Hugging Face Deployment Guide

PF Simulation r34 - public service and schema 10

Public registration, shared admission, durable email, lifecycle accounting and execution recovery. Native HF, managed HF and distributed Linux/Docker deployment paths.

WHERE EACH STEP RUNS

Browser - HF website, email provider or DNS dashboard.
Windows CMD - Python and HF CLI commands on your computer.
Administrator PowerShell - Windows/WSL setup only.
Linux shell - distributed services, workers and database administration.
Docker Desktop - local container tooling when using that option.

START WITH THE PUBLIC SERVICE DEPLOYMENT CHAPTER

Both the Qualified lifecycle package and Recovery Correction are integrated. Conservative accounting remains the default; safe settlement and audited historical reconciliation are explicit controls. Extract the full r34 release into a clean directory and do not replay historical patch installers.

Existing native HF upgrades preserve the Space, secrets, bucket, state key, database credentials and quota history. Schema 10 is additive and requires stopped writers. Follow the full-release upgrade chapter and verify the resulting runtime before opening registration.

Native HF has one fenced controller. Distributed managed HF requires external PostgreSQL and coordinated replicas; it is an architectural choice, not a variable that makes a native controller horizontally scalable. Linux/Docker remains a supported separate path.

CAPACITY MUST BE MEASURED

This release does not establish support for hundreds of thousands of concurrent active users. The included mixed-workload tooling, readiness gates and deployment-specific evidence are required to qualify the provisioned infrastructure, provider quotas and chosen workload before expansion.

COMMAND COPYING

PDF commands wrap visually. Copy exact command lines from the matching packaged Markdown source; every chapter is also attached inside this PDF. Run one command per original line in the named shell. Replace placeholders before running.

Contents

r34 HF SMTP public-readiness correction 6

r34 public service deployment and scale qualification 7

- Choose the architecture before opening registration . . . 7
- Maintain privacy while making the application reachable . . . 7
- Public registration and real account email . . . 8
- Admission is shared across organizations . . . 8
- Schema and lifecycle upgrade . . . 9
- Qualify and expand gradually . . . 9

Public account email delivery 11

- Hugging Face setup . . . 11
- Durable delivery and recovery . . . 12
- Worker placement and inspection . . . 12
- Acceptance before opening registration . . . 13

Public registration, ingress identity and launch evidence 14

- Authentication and administrative authority . . . 14
- Configure the actual proxy contract . . . 14
- Shared limits and overload behavior . . . 15
- Offline review and evidence . . . 16
- HF application and source visibility . . . 17

Account-wide public admission 18

- Accounting and fairness . . . 18
- Migrate existing usage before enabling limits . . . 19
- Verification . . . 20

Assistant execution recovery 21

- HF profiles . . . 21
- Distributed Docker . . . 21
- Bounds and rollout . . . 22

Shared HF accounting and recovery 23

Defaults and online policy changes . . . 23
Migration and historical charges . . . 23
External database recovery protection . . . 24
Administration and operational evidence . . . 25

Consolidated r34: redeploy the existing native HF service 26

1. Extract and configure local tools . . . 26
2. Drain, pause and back up existing state . . . 26
3. Generate and qualify matching code . . . 27
4. Update code pins, preserving state configuration . . . 27
5. Prepare, apply and verify the controller . . . 27
6. Accept the new release before reopening access . . . 28

PF Simulation r34 - complete Hugging Face deployment guide 29

Consolidated r34 starting point . . . 29
1. Create the HF resources and their credentials . . . 30
2. Prepare the full release and Windows tools . . . 31
3. Choose the Docker tooling route . . . 32
4. Build and upload the runtime and private code . . . 32
5. Resolve the public image and qualify one HF Job . . . 33
6. Set up email and generate stable secrets . . . 35
7. Configure the private controller Space . . . 36
8. Upload the controller and verify startup . . . 39
9. Qualify the complete deployment before researcher access . . . 40
10. Troubleshoot the failed checkpoint . . . 41
11. Upgrade an existing r34 deployment . . . 48
Deployment-wide cost admission and storage maintenance . . . 54
Increasing native capacity without replicating the controller . . . 55
Architecture and execution limits . . . 56
Operation, restart, and recovery . . . 57
Compatibility and evidence . . . 58
Official platform references . . . 59

Optional Hugging Face deployment with managed PostgreSQL 60

1. Choose the operation . . . 60
2. Database prerequisites . . . 60
3. Extract the release and build the contexts . . . 61
4. Preserve or configure the Space settings . . . 62
5. Prepare and install maintenance . . . 64
6. Activate and qualify . . . 65
7. Failure, resume and upgrade boundaries . . . 65
Qualification status . . . 66

r34 shared capacity settings 67

Database connections and deadlines . . . 67
HTTP and browser work . . . 68
Assistant and provider admission . . . 69
Jobs, maintenance and state . . . 70

Measuring the configured SaaS capacity 71

Workload and distinct users . . . 71
Start small, then add independent generators . . . 72
Interpret the report and correlate the deployment . . . 73
Mixed workloads and isolated target guard . . . 74
Shard-local roster and synchronized launch . . . 75
Aggregate evidence, soak and failure recovery . . . 76

Existing HF installation: upgrade and recovery 78

1. Record the current installation . . . 78
2. Pause and make a verified recovery copy . . . 78
3. Generate and qualify the replacement . . . 78
4. Apply matching controller settings and code . . . 79
5. Verify recovery and reopen access . . . 79
Recovery and rollback . . . 79

Native HF deployment record template 80

Release and environment . . . 80
Existing-deployment checkpoint . . . 80
Replacement identities . . . 81

Acceptance record . . . 81
Packaged operational entry points . . . 82
Consolidated r34 acceptance addendum . . . 82

r34 Hugging Face Job identity verification 83

Deployment and qualification . . . 83
Acceptance scope . . . 83

r34 Hugging Face PostgreSQL readiness and controller inputs 84

Restored PostgreSQL startup . . . 84
Generated controller Docker context . . . 84
Acceptance scope . . . 84

r34 Hugging Face WAL catch-up and batching 85

Receiver behavior . . . 85
Bounded startup setting . . . 85
Existing deployment . . . 85
Acceptance scope . . . 86

r34 result publication and verification 87

Saved-result behavior . . . 87
Deployment and collection limits . . . 87
Verify a representative workload . . . 87
Troubleshoot publication . . . 88
Validation scope . . . 88

r34 HF SMTP public-readiness correction

The public-readiness checker previously required Resend for HF, although the r34 runtime already supported encrypted SMTP on permitted outbound ports. That checker restriction and the guidance requiring a provider migration were inconsistent with the supported runtime. This correction retains both transports without changing SMTP delivery behavior or weakening live-email requirements.

The existing SMTP2GO configuration is supported:

TEXT | Copy exact lines from the packaged Markdown

```
PF_SAAS_MAIL_MODE=smtp
PF_SAAS_SMTP_HOST=mail.smtp2go.com
PF_SAAS_SMTP_PORT=443
PF_SAAS_SMTP_SSL=true
PF_SAAS_SMTP_STARTTLS=false
```

Preserve the existing verified `PF_SAAS_SMTP_FROM`, SMTP username/password Secrets, session key and account state. No provider migration is required. Resend HTTPS remains an optional transport. HF SMTP configuration checks require a provider-supported endpoint on port 80, 443 or 8080 and exactly one TLS mode. Permitted port numbers alone do not establish connectivity or provider support.

Readiness evidence now binds the effective sender and SMTP endpoint/TLS settings to its configuration fingerprint. Existing evidence becomes invalid when those settings or the checked security source change. Regenerate the report and repeat the applicable verification, password-reset, invitation and single-use-token tests before retaining new evidence. The offline checker does not send mail, prove inbox delivery, open registration or qualify the deployed HF service.

Apply the correction to the extracted release before generating deployment contexts. If contexts were already generated, regenerate them into a new output directory and verify the new bundle; use its resulting hashes and code pins. Do not upload an older generated context or manually edit sealed generated files to bypass identity checks. Preserve existing keys, state and recovery records.

See public email delivery (PUBLIC_EMAIL_DELIVERY.md) and public launch readiness (PUBLIC_LAUNCH_READINESS.md) for setup and acceptance.

r34 public service deployment and scale qualification

This release integrates `PF_r34_HF_Lifecycle_Qualified_20261009` followed by `PF_r34_HF_Recovery_Correction_20261009` into the full release whose incoming SHA256 is `140cd2f3bbf55dc48f4095566c0dc9f367d85772666beb5fa60c87dac80d32de`. It also extends public registration, durable email, account-wide admission, distributed recovery, managed HF accounting and mixed-workload qualification. Do not install those patch ZIPs again over this release.

The software can be configured from one user upward. **This archive does not certify hundreds of thousands of simultaneously active users.** That target requires a provisioned fleet, provider allocations and a successful measured campaign with the intended workload, data volume, regions and failure cases. Changing a numeric limit or accepting 200,000 credentials is not that evidence.

Choose the architecture before opening registration

Deployment	Data authority and execution	Public-service scope
Standalone	Local application and local files	Preserved; no SaaS infrastructure required
Native HF, profile `single`	One fenced controller, supervised PostgreSQL, encrypted HF recovery, HF Jobs	Existing deployment upgrade; bounded pilot after readiness checks; never run two native controllers against one prefix
Managed HF, profile `hf_managed`	Shared external PostgreSQL, encrypted HF objects, scoped leases, HF Jobs	Multiple coordinated API/worker replicas; requires external database, shared-state design, approved HF capacity and network connectivity
Linux/Docker, profile `distributed`	Shared PostgreSQL, private S3, dedicated Docker workers, private shared assistant/BYOS files	Independent API and worker scaling behind a controlled HTTPS ingress

The existing `ignalum/pf-simulation-saas` can first receive the full native upgrade while retaining its identity, users, keys and storage. This does not turn its single controller into a fleet. A migration to managed HF or a distributed Linux/Docker deployment is a separate, explicit architecture step. Neither the release builder nor its readiness checker provisions resources.

For native upgrade follow the existing-service guide (R34_NATIVE_HF_RELEASE_UPGRADE.md). For managed HF follow the managed migration guide (HUGGING_FACE_MANAGED_DEPLOYMENT.md). For Linux/Docker follow the distributed guide (../DISTRIBUTED_SAAS_DEPLOYMENT.md).

Maintain privacy while making the application reachable

For HF, a **Protected** controller Space makes the running application reachable while retaining private repository source. Keep the scientific code Space, Bucket, backups and credential material private. Do not switch the code Space or the full source repository to Public. Protected visibility requires a supporting HF plan; see the official visibility documentation. The deployment tools never change visibility for you.

Use the actual HTTPS application origin as `PF_SAAS_PUBLIC_URL`, with no path. Verify direct and embedded access, cookie/CSRF behavior, origin checks and the exact forwarded-address contract before allowing untrusted users. See public launch readiness (PUBLIC_LAUNCH_READINESS.md). Never enter `0.0.0.0/0` or an invented HF proxy network to make rate limits appear functional. If HF does not supply a verifiable ingress contract for the chosen topology, use a controlled ingress/gateway and prove that its origin cannot be bypassed.

Only platform administrators may open HF accounting/recovery pages or APIs. Organization owners and organization administrators do not acquire this role. Registration creates a non-administrator account and requires email verification. Private assistant workspaces, project grants, BYOK and BYOS remain user scoped.

Public registration and real account email

Keep the existing `PF_SAAS_SECRET_KEY`: it protects sessions and encrypts queued account email. Retain the existing HF state key and BYOK/BYOS credential keys. Do not regenerate them during an upgrade.

For local accounts configure:

TEXT | Copy exact lines from the packaged Markdown

```
PF_SAAS_AUTH_MODE=local
PF_SAAS_REGISTRATION_ENABLED=true
PF_SAAS_PUBLIC_URL=https://YOUR_APPLICATION_ORIGIN
PF_SAAS_MAIL_DELIVERY_WORKER_ENABLED=true
```

Retain existing encrypted SMTP, or choose the optional Resend HTTPS provider. For SMTP2GO on HF, preserve `PF_SAAS_MAIL_MODE=smtp`, host `mail.smtp2go.com`, port `443`, `PF_SAAS_SMTP_SSL=true`, `PF_SAAS_SMTP_STARTTLS=false`, the verified `PF_SAAS_SMTP_FROM` and existing SMTP credentials in Secrets. No provider migration is required. For Resend use `PF_SAAS_MAIL_MODE=resend`, a verified `PF_SAAS_MAIL_FROM` and `PF_SAAS_MAIL_API_KEY` in the secret manager/HF Secrets, never a public Variable. The HTTPS provider uses a fixed endpoint and bounded timeouts. Both transports use the encrypted durable outbox. Registration, password-reset and invitation tokens are committed atomically with their queued email. Delivery is asynchronous; HTTP acceptance and provider acceptance do not prove arrival in an inbox. Follow email setup, retry and diagnosis (PUBLIC_EMAIL_DELIVERY.md), including real delivery and single-use-token checks on an isolated staging account.

HF documents egress on 80, 443 and 8080. The checker accepts SMTP on these ports with exactly one TLS mode; provider support and real delivery still need testing. Ordinary SMTP on 465/587 is not a reliable HF configuration. Linux/Docker may retain encrypted SMTP on its permitted ports. A managed PostgreSQL endpoint on 5432 likewise needs demonstrated permitted connectivity; choose a supported endpoint/port or provider-approved network arrangement. An HTTPS database administration URL is not a PostgreSQL protocol endpoint. See HF networking.

Admission is shared across organizations

The existing organization limits remain. Add explicit account caps to prevent one person multiplying their allowance by creating or joining organizations. These conservative example values are a starting policy, not a capacity claim:

TEXT | Copy exact lines from the packaged Markdown

```
PF_SAAS_ACCOUNT_MAX_CONCURRENT_JOBS=1
PF_SAAS_ACCOUNT_MAX_QUEUED_JOBS=20
PF_SAAS_ACCOUNT_MONTHLY_COMPUTE_SECONDS=36000
PF_SAAS_ACCOUNT_MAX_STORAGE_BYTES=5368709120
PF_SAAS_ACCOUNT_MAX_OWNED_ORGANIZATIONS=1
PF_SAAS_ACCOUNT_MONTHLY_AI_MESSAGES=1000
PF_SAAS_ACCOUNT_AI_MAX_CONCURRENT_REQUESTS=1
PF_SAAS_ACCOUNT_AI_REQUESTS_PER_MINUTE=30
PF_SAAS_ACCOUNT_AI_TOKENS_PER_MINUTE=250000
PF_SAAS_ACCOUNT_AI_MONTHLY_COST_MICRO_USD=0
PF_SAAS_ACCOUNT_ASSISTANT_MAX_QUEUED_OPERATIONS=4
PF_SAAS_ACCOUNT_ASSISTANT_MAX_CONCURRENT_OPERATIONS=1
```

The example leaves monetary AI accounting disabled. To enable a cost cap, set both `PF_SAAS_AI_INPUT_COST_PER_MILLION_MICRO_USD` and `PF_SAAS_AI_OUTPUT_COST_PER_MILLION_MICRO_USD` to reviewed positive conservative tariffs for the allowed providers/models, then set a positive account cost cap. Zero tariffs with a positive cost cap are rejected at startup.

New account caps default to zero (disabled) to preserve existing private deployments. Public readiness requires an explicit bounded policy. Configure the same policy on every API, mail and worker process. A billable provider may charge independently of PF's counters; set provider budgets as well.

The migration attributes historical attempts from their jobs and profiles only where retained operations establish a unique user. Unknown ownership is not guessed. Enabling account caps fails closed on unresolved charged usage; retain the source records and resolve attribution through an audited maintenance procedure before public admission. Creating a new organization does not erase the account's existing reservations or retained bytes.

Schema and lifecycle upgrade

The SaaS schema is **10**. Drain all API writers, assistant continuations and workers; confirm corresponding executions terminal, then create and verify a consistent backup. Native HF applies the additive schema migration while its single controller starts. Distributed Linux/Docker runs the explicit existing `python -m pf_saas init` migration command with the same service configuration before any new application replica serves. Managed HF uses guarded maintenance and verifies authority and charged totals before activation.

Do not mix old and new application processes during migration. A rollback is a coordinated restore of the pre-upgrade database, objects and code, with all writers stopped. Restoring only an older code image is not a schema rollback. The additive migration preserves retained files, source accounting records, users, organization/project grants and established execution authority.

The HF storage policy remains `conservative_lifetime` by default. Platform administration can switch to `safe_lifecycle` with a generation check and an audit reason. Future output reservations settle from verified publication; physical deletion and refund require recorded ownership, terminal execution, recovery safety and idempotent receipts. Legacy charges require a reviewed reconciliation plan and its unchanged digest. Deleting an arbitrary object or `pf-code.zip` is not a valid quota reconciliation method.

Keep the currently approved ceiling. For the deployment already raised to 32 GiB this is `34359738368` in both the encrypted meter and `PF_HF_MAX_STORAGE_BYTES`. Do not set it back to 16 GiB based on an expectation of refunds. First measure the reconciled retained usage and required reserve. Safe accounting prevents avoidable lifetime charges; it cannot guarantee that real retained data and recovery history fit within 16 GiB.

Managed HF additionally coordinates reclamation with external database backup retention. Its audited recovery floor must establish that no older PITR point or native recovery root can still reference deleted data. See managed accounting (HUGGING_FACE_MANAGED_ACCOUNTING.md). An expired timer alone does not prove backup safety.

Execution recovery runs at startup and periodically. It acquires the owner's fence and proves that the corresponding execution has stopped before marking an operation interrupted and releasing temporary reservations. Retained files remain. Docker child executions receive durable launch/terminal journals; ambiguous legacy executions remain held until evidence resolves them. Recovery is a general lifecycle mechanism, not a hard-coded repair for one operation ID.

Qualify and expand gradually

1. Build the complete native/managed contexts from this release and verify the exact private code ZIP and image pins. Check that new mail, recovery, admission and accounting modules are included in the generated contexts.
2. Run an isolated staging migration with a protected copy of representative state. Test registration, verification, reset, invitation, private workflows, permissions, concurrent users, cancellation, recovery and accounting.
3. Run the offline public readiness checker (PUBLIC_LAUNCH_READINESS.md) against configuration and retained ingress/email/isolation/restore evidence.
4. Run the mixed-workload qualification campaign (../deploy/saas/distributed/CAPACITY_QUALIFICATION.md) on an explicitly marked isolated target. Include real active workflows, provider capacity, BYOK/BYOS, failures, recovery, backlog drainage and a soak. Collect database, API, object-store, worker and provider measurements.
5. Approve the observed envelope, configure finite admission and expand public access in stages. At saturation the service must bound/deny work with retry guidance while preserving existing reservations and private results.

The load generator is sharded; the intended user count is separate from executing-job concurrency. For example, 200,000 people browsing or editing is a different compute requirement from 200,000 simultaneous scientific

jobs. Both must be represented explicitly in the campaign contract and provisioned within provider limits. The release's local tests and previous isolated HF qualification records cannot substitute for that campaign.

Public account email delivery

This release supports verification, password reset and organization invitation email through SMTP or the Resend HTTPS API. The default development outbox remains available only in development. Production local authentication requires a real email transport. An accepted registration response does not prove that an email has reached an inbox, or that a new account was created: existing and unknown addresses deliberately receive the same public response.

Hugging Face setup

Use a verified sending domain in your email provider account. Configure its DNS authentication records and check the provider's sending limits and billing. HF Spaces document outbound ports 80, 443 and 8080. Both encrypted SMTP on a provider-supported permitted port and the optional Resend HTTPS transport are supported. Ordinary SMTP submission ports 465 and 587 are unsuitable for that documented Space network. Existing SMTP2GO users do not need to migrate providers. Linux/Docker deployments with suitable egress can retain their encrypted SMTP.

Set these values in the **controller** Space, preserving the existing session key, database, HF state encryption key, storage bindings and code pins:

Setting	Value
`PF_SAAS_AUTH_MODE`	`local` for existing local accounts
`PF_SAAS_PUBLIC_URL`	Exact external HTTPS origin, without a path
`PF_SAAS_REGISTRATION_ENABLED`	`false` during deployment acceptance; `true` only for the intended public opening

Choose one transport. To retain SMTP2GO:

Setting	Value
`PF_SAAS_MAIL_MODE`	`smtp`
`PF_SAAS_SMTP_HOST`	`mail.smtp2go.com`
`PF_SAAS_SMTP_PORT`	`443`
`PF_SAAS_SMTP_SSL`	`true` (implicit TLS)
`PF_SAAS_SMTP_STARTTLS`	`false`
`PF_SAAS_SMTP_FROM`	Existing verified sender
`PF_SAAS_SMTP_USERNAME`, `PF_SAAS_SMTP_PASSWORD`	Existing **Space Secrets** for the SMTP user

Exactly one of implicit TLS and STARTTLS must be enabled. Do not assume a provider supports SMTP on an allowed port: retain its documented endpoint and confirm real delivery from the deployment. Port 443 does not turn SMTP into an HTTP API.

For the optional Resend HTTPS transport, instead use:

Setting	Value
`PF_SAAS_MAIL_MODE`	`resend`
`PF_SAAS_MAIL_FROM`	Verified sender, such as `PF Simulation <accounts@your-domain.example>`
`PF_SAAS_MAIL_API_KEY`	**Space Secret**, restricted to sending email where supported

The API key never belongs in browser JavaScript, a public repository or a Space Variable. The endpoint is fixed at `https://api.resend.com/emails`; configuration cannot redirect credentials to another host. TLS verification is required, redirects and environment-provided HTTP proxies are disabled, transport timeouts are bounded and provider error bodies are not logged or returned to users.

The sender and link origin must match the intended deployment. Reusing production configuration in a staging copy could send real email: isolated acceptance uses mock transports and invented addresses instead. This release's offline tests do not qualify your provider account, domain configuration or real inbox delivery.

Durable delivery and recovery

The schema-v10 migration adds `saas_mail_deliveries`. Existing user identities, password hashes and account tokens remain intact. Registration, reset and invite transactions store their token and encrypted email intent together. A rollback cannot leave a committed token with an uncommitted email record. A worker claims the record in a separate short transaction, commits the claim, and then contacts the provider without holding a SQL lock. Multiple API replicas or dedicated mail workers use row locks and fenced leases against the same PostgreSQL database.

The queued recipient, sender, subject and body are encrypted with a domain-separated key derived from the existing `PF_SAAS_SECRET_KEY`. Token text and provider keys are absent from queue inspection output. Losing or changing the session key before the queue drains makes pending ciphertext unreadable; the worker records a terminal `payload_unavailable` failure. Keep the same key across replicas and upgrades.

Every retry reuses the same immutable message identifier and frozen payload. Resend's idempotency retention is 24 hours. Automatic retry stops within 23 hours of enqueue, or at the token's earlier expiry. Exponential backoff, jitter and attempt limits bound retries. Expired, consumed, replaced and revoked links are not newly submitted when the worker checks them. A token revoked while a network request is already in flight can still appear in an email, but cannot authorize the revoked action.

For SMTP, the stable `Message-ID` is useful for correlation, but the SMTP protocol does not guarantee deduplication. A timeout after acceptance can cause a duplicate email on retry. Its single-use account link still cannot be consumed twice.

`accepted` means that the configured provider accepted the request. PF does not claim final delivery, read receipts or bounce processing. Check the provider's dashboard for bounces and delivery failures. Terminal queue records erase the encrypted message immediately and retain only sanitized operational receipts for the configured retention period. No automatic attempt occurs after a terminal failure; a user may request a new eligible link through the existing UI. Do not manually replay an old message outside its provider idempotency window.

Drain the queue before changing provider, sender identity, provider account or the session key. A changed sender setting does not mutate an already queued request. Provider modes are not silently switched for pending messages.

Worker placement and inspection

Native HF uses its existing single controller's application-lifespan mail worker. Managed HF and Linux/Docker can run a worker in each API process or use dedicated mail-worker containers. For dedicated workers, disable background delivery on API replicas and run the worker with the same database, session key and mail settings:

SH | Copy exact lines from the packaged Markdown

```
# API replicas only:
PF_SAAS_MAIL_DELIVERY_WORKER_ENABLED=false

# Dedicated worker process, after the explicit schema migration:
python -m pf_saas.mail_delivery run

# Read-only operator inspection:
python -m pf_saas.mail_delivery status
```

The explicit `run` command enables delivery in that process even when its inherited API setting disables it. It does not start an HTTP service or simulation worker. This standalone command targets a configured shared

PostgreSQL deployment; use the integrated application worker in native HF, whose database credentials and durable ownership are initialized by the native controller.

Signed-in platform administrators can inspect `GET /api/admin/mail/status` through the existing origin/session boundary. Organization owners and administrators do not have this permission. The response contains aggregate statuses, sanitized error categories and oldest pending timestamp; it excludes addresses, bodies, keys and account-token values. A growing queue or terminal failures requires operator attention. Monitor provider throttling and oldest-pending age as part of the public rollout and mixed-workload capacity qualification.

Setting	Default	Bounds / purpose
`PF_SAAS_MAIL_TIMEOUT_SECONDS`	15	1-60 seconds per transport operation
`PF_SAAS_MAIL_DELIVERY_POLL_SECONDS`	2	0.1-60 seconds while idle
`PF_SAAS_MAIL_DELIVERY_MAX_ATTEMPTS`	8	1-32 attempts within expiry
`PF_SAAS_MAIL_DELIVERY_RETENTION_DAYS`	7	1-90 days of sanitized terminal receipts
`PF_SAAS_MAIL_DELIVERY_WORKER_ENABLED`	`true`	Enable background worker in API processes

When using SMTP, preserve the existing `PF_SAAS_SMTP_*` fields; production requires exactly one of STARTTLS or implicit TLS. SMTP and HTTPS mail share the same durable queue lifecycle. Public-readiness evidence includes the effective sender and SMTP endpoint/TLS settings in its configuration fingerprint; changing those settings requires new evidence. No transport is qualified merely by passing this offline configuration review.

Acceptance before opening registration

Deploy the release and explicit migration with registration closed. Verify the admin-only status route is inaccessible to anonymous users and ordinary accounts. Use an operator-controlled real inbox to exercise registration verification, a single-use password reset, an invitation and invite revocation. Check both PF's acceptance record and actual provider/inbox results. Verify restart recovery and rate-limit behavior in an isolated environment, then enable public registration under the release's broader public-readiness and capacity gates. Do not send synthetic high-volume email to real recipients for a capacity test.

Provider references, checked for this release on 2026-10-09:

- HF Spaces network access
- SMTP2GO SMTP relay and encrypted submission ports
- Resend send-email API
- Resend idempotency semantics

Public registration, ingress identity and launch evidence

This release adds explicit ingress contracts and an offline prerequisite review. It does not change an existing Space's visibility, register users, send test mail, or certify a concurrent-user capacity. Keep the existing session, encryption and storage keys when upgrading. Test the complete public path on a separate staging deployment before opening the production application.

Authentication and administrative authority

Local registration creates an ordinary, unverified account. Email verification is required before login. Owning an organization does not grant platform authority. The accounting page, its JavaScript and its administrative API require a current platform administrator session. CSRF and origin checks still apply to mutations. Never bootstrap every new user as an administrator to make registration work.

Configure production mode, the exact HTTPS public origin, a preserved strong session secret, a production PostgreSQL deployment and working account email. For HF retain a verified encrypted SMTP endpoint on an allowed outbound port, such as SMTP2GO port 443 with `PF_SAAS_SMTP_SSL=true` and `PF_SAAS_SMTP_STARTTLS=false`, or optionally use `PF_SAAS_MAIL_MODE=resend` with its server-side API key and verified sender. The HF checker accepts SMTP on ports 80, 443 or 8080 only with exactly one TLS mode; an allowed port alone does not prove that the provider offers SMTP there. Linux/Docker can also use encrypted SMTP on its permitted ports. Consult the account mail deployment guide for durable delivery, retries and operator failure inspection. A successful registration HTTP response alone does not establish delivery of its verification message. Test verification, password reset and invitation separately.

Set positive account-wide concurrent-job, queued-job, monthly-compute, storage and owned-organization limits as well as organization and deployment limits. These apply independently: creating another organization must not multiply an account's allowance. Authenticated rate admission is also account-wide.

Configure the actual proxy contract

`PF_SAAS_TRUSTED_PROXY_CIDRS` is empty by default. Only an actual socket peer in these explicitly configured networks can supply a forwarded client address. Do not populate it with guessed HF ranges, all private address space, or a default route. If the provider cannot establish a trustworthy forwarding contract, the public ingress remains unqualified. Resolve that with the ingress operator or use an ingress whose identity and forwarding behavior you control.

Keep Uvicorn proxy rewriting disabled (`proxy_headers=False` or `--no-proxy-headers`). Otherwise the boundary may receive an already rewritten peer and cannot verify the transport source. PF never trusts forwarded host or protocol headers to establish the application origin.

Setting	Contract
`PF_SAAS_PROXY_HEADER_MODE=none`	Use the socket client identity; forwarded headers cannot select it. Use only when this is actually the end-client connection, or after PF's authenticated gateway boundary. Leave proxy CIDRs empty.
`PF_SAAS_PROXY_HEADER_MODE=append`	Compatibility default. One trusted immediate proxy appends its observed client IP to X-Forwarded-For. PF uses the last address. The proxy must disable client-port suffixes.
`PF_SAAS_PROXY_HEADER_MODE=single`	The trusted immediate proxy replaces X-Forwarded-For with exactly one observed client IP. Multiple values fail.
`PF_SAAS_PROXY_HEADER_MODE=chain`	All trusted proxy hops are listed explicitly. PF walks from the right and selects the first untrusted address. A chain containing only trusted proxies fails.

Setting	Contract
`PF_SAAS_PROXY_REQUIRE_HEADER=true`	Require the chosen forwarding header from trusted peers. Missing headers fail except GET health/readiness probes. Compatibility default is false; the public review requires true when forwarding is enabled.

Restrict backend network ingress to the intended proxy. The proxy must remove or correctly append incoming forwarding headers according to the chosen contract. Never accept a user-supplied client-IP header as a workaround for shared HF proxy addresses. A single shared IP would cause unrelated users to share the public authentication rate window; trusting arbitrary XFF would instead allow its bypass.

