شرح Ansible

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/

What is Ansible ?

- **Ansible** is an open-source configuration management and provisioning tool, similar to Chef, Puppet or Salt
  - فكرت ال Configuration Management ان انا هبدأ استخدم ال Configuration Management علشان هكونفجر السيرفر بتاعي او السيستم
  - ال Ansible ب Configure اي Machine عندي وبيشتغل علي Protocol ال SSH
- ال Ansible مبيحتجش اي Configuration مختلفه غير ان يكون فيه SSH متظبط مابين المكان اللي ها Run منه ال Ansible ومابين السيرفرات اللي أحنا عايزين نعملها عملية ال Configuration

Why Ansible?

1- No Agent

- يعني ان هو مش محتاج اي Agent ينزل علي السيرفرات اللي بيشتغل عليها

2- Idempotent

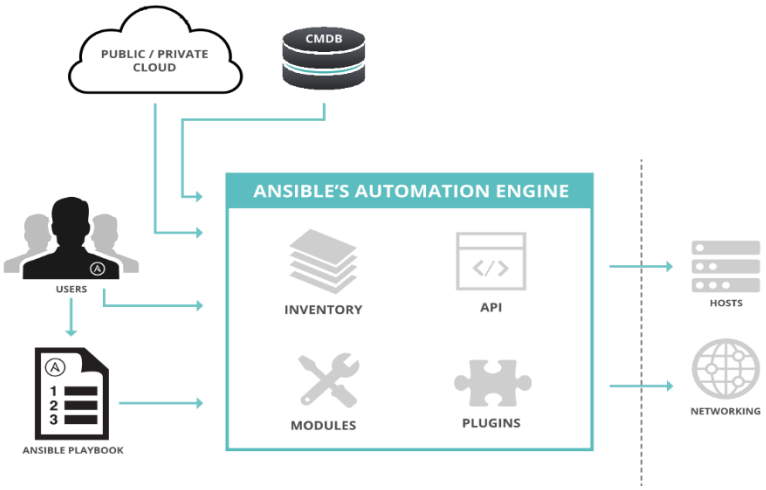
- بيقارن State ب State وبيشوف ال state دي موجوده ولالا لو موجوده مش هيعمل اي Install ليها , ولو مش موجوده هيبدأ يعملها Install .
- طالما هو بيقارن state ب state يبقي ايه الفرق مابين ال Ansible وال Shell Script ان ال Shell بيقولي ال State دي Already install ويطلع error ومش بيكمل الاسكريبت اما ال Ansible هيكمل عادي

3- Declarative Not Procedural

- كلمه **Procedural** يعني انا بمشي Line by Line . كل خطوة بتعتمد على اللي قبلها، فلو حاجة مش جاهزة أو لسه متنفذتش، هيحصل error. يعني لو في دالة Function بتعتمد على متغير أو قيمة معينة ولسه متنفذتش، الكود هيقف عند النقطة دي.
- أما **Declarative**، بتمشي بمنطق مختلف، هو بيقرأ الاسكريبت كله مرة واحدة، مش بالترتيب Line by Line زي ال Procedural. فلو في حاجة في أول الاسكريبت بتعتمد على حاجة في الاخر، مفيش مشكلة، هو هيشوف الحاجات دي ويكمل عادي من غير ما يحصل error، لأنه مش مهتم بترتيب التنفيذ، بس مهتم بالنتيجة النهائية.

4- Tiny Learning Curve

Ansible Concepts and Architecture

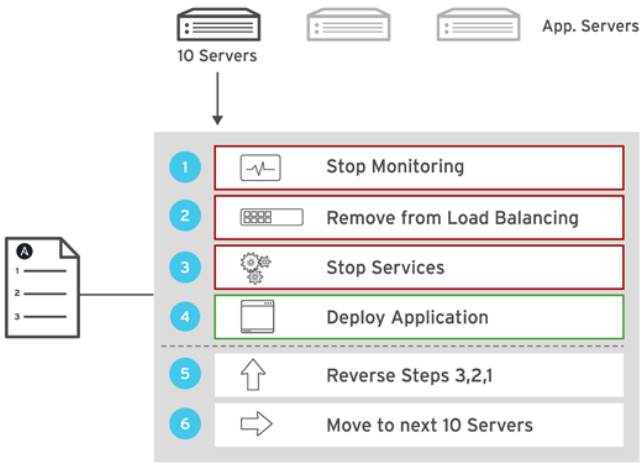


- في نظام Ansible، فيه نوعين من الأجهزة: **control nodes** و**managed hosts**.

- **Control Node**: هو الجهاز اللي بنشغّل عليه Ansible وبيكون فيه كل ملفات المشروع. الجهاز ده ممكن يكون Administrator's laptop، أو سيرفر بيستخدمه أكتر من Administrators

- **Managed Hosts**: دي الأجهزة اللي Ansible بيتحكم فيها وبتكون موجودة في حاجة اسمها inventory، واللي ممكن يكون static text file وبيحتوي على معلومات عن الأجهزة (الـ Hosts) اللي Ansible هيتعامل معاها
- الـ playbook ده ملف مكتوب بلغة YAML وبيحتوي على واحد أو أكتر من plays. كل play بينفذ شوية مهام (tasks) على الأجهزة اللي Ansible بيتحكم فيها. المهام دي بتستخدم حاجة اسمها modules، ودي زي أدوات صغيرة بتعمل شغل زي تعديل ملفات ال system أو تثبيت برامج. الـ playbook بيتميز إنه idempotent، يعني تقدر تشغله أكتر من مرة من غير ما يعيد نفس الخطوات طالما محصلش اي تغيير
- Ansible كمان بيشتغل مع plug-ins عشان تزود إمكانياته وتخليه يشتغل على Different platforms.
- الميزة كمان في Ansible إنه من غير agent، يعني مش محتاج تنزل برامج على الأجهزة اللي بيتحكم فيها، لأنه بيتواصل معاهم عن طريق SSH

The Ansible Way



- Ansible مصمّم بحيث يكون سهل تكتبه وتقراه، وده بيخليك تسعى دايمًا إنك تبسّط الطريقة اللي بتكتب بيها الـ automation.

- الـ Playbooks بتاعة ال Ansible مكتوبة بطريقة واضحة وسهلة القراءة لأنها بتعتمد على ملفات بسيطة ومباشرة. لو كتبت الـ playbooks بشكل صحيح، هتكون زي مستند بيوضح خطوات الـ automation اللي عايز تعملها.

- النقطة المهمة هي إنك تفكر بشكل declarative. يعني بدل ما تكتب خطوات تفصيلية لكل حاجة، أنسبل بيشتغل إنه يحط ال system في الحالة المطلوبة ويعمل التغييرات الضرورية بس.

USE CASES

- ✓ **Configuration Management**
- ✓ **Application Deployment**
- ✓ **Provisioning**
- ✓ **Continuous Delivery**
- ✓ **Security and Compliance**
- ✓ **Orchestration**

SSH Configuration

- قبل اما نبدء نشوف ازاي نستخدم ال Ansible هنشوف ازاي نستخدم ال SSH لان زي ماقولنا ان ال Ansible بيعتمد علي بروتوكول ال SSH
- **فيه عندي طريقتين من ال Authentication**

1- اول طريقه وهي **Password Based authentication** والطريقه دي ان انت كل مره تعمل login علي ال Server بيطلب منك ال Password بتاعت ال user اللي انت بت connect بيه

2- تاني طريقه وهي **SSH key-Based Authentication** ان انت بدل اما تستخدم ال Password بتعمل generate لحاجه اسمها key-pair او Public key و Private key

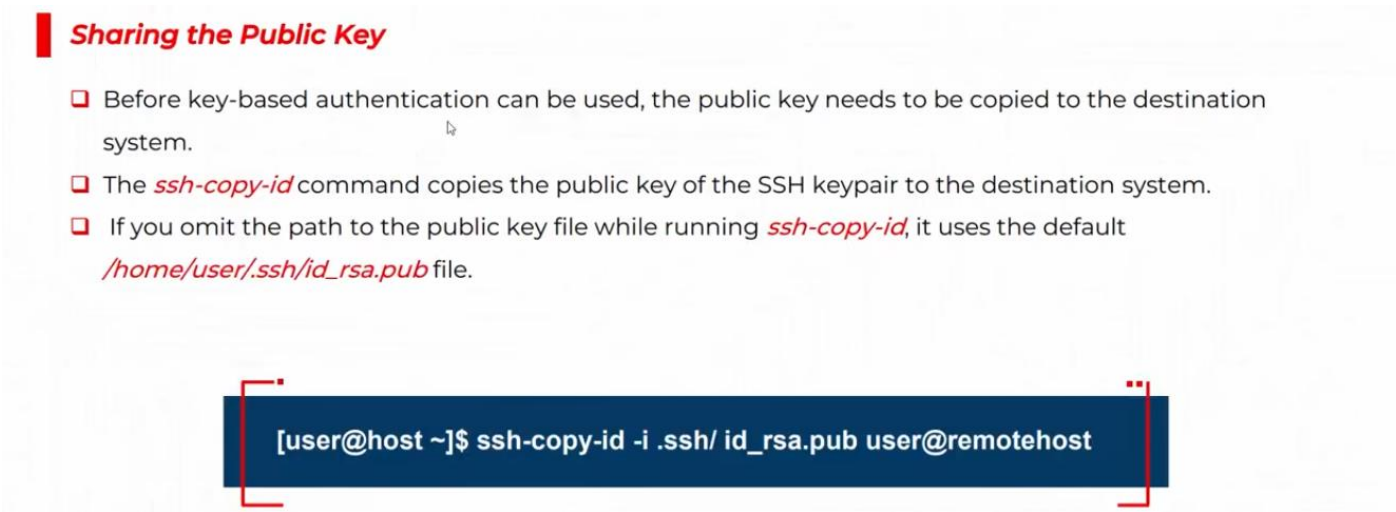
- ف علشان أ configure ال SSH key -Based Authentication عندي كذا Step لازم اعملهم

1- اول حاجه ب generate ال SSH key-pair ال Public key وال Private key علي ال Client

2- بعد كده بعمل copy لل Public key لل Server اللي انا عايز اعمل Login عليه

3- بعد كده بعمل SSH علي السيرفر ده ف ساعتها مش هيطلب مني اي Password لان هو هيعمل Authentication عن طريق ال Key بدل الباسورد

- بستخدم command اسمه ssh-keygen علشان اعمل generate لل Public وال Private key وبيكونوا by default موجودين في ال Home Directory تحت /.ssh وبيبقي اسمهم id_rsa بالنسبه لل Private Key وبالنسبه لل Public Key بيبقي اسمهم id_rsa.pub
- ال Private key ممنوع ان انت تعمله share مع اي حد ف لازم ت secure ال Private key بتاعك عن طريق ان انت ت restrict ال Permissions تعمل deny لاي حد ف علشان تعمل كده هتخلي ال Permission علي ال Private key بتكون 600 وبالنسبه لل Public key هيكون 644



- بعد اما انت عملت generate لل Public key وال Private key ف انت محتاج ان انت تعمل copy لل Public key بتاعك لل Server اللي انت عايز تعمل Login عليه عن طريق ان انت هتستخدم ال command ده **ssh-copy-id**

➢ لو جينا علي ال client ودخلنا تحت /.ssh وعملنا list هنلاحظ ان احنا عندنا ملف واحد وهو known_hosts

```
[root@Client .ssh]# ls
known_hosts
```

➢ لو انت قولتله ssh-keygen انت كده بتقوله ان انت عايز ت Generate Public and Private keys اول اما تضغط enter هيقولك by default ان هو هيعمل Save لل Public وال Private key بتاعك في ال Home directory لل user اللي عمل run لل **ssh-keygen** جوه /root/.ssh/ باسم id_rsa وده ال Private key بعد اما تضغط enter هيسألك يقولك Enter passphrase ان انت عايز تحط Password علي ال Private key بتاعك ولا لا لو ضغط Enter معناها ان انت مش عايز تحط باسورد بعد كده هيعمل generate

```
[root@Client .ssh]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gdvhfqv6+wgEteiCzdCOvc99P6++6PBEXoGoElhJpTg root@MohamedAtef
The key's randomart image is:
+---[RSA 3072]----+
| .oo. .          |
| =.. o + .       |
|E + o + + .      |
| X o o + o .     |
|o B o o S .      |
|  + . + .        |
| .  o + .        |
|  o . =.+..      |
|   o o=O*B=.     |
+----[SHA256]-----+
```

➢ لو عملنا ls تاني هنلاقيه عمل create ل Two file وهما id_rsa و id_rsa.pub

```
[root@Client .ssh]# ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

➢ بعد كده هنعمل copy لل Public key بتاعك للServer باستخدام ssh-copy-id

