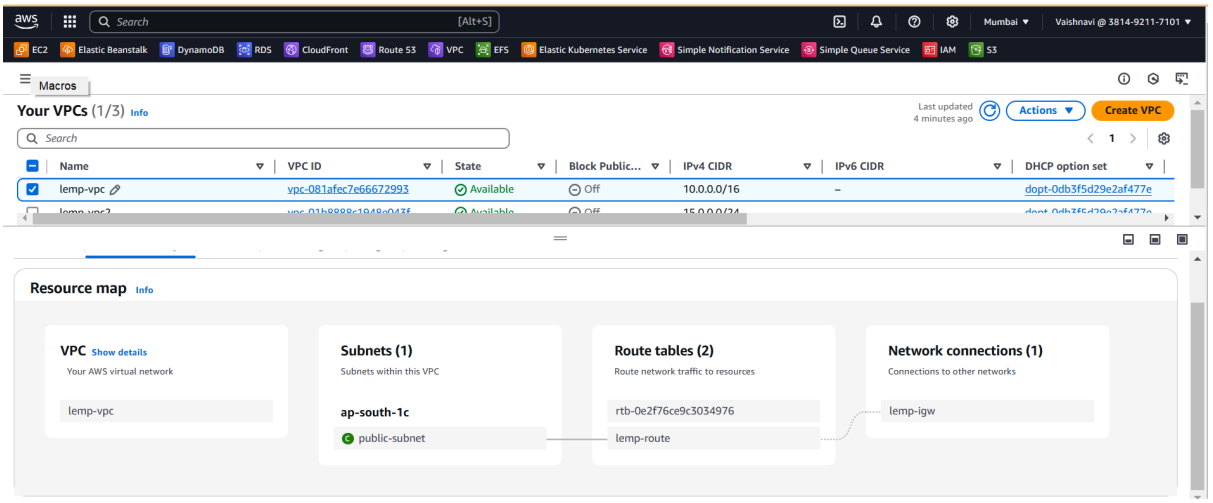
LAMP Server Setup

This document provides a detailed step-by-step guide to set up a LAMP (Linux, Apache, MySQL, PHP) server and deploy a WordPress website integrated with GitHub Actions for continuous deployment.

---

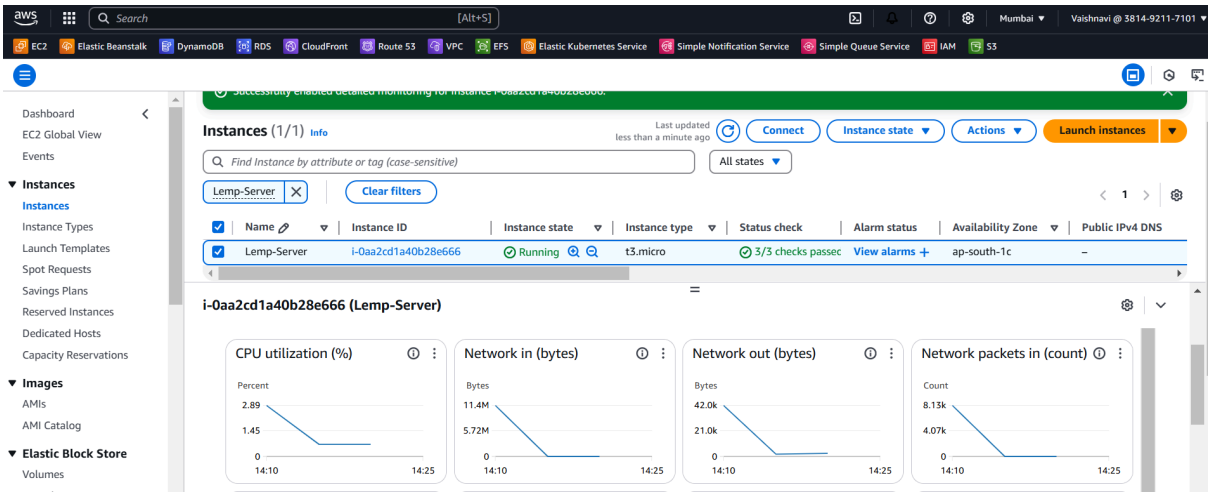
1. Create a VPC and Public Subnet

1. Create a new VPC.
2. Create a public subnet within the VPC.
3. Attach an Internet Gateway (IGW) to the VPC.
4. Configure a route table:
    - Add a route for the IGW.
    - Associate the subnet with the route table.



