14

O‘N TO‘RTINCHI AMALIY MASHG‘ULOT

Korxona ilovalarini dasturlash · 2 soat · 14-mavzu

# Docker stek, GitHub Actions va VPS ga joylashtirish

Jamoa loyihasi bosqichi · 5 ta PHP topshiriq · o‘qituvchi bahosi

Docker | Docker Compose | GitHub Actions | nginx | pg_dump

MASHG‘ULOT MAQSADI

# Amaliy mashg‘ulot oxirida siz quyidagilarni bajara olasiz...

1. **Yig‘ish** — OmborPro uchun production image lar (app va web) va to‘liq Compose stekini: web, app, worker, scheduler, db, redis
2. **Avtomatlashtirish** — GitHub Actions quvurini: Pint, Larastan, Pest (PostgreSQL bilan), image ni GHCR ga yuklash va deploy
3. **Joylashtirish** — ilovani VPS serverga: SSH kalit, ufw, Docker, maxfiy `.env`, migratsiya va keshlar
4. **Himoyalash** — HTTPS ni yoqish, db va redis portlarini yopish, maxfiy qiymatlarni Secrets da saqlash
5. **Tekshirish** — ommaviy URL ishlashini (`/up`, kirish, buyurtma yaratish) hamda zaxira va tiklashni

# Qanday bajariladi va qanday baholanadi?

## Ish tartibi

1. Saytga login va parolingiz bilan kiring: **emro.uz/korxona/kirish.html**
2. **14-amaliy** sahifasini oching (oxirgi slayddagi QR kod).
3. Yo‘riqnoma bo‘yicha jamoa loyihasida bosqichni bajaring, **commit** va **push** qiling.
4. PHP topshiriqlarni yeching: ▶ **Sinab ko‘rish** → ✓ **Topshirish** (yashirin testlar).
5. Loyiha havolasini (GitHub repo, PR yoki tag) topshiring — o‘qituvchi **baho** qo‘yadi.

## Baholash mezonlari

- **Natija** — loyiha bosqichi mezonlar ro‘yxati bo‘yicha bajarilgan.
- **Topshiriqlar** — PHP topshiriqlar testlardan o‘tgan.
- **Kod sifati** — tushunarli nomlar, PSR-12, kichik va mazmunli commitlar.
- **Jamoaviy hissa** — har bir a’zo o‘z qismini tushuntira oladi.
- **Muddat** — muddatdan keyin topshirilgan ish “kechikkan” deb belgilanadi.

01

# 1-qadam: production Docker Compose stek

assets va web image · entrypoint · scheduler va worker · lokal sinov

Dockerfile

compose.yaml

entrypoint

scheduler

# Vite fayllari va nginx image

```
FROM node:20-alpine AS assets
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm ci
COPY . .
RUN npm run build                  # -> public/build

FROM nginx:1.27-alpine AS web
COPY docker/nginx.conf /etc/nginx/conf.d/default.conf
COPY --from=assets /app/public /var/www/public

# app bosqichiga (ma’ruzadagi Dockerfile) qo‘shing:
# COPY --from=assets /app/public/build /var/www/public/build
# COPY --chmod=755 docker/entrypoint.sh /usr/local/bin/entrypoint
# ENTRYPOINT ["entrypoint"]
# CMD ["php-fpm"]
```

▶ NATIJA

```
$ docker build --target web -t ghcr.io/jamoa/omborpro-web:dev .
$ docker build -t ghcr.io/jamoa/omborpro:dev .
$ docker images | grep omborpro
ghcr.io/jamoa/omborpro-web   dev   52MB
ghcr.io/jamoa/omborpro       dev   151MB
```

