In this session, I explored core DevOps concepts like **passwordless authentication**, **Ansible ad-hoc commands**, **playbooks**, and **roles**.

This doc will detail the key learnings and how they can be applied in real-world scenarios.

🔑 Passwordless Authentication in DevOps: A Must-Have for Efficiency

Automation and seamless communication between servers are critical, and **passwordless authentication** is a crucial component of this. Managing multiple servers without needing to enter passwords repeatedly saves time and enhances security.

Step-by-Step Guide:

- **Setting Up the Main and Target Servers**

- Create a **main server** to control operations and several target servers. I used AWS EC2 instances for this setup. The main server will operate remotely to control the target servers.

- **Login and System Update**

- SSH into your servers and update the system for stability:

```bash
```

```
sudo apt update && sudo apt upgrade
```

- **Install Ansible on the Main Server**

- Install **Ansible** for configuration management on the main server:

```bash
sudo apt install ansible
```

- **Generate SSH Keys**

- On the main server, generate SSH keys:

```bash
ssh-keygen
```

- **Copy Public Key to Target Servers**

- Extract your public key and paste it into the `authorized_keys` file on each target server:

```bash
cat ~/.ssh/id_rsa.pub
vim ~/.ssh/authorized_keys
```

- **Establish the Connection**

- From the main server, connect to a target server using its private IP:

```bash
ssh <private_ip_of_target_server>
```

- Done!

 - Once the passwordless connection is established, managing multiple servers becomes seamless.

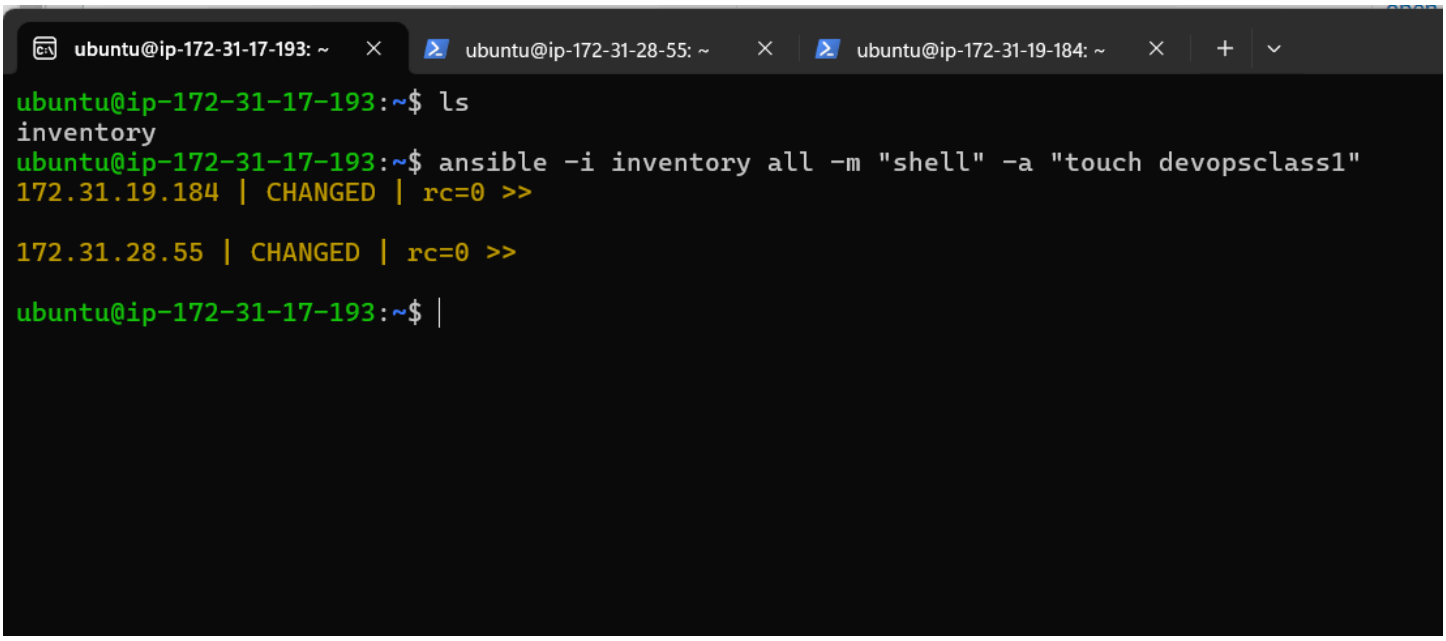
⚙️ Ansible Ad-hoc Commands: Automation Made Simple

