BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
SEMİNER DERSİ PROJESİ

ANSIBLE

RUMEYSA EMİNE ŞAHİN

2022-2023 BAHAR DÖNEMİ

ÖNSÖZ

Bu kılavuz Bursa Teknik Üniversitesi Bilgisayar Mühendisliği 3. Sınıf Bahar Dönemi Seminer dersi kapsamında, Ansible adlı otomasyon aracının tanıtılması ve kullanımı hakkında bilgi verilmesi amacıyla hazırlanmıştır. Ansible, günümüzün karmaşık IT ortamlarında sistem yönetimini ve dağıtımını otomatikleştirmek için kullanılan güçlü bir araçtır. Bu kılavuz, Ansible'ın genel işleyişini, kurulum ve yapılandırma adımlarını, temel bileşenlerini, ekosistemini ve entegrasyonlarını açıklarken, aynı zamanda Ansible ile ilgili bir uygulama örneği sunmaktadır.

RUMEYSA EMİNE ŞAHİN

BURSA 2023

İÇİNDEKİLER

ÖNSÖZ I
İÇİNDEKİLER II
ÖZET III
1. GENEL BİLGİLER 1
1.1 Ansible Nedir? 1
1.2 Otomasyonun Önemi 1
1.3 Ansible Tarihçesi 1
1.4 Ansible Temel Çalışma Prensibi 2
1.5 Ansible Kullanım Alanları 3
2. ANSIBLE KURULUMU VE YAPILANDIRMASI 4
2.1 Ansible'ın Gereksinimleri 4
2.2 Kurulum Adımları 5
3. ANSIBLE TEMEL ELEMANLARI 6
3.1 Kontrol Makinesi (Control Machine) 6
3.2 Hedef Sistemler (Managed Nodes) 6
3.3 Ansible Envanter Dosyası 7
3.3.1 Envanter Dosyası Özellikleri 7
3.3.2 Envanter Dosyası Örnek 7
3.4 Ansible Modüller 9
3.5 Ansible Playbook 10
3.5.1 Playbook Temel Yapı 10
3.5.2 Playbook Öğeleri 11
3.5.3 Playbook Örnekleri 12
3.6 Ansible Ad-Hoc Komutları 15
4. ANSIBLE EKOSİSTEMLERİ 16
4.1 Ansible Tower 16
4.2 Ansible Galaxy 17
5. ANSIBLE ENTEGRASYONLAR 18
5.1 Ansible Ve Git Entegrasyonu 18
5.2 Ansible Ve Docker Entegrasyonu 19
6. ANSIBLE UYGULAMA ÖRNEĞİ 20
7. SONUÇLAR 22
8. KAYNAKLAR 23

ÖZET

Bu kılavuz, Ansible otomasyon aracının temel işleyişini, kurulum ve yapılandırma adımlarını, ekosistemini ve entegrasyonlarını ele alıyor. Ansible, karmaşık IT ortamlarında sistem yönetimi ve dağıtımını otomatikleştirmek için kullanılan güçlü bir araçtır. Kılavuz, Ansible'ın temel bileşenlerini, playbook ve ad-hoc komutlarını, Ansible Tower ve Ansible Galaxy gibi ekosistemlerini tanıtıyor. Ayrıca, Ansible'ın Git ve Docker gibi popüler entegrasyonlarını da inceliyor. Kılavuzun sonunda, Ansible ile gerçekleştirilen bir uygulama örneği sunularak, Ansible'ın pratik kullanımı gösteriliyor.

1. GENEL BİLGİLER

1.1 Ansible Nedir?

Ansible, otomasyon ve yapılandırma yönetim aracıdır. Yazılım kurulumu, yapılandırması, güncellemesi ve yönetimi gibi birçok işlemi otomatikleştirir. Sistem yöneticilerine, yazılım geliştiricilerine ve diğer teknoloji profesyonellerine karmaşık IT altyapılarının yönetimini kolaylaştırmak için tasarlanmış açık kaynaklı bir yazılımdır. Ansible, basit, etkili ve aynı zamanda esnek bir şekilde çalışabilme imkanı sağlar. (1)

1.2 Otomasyonun Önemi

Otomasyon, tekrarlayan ve sürekli olarak gerçekleştirilen görevlerin veya işlemlerin insan müdahalesi olmaksızın bilgisayar sistemleri veya makineler aracılığıyla yapılmasıdır. Otomasyon, tekrar eden işleri otomatikleştirerek iş süreçlerini optimize etmeyi, hataları azaltmayı, verimliliği artırmayı ve insanların daha değerli görevlere odaklanmasını sağlamayı amaçlar. İşletmeler, otomasyon araçları kullanarak verimliliklerini artırabilir, rekabet avantajı elde edebilir ve kaynaklarını daha etkin bir şekilde kullanabilirler.

1.3 Ansible Tarihçesi

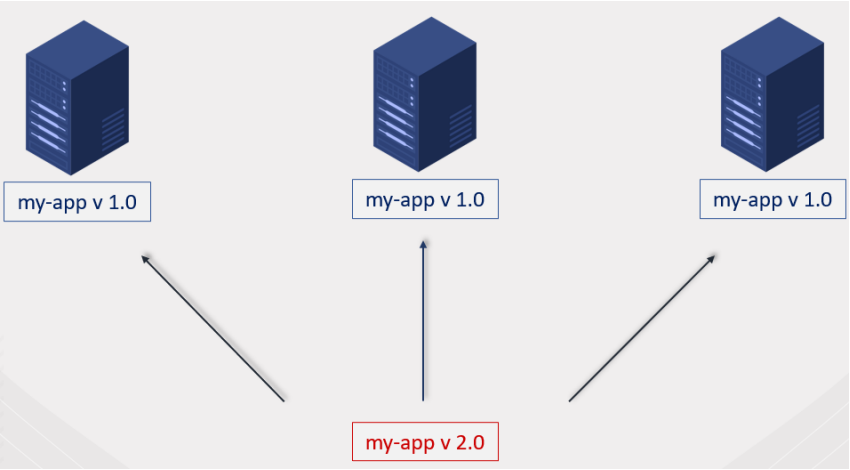
Ansible, 2012 yılında Michael DeHaan tarafından oluşturulan "ansibleworks" adlı bir projenin başlangıcıyla ortaya çıktı. İlk sürümü Ansible 0.1, Şubat 2012'de yayınlandı ve hızla büyüyen bir kullanıcı tabanıyla açık kaynak topluluğunda popülerlik kazandı. 2015 yılında Ansible Inc. adında bir şirket kuruldu ve bu şirket, Ansible'ın geliştirilmesi ve desteklenmesi üzerine odaklandı. Red Hat, 2015'in sonlarında Ansible Inc.'i satın aldı ve Ansible, Red Hat'in bir parçası haline geldi. Bu satın alma, Ansible'ın daha da yaygınlaşmasına ve gelişmesine katkıda bulundu. 2020 itibarıyla, Ansible, önde gelen bir otomasyon ve yapılandırma yönetimi aracı olarak kabul ediliyor ve birçok büyük organizasyon tarafından kullanılıyor.

