Ansible

Notları
Haziran 2023
Kazim Esen

Faydalanılan kaynaklar;

https://docs.ansible.com/
https://medium.com/devopsturkiye/ansible-nedir-dosya-yap%C4%B1s%C4%B1-nas%C4%B1ld%C4%B1r-nas%C4%B1l-kullan%C4%B1l%C4%B1r-4d8c90cdb266
https://www.linuxteknik.com/
https://intellipaat.com/blog/ansible-playbook/?US
https://spacelift.io/blog/ansible-tutorial
https://www.middlewareinventory.com/
https://www.ansiblepilot.com/articles/write-a-variable-to-a-file-ansible-module-copy-vs-template/
https://linuxhint.com/copy-multiple-files-ansible/
https://devopscube.com/ansible-playbook-examples/
https://www.digitalocean.com/community/tutorial_series/how-to-write-ansible-playbooks

Ansible

Ansible, birden fazla sunucuyu tek bir merkezden yönetmeyi ve kontrol etmeyi sağlayan, bunu yaparken herhangi bir agent kurma ihtiyacı gerektirmeyen, Python ve Ruby dilleri ile geliştirilmiş, RedHat'ın ücretsiz ve açık kaynaklı bir otomasyon platformudur. Gerçekleştirilmesi gereken birden fazla ve tekrarlayan görevler olduğunda özellikle idealdir. Uzak bilgisayarların her birine giriş yapıp tek tek görevleri yapmak yerine, bir merkezden tüm bilgisayarların yönetilip, kontrol edilebilmesini sağlar. Uygulama dağıtımında tutarlılığı korur, insan hatasını azaltır, tekrarlayan sıradan görevleri otomatikleştirir. Ansible'ın Puppet, Chef ve Salt gibi başka alternatifleri de var. Ancak Ansible kullanımı kolay ve öğrenmesi basit olduğu için tercih edilmektedir.

Ansible konfigürasyon ve otomasyon için YAML (Yet Another Markup Language) kullanır. Uzak sunucularla iletişim kurmak için bir aracı (agent) yükleme gerektiren diğer otomasyon platformlarının aksine, SSH veya Powershell remoting kullanır.

Ansible temel kavramları:

Inventory

Envanter; yönetilen, yapılandırılan ve kontrol edilmek istenen sunucuların veya düğümlerin listesini içeren bir metin dosyasıdır. Genellikle sunucular, bilgisayar adlarına veya IP adreslerine göre listelenir.

```
10.20.30.40
10.20.30.41
10.20.30.42
```

Bir envanter dosyası, uzak sistemlerin IP adreslerini içerebilir.

```
[webservers]
10.20.30.50
10.20.30.51
10.20.30.52

[dbservers]
10.20.40.60
10.20.40.61
```

Bir envanter dosyası, alternatif olarak, gruplar şeklinde de oluşturulabilir. Sunucular, web ve db şeklinde 2 grup altına yerleştirilmiştir. Bu şekilde olduğunda, grup adlarına görede başvurulabilir. Bu, operasyon süreçlerini daha da basitleştirir. Büyük bir üretim ortamında, birden çok sunucuya sahip birden çok grup olabilir.

Playbook

Bir Ansible çalışma kitabı, Ansible otomasyon aracı tarafından yönetilen bir sunucu yapılandırması için çalışmayı tanımlayan organize bir betik birimidir.

Ansible, playbook'ları kullanarak birden çok sunucunun yapılandırmasını otomatikleştiren bir yapılandırma yönetim aracıdır. Playbook, herhangi bir Ansible yapılandırmasının temel bileşenidir. Bir Ansible playbook'u, her biri yönetilen bir sunucudaki bir yapılandırma için yapılacak işi tanımlayan bir veya daha fazla play içerir. Ansible play'leri YAML'da yazılır. Her play, hedef makineler için ortama özgü parametrelerle bir yönetici tarafından oluşturulur; standart play'ler yoktur.

Ansible play'leri, hedef yönetilen sunucuların çeşitli yönleriyle ilgili olan modüller sayesinde esnektir. Modül betiği Ruby ile yazılmıştır. Yazılım kurulumu ve kullanıcı yönetimi dahil olmak üzere sistem yapılandırmasının birçok bölümü için Modüller mevcuttur. Bir Red Hat şirketi olan Ansible, Ansible'ı kullanan ve destekleyen açık kaynak topluluğu gibi birçok modül sağlar.

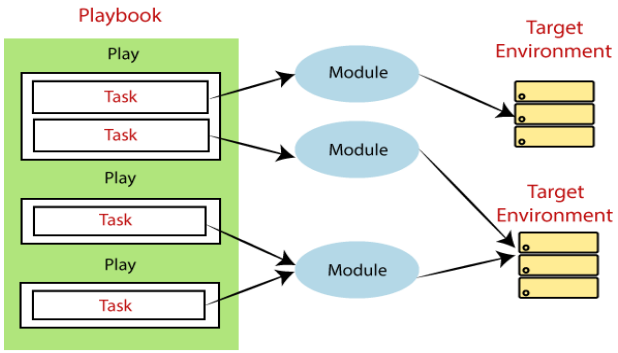
Oyun kitabı bu nedenle modüllerden oluşan oyunlardan oluşur. Yönetici `ansible-playbook` komutunu hedef makinelere karşı çalıştırdığında yürütülür. Yönetici, oyun kitabının yönetimi altındaki hostları belirtmek için bir envanter dosyası kullanmalıdır. Envanter dosyası, Ansible tarafından yönetilen tüm hostların bir listesini içerir ve hostları işlevlerine göre gruplandırma seçeneği sunar. Örneğin, bir yönetici playbook'daki bir grup web sunucusuna bir play ve bir grup veritabanı sunucusuna farklı bir play uygulayabilir. Adı httpd olan bir çalışma kitabı oluşturmak için; `nano httpd.yml` komutu ile bir YAML dosyası oluşturulur. YAML her zaman --- (3 tire) ile başlar ve sırasıyla talimatlar eklenir.

```
---
- name: This installs and starts Apache webserver
  hosts: webservers
  tasks:
  - name: Install Apache Webserver
    yum:
      name=httpd
      state=latest
  - name: check httpd status
    service:
      name=httpd
      state=started
```

Bu çalışma kitabı, envanter dosyasında `webservers` olarak tanımlanan sistemlere Apache web sunucusunu kurar. Ansible daha sonra Apache web sunucusunun başlatılıp başlatılmadığını yani çalışıp çalışmadığını kontrol eder.

Play

Ansible play, bir sunucuda gerçekleştirilecek görevi tanımlayan bir komut dosyası veya talimattır. Başka bir deyişle, bir playbook, her biri bir sunucuda gerçekleştirilecek görevi açıkça şart koşan birden çok komuttan oluşan bir koleksiyondur.



Bir oyun kitabı, sıralı bir listedeki bir veya daha fazla "oyun"dan oluşur.
"Oyun kitabı" ve "oyun" terimleri spor analojileridir.
Her oyun, bir veya daha fazla görev yürüterek
oyun kitabının genel hedefinin bir bölümünü yürütür.
Her görev bir modülü çağırır.

Modules

Modüller, yönetilen hostlarda komutları yürütmek için playbook'larda kullanılan ayrı kod birimleridir.
Her modülü bir bağımsız değişken takip eder.

```
- name: Install apache packages
  yum:
    name=httpd
    state=present
```

Bir modülün temel formatı `anahtar: değer` şeklindedir.
Yandaki YAML dosyasında, `-name` ve `yum` modüllerdir.

Variables

Temel olarak, bir değişken bir değeri temsil eder. Değişkenler, talimatlar bir sistemden diğerine değiştiğinde kullanılır. Bu, özellikle yapılandırma veya çeşitli hizmetler ve özellikler sırasında geçerlidir. 3 ana değişken türü vardır:

- `Playbook variables`
- `Inventory variables`
- `Special variables`

```
vars:
Var name1: 'My first variable'
         Var name2: 'My second variable'
```

Ansible'da değişkenler önce `vars:` ifadesi kullanılarak tanımlanır, ardından değişken adı ve değer gelir. Sözdizimi yanda gösterildiği gibidir.

```
- hosts: webservers
  vars:
    - web_directory:/var/www/html/
```

Bu örnekte, değişken `web_directory`'dir ve ansible'a `/var/www/html/` dizin yapısını oluşturma talimatı verir.

Facts

Ansible'ın bir hostta bir playbook yürüttüğünde toplanan sistem özellikleridir. Özellikler, hostname'i, OS ailesini, CPU tipini ve CPU çekirdeklerini içerir. Kullanılabilecek facts'lara göz atmak için; `ansible localhost -m setup`

```
[kazm@centosAnsible ~]$ ansible localhost -m setup
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.0.120"
```

facts, filtre ile daraltılabilir: `ansible localhost -m setup -a "filter=*ipv4"`

Yapılandırma Dosyaları : Ansible yapılandırma dosyası; ayarların yapıldığı, parametrelerin belirlendiği dosyadır. Varsayılan yapılandırma dosyası, `/etc/ansible/ansible.cfg` 'dir.

Ansible Kurulumu

Ansible'ı CentOS'a kurmak için;
(diğer dağıtımlar için https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html faydalanılabilir)

```
$ sudo yum -y install epel-release
$ sudo yum -y install ansible
```

Komutları Başarılı olarak tamamlandıktan sonra, ansible sürümü kontrol edilebilir;

```
[kazm@centosAnsible ~]$ ansible --version
ansible [core 2.14.2]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/kazm/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.11/site-packages/ansible
  ansible collection location = /home/kazm/.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
[kazm@centosAnsible ~]$
```

Ansible Sunucu ve İstemci Arasında Parolasız SSH Yapılandırması

Ansible sunucuda ssh anahtarı oluşturmak ve bu anahtarı ansible client üzerindeki public key'e kopyalamak gerekir.

```
[kazm@centosAnsible ~]$ ssh-keygen -t rsa -b 4096 -C "kazm@192.168.0.120"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kazm/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kazm/.ssh/id_rsa
Your public key has been saved in /home/kazm/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BQzjtexfstlxQMXMiabzQx4YJhX78D8VnftZxZd+eh8 kazm@192.168.0.120
The key's randomart image is:
   ....
[kazm@centosAnsible ~]$
```

Bu anahtarın ansible sunucudan ansible client'lara `ssh-copy-id` komutu ile kopyalanmalıdır.
Ansible sunucusunda ve ansible istemcisinde SSH ortak anahtarının kopyalanması gerekir.

```
$ ssh-copy-id kazm@192.168.0.120
$ ssh-copy-id kazm@192.168.0.100
....
```

Windows 10'dan Linux hosta parola girmeden bağlanmak için;

```
type $env:USERPROFILE\.ssh\id_rsa.pub | ssh kazm@192.168.0.120 "cat >> .ssh/authorized_keys"
```

Ansible Yönetim Merkezinde `hosts` Dosyası Yapılandırması

Ansible'ın yüklendiği host **yönetim merkezi** rolündedir ve hosts dosyası tüm yönetilen hostların envanterini içerir. Bu dosya, ansible sunucuda `/etc/ansible/hosts` yolunda olabilir ve tüm yönetilen hostname veya IP'leri içermelidir:

```
[kazm@centosAnsible ~]# vi /etc/ansible/hosts
[ansible_clients]
192.168.0.100

[ansible_servers]
192.168.0.120
[kazm@centosAnsible ~]$
```

Statik ve Dinamik Envanterler

Envanter dosyası, yönetilen hostların adlarından veya IP adreslerinden oluşan bir metin dosyasıdır. Yönetilen hostlar, bir grup adı altında kategorize edilebilir. Ansible'da iki tür envanter dosyası vardır: Statik ve Dinamik.