Ansible simplifies automation, and **ad-hoc commands** allow you to perform quick tasks across multiple servers without a playbook.

Example: Creating a File on Target Servers

```bash
ansible -i inventory all -m "shell" -a "touch devopsclass"
```

Output image:



```
ubuntu@ip-172-31-17-193: ~   ubuntu@ip-172-31-28-55: ~   ubuntu@ip-172-31-19-184: ~
ubuntu@ip-172-31-28-55:~$ ls
devopsclass1
ubuntu@ip-172-31-28-55:~$ ls -ltr
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 Sep  4 13:55 devopsclass1
ubuntu@ip-172-31-28-55:~$
```

```
ubuntu@ip-172-31-17-193: ~   ubuntu@ip-172-31-28-55: ~   ubuntu@ip-172-31-19-184: ~
ubuntu@ip-172-31-19-184:~$ ls
devopsclass1
ubuntu@ip-172-31-19-184:~$ ls -ltr
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 Sep  4 13:55 devopsclass1
ubuntu@ip-172-31-19-184:~$
```

- **-i inventory:** Specifies the inventory file containing a list of target servers.

- **all:** Executes the command on all target servers.

- **-m shell:** Specifies the shell module for running shell commands.

- **-a "touch devopsclass":** Creates a file named `devopsclass`.

This ad-hoc command helps in automating simple tasks across multiple servers efficiently.

📜 Ansible Playbooks: Writing Reusable Code for Server Automation

While ad-hoc commands handle quick tasks, **Ansible Playbooks** are essential for automating complex workflows across servers.

My First Ansible Playbook to Install and Start Nginx

```first.yml
---
- name: Install and Start Nginx
  hosts: all
  become: true
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
    - name: Start nginx
      service:
        name: nginx
        state: started
```

Explanation:

- **hosts:** Defines the servers to run the playbook on.

- **become: true:** Grants superuser privileges to execute tasks.

- **tasks:** Each task installs and starts Nginx on the target servers.

