**CI/CD Pipeline Workflow using Jenkins, SonarQube, and Docker**

This pipeline automates code integration, testing, and deployment using three EC2 instances: **Jenkins**, **SonarQube**, and **Docker**. Below is a step-by-step breakdown of the architecture and workflow.

**1. EC2 Instance Setup**

Three EC2 instances are created with appropriate firewall rules:

* **Jenkins** (Port **8080**) – Manages the CI/CD pipeline.
* **SonarQube** (Port **9000**) – Performs code quality analysis.
* **Docker** (Port **80**) – Hosts and deploys the final application.

Each instance has security group rules configured to allow traffic on required ports.

**2. Jenkins Installation & Configuration**

1. Install **Jenkins** on its EC2 instance.
2. Connect Jenkins to **GitHub** using a **webhook** to trigger builds when code changes.
3. Create a **Freestyle Project** (A10001pipeline) that automates the pipeline execution.
4. Install required Jenkins plugins:
   * **SonarQube Scanner** (for code analysis)
   * **Docker Plugin** (for image builds)
   * **SSH Plugins** (for remote deployments)

**3. SonarQube Installation & Integration**

1. Install and start **SonarQube** on its EC2 instance.
2. Create a project in SonarQube and generate a **security token**.
3. Configure **Jenkins** to use SonarQube:
   * Add the SonarQube **server URL** in Jenkins settings.
   * Store the **security token** as a secret credential in Jenkins.
   * Configure the pipeline to run SonarQube analysis on code updates.

**4. Docker Installation & Integration**

1. Install **Docker** on both the **Jenkins** and **Docker EC2 instances**.
2. Enable Docker to run as a **service**.
3. Configure Jenkins to use Docker:
   * Define Docker as a tool in Jenkins settings.
   * Enable Docker build and push in the pipeline.
4. Add a **deployment step** in Jenkins to:
   * Pull the latest Docker image.
   * Stop and remove the old container.
   * Start a new container using the latest image.

**5. CI/CD Pipeline Flow**

**Step-by-Step Execution**

1. **Code Push to GitHub**:
   * A developer makes changes to the index.html file and commits it to GitHub.
   * The **GitHub webhook** notifies Jenkins of the update.
2. **Jenkins Triggers Build**:
   * Jenkins pulls the latest code from GitHub.
   * It triggers **SonarQube** for code quality analysis.
3. **SonarQube Performs Code Analysis**:
   * The code is scanned for vulnerabilities and quality issues.
   * Results are displayed on the SonarQube dashboard.
4. **Docker Builds and Pushes Image**:
   * Jenkins builds a **Docker image** with the updated code.
   * The image is pushed to **Docker Hub**.
5. **Deployment to Docker EC2 Instance**:
   * The Docker EC2 instance pulls the latest image.
   * The old container is stopped and removed.
   * A new container is started, hosting the latest version of the application.
6. **Final Website Update**:
   * The Docker EC2 **public IP** is accessible via a web browser.
   * Any new changes made to index.html in GitHub automatically trigger the pipeline.
   * The website updates seamlessly.