**AS assets**

Node faqat build uchun; yakuniy image larga kirmaydi

**`--target web`**

bitta Dockerfile dan nginx image ini yig‘ish

**`public/build` ikki joyda**

nginx fayllarni beradi, Laravel `manifest.json` ni o‘qiydi

**CMD qayta yoziladi**

`ENTRYPOINT` asos image ning `CMD` ini bekor qiladi

# Konteyner ishga tushganda keshlar

```
#!/bin/sh
# docker/entrypoint.sh — app, worker, scheduler uchun umumiy
set -e
if [ "$APP_ENV" = "production" ]; then
    php artisan optimize      # config, event, route, view keshlari
fi
exec "$@"                     # php-fpm | queue:work | schedule:work

# docker/php.ini
# opcache.enable=1
# opcache.validate_timestamps=0
# opcache.memory_consumption=128
# memory_limit=256M
# upload_max_filesize=20M
# post_max_size=20M
```

▶ NATIJA

```
$ docker compose logs app | head -3
   INFO  Caching framework bootstrap, configuration, and metadata.
  config ..................................... DONE
  routes ..................................... DONE
```

**optimize ishga tushishda**

keshlar serverdagi haqiqiy `.env` qiymatlari bilan yaratiladi

**`exec "$@"`**

asosiy jarayon PID 1 bo‘ladi va signallarni to‘g‘ri oladi

**`validate_timestamps=0`**

konteyner ichida kod o‘zgarmaydi — tekshiruv keraksiz

**`upload_max_filesize`**

nginx dagi `client_max_body_size` bilan mos bo‘lsin

# scheduler va worker ning production sozlamalari

```yaml
  worker:
    <<: *app
    command: php artisan queue:work redis --tries=3 --max-time=3600
    stop_grace_period: 60s
  scheduler:
    <<: *app
    command: php artisan schedule:work
  caddy:
    image: caddy:2-alpine
    ports: ["80:80", "443:443"]
    volumes:
      - ./docker/Caddyfile:/etc/caddy/Caddyfile:ro
      - caddy_data:/data
    depends_on: [web]
    restart: unless-stopped
# web servisidan `ports` olib tashlanadi: tashqariga faqat caddy
# volumes ga qo‘shing: caddy_data: {}
```

▶ NATIJA

```
$ docker compose ps --format "table {{.Service}}\t{{.Status}}"
SERVICE     STATUS
app         Up 2 minutes
caddy       Up 2 minutes
db          Up 2 minutes (healthy)
scheduler   Up 2 minutes
worker      Up 2 minutes
```

**`stop_grace_period`**

worker joriy vazifani tugatishiga vaqt

**`schedule:work`**

har daqiqada `routes/console.php` dagi vazifalarni tekshiradi

**`caddy`**

HTTPS: Let’s Encrypt sertifikatini avtomatik oladi va yangilaydi

**`caddy_data`**

sertifikatlar volume da — qayta ishga tushganda yo‘qolmaydi

Scheduler faqat bitta nusxada ishlashi kerak — aks holda har bir vazifa ikki marta bajariladi.

02

# 2-qadam: GitHub Actions quvuri va deploy job

ci.yml · Secrets · branch protection · teg orqali reliz

GitHub Actions | Secrets | GHCR | Deploy job

DEPLOY JOB

# Teg push qilinganda serverga chiqarish

```yaml
deploy:
  needs: [build]
  if: startsWith(github.ref, 'refs/tags/v')
  runs-on: ubuntu-latest
  environment: production
  steps:
    - name: SSH orqali deploy
      env:
        KEY: ${{ secrets.DEPLOY_SSH_KEY }}
        HOST: ${{ secrets.DEPLOY_HOST }}
        KNOWN: ${{ secrets.DEPLOY_KNOWN_HOSTS }}
      run: |
        mkdir -p ~/.ssh && chmod 700 ~/.ssh
        echo "$KEY" > ~/.ssh/id_ed25519 && chmod 600 ~/.ssh/id_ed25519
        echo "$KNOWN" > ~/.ssh/known_hosts
        ssh deploy@"$HOST" \
          "/opt/omborpro/deploy.sh ${GITHUB_REF_NAME#v}"  # 1.0.0
```

