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

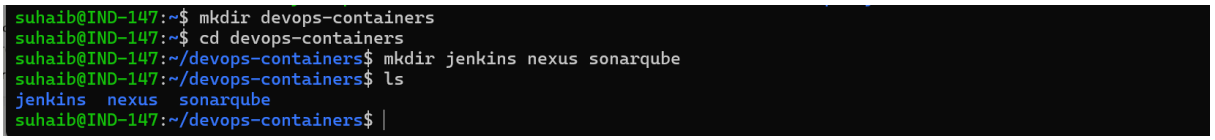
- Docker installed and running
- Azure CLI installed and configured
- Access to Azure Container Registry
- Basic knowledge of Docker and containerization

Step-by-Step Process

Step 1: Environment Setup

1. **Create project directory structure:**

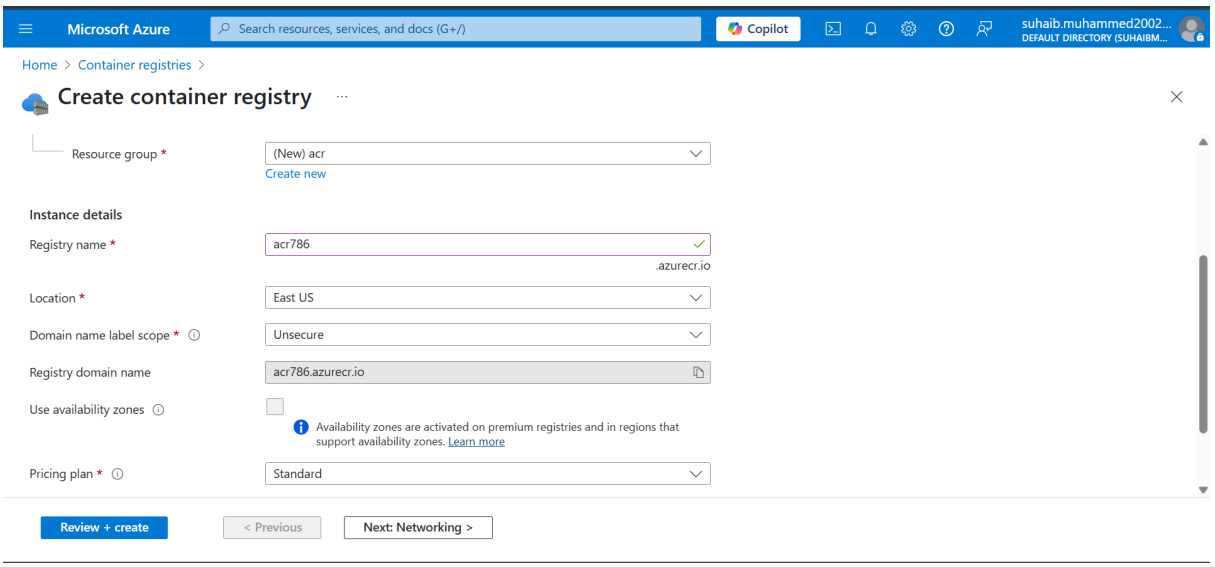
```
mkdir devops-containers
cd devops-containers
mkdir jenkins nexus sonarqube
```



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 git curl && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

