#### **Day17**

### **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**

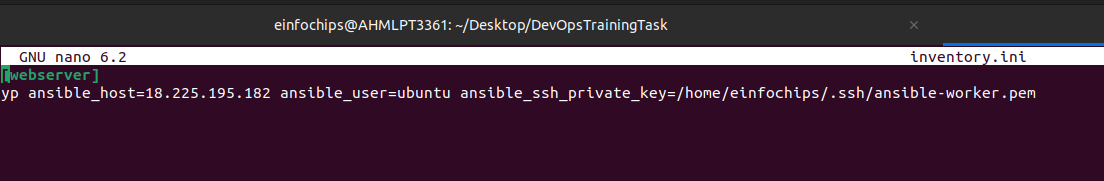
**Objective**: Automate the deployment, configuration, and backup of a PostgreSQL database server on an Ubuntu instance using Ansible.

**Requirements**:

1. **AWS Ubuntu Instance**: You have an Ubuntu server instance running on AWS.
2. **Database Server Deployment**: Deploy and configure PostgreSQL on the Ubuntu instance.
3. **Database Initialization**: Create a database and a user with specific permissions.
4. **Backup Automation**: Set up a cron job for regular database backups and ensure that backups are stored in a specified directory.
5. **Configuration Management**: Use Ansible to handle the deployment and configuration, including managing sensitive data like database passwords.

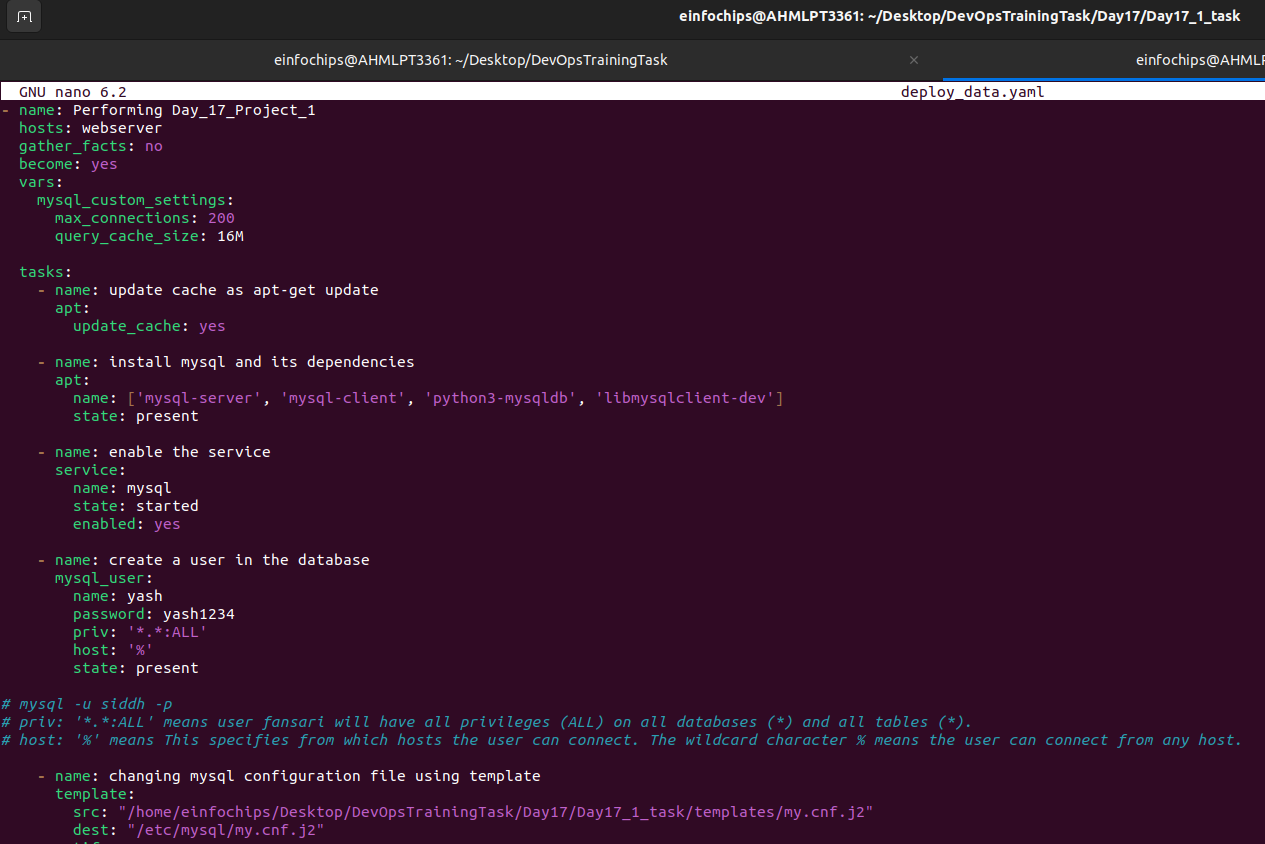
### **Deliverables**

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

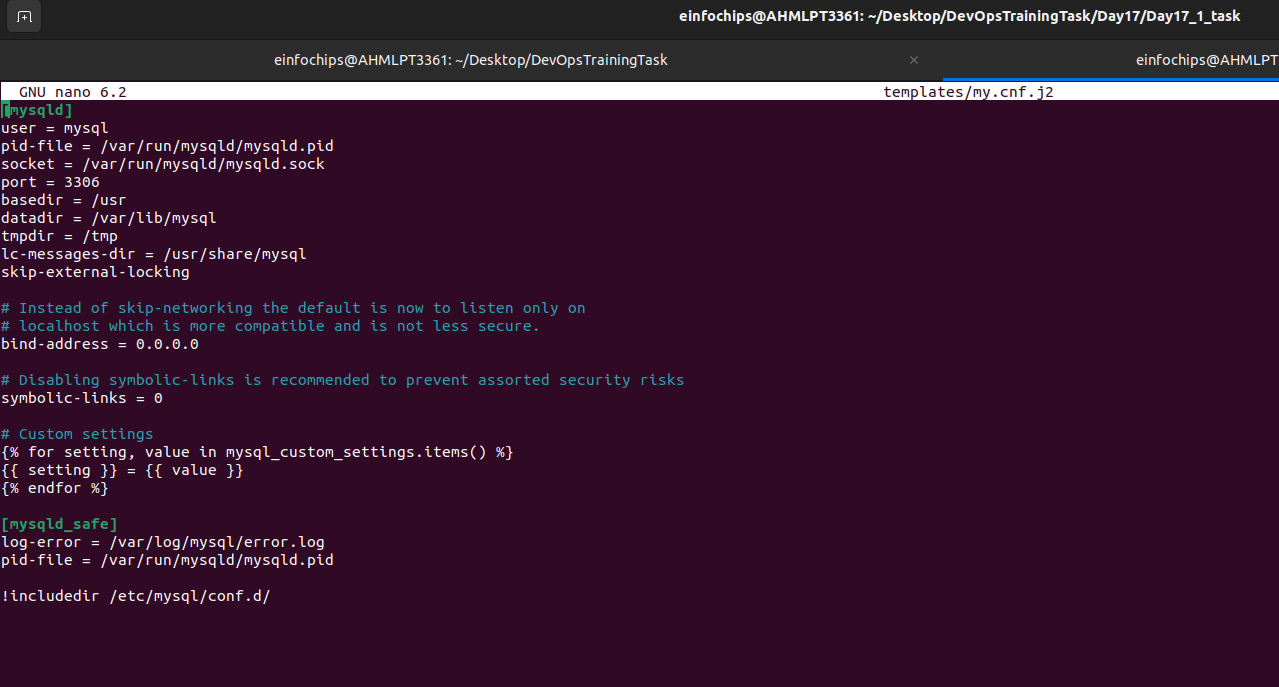


**2. Ansible Playbook**

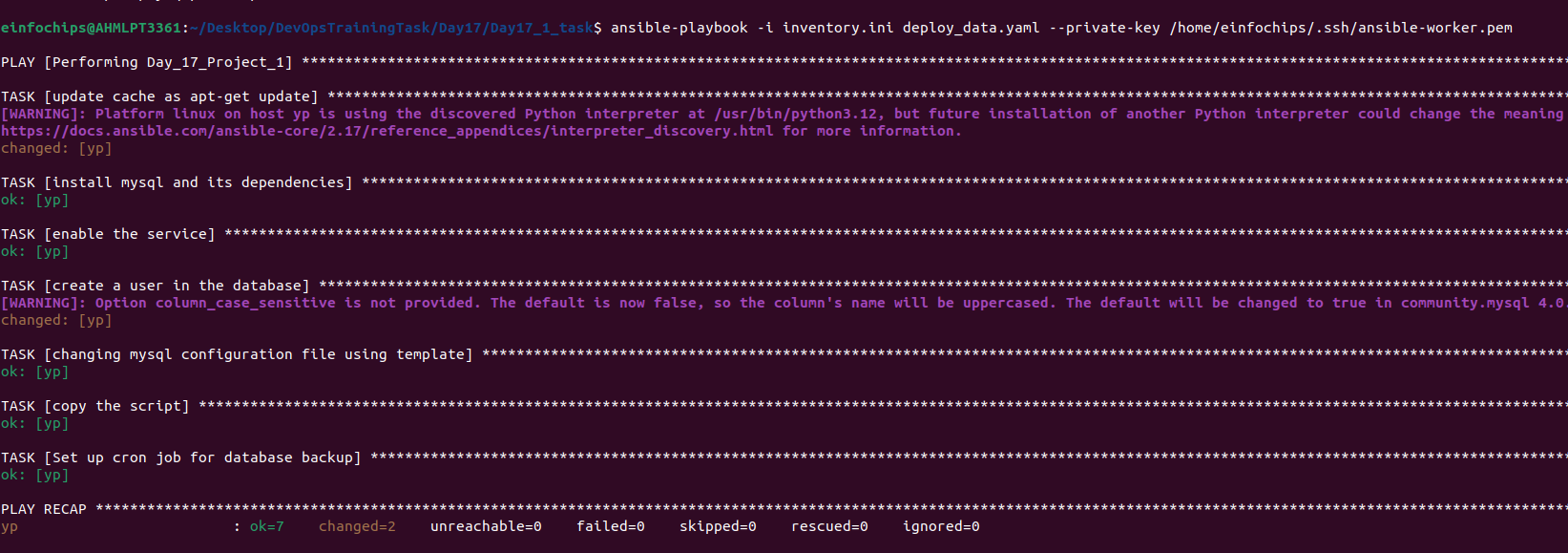
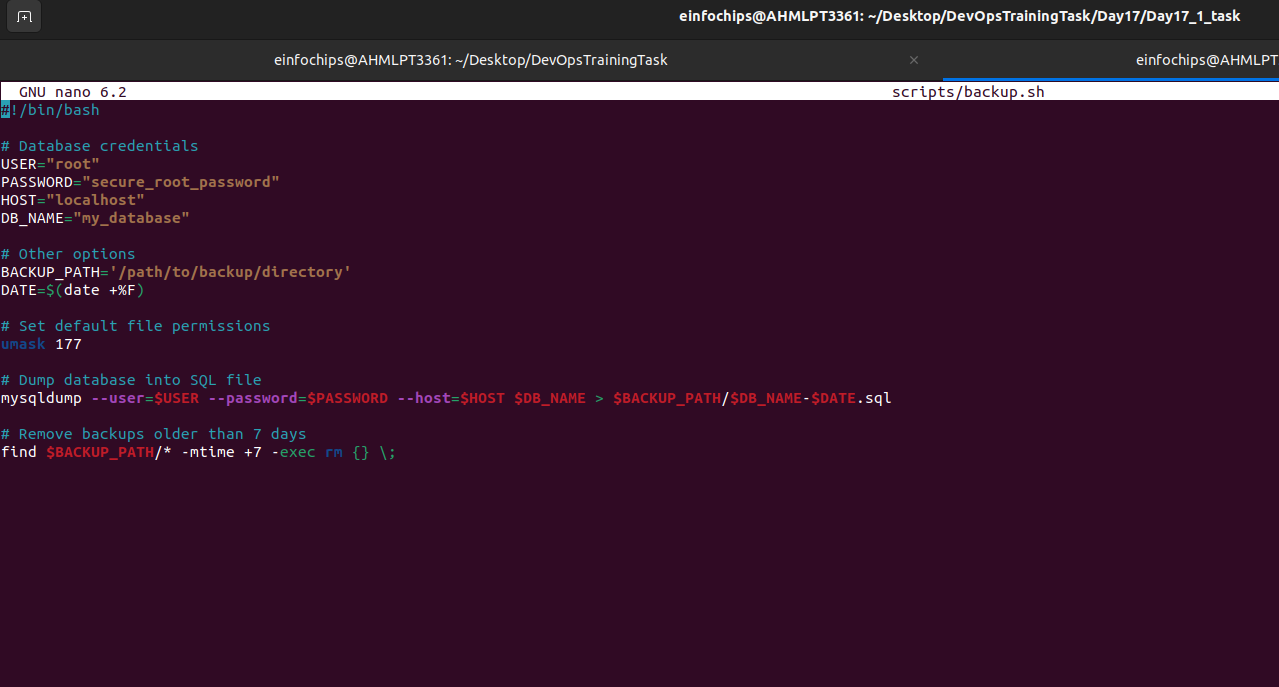
* + **Filename**: deploy\_database.yml
  + **Content**: Automates the installation of PostgreSQL, sets up the database, creates a user, and configures a cron job for backups. It also includes variables for database configuration and backup settings.



1. **Jinja2 Template**
   * **Filename**: templates/pg\_hba.conf.j2
   * **Content**: Defines the PostgreSQL configuration file (pg\_hba.conf) using Jinja2 templates to manage access controls dynamically.



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



**Project 02**

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

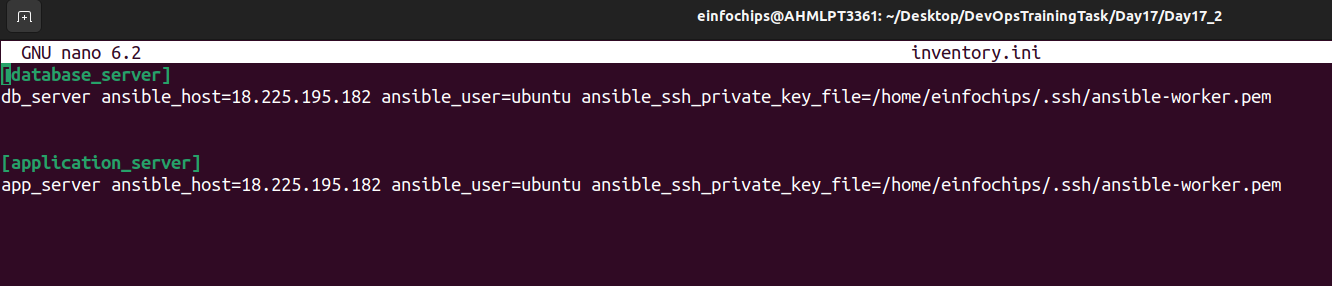
### **Problem Statement**

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

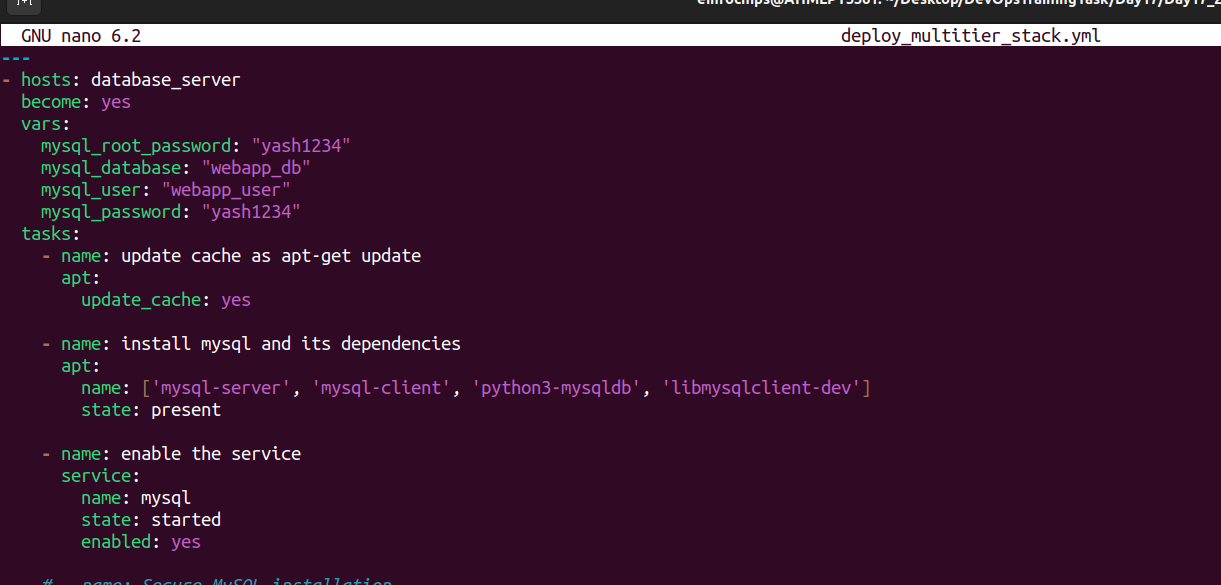
1. **Database Server**: Set up a PostgreSQL database server on one Ubuntu instance.
2. **Application Server**: Set up a web server (e.g., Apache or Nginx) on another Ubuntu instance to host a web application.
3. **Application Deployment**: Ensure the web application is deployed on the application server 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**

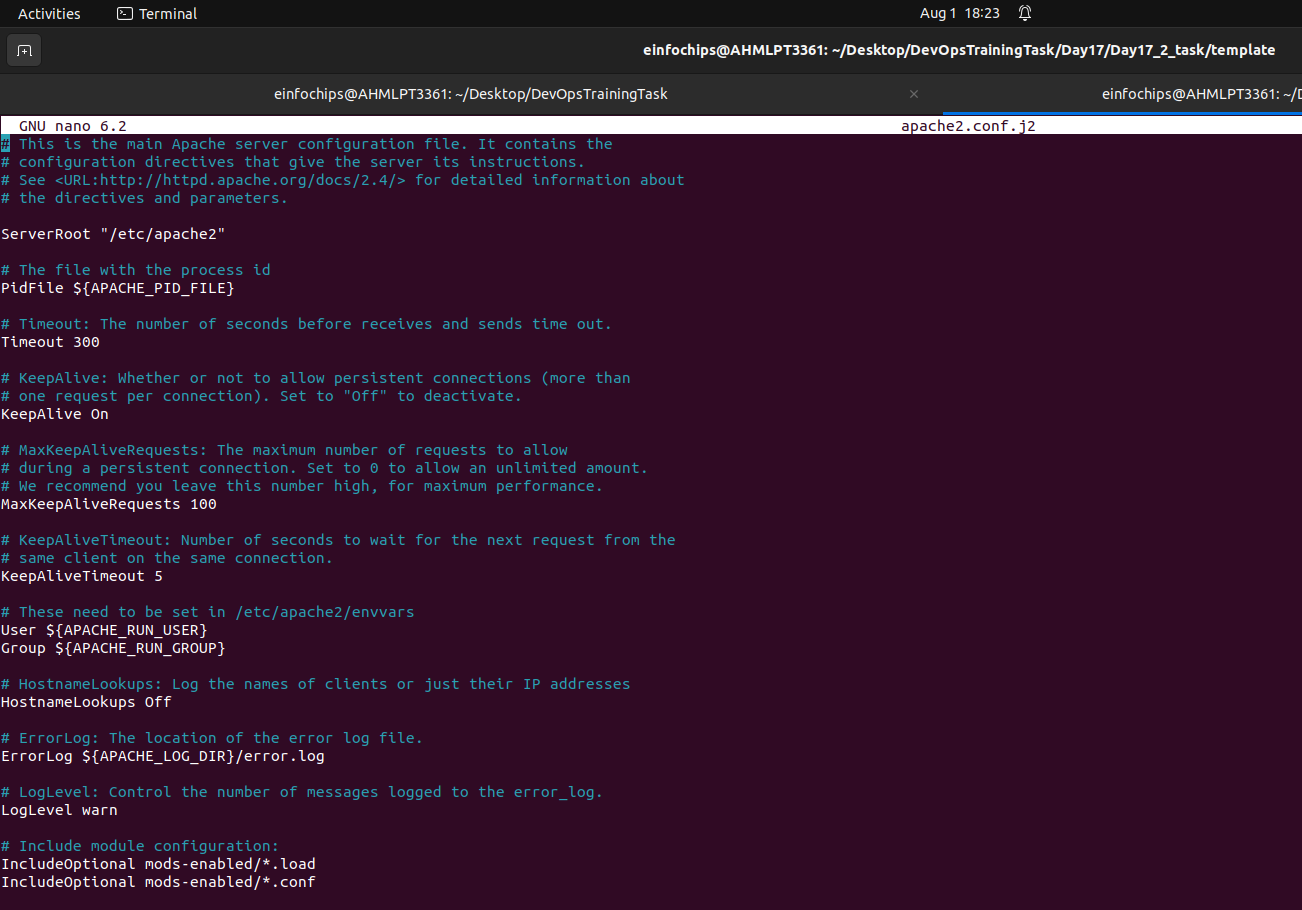
1. **Ansible Inventory File**
   * **Filename**: inventory.ini
   * **Content**: Defines the database server and application server instances, including their IP addresses and connection details.



1. **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.

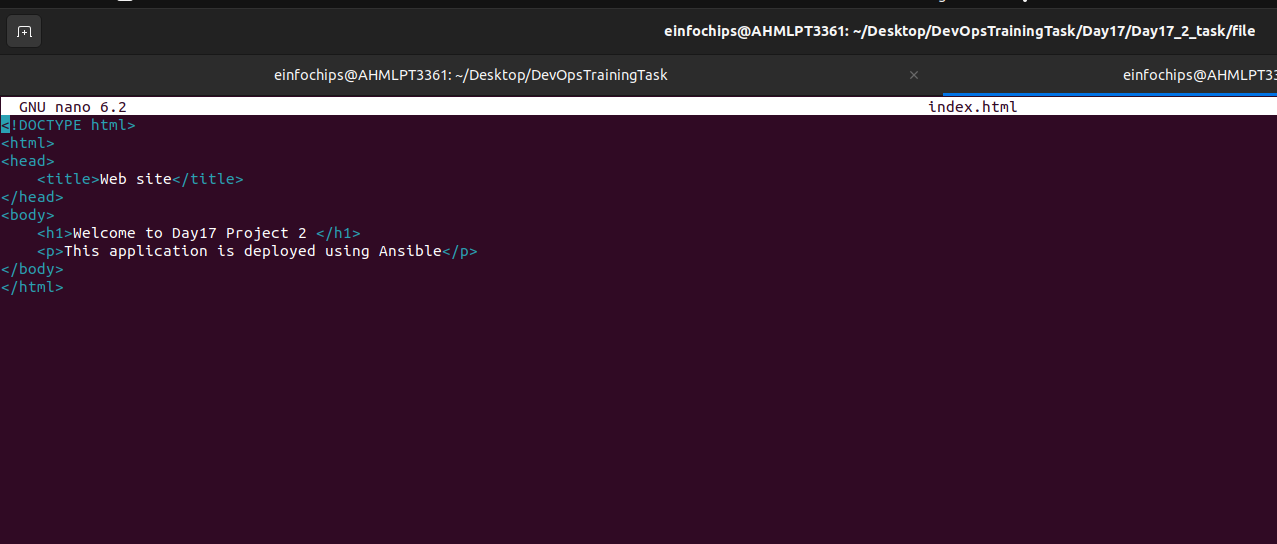


1. **Jinja2 Template**
   * **Filename**: templates/app\_config.php.j2
   * **Content**: Defines a configuration file for the web application that includes placeholders for dynamic values such as database connection details.



**4.A pplication Files**

* + **Filename**: files/index.html (or equivalent application files)
  + **Content**: Static or basic dynamic content served by the web application.



Output:

