گزارش بررسی فنی پلتفرم CloudHost

باگ‌ها، ریسک‌های پروداکشن و موارد بهبود — بیلد، دیتابیس‌ها، GitOps/CI-CD و امنیت اپلیکیشن

تاریخ: ۲ تیر ۱۴۰۴ (2 Jul 2026) · محدوده: کل مخزن cloud-host

BUG — قطعاً می‌شکند | SECURITY — حفره امنیتی | RISK — احتمال شکست در پروداکشن | IMPROVEMENT — بهبود

خلاصه مدیریتی

پروژه معماری خوبی دارد اما در وضعیت فعلی **آماده پروداکشن نیست**. چند دسته مشکل بحرانی وجود دارد که یا هم‌اکنون باگ هستند یا حتماً در پروداکشن (به‌ویژه در شبکه ایران) می‌شکنند:

1. **باگ‌های قطعی بیلد** — برخی Dockerfileها اصلاً build نمی‌شوند (مثلاً Go).
2. **باگ چرخه دوم آپگرید** — سیستم migration در دومین `helm upgrade` قطعاً می‌شکند.
3. **حفره‌های امنیتی مالی** — کاربر می‌تواند کیف پول خود را رایگان شارژ کند و بدون پرداخت دیپلوی کند.
4. **وابستگی به Docker Hub** بدون آینه (mirror) برای ایمیج دیتابیس‌ها و base imageها.
5. **چرخش رمز سرویس‌ها** — رمز Redis/RabbitMQ در هر آپگرید عوض می‌شود و اتصال اپ قطع می‌شود.

۱. فرایند بیلد اپلیکیشن‌ها (Kaniko + Dockerfile) هر رانتایم

باگ‌های قطعی

BUG **Go — سینتکس نامعتبر COPY؛ هر بیلد Go خراب می‌شود**

`backend/src/build/build.service.ts:1311-1314` — دستور `COPY ... 2>/dev/null || true` از ریدایرکت شل پشتیبانی نمی‌کند؛ Kaniko این خطوط را رد می‌کند و بیلد هر اپ Go شکست می‌خورد.

BUG **Node.js — شکست بیلد نادیده گرفته می‌شود**

`backend/src/build/build.service.ts:1034` — `RUN npm run build || echo "..."` ؛ اگر بیلد خطا بدهد باز هم ایمیج ساخته می‌شود و اپ خراب دیپلوی می‌شود. کاربر «بیلد موفق» می‌بیند ولی اپ کار نمی‌کند.

ریسک‌های جدی

RISK **Base imageها بدون آینه، از Docker Hub / GCR / MCR**

همه رانتایم‌ها (`node:` , `php:` , `python:` , `golang:` , `wordpress:`) و ایمیج Kaniko و init pods (`alpine:3.19` , `alpine/git`) مستقیم از رجیستری‌های عمومی pull می‌شوند. آینه فقط برای استک لاگینگ تعریف شده (`configuration.ts:151`). در ایران بیشترین منبع شکست بیلد است.

RISK **Laravel — نبود اکستنشن‌های ضروری PHP**

`backend/src/build/build.service.ts:1085` — فقط `pdo, pdo_mysql, opcache` نصب می‌شود؛ `mbstring, xml, bcmath, zip, fileinfo, tokenizer` که Laravel استاندارد لازم دارد نصب نمی‌شود.

RISK **Python — پروژه‌های pyproject.toml پشتیبانی نمی‌شوند**

`backend/src/build/build.service.ts:1413` — تشخیص‌دهنده `pyproject.toml` را Python می‌شناسد ولی Dockerfile فقط `requirements.txt` نصب می‌کند؛ پروژه‌های Poetry/PDM فقط Flask+gunicorn پیش‌فرض می‌گیرند. اگر install خطا بدهد، fallback خاموش (`2>/dev/null ||`) اپ اشتباه بالا می‌آورد.

RISK **حافظه Kaniko فقط ۴Gi و PVC بیلد بدون StorageClass**

`build.service.ts:588` بیلد Next.js/.NET/Composer اغلب بیشتر می‌خواهد → PVC `build.service.ts:775`. بیلد OOMKilled. `storageClassName` ندارد → در کلاستر بدون SC پیش‌فرض برای همیشه Pending می‌ماند. همچنین `npm install --legacy-peer-deps` به‌جای `npm ci` (خط ۱۰۱۸).

امنیت بیلد

SECURITY **توکن Git داخل spec پاد و تزریق دستور از branch**

`build.service.ts:498-523` — `cloneUrl` با توکن embedشده در command کانتینر → قابل دیدن در `kubectl get pod -o yaml` ، etcd و audit log. همچنین `${branch}` بدون کوت داخل شل → نامی مثل `main; curl evil` کد اجرا می‌کند. بدون اعتبارسنجی URL گیت (SSRF به IPهای داخلی کلاستر). خطر Zip slip / zip bomb در استخراج با `unzip` (خط ۴۶۲) با سقف آپلود ۱۰GiB.

پایداری فرایند

BUG **ری‌استارت backend وسط بیلد → deployment گیر می‌کند**

state — `build.service.ts:56` بیلد در `Map` حافظه است؛ بعد از ری‌استارت، Job روی کلاستر ادامه می‌دهد ولی deployment در وضعیت `BUILDING` گیر می‌کند و reconcile نمی‌شود. همچنین دیپلوی هم‌زمان برای یک اپ قفل ندارد و روی همان Helm release رقابت می‌کنند.

۲. پیش‌نمایش و دیپلوی

پیش‌نمایش با ساخت یک عدد V رقمی پایدار برای هر اپ و host به‌شکل `{userPrefix}-{previewNumber}.{previewRootDomain}` کار می‌کند.

RISK **با ست‌شدن دامنه اختصاصی، پیش‌نمایش بلافاصله حذف می‌شود**

`kubernetes.service.ts:291` — حتی قبل از تأیید DNS؛ کاربر تا وریفای شدن دامنه هیچ آدرس قابل‌دسترسی ندارد. پیش‌نمایش نیازمند DNS wildcard فعال + cert-manager و مقدار `PREVIEW_BASE_DOMAIN` است.

RISK **getPreviewInfo روی هر فراخوانی Service را به NodePort پچ می‌کند**

`kubernetes.service.ts:2955` — عارضه جانبی که ممکن است اپ را ناخواسته روی IP نود باز کند.

BUG **رجیستری per-cluster + fallback بین‌کلاستری → ImagePullBackOff**

`deployments.service.ts:360` — ایمیج روی رجیستری کلاستر A ساخته و push می‌شود، ولی `deployWithClusterFallback` می‌تواند روی کلاستر B دیپلوی کند که آن ایمیج را ندارد.

۳. دیتابیس‌ها و سرویس‌های اختیاری

باگ‌ها

BUG **رمز Redis و RabbitMQ در هر helm upgrade عوض می‌شود**

`redis-deployment.yaml:18`، `rabbitmq-deployment.yaml:19` — `randAlphaNum 16` بدون `lookup` هر بار مقدار جدید تولید می‌کند؛ `resource-policy: keep` فقط جلوی حذف را می‌گیرد نه تغییر. بعد از هر redeploy رمز عوض می‌شود ولی داده PVC رمز قدیمی دارد → قطع اتصال. الگوی درست در چارت پلتفرم (`cloudhost-platform/templates/secret.yaml`) با `lookup` موجود است.

BUG **Health probe رِدیس/مونگو بدون احراز هویت**

`redis-deployment.yaml:80` — `redis-cli ping` بدون `-a` ؛ با `--requirepass` جواب probe → NOAUTH رد → CrashLoopBackOff. همین برای probe مونگو بدون credential.

BUG **MongoDB در snapshot و wp-content restore پشتیبانی نمی‌شوند**

export/restore — `kubernetes.service.ts:4389` فقط Postgres و MySQL دارد؛ اپ Mongo dump خراب می‌گیرد.

restore — `kubernetes.service.ts:4724` محتوای wp-content از طریق Secret ذخیره می‌شود که محدودیت ~۱MiB دارد؛ هر wp-content واقعی بزرگ‌تر است → شکست.

BUG **WordPress + PostgreSQL و WordPress بدون دیتابیس مجاز است**

ایمیج رسمی وردپرس فقط MySQL/MariaDB را می‌شناسد ولی پلتفرم `databaseType: postgresql` یا حتی `none` را می‌پذیرد → سایت بالا نمی‌آید. باید هنگام رانتایم WordPress دیتابیس اجباراً MySQL شود.

ریسک‌ها

- **RISK** **ایمیج همه سرویس‌ها از Docker Hub** بدون مکانیزم آینه در چارت اپ (`postgres:16-alpine` , `mysql:8.0` , ...)؛ override `database.image` هست ولی backend هرگز آن را ست نمی‌کند.
- **RISK** **Deployment + PVC نوع RWO بدون `strategy: Recreate`** برای دیتابیس/Redis/RabbitMQ → در آپگرید ایمیج پاد جدید منتظر ولوم می‌ماند.
- **RISK** **fallback تولید رمز DB** (`kubernetes.service.ts:333`): اگر `dbPassword` خالی باشد هر دیپلوی رمز جدید می‌سازد و با داده قدیمی PVC ناسازگار می‌شود.
- **RISK** **خاموش‌کردن سرویس PVC یتیم جا می‌گذارد** — کاربر آن‌ها را نمی‌بیند ولی هزینه استوریج ادامه دارد.
- **RISK** **دسترسی خارجی host — NodePort اشتباه** (`kubernetes.service.ts:2691`): IP از API server گرفته می‌شود نه worker node؛ رشته اتصال بلااستفاده است. `suspend` هم گرنت‌های NodePort را باطل نمی‌کند.

۴. کنترل‌پلین، GitOps و CI/CD

باگ‌ها (باید قبل از دیپلوی بعدی رفع شوند)

BUG **سیستم migration در آپگرید دوم می‌شکند**

Job — `migrations-job.yaml:50-53` همه فایل‌های SQL را در هر اجرا دوباره اجرا می‌کند بدون جدول ردیابی نسخه.
`001_service_access_grants.sql:2,9` از `CREATE TYPE` بدون گارد استفاده می‌کند → آپگرید دوم:
`ERROR: type already exists` → sync fail.

BUG **migration هوک بعد از دیپلوی backend اجرا می‌شود**

`migrations-job.yaml:10-11` — `post-upgrade` ؛ backend جدید ممکن است قبل از آماده شدن اسکیما بالا بیاید → CrashLoop.
باید `pre-upgrade` باشد.

BUG **نبود base schema و نام ستون اشتباه در migration 015**

هیچ SQL جدول‌های `users` / `applications` را نمی‌سازد؛ روی دیتابیس خالی اولین migration شکست می‌خورد.
`015_application_product_type.sql:5-6` ستون `user_id` می‌سازد ولی entity آن را `userId` تعریف کرده
(`application.entity.ts:150`) → ساخت ایندکس fail.

رمزهای هاردکد شده در گیت

SECURITY **رمزهای الستیک‌سرچ در فایل commitشده**

`backend/k8s/logging/elasticsearch-stack.yaml:21-23` — `ELASTIC_PASSWORD: "CloudHost2024!Secure"` و
`FLUENTBIT_PASSWORD` . باید rotate و از گیت خارج شوند. همین‌ها به‌عنوان default در `configuration.ts:148-150` هستند و در
validate-production بررسی نمی‌شوند.

ریسک‌های CI/CD و کنترل‌پلین

- **RISK** workflow کامیت‌شده **auth کانیکو به Harbor** و **توکن clone** ندارد (→ `.gitea/workflows/build-deploy.yaml:74`)
  push/clone شکست می‌خورد؛ اصلاحات در تغییرات uncommit هستند.
- **RISK** **تست‌ها در مسیر Gitea اجرا نمی‌شوند** (فقط GitHub Actions) → کد خراب می‌تواند به پروداکشن برسد.
- **RISK** ایمیج backend حین بیلد **Helm و kubectl را از اینترنت دانلود می‌کند** (`backend/Dockerfile:16-20`) بدون پروکسی.
- **RISK** `git push` بدون `pull --rebase` (workflow:177) → احتمال half-done deploy.
- **RISK** postgres/redis پلتفرم در `values-abrban.yaml` آینه نشده و imagePullSecret ندارند.
- **RISK** `strategy: Recreate` روی (`backend-deployment.yaml:12` backend) → داون‌تایم کامل API در هر دیپلوی.
- **RISK** بدون resource limits در values پروداکشن → ریسک OOM روی k3s تک‌نود؛ Redis پلتفرم بدون `requirepass` ؛ backup
  پستگرس خاموش.
- **RISK** `docker compose up --build` کامل کار نمی‌کند — backend با `NODE_ENV=production` → `synchronize:false` و
  بدون migration → جدول‌ها موجود نیست.

۵. امنیت و کیفیت کد اپلیکیشن

حفره‌های امنیتی بحرانی (P0)

SECURITY **هر کاربر لاگین‌شده می‌تواند کیف پول خود را رایگان شارژ کند**

`billing-wallet.controller.ts:45-49` — `POST /billing/wallet/charge` بدون درگاه پرداخت مستقیم `chargeWallet` را صدا می‌زند → پول رایگان در پروداکشن. همچنین `gateway/verify` با `PAYMENT_GATEWAY_STUB_ENABLED=true` مبلغ دلخواه را می‌پذیرد.

SECURITY **دور زدن بیلینگ در deploy / start / resources**

`deployments.service.ts:637` `startDeployment` اپ suspendشده را بدون بررسی وضعیت/کیف پول resume می‌کند. `triggerDeployment` (دیپلوی اول) گارد بیلینگ ندارد. `applications.controller.ts:375` `PATCH resources` ارتقا را بدون مسیر پرداخت انجام می‌دهد.

SECURITY **تداخل namespace بین کاربران (۸ کاراکتر اول UUID)**

`kubernetes.service.ts:2687-2689` — `user-${userId.split('-')[0]}` ؛ دو کاربر با ۸ کاراکتر اول یکسان namespace مشترک و دسترسی به workload/secret همدیگر می‌گیرند. همین مشکل در ایزوله‌سازی لاگ الستیک (`elasticsearch.service.ts:676`).

امنیتی (P1)

- **SECURITY** `gitToken` و `dbPassword` در پاسخ API برمی‌گردند (`application.entity.ts:54,114`) — نیاز به `@Exclude` .
- **RISK** عملیات کیف پول بدون (`billing.service.ts:210` transaction/lock) — کسر هم‌زمان می‌تواند overdraw کند.
- **RISK** اسکنر auto-renew idempotent نیست بین رپلیکاها (`app-lifecycle.service.ts:39`) — دو پاد یک اپ را دوبار شارژ می‌کنند.
- **BUG** proration ارتقا همیشه نرخ ساعتی را استفاده می‌کند (`billing.service.ts:755`) → ارتقای ماهانه/سالانه undercharge یا رایگان.
- **SECURITY** توکن‌ها در `localStorage` (`frontend/src/lib/store.ts:43`) → در معرض XSS.
- **SECURITY** refresh token بدون rotation/ابطال و context جعل هویت روی refresh دوباره اعتبارسنجی نمی‌شود (`auth.service.ts:165`).

ریسک‌های متوسط

- **RISK** OTP با `Math.random()` به‌جای (`verification.service.ts:188` CSPRNG) و race در مصرف OTP (خط ۲۲۵).
- **RISK** Swagger بی‌قید در پروداکشن باز است (`main.ts:53`).
- **RISK** secretهای پیش‌فرض ضعیف خارج از پروداکشن (`configuration.ts:75` → `default-jwt-secret`).

اولویت‌بندی برای پروداکشن

باید قبل از هر دیپلوی پروداکشن رفع شود (بلاکر)

1. حذف/گیت کردن `POST /billing/wallet/charge` پشت درگاه پرداخت واقعی.
2. گارد بیلینگ روی `triggerDeployment` ، `startDeployment` و `PATCH resources` .
3. ساخت namespace از کل UUID، نه ۸ کاراکتر اول (تداخل بین‌مستأجری).
4. سیستم migration: جدول ردیابی نسخه یا SQL کاملاً idempotent + هوک `pre-upgrade` + base schema برای نصب تازه.
5. اصلاح `015` (`user_id` → `userId`) و گارد `duplicate_object` برای `CREATE TYPE` در `001` .
6. commit و deploy اصلاحات uncommit شده workflow (توکن Gitea + auth Harbor کانیکو).
7. rotate کردن رمزهای هاردکد الستیک‌سرچ.
8. رفع سینتکس `COPY` در Dockerfile گو و حذف `|| echo` از بیلد Node.

اولویت	اقدام
۹	الگوی `lookup` برای رمز Redis/RabbitMQ (توقف چرخش رمز).
۱۰	probe ردیس/مونگو با احراز هویت.
۱۱	آینه‌کردن base imageهای بیلد + ایمیج دیتابیس‌ها برای شبکه ایران.
۱۲	transaction/lock روی عملیات کیف پول.
۱۳	`strategy: Recreate` روی سرویس‌های stateful و `RollingUpdate` روی backend.
۱۴	رفع ImagePullBackOff در fallback بین‌کلاستری.
۱۵	حذف `gitToken` / `dbPassword` از پاسخ‌ها با `@Exclude` .
۱۶	اعتبارسنجی و کوت `gitBranch` ، انتقال توکن گیت به Secret.
۱۷	پشتیبانی MongoDB در snapshot، restore وردپرس از PVC به‌جای Secret.
۱۸	اجبار MySQL برای رانتایم WordPress.
۱۹	اجرای تست در مسیر Gitea قبل از دیپلوی.
۲۰	resource limits و backup پستگرس روی کنترل‌پلین.