# === Clone v0.7.0 and start with Docker Compose ===

set -e

REPO\_URL="https://github.com/taoc3860-pixel/cloudbooking.git"

GIT\_TAG="v0.6.0"

APP\_DIR="/opt/app/cloudbooking"

# 1) Free port 80 (optional) and get code @ tag

sudo systemctl stop nginx >/dev/null 2>&1 || true

sudo systemctl disable nginx >/dev/null 2>&1 || true

sudo rm -rf "$APP\_DIR"; sudo mkdir -p "$(dirname "$APP\_DIR")"

sudo git clone --depth=1 "$REPO\_URL" "$APP\_DIR" >/dev/null

cd "$APP\_DIR"

git fetch --tags -q

git checkout -f "tags/${GIT\_TAG}" -q

# 2) Minimal .env (keep if already exists)

[ -f .env ] || printf 'JWT\_SECRET=mySuperSecretKey\nPORT=5055\nNODE\_ENV=production\nMONGODB\_URI=\n' | sudo tee .env >/dev/null

# 3) Start (auto-detect docker compose)

if docker compose version >/dev/null 2>&1; then

docker compose up -d --build

else

docker-compose up -d --build

fi

# 4) Show status

if docker compose version >/dev/null 2>&1; then

docker compose ps

else

docker-compose ps

fi

Token

ghp\_bbbltm0CueFLMr66odzwhSTtfpEwTD2t222t

echo ghp\_bbbltm0CueFLMr66odzwhSTtfpEwTD2t222t | docker login ghcr.io -u taoc3860-pixel --password-stdin

# ===== CloudBooking v0.7.0: fault-tolerant compose up (paste & run) =====

set +e

REPO\_URL="https://github.com/taoc3860-pixel/cloudbooking.git"

APP\_DIR="/opt/app/cloudbooking"

GIT\_TAG="v0.7.0"

bold(){ printf "\033[1m%s\033[0m\n" "$\*"; }

ok(){ echo "✅ $\*"; }

warn(){ echo "⚠️ $\*"; }

err(){ echo "❌ $\*"; }

step(){ bold "$1"; shift; eval "$@"; rc=$?; [ $rc -ne 0 ] && warn "返回码=$rc（继续）"; }

# 0) 依赖提示（不退出）

for c in docker; do command -v $c >/dev/null 2>&1 || err "缺少依赖：$c"; done

if docker compose version >/dev/null 2>&1; then DC="docker compose"; else DC="docker-compose"; fi

# 1) 释放端口 & 拉代码@tag

step "停止系统 nginx（避免80占用）" "sudo systemctl stop nginx >/dev/null 2>&1 || true; sudo systemctl disable nginx >/dev/null 2>&1 || true"

step "重置目录并克隆仓库" "sudo rm -rf \"$APP\_DIR\"; sudo mkdir -p \"\$(dirname \"$APP\_DIR\")\"; sudo git clone --depth=1 \"$REPO\_URL\" \"$APP\_DIR\" >/dev/null 2>&1"

cd "$APP\_DIR" 2>/dev/null || { err "进入目录失败：$APP\_DIR"; }

step "切换到 tag $GIT\_TAG" "git fetch --tags -q; git checkout -f \"tags/$GIT\_TAG\" -q"

# 2) 生成/修补 .env

if [ ! -f .env ]; then

step "创建 .env" "printf 'JWT\_SECRET=%s\nPORT=5055\nNODE\_ENV=production\nMONGODB\_URI=%s\n' \

\"\${JWT\_SECRET:-change\_this\_to\_a\_long\_random\_secret}\" \"\${MONGODB\_URI:-mongodb://mongo:27017/cloudbooking}\" | sudo tee .env >/dev/null"

else

# 若缺少关键项则补齐（不覆盖已有值）

grep -q '^JWT\_SECRET=' .env || echo "JWT\_SECRET=${JWT\_SECRET:-change\_this\_to\_a\_long\_random\_secret}" | sudo tee -a .env >/dev/null

grep -q '^PORT=' .env || echo "PORT=5055" | sudo tee -a .env >/dev/null

grep -q '^NODE\_ENV=' .env || echo "NODE\_ENV=production" | sudo tee -a .env >/dev/null

grep -q '^MONGODB\_URI=' .env || echo "MONGODB\_URI=${MONGODB\_URI:-mongodb://mongo:27017/cloudbooking}" | sudo tee -a .env >/dev/null

fi

# 3) 准备 nginx.conf（若缺）

if [ ! -f nginx.conf ]; then

step "创建 nginx.conf" "cat > nginx.conf <<'NGX'

events {}

http {

server {

listen 80;

root /usr/share/nginx/html;

index index.html;

location /api/ {

proxy\_pass http://app:5055/;

proxy\_set\_header Host \$host;

proxy\_set\_header X-Real-IP \$remote\_addr;

proxy\_http\_version 1.1;

proxy\_set\_header Connection \"\";

}

}

}

NGX"

fi

# 4) 自动探测 Dockerfile 路径

DOCKERFILE="Dockerfile"

[ -f backend/Dockerfile ] && DOCKERFILE="backend/Dockerfile"

ok "使用 Dockerfile: $DOCKERFILE"

# 5) 生成/修补 docker-compose.yml（若缺）

if [ ! -f docker-compose.yml ] && [ ! -f compose.yml ]; then

step "创建 docker-compose.yml" "cat > docker-compose.yml <<'YML'

version: "3.9"

services:

app:

build:

context: .

# 若 Dockerfile 在 backend/，compose 会自动发现；如未识别，可取消注释并设置：

# dockerfile: backend/Dockerfile

container\_name: cloudbooking\_app

env\_file: .env

ports:

- "5055:5055"

depends\_on:

- mongo

restart: always

mongo:

image: mongo:6

container\_name: cloudbooking\_mongo

volumes:

- mongo\_data:/data/db

restart: always

nginx:

image: nginx:alpine

container\_name: cloudbooking\_nginx

ports:

- "8080:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf:ro

- ./web:/usr/share/nginx/html:ro

depends\_on:

- app

restart: always

volumes:

mongo\_data:

YML"

fi

# 若 compose 未能自动识别 Dockerfile，临时覆盖（不破坏文件）：用 --build-arg 指明

BUILD\_ARG=""

[ "$DOCKERFILE" != "Dockerfile" ] && BUILD\_ARG="--build-arg DOCKERFILE\_PATH=$DOCKERFILE"

# 6) 启动（遇错不退出）

step "构建并启动（可能需要几分钟）" "$DC up -d --build || true"

# 7) 显示状态与日志（不退出）

bold "===== Compose 状态 ====="

$DC ps || true

bold "===== 近50行 App 日志（若存在）====="

$DC logs --tail=50 app 2>/dev/null || true

bold "===== 近50行 Nginx 日志（若存在）====="

$DC logs --tail=50 nginx 2>/dev/null || true

IP=$(curl -s ifconfig.me || curl -s ipinfo.io/ip || hostname -I 2>/dev/null | awk '{print $1}')

echo

bold "====== 访问方式 & 排错建议 ======"

echo "打开：http://${IP}:8080/"

echo "- 若网页能开但数据不入库：检查 .env 的 MONGODB\_URI（默认已指向 compose 内的 mongo）。"

echo "- 若 app 未监听 5055：确认代码里使用了 \$PORT 或 5055 启动。"

echo "- 查看实时日志：$DC logs -f app"

echo "- 端口占用检查：sudo ss -lntp | grep -E ':80|:8080|:5055'"

ok "流程结束（即使中间有报错也已尽量继续执行）。"