Şekil 1: Red Hat Ansible, Otamasyon Platformu

1.4 Ansible Temel Çalışma Prensibi

Temel çalışma prensibi, bir kontrol makinesi üzerinde bulunan Ansible yazılımı aracılığıyla hedef makinelerde belirli görevleri gerçekleştirmektir. Kontrol makinesindeki Ansible, "inventory" (envanter) adı verilen bir listeyle hedef makineleri belirler ve bu makinelerde çalıştırılacak "playbooks" (oynatıcılar) adı verilen YAML dosyalarını kullanır. Playbook'lar, hedef makinelerde yapılacak işlemleri tanımlar. Ansible, SSH ile doğrudan erişim kullanarak belirtilen görevleri gerçekleştirir.

Diyelim ki bir şirkette birkaç sunucu ve bu sunucularda çalışan bir uygulama var. Sistem yöneticisinin uygulamanın yeni sürümü yayınlandığında yeni sürümü tüm sunuculara dağıtması gerekiyor. Sistem yöneticisinin bu uygulamanın yönetimini manuel olarak gerçekleştirdiğini düşünelim. Her bir sunucuda ayrı ayrı yapılandırma ve güncelleme işlemlerini yapmak zorunda kalır. Bu, zaman alıcı ve hatalara neden olabilecek bir işlemdir. Bu noktada Ansible devreye girer. Hedef cihazlar üzerinde çalışacak görevlerin belirlenmesi, yönetilmesi ve koordine edilmesi için kullanılır. Bu sayede, tüm sunucuların yönetimi için ayrı ayrı işlemler yapmak yerine, Ansible tek bir komutla tüm sunucuların yönetimini sağlar. (2)



Şekil 2: Ansible Temel Çalışma Prensibi

1.5 Ansible Kullanım Alanları

Ansible, açık kaynaklı bir otomasyon aracıdır ve geniş bir kullanım alanına sahiptir. Ansible'ın kullanım alanlarından bazıları aşağıda verilmiştir.

- Yapılandırma Yönetimi: Ansible, sunucuların ve ağ cihazlarının yapılandırma yönetimini kolaylaştırır. Yapılandırma dosyalarınızı ve yapılandırma değişikliklerinizi otomatik olarak dağıtabilir, hızlı ve tutarlı bir şekilde yapılandırma yapabilirsiniz.
- Uygulama Dağıtımı: Ansible, uygulama dağıtım süreçlerini otomatikleştirmek için kullanılabilir. Uygulamanızı birden fazla sunucuya veya buluta dağıtmak için Ansible oyunlarını kullanabilirsiniz. Bu şekilde, yeni bir sürüm yayınladığınızda veya yapılandırma değişiklikleri yaptığınızda hızlı ve tekrarlanabilir bir dağıtım süreci sağlayabilirsiniz.
- Veri Merkezi Otomasyonu: Ansible, büyük ölçekli veri merkezlerinin yönetimini otomatikleştirmek için kullanılabilir. Sunucu oluşturma, konfigürasyon yönetimi, performans izleme, yedekleme ve kurtarma gibi görevleri otomatikleştirebilirsiniz.
- Bulut Hizmetleri Yönetimi: Ansible, popüler bulut hizmet sağlayıcıları ile entegre olabilir ve bulut altyapısının yönetimini kolaylaştırabilir. Sunucuları, depolama kaynaklarını ve ağ yapılandırmalarını otomatik olarak oluşturabilir ve yapılandırabilirsiniz.
- Güvenlik Otomasyonu: Ansible, güvenlikle ilgili görevlerin otomatikleştirilmesi için kullanılabilir. Yama yönetimi, erişim kontrolü, sertifika yönetimi gibi güvenlik işlemlerini otomatikleştirebilir ve güvenlik politikalarını uygulayabilirsiniz.
- Ağ Otomasyonu: Ansible, ağ cihazlarının yapılandırma ve yönetimini otomatikleştirmek için kullanılabilir. Router'lar, switch'ler ve firewalls gibi ağ cihazlarının yapılandırma değişikliklerini yönetebilir ve ağ politikalarını uygulayabilirsiniz.

2. ANSIBLE KURULUMU VE YAPILANDIRMASI

2.1 Ansible'ın Gereksinimleri

Ansible, sunucularla iletişim kurmak ve yönetim işlemlerini gerçekleştirmek için SSH protokolünü kullanır. SSH, sunucular arasında güvenli iletişim ve veri transferi için kullanılan bir ağ protokolüdür. SSH, verileri güvence altına alarak yetkilendirilmemiş erişime karşı korur ve ağ üzerinde güvenli bir kanal oluşturur.Ansible, SSH'nin güvenlik özelliklerini kullanarak sunuculara bağlanır ve yönetim işlemlerini gerçekleştirir. (3)

Ansible'ın SSH kullanarak sunucularla iletişim kurabilmesi için aşağıdaki gereksinimleri karşılaması gerekir:

- Kontrol makinesinde SSH istemcisi yüklü olmalıdır. SSH istemcisi, kontrol makinesinin diğer sunuculara bağlanabilmesi için gerekli olan bileşendir.
- Sunucularda SSH sunucusu etkinleştirilmelidir. SSH sunucusu, diğer makinelerin SSH üzerinden sunucuya bağlanabilmesi için gereklidir.
- Ansible, SSH anahtar tabanlı kimlik doğrulamasını kullanarak sunuculara bağlanır. Anahtar tabanlı kimlik doğrulaması, güvenli ve kullanışlı bir erişim yöntemidir. Parolaların yerine anahtarlar kullanılarak güvenlik artırılır ve otomatik erişim sağlanır. Anahtar tabanlı kimlik doğrulamasında, bir özel anahtar (private key) kontrol makinesinde saklanırken, genel anahtar (public key) sunuculara eklenir. Bu anahtarlar, Ansible'ın sunuculara güvenli bir şekilde bağlanabilmesini sağlar.

  Kontrol makinesinde anahtar çifti oluşturmak için ssh-keygen komutu kullanılır.

```
ssh-keygen -t ed25519 -C "ansible"
```

Kontrol makinesinde oluşturulan genel anahtarı, sunuculara eklemek için ssh-copy-id komutu kullanılır.

```
ssh-copy-id -i ~/.ssh/ansible.pub
```

SSH anahtar tabanlı kimlik doğrulaması yapılandırıldıktan sonra eğer her şey düzgün yapılandırıldıysa, parola sormadan sunucuya SSH ile bağlanabileceksiniz.

