Project 01

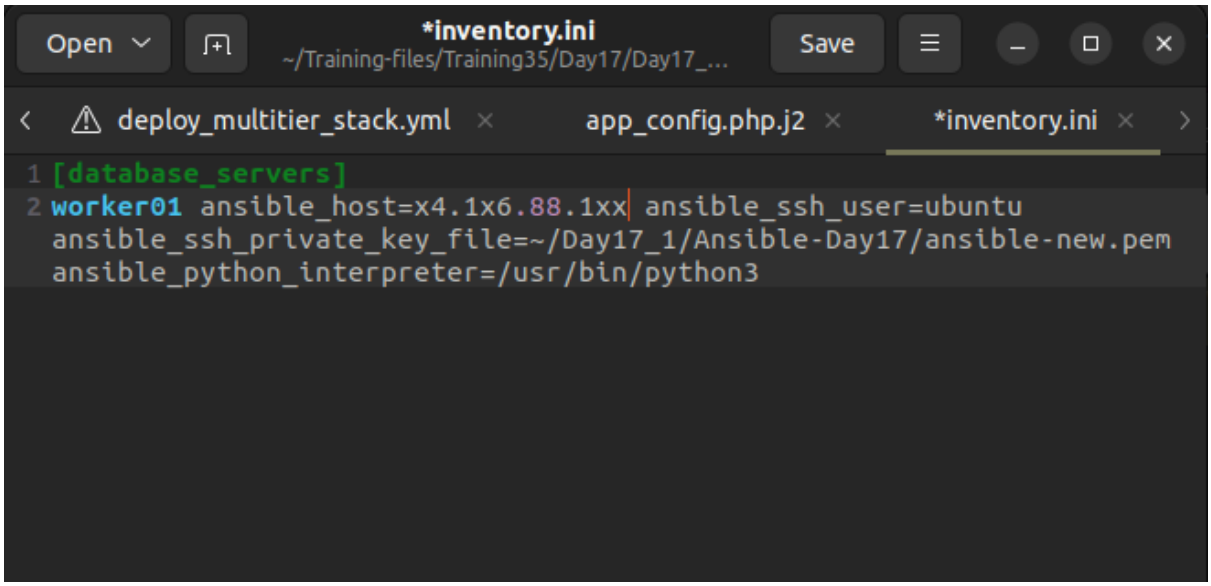
Deploy a Database Server with Backup Automation

Objective: Automate the deployment and configuration of a PostgreSQL database server on an Ubuntu instance hosted on AWS, and set up regular backups.

Problem Statement

Objective: Automate the deployment, configuration, and backup of a PostgreSQL database server on an Ubuntu instance using Ansible.

1. **Ansible Inventory File**
    - **Filename**: `inventory.ini`
    - **Content**: Defines the AWS Ubuntu instance and connection details for Ansible.



2. **Ansible Playbook**
    - **Filename**: `deploy_database.yml`
    - **Content**: Automates the installation of MySQL, sets up the database, creates a user, and configures a cron job for backups. It also includes variables for database configuration and backup settings.

deploy_multitier_stack.yml

```yaml
- name: Deploy and configure MySQL database
  hosts: db_server
  become: yes
  vars:
    db_name: "my_database"
    db_user: "my_user"
    db_password: "user123"

  tasks:
  - name: Install MySQL server
    apt:
      update_cache: yes
      name: "{{ item }}"
      state: present
    with_items:
    - mysql-server
    - mysql-client
    - python3-mysqldb
    - libmysqlclient-dev

  - name: Ensure MySQL service is running
    service:
      name: mysql
      state: started
      enabled: yes

  - name: Create MySQL user
    mysql_user:
      name: "{{ db_user }}"
      password: "{{ db_password }}"
      priv: '*.*:ALL'
      host: '%'
      state: present

  - name: Create MySQL database
```

deploy_multitier_stack.yml

```
      host: '%'
      state: present

  - name: Create MySQL database
    mysql_db:
      name: "{{ db_name }}"
      state: present

- name: Deploy and configure web server and application
  hosts: web_server
  become: yes

  vars:
    db_host: "host_ip"
    db_name: "my_database"
    db_user: "my_user"
    db_password: "user123"

  tasks:
  - name: Install web server
    apt:
      name: nginx
      state: present
      update_cache: yes

  - name: Ensure web server is running
    service:
      name: nginx
      state: started
      enabled: yes

  - name: Deploy application files
    copy:
      src: files/index.html
      dest: /var/www/html/index.html
```

```
  - name: Configure application
    template:
      src: templates/app_config.php.j2
      dest: /var/www/html/app_config.php

  - name: Restart web server to apply changes
    service:
      name: nginx
      state: restarted
```

3. **Jinja2 Template**

- **Filename**: `templates/mysql.cnf.j2`
- **Content**: Defines the MySQL configuration file (`mysql.conf`) using Jinja2 templates to manage access controls dynamically.