# Switch back to jenkins user
USER jenkins

# 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

# 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
```

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

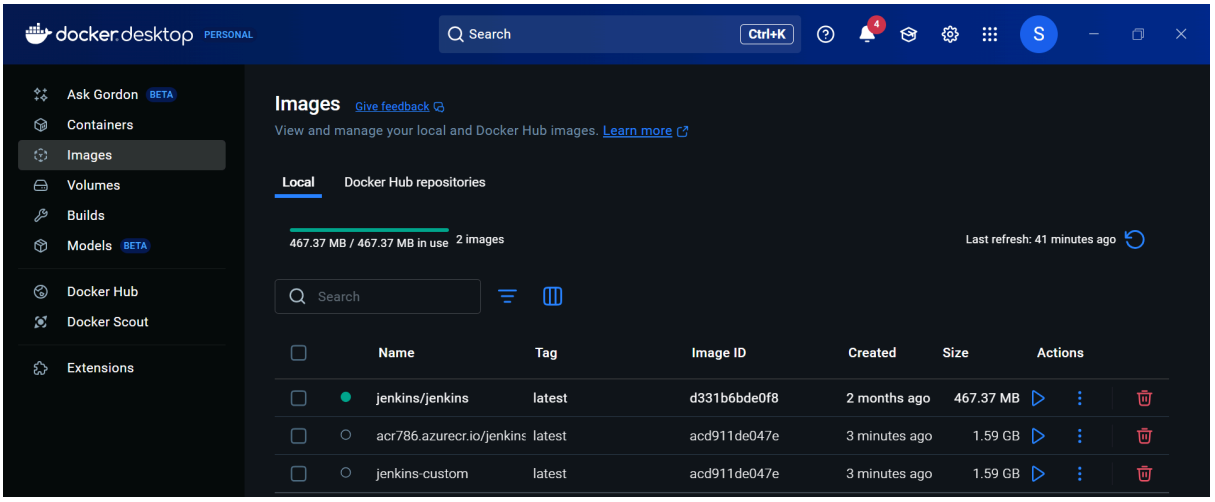
```
# Jenkins Essential Plugins List

# Build Tools
gradle:latest
maven-plugin:latest

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

# Pipeline
workflow-aggregator:latest
pipeline-stage-view:latest
```

4. **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
```

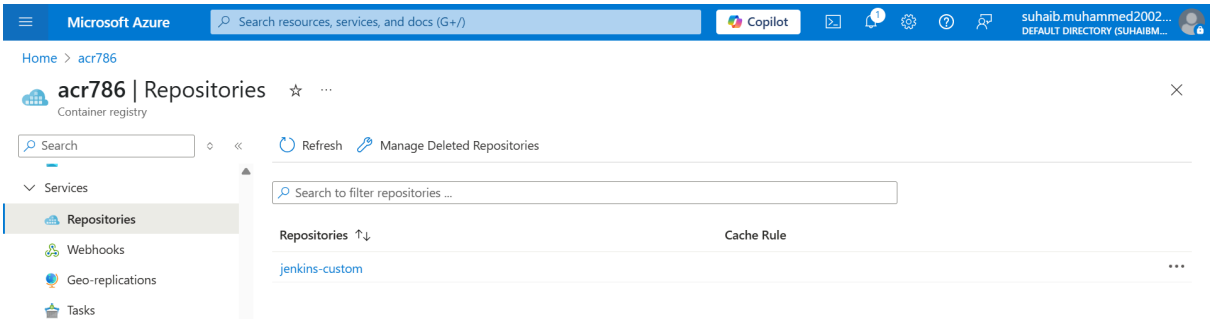
```
 => [6/6] COPY --chown=jenkins:jenkins jenkins.yaml /var/jenkins_home/casc_configs/jenkins.yaml   0.4s
 => exporting to image                                                                  7.6s
 => => exporting layers                                                                 7.4s
 => => writing 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
```

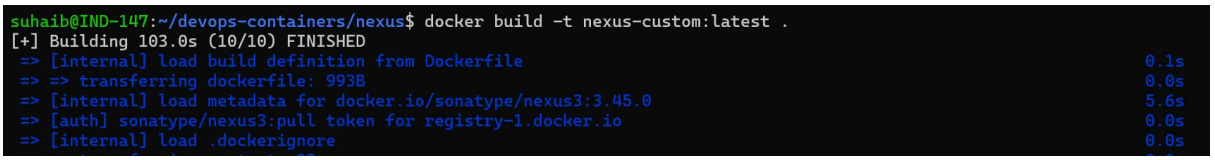
2. **Create Dockerfile**

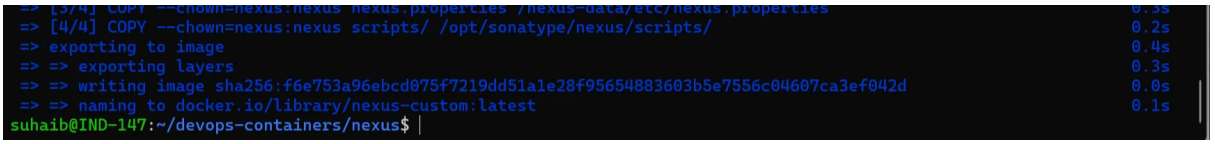
```
# Nexus Repository Manager Dockerfile - Production Ready
FROM sonatype/nexus3:3.45.0

# 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

# 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
```