2.2 Kurulum Adımları

Ansible'ın kurulumu için çeşitli yöntemler bulunmaktadır. Linux tabanlı sistemlerde, sistem paket yöneticisi aracılığıyla Ansible kurulabilir. Örneğin, Ubuntu veya Debian tabanlı bir sistemde apt komutu kullanılabilir. Aşağıda, apt komutu ile Ansible'ın kurulum adımları bulunmaktadır.

Güncel paket listesini indirmek ve Ansible'ı kurmak için aşağıdaki komutlar çalıştırılır:

```
sudo apt update
```

```
sudo apt install ansible
```

Ansible'ın başarıyla yüklendiğini doğrulamak için aşağıdaki komut çalıştırılır:

```
ansible --version
```

Ansible'ı yalnızca bir makineye kurarak, bu makine üzerinden diğer sunuculara yönetim işlemleri yapabilirsiniz. Ansible, merkezi bir kontrol noktası olarak kullanılır ve bu kontrol noktasından diğer sunuculara SSH üzerinden erişerek yönetim işlemlerini gerçekleştirir.

Ansible'ın agentless özelliği, hedef sistemlere özel bir ajan veya ek yazılım kurulumu gerektirmemesidir. Bu sayede, Ansible doğrudan hedef sistemlere SSH veya WinRM gibi iletişim protokolleri üzerinden erişerek işlemleri gerçekleştirir. Agentless yapısı, yönetim sürecini basitleştirir, kurulum ve yapılandırma maliyetlerini azaltır ve daha esnek bir şekilde sistemlere entegrasyon sağlar.

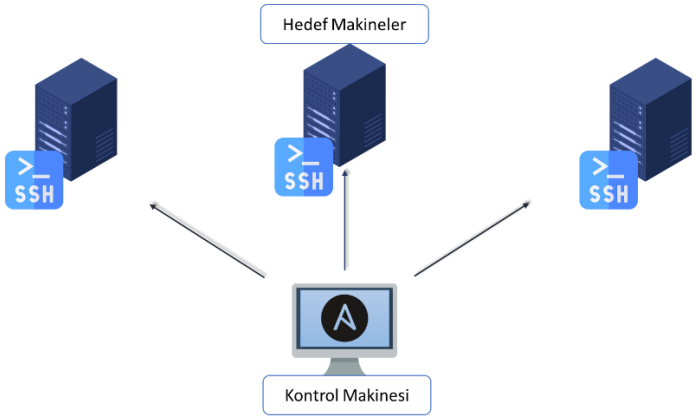
3. ANSIBLE TEMEL ELEMANLARI

3.1 Kontrol Makinesi (Control Machine)

Ansible'ın çalıştırıldığı ve yönetim işlemlerinin gerçekleştirildiği makinedir. Kontrol makinesi, Ansible komutlarını ve playbook'ları çalıştırmak için kullanılır.

3.2 Hedef Sistemler (Managed Nodes)

Ansible ile yönetilmek istenen, yapılandırmaların uygulanacağı ve görevlerin çalıştırılacağı makinelerdir. Bu sistemler genellikle sunucular, ağ cihazları veya sanal makineler olabilir. Ansible, hedef makinelerle genellikle SSH protokolü üzerinden iletişim kurar.



Şekil 3: Kontol Makinesi ve Hedef Makineler Arasındaki Bağlantı

3.3 Ansible Envanter Dosyası

Ansible'ın çalışacağı hedef sistemlerin listesini ve bağlantı bilgilerini içeren dosyadır. Ansible'ın hangi sistemler üzerinde işlemler yapması gerektiğini belirtir. Kontrol makinesi envanter dosyasını kullanarak hedef sistemlere bağlanır.

3.3.1 Envanter Dosyası Özellikleri

- Grup Hiyerarşisi: Envanter dosyasında gruplar ve alt gruplar (grup içinde grup) oluşturabilirsiniz. Bu, hedef sistemleri mantıksal olarak gruplandırmayı ve daha iyi bir organizasyonu sağlar.
- Değişkenler: Envanter dosyasında hedef sistemlere özel değişkenler tanımlayabilirsiniz. Bu değişkenler, playbook'larınızda kullanılarak hedef sistemlere özgü ayarlar yapmanızı sağlar.
- Çeşitli Formatlar: Envanter dosyası, farklı formatlarda oluşturulabilir. INI formatı (plain text) ve YAML formatı (structured data) en yaygın kullanılan formatlardır. Her iki format da gruplar, sistemler ve değişkenlerin tanımlanması için uygun yapılar sağlar.

3.3.2 Envanter Dosyası Örnek

```yaml
---
web:
  hosts:
    webserver1:
      ansible_host: 192.168.1.101
      ansible_user: ubuntu
    webserver2:
      ansible_host: 192.168.1.102
      ansible_user: ubuntu

database:
  hosts:
    dbserver1:
      ansible_host: 192.168.1.201

loadbalancer:
  hosts:
    lbserver1:
      ansible_host: 192.168.1.301

production:
  children:
    web
    database
```

Şekil 4:Envanter Dosyası Örneği

Yukarıdaki envanter dosyası, Ansible ile yönetilen sistemleri ve grupları tanımlamak için kullanılır. Sistemlerin IP adresleri, kullanıcı adları ve diğer özellikleri bu dosyada belirtilir ve Ansible'in bu sistemlere yönelik işlemleri gerçekleştirmesine olanak sağlar.

Bu envanter dosyasında aşağıdaki özelliklere dikkat edebiliriz:

- web, database, ve loadbalancer: Bu gruplar, sistemleri kategorize etmek için kullanılır. Her grup altında, ilgili sistemlerin hosts bölümünde tanımlanır. Örneğin, web grubunda webserver1 ve webserver2 bulunur.
- webserver1 ve webserver2: Bu sistemler, web grubu altında yer alır ve ilgili IP adresleri ansible_host anahtarında belirtilmiştir. Ayrıca, sistemlere bağlanmak için kullanılacak kullanıcı adı ansible_user ile belirtilir. Örneğin, webserver1 için kullanıcı adı ubuntu olarak belirlenmiştir.
- dbserver1: Bu sistem, database grubu altında yer alır ve IP adresi ansible_host anahtarında belirtilmiştir.
- lbserver1: Bu sistem, loadbalancer grubu altında yer alır ve IP adresi ansible_host anahtarında belirtilmiştir.
- production: Bu grup, web ve database gruplarını içeren bir üst grup olarak tanımlanır. Yani, production grubuna ait sistemler, web ve database gruplarında yer alan sistemleri içerir.

