
DevOps Shack

# Ansible Playbook That Performs A Full Stack CI/CD Pipeline

Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps

To create an Ansible playbook that performs a full stack CI/CD pipeline with the described stages, you need to structure the playbook into tasks that represent each stage.

You'll need to install the required dependencies on your Ansible control node and target hosts, including Git, Node.js, Maven, SonarQube, Trivy, Docker, Nexus, Kubernetes, and OWASP ZAP.

Here's the playbook:

**ci_cd_pipeline.yml**

```
---
- name: Full Stack CI/CD Pipeline
  hosts: localhost
  vars:
    git_repo_url: "https://github.com/your-repo.git"
    git_branch: "main"
    project_dir: "/path/to/project"
    sonar_host_url: "http://sonarqube:9000"
    sonar_project_key: "your_project_key"
    sonar_login: "your_sonar_token"
    nexus_url: "http://nexus:8081"
    nexus_repo: "your-repo"
    docker_image_name: "your-docker-image"
    docker_registry: "your-docker-registry"
    k8s_namespace: "your-namespace"
    k8s_deployment_file: "path/to/deployment.yaml"
    zap_host: "zap"
    zap_port: 8080
    notification_email: "team@example.com"
    smtp_server: "smtp.example.com"
    smtp_port: 587
```

```
    smtp_user: "smtp_user"
    smtp_password: "smtp_password"

  tasks:
    - name: Git Checkout
      ansible.builtin.git:
        repo: "{{ git_repo_url }}"
        dest: "{{ project_dir }}"
        version: "{{ git_branch }}"

    - name: Install Node.js Dependencies
      ansible.builtin.shell: |
        cd {{ project_dir }}
        npm install
      when: ansible_os_family == 'Debian'

    - name: Install Maven Dependencies
      ansible.builtin.maven_artifact:
        group_id: "com.example"
        artifact_id: "myapp"
        dest: "{{ project_dir }}"
        state: present
      when: ansible_os_family == 'RedHat'

    - name: Run Test Cases
      ansible.builtin.shell: |
        cd {{ project_dir }}
        npm test

    - name: SonarQube Analysis
      ansible.builtin.shell: |
        cd {{ project_dir }}
        sonar-scanner -Dsonar.projectKey={{ sonar_project_key }} -
Dsonar.sources=. -Dsonar.host.url={{ sonar_host_url }} -Dsonar.login={{
sonar_login }}

    - name: Trivy FS Scan
      community.general.trivy:
        path: "{{ project_dir }}"
        severity: HIGH,CRITICAL

    - name: Build Application
      ansible.builtin.shell: |
        cd {{ project_dir }}
        mvn clean package

    - name: Push Artifact to Nexus
      nexus3_upload:
        nexus_url: "{{ nexus_url }}"
        username: "nexus_user"
        password: "nexus_password"
        artifact: "{{ project_dir }}/target/myapp-1.0.jar"
        repository: "{{ nexus_repo }}"
        group_id: "com.example"
        artifact_id: "myapp"
        version: "1.0"
        packaging: "jar"

    - name: Build & Tag Docker Image
      community.docker.docker_image:
        name: "{{ docker_image_name }}"
```

```
    path: "{{ project_dir }}"
    tag: "latest"

- name: Scan Docker Image Using Trivy
  community.general.trivy:
    image: "{{ docker_image_name }}:latest"
    severity: HIGH,CRITICAL

- name: Push Docker Image
  community.docker.docker_image:
    name: "{{ docker_image_name }}"
    repository: "{{ docker_registry }}/{{ docker_image_name }}"
    tag: "latest"
    push: true

- name: Deploy Application to Kubernetes
  community.kubernetes.k8s:
    kubeconfig: "/path/to/kubeconfig"
    namespace: "{{ k8s_namespace }}"
    definition: "{{ lookup('file', k8s_deployment_file) }}"

- name: Perform Penetration Testing Using OWASP ZAP
  owasp_zap:
    zap_host: "{{ zap_host }}"
    zap_port: "{{ zap_port }}"
    target: "http://your-app-url"
    config:
      spider:
        maxDepth: 5
        threadCount: 5
      scan:
        recurse: true

- name: Send Mail Notifications
  mail:
    host: "{{ smtp_server }}"
    port: "{{ smtp_port }}"
    username: "{{ smtp_user }}"
    password: "{{ smtp_password }}"
    to: "{{ notification_email }}"
    subject: "CI/CD Pipeline Notification"
    body: "The CI/CD pipeline has completed successfully."
```