Kontrol düğümüne Ansible yüklü ve yönetilen hostlara Parolasız SSH bağlantısı yapılandırılmış olmalıdır.

Statik Envanter Dosyası

Grub adı altında; host adları veya IP adresleri kullanılarak oluşturulan yönetilen hostların listesini içeren bir düz metin dosyasıdır. Grub adı köşeli parantez içine, yani `[grup_adı]` alınır. Statik envanter dosyası oluşturma;

```
[kazm@centosAnsible ~]$ mkdir test_lab && cd test_lab && nano hosts
[www_servers]
10.20.30.100

[db_servers]
10.20.40.100

[verimerkezi:children]
www_servers
db_servers
```

Dosya kaydedilir ve çıkılır. Bu envanter dosyasında grublar oluşturuldu: `www_servers` ve `db_servers`. Ayrıca, '`:children`' soneki ile gösterilen bir grup grubunu içeren, `verimerkezi` adlı ek bir grup oluşturuldu.
Ansible, grupların bir grup adı altına yerleştirilmesine de izin verir. Yukarıdaki envanter dosyasında `verimerkezi` altına `www_servers` ve `db_servers` grupları yerleştirilmiştir.

Host envanter dosyasına başvurmak için envanter yönetimi komut sözdizimi;

```
$ ansible {host-pattern} -i /path/of/inventory/file --list-hosts
```

```
[kazm@centosAnsible ~]$ ansible all -i /home/kazm/test_lab/hosts --list-hosts
  hosts (2):
    10.20.30.100
    10.20.40.100
[kazm@centosAnsible ~]$
```

Alternatif olarak, '`all`' argümanı yerine " `*` " joker karakteri de kullanılabilir.

```
$ ansible '*' -i /home/kazm/test_lab/hosts --list-hosts
```

Bir gruptaki hostları listelemek için, host-pattern parametresinde grub adı belirtilmelidir.

```
[kazm@centosAnsible ~]$ ansible www_servers -i /home/kazm/test_lab/hosts --list-hosts
  hosts (1):
    10.20.30.100
[kazm@centosAnsible ~]$
```

Dinamik Envanter Dosyası

AWS gibi envanter dosyasının sunucu ekleyip çıkardıkça sürekli değiştiği bir bulut ortamında, envanter dosyasında tanımlanan hostları takip etmek zordur. IP adreslerini güncellemenin zahmetli hale geldiği bu noktada işin içine dinamik envanter girer.

Dinamik envanter Python, PHP veya başka bir programlama dilinde yazılmış bir betiktir. Sanal bir sunucu durdurulup yeniden başlatıldığında IP adreslerinin değiştiği AWS gibi bulut ortamlarında kullanışlıdır.

Ansible, Google Compute Engine, Amazon EC2 bulut sunucusu, OpenStack, RackSpace, cobbler gibi genel bulut platformları için envanter komut dosyaları geliştirmiştir.

Dinamik envanterin statik envantere göre avantajları;
- Bilgiler betikler kullanılarak toplandığından, dinamik envanterler insan hatasını azaltmada mükemmeldir.
- Envanter yönetiminde minimum çaba gerekir.

Seçilen bir programlama dilinde özelleştirilmiş dinamik envanter yazılabilir. Envanter, uygun seçenekler iletildiğinde JSON'da bir biçim döndürmelidir.

Ansible Ad-Hoc Komutları

Ad hoc Latince'de çok kesin ve özel bir amaç için yapılan bir şey demektir. Ansible ad hoc komutları çok özel bir görev için yazılır ve tek bir görevi bir veya daha fazla yönetilen düğümde çalıştırmanın hızlı bir yoludur. Bir playbook oluşturmadan uzak hostlarda hızlı ve basit görevler gerçekleştirir. Ansible ad-hoc komutu, playbook oluşturmaya gerek kalmadan bazı görevleri basit ama verimli bir şekilde yürütmeye yardımcı olan tek satırlık bir komuttur. Bu tür görevler, bilgisayarlar arasında dosya kopyalama, sunucuları yeniden başlatma, kullanıcı ekleme ve kaldırma ve tek bir paket yüklemeyi içerir.

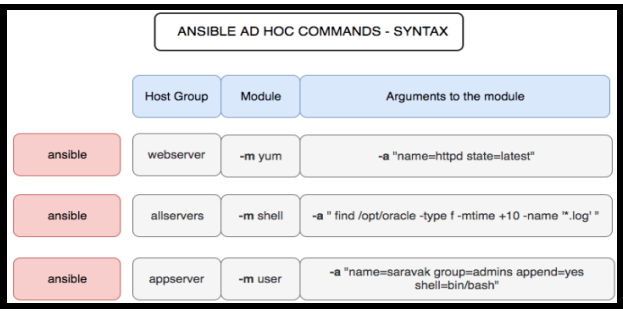
```
$ ansible [host-listesi] -m [modül] -a "[modül seçenekleri]"
```

host-listesi: komutun çalıştırılacağı yönetilen host(lar)
-m: çalıştırılacak modül
-a: modülün gerektirdiği bağımsız değişkenlerin listesi

```
[ansible_clients]
192.168.0.100

[ansible_servers]
192.168.0.120
```

Ansible Ad-Hoc komutlarının bazı uygulamalarını göstermek için kullanılacak envanter (`/etc/ansible/hosts`) dosyası içeriğidir. Adhoc komutlarının en temel kullanımı, bir bilgisayara veya bilgisayar grubuna `ping` atmaktır.

```
[kazm@centosAnsible ~]$ ansible all -m ping
192.168.0.100 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.0.120 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[kazm@centosAnsible ~]$
```



İlk parametre olan `all`, dosyadaki tüm hostları temsil eder ve ikinci parametre olan `-m` modül seçeneği, `ping` komuttur. Belirli bir bilgisayar grubuna ping atmak için, `all` parametresi grup adıyla değiştirilmelidir, `ansible_clients` grubu altındaki hostlara ping atmak için;

```
[kazm@centosAnsible ~]$ ansible ansible_clients -m ping
192.168.0.100 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[kazm@centosAnsible ~]$
```

Standart Linux komutlarını çalıştırmak için `-a` parametresi ile komut çift tırnak içine alınır.
Mesela, uzak sistemlerin sistem çalışma süresini kontrol etmek için;

```
[kazm@centosAnsible ~]$ ansible all -a "uptime"
192.168.0.100 | CHANGED | rc=0 >>
 09:46:46 up 46 min,  2 users,  load average: 0.04, 0.03, 0.00
192.168.0.120 | CHANGED | rc=0 >>
 09:46:46 up 57 min,  2 users,  load average: 0.01, 0.00, 0.00
[kazm@centosAnsible ~]$
```

`# ansible-doc -l`	Adhoc komutuyla kullanılabilecek tüm modülleri açıklamalarıyla birlikte listeler.
`# ansible-doc module_name`	Belirli bir modül hakkında ayrıntılı bilgi görüntüler.
`# ansible-doc yum`	yum modülünün kullanımı ile ilgili ayrıntılı bilgi sağlar.

Ansible ile Paketleri / Hizmetleri Yönetme

Ansible adhoc komutları, yum ve apt paket yöneticileri kullanılarak paketlerin yüklenmesi ve kaldırılması için kullanılabilir.
Envanter dosyasındaki ansible_clients grubudaki "Ubuntu 22.04.1 LTS" hostuna Apache web sunucusunu kurmak için;

```
[kazm@centosAnsible ~]$ ansible ansible_clients -m shell -a "apt update && apt -y install apache2" -b -K
BECOME password:
192.168.0.100 | CHANGED | rc=0 >>
Hit:1 http://tr.archive.ubuntu.com/ubuntu jammy InRelease
    ...
[kazm@centosAnsible ~]$
```

Ansible ile Apache'yi kaldırmak için `install` yerine `purge` girilir.

```
$ ansible ansible_clients -m shell -a "apt -y purge apache2" -b -K
```

Ansible ile Kullanıcı ve Grup Oluşturma

Kullanıcı oluşturmak için, '`user`' modülü kullanılır. `ansible_clients` hostlarında parolası ***p4r0l4*** olan ***jale*** isimli yeni bir kullanıcı oluşturmak için;

```
[kazm@centosAnsible ~]$ ansible ansible_clients -m user -a "name=jale password=p4r0l4" -b -K
BECOME password:
[WARNING]: The input password appears not to have been hashed. The 'password' argument must be encrypted for this module to work properly.
192.168.0.100 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1001,
    "home": "/home/jale",
    "name": "jale",
    "password": "NOT_LOGGING_PASSWORD",
    "shell": "/bin/sh",
    "state": "present",
    "system": false,
    "uid": 1001
}
[kazm@centosAnsible ~]$
```

Yeni kullanıcının oluşturulduğunu doğrulamak için;

```
[kazm@centosAnsible ~]$ ansible ansible_clients -a "id jale"
192.168.0.100 | CHANGED | rc=0 >>
uid=1001(jale) gid=1001(jale) groups=1001(jale)
[kazm@centosAnsible ~]$
```

`ansible ansible_clients -m user -a "name=jale state=absent" -b -K`	Kullanıcıyı siler.

Ayrıcalık Yükseltme

Ansible, –b (become) seçeneğini ve parola istemek için de -K seçeneğini kullanarak ayrıcalık yükseltir, root ayrıcalığı elde eder. '`netstat -pnltu`'yu ayrıcalıklı olarak çalıştırmak için, –b ve root parolasını sormak için -K seçeneğinin kullanımı;

```
[kazm@centosAnsible ~]$ ansible ansible_clients -m shell -a "netstat -pnltu" -b -K
BECOME password:
192.168.0.100 | CHANGED | rc=0 >>
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      694/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      765/sshd: /usr/sbin
tcp6       0      0 :::22                   :::*                    LISTEN      765/sshd: /usr/sbin
udp        0      0 127.0.0.53:53           0.0.0.0:*                           694/systemd-resolve
[kazm@centosAnsible ~]$
```

Başka bir kullanıcı olmak için `--become-user` seçeneği kullanılır.

