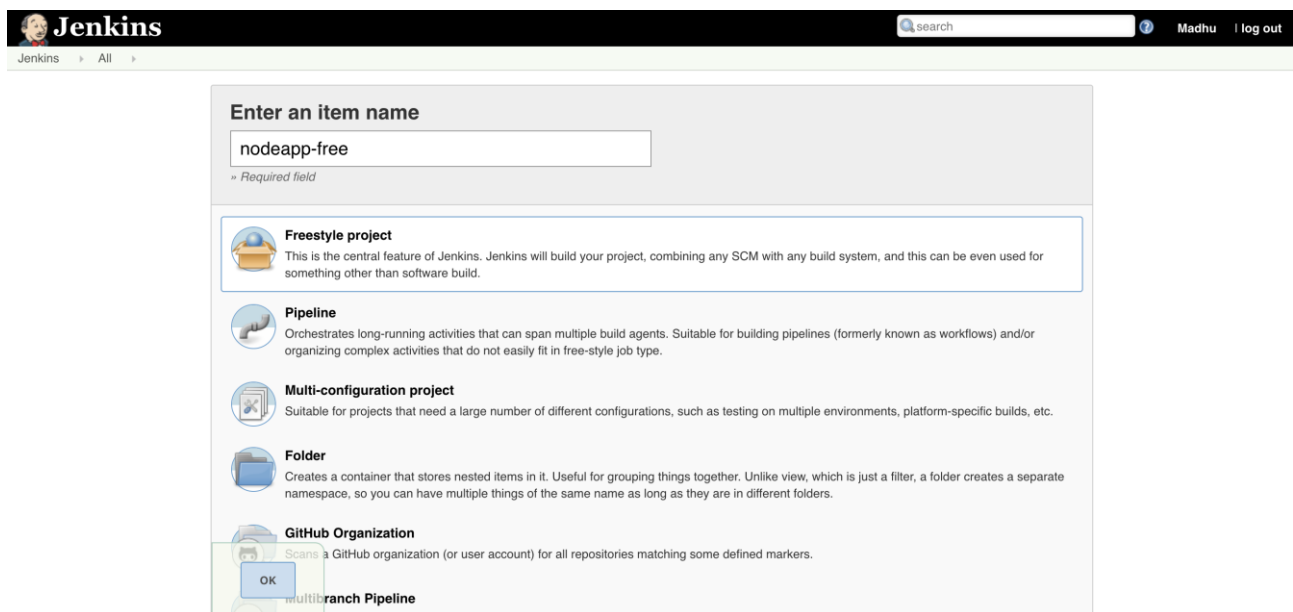


Push Docker image to ECR Repo

Freestyle

=====

1. Navigate to the "Plugin Manager" screen, install the "**Amazon ECR**" install plugin without restart.
2. Create "New item" freestyle. Give any job name.