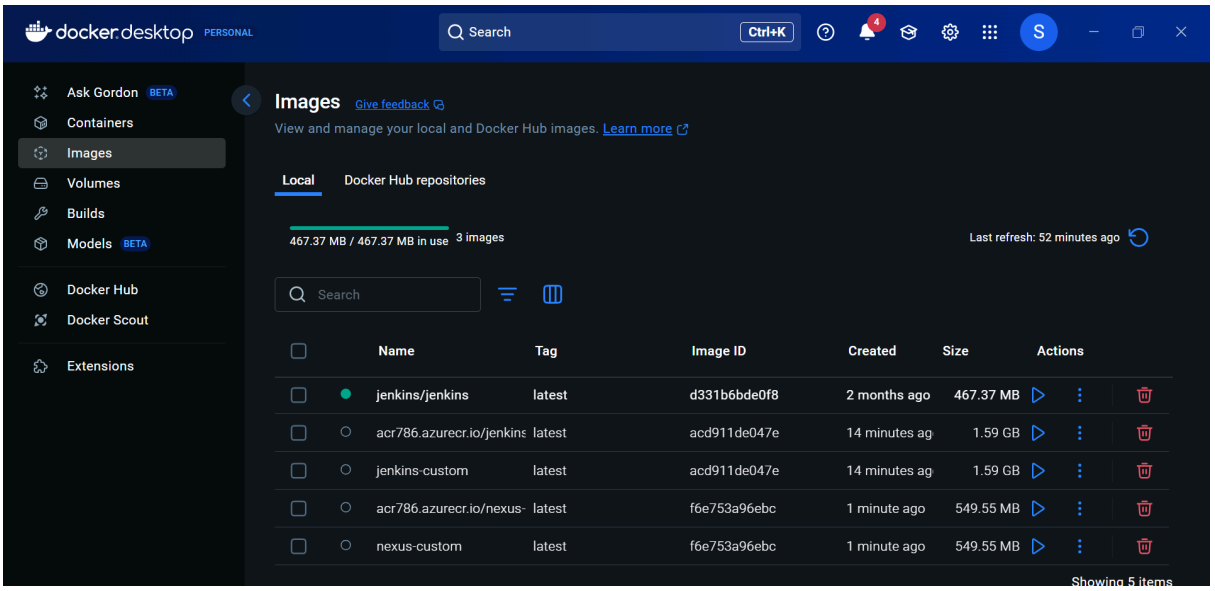
```
# Volume for data persistence
VOLUME ["/nexus-data"]
```