2. Launch EC2 Instance

1. Launch an EC2 instance within the newly created VPC.
2. Enable detailed monitoring of CloudWatch for instance.



---

3. SSH into the Remote Server

Steps to connect:

1. Obtain the private key associated with your EC2 instance. Use the following command to connect to the server:

```
ssh -i "lemp.pem" ubuntu@3.110.100.219
```

2. Once connected, execute the following commands:

```
apt-get update
apt-get upgrade
```

---

4. Install Nginx

Nginx is a lightweight web server required to serve web pages. Execute the following commands:

```
apt-get install nginx
systemctl enable nginx
systemctl start nginx
systemctl status nginx
ufw app list
ufw allow 'Nginx HTTP'
ufw allow 'Nginx HTTPS'
ufw allow ssh
ufw status
ufw enable
systemctl restart nginx
```

---

5. Install MySQL

MySQL is a database server required for managing WordPress data. Execute the following commands:

```
apt install mysql-server
apt-get install mysql-server
mysql_secure_installation
systemctl enable mysql
mysql -u root -p
```

Configure MySQL:

Once inside the MySQL prompt, execute the following SQL commands to create a database and user:

```
CREATE DATABASE wordpress;
CREATE USER 'wp_user'@localhost IDENTIFIED BY 'Vaishnavi@23';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wp_user'@localhost;
FLUSH PRIVILEGES;
exit;
```

---

6. Install PHP

PHP is required for executing WordPress scripts. Execute the following commands:

```
apt install php-fpm php-mysql php php-curl php-gd php-intl php-zip php-mysqli
add-apt-repository ppa:ondrej/php
apt-get install php7.4-fpm
systemctl enable php7.4-fpm
systemctl start php7.4-fpm
```

---

7. Download and Configure WordPress

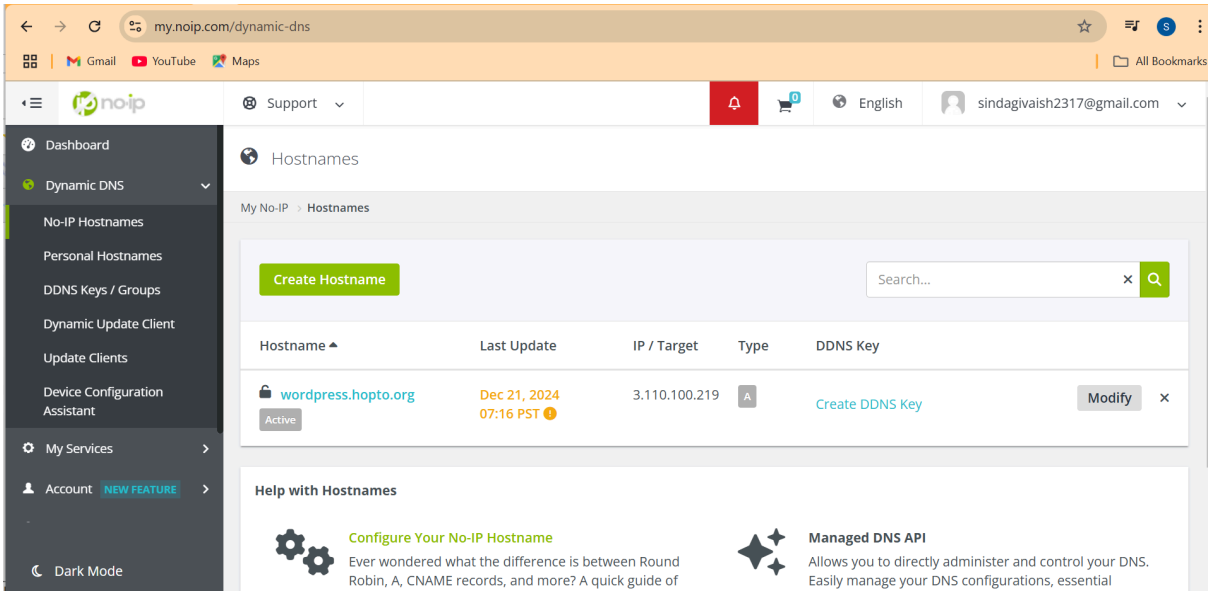
Download WordPress and configure it to connect to your database:

```
cd /var/www/
wget https://wordpress.org/latest.tar.gz
tar -xzvf latest.tar.gz
chown -R www-data:www-data /var/www/html/wordpress
chown -R www-data:www-data /var/www/wordpress
chmod -R 755 /var/www/wordpress
cp /var/www/wordpress/wp-config-sample.php
/var/www/wordpress/wp-config.php
vi /var/www/wordpress/wp-config.php  # Add database details
```