Örneğin, `jale` kullanıcısı olarak '`df -Th`' komutunu çalıştırmak ve parola sormasını istemek için;

```
[kazm@centosAnsible ~]$ ansible ansible_clients -m shell -a 'df -Th' --become-user jale -K
BECOME password:
192.168.0.100 | CHANGED | rc=0 >>
Filesystem                         Type   Size  Used Avail Use% Mounted on
tmpfs                              tmpfs  794M  1.1M  793M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv ext4    30G  8.3G   21G  30% /
tmpfs                              tmpfs  3.9G     0  3.9G   0% /dev/shm
tmpfs                              tmpfs  5.0M     0  5.0M   0% /run/lock
/dev/sda2                          ext4   2.0G  254M  1.6G  14% /boot
/dev/sda1                          vfat   1.1G  6.1M  1.1G   1% /boot/efi
tmpfs                              tmpfs  794M  4.0K  794M   1% /run/user/1000
[kazm@centosAnsible ~]$
```

Sistemleri Hakkında Bilgi Toplama ve Dosya İşlemleri

Facts, bir sistem hakkında ayrıntılı bilgileri ifade eder. Bunlardan birkaçı; IP adresi, sistem mimarisi, bellek ve CPU bilgileridir.

`ansible all -m setup`	Yönetilen hostların tümü hakkında bilgi toplar.
`ansible all -m copy -a "src=/var/log/secure dest=/tmp/"`	Sunucudaki /var/log/secure dosyasını, tüm hostların /tmp dizinine kopyalar.
`ansible ansible_clients -m file -a "dest=/tmp/secure mode=600"`	İzinleri ve dosya sahipliğini değiştirmek için file modülü kullanılır.
`ansible ansible_clients -m file -a "dest=/tmp/secure mode=600 owner=tecmint group=tecmint"`	sahip ve grup değişikliği de yapılabilir.
`ansible ansible_clients -m file -a "dest=/home/tecmint/data mode=755 owner=tecmint group=tecmint state=directory"`	mkdir -p'ye benzer şekilde dizinler oluşturabilir.

`ansible all -m shell -a "shutdown -h 0" -b -K`	Yönetilen hostların tümüne kapat komutu gönderir.

Ansible Playbook nedir?

Bir Ansible Playbook, çok az veya hiç insan müdahalesi olmadan gerçekleştirilen otomasyon görevlerinin ve karmaşık BT etkinliklerinin bir taslağıdır. Playbook'lar, bir Ansible envanterini oluşturan hostların bir grubu, grubu veya kategorizasyonu üzerinde çalıştırılır.

Ansible, otomatikleştirmek istediğiniz şeye bağlanarak ve manuel talimatları gerçekleştiren programları zorlayarak çalışır. Bu uygulamalar, uç noktanın bağlantısına, arayüzüne ve talimatlarına bağlı olarak geliştirilen Ansible modüllerinden yararlanır.

Ansible daha sonra bu modülleri çalıştırır (varsayılan olarak SSH aracılığıyla) ve tamamlandıktan sonra (varsa) kaldırır.

Ansible, günlük BT faaliyetlerini otomatikleştirmenin en basit yöntemidir. Yöneticiler, geliştiriciler ve BT yöneticileri için inanılmaz derecede kısa bir öğrenme eğrisi ile basit, tutarlı, güvenli ve güvenilir olmayı amaçlar.

Ansible'da playbook, Ansible'ın önemli bir özelliği ve her Ansible kurulumunun temelidir.

Ansible playbook, kullanıcıların bir sunucu kurulumunun çalışmasını tanımlayan yapılandırılmış bir komut dosyası koleksiyonu olan Ansible kodunu oluşturduğu bir dosyadır. Genel BT sürecindeki bir dizi aşamayı veya uzak sistemlerde izlemesi gereken ilkeleri tanımlar.

Playbook'lar, belirli bir sırayla gerçekleştirilen bir veya daha fazla play koleksiyonlarıdır. Play, envanterdeki hostlara karşı gerçekleştirilen sıralı bir görevler dizisidir. Yapılması gereken görev play'lerle tanımlanır. Her play'de, yapılandırılacak hostların ve tamamlanması gereken sorumlulukların bir listesi vardır. Standartlaştırılmış play'ler yoktur; her play bir yönetici tarafından yazılmalıdır.

Playbook'larda insan tarafından okunabilen bir veri işleme dili olan YAML kullanılmaktadır. “YAML”, “YAML Ain't Markup Language” anlamına gelen bir dizi kısaltmasıdır.

Ansible Playbook'ları Nasıl Çalışır?

Ansible modülleri görevleri yerine getirir. Bir veya daha fazla Ansible işi birleştirilerek bir oyun oluşturulabilir. Ansible Playbook, iki veya daha fazla oyundan oluşur. Ansible Playbook'lar, hostlarda otomatik olarak çalışan görev koleksiyonlarıdır. Ansible envanteri host gruplarından oluşur.

Ansible Playbook'taki her modül belirli bir görevden sorumludur. Her modül, bir işin ne zaman ve nerede gerçekleştirildiğini ve hangi kullanıcının yaptığını gösteren meta verilere sahiptir. Aşağıdakiler de dahil olmak üzere çeşitli BT görevlerini yürüten çok sayıda Ansible modülü vardır:

- **Cloud management:** oci_vcn, Oracle Bulut Altyapısı ortamlarında sanal bulut ağları oluşturur, siler veya günceller.
- **User management:** selogin, Linux işletim sistemi (OS) kullanıcılarını SELinux kullanıcısıyla eşler ve gitlab_user, GitLab kullanıcılarını oluşturur, günceller veya siler.
- **Networking:** Düzinelerce modül, uygulama programlama arabirimlerini (API'ler) yönetir; Cisco IOS, NXOS ve IOS XR cihazları; yanı sıra F5 BIG-IP hizmetleri.
- **Configuration management:** pip, Python kütüphane bağımlılıklarını yönetirken, assemble, konfigürasyon dosyalarını parçalardan birleştirir.
- **Security:** Openssh_cert, bir OpenSSH hostu veya kullanıcı sertifikaları oluşturur ve ipa_config, global FreeIPA yapılandırma ayarlarını yönetir.

Ansible Playbook Nasıl Oluşturulur?

Ansible playbook'ları oldukça büyük ve sofistike olma yeteneğine sahiptir. Ancak ister kısa ve güzel, ister büyük bir destan gibi olsun, her playbook aşağıdaki standart bölümlere ayrılmıştır:

- **Host:** Playbook'un yürütüleceği hostları belirtir. Bu veriler Ansible envanter dosyasından türetilmiştir. Sonuç olarak, host bölümü bir cihaz listesidir.
- **variable:** Bu bölüm isteğe bağlıdır ve betiğin gerektirdiği değişkenleri sağlar. Gerektiği kadar büyük veya küçük olabilir.
- **Tasks:** Bu bölüm, Modüllerin kullanımını açıklar ve hedef makinenin yapması gereken görevleri tanımlar. Her işe bir ad ve neleri başardığına dair kısa bir açıklama verilir ve playbook yürütüldüğünde listelenir.

Ansible Playbooks nasıl kullanılır?

Ansible, YAML sözdizimini kullanır. YAML dosya uzantılarından .yaml veya .yml yaygın olanlardır.

Ansible Playbook'lar iki şekilde kullanılabilir: komut satırı arabirimi (CLI) aracılığıyla veya Red Hat Ansible Automation Platform'un buton dağıtımları aracılığıyla.

- **CLI'den:** Ansible Playbook'ları başlatmak için, açık kaynak Ansible projesini veya Red Hat Ansible Automation Platform'u yükledikten sonra `ansible-playbook` komutu ile kullanılır.
- **Platform içinden:** Red Hat Ansible Automation Platform'un web tabanlı kullanıcı arabirimi, daha büyük görevlerin (veya iş şablonlarının) parçası olarak kullanılan düğmeli Ansible Playbook dağıtımları sunar. Bu, özellikle BT otomasyonunda yeni olan veya CLI ile çalışma konusunda çok fazla uzmanlığa sahip olmayan kullanıcılar için faydalı olan ekstra önlemler sağlar.

Ansible Playbook'ların Variable'leri Nelerdir?

Ansible, iki sistem tam olarak aynı olmadığından, kullanıcıların sistem farklılıklarıyla başa çıkmalarına yardımcı olmak için değişkenleri kullanır. Değişken adları harflerden, rakamlardan ve alt çizgilerden oluşur, ancak her zaman bir harfle başlamaları gerekir. Değişkenler ayrıca hiçbir zaman boşluk içermez.

Değişkenler, "`vars:`" komutu kullanılarak doğrudan playbook'larda tanımlanabilir. Değişkenler, özel isimlerden bağlantı noktalarına, web sunucularına ve hatta belirli bir komuta kadar her şey olabilir.

Değişkenler, değerleri depolamak için kullanılır. Gruplar ve hostlar, envanter dosyaları, dizi değişkenleri, sözlük değişkenleri ve özel değişkenler için değişkenler vardır.

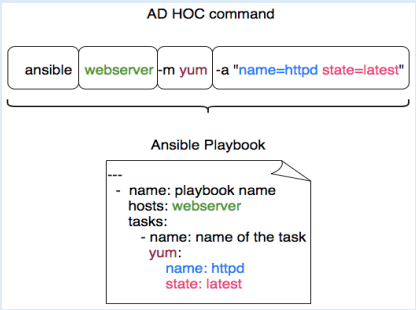
Özel değişkenler, kullanıcı tarafından değiştirilemeyen ve her zaman Ansible tarafından geçersiz kılınan yerleşik değişkenlerdir. Alternatif olarak, değişkenler ayrı bir dosyada tutulabilir ve gerektiğinde vars files komutuyla içe aktarılabilir.

