MDA Multi-Arch CI/CD - Base Image & App Image Standards

This document contains industry-standard, practical examples and ready-to-use artifacts for building **multi-architecture base images** (DevOps team) and **app images** (testers/app teams), promoting images across environments (gate1, staging, live), and deploying/rolling back microservices using Kubernetes. Everything below is designed for Jenkins + Docker Buildx + AWS ECR, but can be adapted to other registries/CI.

Table of Contents

---

1) Goals & Roles

- **DevOps team**: Build and maintain a secure, well-tested *base image* (PHP + Apache + common extensions + wkhtmltopdf + ClamAV, etc.). Publish immutable versioned images and environment promotion tags.
- **Testers / App teams**: Use base image as `FROM <registry>/umbrellaappbaseimg:<pin>` and build app images with code changes. Tag app images per-environment and per-build. Do not rebuild base inside app pipeline.
- **Environments**: `gate1` (integration / gate), `staging` (QA), `live` (production).

---

2) Base Image Best Practices (high level)

- Keep base image minimal. Use official upstream `php:8.4-apache` as base when possible.
- Make image reproducible: include `ARG BUILDDATE` and labels: `org.opencontainers.image.*`.
- Use `docker buildx` to produce multi-arch manifests for `linux/amd64,linux/arm64`.
- Pin versions of apt packages where possible and verify upstream sources.

- Use non-root runtime user. Avoid running application as `root`.
- Separate build-time and runtime dependencies via multi-stage builds (where applicable).
- Hardening: remove package manager caches, set `UMASK`, minimize SUID binaries.
- Supply configuration via mounted files (do not bake secrets in images).
- Provide small entrypoint to handle permissions, db migration hooks, health-check helper.
- Sign and scan images prior to promotion.

---

3) Improved `Dockerfile.base` (multi-arch, hardened)

File: `deployment/docker_build/Dockerfile.base`

```
# syntax=docker/dockerfile:experimental
FROM --platform=$TARGETPLATFORM php:8.4-apache AS base

ARG TARGETPLATFORM
ARG BUILDPLATFORM
ARG TARGETARCH
ARG BUILD_DATE
ARG VCS_REF

LABEL org.opencontainers.image.created=$BUILD_DATE
      org.opencontainers.image.revision=$VCS_REF
      org.opencontainers.image.authors="devops@company.com"
      org.opencontainers.image.version="v13"

ENV DEBIAN_FRONTEND=noninteractive
    APACHE_DOCUMENT_ROOT=/var/www/html/public
    TZ=UTC

# install runtime deps and PHP extensions
RUN set -eux;
    apt-get update;
    apt-get install -y --no-install-recommends
        ca-certificates
        wget gnupg2 apt-transport-https software-properties-common
        sudo supervisor zip unzip git
        libpng-dev libzip-dev libjpeg62-turbo-dev libfreetype6-dev
        libxslt1-dev libxml2-dev pkg-config openssl xfonts-75dpi xfonts-base
        fonts-dejavu-core fonts-liberation locales tzdata
        clamav clamav-daemon
    ;
    # cleanup
    apt-get clean; rm -rf /var/lib/apt/lists/*

# enable apache modules
```

```
RUN a2enmod rewrite headers

# install php extensions (built-in helper)
RUN docker-php-ext-configure gd --with-jpeg --with-freetype;
    docker-php-ext-install -j"$(nproc)" pdo_mysql mysqli exif gettext pcntl
sysvmsg xsl opcache soap sockets zip gd

# conditional multi-arch wkhtmltopdf and libssl
RUN set -eux;
    case "${TARGETARCH}" in
      amd64)
        WKH_VER="0.12.6.1-2";
        WKH_FILE="wkhtmltox_${WKH_VER}.bullseye_amd64.deb";
        SSL_PKG="libssl1.1_1.1.1w-0+deb11u1_amd64.deb";
        ;;
      arm64)
        WKH_VER="0.12.6.1-2";
        WKH_FILE="wkhtmltox_${WKH_VER}.bullseye_arm64.deb";
        SSL_PKG="libssl1.1_1.1.1w-0+deb11u1_arm64.deb";
        ;;
      *) echo "Unsupported arch ${TARGETARCH}"; exit 1 ;;
    esac;
    cd /tmp;
    wget -q "https://github.com/wkhtmltopdf/packaging/releases/download/$
{WKH_VER}/${WKH_FILE}" -O /tmp/${WKH_FILE};
    # if libssl is required for wkhtmltopdf package or older debs
    wget -q "http://ftp.cn.debian.org/debian/pool/main/o/openssl/${SSL_PKG}" -
O /tmp/${SSL_PKG} || true;
    dpkg -i /tmp/${SSL_PKG} || true;
    dpkg -i /tmp/${WKH_FILE} || true;
    apt-get install -f -y || true;
    rm -f /tmp/${WKH_FILE} /tmp/${SSL_PKG};
    rm -rf /var/lib/apt/lists/*

# install fonts package (example installer may require interactivity) - handle
gracefully
RUN set -eux;
    export DEBIAN_FRONTEND=noninteractive;
    apt-get update;
    apt-get install -y --no-install-recommends ttf-mscorefonts-installer ||
true;
    apt-get clean; rm -rf /var/lib/apt/lists/*

# add a non-root user for runtime
RUN groupadd -g 1000 appuser && useradd -r -u 1000 -g appuser -d /var/www -s /
sbin/nologin appuser;
    mkdir -p /var/www/html/public /var/www/html/storage /var/www/html/bootstrap/
cache;
```

```
    chown -R appuser:appuser /var/www/html

# copy default confs
COPY deployment/docker_build/app/000-default.conf /etc/apache2/sites-available/
000-default.conf
COPY deployment/docker_build/app/security.conf /etc/apache2/conf-available/
security.conf
COPY deployment/docker_build/app/php.ini /usr/local/etc/php/php.ini
COPY deployment/docker_build/app/10-wkhtmltopdf.conf /etc/fonts/conf.d/10-
wkhtmltopdf.conf

# entrypoint (keeps as lightweight script for runtime setup)
COPY deployment/docker_build/docker-entrypoint.sh /usr/local/bin/docker-
entrypoint.sh
RUN chmod +x /usr/local/bin/docker-entrypoint.sh

EXPOSE 80
USER appuser
WORKDIR /var/www/html
ENTRYPOINT ["/usr/local/bin/docker-entrypoint.sh"]
CMD ["apache2-foreground"]
```