The optional HF gateway for an external Linux/Docker backend has equivalent `PF_HF_TRUSTED_PROXY_CIDRS`, `PF_HF_PROXY_HEADER_MODE` and `PF_HF_PROXY_REQUIRE_HEADER` settings. Its compatibility default is `chain`. Configure and test these in the gateway process, not only in the backend. The gateway strips incoming forwarding and PF gateway headers, then sends an authenticated client identity to the backend. Its independent gateway secret must stay server-side. The backend's `HFGatewayBoundary` verifies that secret and the canonical origin before using the supplied identity. Keep ordinary backend proxy CIDRs empty and mode `none` for this path. A backend check cannot prove that the public HF gateway has been configured correctly: retain both configuration records with the ingress test evidence.

When the backend enables `PF_SAAS_HF_GATEWAY_SECRET`, the offline review also requires `--gateway-policy gateway-policy.json`. Export the following non-secret values from the actual gateway configuration (these addresses are examples):

JSON | Copy exact lines from the packaged Markdown

```
{
  "public_url": "https://pf.example.org",
  "upstream_url": "https://backend.example.org",
  "trusted_proxy_cidrs": "REPLACE_WITH_VERIFIED_PROXY_NETWORKS",
  "proxy_header_mode": "single",
  "proxy_require_header": true
}
```

Never include the gateway secret in this file. The policy must match the public origin and binds the fixed upstream and forwarding settings into the evidence fingerprint. This remains an operator-exported policy, not a remote settings fetch.

Use two independent external clients to confirm distinct client identities. Send forged XFF prefixes and duplicate/malformed forwarding headers and verify they cannot choose a different identity or evade authentication admission. Test direct origin access without passing the intended ingress. Retain sanitized results and the precise ingress configuration; do not publish client IPs or credentials in public logs or expose a public diagnostic endpoint to reveal them.

Shared limits and overload behavior

Authentication limits use database time and one locked counter per action and hashed identity. Independent API processes share the same counter. Distributed and managed HF authenticated-account rate windows are also database-backed. Native HF remains one fenced controller; its authenticated read-rate window is bounded in process and resets on restart, which avoids per-read WAL writes. That controller is not an independently replicated API tier.

Per-process HTTP and upload admission remains bounded. Saturation returns a retryable response; database pool exhaustion and operational connection failures return sanitized 503 responses with `Retry-After`. Authentication/account rate rejection returns 429. Client retries should honor these bounds and use backoff; they must not create duplicate mutations. Deploy upstream connection/body/rate controls as well: application counters cannot absorb an unbounded network flood.

Offline review and evidence

Run from the release root with the exact deployment's `PF_SAAS_*` environment available locally. The tool reads configuration without creating a session secret or data directory. It neither connects to providers nor changes remote resources. It does not print secrets.

TEXT | Copy exact lines from the packaged Markdown

```
python release_tools/check_public_readiness.py --target-users 200000
```

The first report is expected to fail missing live evidence. It prints the `configuration_sha256` that binds subsequent evidence to non-secret policy settings, security source bytes, database target and optional gateway policy. The mail fingerprint binds the effective sender and SMTP host, port and TLS settings. Changing those settings invalidates prior evidence and requires fresh delivery checks. Credentials are excluded. Record a UTC Unix timestamp and retained test-record references. The schema below deliberately contains false values; replace each only after that specific check succeeds against the target staging deployment. Never turn these into unconditional true defaults in deployment automation.

JSON | Copy exact lines from the packaged Markdown

```
{
  "schema": 1,
  "configuration_sha256": "COPY_THE_REVIEWED_CONFIGURATION_SHA256",
  "checked_at_unix": 0,
  "ingress": {
    "record": "retained-ingress-test-record",
    "distinct_client_addresses": false,
    "forged_forwarding_cannot_select_identity": false,
    "direct_origin_cannot_bypass_ingress": false,
    "raw_asgi_socket_peer_preserved": false
  },
  "authorization": {
    "record": "retained-authorization-test-record",
    "anonymous_admin_denied": false,
    "tenant_admin_denied": false,
    "cross_tenant_private_objects_denied": false,
    "registration_creates_non_admin": false
  },
  "mail": {
    "record": "retained-mail-test-record",
    "verification_delivered_and_consumed": false,
    "password_reset_delivered_and_consumed": false,
    "invitation_delivered_and_consumed": false,
    "used_token_rejected": false
  },
  "operations": {
    "record": "retained-recovery-and-operations-record",
    "backup_restore_verified": false,
    "stopped_execution_recovery_verified": false,
    "overload_is_bounded": false,
    "secret_rotation_and_rollback_documented": false
  }
}
```

For Cognito replace the `mail` group with an `identity_provider` group containing a `record` and these three booleans: `verified_subject_required`, `existing_account_binding_explicit`, `logout_and_session_expiry_verified`. Identity-provider signup/recovery email must also work; changing auth modes is not an implicit migration of existing accounts.

TEXT | Copy exact lines from the packaged Markdown

```
python release_tools/check_public_readiness.py --target-users 200000 --evidence public-launch-evidence.json
```

Exit 0 means the listed configuration prerequisites and operator attestations are complete; exit 1 means some are missing; exit 2 means the input is invalid. Evidence older than 30 days, future-dated evidence and changed

policy fingerprints fail. The report explicitly identifies these records as operator attestations, not independent verification. Preserve the exact release checksum and both gateway/backend settings with the records. Re-run affected checks after code, ingress, authentication, provider, data-residency or deployment topology changes. Normal production startup still performs its full settings validation.

`public_configuration_ready` is a prerequisite check. `public_launch_prerequisites_attested` adds the operator records. `capacity_qualified` and `deployment_authorized` are always false here. Run the separate mixed-workload qualification at the intended active concurrency with actual provisioned databases, workers, providers, storage and ingress. Native single-controller HF cannot be certified for hundreds of thousands simply by raising its environment limits. Distributed Docker and managed HF need measured fleet capacity; managed HF uses external PostgreSQL. The software release does not itself provision that fleet or open production access.

HF application and source visibility

HF documents **protected** Spaces as public applications with private repository source, available on the applicable paid plans. Ordinary public visibility also publishes the repository. Keep the scientific code-only Space and data buckets private. Opening an application does not grant platform administration inside PF.

The native and managed full-release updaters accept a controller that the authenticated Hub metadata explicitly identifies as `visibility="protected"`, or a private controller. They do not change visibility. `private=False` without explicit protected metadata is refused: it does not distinguish protected from public. Conflicting or unknown visibility is also refused. If an older API/client omits protected metadata, stop and resolve metadata support; never remove the source-privacy check or relabel unknown data as protected. Code-only archive uploads continue to require a private repository.

Reference documentation, checked 2026-10-09:

- HF Space visibility
- HF repository visibility API

Local regression tests cover transport spoofing, separate client counters across independent service instances, malformed forwarding rejection, protected-source metadata guards, and anonymous/ordinary-user denial on accounting administration. They exercise isolated fixtures, not public HF ingress or live capacity.

Account-wide public admission

The account layer bounds one user's own work across every organization they belong to. Organization plans, project permissions, private workflow ownership, deployment budgets, provider budgets and execution confinement continue to apply independently. Creating or joining another organization does not grant a second account allowance. A storage destination owner is distinct from the creator charged for retained scientific output.

All settings below use the `PF_SAAS_` prefix. Defaults are `0` (disabled) to preserve existing private deployments. Configure positive account limits before public rollout. The limits supplement, rather than replace, organization and deployment limits. They are process configuration: apply the same values to all API and worker replicas and restart them consistently. These account settings are separate from the HF lifecycle accounting policy switch.

Setting suffix	Account-wide scope
`ACCOUNT_MAX_OWNED_ORGANIZATIONS`	Owned organizations, including archived organizations; creation and promotion both enforce it.
`ACCOUNT_MAX_QUEUED_JOBS`	Queued normal simulation jobs across organizations.
`ACCOUNT_MAX_CONCURRENT_JOBS`	Running normal and assistant executions, including ambiguous provider reservations.
`ACCOUNT_MONTHLY_COMPUTE_SECONDS`	Settled and reserved compute in the UTC billing month.
`ACCOUNT_MAX_STORAGE_BYTES`	Retained assets, attempt outputs, assistant profiles, temporary output reservations and in-flight storage transfers.
`ACCOUNT_MONTHLY_AI_MESSAGES`	Admitted assistant chat operations in the UTC month, including provider failures.
`ACCOUNT_AI_MAX_CONCURRENT_REQUESTS`	Outstanding model HTTP requests, across providers and organizations.
`ACCOUNT_AI_REQUESTS_PER_MINUTE`	Actual/admitted model requests in the shared database minute.
`ACCOUNT_AI_TOKENS_PER_MINUTE`	Conservative token admission with provider-reported settlement when available.
`ACCOUNT_AI_MONTHLY_COST_MICRO_USD`	Estimated model cost using configured tariffs; this is an admission allowance, not an invoice guarantee.
`ACCOUNT_ASSISTANT_MAX_QUEUED_OPERATIONS`	Queued durable investigations across organizations.
`ACCOUNT_ASSISTANT_MAX_CONCURRENT_OPERATIONS`	Admitted background investigation workers across organizations.

Per-account allowances are not divided by the number of users. Deployment-wide ceilings independently bound total infrastructure use. Select values from the workload and provisioned capacity; increasing these numbers is not evidence of throughput. A user can browse and edit while another user's execution runs, subject to the shared infrastructure and request admission limits.

Accounting and fairness

Schema v10 adds attribution to existing source records and installs version 2 transactional usage projections. Source charges remain authoritative; the account dimension is not a second debit. Submission reserves once, idempotent replay returns the original job, and completion exchanges a reservation for settled usage. Cancellation before dispatch releases the reservation. Deletion releases retained storage only through the existing confirmed deletion path. Uncertain provider execution retains its concurrent slot until verified stop reconciliation. A running controller record and its matching provider reservation consume one slot, not two.

Admission locks an organization, then the account, then deployment capacity. Different organizations cannot race the same account allowance. PostgreSQL uses transaction-scoped advisory locks and fixed-size indexed

counter reads; SQLite uses its existing immediate writer transaction for the single-process development profile. The fair scheduler skips account-saturated candidates so another eligible user in the same organization can progress. Backpressure and per-account fairness do not guarantee a specific waiting-time SLA.

Provider counters add an account scope to the existing durable reservation. An unsubmitted request refunds all scopes; an ambiguous outcome retains its slot and conservative charge. Reconciliation releases capacity only with the existing independent stopped-work evidence. Queue interruption checkpoints may temporarily exceed a configured queue cap by the already admitted running population; new submissions still obey the cap.

Migrate existing usage before enabling limits

1. Back up the database and private object bindings. Stop all API and worker writers using the deployment's documented drain/stop procedure.
2. Run the explicit schema migration. Attempt attribution comes from its job; assistant attribution requires a unique retained actor; transfer attribution comes from its immutable scope or source. Durable investigation actor bindings backfill queued and background account scopes. No organization owner's identity is guessed. Existing charges and files remain intact.
3. Inspect unresolved accounting identities using the same private service environment as the migration:

BASH | Copy exact lines from the packaged Markdown

```
python -m pf_saas.account_admission_control report
```

4. Resolve each nonzero unattributed charge against retained job, assistant, provider and operator audit evidence. Old provider reservations may lack an account identifier entirely. If the original account cannot be independently identified, leave the charge intact and keep the affected account caps disabled. Do not assign the charge to an arbitrary owner or reset counters to make the readiness check pass.
5. For a confirmed attribution, create a private JSON evidence record, for example:

JSON | Copy exact lines from the packaged Markdown

```
{
  "source_kind": "ai_reservation",
  "source_id": "THE_EXISTING_RESERVATION_ID",
  "org_id": "THE_ORIGINAL_ORGANIZATION_ID",
  "user_id": "THE_CONFIRMED_ORIGINAL_ACCOUNT_ID",
  "confirmed_attribution": true,
  "evidence_reference": "operator-audit/reference-to-retained-request-evidence"
}
```

Obtain a read-only plan:

BASH | Copy exact lines from the packaged Markdown

```
python -m pf_saas.account_admission_control ai_reservation THE_EXISTING_RESERVATION_ID --user-id
THE_CONFIRMED_ORIGINAL_ACCOUNT_ID --evidence-file /absolute/private/evidence.json --operator-id
local-operator
```

Review its source, destination account, exact retained charges and evidence digest. While writers remain stopped, apply the same command with `--apply --writers-stopped --expected-plan-sha256 THE_REVIEWED_PLAN_HASH`. A changed source invalidates the plan. The transaction preserves quantities, access rights and execution state and appends an `account.usage_attributed` audit event. It cannot reassign an already attributed charge. Re-running after a completed attribution requires a fresh plan and makes no extra debit.

The same interface accepts `attempt`, `assistant_profile`, and `storage_mutation`; the latter also requires `--resource-kind asset` or `--resource-kind job`. Background investigations must be repaired through their authoritative retained actor binding and migration, not this command.

6. Configure consistent limits and run readiness checks before starting public traffic. Startup refuses enabled caps that would omit nonzero legacy usage, unowned active provider requests, or unattributed current-month model cost. Confirm public registration verification and administrator-only platform controls separately.

The attribution command never claims that a remote execution stopped and never refunds usage. Execution recovery and HF lifecycle reconciliation retain their own evidence, ownership and audit requirements.

Verification

`tests_saas/test_account_admission.py` runs the same account admission cases on SQLite and on a disposable PostgreSQL schema when `PF_TEST_POSTGRESQL_URL` is set. PostgreSQL cases otherwise skip explicitly. The tests cover concurrent cross-organization admission, source settlement, temporary transfer retention, ambiguous provider slots, queue fairness, organization ownership, and migration idempotency.
`tests/test_account_ai_admission.py` covers the shared provider/background/queue account scopes without contacting any paid provider. These tests establish functional boundaries; they do not qualify hundreds of thousands of active users. Use the release's mixed-workload qualification procedure on provisioned isolated infrastructure for that claim.

Assistant execution recovery

The native HF, managed HF, and distributed Docker profiles run recurring recovery of abandoned assistant operations. This is a lifecycle service, not a script tied to one operation ID. A queued simulation can use an execution slot again only after recovery has proved that the assistant execution occupying it has stopped. An expired lease, timeout, cancellation request, or empty provider listing is insufficient proof.

Recovery marks an abandoned running assistant operation `interrupted`, clears its temporary compute and storage reservations, records the retained private byte count, and writes an audit event in the same database transaction. Unknown elapsed compute is charged conservatively at the reserved allowance. Recovery does not delete retained private files. A lost response can be retried: the current database state prevents a second settlement. Diagnostics distinguish a confirmed accounting commit from a later owner-release failure and omit exception text, SQL, credentials, and private content.

HF profiles

Native HF retains its controller fence in addition to the private profile lease and deployment-reservation lock. Managed HF uses shared PostgreSQL profile leases, reservation-row locking, and the durable HF child-job journal; it does not pretend to own the native single-controller fence. Both inspect exact provider job identities and fingerprints. Confirmed terminal evidence is retained so recovery does not depend on the provider keeping completed jobs forever.

Managed startup runs a bounded sweep before starting worker dispatch. The application lifespan continues scheduled fleet maintenance. The accounting administrator page exposes the same evidence-based recovery service; ordinary users and organization administrators cannot invoke platform-administration APIs. Retrying one operation applies the same checks as background recovery and has no force-refund switch.

Distributed Docker

Every new assistant execution reservation enrolls in the durable recovery protocol before submission. Each Docker child is journalled before launch and carries labels binding it to its exact reservation and child ID. The journal stores a hash of the Docker daemon identity. Another host cannot use its own absent-container result as proof for that execution.

The shared private directory must provide coherent POSIX `flock`, atomic rename, and `fsync` across every replica. A per-profile kernel lock complements the SQL publication lease. A still-running process retains this lock even when its SQL lease expires; recovery cannot reconcile its files concurrently. Do not delete `.owner-fences` files or replace their storage during execution. An object-storage mount without these filesystem guarantees is unsupported.

Once recovery holds the profile lock and SQL lease, it can inspect and remove an abandoned child's exact authenticated Docker object. The immutable container ID is committed before forced removal. A retry can therefore distinguish a known object's permanent absence from an unbound name that might still be created by a delayed Docker request. Charges stay reserved if the daemon is unavailable, identities disagree, or submission remains uncertain. Run a recovery-capable API replica on each Docker daemon with unfinished children; draining a host must retain that capability until its children settle.

Older ambiguous Docker reservations have no durable child journal. The upgrade preserves them and reports `submission_unresolved`; it does not invent execution evidence or silently refund them. An operator must establish stopped execution and use the deployment's audited reconciliation procedure. Existing finished/unsubmitted reservations retain their durable terminal/no-submission meaning.

Bounds and rollout

The shared settings `PF_SAAS_ASSISTANT_RECOVERY_BATCH_SIZE` (default 8, maximum 32), `PF_SAAS_ASSISTANT_RECOVERY_MAX_SECONDS` (default 15, maximum 60), and `PF_SAAS_ASSISTANT_RECOVERY_INTERVAL_SECONDS` (default 15) bound distributed and managed maintenance. A single in-flight provider call may extend a sweep past its selection deadline; every provider call has a separate timeout. Currently valid private leases are excluded in SQL as a candidate-selection optimization. Every selected operation still receives the full ownership and termination checks.

Apply the additive SaaS schema migration before starting distributed replicas. Stop or drain old API processes before starting processes that use the new filesystem lock protocol. Keep the same database, private storage, encryption keys, and Docker daemon identities during upgrade. Local protocol tests cover crash and retry behavior; they do not qualify production provider behavior, shared filesystem guarantees, or large-scale capacity. Include mixed-workload and failure-recovery testing on the intended infrastructure before public rollout.

Shared HF accounting and recovery

The `hf_managed` profile uses an external, writable PostgreSQL database for authoritative application state and encrypted Hugging Face objects for files. It is a separate deployment path from the native single-controller HF profile. These accounting changes do not convert a native Space into a replicated deployment. Use the managed migration and deployment guide first.

Defaults and online policy changes

The default remains `conservative_lifetime`. New writes collect immutable operation identities and durable write-closure receipts in either mode. In conservative mode, deletion or completion does not automatically refund the deployment storage allowance.

A platform administrator can select `safe_lifecycle` at `/admin/hf-accounting`. The policy is held in shared PostgreSQL and takes effect across replicas without a restart. Each change requires the observed policy generation and an audit reason. An organization owner or administrator does not receive platform administration authority. The HTML, script and every API use the existing authenticated platform role, same-origin checks and CSRF protection.

Safe lifecycle accounting:

- settles a future attempt's maximum output reservation only after its provider execution is durably terminal and its exact immutable output inventory is verified;
- releases temporary transfer storage only after the publication or discard authority, writer closure and exact physical absence have been established;
- releases artifact and private-state storage only after all live references have ended, the publisher is closed, physical deletion is verified and the database recovery protection described below allows deletion;
- updates the operation receipt and aggregate charge in one database transaction;
- retains a permanent SQL tombstone so a retried admission cannot recreate a settled or released immutable payload.

Switching back stops new credits; completed credits remain valid. Policy changes never recalculate usage from a bucket listing or reset the quota. A stale policy generation cannot authorize a new transition. Receipts with uncertain remote writes remain charged until independent writer-closure evidence is available.

The accounting ceiling measures admitted HF object bytes. The size, replication, indexes and backup retention of the external PostgreSQL database require their own provider capacity and monitoring. SQL audit/tombstone storage is not billed against the HF object meter; it grows with operation history and must be included in database sizing.

Migration and historical charges

Pause and drain the whole managed fleet before applying the release's schema upgrade. Serving startup validates existing schema and accounting state and does not create an empty replacement ledger. The explicit migration calls `initialize_managed_accounting_schema(engine)` after the imported meter exists.

The application database role must be able to inspect its PostgreSQL cluster identity and timeline. The database administrator grants only the needed functions, not superuser access, for example (replace `pf_app` with the actual application role):

SQL | Copy exact lines from the packaged Markdown

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_system(),
  pg_catalog.pg_control_checkpoint() TO pf_app;
```

Serving startup verifies these reads before admitting users. A provider that does not expose these PostgreSQL identity functions is not qualified for this managed lifecycle implementation. Changing hosts, restoring a snapshot, or promoting a different timeline does not authorize interpreting an old publisher XID in a new timeline: affected private-state receipts stay held for explicit recovery reconciliation.

Before enabling managed artifact or private-state reclamation, generate an external recovery epoch and set the **same value on every replica**:

BASH | Copy exact lines from the packaged Markdown

```
python -c "import uuid; print(uuid.uuid4().hex)"
# Set PF_HF_MANAGED_RECOVERY_EPOCH to that 32-character lowercase hex value.
```

This is an operator-controlled deployment setting, not a secret and not a value restored from the database. Without it the application can retain its normal conservative operation, but safe artifact/private-state credit remains blocked. Setting it before new private-state writes allows those receipts to be eligible later; old receipts with a missing or different epoch are never automatically rebound.

For **every database restore, fork or replacement**, stop and fence the entire old fleet, choose the supported matching database/object recovery point, generate a new epoch, and configure it on every replacement replica before any writer starts. Rotation invalidates the previous recovery-floor attestation and old private-state/deletion proofs. Review the provider's new recovery inventory and record a new floor before allowing reclamation. Existing charges stay held until their new authority is explicitly reconciled; rotation never refunds or resets usage. Do not copy an old epoch into a restored deployment.

A byte-for-byte physical clone using the same database identity, timeline and deployment settings cannot be distinguished from the original by application software alone. The external epoch rotation is therefore a required recovery runbook step, in addition to the automatic cluster/timeline checks.

Both native meter schema 1 and the Qualified lifecycle patch's schema 2 are accepted. Import preserves the exact total, ceiling, WAL reserve, every active reservation, lifecycle transition and the complete native meter provenance. Schema 2 policy and generation are preserved. Native encrypted retirement receipts and closed recovery-generation fences continue preventing replay after cutover. Preserve the original encrypted object namespace and state key.

An existing managed database is upgraded additively. Every pre-upgrade managed reservation is classified as legacy; no existing byte is automatically removed. Previously enabled recovery protection stays enabled when importing a native meter whose lifecycle mode has since been turned off. The same native snapshot may be imported again after an interrupted migration, but a different snapshot cannot replace an existing import. Audit and residual usage that cannot be attributed exactly remain charged.

The administrator's reconciliation preview matches historical **operation identities**, not equal byte amounts. The current verifier can reconcile retired compact output reservations with the historical `:bounded-output-v2` identity, a durably stopped provider execution, publication/discard authority and verified absence of both exact output keys and the output prefix. Applying the reviewed batch recomputes the evidence and checks the review digest. Unverifiable entries, old per-file formats without retained inventories, and other unexplained charges remain held. This is intentional preservation of evidence, not a claim that all historical allowance can always be recovered.

Private-state receipts created on the native database retain their native transaction identity. Managed GC does not interpret those XIDs in the external database, or infer closure from coincidentally equal XID numbers. They remain held for separately verified migration reconciliation. New managed private-state receipts use the new shared database publication authority.

External database recovery protection

An older database backup can still reference a file that the current database has deleted. SQL commit alone is therefore insufficient evidence for deleting its encrypted object. Managed accounting records an immutable retirement time for each artifact allocation or private-state object and waits for a verified recovery boundary strictly later than that retirement.

The database backup coordinator or platform operator must:

1. Inspect the external provider's complete retained backup/PITR inventory and the native recovery roots retained during cutover.

2. Retire or make permanently unavailable every recovery point older than the proposed boundary. Fence/decommission any previous database authority.
3. Verify that the current external recovery epoch matches every active replica. Preserve the reviewed inventory as operational evidence and calculate its SHA256.
4. Use **Database recovery protection** in `/admin/hf-accounting` to record the oldest recoverable UTC timestamp, inventory reference, evidence SHA256 and reason, explicitly attesting to the retirement of all older roots.

PF binds this authenticated operator attestation to the current external epoch. It does not independently query a cloud backup provider, and it does not infer backup retirement from elapsed time, an empty object listing or a retention setting. An automated backup coordinator may use the same authenticated administrative operation only after independently verifying the provider inventory. Merely advancing a date to obtain a quota refund is unsafe.

The API is `POST /api/admin/hf-accounting/recovery-floor` and accepts:

JSON | Copy exact lines from the packaged Markdown

```
{
  "floor_utc": "2026-10-09T12:00:00Z",
  "expected_generation": 0,
  "evidence_sha256": "64-lowercase-hex-characters-from-your-reviewed-inventory",
  "source": "Your provider inventory evidence reference",
  "native_roots_retired": true,
  "all_older_recovery_points_retired": true,
  "reason": "Explain the independently verified backup retirement"
}
```

The example hash is descriptive and must be replaced with a real SHA256. A future timestamp, stale generation, missing attestation or backwards boundary is rejected. First enabling protection also waits for every previously admitted conservative physical purge to confirm completion. The pending purge journal survives a process crash and is completed only by an exact-scope deletion retry. After a valid boundary advances, bounded background maintenance can delete and credit eligible objects. No boundary means no safe automatic artifact or private-state reclamation. Output settlement can still recover unused maximum-output allowance because it does not delete retained file data.

Recovery protection is permanent once enabled: switching quota policy back to conservative does not disable it. A never-enabled conservative deployment keeps its prior physical purge behavior and issues no lifecycle credits.

Never restore an older database snapshot against a namespace whose referenced objects have been reclaimed. Recovery must select a point at or after the recorded boundary with its matching object namespace. Restoring or replacing database authority is a controlled fleet-wide recovery/migration, not an online connection-string edit.

Administration and operational evidence

`GET /api/admin/hf-accounting` returns shared policy, charges, recovery boundary, receipt counts and execution recovery status. `GET /api/admin/hf-accounting/audit` provides bounded pages of policy, credit, reconciliation and recovery-boundary events. Use the returned `next_cursor` to traverse retained events; IDs are pagination identities rather than chronological sequence numbers.

Database admission, lifecycle changes and audit writes share a consistent lock order. Remote imported-receipt checks finish before the aggregate meter lock. Operator receipt counts use grouped SQL queries; immutable native provenance is not reread for every ordinary admission. Background cleanup uses bounded work and shared maintenance coordination across replicas.

The included tests cover exact accounting, policy fencing, legacy permits, platform access, migration preservation, writer/reference/recovery barriers and real PostgreSQL concurrent credit. Run the PostgreSQL suite with `PF_TEST_POSTGRESQL_URL` pointing to an isolated qualification database. It creates and removes only its own randomized schemas. Unit tests and these integration tests do not prove a particular public-user capacity or qualify a provider's backup inventory. Those require the release's workload and deployment qualification process against the actual provisioned environment.

Consolidated r34: redeploy the existing native HF service

This is the native HF upgrade route for the consolidated full r34 release. It preserves your controller/code/runtime Spaces, Bucket and prefix, tokens, Variables, Secrets, users, projects and encrypted state. No external PostgreSQL service is needed. Do not apply the earlier patch ZIPs after this release. The complete fresh-install procedure remains in HUGGING_FACE_NATIVE_DEPLOYMENT.md (HUGGING_FACE_NATIVE_DEPLOYMENT.md).

The release incorporates the approved native latency, startup, response, runtime, WAL batching, ownership queue and persistence response recovery fixes, plus applicable shared application improvements. It retains native ownership fencing and durable acknowledgement; there is no HTTP ownership bypass. Schema **10** is applied on native startup. Prepare a verified backup before startup: schema-9 or older application code must not serve the migrated database. This full release includes both Qualified Lifecycle and Recovery Correction packages, public email, account-wide admission and execution recovery. See R34_PUBLIC_SERVICE_DEPLOYMENT.md (R34_PUBLIC_SERVICE_DEPLOYMENT.md) before opening registration.

1. Extract and configure local tools

Check the new archive against its companion `.sha256`, extract the `.xz`, then extract the `.zip` into a new folder. Preserve the old release and generated contexts. In **Windows CMD**, set actual paths and existing HF resource names:

BAT | Copy exact lines from the packaged Markdown

```
certutil -hashfile "FULL_PATH_TO_NEW_RELEASE.zip.xz" SHA256
set "PF_PROJECT_ROOT=D:\pf-r34-consolidated\privileged-frame-simulation"
set "PF_HF_TOOLS=%USERPROFILE%\pf-hf-tools-r34-20261005"
set "PF_HF_OWNER=YOUR_HF_USERNAME_OR_ORGANIZATION"
set "PF_HF_APP_REPO=%PF_HF_OWNER%/pf-simulation-saas"
set "PF_HF_CODE_REPO=%PF_HF_OWNER%/pf-simulation-runner"
set "PF_HF_RUNTIME_REPO=%PF_HF_OWNER%/pf-runtime-r34"
set "PF_HF_BUCKET_NAME=pf-simulation-state"
set "PF_HF_BACKUP_BUCKET_NAME=pf-simulation-state-backup-REPLACE_WITH_UNIQUE_SUFFIX"
set "PF_HF_UPLOAD=%USERPROFILE%\pf-r34-consolidated-generated"
set "PF_HF_DEPLOY_STATE=%USERPROFILE%\pf-r34-consolidated-deployment"
call "%PF_HF_TOOLS%\Scripts\activate.bat"
cd /d "%PF_PROJECT_ROOT%"
python -c "import sys,huggingface_hub; print(sys.executable); print(sys.version.split()[0]);
print(huggingface_hub.__version__)"
hf auth whoami
```

Use the existing Python 3.12 tools environment with `huggingface_hub==1.32.0` and `truststore>=0.10,<1`; see section 2 of the full guide if preparation is needed. The generated and deployment paths must not already exist. Choose new suffixes if they do. Local `set` statements do not change HF Variables.

2. Drain, pause and back up existing state

Keep users and scheduled clients out of the maintenance window. Let simulations, private workflows, assistant work and result collection finish, or cancel through PF and verify terminal HF Jobs state. Confirm no other controller writes this Bucket/prefix. Record current controller/code commits, code SHA256, runtime image, Variables and secure references to stable keys. Do not regenerate keys.

BAT | Copy exact lines from the packaged Markdown

```
hf spaces pause "%PF_HF_APP_REPO%"
hf spaces info "%PF_HF_APP_REPO%" --expand sha,runtime
```

Require `PAUSED`. Complete the private Bucket-to-Bucket backup and source/backup manifest comparison in section 11.3 (HUGGING_FACE_NATIVE_DEPLOYMENT.md#113-create-and-verify-a-private-hf-backup). Keep the verified copy separate from active state. It includes encrypted quota usage and reservation history. Preserve

`PF_HF_MAX_STORAGE_BYTES` at its approved value; for the service already raised to 32 GiB it is **34359738368**. Keep the encrypted meter and Space Variable aligned. Do not reduce it during this upgrade. Deleting files does not automatically refund conservative lifetime admission.

This backup is the only additional resource in this upgrade procedure; existing Spaces, production Bucket, tokens and runtime Secrets are reused. If you already have an equivalent verified, consistent pre-upgrade backup under your recovery policy, retain its identity and evidence instead of creating a duplicate.

3. Generate and qualify matching code

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\build_space.py --output "%PF_HF_UPLOAD%"
python deploy\saas\huggingface_native\deploy_release.py verify --bundle "%PF_HF_UPLOAD%"
python deploy\saas\huggingface_native\deploy_release.py --system-ca upload-code --bundle "%PF_HF_UPLOAD%"
--repo "%PF_HF_CODE_REPO%"
```

Require `local_bundle_verified`, then `private_code_uploaded`. Save the returned `code_revision` and `code_sha256`. The latter is the **private code ZIP** hash, not the full `.zip.xz` release hash. The upload is guarded by the current private code repository commit and verified against exact remote file identities.