To run the playbook:

```bash

ansible-playbook -i <inventory_file> first.yml

```

Output image of practiced Ansible-playbook for Installing and running Nginx:

```
ubuntu@ip-172-31-17-193: ~   ubuntu@ip-172-31-28-55: ~   ubuntu@ip-172-31-19-184: ~

ubuntu@ip-172-31-17-193:~$ ls
inventory  practice.yml
ubuntu@ip-172-31-17-193:~$ ansible-playbook -i inventory practice.yml

PLAY [Install nginx and Start nginx] ***********************************************************************************

TASK [Gathering Facts] *************************************************************************************************
ok: [172.31.19.184]
ok: [172.31.28.55]

TASK [Install nginx] ***************************************************************************************************
changed: [172.31.19.184]
changed: [172.31.28.55]

TASK [Start nginx] *****************************************************************************************************
ok: [172.31.19.184]
ok: [172.31.28.55]

PLAY RECAP *************************************************************************************************************
172.31.19.184              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.28.55               : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-17-193:~$
```

```
ubuntu@ip-172-31-17-193: ~   ubuntu@ip-172-31-28-55: ~

ubuntu@ip-172-31-28-55:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset:
 enabled)
     Active: active (running) since Wed 2024-09-04 14:06:47 UTC; 4min 34s ag
o
       Docs: man:nginx(8)
    Process: 2089 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_pr
ocess on; (code=exited, status=0/SUCCESS)
    Process: 2091 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
 (code=exited, status=0/SUCCESS)
   Main PID: 2092 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 1.7M (peak: 1.8M)
        CPU: 12ms
     CGroup: /system.slice/nginx.service
             ├─2092 "nginx: master process /usr/sbin/nginx -g daemon on; mas
ter_process on;"
             └─2093 "nginx: worker process"

Sep 04 14:06:47 ip-172-31-28-55 systemd[1]: Starting nginx.service - A high
performance web server and a reverse proxy >
Sep 04 14:06:47 ip-172-31-28-55 systemd[1]: Started nginx.service - A high p
erformance web server and a reverse proxy s>
lines 1-16/16 (END)...skipping...
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset>
     Active: active (running) since Wed 2024-09-04 14:06:47 UTC; 4min 34s a>
       Docs: man:nginx(8)
    Process: 2089 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_p>
    Process: 2091 ExecStart=/usr/sbin/nginx -g daemon on; master_process on>
   Main PID: 2092 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 1.7M (peak: 1.8M)
        CPU: 12ms
     CGroup: /system.slice/nginx.service
             ├─2092 "nginx: master process /usr/sbin/nginx -g daemon on; ma>
             └─2093 "nginx: worker process"

Sep 04 14:06:47 ip-172-31-28-55 systemd[1]: Starting nginx.service - A high>
Sep 04 14:06:47 ip-172-31-28-55 systemd[1]: Started nginx.service - A high >
~
~
~
~
~
```

```
ubuntu@ip-172-31-19-184: ~

ubuntu@ip-172-31-19-184:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset:
 enabled)
     Active: active (running) since Wed 2024-09-04 14:06:47 UTC; 4min 44s ag
o
       Docs: man:nginx(8)
    Process: 2141 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_pr
ocess on; (code=exited, status=0/SUCCESS)
    Process: 2147 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
 (code=exited, status=0/SUCCESS)
   Main PID: 2149 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 1.7M (peak: 1.9M)
        CPU: 16ms
     CGroup: /system.slice/nginx.service
             ├─2149 "nginx: master process /usr/sbin/nginx -g daemon on; mas
ter_process on;"
             └─2150 "nginx: worker process"

Sep 04 14:06:47 ip-172-31-19-184 systemd[1]: Starting nginx.service - A high
 performance web server and a reverse proxy>
Sep 04 14:06:47 ip-172-31-19-184 systemd[1]: Started nginx.service - A high
performance web server and a reverse proxy >
lines 1-16/16 (END)...skipping...
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset>
     Active: active (running) since Wed 2024-09-04 14:06:47 UTC; 4min 44s a>
       Docs: man:nginx(8)
    Process: 2141 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_p>
    Process: 2147 ExecStart=/usr/sbin/nginx -g daemon on; master_process on>
   Main PID: 2149 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 1.7M (peak: 1.9M)
        CPU: 16ms
     CGroup: /system.slice/nginx.service
             ├─2149 "nginx: master process /usr/sbin/nginx -g daemon on; ma>
             └─2150 "nginx: worker process"

Sep 04 14:06:47 ip-172-31-19-184 systemd[1]: Starting nginx.service - A hig>
Sep 04 14:06:47 ip-172-31-19-184 systemd[1]: Started nginx.service - A high>
~
~
~
~
~
```

🔄 Ansible Roles: Simplifying Complex Workflows

Ansible roles help in breaking down playbooks into reusable components, making them ideal for managing complex infrastructure.

