Vadakathi Muhammed Suhaib
Technical Apprentice
Emp ID: X48GRSTML
muhammed.suhaib@cprime.com

Write Optimal Dockerfiles for Jenkins, Nexus, and SonarQube

Phase 1: Container Image Preparation

Task 1: Write Optimal Dockerfiles for Jenkins, Nexus, and SonarQube

Overview

This document provides a step-by-step process for creating secure and production-ready Docker images for Jenkins, Nexus, and SonarQube, then pushing them to Azure Container Registry (ACR).

Prerequisites

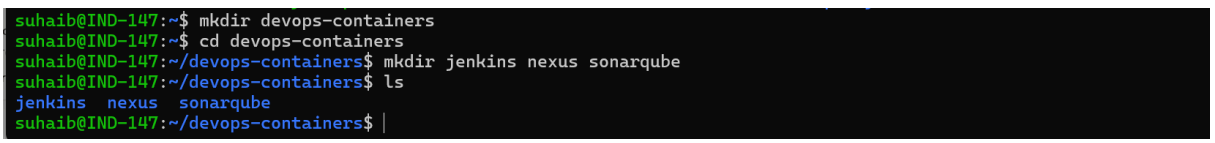
- Docker installed and running
- Azure CLI installed and configured
- Access to Azure Container Registry
- Basic knowledge of Docker and containerization

Step-by-Step Process

Step 1: Environment Setup

1. **Create project directory structure:**

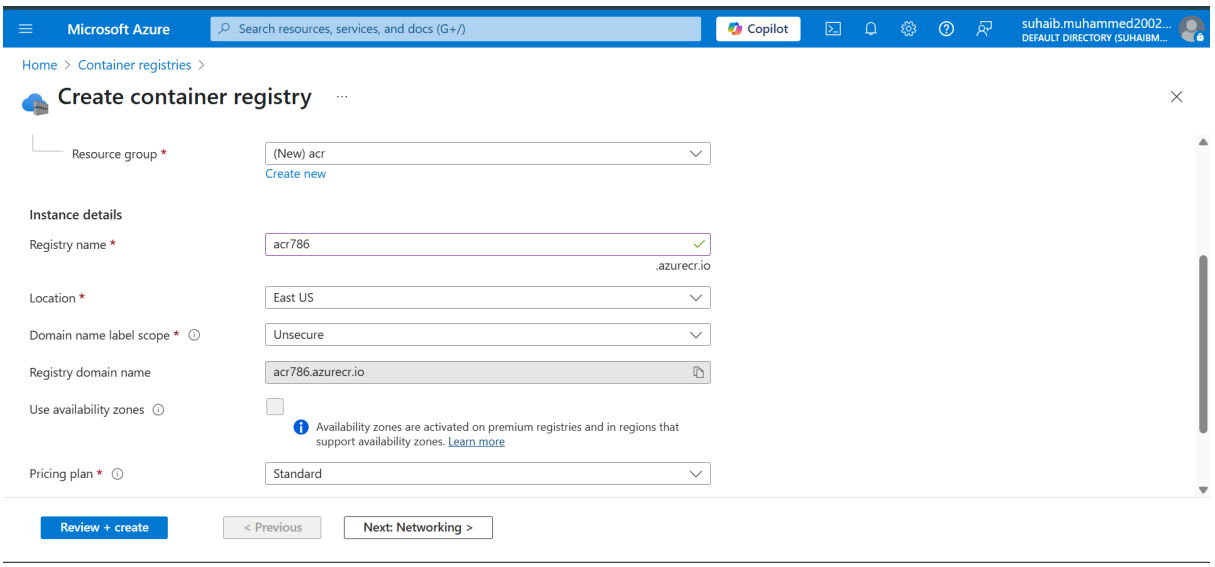
```
mkdir devops-containers
cd devops-containers
mkdir jenkins nexus sonarqube
```



2. **Install Azure CLI and Docker**

```
# Install Azure CLI (Ubuntu/Debian)
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

# Install Docker (Ubuntu/Debian)
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker $USER
```

3. **Create a Container Registry in Azure Portal and Note the name**



4. **Login to Azure and ACR:**

```
az login
az acr login --name <your-acr-name>
```

```
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue
the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow w
ith `az login --use-device-code`.

Retrieving tenants and subscriptions for the selection...

[Tenant and subscription selection]

No     Subscription name    Subscription ID                       Tenant
-----  -------------------  ------------------------------------  -----------------
[1] *  Azure for Students   0f9ec8b3-d366-4f81-9873-dbbde1e72b8c  Default Directory

The default is marked with an *; the default tenant is 'Default Directory' and subscription is 'Azure for Students' (0f9
ec8b3-d366-4f81-9873-dbbde1e72b8c).

Select a subscription and tenant (Type a number or Enter for no changes): 1

Tenant: Default Directory
Subscription: Azure for Students (0f9ec8b3-d366-4f81-9873-dbbde1e72b8c)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about i
t and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subsc
riptions by default.

suhaib@IND-147:~/devops-containers$
```

```
suhaib@IND-147:~/devops-containers$ az acr login --name acr786
Login Succeeded
suhaib@IND-147:~/devops-containers$
```

3. **Set environment variables:**

```
export ACR_NAME="acr786"
export ACR_LOGIN_SERVER="${ACR_NAME}.azurecr.io"
export RESOURCE_GROUP="acr"
```

```
suhaib@IND-147:~/devops-containers$ export ACR_NAME="acr786"
suhaib@IND-147:~/devops-containers$ export ACR_LOGIN_SERVER="${ACR_NAME}.azurecr.io"
suhaib@IND-147:~/devops-containers$ export RESOURCE_GROUP="acr"
```

Step 2: Jenkins Container Preparation

1. **Navigate to Jenkins directory:**

```
cd jenkins
```

2. **Create Dockerfile**

```
# Jenkins Dockerfile - Production Ready
FROM jenkins/jenkins:lts-jdk17

# Switch to root to install packages
USER root
```

```
# Install additional tools and clean up in single layer
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        curl \
        wget \
        git \
        unzip \
        ca-certificates \
        apt-transport-https \
        gnupg \
        lsb-release && \
    # Install Docker CLI
    curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -
    echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyring
    apt-get update && \
    apt-get install -y docker-ce-cli && \
    # Install kubectl
    curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stabl
    install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl && \
    rm kubectl && \
    # Clean up
    apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# Install Azure CLI
RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash && \
    rm -rf /var/lib/apt/lists/*

# Switch back to jenkins user
USER jenkins

# Skip initial setup wizard
ENV JAVA_OPTS="-Djenkins.install.runSetupWizard=false"
ENV JENKINS_OPTS="--httpPort=8080 --httpsPort=-1"

# Copy plugins list and install plugins
COPY plugins.txt /usr/share/jenkins/ref/plugins.txt
```

```
RUN jenkins-plugin-cli --plugin-file /usr/share/jenkins/ref/plugins.txt

# Copy custom configuration
COPY --chown=jenkins:jenkins jenkins.yaml /var/jenkins_home/casc_configs/

# Set JVM options for production
ENV JAVA_OPTS="-Xmx2g -Xms1g -XX:+UseG1GC -XX:+UseContainerSuppo

# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=5m --retries=3 \
   CMD curl -f http://localhost:8080/login || exit 1

# Expose port
EXPOSE 8080 50000

# Use default entrypoint from base image
```

