14

O‘N TO‘RTINCHI MA’RUZA

Korxona ilovalarini dasturlash · 14-mavzu

Joriy etish va DevOps

Docker image · Compose stek va nginx · GitHub Actions · production sozlamalari · zaxira nusxalar · migratsiyalar

Docker | Docker Compose | GitHub Actions | nginx | pg_dump

ISINISH

OmborPro jamoa noutbugida mukammal ishlaydi. Ertadan 40 ta xodim undan internet orqali foydalanadi. Nima noto‘g‘ri ketishi mumkin?

Hozircha javob bermang — noutbuk va production server orasidagi farqlarni sanab chiqing.

1

Serverdagi PHP versiyasi va kengaytmalar noutbukdagidan farq qilsa, nima bo‘ladi?

2

Yangi versiyani chiqarayotganda migratsiya yarmida xato bersa, buyurtmalar nima bo‘ladi?

3

Server diski ishdan chiqsa, kechagi hisob-fakturalarni qayerdan tiklaysiz?

O‘QUV MAQSADLARI

Bu ma’ruza oxirida siz quyidagilarni bajara olasiz…

1. **Tushuntirish** — image, konteyner, qatlam va multi-stage build tushunchalarini hamda “bir marta yig‘, hamma joyda ishlat” tamoyilini
2. **Yozish** — php-fpm uchun production Dockerfile ni: `pdo_pgsql`, `redis`, `opcache`, `composer install --no-dev`
3. **Loyihalash** — Docker Compose stekini (web, app, worker, db, redis) healthcheck, volume va ichki tarmoq bilan
4. **Sozlash** — nginx ni Laravel uchun: `root public`, `try_files`, `fastcgi_pass app:9000`
5. **Qurish** — GitHub Actions da lint, test va build joblaridan iborat CI/CD quvurini
6. **Qo‘llash** — production sozlamalari va keshlarini: `APP_DEBUG=false`, `config:cache`, `optimize`, `queue:restart`
7. **Baholash** — migratsiyaning production uchun xavfini, zaxira nusxa olish va tiklash strategiyasini

01

Docker: image, konteyner va Dockerfile

Image va qatlamlar · multi-stage build · php-fpm kengaytmalari · .dockerignore

Docker | Image | Multi-stage | php-fpm

Image, konteyner, qatlam, registry

Docker ilovani muhiti bilan birga qadoqlaydi: noutbukda, CI da va serverda bir xil image ishlaydi.

Image

Kod + PHP + kengaytmalar + sozlamalar; faqat o‘qiladigan shablon. Masalan `ghcr.io/jamoa/ombor pro:1.4.0`.

Konteyner

Image dan ishga tushirilgan izolyatsiyalangan jarayon; o‘chirilsa, ichidagi yozuvlar yo‘qoladi (volume dan tashqari).

Qatlam (layer)

Har bir `RUN` / `COPY` natijasi; o‘zgarmagan qatlam keshdan olinadi.

Registry

Image lar ombori: Docker Hub, GitHub Container Registry (`ghcr.io`).

Konteyner — virtual mashina emas: u host yadrosini ulashadi, shuning uchun soniyalarda ishga tushadi.

Asosiy ko‘rsatmalar

Ko‘rsatma	Vazifasi	OmborPro misoli
FROM	Asos image (bosqich boshlanishi)	`FROM php:8.2-fpm-alpine`
WORKDIR	Ishchi papka	`WORKDIR /var/www`
COPY	Fayllarni image ga nusxalash	`COPY composer.json composer.lock ./`
RUN	Build paytida buyruq bajarish	`RUN docker-php-ext-install pdo_pgsql`
USER	Kim nomidan ishlash	`USER www-data`
CMD / ENTRYPOINT	Konteyner ishga tushganda bajariladigan buyruq	`CMD ["php-fpm"]`

`ENV` va `ARG` ga parol yozmang — ular image tarixida (`docker history`) ko‘rinadi.

MULTI-STAGE BUILD

OmborPro uchun production Dockerfile

```dockerfile
# 1-bosqich: PHP bog‘liqliklari
FROM composer:2 AS vendor
WORKDIR /app
COPY composer.json composer.lock ./
RUN composer install --no-dev --no-scripts --no-autoloader
COPY . .
RUN composer dump-autoload --optimize --no-dev --no-scripts

# 2-bosqich: production image (php-fpm)
FROM php:8.2-fpm-alpine
RUN apk add --no-cache postgresql-dev $PHPIZE_DEPS \
 && docker-php-ext-install pdo_pgsql opcache pcntl \
 && pecl install redis && docker-php-ext-enable redis
COPY docker/php.ini /usr/local/etc/php/conf.d/omborpro.ini
WORKDIR /var/www
COPY --from=vendor /app /var/www
RUN chown -R www-data:www-data storage bootstrap/cache
USER www-data
```