## Notes:

1. **Variables**: Adjust the variables to match your environment and project specifics.
2. **Roles and Tasks**: The tasks for each step (e.g., `ansible.builtin.git`, `community.general.trivy`, `community.docker.docker_image`, etc.) must be installed on your Ansible control node. You can install required collections using `ansible-galaxy collection install <collection_name>`.
3. **OWASP ZAP**: This requires the `owasp_zap` module, which may need custom implementation or additional setup if not directly available in Ansible.
4. **Email Notification**: This uses the `mail` module for sending emails. Ensure your SMTP server and credentials are correctly configured.

# Each Stage in Detail

## 1. Playbook Header

```
- name: Full Stack CI/CD Pipeline
  hosts: localhost
  vars:
    git_repo_url: "https://github.com/your-repo.git"
    git_branch: "main"
    project_dir: "/path/to/project"
    sonar_host_url: "http://sonarqube:9000"
    sonar_project_key: "your_project_key"
    sonar_login: "your_sonar_token"
    nexus_url: "http://nexus:8081"
    nexus_repo: "your-repo"
    docker_image_name: "your-docker-image"
    docker_registry: "your-docker-registry"
    k8s_namespace: "your-namespace"
    k8s_deployment_file: "path/to/deployment.yaml"
    zap_host: "zap"
    zap_port: 8080
    notification_email: "team@example.com"
    smtp_server: "smtp.example.com"
    smtp_port: 587
    smtp_user: "smtp_user"
    smtp_password: "smtp_password"
```

**Explanation:** This section sets up the playbook metadata and variables:

- `name`: The name of the playbook.
- `hosts`: Specifies that the playbook will run on the localhost.
- `vars`: Declares variables used throughout the playbook, such as Git repository details, SonarQube settings, Nexus repository details, Docker image information, Kubernetes namespace, OWASP ZAP settings, and email notification details.

## 2. Git Checkout

```
- name: Git Checkout
  ansible.builtin.git:
    repo: "{{ git_repo_url }}"
    dest: "{{ project_dir }}"
    version: "{{ git_branch }}"
```

**Explanation:** This task clones the specified Git repository to the defined project directory and checks out the specified branch.

## 3. Install Node.js Dependencies

```
- name: Install Node.js Dependencies
  ansible.builtin.shell: |
    cd {{ project_dir }}
    npm install
  when: ansible_os_family == 'Debian'
```

**Explanation:** This task installs Node.js dependencies using `npm install` if the operating system family is Debian. It ensures the project has all necessary Node.js packages.

## 4. Install Maven Dependencies

```
- name: Install Maven Dependencies
  ansible.builtin.maven_artifact:
    group_id: "com.example"
    artifact_id: "myapp"
    dest: "{{ project_dir }}"
    state: present
  when: ansible_os_family == 'RedHat'
```

**Explanation:** This task installs Maven dependencies for the project if the operating system family is RedHat. It ensures that the required Maven artifacts are present in the project directory.

## 5. Run Test Cases

```
- name: Run Test Cases
  ansible.builtin.shell: |
    cd {{ project_dir }}
    npm test
```

**Explanation:** This task runs the project's test cases using `npm test`, ensuring the code passes all tests before proceeding.