3. **Create plugins.txt file:**

```
# Jenkins Essential Plugins List

# Build Tools
ant:latest
gradle:latest
maven-plugin:latest

# SCM
git:latest
github:latest
github-branch-source:latest
bitbucket:latest

# Pipeline
workflow-aggregator:latest
pipeline-stage-view:latest
pipeline-github-lib:latest
pipeline-build-step:latest
pipeline-input-step:latest
```

```
# Build Environment
build-timeout:latest
timestamper:latest
ws-cleanup:latest
# environment-injector:latest  # REMOVED - causing 404 errors

# Authentication & Authorization
matrix-auth:latest
ldap:latest
active-directory:latest
role-strategy:latest

# Notifications
email-ext:latest
mailer:latest
slack:latest

# Testing & Quality
junit:latest
jacoco:latest
sonar:latest
performance:latest

# Deployment
deploy:latest
ssh-slaves:latest
ssh-agent:latest
publish-over-ssh:latest

# Cloud & Container
docker-plugin:latest
docker-workflow:latest
kubernetes:latest
azure-container-agents:latest

# Monitoring & Logging
monitoring:latest
```

```
log-parser:latest
build-monitor-plugin:latest

# Utilities
credentials-binding:latest
parameterized-trigger:latest
copyartifact:latest
build-name-setter:latest
description-setter:latest

# Security
credentials:latest
plain-credentials:latest
ssh-credentials:latest
```

4. **Create jenkins.yaml file:**

```
jenkins:
  systemMessage: "Jenkins configured automatically by Configuration as Code

  # Global security settings
  globalNodeProperties:
    - envVars:
        env:
          - key: "JAVA_HOME"
            value: "/opt/java/openjdk"
          - key: "PATH"
            value: "$PATH:/opt/java/openjdk/bin"

  # Security realm
  securityRealm:
    local:
      allowsSignup: false
      users:
        - id: "admin"
          password: "${JENKINS_ADMIN_PASSWORD:-admin123}"
          properties:
            - "hudson.security.HudsonPrivateSecurityRealm$Details":
```

```
            passwordHash: "${JENKINS_ADMIN_PASSWORD_HASH}"

  # Authorization strategy
  authorizationStrategy:
    globalMatrix:
      permissions:
        - "Overall/Administer:admin"
        - "Overall/Read:authenticated"

  # Global tool configuration
  tool:
    git:
      installations:
        - name: "Default"
          home: "/usr/bin/git"

    maven:
      installations:
        - name: "Maven-3.9"
          properties:
            - installSource:
                installers:
                  - maven:
                      id: "3.9.5"

    dockerTool:
      installations:
        - name: "Docker"
          properties:
            - installSource:
                installers:
                  - fromDocker:
                      version: "latest"

# Global security configuration
security:
  # Prevent Cross Site Request Forgery exploits
  crumbIssuer:
```

```
    standard:
      excludeClientIPFromCrumb: false

  # Global security settings
  globalJobDslSecurityConfiguration:
    useScriptSecurity: true

# Unclassified configuration
unclassified:
  # Location configuration
  location:
    adminAddress: "admin@company.com"
    url: "http://jenkins:8080/"

  # Global libraries configuration
  globalLibraries:
    libraries:
      - name: "shared-library"
        defaultVersion: "main"
        retriever:
          modernSCM:
            scm:
              git:
                remote: "https://github.com/company/jenkins-shared-library.git"
```

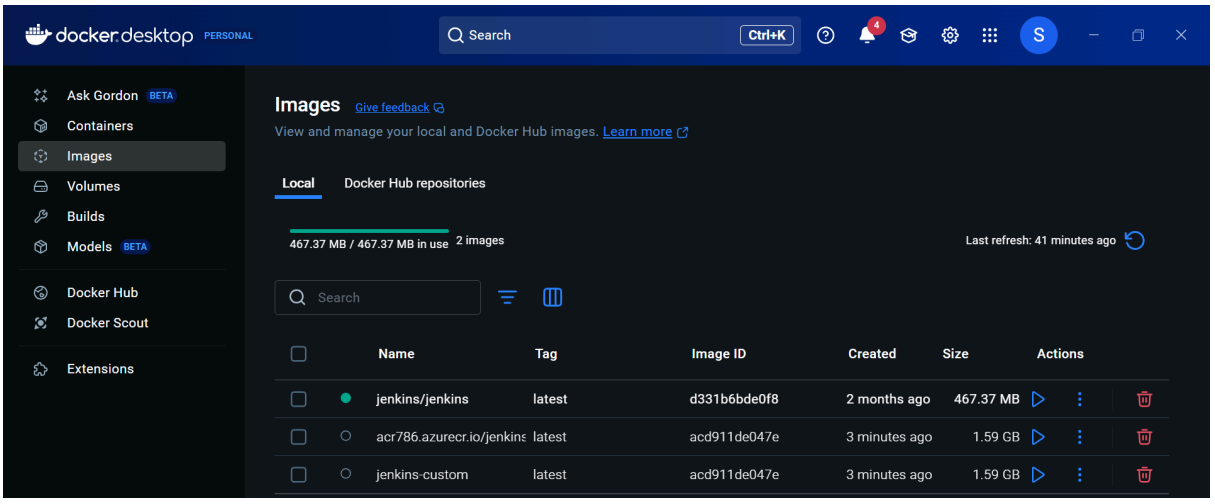
```
suhaib@IND-147:~/devops-containers$ cd jenkins
suhaib@IND-147:~/devops-containers/jenkins$ sudo nano Dockerfile
[sudo] password for suhaib:
suhaib@IND-147:~/devops-containers/jenkins$ sudo nano plugins.txt
suhaib@IND-147:~/devops-containers/jenkins$ sudo nano jenkins.yaml
suhaib@IND-147:~/devops-containers/jenkins$ ls
Dockerfile  jenkins.yaml  plugins.txt
suhaib@IND-147:~/devops-containers/jenkins$
```

