To build a complete CI/CD pipeline for an e-commerce front-end (e.g., built using React, Angular, etc.), with a Python back-end and a database, leveraging Jenkins, Docker, and automation tools, you will need to define several components:

1. **Jenkinsfile**: A pipeline configuration that automates the build, test, and deployment processes.
2. **Dockerfile**: To containerize both the front-end and back-end applications.
3. **CI/CD Pipeline**: A fully automated pipeline from code commit to deployment.

**1. CI/CD Workflow Breakdown:**

1. **Checkout Code**: Pull the latest code from the Git repository.
2. **Build Frontend**: Use Docker to build the front-end container.
3. **Build Backend**: Similarly, use Docker to build the back-end container.
4. **Test Frontend**: Run unit tests for the front-end (e.g., using Jest for React).
5. **Test Backend**: Run unit tests for the back-end (e.g., using Pytest for Python).
6. **Push Docker Images**: Push the built Docker images to a Docker registry (Docker Hub, AWS ECR, etc.).
7. **Deploy**: Use docker-compose to deploy the entire stack (frontend, backend, and database) or a similar tool.
8. **Cleanup**: Clean up the local Docker images to free up space.

**2. Additional Configurations for Jenkins:**

* **Install Plugins**:
  + Docker Pipeline Plugin
  + Git Plugin (for SCM checkout)
  + NodeJS Plugin (for frontend if using Node.js)
* **Environment Variables**:
  + You can store credentials, image names, and other variables in Jenkins' environment configurations or as secret variables.

**3. Docker Compose Deployment:**

In your Jenkins pipeline, the **Deploy** stage uses docker-compose to run the application. This is an easy way to manage multi-container setups with dependencies (frontend, backend, and database).

You can also adapt the deployment stage to use **Kubernetes** or **AWS ECS** if you're scaling up.

**Conclusion:**

* **Dockerfiles**: Containerize your front-end and back-end applications for consistent, reproducible deployments.
* **Jenkinsfile**: Automate your build, test, and deployment processes using a CI/CD pipeline.
* **Docker Compose**: Handle multi-container setups locally or in production.
* **CI/CD**: Ensure that your e-commerce application goes through the stages of building, testing, and deploying automatically with every change to the repository.