## 6. SonarQube Analysis

```
- name: SonarQube Analysis
  ansible.builtin.shell: |
    cd {{ project_dir }}
    sonar-scanner -Dsonar.projectKey={{ sonar_project_key }} -
Dsonar.sources=. -Dsonar.host.url={{ sonar_host_url }} -Dsonar.login={{
sonar_login }}
```

**Explanation:** This task performs static code analysis using SonarQube to ensure code quality and identify potential issues.

## 7. Trivy FS Scan

```
- name: Trivy FS Scan
  community.general.trivy:
    path: "{{ project_dir }}"
    severity: HIGH,CRITICAL
```

**Explanation:** This task uses Trivy to scan the filesystem of the project directory for vulnerabilities of high and critical severity.

## 8. Build Application

```
- name: Build Application
  ansible.builtin.shell: |
    cd {{ project_dir }}
    mvn clean package
```

**Explanation:** This task builds the application using Maven, packaging the application into a deployable format (e.g., a JAR file).

## 9. Push Artifact to Nexus

```
- name: Push Artifact to Nexus
  nexus3_upload:
    nexus_url: "{{ nexus_url }}"
    username: "nexus_user"
    password: "nexus_password"
    artifact: "{{ project_dir }}/target/myapp-1.0.jar"
    repository: "{{ nexus_repo }}"
    group_id: "com.example"
    artifact_id: "myapp"
    version: "1.0"
    packaging: "jar"
```

**Explanation:** This task uploads the built artifact to a Nexus repository, making it available for distribution.

## 10. Build & Tag Docker Image

```
- name: Build & Tag Docker Image
  community.docker.docker_image:
    name: "{{ docker_image_name }}"
    path: "{{ project_dir }}"
    tag: "latest"
```

**Explanation:** This task builds a Docker image from the project directory and tags it as "latest".

## 11. Scan Docker Image Using Trivy

```
- name: Scan Docker Image Using Trivy
  community.general.trivy:
    image: "{{ docker_image_name }}:latest"
    severity: HIGH,CRITICAL
```

**Explanation:** This task scans the Docker image for vulnerabilities of high and critical severity using Trivy.

## 12. Push Docker Image

```
- name: Push Docker Image
  community.docker.docker_image:
    name: "{{ docker_image_name }}"
    repository: "{{ docker_registry }}/{{ docker_image_name }}"
    tag: "latest"
    push: true
```

**Explanation:** This task pushes the Docker image to a specified Docker registry.

## 13. Deploy Application to Kubernetes

```
- name: Deploy Application to Kubernetes
  community.kubernetes.k8s:
    kubeconfig: "/path/to/kubeconfig"
    namespace: "{{ k8s_namespace }}"
    definition: "{{ lookup('file', k8s_deployment_file) }}"
```

**Explanation:** This task deploys the application to a Kubernetes cluster using the provided deployment YAML file.

## 14. Perform Penetration Testing Using OWASP ZAP

```
- name: Perform Penetration Testing Using OWASP ZAP
  owasp_zap:
    zap_host: "{{ zap_host }}"
    zap_port: "{{ zap_port }}"
    target: "http://your-app-url"
    config:
      spider:
        maxDepth: 5
        threadCount: 5
      scan:
        recurse: true
```

**Explanation:** This task performs automated penetration testing using OWASP ZAP to identify security vulnerabilities in the deployed application.

## 15. Send Mail Notifications

```
- name: Send Mail Notifications
  mail:
    host: "{{ smtp_server }}"
    port: "{{ smtp_port }}"
    username: "{{ smtp_user }}"
    password: "{{ smtp_password }}"
    to: "{{ notification_email }}"
    subject: "CI/CD Pipeline Notification"
    body: "The CI/CD pipeline has completed successfully."
```

**Explanation:** This task sends an email notification to the specified email address, informing the team that the CI/CD pipeline has completed successfully.

Each section of this playbook automates a step in the CI/CD process, ensuring the continuous integration and delivery of the application from source code management to deployment and testing.