mysql.cnf.j2

```
# Here is entries for some specific programs
# The following values assume you have at least 32M ram

!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mysql.conf.d/

```

4. **Backup Script**
   - **Filename**: `scripts/backup.sh`
   - **Content**: A script to perform the backup of the MySQL database. This script should be referenced in the cron job defined in the playbook.

backup.sh

```
#!/bin/bash

DB_NAME=$1
DB_USER=$2
DB_PASSWORD=$3
BACKUP_DIR=$4
DATE=$(date +%Y%m%d)
BACKUP_FILE="${BACKUP_DIR}/${DB_NAME}_${DATE}.sql.gz"

# Create backup
mysqldump -u $DB_USER -p$DB_PASSWORD $DB_NAME | gzip > $BACKUP_FILE

# Optional: Remove backups older than 7 days
find $BACKUP_DIR -type f -name "*.gz" -mtime +7 -exec rm {} \;

```

```
einfochips@AHMLPT2484:~/Day17_1/Ansible-Day17$ nano inventory.ini
einfochips@AHMLPT2484:~/Day17_1/Ansible-Day17$ nano deploy_database.yml
einfochips@AHMLPT2484:~/Day17_1/Ansible-Day17$ ansible-playbook -i inventory.ini deploy_database.yml

PLAY [Deploy and configure MySQL database server] ******************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [worker01]

TASK [Install MySQL] ***********************************************************************************************************
ok: [worker01] => (item=mysql-server)
ok: [worker01] => (item=mysql-client)
ok: [worker01] => (item=python3-mysqldb)
ok: [worker01] => (item=libmysqlclient-dev)

TASK [Copy MySQL configuration file] *******************************************************************************************
ok: [worker01]

TASK [Ensure MySQL service is running] *****************************************************************************************
ok: [worker01]

TASK [Create MySQL database] ***************************************************************************************************
ok: [worker01]

TASK [Create MySQL user and grant privileges] **********************************************************************************
[WARNING]: Option column_case_sensitive is not provided. The default is now false, so the column's name will be uppercased. The default will
be changed to true in community.mysql 4.0.0.
ok: [worker01]

TASK [Create backup directory] *************************************************************************************************
ok: [worker01]

TASK [Copy MySQL backup script] ************************************************************************************************
ok: [worker01]

TASK [Set up cron job for daily backups] ***************************************************************************************
ok: [worker01]

PLAY RECAP *********************************************************************************************************************
worker01                   : ok=9    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

einfochips@AHMLPT2484:~/Day17_1/Ansible-Day17$
```

Project 02

Objective: Automate the setup of a multi-tier web application stack with separate database and application servers using Ansible.

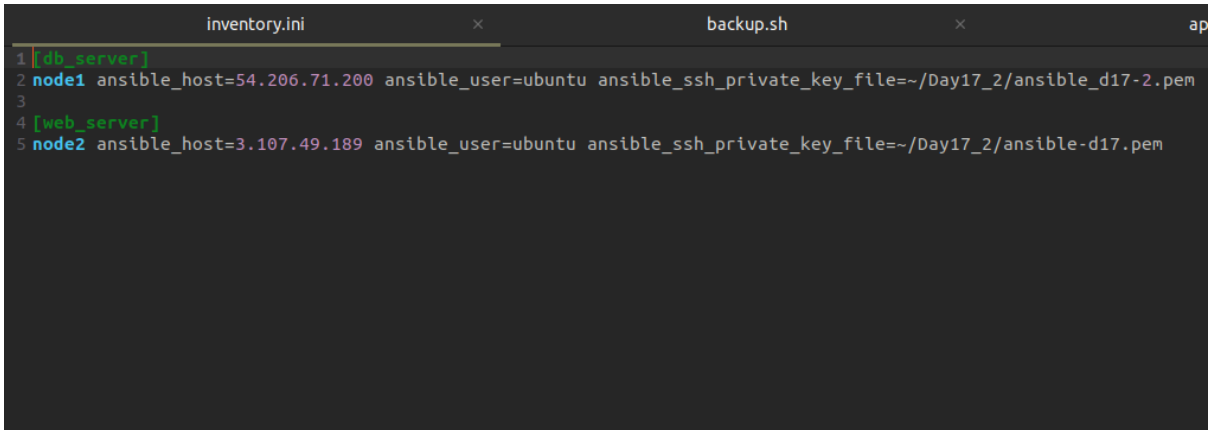
Problem Statement

Objective: Automate the deployment and configuration of a multi-tier web application stack consisting of:

1. **Database Server**: Set up a PostgreSQL database server on one Ubuntu instance.
2. **Application Server**: Set up a web server (e.g., Apache or Nginx) on another Ubuntu instance to host a web application.
3. **Application Deployment**: Ensure the web application is deployed on the application server and is configured to connect to the PostgreSQL database on the database server.
4. **Configuration Management**: Use Ansible to automate the configuration of both servers, including the initialization of the database and the deployment of the web application.

Deliverables

1. **Ansible Inventory File**
    - **Filename**: `inventory.ini`
    - **Content**: Defines the database server and application server instances, including their IP addresses and connection details.



2. **Ansible Playbook**
    - **Filename**: `deploy_multitier_stack.yml`
    - **Content**: Automates:
        - The deployment and configuration of the PostgreSQL database server.
        - The setup and configuration of the web server.
        - The deployment of the web application and its configuration to connect to the database.

deploy_multitier_stack.yml

```yaml
- name: Deploy and configure MySQL database
  hosts: db_server
  become: yes
  vars:
    db_name: "my_database"
    db_user: "my_user"
    db_password: "user123"

  tasks:
  - name: Install MySQL server
    apt:
      update_cache: yes
      name: "{{ item }}"
      state: present
    with_items:
    - mysql-server
    - mysql-client
    - python3-mysqldb
    - libmysqlclient-dev

  - name: Ensure MySQL service is running
    service:
      name: mysql
      state: started
      enabled: yes

  - name: Create MySQL user
    mysql_user:
      name: "{{ db_user }}"
      password: "{{ db_password }}"
      priv: '*.*:ALL'
      host: '%'
      state: present

```

deploy_multitier_stack.yml

```yaml
  - name: Create MySQL database
    mysql_db:
      name: "{{ db_name }}"
      state: present

- name: Deploy and configure web server and application
  hosts: web_server
  become: yes

  vars:
    db_host: "host_ip"
    db_name: "my_database"
    db_user: "my_user"
    db_password: "user123"

  tasks:
  - name: Install web server
    apt:
      name: nginx
      state: present
      update_cache: yes

  - name: Ensure web server is running
    service:
      name: nginx
      state: started
      enabled: yes

  - name: Deploy application files
    copy:
      src: files/index.html
      dest: /var/www/html/index.html
```

```yaml

  - name: Configure application
    template:
      src: templates/app_config.php.j2
      dest: /var/www/html/app_config.php

  - name: Restart web server to apply changes
    service:
      name: nginx
      state: restarted
```

3. **Jinja2 Template**
    - **Filename**: `templates/app_config.php.j2`
    - **Content**: Defines a configuration file for the web application that includes placeholders for dynamic values such as database connection details.

Open

app_config.php.j2
~/Day17_2/templates

```
<?php
$servername = "{{ db_host }}";
$username = "{{ db_user }}";
$password = "{{ db_password }}";
$dbname = "{{ db_name }}";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Output:

einfochips@AHMLPT2484: ~/Day17_2

```
einfochips@AHMLPT2484:~/Day17_2$ nano deploy_multitier_stack.yml
einfochips@AHMLPT2484:~/Day17_2$ nano inventory.ini
einfochips@AHMLPT2484:~/Day17_2$ ansible-playbook -i inventory.ini deploy_multitier_stack.yml

PLAY [Deploy and configure MySQL database] *************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [node1]

TASK [Install MySQL server] ****************************************************************************************************************************************
ok: [node1] => (item=mysql-server)
ok: [node1] => (item=mysql-client)
ok: [node1] => (item=python3-mysqldb)
ok: [node1] => (item=libmysqlclient-dev)

TASK [Ensure MySQL service is running] *****************************************************************************************************************************
ok: [node1]

TASK [Create MySQL user] *******************************************************************************************************************************************
[WARNING]: Option column_case_sensitive is not provided. The default is now false, so the column's name will be uppercased. The default will
be changed to true in community.mysql 4.0.0.
changed: [node1]

TASK [Create MySQL database] ***************************************************************************************************************************************
changed: [node1]

PLAY [Deploy and configure web server and application] **************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [node2]

TASK [Install web server] *******************************************************************************************************************************************
ok: [node2]

TASK [Ensure web server is running] ********************************************************************************************************************************
[WARNING]: sftp transfer mechanism failed on [3.107.49.189]. Use ANSIBLE_DEBUG=1 to see detailed information
ok: [node2]
```

```
TASK [Install web server] ******************************************************************************************
ok: [node2]

TASK [Ensure web server is running] ********************************************************************************
[WARNING]: sftp transfer mechanism failed on [3.107.49.189]. Use ANSIBLE_DEBUG=1 to see detailed information
ok: [node2]

TASK [Deploy application files] ************************************************************************************
ok: [node2]

TASK [Configure application] ***************************************************************************************
changed: [node2]

TASK [Restart web server to apply changes] *************************************************************************
changed: [node2]

PLAY RECAP *********************************************************************************************************
node1                      : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2                      : ok=6    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. **Application Files**
    - **Filename**: `files/index.html` (or equivalent application files)
    - **Content**: Static or basic dynamic content served by the web application.