```
[root@Client .ssh]# ssh-copy-id root@172.16.76.133
```

➢ بعد كده انت تقدر تعمل connect علي Server بدون اي مشكله

- ازاي هنعمل customize لل open SSH Service configuration انا عندي ال main configuration file الخاص بال SSH Service موجود في /etc/ssh/sshd_config/ لو فتحنا ال sshd_config هنلاقي Parameters اسمه PermitRootLogin ال Parameters ده لو انت عايز تعمل deny او restrict ان مفيش اي يحد يقدر ي access ال Server بتاعك remotely باستخدام ال root ف انت بتخلي الPermitRootLogin ب No ودي حاجه recommended
- اي تغيير في ال Configuration file بتبقي محتاج ان انت تعمل reload لل Service عن طريق استخدام

```
[root@MohamedAtef ~]# systemctl reload sshd
```

Prohibiting password-based authentication for SSH

- The OpenSSH server uses the *PasswordAuthentication* parameter in the */etc/ssh/sshd_config* configuration file to control whether users can use password-based authentication to log in to the system.
- The default value of *yes* for the PasswordAuthentication causes the SSH server to allow users to use password-based authentication while logging in.
- The value of *no* for PasswordAuthentication prevents users from using password-based authentication.
- Keep in mind that whenever you change the /etc/ssh/sshd_config file, you must reload the sshd service for changes to take effect.

- فيه Parameter تاني اسمه *PasswordAuthenticaton* هل انت عايز ت enable ال Password based Authentication احنا قولنا ان فيه نوعين من ال Authentication في ال SSH فيه نوع اسمه Password based Authentication والنوع التاني key-based Authentication ال recommended وال More Secure ان انت تستخدم ال Key-Based Authentication مش ال Password Authentication . ف انت لازم تعمل Disable لل root access و Disable لل Password Authentication وطبعا متنساش في نفس الوقت لازم تبقي عامل Enable لل SSH Key او ال Key Based Authentication وعامل generate لل Public وال Private Keys علشان تقدر تعمل Login علي السيستم بتاعك

Installing Ansible

- تثبيت Ansible سهل جدًا. كل اللي محتاج تعمله هو تثبيت ال ansible على الـ control node اللي هتشتغل منه. الأجهزة اللي ال Ansible بيتحكم فيها الـ managed hosts مش محتاجة يكون عليها Ansible.

- الـ control node لازم يكون نظام تشغيل Linux أو UNIX. أنظمة Windows مش بتدعم تشغيل Ansible كـ control node، بس ممكن تدير أجهزة ال Windows كـ managed hosts.
- كمان لازم تكون عامل install ل Python 3 او Python2.7 معمول ليه install على الـ control node علشان ميحصلش اي مشكله في ال Install
- ف الافضل انت تعمل Install عن طريق ال Documentation علي حسب ال OS اللي شغال عليه

> ***Installation Guide — Ansible Documentation***
>
> http://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

- ف انا شغال علي redhat 9 ف هستخدم ال yum install ansible -y

```
[root@Control ~]# yum install ansible-core.x86_64
```

- ممكن نستخدم ansible --version علشان نتاكد ان ال Ansible اتعمل ليه Install ونشوف ال Version الخاص بيه

```
[root@Control ~]# ansible --version
ansible [core 2.14.2]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.11/site-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.11.2 (main, Feb 16 2023, 00:00:00) [GCC 11.3.1 20221121 (Red Hat 11.3.1-4)]
(/usr/bin/python3.11)
  jinja version = 3.1.2
  libyaml = True
```

Building an Ansible Inventory

• Defining the Inventory

- ال Inventory زي ماقولنا قبل كده هو عباره عن file بنعرف فيه الايبهات بتاعت السيرفرات اللي هيبدأ ال Ansible بتاعنا انه يتعامل معاها ف ممكن نعرف فيه ال IP او ال Hostname وممكن كمان يتقسم ل Groups
- وال Default Location هو ال /etc/ansible/hosts ويفضل ان احنا منستخدمش ال file اللي اسمه hosts اللي موجود في ال Default Locationالافضل ان احنا نستخدم ال Inventory علشان يكون اكثر تنظيم ويفضل ان يكون فيه file لكل مشروع
- ف علشان نستخدم ال Inventory هنحتاج ان احنا نعمل create ل file باسم Inventory ويفضل انه يكون جوه folder ك تنظيم ف احنا هنعمل folder باسم مثلا ansible وجواه هنعمل file باسم inventory
- ملف ال Inventory بيكون بالشكل ده بيكون مكتوب بصيغه INI Format ممكن نعرف فيه ال Hostname زي web1.example.com و db1.example.com او ال IP الطريقه دي بتسمي ال Static Inventory

```
web1.example.com
web2.example.com
db1.example.com
db2.example.com
192.0.2.42
```

- ممكن كمان ان احنا ننظم ملف ال Inventory عن طريق ان احنا بنظم ال Hosts في Groups ان كل مجموعه بيتم تسميتها في بدايه ال Section عن طريق ان انا بكتب بين قوسين [ ] اسم الجروب زي مثلا webservers وبعدين بكتب تحته اسامي ال Hosts وال IPs اللي من نفس ال Group

```
[webservers]
web1.example.com
web2.example.com

[databases]
db1.example.com
db2.example.com
192.0.2.42
```

- في Ansible تقدر تعمل Nested Groups يعني مجموعة تكون جواها مجموعات تانية، وده بيسهل عليك تنظيم ال Hosts اللي بتشتغل عليها . ف لو احنا عايزين نعمل Nested Groups هستخدم children: مع ال group الجديد يعني ال group اللي اسمه usa فيه الاجهزه دي washington1 و washington2 و مجموعة canada فيها الأجهزة ontario01 وontario02 ف لو انا عايز عايز الجروب الجديد اللي اسمه north-america يكون موجود فيه الجروب اللي اسمه usa و Canada هستخدم children بعد اسم ال north-america وبعدين هكتب اسامي ال Groups
- يعني north-america بقت بتضم كل الأجهزة اللي في المجموعتين اللي هما
  - washington1.example.com
  - washington2.example.com
  - ontario01.example.com
  - ontario02.example.com

```
[usa]
washington1.example.com
washington2.example.com
[canada]
ontario01.example.com
ontario02.example.com
[north-america:children]
canada
usa
```

- ممكن كمان ان انت تسخدم ال Ranges في ال Host names وال IP Addresses عن طريق ان انا بحدد البدايه وبعدين النهايه [START:END]
  - يعني لو انا عايز احدد مجموعه من ال IP بدل اما انا اكتب ال IP كلها لا انا هحدد ال IP اللي انا عايز اشتغل عليهم زي مثلا 192.168.1.[10:20] انا هنا حددت من اول 10 لحد 20
  - ونفس الموضوع لو انا عندي اكتر من hostname ممكن ان انا استخدم ال range زي مثلا server[01:20].example.com انا هنا قولتله ان انا عايز اعمل matchesلكل ال hosts من اول server01.example.com ل server20.example.com

```
[usa]
washington[1:2].example.com
192.168.1.[10:20]

[canada]
ontario[01:02].example.com
[a:c].dns.example.com
```

- في الشرح هنشتغل علي Three Machines ف هيكون موجود Machine هتكون ال control-node ودي اللي بينزل عليها ال Ansible وبيتم عليها كل ال Configurations وهيكون اسمها Control node والاتنين التانيين هيكونوا Node1 و Node2
- هنطبق اللي احنا شرحناه عن طريق ان احنا هنعمل Create لل Machines ف ممكن ان احنا نعمل createعلي ال VMware Workstation او عن طريق ال AWS Cloud او Azure Cloud علي حسب بتفضل تشتغل علي ايه
- ف اللي احنا عايزين نعمله هنعمل Ansible يكون Configure عليهم كلهم ان انا ازاي اعرف من ال Control node اعمل Run لل Ansible ف انا محتاج ان ال SSH يكون متظبط ومحتاج يكون ال ansible نازل علي ال Control Node وكمان as a best practice ان يكون عندي dedicated user لل Ansible بمعني ان يكون فيه user علي ال Control-node هو هو نفس ال User اللي علي ال node1 و node2 وال user ده بس اللي يستعمل Ansible وكمان عايزين نستعمل ال Hostname بدل ال IP لان يفضل ان احنا نستخدم ال Hostname لان ممكن ال IP يتغير ف احنا عايزين علي ال Control-Node نقدر نوصل ل Node1 و Node2 عن طريق Two Domain Name واحد هنسميه node1.example.com والتاني هنسميه node2.example.com
- ف احنا هنروح علي ال Control-node ونعمل Install لل Ansible

```
[root@control-node ~]#  yum install ansible-core.x86_64
```

- بعد كده هنغير ال Hostname لكل ال Machine اللي عندي
  - لل control-node هنخليه control-node.example.com

```
[root@control-node ~]# hostnamectl set-hostname control-node.example.com
```

  - ول node1 هنخليه node1.example.com

```
[root@node1 ~]# hostnamectl set-hostname node1.example.com
```

  - ول node2 هنخليه node2.example.com

```
[root@node2 ~]# hostnamectl set-hostname node2.example.com
```

- وبعدين زي ماقولنا ان احنا عايزين ال Machine اللي احنا واقفين عليها تكون شايفه ال Two Nodes بال Hostname مش بال IPs ف هندخل علي ال Host بتاعي عن طريق /etc/hosts وهندخل جوه الملف وهنكتب فيه الاتي ال IP وال name وال Hostname لل control-node وال node1 وnode2

```
192.168.1.2   control    control-node.example.com
192.168.1.3   node1      node1.example.com
192.168.1.5   node2      node2.example.com
```

- بعد كده هنعمل create لل User اللي اتكلمنا عليه وهنسمي ال user ده مثلا ansible ونديله password وال user ده هعمله create علي ال control-node وnode1 و node2

```
# useradd ansible
#  passwd ansible
```

- ال user ده لازم يكون ليه Sudo Permissions علشان يشتغل كأنه ال root ف انا هدخل تحت المساردة /etc/sudoers.d/ واعمل create ل file باسم ansible وهنحط فيه ال permissions الخطوه دي هتم علي ال control-node و node1 و node2 باستخدام ال root مش ال user

```
anisble ALL=(ALL) NOPASSWD: ALL
```

- بعد كده علي ال control-node هندخل علي ال user اللي اسمه ansible باستخدام su – ansible و هنعمل create ل ال Public key وال Private key فهنستخدم الامر ssh-keygen

```
[root@control-node ~]# su - ansible
[ansible@control-node ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Created directory '/home/ansible/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:XbPvtik2KyLV44aJ7col2YfvZh2rV0ss6kqmGg/65uU ansible@control-node.example.com
The key's randomart image is:
+---[RSA 3072]----+
|            |
|            |
|         o  |
|       . . o  |
|       S.. ..  |
|      o..o o.+  |
|   o ++*+.+ *.. |
|  ..Bo*=oB B.o. |
| .++.E+oO==.=+. |
+----[SHA256]-----+
```

- لو دخلنا تحت /home/ansible/.ssh/ هنلاقي اتعمل create لل Public وال Private key

```
[ansible@control-node ~]$ cd .ssh/
[ansible@control-node .ssh]$ ls
id_rsa  id_rsa.pub
```

- بعد كده لازم ان احنا نضيف ال Public key بتاع ال control node اللي احنا عملناه واضيفه في node1 و node2 ف علشان اعمل كده هستخدم ال command ده ssh-copy-id وال command ده كل اللي هيعمله انه by default هيدور عندي علي ال id_rsa.pub وهياخد ال public key ده وهيبعته لل Host Name وهيحطه في ال authorized key تحت ال user اللي اسمه ansible في node1 وnode2

```
[ansible@control-node ~]$ ssh-copy-id node1.example.com
[ansible@control-node ~]$ ssh-copy-id node2.example.com
```

- كده احنا ضفنا ال Public key الخاص بال server اللي هو ال ansible الموجود في ال control-node في node1 وnode2 ف لو احنا داخلين علي ال user ده علي ال control وجينا عملنا ssh علي اي node هقدر انه يعمل ssh بدون اي مشكله

```
[ansible@control-node ~]$ ssh node1.example.com
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sun Oct 20 01:03:26 2024
[ansible@node1 ~]$
```

- كده احنا ظبطنا كل حاجه خاصه بال ssh وال ip وال hosts
- تعال بقي نشوف ازاي نستخدم ال Ansible ف ال ansible علشان يعمل connect لازم يكون عارف ال IPs او ال Host name بتاعت ال Machine اللي عايز اوصلها ف احنا هنحط الكلام ده في ملف اسمه Inventory وزي ماقولنا قبل كده ان ال Default Location الخاص بال ansible هو ال /etc/ansible/hosts ويفضل ان احنا منستخدمش ال file اللي اسمه hosts اللي موجود في ال Default Locationالافضل ان احنا نستخدم ال Inventory
- ف هنعمل create ل folder اسمه وليكن Test وهنعمل create جواه ل file باسم inventory ونكتب في الملف ده ال Hostname الخاص بال Machines الكلام ده علي ال user اللي اسمه ansible علي ال control-node

```
[ansible@control-node ~]$ mkdir test
[ansible@control-node ~]$ touch inventory
[ansible@control-node ~]$ vim inventory
[nodes]
node1.example.com
node2.example.com
```

- لو احنا عايزين نتاكد ان هو شايف الكلام اللي احنا ضيفناه في ال inventory هستخدم