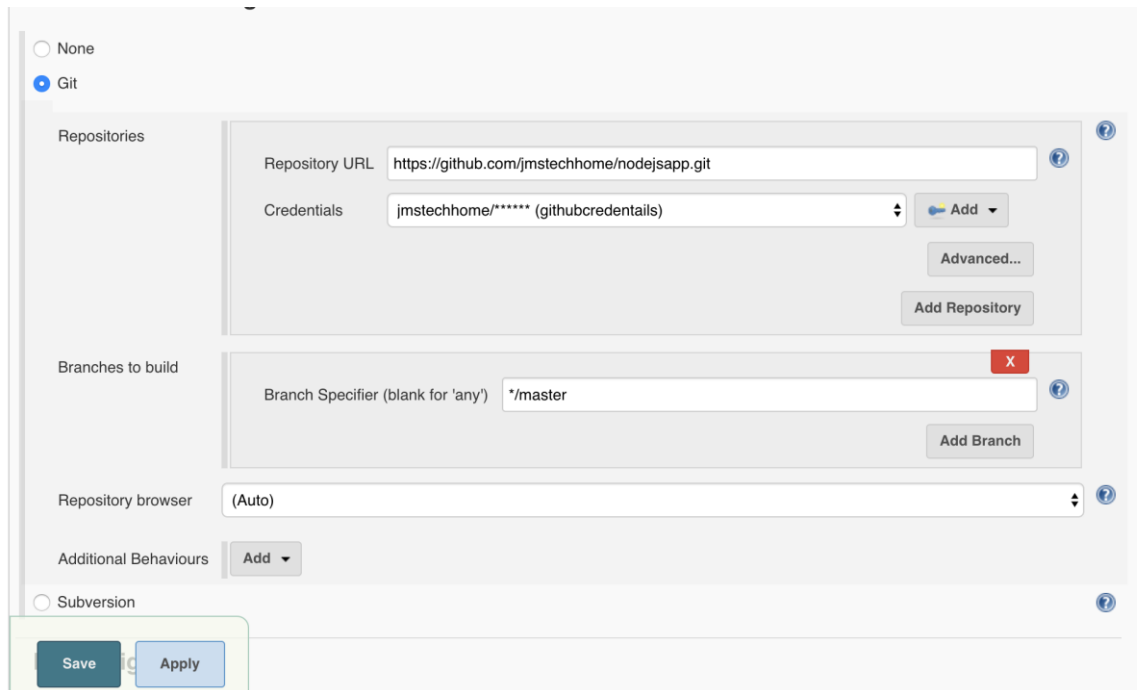


3. Amazon ECR plugin implements a Docker Token producer to convert Amazon credentials to Jenkins' API used by (mostly) all Docker-related plugins.

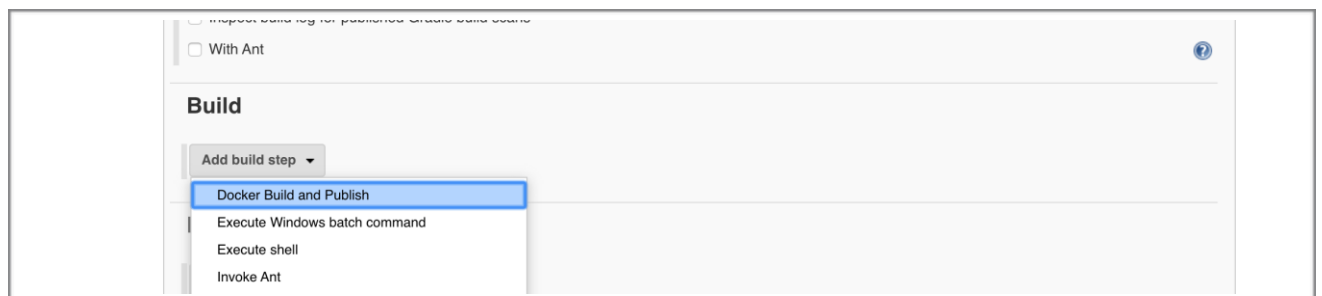
In this producer, you can select your existing registered Amazon credentials for various Docker operations in Jenkins, for sample using **CloudBees Docker Build and Publish plugin**:

Plugin to run your tests in Bitbar Cloud.		
<input type="checkbox"/>	BlazeMeter	4.7
This plugin integrates BlazeMeter - the cloud load testing service to Jenkins.		
<input checked="" type="checkbox"/>	CloudBees Docker Build and Publish	1.3.2
This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry.		
<input type="checkbox"/>	GCloud SDK	0.0.3
GCloud SDK Plugin allows the invocation of the gcloud CLI as a job step.		
<input type="checkbox"/>	IBM Application Security on Cloud	1.2.6
Nouvola DiveCloud		

4. Go to Jenkins job configure in build section select **docker build and publish** option.

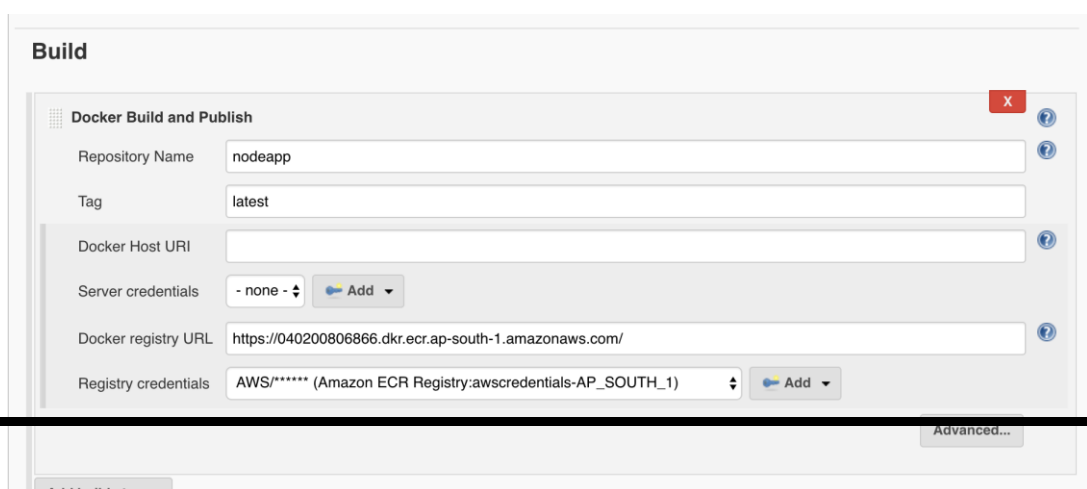


The screenshot shows the Jenkins job configuration page, specifically the 'Build' section. The 'Repository' is set to 'Git' with the URL 'https://github.com/jmstechhome/nodejsapp.git' and credentials 'jmstechhome/***** (githubcredentials)'. The 'Branches to build' section shows the 'Branch Specifier (blank for \'any\')' set to '*/master'. The 'Repository browser' is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. The 'Subversion' option is not selected. At the bottom, there are 'Save' and 'Apply' buttons.



