره سازي role ها و playbook ها در يک version control

. راههاي مديريت امنيت

با استفاده از ان مي توانيم فايلها را encrypt و decrypکنيم.

در اين روش فقط کافي است تا پسور vault را حفظ کنيم و بقيه فايلهاي پسورد را بوسيله ان encrypt کنيم.

ansible-vault create foo.yml

ansible-vault edit foo.yml

ansible-vault encrypt foo.yml

ansible-vault decrypt foo.ym

براي اينکه به ansible بگوييم که در هنگام اجرا پسورد را از ما بگيرد از دستور زير استفاده ميکنيم.

ansible-playbook --ask-vault-pass

در فولدر library ماژولهاي custom قرار ميگيرند. که به زبان python نوشته شده اند.

براي غير فعال کردن اين موضوع که براي sudo از ما پسور نخواهد دو کار بايد انجام دهيم:

* Vim /etc/sudoers

%sudo ALL=(ALL) NOPASSWD:ALL

* Vim /etc/ansible/ansible.conf
  + Sudo\_pass = false

ماژول file :

Ansible all –m file –a ‘path=/appserver/test’

Ansible all –s –m file –a ‘path=/appserver/test state=directory mode=0700 owner=root’

-s يعني sudo

ماژول raw براي اجراي يک دستور shell روي سرورهاي مقصد مي باشد. ( مي توانيم در اين دستور بگوييم که از shell يا zsh و يا ... براي اجراي دستورات استفاده شود.)

until :

به تعداد دفعات شرط مورد نظر را چک مي کند.

- shell: /usr/bin/foo

register: result

until: result.stdout.find("all systems go") != -1

retries: 5

delay: 10

با -- tag مي توانيم بگوييم که فقط انهايي که در اين playbook داراي اين تگ هستند اجرا شوند.

استفاده از role ها بصورت لوکال:

ماژول synchroniz :

Each character is a code that can be translated if you read the section for -i, --itemize-changes in man rsync

Decoding your example log file from the question:

### >f.st......

> - the item is received

f - it is a regular file

s - the file size is different

t - the time stamp is different

### .d..t......

. - the item is not being updated (though it might have attributes

that are being modified)

d - it is a directory

t - the time stamp is different

### >f+++++++++

> - the item is received

f - a regular file

+++++++++ - this is a newly created item

The relevant part of the rsync man page:

-i, --itemize-changes

برای کم کردن لاگ ansible بصورت موقت :

export ANSIBLE\_STDOUT\_CALLBACK=skippy

و بصورت دائمی در فایل ansible.config انرا باید عوض کنیم.

stdout\_callback = dense

ansible awx :

يک پروژه web مي باشد که براي مديريت وظايف استفاده مي شود.

در اين نرم افزار مي توانيم يک پروژه تعريف کنيم که در واقع همان task هايي است که بايد انجام شود. که مي تواند اين task بصورت دستي تعريف شود ( در ادرس /var/lib/awx/projects ) و هم مي تواند به يک url گيت متصل شود و از انجا بخواند که در اين صورت اگر در گيت task هارا تغيير دهيم بصورت اتوماتيک با اين ابزار سينک مي شود.

نکته:

برای اجرای دستورات روی دو گروه از هاستها بصورت زیر عمل میکنیم:

ansible-playbook /etc/ansible/playbooks/syncSource.yml --limit '10.15.8.145:10.15.8.137' –check

همانطور که مشاهده می شود با : باید انها را از هم جدا کنیم.