5. **Build Jenkins image:**

```
docker build -t jenkins-custom:latest .
```

```
suhaib@IND-147:~/devops-containers/jenkins$ docker build -t jenkins-custom:latest .
[+] Building 889.3s (12/12) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.1s
 => => transferring dockerfile: 2.15kB                                                                  0.0s
 => [internal] load metadata for docker.io/jenkins/jenkins:lts-jdk17                                    5.8s
 => [auth] jenkins/jenkins:pull token for registry-1.docker.io                                          0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [1/6] FROM docker.io/jenkins/jenkins:lts-jdk17@sha256:3cc41bac7bdeba7fef4c5421f72d0143b08b288362e  80.0s
 => => resolve docker.io/jenkins/jenkins:lts-jdk17@sha256:3cc41bac7bdeba7fef4c5421f72d0143b08b288362e53  0.1s
 => => sha256:3cc41bac7bdeba7fef4c5421f72d0143b08b288362e539143aed454a6c7dade5 3.12kB / 3.12kB          0.0s
 => => sha256:3e6b9d1a95114e19f12262a4e8a59ad1d1a10ca7b82108adcf0605a200294964 48.49MB / 48.49MB       16.1s
 => => sha256:980fcecbb536c6bf3fdfbaf08c92a058d3b185d17020f0bf34d67606b7a30319 2.58kB / 2.58kB          0.0s
 => => sha256:a51d3f0ef2f6393404de53b542adf033434a6309b802ec82dfd64359c27c0d31 12.79kB / 12.79kB        0.0s
 => => sha256:1c0b6121482a147ae2c82071540baa45fec77656d8c293bc06acc64a3f11eaff 61.64MB / 61.64MB       17.9s
 => => sha256:cbd1283dc3b2995d90fbd76927489767bcf8ff4d19491ccd841d5f6278c3a142 4.98MB / 4.98MB          4.8s
 => => sha256:b9b1866598bda4299aea6407b959986c424c1f4f7749d08efee20e028e14da59 1.24kB / 1.24kB          5.2s
 => => sha256:88ce37b37f54a85d80240133b416c71d0a5e396487d324fd943f3f064ed5c85c 182B / 182B              5.5s
 => => sha256:57daf441aeec774a343cefc1499ac351d7560c6f1ef6b763328b3fae6cf70fe0 94.17MB / 94.17MB       75.5s
 => => sha256:10e8fd64a50341a578e2d37ff610a9de28addeb573c664ca37b5e4b1c049f112 189B / 189B            19.0s
 => => extracting sha256:3e6b9d1a95114e19f12262a4e8a59ad1d1a10ca7b82108adcf0605a200294964               5.0s
 => => sha256:d535607246585ad3f5735644d99e827d5effe27493143519cceb2a68cb21b72e 6.29MB / 6.29MB         20.0s
 => => sha256:287a1e60bbedc2f67a2fc9d8d219aa6abb6f90413d4c7d34a076cc434645c0b5 62.96MB / 62.96MB       33.9s
 => => sha256:facffd25c99beea76b87223fdef05c3c9c84e3fb6acc358972e86287df488e94 1.92kB / 1.92kB          20.4s
 => => sha256:830c9e42c618db4bdf7c1266ffb314c0b228197487a40c27ff3eec5503f96f54 1.29kB / 1.29kB          20.7s
 => => sha256:5d6d2a5d18a4c8c1a0066cd1089deb1bb52c87d376e1fd6a985b7bef9c06cd69 390B / 390B              21.0s
 => => extracting sha256:1c0b6121482a147ae2c82071540baa45fec77656d8c293bc06acc64a3f11eaff               4.2s
```

```
 => => extracting sha256:cbd1283dc3b2995d90fbd76927489767bcf8ff4d19491ccd841d5f6278c3a142               0.1s
 => => extracting sha256:b9b1866598bda4299aea6407b959986c424c1f4f7749d08efee20e028e14da59               0.0s
 => => extracting sha256:88ce37b37f54a85d80240133b416c71d0a5e396487d324fd943f3f064ed5c85c               0.0s
 => => extracting sha256:57daf441aeec774a343cefc1499ac351d7560c6f1ef6b763328b3fae6cf70fe0               1.0s
 => => extracting sha256:10e8fd64a50341a578e2d37ff610a9de28addeb573c664ca37b5e4b1c049f112               0.0s
 => => extracting sha256:d535607246585ad3f5735644d99e827d5effe27493143519cceb2a68cb21b72e               0.2s
 => => extracting sha256:287a1e60bbedc2f67a2fc9d8d219aa6abb6f90413d4c7d34a076cc434645c0b5               0.9s
 => => extracting sha256:facffd25c99beea76b87223fdef05c3c9c84e3fb6acc358972e86287df488e94               0.0s
 => => extracting sha256:830c9e42c618db4bdf7c1266ffb314c0b228197487a40c27ff3eec5503f96f54               0.0s
 => => extracting sha256:5d6d2a5d18a4c8c1a0066cd1089deb1bb52c87d376e1fd6a985b7bef9c06cd69               0.0s
 => [internal] load build context                                                                       0.1s
 => => transferring context: 1.29kB                                                                     0.0s
 => [2/6] RUN apt-get update &&     apt-get install -y --no-install-recommends        curl        wget  150.0s
 => [3/6] RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash &&     rm -rf /var/lib/apt/lists/*      378.4s
 => [4/6] COPY plugins.txt /usr/share/jenkins/ref/plugins.txt                                           0.3s
 => [5/6] RUN jenkins-plugin-cli --plugin-file /usr/share/jenkins/ref/plugins.txt                     266.1s
 => [6/6] COPY --chown=jenkins:jenkins jenkins.yaml /var/jenkins_home/casc_configs/jenkins.yaml         0.4s
 => exporting to image                                                                                  7.6s
 => => exporting layers                                                                                 7.4s
 => => writing image sha256:acd911de047eaca436eb8db65be1ea8aa29e1a3f998abd1f0c336f5e2f88dd80            0.0s
 => => naming to docker.io/library/jenkins-custom:latest                                                0.0s
suhaib@IND-147:~/devops-containers/jenkins$
```