▶ NATIJA

```
$ docker build -t ghcr.io/jamoa/omborpro:1.4.0 .
 => CACHED [vendor 4/6] RUN composer install --no-dev ...
 => [stage-1 6/7] COPY --from=vendor /app /var/www
 => naming to ghcr.io/jamoa/omborpro:1.4.0  done
$ docker images ghcr.io/jamoa/omborpro
ghcr.io/jamoa/omborpro   1.4.0   7c1e...   148MB
```

AS vendor

birinchi bosqich nomi; undan faqat natija (`/app`) olinadi

`composer.json` avval

kod o‘zgarsa ham paketlar qatlami keshdan olinadi (CACHED)

`--no-dev`

Pest, Debugbar production image ga kirmaydi

`pcntl`

`queue:work` SIGTERM ni ushlab, joriy vazifani tugatishi uchun

`USER www-data`

konteyner root huquqisiz ishlaydi

php-fpm asos image ning `CMD ["php-fpm"]` buyrug‘i meros qilib olinadi — qayta yozish shart emas.

QATLAM KESHI

Ko‘rsatmalar tartibi build tezligini belgilaydi

Noto‘g‘ri: avval butun kod

- `COPY . .` → `RUN composer install`
- Bitta Blade fayl o‘zgarsa ham kesh buziladi
- Har build da 80+ paket qayta yuklanadi
- `.dockerignore` yo‘q — `.env` va `vendor` image ga tushadi

To‘g‘ri: kam o‘zgaradigani yuqorida

- `COPY composer.json composer.lock ./` → `install`
- Keyin `COPY . .` va `dump-autoload`
- Paketlar qatlami faqat `composer.lock` o‘zgarganda yangilanadi
- `.dockerignore`: `.git`, `.env`, `vendor`, `node_modules`

Qoida: Dockerfile ni “eng barqaror”dan “eng tez o‘zgaradigan”ga qarab yozing.

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

1-bo‘lim yakuni: Docker: image, konteyner va Dockerfile

1

Image bilan konteynerning farqi nimada?

Savol 1

2

Nima uchun `composer.json` ni manba koddan oldin nusxalaymiz?

Savol 2

3

Multi-stage build production image ga qanday foyda beradi?

Savol 3

MA’RUZADAGI MASHQ

Har kim o‘z qurilmasida · 10 daqiqa · Oson · 10 ball

1-topshiriq: Docker image teglari (semver)

GitHub Actions git teg (`v1.4.2`) asosida Docker image uchun teglar ro‘yxatini yasaydi. Teg `vX.Y.Z` yoki `X.Y.Z` ko‘rinishida, ixtiyoriy pre-release qismi bilan (`-rc.1`, `-beta`: harf, raqam, nuqta). Barqaror versiya uchun teglar: `X.Y.Z`, `X.Y`, `X`, `latest`. Agar asosiy versiya `0` bo‘lsa — faqat `0.Y.Z` va `0.Y` (API hali barqaror emas). Pre-release versiya faqat to‘liq tegini oladi. Format noto‘g‘ri bo‘lsa — `InvalidArgumentException`.

Imzo: `function image_teglari(string $teg): array`

Qaytaradi: Satrlar ro‘yxati, aynan yuqoridagi tartibda (boshidagi `v` olib tashlanadi).

MISOL · CHAQIRUV

```
image_teglari('v1.4.2')
```

MISOL · NATIJA

```
["1.4.2","1.4","1","latest"]
```

Telefon kamerasi bilan skanerlang

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

02

Docker Compose stek: web, app, worker, db, redis

Servislar · ichki tarmoq · depends_on va healthcheck · volume · nginx + php-fpm

Docker Compose | healthcheck | nginx | php-fpm

OMBORPRO STEKI

Oltita servis — bitta compose.yaml

Bir xil `omborpro` image dan uchta servis ishlaydi; farqi — faqat buyruq.

1 web — nginx

80/443 port, statik fayllar, `.php` ni `app:9000` ga uzatadi.

2 app — php-fpm

Laravel: Blade + Livewire sahifalar va `/api/v1` REST API.

3 worker

`php artisan queue:work` — hisobotlar, kam qoldiq bildirishnomalari.

4 scheduler

`php artisan schedule:work` — har daqiqada rejalashtirilgan vazifalar.

5 db — PostgreSQL 16

Faqat ichki tarmoqda; ma’lumot `pgdata` volume da.

6 redis — Redis 7

Kesh, sessiya va navbat; tashqariga port ochilmaydi.

web, app, worker — umumiy sozlama yakor bilan

```yaml
x-app: &app
  image: ghcr.io/jamoa/omborpro:${TAG:-latest}
  env_file: .env
  volumes: [storage:/var/www/storage/app]
  restart: unless-stopped
  depends_on:
    db: { condition: service_healthy }
    redis: { condition: service_started }
services:
  web:
    image: ghcr.io/jamoa/omborpro-web:${TAG:-latest}
    ports: ["80:80"]
    depends_on: [app]
    restart: unless-stopped
  app: *app
  worker:
    <<: *app
    command: php artisan queue:work --tries=3 --max-time=3600
```

