# Session 1

Available plugins
Plugin :
Deploy to container
Manage jenkins
We have to go to plugins
 Deploy tp container
Click on install

## 1️⃣ Copying the WAR File

- You built the project in **Jenkins**, which generated the **EcommerceApp.war** file.
- We copied the WAR file from Jenkins to **Tomcat's webapps/ folder**, where Tomcat looks for applications.

## 2️⃣ Extracting the WAR File

- Normally, Tomcat **automatically extracts** WAR files into a folder when it starts.
- In this case, it **didn't extract it**, so we did it manually using:
  bash
  CopyEdit

```
sudo unzip EcommerceApp.war -d EcommerceApp/
```

- This created a folder **EcommerceApp/** inside webapps/ that contained all the necessary files.

## 3️⃣ Restarting Tomcat

- After extracting the WAR file, Tomcat needed to **reload** the application.
- We restarted Tomcat using:
  bash
  CopyEdit
  sudo ./catalina.sh stop
- sudo ./catalina.sh start
- This forced Tomcat to detect the extracted files and **deploy the app**.

## 4️⃣ Accessing the App

- Once Tomcat deployed the application, we accessed it at:
  arduino
  CopyEdit

http://localhost:8080/EcommerceApp/

- Now the application is **running correctly!** 🎉

## 📌 What You Learned

✅ **How to deploy a Java web app using Tomcat**
✅ **How Tomcat processes WAR files**
✅ **How to troubleshoot deployment issues**
✅ **How to manually extract and deploy a WAR file**
✅ **How to restart Tomcat to apply changes**

## 🔷 Next Steps

Now that your homework is done, here are some things you can explore next:
🔷 **Automate the deployment with Jenkins** → Configure Jenkins to automatically deploy your WAR file to Tomcat.
🔷 **Set up GitHub Webhooks** → So that whenever you push code to GitHub, Jenkins builds and deploys automatically.
🔷 **Use Docker for Deployment** → Instead of manually copying files, deploy your app in a **Docker container** for easier management.