For deploying Java artifacts, you need a place to store the artifacts. Naturally, it is recommended to use a Maven repository, however it is possible to point Maven to a local filesystem as a repository, mainly for testing things out. The configuration needs to go in your $M2_HOME/settings.xml file. A minimal example that stores the deployed artifacts in /tmp could be the following:

**$M2_HOME/settings.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0 http://maven.apache.org/xsd/settings-1.1.0.xsd">
  <profiles>
    <profile>
      <id>no-repository</id>
      <properties>
        <distributionManagementURL.release>file:///tmp/</distributionManagementURL.release>
        <distributionManagementURL.snapshot>file:///tmp/</distributionManagementURL.snapshot>
        <project.distributionManagement.repository.url>file:///tmp/</project.distributionManagement.repository.
url>
        <project.distributionManagement.snapshotRepository.url>file:///tmp/</project.distributionManagement.
snapshotRepository.url>
      </properties>
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>no-repository</activeProfile>
  </activeProfiles>
</settings>
```

In your module you should compile/install as usual and execute:

```
make clean all install
make deploy
```

The command deploy would call Maven on all JARFILES and COMPONENTS_JARFILES defined in the Makefile, and end up deploying them in the configured repositories (/tmp in this case).

Currently at ALMA we're using an internal repository based on JFrog Artifactory for Java artifacts, which has a free community version.

There are some environment variables which help with the Maven deployment of artifacts:

- ACS_DEPLOY_VERSION: The version (Default: If undefined, the version is deduced from $ALMASW_RELEASE: 2023DEC --> 2023.12. If it has a different format then $ALMASW_RELEASE is used as is for the version) for the artifact to be deployed.
- ACS_DEPLOY_GROUP: The group (almasw) by default to be used for the deployment. The root Makefile defines this variable to be alma.acs. So all artifacts use this group when building ACS (make build + make deploy at repository root level)

Other adjustments can be made through the settings.xml like triggering different profiles (release, testing, development). In our case we have the following variables to enable/disable profiles:

- ACS_MAVEN_PROF: Defines the repository to be used (no-repository, ro-repository, rw-repository, or whatever you prefer)
- ACS_MAVEN_ENV: Defines the stage to be used (develop, testing, release).

These last two variables aren't defined anywhere in ACS and are rather used for convenience in the settings.xml file in your $M2_HOME.

For deploying Python packages, you need a repository, and as far as we've seen it's not trivial to use a local file system for testing. Here you need a .pypirc file in your home. A minimal example would be:

**.pypirc**

```
[distutils]
index-servers =
    pypi
    testpypi
    develop

[pypi]
repository = https://upload.pypi.org/legacy/

[testpypi]
repository = https://test.pypi.org/legacy/

[develop]
repository = https://<some_url>/develop/
username = <user>
password = <pass>
```

Then you would go to your component module as usual and execute:

```
make clean all install
make deploy
```

The command deploy would call setup.py and twine on all PY_MODULES and PY_PACKAGES. And end up deploying them in the configured repositories.

Currently at ALMA we're using an internal repository based on pypiserver, which is freely available at official PyPI, and you can find a Docker image in DockerHub.

There are some environment variables which help with the Python deployment of packages:

- ACS_DEPLOY_VERSION: The version (Default: If undefined, the version is deduced from $ALMASW_RELEASE: 2023DEC --> 2023.12. If it has a different format then $ALMASW_RELEASE is used as is for the version) for the artifact to be deployed.
- ACS_PYPI_ENV: In this case is mandatory, because the stage is used as the repository to be used by twine to upload the packages. It must match one of the repositories defined in .pypirc.