Auctio

MANUALE DI INSTALLAZIONE ED ESERCIZIO

Il gestore del servizio

Installare, configurare, aprire gli account, tenere in esercizio: il servizio, i fornitori, i controlli, le emergenze.

Auctio — deposito cifrato delle offerte tecniche oltre i limiti di Sintel · https://auctio.it · versione del manuale: settembre 2026 (software 0.1, WP17) · ThinkDifferent S.r.l.

Indice

1. Cosa gira dove

Auctio è un unico binario Go (`auctio`) che serve le pagine e l'API, esegue lo scheduler dei lavori periodici e applica le migrazioni dello schema all'avvio. In produzione gira come **pod** di due container (podman rootless, quadlet di systemd): il servizio (`auctio`, immagine `localhost/auctio`, utente non privilegiato, filesystem in sola lettura) e PostgreSQL 16 (`auctio-db`, immagine bloccata per digest). Solo la porta del servizio esce dal pod, su loopback (`127.0.0.1:8630`); davanti c'è nginx, che termina TLS.

```
/var/auctio/
  auctio.env        l'ambiente del servizio                          0600
  auctio-db.env     l'ambiente di PostgreSQL                        0600
  secrets/          pgpass, db.password, smtp.password, tsa-chain.pem 0700, file 0600
  data/             keys/, log/ (registro), blobs/ (depositi cifrati) 0700
  pg/               i dati di PostgreSQL                             0700
  backup/           gli archivi notturni, gli ultimi quattordici     0700
~/.config/containers/systemd/
  auctio.pod  auctio-db.container  auctio.container
```

Ciò che deve sopravvivere a tutto è in `data/`: le chiavi (`keys/seal.pem`, il sigillo; `keys/secrets.key`, la chiave dei segreti a riposo), il registro trasparente (`log/`, tile e checkpoint, con l'origine bloccata in `log/origin`) e i depositi cifrati (`blobs/`). Il database tiene account, gare, depositi, ricevute, verbali, notifiche, il registro di sistema a catena di hash.

2. Requisiti e fornitori esterni

COSA	SERVE PER	NOTE
Host Linux con podman ≥ 4.4 (rootless) e systemd; 2 CPU, 4 GB, disco per i depositi	il pod	SELinux ammesso (le etichette sono nelle unit); un utente dedicato con `loginctl enable-linger`
nginx (o Plesk) con certificato TLS	il proxy in ingresso	capitolo 5
Autorità di marcatura temporale RFC 3161	la marca sulle ricevute e sui verbali	per valore legale pieno: un fornitore qualificato eIDAS (Aruba, InfoCert, Namirial…); URL, credenziali, catena dei certificati in PEM. In produzione il servizio rifiuta di partire senza (`AUCTIO_TIMESTAMPING=rfc3161`)
Ancoraggio	rendere immodificabile il registro	OpenTimestamps (Bitcoin) senza alcun account: i calendari pubblici; in aggiunta, opzionale, una catena EVM con un portafoglio (`docs/ANCHORING.md`)
Posta elettronica (SMTP con TLS)	conferme, codici, notifiche	una casella dedicata (es. `noreply@auctio.it`); opzionale una casella **PEC** per le comunicazioni a valore legale (scarico, chiusura, distruzione)
Accesso a VIES	la verifica della partita IVA alla registrazione	rete in uscita verso `ec.europa.eu`
Una macchina di build (Git, Go, docker o podman)	costruire l'immagine e verificarla	mai l'host di produzione
Un gestore di password fuori dall'host	l'identità dei backup, le credenziali dei fornitori	capitolo 11

3. Installazione da zero

La procedura completa è `docs/RUNBOOK.md` §2; qui l'ordine e i punti che non si possono sbagliare.

1. **Costruite e verificate l'immagine**
   sulla macchina di build, alla versione taggata che intendete installare:

```
git clone git@git.nexlab.net:nexlab/auctio.git && cd auctio && git checkout wp-17
deploy/build.sh --verify          # costruisce, salva dist/auctio-image.tar, confr
onta il binario
scp dist/auctio-image.tar dist/auctio-image.sha256 utente@host:
```

`--verify` ricompila il binario localmente e lo confronta byte per byte con quello nell'immagine: `VERIFIED`, o la build si ferma. Annotate il digest.

2. **Generate l'identità dei backup**
, sul vostro portatile, mai sull'host: `auctio backup keygen > auctio-backup.key`. Il file va nel gestore di password e in una copia stampata in cassaforte; la parte pubblica (`age1…`, stampata su stderr) va in `AUCTIO_BACKUP_RECIPIENTS`. Persa l'identità, ogni archivio è illeggibile.

3. **Sull'host, come root una volta:**
l'utente dedicato con linger, le directory:

```
useradd -m -s /bin/bash auctio && loginctl enable-linger auctio
mkdir -p /var/auctio/{data,pg,backup,secrets} && chown -R auctio:auctio /var/aucti
o && chmod 700 /var/auctio/*
```

4. **Come utente del servizio:**
caricate l'immagine, scrivete ambiente e segreti, riassegnate le directory all'utente non privilegiato dell'immagine:

```
sha256sum -c auctio-image.sha256 && podman load -i auctio-image.tar
podman tag localhost/auctio:<versione> localhost/auctio:latest
cp deploy/auctio.env.example /var/auctio/auctio.env && chmod 600 /var/auctio/aucti
o.env   # ogni CHANGE-ME
printf 'POSTGRES_USER=auctio\nPOSTGRES_PASSWORD_FILE=/secrets/db.password\nPOSTGRE
S_DB=auctio\n' > /var/auctio/auctio-db.env
cd /var/auctio/secrets
head -c 24 /dev/urandom | base64 | tr -d '\n' > db.password
printf '127.0.0.1:5432:auctio:auctio:%s\n' "$(cat db.password)" > pgpass
# smtp.password, pec.password, tsa-chain.pem (e le credenziali TSA) come li danno
i fornitori
chmod 600 /var/auctio/secrets/*
podman unshare chown -R 65532:65532 /var/auctio/data /var/auctio/backup /var/aucti
o/secrets
```

5. **Le chiavi, prima del primo avvio.**
In produzione il servizio non inventa mai un sigillo (uno nuovo renderebbe orfana ogni ricevuta):

```
alias auctio-run='podman run --rm -v /var/auctio/data:/data:z -v /var/auctio/secre
ts:/secrets:ro,z --env-file /var/auctio/auctio.env'
auctio-run localhost/auctio:latest keys init
auctio-run localhost/auctio:latest keys show      # il KID del sigillo: annotatelo
```

6. **Le unit e il primo avvio:**

```
mkdir -p ~/.config/containers/systemd
cp deploy/auctio.pod deploy/*.container ~/.config/containers/systemd/
systemctl --user daemon-reload && systemctl --user start auctio-pod
systemctl --user status auctio-db auctio      # due verdi, "healthy"
podman logs auctio | head -20
```

Il servizio applica lo schema, avvia lo scheduler e dichiara `production=true fake_adapters=null` nella prima riga di log: se elenca adattatori *fake*, l'ambiente è sbagliato e il servizio si è rifiutato di partire.

7. **Il primo gestore:**

```
podman exec auctio /auctio user add --role manager --login <nome> --email <indiriz
zo> --password-file /secrets/manager.initial
```

(la password in un file sotto `secrets/`, scritto con `podman unshare`; cancellatelo dopo il primo accesso). Al primo accesso il gestore configura l'app di autenticazione.

8. **Il proxy**
(capitolo 5), poi la prova: `curl https://<dominio>/healthz` → `{"database":"ok","status":"OK"}`; un giro completo come nel manuale della stazione appaltante, con una gara di prova a scadenza ravvicinata; la verifica della ricevuta con `auctio-verify`.

---

4. La configurazione

Tutto è ambiente (`/var/auctio/auctio.env`); i segreti sono file sotto `/secrets`, mai valori nell'ambiente. L'elenco completo con i valori predefiniti è `docs/CONFIGURATION.md`; qui ciò che una produzione deve decidere.

VARIABILE	VALORE IN PRODUZIONE	PERCHÉ
`AUCTIO_PRODUCTION`	`1`	accende la guardia: rifiuta adattatori finti, esige URL pubblico, contatti, informativa, origine del registro, TSA reale
`AUCTIO_MIGRATE_AT_START`, `AUCTIO_WORKER_INLINE`	`1`, `1`	un solo processo: schema applicato all'avvio, scheduler dentro il servizio
`AUCTIO_DATABASE_URL`	`postgres://auctio@127.0.0.1:5432/auctio?sslmode=disable&passfile=/secrets/pgpass`	la password da file; il database è nello stesso pod
`AUCTIO_PUBLIC_URL`	`https://auctio.it`	l'indirizzo in ogni collegamento spedito e stampato
`AUCTIO_LOG_ORIGIN`	`auctio.it/log/v1`	il nome del registro in ogni checkpoint, **per sempre**: bloccato al primo avvio, il servizio rifiuta di cambiarlo
`AUCTIO_OPERATOR_NAME`, `_CONTACT`, `_EMAIL`, `_LEGAL`, `AUCTIO_PRIVACY_URL`	ragione sociale; recapiti; indirizzo per le stazioni appaltanti; sede legale, P. IVA, PEC; `https://auctio.it/privacy`	il piè di pagina di ogni pagina, l'autore dei documenti, l'informativa
`AUCTIO_TIMESTAMPING=rfc3161`, `AUCTIO_TSA_URL`, `AUCTIO_TSA_USERNAME`, `AUCTIO_TSA_PASSWORD_FILE`, `AUCTIO_TSA_CA_BUNDLE`, `AUCTIO_TSA_POLICY`	come da fornitore; la catena in `/secrets/tsa-chain.pem`	la marca temporale; `auctio timestamp probe` prova l'endpoint
`AUCTIO_ANCHORING`	`ots` (o `ots,evm`)	l'ancoraggio dei tree head
`AUCTIO_NOTIFICATIONS=smtp`, `AUCTIO_SMTP_HOST`, `_PORT`, `_TLS`,	la casella del servizio; `tls` su 465 o `starttls` su 587	la posta; senza PEC le comunicazioni

`_FROM`, `_USERNAME`, `_PASSWORD_FILE`; `AUCTIO_PEC_*`		legali partono dalla casella e-mail
`AUCTIO_VAT_CHECK`	`vies`	la partita IVA verificata alla registrazione
`AUCTIO_TRUSTED_PROXIES`	l'indirizzo da cui nginx si connette, come lo mostra la prima riga di log	solo di lì si crede a `X-Forwarded-For`; dietro pasta il loopback dell'host appare come l'indirizzo pubblico dell'host
`AUCTIO_BACKUP_RECIPIENTS`, `AUCTIO_BACKUP_DIR=/backup`, `AUCTIO_BACKUP_KEEP`	le chiavi pubbliche age; 14	l'archivio notturno; senza destinatari il lavoro fallisce ogni notte, apposta
`AUCTIO_WORKER_NIGHT`, `TZ`	`03:30`, `Europe/Rome`	l'ora dei lavori notturni; le date nelle pagine e nei PDF
`AUCTIO_LANGUAGE`	`it`	pagine, messaggi, PDF

Limiti e ritmi con un predefinito sensato, da toccare solo con un motivo: `AUCTIO_CHUNK_DEFAULT` (dimensione dei blocchi), `AUCTIO_SERVE_MAX`, `AUCTIO_DISK_RESERVE` (spazio libero sotto il quale i caricamenti rispondono 507), `AUCTIO_UPLOAD_EXPIRY` (72 h), `AUCTIO_SESSION_TTL` (12 h) e `_IDLE` (30 min), `AUCTIO_STH_INTERVAL` (10 min), `AUCTIO_CHECKPOINT_INTERVAL` (10 s), gli intervalli dei lavori `AUCTIO_JOB_*_INTERVAL`.

> **Tre cose non si cambiano mai dopo il primo avvio:** `AUCTIO_LOG_ORIGIN`, la directory `data/log/` (né spostata né rinominata), il sigillo se non con `auctio keys rotate`.

5. Il proxy in ingresso

nginx termina TLS e inoltra a `127.0.0.1:8630`. La forma del blocco è in `deploy/auctio.it/zz020_auctio.conf` (l'installazione di riferimento, su Plesk): `proxy_http_version 1.1`, gli header `Host`, `X-Real-IP`, `X-Forwarded-For`, `X-Forwarded-Proto`, **nessun** `X-Forwarded-Prefix` (Auctio non accetta un percorso di base), `client_max_body_size 64m` (il PATCH del browser arriva al massimo a 32 MiB),

`proxy_request_buffering off` e `proxy_buffering off` (un deposito da 600 MB non deve toccare il disco di nginx), `proxy_read_timeout 600s`. HTTP → HTTPS, HSTS.

Su un host Plesk, un file in `/etc/nginx/conf.d/` con un nome che ordini *dopo* `zz010_psa_nginx.conf` (per esempio `zz020_auctio.conf`): incluso prima, diventerebbe il server predefinito dell'indirizzo e catturerebbe il pannello. Il certificato Let's Encrypt con `acme.sh` e la sfida HTTP su `/.well-known/acme-challenge/`, rinnovato da cron con ricarica di nginx.

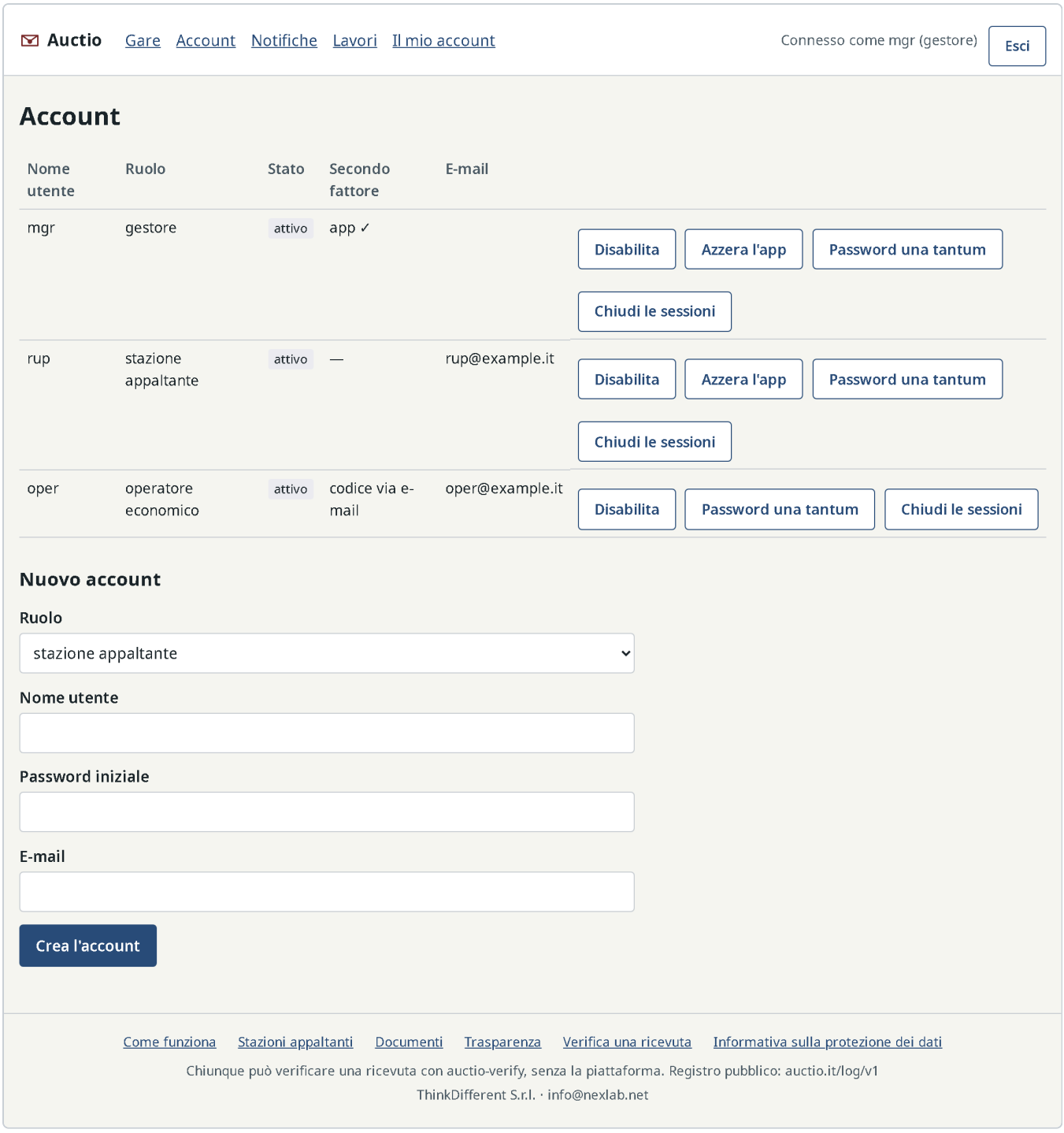
Dopo il primo avvio leggete nel log la riga di una richiesta arrivata attraverso nginx: l'`ip` che mostra è quello da mettere in `AUCTIO_TRUSTED_PROXIES`. Finché non lo fate, il servizio registra l'indirizzo del proxy anziché quello del client e lo avvisa nel log una volta al minuto.

---

6. Gli account

Tre ruoli. Un account ha un ruolo solo.

RUOLO	CHI LO CREA	SECONDO FATTORE	COSA PUÒ FARE
gestore (manager)	la riga di comando (il primo), poi la pagina «Account»	app di autenticazione (TOTP), configurata al primo accesso	gare, account, notifiche, lavori; mai leggere un deposito
stazione appaltante (authority)	il gestore, dalla pagina «Account» o da riga di comando, dopo verifica dell'ente	app di autenticazione	chiave e apertura delle gare assegnate
operatore economico (operator)	si registra da solo (`/register`), con conferma dell'e-mail e verifica della partita IVA	codice via e-mail a ogni accesso	depositare nelle gare aperte



«Account»: creazione, disattivazione, azzeramento dell'app, password una tantum, chiusura delle sessioni.

Creare una stazione appaltante

Dalla pagina «Account»: ruolo *stazione appaltante*, nome utente, e-mail, la password iniziale che scegliete voi. Consegnatela per un canale diverso da quello con cui avete comunicato il nome utente (telefono, PEC). Oppure da terminale: `podman exec auctio /auctio user add --role authority --login <nome> --email <indirizzo> --password-file /secrets/<file>`.

Le situazioni ricorrenti

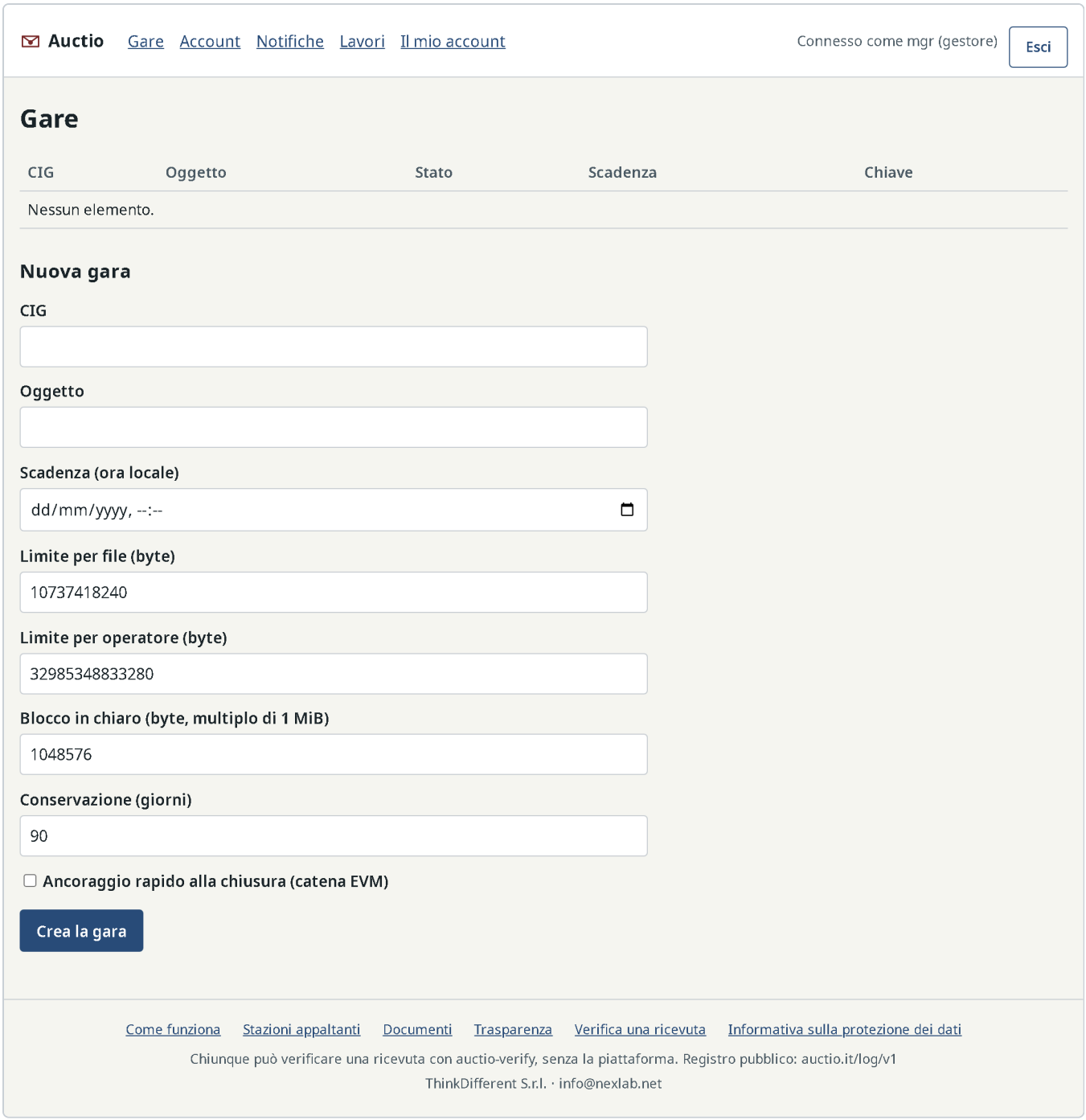
- **telefono perso** (gestore o stazione appaltante): «Azzera l'app» — l'account riconfigura l'autenticatore al prossimo accesso; prima, accertatevi di chi chiede;

- **password dimenticata**: gli operatori usano «Password dimenticata?»; per gestori e stazioni appaltanti «Password una tantum», stampata una volta, con chiusura delle sessioni;
- **partita IVA non verificata** (VIES non rispondeva): `auctio user verify-vat LOGIN` ritenta; `auctio user activate LOGIN` conferma a mano una registrazione la cui e-mail non arriva;
- **uscita di una persona**: «Disabilita» chiude l'account e ogni sessione; i suoi depositi e le sue gare restano.

---

7. Le gare

La gara la crea il gestore su richiesta della stazione appaltante: CIG, oggetto, scadenza (giorno, ora e minuto, ora italiana — la stessa di Sintel), limiti per file e per operatore, giorni di conservazione dopo l'apertura, eventuale ancoraggio rapido (EVM) per la chiusura.



«Gare»: l'elenco e il modulo di creazione.

1. **Creare**
   la gara: nasce in *bozza*.
2. **Assegnare**
   la stazione appaltante (una o più) dalla pagina della gara.
3. **Attendere la chiave.**
   La stazione appaltante genera e pubblica la chiave; la pagina della gara mostra l'impronta quando c'è.
4. **Aprire**

la gara («Apri la gara»): da questo momento accetta depositi e la chiave non può più cambiare. Comunicate alla stazione appaltante l'indirizzo della pagina pubblica (`/t/<identificativo>`).

5. **Non fare nulla**
   fino alla scadenza: la chiusura è automatica, al minuto, con verbale, marca, iscrizione nel registro e notifiche. Se la stazione appaltante proroga, «Sposta la scadenza»; se annulla, «Annulla la gara» (i depositi vengono distrutti).

6. **Dopo l'apertura**
   da parte della stazione appaltante, la conservazione decorre e la distruzione avviene da sola al termine. «Dichiara completata l'apertura» esiste anche qui, per una stazione appaltante che non l'ha fatto.

A1B2C3D4E5 — Servizio di progettazione esecutiva — lotto 2

bozza · /t/14a074fc6bf6f18656673bf4abcc9080

Scadenza	19/09/2026 13:34 CEST (`2026-09-19T11:34:00Z`)
Impronta della chiave pubblica della gara	La chiave della gara non è ancora pubblicata.
Dimensione massima del file	10.00 GiB
Volume massimo per operatore	30.0 TiB
Dimensione dei blocchi	1.0 MiB
Conservazione dopo l'apertura	90 giorni

Apri la gara | Annulla la gara

Salva

CIG

A1B2C3D4E5

Oggetto

Servizio di progettazione esecutiva — lotto 2

Scadenza (ora locale)

19/09/2026, 13:34

Limite per file (byte)

10737418240

Limite per operatore (byte)

32985348833280

Blocco in chiaro (byte, multiplo di 1 MiB)

1048576

Conservazione (giorni)

90

☐ **Ancoraggio rapido alla chiusura (catena EVM)**

Salva

Stazione appaltante

- rup Rimuovi

Depositi

Identificativo	Operatore	Stato	Dimensione	Completato il
Nessun elemento.				

Registro di sistema della gara

Quando	Evento	Attore
19/09/2026 13:29 CEST	`tender.created` `14a074fc6bf6f18656673bf4abcc9080`	`ff8e8346cf5fe81cfd212b9e80196dcb`

La gara: stato, stazioni appaltanti assegnate, depositi, il registro di sistema della gara, le azioni.

A1B2C3D4E5 — Servizio di progettazione esecutiva — lotto 2

in apertura · /t/14a074fc6bf6f18656673bf4abcc9080

Scadenza	19/09/2026 13:34 CEST (`2026-09-19T11:34:00Z`)
Impronta della chiave pubblica della gara	`697dbcf3b2009ce55718a80c59a98eb10ee047dbd014cdb11cbb78d60c655ffc`
Dimensione massima del file	10.00 GiB
Volume massimo per operatore	30.0 TiB
Dimensione dei blocchi	1.0 MiB
Conservazione dopo l'apertura	90 giorni
Chiusa il	19/09/2026 13:34 CEST

Dichiara completata l'apertura

Stazione appaltante

- rup Rimuovi

Depositi

Identificativo	Operatore	Stato	Dimensione	Completato il
`7cfe55296ca4238797bea8466b9e77c0`	`7ae84617f8bea79c1e868cbd72e4774e`	completato	6.0 MiB	19/09/2026 13:30 CEST

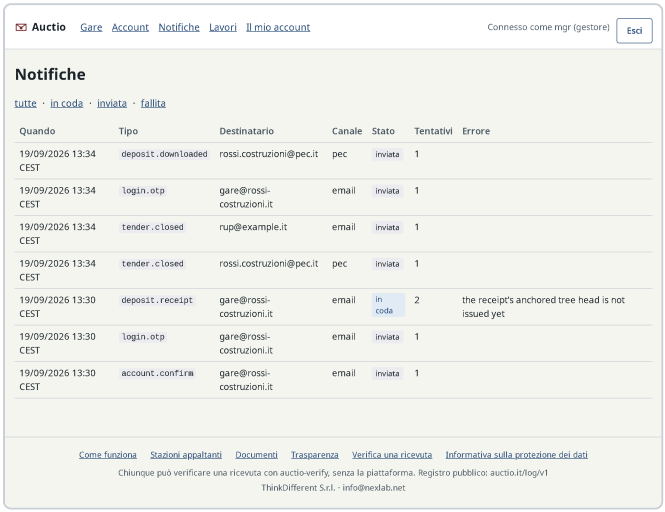
Registro di sistema della gara

Quando	Evento	Attore
19/09/2026 13:29 CEST	`tender.created 14a074fc6bf6f18656673bf4abcc9080`	`ff8e8346cf5fe81cfd212b9e80196dcb`
19/09/2026 13:29 CEST	`tender.authority_assigned 1b787a00fdd0a0d868695409fc82de0d`	`ff8e8346cf5fe81cfd212b9e80196dcb`
19/09/2026 13:29 CEST	`tender.keys_published 14a074fc6bf6f18656673bf4abcc9080`	`1b787a00fdd0a0d868695409fc82de0d`
19/09/2026 13:30 CEST	`tender.opened 14a074fc6bf6f18656673bf4abcc9080`	`ff8e8346cf5fe81cfd212b9e80196dcb`
19/09/2026 13:30 CEST	`deposit.started 7cfe55296ca4238797bea8466b9e77c0`	`7ae84617f8bea79c1e868cbd72e4774e`
19/09/2026 13:30 CEST	`deposit.chunk_received 7cfe55296ca4238797bea8466b9e77c0`	`7ae84617f8bea79c1e868cbd72e4774e`
19/09/2026 13:30 CEST	`deposit.chunk_received 7cfe55296ca4238797bea8466b9e77c0`	`7ae84617f8bea79c1e868cbd72e4774e`
19/09/2026 13:30 CEST	`deposit.completed 7cfe55296ca4238797bea8466b9e77c0`	`7ae84617f8bea79c1e868cbd72e4774e`
19/09/2026 13:30 CEST	`timestamp.obtained 7cfe55296ca4238797bea8466b9e77c0`	`system`
19/09/2026 13:30 CEST	`receipt.issued 7cfe55296ca4238797bea8466b9e77c0`	`system`
19/09/2026 13:34 CEST	`tender.closed 14a074fc6bf6f18656673bf4abcc9080`	`cli`
19/09/2026 13:34 CEST	`closure.issued 14a074fc6bf6f18656673bf4abcc9080`	`system`
19/09/2026 13:34 CEST	`deposit.downloaded 7cfe55296ca4238797bea8466b9e77c0`	`1b787a00fdd0a0d868695409fc82de0d`
19/09/2026 13:34 CEST	`chunk.served 7cfe55296ca4238797bea8466b9e77c0`	`1b787a00fdd0a0d868695409fc82de0d`
19/09/2026 13:34 CEST	`opening.outcome 7cfe55296ca4238797bea8466b9e77c0`	`1b787a00fdd0a0d868695409fc82de0d`

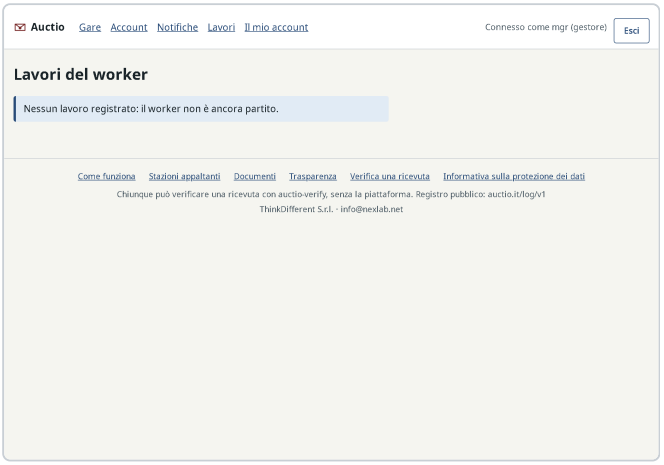
La gara chiusa: il verbale, i depositi elencati.

8. L'area del gestore

- **Gare** e la pagina di ogni gara: capitolo 7; in fondo, il registro di sistema della gara (chi ha fatto cosa, quando).
- **Account**: capitolo 6.
- **Notifiche**: la coda di uscita della posta — in attesa, inviate, fallite — con «Riprova». Un messaggio fallito ripetutamente è un problema del fornitore di posta o un indirizzo errato.
- **Lavori**: l'ultima esecuzione di ogni lavoro periodico con esito e riepilogo. È la prima pagina da guardare quando qualcosa non torna.



«Notifiche»: la coda di uscita.



«Lavori»: ogni lavoro con esito e riepilogo.

9. Gli endpoint

Tutto passa dallo stesso servizio, sulla stessa porta. Le pagine e l'API condividono sessione, autorizzazioni e regola CSRF; la matrice completa route × ruolo è `docs/AUTHORIZATION.md`, generata da un test.

PERCORSO	CHI	COSA
`/`, `/come-funziona`, `/stazioni-appaltanti`, `/trasparenza`, `/privacy`, `/documenti`	tutti	il sito
`/t/{id}`	tutti	la pagina pubblica di una gara; dopo la chiusura, il verbale
`/login`, `/register`, `/confirm`, `/reset`, `/account`	tutti / sessione	accesso, registrazione, conferma, reset, account
`/me`, `/t/{id}/deposit`	operatore	le gare, il deposito
`/authority`, `/t/{id}/key`, `/t/{id}/open`	stazione appaltante	le gare, la chiave, l'apertura
`/manager/tenders`, `/manager/accounts`, `/manager/notifications`, `/manager/jobs`	gestore	l'area del gestore
`/verify/`	tutti	il verificatore nel browser (WebAssembly)
`/api/…`	per ruolo	l'API JSON (`docs/API.md`): sessione, account, gare, depositi (tus per il caricamento), ricevute e verbali in JSON e PDF
`/api/status`	tutti	versione, schema, impronte dei moduli client
`/healthz`	tutti (nginx, systemd)	`{"database":"ok","status":"OK"}` o 503; è il controllo di salute del container
`/log/checkpoint`, `/log/tile/…`, `/log/keys`, `/log/sth`, `/log/sth/{n}`, `/log/proof/inclusion/{i}`, `/log/proof/consistency/{a}/{b}`	tutti	il registro trasparente nel formato tlog-tiles, le chiavi, i tree head ancorati, le prove
`127.0.0.1:8631/healthz` (dentro il pod)	il gestore	la salute dello scheduler, con lo stato di ogni lavoro; `auctio worker health`

10. L'esercizio: cosa gira da solo, cosa controllare

Lo scheduler dentro il servizio esegue i lavori qui sotto; ognuno esiste anche come comando (`docs/OPERATIONS.md`).

LAVORO	QUANDO	COSA FA
`tender.close`	alla scadenza più vicina, al secondo	chiude le gare scadute: verbale, marca, tree head, notifiche
`notify.drain`	ogni minuto	spedisce la posta in coda
`receipt.reissue`	ogni 5 min	emette le ricevute rinviate da un'interruzione della TSA
`sth.issue`	ogni 10 min se il registro è cresciuto	iscrive la testa del registro di sistema nel registro trasparente, emette un tree head marcato e ancorato
`anchor.upgrade`	ogni 10 min	conferma gli ancoraggi in attesa
`notify.remind`, `deposit.expire`, `tender.destroy`	ogni ora	promemoria di scadenza, eliminazione dei caricamenti incompleti (72 h), distruzione dopo la conservazione
`session.purge`, `deposit.scrub`, `integrity.verify`, `backup.nightly`	di notte (`AUCTIO_WORKER_NIGHT`)	pulizia; ricalcolo dell'impronta di ogni deposito; verifica del registro di sistema e del registro trasparente contro l'ultima testa ancorata; l'archivio cifrato

Ogni giorno (un minuto)

```
podman exec auctio /auctio worker status     # ogni lavoro ok
podman exec auctio /auctio notify list failed
podman exec auctio /auctio backup list       # un archivio nuovo ogni notte
df -h /var/auctio
```

Ogni settimana

Sulla macchina di build `make audit` (vulnerabilità note, SBOM, moduli più nuovi) e `deploy/pin.sh` (un `MOVED` segnala un'immagine di base aggiornata: leggete le note, aggiornate il pin, ricostruite con `--verify`, installate al prossimo rilascio). Sull'host: che il rsync notturno di `/var/auctio/data/blobs` e `/var/auctio/backup` sia andato.

Una volta l'anno, o quando una persona che poteva leggere il file lascia

`auctio keys rotate` ritira il sigillo corrente e ne crea uno nuovo; i sigilli ritirati restano per la verifica e `/log/keys` li elenca tutti. La chiave dei segreti non si ruota mai.

---

11. Backup, prova di ripristino, ripristino

Ogni notte, dopo la verifica d'integrità, il servizio scrive `/var/auctio/backup/auctio-backup-<istante UTC>.tar.age`: un archivio cifrato per `AUCTIO_BACKUP_RECIPIENTS` con ogni tabella del database (copiata in una sola transazione), le chiavi, il registro trasparente, l'origine. **I depositi cifrati non sono nell'archivio**: sono immutabili e grandi; il rsync notturno dell'host deve portare fuori `/var/auctio/data/blobs` e `/var/auctio/backup` (i file appartengono a un uid subordinato: il rsync gira come root, o con `podman unshare`).

> L'archivio è illeggibile senza l'identità (`auctio-backup.key`): una copia della directory dei backup non rivela nulla; perdere l'identità perde ogni archivio. L'identità non sta mai sull'host né dove stanno gli archivi.

La prova, prima di ogni rilascio

Su una macchina che non è l'host: un PostgreSQL vuoto, una directory dati vuota con i blob copiati dentro, l'identità, l'archivio più recente:

```
podman run -d --name drill-db -e POSTGRES_USER=auctio -e POSTGRES_PASSWORD=drill -e POSTGRES
_DB=auctio -p 127.0.0.1:5433:5432 docker.io/library/postgres:16
mkdir -p /tmp/drill/data && rsync -a host:/var/auctio/data/blobs /tmp/drill/data/
AUCTIO_DATABASE_URL=postgres://auctio:drill@127.0.0.1:5433/auctio AUCTIO_DATA_DIR=/tmp/dril
l/data \
  auctio restore auctio-backup-<istante>.tar.age --identity auctio-backup.key
```

`restore` carica l'archivio in un database vuoto e una directory vuota, poi verifica il registro di sistema, il registro trasparente e, con i blob, ogni deposito: un esito pulito è la prova che il backup vale. Annotate data ed esito.

Il ripristino dopo una perdita

La stessa procedura sull'host ricostruito (capitolo 3 fino alle unit, senza `keys init`: le chiavi sono nell'archivio), poi il pod. Il servizio riparte con lo stesso sigillo, lo stesso registro, la stessa origine: ricevute e verbali emessi prima restano verificabili. `docs/RUNBOOK.md` §5 in dettaglio.

---

12. Aggiornamento

```
# macchina di build
git checkout <tag> && deploy/build.sh --verify && scp dist/auctio-image.* utente@host:
# host
sha256sum -c auctio-image.sha256 && podman load -i auctio-image.tar
podman tag localhost/auctio:<nuova> localhost/auctio:latest
systemctl --user restart auctio-pod
systemctl --user status auctio-db auctio
```

Il servizio applica ciò che lo schema ha guadagnato (nulla, se non è cambiato) e riparte. Leggete prima le note di rilascio: dicono se lo schema cambia, cosa guadagna l'ambiente, se le unit sono cambiate (allora ricopiatele e `daemon-reload`). Aggiornate di notte: le chiusure avvengono alle scadenze. **Ritorno indietro**: `podman tag localhost/auctio:<precedente> localhost/auctio:latest` e riavvio; se lo schema era avanzato, `migrate down` una volta per migrazione aggiunta, poi il riavvio.

13. Emergenze

SINTOMO	CAUSA PROBABILE	COSA FARE
I caricamenti rispondono 507	spazio libero sotto `AUCTIO_DISK_RESERVE`	liberate: journal (`journalctl --user --vacuum-time=30d`), immagini vecchie (`podman image prune`), archivi oltre i quattordici se il rsync li ha. **Mai** sotto `data/`
Le ricevute risultano «rinviate»	la TSA non risponde	`auctio timestamp probe` ; quando torna, `receipt.reissue` le emette; se la CA della TSA è cambiata, aggiungete la nuova catena a `tsa-chain.pem`
Ancoraggi in attesa da ore	i calendari OpenTimestamps o la catena EVM non raggiungibili	si confermano da soli quando tornano; `auctio anchor status`
Posta in coda o fallita	il fornitore SMTP, o credenziali scadute	`auctio notify probe <indirizzo>` , poi «Riprova» dalla pagina o `auctio notify retry ID`
`deposit.scrub` fallisce	un blob danneggiato o cancellato	è un incidente: il lavoro nomina il deposito; ripristinate il blob dal rsync; informate la stazione appaltante se la gara è aperta
`integrity.verify` fallisce	il registro di sistema o il registro trasparente non tornano con l'ultima testa ancorata	è un incidente di sicurezza: fermate il servizio, conservate tutto, `auctio verify-log` e `auctio log verify` dicono dove; `docs/THREAT-MODEL.md`
Il servizio rifiuta di partire: «schema version mismatch», «seal key», «log origin»	binario e dati non coerenti	leggete il messaggio: è esatto; non forzate mai; capitolo 12 per il ritorno indietro
Il pannello dell'host o un altro sito sono spariti dietro Auctio	il file di nginx è incluso prima di quelli dell'host	capitolo 5: il nome del file deve ordinare dopo
Contenzioso su un deposito	—	tutto è verificabile da fuori: `auctio-verify receipt --file … --closure …` con `/log/keys` salvato; il registro di sistema della gara nella pagina del gestore; `docs/VERIFICA.md`

14. Mai

- Scaricare `localhost/auctio:latest` o `postgres:16` da un registro in produzione: l'immagine è il tar verificato, il database è il digest nella unit.
- Due pod sulla stessa directory dati o sullo stesso database.
- Modificare a mano sotto `data/` ; rinominare o spostare `log/` ; cambiare `AUCTIO_LOG_ORIGIN` .
- Tenere l'identità dei backup sull'host, o insieme agli archivi.
- Ripristinare su qualcosa che non è vuoto: `restore` rifiuta; non fatelo smettere.

- Aprire una gara senza che la stazione appaltante abbia verificato il file della chiave e stampato il kit.

---

15. La riga di comando

`podman exec auctio /auctio <comando>`, con l'ambiente del container; ogni comando ha `--help`.

```
serve                 il servizio (con AUCTIO_WORKER_INLINE anche lo scheduler)
migrate up|down|status
verify-log [--file export.jsonl]      il registro di sistema, ricalcolato
keys init|show|rotate                 il sigillo e la chiave dei segreti
user add|list|activate|deactivate|reset-totp|reset-password|verify-vat|otp|sessions purge
tender list|show|create|close-due|destroy-due
deposit list TENDER|expire|scrub
log show|sth|verify|reissue           il registro trasparente
timestamp probe|verify                l'adattatore RFC 3161
anchor status|wallet|upgrade|verify|ots info|contract deploy
notify drain|list [STATE]|retry ID|remind|probe ADDRESS
worker [status|run JOB|jobs|health]
backup [list|keygen]                  l'archivio cifrato, ora
restore ARCHIVE --identity FILE
health                                /healthz del servizio in esecuzione
load …                                la prova di carico (mai in produzione)
version
```

Lo strumento di verifica esterno, `auctio-verify`, è distribuito con ogni rilascio per Linux, macOS e Windows con il file `SHA256SUMS`; la guida è `docs/VERIFICA.md`.