```
[ansible@control-node ~]$ ansible  -i  inventory   all   --list-hosts
  hosts (2):
    node1.example.com
    node2.example.com
```

- لما تيجي تشغل أوامر Ansible لو عايز تحدد ملف الـ inventory اللي فيه الأجهزة اللي عايز تديرها، لازم تستخدم الخيار **-i** في الأمر. ده بيخلي Ansible يعرف يستخدم ملف الـ inventory اللي انت محدده بدل ما يعتمد على الملف الافتراضي اللي في /etc/ansible/hosts.
- ممكن كمان ان انا استخدم اسم ال group

```
[ansible@control-node ~]$ ansible   -i   inventory   nodes   --list-hosts
  hosts (2):
    node1.example.com
    node2.example.com
```

- ممكن كمان ان انا احددله hostname معين ان انا عايز اتاكد ان ال hostname موجود ولالا

```
[ansible@control-node ~]$ ansible   -i   inventory   node1.example.com   --list-hosts
  hosts (1):
    node1.example.com
```

- ممكن كمان لو ملف ال Inventory موجود فيه groups كتيره و Hostname وIP وانا عايز اعمل List لكل ال Hosts اللي مبتكونش من ضمن ال groups هستخدم ungrouped

```
[ansible@control-node ~]$ ansible   -i   inventory  ungrouped  --list-hosts
[WARNING]: No hosts matched, nothing to do
  hosts (0):
```

- لو احنا عايزين نتاكد ان ال ansible هيقدر ي connect معاهم او شايف السيرفرات التانيه ومفيش اي مشكله هنستخدم module مشهور جدااا اسمه Ping وده وظيفته انه بيعمل connect علي ال remote host وبيرجعلك رساله فيها pong ف هو بيعمل test لل connections فهنستخدم **-m** وده عباره عن ال module اللي ال ansible هيشغله علي الاجهزه وبعدين هنستخدم ping الطريقه دي اسمها ad-hoc commands وهنشرحها كمان شويه

```
[ansible@control-node ~]$ ansible   -i  inventory  nodes  -m ping
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
node2.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

ف هنلاقيه هنا باعتلي ping و pong ال pong من ناحيه ال server الاولاني اللي هو node1.example.com و pong من ناحيه ال server التاني اللي هو node2.example.com

Configuring Ansible

- إعدادات Ansible بتتغير من خلال تعديل ملف ال configuration الخاص بيه. Ansible بيدور على ملف ال configuration في أماكن مختلفة على الجهاز اللي بتتحكم من خلاله (control node)، وبيستخدم أول ملف يلاقيه بناءً على الترتيب ده

1- اول حاجه بيبص في المسار ده **/etc/ansible/ansible.cfg** و ده ال Default file اللي بينزل مع ال Ansible، وبيتم استخدامه لو مفيش أي ملفات configuration تانية موجودة. ده بيكون ملف ال configuration الأساسي اللي Ansible بيبدأ بيه.

2- تاني حاجه بيبص في **~/.ansible.cfg** وهنا Ansible بيدور في Home Directory الخاص بال User ذات نفسه على ملف اسمه ansible.cfg. لو الملف ده موجود، Ansible هيستخدمه بدل الملف ال Default اللي في **/etc/ansible/ansible.cf**

3- تالت حاجه بيبص في **./ansible.cfg** لو لاقي ملف ال configuration اللي اسمه ansible.cfg موجود في ال Directory اللي انت واقف فيه اللي بتشغل منه ال Ansible هيستخدمه بدل الملف ال Default أو ملف ال Home Directory الخاص بال User. الميزة هنا إنك تقدر تجهز ملفات configuration مختلفة لكل مشروع وكل مشروع بيكون ليه إعداداته الخاصة.

- ولو احنا عايزين نعرف ال ansible هيستخدم انهي configuration file هستخدم --version

```
[ansible@control-node test]$ ansible --version
ansible [core 2.14.2]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
```

هتلاقيه هنا قايلك ان ال configuration file اللي هيستخدمه موجو في المكان ده /etc/ansible/ansible.cfg وده ال default

- لو احنا عملنا create ل ansible.cfg ونفذنا ال ansible --version في نفس مكان ال file هيختار ال configuration file اللي احنا عملنا ليه create بدل ال default

```
[ansible@control-node test]$ touch ansible.cfg
[ansible@control-node test]$ ansible --version
ansible [core 2.14.2]
  config file = /home/ansible/test/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
```

هتلاقيه هنا قايلك ان ال configuration file اللي هيستخدمه موجو في المكان ده /home/ansible/test/ansible.cfg

Managing Settings In The Configuration File

- ملف ال configuration بيتكون من sections مختلفة، وكل section بيتحكم في مجموعة معينة من الإعدادات. وال Sections دي بتتكتب مابين قوسين [ ] ومن اشهر ال Sections المستخدمه واللي احنا هنتعامل معاها هما [defaults] و [privilege_escalation].

1- اول Section وهو [defaults] ده ال section الأساسي اللي بيتحكم في إعدادات Ansible العامة.

2- تاني Section وهو [privilege_escalation] وده ال Section اللي بيحدد فيها ال behavior بتاع ال Ansible لو هو عايز يرفع ال Permissions بتاعته

- وبينكتبوا في ملف ال configuration بالشكل ده ف احنا هنعمل create ل ansible.cfg وهنكتب فيه الاتي

```
[defaults]
inventory = inventory #هنا انا بحددله مسار ال Inventory موجود فين انا هنا كتبت inventory لان انا واقف في نفس المكان اللي فيه الملف
remote_user = ansible # هنا انا بحددله ان لو معنديش the same remote user يستخدم ال user اللي اسمه ansible
host_key_checking = False # هنا لو عايز اعطل انه كل اما اعمل connect باستخدام ssh ميطلبش مني التأكيد

[privilege_escalation] # هنا لو عايز اعمل enable او disable لل privilege escalation علشان ال user ده يقدر انه يستخدم صلاحيات ال sudo
become = True
become_user = root # هنا انا بقوله استعمل ال Permissions بتاع تال root
become_method = sudo # هنا انا بقوله استعمل ال Permissions بتاعت ال root بأستخدام ال sudo
become_ask_pass = False # هنا انا بقوله انه ميطلبش password
```

- ف ممكن ان احنا نعمل Ping بس هنا احنا ممكن نستغني عن i inventory- لان انا محددله في ملف ال configuration اللي هو ansible.cfg محددله مكان ال inventory في ال section الخاص بال defaults

```
[ansible@control-node test]$ ansible  nodes  -m  ping
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
node2.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Ad-Hoc Commands

- الـ ad-hoc commands في Ansible هي أوامر بسيطة وسريعة تستخدم لإجراء مهام معينة على الأجهزة بدون الحاجة إلى كتابة playbooks كاملة. الأوامر دي بتكون مناسبة لما يكون عندك حاجة سريعة عايز تنفذها على جهاز أو مجموعة أجهزة، زي تشغيل أمر معين، التحقق من حالة الأجهزة، أو حتى تعديل إعدادات بسيطة. وتقدر انك تنفذ module واحد فقط باستخدام ال ad-hoc commands ف احنا نقدر نعمل بيه اي task زي مثلا لو انا عايز اضيف user معين او اعدل عليه او ان انا اعمل update او اعمل install ل اي package ف فيه عندي modules كتيره جدااا ممكن نستخدمهاا
- ودي الطريقه اللي بتتكتب بيها

```
Ansible  [-i inventory]  host-pattern  -m  module  [-a 'module arguments']
```

- فيه module مهم جدااا جدااا ممكن نستخدمه علشان نطبع رساله ال module ده اسمه debug وال debug هي دي الطريقه الوحيده اللي ال ansible يقدر يطبع بيها رساله

```
[ansible@control-node test]$ ansible nodes -m debug -a 'msg="Hello Everyone"'
node1.example.com | SUCCESS => {
    "msg": "Hello Everyone"
}
node2.example.com | SUCCESS => {
    "msg": "Hello Everyone"
}
```

- فيه module تاني ممكن نستخدمه اسمه user وممكن نستخدم في ال argument ال name وال state في ال name معناها انشاء او تعديل ال user وال state هنا انا قولتله present علشان لازم ال user اللي اسمه Mohamed يكون موجود ولو مش موجود هيعمله create

```
[ansible@control-node test]$ ansible node1.example.com  -m user -a 'name=mohamed state=present'
node1.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 4001,
    "home": "/home/mohamed",
    "name": "mohamed",
    "shell": "/bin/bash",
```

هتلاقيه هنا قايلك ان ال changed تم وعمل create لل home directory لل user

- معظم ال Modules في Ansible بتكون idempotent، وده معناه إنك تقدر تشغلها أكتر من مرة بأمان وبدون اي مشكله ولو النظام بالفعل في الحالة الصحيحة، مش هيحصل أي تغيير بمعني يعني لو نفذت نفس الـ ad-hoc command اللي بيستخدم module الـ user مرة تانية لو ال user اللي اسمه mohamed موجود بالفعل Ansible هيعمل check ويلاقي إن الحالة المطلوبة موجودة فعلاً، ومش هيعمل أي تعديل. وال output اللي هيظهر إن مفيش أي تغيير حصل

```
[ansible@control-node test]$ ansible node1.example.com  -m user -a 'name=mohamed state=present'
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "append": false,
    "changed": false,
    "comment": "",
    "group": 4001,
    "home": "/home/mohamed",
    "move_home": false,
    "name": "mohamed",
```

هتلاقيه هنا اهو قايلك ان عمليه ال append ب false والتغير ب false

- ممكن كمان ان احنا ننفذ command معين باستخدام Module اسمه command وليكن انا عايز اعمل create ل file اسمه devops في ال host اللي اسمه node1.example.com

```
[ansible@control-node test]$ ansible   node1.example.com   -m   command   -a   'touch devops'
node1.example.com | CHANGED | rc=0 >>
```

- ممكن نستخدم نفس ال module علشان ننفذ command ال ls علشان نتاكد ان ال file اتعمل ليه create

```
[ansible@control-node test]$ ansible   node1.example.com   -m   command   -a   ls
node1.example.com | CHANGED | rc=0 >>
devops
```

هتلاقيه عمل list لل files اللي موجوده وهتلاقي هنا ال devops موجود

- ممكن كمان نعمل install ل Package معينه او Service معينه وليكن لو عايز اعمل install ل httpd علي node1 ف هنستخدم module اسمه yum

```
[ansible@control-node test]$ ansible   node1.example.com   -m   yum   -a "name=httpd  state=present"
```

- ممكن كمان ان احنا نستخدم Module اسمه service علشان اعمل start ل service معينه

```
[ansible@control-node test]$ ansible node1.example.com -m service  -a  "name=httpd  state=started"
```

- ف موجود هنا في ال Documentation كل ال Modules اللي ممكن تستخدمها

> ***Introduction to Ad-Hoc Commands: Ansible Documentation***
>
> https://docs.ansible.com/ansible/latest/command_guide/intro_adhoc.html

- في الجدول موجود بعض ال modules الرئيسيه اللي ممكن نستخدمها ومتقسمه علي حسب وظيفتها

MODULE CATEGORY	MODULES
Files modules	• `copy`: Copy a local file to the managed host • `file`: Set permissions and other properties of files • `lineinfile`: Ensure a particular line is or is not in a file • `synchronize`: Synchronize content using **`rsync`**
Software package modules	• `package`: Manage packages using autodetected package manager native to the operating system • `yum`: Manage packages using the YUM package manager • `apt`: Manage packages using the APT package manager • `dnf`: Manage packages using the DNF package manager • `gem`: Manage Ruby gems • `pip`: Manage Python packages from PyPI
System modules	• `firewalld`: Manage arbitrary ports and services using firewalld • `reboot`: Reboot a machine • `service`: Manage services • `user`: Add, remove, and manage user accounts
Net Tools modules	• `get_url`: Download files over HTTP, HTTPS, or FTP • `nmcli`: Manage networking • `uri`: Interact with web services

- ممكن ان احنا نستخدم ال configurations اللي استخدمناها في ملف ال ansible.cfg مع ال ad hoc command ولو احنا استخدمنا اي configuration مع ال ad hoc ف بيكون لل ad hoc الاولويه عن ملف ال configuration في التنفيذ
- الجدول ده بيوضح ال options اللي ممكن تستخدمها مع ال ad hoc command

CONFIGURATION FILE DIRECTIVES	COMMAND-LINE OPTION
`inventory`	**`-i`**
`remote_user`	**`-u`**
`become`	**`--become`**, **`-b`**
`become_method`	**`--become-method`**
`become_user`	**`--become-user`**
`become_ask_pass`	**`--ask-become-pass`**, **`-K`**

ANSIBLE PLAYBOOKS

- ال Playbook هو عباره عن YAML File ال Yaml file ليه شكل معين في ال Syntax وده بيكون جواه ال script اللي نقدر نقول عليه ال code اللي ب describe ال state بتاعت ال system
- وكل file من ال Playbook بيحتوي علي مجموعه من ال Plays او Play واحده وكل Play بيحتوي علي مجموعه من ال Tasks وكل Task بيتكون من module معين مع ال Parameters الخاصه بيه
- ف ال Playbook بتوفر لك طريقة منظمة و مرنة وقوية لإدارة الأجهزة وتنفيذ ال Tasks بشكل متكرر و منظم و مستقر.

- **مميزات ال Playbook**
  - تقدر ان انت تكتب مجموعة من ال Tasks مرة واحدة وتنفذها على أي عدد من الأجهزة في أي وقت من غير ما تحتاج ان انت تكتب أوامر جديدة في كل مرة
  - ممكن كمان بدل اما انت تستخدم ال ad-hoc commands علشان تنفذ command واحد تقدر تستخدم الـ playbooks علشان تنفذ Tasks كتيره ومعقدة بشكل منظم.
  - لما تستخدم الـ Playbooks بتضمن إن نفس ال Tasks هتتنفذ بنفس الطريقة على كل الأجهزة اللي عندك وبالتالي بتضمن استقرار اكتر بدون اي مشاكل او errors ممكن تحصل
  - تقدر ان انت تعيد استخدام ال playbooks بسهولة عن طريق ان انت تعدل ال Tasks أو ال variables لتناسب الاحتياج الخاص بيك وبالتالي ده بيقلل وقت كتابة ال Tasks الجديدة.

Formatting an Ansible Playbook

- ده كده مثال بسيط لل Playbook بينكتب بالطريقه دي