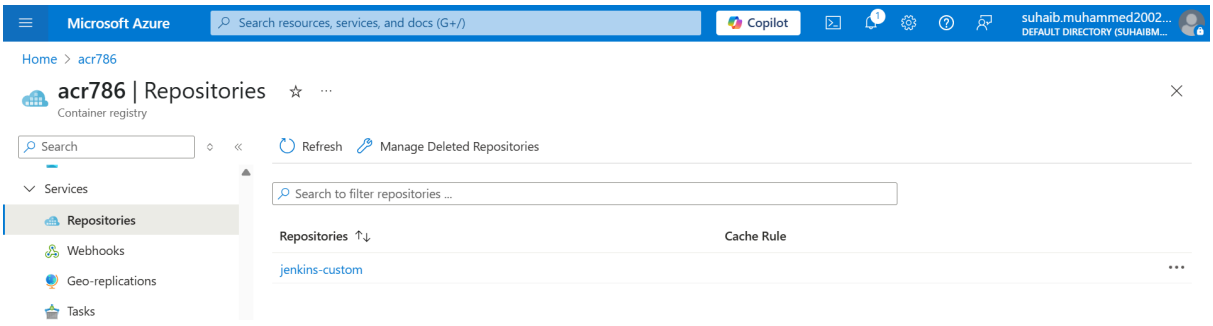


6. **Tag and push to ACR:**

```
docker tag jenkins-custom:latest ${ACR_LOGIN_SERVER}/jenkins-custom:latest
```

```
docker push ${ACR_LOGIN_SERVER}/jenkins-custom:latest
```

```
suhaib@IND-147:~/devops-containers/jenkins$ docker tag jenkins-custom:latest ${ACR_LOGIN_SERVER}/jenkins-custom:latest
suhaib@IND-147:~/devops-containers/jenkins$ docker push ${ACR_LOGIN_SERVER}/jenkins-custom:latest
The push refers to repository [acr786.azurecr.io/jenkins-custom]
37cb307cd408: Pushed
2f26cf806ba8: Pushed
4417e0166673: Pushed
0ea6bbd25352: Pushed
9693677920ff: Pushed
d0f54ffdf691: Pushed
b20d768f3176: Pushed
b8cd222ec471: Pushed
d4192ba56618: Pushed
0a7c1cf94bc4: Pushed
d6ab3a452f20: Pushed
5a84a1f3379e: Pushed
a64e8fbcdbfe: Pushed
e2336f3e92cf: Pushed
649a2f525af6: Pushed
d9dc08411f8e: Pushed
2f7436e79a0b: Pushed
latest: digest: sha256:cc860383121562b854a8216f13be094a017d0ef074cb1d1d173ffdf5e343cbbd size: 3886
suhaib@IND-147:~/devops-containers/jenkins$
```



Step 3: Nexus Container Preparation

1. **Navigate to Nexus directory:**

```
cd ../nexus
```

2. **Create Dockerfile**

```
# Nexus Repository Manager Dockerfile - Production Ready
FROM sonatype/nexus3:3.45.0

# Switch to root for setup
USER root

# Create nexus data directory with proper permissions
RUN mkdir -p /nexus-data/etc && \
    chown -R nexus:nexus /nexus-data
```

```
# Switch back to nexus user
USER nexus

# Copy custom configuration
COPY --chown=nexus:nexus nexus.properties /nexus-data/etc/nexus.propert
COPY --chown=nexus:nexus scripts/ /opt/sonatype/nexus/scripts/

# Set JVM options for production
ENV INSTALL4J_ADD_VM_PARAMS="-Xms2g -Xmx2g -XX:MaxDirectMemory

# Configure Nexus properties
ENV NEXUS_SECURITY_RANDOMPASSWORD=false

# Health check
HEALTHCHECK --interval=30s --timeout=15s --start-period=10m --retries=3
    CMD curl -f http://localhost:8081/service/rest/v1/status || exit 1

# Expose ports
EXPOSE 8081

# Volume for data persistence
VOLUME ["/nexus-data"]

# Use default entrypoint from base image
```

3. **Create initialization script:**

```
mkdir scripts
# Create nexus-init.groovy
```

```
import org.sonatype.nexus.repository.Repository
import org.sonatype.nexus.repository.manager.RepositoryManager
import org.sonatype.nexus.repository.maven.LayoutPolicy
import org.sonatype.nexus.repository.maven.VersionPolicy
import org.sonatype.nexus.repository.config.Configuration
import org.sonatype.nexus.repository.storage.WritePolicy
```

```
// Get repository manager
repositoryManager = container.lookup(RepositoryManager.class.getName())

// Create Maven repositories
def createMavenProxy(String name, String remoteUrl) {
    def existingRepo = repositoryManager.get(name)
    if (existingRepo == null) {
        log.info("Creating Maven proxy repository: " + name)

        Configuration config = new Configuration(
            repositoryName: name,
            recipeName: 'maven2-proxy',
            online: true,
            attributes: [
                maven: [
                    versionPolicy: VersionPolicy.MIXED,
                    layoutPolicy: LayoutPolicy.STRICT
                ],
                proxy: [
                    remoteUrl: remoteUrl,
                    contentMaxAge: 1440,
                    metadataMaxAge: 1440
                ],
                httpclient: [
                    blocked: false,
                    autoBlock: true
                ],
                storage: [
                    blobStoreName: 'default',
                    strictContentTypeValidation: true
                ]
            ]
        )

        repositoryManager.create(config)
        log.info("Created Maven proxy repository: " + name)
    } else {
        log.info("Maven proxy repository already exists: " + name)
```

```
    }
}

def createMavenHosted(String name) {
    def existingRepo = repositoryManager.get(name)
    if (existingRepo == null) {
        log.info("Creating Maven hosted repository: " + name)

        Configuration config = new Configuration(
            repositoryName: name,
            recipeName: 'maven2-hosted',
            online: true,
            attributes: [
                maven: [
                    versionPolicy: VersionPolicy.MIXED,
                    layoutPolicy: LayoutPolicy.STRICT
                ],
                storage: [
                    blobStoreName: 'default',
                    strictContentTypeValidation: true,
                    writePolicy: WritePolicy.ALLOW_ONCE
                ]
            ]
        )

        repositoryManager.create(config)
        log.info("Created Maven hosted repository: " + name)
    } else {
        log.info("Maven hosted repository already exists: " + name)
    }
}

def createMavenGroup(String name, List<String> memberNames) {
    def existingRepo = repositoryManager.get(name)
    if (existingRepo == null) {
        log.info("Creating Maven group repository: " + name)

        Configuration config = new Configuration(
```

```
            repositoryName: name,
            recipeName: 'maven2-group',
            online: true,
            attributes: [
                maven: [
                    versionPolicy: VersionPolicy.MIXED,
                    layoutPolicy: LayoutPolicy.STRICT
                ],
                group: [
                    memberNames: memberNames
                ],
                storage: [
                    blobStoreName: 'default',
                    strictContentTypeValidation: true
                ]
            ]
        )

        repositoryManager.create(config)
        log.info("Created Maven group repository: " + name)
    } else {
        log.info("Maven group repository already exists: " + name)
    }
}

// Create Docker repositories
def createDockerHosted(String name, int httpPort) {
    def existingRepo = repositoryManager.get(name)
    if (existingRepo == null) {
        log.info("Creating Docker hosted repository: " + name)

        Configuration config = new Configuration(
            repositoryName: name,
            recipeName: 'docker-hosted',
            online: true,
            attributes: [
                docker: [
                    httpPort: httpPort,
```

```
                httpsPort: null,
                forceBasicAuth: true,
                v1Enabled: false
            ],
            storage: [
                blobStoreName: 'default',
                strictContentTypeValidation: true,
                writePolicy: WritePolicy.ALLOW
            ]
        ]
    )

    repositoryManager.create(config)
    log.info("Created Docker hosted repository: " + name)
  } else {
    log.info("Docker hosted repository already exists: " + name)
  }
}

// Create repositories
try {
  // Maven repositories
  createMavenProxy('maven-central', 'https://repo1.maven.org/maven2/')
  createMavenProxy('maven-google', 'https://maven.google.com/')
  createMavenProxy('gradle-plugins', 'https://plugins.gradle.org/m2/')

  createMavenHosted('maven-releases')
  createMavenHosted('maven-snapshots')

  createMavenGroup('maven-public', ['maven-releases', 'maven-snapshots',

  // Docker repositories
  createDockerHosted('docker-hosted', 8082)

  log.info("Repository initialization completed successfully")

} catch (Exception e) {
```

```
    log.error("Error during repository initialization: " + e.getMessage(), e)
}
```

