This document walks through steps for automated deployment of WordPress application on ECS using open source tools.

Assumptions:

- Code is hosted on GitHub
- Container repository is hosted on DockerHub
- CICD server is Jenkins on hosted on EC2 with required plugins and tools like ecs, docker cli, terraform

Configuration files:

Below files are required to be created and kept in the root directory of the code repository.

- Jenkinsfile - This file contains stage/steps declarations for checking out code, Building docker image and pushing it to registry, updating task definition file with new docker image id and running terraform apply to update the ecs service.
- Dockerfile - This file contains instructions for creating docker image out of code

GitHub branching strategy:

- master/main branch – This is for production deployment ONLY
- development branch – This is for development purposes and separate CICD job configured for this branch to deploy on a sandbox environment. Code from this branch ONLY be merged with master/main branch.
- feature branch/es - This is for individual feature/tasks development. Always created from the latest development branch. After work finishes PR should be raised with the development branch. Once merged with development then delete the branch.

Pipeline for development

Pipeline job will have following stages:

- Code checkout from development/feature branch using poll SCM (GitHub webhooks can also be configured, but for that Jenkins need to be public accessible)
- Building docker image from code
- Pushing docker image to DockerHub
- Checkout terraform code
- Update new docker image id in task definition file
- Deploy the build to the SANDBOX environment. Run the terraform plan/apply command to update ECS service
- Running vulnerability scanner/testing to scan vulnerabilities and exploits in WordPress core, plugins and themes
- Sendout the email to developers if there are any errors during build deployment

Note: Docker image tag can be <branch name> + build job number

Pipeline for production deployment

Pipeline job will have following stages:

- Code checkout from master using poll SCM (GitHub webhooks can also be configured, but for that Jenkins need to be public accessible)
- Building docker image from code
- Pushing docker image to DockerHub
- Checkout terraform code
- Update new docker image id in task definition file
- Deploy the build to the TEST environment. Run the terraform plan/apply command to update ECS service (This will keep the IaC code updated and detect if any unwillingly changes made to infrastructure)
- Running vulnerability scanner/testing to scan vulnerabilities and exploits in WordPress core, plugins and themes
- Execute the automated Load/Performance test against the environment. Post the result on the dashboard.
- Upon passing tests, deploy to the production environment
- Send emails to development and devops team if any errors

Note: For docker image tagging we can use GitHub tags. Tags should be created when merging development branch with master/main branch.

Security for Deployment server:

- Access to Jenkins EC2 instances should be restricted, only required ports 22, 8080 should be allowed in SG group and only from trusted location and AWS environment.
- Access to Jenkins UI should be restricted to specific IP's
- Jenkins instances should have IAM role to access required AWS resources with strict permissions
- GitHub and Jenkins communication should be configured with service account and this account access will be restricted to group of responsible people