```
---
- name: Configure important user consistently
  hosts: node1.example.com
  remote_user: ansible
  tasks:
    - name: Ensure Ahmed exists with UID 4002
      ansible.builtin.user:
        name: Ahmed
        uid: 4002
        state: present
```

- اهم حاجه ان يكون extension ملف ال playbook يكون yml. احنا مثلا هنسميه site.yml
- اول حاجه في ملف ال Playbook لازم يبدء ب (---) Three dashes
- وزي ماقولنا ان ال playbook عباره عن list of plays ممكن يكون ال file جواه اكتر من play وكل play بت run مجموعه من ال Tasks وال Task عباره عن object وال play عباره عن object ف علشان ابدء اكتب task لازم ابدء اكتب داش – والمتعارف عليه وده best practice انك تحط name لل Playbook بتاعتك انك تقول انت بتعمل ايه زي ماموجود في المثال فوق ان احنا محددنله ال name عباره عن ايه Configure important user consistently وده بيكون option عايز تحط name ولا لا
- بعد كده بنديله ال Host Pattern اللي كنا بنكتبها في ال Ad-Hoc command ف احنا كتبنا ال node بتاعتنا اللي هي node1.example.com
- بعد كده فيه Parameters مهم جدااا لازم نستخدمه وهو ال remote_user وده احنا بنحددله مين ال user اللي هيستخدمه علشان يعمل run لل configuration ولو انت محددتلهوش ال remote_user بياخد ال username ال default بتاع الجهاز اللي انت شغال عليه
- بعد كده بنكتب ال tasks ودي لازم تتكتب ودي عباره عن list وكل List بيبقي ليها اسم name زيها زي ال Playbook وده بيكون best practice انك تعبر عن ال Tasks بتاعتك ف قولنا ان - name: Ensure Ahmed exists with UID 4000
- بعد كده في نفس المستوي بتاع ال object اللي اسمه name بنكتب ال module اللي احنا عايزينه واسم ال module بيكون عباره عن ال namespace.collections.modules وده بيكون ال fully coalified name لل Modules زي مثلا انا عايز استخدم ال module الخاص بال user ف هستخدم ansible.builtin.user او ممكن تكتب فقط user وده عباره عن alias
- بعد كده بنكتب ال Parameters الخاصه بال module اللي احنا استخدمناه وده احنا بنعرفه من ال search او ال Documentation زي المثال احنا استخدمنا ال name وال uid وال state

Running Playbooks

- علشان اعمل run لل playbook هستخدم ansible-playbook وبعدين اسم ال playbook

```
[ansible@control-node test]$ ansible-playbook site.yml
```

- دايما قبل اما نعمل run لل playbook لازم نعمل حاجه اسمها syntax check وده ال best practice وده وظيفته انه بيتاكد ان ال yml file بتاعك مكتوب صح ومفهوش اي مشكله لو انت كاتب كل حاجه صح هيطلعلك اسم ال Playbook بتاعك ال yml file بتاعك وفي حالتنا هنا هو ال site.yml

```
[ansible@control-node test]$ ansible-playbook   site.yml   --syntax-check

playbook: site.yml
```

- لو لنفرض ان فيه مشكله في ال syntax وعملنا تاني syntax-check هتلاقيه مطلعلك error وقايلك المشكله فين بالظبط وهنا انا كنت شايل حرف ال n من ال attribute اللي اسمه name

```
[ansible@control-node test]$ ansible-playbook   site.yml   --syntax-check
ERROR! 'ame' is not a valid attribute for a Play
The error appears to be in '/home/ansible/test/site.yml': line 2, column 3, but may
be elsewhere in the file depending on the exact syntax problem.
The offending line appears to be:
---
- ame: Configure important user consistently
  ^ here
```

- بعد اما نتاكد ان ال syntax صح ومفيش اي مشكله نعمل بعد كده run لل playbook

```
[ansible@control-node test]$ ansible-playbook   site.yml

PLAY [Configure important user consistently] ***********************************

TASK [Gathering Facts] *********************************************************
ok: [node1.example.com]                    لما تلاقي ok معناها ان محصلش اي تغيير ان مفيش change

TASK [Ensure Ahmed exists with UID 4002] ***************************************
changed: [node1.example.com]    هنا هتلاقيه قايلك changed معني كده ان حصل تغيير في ال host ده ان ان هو عمل
                                create لل user اللي اسمه Ahmed واداله UID ب 4002

PLAY RECAP *********************************************************************
node1.example.com   : ok=2   changed=1   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
                                هتلاقيه هنا قايلك ان ال change اللي حصل 1
```

- لو جينا عملنا run لل playbook تاني هتلاقيه قايلك ok وان مفيش اي تغير حصل لان ال user معمول ليه create قبل كده وموجود

```
[ansible@control-node test]$ ansible-playbook   site.yml

PLAY [Configure important user consistently] ***********************************

TASK [Gathering Facts] *********************************************************
ok: [node1.example.com]

TASK [Ensure Ahmed exists with UID 4002] ***************************************
ok: [node1.example.com]

PLAY RECAP *********************************************************************
node1.example.com   : ok=2   changed=0  unreachable=0   failed=0  skipped=0   rescued=0   ignored=0
```

- لازم دايما قبل اما تيجي تعمل run لل Ansible Playbook بتاعتك لازم تتاكد انك تقدر تعملها run اكتر من مره لو حاجتك صح مش هتغير حاجه وهيديك Ok

- **فيه عندي اكتر من options من نستخدمهم لو عايزين نعرف تفاصيل اكتر واحنا بنفذ ال Tasks**

  - اول اوبشن وهو v- وده احنا بنستخدمه لو عايز نعرف معلومات اكتر زي التغييرات اللي حصلت

```
[ansible@control-node test]$ ansible-playbook   -v  site.yml
```

  - تاني اوبشن وهو vv- وده بيعرضلي ال task configuration وال task results

```
[ansible@control-node test]$ ansible-playbook   -vv  site.yml
```

  - تالت اوبشن وهو vvv- وده بيعرض معلومات عن connections بين Ansible وال hosts

```
[ansible@control-node test]$ ansible-playbook   -vvv  site.yml
```

  - رابع اوبشن وهو vvvv- وده بيعرضلي تفاصيل كتيره جدا و بيعرض جميع المعلومات المتاحة زي مثلا ال users اللي بيستخدموا في ال Hosts علشان ينفذو script معين وايه هي ال scripts اللي اتنفذت وهكذا.

```
[ansible@control-node test]$ ansible-playbook   -vvvv  site.yml
```

  - Configuring the Output Verbosity of Playbook Execution

OPTION	DESCRIPTION
`-v`	The task results are displayed.
`-vv`	Both task results and task configuration are displayed
`-vvv`	Includes information about connections to managed hosts
`-vvvv`	Adds extra verbosity options to the connection plug-ins, including users being used in the managed hosts to execute scripts, and what scripts have been executed

Dry Run

- ال **Dry Run** احنا بنستخدمها لو احنا عايزين نعمل Run لل Playbook بدون اي تغيير ممكن يحصل الغرض منه ان انا اتاكد ان كل الخطوات هتتنفذ صح قبل ما تبدأ في التغيير الفعلي يعني بالظبط كانه بيعمل محاكاه يعني هيقولك هيحصل ايه لو ال Tasks دي اتنفذت بس من غير مايعدل اي حاجه وده بيسمي وضع ال Check mode
- علشان استخدم الطريقه دي هستخدم اوبشن C- وال C هنا بتكون Capital ي اما تستخدم check--
- هتلاقيه هنا قايلك ال change هيتم فين بالظبط وايه اللي مش هيحصل فيه تغيير ف هو هنا مش هينفذ ال Task فعلا علي ال Host لا هو هيوريك اللي المفروض يحصل ف هو بيقرأ ال Playbook وبعدين يشوف ال Tasks اللي هتتنفذ وبعد كده هيعمل محاكاه هيقولك ايه اللي هيحصل لو ال Tasks دي اتنفذت بس مش هيغير اي حاجه

```
[ansible@control-node test]$ ansible-playbook  -C  site.yml

PLAY [Configure important user consistently] ***********************************

TASK [Gathering Facts] *********************************************************
ok: [node1.example.com]

TASK [Ensure Ahmed exists with UID 4002] ***************************************
changed: [node1.example.com]

PLAY RECAP *********************************************************************
node1.example.com   : ok=2   changed=1  unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
```

مثال عن ال Playbook وهيتم فيه شرح كل ماسبق

- زي ماقولنا ان في بدايه ملف ال playbook لازم ابدأ ب (---) Three dashes ولو احنا مكتبنهمش مش هيأثروا علي اي حاجه وبعد كده اتفقنا ان file ال playbooks بينقسم ل Plays وكل play بتبدأ ان بكتب – dash وبعدين اكتب ال name: وده بيكون اسم او وصف من اختيارنا واحنا هنا في المثال هنعمل Two Tasks واحد هيعمل install للPackage الخاصه بال apache server وال Task التانيه هتعمل Started للapached ف هنكتب مثلاInstall and Start Apache on RedHat بعد كده بنكتب ال hosts: وفايده ال hosts دي ان ده ال section اللي بيبدأ يعرفوا انت هتستخدم وهترن علي انهي group of hosts من ال Inventory file بتاعك ف انا هقوله ان انا عايز ارن علي ال group اللي اسمه servers وبعدين تحت ال hosts بكتب ال Tasks لان زي ماقولنا ان كل Play بتحتوي علي مجموعه من ال Tasks ف كل Task بنبدء نديها name: وده زي ماقولنا بيكون اسم من اختيارنا ف اول task هنعمل install للapache server ف هنكتب في ال name مثلا install apache وتحت ال name تبدأ تختار ال module اللي انت هتستخدمه علشان تعمل installations وده بيعتمد علي انت شغال علي انهي servers يعني ال Module هيختلف من ان انت شغال علي redhat او شغال علي ubuntu ف علي حسب ال case بتاعتك انت هتختار ف علشان انا شغال علي redhat ف هستخدم module اسمه yum وتحت ال yum ابدا اكتب ال Parameters اللي بياخدها ال yum وال yum بياخد ال name وده بيكون فيه اسم ال Package اللي هعملها install وفي حالتي وهي ال httpd ودي ال package الخاصه بال apache server وبياخد كمان parameter اسمه :state وهقوله present ان انا بقوله ان انا عايز ال Package يكون معمول ليها Install
- نفس الكلام هنعمله علي ال Task التانيه وهنستخدم ال module اللي هيخليني اعمل enable و start

```
---
- name: Install and Start Apache on RedHat
  hosts: servers
  tasks:
    - name: Install Apache on RedHat
      yum:
        name: httpd
        state: present

    - name: Ensure Apache is running on RedHat
      service:
        name: httpd
        state: started
        enabled: true
```

- تعال نعمل syntax-check علشان نتاكد هل احنا كاتبين ال playbook صح ولالا وزي ماقولنا طالما رجعلي اسم ال Playbook يبقي كده ال syntax مظبوط

```
[ansible@control-node test]$ ansible-playbook site.yml --syntax-check

playbook: site.yml
```

- بعد كده هنعمل Run لل Playbook ف هو هيبدء يعمل Gathering Facts ودي وظيفتها ان هو هيبدأ يعمل Test لل connectivity مابينه ومابين ال Server وبعد كده هيبدأ يعمل Install لل Apache علي السيرفرات اللي عندي ف هتلاقيه قايلك changed ودي ال state اللي حصلت ان طالما قالك changed يبقي حصل تغيير وعمل install وبعد كده هيعمل enable وهيتاكد ان ال apache شغاله علي السيرفرات اللي عندي

```
[ansible@control-node test]$ ansible-playbook site.yml

PLAY [Install and Start Apache on RedHat] ***************************************************

TASK [Gathering Facts] **********************************************************************
ok: [node2.example.com]
ok: [node1.example.com]

TASK [Install Apache on RedHat] *************************************************************
changed: [node2.example.com]
changed: [node1.example.com]

TASK [Ensure Apache is running on RedHat] ***************************************************
changed: [node2.example.com]
changed: [node1.example.com]

PLAY RECAP **********************************************************************************
node1.example.com  : ok=3  changed=2  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
node2.example.com  : ok=3  changed=2  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

Ansible Variables

- ال **Variables** في Ansible هي طريقة بنستخدمها علشان نعمل store لل Values زي أسماء ال users و البرامج وال Services اللي ممكن نستخدمها في أماكن مختلفه في ال Project.