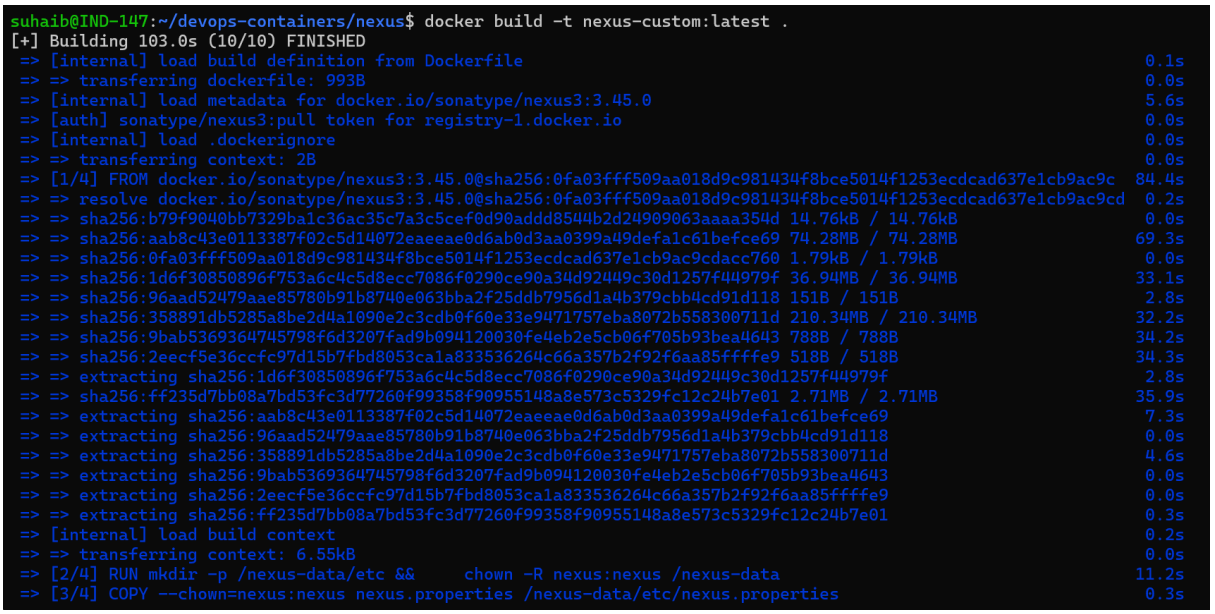
4. **Create nexus.properties:**

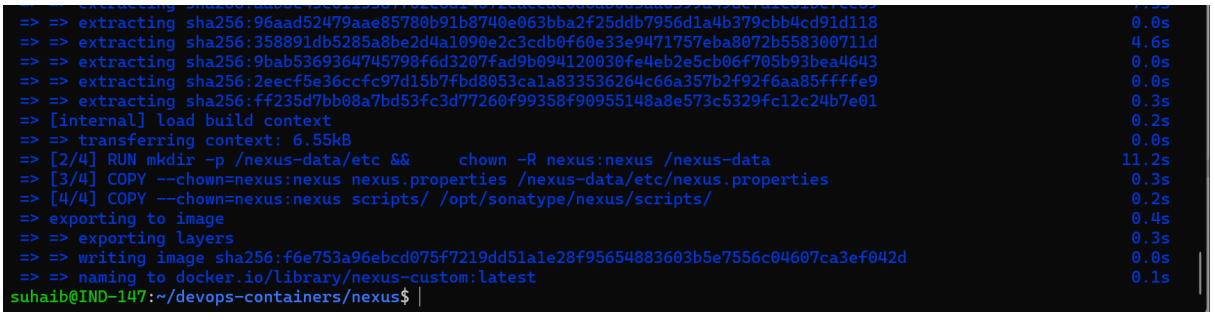
```
# Nexus Repository Manager Configuration
# Production-ready settings

# Data directory
nexus-args=${jetty.etc}/jetty.xml,${jetty.etc}/jetty-http.xml,${jetty.etc}/jetty-re
nexus-context-path=/

# HTTP settings
application-port=8081
application-host=0.0.0.0

# Security settings
nexus.security.randompassword=false
nexus.security.userSource=default

# Performance settings
nexus.scripts.allowCreation=true
nexus.cleanup.retainDays=30

# Database settings (using H2 by default)
nexus.datastore.enabled=true
nexus.datastore.nexus.name=nexus
nexus.datastore.nexus.type=jdbc
nexus.datastore.nexus.jdbcUrl=jdbc:h2:file:./sonatype-work/nexus3/db/nexus

# Logging
nexus.log.level=INFO

# Clustering (disabled for single instance)
nexus.clustered=false

# Blob store settings
nexus.blobstore.quota.warnOnPercentage=80
```

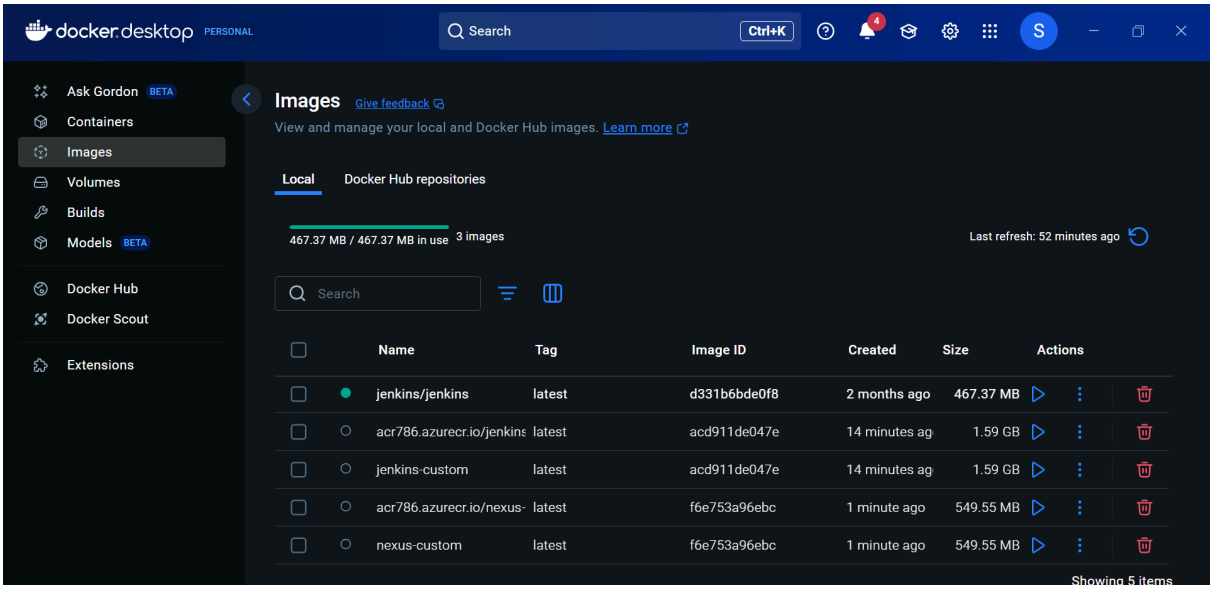
```
nexus.blobstore.quota.errorOnPercentage=90

# Task settings
nexus.quartz.jobStore.isClustered=false

# System user
nexus.security.systemUser=admin
```

5. **Build Nexus image:**

```
docker build -t nexus-custom:latest .
```



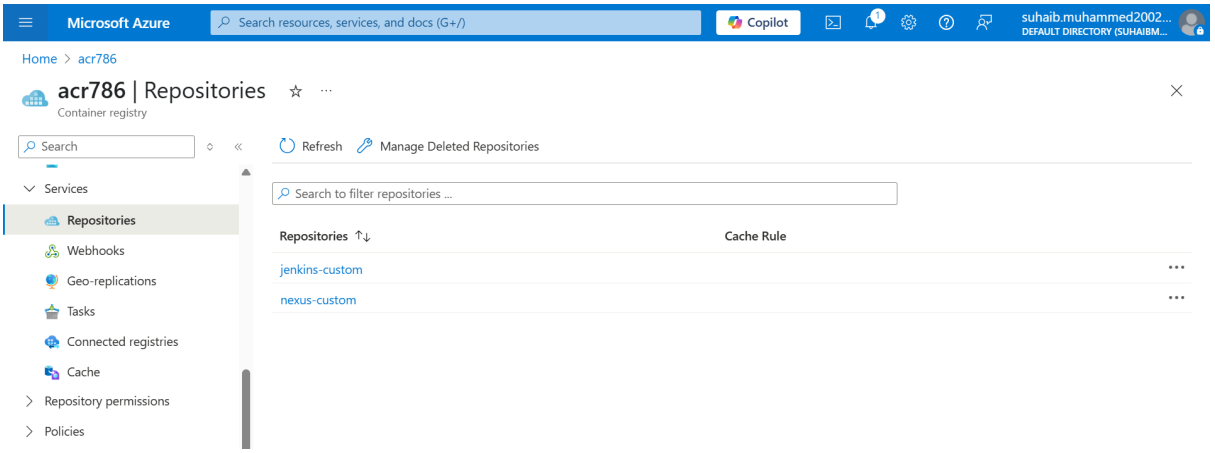




6. **Tag and push to ACR:**

```
docker tag nexus-custom:latest ${ACR_LOGIN_SERVER}/nexus-custom:latest
docker push ${ACR_LOGIN_SERVER}/nexus-custom:latest
```

```
suhaib@IND-147:~/devops-containers/nexus$ docker tag nexus-custom:latest ${ACR_LOGIN_SERVER}/nexus-custom:latest
suhaib@IND-147:~/devops-containers/nexus$ docker push ${ACR_LOGIN_SERVER}/nexus-custom:latest
The push refers to repository [acr786.azurecr.io/nexus-custom]
146809157194: Pushed
fc37c8ee8297: Pushed
843d86bde672: Pushed
46e0ebe35cf5: Pushed
b4b58f37e833: Pushed
bd7bb9fea8ed: Pushed
b0e0e2d07b9e: Pushed
c8e786974da2: Pushed
1352aff765be: Pushed
00a3d7f85a72: Pushed
latest: digest: sha256:6cde6cc756aec63b74487e36abc620960c4c960680125baa33785aa013392634 size: 2409
suhaib@IND-147:~/devops-containers/nexus$
```



Step 4: SonarQube Container Preparation

1. **Navigate to SonarQube directory:**

```
cd ../sonarqube
```

2. **Create Dockerfile** (see Deliverables section below)

```
# SonarQube Dockerfile - Production Ready
FROM sonarqube:10.3-community

# Switch to root for setup
USER root

# Create necessary directories
RUN mkdir -p /opt/sonarqube/conf && \
    mkdir -p /opt/sonarqube/data && \
    mkdir -p /opt/sonarqube/logs && \
    mkdir -p /opt/sonarqube/extensions/plugins && \
    chown -R sonarqube:sonarqube /opt/sonarqube

# Switch back to sonarqube user
USER sonarqube

# Copy custom configuration
COPY --chown=sonarqube:sonarqube sonar.properties /opt/sonarqube/conf/

# Set JVM options for production
ENV SONAR_JAVA_PATH="/opt/java/openjdk/bin/java"
ENV SQ_JAVA_OPTS="-Xmx2g -Xms1g -XX:+UseG1GC -XX:+UseContainerSu

# Configure SonarQube
ENV SONAR_WEB_HOST="0.0.0.0"
ENV SONAR_WEB_PORT="9000"
ENV SONAR_WEB_CONTEXT=""

# Health check
HEALTHCHECK --interval=30s --timeout=15s --start-period=5m --retries=3 \
    CMD curl -f http://localhost:9000/api/system/status | grep -q '"status":"UP'
```

```
# Expose port
EXPOSE 9000

# Volume for data persistence
VOLUME ["/opt/sonarqube/data", "/opt/sonarqube/logs", "/opt/sonarqube/ext

# Use default entrypoint from base image
```

3. **Create configuration files** sonar.properties**:**

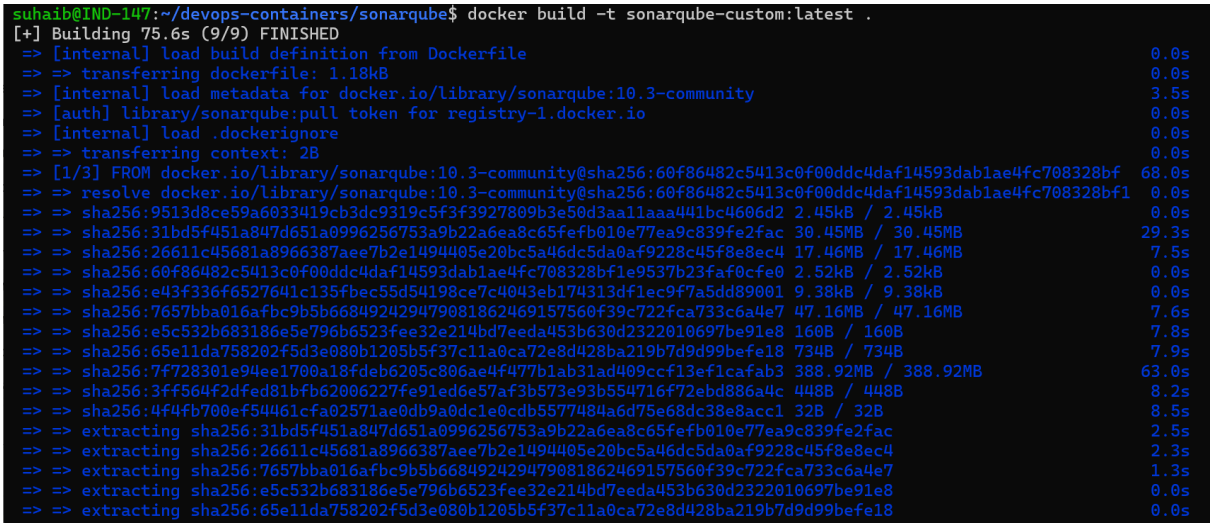
```
# SonarQube Configuration
# Production-ready settings

# Web server settings
sonar.web.host=0.0.0.0
sonar.web.port=9000
sonar.web.context=

# Database settings (using embedded H2 by default)
# For production, consider using PostgreSQL or Oracle
sonar.jdbc.url=jdbc:h2:tcp://localhost:9092/sonar;NON_KEYWORDS=VALUE
sonar.jdbc.username=sonar
sonar.jdbc.password=sonar

# Path settings
sonar.path.data=/opt/sonarqube/data
sonar.path.temp=/opt/sonarqube/temp
sonar.path.logs=/opt/sonarqube/logs

# Elasticsearch settings
sonar.search.host=127.0.0.1
sonar.search.port=9001
sonar.search.javaOpts=-Xmx1g -Xms1g -XX:MaxDirectMemorySize=256m -X

# Security settings
sonar.forceAuthentication=true
sonar.security.realm=sonar
```

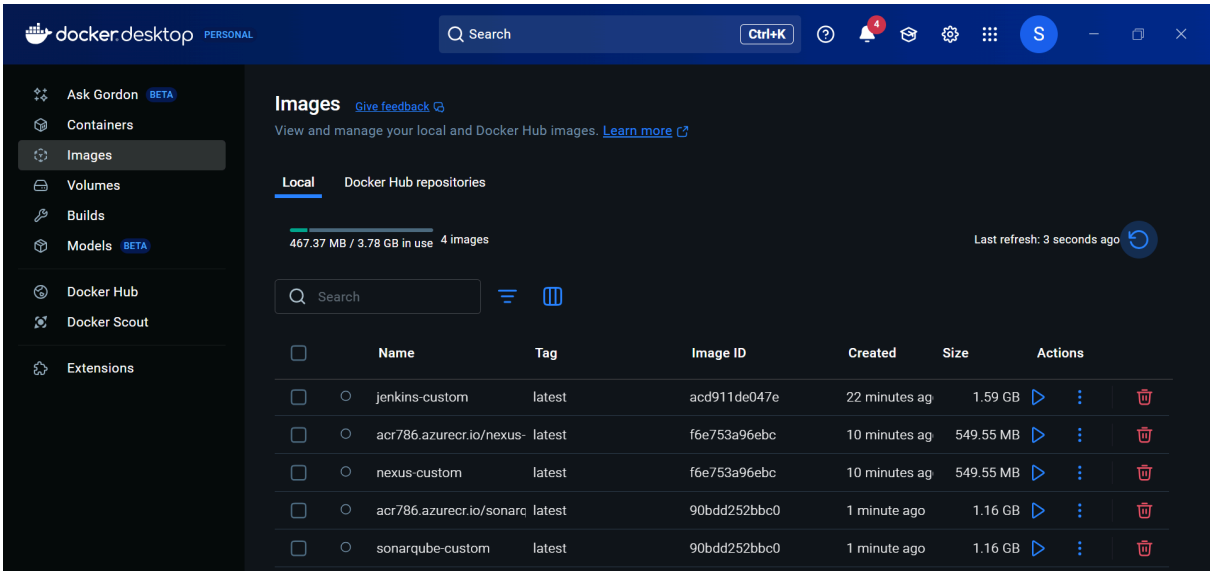
```
# Performance settings
sonar.ce.javaOpts=-Xmx2g -Xms1g -XX:+UseG1GC
sonar.web.javaOpts=-Xmx2g -Xms1g -XX:+UseG1GC

# Logging
sonar.log.level=INFO
sonar.path.logs=/opt/sonarqube/logs

# Update center
sonar.updatecenter.activate=true

# Technical debt settings
sonar.technicalDebt.hoursInDay=8
sonar.technicalDebt.developmentCost=30

# Quality gate settings
sonar.qualitygate.ignoreSmallFiles=true

# Analysis settings
sonar.analysis.detectedLanguages=java,javascript,typescript,python,csharp,g

# Plugin settings
sonar.plugins.risk.consent=ACCEPTED

# LDAP settings (uncomment and configure if using LDAP)
# sonar.security.realm=LDAP
# ldap.url=ldap://localhost:10389
# ldap.bindDn=cn=sonar,ou=users,o=mycompany
# ldap.bindPassword=sonar
# ldap.user.baseDn=ou=users,o=mycompany
# ldap.user.request=(&(objectClass=inetOrgPerson)(uid={login}))
# ldap.user.realNameAttribute=cn
# ldap.user.emailAttribute=mail
# ldap.group.baseDn=ou=groups,o=mycompany
# ldap.group.request=(&(objectClass=posixGroup)(memberUid={uid}))

# Email settings (configure for notifications)
# email.smtp_host.secured=localhost
```

```
# email.smtp_port.secured=587
# email.smtp_username.secured=
# email.smtp_password.secured=
# email.from=noreply@sonarqube.company.com
# email.prefix=[SONARQUBE]

# License (for commercial editions)
# sonar.license.secured=

# Compute Engine settings
sonar.ce.workerCount=2
```