Compare the newly generated `runtime.zip` against the retained context used for your qualified image, as in section 11.4. Reuse its digest only if those bytes and image provenance match. Otherwise upload only the generated public `runtime/`, wait for its build and obtain the new immutable digest using sections 4-5. Keep code Spaces private. The controller remains Private during maintenance; use Protected visibility only after the public readiness gates pass.

Set local pins from the verified results:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_CODE_REVISION=NEW_40_CHARACTER_COMMIT"
set "PF_HF_CODE_SHA256=NEW_64_CHARACTER_PRIVATE_CODE_SHA256"
set "PF_HF_JOB_IMAGE=YOUR_REGISTRY_IMAGE_AT_SHA256_DIGEST"
python deploy\saas\huggingface_native\qualify.py --namespace "%PF_HF_OWNER%" --runtime-image
"%PF_HF_JOB_IMAGE%" --code-repo "%PF_HF_CODE_REPO%" --code-revision "%PF_HF_CODE_REVISION%" --code-sha256
"%PF_HF_CODE_SHA256%" --flavor cpu-basic --system-ca
```

Review the dry-run specification. Run the same command with `--submit` to launch one billable confinement/scientific-runtime qualification Job. Require `COMPLETED`, `confinement_probe_passed=true` and `scientific_runtime_probe_passed=true`. Use your intended supported flavor in both calls. If submission is uncertain, inspect the existing Job before retrying. Qualify a distinct assistant image separately against the same new code pins.

4. Update code pins, preserving state configuration

In the existing controller's HF **Settings**, set `PF_HF_CODE_REPO`, `PF_HF_CODE_REVISION`, `PF_HF_CODE_SHA256` and the qualified `PF_HF_JOB_IMAGE`. Keep the existing namespace, Bucket, prefix, quotas, stable Secrets, email configuration, public origin and budgets. Preserve `PF_SAAS_DEPLOYMENT_PROFILE=single`; no managed-DB URL is introduced. Remove `PF_HF_BOOTSTRAP` from both Variables and Secrets for this initialized service. Do not reinstate first-time administrator credentials.

Review shared bounded settings in the shared capacity settings guide (R34_SHARED_CAPACITY_SETTINGS.md); retain finite admitted capacity within the native controller's connection and worker limits. Defaults are not a claim of hundreds of thousands of concurrent users. If editing HF settings restarts the controller, pause it again before proceeding.

5. Prepare, apply and verify the controller

Run these **one at a time**, stopping when a command fails:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\deploy_release.py --system-ca prepare --bundle "%PF_HF_UPLOAD%"
--space "%PF_HF_APP_REPO%" --state-dir "%PF_HF_DEPLOY_STATE%"
python deploy\saas\huggingface_native\deploy_release.py --system-ca apply --state-dir "%PF_HF_DEPLOY_STATE%"
hf spaces wait "%PF_HF_APP_REPO%" --timeout 25m
python deploy\saas\huggingface_native\deploy_release.py --system-ca status --state-dir
"%PF_HF_DEPLOY_STATE%"
hf spaces logs "%PF_HF_APP_REPO%" --tail 200
```

- `prepare` makes no remote changes. It verifies the generated seal, immutable private code, current controller revision and public configuration fingerprint. It lists exact obsolete release-file deletions, including the old extra qualification registries responsible for the earlier clean-check mismatch. Unexpected unrelated files stop it.
- `apply` confirms pause, rechecks the plan and submits one parent-guarded commit, then requests restart. It does not edit Variables, Secrets, Bucket objects, quota values or database rows. The running new application handles schema migration during its ordinary native startup.
- `status` verifies exact controller bytes and the running revision. Require `full_release_installed_running`, `source_bytes_verified=true` and `running_revision_verified=true`. The flags for HTTP readiness and performance remain false until separately measured; `RUNNING` alone does not prove them.

If apply has an uncertain response, run `status` before another write. It can recognize an exact completed commit and record it without repeating it. If that commit remains `PAUSED`, use the existing `hf spaces restart` command, wait, then check status. If preparation's revision or configuration is stale, inspect the change and prepare into a new directory; never force past the guard.

6. Accept the new release before reopening access

Sign in using an existing PF account at the direct application URL. Check:

1. Existing organizations, projects, saved runs, attachments, BYOK and BYOS state.
2. Archive and restore a disposable organization and project, recording request and visible UI completion timing. Confirm no new ownership-bypass mode.
3. A small simulation through collection, plots/report and authenticated Run ZIP download; a separate cancellation through terminal provider state.
4. A new saved record surviving a controlled restart, with the same quota limit, preserved usage history and no duplicate provider submission.
5. Representative private workflow and assistant operations with bounded settings.

Retain the full-release checksum, generated manifests, code/runtime/controller pins and acceptance results. The user's successful earlier native patch tests are valuable prior evidence; the newly consolidated release still needs this acceptance run. No live HF deployment or live performance result is implied by the packaged offline tests.

If startup or acceptance fails, preserve the same state and obtain the first relevant log error. Do not repeatedly restart, enable bootstrap, reset counters, change keys/prefix, reapply old patches or switch back to schema-9 or older code against migrated data. Use a reviewed forward repair or a verified compatible state restore as described in the recovery guide (HUGGING_FACE_R34_RECOVERY.md).

PF Simulation r34 - complete Hugging Face deployment guide

Current public-service and schema-10 upgrade instructions: r34 public service deployment (R34_PUBLIC_SERVICE_DEPLOYMENT.md). This includes both 2026-10-09 lifecycle packages. Large active-user capacity requires the measured qualification described there.

Use the **complete r34 release supplied with this guide**. Extract it into a new local project folder; this installation does not require the earlier compatibility update ZIP or separately downloaded repair scripts. Compare the archive's SHA256 with its companion checksum before extraction. Keep the complete release and its validation records.

For receiver startup and durability settings, also read WAL catch-up and batching (R34_HF_WAL_CATCHUP_AND_BATCHING.md).

This native mode runs the PF SaaS application, PostgreSQL process, scientific Jobs and primary durable storage on Hugging Face. No separate Linux server is required. Email delivery and users' BYOK/BYOS providers remain external services. The standalone and ordinary Linux/Docker deployment instructions remain separate.

If you already deployed: a clean local extraction is not a remote reset. Preserve your private Spaces, bucket, deployment prefix and all encryption, session and database keys. Do not discard working state or substitute fresh keys. Sections 1-9 cover a new installation. For an earlier r34 deployment that is already running, use the complete upgrade procedure in section 11 (#11-upgrade-an-existing-r34-deployment). The recovery guide (HUGGING_FACE_R34_RECOVERY.md) explains the recovery boundaries.

Where each action happens

Label	Open this
Browser - Hugging Face	The HF website, signed in as the deployment operator.
Windows CMD	Start menu → type **cmd** → Command Prompt; ordinary privileges.
Administrator PowerShell	Start menu → PowerShell → **Run as administrator**; only explicitly labelled Windows setup/repair commands.
Docker Desktop	The Docker Desktop application on your PC; optional for the HF cloud-build route.
Browser - email/DNS provider	SMTP2GO and the provider managing your domain's DNS, such as Network Solutions.

Paste each command as one line in the named shell. Do not copy a prompt such as `D:\>`, PDF line numbers, or visual line wraps. Windows CMD uses `%NAME%` variables; PowerShell uses different syntax. Commands here are CMD unless labelled otherwise. Replace `YOUR_HF_USERNAME` and other `REPLACE_...` placeholders before running. Run setup and upload commands **one at a time** and stop if a command fails. Only the explicitly labelled read-only diagnostic batches may continue after an individual failure.

Milestones: create resources → prepare Windows tools → build/upload the three contexts → record immutable pins → run one confinement test → configure email and controller → verify startup → qualify the complete service. A successful upload or runner test is not proof that the complete SaaS deployment is ready.

Consolidated r34 starting point

The October 2026 consolidated release incorporates the approved native Latency, Application Latency (final corrected form), Startup Recovery, Response Correction, Runtime Recovery, WAL Batching, Ownership Queue and Persistence Response Recovery changes. It also contains the shared paging, bounded admission, database-pool, lease/recovery and polling improvements. Do not replay the historical patch ZIPs. User-reported live responsiveness of the prior native patch combination is recorded separately from this release's offline regression results.

Native HF remains one PostgreSQL authority inside the private controller, with its encrypted state and WAL in the existing Bucket. No external database service is required. The optional managed PostgreSQL route is a separate explicit choice; see deployment choices (R34_CONSOLIDATED_DEPLOYMENT.md). Preserve `PF_SAAS_DEPLOYMENT_PROFILE=single` for this guide.

The shared application schema is now **10**. Native startup restores the existing cluster, applies its additive schema upgrade and initializes shared counters and admission tables before serving. Existing users, keys and quota reservations are retained. Take the section-11 backup first: returning to schema-9 or older code is not a code-only rollback after migration. External PostgreSQL migrations are not part of the native route.

For the shortest existing-Space route, see the consolidated native upgrade checklist (R34_NATIVE_HF_RELEASE_UPGRADE.md).

1. Create the HF resources and their credentials

Where: Browser - Hugging Face.

1. Create an account at https://huggingface.co/join, verify its email, and enable account security/recovery controls. Review current eligibility and charges for Docker Spaces, Jobs and Buckets before purchasing. HF currently requires a paid eligible account to create Docker Spaces; CPU Basic having no hourly Space charge does not make the whole deployment free.
2. At https://huggingface.co/new-space, create these **three Docker Spaces**. Select your username/organization as owner, **Docker** as SDK, and a blank Docker template if offered. The names below are used by the commands; use other names only if you change every matching reference.
3. At https://huggingface.co/new-bucket, create the private state Bucket below. Create it **before** configuring the token so it can be included in its scope.

Resource name	Type / visibility	What will be uploaded
`pf-runtime-r34`	**New Public Docker Space**	Only generated `runtime/`: public dependencies and generic bootstrap.
`pf-simulation-runner`	**Private Docker Space**	Generated `jobs/`: private PF code payload.
`pf-simulation-saas`	**Private Docker Space**	Generated `space/`: trusted SaaS controller.
`pf-simulation-state`	**Private Storage Bucket**	Encrypted service state/artifacts, written by the controller.

Never make a previously private runner/controller Space public. Its Git history may contain proprietary code. A new public runtime Space must contain only the reviewed `runtime/` context, with no PF source, user files or secrets. Keep the Bucket private. Existing private code/controller Spaces may be reused.

4. Open https://huggingface.co/settings/tokens. Create a **fine-grained** setup token, for example `pf-r34-setup`, with read/write access to those three Spaces, read/write access to **the state Bucket**, and permission to start/manage Jobs. Jobs management includes inspecting and cancelling the operator's Jobs. Space permissions do not implicitly include a separate private Bucket.
5. In the resource selector, search for the Bucket name and select the **Bucket** belonging to your owner, if it is offered there. Confirm its resource type and full owner/name; do not select a similarly named Space. If the UI cannot find it, check the Bucket's exact URL and available scope controls rather than granting every resource or recreating the Bucket.
6. Save the token securely. In that token's menu choose **Generate S3 credentials** and save both the `HFAK...` access-key ID and matching secret access key. The secret is displayed once. A separate narrower controller token may later replace the setup token: it needs Jobs management, private code access and read/write access to the deployment Bucket, not upload rights to all Spaces.

HF-native CLI access uses the **HF token**; the S3 API uses the **S3 pair** derived from an HF token. S3 permissions inherit the parent token's scope. These are different credentials; an S3 key is not an `hf auth login` token. HF S3 credential documentation.

Checkpoint: all four resources exist with the stated visibility; the token's scope includes the Bucket as well as the Spaces. Ordinary PF users must never receive this operator token.

2. Prepare the full release and Windows tools

Where: Windows CMD. To check the download, replace the path in this example and compare the result with the separately supplied checksum:

BAT | Copy exact lines from the packaged Markdown

```
certutil -hashfile "FULL_PATH_TO_DOWNLOADED_RELEASE_ARCHIVE" SHA256
```

Where: File Explorer / archive utility. If the download ends in `.zip.xz`, extract the outer `.xz` first to obtain the `.zip`, then extract that ZIP into a fresh directory. 7-Zip or another compatible utility can open the outer layer; Explorer's **Extract All** can extract the resulting ZIP. Do not run scripts inside an archive preview or overwrite an earlier project. Locate the project root containing `run_pf_saas.py`, `pf_hf_native` and `deploy`. All helpers used below are already under `deploy\saas\huggingface_native` in that release. Do not use earlier scripts from Downloads.

Where: Windows CMD. Set your nonsecret working values. Change the project path to your actual extraction, and the owner to your HF username/organization:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_PROJECT_ROOT=D:\privileged-frame-simulation"
set "PF_HF_OWNER=YOUR_HF_USERNAME"
set "PF_HF_TOOLS=%USERPROFILE%\pf-hf-tools-r34"
set "PF_HF_UPLOAD=%USERPROFILE%\pf-hf-native-upload-r34-full"
set "PF_HF_DEPLOY_STATE=%USERPROFILE%\pf-hf-native-deploy-r34-full"
set "PF_HF_RUNTIME_REPO=%PF_HF_OWNER%/pf-runtime-r34"
set "PF_HF_CODE_REPO=%PF_HF_OWNER%/pf-simulation-runner"
set "PF_HF_APP_REPO=%PF_HF_OWNER%/pf-simulation-saas"
set "PF_HF_BUCKET_NAME=pf-simulation-state"
cd /d "%PF_PROJECT_ROOT%"
py -3.12 --version
```

These `set` values last only in this CMD window. If you reopen it, repeat this block and the activation command below. They do **not** configure your HF Space.

Use 64-bit Python **3.12**. If it is missing, install it from Python's official Windows downloads and confirm `py -3.12 --version` works. The operator tooling was exercised with Python 3.12.10; use a maintained compatible 3.12 installation and record the patch version. Do not let an unrelated Python 3.13 environment silently run these tools.

Create the tools environment only when `PF_HF_TOOLS` names a new directory. An existing correctly configured 3.12 environment can instead be activated; do not overwrite an existing 3.13 environment.

BAT | Copy exact lines from the packaged Markdown

```
py -3.12 -m venv "%PF_HF_TOOLS%"
call "%PF_HF_TOOLS%\Scripts\activate.bat"
python -m pip install "huggingface_hub==1.32.0" "truststore>=0.10,<1"
where python
where hf
python -c "import sys,huggingface_hub; print(sys.executable); print(sys.version.split()[0]);
print(huggingface_hub.__version__)"
hf auth login
hf auth whoami
hf buckets info "%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%"
```

In the login menu choose **Paste an access token**, if offered, and enter the setup token only at the private prompt. Do not use `--token SECRET` in a command. The first `python` and `hf` paths should be inside the tools environment; versions should show Python 3.12 and Hub SDK 1.32.0. A successful Bucket information check confirms this HF token can see it; it does not yet prove S3 writes/durability. If any checkpoint fails, use section 10 before proceeding.

3. Choose the Docker tooling route

Both routes deploy Docker images on HF. **The cloud-build route used in sections 4-8 needs Python and the HF CLI, but no local Docker daemon.** HF builds the uploaded Dockerfiles. The packaged registry helper reads image metadata without Docker. It does not replace the subsequent live HF Job test.

Standard option: install Docker Desktop for local Docker commands

Where: your Windows PC, then Administrator PowerShell / Docker Desktop.

1. Check Docker's current Windows requirements: supported 64-bit Windows, CPU SLAT support, at least 8 GB RAM, and hardware virtualization enabled in BIOS/UEFI. The WSL 2 backend requires WSL 2.1.5 or later. On a managed PC/virtual machine, confirm administrator/host support.
2. **Administrator PowerShell:** if WSL is absent, install it without a separate Linux distribution, then restart Windows when requested:

POWERSHELL | Copy exact lines from the packaged Markdown

```
wsl --install --no-distribution
```

If WSL is already installed, update it instead:

POWERSHELL | Copy exact lines from the packaged Markdown

```
wsl --update
```

3. Install Docker Desktop from its official page. Select the **WSL 2 backend**, start Docker Desktop, and complete its first-run prompts.
4. **Docker Desktop:** use **Linux containers** and, where shown, **Settings → General → Use the WSL 2 based engine**. Wait for the engine to run.
5. **Windows CMD:** check the local server:

BAT | Copy exact lines from the packaged Markdown

```
wsl --version
docker version
docker info --format "{{.OSType}}"
```

Expected: both **Client** and **Server** sections, followed by `linux`. A client version alone is not a working engine. A separate Ubuntu installation is not needed for these CMD commands. Microsoft WSL options.

After the runtime has built in section 4, you can optionally authenticate your local Docker client for registry operations. **Windows CMD:**

BAT | Copy exact lines from the packaged Markdown

```
docker login registry.hf.space
```

At its prompts, enter your **personal HF username** and an HF token as the password. An organization owner name is not the username of the person logging in. Keep the token out of command arguments. This local login is optional and does not replace section 5's anonymous public-image check or live HF Job test.

If local Docker cannot run, continue with the cloud-build route below. The virtualization and named-pipe HTTP 500 troubleshooting in section 10 explains local repair; neither problem requires changing PF's Linux/Docker server edition.

4. Build and upload the runtime and private code

Where: Windows CMD, activated tools environment, project root.

BAT | Copy exact lines from the packaged Markdown

```
cd /d "%PF_PROJECT_ROOT%"
python deploy\saas\huggingface_native\build_space.py --output "%PF_HF_UPLOAD%"
```

`PF_HF_UPLOAD` must name a **new, nonexistent** output directory. If it already exists, choose a new name, change that variable, and keep using it for every upload. Do not mix an old `pf-hf-native-upload` folder with the current build.

The packaged compatibility-named alias below delegates to the same current three-context builder; it is an alternative command, not an additional patch:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\build_space_windows_fix.py --output "%PF_HF_UPLOAD%"
```

Run **one** builder command, not both against the same output. Both accept `--source` and `--output`. The builder produces `runtime/`, `jobs/`, `space/` and matching ZIPs/manifests. It checks an explicit input allowlist and rejects symlinks. The controller context includes `base-code-manifest.json`; keep it with its matching `source/` tree and `jobs/pf-code.zip`. Native startup checks this inventory against `PF_HF_CODE_SHA256` before admitting a new execution. Verify that the generated runner and controller describe the same actual payload:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import hashlib,json,sys; from pathlib import Path; p=Path(sys.argv[1]);
a=json.loads((p/'jobs'/'code-manifest.json').read_text(encoding='utf-8'));
b=json.loads((p/'space'/'base-code-manifest.json').read_text(encoding='utf-8'));
h=hashlib.sha256((p/'jobs'/'pf-code.zip').read_bytes()).hexdigest(); assert a['sha256']==b['sha256']==h,
'Generated contexts do not match'; print('VERIFIED_CODE_SHA256='+h)" "%PF_HF_UPLOAD%"
```

Require `VERIFIED_CODE_SHA256=` followed by the hash from this build. Do not substitute the full release archive checksum for this private-code checksum.

Review `runtime/` before public upload. Upload only each context's contents, never the full release, parent directory, live data or environment files. Do not add any secrets to the public runtime Space.

BAT | Copy exact lines from the packaged Markdown

```
hf upload "%PF_HF_RUNTIME_REPO%" "%PF_HF_UPLOAD%\runtime" . --repo-type space
python deploy\saas\huggingface_native\deploy_release.py --system-ca upload-code --bundle "%PF_HF_UPLOAD%"
--repo "%PF_HF_CODE_REPO%" --fresh
python -c "import sys; from huggingface_hub import HfApi;
print(HfApi().space_info(repo_id=sys.argv[1]).sha)" "%PF_HF_CODE_REPO%"
python -c "import json,sys; print(json.load(open(sys.argv[1], encoding='utf-8'))['sha256'])"
"%PF_HF_UPLOAD%\jobs\code-manifest.json"
```

Save the last two outputs: the private upload's exact **40-character commit** and the payload's **64-character SHA256**. They must come from **this build/upload**, not a previous chat or release. In **Browser - HF**, wait for the public runtime Space's build to succeed. The runtime/code pages are not the PF application.

Checkpoint: successful uploads, a built public runtime, and two recorded private-code pins. A failed upload may leave uncertainty about remote commits; check Files/history before deciding nothing changed. Xet 403 recovery is in section 10. Do not delete repository files/history as a routine workaround.

5. Resolve the public image and qualify one HF Job

Where: Browser - Hugging Face, public runtime Space. Open its **Run with Docker** / **Run locally** instructions (possibly in the page menu) and copy the actual `registry.hf.space/...:latest` image reference. Do not guess it from the Space URL, and do not use the private runner's old image.

Where: Windows CMD. Replace the placeholder with that image:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_RUNTIME_TAG=REPLACE_WITH_ACTUAL_PUBLIC_REGISTRY_IMAGE_LATEST"
python deploy\saas\huggingface_native\hf_registry_digest.py --anonymous --system-ca --image
"%PF_HF_RUNTIME_TAG%"
```

This packaged helper requires no Docker daemon. `--system-ca` uses Windows' trusted certificate authorities via `truststore`; TLS verification remains on. The anonymous check must succeed without reading/sending an HF token. Record its complete `registry_image` value ending in `@sha256:` plus 64 hex characters. Public visibility alone is not proof that a particular registry route is anonymously accessible. An authenticated metadata lookup is not a substitute.

Set the three outputs below. The private commit and payload hash are lowercase hexadecimal; no angle brackets or spaces belong inside these values.

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_JOB_IMAGE=REPLACE_WITH_FULL_REGISTRY_IMAGE_AT_SHA256_DIGEST"
set "PF_HF_CODE_REVISION=REPLACE_WITH_40_HEX_PRIVATE_COMMIT"
set "PF_HF_CODE_SHA256=REPLACE_WITH_64_HEX_PAYLOAD_SHA256"
python deploy\saas\huggingface_native\qualify.py --namespace "%PF_HF_OWNER%" --runtime-image
"%PF_HF_JOB_IMAGE%" --code-repo "%PF_HF_CODE_REPO%" --code-revision "%PF_HF_CODE_REVISION%" --code-sha256
"%PF_HF_CODE_SHA256%" --flavor cpu-basic --system-ca
```

This is a **dry run**: no Job is submitted. Review the image, private revision, hash, empty secrets and read-only `/pf-code` mount. The bootstrap verifies the payload before extracting/importing it. When ready, the following command explicitly submits **one paid Job**, with a five-minute provider timeout:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\qualify.py --namespace "%PF_HF_OWNER%" --runtime-image
"%PF_HF_JOB_IMAGE%" --code-repo "%PF_HF_CODE_REPO%" --code-revision "%PF_HF_CODE_REVISION%" --code-sha256
"%PF_HF_CODE_SHA256%" --flavor cpu-basic --system-ca --submit
```

The helper supplies the short name `pf-r34-confinement-check`. Save the Job ID/URL. Acceptance requires verified provider identity, a passing mandatory confinement probe, successful confined FMPy Dahlquist and QuTiP decay probes, successful exit and provider stage **COMPLETED**. Require `self_test_completed: true` and `scientific_runtime_probe_passed: true`. These small dependency probes do not qualify every scientific workflow or the complete SaaS deployment. Merely starting or pulling an image is insufficient. Build the runtime context from this release and record its new digest; updating private code while retaining an earlier dependency image does not install the required scientific packages.

If the helper reports `RuntimeError`, a timeout or an uncertain response after printing a Job ID, **do not submit again**. Inspect the existing Job in section 10 first. Completed logs can establish that execution ran, but do not by themselves resolve a failed provider-identity check. A generic RuntimeError does not identify its cause: resolve the helper error and compare the actual Job with the expected identity before service admission. A cancellation request is not proof the VM stopped. Failed confinement remains blocking; do not remove the probe or request privileges.

The self-test runs outside PF's admission ledger and can consume HF credits separately. Cloud Space operation/build resources are also outside that ledger. Rebuilding code changes its commit/hash; rebuilding a runtime may change its digest. Requalify new identities. Tags, `hf.co/spaces/OWNER/SPACE` shorthand and digests copied from another deployment are not production pins.

Checkpoint: record this runtime digest, private repo/commit/hash, flavor and successful Job evidence. These CMD variables are only local notes. **You must enter their actual values into the controller's HF Variables in section 7.**

6. Set up email and generate stable secrets

Email delivery

Where: Browser - SMTP2GO, then your DNS provider. Production local authentication requires functioning account email even while self-registration is disabled. The r34 mailer supports encrypted SMTP and optional Resend HTTPS. This section retains the SMTP2GO path; a provider migration is not required. See public email delivery (PUBLIC_EMAIL_DELIVERY.md) for both transports and their shared durable queue. HF Spaces currently permits outgoing ports **80, 443 and 8080**; the usual SMTP ports 587/465 are not usable under that published policy. HF networking.

A compatible documented setup is SMTP2GO SMTP relay:

1. Create the SMTP2GO account and review its plan/quota.
2. In **Sending → Verified Senders**, add your sending domain. In the domain's actual DNS manager, add the exact three CNAME records SMTP2GO displays. They handle return-path/SPF, DKIM and tracking. Use your displayed values, not another user's selector numbers.
3. For Network Solutions DNS: **Domains → your domain → Advanced Tools → Advanced DNS Records → Manage → Add Record → CNAME → Other Host**. Its form may append the domain; use the host part only when it does. Network Solutions setup.
4. Add records without replacing website, MX, nameserver, SPF or existing DKIM records. SMTP2GO requires no change to existing SPF/MX. If `link` is already used, choose an unused tracking hostname in SMTP2GO. Preserving the existing hosting records keeps your Network Solutions website and mailbox in place.
5. In SMTP2GO click **Verify**. This checks DNS; it does not add records. Inspect each failure's tooltip. Propagation is a possible delay, not a substitute for checking the records were saved.
6. Create an **SMTP User** under **Sending → SMTP Users**. Save its username and password. These are not the website login, API key, mailbox password or PF administrator password. Open/click tracking can remain disabled.

For each DNS row, the **Host/Name** is the alias under your own domain; the **Alias/Target** is SMTP2GO's supplied destination, such as `return.smtp2go.net`, `dkim.smtp2go.net` or `track.smtp2go.net`. Targets are hostnames, without `https://` or a path. Use the values displayed for your account. TTL is in seconds: `3600` = 1 hour, `7200` = 2 hours, `14400` = 4 hours. Select the matching duration if the DNS form uses a dropdown. TTL is caching behavior, not a guaranteed verification waiting time.

Choose the actual sender mailbox you want recipients to see, for example `support@YOUR_VERIFIED_DOMAIN`. Domain verification authorizes sending; it does not choose or create that mailbox. Enter the chosen address as `PF_SAAS_SMTP_FROM`; it is not the SMTP username or your HF account login.

Use `mail.smtp2go.com`, port **443**, **SSL=true**, **STARTTLS=false** in section

7. The public-readiness checker requires exactly one TLS mode and an allowed outbound port for HF SMTP. Another provider's encrypted SMTP endpoint on an allowed port can be used after verification. Changing Network Solutions' normal SMTP port to 443 alone does not make its server support it. Verified sender details.

Application and storage keys

Where: Windows CMD, privately. Generate these only for a genuinely new deployment. Save each output in a password manager/offline recovery record, labelled with its exact Secret name; then enter it in HF Secrets. These commands print secrets, so do not share their output.

Run once for `PF_HF_STATE_KEY`:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import secrets; print(secrets.token_hex(32))"
```

Run the following **twice**, keeping separate outputs for `PF_SAAS_ASSISTANT_CREDENTIALS_KEY` and `PF_SAAS_BYOS_CREDENTIALS_KEY`:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import base64,secrets; print(base64.urlsafe_b64encode(secrets.token_bytes(32)).decode())"
```

Run this separately for each of `PF_SAAS_SECRET_KEY`, `PF_HF_PG_ADMIN_PASSWORD`, `PF_HF_PG_APP_PASSWORD` and `PF_HF_PG_WAL_PASSWORD`:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import secrets; print(secrets.token_urlsafe(48))"
```

Choose/save a separate strong `PF_HF_ADMIN_PASSWORD` for your first PF login. Do not reuse these values. Keep state, database, session and vault keys stable across restarts/rebuilds: losing or replacing them can make saved data unrecoverable. HF Secrets cannot be read back as a recovery method; retain your own secure copy.

7. Configure the private controller Space

Where: Browser - Hugging Face → your `pf-simulation-saas` **Space → Settings → Variables and secrets.** Add the following names to **Variables** or **Secrets** as indicated. Enter literal values, without CMD `set` syntax, outer quotes, or unexpanded `%PF_HF_OWNER%` placeholders. Every value is case-sensitive.

Get HF's actual assigned app host in **Windows CMD**:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import sys; from huggingface_hub import HfApi;
print(HfApi().space_info(repo_id=sys.argv[1]).host)" "%PF_HF_APP_REPO%"
```

Use that host as an HTTPS origin, for example `https://ACTUAL-HOST.hf.space`, with no trailing path; keep an existing `https://` prefix rather than adding it twice. If empty, obtain it from the Space's open-app link after its initial build. Do not guess it from the account name.

