Lab Title: Deployment Tools

Lab Number: 3

Objective

To understand the role of deployment tools in Agile software development and to gain practical experience using tools like GitHub Actions and Docker for automating the deployment process.

Theory

In Agile and DevOps environments, delivering working software frequently is a key goal. Manual deployments are slow, error-prone, and inconsistent. Deployment tools help automate the build, test, and release process, making software delivery faster and more reliable.

Some popular deployment tools include:

1. Jenkins – Used for continuous integration and deployment pipelines.

2. Docker – Packages apps into containers to ensure they run the same in all environments.

3. Kubernetes – Manages and scales containerized applications (works with Docker).

4. GitHub Actions – Automates workflows directly within GitHub.

5. Ansible – Automates configuration and deployment to multiple servers.

Implementation

Using GitHub Actions for Deployment

GitHub Actions allows us to automatically deploy our application when code is pushed to a branch.

Steps:

1. In the GitHub repo, create a folder `.github/workflows/`
2. Inside it, create a file `deploy.yml`

```
name: Deploy Website

on:
  push:
    branches: [ main ]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
```

```
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Deploy
      run: echo "Website Deployed Successfully!"
```

> Now, whenever you push to the `main` branch, this GitHub Action will run automatically.

Using Docker for Deployment

Docker allows you to package your application and all its dependencies into a single container image that runs the same everywhere.

Steps:

1. Create a simple web app (e.g., `app.py` for Python Flask or `index.html` for static site)
2. Create a `Dockerfile` in the project root:

```
# Use Python base image
FROM python:3.10-slim

# Set working directory
WORKDIR /app

# Copy files
COPY . .

# Install dependencies
RUN pip install flask

# Run the app
CMD ["python", "app.py"]
```

3. Build and run your Docker container:

```
docker build -t myapp .
docker run -p 5000:5000 myapp
```

Now your app is running in a container and can be deployed anywhere Docker is supported.

Result

- Successfully created a deployment pipeline using **GitHub Actions**.
- Successfully containerized and ran an application using **Docker**.
- Understood how automation tools reduce manual errors and speed up the release cycle.

Conclusion

Deployment tools like **Docker** and **GitHub Actions** are crucial in Agile and DevOps processes. Docker ensures consistency by packaging apps into containers. GitHub Actions automates deployment workflows. These tools make modern software development faster, more reliable, and easier to manage, especially when code changes are frequent.