Notes: - `USER appuser` ensures containers run as non-root. If you need root at startup for certain init steps, make entrypoint escalate temporarily. - Keep `docker-entrypoint.sh` short: adjust permissions, run migrations (optionally), start services.

---

4) App Dockerfile (testers) — `Dockerfile.app`

File: `deployment/docker_build/Dockerfile.app`

```
FROM 015227858865.dkr.ecr.eu-west-1.amazonaws.com/umbrellaappbaseimg:v13

# Ensure we run as the expected user inside the base image
USER 1000:1000
WORKDIR /var/www/html

# Copy application code only
COPY ./api .
COPY ./api/resources/views/hmrc-inbox public/api/resources/views/hmrc-inbox/
COPY ./deployment/docker_build/app/openssl.cnf /usr/lib/ssl/openssl.cnf

# set file perms for runtime writable paths (done as non-root user if base image
gave sudo)
# If base image does not include sudo, rely on Dockerfile's USER root for chown
in build time
```

```
USER root
RUN chown -R 1000:1000 /var/www/html && chmod -R 750 /var/www/html
USER 1000

# small healthcheck
HEALTHCHECK --interval=30s --timeout=3s --start-period=30s --retries=3
  CMD curl -f http://localhost/ || exit 1
```

---

5) Jenkins pipelines

5.1 DevOps pipeline: Build & push base image (multi-arch)

File: `jenkins/base-image/Jenkinsfile`

```
pipeline {
  agent { label 'docker-builder' }
  environment {
    REGISTRY = '015227858865.dkr.ecr.eu-west-1.amazonaws.com'
    IMAGE_NAME = 'umbrellaappbaseimg'
    IMAGE_VERSION = 'v13'
    DOCKER_BUILDX = 'docker-buildx'
  }

  stages {
    stage('Checkout') { steps { checkout scm } }

    stage('Login to ECR') {
      steps {
        withAWS(region: 'eu-west-1', credentials: 'aws-ecr-credentials') {
          sh 'aws ecr get-login-password --region eu-west-1 | docker login --
username AWS --password-stdin ${REGISTRY}'
        }
      }
    }

    stage('Setup buildx') {
      steps {
        sh '''#!/bin/bash
          docker buildx create --use --name multi || true
          docker buildx inspect --bootstrap
        '''
      }
    }
```

```
    stage('Build & Push Multi-arch') {
      steps {
        sh '''#!/bin/bash
          BUILD_DATE=$(date --utc +%Y-%m-%dT%H:%M:%SZ)
          VCS_REF=$(git rev-parse --short HEAD)

          docker buildx build --push
            --platform linux/amd64,linux/arm64
            --build-arg BUILD_DATE=${BUILD_DATE}
            --build-arg VCS_REF=${VCS_REF}
            -t ${REGISTRY}/${IMAGE_NAME}:${IMAGE_VERSION}
            -t ${REGISTRY}/${IMAGE_NAME}:${IMAGE_VERSION}-${VCS_REF}
            deployment/docker_build -f deployment/docker_build/Dockerfile.base
        '''
      }
    }

    stage('Optional: Tag to env (manual)') {
      when { expression { return params.PROMOTE_TO != null &&
params.PROMOTE_TO != '' } }
      steps {
        script {