- **Examples of values that variables might contain include:**
  - ✓ Users to create
  - ✓ Packages to install
  - ✓ Services to restart
  - ✓ Files to remove
  - ✓ Archives to retrieve from the internet

- ال Variables ممكن تحطها في اماكن كتيره جدا
- ال Variables بيكون ليها 3 scope
  - اول scope فيهم ليه علاقه بال Inventory scope
  - تاني scope ليه علاقه بال Plays scope
  - تالت scope وهو Global scope
- اعلي scope فيهم وهو ال Global واقل واحد وهو ال Inventory
- لو انا عايز احط ال Variable في ملف ال Inventory ف هتكون بالشكل ده
  - لو ل Host Variables هنكتب جنب ال host ال variables بالشكل ده

```
[server]
node1.example.com    ansible_user=mohamed
```

  - اما لو لل group variable هنكتبه بالشكل ده

```
[servers]
node1.example.com
node2.example.com

[servers:vars]
user=mohamed
```

  - وعلشان نعرف ال variable ده في ال Playbook هنكتبه بالشكله ده

"{{ variable_name }}"

➢ هنشوف مثال علي ان احنا نكتب variable في ال Inventory ونعرفه في ال playbook

ده ملف ال playbook

```
---
- name: Example Playbook
  hosts: servers
  tasks:
    - name: Create a user
      user:
        name: "{{ user }}"
        state: present
```

ده ملف ال Inventory

```
[servers]
node1.example.com
node2.example.com

[servers:vars]
user=mohamed
```

- في المثال ده احنا عملنا variable في ملف ال Inventory وال variable ده علي ال groupكله اللي هو ال servers وال variable ده عباره عن user وجواه value اسمه Mohamed وفي ملف ال playbook عرفت في ال parameter اللي اسمه name كتبت ال variable "{{ user }}"

- هنشوف مثال تاني بس هنعمل ال variable علي ملف ال playbook الغرض من المثال ده ان انا اعمل playbook اقدر من خلاله ان انا اعمل create ل Directory تحت مسار معين ف علشان ممكن استخدم ال directory ده في اكتر من مكان ف هنستخدم ال varaibles انا ممكن اعمل section اسمه :vars وبعدين ابدأ اديها اسامي ال Variables ف هديها variable اسمه dir_name وال value بتاعته هتكون DevOps وال variable التاني اسمه :dir_path وال value بتاعته هتكون /tmp وعلشان احنا بنعمل create ل Directory هنستخدم ال module المسئول عن عمل ال create وهو اسمه file وال module ده بياخد Parameters زي ال Path وده احنا بنحدد فيه المسار بالكامل اللي هنعمل فيه ال create وبنعرف فيه ال Variable بالطريقه دي " {{ dir_path }}/{{ dir_name }}" وال Parameter التاني اسمه state وده انا بحددله ان انا عايز اعمل create ل Directory

```
---
- name: Create a directory using variables
  hosts: node1.example.com
  become: yes
  vars:
    dir_name: "DevOps"          #directory اسم ال
    dir_path: "/tmp"            #directory المسار اللي هيتعمل فيه ال
  tasks:
    - name: Create a directory
      file:
        path: "{{ dir_path }}/{{ dir_name }}"
        state: directory
    - name:
      command: "ls {{ dir_path }}/{{ dir_name }}"
```

انا هنا اهو استخدمت ال module اللي اسمه file وفي ال Path انا هنا بحدد المثال بالكامل اللي فيه ال Directory ده وعلشان احنا هنعمل لل create ده لل directory تحت /tmp ف كتبنا ال path بالشكل ده {{ dir_path }}/{{ dir_name }} علشان يفهم ان المسار اللي انا عايزه هو /tmp/DevOps

في الاخر هنا انا استخدمت module اسمه command علشان اعمل list لمحتوي ال directory بعد اما يعمل create ال module ده بنحظر ان احنا نستخدمه لان ال module ده مش Idempotent ف احنا بنستخدمه في اضيق الحالات

- زي ماقولنا ان ال module اللي اسمه command بنستخدمه فقط في اضيق الحالات علشان مش idempotent بمعني لو انت شغلت ال playbook اكتر من مره هيفضل ينفذ نفس الامر كل مره حتي لو النتيجه موجوه وهيفضل يقولك changed مش Ok ومش هيطلعلك ال output وبنستخدم ال command لو احنا ملاقناش اي module يقدر ينفذ ال Task اللي احنا عايزينه ف بنروح لل command
- تعال نعمل run لل playbook اول اما عملنا run قالنا انه حصل changed

```
[ansible@control-node test]$ ansible-playbook site.yml
PLAY [Create a directory using variables] ***************************************************

TASK [Gathering Facts] **********************************************************************
ok: [node1.example.com]

TASK [Create a directory] ********************************************************************
changed: [node1.example.com]

TASK [List the contents of the directory] ***************************************************
changed: [node1.example.com]

PLAY RECAP **********************************************************************************
node1.example.com : ok=3  changed=2  unreachable=0  failed=0   skipped=0   rescued=0   ignored=0
```

- لو عملنا run تاني لل playbook هنلاقيه زي ماقولنا انه لسه في ال Task اللي بتعمل list لل contents بتاعت ال Directory اللي استخدمنا فيها ال command لسه ب changed ومش عارضلي ال standard output

```
[ansible@control-node test]$ ansible-playbook site.yml
PLAY [Create a directory using variables] ***************************************************

TASK [Gathering Facts] **********************************************************************
ok: [node1.example.com]

TASK [Create a directory] ********************************************************************
changed: [node1.example.com]

TASK [List the contents of the directory] ***************************************************
changed: [node1.example.com]

PLAY RECAP **********************************************************************************
node1.example.com : ok=3  changed=2  unreachable=0  failed=0   skipped=0   rescued=0   ignored=0
```

Register and Debug

- علشان نقدر نشوف ال standard output(stdout) بتاعت اي Module في Ansible بنحتاج نعمل حاجه اسمها register and Debug
- ال **Register** بيكون Parameters مش مرتبط ب Module معين علشان كده احنا مش بنحطه جوه ال module يعني بيكون علي نفس محاذاة ال name بتاع ال Task ف بنكتب :register وبعدين نديله اسم variable وليكن انا عايز اعمل register في variable اسمه results يعني كأني بقوله ان انت هتخزن ال stdout في variable اسمه results
- وعلشان اقدر ان انا اطبع ال results فيه module تاني اسمه **Debug** وال Debug ده مابياخدش غير Parameters واحد بس ي اما msg ي اما var ي اما verbosity ف احنا هنستخدم ال var وبعدين اديله اسم ال variable اللي انا عايز اعمله debug

➢ لو احنا استخدمنا نفس المثال السابق وعايزين نعرض ال stdout بتاع ال command اللي بيعمل list لل contents بتاعت ال Directory للتنويه انا عملت create ل file باسم MyFile جوه ال Directory علشان لما يعمل list يكون موجود اي file

```
---
- name: Create a directory using variables
  hosts: node1.example.com
  become: yes
  vars:
    dir_name: "DevOps"
    dir_path: "/tmp"
  tasks:
    - name: Create a directory
      file:
        path: "{{ dir_path }}/{{ dir_name }}"
        state: directory
   - name:
      command: "ls {{ dir_path }}/{{ dir_name }}"
      register: results

    - name: print stdout
      debug:
        var: results
```

➢ لو عملنا run لل playbook هنلاقيه عمل Print لل stdout

```
[ansible@control-node test]$ ansible-playbook site.yml

PLAY [Create a directory using variables] ***********************************************************
TASK [Gathering Facts] ******************************************************************************
ok: [node1.example.com]

TASK [Create a directory] ****************************************************************************
ok: [node1.example.com]

TASK [List the contents of the directory] ************************************************************
changed: [node1.example.com]
TASK [print stdout] **********************************************************************************
ok: [node1.example.com] => {
    "results": {
        "changed": true,
        "cmd": [
            "ls",
            "/tmp/DevOps"
        ],
        "delta": "0:00:00.011253",
        "end": "2024-10-21 14:22:29.970564",
        "failed": false,
        "msg": "",
        "rc": 0,
        "start": "2024-10-21 14:22:29.959311",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "MyFile",
        "stdout_lines": [
            "MyFile"
        ]
    }
}
PLAY RECAP *******************************************************************************************
node1.example.com  : ok=4 changed=1  unreachable=0  failed=0    skipped=0    rescued=0    ignored=0
```

هتلاقيه هنا جايبلك تفاصيل عن ال results وبيكون عباره عن json object

هتلاقيه هنا اهو جايبلك ال stdout وهو MuFile

➢ لو احنا عايزين نعرض بس ال stdout من غير باقي التفاصيل الخاصه بال results هستخدم في ال playbook بدل في ال section الخاص بال debug في ال var بدل اما اكتب results بس هكتب results.stdout

```
- name: print stdout
    debug:
      var: results.stdout
```

Interactive input: prompts

- الطريقه دي بستخدمها لو انا عايز اطلب من ال user ان اول اما يعمل Run لل Playbook يطلب منه مثلا Password او Username وهكذا
- لو انت مثلا عايز تدخل قيمه password ف مش هينفع ان انت تكتب ال Password في ال Playbook واي حد يقدر يشوفه ف فيه طريقه ان بدل اما استخدم vars بستخدم section تاني اسمه **vars_prompt** وده بيكون عباره عن section لل variables اللي انت هتدخلهم في ال Run Time من خلال ال CLI وانت بتنفذ ال Playbook
- ف ال section ده بكتب تحتيه ال name وليكن هكتب password وبعدين بكتب Prompt وده انا بكتب فيه ال Massage اللي هو هيطلعهالي وليكن ?What is your password وبعدين بكتب ال Private وده انا بحدد لو انا عايز وانا بكتب الباسورد تظهر في ال Terminal ولا لا ف لو انا اخترت private: yes يبقي كده وانا بكتب مش هيبقي ظاهر ولو كتبت no هيظهر في ال Terminal
- في المثال ده لو نفذناه هيطلب مني قبل اما انفذ ال Playbook ال username وال Password

```
---
- hosts: all
  vars_prompt:
    - name: username
      prompt: What is your username?
      private: yes

    - name: password
      prompt: What is your password?

  tasks:
    - name: Print a message
      ansible.builtin.debug:
        msg: 'Logging in as {{ username }}'
```

- لو نفذنا ال playbook هيطلب مننا ال Username والباسورد بالشكل ده

```
[ansible@control-node test]$ ansible-playbook  site.yml
What is your username?: mohamed
What is your password?:
```

Managing Facts

- هنتكلم عن ال Facts ودي عباره عن مجموعه من ال Variables بت Describe ال Host او الجهاز بتاعي وال Ansible بيبقي عارف هو بيجبها ازاي . اول اما انت بتعمل Run علي Host معين بيروح اول حاجه بيعملها بيعمل Gathering لل Facts ف بيجبلك Variables بتفيدك جدا جدا ان انا اعرف معلومات عن ال Host ده زي مثلا ال Host name وال kernel version وال network interfaces وال IP Addresses وال version الخاص بال Operating system وال Memory وال CPUs وكمان ال devices الموجوده علي ال Host ومعلومات كتيره جدا

➢ علشان اشوف ال Ansible Facts هنستخدم module اسمه setup ف هنلاقي معلومات كتيره جدااا خاصه بال host اللي احنا كاتبينه وفيه معلومات ازيد من كده بكتير

```
[ansible@control-node test]$ ansible   node1.example.com  -m  setup

node1.example.com | SUCCESS => {
   "ansible_facts": {
      "ansible_all_ipv4_addresses": [
         "192.168.1.3"
      ],
      "ansible_all_ipv6_addresses": [
         "fe80::20c:29ff:fe25:3a7"
      ],
      "ansible_apparmor": {
         "status": "disabled"
      },
      "ansible_architecture": "x86_64",
      "ansible_bios_date": "05/22/2023",
      "ansible_bios_vendor": "VMware, Inc.",
      "ansible_bios_version": "VMW201.00V.21805430.B64.2305221830",
      "ansible_board_asset_tag": "NA",
      "ansible_board_name": "440BX Desktop Reference Platform",
  "ipv4": {
            "address": "192.168.1.3",
            "broadcast": "192.168.1.255",
            "netmask": "255.255.255.0",
            "network": "192.168.1.0",
            "prefix": "24"
"ansible_processor": [
         "GenuineIntel",
         "Intel(R) Core(TM) i7-2640M CPU @ 2.80GHz",
```