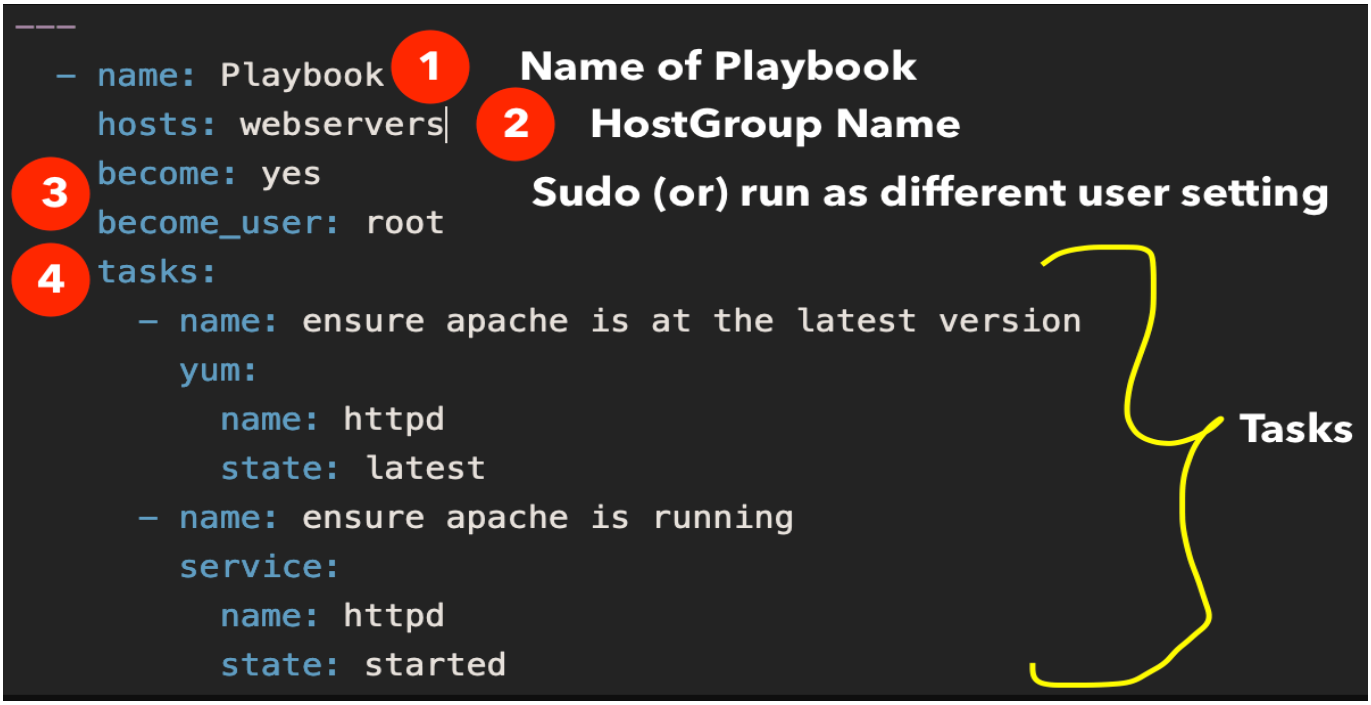
Gerekli değilse, playbook'un değişkenleri içermesi gerekmediğini unutmamak önemlidir. Tamamen isteğe bağlıdırlar.

Ansible Playbook'ları Örneği:

Ansible, bulut tabanlı REST API'leri, Linux ve Windows sistemleri, ağ cihazları ve çok daha fazlasını içeren çok çeşitli cihaz türleri ile iletişim kurabilir. İki farklı sunucu türünü otomatik olarak güncelleyen iki Ansible modülü örneği:

```
- name: Playbook
    hosts: webservers
    become: yes
    become_user: root
    tasks:
      - name: ensure an apache is at the latest version
        yum:
          name: httpd
          state: latest
      - name: ensure apache is running
        service:
          name: httpd
          state: started
```





Bu kısa ansible-playbook örneği, Apache kurulumunun tamamlanıp hazır hale gelmesi için yeterlidir. Her satırın açıklaması:

- **name** Playbook'un adı
- **hosts** Genellikle bir host grubu olarak gruplanan ve bir envanter dosyasında tanımlanan host koleksiyonu.
- **become** oturum açmış kullanıcıdan farklı bir kullanıcı 'olmaya' izin verir. **become:yes** yönergesi ayrıcalıkları yükseltir ve install, update, reboot gibi root ayrıcalıkları gerektiren görevlere izin verir. Playbook'un `sudo` olarak çalışması gerektiğini belirtir.
- **become_user** başka bir kullanıcı olmak için kullanılabilecek bir yönerge, geçiş yapmak istenilen kullanıcı adı belirtilir, sudo su – user'a benzer. Oturum açtıktan sonra uzaktaki bilgisayarda sudo kullanıcısına geçmeyi sağlar.
- **tasks** tamamlanacak bir faaliyetler koleksiyonudur, tüm görevler bunun altında belirtilir.

Daha sonra biri **yum** diğeri **service** olmak üzere iki modüllü iki jobumuz var.
İlk **yum** işindeki en son değişken, yukarıda belirtilen **httpd** paketinin henüz kurulu değilse kurulması (veya) kuruluysa mevcut en son sürüme güncellenmesi gerektiğini belirtir. Varsa güncellenmesi istemiyorsa, state öğesi **state:start** olarak değiştirilir .

Ansible'ın çoğu modülü idempotent (etkisiz, bağımsız)'tir, bu da değişikliklerin yalnızca gerektiğinde uygulandığı anlamına gelir. `copy` gibi idempotent bir modülle bir komut ikinci kez çalıştırıldığında, (dosya mevcut olduğundan) herhangi bir işlem yapmadan görevin başarılı olduğu belirtilir.

```
[kazm@centosAnsible ~]$ ansible all -i /etc/ansible/hosts -m ansible.builtin.copy -a "src=./ansible-test dest=/tmp/ansible-test"
192.168.0.100 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
```

Aynı komutu ikinci kez çalıştırdıktan sonra, komut çıktısının renkleri de farklılaşır:

```
[kazm@centosAnsible ~]$ ansible all -i /etc/ansible/hosts -m ansible.builtin.copy -a "src=./ansible-test dest=/tmp/ansible-test"
192.168.0.100 | SUCCESS => {
    ...
    },
    "changed": false,
```

Plays ve Playbooks

Ansible, özel görevlerin yürütülmesi için playbook'larda kullanılan modüller adı verilen bağımsız betiklerle birlikte gelir.

Modüller, paket yönetimi, dosyaların arşivlenmesi ve kopyalanması gibi görevleri otomatikleştirmek için kullanışlıdır. Yapılandırma dosyalarında ince ayarlar yapmaya ve yönlendiriciler, anahtarlar, yük dengeleyiciler, güvenlik duvarları ve bir dizi başka cihaz gibi cihazları yönetmeye olanak tanırlar.

Paket Yönetimi

Sistem yöneticileri tarafından yürütülen en önemli ve sık görevlerdendir. Ansible, hem RedHat hem de Debian tabanlı sistemlerde paket yönetimi görevlerini yürütmeye yardımcı olan modüllere sahiptir.

Debian tabanlı APT paket yönetimi için `apt` modülü, YUM paket yönetimi için eski `yum` modülü ve daha yeni RHEL dağıtımlarıyla ilişkili `dnf` modülü bulunmaktadır.

Aşağıda, modüllerin bir çalışma kitabında nasıl kullanılabileceğine dair birkaç örnek verilmiştir:

Apache Web Sunucusunu RHEL 8'e Kurmak

```
---
- name: install Apache webserver
  hosts: webservers

  tasks:
    - name: install httpd
      dnf:
        name: httpd
        state: latest
```

Apache Web Sunucusunu Debian 10'a Kurmak

```
---
- name: install Apache webserver
  hosts: databases

  tasks:
    - name: install Apache webserver
      apt:
        name: apache2
        state: latest
```

Service Modülü

Hizmet modülü, hizmetlerin başlatılmasını, durdurulmasını, güncellenmesini, yükseltilmesini ve yeniden yüklenmesini sağlar.

Apache Web Sunucusunu Başlatma

```
---
- name: Start service httpd, if not started
  service:
    name: httpd
    state: started
```

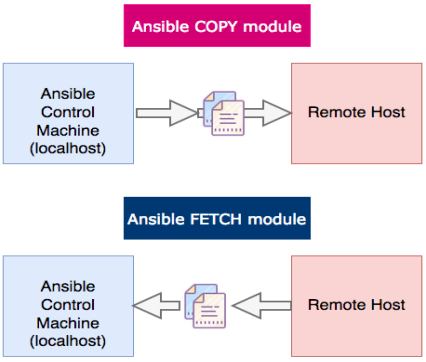
Apache Web Sunucusunu Durdurma

```
---
- name: Stop service httpd
  service:
    name: httpd
    state: stopped
```

Bir Ağ Arayüzünü Yeniden Başlatma

```
---
- name: Restart network service for interface eth0
  service:
    name: network
    state: restarted
    args: enp2s0
```

Kopyalama Modülü



Copy modülü, yerel veya uzak makinedeki dosya veya dizinde basit bir kopyalama yürütür.
Gereksinime göre uygun bir `copy` modeli kullanılmalıdır.
- Dosyaları yerel bir kaynaktan yerel bir hedefe kopyalama
- Dosyaları uzak bir kaynaktan uzak bir hedefe kopyalama (remote_src)
- Dosyaları yerel bir kaynaktan uzak bir hedefe kopyalama

Öte yandan dosyaları uzak kaynaktan yerele kopyalamak için `fetch` modülü kullanılabilir.
copy modülü: Dosyaları yerel olarak ansible kontrol makinesinden,
uzak makineye, host/host grubuna kopyalar.
fetch modülü: Dosyaları uzak hostdan yerel olarak ansible kontrol makinesine kopyalar.

```
---
# copy_file.yml

- name: copy files to destination
  hosts: localhost
  connection: local
  tasks:
    - name: src.txt, dest.txt olarak kopyalar
      copy:
        src: files/src.txt
        dest: files/dest.txt
      tags:
        - simple_copy
```

Bu playbook tek bir play'dan oluşuyor. Bu playbook'u yürütmek için varsayımlar:
1. Playbook ile aynı konumda bir files dizini var
2. files dizininde bir src.txt dosyası var.

Play, "src"yi "dest"ine kopyalamak için copy modülünü kullanan bir görevden oluşur.
Varsayılan olarak, ansible copy modülü hedefe bir kopyalama yapar.
Dosya varsa kopyalama yapılmaz.

```
[kazm@centosAnsible test-an]$ ansible-playbook -i /etc/ansible/hosts copy_file.yml
PLAY [copy files to destination]
*********************************************************************************************************************************************************
TASK [Gathering Facts]
*********************************************************************************************************************************************************
ok: [localhost]
TASK [copy src.txt as dest.txt in the same dir]
*********************************************************************************************************************************************************
changed: [localhost]
PLAY RECAP
*********************************************************************************************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[kazm@centosAnsible ~]$
```

```
---
# copy_file.yml
- name: copy files to destination
  hosts: localhost
  connection: local
  tasks:
    - name: copy src.txt to a non existing
directory
      copy:
        src: files/src.txt
        dest: files/not_dir/
      tags:
        - dir_not_exist
```

Dosyayı var olmayan bir dizine kopyalama
Hedef dizin mevcut değilse, copy modülü onu oluşturma ve dosyayı kaynak dosya adıyla aynı ada sahip yeni dizine kopyalama işini üstlenir.

Yalnızca dizinin içeriğinin, dış dizini dışarıda bırakarak kopyalanması gerekiyorsa, src, src dizininin içeriğini temsil eden bir eğik çizgi / ile bitmelidir.

Uzak Hostlar Arasında Dosya Kopyalama

Dosyaları bir hostdan başka bir hosta kopyalamak için, ilk akla gelen SCP'dir. Bir Ansible Control Master'dan uzak hostlara dosya kopyalamak için, COPY (scp) modülü gayet iyidir. Ancak dosyaları uzak hostlar arasında kopyalamak veya uzaktan uzağa dosyaları kopyalamak için daha fazlasına ihtiyaç var. Ansible'daki hostlar arasında dosya kopyalama için copy , fetch , sync gibi bazı modüller kullanılır. Copy modülü daha önce incelendi.

Ansible Fetch, dosyaları uzak sunucudan kontrol makinesine çeker. Ansible Synchronize ise, dosyaları uzak sunucular (veya) hedef hostlar arasında kopyalar. Bu daha çok Ansible yardımıyla RSYNC yapmak gibidir.