Variable	Value/purpose
`PF_HF_NAMESPACE`	Your HF owner from `PF_HF_OWNER`; type the actual username/organization.
`PF_HF_BUCKET`	Bare bucket name, without namespace or `/`.
`PF_HF_PREFIX`	Stable deployment prefix, for example `pf-saas`. Never change it to bypass a full ledger.
`PF_HF_JOB_IMAGE`	Public runtime image reference ending in `@sha256:` and 64 hex characters, qualified in section 5.
`PF_HF_CODE_REPO`	Private Space repository containing the generated code payload, e.g. `YOUR_HF_USERNAME/pf-simulation-runner`.
`PF_HF_CODE_REVISION`	Exact 40-character Git commit of that private payload upload; never a branch/tag.
`PF_HF_CODE_SHA256`	SHA256 of the generated code archive, recorded by the builder.
`PF_HF_ASSISTANT_IMAGE`	Optional distinct public runtime digest for private workflows/assistant code; defaults to the job image. The same pinned private code payload is used.
`PF_HF_JOB_FLAVOR`	`cpu-basic` or `cpu-upgrade`, qualified for the actual workload.
`PF_HF_BOOTSTRAP`	`true` for the first empty deployment only; remove/set `false` after successful initialization. It never replaces an existing remote cluster.
`PF_HF_PG_START_TIMEOUT_SECONDS`	Bounded PostgreSQL startup/recovery wait, default `600`; use `30` to `3600` seconds as justified by restore qualification.

Variable	Value/purpose
PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS	Separate bounded wait for the durable WAL receiver to finish catch-up and become synchronous; default 600, integer range 30 through 3600. It does not replace the PostgreSQL process-start timeout.
PF_HF_MAX_STORAGE_BYTES	Deployment lifetime byte-admission ceiling, e.g. 10737418240 for 10 GiB. Includes retained encrypted persistence.
PF_HF_WAL_RESERVE_BYTES	Capacity reserved for essential WAL/base backups, e.g. 268435456. Must be positive and below the total ceiling.
PF_HF_MAX_STATE_BYTES	Bound per private state scope, e.g. 2147483648.
PF_HF_MAX_TRANSFER_FILES	Maximum transfer/state inventory entries, e.g. 20000.
PF_SAAS_PUBLIC_URL	Direct application origin, e.g. https://YOUR-ACTUAL-SUBDOMAIN.hf.space; copy the actual value from HF.
PF_SAAS_DEPLOYMENT_BUDGET_ENABLED	Required: true.
PF_SAAS_DEPLOYMENT_MONTHLY_COMPUTE_SECONDS	Finite shared monthly compute-admission ceiling, e.g. 3600 for an initial trial; committed charges include active reservations. Normal jobs reserve their timeout plus a 60-second launch/stop margin. See queued-run troubleshooting before adjusting it.
PF_SAAS_WORKER_SLOTS	Start with 1; integer range 1 through 64. This starts real internal dispatcher threads under the same controller; it does not create another database writer.
PF_SAAS_DEPLOYMENT_MAX_CONCURRENT_JOBS	Start with 1; shared active execution-reservation cap, independent of worker slots. Increase only after load qualification and budget review.
PF_SAAS_USER_MAX_CONCURRENT_JOBS	Start with 1; active scientific/slot-occupying assistant operations per user within an organization. Must be positive when worker slots exceed one.
PF_SAAS_USER_MAX_QUEUED_JOBS	Start with 20; queued scientific jobs per user within an organization. Must be positive when worker slots exceed one.
PF_SAAS_DEPLOYMENT_MAX_QUEUED_JOBS	Shared queued scientific-job ceiling across all native HF tenants, default 10000; set a smaller deliberate trial limit if desired.
PF_SAAS_MAX_CONCURRENT_JOBS	1 initially; per-organization limit, separate from deployment concurrency.
PF_SAAS_API_MAX_INFLIGHT_REQUESTS / API_MAX_INFLIGHT_UPLOADS	Per-process HTTP/upload concurrency; defaults 64 and 2. These protect request handling, not scientific execution capacity.
PF_SAAS_DATABASE_POOL_SIZE / DATABASE_MAX_OVERFLOW	Shared controller SQL pool bounds, defaults 5 and 5; shared assistant pools are budgeted separately.
PF_SAAS_DATABASE_POOL_TIMEOUT_SECONDS / DATABASE_LEASE_POOL_SIZE	Main-pool wait in seconds (5) and independent lease connections (1). Main+overflow+lease+enabled assistant pools must not exceed 25 in this native profile.
PF_SAAS_API_USER_REQUESTS_PER_MINUTE	Authenticated per-user request admission; native HF uses a bounded in-memory counter to avoid a durable SQL write on each GET (initial example 600). This is separate from in-flight concurrency; counters reset on restart.
PF_SAAS_API_RATE_LOCAL_MAX_ACCOUNTS	Native counter bound, default 10000; a full active counter map returns 429 for new accounts. See shared settings (R34_SHARED_CAPACITY_SETTINGS.md).
PF_SAAS_AI_ADMISSION_ENABLED	Keep true for hosted operator/BYOK/autonomous work; configure the shared deployment/organization/provider concurrency, RPM, TPM and estimated-cost bounds from the shared settings guide.

Variable	Value/purpose
PF_SAAS_MAX_JOB_SECONDS	Per-invocation timeout, e.g. 600; allow for CPU-intensive scientific needs.
PF_SAAS_JOB_PREPARE_SECONDS	Optional preparation deadline in seconds; default 300.
PF_SAAS_JOB_TRANSFER_SECONDS	Optional input-transfer deadline in seconds; default 300.
PF_SAAS_JOB_SUBMIT_SECONDS	Optional submission-handling deadline in seconds; default 60.
PF_SAAS_JOB_COLLECT_SECONDS	Optional output-collection deadline in seconds; default 300.
PF_SAAS_JOB_CPUS	2 or less in this native profile.
PF_SAAS_JOB_MEMORY_MB	4096 initially; adjust after qualification, never above 14336 in this profile.
PF_SAAS_ASSISTANT_BYOK_ENABLED	true to allow users' supported AI provider keys.
PF_SAAS_BYOS_ENABLED	true to allow supported user storage destinations.
PF_SAAS_REGISTRATION_ENABLED	Keep false during private qualification; enable only with the configured account/email flow.
PF_SAAS_MAIL_MODE	Required for default local authentication in production: smtp.
PF_SAAS_SMTP_HOST	mail.smtp2go.com for the documented SMTP2GO route.
PF_SAAS_SMTP_PORT	443 for SMTP2GO implicit TLS; do not use blocked 587/465.
PF_SAAS_SMTP_FROM	Your chosen real mailbox at the verified domain, e.g. support@YOUR_VERIFIED_DOMAIN; not the SMTP username.
PF_SAAS_SMTP_STARTTLS	false with the documented SMTP2GO port-443 setup.
PF_SAAS_SMTP_SSL	true with the documented SMTP2GO port-443 setup.
PF_HF_ADMIN_EMAIL	Your first PF administrator's email address; set it for first initialization.
PF_HF_ADMIN_NAME	Your first PF administrator's display name.

Secret	Purpose
PF_HF_TOKEN	Trusted Jobs control-plane token with access to the private code Space and required Bucket mounts. Never forwarded to scientific code.
PF_HF_S3_ACCESS_KEY_ID	Dedicated Bucket S3 access key.
PF_HF_S3_SECRET_ACCESS_KEY	Matching S3 secret.
PF_HF_STATE_KEY	Independent 32-byte key, encoded as 64 hex characters or standard Base64.
PF_HF_PG_ADMIN_PASSWORD	Persistent PostgreSQL administration password, at least 32 printable characters.
PF_HF_PG_APP_PASSWORD	Distinct application database password.
PF_HF_PG_WAL_PASSWORD	Distinct replication-role password.
PF_SAAS_SECRET_KEY	Stable independent SaaS session/signing key.
PF_SAAS_ASSISTANT_CREDENTIALS_KEY	Stable independent BYOK vault key: URL-safe Base64 encoding of 32 bytes.
PF_SAAS_BYOS_CREDENTIALS_KEY	Stable independent BYOS vault key: URL-safe Base64 encoding of 32 bytes.

Secret	Purpose
PF_SAAS_SMTP_USERNAME	Username from SMTP2GO's SMTP Users page.
PF_SAAS_SMTP_PASSWORD	Matching SMTP User password, not an HTTP API key or your PF login password.
PF_HF_ADMIN_PASSWORD	First operator password; remove after successful first login. Existing accounts are never reset/promoted by this path.

The launcher owns production mode, the local PostgreSQL connection and the assistant/BYOS state directories. Do not set an external database URL, hybrid gateway secret or separate worker for this native mode. Existing SaaS settings still govern supported AI providers, endpoint allowlists, organizations and quotas. The defaults above are starting limits, not validated capacity for every workload.

Checkpoint: the six runner values (namespace, image, code repository, revision, hash and flavor) are only part of the configuration. Bucket access, all stable keys/passwords, SMTP and deployment budgets must also be configured. Keep `PF_SAAS_REGISTRATION_ENABLED=false` and the Space Private during validation.

8. Upload the controller and verify startup

Where: Windows CMD. Use the same current output directory from section 4:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\deploy_release.py verify --bundle "%PF_HF_UPLOAD%"
python deploy\saas\huggingface_native\deploy_release.py --system-ca prepare --bundle "%PF_HF_UPLOAD%"
--space "%PF_HF_APP_REPO%" --state-dir "%PF_HF_DEPLOY_STATE%" --fresh
python deploy\saas\huggingface_native\deploy_release.py --system-ca apply --state-dir "%PF_HF_DEPLOY_STATE%"
```

Use `--fresh` only for a newly created empty controller Space. For an existing native deployment, use section 11 without that flag. `prepare` verifies local seals and private code pins without changing HF. `apply` confirms pause, commits against the reviewed repository revision and requests restart. Keep the printed revision and local preparation directory. It does not change configuration, Secrets or durable data. Do not add blanket `--delete` flags.

After apply succeeds, wait for the controller:

BAT | Copy exact lines from the packaged Markdown

```
hf spaces wait "%PF_HF_APP_REPO%" --timeout 25m
python deploy\saas\huggingface_native\deploy_release.py --system-ca status --state-dir
"%PF_HF_DEPLOY_STATE%"
hf spaces logs "%PF_HF_APP_REPO%" --tail 150
```

Require `source_bytes_verified=true`, `running_revision_verified=true`, provider stage `RUNNING` and application startup completion in the runtime logs. A terminal build/runtime error ends the wait early; inspect its logs even when the wait command returns an error. The CLI's `25m` wait is only an observation window, separate from the controller's bounded startup settings.

Where: Browser - Hugging Face.

1. Open the controller's build logs, then its runtime/container logs. A completed Docker build is distinct from successful application startup.
2. The first empty deployment needs `PF_HF_BOOTSTRAP=true`. The supervisor creates PostgreSQL, persists a verified base backup, starts synchronous remote WAL, initializes ledgers, restores private state and then exposes port 7860. Existing remote state always takes the restore path. Failure keeps startup closed.
3. Open the **direct application URL** in its own browser tab. If the HF page's embedded frame is blocked, use the direct URL; do not weaken frame/CSRF/cookie protections. Private Space access still requires HF owner/collaborator access.

4. Log into **PF Simulation** with `PF_HF_ADMIN_EMAIL` and `PF_HF_ADMIN_PASSWORD`. Use this PF administrator password, not an HF token, S3 secret, PostgreSQL password or SMTP password. Existing users are never reset or promoted by bootstrap values.
5. After confirmed successful durable initialization and first login, remove `PF_HF_ADMIN_PASSWORD` and remove `PF_HF_BOOTSTRAP` (absence defaults to `false`). Preserve every other persistent key/password, bucket and prefix. Confirm the resulting restart restores the same application state. Use the account's normal password flow afterward, not bootstrap variables.

If a setting is missing or logs contain a generic `DurabilityError`, use section 10. Do not generate new encryption keys or reset storage to get past recovery.

If HF remains **Starting** with sparse logs, use this **read-only Windows CMD** batch and inspect the latest runtime logs in the browser:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import sys; from huggingface_hub import HfApi;
print(HfApi().get_space_runtime(repo_id=sys.argv[1]).stage)" "%PF_HF_APP_REPO%"
hf buckets info "%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%"
hf buckets list "%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%"
```

These checks do not reveal Secret values or prove recovery has finished. `PF_HF_PG_START_TIMEOUT_SECONDS=600` bounds the PostgreSQL startup/recovery phase, not the total Docker build or full controller initialization. The separate `PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS` wait defaults to `600` seconds and accepts integers from `30` through `3600`. Healthy WAL uploads can continue while PostgreSQL reports `catchup` or `potential`; those states do not yet satisfy the unchanged `streaming`/`sync` readiness checks. The receiver batches contiguous data within a 100-millisecond collection window, up to 4 MiB; remote publication time is additional. See the maintenance note (R34_HF_WAL_CATCHUP_AND_BATCHING.md) for the acknowledgement and startup boundaries. Sparse logs or an elapsed interval alone are not grounds for a force restart or storage reset.

After completing section 9, a SaaS trial for ordinary PF users can use **Protected** Space visibility: source stays private while the app is accessible. Check current HF plan eligibility. PF users then authenticate to PF and retain organization/project permissions; they do not need repository collaborator access or the operator's HF token. Do not change the controller to Public.

9. Qualify the complete deployment before researcher access

Record the release hash, builder manifests, runtime image digest, private code repository and exact commit, payload SHA256, controller Space revision, HF hardware/flavor, package versions, test inputs, Job IDs, timestamps, and observed results. Rebuilds and changes to those identities require qualification again. Required live gates include:

Run destructive fault tests in a **separate disposable validation deployment**, with its own private Bucket/prefix and independently generated keys. Never corrupt or remove WAL, database heads or artifacts, introduce a conflicting controller, or exhaust storage against a deployment holding retained research data. Ordinary restart/persistence checks should use known test records.

1. **Provider capabilities:** public digest image pull; exact-commit private repository mount; rejection of changed/missing payloads; dedicated Job execution; mandatory Landlock ABI 6/seccomp probe; read-only input and scoped output mounts; conditional Bucket create/update semantics and read-after-write visibility. An image download alone does not qualify execution.
2. **Research execution:** real representative relativity and quantum workloads, private Python workflows, active override layers, report rendering/export, literature attach/upload/remove, and autonomous investigation pause/recovery.
3. **Tenant separation:** two ordinary users in different organizations/projects; attempted cross-owner access to jobs, attachments, results, state, credentials, and BYOS destinations must be rejected. Verify execution cannot read controller files, another workload's data, controller secrets, or its Bucket credentials, or signal/trace the runner. Exercise direct sockets, inherited descriptors, symlink/hardlink escapes, subprocesses, and attempted privilege changes. Landlock is not a separate filesystem/PID namespace: some names and metadata may remain observable, so do not certify complete namespace invisibility.

4. **Durability:** acknowledge a known transaction/upload, forcibly replace the disposable controller and its local disk, and verify the acknowledged state, budget, credential references, artifacts, and private continuation checkpoints recover. Exercise WAL corruption, missing objects, and conflicting controllers: each must fail closed rather than serve older state.
5. **Faults and costs:** storage outage, expired token, provider API timeout around submission, cancellation, process/resource exhaustion, Job failure, controller restart while a Job is running, and exhausted global/storage budgets. Confirm no duplicate remote dispatch, false success, or unproven reservation refund. Attempt detached and double-forked children; verify termination of the entire dedicated Job before releasing its reservation. Per-process rlimits are not a delegated cgroup limit over a process tree; VM capacity and provider timeout remain the aggregate boundary.
6. **SaaS access:** secure cookies/CSRF at the canonical HTTPS origin, user account registration or invitations, SMTP, upload sizes, download integrity, BYOK/BYOS, and admission under concurrent users.

Passing local fixtures is not a substitute for these gates. If a gate fails, keep registration and researcher access closed, preserve the failure evidence, and fix/requalify the implementation. The same evidence standard applies to Linux/Docker deployment validation.

10. Troubleshoot the failed checkpoint

Use the matching subsection; do not repeat the entire deployment. Read-only checks can be run together. Review outputs before sharing: paths, computer names, bucket names and application logs can be private. Never share HF tokens, S3 secrets, encryption keys, passwords or an unfiltered environment dump.

Identify the controller's current stage

Where: Windows CMD. Inspect the same startup attempt before taking another action:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import sys; from huggingface_hub import HfApi;
print(HfApi(token=True).get_space_runtime(sys.argv[1]).stage)" "%PF_HF_APP_REPO%"
hf spaces logs "%PF_HF_APP_REPO%" --build --tail 150
hf spaces logs "%PF_HF_APP_REPO%" --tail 150
```

Stage or finding	Action
`BUILDING`	Follow the current build logs; wait for that build to finish.
`BUILD_ERROR`	Locate the first build error. Verify the upload contains the entire current generated `space/` context, including `Dockerfile`, `.dockerignore`, `base-code-manifest.json` and its matching `source/`. The packaged builder includes the required controller build inputs. Correct the input or configuration identified by the error before another build.
`APP_STARTING`	Allow PostgreSQL restore and synchronous receiver catch-up to finish within their configured bounds. A completed Docker build alone does not establish database readiness.
`RUNTIME_ERROR`	Retain the controller commit, timestamps, runtime logs and the configured nonsecret pins. Check the specific failing prerequisite; do not infer a credential failure from a generic `OperationalError` or `RuntimeError`.
`RUNNING` with application startup complete	Proceed to application checks; this is not yet proof of a successful simulation or full service qualification.

For database startup, verify unchanged bucket/prefix and persistent keys, controller access to that storage, storage/WAL capacity, and the two independent startup bounds. This release waits for PostgreSQL readiness and batches WAL catch-up before admitting the application. Set `PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS=600` for the normal receiver startup budget; accepted values are `30-3600`. A longer CLI wait cannot change that internal budget. Increase a bound only after establishing that recovery is making healthy progress. Keep the synchronous readiness requirement intact.

If the cause remains uncertain, pause the failed controller, retain the verified backup, and obtain a review of the sanitized logs and exact release/pins. Do not change durable state to make an unexplained startup failure disappear.

Run stays Queued with no attempt

`attempt_count=0` means the PF worker has not claimed the run. Check admission before resubmitting. The Space being `RUNNING` only establishes application startup; it does not establish that the dispatcher can reserve another run.

The deployment allowance is shared across organizations, simulations and admitted assistant work. An organization's **Usage** page and `hf jobs ps` do not report that ledger. A terminal HF Job list does not prove that all PF reservations have settled. Retained charges can include conservatively accounted interrupted or ambiguous work; do not delete attempts or ledger records to recover allowance.

In the **private configured service environment connected to the actual service database**, an operator can inspect the packaged deployment control status:

BASH | Copy exact lines from the packaged Markdown

```
python -m pf_saas.deployment_control status
```

This is not a command for an unconfigured Windows tools environment. Do not expose the database or copy its credentials into a shared terminal to run it. Read `usage.compute_committed_seconds`, `usage.compute_remaining_seconds`, `usage.monthly_compute_seconds`, `usage.active_reservations` and `unresolved`. Committed seconds already include active reservations: do not add them twice. If operator database access is unavailable, obtain an authorized ledger check; do not infer remaining allowance from tenant usage, elapsed wall time or HF Jobs.

For a normal job with a 600-second timeout, r34 requests **660 deployment reservation seconds**: 600 execution seconds plus the 60-second launch/stop margin. This is an admission reservation, not a claim that 660 seconds of scientific computation occurred or an exact HF invoice charge. Different job limits and assistant workflows can require different reservations.

As a conditional example, if the current UTC month's verified committed total is **5,940 seconds**, the configured ceiling is **5,940**, and the intended next reservation is **660**, a ceiling of **6,600** supplies that one reservation's headroom. These are example values, not release defaults. Check global concurrency, optional monetary admission and organization limits as well; raising the compute ceiling does not resolve those separate blockers.

If the operator approves that additional allowance:

1. Stop new submissions across the deployment. Settle current executions and result collection. Review queued runs and investigations: increasing the ceiling can admit existing work automatically, so do not leave unintended work queued. Confirm provider termination where applicable.
2. Record the current nonsecret settings, immutable controller/runner pins and the verified monthly ledger values. Pause the controller as in section 11.2.
3. In **Browser - HF → controller Settings → Variables**, change only `PF_SAAS_DEPLOYMENT_MONTHLY_COMPUTE_SECONDS` to the deliberately chosen new ceiling. Keep `PF_SAAS_DEPLOYMENT_BUDGET_ENABLED=true`, the concurrency and monetary bounds, and the storage/state settings. Do not change the prefix, reset accounting, or alter the worker lease/durability requirements.
4. If the settings change starts the Space, pause it again. Read back the new ceiling and verify that the other settings and code pins are unchanged. For the example above, the operator record would state `DEPLOYMENT_CEILING_UPDATED=5940->6600` and `ADDITIONAL_ALLOWANCE_SECONDS=660`.
5. Restart once, require `RUNNING` and normal startup logs, then submit or retry only the intended bounded run. Require completion and accessible saved results as in section 11.7. Recheck admission before another run.

The ceiling is a total for the UTC billing month, not a refill balance. Raising it preserves the existing charges. It also permits additional paid provider work; it is not an account-wide HF billing cap. If admission has sufficient headroom and the run remains queued, retain its job details and controller logs for dispatcher investigation instead of repeatedly increasing the ceiling.

Worker lease or result-publication failure

An interrupted attempt and an HF Job's final status describe different layers. Read the PF attempt's phase, timestamps, `execution_started_at`, error and cleanup state along with the matching provider identity. An absent execution timestamp does not establish that scientific execution started. Total wall time includes preparation, transfers and publication in addition to provider execution.

A `PF_WORKER_HEARTBEAT` controller log event identifies a failed worker renewal with bounded timing and exception-category fields. Preserve it with the exact controller and private-code pins. It is evidence for investigation, not proof of a particular storage or database root cause. Keep lease expiry, synchronous WAL acknowledgement and ownership fencing enabled; do not increase unrelated compute or collection limits to hide lost ownership.

For collection or publication errors, use result publication and verification (R34_ARTIFACT_PUBLICATION_REPAIR.md). A complete transfer byte counter or provider `COMPLETED` status is insufficient: PF must commit its final attempt and make the saved results accessible.

Measure the stages of a slow run

Read the authorized job/attempt `timings` record and the controller's structured `PF_EXECUTION_TIMING` log. Older attempts can have no timing record. These measurements use `controller_monotonic_wall` and distinguish preparation, uploading, submission, provider queueing, observed execution and collection. Collection stages distinguish transfer, import, publication and commit when observed. Heartbeat, progress and publication operation aggregates can overlap phases; do not add them to phase totals as additional elapsed time.

The database timing snapshot precedes its own final commit acknowledgement; the final controller log can include that acknowledgement's duration. The downloadable `timings.json` artifact is an earlier snapshot made before artifact publication. `provider_runtime_seconds` is `null` when unavailable. An observed executing phase is not an exact HF Job runtime or billing measurement, and `accounted_compute_seconds` remains PF's separate internal accounting measure. Use the matching provider record for provider runtime and billing. These measurements locate delay without promising a particular completion time.

The workbench adapts automatic polling to activity, page visibility and errors. Optional controller Variables are `PF_SAAS_UI_POLL_ACTIVE_MS` (5000), `PF_SAAS_UI_POLL_IDLE_MS` (30000), `PF_SAAS_UI_POLL_HIDDEN_MS` (60000), `PF_SAAS_UI_POLL_MAX_BACKOFF_MS` (120000) and `PF_SAAS_UI_POLL_QUALIFICATION_MS` (60000). Values are milliseconds; keep active at or below idle, idle at or below hidden, and hidden at or below maximum backoff. Manual refresh and cancellation remain explicit immediate requests. Polling settings change display request frequency, not execution capacity.

Python 3.13 / Python 3.12 / package or PATH mismatch

Where: Windows CMD. The environment name in the prompt is not proof of the interpreter actually running:

BAT | Copy exact lines from the packaged Markdown

```
where python
where hf
python -c "import sys; print(sys.executable); print(sys.version.split()[0])"
python -m pip show huggingface_hub truststore
```

Activate the section-2 environment and repeat the checks. If an old environment was built with Python 3.13, create a different new 3.12 environment; do not overwrite/delete it as a repair. Use `python -m pip` inside the selected environment. If `hf` still resolves elsewhere, invoke the expected executable:

BAT | Copy exact lines from the packaged Markdown

```
call "%PF_HF_TOOLS%\Scripts\activate.bat"
"%PF_HF_TOOLS%\Scripts\hf.exe" auth whoami
```

Installing a package in one environment does not install it into another. A Downloads copy of a helper is not necessarily the helper packaged in this release.

Upload fails with Xet HTTP 403

Where: Windows CMD. First confirm the destination and current context path. For a controller operation, first inspect the saved preparation and exact remote revision. The following read-only check can reconcile an uncertain commit response; it does not repeat the upload:

BAT | Copy exact lines from the packaged Markdown

```
hf auth whoami
dir "%PF_HF_UPLOAD%\space\Dockerfile"
set "HF_HUB_DISABLE_XET=1"
python deploy\saas\huggingface_native\deploy_release.py verify --bundle "%PF_HF_UPLOAD%"
python deploy\saas\huggingface_native\deploy_release.py --system-ca prepare --bundle "%PF_HF_UPLOAD%"
--space "%PF_HF_APP_REPO%" --state-dir "%PF_HF_DEPLOY_STATE%" --fresh
python deploy\saas\huggingface_native\deploy_release.py --system-ca apply --state-dir "%PF_HF_DEPLOY_STATE%"
```

Set `HF_HUB_DISABLE_XET` **before** launching the next CLI process, in the same CMD window. It disables the optional Xet transfer path; it does not bypass access control or guarantee every 403 is resolved. For an uncommitted preparation, resolve the network error and rerun the same `apply`; its parent-revision guard prevents replacing a concurrent commit. For runtime/code uploads, inspect the existing commit before deciding to retry their matching section-4 command with this setting. Do not upload `space/` to the public runtime. HF environment-variable reference.

After the intended upload has succeeded and you have verified its commit, clear the temporary override before later HF operations:

BAT | Copy exact lines from the packaged Markdown

```
set "HF_HUB_DISABLE_XET="
```

If still denied, check that the active token has write access to that exact Space and that the network can reach HF's transfer services. Keep certificate validation enabled. A failed attempt does not prove no commit occurred: inspect **Files → history** and compare its commit with your recorded upload. Align with the current generated folder, without `--delete`.

Registry TLS, authentication or missing Docker instructions

Where: Windows CMD.

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\hf_registry_digest.py --version
python deploy\saas\huggingface_native\hf_registry_digest.py --diagnose --system-ca
python deploy\saas\huggingface_native\hf_registry_digest.py --anonymous --system-ca --image
"%PF_HF_RUNTIME_TAG%"
```

The diagnostic reads no HF token. A registry-root 401 can be a normal anonymous challenge; the selected image's anonymous manifest/digest check must still pass. `CERTIFICATE_VERIFICATION_FAILED` calls for trusted-CA/clock/proxy repair or `--system-ca`, not `verify=False`, insecure curl or disabled TLS checks. If the helper reports that `--system-ca` conflicts with `SSL_CERT_FILE` or `SSL_CERT_DIR`, retain your approved CA configuration and run the helper without `--system-ca`. Do not clear an organization's approved CA settings or disable verification merely to remove the error.

In **Browser - HF**, use the **public runtime** Space's page menu/run-locally instructions after a successful build. HF calls the feature “Run with Docker”, but its position may change. A private runner's registry login can verify metadata yet still fail an HF Job pull. Do not make that private repository public. If the current public image still gives 401, resolve its public access or publish only the reviewed public runtime context to another anonymously pullable OCI registry. The packaged `hf_registry_digest.py` accepts only `registry.hf.space`; for another registry, use that registry's supported method to verify anonymous access to the exact image digest, then run the same `qualify.py` qualification.