- ممكن كمان ان احنا بدل اما نعرض كل ال Facts نعمل filter علي حاجه معينه احنا عايزنها وليكن مثلا عايز نعمل filter علي ال ipv4 ف هنستخدم 'filter=ansible_default_ipv4' -a

```
[ansible@control-node test]$ ansible  node1.example.com -m setup -a   'filter=ansible_default_ipv4'
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "192.168.1.3",
            "alias": "ens160",
            "broadcast": "192.168.1.255",
            "gateway": "192.168.1.1",
            "interface": "ens160",
            "macaddress": "00:0c:29:25:03:a7",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "192.168.1.0",
```

- ممكن كمان تعمل filter علي ال dns وال nameserver

```
[ansible@control-node test]$ ansible   node1.example.com   -m   setup   -a   'filter=ansible_dns'
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_dns": {
            "nameservers": [
                "192.168.1.1"
```

- ممكن كمان نعمل filter علي ال Interfaces

```
[ansible@control-node test]$ ansible   node1.example.com   -m   setup   -a   'filter=ansible_interfaces'
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_interfaces": [
            "ens160",
            "lo"
```

- او لو انا عايز اعرف نوع ال kernel اللي علي ال host

```
[ansible@control-node test]$ ansible   node1.example.com   -m   setup   -a   'filter=ansible_kernel'
node1.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_kernel": "5.14.0-284.11.1.el9_2.x86_64",
```

- ف انت تقدر تعمل filter علي اي جزء انت عايزه وفيه امثله كتيره جدااا علي الجزئيه دي ودايما استعمل ال Documentation وانك تعمل search لان كل ده مبيتحفظش المهم تكون فاهم

- ممكن ان احنا نستخدم ال facts في ال Playbook عن طريق ان احنا نستخدم module ال debug مع ال Parameters اللي اسمه msg علشان اطبع رساله
- فيه عندي طريقتين علشان اعرف اكتب ال variable الخاص بال fact ف ال Playbook ي اما اكتبها بالاسلوب ده

```
ansible_facts['default_ipv4']['address']
```

- او الاسلوب ده

```
ansible_facts.default_ipv4.address
```

- ف هتتكتب بالشكل ده في ال Playbook

```
---
- hosts: all
  tasks:
   - name: Prints various Ansible facts
     debug:
       msg: "The default IPv4 address of {{ ansible_facts.fqdn }} is {{ ansible_facts.default_ipv4.address }}"
```

- الجدول ده فيه بعض الامثله عن ال Ansible Fact

FACT	VARIABLE
Short host name	ansible_facts['hostname']
Fully qualified domain name	ansible_facts['fqdn']
Main IPv4 address (based on routing)	ansible_facts['default_ipv4'] ['address']
List of the names of all network interfaces	ansible_facts['interfaces']
Size of the /dev/vda1 disk partition	ansible_facts['devices']['vda'] ['partitions']['vda1']['size']
List of DNS servers	ansible_facts['dns'] ['nameservers']
Version of the currently running kerne	ansible_facts['kernel']

Turning Off Fact Gathering

- لو احنا مش عايزين نستخدم ال Fact Gathering لسبب معين مثلا لو انا عايز ال Taskيتنفذ بسرعه ان انا مش عايزه يعمل الخطوه بتاعت ال TASK [Gathering Facts] وانا بعمل run لل playbook ف انا بعمل كده عن طريق ان انا بقوله gather_facts: no في ملف ال Playbook

```
---
- name: This play gathers no facts automatically
  hosts: large_farm
  gather_facts: no
```

Templates

- ال Ansible من القوه بتاعته ان انت ممكن تعمل Templates ان انت تستعمل Variables وبعد كده تعوض بيها في files . ف ال Ansible بيستخدم حاجه اسمها Jinja2 ال Template فكرتها ببساطه انك اوقات كتيره جدا في ال Ansible code بتاعك ان انت عايز ت configure server انك تعمل edit في ال configuration file او تباصي configuration file بشكل معين وال configuration file بيكون معظمها file كبيره جدا وبيكون انت عايز تغير فيه شويه حاجات بسيطه من server للتاني زي مثلا معلومه زي ال ip او اسم ال host وهكذا ف هو عمل حاجه حلوه انك تحط ال configuration file بتاعتك في ملف واحد
- يعني لو عندك مثلاً 10 سيرفرات وعايز تعمل Configuration file واحد لكن كل سيرفر له إعدادات مختلفة زي الـ IP أو اسم الجهازف انت بدل ما تكتب 10 ملفات بتكتب configuration file واحد بس فيه ال Variables وAnsible هيغير القيم تلقائيا لكل سيرفر
- بنكتب ملف ال template بلغة اسمها Jinja2 وبتحط فيه ال Variables اللي عايز Ansible يستبدلها. لازم ال extensions بتاعته تكون **.j2** بعد كده، لما تشغل الـ Ansible ال playbook بياخد ال Template ده، ويبدل ال Variables بالقيم اللي تخص كل جهاز، وبعدين ينسخه في المكان اللي إنت محدده.
- ال template module في Ansible بيستخدم لنقل ملفات الـ templates من جهاز الـ control الجهاز اللي بتشغل منه Ansible للأجهزة اللي عندي ال Hosts وال Parameters اللي احنا بنستخدمهم مع ال templates هما
  - ال :src وده بيكون مسار ملف templateاللي موجود على جهاز الـ control
  - ال :dest وده بيكون المسار اللي هينسخ فيه الملف النهائي على الجهاز اللي احنا بنحدده
- في المثال ده عملنا ملف لل template واسمه myconfig.j2 وكتبنا فيه variable اسمه name وبعدين في ملف ال playbook ضفنا ال عرفنا ال variable اللي اسمه myname وال value بتاعته ب Mohamed وبعدين ضفنا ال module الخاص بال template وبعدين حددنا ال src وده مسار ملف ال template واسمه myconfig وبعدين حددنا ال dest وده المكان اللي هيتم نسخ فيه الملف واحنا سمينا الملف ده final والمسار تحت /tmp

ده ملف ال playbook واسمه site.yml

```
---
- hosts: node1.example.com
  vars:
    myname: mohamed
  tasks:
  - name: copy template
    template:
      src: myconfig.j2
      dest: /tmp/final
```

ده ملف ال template وهنسميه myconfig.j2

```
name : {{ myname }}
```

Conditions

- ال ansible هو زيه زي اي اسكريبت او Coding language بيديك شويه امكانيات زي ال Programming Language زي ان انت تعمل Condition وبستخدمها عن طريق when وده بيمكنا ان لو الشرط اتحقق ال Task هيتعملها Run ولو متحققتش ال Task مش هيتعمله Run هيعمل skipping لل Task دي
- يعني مثلا ممكن اقوله انه يعمل Run لل Task دي لو ال operating system من ال family بتاعت redhat ف هقوله ده عن طريق when: ansible_facts.os_family == "RedHat" انا هنا استخدمت ansible_facts.os_family لان زي ماقولنا قبل كده ان ال Gathering Fact بيجمعلي معلومات عن ال Host او ال Server ف هو هيشوف ال os_family ايه وهيشوف هينفذ ال Task ولالا ونفس الموضوع في ال Task التانيه

```
---
- hosts: all
  tasks:
    - name: Install package on RedHat systems
      yum:
        name: httpd
        state: present
      when: ansible_facts.os_family == "RedHat"

    - name: Install package on Debian systems
      apt:
        name: apache2
        state: present
      when: ansible_facts.os_family == "Debian"
```

- لو عملنا run لل playbook

```
[ansible@control-node test]$ ansible-playbook   site.yml

PLAY [node1.example.com] ********************************************************************
TASK [Gathering Facts] **********************************************************************
ok: [node1.example.com]
TASK [Install package on RedHat systems] ****************************************************
ok: [node1.example.com]
TASK [Install package on Debian systems] ****************************************************
skipping: [node1.example.com]
PLAY RECAP **********************************************************************************
node1.example.com  : ok=2    changed=0 unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

هتلاقيه هنا عمل Skipping لل Task التانيه لانها محققتش ال condtion

- مثال تاني ممكن ان احنا نباصي Variable ان لو ال Variable ده ب True ينفذ ال Task

```
---
- hosts: all
  vars:
    allow_service_restart: true
  tasks:
    - name: Restart the web service if allowed
      service:
        name: apache2
        state: restarted
      when: allow_service_restart
```

- ممكن كمان ان احنا نستخدم اكتر من condition في نفس ال Task باستخدام ال **and** وال **or**
  - زي ماحنا عارفين ان لو استخدمت ال and معناها ان لازم الشرطين يكونوا متحققين علشان ال Task تتنفذ. ولو استخدمت ال or معناها ان لازم شرط من الاتنين يكون متحقق علشان ال Task تتنفذ

```
---
- hosts: all
  tasks:
    - name: Run task if both conditions are true
      command: echo "Both conditions met"
      when: ansible_facts.os_family == "RedHat" and ansible_facts.default_ipv4.address == "192.168.1.10"

    - name: Run task if any condition is true
      command: echo "One condition met"
      when: ansible_facts.os_family == "Debian" or ansible_facts.default_ipv4.address == "192.168.1.20"
```

- ممكن كمان تعمل condition علشان تشوف لو فيه file موجود مثلا ينفذ ال Task بناءا علي ال file

```
---
- hosts: all
  tasks:
    - name: Check if a file exists
      stat:
        path: /etc/somefile
      register: file_check

    - name: Delete the file if it exists
      file:
        path: /etc/somefile
        state: absent
      when: file_check.stat.exists
```

Loops

- وده احنا بنستخدمه لو عايزين نكرر Task اكتر من مره ب Parameters مختلفه بحيث ان انت تختصر ال Syntax بتاعك يعني بتوفر علينا ان احنا نكتب نفس ال Task اكتر من مره
- يعني لو احنا عايزين نعمل create ل اكتر من user بدل اما نعمل Task لكل user علشان نعمله create لا احنا ممكن نستخدم Task واحده باستخدام ال loop وهيعملي create لكل ال users

➢ يعني مثلا لو احنا عايزين نعمل create لusers بأسم MohamedAtef و AhmedAtef و Abdo هنكتب loop وتحتها هنكتب ال users اللي احنا عايزين نعملهم create بالشكل ده وبعدين هنضيف variable في ال name باسم item ف اللي هيحصل ان ال loop هتباصي كل مره في كل loop القيمه لل variable اللي اسمه item

```
---
- hosts: node1.example.com
  tasks:
    - name: Create multiple users
      user:
        name: "{{ item }}"
        state: present
      loop:
        - MohamedAtef
        - AhmedAtef
        - abdo
```

➢ لو عملنا run لل playbook هنلاقيه بيدينا state لكل item في ال loop ف هتلاقيه بيقولك ان item بتاع MohamedAtef حصله changed و item بتاع AhmedAtef حصله changed وهكذا

```
[ansible@control-node test]$ ansible-playbook  site.yml

PLAY [node1.example.com] ********************************************************************

TASK [Gathering Facts] **********************************************************************
ok: [node1.example.com]

TASK [Create multiple users] ****************************************************************
changed: [node1.example.com] => (item=MohamedAtef)
changed: [node1.example.com] => (item=AhmedAtef)
changed: [node1.example.com] => (item=abdo)

PLAY RECAP **********************************************************************************
node1.example.com  : ok=2  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

➢ ممكن كمان نعمل create ل users مع اضافته في Groups

```
- name: Add several users
  ansible.builtin.user:
    name: "{{ item.name }}"
    state: present
    groups: "{{ item.groups }}"
  loop:
    - { name: 'testuser1', groups: 'wheel' }
    - { name: 'testuser2', groups: 'root' }
```

➢ مثال تاني ممكن ان انت تستخدم ال loop لو انت عايز تعمل install ل اكتر من Package

```
---
- hosts: all
  tasks:
   - name: Install multiple packages
     apt:
       name: "{{ item }}"
       state: present
     loop:
       - git
       - nginx
       - curl
```

➢ **ملاحظة :-**

- فيه اصدارات قديمه من ال Ansible بتستخدم with_items بدل ال loop ف لو مشتغلش معاك اكتب with_items بدل Loop

```
---
- hosts: all
  tasks:
   - name: Install packages
     apt:
       name: "{{ item }}"
       state: present
     with_items:
       - git
       - nginx
       - curl
```

Handlers

- ال Handlers هي عباره عن Tasks بتتكتب بنفس ال Modules وبنفس كل حاجه ولكن ال Tasks دي مش بت Run في ال sequence بتاع ال Code . يعني بت Run لو حصل Change ل Task واحده معينه موجوده في ال Playbook نفسها
- يبقي فكره ال Handlers ان هي عباره عن Tasks بتتكتب في ال Playbook تحت ال section handlers وبتكون مربوطه ب Tasks تانيه ب اي Change يحصل فيها باستخدام ال Parameters اللي اسمه :notify
- يبقي انا بعمل section باسم handlers في نهايه ال Playbook وبعدين تحت ال handlers بضيف ال name ولازم ال name ده يكون مش متكرر يكون فريد علشان هستخدمه في ال notify وعلشان اربط ال handlers بال Task التانيه هستخدم notify ودي انا بكتب فيها اسم ال handlers
- **فيه كذا معلومه لازم تاخد بالك منها**

- ان ال handlers عباره عن Special Kind of Task بمعني ان هي مش بيحصلها Run في ال Playbook الا اذا حد عمل notify واللي عمل notify ده يكون ال Status بتاعته as a task تكون changed متكونش ok لو ادتلنا ok يبقي كده انا مش محتاج ان ال handlers يتعمله run
- ممكن ان احنا نعمل notify ل اكتر من Handlers وال notify بتاخد حاجه واحده بس اوبتاخد List
- ودايما ال Handlers بي run بعد اما ال Tasks كلها بتخلص
- ممكن كمان ان اكتر من Task تنادي علي نفس ال Handlers بس ده مش معناه ان ال Handlers هيتنفذ اكتر من مره

- عادةً بنستخدم الـ handlers عشان نعمل restart لل Services أو reboot للجهاز.