▶ NATIJA

```
deploy  ✓ SSH orqali deploy
  ✓ Container omborpro-app-1     Recreated
  ✓ Container omborpro-worker-1  Recreated
  Deploy 1.0.0: OK
```

**`if: refs/tags/v`**

faqat `v1.0.0` kabi teglarda ishlaydi

**`environment: production`**

GitHub da tasdiq va alohida secret lar uchun

**`known_hosts`**

server kaliti oldindan ma’lum — MITM hujumidan himoya

**`${GITHUB_REF_NAME#v}`**

`v1.0.0` → `1.0.0`: image tegi

SECRETS

# Quvur uchun kerakli maxfiy qiymatlar

| Nomi | Qayerda ishlatiladi | Qanday olinadi |
|---|---|---|
| `GITHUB_TOKEN` | GHCR ga login (build) | GitHub avtomatik beradi |
| `DEPLOY_HOST` | SSH manzili | VPS IP yoki `omborpro.example.uz` |
| `DEPLOY_SSH_KEY` | Serverga kirish | `ssh-keygen -t ed25519 -f deploy_key` |
| `DEPLOY_KNOWN_HOSTS` | Server kalitini tekshirish | `ssh-keyscan -H <host>` (bir marta, tekshirib) |
| **Server** `.env` | Laravel sozlamalari | Faqat serverda, `chmod 600` — GitHub da emas |

Ochiq kalit (`deploy_key.pub`) serverdagi `~deploy/.ssh/authorized_keys` ga qo‘shiladi.

RELIZ OQIMI

# Kod o‘zgarishidan production gacha

1

## Feature tarmoq → PR

lint va test avtomatik ishlaydi; kamida bitta review

2

## main ga merge

build: image `sha` tegi bilan GHCR ga

3

## git tag v1.0.0

build: image `1.0.0` tegi bilan

4

## deploy job

SSH → `deploy.sh 1.0.0` → zaxira, migratsiya, `up -d`

5

## Tutun testi

`/up` = 200; muammo bo‘lsa `deploy.sh 0.9.0` bilan qaytish

Ma’lumotlar bazasi orqaga qaytmaydi — shuning uchun migratsiyalar orqaga mos (expand/contract) bo‘lishi shart.

03

# 3-qadam: VPS ga joylashtirish va ommaviy URL

Server tayyorlash · ufw · Docker · HTTPS · tutun testi · topshirish

VPS | ufw | HTTPS | Smoke test

SERVER TAYYORLASH

# Ubuntu VPS: foydalanuvchi, ufw, Docker

```
# root sifatida, bir marta
apt update && apt -y upgrade
adduser --disabled-password --gecos "" deploy
mkdir -p /home/deploy/.ssh
cp ~/.ssh/authorized_keys /home/deploy/.ssh/
chown -R deploy:deploy /home/deploy/.ssh

ufw allow OpenSSH
ufw allow 80/tcp && ufw allow 443/tcp
ufw enable

curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh                  # rasmiy o‘rnatish skripti
usermod -aG docker deploy
mkdir -p /opt/omborpro/backups && chown -R deploy /opt/omborpro
# /etc/ssh/sshd_config: PasswordAuthentication no, PermitRootLogin no
```

▶ NATIJA

```
$ ufw status
Status: active
To                         Action      From
OpenSSH                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
```

**deploy foydalanuvchi**

root emas; faqat SSH kalit bilan kiradi

**ufw**

faqat 22, 80, 443 — qolgani yopiq

**docker guruhi**

root ga teng huquq — faqat deploy foydalanuvchisiga

**sshd_config**

parol va root kirishini o‘chirish — brute-force dan himoya

`sshd_config` ni o‘zgartirishdan oldin yangi oynada `deploy` bilan kira olishingizni tekshiring — aks holda serverdan qulflanib qolasiz.