YAML dosyasının başında üç tire (---) kullanılmıştır. Bu, YAML belgesinin başlangıcını belirtir. YAML formatı, daha okunabilir ve yapısaldır. Anahtarlar ve değerler, girintileme (indentation) ile belirtilir. Gruplar ve sistemler iç içe geçmiş anahtarlarla temsil edilir. Değişkenler, anahtar-değer çiftleri olarak tanımlanır.

3.4 Ansible Modüller

Ansible modüller, Ansible'in farklı sistemler üzerinde çeşitli görevleri gerçekleştirmek için kullanılan fonksiyonel birimlerdir. Modüller, Ansible'in temel yapı taşlarıdır ve işletim sistemi yönetimi, ağ yapılandırması, veritabanı yönetimi, bulut hizmetleri entegrasyonu gibi çeşitli görevler için kullanılabilir.

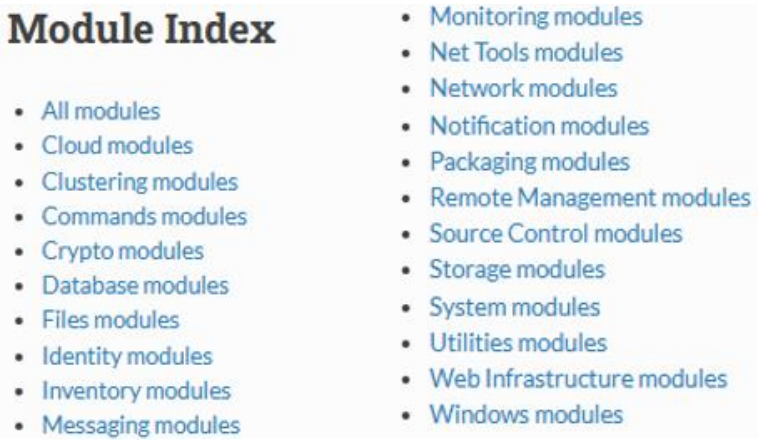
Her modül, belirli bir görevi gerçekleştirmek için kullanılır ve her defasında aynı sonucu üretecek şekilde tasarlanmıştır. Bu sayede, Ansible ile yapılan işlemler, sistemler üzerinde istenen durumu sağlama veya mevcut durumu koruma amacıyla güvenli bir şekilde tekrarlanabilir.

Ansible modüllerine, playbooks veya ad-hoc komutlar aracılığıyla erişilebilir. Playbooklar, modüllerin bir araya getirilerek karmaşık otomasyon senaryolarının oluşturulmasına olanak tanırken, ad-hoc komutlar ise tek seferlik görevlerin hızlı bir şekilde yürütülmesini sağlar.

Modül Örnekleri:

- apt modülü: Ubuntu veya Debian tabanlı sistemlerde paket yönetimi için kullanılır. Paketleri yükleme, kaldırma veya güncelleme gibi işlemler bu modülle gerçekleştirilebilir.
- service modülü: Hizmetlerin durumunu yönetmek için kullanılır. Hizmetin başlatılması, durdurulması veya yeniden başlatılması gibi işlemler bu modül aracılığıyla gerçekleştirilebilir.
- copy modülü: Dosyaları hedef sunuculara kopyalamak için kullanılır.
- command modülü: Komutları doğrudan hedef sunucuda çalıştırmak için kullanılır.

Bu örnekler, sadece birkaç Ansible modülünü temsil etmektedir. Ansible, birçok başka modül sağlar ve belirli bir işlem veya sistem üzerinde yapılabilecek çeşitli görevleri destekler. Her modülün kendi belgeleri vardır ve Ansible belgelerinde modül referansına ulaşabilirsiniz. (4)



Şekil 5:Ansible Module Index

3.5 Ansible Playbook

Hedef makinelerin üzerinde çalıştırılacak görevlerin listesi ve bu görevlerin nasıl gerçekleştirileceğine dair talimatları içerir.

Ansible playbook, Ansible'in otomasyon süreçlerini tanımlamak ve yürütmek için kullanılan bir dosya veya senaryo koleksiyonudur. Playbooklar, bir veya daha fazla görevi ve bunların hedef sistemlerdeki çalışma sırasını içeren yapılandırma ve otomasyon talimatlarını içerir. (5)

Her playbook, birden çok oynatıcıdan (play) oluşur ve her oynatıcı, belirli bir hedef üzerinde çalışacak bir dizi görevi temsil eder. Her görev, bir modülün kullanılmasıyla birlikte belirli bir işlemi gerçekleştirir.

Playbooklar, YAML formatında yazılır. YAML formatı, yapılandırmaları ve görevleri hiyerarşik bir şekilde ifade etmeyi kolaylaştırır. Bu sayede, playbookları anlamak, düzenlemek ve paylaşmak kolaylaşır. (6)

3.5.1 Playbook Temel Yapı

```
1   ---
2 - name: Playbook Name
3   hosts: target_hosts
4   become: true
5   become_user: root
6   vars:
7     variable_name: variable_value
8   tasks:
9     - name: Task 1
10      module_name:
11        parameter1: value1
12        parameter2: value2
13
14    - name: Task 2
15      module_name:
16        parameter1: value1
17        parameter2: value2
```

Şekil 6:Playbook Dosyalarının Temel Yapısı

Yukarıdaki örnekte, playbook aşağıdaki unsurlardan oluşur:

- name: Playbook'a verilen isimdir. İsim, playbook'ın amacını açıklamak için kullanılır ve isteğe bağlıdır.
- hosts: Playbook'un hangi hedef sistemler üzerinde çalışacağı belirtilir. Bu hedefler, IP adresleri, hostname'ler, gruplar veya dinamik envanter kullanılarak tanımlanabilir.

- become ve become_user: Playbook'un işlemleri hangi kullanıcı veya kullanıcılar yetkisiyle gerçekleştireceği belirtilir. ‘become: true’ ifadesi, işlemlerin kök kullanıcı (root) yetkileriyle gerçekleştirileceğini belirtir. ‘become_user’ ise hangi kullanıcı hesabının kullanılacağını belirler.
- vars: Playbook boyunca kullanılacak değişkenlerin tanımlandığı bölümdür. Bu değişkenler, playbook'ın farklı senaryolara uyum sağlamasını ve yapılandırmaları esnek hale getirmesini sağlar.
- tasks: Yürütülecek görevlerin listesidir. Her bir görev, bir modül ve o modülün parametreleri ile birlikte belirtilir.

Playbooklar, sistem yapılandırmaları, uygulama dağıtımları, hizmet yönetimi ve diğer otomasyon senaryolarını tanımlamak için kullanılır. Modüler yapıları sayesinde playbook'lar, tekrar kullanılabilir, okunabilir ve yönetilebilir hale gelir.