➢ **تعال نشوف مثال :-**

➢ لو هعدل في ال Configuration file بتاعت ال SSH هحتاج اعمل restart لل SSH ف لو انا استخدمت الطريقه العاديه باستخدام ال Task بدون استخدام ال Handlers هيفضل يعمل restart كل اما ننفذ ال Playbook واحنا مش عايزين كده احنا عايزين لما يلاقي تغيير حصل في ال Task الخاصه بال configuration يعمل restart ولو ملقاش اي تغيير ميعملش اي حاجه ف علشان نحل المشكله دي هنستخدم ال handlers

➢ ده كده مثال لو عدلت في إعدادات الـ SSH في ملف الـ /etc/ssh/sshd_config

```
- hosts: node1.example.com
  tasks:
    - name: Update SSH configuration
      copy:
        src: /path/to/sshd_config
        dest: /etc/ssh/sshd_config
      notify: restart ssh

  handlers:
    - name: restart ssh
      service:
        name: ssh
        state: restarted
```

2-انا هنا ضفت ال sections الخاص بال Handler وتحتيه كتبت ال name وسميته restart ssh وزي ماقولنا لازم الاسم ده يكون فريد مش متكرر وبعد كده ضفت ال modules اللي هيخليني اعمل restarted وهو service وده بياخد two parameters بياخد ال name وده اسم ال services وهي ال ssh وبياخد ال state ودي الحاله اللي هينفذها وهي restarted

1- علشان ال handlers يتنفذ لازم اعمل ربط مابين ال handlers ومابين ال Task اللي انا عايز ال handlers يتنفذ لو حصل تغيير في ال Task دي وده عن طريق ان انا بضيف notify في ال Task وبديلها نفس اسم ال handlers

➢ وده مثال لو هعمل update لل nginx ومحتاج ان اعمل restart لل services لو حصل اي تغيير

```
---
- hosts: all
  tasks:
    - name: Install or update nginx
      ansible.builtin.yum:
        name: nginx
        state: latest
      notify: Restart Nginx

  handlers:
    - name: Restart Nginx
      ansible.builtin.service:
        name: nginx
        state: restarted
```

- ده مثال عن اكتر من task بتنادي او بتشاور علي نفس ال handler عن طريق استخدام ال notify في كل Task . وده معناه ان انت ممكن تنادي على نفس الـ handler من اكتر من tasks وال handler هيتنفذ مره واحده بس في النهايه

```
---
- name: Example Playbook
  hosts: localhost
  tasks:
    - name: Install package
      apt:
        name: nginx
        state: present
      notify: Restart Nginx

    - name: Update configuration file
      copy:
        src: my_nginx.conf
        dest: /etc/nginx/nginx.conf
      notify: Restart Nginx

    - name: Remove old package
      apt:
        name: old-package
        state: absent
      notify: Restart Nginx

  handlers:
    - name: Restart Nginx
      service:
        name: nginx
        state: restarted
```

Error handling in playbooks

- في Ansible تتقدر تتحكم في ال errors وال Failures اللي ممكن تحصل وانت بت run الـ tasks عن طريق ان انت ممكن تستخدم **ignore_errors: yes** ودي بيخليني ان انا اتخطي او اتجاهل ال errors اللي ممكن تحصل والاستمرار في تنفيذ ال Tasks في ال Playbook حتي لو حصلت مشكله في Task معينه ف احنا بنستخدمها لو احنا ممكن نتوقع ان ممكن يحصل مشكله في ال Task

➢ تعال مثلا نعمل install ل Package مش موجوده ونشوف لما نعمل run لل playbook هيطلعنا ايه وهل هيكمل ال Task ولا لا انا قايله اعمل Install ل package اسمها ay_kalam وبعد اما يعمل install المفروض لو مفيش اي مشكله هينفذ ال Task اللي بعدها ودي عباره عن ان انا بطبع رساله عباره عن The previous task failed, but this task is still running

```
---
- name: Example Playbook
  hosts: node1.example.com
  tasks:
    - name: install a non-existent package
      yum:
        name: ay_kalam
        state: present

    - name: Continue execution
      debug:
        msg: "The previous task failed, but this task is still running!"
```

➢ لو عملنا run لل Playbook هنلاقيه طلع error ومكملش ال Task

```
[ansible@control-node test]$ ansible-playbook   site.yml

PLAY [Example Playbook] ********************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [node1.example.com]

TASK [install a non-existent package] ******************************************************
fatal: [node1.example.com]: FAILED! => {"changed": false, "failures": ["No package ay_kalam
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}

PLAY RECAP *********************************************************************************
node1.example.com  : ok=1  changed=0  unreachable=0  failed=1  skipped=0  rescued=0  ignored=0
```

هنلاقيه هنا قايلك ان فيه failures ان ال Package اللي اسمها ay_kalam مش موجوده وان ال Task التانيه مش هتتنفذ

هنلاقيه هنا قايلك ان فيه failed واحد

1. Ignoring Failed Commands

- علشان نحل المشكله دي وان لو حصل اي errors زي اللي شوفناها يتم تجاهلها ويكمل باقي Tasks هنستخدم زي ماقولنا **ignore_errors**

➢ ف احنا هنضيف ال **ignore_errors: yes** في ال Playbook بالطريقه دي ان انا بكتبها تحت ال Task اللي انا عايز ان لو حصل مشكله يعمل ignore

```
---
- name: Example Playbook
  hosts: node1.example.com
  tasks:
    - name: install a non-existent package
      yum:
        name: ay_kalam
        state: present
      ignore_errors: yes

    - name: Continue execution
      debug:
        msg: "The previous task failed, but this task is still running!"
```

➢ لو عملنا Run تاني لل Playbook هتلاقيه قايلنا ان حصل error في ال Task الخاصه بال Install وعمل ignoring ليها عمل تجاهل ليها ونفذ باقي ال Task وطلعلنا الرساله اللي كنا كاتبينها

```
[ansible@control-node test]$ ansible-playbook   site.yml

PLAY [Example Playbook] ********************************************************************
TASK [Gathering Facts] *********************************************************************
ok: [node1.example.com]

TASK [install a non-existent package] ******************************************************
fatal: [node1.example.com]: FAILED! => {"changed": false, "failures": ["No package ay_kalam
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
...ignoring

TASK [Continue execution] ******************************************************************
ok: [node1.example.com] => {
    "msg": "The previous task failed, but this task is still running!"
}

PLAY RECAP *********************************************************************************
node1.example.com  : ok=3  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=1
```

هنا اهو حددلك المشكله الموجوده في ال Task

هتلاقيه قايلك ان عمل ignoring لل Task دي

هتلاقيه هنا اهو نفذ ال Task اللي بعدها بدون اي مشكله عمل Print لل msg

هتلاقيه هنا قايلك ان عمل عمليه ignored واحده بس

2. Ignoring Unreachable Host Errors

- بنستخدم الطريقه دي لو احنا عايز يحصل تجاهل او Ignore ل host معين بمعني ان ممكن وانت بتعمل Run ل Playbook ميقدرش انه يوصل لل Host ف لو هو معرفش ي connect مع ال host هيحصل error احنا بقي ممكن نستخدم الطريقه دي علشان يتجاهل الhost ده ويكمل ال Task علي باقي ال Hosts اللي عندي ف بنستخدم **ignore_unreachable: yes** في ال Playbook

➢ في المثال ده انا قايله اعمل Task بتعمل Ping علي ال Hosts اللي عندي وقايلك كمان ان لو معرفتش توصل ل Host معين اعمله Ignore وكمل باقي ال Task

```
---
- name: Playbook with ignore_unreachable
  hosts: all
  ignore_unreachable: yes
  tasks:
    - name: Ping hosts
      ping:
```

➢ لو نفذنا ال Playbook هنلاقيه عمل Ignore لل host اللي معرفش يوصله

```
[ansible@control-node test]$ ansible-playbook  site.yml

PLAY [Playbook with ignore_unreachable] ****************************************************************
هتلاقيه هنا اهو في ال Gathering Facts انه وهو بيجمع معلومات عن ال Hosts مقدرش انه يوصل لل host اللي اسمه node2.example.com عن طريق ال SSH وعمل ليها ignoring
TASK [Gathering Facts] *********************************************************************************
fatal: [node2.example.com]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host node2.example.com port 22: No route to host", "unreachable": true}
...ignoring
ok: [node1.example.com]
وهنا قالك node1 ب ok

TASK [Ping hosts] **************************************************************************************
وهنا في الجزء الخاص بال Task قالك ان ال Task اتنفذت علي node1 بدون اي مشكله
ok: [node1.example.com]
fatal: [node2.example.com]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host node2.example.com port 22: No route to host", "unreachable": true}
...ignoring
وبالنسبه ل node2 حصل error لانه معرفش يوصله وبعدين عمله ignoring

PLAY RECAP *********************************************************************************************
node1.example.com  : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
node2.example.com  : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=2
```

3. failed_when

- احنا بنستخدم الطريقه دي لو احنا عايزين نحط condition معين لو حصل اي error في Task معين بناءا علي ال نتائج ال Task او ال Output يعمل failed لل Task

➢ في المثال ده احنا حطينا condition باستخدام ال failed_when ان لو result.stdout ما فيهاش كلمة 'World' اعتبر ال Task دي Failed

```
---
- name: Example Playbook using failed_when
  hosts: localhost
  tasks:
    - name: Run a command
      command: /bin/echo "Hello, World!"
      register: result

    - name: Fail the task if output does not contain 'World'
      debug:
        msg: "Command succeeded, and output contains 'World'."
      failed_when: "'World' not in result.stdout"
```

➢ لو نفذنا ال Playbook

```
[ansible@control-node test]$ ansible-playbook site.yml

PLAY [Example Playbook using failed_when] ***************************************************

TASK [Gathering Facts] **********************************************************************
ok: [localhost]

TASK [Run a command] ************************************************************************
changed: [localhost]

TASK [Fail the task if output does not contain 'World'] *************************************
ok: [localhost] => {
    "msg": "Command succeeded, and output contains 'World'."
}

PLAY RECAP **********************************************************************************
localhost    : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. max_fail_percentage

- بستخدم ال **max_fail_percentage** لو احنا عايزين نحدد نسبة ال Hosts المسموح ليهم انهم يحصلهم error او Failure قبل ايقاف تشغيل ال Playbook بشكل كامل

➢ بتكون بالشكل ده ان احنا بنكتب ال Max_fail_percentage وبعدين بنحدد النسبه اللي احنا عايزنها ف في المثال احنا كاتبين ان النسبه 20 يعني لو ال Task حصل فيها مشاكل في ال % 20 اعمل ايقاف لل Playbook

```
---
- name: Example Playbook with max_fail_percentage
  hosts: all
  max_fail_percentage: 20
  tasks:
    - name: Install package
      apt:
        name: nginx
        state: present
```

5. any_errors_fatal

- احنا بنستخدم any_erroes_fatal ان لو حصل اي مشكله في اي Host هيحصل ايقاف لل Playbook كلها علي كل ال Hosts مره واحده

```
---
- name: Example Playbook with any_errors_fatal
  hosts: all
  any_errors_fatal: yes
  tasks:
    - name: Run a command that fails
      command: /bin/false