HTTPS

# Sertifikat olishning ikki yo‘li

## Caddy (stek ichida)

- `Caddyfile`: `domen { reverse_proxy web:80 }`
- Let’s Encrypt sertifikatini avtomatik oladi va yangilaydi
- Tashqariga faqat Caddy ning 80/443 portlari
- Talaba loyihasi uchun eng sodda yo‘l

## Host nginx + certbot

- Server da alohida nginx teskari proksi
- `certbot --nginx -d domen` bilan sertifikat
- Yangilash — certbot taymeri
- Bir serverda bir nechta loyiha bo‘lsa qulay

Ikkala holatda ham DNS A yozuvi server IP siga qaratilgan bo‘lishi va 80-port ochiq bo‘lishi kerak (HTTP-01 tekshiruvi).

TUTUN TESTI

# Deploydan keyin 5 daqiqalik tekshiruv

| Tekshiruv | Buyruq / harakat | Kutilgan natija |
|---|---|---|
| **Health endpoint** | `curl -i https://domen/up` | `HTTP/2 200` |
| **Yashirin fayllar** | `curl -i https://domen/.env` | `403` yoki `404` |
| **Kirish** | Demo `menejer` bilan login | Bosh sahifa ochiladi |
| **Buyurtma oqimi** | Buyurtma yaratish → tasdiqlash | Qoldiq rezervlanadi |
| **Navbat** | Hisobotni so‘rash | Worker logida job `DONE` |
| **Portlar** | Boshqa kompyuterdan `nc -zv IP 5432` | Ulanish rad etiladi |
| **Zaxira** | `./backup.sh` → test bazaga tiklash | Yozuvlar soni mos |

LOYIHA BOSQICHI

# Natija va topshirish

**Natija: OmborPro to‘liq Docker Compose stekida (web, app, worker, scheduler, db, redis) VPS serverda ommaviy URL orqali ishlaydi; har bir `v*` teg GitHub Actions da lint, test va build dan o‘tib, avtomatik joylashtiriladi.**

## Baholash mezonlari

- ✓ Dockerfile multi-stage: composer `--no-dev`, `pdo_pgsql`, `redis`, `opcache`, `pcntl`; `USER www-data`; `.dockerignore` mavjud
- ✓ `compose.yaml` da web, app, worker, scheduler, db, redis; db uchun healthcheck va `service_healthy`; db/redis portlari ochilmagan
- ✓ nginx konfiguratsiyasi: `root /var/www/public`, `try_files`, `fastcgi_pass app:9000`, yashirin fayllar uchun `deny all`
- ✓ GitHub Actions: `lint` (Pint `--test`, Larastan), `test` (Pest + PostgreSQL servis), `build` (GHCR) — oxirgi ishga tushirish yashil
- ✓ Teg orqali deploy ishlaydi: `deploy.sh` da zaxira, `migrate --force`, `up -d`, `queue:restart`, tutun testi
- ✓ Production `.env`: `APP_ENV=production`, `APP_DEBUG=false`, `APP_KEY` o‘rnatilgan; `.env` git da yo‘q
- ✓ Ommaviy URL ochiladi (HTTPS yoki kelishilgan HTTP), `/up` 200 qaytaradi, `/.env` ochilmaydi
- ✓ `backup.sh` cron da; README da tiklash mashqi natijasi va tutun testi jadvali yozilgan

📤 GitHub repozitoriy havolasi, reliz tegi (masalan `v1.0.0`) va shu teg uchun yashil GitHub Actions ishga tushirish havolasi hamda ommaviy URL ni topshiring; README da deploy bo‘limi, tutun testi jadvali va demo foydalanuvchi (test parol) ko‘rsatilsin.

Telefon kamerasi bilan skanerlang

Loyiha havolasini amaliy sahifasidagi «Loyiha bosqichi» bo‘limida topshiring.