The screenshot shows the Jenkins job configuration page, specifically the 'Build' section. The 'Add build step' dropdown menu is open, showing options: 'Docker Build and Publish', 'Execute Windows batch command', 'Execute shell', and 'Invoke Ant'. The 'Docker Build and Publish' option is highlighted.

5. Add below details . Make sure before adding this details you must Create a repo in ecr. And must be added aws credentials.



The screenshot shows the Jenkins job configuration page, specifically the 'Docker Build and Publish' step. The 'Repository Name' is set to 'nodeapp' and the 'Tag' is set to 'latest'. The 'Docker Host URI' is empty. The 'Server credentials' are set to 'none'. The 'Docker registry URL' is set to 'https://040200806866.dkr.ecr.ap-south-1.amazonaws.com/'. The 'Registry credentials' are set to 'AWS/***** (Amazon ECR Registry:awscredentials-AP_SOUTH_1)'. An 'Advanced...' button is visible at the bottom right.

ECR > Repositories > Create repository

Create repository

Repository configuration

Repository name

040200806866.dkr.ecr.ap-south-1.amazonaws.com/

A namespace can be included with your repository name (e.g. namespace/repo-name).

Tag immutability

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☒ Disabled

Scan on push

Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

☒ Disabled

Cancel

Create repository

Now build the job and see the output.

If you see below error because of Jenkins user connect docker commands without sudo. Then you must add Jenkins user into docker group and restart the Jenkins.

\$ sudo usermod -aG docker jenkins

```
console message: node application
First time build. Skipping changelog.
[nodeapp-free] $ docker build -t 040200806866.dkr.ecr.ap-south-1.amazonaws.com/nodeapp:latest --pull=true
/var/lib/jenkins/workspace/nodeapp-free
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post
http://%2Fvar%2Frun%2Fdocker.sock/v1.39/build?
buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuquota=0&cpuusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile
&labels=%7B%7D&memory=0&memswap=0&networkmode=default&pull=1&rm=1&session=7mdlhblh2e33kjyiedgx7nbt&shmsize=0&t=040200806866.dkr
.ecr.ap-south-1.amazonaws.com%2Fnodeapp%3Alatest&target=&ulimits=null&version=1: dial unix /var/run/docker.sock: connect:
permission denied
Build step 'Docker Build and Publish' marked build as failure
Finished: FAILURE
```

Pipeline

=====

1. Create a pipeline job.
2. Write below declarative pipeline code.

```
pipeline {
  agent any
  environment {
    {
      VERSION = "${BUILD_NUMBER}"
      PROJECT = 'nodeapp'
      IMAGE = "$PROJECT:$VERSION"
      ECRURL = 'https://040200806878.dkr.ecr.ap-south-1.amazonaws.com'
      ECRURED = 'ecr:ap-south-1:awscredentials'
    }
  }
  stages {
    stage('GetSCM') {
      steps {
        // Get some code from a GitHub repository
        git 'https://github.com/jmstechhome/nodejsapp.git'
      }
    }
    stage('Image Build'){
      steps{
        script{
          docker.build('$IMAGE')
        }
      }
    }
    stage('Push Image'){
      steps{
        script
        {
          docker.withRegistry(ECRURL, ECRURED)
          {
            docker.image(IMAGE).push()
          }
        }
      }
    }
  }
}
```

```

    }
  }
}

post
{
  always
  {
    // make sure that the Docker image is removed
    sh "docker rmi $IMAGE | true"
  }
}
}

```

[Delete Pipeline](#)
[Configure](#)
[Full Stage View](#)
[Rename](#)
[Pipeline Syntax](#)

[Recent Changes](#)

Stage View

Average stage times:
(Average full run time: ~6s)

	GetSCM	Image Build	Push Image	Declarative: Post Actions
Average	1s	1s	1s	362ms
#8 Jan 20 15:23	877ms Success Logs	1s	1s	362ms
#7 Jan 20 15:20	No Changes	1s	1s	

Build History

find

#	Time
#8	20-Jan-2020 09:53
#7	20-Jan-2020 09:50
#6	20-Jan-2020 09:47
#5	20-Jan-2020 09:46

And go and check ecr repo the images is pushed or not.

If you basic auth credentials issue: please do below steps

```

sudo apt install awscli
sudo apt-get install python3-pip
sudo pip3 install --upgrade awscli
aws configure

```

eval \$(aws ecr get-login --no-include-email --region ap-south-1 | sed 's|https://||')