3.5.2 Playbook Öğeleri

- Handlers: Playbook'un sonunda çalıştırılacak belirli görevleri tanımlar. Handler'lar, bir oynatıcı içindeki görevlerin sonucuna bağlı olarak çalıştırılmak üzere belirtilir. Örneğin, bir hizmetin yeniden başlatılması veya yapılandırma dosyasının yeniden yüklenmesi gibi işlemler bir handler tarafından gerçekleştirilebilir. Handlers, belirli bir durumun meydana gelmesi durumunda tetiklenir.
- Conditions (Koşullar): Playbook içinde belirli koşullara dayalı olarak işlemlerin yönlendirilmesini sağlar. Koşullar, ‘when’ ifadesiyle belirtilir. Koşullar, hedef sistemlerin durumuna, değişkenlere veya diğer faktörlere bağlı olarak belirli işlemleri etkinleştirir veya devre dışı bırakır.
- Loops (Döngüler): Playbook içinde belirli görevlerin tekrarlanmasını sağlar. Döngüler, belirli bir liste veya sözlük üzerinde döner ve her döngü adımında belirtilen görevleri çalıştırır. Döngüler, yapılandırma tekrarını önler ve playbook'un daha kısa ve okunabilir olmasını sağlar.
- Roles(Roller): Bir playbook içinde belirli bir işlevselliği veya bir bileşeni temsil eden yapısal bir öğedir. Roller, playbook'ları daha modüler hale getirir ve daha iyi organize etmenizi sağlar. İşlevsel birimlerin veya bileşenlerin tekrar kullanılabilir olmasını sağlar ve aynı rolleri farklı projelerde veya senaryolarda yeniden kullanabilmenizi sağlar. Örneğin, bir web sunucusu rolü, farklı projelerdeki tüm web sunucularının yapılandırmasını ve yönetimini kolaylaştırır.

3.5.3 Playbook Örnekleri

```yaml
---
- name: Web Server Configuration
  hosts: webserver
  become: true

  tasks:
    - name: Install Apache Web Server
      apt:
        name: apache2
        state: present

    - name: Start Apache Service
      service:
        name: apache2
        state: started
        enabled: true
```

Şekil 7: Playbook Dosyası Örnek 1

Bu playbook, "Web Server Configuration" isimli bir görev kümesini ifade eder. Bu görevler, "webserver" adlı hedefler üzerinde root yetkisiyle çalışır (become: true).

Görevler, Apache web sunucusunun kurulumunu gerçekleştirir. Ardından Apache servisini başlatır ve otomatik olarak başlaması için yapılandırır. Her görev, bir ad (name) ve modül içerir. Örnekte iki görev bulunmaktadır.

İlk görev, "Install Apache Web Server" adını taşır ve "apt" modülü kullanarak "apache2" paketini kurulumunu gerçekleştirir. "state: present" parametresi, paketin yüklü olmasını sağlar.

İkinci görev, "Start Apache Service" adını taşır ve "service" modülü kullanarak "apache2" servisini başlatır ve otomatik başlaması için yapılandırır. "state: started" servisin çalışmasını, "enabled: true" ise otomatik başlamasını sağlar.

```yaml
tasks:
  - name: Install Apache Web Server
    apt:
      name: apache2
      state: present

  - name: Copy HTML Files
    copy:
      src: index.html
      dest: /var/www/html/

  - name: Start Apache Service
    service:
      name: apache2
      state: started
      enabled: true

vars:
  website_title: "My Website"
  website_description: "Welcome to my website!"

pre_tasks:
  - name: Update System Packages
    apt:
      update_cache: yes

post_tasks:
  - name: Clean Up Temporary Files
    file:
      path: /tmp/temp_file
      state: absent

handlers:
  - name: Restart Apache Service
    service:
      name: apache2
      state: restarted
```

Şekil 8: Playbook Dosyası Örnek 2

Bu playbook, Apache Web Sunucusu'nun kurulumunu gerçekleştirir, bir HTML dosyasını hedef sistemlere kopyalar, servisi başlatır ve belirli görevleri gerçekleştirirken değişkenler, pre-tasks, post-tasks ve handler'ları kullanır.

Görevler:

- "Install Apache Web Server" adlı görev, apt modülünü kullanarak Apache Web Sunucusu'nu kurar.
- "Copy HTML Files" adlı görev, copy modülünü kullanarak belirli bir HTML dosyasını hedef sistemlere kopyalar.
- "Start Apache Service" adlı görev, service modülünü kullanarak Apache servisini başlatır.

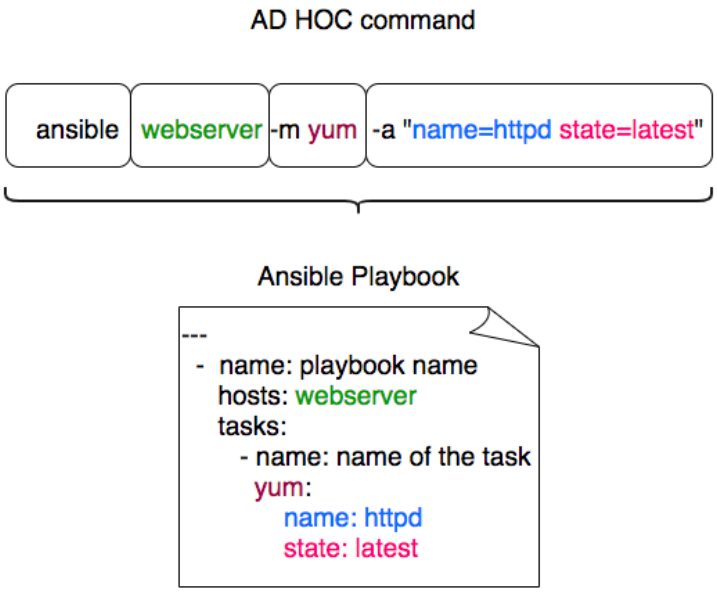
Değişkenler: website_title ve website_description gibi değişkenler, playbook içinde kullanılmak üzere tanımlanmıştır.

‘pre_tasks’ ve ‘post_tasks’ öğeleri, Ansible playbook'larında özel olarak tanımlanan bölümlerdir. Bu öğeler, playbook çalıştırılmadan önce veya sonra belirli görevleri gerçekleştirmek için kullanılır.

Pre-Tasks: "Update System Packages" adlı görev, playbook çalışmadan önce sistem paketlerini günceller.

Post-Tasks: "Clean Up Temporary Files" adlı görev, playbook çalıştıktan sonra geçici dosyaları siler.

Handler'lar: "Restart Apache Service" adlı handler, tetiklendiğinde Apache servisini yeniden başlatır.