4. **Build SonarQube image:**

```
docker build -t sonarqube-custom:latest .
```

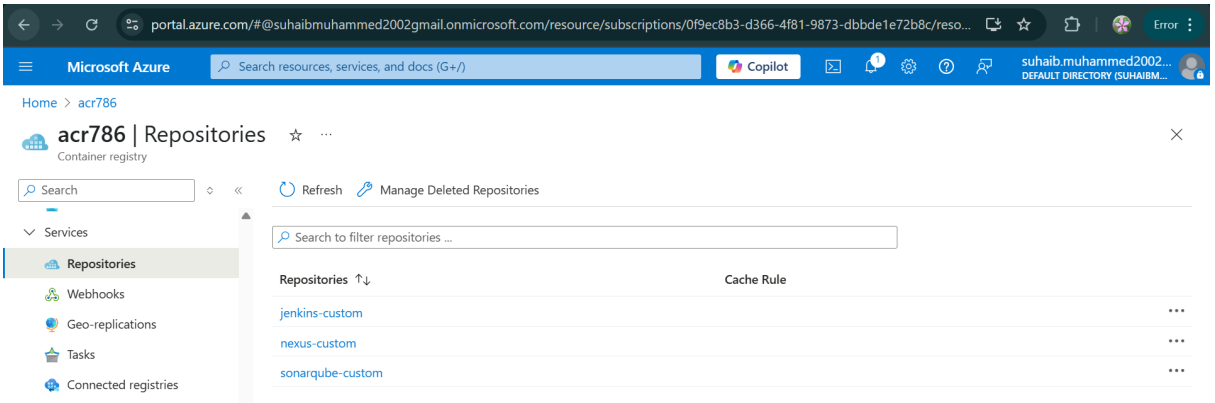




5. **Tag and push to ACR:**

```
docker tag sonarqube-custom:latest ${ACR_LOGIN_SERVER}/sonarqube-custom:latest
docker push ${ACR_LOGIN_SERVER}/sonarqube-custom:latest
```

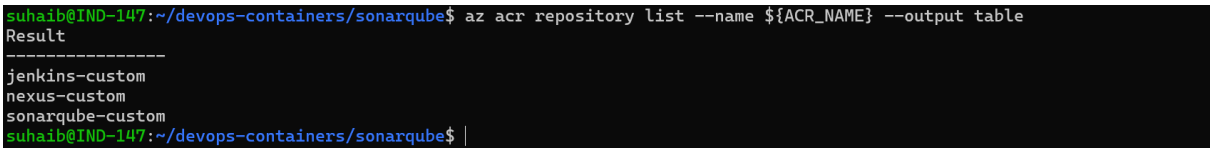
```
suhaib@IND-147:~/devops-containers/sonarqube$ docker tag sonarqube-custom:latest ${ACR_LOGIN_SERVER}/sonarqube-custom:latest
suhaib@IND-147:~/devops-containers/sonarqube$ docker push ${ACR_LOGIN_SERVER}/sonarqube-custom:latest
The push refers to repository [acr786.azurecr.io/sonarqube-custom]
4872aaf18dc6: Pushed
132bbc454edf: Pushed
5f70bf18a086: Pushed
7974cf046f1e: Pushed
207c8089a5dd: Pushed
5dd694b04e2e: Pushed
8fec61bdd63c: Pushed
0c4355b14662: Pushed
ab995379f7a6: Pushed
1a102d1cac2b: Pushed
latest: digest: sha256:e346f12617fa8cf219054c78d6a38e198de9494e34bac6fdf6d6cfcbab099a38 size: 2415
suhaib@IND-147:~/devops-containers/sonarqube$
```



Step 5: Verification

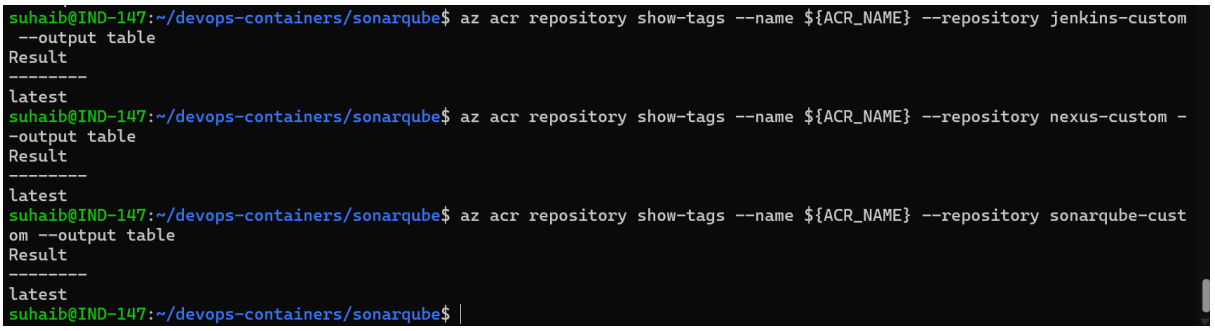
1. **Verify images in ACR:**

```
az acr repository list --name ${ACR_NAME} --output table
```



2. **Check image details:**

```
az acr repository show-tags --name ${ACR_NAME} --repository jenkins-cu
stom --output table
az acr repository show-tags --name ${ACR_NAME} --repository nexus-cus
tom --output table
az acr repository show-tags --name ${ACR_NAME} --repository sonarqube
-custom --output table
```



Security Considerations

Image Security Best Practices Applied:

- **Non-root users**: All containers run with dedicated non-root users
- **Minimal base images**: Using official slim/alpine variants where possible
- **Layer optimization**: Commands combined to reduce layers
- **Secret management**: No hardcoded secrets in images
- **Vulnerability scanning**: Regular base image updates

Security Scanning Commands:

```
# Scan images for vulnerabilities
docker scout cves jenkins-custom:latest
docker scout cves nexus-custom:latest
docker scout cves sonarqube-custom:latest
```

Optimization Notes

Size Optimization:

- Multi-stage builds used where applicable
- Package cache cleanup in same RUN layer
- Only essential packages installed
- Unused files and directories removed

Performance Optimization:

- Pre-configured with optimal JVM settings
- Essential plugins/repositories pre-installed
- Proper health checks implemented

Troubleshooting

Common Issues and Solutions:

1. **Build failures due to network timeouts:**

```
docker build --network=host -t image-name .
```

2. **Permission issues:**

```
# Ensure proper user permissions in Dockerfile
# Use --chown flag in COPY commands
```

3. **ACR authentication issues:**

```
az acr login --name ${ACR_NAME}
```

```
# Or use service principal authentication
```

4. **Large image sizes:**

```
# Use docker system prune to clean up
docker system prune -a
# Analyze image layers
docker history image-name:latest
```