---

8. Set Up Domain Name

1. Create a domain name using No-IP.



---

9. Install SSL Certificate

To secure your website, install an SSL certificate:

```
apt install certbot python3-certbot-nginx -y
certbot --nginx -d http://wordpress.hopto.org/ -d http://wordpress.hopto.org/
certbot --nginx -d wordpress.hopto.org -d wordpress.hopto.org
certbot install --cert-name wordpress.hopto.org
mv /etc/nginx/sites-enabled/default /etc/nginx/sites-available/wordpress
vi /etc/nginx/sites-available/wordpress
```

```
server {
server_name wordpress.hopto.org wordpress.hopto.org;

    root /var/www/wordpress;
    index index.php index.html index.htm;

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

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
```

```
    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/wordpress.hopto.org/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/wordpress.hopto.org/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}


server {
    if ($host = wordpress.hopto.org) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name wordpress.hopto.org wordpress.hopto.org;
    return 404; # managed by Certbot


}
```

Make symbolic link in sites-enabled and restart nginx to reflect changes

```
ln -s /etc/nginx/sites-available/wordpress /etc/nginx/sites-enabled/
nginx -t
systemctl restart nginx
systemctl reload nginx
```

Verify changes by browsing domain name i.e https://wordpress.hopto.org/wp-admin/install.php

---

10. Nginx Configuration Changes

Optimize Nginx server configuration, enable caching and gzip compression:

```
worker_connections 1024;
client_max_body_size 10m;

http {
    gzip on;
    gzip_vary on;
    gzip_comp_level 6;
```

```
    gzip_min_length 1000;
    gzip_types text/plain text/css application/javascript
application/json application/xml application/xml+rss text/javascript;

    fastcgi_cache_path /var/cache/nginx levels=1:2 keys_zone=MYCACHE:10m
inactive=60m;

    server {
        listen 80;
        server_name example.com;

        location / {
            try_files $uri $uri/ =404;
        }

        # Static file caching
        location ~*
\.(jpg|jpeg|png|gif|css|js|ico|woff|woff2|svg|ttf|eot|otf|webp)$ {
            expires 30d;
            add_header Cache-Control "public, no-transform";
        }

        # Enable fastcgi caching
        location ~ \.php$ {
            fastcgi_cache MYCACHE;
            fastcgi_cache_valid 200 60m;
            fastcgi_cache_min_uses 1;
            fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
            include fastcgi_params;
        }
    }
}
```

11. Nginx configuration changes

optimize Nginx server configuration, efficient caching and gzip compression and understanding of performance optimization techniques.

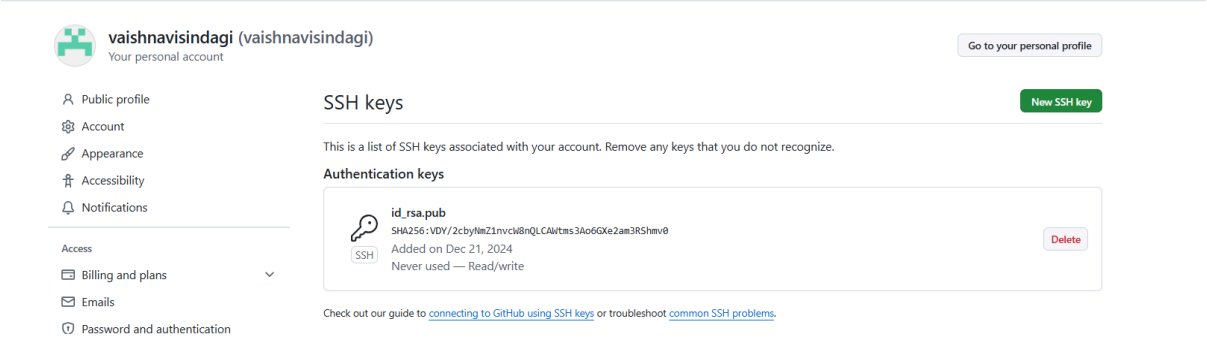
```
worker_processes auto;
worker_connections 1024;
client_max_body_size 10m;

http {
    gzip on;
    gzip_vary on;
    gzip_comp_level 6;
```

```
    gzip_min_length 1000;
    gzip_types text/plain text/css application/javascript
application/json application/xml application/xml+rss text/javascript;

    fastcgi_cache_path /var/cache/nginx levels=1:2 keys_zone=MYCACHE:10m
inactive=60m;

    server {
        listen 80;
        server_name example.com;

        location / {
            try_files $uri $uri/ =404;
        }

        # Static file caching
        location ~*
\.(jpg|jpeg|png|gif|css|js|ico|woff|woff2|svg|ttf|eot|otf|webp)$ {
            expires 30d;
            add_header Cache-Control "public, no-transform";
        }

        # Enable fastcgi caching
        location ~ \.php$ {
            fastcgi_cache MYCACHE;
            fastcgi_cache_valid 200 60m;
            fastcgi_cache_min_uses 1;
            fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
            include fastcgi_params;
        }
    }
}
```

12. Set Up Git Authentication

To enable deployment from GitHub, set up SSH authentication:

```
cd /var/www/wordpress
ssh-keygen -t rsa -b 4096 -C "sindagivaish2317@gmail.com"
cat ~/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
less /root/.ssh/id_rsa.pub  # Add public key in GitHub
```