5. **Build Nexus image:**

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



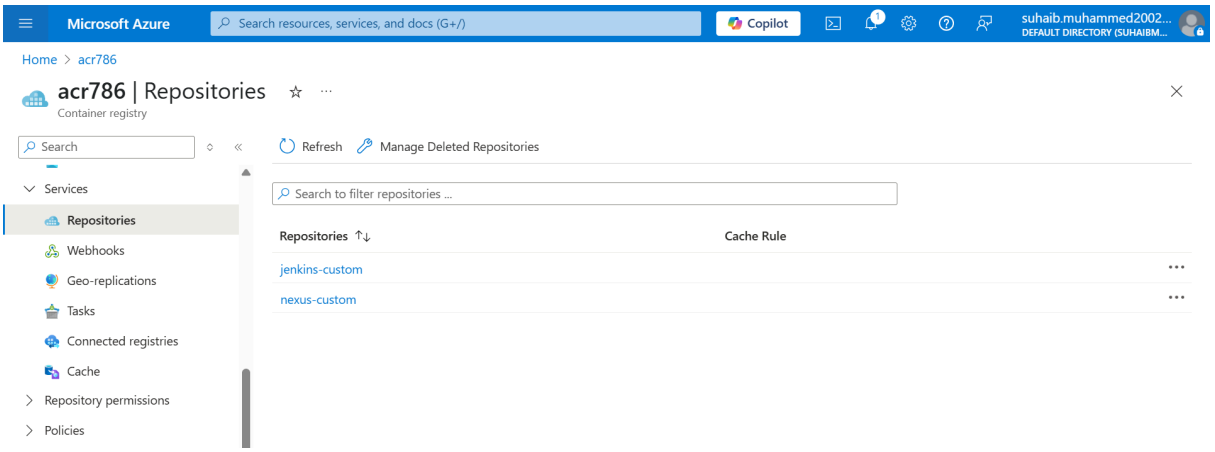




6. **Tag and push to ACR:**

```
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
```

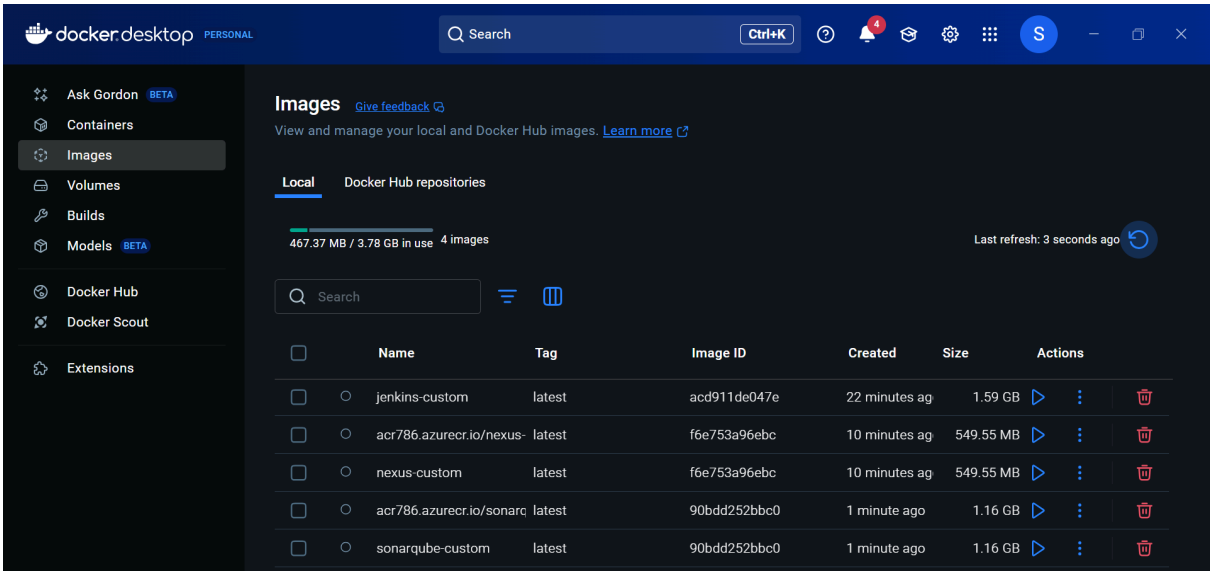
```
# Switch back to sonarqube user
USER sonarqube

# 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
```

4. **Build SonarQube image:**

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

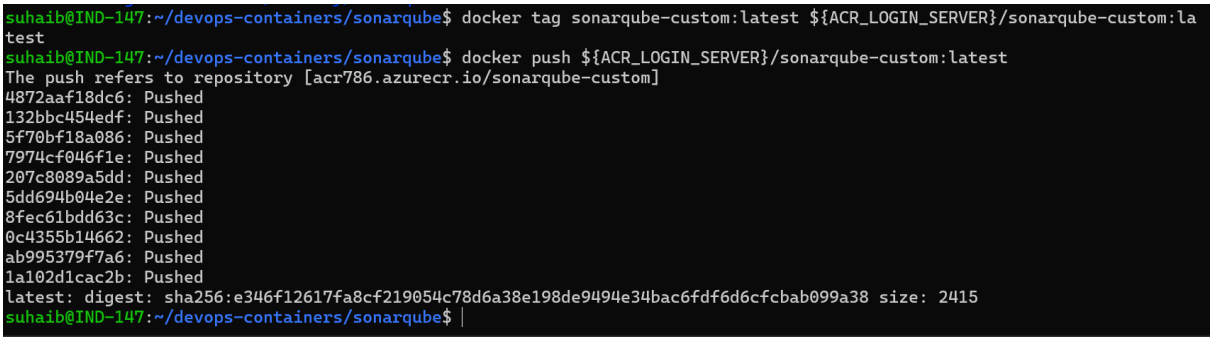
```
suhaib@IND-147:~/devops-containers/sonarqube$ docker build -t sonarqube-custom:latest .
[+] Building 75.6s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 1.18kB                                                                  0.0s
 => [internal] load metadata for docker.io/library/sonarqube:10.3-community                             3.5s
 => [auth] library/sonarqube:pull token for registry-1.docker.io                                        0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [1/3] FROM docker.io/library/sonarqube:10.3-community@sha256:60f86482c5413c0f00ddc4daf14593dab1ae4fc708328bf  68.0s
```