▶ NATIJA

```
$ TAG=1.4.0 docker compose up -d
 ✓ Container omborpro-db-1      Healthy
 ✓ Container omborpro-redis-1   Started
 ✓ Container omborpro-app-1     Started
 ✓ Container omborpro-worker-1  Started
 ✓ Container omborpro-web-1     Started
```

`x-app: &app`

takrorlanadigan sozlama; `*app` va `<<: *app` bilan qayta ishlatiladi

`${TAG:-latest}`

versiya deploy paytida beriladi: `TAG=1.4.0`

`service_healthy`

`app` faqat PostgreSQL tayyor bo‘lgach ishga tushadi

`omborpro-web`

nginx image: `nginx.conf` va `public/` (CSS/JS) ichida

`--max-time=3600`

worker soatda bir marta o‘zini toza qayta ishga tushiradi

`version:` kaliti eskirgan — Compose v2 uni e’tiborsiz qoldiradi, yozmang.

db va redis: volume va healthcheck

```yaml
  db:
    image: postgres:16-alpine
    environment:
      POSTGRES_DB: omborpro
      POSTGRES_USER: omborpro
      POSTGRES_PASSWORD: ${DB_PASSWORD:?DB_PASSWORD kerak}
    volumes: [pgdata:/var/lib/postgresql/data]
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U omborpro -d omborpro"]
      interval: 5s
      retries: 10
    restart: unless-stopped
  redis:
    image: redis:7-alpine
    volumes: [redisdata:/data]
    restart: unless-stopped
volumes: { pgdata: {}, redisdata: {}, storage: {} }
```

▶ NATIJA

```
$ docker compose ps db
NAME             IMAGE                STATUS
omborpro-db-1    postgres:16-alpine   Up 3 minutes (healthy)
```

ports yo‘q

db va redis faqat ichki tarmoqda: `DB_HOST=db`

`${DB_PASSWORD:?...}`

o‘zgaruvchi berilmasa, Compose xato bilan to‘xtaydi

`pg_isready`

baza ulanish qabul qilsa — `healthy`

volumes

`docker compose down` ma’lumotni saqlaydi; `down -v` — o‘chiradi!

NGINX

Laravel uchun nginx konfiguratsiyasi

```
server {
    listen 80;
    server_name omborpro.example.uz;
    root /var/www/public;
    index index.php;
    client_max_body_size 20m;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }
    location ~ \.php$ {
        fastcgi_pass app:9000;
        fastcgi_param SCRIPT_FILENAME
            $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }
    location ~ /\.(?!well-known).* { deny all; }
}
```

▶ NATIJA

```
$ curl -I http://omborpro.example.uz/.env
HTTP/1.1 403 Forbidden
$ curl -I http://omborpro.example.uz/orders
HTTP/1.1 302 Found
Location: http://omborpro.example.uz/login
```

`root /var/www/public`

faqat `public` ochiq; `.env` va `app/` brauzerga ko‘rinmaydi

`try_files`

fayl bo‘lsa — o‘zi, aks holda `index.php` (Laravel marshrutlari)

`fastcgi_pass app:9000`

Compose servis nomi; `127.0.0.1` emas

`$realpath_root`

symlink lar to‘g‘ri yechiladi; yo‘l app konteynerida ham bir xil

deny all

`.env`, `.git` kabi yashirin fayllar — 403

`client_max_body_size` — Excel import (8-mavzu) uchun fayl hajmi chegarasi; sukut bo‘yicha 1 MB.

TRY_FILES MODELI

nginx so‘rovni qayerga yuboradi?

```php
// nginx qoidalarining soddalashtirilgan modeli
function nginx(string $uri, array $files): string
{
    $path = parse_url($uri, PHP_URL_PATH);
    $query = parse_url($uri, PHP_URL_QUERY) ?? '';
    if (preg_match('#/\.(?!well-known)#', $path)) {
        return '403 (deny all)';
    }
    if (in_array($path, $files, true)) {
        return "statik fayl: $path";
    }
    return "php-fpm: /index.php?$query";
}
$files = ['/favicon.ico', '/build/app.css'];
foreach (['/build/app.css', '/orders?page=2', '/.env',
          '/api/v1/products'] as $u) {
    echo str_pad($u, 18), '→ ', nginx($u, $files), PHP_EOL;
}
```

▶ NATIJA

```
/build/app.css    → statik fayl: /build/app.css
/orders?page=2    → php-fpm: /index.php?page=2
/.env             → 403 (deny all)
/api/v1/products  → php-fpm: /index.php?
```

regex location

nginx da regex `location` lar prefiks `/` dan ustun turadi

`$files`

`public/` dagi haqiqiy fayllar — PHP ishga tushmaydi, tez

`index.php?$query`

`page=2` Laravel ga `$request->query('page')` bo‘lib yetadi

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

2-bo‘lim yakuni: Docker Compose stek: web, app, worker, db, redis

1

Nima uchun `.env` da `DB_HOST=127.0.0.1` emas, `DB_HOST=db` yoziladi?

Savol 1

2

`depends_on` va `condition: service_healthy` ning farqi nimada?

Savol 2

3

nginx `root` ni loyiha ildiziga qaratsak, qanday xavf paydo bo‘ladi?