Qualification reports RuntimeError or stops after a Job ID

Where: Windows CMD. Copy the already-created Job ID; these commands inspect it and do not submit another paid Job:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_TEST_JOB=REPLACE_WITH_EXISTING_JOB_ID"
hf jobs inspect "%PF_HF_TEST_JOB%" --namespace "%PF_HF_OWNER%" --format json
hf jobs logs "%PF_HF_TEST_JOB%" --namespace "%PF_HF_OWNER%"
```

Use this inspection for the credential-free qualification Job, and review it privately before sharing. JSON avoids the human display truncating pins or mount details. Check the same Job in **Browser - HF → Jobs**. Record terminal status, image, command, revision/mount and confinement output. A generic `RuntimeError` has no proven cause until the identity comparison is examined. Logs showing a passed probe and **COMPLETED** are evidence for that run, not permission to ignore a failed metadata check or open the full SaaS to users. Resolve the error's cause before admission; retain reservations until actual provider termination is confirmed. If no ID was printed, inspect the account's Jobs list before retrying.

The packaged qualifier supports HF's empty-secrets metadata representations (`None`, `{}` and `[]`) while rejecting nonempty or malformed values. Use the qualifier packaged in this release. Keep the public-runtime/private-payload flow, mandatory confinement, empty secrets and read-only code mount intact.

Bucket exists in the browser, but CLI says BucketNotFound / access denied

Where: Windows CMD.

BAT | Copy exact lines from the packaged Markdown

```
hf auth whoami
if defined HF_TOKEN (echo HF_TOKEN override is set) else (echo HF_TOKEN override is not set)
hf buckets info "%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%"
```

Where: Browser - HF. Compare the exact Bucket owner/name with its URL. Then check the **active CLI token's** fine-grained scope: three selected Spaces do not include the state Bucket. Add only the needed Bucket read/write grant using the available resource/scope controls, save, and retry. Matching `whoami` proves account identity, not resource permission.

An `HF_TOKEN` environment variable can override the saved login. If it is an unintended old override, clear it only in this CMD window and select the intended token interactively:

BAT | Copy exact lines from the packaged Markdown

```
set "HF_TOKEN="
hf auth login --force
hf buckets info "%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%"
```

Do not rotate S3 keys to fix a CLI HF-token error. Creating a Bucket after a token does not itself require invalidation. S3 keys inherit their parent token's scope, but do not assume an unverified permission change fixed existing S3 access: test the controller/S3 path separately. If you replace/revoke that parent token, update every affected HF token Secret and generate/update the corresponding S3 pair. Separate valid HF and S3 parent tokens need not be identical. Preserve `PF_HF_STATE_KEY`, the Bucket, prefix and existing objects.

Email sender stays Unverified / email delivery fails

Where: Browser - SMTP2GO and DNS provider. Open the domain's record details, not just the summary row. Confirm all three CNAMEs exist at the authoritative DNS provider, with no doubled domain suffix, wrong record type or replaced hosting record. Read the failed record's tooltip. DNS propagation can take time, but waiting does not correct a missing/wrong record. **Tracking Disabled** is a separate setting, not evidence of the verification failure's cause.

Where: Windows CMD. Replace these placeholders with the three **full** CNAME hostnames from SMTP2GO, then run this read-only DNS check:

BAT | Copy exact lines from the packaged Markdown

```
nslookup -type=CNAME REPLACE_WITH_FULL_RETURN_PATH_HOSTNAME
nslookup -type=CNAME REPLACE_WITH_FULL_DKIM_HOSTNAME
nslookup -type=CNAME REPLACE_WITH_FULL_TRACKING_HOSTNAME
```

Each answer should point to its supplied SMTP2GO target. A missing answer may reflect an unsaved/wrong record, the wrong DNS provider or a cached result; check the provider's saved rows and SMTP2GO's error before deciding to wait.

For delivery, confirm `mail.smtp2go.com`, port `443`, SSL `true`, STARTTLS `false`, a verified From address and the **SMTP User** credentials. Check SMTP2GO's activity report for rejection/quota details. Do not put its HTTP API key in the SMTP password field or switch to development outbox mode in production.

Build succeeds, but startup reports DurabilityError

Where: Browser - controller runtime logs and Settings. A generic `DurabilityError` is not enough to diagnose a single cause. Capture the time and sanitized error context. Check the configured owner, bare Bucket name, prefix, S3 credential pairing/permissions, unchanged state/database keys and available storage/WAL allowance. A successful CLI Bucket lookup tests HF-token visibility, not S3 writes, conditional updates, encryption or PostgreSQL restore.

Do not delete durable heads, WAL, ledgers or objects; change the prefix; generate replacement state/vault keys; force bootstrap; or run a second controller to bypass the failure. An existing deployment must recover its existing authenticated state. Keep researcher access closed until the actual failure is resolved.

App frame blocked, or administrator login fails

Where: Browser. Open the direct `https://...hf.space` origin obtained in section 7; set `PF_SAAS_PUBLIC_URL` to that exact origin. HF private access and PF login are separate. Do not disable frame, cookie or CSRF protections.

Use the PF administrator email/password from first initialization. Changing `PF_HF_ADMIN_PASSWORD` later does not reset an existing account. If that password is lost and SMTP works, use PF's **Forgot password?** flow. HF's masked Secrets cannot recover the old value. Keep stable encryption/session/database keys.

“Virtualization support not detected”

This is a local Windows/Docker startup problem. Treat three prerequisites separately: **CPU virtualization in BIOS/UEFI**, the **Windows WSL/Virtual Machine Platform components**, and **hypervisor startup**. Installing Python or changing HF tokens will not enable any of these.

Where: ordinary Windows CMD. Run these read-only checks. You can paste the complete block; one failed command does not invalidate the other results.

BAT | Copy exact lines from the packaged Markdown

```
ver
systeminfo
wsl --status
wsl --version
wsl --list --verbose
powershell -NoProfile -Command "Get-CimInstance Win32_Processor | Select-Object
Name,VirtualizationFirmwareEnabled,VMMonitorModeExtensions,SecondLevelAddressTranslationExtensions"
```

Also open **Task Manager → Performance → CPU** and inspect **Virtualization**. `systeminfo` may say a hypervisor has already been detected instead of listing each Hyper-V requirement; that is not evidence that virtualization is missing. Processor-property output is supporting evidence, especially on virtual machines, rather than the sole pass/fail test. Docker virtualization troubleshooting; Microsoft processor-property reference.

Finding	Next action
Task Manager shows virtualization disabled	Follow the PC manufacturer's BIOS/UEFI instructions to enable CPU virtualization, commonly named Intel VT-x/Virtualization Technology or AMD-V/SVM. Restart Windows.
WSL is missing or its modern version command is unavailable	Complete or update WSL using the standard setup in section 3, then restart when requested.
Firmware virtualization is enabled, but WSL reports a missing required feature	Check Windows components and hypervisor startup as below.
This Windows PC is itself a virtual machine or managed desktop	Ask its administrator whether nested virtualization is supported and enabled on the host. Guest settings alone cannot expose missing host capabilities. Use the HF-hosted alternative if unavailable.

Where: PowerShell as administrator. Inspect the Windows feature states:

POWERSHELL | Copy exact lines from the packaged Markdown

```
Get-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform
Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
bcdedit /enum | findstr /i hypervisorlaunchtype
```

If WSL is not installed correctly, first use `wsl --install --no-distribution`. For a manual/older component-based WSL installation, the documented component repair is:

POWERSHELL | Copy exact lines from the packaged Markdown

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

Restart Windows after enabling the components; `/norestart` only postpones that restart. Modern packaged WSL can differ from the older optional-component setup, so a disabled legacy WSL component alone is not proof that a working WSL 2 installation is broken. Microsoft manual WSL installation; modern WSL installation options.

If the boot-configuration check explicitly reports `hypervisorlaunchtype Off`, and the computer's administrator permits the change, enable hypervisor startup:

POWERSHELL | Copy exact lines from the packaged Markdown

```
bcdedit /set hypervisorlaunchtype auto
```

Restart Windows, start Docker Desktop, and repeat `docker version`. An empty filtered result is not the same as an explicit `Off`; do not change boot settings merely because that line is absent. Firmware and corporate-managed settings should be handled by the machine's administrator. Microsoft WSL startup troubleshooting; nested virtualization requirements.

Docker login returns a named-pipe HTTP 500

Example of a local Docker engine endpoint:

TEXT | Copy exact lines from the packaged Markdown

```
500 Internal Server Error ... dockerDesktopLinuxEngine/v1.56/auth
check if the server supports the requested API version
```

The named pipe identifies the local Docker Desktop engine connection. This message alone does not establish that HF rejected the token or that API version 1.56 is unsupported. Check the local engine before retrying authentication.

Where: ordinary Windows CMD. Read-only checks:

BAT | Copy exact lines from the packaged Markdown

```
where docker
docker context show
docker context ls
docker version
docker info --format "{{.OSType}}"
if defined DOCKER_API_VERSION (echo DOCKER_API_VERSION is set) else (echo DOCKER_API_VERSION is not set)
if defined DOCKER_HOST (echo DOCKER_HOST is set) else (echo DOCKER_HOST is not set)
if defined DOCKER_CONTEXT (echo DOCKER_CONTEXT is set) else (echo DOCKER_CONTEXT is not set)
```

1. If the **Server** section is missing or errors, open Docker Desktop, inspect its startup error, and resolve the local engine/virtualization problem first. Restart Docker Desktop through its **Troubleshoot** menu when appropriate; do not use factory reset as the first repair.
2. If the selected context targets a different engine, choose the intended Docker Desktop Linux context shown by `docker context ls`. If it is named `desktop-linux`, this one-command check explicitly selects it:

BAT | Copy exact lines from the packaged Markdown

```
docker --context desktop-linux version
```

Context names and environment overrides determine which engine receives a command; do not remove an intentional remote-engine configuration. Docker contexts.

3. If `DOCKER_API_VERSION` was set for earlier troubleshooting, remove that override **in this CMD window** and retry:

BAT | Copy exact lines from the packaged Markdown

```
set "DOCKER_API_VERSION="
docker version
```

Docker normally negotiates a compatible API version. Do not guess an older API version merely because `/v1.56/` appears in the error. Use the Client/Server output to investigate a genuine version mismatch, and update Docker Desktop through its normal update mechanism if necessary. Docker Engine API negotiation.

4. Once Client and Server respond normally, retry `docker login registry.hf.space` interactively. Enter your personal HF username and token at its prompts; do not substitute an organization owner name or put the token in the command.
5. If Docker remains unavailable, use the packaged `deploy\saas\huggingface_native\hf_registry_digest.py` and the HF-hosted build path. A registry-metadata result establishes only metadata access; the subsequent HF Job test still establishes whether HF can pull and execute the image.

Collecting diagnostics: review system, context and log output before sharing it: computer/domain names or network details may be private. Use the formatted Docker check above rather than an unrestricted configuration/environment dump. Never share HF tokens, S3 secrets, passwords or PF encryption keys.

11. Upgrade an existing r34 deployment

Use this route when an earlier r34 release is already deployed and running. It upgrades the existing private controller and runner while retaining the existing application data. It does not repeat first-time resource creation, email setup, secret generation or database initialization. The complete release contains the approved native response/durability corrections and the shared application improvements. Historical patch installers are not required.

Keep the same state Bucket, deployment prefix, database passwords, state key, session key and BYOK/BYOS vault keys. Preserve `PF_HF_MAX_STORAGE_BYTES` at the already approved value: for the deployment raised to 32 GiB this is `34359738368`. Keep the encrypted quota counter, reservation IDs and audit history. Conservative lifetime admission remains the default; only explicit safe-lifecycle policy and verified reconciliation can reclaim eligible charges. Keep the existing controller's valid HF and Bucket S3 credentials. **This upgrade and its HF-to-HF backup require no new S3 configuration.** Preserve the old release, its generated contexts, the old controller

revision and its qualified code/image pins for recovery review. The new full archive and the newly generated private-code archive have distinct SHA256 values; neither is the runner's Git commit or the runtime image digest.

11.1 Prepare the new release and record the running deployment

Where: Windows CMD / File Explorer. Compare the downloaded full archive against its companion checksum, then extract `.xz` and `.zip` into a **fresh** project directory as in section 2. Retain the earlier extraction separately.

BAT | Copy exact lines from the packaged Markdown

```
certutil -hashfile "FULL_PATH_TO_DOWNLOADED_RELEASE_ARCHIVE" SHA256
```

Open the already configured Python 3.12 tools environment. Set the following values for this upgrade. Replace every `REPLACE_...` value and use the actual existing resource names if they differ from these examples:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_PROJECT_ROOT=D:\pf-r34-upgrade\privileged-frame-simulation"
set "PF_HF_TOOLS=%USERPROFILE%\pf-hf-tools"
set "PF_HF_OWNER=YOUR_HF_USERNAME"
set "PF_HF_APP_REPO=%PF_HF_OWNER%/pf-simulation-saas"
set "PF_HF_CODE_REPO=%PF_HF_OWNER%/pf-simulation-runner"
set "PF_HF_RUNTIME_REPO=%PF_HF_OWNER%/pf-runtime-r34"
set "PF_HF_BUCKET_NAME=pf-simulation-state"
set "PF_HF_BACKUP_BUCKET_NAME=pf-simulation-state-backup-REPLACE_WITH_UNIQUE_SUFFIX"
set "PF_HF_UPLOAD=%USERPROFILE%\pf-hf-native-upload-r34-REPLACE_WITH_UNIQUE_SUFFIX"
set "PF_HF_DEPLOY_STATE=%USERPROFILE%\pf-hf-native-deploy-r34-REPLACE_WITH_UNIQUE_SUFFIX"
call "%PF_HF_TOOLS%\Scripts\activate.bat"
cd /d "%PF_PROJECT_ROOT%"
where python
where hf
python -c "import sys,huggingface_hub; print(sys.executable); print(sys.version.split()[0]);
print(huggingface_hub.__version__)"
```

Use Python 3.12 and the guide's Hub SDK version, `1.32.0`, with `truststore`. If the existing environment needs preparation, use section 2; do not create a new virtual environment over it. `PF_HF_UPLOAD` names a new generated-context directory. These CMD variables do not update the remote Space.

Where: Browser - HF → controller Settings. Record the existing Variables, resource visibility/hardware, and secure recovery references for its Secrets. Specifically retain `PF_HF_JOB_IMAGE`, `PF_HF_ASSISTANT_IMAGE` if set, `PF_HF_CODE_REPO`, `PF_HF_CODE_REVISION`, `PF_HF_CODE_SHA256`, job flavor, namespace, Bucket and prefix, startup bounds and budget settings. HF cannot show Secret values back to you; use the original secure recovery record. Record the controller's current commit from **Files → history**. Keep these old identities separate from the new values recorded later.

11.2 Stop new work, finish Jobs and pause the controller

Where: PF application and Browser - HF. Arrange a maintenance window and stop users, API clients and scheduled automation from submitting work. Keep registration disabled. Restrict researcher access for the upgrade using the deployment's existing access controls; when necessary, use Private Space visibility and coordinate with HF collaborators who retain access.

Let current PF simulations, private workflows, assistant work and autonomous investigations finish and collect their results, or cancel them through PF and verify cancellation in HF Jobs. Check the HF Jobs dashboard for any remaining deployment-owned Jobs, including previously submitted qualification Jobs. A cancellation request alone is not proof that a Job has stopped. Do not pause until active work and its result collection have settled.

Where: Windows CMD. With new submissions still stopped:

BAT | Copy exact lines from the packaged Markdown

```
hf spaces pause "%PF_HF_APP_REPO%"
python -c "import sys; from huggingface_hub import HfApi;
```

BAT (CONTINUED) | Copy exact lines from the packaged Markdown

```
s=HfApi(token=True).get_space_runtime(sys.argv[1]).stage; print(s); raise SystemExit(0 if s=='PAUSED' else
1)" "%PF_HF_APP_REPO%"
```

Require `PAUSED`. Pausing the controller does **not** cancel separately running HF Jobs. Confirm that no other controller or writer uses this deployment's Bucket/prefix. Keep the controller paused through backup, code qualification and configuration. After any HF settings change that starts the Space, pause it again before proceeding.

11.3 Create and verify a private HF backup

Where: Browser - HF. Open https://huggingface.co/new-bucket, signed in as the operator for the deployment owner. Create a new **Private** Bucket in the **same storage region** as the state Bucket. The storage region must match; a prewarming region is a different setting. Use the exact unique name chosen in `PF_HF_BACKUP_BUCKET_NAME`.

For example, `pf-simulation-state-backup-20261001` is an arbitrary name for one maintenance window, not a required name or a Bucket to reuse for every upgrade. Choose a new suffix for each backup. The destination must be empty before copying. Keep the original state Bucket selected in the controller settings; the backup is a separate recovery copy.

At https://huggingface.co/settings/tokens, select an operator token with its existing Space upload/management and Jobs permissions plus these Bucket grants:

Resource	Required access for this backup
Existing state Bucket	Read
New backup Bucket	Read and write

Preserve any existing permissions the operator token needs for other deployment actions. The controller's existing storage credentials are separate and stay unchanged. Bucket grants are independent of Space grants; account identity alone does not establish access to either Bucket.

Where: Windows CMD. Select that token explicitly for this maintenance session. Clearing `HF_TOKEN` here removes the environment override of the saved login. Enter the token only at the private login prompt:

BAT | Copy exact lines from the packaged Markdown

```
set "HF_TOKEN="
hf auth login --force
hf auth whoami
hf buckets info "%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%"
hf buckets info "%PF_HF_OWNER%/%PF_HF_BACKUP_BUCKET_NAME%"
python -c "import os; from huggingface_hub import HfApi; a=HfApi(token=True);
s=os.environ['PF_HF_OWNER']+'/'+os.environ['PF_HF_BUCKET_NAME'];
d=os.environ['PF_HF_OWNER']+'/'+os.environ['PF_HF_BACKUP_BUCKET_NAME']; assert s!=d, 'Source and backup must
 differ'; assert a.get_space_runtime(os.environ['PF_HF_APP_REPO']).stage=='PAUSED', 'Controller must be
paused'; assert a.bucket_info(d).private is True, 'Backup must be private'; assert not
list(a.list_bucket_tree(d,recursive=True)), 'Use a new empty backup Bucket';
print('BACKUP_DESTINATION_READY')"
```

Require `BACKUP_DESTINATION_READY`, then copy entirely within HF:

BAT | Copy exact lines from the packaged Markdown

```
hf cp "hf://buckets/%PF_HF_OWNER%/%PF_HF_BUCKET_NAME%/"
"hf://buckets/%PF_HF_OWNER%/%PF_HF_BACKUP_BUCKET_NAME%/"
```

After `Copied`, verify file paths, sizes and Xet hashes. This reads the source, the backup and the source again, checks private visibility and confirms that the controller was paused before and after the comparison:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import os; from huggingface_hub import HfApi,BucketFile; a=HfApi(token=True);
s=os.environ['PF_HF_OWNER']+'/'+os.environ['PF_HF_BUCKET_NAME'];
d=os.environ['PF_HF_OWNER']+'/'+os.environ['PF_HF_BACKUP_BUCKET_NAME']; c=os.environ['PF_HF_APP_REPO'];
read=lambda b:{f.path:(f.size,f.xet_hash) for f in a.list_bucket_tree(b,recursive=True) if
isinstance(f,BucketFile)}; p=a.get_space_runtime(c).stage=='PAUSED'; x=read(s); y=read(d); z=read(s);
ok=s!=d and p and a.get_space_runtime(c).stage=='PAUSED' and a.bucket_info(d).private is True and bool(x)
and all(v[1] for v in x.values()) and x==y==z; print('BACKUP MANIFEST VERIFIED' if ok else 'BACKUP NOT
VERIFIED'); print('Source files:',len(x),'Backup files:',len(y),'Bytes:',sum(v[0] for v in x.values()));
raise SystemExit(0 if ok else 1)"
```

Require `BACKUP MANIFEST VERIFIED`. Save the source/destination names, maintenance time, counts and command result with the upgrade record. Counts and bytes depend on your deployment; do not require another deployment's numbers. This establishes an unchanged matching copy manifest. It does not constitute a restore drill or independently prove decryption/application recovery. If it does not verify, keep the controller paused and resolve the backup before changing deployed code.

11.4 Generate matching contexts and choose the runtime

Where: Windows CMD, new project root. Build once into the new output folder:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\build_space.py --output "%PF_HF_UPLOAD%"
python -c "import hashlib,json,sys; from pathlib import Path; p=Path(sys.argv[1]);
a=json.loads((p/'jobs'/'code-manifest.json').read_text(encoding='utf-8'));
b=json.loads((p/'space'/'base-code-manifest.json').read_text(encoding='utf-8'));
h=hashlib.sha256((p/'jobs'/'pf-code.zip').read_bytes()).hexdigest(); assert a['sha256']==b['sha256']==h,
'Generated contexts do not match'; print('VERIFIED_CODE_SHA256='+h)" "%PF_HF_UPLOAD%"
```

Record the new `VERIFIED_CODE_SHA256`. Use this same `jobs/` and `space/` pair through the rest of the upgrade. The complete release builds its required controller inputs directly; no separately downloaded repair or diagnostic wrapper is needed.

Earlier r34 builds added the supported FMU and QuTiP runtime dependencies. Use the current generated context as the compatibility reference rather than assuming any r34 runtime image matches. Qualification must report both `confinement_probe_passed=true` and `scientific_runtime_probe_passed=true`. The standalone edition retains its existing optional-dependency choices.

The public runtime can be reused when its generated context is unchanged and the recorded digest belongs to the previously qualified deployment. Compare the new `runtime.zip` with the retained `runtime.zip` used to build that image:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_PREVIOUS_UPLOAD=FULL_PATH_TO_RETAINED_PREVIOUS_CONTEXTS"
python -c "import hashlib,sys; from pathlib import Path; digest=lambda
p:hashlib.sha256(Path(p).read_bytes()).hexdigest(); a=digest(Path(sys.argv[1])/'runtime.zip');
b=digest(Path(sys.argv[2])/'runtime.zip'); assert a==b, 'Runtime contexts differ; build and qualify the
current runtime'; print('RUNTIME_CONTEXT_UNCHANGED='+b)" "%PF_HF_PREVIOUS_UPLOAD%" "%PF_HF_UPLOAD%"
```

When that comparison passes and the retained context is tied to the old image in your deployment record, use its existing pinned `PF_HF_JOB_IMAGE` digest. The new code still needs a new qualification Job. If the runtime differs, the reference context is unavailable, or its image provenance is uncertain, follow sections 4-5 to upload only the current `runtime/`, wait for its build, verify anonymous registry access and obtain its immutable digest. Keep both private Spaces private. Never use a `:latest` tag as the production pin.

11.5 Upload the private payload and qualify its exact identity

Where: Windows CMD. Upload the generated private runner context:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\deploy_release.py --system-ca upload-code --bundle "%PF_HF_UPLOAD%"
--repo "%PF_HF_CODE_REPO%"
```

Save the helper's `code_revision` and `code_sha256`; these identify the exact verified upload. The helper uses a guarded parent commit and checks the remote file inventory. Keep concurrent operator changes out of this maintenance window. Set the new commit, the code hash from section 11.4 and the selected runtime digest. Do not reuse pins from another full archive or an earlier qualification just because both releases are named r34:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_CODE_REVISION=REPLACE_WITH_NEW_40_HEX_PRIVATE_COMMIT"
set "PF_HF_CODE_SHA256=REPLACE_WITH_NEW_64_HEX_PAYLOAD_SHA256"
set "PF_HF_JOB_IMAGE=REPLACE_WITH_FULL_REGISTRY_IMAGE_AT_SHA256_DIGEST"
python deploy\saas\huggingface_native\qualify.py --namespace "%PF_HF_OWNER%" --runtime-image
"%PF_HF_JOB_IMAGE%" --code-repo "%PF_HF_CODE_REPO%" --code-revision "%PF_HF_CODE_REVISION%" --code-sha256
"%PF_HF_CODE_SHA256%" --flavor cpu-basic --system-ca
```

Review this **dry run**: exact image, repository, revision and hash; no secrets, no exposed service/SSH, and the read-only `/pf-code` mount at that revision. The following command deliberately submits **one paid qualification Job**:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\qualify.py --namespace "%PF_HF_OWNER%" --runtime-image
"%PF_HF_JOB_IMAGE%" --code-repo "%PF_HF_CODE_REPO%" --code-revision "%PF_HF_CODE_REVISION%" --code-sha256
"%PF_HF_CODE_SHA256%" --flavor cpu-basic --system-ca --submit
```

Use the actual intended supported flavor instead of `cpu-basic` if needed, in both commands, and retain the qualified flavor. Require provider stage `COMPLETED`, `self_test_completed: true` and `confinement_probe_passed: true`. Record the Job ID, URL and result. `full_saas_live_qualified: false` correctly means application acceptance is still outstanding.

If a Job ID is printed before an uncertain helper response, inspect that same Job; do not blindly submit another paid Job:

BAT | Copy exact lines from the packaged Markdown

```
set "PF_HF_TEST_JOB=REPLACE_WITH_ALREADY_SUBMITTED_JOB_ID"
hf jobs inspect "%PF_HF_TEST_JOB%" --namespace "%PF_HF_OWNER%" --format json
hf jobs logs "%PF_HF_TEST_JOB%" --namespace "%PF_HF_OWNER%"
```

Compare its immutable identity and confinement evidence as in section 10. If no Job ID was returned, check the HF Jobs dashboard before retrying. Resolve any failed qualification check before bringing the upgraded controller online.

11.6 Set remote pins, upload the matching controller and restart

Where: Browser - HF → controller Settings → Variables and secrets. Keep all existing storage, key, SMTP, origin and budget configuration. Remove `PF_HF_BOOTSTRAP` if it is present, whether as a Variable or a Secret; absence defaults to `false`. This existing deployment must restore its durable state. Do not restore first-time administrator password settings.

Set these **remote Space Variables**, using the actual values just qualified:

Variable	Value
`PF_HF_CODE_REPO`	Current private runner repository.
`PF_HF_CODE_REVISION`	New immutable upload commit from section 11.5.

Variable	Value
PF_HF_CODE_SHA256	New verified private-code SHA256 from section 11.4.
PF_HF_JOB_IMAGE	Qualified immutable runtime digest, retained or newly built.
PF_HF_JOB_FLAVOR	Flavor used for the qualification.
PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS	600; separate from the PostgreSQL startup timeout.

If PF_HF_ASSISTANT_IMAGE is explicitly configured to a different runtime, qualify that image against the **new code commit and SHA256** before resuming those workflows, even when the image digest itself is unchanged. Repeat section 11.5's dry run and paid self-test with that image as --runtime-image, keeping the same new code pins, and record its result separately. Preserve the main PF_HF_JOB_IMAGE pin. Local CMD set commands alone do not change remote Variables. Check the saved values from CMD:

BAT | Copy exact lines from the packaged Markdown

```
python -c "import os; from huggingface_hub import HfApi; a=HfApi(token=True);
v=a.get_space_variables(os.environ['PF_HF_APP_REPO']);
keys=('PF_HF_CODE_REPO','PF_HF_CODE_REVISION','PF_HF_CODE_SHA256','PF_HF_JOB_IMAGE'); assert all(k in v and
v[k].value==os.environ[k] for k in keys), 'Controller variables do not match the qualified pins'; assert
v.get('PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS') is not None and
v['PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS'].value=='600', 'Unexpected receiver startup deadline'; assert
'PF_HF_BOOTSTRAP' not in v, 'Remove bootstrap from Variables'; print('CONTROLLER_CODE_PINS_VERIFIED')"
```

Require CONTROLLER_CODE_PINS_VERIFIED; also confirm bootstrap is absent from Secrets in the HF settings. Verify the flavor matches the qualification. If changing settings restarted the controller, pause it again. Upload only the matching current space/ context, then restart after the upload succeeds:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_native\deploy_release.py verify --bundle "%PF_HF_UPLOAD%"
python deploy\saas\huggingface_native\deploy_release.py --system-ca prepare --bundle "%PF_HF_UPLOAD%"
--space "%PF_HF_APP_REPO%" --state-dir "%PF_HF_DEPLOY_STATE%"
python deploy\saas\huggingface_native\deploy_release.py --system-ca apply --state-dir "%PF_HF_DEPLOY_STATE%"
hf spaces wait "%PF_HF_APP_REPO%" --timeout 25m
python deploy\saas\huggingface_native\deploy_release.py --system-ca status --state-dir
"%PF_HF_DEPLOY_STATE%"
hf spaces logs "%PF_HF_APP_REPO%" --tail 150
```

Run these individually. prepare is read-only and seals the current revision, new code pins, configuration fingerprint and exact generated files in a new local state directory. It schedules deletion of obsolete release-owned source and known historical registry extras; unexpected unrelated files stop the preparation. apply confirms pause, checks those pins again, submits one guarded commit and requests restart. It never edits Space Variables, Secrets, Buckets or quota data. If a write response is uncertain, run status first; do not repeat an upload. A reconciled commit that remains PAUSED needs only the ordinary hf spaces restart command, followed by wait and status. Require source_bytes_verified=true and running_revision_verified=true. HTTP and performance acceptance remain separate browser checks.

Save the controller commit. The native startup restores durable state and runs the packaged application migrations; do not run a local Windows migration against the hosted database. Require RUNNING plus Application startup complete in the runtime logs. If the wait ends in an error, still retrieve the logs and follow section 10; do not repeatedly restart or change database state. Build logs are available separately:

BAT | Copy exact lines from the packaged Markdown

```
hf spaces logs "%PF_HF_APP_REPO%" --build --tail 150
```

11.7 Verify retained data and execution before reopening access

See result publication and verification (R34_ARTIFACT_PUBLICATION_REPAIR.md) for the separate transfer, import and publication counters, safe failure codes and saved-result acceptance checks.

Keep the maintenance window in place while checking the direct application URL:

1. Sign in with an **existing PF account**. Confirm its organizations, projects, retained simulation records, attachments and results. Confirm that expected BYOK/BYOS settings remain available without exposing credential values.
2. Submit one small representative simulation. Confirm one expected HF Job, completed result collection, and the expected plots/report/downloads in PF.
3. Run a separate small cancellation test. Cancel through PF, confirm provider termination and the application's final cancellation state, and check that it can admit the next permitted workload.
4. Verify normal durable writes and persistence using known test records. Schedule any destructive restore/fault qualification in a separate disposable validation deployment, as described in section 9.
5. Record outcomes against this full release checksum, new code hash, runner commit, runtime digest, controller commit, variables and Job IDs. Perform the remaining applicable service/authorization checks from section 9 before treating the complete deployment as qualified.

`RUNNING` and the confinement self-test are separate checkpoints; neither proves that these application checks passed. Reopen the previous intended researcher access only after the required checks succeed. Retain the private backup and old deployment records for your recovery/retention policy; do not automatically delete them on the first successful startup.

11.8 If the upgrade cannot be accepted

Stop new submissions and settle or cancel any test Jobs, then pause the controller. Preserve the verified pre-upgrade backup, current state Bucket, keys, old/new deployment identities and failure evidence. A controller startup can claim durable ownership and publish recovery/WAL state even before the UI opens, so failure does not establish that remote state was untouched.

Review compatibility before selecting rollback. Reverting only a code pin is not a complete rollback: the controller source, private payload/revision/hash, runtime and configuration must form a known matching qualified deployment. An older controller is not automatically compatible with state written by a newer one. If recovery requires the pre-upgrade Bucket copy, use a reviewed restore procedure and explicitly account for changes made after that copy. Do not copy it over the active Bucket, enable bootstrap, change keys/prefix, or start a second controller as an improvised rollback. Resume service only after restore and application acceptance have been established.

Deployment-wide cost admission and storage maintenance

Compute admission is **deployment-wide**, in addition to existing user/org quotas. Set finite compute seconds, concurrency, runtime, input/output, and storage bounds before enabling researcher access. The optional monetary estimate uses integer micro-USD settings:

- `PF_SAAS_DEPLOYMENT_BUDGET_MONTHLY_MICRO_USD`
- `PF_SAAS_DEPLOYMENT_BUDGET_FIXED_MICRO_USD`
- `PF_SAAS_DEPLOYMENT_BUDGET_SAFETY_MICRO_USD`
- `PF_SAAS_DEPLOYMENT_BUDGET_HOURLY_MICRO_USD`

Enter current provider rates for the selected hardware and a conservative allowance for fixed costs. Reserving a Job includes bounded startup time. An ambiguous dispatch is not retried blindly, and its reservation is not released until provider termination can be established. A timed-out local HTTP request does not prove that a remote Job stopped.

These controls stop **PF-controlled new work**. They are not a contractual cap on the HF invoice: subscriptions, Space uptime, storage, API operations, provider billing granularity, other account activity, and stale/orphaned

resources can still incur charges. User BYOK and BYOS services have their own billing controls. Review HF billing and provider-level limits directly; do not describe the PF estimate as an absolute maximum bill.

The storage meter conservatively retains reservations after ambiguous writes and deletions. It does not automatically recycle capacity or delete backup/WAL history. Near exhaustion, stop admitting work and conduct verified maintenance before the reserved WAL capacity is exhausted. Do not reset/delete the meter, change the prefix, or increase concurrency merely to bypass a stopped service.

To review or increase the byte ceiling deliberately, first stop the controller and establish that provider Jobs are stopped. In a private operator environment with the same bucket/prefix and credentials (not an ordinary unconfigured Windows shell), use:

BASH | Copy exact lines from the packaged Markdown