Metod1: Fetch modülünü kullanarak
uzak hostlar arasında dosya kopyalama

Metod2: Synchronize modülünü kullanarak
uzak hostlar arasında dosya kopyalama
Synchronize (Push) İtme
Synchronize (Pull) Çekme

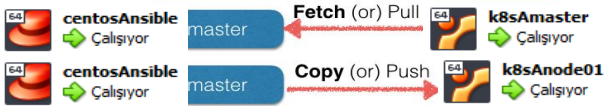
Metod 1: Fetch modülünü kullanarak k8sAmaster'dan k8sAnode01'e kopyalama

Bu yöntemde uzak düğümler arasında SSH Kimlik Doğrulaması ***gerekmez.***

Uzak düğümler arasında SSH Anahtarı tabanlı Kimlik Doğrulaması etkin olmadığında, uzak düğümler arasında dosya aktarımı için en iyi ve en kolay seçenektir. Yine de, en kısa yol (veya) kestirme yol değildir. Yaklaşım ne olursa olsun, birincil odak noktası işi bitirmek olduğu durumlarda işe yarar. Bu yöntem iki aşamalı bir süreçtir.

Adım 1: Dosya uzak sunucudan (kaynak) ansible master'a (Fetch) getirilir.

Adım 2: Dosya Ansible master'dan uzak sunucuya (hedef) (Copy) itilir.



Burada centosAnsible, dosyaların geçici olarak saklandığı ve ardından aktarıldığı bir arabellek görevi görür.
Yukarıda belirtilen görevlerin ikisini de gerçekleştiren playbook `copy_file.yml`

```
---
 - hosts: k8sA
   tasks:
     - name: Dosya k8sAmaster'dan centosAnsible (ansible master)'a (fetch) getirilir
       run_once: yes
       fetch: src=/tmp/k8sAmasterTOk8sAnode01.pdf dest=buffer/ flat=yes
       when: inventory_hostname == 'k8sAmaster'

     - name: Dosya centosAnsible (ansible master)'dan k8sAnode01'e (copy) kopyalanır
       copy: src=buffer/k8sAmasterTOk8sAnode01.pdf dest=/tmp/
       when: inventory_hostname == 'k8sAnode01'
```

```
[kazm@centosAnsible ~]$ cat/etc/ansible/hosts
[ansible_servers]
centosAnsible

[k8s_servers]
k8sAmaster

[k8s_nodes]
k8sAnode01
k8sAnode02
k8sAnode03
k8sAnode04

[k8sA:children]
k8s_servers
k8s_nodes
```

```
[kazm@centosAnsible Lab]$ ansible-playbook -i /etc/ansible/hosts copy_file.yml

PLAY [k8sA] ********************************************************************************************************************************************************

TASK [Gathering Facts]
*********************************************************************************************************************************************************************
ok: [k8sAmaster]
ok: [k8sAnode01]
ok: [k8sAnode02]
ok: [k8sAnode04]
ok: [k8sAnode03]

TASK [Dosya k8sAmaster'dan centosAnsible (ansible master)'a (fetch) getirilir] ************************************************************************************
changed: [k8sAmaster]

TASK [Dosya centosAnsible (ansible master)'dan k8sAnode01'e (copy) kopyalanır] ************************************************************************************
skipping: [k8sAmaster]
skipping: [k8sAnode02]
skipping: [k8sAnode03]
skipping: [k8sAnode04]
changed: [k8sAnode01]

PLAY RECAP *********************************************************************************************************************************************************
k8sAmaster                 : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
k8sAnode01                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode02                 : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
k8sAnode03                 : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
k8sAnode04                 : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```

Bu `copy_file.yml` dosyasındaki son satır;

`when: inventory_hostname == 'k8sAnode01'"`

değiştirilerek

`when: inventory_hostname is match('k8sAnode0.*')`

playbook tekrar çalıştırılırsa; adında `k8sAnode0` tüm node'lara dosya kopyalanır.

```
[kazm@centosAnsible Lab]$ ansible-playbook -i /etc/ansible/hosts copy_file.yml
      ...
PLAY RECAP *********************************************************************************************************************************************************
k8sAmaster                 : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
k8sAnode01                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode02                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode03                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode04                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```

Koşula dayalı yürütmeyi gerçekleştirmek için "`when`" kullanılır ve `inventory_hostname` yürütme anındaki uzak host adını işaret edecek ansible yerleşik değişkendir.

Ansible'nın `inventory_hostname` ve `ansible_hostname` yerleşik değişkenlerinin her ikisi de makinenin adını verir. `inventory_hostname` makinenin adını envanter komut dosyasından veya uygun yapılandırma dosyasından alır. Değer, tanımladığımız yapılandırma dosyasından alındığından, gerçek çalışma zamanı `hostname` değeri, bu değişken tarafından döndürülen değerden farklı olabilir.

Çoğunlukla, yerel geliştirme ortamlarında, sanal makineler için tam nitelikli bir hostname veya DNS adlarına sahip olunmaz. Bu nedenle, host grubu altındaki hostlara başvurmak için genellikle kontrol makinesindeki `/etc/hosts` dosyasına girişler yapılır.

`inventory_hostname`'in sorumluluğu, playbook'u başlatırken `-i` ile başvurulan varsayılan `ansible_hosts` dosyasında veya herhangi bir `hostfile`'da belirtilen bu host adlarına/IP'lere atıfta bulunmaktır. Bu sadece kontrol makinesinde belirtilen yerel bir kimlik olabilir. Hem **playbook** hem de **ad_hoc** komutunda kullanılabilir.

`ansible_hostname` yerleşik değişkeni de, tıpkı `inventory_hostname` gibi uzak hostun adını tutar, fark, `ansible_hostname` uzak makinenin adını **playbook**'un `gather_facts` bölümü sırasında toplanan bilgilerden almasıdır. `ansible_hostname` uzak makinenin `uname -n` veya `hostname` komut çıktısını alır. Bu, bağlanmak için kullandığınızdan veya ansible hosts dosyasında tanımlanandan farklı olabilir. `ansible_hostname gather_facts` adımına dayalı olduğundan, **ad_hoc** komutunda mevcut değildir.

Metod 2: Synchronize modülünü kullanarak `k8sAnode01`'den `k8sAnode02`'ye kopyalama
Uzak düğümler arasında SSH Anahtarı tabanlı kimlik doğrulama ***etkinleştirilmelidir.***

Senkronize modülünün sorunsuz çalışması için, uzak düğümler arasında SSH Anahtarı tabanlı kimlik doğrulamanın etkinleştirilmesi gerekir. Aksi takdirde senkronizasyon görevi ve Ansible da takılıp kalır.

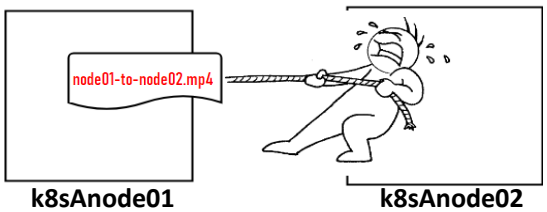
Başka bir deyişle, uzak düğümler, şifreyi manuel olarak girmek zorunda kalmadan birbirleriyle oturum açabilmelidir. Bu genellikle SSH anahtarı yardımıyla yapılır.

Daha önce de belirtidiği gibi Ansible `Synchronize`, Ansible `RSYNC`'ye daha çok benzer. `rsync` Linux'un tipik bir özelliğidir.

SSH Anahtarı tabanlı kimlik doğrulaması hazır olduğunda; (playbook) oyunu oynanabilir.

Synchronize modülünü kullanarak dosyaları kopyalamanın iki yöntemi vardır.
Synchronize Pull (Senkronize Çekme)
Synchronize Push (Senkronize İtme)
Hem **Synchronize Pull** hem de **Synchronize Push** yöntemini kullanarak dosyayı `k8sAnode01`'den `k8sAnode02`'ye kopyalayan playbooklar aşağıdadır.

Synchronize Pull (Çekme)
Hedeflenen: Kaynak Sunucu (`k8sAnode01`)
Yürütülen: Hedef Sunucu (`k8sAnode02`)
Açıklama: Dosyanın varması gereken hedef sunucuda yürütüldüğü noktasından bakıldığı için; görev, dosyayı kaynak sunucudan çekmektir.



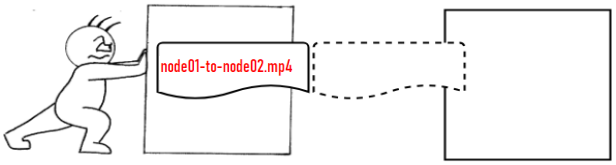
Delegasyon ile `k8sAnode02` hedef sunucuda yürütülen ve dosyayı k8sAnode01'den k8sAnode02'ye çeken (`pull`) playbook:

```
---
- name: Sync Pull task - Hedef hostda yürütülür "{{groups['k8s_nodes'][1]}}"
  hosts: "{{groups['k8s_nodes'][0]}}"
  user: kazm
  tasks:
    - name: Method Pull ile dosyayı k8sAnode01'den k8sAnode02'ye kopyalama
      tags: sync-pull
      synchronize:
        src: "{{ item }}"
        dest: "{{ item }}"
        mode: pull
      delegate_to: "{{groups['k8s_nodes'][1]}}"
      register: syncfile
      run_once: true
      with_items:
       - "/tmp/node01-to-node02.mp4"
```

```
[kazm@centosAnsible Lab]$ ansible-playbook -i /etc/ansible/hosts metod2.yml

PLAY [Sync Pull task - Hedef hostda yürütülür "k8sAnode02"] ********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [k8sAnode01]

TASK [Method Pull ile dosyayı k8sAnode01'den k8sAnode02'ye kopyalama] ***********************************************************************************************
changed: [k8sAnode01 -> k8sAnode02] => (item=/tmp/node01-to-node02.mp4)

PLAY RECAP *********************************************************************************************************************************************************
k8sAnode01                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```

hosts direktifi ilk sunucuyu hedeflediğinden, playbook ilk sunucuda yürütülecek şekilde ayarlanır.
`hosts: "{{groups['k8s_nodes'][0]}}"` k8sAnode01
Ancak, bu görev aslında ikinci sunucu `k8sAnode02` üzerinde `delege_to` yardımıyla yürütülür.

Synchronize Push (İtme)
Hedeflenen: Hedef Sunucu (`k8sAnode02`)
Yürütülen: Kaynak Sunucu (`k8sAnode01`)
Açıklama: Dosyanın zaten mevcut olduğu kaynak sunucuda yürütüldüğü için; görev, dosyayı hedef sunucuya itmektir.



`k8sAnode01` kaynak sunucuda yürütülen ve dosyayı 01'den 02'ye iten (`pushes`) playbook:

```
- name: Sync Push task - Kaynak hostda yürütülür "{{groups['k8s_nodes'][0]}}"
  hosts: "{{groups['k8s_nodes'][1]}}"
  user: kazm
  tasks:
    - name: Method Push'u kullanarak dosyayı k8sAnode01'den k8sAnode02'ye kopyalama
      tags: sync-push
      synchronize:
        src: "{{ item }}"
        dest: "{{ item }}"
        mode: push
      delegate_to: "{{groups['k8s_nodes'][0]}}"
      register: syncfile
      with_items:
       - "/tmp/node01-to-node02.mp4"
```

`hosts` direktifi ilk sunucuyu hedeflediğinden, playbook ikinci sunucuda çalışacak şekilde ayarlanır.
`hosts: "{{group[k8s_nodes][1]}}"` k8sAnode02
Ancak, bu görev aslında ilk sunucu `k8sAnode01` üzerinde `delege_to` yardımıyla yürütülür.

Playbook, host-based iki görevden oluşur. İlki Synchronize Pull ile `k8sAnode02`'de ve ikincisi Synchronize Push ile `k8sAnode01`'de çalışır.

Playbook anlaşılır olsa da, `"{{groups['k8s_nodes'][0]}}"` gibi satırların biraz daha fazla açıklanması gerekir.
Tıpkı bir dizi gibi **"k8s_nodes"** adlı host grubunun ilk öğesini (veya) sunucusunu temsil eder.