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · Oson · 10 ball

# 4-topshiriq: .env qatorini tahlil qilish

`.env` faylining bitta qatorini tahlil qiling. Bo‘sh qator yoki `#` bilan boshlanadigan izoh — `null`. Aks holda qator `KALIT=qiymat` ko‘rinishida: kalit va qiymat chetidagi bo‘sh joylar olib tashlanadi; birinchi `=` bo‘yicha bo‘linadi. Kalit `^[A-Z][A-Z0-9_]*$` ga mos bo‘lmasa yoki `=` umuman bo‘lmasa — `InvalidArgumentException`. Qiymat qo‘shtirnoq yoki tirnoq bilan boshlansa — qiymat shu tirnoqdan keyingi birinchi xuddi shunday tirnoqqacha bo‘lgan qism (ichidagi `#` saqlanadi, yopuvchi tirnoqdan keyingisi e’tiborsiz); tirnoq yopilmagan bo‘lsa — `InvalidArgumentException`. Tirnoqsiz qiymatda `#` (bo‘sh joy + `#`) dan keyingi qism izoh hisoblanadi va kesiladi.

**Imzo:** `function env_qator(string $qator): ?array`

**Qaytaradi:** `null` yoki `[kalit, qiymat]` — ikki satrdan iborat ro‘yxat.

MISOL · CHAQIRUV

```
env_qator('APP_NAME="OmborPro ERP"')
```

MISOL · NATIJA

```
["APP_NAME","OmborPro ERP"]
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · O‘rta · 20 ball

# 5-topshiriq: Tashqariga ochiq portlarni aniqlash

Docker Compose `ports` yozuvlari tahlil qilinadi: `servis => ['80:80', '127.0.0.1:5432:5432', ...]`. Yozuv `HOST:KONTEYNER` yoki `IP:HOST:KONTEYNER` ko‘rinishida (portlar — raqamlar, IP — `d.d.d.d`); boshqa format — `InvalidArgumentException`. IP ko‘rsatilmagan yoki `0.0.0.0` bo‘lsa — port internetga ochiq (Docker ufw qoidalarini chetlab o‘tadi!). Ochiq va host porti 80 yoki 443 bo‘lmagan har bir yozuv uchun `servis:hostport` satrini qaytaring, alifbo tartibida, takrorsiz.

**Imzo:** `function ochiq_portlar(array $servislar): array`

**Qaytaradi:** Satrlar ro‘yxati, masalan `['db:5432', 'redis:6379']`; xavfli port bo‘lmasa — `[]`.

MISOL · CHAQIRUV

```
ochiq_portlar(['web' => ['80:80', '443:443'], 'db' =>
['5432:5432'], 'redis' => []])
```

MISOL · NATIJA

```
["db:5432"]
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · O‘rta · 20 ball

# 6-topshiriq: Servislarni ishga tushirish tartibi

`depends_on` grafi berilgan: `servis => [bog‘liq servislar]`. Har bir servis o‘zi bog‘liq bo‘lgan barcha servislardan keyin ishga tushadi. Har qadamda bog‘liqliklari tayyor bo‘lgan servislar ichidan **alifbo bo‘yicha eng kichigini** tanlang (Kahn algoritmi). Grafda yo‘q servisga bog‘liqlik bo‘lsa — `InvalidArgumentException`; aylana (sikl) bo‘lsa — `DomainException`.