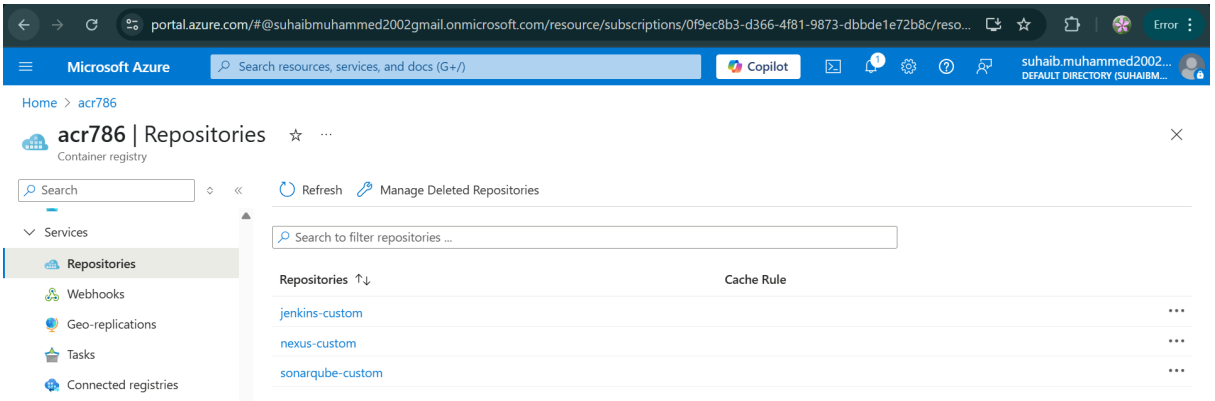
```
 => [3/3] COPY --chown=sonarqube:sonarqube sonar.properties /opt/sonarqube/conf/sonar.properties        0.1s
 => exporting to image                                                                                  1.1s
 => => exporting layers                                                                                 1.1s
 => => writing image sha256:90bdd252bbc04651a0830fc7974f53e33bc2b584ed07dbe9dd4cb81e92765454            0.0s
 => => naming to docker.io/library/sonarqube-custom:latest                                              0.0s
suhaib@IND-147:~/devops-containers/sonarqube$
```



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
```





Step 5: Verification

1. **Verify images in ACR:**

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

```
suhaib@IND-147:~/devops-containers/sonarqube$ az acr repository list --name ${ACR_NAME} --output table
Result
----------------
jenkins-custom
nexus-custom
sonarqube-custom
suhaib@IND-147:~/devops-containers/sonarqube$
```

2. **Check image details:**

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

```
suhaib@IND-147:~/devops-containers/sonarqube$ az acr repository show-tags --name ${ACR_NAME} --repository jenkins-custom
 --output table
Result
--------
latest
suhaib@IND-147:~/devops-containers/sonarqube$ az acr repository show-tags --name ${ACR_NAME} --repository nexus-custom -
-output table
Result
--------
latest
suhaib@IND-147:~/devops-containers/sonarqube$ az acr repository show-tags --name ${ACR_NAME} --repository sonarqube-cust
om --output table
Result
--------
latest
suhaib@IND-147:~/devops-containers/sonarqube$
```

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
```