          env.PROMOTE = params.PROMOTE_TO
        }
        sh '''#!/bin/bash
          # re-tag the built image to environment (e.g., gate1, staging, live) -
this is done only after validation
          aws ecr get-login-password --region eu-west-1 | docker login --
username AWS --password-stdin ${REGISTRY}
          docker pull ${REGISTRY}/${IMAGE_NAME}:${IMAGE_VERSION}
          docker tag ${REGISTRY}/${IMAGE_NAME}:${IMAGE_VERSION} ${REGISTRY}/$
{IMAGE_NAME}:${PROMOTE}
          docker push ${REGISTRY}/${IMAGE_NAME}:${PROMOTE}
        '''
      }
    }
  }
}
```

Note: Promotion stage should usually be gated by manual approval step (Jenkins input or an automated policy) and should only be allowed by authorized accounts.

5.2 App pipeline: Build & push app images per environment

File: `jenkins/app-image/Jenkinsfile`

```
pipeline {
  agent { label 'docker-builder' }
  parameters {
    string(name: 'ENV', defaultValue: 'gate1', description: 'deploy env tag:
gate1|staging|live')
  }
  environment {
    REGISTRY = '015227858865.dkr.ecr.eu-west-1.amazonaws.com'
    IMAGE_NAME = 'umbrellaapp'
  }

  stages {
    stage('Checkout') { steps { checkout scm } }

    stage('Login to ECR') {
      steps {
        withAWS(region: 'eu-west-1', credentials: 'aws-ecr-credentials') {
          sh 'aws ecr get-login-password --region eu-west-1 | docker login --
username AWS --password-stdin ${REGISTRY}'
        }
      }
    }

    stage('Build & Push Multi-arch App Image') {
      steps {
        sh '''#!/bin/bash
          GIT_SHA=$(git rev-parse --short HEAD)
          BUILD_TAG=${ENV}-${GIT_SHA}

          docker buildx create --use --name multi || true
          docker buildx inspect --bootstrap

          docker buildx build --push
            --platform linux/amd64,linux/arm64
            -t ${REGISTRY}/${IMAGE_NAME}:${BUILD_TAG}
            -t ${REGISTRY}/${IMAGE_NAME}:${ENV}
            -f deployment/docker_build/Dockerfile.app .

          echo "Pushed ${REGISTRY}/${IMAGE_NAME}:${BUILD_TAG} and ${REGISTRY}/$
{IMAGE_NAME}:${ENV}"
        '''
      }
    }

    stage('Register Build Metadata') {
      steps {
        // optional: push metadata to artifact tracker, notify teams, create
```

```
release note
        echo 'Register build metadata (optional)'
      }
    }
  }
}
```

---

6) Tagging & Promotion Strategy (recommended)

- **Base image**: `umbrellaappbaseimg:v13`, `umbrellaappbaseimg:v13-<gitsha>`; promoted tags `umbrellaappbaseimg:gate1`, `:staging`, `:live` (manually promoted after tests).
- **App image**: `umbrellaapp:gate1-<gitsha>`, `umbrellaapp:staging-<gitsha>`, `umbrellaapp:live-<gitsha>` plus environment alias tags `umbrellaapp:gate1` etc.
- Never rely on `latest` for environments. Use environment-specific alias tags controlled by the CI promotion pipeline.