---

13. Install Git and Configure Repository

To manage the WordPress files using Git, execute the following commands:

```
apt-get install git
cd /var/www/wordpress
git config --global user.email "sindagivaish2317@gmail.com"
git config --global user.name "vaishnavisindagi"
git init
git remote set-url origin https://github.com/vaishnavisindagi/wp-deployment.git
git remote add origin https://github.com/vaishnavisindagi/wp-deployment.git
git remote -v
git branch main
git add .
git commit -m "wordpress-deployment"
git push -u origin main
```

---

12. Configure GitHub Actions for Deployment

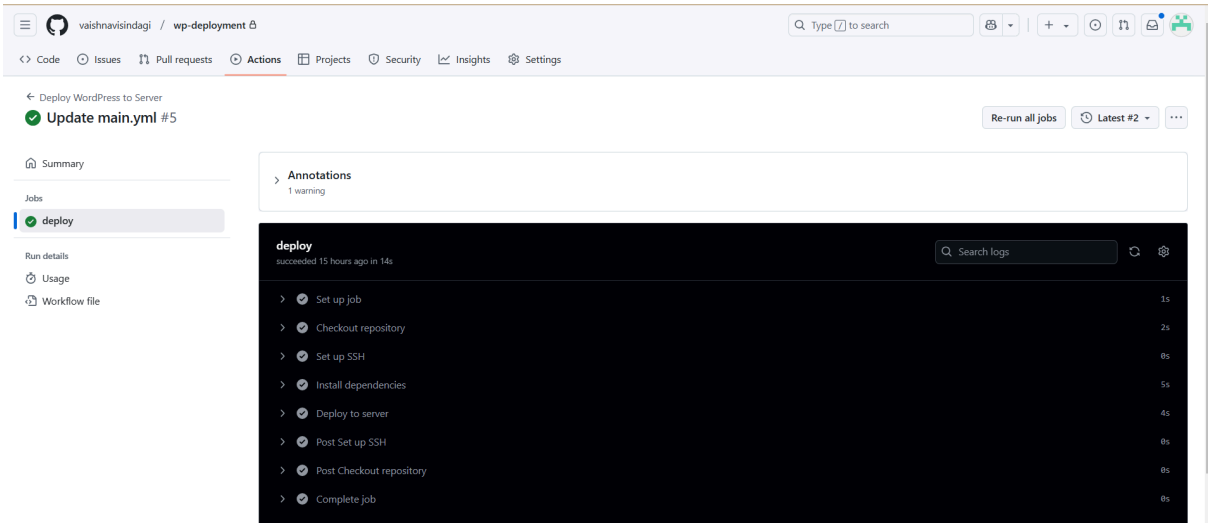
Steps:

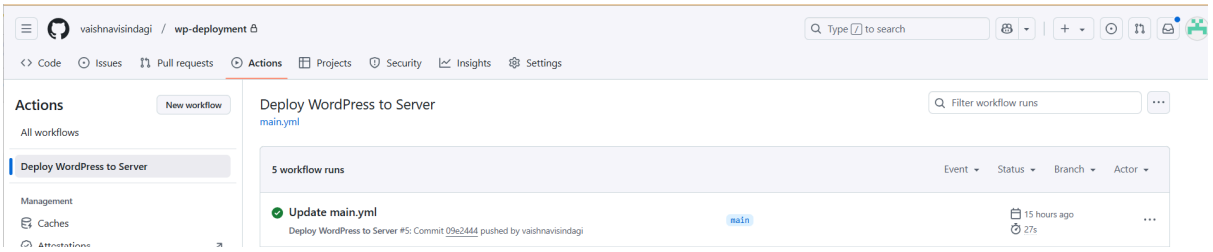
1. Go to **GitHub Repository > Actions > Create Workflow**.
2. Select **Set up a workflow yourself** and create a YAML file with the following content:

```
name: Deploy WordPress to Server

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest
```

```yaml
steps:
  - name: Checkout repository
    uses: actions/checkout@v3

  - name: Set up SSH
    uses: webfactory/ssh-agent@v0.5.3
    with:
      ssh-private-key: ${{ secrets.KEY }}

  - name: Install dependencies
    run: |
      sudo apt-get update

  - name: Deploy to server
    run: |
      ssh -o StrictHostKeyChecking=no root@3.110.100.219 << 'EOF'
        cd /var/www/wordpress
        git pull origin main
        sudo systemctl restart nginx
      EOF
```

3. Navigate to **Settings > Secrets and Variables > Actions**.
4. Add the SSH private key under the **Secrets** section.





WordPress installation URL: https://wordpress.hopto.org/wp-admin/install.php

By following these steps, i have successfully set up a LAMP server, deployed WordPress, and configured GitHub Actions for automated deployment.