```
python -m pf_hf_native.quota inspect
python -m pf_hf_native.quota raise-limit --new-limit-bytes NEW_CEILING --reason "Reviewed storage capacity
increase" --confirm-supervisor-stopped
```

Use the existing ceiling in `PF_HF_MAX_STORAGE_BYTES` for this maintenance action; after its successful conditional update, set the new ceiling in the Space and restart. The operation retains usage/reservations and an audit record; it is an explicit operator cost decision, not a reset. Keep the WAL reserve unchanged unless a separately reviewed migration supports changing it.

Increasing native capacity without replicating the controller

Concurrent users are not the same as simultaneous simulation runs. Many researchers can be signed in while only a bounded subset of jobs execute; queued jobs also consume admission/storage reservations. Configure all three rather than treating a user count as a worker count.

Start with one slot and one active run. After the existing single-run, cancellation, recovery and publication checks pass, a small two-run trial can use:

Space Variable	Initial one-run deployment	Two-run qualification trial
`PF_SAAS_WORKER_SLOTS`	1	2
`PF_SAAS_USER_MAX_CONCURRENT_JOBS`	1	1
`PF_SAAS_USER_MAX_QUEUED_JOBS`	20	20
`PF_SAAS_MAX_CONCURRENT_JOBS`	1	2
`PF_SAAS_DEPLOYMENT_MAX_CONCURRENT_JOBS`	1	2

This example allows two different users in the same organization to run concurrently, subject to current compute/storage reservations and provider capacity. A single user remains at one active run; increase that user's applicable cap deliberately if the trial specifically needs two runs by the same user. Raising these limits does not replenish consumed monthly allowance, storage admission or provider credits.

Set the Variables on the existing controller Space, retain `PF_SAAS_DEPLOYMENT_PROFILE=single`, and use a controlled restart after draining existing work. Do not create another controller or point an ordinary external worker at the native database. Submit three bounded jobs across authorized users: verify two actual provider executions overlap, the third remains queued, and it starts only after a slot is released. Then exercise cancellation, a controller interruption and concurrent downloads/private-state mutations; inspect attempt IDs, tenant isolation, lease gaps, immutable manifests and charged reservations. Each live trial consumes provider resources.

Increase to the next measured level only after recording response latency, queue delay, provider startup/runtime, upload/commit latency, controller memory/CPU, PostgreSQL connection/lock pressure and

durable WAL latency. Native HTTP request admission and the isolated lease connection pool protect different resources. Start with the default SQL pools; increase lease capacity only within the profile's combined pool ceiling and after measuring commit delay. HF API/dispatcher threads share one main database client. Optional assistant stores and maintenance commands use additional connections; PostgreSQL remains capped at 60 connections. Keep one maintenance action at a time.

When API, database, durable-writer or provider limits require independently scalable tiers, use the distributed deployment guide (../DISTRIBUTED_SAAS_DEPLOYMENT.md). That migration adds shared PostgreSQL/object storage, independent API replicas and dedicated worker hosts. It preserves the same user, organization and deployment admission concepts, while changing the infrastructure. The capacity qualification guide (../deploy/saas/distributed/CAPACITY_QUALIFICATION.md) separates authenticated-user load measurements from scientific throughput. Neither a 64-slot setting nor a large distributed ceiling certifies hundreds of thousands of concurrent users or simulations.

Architecture and execution limits

This native HF profile remains one controller, with configurable internal dispatcher capacity. `PF_SAAS_WORKER_SLOTS` starts that many worker threads, each responsible for at most one scientific attempt at a time. The default is one and the software ceiling is 64; this is a bound, not a qualified load level. Per-user, organization and deployment reservations still decide which work can start. Increasing a quota without available slots and provider capacity does not create additional execution throughput. For separate API replicas and multiple Linux/Docker worker hosts sharing PostgreSQL and object storage, use DISTRIBUTED_SAAS_DEPLOYMENT.md (../DISTRIBUTED_SAAS_DEPLOYMENT.md). That profile provides configurable capacity and admission controls; neither profile claims qualification for hundreds of thousands of concurrent simulations. Establish capacity, provider quotas, cost controls and failure recovery for the actual deployment before admitting that workload.

Component	Location and responsibility
Trusted controller	One Docker Space runs the PF SaaS API, the configured pool of dispatcher threads, and one native supervisor/durable owner.
Database	PostgreSQL 15 on ordinary local Space disk; encrypted base backups and synchronous WAL are persisted directly through the HF Bucket S3 API.
Scientific execution	A separate HF Job VM for each invocation, with a public dependency image pinned by digest, a private code payload pinned by commit and SHA256, and mandatory Landlock/seccomp confinement.
Persistent artifacts	A private HF Bucket; application metadata references encrypted objects.
Private assistant/BYOS state	Restored before the API starts and persisted through the native state adapter.
Credentials	Space Secrets; user BYOK/BYOS credentials keep the existing owner binding and vault encryption. Jobs receive no controller credentials.

Space disk is a working copy. PostgreSQL does **not** use `hf-mount` or a bucket mount as its data directory. An acknowledged transaction depends on the remote WAL receiver durably publishing its encrypted bytes and advancing a conditional head. Loss of bucket access, a conflicting controller, or exhausted WAL capacity closes admission; there is no asynchronous fallback. Crash recovery starts from the remote base backup and authenticated WAL before exposing port 7860.

Only one controller is supported. Do not enable replicas or start a second ordinary `run_pf_saas.py worker` against the native database. A replacement controller fences the previous owner. The public API and all worker threads share the same process, service database pools and ownership guard.

The default public runtime image contains the scientific and reporting dependencies from the existing SaaS route and a generic checksum-verifying bootstrap. It contains no PF source, credentials, user data, or private workflows. PF code is delivered separately from a private repository at an exact commit and checked before use. Optional licensed/external research toolchains still require their applicable software, configuration, and rights; HF

hosting does not supply MATLAB licenses, external hardware, or arbitrary remote services. Custom runtime images must preserve the security contract and be requalified. A public runtime image must contain only material that you have reviewed and may redistribute.

Execution boundary and supported behavior

The HF runner uses one dedicated Job/VM per invocation and never pools mutually untrusted researchers in one Job. Its unprivileged supervisor checks Landlock ABI 6 or later, seccomp, `no_new_privs`, process cleanup, and the effective access restrictions before executing a submitted program. Missing support or a failed probe blocks execution. This profile does not require Bubblewrap, a Docker socket, root, nested namespaces, mount privileges, or the experimental HF Sandbox API.

Within the Job, the submitted program can read the prepared `/source`, workload helper files in `/trusted`, and explicitly allowed runtime dependencies, and write its assigned work tree. The active bootstrap/runner source at `/opt/pf/source` is not readable by the workload. The dedicated Job's initially empty `/dev/shm` is writable for POSIX semaphores and shared-memory IPC; its capacity is bounded by the VM's memory. Landlock controls access to content; it does not make this a separate filesystem or PID namespace, and some path/process metadata may remain visible. Metadata-changing operations such as `chmod`, `utime`, and extended-attribute writes are blocked globally by seccomp because Landlock does not mediate all those operations, including on the work tree. Controller secrets and other tenants' inputs are not mounted into the Job. Landlock signal/abstract-socket scoping and seccomp protect the supervisor from the workload even though both use the image's unprivileged UID.

Network sockets are blocked for untrusted programs, including local listeners and direct external API calls. Anonymous Unix socket pairs remain available for in-workload communication. Existing trusted BYOK/BYOS orchestration remains in the controller; do not place those credentials in the Job. Programs requiring a local server, `multiprocessing.Manager`, or direct network access may need an approved compatible workflow design. Ordinary multiprocessing pools, threads, subprocesses, and new sessions/process groups are supported by the confinement design. The supervisor acts as a subreaper and terminates/reaps detached descendants before output collection. Test the actual scientific libraries and external-tool adapters before promising support for a particular workflow.

Resource limits bound each process; the selected VM and provider timeout bound the complete Job. Process-count limits are per UID, and output-directory checks are admission/collection checks, not hard disk quotas or delegated cgroup limits. The runner closes inherited descriptors, removes untrusted environment entries, and reaps its children before output collection. The controller retains capacity and budget reservations until the provider confirms the whole Job has terminated. A passing probe is enforcement evidence, not a security certification or signed attestation. The broader live checks in section 9 remain mandatory.

Administrator-approved research profiles can select additional immutable images for optional scientific toolchains. Keep operator license files and service keys out of images used for arbitrary private code; preserve the existing profile policy and qualify each selected image. A license requiring unrestricted outbound network access is not compatible with the native untrusted-workload network policy.

The application Space and Jobs can use different hardware. Choose Space memory and local disk sufficient for PostgreSQL, private state restore, API requests, and temporary transfers. Free/sleeping hardware is not a guarantee of continuous availability. Avoid scheduled sleep during active investigations.

The initial native recovery implementation bounds the extracted database backup at 8 GiB and stages compressed backup/WAL data in memory. Qualify the chosen Space's RAM, disk, and recovery time against your retained data volume before increasing storage allowances; this profile does not establish arbitrary database scale. A larger storage ceiling alone does not raise the restore bound.

Operation, restart, and recovery

- Start/restart only the native application Space. Do not manually start an additional controller or change PostgreSQL to asynchronous commit.
- The supervisor closes API admission if its worker, WAL receiver, or ownership guard fails. Provider Jobs may outlive a crashed controller; the durable journal retains their identity and unresolved budget reservations.

- Ordinary restart restores from the encrypted remote head. Never create a fresh empty cluster to conceal failed recovery. PostgreSQL major-version changes require a tested migration; do not point a different major binary at a backup.
- Before upgrades, retain the release/image identity, keys, and a verified recoverable backup. Stop new work, let active work finish/cancel with terminal confirmation, then replace the controller and perform the recovery checks.
- Monitor the HF Jobs list, account billing, local disk, durable byte meter, and remaining WAL reserve. HF bucket storage is mutable and non-versioned; the application's immutable encrypted objects are not protection from an operator deleting the bucket or losing the encryption key.
- Keep an independently retained, verified recovery copy for operational disaster recovery. A second HF Bucket can keep hosting within HF, but validate the copy's complete head/base/WAL/object dependency set before calling it a backup.
- An uncertain operation is an operator reconciliation case. Establish remote Job termination and durable receipt state before releasing reservations or removing transfer objects. Never infer termination solely from elapsed time.

From a private operator shell inside the running trusted controller environment, inspect the durable reservation ledger with:

BASH | Copy exact lines from the packaged Markdown

```
python -m pf_hf_native.control status
python -m pf_hf_native.control reconcile RESERVATION_ID
python -m pf_hf_native.control cancel RESERVATION_ID
```

Use a reservation identifier returned by `status`. `reconcile` checks provider identity and terminal state without requesting cancellation; `cancel` requests cancellation and then checks again. Neither command releases concurrency until all recorded child Jobs are positively confirmed stopped. A successful check retains the entire reserved compute charge. An uncertain result returns a nonzero exit status and keeps the reservation; investigate provider state and retry reconciliation after it becomes conclusive. Missing submission evidence does not establish that a Job never launched. Do not edit database rows to force a refund. Terminal transfer objects left by a crashed controller are eligible for cleanup on the next normal native startup.

Storage metering is a lifetime admission bound, not measured current occupancy or a monthly reset. Base backups and WAL are retained, and deleted/uncertain objects remain charged. If the meter's reservation index fills, the offline `pf_hf_native.quota compact-index` maintenance command can consolidate its entries while preserving all charged usage; review its `--help` and stop the supervisor first. Replayed old operations can then be charged again. There is no automatic history garbage collection in this release.

Compatibility and evidence

The native profile is selected by its own Dockerfile/launcher and the explicit `huggingface` execution/storage settings. Existing standalone and Linux/Docker launch commands, requirements files, and reference infrastructure remain their existing routes. The optional hybrid gateway remains available. Native Python dependencies are pinned separately in `deploy/saas/huggingface_native/requirements.lock.txt`.

Local tests under `tests/test_hf_native_*.py` exercise contracts, fault handling, confinement policy, persistence/restore rules, and deployment packaging. They include checks that the public runtime context contains no private PF code and that ordinary Linux/Docker and standalone entry points retain their defaults. Local kernel execution is recorded separately from mocked provider contracts. The local Linux 6.12.107 VM checks exercised Landlock ABI 6 enforcement with Python 3.12.14, NumPy linear algebra, multiprocessing pools, threads, subprocesses, and detached-descendant cleanup on timeout/cancellation. These checks did not run in an HF Job or validate its image pull, repository mounts, or SaaS recovery. The native Jobs serialization test uses the real pinned `huggingface_hub==1.32.0` SDK with mocked HTTP transport; it validates client request/response contracts without contacting or executing on HF. They must be reported separately from actual HF execution, PostgreSQL crash recovery, independent security assessment, scientific validation, and billing qualification. No live deployment or provider purchase is performed by the builder or by the release tests.

Use the deployment record template (HUGGING_FACE_DEPLOYMENT_RECORD.md) to record your own identities, commands, outcomes and remaining acceptance checks. Prior successful Jobs qualify only their recorded image/code/hardware combination; they are not reusable pins or full SaaS approval for a newly generated payload.

Official platform references

- Spaces overview and visibility
- Docker Spaces
- Space image access
- HF Jobs images
- HF Jobs configuration and volumes
- HF Jobs VM and nested-namespace limitations
- Linux Landlock security and limitations
- Storage Buckets
- Bucket permissions
- HF CLI copy
- Bucket S3 credentials and addressing
- PostgreSQL synchronous replication
- HF CLI 1.32.0 commands
- HF token scopes
- Docker Desktop Windows installation
- Microsoft WSL commands
- SMTP2GO SMTP ports and credentials
- SMTP2GO verified senders
- SMTP2GO setup at Network Solutions

Review current account eligibility, prices, and platform behavior when deploying; documentation and service capabilities can change independently of r34.

Optional Hugging Face deployment with managed PostgreSQL

This is an explicit alternative to native HF (HUGGING_FACE_NATIVE_DEPLOYMENT.md), not a prerequisite for its performance corrections. The consolidated full r34 release contains both options. Native HF still uses its supervised local PostgreSQL process and encrypted HF Bucket recovery. Managed HF uses a dedicated remote PostgreSQL database for durable application state and encrypted HF Bucket objects for artifacts. HF Jobs, private code, BYOK/BYOS, private workflows and the scientific runner remain supported.

Choose this guide only when adopting the external database architecture. To redeploy the already working native installation, use R34_NATIVE_HF_RELEASE_UPGRADE.md (R34_NATIVE_HF_RELEASE_UPGRADE.md). Do not install historical patch ZIPs on top of the consolidated release.

1. Choose the operation

Operation	Controller repository	Database and object state	Tool mode
New managed installation	New private Docker Space, or unused private blank Space	Empty dedicated PostgreSQL public schema and empty HF object prefix	`fresh`
Existing native HF to managed	Existing private native controller Space	Existing native Bucket/key/prefix, empty dedicated target PostgreSQL public schema	`migrate`
Existing managed HF code upgrade	Existing private managed controller Space	Existing authoritative schema-9 or schema-10 PostgreSQL, same Bucket/key/prefix	`upgrade`

`fresh` refuses a nonempty database or object prefix; it must never reset an existing deployment. `migrate` checks the sealed deployed native controller, restores its database, verifies rows and referenced encrypted objects, imports its quota reservations and audits, applies schema 10, and retires native database authority before admitting the managed application. It requires stopped or explicitly paused work, not just expired leases.

`upgrade` is a guarded additive schema-9 to schema-10 upgrade (and an idempotent schema-10 refresh). Pause every replica before maintenance. The gate verifies stopped work, preserves the ready authority marker and quota totals, adds the accounting/recovery/mail tables and backfills attributable account usage. It does not reimport native state or reset charges. Older or future schemas are rejected by this managed upgrade route. See public service deployment (R34_PUBLIC_SERVICE_DEPLOYMENT.md) and managed accounting (HUGGING_FACE_MANAGED_ACCOUNTING.md).

All three operations use two guarded commits: maintenance first, application second. The tools preserve Space identity, access tokens, Variables and Secrets. They do not create or bill a database, create HF resources, configure Secrets, reset quota, or execute paid scientific/provider tests. The operator explicitly configures the optional new database.

2. Database prerequisites

For `fresh` and `migrate`, provision a dedicated PostgreSQL database whose operator account can create and migrate the application schema. Use PostgreSQL 15 or a compatible newer version, and verify compatibility with the bundled PostgreSQL 15 client tools. PostgreSQL 15 documentation describes forward loading of `pg_dump` output into newer server versions; the maintenance gate therefore accepts 15 and newer rather than excluding 16/17. This is a compatibility expectation, not a tested provider certificate. The application must connect to a writable primary with TLS, `fsync=on` and `synchronous_commit=on` (or `remote_apply`). Use a direct endpoint or a **session affine** pool; transaction pooling is unsupported because scoped session advisory locks must survive transaction boundaries.

For `upgrade`, retain the existing database and verify its backup first.

Place the database near the HF controller. Confirm actual PostgreSQL protocol connectivity from HF before migration; ordinary port 5432 may be blocked by HF egress policy. Use a provider-supported permitted endpoint/port or approved network arrangement, retaining verify-full TLS. An HTTPS API URL is not a PostgreSQL endpoint. Configure durable backups/PITR and exercise a restore before relying on it. Record connection capacity, retention, regional latency and failover behavior. Startup probes test protocol and server settings, not provider disk durability or restoration success.

The database URL must contain an explicit username/password/database, remote host, `sslmode=verify-full` and `connect_timeout=1..30`, for example:

TEXT | Copy exact lines from the packaged Markdown

```
postgresql+psycopg://USER:URL_ENCODED_PASSWORD@DB_HOST:5432/PF_DATABASE?sslmode=verify-full&sslrootcert=/etc
/ssl/certs/ca-certificates.crt&connect_timeout=10
```

Use the provider's correct trusted CA configuration. A private provider CA must be securely provisioned at its configured path in the container; never disable certificate or hostname checks. The packaged public system CA bundle is appropriate only when it validates that provider's certificate. No DSN or password belongs in source, a public Variable, shell history, or a shared log. The validator rejects host overrides, `options`, duplicate TLS values and unbounded connection timeouts.

3. Extract the release and build the contexts

Verify the full release's `.sha256`, extract both the `.xz` and `.zip` layers into a **new** project directory, and retain the archive and qualification records. Existing HF installations retain all original resources and keys.

Prepare the Python 3.12 HF tools environment and login as described in native guide section 2 (HUGGING_FACE_NATIVE_DEPLOYMENT.md#2-prepare-the-full-release-and-windows-tools). The operator tools use `huggingface_hub==1.32.0`. The following commands are **Windows CMD**, one line at a time. Replace the project path and owner before running. An existing correctly configured tools environment can be reused.

BAT | Copy exact lines from the packaged Markdown

```
set "PF_PROJECT_ROOT=D:\privileged-frame-simulation"
set "PF_HF_OWNER=YOUR_HF_USERNAME"
set "PF_HF_APP_REPO=%PF_HF_OWNER%/pf-simulation-saas"
set "PF_HF_CODE_REPO=%PF_HF_OWNER%/pf-simulation-runner"
set "PF_HF_RUNTIME_REPO=%PF_HF_OWNER%/pf-runtime-r34"
set "PF_HF_MANAGED_UPLOAD=%USERPROFILE%\pf-hf-managed-r34-generated"
set "PF_HF_MANAGED_STATE=%USERPROFILE%\pf-hf-managed-r34-deployment"
cd /d "%PF_PROJECT_ROOT%"
call "%USERPROFILE%\pf-hf-tools-r34\Scripts\activate.bat"
hf auth whoami
python deploy\saas\huggingface_managed\build_space.py --output "%PF_HF_MANAGED_UPLOAD%"
```

The output and state directories must be new. The builder is offline and does not read operator credentials. It produces:

Context	Purpose
`runtime/`	Public-safe third-party dependencies and generic verifier only
`jobs/`	Private scientific code archive and immutable identity
`maintenance-fresh/`	Empty-target initialization, no application or workers
`maintenance-migrate/`	Native restoration/import/retirement, no application or workers
`maintenance-upgrade/`	Existing managed schema-9/10 additive upgrade and authority/quota verification, no application or workers
`space/`	Managed application, admitted only after the database marker is ready

Do not upload the whole release or parent generated directory. Keep the controller Private during maintenance. After public readiness, Protected visibility may expose its application while retaining private source. Never make the scientific code repository Public. The managed contexts use the same scientific code digest as their private `jobs/` payload.

For a new installation, create the private controller/code Spaces, private Bucket, scoped HF token and S3 credential pair using native guide section 1. Create a separate public dependency-only runtime Space if one is needed. For migration, keep the existing resources and keys. Build/upload the runtime and jobs contexts and obtain the exact immutable pins using native guide sections 4-6 (HUGGING_FACE_NATIVE_DEPLOYMENT.md#4-build-and-upload-the-runtime-and-private-code) with **this managed-generated output directory** in place of the native one. Do not run the native controller upload or `PF_HF_BOOTSTRAP=true` steps.

The key upload and identity checks are:

BAT | Copy exact lines from the packaged Markdown

```
hf upload "%PF_HF_RUNTIME_REPO%" "%PF_HF_MANAGED_UPLOAD%\runtime" . --repo-type space
hf upload "%PF_HF_CODE_REPO%" "%PF_HF_MANAGED_UPLOAD%\jobs" . --repo-type space
hf spaces info "%PF_HF_CODE_REPO%" --expand sha
python -c "import json,sys; print(json.load(open(sys.argv[1], encoding='utf-8'))['sha256'])"
"%PF_HF_MANAGED_UPLOAD%\jobs\code-manifest.json"
```

Use the native guide's registry helper to resolve the runtime image digest and run its confinement qualification. Record the **40-character code commit**, **64-character code archive SHA256**, and **OCI image SHA256 digest**. Updating code only is insufficient: the controller and private Jobs pins must agree. For migration, complete the drain and pause at the beginning of section 4 **before changing any controller code pin or database setting**; automatic restart of old controller source against new pins would fail its identity gate.

4. Preserve or configure the Space settings

Migration or managed upgrade: first drain scientific and AI/provider work, pause autonomous investigations at safe checkpoints, finish/reconcile pending uploads and BYOS operations, and pause the controller and any other application replicas. Keep a provider-independent recovery copy of native encrypted state and preserve every original decryption/database key. Do not treat an ambiguous provider timeout as proof that the job stopped. The maintenance program independently refuses unresolved work. For the native single-controller deployment, stop user admission before beginning this maintenance window.

All modes: in the controller Space's **Settings**, configure the required Secrets and Variables. Setting updates can trigger rebuilding/restarting; keep all application instances paused while preparing cutover. No managed profile Variable is required: the generated managed Dockerfile and launcher explicitly select it. Never change a currently serving native controller's deployment profile manually.

Setting	Where	Action
`PF_SAAS_DATABASE_URL`	Secret	Verified remote PostgreSQL DSN; preserve the existing DSN for managed upgrades
`PF_HF_TOKEN`, `PF_HF_S3_ACCESS_KEY_ID`, `PF_HF_S3_SECRET_ACCESS_KEY`	Secrets	Preserve existing scoped credentials, or create for fresh installation
`PF_HF_STATE_KEY`	Secret	Preserve exactly on migration; independent 32-byte state key for fresh installation
`PF_SAAS_SECRET_KEY`	Secret	Preserve session-signing key; create independent key only for a new installation
`PF_SAAS_ASSISTANT_CREDENTIALS_KEY`, `PF_SAAS_BYOS_CREDENTIALS_KEY`	Secrets	Preserve existing encrypted BYOK/BYOS keys and enabled-provider configuration

Setting	Where	Action
`PF_HF_PG_ADMIN_PASSWORD`, `PF_HF_PG_APP_PASSWORD`, `PF_HF_PG_WAL_PASSWORD`	Secrets	Preserve original native credentials during migration; managed serving itself does not launch local PostgreSQL
`PF_HF_NAMESPACE`, `PF_HF_BUCKET`, `PF_HF_PREFIX`	Variables	Preserve exact native identity on migration; select an empty prefix for fresh mode
`PF_HF_CODE_REPO`, `PF_HF_CODE_REVISION`, `PF_HF_CODE_SHA256`	Variables	New private Jobs repository pins from section 3
`PF_HF_JOB_IMAGE`, optional `PF_HF_ASSISTANT_IMAGE`	Variables	Qualified immutable public dependency image digest, with no private PF source
`PF_HF_MAX_STORAGE_BYTES`	Variable	Preserve the actual approved source meter limit; the existing service raised to 32 GiB uses 34359738368. Historical 16 GiB installations use 17179869184 only while that remains their approved ceiling.
`PF_HF_WAL_RESERVE_BYTES`	Variable	Preserve source reservation policy; default 268435456 when applicable
`PF_HF_BOOTSTRAP`	Variable	Absent or `false` for every managed mode
`PF_SAAS_PUBLIC_URL` and email/authentication settings	Variables/Secrets as in native guide	Preserve configured origin and functioning authentication/email
`PF_SAAS_DEPLOYMENT_BUDGET_ENABLED`	Variable	`true`; retain configured compute/cost/queue ceilings
`PF_SAAS_USER_MAX_CONCURRENT_JOBS`	Variable	Explicit positive value; 1 is a conservative initial setting
`PF_SAAS_USER_MAX_QUEUED_JOBS`	Variable	Explicit positive value; 20 is a conservative initial setting
`PF_SAAS_AI_ADMISSION_ENABLED`	Variable	`true`
`PF_SAAS_API_USER_REQUESTS_PER_MINUTE`	Variable	Positive value; default 600
`PF_SAAS_SCALING_AUTO_SCALE_IN`	Variable	`false`; the AWS scale-in controller is not an HF control plane

Retain positive existing limits rather than replacing working values with these examples. Remove `PF_SAAS_HF_GATEWAY_SECRET`: the managed/native direct controller is distinct from the legacy Linux-backend gateway. Do not reuse the HF state key for session, assistant or BYOS credential encryption.

For a fresh managed installation only, set `PF_HF_ADMIN_EMAIL` and `PF_HF_ADMIN_PASSWORD` Secrets to initialize the first operator after activation; remove the password Secret after verifying login. Never use these to reset or promote an existing account. Complete the native guide's full email, authentication, BYOK/BYOS and deployment-budget configuration rather than treating the shortened table above as a complete fresh-service credential setup.

Managed HF can retain SMTP2GO encrypted SMTP on port 443 with `PF_SAAS_SMTP_SSL=true` and `PF_SAAS_SMTP_STARTTLS=false`, its verified sender and existing credentials. Resend HTTPS is optional. Both transports require real account-email acceptance evidence before public launch; see public email delivery (PUBLIC_EMAIL_DELIVERY.md). The checker binds sender, SMTP endpoint and TLS policy into that evidence's configuration fingerprint.

Set connection capacity before preparing. Main-pool capacity must be at least `PF_SAAS_WORKER_SLOTS + 4`; default main pool 5 plus overflow 5 supports one worker. Include main, lease and assistant pools across **all** API and worker processes/replicas plus the administrator reserve in the provider's connection budget. The default one-process configuration estimates 23 total connections including ten reserved administration connections. Configured process counts must describe the actual maximum fleet. They do not create replicas or increase the database provider's cap. See the shared settings reference linked in R34_CONSOLIDATED_DEPLOYMENT.md (R34_CONSOLIDATED_DEPLOYMENT.md).

5. Prepare and install maintenance

Choose **one** preparation command. All modes verify the newly uploaded scientific archive and exact remote controller bytes. `prepare` is read-only and saves a local source snapshot; the state directory contains private code and must remain private.

Fresh installation:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py prepare --bundle "%PF_HF_MANAGED_UPLOAD%" --space
"%PF_HF_APP_REPO%" --mode fresh --state-dir "%PF_HF_MANAGED_STATE%"
```

Existing native installation to managed PostgreSQL:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py prepare --bundle "%PF_HF_MANAGED_UPLOAD%" --space
"%PF_HF_APP_REPO%" --mode migrate --state-dir "%PF_HF_MANAGED_STATE%"
```

Already managed schema-9 or schema-10 installation:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py prepare --bundle "%PF_HF_MANAGED_UPLOAD%" --space
"%PF_HF_APP_REPO%" --mode upgrade --state-dir "%PF_HF_MANAGED_STATE%"
```

Verify `prepared_no_remote_changes`, repository, mode and source revision. Then:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py maintenance --state-dir "%PF_HF_MANAGED_STATE%"
hf spaces wait "%PF_HF_APP_REPO%" --timeout 25m
python deploy\saas\huggingface_managed\deploy_release.py status --state-dir "%PF_HF_MANAGED_STATE%"
```

Maintenance pauses the Space before an expected-parent commit and requests a restart. `RUNNING` alone only means the maintenance server is reachable; it is not application readiness. Status must verify the runtime revision **and** an authenticated receipt with `managed_fresh_ready`, `managed_migration_ready`, or `managed_upgrade_ready`, the same deployment ID/source revision, `maintenance_only=true` and `application_serving=false`. A migration receipt additionally confirms native authority retirement. The upgrade guard requires `existing_authority_preserved=true`, `schema_version=10`, `quota_charges_preserved=true` and `migration_scope=additive_schema_and_account_attribution`. The upgrade operation emits `data_changed=true` because it can apply additive schema and attribution work; this is not a read-only gate, and that flag is not a row-by-row change count. Do not require `data_changed=false` for a schema-10 refresh. Do not change settings between preparation and activation.

The authenticated receipt request rejects redirects and uses verified HTTPS. For a managed Windows/system CA, install `truststore` in the tools environment and place `--system-ca` before each subcommand, for example:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py --system-ca status --state-dir
"%PF_HF_MANAGED_STATE%"
```

The flag applies system certificate trust to Hub and receipt requests while retaining certificate/hostname verification. An enterprise proxy/403/TLS failure still blocks activation; fix approved network/CA access and rerun status. Do not bypass receipt verification or disable TLS.