Şekil 9: Ad-Hoc Komutlarının ve Playbook Dosyalarının Karşılaştırılması

Hedef makinede, oluşturulan playbook’ları çalıştırmak için aşağıdaki komut kullanılır.

```
ansible-playbook -i <envanter_dosyası> <playbook_dosyası>
```

3.6 Ansible Ad-Hoc Komutları

Ad-Hoc komutları, komut satırı aracılığıyla hızlı ve geçici bir şekilde hedef makinelerde belirli görevleri gerçekleştirmek için kullanılır. Ansible komut satırı aracılığıyla doğrudan yazılır ve hedef makineler üzerinde tek bir görev veya modül çalıştırılabilir. (7)

Ad-hoc komutları 'ansible' komutuyla çalıştırılır ve aşağıdaki gibi bir formata sahiptir:

ansible <hosts> -m <module> -a '<arguments>'

- <hosts>: Ad-hoc komutun uygulanacağı hedef sistemlerin veya grupların adı veya IP adresi.
- <module>: Kullanılacak Ansible modülünün adı. Örneğin, command, copy, apt, service gibi.
- <arguments>: Modülün gerektirdiği argümanlar. Örneğin, bir dosya kopyalama ad-hoc komutunda, kaynak ve hedef dosya yolları gibi argümanlar verilir.

Ad-hoc komutları, hızlı ve geçici çözümler için kullanışlıdır. Ancak daha karmaşık ve tekrarlanabilir görevler için playbook kullanmanız daha uygundur.

Ad-Hoc Komutlarına Örnek Ve Açıklama:

ansible all -m ping => SSH bağlantısı kontrol etmek için kullanılır.

ansible all --list-hosts => Ansible ile yönetilen tüm sunucuların listesini göstermek için kullanılır.

ansible all -m gather_facts => Tüm sunuculara bağlanacak ve her sunucudan sistem bilgilerini, ağ bilgilerini, donanım bilgilerini ve diğer faktörleri toplayacaktır. Sunucular hakkında ayrıntılı bilgilere erişmek için kullanılır.

ansible webservers -m apt -a 'name=nginx state=present' => Bu komut, "webservers" adlı hedef sistemlere, Nginx paketini kurmayı sağlar.

ansible web_servers -m service -a "name=nginx state=restarted" => Bu ad-hoc komutu, "web_servers" adlı bir grup sunucuya service modülünü kullanarak "name=nginx state=restarted" parametreleriyle "nginx" servisini yeniden başlatmayı amaçlar.

ansible webserver -m copy -a 'src=/path/to/local/file dest=/remote/path' => Bu komut, "webserver" adlı hedef sistemlere, yerel bir dosyayı belirtilen uzak konuma (/remote/path) kopyalar.

ansible servers -m shell -a "rm -rf path/to/folder/*" => Bu komut, "servers" adlı bir grup sunucuya shell modülünü kullanarak "rm -rf path/to/folder/*" komutunu yürütmeye çalışır. Komut, "path/to/folder/" dizinindeki tüm dosya ve alt dizinleri silmeyi amaçlar.

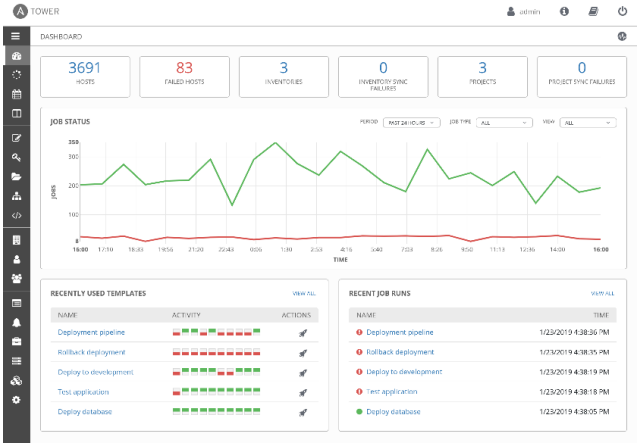
4. ANSIBLE EKOSİSTEMLERİ

4.1 Ansible Tower

Ansible Tower, Ansible otomasyon motorunu ve iş akışlarını merkezi bir şekilde yönetmek, takip etmek ve görselleştirmek için kullanılan bir web tabanlı bir uygulamadır. Tower, Ansible projelerini, rol ve iş akışlarını yönetmek ve takip etmek için kullanıcı dostu bir arayüz sunar.

Ansible Tower'ın başlıca özellikleri şunlardır:

- Merkezi Kontrol ve Yönetim: Ansible Tower, bir merkezi yönetim noktası olarak işlev görür. Birden çok Ansible projeyi ve rolü tek bir arayüzden yönetebilirsiniz.
- İş Akışları ve Sıralama: Tower, iş akışlarını ve sıralamaları oluşturmanıza olanak tanır. İş akışları, birden fazla adımdan oluşan otomasyon süreçlerini ifade eder. Sıralama ise bu adımların belirli bir sıra ile gerçekleştirilmesini sağlar.
- Görselleştirme ve İzleme: Ansible Tower, projelerinizi ve iş akışlarınızı görsel olarak temsil eder. İş akışlarının ilerlemesini, başarı durumunu, hata mesajlarını ve sonuçları izleyebilirsiniz. Böylece, iş süreçlerinizi daha iyi takip edebilir ve sorunları hızlı bir şekilde tespit edebilirsiniz.
- Rol Tabanlı Erişim Kontrolü: Tower, kullanıcıların ve ekiplerin belirli projeleri, rolleri ve iş akışlarını erişim düzeylerine göre kontrol etmelerini sağlar. Bu, güvenlik ve yetkilendirme açısından esneklik sağlar ve farklı kullanıcı rolleriyle çalışmanızı kolaylaştırır.
- Zamanlanmış Görevler: Ansible Tower, belirli zamanlarda otomatik olarak çalışacak görevlerin zamanlamasını yapmanıza izin verir. Bu özellik, tekrar eden görevleri otomatikleştirmenizi ve planlı bakım işlemlerini kolaylaştırır.
- API Desteği: Tower, RESTful API desteği sunar. Bu, diğer araçlar ve sistemlerle entegrasyon sağlamanızı ve otomasyon süreçlerinizi genişletmenizi sağlar. (8)



