Integration of TMAS in Jenkins

Step 1 – Install Docker and Jenkins on your Server

1. Update your system:

```
sudo apt update -y && sudo apt upgrade -y
```

2. Install Docker:

```
sudo apt install docker.io -y
```

3. Enable and start Docker service:

```
sudo systemctl enable docker
sudo systemctl start docker
```

4. Install Jenkins (on Ubuntu):

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \
    /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
    https://pkg.jenkins.io/debian binary/ | sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update -y
sudo apt install openjdk-17-jdk -y   # Jenkins needs Java
sudo apt install jenkins -y
```

5. Add Jenkins user to Docker group (so Jenkins can run Docker builds):

```
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
```

Step 2 – Install TMAS CLI and AWS CLI

1. Download and install TMAS CLI:

```
wget https://cli.artifactscan.cloudone.trendmicro.com/tmas-cli/latest/tmas-cli_Linux_x86_64.tar.gz
tar -xvf tmas-cli_Linux_x86_64.tar.gz
chmod +x tmas
sudo mv tmas /usr/local/bin/
```

Test installation:

```
tmas version
```

2. Install AWS CLI v2:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Verify:

```
aws --version
```

Step 3 – Configure AWS IAM User

1. Create an **IAM user** with following permissions:
    a. `AmazonEC2ContainerRegistryFullAccess`
    b. `AmazonEKSClusterPolicy` (if you want to later deploy to EKS)

c. `AmazonECRPublicReadOnly`

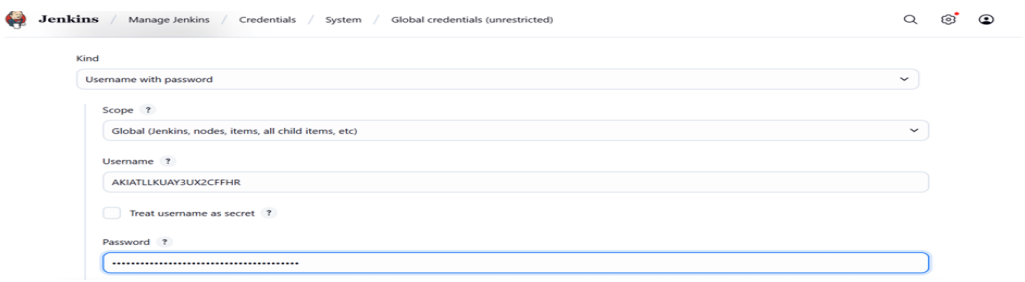
2. Generate **AWS ACCESS KEY** and **AWS SECRET ACCESS KEY**.
3. Configure locally:

```
aws configure
```

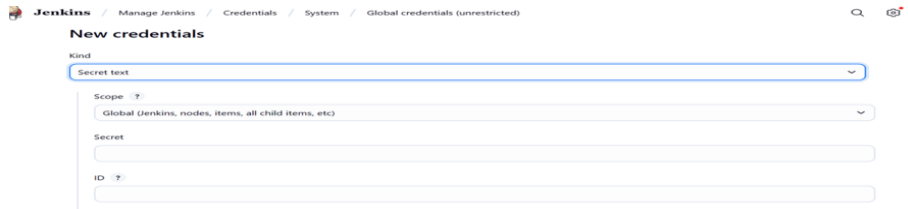
Provide Access Key, Secret, and default region (e.g., `ap-south-1`).

Step 4 – Add Credentials in Jenkins

1. Go to **Manage Jenkins → Credentials → Global → Add Credentials**.
   a. Choose **Kind: Username and Password**
      i. Username = AWS_ACCESS_KEY
      ii. Password = AWS_SECRET_ACCESS_KEY
      iii. ID = `aws-creds`

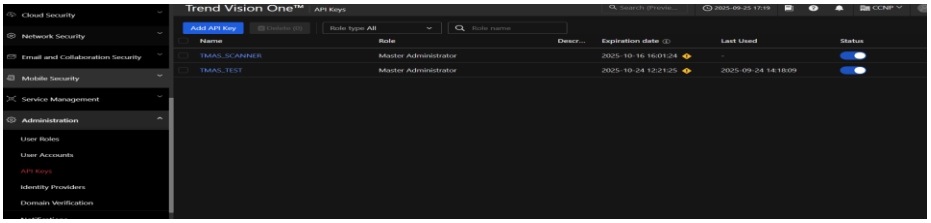


   b. Choose **Kind: Secret text** for TMAS API key
      i. Secret = `<YOUR_TMAS_API_KEY>`
      ii. ID = `tmas-api-key`



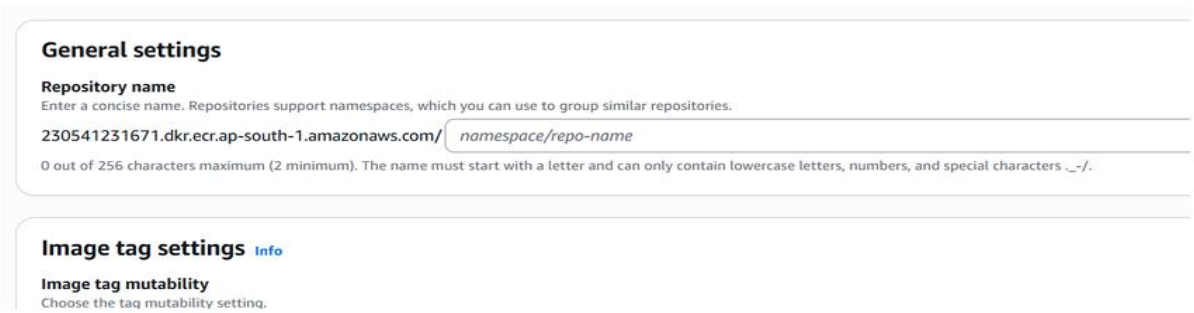
Step 5 – Generate TMAS API Key

1. Login to **Trend Micro Cloud One console** → Go to **Administration → API Keys**.
2. Create API Key with **Master Admin permission**.
3. Save this key and add it as a Jenkins credential (`tmas-api-key`).



Step 6 – Create AWS ECR Repository

1. In AWS Console → **ECR** → Create repository.
   Example name: `myapp-repo`.



2. Clone your app code to Jenkins server:

```
git clone https://github.com/mohitverma4109/two-tier-flask-app.git
cd two-tier-flask-app
```

Step 7 – Install Jenkins Plugins

- Go to **Manage Jenkins → Plugins → Available Plugins**.
- Install:

- **Pipeline**
- **Pipeline Stage View**
- **Git plugin**
- **Docker Pipeline**

Restart Jenkins after installation.

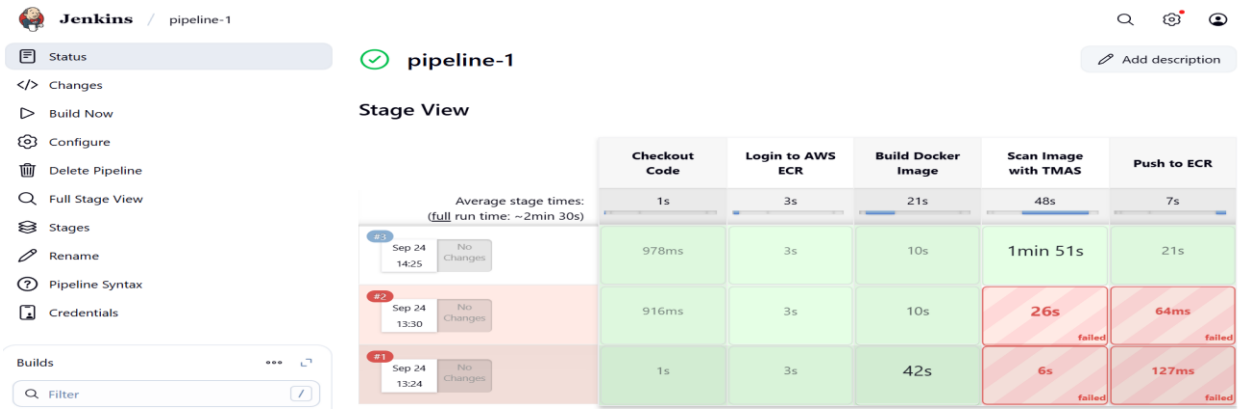
Step 8 – Jenkins Pipeline

Here's a complete pipeline (`Jenkinsfile`) with **extra details** added:

```
pipeline {
    agent any

    environment {
        AWS_REGION     = 'ap-south-1'
        AWS_ACCOUNT_ID = '230541231671'
        ECR_REPO       = 'myapp-repo'    // Your ECR repo name
        IMAGE_TAG      = "latest"
        TMAS_URL       = 'https://container.trendmicro.com'
    }

    stages {
        stage('Checkout Code') {
            steps {
         git branch: 'master',
                    url: 'https://github.com/mohitverma4109/two-tier-
flask-app.git'
            }
        }

        stage('Login to AWS ECR') {
            steps {
                withCredentials([usernamePassword(
                    credentialsId: 'aws-creds',
                    usernameVariable: 'AWS_ACCESS_KEY_ID',
                    passwordVariable: 'AWS_SECRET_ACCESS_KEY'
```

```
                )]) {
                    sh '''
                        aws configure set aws_access_key_id
$AWS_ACCESS_KEY_ID
                        aws configure set aws_secret_access_key
$AWS_SECRET_ACCESS_KEY
                        aws configure set default.region $AWS_REGION

                        aws ecr get-login-password --region
$AWS_REGION | \
                          docker login --username AWS \
                          --password-stdin
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com
                    '''
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                sh '''
                    docker build -t $ECR_REPO:$IMAGE_TAG .
                    docker tag $ECR_REPO:$IMAGE_TAG \

$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO:$IMAGE_TAG
                '''
            }
        }

        stage('Scan Image with TMAS') {
            steps {
                withCredentials([string(credentialsId: 'tmas-api-key',
variable: 'TMAS_API_KEY')]) {
                    sh '''
                        export TMAS_API_KEY=$TMAS_API_KEY
                        export TMAS_URL=$TMAS_URL

                        echo "Running TMAS scan on Docker image..."
                        tmas scan
```

```
docker:$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO:$IM
AGE_TAG \
                        -V -M -S --region $AWS_REGION
                    '''
                }
            }
        }

        stage('Push to ECR') {
            steps {
                sh '''
                    docker push
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO:$IMAGE_TAG
                '''
            }
        }
    }

    post {
        always {
            echo "Pipeline execution completed. Check Trend Micro
console for scan reports."
        }
    }
}
```



Outcome

- Pull code from GitHub.
- Build a Docker image.

- Scan it with TMAS for vulnerabilities.
- Push the image to AWS ECR