6. Activate and qualify

Only after successful maintenance:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py activate --state-dir "%PF_HF_MANAGED_STATE%"
hf spaces wait "%PF_HF_APP_REPO%" --timeout 25m
python deploy\saas\huggingface_managed\deploy_release.py status --state-dir "%PF_HF_MANAGED_STATE%"
hf spaces logs "%PF_HF_APP_REPO%" --tail 200
```

Activation independently rechecks the maintenance receipt before switching the Docker entry point. Expect `active_code_verified`, the active commit as `runtime_sha`, and `running_revision_verified=true`. Then open the PF origin and verify authenticated login, organization/project archive and restore, private workflows, BYOK/BYOS access, a controlled scientific run, authenticated ZIP retrieval, persistence through a restart, and the provider database backup/restore procedure. Record the real requests' status, timing and results. Offline tests or an HF stage do not establish user-visible latency.

Managed serving checks the ready marker, schema 10, writable TLS PostgreSQL and imported/fresh quota. It never falls back to an empty local database. Startup reports phase and exception class without raw SQL, DSNs or private data. `PF_HF_MANAGED_WORKERS_ENABLED=false` makes an instance API-only; planning multiple replicas also requires accurate shared connection/provider admission budgets.

7. Failure, resume and upgrade boundaries

Keep the generated contexts and prepared state directory. If a commit call fails without a definite outcome, do not repeat it blindly:

BAT | Copy exact lines from the packaged Markdown

```
python deploy\saas\huggingface_managed\deploy_release.py reconcile --state-dir "%PF_HF_MANAGED_STATE%"
```

The tool accepts only its exact expected child commit and bytes. A reconciled commit does not automatically restart; use `hf spaces restart` and repeat status. If no commit occurred, the Space remains paused and that stage can be retried. Do not delete prepared state to bypass a disagreement with remote history.

For a failed maintenance container, inspect its `PF_MANAGED_MIGRATION` phase, correct the external cause, and restart **the same maintenance revision**. Fresh initialization resumes only its exact marker/identity; it cannot adopt native state. Native migration resumes only an exact verified copy or staged target. A partial foreign target is never overwritten. Do not manually zero quota, clear ambiguous jobs, delete the native head, or promote a partly imported target.

Once native migration maintenance may have started, **a code-only native rollback is not supported**. The encrypted schema-2 retirement tombstone intentionally prevents old native binaries from becoming a second authority. Keep both stores, keys and receipts. Repair forward, or use an independently reviewed reverse data migration with both sides stopped and verified. The original native quota reservations, retry identities and limit history remain charged after migration; deleting artifact files does not refund conservative lifetime accounting.

For an already managed deployment, use `--mode upgrade` with new generated contexts and a new state directory. Back up the authoritative database, drain work and pause **every** application/worker replica before changing code pins. The guarded maintenance step must report `managed_upgrade_ready`. It performs the additive schema-9 to schema-10 migration, or verifies an already schema-10 deployment without reimporting native state or resetting charges. It preserves the authority marker and charged totals. Schema 8, future schema versions, a mismatched marker, changed quota policy or unresolved work are blocked. Do not switch to `fresh` or `migrate` to bypass that gate. Schemas other than 9 or 10 require a separately reviewed and qualified forward migration with database recovery available. Native full-release upgrades remain independently supported by the native guide.

Qualification status

This release includes local managed-runtime, migration-state, deployment-guard, shared-state, deadline and schema regression tests, including mocked Hub operations and SQLite semantic fixtures. The retained isolated HF Job evidence under `validation_r34_public_service/isolated_postgres/` records 144 passing checks with no skips, including real PostgreSQL migration, accounting, admission, execution recovery, physical restore and a scientific worker regression. That Job used temporary test data and loopback test services; external mail, storage and AI-provider behavior remained fixtures. Ordinary local PostgreSQL-only tests still require an explicit test database and skip when it is absent.

No production managed-database cutover, live account-email delivery, production provider workload, regional failover or large-fleet load test is certified by this evidence. The migration runs actual bounded TLS/session-affinity/transaction probes on the operator's chosen target before moving authority; those probes do not replace that target's backup/restore or load qualification. Keep this optional mode separate from the user-confirmed native HF responsiveness improvement.

r34 shared capacity settings

Current public-service and schema-10 upgrade instructions: r34 public service deployment (R34_PUBLIC_SERVICE_DEPLOYMENT.md). This includes both 2026-10-09 lifecycle packages. Large active-user capacity requires the measured qualification described there.

These hosted controls are part of the complete release, independently of whether HF uses native persistence or optional external PostgreSQL. Use the actual runtime settings in `pf_saas/settings.py` as the configuration contract. Names below all have the `PF_SAAS_` prefix. The scope matters: a per-process cap does not become a whole-fleet cap by adding replicas. Standalone does not read these hosted settings.

Database connections and deadlines

Setting (after `PF_SAAS_`)	Default	Scope and meaning
`DATABASE_POOL_SIZE`	5	Main SQL pool per database client
`DATABASE_MAX_OVERFLOW`	5	Bounded extra main connections per client
`DATABASE_LEASE_POOL_SIZE`	1	Lease engine connections; no overflow
`DATABASE_ASSISTANT_POOL_SIZE`	2	Assistant SQL engine pool
`DATABASE_ASSISTANT_MAX_OVERFLOW`	0	Assistant overflow
`DATABASE_POOL_TIMEOUT_SECONDS`	5	Wait to acquire a pooled connection
`DATABASE_CONNECT_TIMEOUT_SECONDS`	5	PostgreSQL connection establishment
`DATABASE_STATEMENT_TIMEOUT_MS`	30000	PostgreSQL statement timeout
`DATABASE_LOCK_TIMEOUT_MS`	10000	PostgreSQL lock wait; cannot exceed statement timeout
`DATABASE_IDLE_TRANSACTION_TIMEOUT_MS`	60000	PostgreSQL idle-in-transaction timeout
`DEPLOYMENT_API_PROCESSES`	1	Whole deployment's API process count
`DEPLOYMENT_WORKER_PROCESSES`	0	Whole deployment's additional worker process count; set to actual count
`DATABASE_FLEET_MAX_CONNECTIONS`	100	Configured service budget ceiling, within the database's real capacity
`DATABASE_RESERVED_CONNECTIONS`	10	Administration/migration/other clients outside declared service pools

Linux/Docker and managed PostgreSQL profiles use session advisory resource locks and SQL generation checks. Use direct or session-affine connections; transaction/statement pooling is unsupported for those service clients. The on-host reference already uses a direct local connection.

Native HF retains local per-resource filesystem locks on its single controller, with its global durable ownership/WAL fencing still enforced. It does not add SQL generation writes solely to coordinate these local resource operations. This native exception does not authorize a second controller or remove durable publication acknowledgement.

For PostgreSQL profiles other than native HF, the main pool plus overflow must total at least four connections for shared resource-lock, publication and callback work. Optional managed HF requires at least `WORKER_SLOTS + 4`. Native HF retains its original per-pool bounds and combined connection ceiling; the additional SQL resource-lock minimum does not apply to its local lock path. The ordinary 5+5 main/overflow defaults satisfy all profiles at one worker slot.

The conservative validation budget is:

`(main pool + overflow + lease pool + enabled assistant pool + assistant overflow) * (API processes + worker processes) + reserved connections`.

The reference host and local Compose explicitly declare one API plus one worker; with enabled assistant defaults their conservative budget is 36 connections. The distributed example also declares one API plus one worker; with assistant disabled it is 32. Update the counts before growing the fleet. Different pool sizes across replicas require an equivalent conservative fleet calculation; use identical settings in the supplied topology. A runtime declaration does not change the PostgreSQL server's `max_connections` or reserve sockets itself. Native HF additionally limits its combined main/lease/assistant configured pools to 25, leaving headroom for its local PostgreSQL control and WAL services.

Connection, lock and statement limits bound different operations; they are not one end-to-end request deadline. Batch migration/backfill time must be qualified on a copy of the actual database. For a large migration, use an explicitly reviewed temporary migration-only statement limit; retain the application limit when normal services restart. No service should run mixed schema versions.

HTTP and browser work

Setting	Default	Meaning
`API_MAX_INFLIGHT_REQUESTS`	64	Concurrent requests per API process, including streams
`API_MAX_INFLIGHT_UPLOADS`	2	Concurrent large uploads within that request cap
`API_USER_REQUESTS_PER_MINUTE`	600	Authenticated account request rate; native HF uses its single-controller memory counter, other hosted profiles use shared SQL; 0 disables in profiles that permit it
`API_RATE_LOCAL_MAX_ACCOUNTS`	10000	Maximum active account counters in the native HF memory limiter
`MAX_PARALLEL_EXPORTS`	2	ZIP preparation/delivery operations per API process
`MAX_REQUEST_BYTES`	8388608	Ordinary request body limit
`WORKFLOW_MAX_REQUEST_BYTES`	25165824	Workflow request limit
`MAX_UPLOAD_BYTES`	104857600	Uploaded asset limit
`MAX_ARTIFACT_BYTES`	536870912	Artifact limit
`UI_POLL_ACTIVE_MS`	5000	Active-work polling
`UI_POLL_IDLE_MS`	30000	Idle polling
`UI_POLL_HIDDEN_MS`	60000	Hidden-tab polling
`UI_POLL_MAX_BACKOFF_MS`	120000	Error backoff ceiling
`UI_POLL_QUALIFICATION_MS`	60000	Qualification refresh interval

Native HF uses a bounded in-memory account limiter on its single controller so ordinary authenticated reads do not create a durable SQL/WAL write solely to count HTTP requests. It uses monotonic one-minute windows, resets its rate counters on controller restart, and returns `429` when an account exhausts its minute allowance or a new account cannot fit the active-counter bound. This counter is intentionally not a durable quota/billing ledger. Native HF must still have one controller; it is not a shared counter across replicas.

Linux/Docker and managed/distributed profiles retain atomic SQL account counters for requests across their service processes. Authentication throttling, authorization, native ownership fencing and durable job/provider/usage accounting remain in place. The native limiter does not bypass those checks.

Polling intervals must be 1,000..600,000 ms and increase from active through idle, hidden and maximum backoff. Shared browser scheduling adds jitter and avoids overlapping polls; stale selections are ignored. Authorized list routes use cursor pages (default 100, maximum 500), and the UI offers further pages. These reduce unnecessary work without changing project authorization.

Assistant and provider admission

Setting	Default	Meaning
ASSISTANT_BACKGROUND_WORKERS	4	Background threads per service process
ASSISTANT_BACKGROUND_QUEUE_SIZE	32	Bounded pending background work per process
ASSISTANT_TENANT_QUEUE_SIZE	16	Tenant queue bound within that executor
ASSISTANT_CLEANUP_WORKERS	2	Cleanup executor bound
ASSISTANT_ATTACHMENT_MAX_BYTES	4194304	Assistant attachment limit
ASSISTANT_SNAPSHOT_MAX_BYTES	268435456	Assistant snapshot limit
AI_ADMISSION_ENABLED	true	Enable durable shared hosted provider admission
AI_MAX_CONCURRENT_REQUESTS	8	Deployment provider-request concurrency
AI_TENANT_MAX_CONCURRENT_REQUESTS	2	Per-tenant provider-request concurrency
AI_PROVIDER_MAX_CONCURRENT_REQUESTS	4	Per-provider concurrency
AI_REQUESTS_PER_MINUTE	120	Deployment provider-request rate
AI_TENANT_REQUESTS_PER_MINUTE	30	Tenant provider-request rate
AI_PROVIDER_REQUESTS_PER_MINUTE	120	Provider request rate
AI_TOKENS_PER_MINUTE	1000000	Deployment estimated token admission
AI_TENANT_TOKENS_PER_MINUTE	250000	Tenant estimated token admission
AI_PROVIDER_TOKENS_PER_MINUTE	1000000	Provider estimated token admission
AI_MONTHLY_COST_MICRO_USD	0	Deployment estimated cost cap; 0 disables this cap
AI_TENANT_MONTHLY_COST_MICRO_USD	0	Tenant estimated cost cap
AI_PROVIDER_MONTHLY_COST_MICRO_USD	0	Provider estimated cost cap
AI_INPUT_COST_PER_MILLION_MICRO_USD	0	Conservative input-token tariff needed for a cost cap
AI_OUTPUT_COST_PER_MILLION_MICRO_USD	0	Conservative output-token tariff needed for a cost cap

Tenant/provider concurrency cannot exceed deployment concurrency. A positive cost ceiling requires both positive conservative tariffs. Zero cost ceilings are explicit absence of that particular cost cap, not free provider use. These are application estimates and admissions, not provider invoices. BYOK and operator credentials remain private; institution and project disclosure policy still applies. Ambiguous external outcomes retain

accounting until reconciled; a timeout alone does not authorize duplicate requests or refund a reservation. Scientific execution capacity is separately admitted even when initiated by AI.

Jobs, maintenance and state

Setting	Default	Meaning
`WORKER_SLOTS`	1	1..64 scientific execution slots per worker host/native HF controller
`SCHEDULER_BATCH_SIZE`	100	Bounded queue candidate selection (maximum 1000)
`USER_MAX_CONCURRENT_JOBS` / `USER_MAX_QUEUED_JOBS`	0 / 0	Single-profile unset per-user caps; distributed/managed require positive values
`MAX_CONCURRENT_JOBS` / `MAX_QUEUED_JOBS`	1 / 20	Default organization plan caps
`DEPLOYMENT_BUDGET_ENABLED`	false	Optional single-host, mandatory distributed/HF execution admission
`DEPLOYMENT_MAX_CONCURRENT_JOBS` / `DEPLOYMENT_MAX_QUEUED_JOBS`	1 / 10000	Shared admitted execution/queued caps
`DEPLOYMENT_MONTHLY_COMPUTE_SECONDS`	36000	Shared execution reservation allowance when enabled
`MONTHLY_COMPUTE_SECONDS`	36000	Default organization allowance
`MAX_JOB_SECONDS`	3600	Scientific execution limit
`JOB_PREPARE_SECONDS` / `JOB_TRANSFER_SECONDS` / `JOB_COLLECT_SECONDS`	300 / 300 / 300	Independent lifecycle phase limits
`JOB_SUBMIT_SECONDS`	60	Submission phase limit
`LEASE_SECONDS`	60	Job lease interval
`PROGRESS_INTERVAL_SECONDS`	5	Coalesced progress interval
`MAINTENANCE_INTERVAL_SECONDS`	5	Bounded maintenance cadence
`JOB_MEMORY_MB` / `JOB_CPUS` / `JOB_PIDS_LIMIT`	4096 / 2 / 128	Scientific execution resource bounds

Retain existing explicit organization/deployment storage and cost settings when upgrading; defaults are not a quota reset. See the distributed guide for worker host scaling, drain/termination settings and measured-capacity acceptance, and the native guide for native encrypted quota/persistence controls. Optional managed HF has additional TLS, readiness, state-migration and process settings in its own guide. Shared settings do not imply replicated native HF writers or a change to the scientific model.

Measuring the configured SaaS capacity

The full public-release integration adds `release_tools/saas_capacity_mixed.py` for real authenticated writes, storage, scientific-job and assistant journeys. The original read-only probe remains available unchanged for cheap baseline measurement. Neither tool converts a configured user count into a capacity guarantee. This release has not been load-qualified for 200,000 concurrent users; that acceptance campaign requires the actual provisioned deployment and generators.

The deployment can start with one user and one execution slot. Grow the API replicas, shared PostgreSQL capacity and job worker fleet independently as the measured workload grows. A user count is not an execution-slot count: 200,000 signed-in users with one request every five seconds offer about 40,000 HTTP requests per second, while their scientific jobs have a separate arrival rate, runtime, queue, budget and concurrency limit.

`release_tools/saas_capacity_probe.py` provides a repeatable authenticated HTTP workload. It uses the installed `httpx` dependency and never creates accounts, submits jobs, starts assistants, uploads data, changes configuration, or calls paid scientific providers. GET requests still consume deployment resources and may update authentication activity timestamps. Run it against an authorized test deployment with representative data and monitored resource costs.

Workload and distinct users

Create and sign in distinct test accounts using the application's ordinary account workflow. Give each account access to its stated organization and project. Export only the existing `pf_session` cookie value and the corresponding IDs into a private JSON file. The application authenticates these routes through that cookie; a bearer token is not a substitute. This tool does not generate accounts, credentials or placeholder identities for a target population.

Example file shape, with illustrative values that must be replaced:

JSON | Copy exact lines from the packaged Markdown

```
{
  "sessions": [
    {
      "cookie": "EXISTING_PF_SESSION_COOKIE_VALUE",
      "user_id": "existing_user_id",
      "project_id": "existing_project_id",
      "org_id": "existing_org_id",
      "job_id": "optional_existing_job_id"
    }
  ]
}
```

On Unix, the credential file must be owned by the running user with mode `0600` or `0400`. Store it outside the release source and all published evidence. On Windows, use a private account directory and its access controls. Never paste cookies into shell arguments, source control, reports or shared logs. The tool rejects duplicate cookies and duplicate user IDs and verifies each account's identity using `/api/me` before continuing its journey. It does not print credentials, individual IDs, response bodies or HTTP exception messages.

Two fixed, read-only profiles are available:

Profile	Repeating request sequence	Additional input
`browse`	`/api/me`, `/api/catalog`, `/api/jobs`, `/api/usage`, `/api/jobs`	Organization and project IDs
`monitor`	`/api/me`, `/api/jobs`, existing job detail three times	A preexisting accessible `job_id` per user

Every request carries that user's cookie and project selection. The usage request uses that user's organization. Each user has at most one outstanding request. `--cadence-seconds` sets the interval between successive requests, not between five-request cycles; it represents an explicit browsing or polling assumption. Real browser background tabs, idle sessions, export traffic and job-specific polling can have different rates. Choose representative account, organization, project and job distributions; an empty demonstration account is not a representative large tenant.

Start small, then add independent generators

Run the command from the release root with its SaaS Python environment. Choose a new report filename for every run. The tool refuses existing output paths.

BASH | Copy exact lines from the packaged Markdown

```
python release_tools/saas_capacity_probe.py \
  --base-url https://your-test-deployment.example \
  --credentials /private/pf-test-sessions.json \
  --users 1 --duration-seconds 180 --ramp-seconds 10 \
  --cadence-seconds 5 --max-rps 2 --max-in-flight 4 \
  --min-samples-per-route 5 \
  --output /private/capacity-one-user.json
```

Use stages such as 1, 100, 1,000, 10,000, 100,000 and 200,000 or more distinct users. `--users N` is the global target. Its numeric validation range is 1 through 1,000,000, but the complete credential file must also fit the tool's 64 MiB input limit. The numeric range does not guarantee that a million-user roster can be loaded. With this release's 64-character session cookies and 32-character IDs, a 200,000-user roster is approximately 41-49 MiB for `browse`, or 49-59 MiB for `monitor`, depending on JSON whitespace. At 250,000 users, a compact `monitor` roster is about 62 MiB and expanded whitespace can exceed the limit. Measure the actual file; omit unused job IDs for `browse` and unnecessary JSON whitespace.

Every generator parses the entire roster and checks global identity/cookie uniqueness before retaining its selected shard. Allow memory substantially above the file size for parsed JSON, session objects and uniqueness sets during startup. Sharding bounds active requests and retained session scheduling state; it does not remove that initial full-roster memory cost or the 64 MiB file limit. Larger credential sets require a separately qualified load generator; this tool does not accept independently supplied shard-local rosters.

For large targets, run multiple load-generator machines, each with the same ordered credential roster and a different zero-based `--shard-index`. Shard `i` selects roster indexes `i, i + shards, i + 2*shards, ...`, with no overlap between correctly configured shards. All generators must agree on `--users`, `--shards`, the roster, profile, cadence, ramp, duration and planned start time. The CLI starts immediately: coordinate launch externally and use the reported UTC start times to confirm overlapping measurement windows. Per-shard files are not an automatic global aggregate or proof of simultaneous activity.

For example, 200,000 users over 200 generators gives 1,000 users per generator. At a five-second cadence each shard's nominal demand is 200 RPS. This illustrative command is for shard 0; launch indexes 0 through 199 on the independently provisioned generators only after lower stages have passed:

BASH | Copy exact lines from the packaged Markdown

```
python release_tools/saas_capacity_probe.py \
  --base-url https://your-test-deployment.example \
  --credentials /private/pf-test-sessions.json \
  --users 200000 --shards 200 --shard-index 0 \
  --duration-seconds 1200 --ramp-seconds 300 \
  --cadence-seconds 5 --max-rps 250 --max-in-flight 128 \
  --timeout-seconds 10 --min-samples-per-route 1000 \
  --max-error-rate 0.01 --max-p95-ms 1000 --max-schedule-lag-ms 1000 \
  --output /private/capacity-200000-shard-000.json
```

Those numbers are a workload specification, not an established hardware sizing recommendation or a claim that 200 generators are necessary or sufficient. Determine each generator's safe capacity from its own CPU, network, connection, memory and scheduling-lag measurements. The scheduler keeps small records per user and at most

`--max-in-flight` asynchronous request tasks and reused connections (1-4,096 per generator). It does not start one thread or one running task per virtual user. `--max-rps` is a strict per-generator pacing ceiling; there is no catch-up request burst. A ceiling below the shard's nominal demand fails the observed-profile checks. Ramp time determines each user's first request, spread across the global target. Allow enough time after the last user arrives for multiple complete profiles and representative steady-state data.

Requests have an overall timeout and a bounded response size. Remote origins require HTTPS; plain HTTP is allowed only for local fixtures. Redirects are not followed, and environment proxy settings are not inherited. No automatic retries hide failed requests. A failed identity check or an HTTP 401/403 stops that user's journey and fails the session-coverage check; refresh expired sessions before a new independent run.

Interpret the report and correlate the deployment

The JSON records the exact target and shard configuration, users actually verified, users completing a successful full journey, route sample counts, status/error counts, bytes, mean/max/p50/p95/p99 latency, achieved RPS, peak outstanding requests, scheduling lag, missed cadences and requests already due but not started before the measurement deadline. It waits for bounded in-flight requests to finish and includes that drain time in achieved RPS. This is a paced closed-loop session workload; overload can suppress later requests. Suppressed cadences and horizon work are reported explicitly and make the profile fail. Percentiles are conservative histogram upper bounds with bins at most 5% wide; they must not be averaged across shards to manufacture global percentiles.

Exit code 0 means this shard met all declared checks: each target identity was verified, each user completed a successful journey, every route met the sample minimum and p95 threshold, error rate and scheduling lag stayed within bounds, the RPS ceiling covered nominal demand, and no due work or cadence was lost. Exit code 1 means one or more checks failed; inspect the report before increasing load. Exit code 2 means the run could not produce a completed report. The report always says `capacity_qualified: false`: a passing read-only profile is one piece of production qualification evidence, not a certification of that user count or scientific-job throughput.

Record the release hash, deployment topology and resource limits with every campaign. Correlate all shard windows with API p95/p99 and 503/429 rates, PostgreSQL connections/latency/locks, object-store latency, worker queues and claims, publication recovery, network/CPU/memory and generator saturation. Relevant controls in this release are:

Control	Scope and relationship
API replica count	Set in the deployment platform; each replica runs its own API process
`PF_SAAS_API_MAX_INFLIGHT_REQUESTS`	Per-process HTTP admission bound, default 64
`PF_SAAS_API_MAX_INFLIGHT_UPLOADS`	Per-process concurrent upload admission, default 2
`PF_SAAS_DATABASE_POOL_SIZE` and `PF_SAAS_DATABASE_MAX_OVERFLOW`	Main PostgreSQL pool per service process, defaults 5 and 5
`PF_SAAS_DATABASE_POOL_TIMEOUT_SECONDS`	Main-pool acquisition wait, default 5 seconds
`PF_SAAS_DATABASE_LEASE_POOL_SIZE`	Separate lease/heartbeat pool, default 1 with no overflow
`PF_SAAS_MAX_PARALLEL_EXPORTS`	Independent per-process export bound, default 2
`PF_SAAS_WORKER_SLOTS`	Job execution concurrency per controller or worker host; distinct from signed-in users

Budget PostgreSQL connections across every API replica, worker process and enabled assistant store, rather than assigning each replica the database's entire connection budget. The assistant store has its own pool. HF native runs one durable controller with a bounded local database connection budget; the distributed deployment is the multi-replica/multi-host path. Follow the distributed deployment guide (../../../DISTRIBUTED_SAAS_DEPLOYMENT.md) for shared state, worker admission, budget and scaling requirements.

Before declaring support for a specified concurrent-user population, also qualify representative job submission/queue/execution/publication, assistant and private-storage operations, uploads/downloads, sign-in/session renewal, tenant isolation, long-running soak behavior, safe scale-out/scale-in, and replica/worker/database failure recovery. These operations need a separate explicit test workload and cost budget; this read-only tool does not perform them. Preserve failed runs as well as successful runs, and qualify the complete campaign at the target population before changing any capacity claim.

Mixed workloads and isolated target guard

Use the mixed tool on a separate, disposable deployment with synthetic data, dedicated provider budgets, and test mailbox accounts. Configure `PF_SAAS_CAPACITY_CAMPAIGN_ID` to a unique 16-128-character identifier containing only letters, digits, underscores and dashes. `/healthz` advertises it in `X-PF-Capacity-Campaign`. The generator checks the exact header before sending any authenticated request. Keep this setting empty on the production service. The marker records an operator assertion that this is an isolated campaign; it cannot prove infrastructure separation. It grants no authentication or authorization bypass and is not a secret.

Mutations additionally require `--allow-writes`. Scientific job submission requires `--allow-compute`; provider messages require both `--allow-provider` and `--allow-compute` because an assistant can invoke execution tools. A prompt asking it not to execute is not a security boundary. Email requests require `--allow-email` and must address owned test mailboxes. Do not point any of these workloads at the current production HF controller. The harness does not change Space visibility, provision resources, stop workers or inject infrastructure faults automatically.

Scenario	Actual exercised behavior	Required fixture or permission
`browse`	Catalog, job page and organization usage reads	Existing account, organization, project
`login`	Local-password login, fresh session cookie and `/api/me` identity/CSRF check	Account email/password; ordinary login rate limits remain active
`project`	Update disposable project description; read it back and compare	Editor/owner fixture and `--allow-writes`
`asset`	Upload a small CSV through multipart API, list its identity, delete it	`--allow-writes`; measures the declared small-upload workload
`job`	Idempotent submit, poll terminal state, require `completed`, read bound results	Reviewed `--job-payload`, `--allow-writes` `--allow-compute`
`result`	Completed-job detail, bound results and bounded streamed ZIP download	Retained completed fixture job
`assistant`	Capture completed job, upload private attachment, read it back, remove attachment	Enabled assistant, completed fixture job, `--allow-writes`
`assistant_message`	Submit evidence-summary question; require completed message and investigation; verify persisted message identity	Existing private session, provider credentials, all three write/provider/compute opt-ins
`isolation`	Verify own project/job exist, deny another account's project, job, results and ZIP; optional private assistant denial	Inaccessible fixtures owned by other campaign accounts

Scenario	Actual exercised behavior	Required fixture or permission
`email`	Request password-reset email through public route	`--allow-writes --allow-email`; owned test mailbox

`--mix browse=3,project=1,asset=1,isolation=1` is a deterministic weighted cycle of journeys. Account index offsets distribute the initial scenarios. Cadence is the interval between journey starts, unlike the original probe's per-request cadence. A journey can issue several HTTP requests. Choose weights and cadence from measured real user activity; they are not inferred from session count. Every target user must verify identity, complete a useful journey during the steady window, and complete isolation checks when that lane is enabled. Every configured scenario must meet `--min-journeys-per-scenario` on every shard.

Job/provider/email cost attempts have per-user hard ceilings (defaults one). Attempts consume the ceiling before the request, even if the response times out. Subsequent visits to an exhausted cost lane perform browsing and are counted as browsing, never as completed jobs/messages/emails. Set explicit larger ceilings to sustain a representative arrival rate throughout a soak. These are generator attempt limits, not provider billing guarantees: an assistant message may make several model calls or execute tools. Configure application/provider budgets separately and include them in the campaign record. A campaign of 200,000 users with one allowed job each can request 200,000 real jobs.

There are no automatic HTTP retries. An uncertain mutation disables that user, records `uncertain_mutations`, and fails the workload. Reconcile retained objects/jobs before a new run. Successful asset/attachment cleanup is performed through ordinary authorized APIs. The tool does not remove completed scientific jobs, existing assistant sessions, or project audit records; retained outputs are necessary for publication/accounting inspection. Dispose of the isolated deployment only after preserving evidence and reviewing running provider work. New mutations are forbidden once the measurement deadline passes, including requests delayed by generator pacing. Already-started work may finish bounded read-only polling. A successful HTTP write followed by unknown resource identity, failed readback or unfinished cleanup also stops that user and requires reconciliation; it is never automatically repeated on the next journey.

Shard-local roster and synchronized launch

Unlike the original probe, the mixed tool streams **shard-local JSONL** files. Each line is one account and contains its global `index`. Shard `i` must contain exactly indexes `i, i + shards, ...` in increasing order, below the global `--users` target. No generator parses a million-account global JSON document. The file limit is 512 MiB and each line is limited to 64 KiB; credentials remain private and permissions follow the original probe's ownership/mode rules.

Example for the first account of a two-account campaign:

JSON | Copy exact lines from the packaged Markdown