Savol 3

MA’RUZADAGI MASHQ

Har kim o‘z qurilmasida · 10 daqiqa · O‘rta · 20 ball

2-topshiriq: Konteyner healthcheck holati

Docker har `interval` da healthcheck buyrug‘ini bajaradi. Konteyner `starting` holatida boshlanadi. Muvaffaqiyatli tekshiruv (`true`) holatni `healthy` qiladi va ketma-ket xatolar hisoblagichini 0 ga tushiradi. Muvaffaqiyatsiz tekshiruv (`false`) hisoblagichni oshiradi; hisoblagich `retries` ga yetsa — `unhealthy`, aks holda holat o‘zgarmaydi. Har bir tekshiruvdan keyingi holatlar ro‘yxatini qaytaring. `retries < 1` bo‘lsa — `InvalidArgumentException`.

Imzo: `function healthcheck_holatlari(array $natijalar, int $retries): array`

Qaytaradi: `$natijalar` uzunligidagi satrlar ro‘yxati: `starting`, `healthy` yoki `unhealthy`.

MISOL · CHAQIRUV

```
healthcheck_holatlari([false, false, true, true], 3)
```

MISOL · NATIJA

```
["starting","starting","healthy","healthy"]
```

Telefon kamerasi bilan skanerlang

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

03

CI/CD: GitHub Actions bilan lint, test, build

Uzluksiz integratsiya · workflow va job · servis konteyner · GHCR ga image

CI/CD | GitHub Actions | Pint | Pest | GHCR

ci.yml: hodisalar va lint job

```yaml
name: CI
on:
  push: { branches: [main], tags: ['v*'] }
  pull_request:
permissions: { contents: read, packages: write }
jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: shivammathur/setup-php@v2
        with: { php-version: '8.2', tools: composer:v2 }
      - run: composer install --no-interaction --prefer-dist
      - run: vendor/bin/pint --test
      - run: vendor/bin/phpstan analyse --memory-limit=1G
```

▶ NATIJA

```
lint  ✓ Run vendor/bin/pint --test
      PASS ........................... 214 files
lint  ✓ Run vendor/bin/phpstan analyse --memory-limit=1G
      [OK] No errors
```

on: push / pull_request

har push, PR va `v*` teg da ishga tushadi

setup-php@v2

PHP 8.2 va Composer ni o‘rnatadi

pint --test

faqat tekshiradi; uslub buzilsa — job qizil

phpstan analyse

Larastan: statik tahlil, `phpstan.neon` dagi daraja bilan

Pest testlari PostgreSQL servis konteyneri bilan

```yaml
test:
  runs-on: ubuntu-latest
  services:
    postgres:
      image: postgres:16
      env: { POSTGRES_PASSWORD: secret, POSTGRES_DB: testing }
      ports: ['5432:5432']
      options: --health-cmd pg_isready --health-retries 10
  env: { DB_CONNECTION: pgsql, DB_HOST: 127.0.0.1,
         DB_DATABASE: testing, DB_USERNAME: postgres,
         DB_PASSWORD: secret }
  steps:
    - uses: actions/checkout@v4
    - uses: shivammathur/setup-php@v2
      with: { php-version: '8.2', extensions: pdo_pgsql }
    - run: composer install --no-interaction --prefer-dist
    - run: cp .env.example .env && php artisan key:generate
    - run: vendor/bin/pest
```

▶ NATIJA

```
PASS  Tests\Feature\OrderServiceTest
✓ buyurtmani tasdiqlash qoldiqni rezervlaydi      0.21s
✓ mavjud qoldiqdan ko‘p rezerv taqiqlanadi        0.08s
Tests:    86 passed (241 assertions)
Duration: 9.42s
```

`services: postgres`

job davomida yashaydigan haqiqiy PostgreSQL 16

`--health-cmd`

testlar baza tayyor bo‘lgach boshlanadi

job env

muhit o‘zgaruvchilari `.env` dagi qiymatlardan ustun turadi

`key:generate`

test uchun vaqtinchalik `APP_KEY`

Image ni yig‘ish va GHCR ga yuklash

```yaml
build:
  needs: [lint, test]
  if: github.event_name == 'push'
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v4
    - uses: docker/login-action@v3
      with:
        registry: ghcr.io
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}
    - uses: docker/build-push-action@v6
      with:
        context: .
        push: true
        tags: |
          ghcr.io/jamoa/omborpro:${{ github.sha }}
          ghcr.io/jamoa/omborpro:latest
```

▶ NATIJA

```
build  ✓ docker/build-push-action@v6
  #12 pushing ghcr.io/jamoa/omborpro:9f3c2a1... done
  #12 pushing ghcr.io/jamoa/omborpro:latest done
```

`needs: [lint, test]`

ikkalasi yashil bo‘lmasa, build boshlanmaydi

`if: push`

PR lardan image yuklanmaydi

`GITHUB_TOKEN`

GitHub har ishga tushirish uchun beradigan vaqtinchalik token

`github.sha`

har bir image qaysi commit dan yig‘ilgani aniq

Image nomi `ghcr.io` da kichik harflarda bo‘lishi shart. Semver teglar uchun `docker/metadata-action@v5` qulay.

▶ INTERAKTIV

CI mi yoki CD mi?

Har bir qadamni quvurning to‘g‘ri qismiga joylashtiring. (Interaktiv slaydlarda bosib bajariladi.)

`vendor/bin/pint --test`

`vendor/bin/phpstan analyse`

PostgreSQL servis konteyneri bilan Pest

Branch protection: faqat yashil PR

Image ni `ghcr.io` ga push qilish

SSH orqali `docker compose pull`

Serverda `migrate --force`

Deploydan keyin `/up` tutun testi

CI: kodni tekshirish

CD: yetkazish va joylashtirish

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

3-bo‘lim yakuni: CI/CD: GitHub Actions bilan lint, test, build

1

Continuous Delivery va Continuous Deployment ning farqi nimada?

Savol 1

2

Nima uchun CI da `pint` emas, `pint --test` ishlatiladi?

Savol 2

3

Build job qachon ishlaydi va nima uchun `needs` kerak?

Savol 3

04

Production konfiguratsiyasi va keshlash

.env va maxfiy ma’lumotlar · APP_DEBUG=false · config/route/view kesh · optimize · OPcache

.env | APP_DEBUG | config:cache | optimize

Production uchun asosiy sozlamalar

```
APP_NAME=OmborPro
APP_ENV=production
APP_DEBUG=false
APP_KEY=base64:...            # key:generate --show
APP_URL=https://omborpro.example.uz
APP_TIMEZONE=Asia/Tashkent
LOG_LEVEL=warning
DB_CONNECTION=pgsql
DB_HOST=db
DB_DATABASE=omborpro
DB_USERNAME=omborpro
DB_PASSWORD=                  # faqat serverda to‘ldiriladi
CACHE_STORE=redis
QUEUE_CONNECTION=redis
SESSION_DRIVER=redis
SESSION_SECURE_COOKIE=true
REDIS_HOST=redis
```

▶ NATIJA

```
$ docker compose exec app php artisan about --only=environment
  Application Name ..................... OmborPro
  Environment ........................... production
  Debug Mode ................................... OFF
  Timezone ............................. Asia/Tashkent
```

`APP_DEBUG=false`

xato sahifasida parol va stack trace ko‘rinmaydi

`APP_KEY`

bir marta yaratiladi; o‘zgarsa sessiyalar va shifrlangan ma’lumot buziladi

`DB_HOST=db`

Compose servis nomi

`…=redis`

kesh, navbat va sessiya Redis da — bir nechta app konteyneri uchun ham umumiy

`.env` — git da emas. Serverda: `chmod 600 .env`, egasi — deploy foydalanuvchisi.

KESHLAR

Deploy paytidagi artisan buyruqlari

Buyruq	Nima qiladi	Eslatma
`config:cache`	Konfiguratsiyani bitta faylga yig‘adi	Keyin `.env` o‘qilmaydi
`route:cache`	Marshrutlar ro‘yxatini keshlaydi	Marshrutlar ko‘p bo‘lsa sezilarli tezlik
`view:cache`	Blade shablonlarini oldindan kompilyatsiya qiladi	Birinchi so‘rov sekinlashmaydi
`event:cache`	Hodisa → tinglovchi xaritasini keshlaydi	Avto-topish har so‘rovda ishlamaydi
`optimize`	Laravel 12: yuqoridagi to‘rttasini bajaradi	Har deployda
`optimize:clear`	Barcha keshlarni tozalaydi	Muammo bo‘lsa birinchi qadam
`queue:restart`	Worker larga “joriy vazifadan keyin qayta ishga tush” signali	Kod yangilangandan keyin

Lokal ishlab chiqishda keshlarni yoqmang — `.env` va marshrut o‘zgarishlari ko‘rinmay qoladi.

TEKSHIRUV SKRIPTI

Production .env ni deploydan oldin tekshirish

```php
function prodXatolar(array $env): array
{
    $x = [];
    if (($env['APP_ENV'] ?? '') !== 'production') {
        $x[] = 'APP_ENV=production bo‘lishi kerak';
    }
    if (($env['APP_DEBUG'] ?? 'true') !== 'false') {
        $x[] = 'APP_DEBUG=false bo‘lishi kerak';
    }
    if (!str_starts_with($env['APP_KEY'] ?? '', 'base64:')) {
        $x[] = 'APP_KEY o‘rnatilmagan';
    }
    return $x;
}
$env = ['APP_ENV' => 'production', 'APP_DEBUG' => 'true'];
echo implode(PHP_EOL, prodXatolar($env)), PHP_EOL;
echo count(prodXatolar(['APP_ENV' => 'production',
    'APP_DEBUG' => 'false', 'APP_KEY' => 'base64:a2V5'])), PHP_EOL;
```

▶ NATIJA

```
APP_DEBUG=false bo‘lishi kerak
APP_KEY o‘rnatilmagan
0
```

`?? 'true'`

qiymat yo‘q bo‘lsa — eng xavfli holat deb hisoblaymiz

`!== 'false'`

`.env` dagi qiymat — satr; `(bool) 'false'` esa `true`!

`base64:`

`key:generate` yaratgan kalit formati

`count(...) === 0`

xato yo‘q — deploy davom etadi

config:cache dan keyin env() qayerda ishlaydi?

Noto‘g‘ri: env() kod ichida

- `CbuRateService`: `Http::get(env('CBU_URL'))`
- Lokal muhitda ishlaydi — kesh yo‘q
- Production da `config:cache` → `.env` o‘qilmaydi → `null`
- Xato faqat serverda chiqadi — eng qimmat xato

To‘g‘ri: env() faqat config/ da

- `config/services.php`: `'cbu' => ['url' => env('CBU_URL')]`
- Kodda: `config('services.cbu.url')`
- Qiymat kesh faylida saqlanadi
- Code review va Larastan qoidasi bilan nazorat qilinadi

Sukut qiymat ham config faylida beriladi: `env('CBU_URL', 'https://cbu.uz/...')`.

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

4-bo‘lim yakuni: Production konfiguratsiyasi va keshlash

1

Production da `APP_DEBUG=true` qolsa, qanday xavf bor?

Savol 1

2

Laravel 12 da `php artisan optimize` qaysi keshlarni yaratadi?

Savol 2

3

`config:cache` dan keyin controller ichidagi `env('CBU_URL')` nima qaytaradi va nima uchun?

Savol 3

05

Zaxira nusxalar (backup) va tiklash

RPO va RTO · pg_dump va pg_restore · 3-2-1 qoidasi · saqlash siyosati · tiklashni sinash

pg_dump | 3-2-1 | Retention | RPO / RTO

ZAXIRA STRATEGIYASI

Nimani, qancha va qayerda saqlaymiz?

1 RPO

Qancha ma’lumot yo‘qotishga rozimiz: kunlik dump → 24 soatgacha.

2 RTO

Tizim qancha vaqtda tiklanishi kerak: masalan, 2 soat ichida.

3 3 nusxa

Asl baza + kamida 2 ta zaxira.

4 2 vosita

Server diski + obyektli ombor (S3-mos) yoki boshqa server.

5 1 offsite

Kamida bitta nusxa boshqa data-markazda.

6 Shifrlash va sinov

Dump shifrlanadi; oyiga bir marta test bazaga tiklanadi.

Docker volume va RAID — zaxira emas: ular `down -v`, xato `DELETE` yoki hosting bloklanishidan himoya qilmaydi.

Kunlik zaxira skripti va cron

```bash
#!/usr/bin/env bash
# /opt/omborpro/backup.sh — har kuni cron orqali
set -euo pipefail
cd /opt/omborpro
FILE="backups/omborpro_$(date +%F_%H%M).dump"
docker compose exec -T db \
    pg_dump -U omborpro -Fc omborpro > "$FILE"
test -s "$FILE"                    # bo‘sh fayl — xato
rclone copy "$FILE" offsite:omborpro-backups
find backups -name '*.dump' -mtime +14 -delete

# crontab -e   (server vaqti: Asia/Tashkent)
# 30 2 * * * /opt/omborpro/backup.sh >> /var/log/omb-backup.log 2>&1

# Tiklash mashqi — ALOHIDA test bazaga:
# docker compose exec -T db pg_restore --clean --if-exists \
#     -U omborpro -d omborpro_restore_test < backups/oxirgi.dump
```

▶ NATIJA

```
$ ls -lh /opt/omborpro/backups | tail -2
-rw-r--r-- 1 deploy deploy 18M Oct 26 02:30 omborpro_2026-10-
26_0230.dump
-rw-r--r-- 1 deploy deploy 18M Oct 27 02:30 omborpro_2026-10-
27_0230.dump
```

`-T`

TTY siz: ikkilik dump buzilmaydi

`-Fc`

siqilgan custom format, `pg_restore` bilan tanlab tiklanadi

`test -s`

fayl bo‘sh bo‘lsa skript xato bilan tugaydi

rclone copy

offsite nusxa (3-2-1 dagi “1”)

`-mtime +14`

lokal nusxalar 14 kundan keyin o‘chiriladi

SAQLASH SIYOSATI

Qaysi nusxalar qoladi? (kunlik + haftalik)

```php
function saqlanadi(array $sanalar, int $kunlik, int $hafta): array
{
    rsort($sanalar);
    $keep = array_slice($sanalar, 0, $kunlik);
    $yak = array_filter($sanalar, fn ($s) =>
        (new DateTimeImmutable($s))->format('N') === '7');
    $keep = array_unique([...$keep, ...array_slice($yak, 0, $hafta)]);
    rsort($keep);
    return $keep;
}
$bugun = new DateTimeImmutable('2026-10-27');
$sanalar = [];
for ($i = 0; $i < 30; $i++) {
    $sanalar[] = $bugun->modify("-$i day")->format('Y-m-d');
}
$keep = saqlanadi($sanalar, 3, 3);
echo implode(PHP_EOL, $keep), PHP_EOL;
echo 'O‘chiriladi: ', count($sanalar) - count($keep), PHP_EOL;
```

▶ NATIJA

```
2026-10-27
2026-10-26
2026-10-25
2026-10-18
2026-10-11
O‘chiriladi: 25
```

rsort

`Y-m-d` satrlari alifbo bo‘yicha ham sana bo‘yicha saralanadi

format('N') === '7'

ISO hafta kuni: 7 — yakshanba (haftalik nusxa)

array_unique

25-oktabr ham kunlik, ham haftalik — bir marta sanaladi

30 → 5

disk 6 barobar kam band bo‘ladi, 3 hafta orqaga tiklash mumkin

Amaliy topshiriqda to‘liq GFS: ISO haftalar (`o-W`) va noto‘g‘ri sanalar tekshiruvi bilan.

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

5-bo‘lim yakuni: Zaxira nusxalar (backup) va tiklash

1

RPO va RTO ning farqi nimada?

Savol 1

2

Nima uchun `docker compose exec` ga `-T` bayrog‘i qo‘shiladi?

Savol 2

3

3-2-1 qoidasini OmborPro misolida tushuntiring.

Savol 3

06

Production migratsiyalari va xavfsiz deploy

migrate --force · expand/contract · qulflar va CONCURRENTLY · texnik rejim · deploy skripti

migrate --force

Zero-downtime

Expand/contract

Maintenance mode

EXPAND / CONTRACT

Ustunni uzilishsiz qayta nomlash

✗ Bir qadamda: renameColumn

- `$table->renameColumn('method', 'channel')`
- Migratsiya tugadi — eski konteyner hali `method` ga yozadi
- To‘lov webhooklari 500 xato beradi
- Orqaga qaytarish ham uzilish bilan

✓ Expand/contract: 3 reliz

- 1.5: `channel` qo‘shiladi, kod ikkalasiga yozadi
- 1.5: eski yozuvlar qismlarga bo‘lib ko‘chiriladi
- 1.6: kod faqat `channel` ni o‘qiydi
- 1.7: `method` ustuni o‘chiriladi (contract)

Har bir reliz alohida deploy qilinadi va har birida eski + yangi kod birga ishlay oladi.

CONCURRENTLY

Katta jadvalga indeksni qulfsiz qo‘shish

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Support\Facades\DB;

return new class extends Migration
{
    // CONCURRENTLY tranzaksiya ichida ishlamaydi
    public $withinTransaction = false;

    public function up(): void
    {
        DB::statement('CREATE INDEX CONCURRENTLY IF NOT EXISTS '
            . 'invoices_status_due_at_idx '
            . 'ON invoices (status, due_at)');
    }
};
```

▶ NATIJA

```
$ php artisan migrate --force

   INFO  Running migrations.

  2026_10_27_090000_add_invoices_status_due_index .. 2s DONE
```

`$withinTransaction = false`

Laravel bu migratsiyani tranzaksiyaga o‘ramaydi

CONCURRENTLY

indeks qurilayotganda `invoices` ga yozish bloklanmaydi

IF NOT EXISTS

yarim yo‘lda to‘xtagan migratsiyani qayta ishga tushirish xavfsizroq

`--force`

production da tasdiq so‘ralmaydi

CONCURRENTLY xato bilan to‘xtasa, yaroqsiz (INVALID) indeks qolishi mumkin — uni o‘chirib, qayta yarating.

DEPLOY SKRIPTI

Serverdagi deploy.sh — har qadam yozilgan

```
#!/usr/bin/env bash
# ishlatish: ./deploy.sh 1.4.0
set -euo pipefail
cd /opt/omborpro
export TAG="$1"

./backup.sh                          # 1) avval zaxira
docker compose pull                  # 2) yangi image lar
docker compose run --rm app \
    php artisan migrate --force      # 3) orqaga mos migratsiya
docker compose up -d --remove-orphans  # 4) konteynerlarni yangilash
docker compose exec -T app php artisan queue:restart
curl -fsS https://omborpro.example.uz/up > /dev/null \
    && echo "Deploy $TAG: OK"       # 5) tutun testi
```

▶ NATIJA

```
$ ./deploy.sh 1.4.0
 ✓ app Pulled    ✓ web Pulled
  2026_10_27_090000_add_invoices_status_due_index .. DONE
 ✓ Container omborpro-app-1      Recreated
 ✓ Container omborpro-worker-1   Recreated
Deploy 1.4.0: OK
```

`set -euo pipefail`

biror qadam xato bersa — skript darhol to‘xtaydi

`run --rm app`

migratsiya yangi image bilan, eski konteynerlar ishlab turganda

`up -d`

faqat image i o‘zgargan konteynerlar qayta yaratiladi

`/up`

Laravel 12 ning o‘rnatilgan health route i — 200 bo‘lsa, ilova ko‘tarildi

`php artisan optimize` konteyner ishga tushganda entrypoint skriptida bajariladi (amaliy mashg‘ulotda).

▶ INTERAKTIV

Production migratsiyalari

Har bir vaziyat uchun to‘g‘ri yechimni tanlang. (Interaktiv slaydlarda bosib bajariladi.)

1 **Deploy skriptida migratsiya qanday ishga tushiriladi?**

A) `php artisan migrate:fresh`
B) `php artisan migrate --force`
C) `php artisan migrate --pretend`
D) `php artisan db:wipe`

2 **5 million qatorli `stock_movements` ga indeks qo‘shishning xavfsizroq usuli?**

A) Oddiy `$table->index()` — tez tugaydi
B) Texnik rejimsiz `ALTER TABLE ... TYPE`
C) `CREATE INDEX CONCURRENTLY` + `$withinTransaction = false`
D) Indeksni faqat lokal bazaga qo‘shish

3 **`payments.method` ustunini uzilishsiz `channel` ga aylantirish uchun nima qilinadi?**

A) Expand/contract: yangi ustun, ikkalasiga yozish, ko‘chirish, keyin eskisini o‘chirish
B) Bitta migratsiyada `renameColumn`
C) `migrate:rollback` va qayta yaratish
D) Ustunni o‘chirib, yangisini qo‘shish

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

6-bo‘lim yakuni: Production migratsiyalari va xavfsiz deploy

1

Nima uchun deploy skriptida `migrate` emas, `migrate --force` yoziladi?

Savol 1

2

Expand/contract usulining mohiyati nimada?

Savol 2

3

Texnik rejimda (`down --secret`) foydalanuvchi va administrator nimani ko‘radi?

Savol 3

MA’RUZADAGI MASHQ

Har kim o‘z qurilmasida · 10 daqiqa · O‘rta · 20 ball

3-topshiriq: Migratsiya xavfini baholash

Production ga chiqishdan oldin bitta SQL buyrug‘ining (PostgreSQL) xavf darajasini aniqlang (katta-kichik harf farqsiz). `xavfli`: `DROP TABLE`, `DROP COLUMN`, `RENAME`, `ALTER COLUMN <ustun> TYPE` yoki `SET DATA TYPE`, yoki `ADD COLUMN` + `NOT NULL` bo‘lib, `DEFAULT` yo‘q. `ehtiyot`: `CREATE INDEX` / `CREATE UNIQUE INDEX CONCURRENTLY` siz, yoki `SET NOT NULL`. Qolganlari — `xavfsiz`. Bir nechta qoida mos kelsa — eng yuqori daraja. Bo‘sh (yoki faqat bo‘sh joydan iborat) SQL — `InvalidArgumentException`.

Imzo: `function migratsiya_xavfi(string $sql): string`

Qaytaradi: `'xavfsiz'`, `'ehtiyot'` yoki `'xavfli'`.

MISOL · CHAQIRUV	MISOL · NATIJA
`migratsiya_xavfi('ALTER TABLE products DROP COLUMN unit')`	`"xavfli"`

Telefon kamerasi bilan skanerlang

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

XULOSA

Asosiy xulosalar

1 Bir marta yig‘, hamma joyda ishlat

Multi-stage Dockerfile: composer `--no-dev`, `pdo_pgsql`, `redis`, `opcache`; bir xil image test va production da.

2 Compose — butun stek

web, app, worker, db, redis: ichki tarmoq, volume, healthcheck va `service_healthy`.

3 nginx + php-fpm

`root public`, `try_files ... /index.php?$query_string`, `fastcgi_pass app:9000`.

4 CI/CD quvuri

Pint `--test`, Larastan, Pest (PostgreSQL bilan) → image GHCR ga → deploy.

5 Production sozlamalari

`APP_DEBUG=false`, maxfiy `.env`, `optimize`, `env()` faqat config da, `queue:restart`.

6 Zaxira va migratsiya

pg_dump `-Fc` + 3-2-1 + tiklash mashqi; `migrate --force`, expand/contract, CONCURRENTLY.

TAKRORLASH

Yakunlashdan oldin sakkizta savol

1. Image va konteyner farqi?
2. Nima uchun `COPY composer.json COPY . .` dan oldin?
3. `depends_on` yetarli emasmi?
4. nginx da `fastcgi_pass` qayerga qaratiladi?
5. CI da `pint --test` nima qiladi?
6. `config:cache` dan keyin `env()`?
7. 3-2-1 qoidasi?
8. Production da migratsiya buyrug‘i?

AMALIY MASHG‘ULOT

14-amaliy mashg‘ulot: 5 ta PHP topshiriq va loyiha bosqichi

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.

CHIQISH CHIPTASI

Darsni tark etishdan oldin

Javoblaringizni qog‘ozga yozing va o‘qituvchiga topshiring.

1

OmborPro ni yangi VPS ga chiqarish uchun bajariladigan 6 ta asosiy qadamni tartib bilan yozing.

2

Sizning loyihangizdagi qaysi migratsiya production da eng xavfli bo‘lardi va uni qanday xavfsiz qilasiz?

3

Bugungi mavzuda sizga eng noaniq bo‘lib qolgan qism qaysi?