---

7) Kubernetes manifests + Kustomize overlays

base: `k8s/base/deployment.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: umbrellaapp
  labels:
    app: umbrellaapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: umbrellaapp
  template:
    metadata:
      labels:
        app: umbrellaapp
    spec:
      containers:
        - name: umbrellaapp
          image: 015227858865.dkr.ecr.eu-west-1.amazonaws.com/
umbrellaapp:REPLACEME
          ports:
            - containerPort: 80
          readinessProbe:
```

```
            httpGet:
              path: /
              port: 80
            initialDelaySeconds: 10
            periodSeconds: 10
          livenessProbe:
            httpGet:
              path: /health
              port: 80
            initialDelaySeconds: 30
            periodSeconds: 20
```

overlay: `k8s/overlays/live/kustomization.yaml`

```
resources:
  - ../../base
images:
  - name: 015227858865.dkr.ecr.eu-west-1.amazonaws.com/umbrellaapp
    newTag: live-35
```

Deploy using:

```
kubectl apply -k k8s/overlays/live
```

Rollback: set `newTag: live-34` and re-apply, or use `kubectl rollout undo deployment/umbrellaapp -n live`.

---

8) Makefile (devops/tester helper)

```
REG=015227858865.dkr.ecr.eu-west-1.amazonaws.com
APP=umbrellaapp
BASE=umbrellaappbaseimg

.PHONY: build-base push-base build-app push-app deploy-live deploy-staging
promote-base rollback-live

build-base:
    docker buildx create --use || true
    docker buildx build --platform linux/amd64,linux/arm64 -t ${REG}/$
{BASE}:v13 -f deployment/docker_build/Dockerfile.base deployment/docker_build --
load
```

```
push-base:
    aws ecr get-login-password --region eu-west-1 | docker login --username AWS
--password-stdin ${REG}
    docker buildx build --platform linux/amd64,linux/arm64 -t ${REG}/$
{BASE}:v13 -t ${REG}/${BASE}:v13-$(shell git rev-parse --short HEAD) --push -f
deployment/docker_build/Dockerfile.base deployment/docker_build

build-app:
    docker buildx create --use || true
    docker buildx build --platform linux/amd64,linux/arm64 -t ${REG}/$
{APP}:local -f deployment/docker_build/Dockerfile.app . --load

push-app:
    aws ecr get-login-password --region eu-west-1 | docker login --username AWS
--password-stdin ${REG}
    docker buildx build --platform linux/amd64,linux/arm64 -t ${REG}/${APP}:$
(ENV)-$(shell git rev-parse --short HEAD) -t ${REG}/${APP}:$(ENV) --push -f
deployment/docker_build/Dockerfile.app .

promote-base:
    # re-tag a base image to an environment (manual, requires permission)
    aws ecr get-login-password --region eu-west-1 | docker login --username AWS
--password-stdin ${REG}
    docker pull ${REG}/${BASE}:v13
    docker tag ${REG}/${BASE}:v13 ${REG}/${BASE}:$(ENV)
    docker push ${REG}/${BASE}:$(ENV)

deploy-live:
    kubectl apply -k k8s/overlays/live

rollback-live:
    kubectl rollout undo deployment/umbrellaapp -n live
```

---

9) Security & Quality Gates

- **Image scanning**: Integrate Snyk/Trivy/Clair into Jenkins pipeline. Fail build for critical CVEs.
- **Image signing**: Use cosign/notation to sign images post-build. Validate signatures during promotion.
- **Vulnerability policy**: block promotion for images with high/critical CVEs.
- **Immutable tags**: never overwrite tags like `live-<sha>` or `v13-<sha>`.
- **RBAC**: restrict who can promote tags or create `live` alias.
- **Secrets**: use K8s Secrets or external secret manager (avoid baking secrets in images).

---

10) Operational notes & troubleshooting

- If wkhtmltopdf fails on some arch, maintain a pinned artifact repo or build wkhtmltopdf from source in CI for that arch.
- For base-image changes: bump `v13 -> v14` and run a smoke test job that deploys to a disposable `gate1-smoke` namespace to validate before promoting.
- Monitor image sizes; avoid heavy packages in base image if not needed by most apps.

Appendix: Entry-point example

`deployment/docker_build/docker-entrypoint.sh`

```
#!/bin/bash
set -e

# set correct permissions if mounted volumes exist
chown -R appuser:appuser /var/www/html || true

# optionally run migrations if env var
if [ "$APP_MIGRATE" = "true" ]; then
  echo "Running migrations"
  # php artisan migrate --force
fi

exec "$@"
```

Next steps for your team

1. Save base `Dockerfile.base` into `deployment/docker_build/` and the `Dockerfile.app` for app teams.
2. Configure Jenkins builders with `docker` and `docker buildx` available (or use a privileged builder pool).
3. Add image scanning and signing steps in pipeline.
4. Use the Makefile and Kustomize overlays for fast local iteration and production deployment.

If you want, I can now: - produce a Helm chart variant of the k8s deployment, - create a Jenkins `promotion` job (UI + approval step) with RBAC policy snippets, - or generate a minimal Trivy/Snyk scan step to add into Jenkinsfile.

End of document.