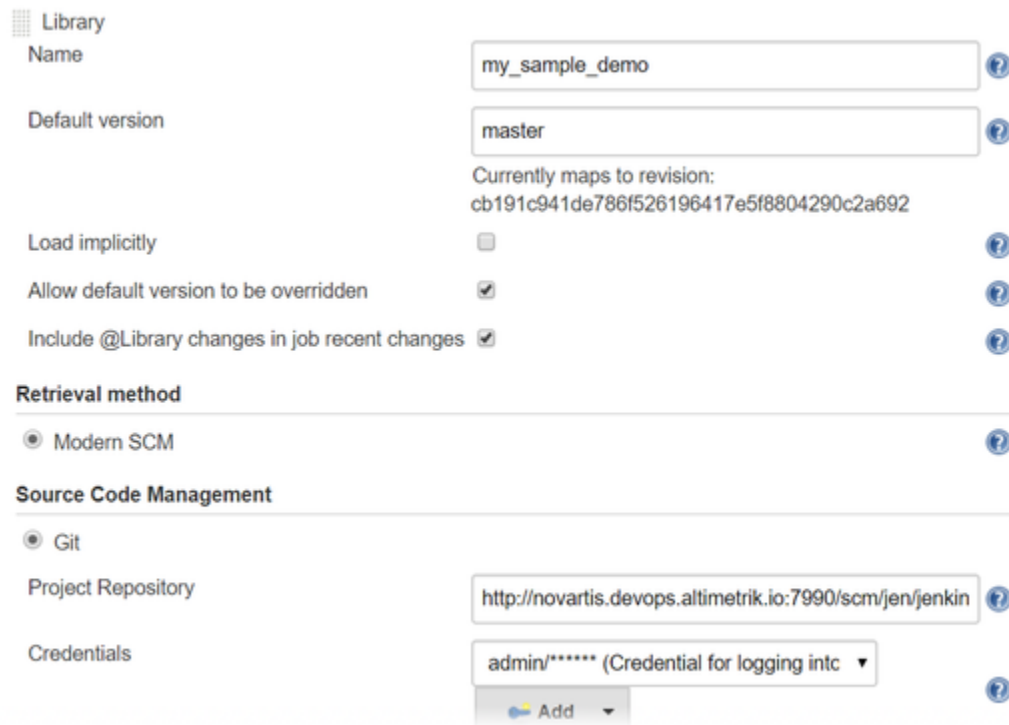


# Docker Image Creation Using Global Library

## Global library configuration in Jenkins

To do so, go into **Manage Jenkins -> Configure System** and find the **Global Pipeline Libraries** section. The shared library will be loaded on the fly from a git repository, in every job. It is never cached.



The screenshot shows the Jenkins configuration page for Global Pipeline Libraries. It includes a sidebar with a 'Library' icon. The main form has the following fields:

- Name:** my\_sample\_demo
- Default version:** master
- Currently maps to revision:** cb191c941de786f526196417e5f8804290c2a692
- Load implicitly:** ☐
- Allow default version to be overridden:** ☒
- Include @Library changes in job recent changes:** ☒
- Retrieval method:** Modern SCM
- Source Code Management:** Git
- Project Repository:** http://novartis.devops.altimetrik.io:7990/scm/jen/jenkin
- Credentials:** admin/\*\*\*\*\* (Credential for logging into...)

The Project repository is the url of your git repo where Global library code is present, [Shared Library](#)

## Shared Library:

Shared Library

```
#!/usr/bin/groovy
def call(Map config) {

    node {
        stage('Initialize') {
            def dockerHome = tool 'myDocker'
            env.PATH = "${dockerHome}/bin:${env.PATH}"
        }

        stage('Checkout') {
            echo "Checking out the sources..."
            checkout scm
        }

        stage('Build') {
            echo "Build Stage"
        }

        stage('Test') {
            echo "Test Stage"
        }

        stage('Package and Build Image') {
            sh 'python setup.py bdist'

            // Build Image

            if (config.DockerFilePath) {
                sh "cp ${config.PackagePath} ${config.DockerFilePath}"
                def customImage =
                    docker.build("novartis.devops.altimetrik.io:8081/docker-local/" +
                        "${config.ImageName}:${config.ImageVersion}",
                        "${config.DockerFilePath}")
            } else {
                sh "cp ${config.PackagePath} $WORKSPACE"
                def customImage =
                    docker.build("novartis.devops.altimetrik.io:8081/docker-local/" +
                        "${config.ImageName}:${config.ImageVersion}")
            }

        }

    }
}
}
```

1. In order to create a Docker image, the **Docker Pipeline** plugin provides a `build()` method for creating a new image, from a Dockerfile in the repository, during a Pipeline run. Example, `docker.build("novartis.devops.altimetrik.io:8081/docker-local/" + "${config.ImageName}:${config.ImageVersion}")`

2. `config.ImageName` and `config.ImageVersion` will be passed from `JenkinsFile` as defined below.