Experimenting with JBoss-standalone Role

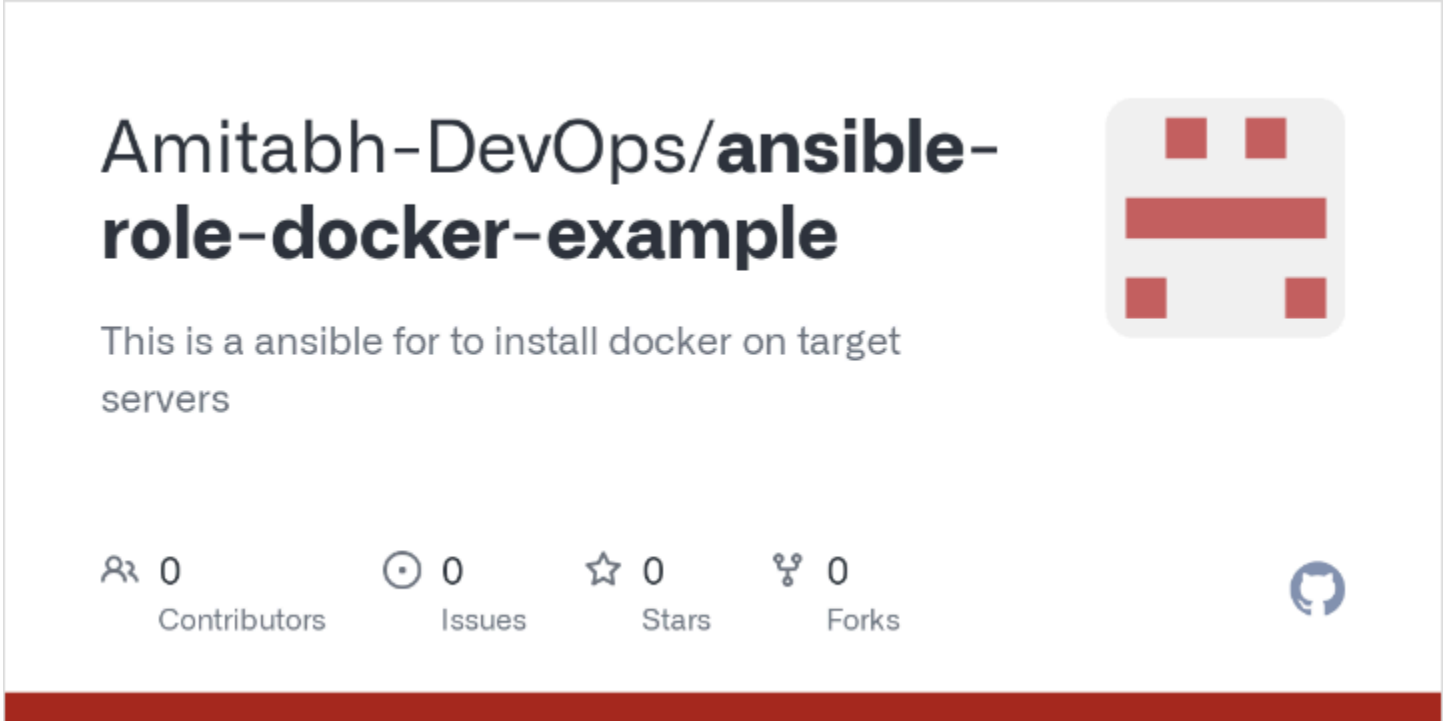
- I attempted to create a role for **JBoss-standalone**, but encountered issues, likely due to outdated configurations.

<u>Successful Docker Installation Role</u>

- I successfully used a pre-built **Docker role** from **Ansible Galaxy** by Jeff Geerling.

Docker Role from Ansible Galaxy:

🔗 Ansible Role for Docker :



Roles make managing large-scale infrastructure more efficient by organizing tasks into smaller components.

🎓 Key Takeaways from This Lecture

1. **Passwordless Authentication**:

   - Essential for managing multiple servers efficiently and securely.

2. **Ansible Ad-hoc Commands**:

   - Ideal for quick, one-off tasks across multiple servers.

3. **Ansible Playbooks**:

   - Automates complex, repeatable tasks across servers.

4. **Ansible Roles**:

   - Simplifies complex workflows by breaking them into reusable components