Task: Setting Up a Multi-Server Java Application Build, Store, and Deployment System

Objective:

The objective of this task is for students to set up and configure a multi-server environment for building, storing, and deploying a Java web application. The task is divided into three sections, with each server playing a specific role in the development and deployment pipeline:

1. **Java Build Server**: A server where students will build the project code.

2. **Nexus Repository Server**: A repository to store the artifacts (WAR files) generated by the build process.

3. **Deploy Server**: A server to deploy the final application and make it accessible via a web browser.

The task involves configuring each server for its respective role and ensuring that they can interact with each other seamlessly.

Requirements:

- **GitHub Repository**: Students will use the Java Web Calculator App - https://github.com/mrtechreddy/Java-Web-Calculator-App.git as the source code.

- **Three Servers**: One for building the code, one for hosting the Nexus repository, and one for deploying the application.

---

1. Java Build Server Setup

Task Overview:

This server is where students will clone the project, build it using Maven, and deploy the resulting artifact to the Nexus repository.

Steps:

- **Clone the Project**: Students will begin by cloning the GitHub repository containing the source code of the Java Web Calculator App.

- **Build the Application**: Using Apache Maven, students will build the project, which will generate a `.war` file in the `target/` directory. The `.war` file is the deployable artifact.

- **Configure Maven for Deployment**: Students will need to configure Maven to upload the generated `.war` file to a Nexus repository. They will modify their `pom.xml` and Maven `settings.xml` to include the repository URL and authentication credentials for Nexus.

- **Deploy the Artifact**: Once the build completes successfully, the students will deploy the generated artifact to the Nexus repository. They will do this by running the `mvn deploy` command. The artifact will then be stored in the Nexus repository for future use by the deploy server.

Deliverables:

- Cloned project directory on the build server.

- `.war` file generated in the `target/` directory.

- A successful deployment to the Nexus repository.

---

2. Nexus Repository Server Setup

Task Overview:

The Nexus repository server will be used to store and manage the build artifacts (such as the `.war` file). Students will configure the repository server to host the necessary Maven repositories and ensure that the build server can push artifacts to it.

Steps:

- **Install Nexus Repository Manager**: Students will need to install Nexus Repository Manager on this server. They should ensure that the Nexus service is running and accessible from the build and deploy servers.

- **Configure Repositories**: Students will create a Maven repository in Nexus where they can upload the `.war` files. This will involve creating both a release repository (for production-ready artifacts) and possibly a snapshot repository (for development artifacts).

- **Set Up Authentication**: The repository will require authentication for both uploading and downloading artifacts. Students will configure the repository's authentication settings and update their `settings.xml` files on the build server with the correct credentials.

- **Monitor the Nexus Server**: Students will check that the server is accessible from the build server, and verify that artifacts are being uploaded correctly. They will also need to ensure that the repository has sufficient disk space and is properly maintained.

Deliverables:

- Nexus repository server set up with proper repository configurations.
- Authentication configured for the build and deploy servers.
- Verification that the artifact can be uploaded and stored in the Nexus repository.

---

3. Deploy Server Setup

Task Overview:

This server will host the deployed web application. The students will configure Apache Tomcat on this server to fetch the `.war` file from the Nexus repository and deploy it.

Steps:

- **Install Apache Tomcat**: Students will install and configure Apache Tomcat on the deploy server. Apache Tomcat will act as the servlet container where the `.war` file will be deployed.

- **Configure Maven for Deployment**: The deploy server will need to be able to pull the `.war` file from the Nexus repository. Students will configure Maven or a script to download the `.war` file from the Nexus repository to the deploy server.

- **Automate Artifact Deployment**: Once the `.war` file is downloaded, students will automate the deployment to Apache Tomcat by copying the `.war` file into the Tomcat `webapps/` directory. They will then start Tomcat to deploy the application.

- **Test the Application**: After the deployment, students will test the application by accessing it via a web browser at

`http://<DEPLOY_SERVER_IP>:8080/<artifact-name>`.

Deliverables:

- Apache Tomcat server installed and configured on the deploy server.
- Automation scripts (if used) to pull the `.war` file and deploy it to Tomcat.
- A running instance of the Java Web Calculator app accessible via a web browser.

---

4. End-to-End Testing

Task Overview:

Once the individual servers are set up, students will test the entire workflow. The Java build server should be able to build and deploy the `.war` file to Nexus, and the deploy server should be able to fetch and deploy the `.war` file to Tomcat.

Steps:

- **Trigger Build and Deployment**: On the build server, students will run `mvn clean install` to build the project, followed by `mvn deploy` to push the artifact to Nexus.
- **Verify Artifact in Nexus**: Students will log in to the Nexus web interface and verify that the artifact was uploaded correctly.
- **Deploy to Tomcat**: On the deploy server, students will fetch the `.war` file from Nexus and deploy it to Apache Tomcat.
- **Access the Application**: Finally, students will test the deployed application by navigating to `http://<DEPLOY_SERVER_IP>:8080/<artifact-name>` and ensuring the Java Web Calculator is accessible and functioning as expected.

Deliverables:

- A fully functional Java Web Calculator application running on Tomcat.
- Documentation or a report detailing the steps taken and any issues encountered during setup.