    - name: This task will not run if any host fails
      debug:
        msg: "This task is skipped due to a fatal error."
```

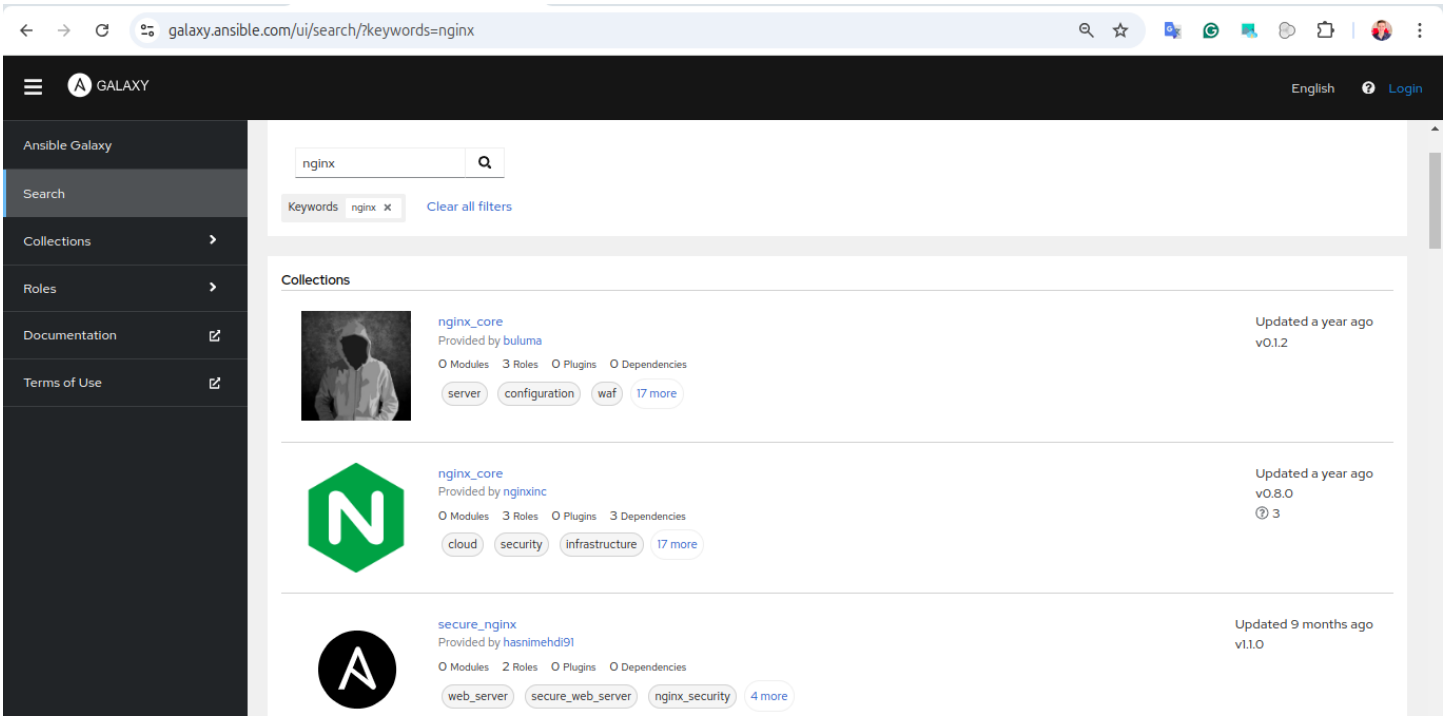
Roles

- ال Roles هي شبه ال Modules في ال Terraform او في ال Programming فكرتها ببساطه انك تنظم التاسكات بتاعتك كلها في اماكن closed يقدر اي حد يستخدمها في اكتر من Project وتقدر تعملها Publish في ال Ansible Galaxy
- ال Roles بتعتمد علي Structure معين من ال Folders ف انا بعمل role لكل حاجه عندي في الكود
- ف ال Role عباره عن Folder وجوه ال folder مجموعه من ال folders متقسمه علي حسب ال use case بتاعتي
- وليكن مثلا هعمل Role لل web مثلا ف هبدء اعمل folder باسم web
  - وبعدين ال web ده هيكون جواه مجموعه من ال Folders
  - ف هبدأ اعمل Folder لل tasks وجوه tasks بنعمل file اسمه main.yml وفي ال main.yml ببدأ ان انا احط فيه ال Tasks بتاعتي كلها
  - وبنعمل folder لل files وده احنا بنحط فيه اي file بنعمله copy
  - وكمان بنعمل folder لل templates وبنحط فيها اي template عندي
  - وبيكون عندنا folder لل vars وجوه vars بيكون موجوده file ب اسم main.yml وبنحط في ال file ده كل ال variables اللي عندي
  - وكمان بنعمل folder لل handlers وجواه بيكون فيه main.yml وفي ال main.yml بكتب ال Tasks بتاعت ال handlers
  - ف بيكون ال Structure بالشكل ده وهنشوف ازاي بال Ansible Galaxy هنعمل create لل Roles دي علي طول وتكون جاهزه علشان اكتب فيها الكود بتاعي

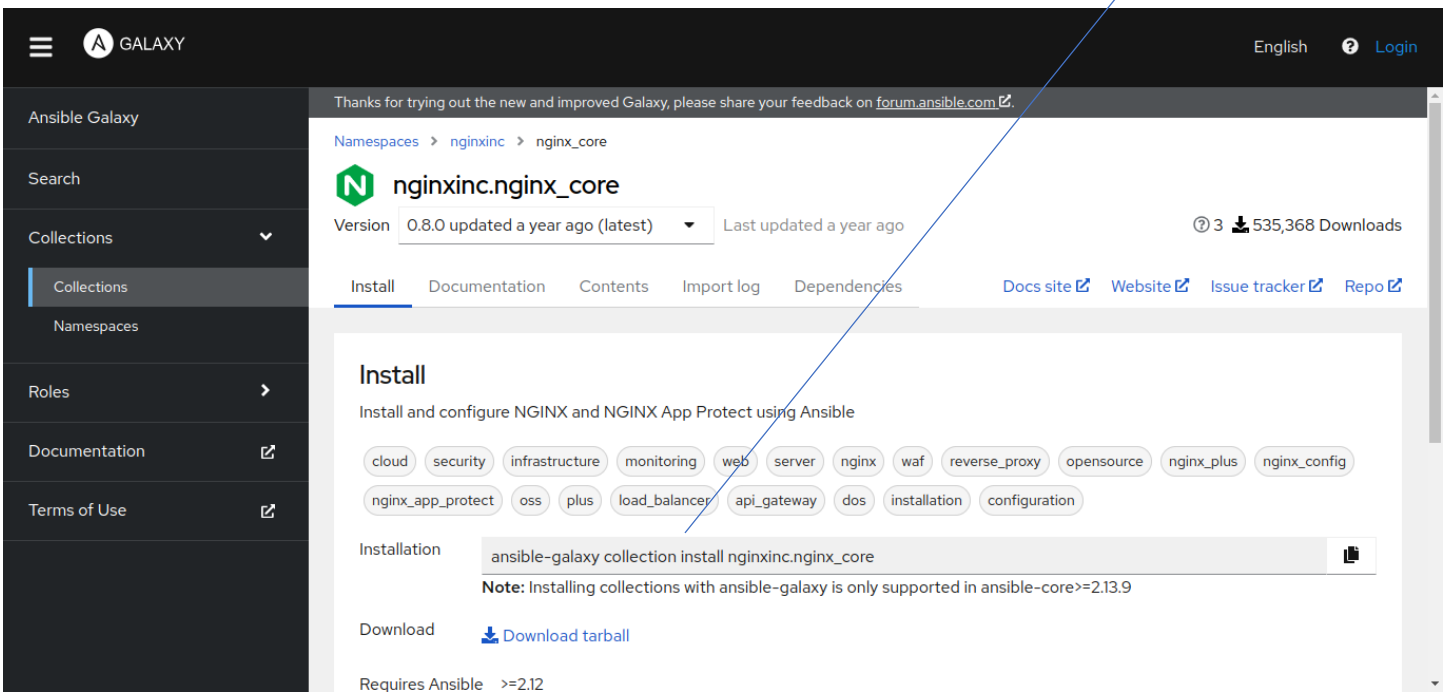
```
[user@host roles]$ tree user.example
user.example/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Ansible Galaxy

- ال Ansible Galaxy هي شبه ال Docker Hub وال GitHub ف هتلاقي ناس كاتبه عليها ansible roles جاهزه تقدر تعملها Install وتستخدمها وكمان تقدر تعمل Publish
- وده كده شكل ال Web Site الخاص بال Ansible Galaxy



- ف انت تقدر تختار ال Package وتعملها install عن طريق ال Command اللي بيكون موجود لما تفتح ال Package



- هنشوف ازاي نستخدم ال Ansible Galaxy command علشان يعملنا Create لل Folder Structure الخاص بال Roles بحيث ان انا ابدأ اشتغل فيه واكتب الكود بتاعي
- وبنعمل ده عن طريق ان احنا بنقوله ansible-galaxy وبعدين init وبعدين اسم ال Role

```
[ansible@control-node test]$ ansible-galaxy   init    my_role
- Role my_role was created successfully
```

- لو عملنا ls لل role اللي اسمها my_role خلي بالك ال role دي عباره عن folder

```
[ansible@control-node test]$ cd my_role/
[ansible@control-node my_role]$  ls
defaults   files   handlers   meta   README.md   tasks   templates   tests   vars
```

- ولازم في ملف ال Playbook اضيف ال section الخاص بال rules وبيكون بالشكل ده

```
---
- name: Deploy Nginx
  hosts: web_servers
  roles:
    - my_role
```

بكتب :roles وتحتها بكتب اسم ال role اللي هو ال folder اللي فيه كل ال Folders

Managing Secrets

- من ضمن المواضيع المهمه جدا في ال Ansible وهي ال Managing Secrets ان انا ازاي اعمل security لل Data بتاعتي ومن ضمن الطرق اللي بنستخدمها وهي ال Ansible Vault
- **Ansible Vault** is a powerful tool for protecting sensitive data within Playbooks. It allows you to easily encrypt and decrypt data, ensuring the security of important information while using Ansible.
- ال Ansible Vault دي الاداه اللي انا بستخدمها علشان اعمل encrypt و decrypt لل data

• Creating an Encrypted File

➢ علشان اعمل create ل Encrypted File بستخدم الامر ده وبيطلب منك ال password اللي هتعملها لل file ده اول اما تدخله الباسورد هيفتح معاك ال File علي طول

```
[ansible@control-node ~]$ ansible-vault   create   secret.yml
New Vault password: 123456
Confirm New Vault password: 123456
```

➢ لو احنا فتحنا ال file ده ب اي editor زي مثلا vim او عملنا cat علشان نعرض محتوي الملف هتلاقي ال file ده متشفر بالشكل ده

```
$ANSIBLE_VAULT;1.1;AES256
30356232363231316233666538353335613562316664356137343566623730373266346130613032
39353733666366616431386639366361363339623831366 10a32663035396434386533636346362
39376433623130663236396639316361623734626537313037623361333237613065306362663961
6132323939306462350a313266633764356230313634626330616239623239383833616465303131
6137
```

• Viewing an Encrypted File

➢ علشان نشوف محتوي ال file ده هنستخدم الامر ده ف بيطلب منك الباسورد وبعدين بيعرضلك محتوي الملف

```
[ansible@control-node ~]$ ansible-vault   view   secret.yml
Vault password: 123456
Hello World
```

- **Editing an Existing Encrypted File**

➢ علشان نعمل edit لل file هنستخدم

```
[ansible@control-node ~]$ ansible-vault  edit  secret.yml
```

- **Encrypting an Existing File**

➢ لو انا عايز اعمل encrypt ل File موجود

```
[ansible@control-node ~]$ ansible-vault  encrypt  secret1.yml
New Vault password: 123456
Confirm New Vault password: 123456
Encryption successful
```

- **Encrypting an Existing File**

➢ لو انا عايز اعمل decrypt ل file

```
[ansible@control-node ~]$ ansible-vault  decrypt  secret1.yml
Vault password: 123456
Decryption successful
```

- **Changing the Password of an Encrypted File**

➢ علشان اغير باسورد ملف معمول ليه Encrypt

```
[ansible@control-node ~]$ ansible-vault rekey secret.yml
Vault password: 123456
New Vault password: 123
Confirm New Vault password: 123
Rekey successful
```

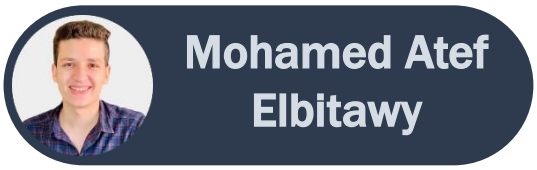
Playbooks and Ansible Vault

- لو انا عندي ملف ال Playbook معمول ليه Encrypt وجيت انك تعمله Run هيظهرلك error

```
[ansible@control-node test]$ ansible-playbook   site.yml
ERROR! Attempting to decrypt but no vault secrets found
```

- علشان اقدر اعمل Run ل Playbook معمول ليه Encrypt هستخدم اوبشن --ask-vault-pass

```
[ansible@control-node test]$ ansible-playbook   --ask-vault-pass   site.yml
Vault password:123456
```



شرح Ansible

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy