📘 PROJECT REPORT- mainly errors we delt with

Automated WordPress Deployment on AWS EC2 using Ansible (RHEL 9/10, Nginx, PHP-FPM, MariaDB, SSL)

---

1. Project Overview

This project demonstrates the **end-to-end automation** of a production-ready WordPress website on **AWS EC2** using **Ansible**.

The deployment was designed to be:

- Secure (SELinux enforced, HTTPS enabled)
- Repeatable (Ansible roles & idempotent playbooks)
- Production-grade (Nginx + PHP-FPM + MariaDB)
- RHEL-compatible (no snap/certbot dependency)

The project intentionally started with **manual setup** to understand real-world Linux problems, then transitioned into **full automation using Ansible**.

---

2. Infrastructure Details

Component	Value
Cloud Provider	AWS EC2
Region	ap-south-1 (Mumbai)
OS	RHEL 9 / RHEL 10
Web Server	Nginx

PHP	PHP 8.x (PHP-FPM)
Database	MariaDB 10.x
CMS	WordPress
SSL	Let’s Encrypt (acme.sh)
Automation Tool	Ansible
Domain	`logstellthetruth.site`
DNS Provider	GoDaddy

---

3. Architecture Overview

```
User Browser
   ↓ HTTPS (443)
Nginx
   ↓ FastCGI (Unix Socket)
PHP-FPM
   ↓ TCP (localhost)
MariaDB
```

- WordPress files hosted under `/home/ec2-user/wordpress/public`
- SELinux **enabled and enforced**
- SSL certificates auto-issued and auto-renewed

---

4. Ansible Project Structure

```
/home/anandu/
├── ansible.cfg
├── inventory/
│   └── hosts.ini
├── group_vars/
```

```
│   └── web.yml
├── playbooks/
│   ├── phase5-mariadb.yml
│   ├── phase6-wordpress-files.yml
│   ├── phase7-nginx-vhost.yml
│   ├── phase8-wp-config.yml
│   └── phase9-ssl.yml
├── roles/
│   ├── nginx/
│   │   ├── tasks/
│   │   │   ├── main.yml
│   │   │   └── ssl.yml
│   │   ├── handlers/
│   │   │   └── main.yml
│   │   └── templates/
│   │       ├── wordpress.conf.j2
│   │       └── nginx-ssl.conf.j2
│   ├── wordpress/
│   │   ├── tasks/
│   │   │   └── main.yml
│   │   └── templates/
│   │       └── wp-config.php.j2
│   └── ssl/
│       └── tasks/
│           └── main.yml
```

---

5. Phase-wise Implementation & Commands

PHASE 1 — Ansible Control Node Setup

Installed Ansible on VMware RHEL

```
sudo dnf install ansible-core -y
ansible --version
```

PHASE 2 — Inventory & Connectivity

Inventory File

`inventory/hosts.ini`

```
[web]
wordpress ansible_host=13.235.99.251 ansible_user=ec2-user
```

SSH Key Setup

```
scp wordpress-key.pem anandu@192.168.133.128:/home/anandu/.ssh/
chmod 400 ~/.ssh/wordpress-key.pem
```

Connectivity Test

```
ansible -i inventory/hosts.ini web -m ping
```

✅ Result: `pong`

PHASE 3 — Nginx & PHP-FPM Setup (Automated)

Installed Packages

```
sudo dnf install nginx php php-fpm php-mysqlnd -y
sudo systemctl enable --now nginx php-fpm
```

PHP-FPM Socket Verification

```
ls -l /run/php-fpm/www.sock
```

Observed Issue (RHEL 10):

```
srw-rw----+ root root /run/php-fpm/www.sock
```

Fix

Configured PHP-FPM to run as `nginx` user.

PHASE 4 — MariaDB Automation

Initial Error

```
ERROR! couldn't resolve module 'mysql_user'
```

Root Cause

MySQL modules are part of `community.mysql` collection.

Fix

```
ansible-galaxy collection install community.mysql
```

Database Creation Commands (Executed via Ansible)

```
CREATE DATABASE wordpress_db;
CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'StrongPassword@123';
GRANT ALL PRIVILEGES ON wordpress_db.* TO 'wp_user'@'localhost';
FLUSH PRIVILEGES;
```

PHASE 5 — WordPress Files Deployment

Initial Error

```
HTTPSConnection.__init__() got an unexpected keyword argument
'cert_file'
```

Root Cause

Python SSL incompatibility with Ansible `get_url` on RHEL.

Fix

Switched to `curl`.

```
- name: Download WordPress using curl
  command: curl -o /tmp/wordpress.tar.gz
https://wordpress.org/latest.tar.gz
```

PHASE 6 — SELinux & Permissions (CRITICAL PHASE)

Error Observed

```
File not found
Primary script unknown
stat() failed (13: Permission denied)
```

Root Cause

- WordPress hosted inside `/home`
- SELinux + Linux DAC blocking traversal

Fixes Applied

```
sudo setsebool -P httpd_enable_homedirs on
sudo chmod o+x /home
sudo chmod o+x /home/ec2-user
sudo chcon -R -t httpd_sys_content_t /home/ec2-user/wordpress
```

Verification

```
namei -l /home/ec2-user/wordpress/public
```

PHASE 7 — Nginx Virtual Host Automation

HTTP Config Template

`roles/nginx/templates/wordpress.conf.j2`

```
server {
    listen 80;
    root /home/ec2-user/wordpress/public;
    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include fastcgi.conf;
        fastcgi_pass unix:/run/php-fpm/www.sock;
    }
}
```

PHASE 8 — wp-config.php Automation

Error

```
'wp_db_name' is undefined
```

Root Cause

`group_vars` not explicitly loaded.

Fix

```
vars_files:
  - ../group_vars/web.yml
```

wp-config Template

`roles/wordpress/templates/wp-config.php.j2`

```
define('DB_NAME', '{{ wp_db_name }}');
define('DB_USER', '{{ wp_db_user }}');
define('DB_PASSWORD', '{{ wp_db_password }}');
define('DB_HOST', '{{ wp_db_host }}');
{{ wp_salts }}
```

Result

WordPress setup page loads **without DB input**

PHASE 9 — SSL Automation (Let's Encrypt)

Initial Failure

```
no valid A records found
```

Root Cause

DNS not pointing to EC2.

DNS Fix (GoDaddy)

Type	Host	Value
A	@	13.235.99.251
CNAME	www	logstellthetruth.site

SSL Issue #2

```
ERR_CONNECTION_REFUSED
```

Root Cause

SSL vhost not deployed.

Fix

Correct role/template separation.

Final SSL Config

roles/nginx/templates/nginx-ssl.conf.j2

```
server {
    listen 443 ssl;
    server_name logstellthetruth.site www.logstellthetruth.site;

    ssl_certificate /etc/nginx/ssl/logstellthetruth.site.crt;
    ssl_certificate_key /etc/nginx/ssl/logstellthetruth.site.key;

    root /home/ec2-user/wordpress/public;
}
```

Final Verification

```
ss -tulpn | grep nginx
```

Result:

```
80 LISTEN
443 LISTEN
```

---

6. Final Outcome

✅ WordPress fully automated
✅ HTTPS enabled with auto-renew
✅ SELinux enforced
✅ No manual server changes
✅ Repeatable deployment

---

7. Key Learnings (Interview-Critical)

- SELinux **must be handled**, not disabled
- `/home` hosting requires **execute permissions**

- DNS is mandatory for SSL
- Ansible roles are **isolated**
- Logs are more important than guesses

---

8. Conclusion

This project demonstrates **real-world DevOps engineering**, not tutorial automation.
It involved **debugging, design decisions, security hardening, and clean automation**.

The final system is production-ready, secure, and fully automated using Ansible.