```
[kazm@centosAnsible Lab]$ ansible k8s_nodes --list-hosts
  hosts (4):
    k8sAnode01
    k8sAnode02
    k8sAnode03
    k8sAnode04
[kazm@centosAnsible Lab]$ cat /etc/ansible/hosts | grep -i "\[k8s_nodes\]" -A 4
[k8s_nodes]     Host Grup Adı
k8sAnode01      Host Grubundaki İlk Sunucu
k8sAnode02
k8sAnode03
k8sAnode04
[kazm@centosAnsible Lab]$
```

Playbook yürütüldükten sonraki çıktı;

```
[kazm@centosAnsible Lab]$ ansible k8s_nodes -m shell -a "ls -lrt /tmp/node01-to-node02.mp4" --user=kazm --limit=k8sAnode02
k8sAnode02 | FAILED | rc=2 >>
ls: cannot access '/tmp/node01-to-node02.mp4': No such file or directorynon-zero return code
[kazm@centosAnsible Lab]$ ansible-playbook -i /etc/ansible/hosts push_metod2.yml

PLAY [Sync Push task - Kaynak hostda yürütülür "k8sAnode01"] *******************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [k8sAnode02]

TASK [Method Push'u kullanarak dosyayı k8sAnode01'den k8sAnode02'ye kopyalama] *************************************************************************
changed: [k8sAnode02 -> k8sAnode01] => (item=/tmp/node01-to-node02.mp4)

PLAY RECAP *********************************************************************************************************************************************
k8sAnode02                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$ ansible k8s_nodes -m shell -a "ls -lrt /tmp/node01-to-node02.mp4" --user=kazm --limit=k8sAnode02
k8sAnode02 | CHANGED | rc=0 >>
-rw-rw-r-- 1 kazm kazm 85078842 May 27 11:11 /tmp/node01-to-node02.mp4
```

İlk ve son komut, `k8sAnode02`'deki `/tmp` dizini altındaki dosyayı kontrol eden bir Ansible **ad-hoc** komutudur.
Kullanılan host grubunda dört sunucu var ancak komutu yalnızca bir sunucuda yürütmek için `--limit` kullanıldı.

Hem **Fetch** hem de **Synchronize** modülleri kullanılarak bir dosya **k8sAnode01**'den **k8sAnode02**'ye başarıyla kopyalandı.

Ansible ile Yerel(Local)'den Uzak(Remote)'a Dosya Kopyalama

Ansible kontrol makinesindeki `index.html` dosyasını uzak sunucu grubuna kopyalayan playbook;

```
- name: Yerelden Uzağa Dosya Kopyalama
  hosts: k8sA
  tasks:
    - name: playbook ile dosya kopyalama
      copy:
        src: ~/Downloads/index.html
        dest: /tmp
        mode: 0644
```

yml dosyasındaki talimatların ayrıntıları ve açıklaması;

- **hosts:** hosts dosyasında tanımlanmış bir hedef host grubu
- **tasks:** tüm görevler (oyunlar) bunun altında tanımlanır
- **become:** bu, ansible'a `become_user` ile başka bir kullanıcı belirtilmediği sürece ilgili görevi bir **sudo user** `root` olarak yürütmesini söyler.
- **copy:** bu görevde kullanılacak modülün adı
- **src:** Playbookun veya ad-hoc komutunun yürütüldüğü yerel makinedeki dizinlerin ve belgelerin orijinal kaynak dosya yoludur. Bağlama bağlı olarak, bu yol göreceli veya mutlak olabilir. Kaynak konum dizin ise, belgeler ve alt dizinler yinelemeli olarak kopyalanır. Kaynak adres bir sözlük ise ve / ile bitiyorsa, dizinin bilgilerini çoğaltır. Alternatif olarak, verilen adı taşıyan dizini hedeflenen yola çoğaltır. Uzak sunucuda dosyayı aramak için `remote_src`'da ayarlayabilir.
- **dest:** dosyanın kopyalanması gereken uzak sunucu/hostdaki hedef yolu
- **mode:** kopyalandıktan sonra dosyanın iznini ayarlar. `0644` dosyanın iznini `rw- r-- r--` olarak ayarlanacaktır.

copy modülünün **owner:** ve **group:** seçenekleri; kopyalandıktan sonra hedef sunucudaki dosyanın sahibini ve grubunu ayarlar.

Bu playbook, ad hoc komut olarak da tek bir satırda yürütülebilir.

```
$ ansible k8sA -m copy -a "src=~/Downloads/index.html dest=/tmp mode=0644" # owner=apache group=apache
```

Playbook yürütüldüğünde çıktı aşağıdaki gibi olacaktır;

```
[kazm@centosAnsible Lab]$ ansible k8sA -m shell -a "ls -lrt /tmp/index.html"
k8sAnode01 | FAILED | rc=2 >>
ls: cannot access '/tmp/index.html': No such file or directorynon-zero return code
k8sAnode04 | FAILED | rc=2 >>
ls: cannot access '/tmp/index.html': No such file or directorynon-zero return code
k8sAnode03 | FAILED | rc=2 >>
ls: cannot access '/tmp/index.html': No such file or directorynon-zero return code
k8sAmaster | FAILED | rc=2 >>
ls: cannot access '/tmp/index.html': No such file or directorynon-zero return code
k8sAnode02 | FAILED | rc=2 >>
ls: cannot access '/tmp/index.html': No such file or directorynon-zero return code
[kazm@centosAnsible Lab]$ ansible-playbook -i /etc/ansible/hosts copy_file-local_to_remote.yml

PLAY [Yerelden Uzağa Dosya Kopyalama] ******************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [k8sAnode02]
ok: [k8sAnode03]
ok: [k8sAnode01]
ok: [k8sAmaster]
ok: [k8sAnode04]

TASK [playbook ile dosya kopyalama] ********************************************************************************************************************************
changed: [k8sAnode03]
changed: [k8sAnode01]
changed: [k8sAmaster]
changed: [k8sAnode02]
changed: [k8sAnode04]

PLAY RECAP *********************************************************************************************************************************************************
k8sAmaster                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode01                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode02                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode03                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode04                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$ ansible k8sA -m shell -a "ls -lrt /tmp/index.html"
k8sAmaster | CHANGED | rc=0 >>
-rw-r--r-- 1 kazm kazm 286 May 29 10:48 /tmp/index.html
k8sAnode02 | CHANGED | rc=0 >>
-rw-r--r-- 1 kazm kazm 286 May 29 10:48 /tmp/index.html
k8sAnode03 | CHANGED | rc=0 >>
-rw-r--r-- 1 kazm kazm 286 May 29 10:48 /tmp/index.html
k8sAnode04 | CHANGED | rc=0 >>
-rw-r--r-- 1 kazm kazm 286 May 29 10:48 /tmp/index.html
k8sAnode01 | CHANGED | rc=0 >>
-rw-r--r-- 1 kazm kazm 286 May 29 10:48 /tmp/index.html
[kazm@centosAnsible Lab]$
```

Ansible ile Yerel(Local)'den Uzak(Remote)'a Dizin Kopyalama

Ansible copy modülü ile dizin kopyalamanın iki varyasyon vardır.

- **Tip 1:** Kaynak Dizinin içeriği kopyalanır (ana dizini değil)
- **Tip 2:** Kaynak Dizin ve içeriği kopyalanır

Tip#1'de Ansible, yalnızca `src` dizininin içeriği hedef dizine kopyalanır.
Tip#2'de Ansible, dizini, dizinin içeriğiyle birlikte uzak sunucuya kopyalar. Yani uzak sunucuda dizini oluşturduktan sonra içine dosyaları ve varsa alt dizinleri kopyalar. `tree` komutu bu noktayı anlamayı kolaylaştırır.

Tip#1 Ansible ile dizin içeriği yinelemeli bir şekilde kopyalanır.
Yalnızca dizinin içeriği kopyalanır. İçerik bir dizin (alt dizin) veya bir dosya olabilir. Uygun bir AD HOC komutu;

```
$ ansible k8s_servers -m copy -a "src=~/Lab/ dest=/dizin"
```

Aynı kopyalama işini yapan playbook dosyası (`copy_dir1.yml` olarak adlandırıldı) içeriği;

```
- name: Ansible Dizin Copy Örneği - Local'dan Remote'a
  hosts: k8s_servers
  tasks:
    - name: Dizin içeriğini kopyalama (alt dizinler/dosyalar)
      copy:
        src: ~/Lab/
        dest: /dizin
```

Adhoc ve playbook yürütülmeden önce hedef sunucu olarak seçilen `k8s_servers` grubunun tek üyesi olan `k8sAmaster` sunucusundaki hedef dizin içeriğinin boş olduğu `tree` komutu ile de doğrulanacaktır.

Ayrıca hem adhoc komutu çalıştırıldıktan sonra hem de playbook yürütüldükten sonra yine aynı şekilde `tree` komutu ile dizin içeği gözlemlenecektir.

```
[kazm@k8sAmaster:/dizin$ tree /dizin
/dizin

0 directories, 0 files
[kazm@k8sAmaster:/dizin$
```

```
[kazm@centosAnsible Lab]$ ansible k8s_servers -m copy -a "src=~/Lab/ dest=/dizin"
k8sAmaster | CHANGED => {
    "changed": true,
    "dest": "/dizin/",
    "src": "/home/kazm/Lab"
}
[kazm@centosAnsible Lab]$  ansible-playbook -i /etc/ansible/hosts copy_dir1.yml

PLAY [Ansible Dizin Copy Örneği - Local'dan Remote'a] **************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [k8sAmaster]

TASK [Dizin içeriğini kopyalama (alt dizinler/dosyalar)] ***********************************************************************************************************
changed: [k8sAmaster]

PLAY RECAP *********************************************************************************************************************************************************
k8sAmaster                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```

Hem adhoc hem de playbook yürütüldükten sonra hedef sunucuda dizin içeği;

```
[kazm@k8sAmaster:/dizin$ tree /dizin
/dizin
├── buffer
│   └── node01-to-node02.mp4
├── copy_dir1.yml
└── copy_dir2.yml

1 directory, 3 files
[kazm@k8sAmaster:/dizin$
```

Tip#2 Kaynak Dizin ve içeriği kopyalanır.

Ansible; dizini ve dizinin içeriğini özyinelemeli bir şekilde uzak sunucuya kopyalar.

Yani uzak sunucuda dizini oluşturduktan sonra içine dosyaları ve varsa alt dizinleri kopyalar. Adhoc komutu;

```
$ ansible k8s_servers -m copy -a "src=~/Lab dest=/dizin"
```

Aynı kopyalama işini yapan playbook dosyası (`copy_dir2.yml` olarak adlandırıldı) içeriği;

```
- name: Ansible Dizin Copy Örneği - Local'dan Remote'a
  hosts: k8s_servers
  tasks:
    - name: Dizin içeriğini kopyalama (alt dizinler/dosyalar)
      copy:
        src: ~/Lab
        dest: /dizin
```

```
[kazm@centosAnsible Lab]$ ansible k8s_servers -m shell -a "tree /dizin" && ansible k8s_servers -m copy -a "src=~/Lab dest=/dizin" && ansible k8s_servers -m shell -a "tree /dizin"
k8sAmaster | CHANGED | rc=0 >>
/dizin

0 directories, 0 files
k8sAmaster | CHANGED => {
    "changed": true,
    "dest": "/dizin/",
    "src": "/home/kazm/Lab"
}
k8sAmaster | CHANGED | rc=0 >>
/dizin
└── Lab
    ├── buffer
    │   └── node01-to-node02.mp4
    ├── copy_dir1.yml
    └── copy_dir2.yml

2 directories, 3 files
[kazm@centosAnsible Lab]$  ansible-playbook -i /etc/ansible/hosts copy_dir2.yml

PLAY [Ansible Dizin Copy Örneği - Local'dan Remote'a] **************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [k8sAmaster]

TASK [Dizin ve dizin içeriğini kopyalama (ana dizin, alt dizinler/dosyalar)] ***************************************************************************************
changed: [k8sAmaster]

PLAY RECAP *********************************************************************************************************************************************************
k8sAmaster                 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```

Hedef sunucu dizin içeği;

Playbook (veya adhoc) yürütülmeden önce	Playbook (veya adhoc) yürütüldükten sonra

`[kazm@k8sAmaster:/dizin$ tree /dizin`
`/dizin`
 	
`2 directories, 3 files`
`[kazm@k8sAmaster:/dizin$`
	`[kazm@k8sAmaster:/dizin$ tree /dizin`
	`/dizin`
	`└── Lab`
	`├── buffer`
	`│ └── node01-to-node02.mp4`
	`├── copy_dir1.yml`
	`└── copy_dir2.yml`
	
	`2 directories, 3 files`
	`[kazm@k8sAmaster:/dizin$`

copy modülü parametreleri:

Parametre	Tanım
src	Yerel makinedeki kaynak dosya veya dizin yoludur.
dest	Uzak makinedeki hedef dosya veya dizin yoludur.
backup	Yes olarak ayarlanırsa, uzak makinede orijinal dosyanın bir yedek dosyasını oluşturur.
checksum	Yes olarak ayarlanırsa, dosyaların aynı olup olmadığını belirlemek için kaynak ve hedef dosyaları karşılaştırmak üzere bir checksum kullanır. Aynı değillerse, dosya kopyalanır.
force	Yes olarak ayarlanırsa, zaten var olsa bile hedef dosya veya dizinin üzerine yazacaktır.
mode	Hedef dosya veya dizindeki izinleri ayarlar.
owner	Hedef dosyanın veya dizinin sahibini ayarlar.
group	Hedef dosya veya dizinin grubunu ayarlar.

copy modülünde döngü kullanarak birden çok dosyayı uzak hedefe kopyalama

```
- name: Copy multiple files to remote server
  hosts: my_server
  tasks:
    - name: Copy files
      copy:
        src: "{{ item }}"
        dest: "/path/on/remote/machine/"
      with_items:
        - /path/to/local/file1
        - /path/to/local/file2
        - /path/to/local/file3
```

Mutlak yol, / kök dizininden başlayarak bir dosya veya dizine giden tam yolu belirtir. Mutlak yol her zaman eğik çizgi (/) ile başlar ve dosyaya veya dizine giden tam yolu belirtir.
Uzak bir sunucuya birden çok dosya kopyalamak için copy modülü içinde bir döngü kullanılabilir.

with_items parametresi, uzak makineye kopyalanacak dosyaların bir listesini belirtir.
src parametresi, with_items listesindeki her bir öğeye referans olan {{ item }} olarak ayarlanır. Bu, kopyalama modülünün dosya listesi üzerinde yineleme yapmasına ve bunları birer birer kopyalamasına olanak tanır.
dest parametresi, uzak makinedeki hedef dizini belirtir. Bu örnekte, dosyalar /path/on/remote/machine/ dizinine kopyalanır.
item başvurusu, her dosyanın adını hedef dizin yoluna eklemek için kullanılır.
Bu görev yürütüldüğünde, Ansible her dosyayı yerel makineden uzak makineye kopyalayacaktır. Ansible'ın daha yeni sürümlerinde (2.5 ve sonrası) with_items yerine loop anahtar sözcüğü kullanılabilir.

Birden çok dosyayı birden çok hedefe kopyalama

Birden çok dosyayı yerel bir makineden uzak bir makineye kopyalamak için copy modülünü kullanan bir playbook görevidir. Dosyalar, with_items parametresi kullanılarak bir sözlük listesinde belirtilir.

```
- name: Copy some files to /etc/myapp/
  copy:
    src: "{{ item.src }}"
    dest: "{{ item.dest }}"
    owner: root
    group: root
    mode: u=rw,g=rw,o=r
  with_items:
    - { src: app1.conf, dest: /etc/app1/ }
    - { src: app2.conf, dest: /etc/app2/ }
    - { src: app3.conf, dest: /etc/app3/ }
```

- Kaynak ve hedef yolları, src ve dest anahtar/değer çiftlerini içeren sözlükler listesinde dolaşan with_items yönergesi kullanılarak oluşturulur.
- Döngünün her tekrarı için src ve dest yolları, sırasıyla src ve dest yönergeleri kullanılarak kopya modülüne iletilir.

with_items: Kopyalanacak dosyaları tanımlayan sözlüklerin listesi. Her sözlük, yerel makinede kaynak dosya yolunu belirten bir src anahtarına ve uzak makinede hedef dosya yolunu belirten bir dest anahtarına sahiptir.

Uzak hostdaki konumlar arasında dosyala kopyalama

copy modülü, dosyaları aynı uzak makinede bir dizinden diğerine kopyalamayı sağlar. Ancak bu sadece dosyalar içindir, dizinler için değildir. Ansible'a niyetimizi bildirmek için remote_src parametresini kullanabiliriz.

```
- name: Uzak hostdaki konumlar arasında dosya copy
  hosts: k8sA
  tasks:
    - name: Uzak hostdaki konumlar arasında kopyalama
      copy:
        src: "{{ ansible_env.HOME }}/test_file"
        remote_src: true
        dest: "{{ ansible_env.HOME }}/test_file2"
```

Burada, copy modülü uzak hostdaki bir dosyayı bir konumdan diğerine kopyalar.
remote_src, kaynak dosya uzak hostda ise true olarak ayarlanır.
src ve dest parametrelerindeki değişken referanslarının çift kaşlı ayraçlar içine alınmalıdır.

`src`, kaynak dosya yolu `ansible_env.HOME` değişkeni kullanılarak `$HOME/test_file` olarak ayarlanır.
`dest`, hedef dosya yolu `ansible_env.HOME` değişkeni kullanılarak `$HOME/test_file2` olarak ayarlanır.

Playbook'un yürütülmesini kontrol etme: stratejiler ve daha fazlası

Varsayılan olarak Ansible, 5 çatal kullanarak herhangi bir hostda bir sonraki göreve başlamadan önce her görevi bir play'den etkilenen tüm hostlarda çalıştırır. Bu varsayılan davranış değiştirilmek istendiğinde, farklı bir strateji eklentisi kullanılabilir, çatal sayısı değiştirilebilir veya serial gibi birkaç anahtar kelimeden biri uygulanabilir.

Strateji seçme

Varsayılan davranış, doğrusal stratejidir. Ansible, hata ayıklama stratejisi ve her hostun oyunun sonuna kadar olabildiğince hızlı çalışmasına izin veren ücretsiz (free) strateji dahil olmak üzere başka stratejiler sunar:

```
- hosts: all
  strategy: free
  tasks:
  # …
```

Yan tarafta gösterildiği gibi her oyun için farklı bir strateji seçebilir veya tercih edilen strateji genel olarak `ansible.cfg`'de varsayılan dörtlük altında ayarlanabilir.

```
[defaults]
strategy = free
```

Tüm stratejiler, strateji eklentileri olarak uygulanır. Nasıl çalıştığına ilişkin ayrıntılar için her bir strateji eklentisinin belgeleri incelenebilir.

```
[defaults]
forks = 30
```

Çatal (fork) sayısını ayarlama

İşlemci gücü varsa ve daha fazla çatal kullanmak istenirse `ansible.cfg`'de ayarlanabilir veya Ansible'ın kaç tane çatal veya paralel işlem çalıştırabileceği `-f` bayrağı ile yapılandırılabilir:

```
$ ansible-playbook -f 30 my_playbook.yml
```

Ansible paralel doğası sayesinde, uzak hostlarda komutları birden çok çatal (fork) kullanarak çalıştırır veya görevi tüm sunucularda basitçe paralel olarak yürütmeyi sağlar, bu zamandan kazandırır.

Yürütmeyi kontrol etmek için anahtar kelimeler kullanma

Stratejilere ek olarak, birkaç anahtar kelime de oyunun yürütülmesini etkiler. `serial` ile aynı anda yönetilmek istenen hostların sayısı, yüzdesi veya sayı listesi ayarlanabilir. Ansible, bir sonraki host grubunu başlatmadan önce belirtilen sayıda veya yüzdede hostda oyunu tamamlar. `throttle` (kısma) ile bir bloğa veya göreve atanan işçi sayısı sınırlanabilir. Ansible'ın bir gruptaki bir sonraki hostu sırayla yürütmek için nasıl seçtiği kontrol edilebilir. `run_once` ile tek bir hostda görev çalıştırılabilir. Bu anahtar kelimeler strateji değildir. Bir oyuna, bloğa veya göreve uygulanan direktifler veya seçeneklerdir.

Oynatmanın yürütülmesini etkileyen diğer anahtar sözcükler arasında `ignore_errors`, `ignore_unreachable` ve `any_errors_fatal` bulunur.