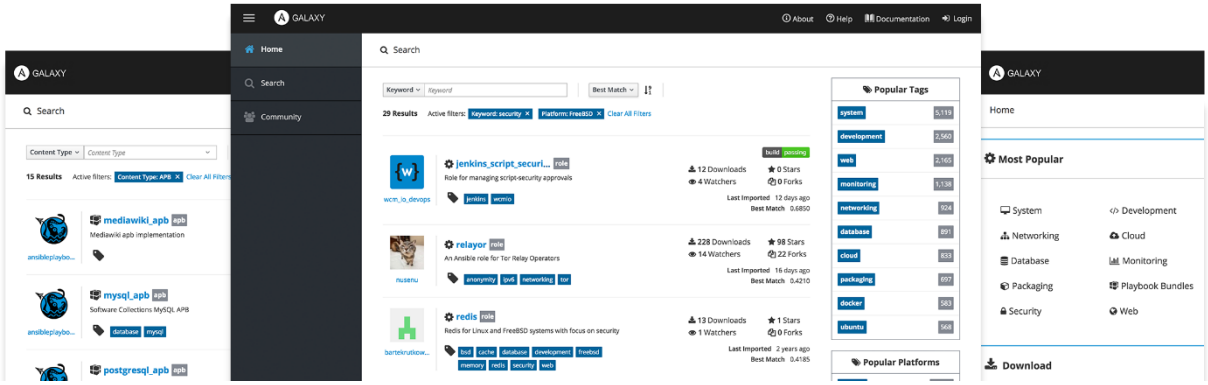
Şekil 10:Ansible Tower

4.2 Ansible Galaxy

Ansible Rollerinin paylaşıldığı ve topluluk tarafından oluşturulan bir depodur. Ansible Galaxy'de binlerce rol bulunur ve bu rolleri projelerinizde kullanarak hızlı bir şekilde yapılandırma ve otomasyon işlemleri gerçekleştirebilirsiniz.

Ansible Galaxy'nin temel özellikleri ve avantajları şunlardır:

- Rol Paylaşımı: Ansible Galaxy, Ansible rollerini paylaşmanızı sağlar. Topluluk üyeleri, projelerinde başarılı bir şekilde kullandıkları rolleri Galaxy'e yükleyebilir ve diğer kullanıcılarla paylaşabilir.
- Rol Sürüm Kontrolü: Ansible Galaxy, rollerin sürüm kontrolünü destekler. Roller, değişiklikler yapıldıkça sürüm numaralarıyla işaretlenebilir. Bu sayede rollerin belirli bir sürümünü kullanabilir veya güncellemeleri takip edebilirsiniz.
- Rol Bağımlılıkları: Ansible Galaxy'de paylaşılan roller, diğer rollerle bağımlılık ilişkilerine sahip olabilir. Bu, bir rolü kullanırken ona ihtiyaç duyulan diğer rolleri otomatik olarak indirme ve yapılandırma imkanı sağlar. (9)



Şekil 11: Ansible Galaxy

Bu, Ansible ekosisteminde yer alan temel bileşenlerden sadece birkaçıdır. Ansible, geniş bir topluluk desteği ve sürekli olarak geliştirilen yeni ekosistem bileşenleriyle dinamik bir yapıya sahiptir.

5. ANSIBLE ENTEGRASYONLAR

5.1 Ansible Ve Git Entegrasyonu

Ansible ve Git entegrasyonu, Ansible projelerinin ve yapılandırma dosyalarının Git versiyon kontrol sistemine entegre edilerek yönetilmesini sağlar. Bu entegrasyon, kod değişikliklerinin takip edilmesini, sürüm kontrolünü ve işbirliğini kolaylaştırır. (10)

Ansible projenizi Git ile entegre etmek için aşağıdaki adımları izleyebilirsiniz:

- Git Yükleme: İlk olarak, sisteminizde Git'in yüklü olduğundan emin olmanız gerekmektedir. Git'i resmi web sitesinden indirebilir ve sisteminize kurabilirsiniz.
- Ansible Projesinin Git Reposuna Bağlanma: Ansible projesini Git reposuna bağlamak için, projenizin bulunduğu dizine gidin ve bir Git deposu başlatın. Komut satırında 'git init' komutunu kullanabilirsiniz.
- Yapılandırma Dosyalarını Eklenme: Ansible projesindeki yapılandırma dosyalarını Git'e eklemeniz gerekmektedir. Bu dosyalar genellikle playbook.yml, inventory.ini gibi dosyalardır.

```
git add playbook.yml
git add inventory.ini
```

- İlk Commit: Eklediğiniz dosyaları bir commit'e dahil etmelisiniz. Bu commit, değişikliklerin bir sürümünü oluşturur.

```
git commit -m "İlk commit"
```

- Uzak Bir Git Deposu Ekleme: Projenizi bir Git deposuna yüklemek ve uzaktaki bir depoyla senkronize olmak istiyorsanız, bu adımı takip etmelisiniz. Uzak bir Git deposu eklemek için şu komutu kullanabilirsiniz:

```
git remote add origin <repo-url>
```

- Değişiklikleri Gönderme ve Çekme: Ansible projenizde değişiklik yaptığınızda, değişiklikleri Git deposuna göndermek (push) ve başkalarının değişikliklerini çekmek (pull) için aşağıdaki komutları kullanabilirsiniz:

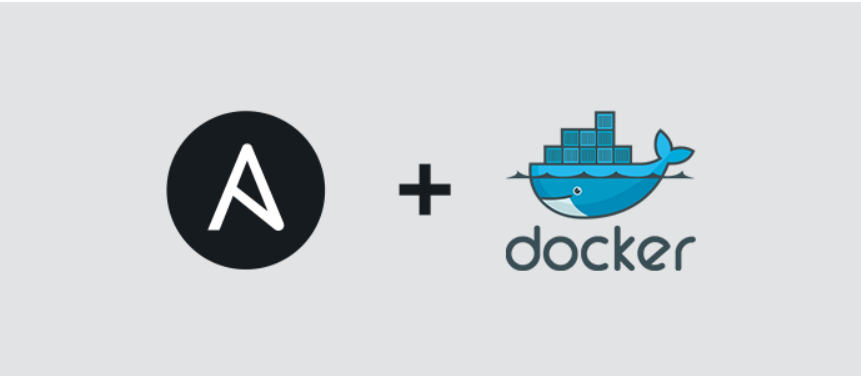
```
git push origin <branch-name>
git pull origin <branch-name>
```

5.2 Ansible Ve Docker Entegrasyonu

Ansible ve Docker entegrasyonu, Ansible ile Docker konteynerlerinin yönetimini ve dağıtımını kolaylaştıran bir süreçtir. Ansible, Docker API'si üzerinden Docker konteynerlerini oluşturabilir, çalıştırabilir, durdurabilir ve yönetebilir.