**Imzo:** `function ishga_tushirish_tartibi(array $deps): array`

**Qaytaradi:** Servis nomlari ro‘yxati — ishga tushirish tartibida.



Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

MISOL · CHAQIRUV

```
ishga_tushirish_tartibi(['web' => ['app'], 'app' => ['db', 'redis'], 'worker' => ['db', 'redis'], 'db' => [], 'redis' => []])
```

MISOL · NATIJA

```
["db","redis","app","web","worker"]
```

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · Qiyin · 30 ball

# 7-topshiriq: CI pipeline bosqichlari va davomiyligi

GitHub Actions joblari berilgan: `nom => ['needs' => [...], 'min' => daqiqa]`. `needs` siz job 1-bosqichda; aks holda bosqichi = `needs` dagi joblar bosqichlarining eng kattasi + 1. Bir bosqichdagi joblar parallel ishlaydi. Job tugash vaqti = o‘z `min` + `needs` dagilarning eng kech tugash vaqti; pipeline davomiyligi — eng kech tugash vaqti (kritik yo‘l). Noma’lum job `needs` da bo‘lsa — `InvalidArgumentException`; sikl bo‘lsa — `DomainException`.

**Imzo:** `function ci_bosqichlar(array $jobs): array`

**Qaytaradi:** `['bosqichlar' => [[...], [...]], 'davomiylik' => int]` — har bir bosqich ichida nomlar alifbo tartibida; joblar bo‘lmasa `['bosqichlar' => [], 'davomiylik' => 0]`.

MISOL · CHAQIRUV

```
ci_bosqichlar(['lint' => ['needs' => [], 'min' => 2], 'test' => ['needs' => [], 'min' => 6], 'build' => ['needs' => ['lint', 'test'], 'min' => 4], 'deploy' => ['needs' => ['build'], 'min' => 1]])
```

MISOL · NATIJA

```
{"bosqichlar":[["lint","test"],["build"],["deploy"]],"davomiylik":11}
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · Qiyin · 30 ball

# 8-topshiriq: Zaxira nusxalarni saqlash siyosati (GFS)

Har kungi `pg_dump` nusxalarining sanalari berilgan (`Y-m-d`, tartibsiz, takrorlanishi mumkin). Saqlanadi: (1) eng yangi `$kunlik` ta turli sana; (2) eng yangi `$haftalik` ta ISO hafta (`o-W`) ning har biridan shu haftadagi eng yangi sana. Qolganlari o‘chiriladi. Sana formati noto‘g‘ri yoki mavjud bo‘lmagan sana (`2026-02-30`) bo‘lsa, yoki `$kunlik`/`$haftalik` manfiy bo‘lsa — `InvalidArgumentException`.

**Imzo:** `function saqlanadigan_nusxalar(array $sanalar, int $kunlik, int $haftalik): array`

**Qaytaradi:** Saqlanadigan sanalar ro‘yxati — takrorsiz, kamayish tartibida (eng yangisi birinchi).

MISOL · CHAQIRUV

```
saqlanadigan_nusxalar(['2026-10-27', '2026-10-26', '2026-10-25', '2026-10-24', '2026-10-19', '2026-10-18', '2026-10-12', '2026-10-05'], 2, 3)
```

MISOL · NATIJA

```
["2026-10-27","2026-10-26","2026-10-25","2026-10-18"]
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

TOPSHIRIQLAR VA TOPSHIRISH

# 14-amaliy: 8 ta PHP topshiriq

1 **Docker image teglari (semver)**
Oson · 10 ball · ma’ruzada

2 **Konteyner healthcheck holati**
O‘rta · 20 ball · ma’ruzada

3 **Migratsiya xavfini baholash**
O‘rta · 20 ball · ma’ruzada

4 **.env qatorini tahlil qilish**
Oson · 10 ball

5 **Tashqariga ochiq portlarni aniqlash**
O‘rta · 20 ball

6 **Servislarni ishga tushirish tartibi**
O‘rta · 20 ball

7 **CI pipeline bosqichlari va davomiyligi**
Qiyin · 30 ball

8 **Zaxira nusxalarni saqlash siyosati (GFS)**
Qiyin · 30 ball

Telefon kamerasi bilan skanerlang

PHP kod saytda yoziladi va server tomonidan avtomatik tekshiriladi.

Loyiha havolasini ham shu sahifada topshirasiz; o‘qituvchi baho qo‘yadi.