```
---
# batch_serial.yml
- name: test play
  hosts: k8sA
  serial: 3
  gather_facts: False

  tasks:
    - name: first task
      command: hostname
    - name: second task
      command: hostname
```

Toplu iş (batch) boyutunun `serial` ile ayarlanması

Varsayılan olarak Ansible, her oyunun `hosts:` alanında ayarlanan kalıptaki tüm hostlara karşı paralel olarak çalışır. Bir seferde yalnızca birkaç makine yönetilmek istenirse, örneğin sıralı bir güncelleme sırasında, `serial` anahtar sözcüğünü kullanarak Ansible'ın tek seferde kaç hostu yönetmesi gerektiği tanımlanabilir.

Yandaki örnekte, "`k8sA`" grubunda 6 host olsaydı, Ansible sonraki 3 hosta geçmeden önce oyunu tamamen (her iki görevi) 3 hostda yürütürdü:

```
[kazm@centosAnsible Lab]$ ansible-playbook batch_serial.yml

PLAY [test play] ***************************************************************************************************************************************************

TASK [first task] **************************************************************************************************************************************************
changed: [k8sAmaster]
changed: [k8sAnode02]
changed: [k8sAnode01]

TASK [second task] *************************************************************************************************************************************************
changed: [k8sAmaster]
changed: [k8sAnode02]
changed: [k8sAnode01]

PLAY [test play] ***************************************************************************************************************************************************

TASK [first task] **************************************************************************************************************************************************
changed: [k8sAnode04]
changed: [k8sAnode03]

TASK [second task] *************************************************************************************************************************************************
changed: [k8sAnode03]
changed: [k8sAnode04]

PLAY RECAP *********************************************************************************************************************************************************
k8sAmaster                 : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode01                 : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode02                 : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode03                 : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
k8sAnode04                 : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```

Toplu iş boyutunun `serial` olarak ayarlanması, Ansible hatalarının kapsamını host listesinin tamamına değil, toplu iş boyutuna değiştirir. Bu davranışı değiştirmek için `ignore_unreachable` veya `max_fail_percentage` kullanılabilir.

```
---
- name: test play
  hosts: k8sA
  serial: "40%"
```

Ayrıca `serial` anahtar sözcüğü ile bir yüzde belirtilebilir. Ansible, geçiş başına sunucu sayısını belirlerken bir oyundaki toplam sunucu sayısına yüzdeyi uygular.

```
---
- name: test play
  hosts: k8sA
  serial:
    - 1
    - 5
    - 10
```

Host sayısı geçiş sayısına eşit olarak bölünmüyorsa, son geçiş kalanı içerir. `serial: "30%"` ise ve 20 host varsa, ilk üç grup 6 ve son grup 2 host içerir.

Parti boyutları yandaki gibi bir liste olarak da belirlenebilir. Yandaki örnekte, ilk toplu iş tek bir host, sonraki 5 host ve (geriye kalan host varsa), takip eden her toplu iş ya 10 host ya da 10'dan azsa kalan tüm hostları içerir.

```
---
- name: test play
  hosts: k8sA
  serial:
    - "10%"
    - "20%"
    - "100%"
```

Birden çok parti boyutu yüzde olarak listelenebilir. Ayrıca değerler karıştırabilir ve eşleştirilebilir.

Yüzde ne kadar küçük olursa olsun, geçiş başına host sayısı her zaman 1 veya daha fazla olacaktır.

```
---
- name: test play
  hosts: k8sA
  serial:
    - 1
    - 5
    - "20%"
```

`throttle` ile yürütmeyi kısıtlama

`throttle` (kısma) anahtar sözcüğü, belirli bir görev için çalışan sayısını sınırlar. Blok ve görev seviyesinde ayarlanabilir. CPU-yoğun olabilecek veya hız sınırlayıcı bir API ile etkileşime girebilecek görevleri kısıtlamak için `throttle` kullanılır.

```
tasks:
- command: /path/to/cpu_intensive_command
  throttle: 1
```

Paralel olarak yürütülecek çatal sayısı veya makine sayısı zaten kısıtlandıysa, `throttle` ile işçi sayısı azaltılabilir, ancak artırılamaz. Eğer ikisi birlikte kullanılıyorsa, etki etmesi için `throttle` ayarının `forks` veya `serial` ayarından daha düşük olması gerekir.

Envantere göre yürütmeyi sıralama

`order` anahtar sözcüğü, hostların çalıştırılma sırasını kontrol eder. `order` için olası değerler şunlardır:
- **inventory:** (varsayılan) İstenen seçim için envanter tarafından sağlanan sıralama (aşağıdaki nota bakın)
- **reverse_inventory:** Yukarıdakiyle aynı, ancak döndürülen liste tersine çevriliyor
- **sorted:** Ada göre alfabetik olarak sıralanmıştır
- **reverse_sorted:** İsme göre ters alfabetik sırada sıralanmıştır
- **shuffle:** Her çalıştırmada rastgele sıralanır

"envanter" sırası, envanter kaynak dosyasında hostların/grupların tanımlanma sırasına değil, "derlenmiş envanterden bir seçimin döndürülme sırasına" eşittir. Bu, geriye dönük uyumlu bir seçenektir ve tekrar üretilebilirken, normalde öngörülebilir değildir. Envanterin doğası gereği, barındırma kalıpları, limitler, envanter eklentileri ve birden fazla kaynağa izin verme yeteneği nedeniyle, böyle bir siparişi iade etmek neredeyse imkansızdır. Basit durumlarda bu, dosya tanımı sırası ile eşleşebilir, ancak bu garanti edilmez.

```
- command: /opt/application/upgrade_db.py
  run_once: true
```

`run_once` ile tek bir makinede çalışıyor olma

Bir görevin yalnızca ana makine grubundaki ilk hostda çalışması isteniyorsa, bu görevde `run_once` değeri `true` olarak ayarlanmalıdır.
Ansible, bu görevi geçerli batch'deki ilk hostda yürütür ve tüm sonuçları ve olguları aynı batch'teki tüm hostlara uygular. Bu yaklaşım, aşağıdaki gibi bir göreve koşullu uygulamaya benzer:

```
- command: /opt/application/upgrade_db.py
  when: inventory_hostname == k8sA[0]
```

Ansible, bu görevi geçerli batch'deki ilk hostda yürütür ve tüm sonuçları ve olguları aynı batch'teki tüm hostlara uygular. Bu yaklaşım, buradaki gibi bir göreve koşul uygulamaya benzer.

```
- command: /opt/application/upgrade_db.py
  run_once: true
  delegate_to: "{{groups['k8s_nodes'][2]}}"
```

Ancak `run_once` ile sonuçlar tüm hostlara uygulanır. Görevi batch'deki ilk host yerine belirli bir hostda çalıştırmak için görevi devredin:

Delegasyonda her zaman olduğu gibi, eylem, yetki verilen hostda yürütülür, ancak bilgiler yine de görevdeki orijinal hosta aittir.

`serial` ile birlikte kullanıldığında, `run_once` olarak işaretlenen görevler, her seri partide bir hostda çalıştırılacaktır. Görevin seri moddan bağımsız olarak yalnızca bir kez çalıştırılması gerekiyorsa,
`when: inventory_hostname == ansible_play_hosts_all[0]` yapısı kullanılmalıdır.

Herhangi bir koşullu (başka bir deyişle, when:), görevin çalışıp çalışmadığına karar vermek için 'ilk host' değişkenlerini kullanır, başka hiçbir host test edilmez.

Tüm hostlar için olguyu (fact) ayarlama varsayılan davranışından kaçınmak isteniyorsa, belirli görev veya blok için "`delegate_facts: True`" olarak ayarlanmalıdır.

Fact Değişkenini Kullanarak OS'ni Kontrol Eden Playbook

Playbook'un OS ile uyumlu olduğundan ve OS ile ilgili bir modül kullandığından emin olmak için bir playbook'da OS'yi `when` ile kontrol etmek gerekir. Bu, playbook çalıştırıldığında OS hakkında tüm bilgileri toplayan `facts` ile yapılır.

İşletim sisteminin Ubuntu mu yoksa Centos mu olduğunu kontrol eden ve platforma göre apache yükleyen örnek bir playbook. Sistem gerçeklerini toplamak için "`gather_facts: yes`" belirtiyoruz.

`gather-facts` ile istemci sistemden global değişkenlerden bilgiler alınır.
Örnek playbook dosyası `os-base-fact.yml` içeriği;

```
---
- name: İşletim sistemini kontrol
  hosts: all
  gather_facts: yes
  tasks:
    - name: OS bazlı paket kurma - Ubuntu
      apt:
        name: apache2
        state: present
      when: ansible_distribution == "Ubuntu"
    - name: OS bazlı paket kurma - CentOS httpd paket kurulumu
      dnf:
        name: httpd
        state: latest
      when: ansible_distribution == "CentOS"
    - name: OS bazlı paket kurma - CentOS httpd service kurulumu
      service:
        name: httpd
        enabled: true
        state: started
      when: ansible_distribution == "CentOS"
    - name: OS bazlı paket kurma - CentOS index.html yaratma
      copy:
        dest: /var/www/html/index.html
        content: |
          Enable SysAdmin Demo:
          Ansible Profiling with Callback Plugin
          Custom Web Page
      when: ansible_distribution == "CentOS"
```

```
[kazm@centosAnsible Lab]$ ansible-playbook os-base-fact.yml -b -K
BECOME password:

PLAY [İşletim sistemini kontrol] ***********************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [k8sAnode02]
ok: [k8sAnode01]
ok: [k8sAmaster]
ok: [k8sAnode03]
ok: [centosAnsible]
ok: [k8sAnode04]

TASK [OS bazlı paket kurma - Ubuntu] *******************************************************************************************************************************
skipping: [centosAnsible]
changed: [k8sAnode02]
changed: [k8sAnode04]
changed: [k8sAnode03]
changed: [k8sAnode01]
changed: [k8sAmaster]

TASK [OS bazlı paket kurma - CentOS httpd paket kurulumu] **********************************************************************************************************
skipping: [k8sAmaster]
skipping: [k8sAnode01]
skipping: [k8sAnode02]
skipping: [k8sAnode03]
skipping: [k8sAnode04]
changed: [centosAnsible]

TASK [OS bazlı paket kurma - CentOS httpd service kurulumu] ********************************************************************************************************
skipping: [k8sAmaster]
skipping: [k8sAnode01]
skipping: [k8sAnode02]
skipping: [k8sAnode03]
skipping: [k8sAnode04]
changed: [centosAnsible]

TASK [OS bazlı paket kurma - CentOS index.html yaratma] ************************************************************************************************************
skipping: [k8sAmaster]
skipping: [k8sAnode01]
skipping: [k8sAnode02]
skipping: [k8sAnode03]
skipping: [k8sAnode04]
ok: [centosAnsible]

PLAY RECAP *********************************************************************************************************************************************************
centosAnsible              : ok=4    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
k8sAmaster                 : ok=2    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
k8sAnode01                 : ok=2    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
k8sAnode02                 : ok=2    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
k8sAnode03                 : ok=2    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
k8sAnode04                 : ok=2    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0

[kazm@centosAnsible Lab]$
```