Ansible ile Docker entegrasyonunu sağlamak için aşağıdaki adımları izleyebilirsiniz:

- Ansible ve Docker Kurulumu: İlk olarak, hem Ansible'in hem de Docker'ın sisteminizde yüklü olduğundan emin olmanız gerekmektedir.
- Docker Modülünün Kullanılması: Ansible, Docker ile etkileşime geçmek için Docker modülünü sağlar. Docker modülü, Docker API'si üzerinden Docker konteynerlerini oluşturmanıza, başlatmanıza, durdurmanıza, silmenize ve diğer işlemleri gerçekleştirmenize olanak tanır. Ansible playbook'larınızda Docker modülünü kullanarak Docker konteynerlerini yönetebilirsiniz.
- Docker Playbook'larının Oluşturulması: Docker ile ilgili işlemleri gerçekleştiren Ansible playbook'ları oluşturmanız gerekmektedir. Örneğin, Docker konteynerlerini oluşturmak, konteynerlara yapılandırma uygulamak, Docker imajlarını güncellemek veya konteynerleri durdurmak gibi görevleri içerebilir.
- Docker Inventory'nin Ayarlanması: Ansible Inventory dosyasında, Docker konteynerlerinin IP adresleri, bağlantı bilgileri ve diğer ayrıntıları yer almalıdır. Bu, Ansible'in Docker konteynerleriyle iletişim kurmasını ve işlemleri gerçekleştirmesini sağlar.



Şekil 12:Ansible Ve Docker Entegrasyonu

6. ANSIBLE UYGULAMA ÖRNEĞİ

Bir Flask web uygulamasını Docker konteynerinde çalıştırmak için Ansible playbook'larını kullanacağız. Ansible'in Docker entegrasyonunu kullanarak uygulama dağıtımını otomatikleştireceğiz.

Uygulama Adımları:

1. Projenizin kök dizinine bir playbooks adlı bir klasör oluşturun.
2. Playbooks klasörü içinde, deploy.yml adında bir Ansible playbook dosyası oluşturun.
3. 'deploy.yml' dosyasını açın ve aşağıdaki gibi içeriğini doldurun:

```yaml
---
- name: Docker uygulama dağıtımı
  hosts: target_server
  become: true
  tasks:
    - name: Docker imajının indirilmesi
      docker_image:
        name: python:3.8-slim
        source: pull

    - name: Uygulama dizinini oluşturma
      file:
        path: /opt/flaskapp
        state: directory

    - name: Uygulama dosyalarının kopyalanması
      copy:
        src: app/
        dest: /opt/flaskapp/

    - name: Gerekli Python paketlerinin kurulması
      pip:
        requirements: /opt/flaskapp/requirements.txt

    - name: Flask uygulamasının başlatılması
      docker_container:
        name: flaskapp
        image: python:3.8-slim
        command: python /opt/flaskapp/app.py
        ports:
          - "5000:5000"
        volumes:
          - "/opt/flaskapp:/opt/flaskapp"
```

Şekil 13: Docker Entegrasyonunda Kullanılan Playbook Örneği

Bu playbook, Ansible aracılığıyla bir Flask uygulamasını Docker konteynerinde dağıtmak için kullanılır. İşlevleri şu adımları içerir:

- docker_image modülü aracılığıyla python:3.8-slim Docker imajını indirir.
- file modülü aracılığıyla /opt/flaskapp dizinini oluşturur.
- copy modülü aracılığıyla app/ dizinindeki uygulama dosyalarını /opt/flaskapp/ dizinine kopyalar.
- pip modülü aracılığıyla /opt/flaskapp/requirements.txt dosyasındaki Python paketlerini kurar.
- docker_container modülü aracılığıyla flaskapp adında bir Docker konteyneri oluşturur.
- Konteyner, python:3.8-slim imajını kullanır.
- python /opt/flaskapp/app.py komutunu çalıştırarak Flask uygulamasını başlatır.
- 5000 portunu hedef sunucunun 5000 portuna yönlendirir.
- /opt/flaskapp dizinini konteynerin /opt/flaskapp dizinine monte eder.

4. ‘hosts’ dosyasını oluşturun ve hedef sunucunun IP adresini veya DNS adını belirtin:

```
[target_servers]
192.168.0.101
192.168.0.102
```

5. Terminali açın ve aşağıdaki komutu çalıştırın:

```
ansible-playbook playbooks/deploy.yml -i hosts
```

Bu komut, deploy.yml playbook'unu hedef sunucuda çalıştırır ve Docker konteynerini oluşturur.

Sonuç olarak, Ansible playbook'u aracılığıyla Docker konteyneri başarıyla oluşturulur ve Flask uygulaması hedef sunucuda çalışır hâle gelir.

7. SONUÇLAR

Bu kılavuzda, Ansible otomasyon aracının genel işleyişini, kurulum adımları, temel bileşenleri ve kullanım alanları ele alındı. Ansible'ın basit yapılandırması ve anlaşılır syntax yapısı, otomasyon süreçlerini kolaylaştırmada önemli bir avantaj sağlamaktadır. Ansible Tower ve Ansible Galaxy gibi ekosistemler, Ansible'ın daha geniş bir kapsamda kullanılmasını desteklemekte ve yönetim arayüzleri ile rol/modül depolarıyla kullanıcıların verimliliğini artırmaktadır. Ayrıca, Ansible'ın Git ve Docker entegrasyonları, yazılım dağıtım süreçlerinde hızlı ve güvenilir bir şekilde uygulama yönetimi sağlamaktadır. Sonuç olarak, Ansible, otomasyon süreçlerinde etkili bir araç olarak kullanılarak zaman ve kaynak tasarrufu sağlayabilir, verimliliği artırabilir ve işletmelere rekabet avantajı sağlayabilir.

8. KAYNAKLAR

1. https://www.ansible.com/. [Çevrimiçi]

2. https://www.youtube.com/watch?v=1id6ERvfozo. [Çevrimiçi]

3. https://www.learnlinux.tv/getting-started-with-ansible-02-ssh-overview-setup/. [Çevrimiçi]

4. https://docs.ansible.com/ansible/2.8/modules/modules_by_category.html. [Çevrimiçi]

5. https://www.learnlinux.tv/getting-started-with-ansible-06-writing-our-first-playbook/. [Çevrimiçi]

6. https://www.digitalocean.com/community/tutorial_series/how-to-write-ansible-playbooks. [Çevrimiçi]

7. https://www.learnlinux.tv/getting-started-with-ansible-04-executing-ad-hoc-commands/. [Çevrimiçi]

8. https://www.simplilearn.com/what-is-ansible-tower-article. [Çevrimiçi]

9. https://galaxy.ansible.com/. [Çevrimiçi]

10. https://www.learnlinux.tv/getting-started-with-ansible-03-setting-up-the-git-repo/. [Çevrimiçi]

11. https://www.javatpoint.com/ansible

12. https://linuxhint.com/ansible-tutorial-beginners/

13. https://www.youtube.com/watch?v=5hycyr-8EKs

14. https://chat.openai.com/