**Note :-** It will search in the root directory for the `Dockerfile`, if `DockerFilePath` is not defined.

To refer shared library in your `JenkinsFile` please use the following code snippet, Example: Sample repo for Integrating with shared library is available at [Demo Repo](#).

```
@Library('my-shared-library') _

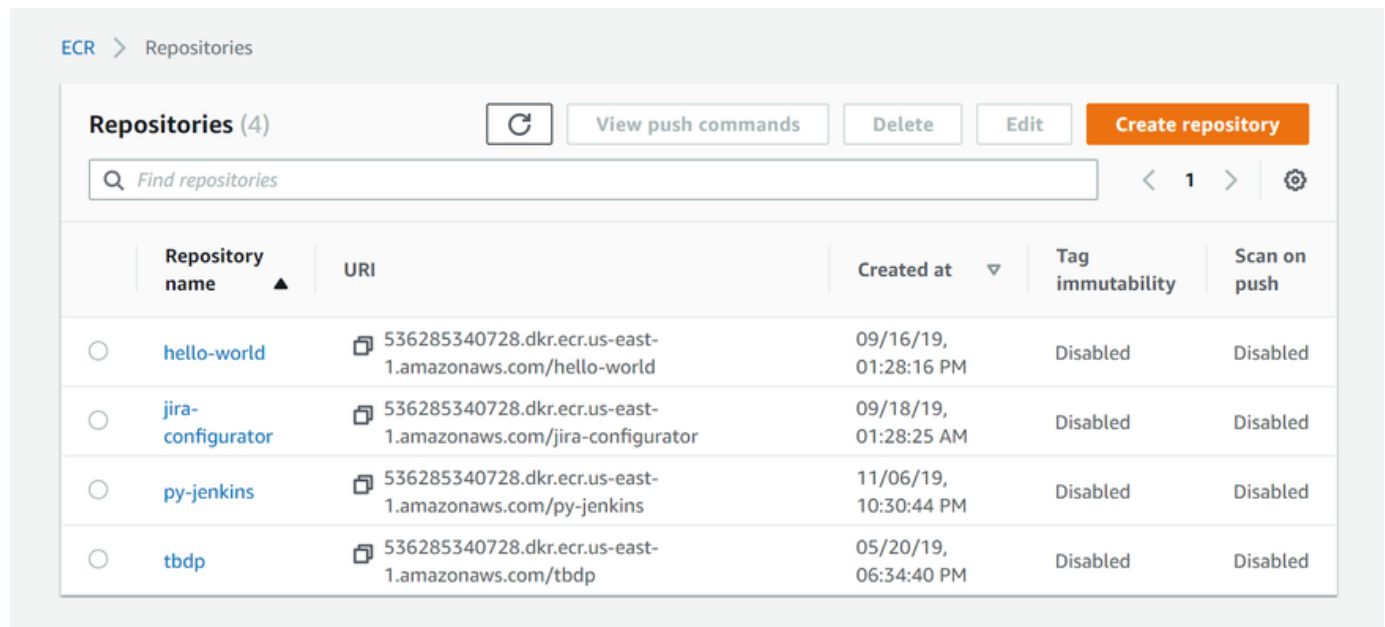
allInOne(
  ImageName: 'py-jenkins',
  ImageVersion: '0.1.0',
  DockerFilePath: './dockerfiles',
  PackagePath: '$WORKSPACE/dist/*'
)
```

To integrate this jenkins file we can use a pipeline project.

## Push Image To ECR

### Create the ECR Repository

- Log in to your AWS Console
- Open the **Elastic Container Registry(ECR)** service.
- Click the **Create repository** button in the **Repositories** tab.



The screenshot shows the AWS Elastic Container Registry (ECR) console. The breadcrumb navigation at the top reads "ECR > Repositories". The main heading is "Repositories (4)". To the right of the heading are buttons for "View push commands", "Delete", "Edit", and a prominent orange "Create repository" button. Below the heading is a search bar with the placeholder text "Find repositories". To the right of the search bar are pagination controls showing "< 1 >" and a settings gear icon. The main content is a table with the following columns: "Repository name" (with a sort icon), "URI", "Created at", "Tag immutability", and "Scan on push". There are four repositories listed:

Repository name	URI	Created at	Tag immutability	Scan on push
hello-world	536285340728.dkr.ecr.us-east-1.amazonaws.com/hello-world	09/16/19, 01:28:16 PM	Disabled	Disabled
jira-configurator	536285340728.dkr.ecr.us-east-1.amazonaws.com/jira-configurator	09/18/19, 01:28:25 AM	Disabled	Disabled
py-jenkins	536285340728.dkr.ecr.us-east-1.amazonaws.com/py-jenkins	11/06/19, 10:30:44 PM	Disabled	Disabled
tbdp	536285340728.dkr.ecr.us-east-1.amazonaws.com/tbdp	05/20/19, 06:34:40 PM	Disabled	Disabled

- Give a name to your repository. We can use the Image name as the name of the repository. Then, click the "Next" button.

### Add AWS Credentials to Jenkins

- From the home screen, hit the **Credentials** link in the left-side bar.
- Determine where you want to put your credentials. If unsure, go into the **Global credentials**.

- Click the **Add Credentials** link in the left-side navigation.
- For **Kind**, select **AWS Credentials**.
- Enter the *Access ID* and *Secret Access Key* for the AWS user that has access to the ECR repository.
- In the **Advanced** button, specify an ID that will make sense to you (so you don't have to remember a randomly generated UUID).

Kind	AWS Credentials
Scope	Global (Jenkins, nodes, items, all child items, etc)
Access Key ID	some_access_id
Secret Access Key	.....
Description	
ID	demo-ecr-credentials

OK

Install required plugins (if not already installed)

- Pipeline
- Docker Pipeline Plugin
- Amazon ECR Plugin

### Groovy script

```
docker.withRegistry("https://your.ecr.domain.amazonaws.com",
"ecr:us-east-1:credential-id"){
docker.image("your-image-name").push()
}
```

In order to obtain an ECR login credential, you must use the ecr provider prefix.