```
{"index":0,"user_id":"actual_user_a","org_id":"actual_org_a","project_id":"actual_project_a","cookie":"actua
l_pf_session_value_a","job_id":"actual_retained_job_a","foreign_project_id":"actual_project_b","foreign_job_
id":"actual_retained_job_b"}
```

Account B must own those foreign fixtures, and its own row points back at A's fixtures. Merely inventing unknown IDs that return 404 is insufficient: the aggregate verifies that every foreign project and job belongs to another tested account and that all accounts exercised their own fixture reads. Use separate organizations or properly restricted projects, not fixtures that are intentionally shared. For private assistant isolation add `foreign_assistant_session_id`; its existence and owner must also be recorded and reviewed in the external fixture inventory.

For login coverage, add `email` and `password`; a cookie can be omitted and the tool will authenticate normally before the first journey. For provider messages add `assistant_session_id`. Never publish these files, login cookies/passwords, private prompts or response bodies. Aggregate reports retain SHA256 account and fixture fingerprints to detect reused identities across rosters; protect those pseudonymous reports according to the campaign's data policy.

Prepare accounts and representative completed jobs through the ordinary test deployment/operator workflow. The load generator does not silently seed or mark unverified public registrations as verified. Registration, email verification, invitation and reset-token consumption need a separate mailbox acceptance campaign. Record its actual delivery receipts and token outcomes; HTTP 202 by itself is not delivery evidence.

Start with two accounts and the non-compute mix, using the exact SHA256 of the release actually installed on the target:

BASH | Copy exact lines from the packaged Markdown

```
python release_tools/saas_capacity_mixed.py run \
  --base-url https://isolated-pf-test.example \
  --campaign-id pf-isolated-campaign-20261009 \
  --release-sha256 "$DEPLOYED_RELEASE_SHA256" \
  --credentials /private/campaign-shard-0.jsonl \
  --users 2 --allow-writes \
  --mix browse=3,project=1,asset=1,isolation=1 \
  --duration-seconds 300 --ramp-seconds 10 --min-steady-seconds 240 \
  --cadence-seconds 10 --max-rps 20 --max-in-flight 4 \
  --output /private/mixed-two-users.json
```

One-user functional scenarios work by selecting a mix without isolation; cross-account isolation and the aggregate acceptance gate require at least two real accounts. Increase populations in measured stages. At 200,000 users and 200 generators, every generator retains 1,000 users' scheduling records. Use the same `--users 200000 --shards 200` and distinct `--shard-index 0` through `199`, with the corresponding private roster on each generator. Compute one future UTC epoch and distribute that identical `--start-epoch` to all generators; do not calculate a separate epoch independently on each host. Synchronize host clocks. Late-starting generators refuse to run. The aggregate permits at most one second of launch spread and requires overlapping steady measurement time.

Each generator has at most `--max-in-flight` active journeys and reused HTTP connections (maximum 4,096), rather than one task/thread per target user. `--max-rps` paces actual HTTP starts across all journeys with no catch-up burst. Network timeouts, whole-journey timeouts, bounded JSON responses and streamed download byte bounds cap work. Polling a scientific job occupies an active journey slot; budget generator slots and shards for the measured job duration. A saturated generator cannot pass merely because all accounts logged in: scheduling lag, missed journeys and users inactive in the steady window fail the checks. Ordinary auth/account/API limits remain enforced and 429/503 count as workload failures. Do not globally disable them to manufacture a result.

Choose `--journey-timeout-seconds` longer than the scientific job's timeout plus its allowed queue/publication delay. Use `--job-payload` with an actual validated PF request JSON and bounded `resource_limits.timeout_seconds`. Its SHA256 is recorded; mismatched scientific payloads cannot be aggregated. Capture separate campaigns for representative heavier workflows and larger exports. The small CSV lane does not qualify large-file ingestion. No single fixed workload mix represents every scientific workflow or assistant conversation.

Aggregate evidence, soak and failure recovery

BASH | Copy exact lines from the packaged Markdown

```
python release_tools/saas_capacity_mixed.py aggregate \
  --report /private/mixed-shard-0.json \
  --report /private/mixed-shard-1.json \
  --output /private/mixed-campaign-aggregate.json
```

Supply every shard report exactly once. Aggregate rejects missing/duplicate shards, mismatched campaign configuration/workload, duplicate account identities, inconsistent counts/histogram totals, invalid observation windows and insufficient steady overlap. It merges histogram bins for global percentiles; it never averages p95 values. Each report includes sample/status/ error counts, response bytes, successful journeys by scenario, failed journey categories, actual steady request rate, scheduling lag, missed work, verified active users and bounded cost attempts. A write returning 200 with the wrong resource or failed readback is a failed journey. Job success

requires terminal completion and matching result identity; a provider response with a partial or failed investigation does not pass. Exit codes are 0 for the declared observed workload passing, 1 for completed failing evidence, and 2 for an incomplete run. Output paths are exclusively created; preserve unsuccessful evidence too.

The supplied release SHA256 is an operator declaration. Independently verify the deployed code archive and image/source pins; the campaign marker verifies only the target's configured campaign identifier. Local report consistency checks do not authenticate evidence files or replace an independent review.

The aggregate still emits `capacity_qualified: false`. To accept a production capacity target, preserve and review all of the following for the same release, deployment topology, time window and workload mix:

1. Resource inventory: API replicas, database instance/pools, worker slots, provider quotas, object storage, shared assistant filesystem, mail service, generator hardware and network. Include configured budgets and provider account concurrency limits.
2. Sustained mixed workload at target active users, with a declared soak length, job arrival/completion rates, queue-delay limits, p95/p99 latency thresholds, request/error rates, object bytes and every shard's activity. Choose a soak long enough to expose session renewal, recurring maintenance and accounting retention; the CLI allows up to 24 hours per campaign.
3. Correlated API/database/lease/object/worker/generator CPU, memory, connections, latency, queue and saturation metrics. HTTP throughput alone cannot determine the capacity or headroom of a downstream scientific provider.
4. Scheduled, operator-controlled test-environment faults: terminate an API replica during writes, stop an executing worker, interrupt object access, and exercise database failover using the database operator's procedure. Timestamp each event, retain failed requests, and show subsequent recovery, bounded queue drain, no duplicate publication, and no premature refunds. The generator does not execute fault commands or certify recovery from a restored HTTP 200 alone.
5. Audited quota lifecycle evidence before/after successful, failed, interrupted and deleted attempts; conservative-mode compatibility; safe-mode settlement; historical charge reconciliation and retained file preservation.
6. Real registration/verification/reset/invitation mailbox receipts, replay and expired-token rejection, shared auth rate limits, ordinary-user denial of platform administration, BYOK/BYOS isolation/rotation, and retained artifact integrity/scientific validation for representative jobs.

HF native remains one durable controller and is not the topology to certify by replicating that writer. Multi-replica campaigns must use the documented distributed Docker or managed-HF architecture with external shared PostgreSQL and the required storage/lease contracts. Linux/Docker and HF campaigns need separate infrastructure evidence even when common application tests pass. Publish a capacity claim only for the tested workload, resources, limits and failure assumptions. A target of hundreds of thousands of concurrently active users is a deployment acceptance requirement, not a release-unit-test result.

Existing HF installation: upgrade and recovery

For an already running native HF deployment, follow section 11 of the native HF guide (HUGGING_FACE_NATIVE_DEPLOYMENT.md#11-upgrade-an-existing-r34-deployment). It supplies the complete Windows CMD and HF website sequence. This companion summarizes the preservation and acceptance requirements. New installations start at section 1 of that guide and use the full r34 release. The concise current upgrade route is R34_NATIVE_HF_RELEASE_UPGRADE.md (R34_NATIVE_HF_RELEASE_UPGRADE.md). The consolidated release includes all approved native fixes plus shared changes; there is no historical patch installation chain.

1. Record the current installation

Where: secure operator records and the HF website. Record the controller, private code and public runtime Space names, state Bucket and prefix, immutable runtime digest, private code commit and code SHA256, controller commit, and current configuration. Use the deployment record template (HUGGING_FACE_DEPLOYMENT_RECORD.md). Keep existing database, session, state-encryption and BYOK/BYOS keys in your private secret store. HF does not reveal secret values after they are saved.

Preserve the existing Bucket, prefix and stable secrets throughout an upgrade. Keep `PF_HF_BOOTSTRAP` absent for an initialized service. Preserve the approved `PF_HF_MAX_STORAGE_BYTES` value (16 GiB is `17179869184`) and encrypted quota reservations/usage. Never lower it to a fresh-install example or reset the meter. The new code restores and migrates its existing database; creating a new prefix or keys is a different installation and does not recover the old one.

2. Pause and make a verified recovery copy

Where: PF/HF website, then Windows CMD. Close admission to new work, allow active runs to finish or cancel them, and verify their terminal provider state. Confirm that there is no other controller using the same Bucket and prefix. Pause the controller and require `PAUSED` before copying durable state.

Where: HF website. Create a new, empty, private backup Bucket in the source Bucket's storage region. Choose a unique name for each maintenance window. `pf-simulation-state-backup-20261001` is an example name, not a required name or a Bucket to reuse. Give the local HF token read access to the source and read and write access to this destination, while retaining its deployment scopes.

Where: Windows CMD. Use the guide's HF Bucket-to-Bucket copy and verification commands. Keep the controller paused until source, destination and a second source listing agree on every file path, byte length and Xet hash, the source is nonempty, and the destination is private. Record the file and byte counts. The backup uses HF authentication and requires no additional S3 configuration. Preserve the existing runtime storage credentials.

A verified copy is a recovery checkpoint. It does not prove that a restore into a separate environment has succeeded; record any restore drill separately.

3. Generate and qualify the replacement

Where: Windows CMD with the tools environment active. Verify the full archive against its accompanying SHA256 file and extract it into a clean directory. Keep the preceding release and its upload folders available. Use the packaged `deploy\saas\huggingface_native\build_space.py` with a new output directory; it generates `runtime`, `jobs` and `space` together. Check that the private code manifest and controller base-code manifest report the same generated SHA256.

Use the packaged `deploy_release.py upload-code` command for the generated `jobs/` contents in the existing private code Space. Record its exact returned `code_revision` and `code_sha256`; it verifies the committed remote inventory. Use those exact identities with the compatible, qualified, immutable public runtime image. Use the generated runtime context to establish the dependency contract, including FMU and QuTiP. Build and qualify a changed public runtime and record its new digest. An unchanged compatible runtime may be retained only when its context and image provenance are verified as described in section 11.4.

Preview the pinned qualification specification and then submit the reviewed self-test. `--submit` starts a billable HF Job. If a Job ID is returned, inspect that existing Job before considering another submission. Require completion and both `confinement_probe_passed=true` and `scientific_runtime_probe_passed=true` for the newly uploaded code. A passing self-test is one acceptance item, not a full SaaS or restore qualification.

4. Apply matching controller settings and code

Where: HF website or the explicit HF CLI commands in section 11. Windows CMD `set` commands only configure the local terminal. Save the new private-code repository, commit and SHA256 in the controller's HF **Variables**, retain its qualified runtime image, and set or retain the bounded receiver-startup setting. The default `PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS` is 600 seconds; its allowed range is 30-3600. Verify the saved non-secret pins before uploading.

Keep the existing state binding and stable secrets, with `PF_HF_BOOTSTRAP` absent. Use the packaged `deploy_release.py verify`, `prepare`, `apply`, then `status` sequence for the matching generated bundle. Preparation verifies exact code pins; apply confirms pause before a parent-guarded commit, removes only recognized obsolete release-owned files and restarts. It refuses unexpected files or changed public Variables/revisions. Settings may themselves trigger provider restarts, so keep researcher access closed throughout maintenance. Inspect `status` before retrying an uncertain write; it can reconcile an exact committed bundle without repeating the mutation.

5. Verify recovery and reopen access

Require `RUNNING`, `Application startup complete` and the HTTP server startup message. Then open the actual direct `.hf.space` address in a normal browser tab. Sign in with the existing PF administrator account, open a previously saved project and download a known artifact. Run a small simulation through result collection, inspect its plots/report and verify a newly saved record survives a controlled restart. Test cancellation on a separate bounded run and apply the remaining authorization and operational acceptance checks in the guide.

Reopen researcher access only after the applicable checks pass. Use the guide's troubleshooting section if build or startup does not finish successfully; retain the state and configuration while identifying the failing phase.

Recovery and rollback

Retain the pre-upgrade backup and its matching code/configuration records. The consolidated shared application schema is **10**, upgraded during native startup. A schema-9 or older controller cannot run against it. Reverting a controller commit alone does not reverse that migration. After startup has migrated data, use a reviewed forward repair or restore a separately verified pre-upgrade state with its complete matching old configuration/code. Account explicitly for writes after the backup; do not present code checkout as data rollback. Validate recovery in a separate isolated installation with the recorded compatible code and keys, without running a second writer against the production Bucket and prefix.

The full release includes those earlier compatibility fixes and the approved native latency, startup, response, runtime, WAL batching, ownership queue and persistence response recovery corrections. Follow the current full-release sequence rather than assembling those corrections separately.

Native HF deployment record template

Keep one private record for each deployment or upgrade. Follow the native deployment guide (HUGGING_FACE_NATIVE_DEPLOYMENT.md); for an existing running r34 installation, start at section 11 (HUGGING_FACE_NATIVE_DEPLOYMENT.md#11-upgrade-an-existing-r34-deployment). This template records the successful sequence and acceptance evidence for your installation. It contains no deployment-specific credentials or default pins.

Release and environment

Item	Record
Maintenance date/time and operator	Your values, including timezone
Full release filename and verified archive SHA256	From the delivered archive and accompanying checksum
Previous release and rollback reference	Matching prior archive, configuration and backup record
Local extraction directory	Clean full-release directory
Generated upload directory	New output containing matching `runtime/`, `jobs/`, `space/`
Tools environment	Python and `huggingface_hub` versions
HF owner and controller Space	Your namespace and repository
Public dependency-runtime Space	Your public dependency-only repository
Private code Space	Your private repository
Production state Bucket and prefix	Existing binding, preserved during upgrades
Direct application URL	Actual `.hf.space` address from HF

Keep database passwords, HF credentials, session/state encryption keys, BYOK/BYOS keys and SMTP passwords in your private secret store, separately from this record. Record secret names or secure references only.

Existing-deployment checkpoint

Complete these items before changing production code or pins. For a new installation, mark this section not applicable and record first initialization under the guide's clean-install procedure.

Check	Record
Researcher admission closed	Time and method
Active runs completed or cancelled	Provider terminal states confirmed
Controller paused	`PAUSED` confirmed before and after the copy
Other writers	No second controller using the same state Bucket/prefix
Private backup destination	New empty private HF Bucket, unique name and matching region
Token access	Source read, destination read/write, existing deployment access retained
Copy	Successful HF Bucket-to-Bucket copy
Manifest verification	`BACKUP MANIFEST VERIFIED`, matching paths/sizes/Xet hashes, file and byte counts

Check	Record
Stable configuration preserved	Same production Bucket/prefix and existing secret references
Bootstrap setting	PF_HF_BOOTSTRAP absent

The backup Bucket name is an operator choice. For example, `pf-simulation-state-backup-20261001` identifies one maintenance date; choose a fresh name for another upgrade. This copy needs no additional S3 setup. A manifest verification is distinct from a separately tested restore.

Replacement identities

These values identify different artifacts; record each in its own field. Read new values from the generated manifests and actual provider responses.

Identity	Record
Public runtime image	`registry.hf.space/…@sha256:…` verified immutable digest
Private code archive SHA256	`jobs/code-manifest.json` field `sha256`
Controller base-code SHA256	Matching `space/base-code-manifest.json` field `sha256`
Private code repository commit	Exact 40-character uploaded commit
Qualification Job	Job ID, URL, inspected identity, terminal stage and result
Saved controller pins	Read-back matches newly qualified repository/commit/code hash/runtime
Receiver startup deadline	Recorded value or default 600 seconds
Controller repository commit	Commit containing matching generated `space/` contents

A full-release archive SHA256 is not the private-code SHA256, a Git commit or an OCI runtime digest. A new full-release build can produce a new private payload; qualify the exact generated code that will be deployed.

Acceptance record

Record pass, fail or pending, the date and evidence for every applicable item.

Check	Evidence to retain
Qualified code self-test	Completed Job and passing confinement result for recorded pins
Controller container build	Successful build for the recorded controller commit
Application startup	`RUNNING`, application startup complete and server listening
Restored access	Existing administrator login and previous project/artifact access
Simulation	Small real run completes through result collection and plots/report
New durable write	Saved record survives a controlled restart
Cancellation	Separate bounded run reaches confirmed terminal provider state
Private workflows and BYOK/BYOS	Applicable configured paths checked
Authorization	Applicable project/tenant boundaries checked
Email and invitations	End-to-end check if configured

Check	Evidence to retain
Recovery	Separate restore drill and compatibility result, when performed
Researcher access reopened	Time and approving operator

Record the confinement probe's declared scope accurately. A successful Job or web-server startup does not by itself establish complete SaaS security, restore, scientific-workload, concurrency or billing qualification. Mark unperformed checks pending rather than inferring that they passed.

Packaged operational entry points

Packaged path	Purpose
`deploy/saas/huggingface_native/build_space.py`	Canonical Windows-safe three-context builder with controller manifest inclusion
`deploy/saas/huggingface_native/build_space_windows_fix.py`	Compatibility entry point delegating to the same current builder
`deploy/saas/huggingface_native/hf_registry_digest.py`	Immutable public runtime digest lookup, anonymous verification and optional system CA trust
`deploy/saas/huggingface_native/qualify.py`	Preview/submit the pinned qualification Job and verify its provider-reported identity

Use the scripts included with the extracted full release. The main guide and its packaged PDF provide the current commands, troubleshooting and later existing-deployment upgrade route.

Consolidated r34 acceptance addendum

Item	Record
Application schema	10 after startup; prior schema-9 code is not a code-only rollback
Deployment profile	Native HF `single`, or separately qualified optional `hf_managed`
Guarded preparation	Path to `deployment.json`, parent revision and cleanup list
Exact controller verification	`source_bytes_verified`, expected/running revision and status timestamp
Existing quota preservation	Configured limit, encrypted usage/reservation history retained; no reset/refund
Archive/restore response	Request and visible UI completion timings for organization and project
Shared capacity settings	SQL/assistant connection budgets, API/AI limits, queue, worker and recovery bounds
Evidence separation	Offline regression report, prior user-reported native result, new live acceptance result
Recovery boundary	Verified pre-upgrade state copy and complete matching old code/configuration

r34 Hugging Face Job identity verification

Consolidated-release note: this document records an earlier r34 correction.

Its retained technical behavior is included in the full release. Use

the current native upgrade sequence (R34_NATIVE_HF_RELEASE_UPGRADE.md) for

deployment and schema-10 recovery rules; do not reinstall a historical patch.

The qualification helper and normal simulation Job verifier accept `None`, `{}` or `[]` as an empty provider `secrets` field. Nonempty secret containers and unexpected values remain invalid. Verification also checks the submitted image, private-code identity, environment, read-only code mount, network configuration and exposed endpoints. No service credentials are supplied to scientific Jobs.

Deployment and qualification

Build private `jobs` and controller `space` contexts together from the complete release. Record the new private upload's exact commit and generated content hash, then qualify that pair with its compatible dependency-only runtime digest. Apply the matching controller settings and context using section 11 of the native deployment guide (HUGGING_FACE_NATIVE_DEPLOYMENT.md#11-upgrade-an-existing-r34-deployment). Preserve the existing durable state, Bucket configuration and encryption keys.

If submission returns a Job ID but later verification is uncertain, inspect that same Job before submitting again. Provider completion and successful verification of the expected identity are both required. A result for one private-code revision does not qualify a later revision.

Acceptance scope

Provider-contract fixtures check accepted empty representations, malformed responses and secret-bearing responses. Their receipts identify the exact code and runtime tested. Such fixtures do not establish live service acceptance, cross-tenant security, database restoration or scientific benchmark results. Complete the guide's applicable deployment acceptance checks separately.

r34 Hugging Face PostgreSQL readiness and controller inputs

Consolidated-release note: this document records an earlier r34 correction.

Its retained technical behavior is included in the full release. Use

the current native upgrade sequence (R34_NATIVE_HF_RELEASE_UPGRADE.md) for

deployment and schema-10 recovery rules; do not reinstall a historical patch.

Restored PostgreSQL startup

A successful `pg_ctl start` does not establish application readiness while a restored cluster remains in standby recovery. The native launcher waits for PostgreSQL's process status to reach `ready` before opening its first SQL connection. The wait shares the configured startup deadline (600 seconds by default), checks process identity and treats disappearance, malformed status or an expired deadline as startup failure. Startup cleanup stops the affected process. Database recreation, recovery bypass and replacement keys are not part of this procedure.

Durability, replay and synchronous commit checks remain required after process readiness. A restored database is not accepted merely because its process exists. See WAL catch-up and batching (R34_HF_WAL_CATCHUP_AND_BATCHING.md) for the separate receiver readiness requirement and startup bound.

Generated controller Docker context

The controller Dockerfile copies `base-code-manifest.json`, generated alongside its matching private code payload. The canonical builder includes this root-level input in the private controller context and records all generated inputs in the bundle manifest.

Generate private code and controller contexts together from the same complete release. The controller's inventory and configured code hash must match the private payload. A compatible qualified public runtime digest can be retained when its generated dependency-only context is unchanged; the newly built private payload still needs its own recorded hash, immutable commit and qualification.

Use a clean extraction and output directory, retain a verified stopped-state backup and stable deployment configuration, and follow section 11 of the native deployment guide (HUGGING_FACE_NATIVE_DEPLOYMENT.md#11-upgrade-an-existing-r34-deployment). No new S3 configuration or first-time bootstrap is needed for an existing service.

Acceptance scope

Software fixtures cover bounded readiness transitions, failure handling, controller input inclusion and matching bundle inventories. Their receipts remain bound to the tested source. Local fixtures and separately recorded PostgreSQL exercises do not establish complete HF production acceptance. Check restored accounts and projects, durable writes, a representative simulation, cancellation and the applicable authorization boundaries after deployment. A successful build or confinement probe alone is not full deployment acceptance.

r34 Hugging Face WAL catch-up and batching

Consolidated-release note: this document records an earlier r34 correction.

Its retained technical behavior is included in the full release. Use

the current native upgrade sequence (R34_NATIVE_HF_RELEASE_UPGRADE.md) for

deployment and schema-10 recovery rules; do not reinstall a historical patch.

The native controller's durable WAL receiver batches contiguous write-ahead log bytes while retaining synchronous durable acknowledgement and single-writer ownership checks. Restored checkpoints can require receiver catch-up before the application is admitted.

Receiver behavior

The receiver collects contiguous WAL bytes in bounded batches, up to 4 MiB, with a 100-millisecond collection limit. An idle input stream still flushes when that short collection deadline arrives, so a lone small transaction does not depend on another message. Complete-record verification and remote publication occur for the batch rather than each individual network message. Remote verification and publication time is additional; the collection window is not a promise of a completed durable write within 100 milliseconds.

The safety conditions are unchanged: receiver feedback acknowledges only the complete-record boundary confirmed by durable publication. Buffered bytes, the primary's advertised WAL end and an uploaded but unpublished object do not authorize feedback. Gaps, overlaps, source identity or timeline mismatches, lost ownership and storage failures still prevent a successful acknowledgement. Cancellation does not turn unacknowledged buffered bytes into durable progress.

Controller startup still requires one receiver in `streaming` and `sync` state whose flush position covers the confirmed durable acknowledgement. A connected receiver in `catchup` or `potential` state does not pass readiness.

Bounded startup setting

`PF_HF_WAL_RECEIVER_START_TIMEOUT_SECONDS` controls the separate receiver startup wait. Its default is **600 seconds**, and accepted values are integers from **30 through 3600**. On HF, this is an optional controller Space variable; leaving it unset uses the default. It does not change `PF_HF_PG_START_TIMEOUT_SECONDS`, the PostgreSQL process startup budget, or a simulation's execution limit.

The receiver wait remains bounded. Increasing this limit gives healthy catch-up more time; it does not bypass failed publication, permit asynchronous operation or waive the readiness checks. Invalid values are rejected.

Existing deployment

Follow section 11 of the native deployment guide (HUGGING_FACE_NATIVE_DEPLOYMENT.md#11-upgrade-an-existing-r34-deployment) for an already running service. It covers pausing, creating and verifying a private HF Bucket copy, and replacing code while preserving existing stable keys, Bucket configuration and state. This maintenance needs no new S3 configuration, database recreation or first-time bootstrap. Compare the current generated runtime context with the retained qualified one; rebuild and qualify it when the context or image provenance differs. Pin the resulting immutable digest.

Use a fresh extraction of the full release; its receiver implementation is already included. Generate new private `jobs` and controller `space` contexts together. Upload the private payload, record its exact commit and new content hash, qualify those identities, then update the controller pins and upload the matching controller context. A previously generated private-code hash must not be reused after these source changes.

After restarting, wait for receiver catch-up and normal application admission. Check restored accounts and projects, a small real simulation, durable writes and the existing cancellation and authorization acceptance checks. If startup still fails, inspect the same attempt before repeatedly restarting it or replacing configuration.

Acceptance scope

Software fixtures cover batching limits, idle flushes, contiguous ordering, durable feedback, failure/cancellation handling and startup setting validation. Their commands and results remain bound to their recorded source identities. Mock storage latency and request-count tests do not measure live HF Bucket throughput or establish PostgreSQL recovery. Consult the current release record for newly executed checks; complete deployment recovery and live acceptance separately against the exact deployed code.

r34 result publication and verification

Consolidated-release note: this document records an earlier r34 correction.

Its retained technical behavior is included in the full release. Use

the current native upgrade sequence (R34_NATIVE_HF_RELEASE_UPGRADE.md) for

deployment and schema-10 recovery rules; do not reinstall a historical patch.

Saved-result behavior

The shared SaaS worker submits an artifact inventory through a common publication interface. Adapters can implement batch storage; individual-file adapters retain their normal routing. Native HF storage combines file bytes into bounded, verified packs and publishes the logical file descriptors together. Individual reports, figures, CSV files and other downloads retain their original names and access controls. Existing individually stored artifacts remain readable.

Packing reduces publication operations but has a read tradeoff: accessing a small file can fetch its entire containing pack. Repeated individual downloads may therefore transfer more bytes than an individual-file representation. Publication request counts and read amplification are separate performance measures.

The worker maintains its lease during result publication and records bounded failure codes, operation stages, publication counters and elapsed time. Exception messages, signed URLs, database connection strings and credentials are excluded from those diagnostics. Database finalization, cancellation, ownership loss and uncertain cleanup are checked before results can be exposed. Only a successfully committed final attempt can expose completed scientific results.

Collection progress distinguishes result transfer, result import and result publication. Byte totals can differ between compressed transfer and expanded outputs. Finishing one counter does not mean that the entire run has completed. Use the final PF run status and access to saved results as the acceptance checks.

Storage selection and BYOS routing are preserved. Native HF pack optimization applies to that adapter; the common interface does not claim the same remote operation reduction for every provider. Scientific equations, benchmark inputs and modeled results are unchanged by the publication mechanism.

Deployment and collection limits

For an existing deployment, follow section 11 of the native deployment guide (HUGGING_FACE_NATIVE_DEPLOYMENT.md#11-upgrade-an-existing-r34-deployment) using a clean extraction of the complete release. It already includes the publication implementation. Build the controller and private code together, qualify the exact new private-code revision and hash, and deploy their matching contexts and pins. Preserve the existing Bucket, prefix, keys and account records; leave `PF_HF_BOOTSTRAP` absent. No separate patch or diagnostic installation is required. A compatible qualified public runtime digest can be retained under section 11.4.

`PF_SAAS_JOB_COLLECT_SECONDS` bounds collection work separately from scientific execution. Its default is 300 seconds. An operator can retain or select a larger bounded value, such as 900, after checking representative output sizes and storage latency. This does not resolve permission, quota, integrity, lost-lease or durable publication errors. Preserve a justified existing value during an upgrade rather than increasing it automatically.

Native HF packs use descriptors that the current release reads alongside the individual-file format. Older code may not support newly committed packed artifacts. Retain the verified backup and review storage compatibility before recovery to an older release; a code-only rollback is not a storage rollback.

Verify a representative workload

1. Require `RUNNING` and `Application startup complete` after deployment. Open the workbench and check existing organizations, projects and retained results.
2. Submit one small representative simulation within the deployment's remaining compute, concurrency and storage allowances. Record its configuration and IDs.

3. Observe collection through transfer, import and publication. A full byte counter is an intermediate checkpoint; wait for the final PF status.
4. Require **Completed**. Open the report and figures, download the run ZIP and representative JSON/CSV files, and confirm that the files open as expected. Verify that the saved run remains accessible after reloading the application.
5. Record the run/attempt IDs, final status, elapsed time, cleanup state, release checksum, private-code hash, runner commit and controller commit.

A single completed workload establishes that workflow's observed execution and result access for its recorded deployment. It does not qualify all workflows, all storage providers, production security or crash recovery.

Troubleshoot publication

Capture the run and attempt fields `status`, `returncode`, `error`, `phase`, `progress`, `wall_elapsed_seconds`, `cleanup_pending` and `cleanup_state`, along with the matching safe controller log event. Keep credentials and private configuration out of shared diagnostics. An `ARTIFACT_...` code and stage identify the category and operation to investigate; they do not establish the provider's root cause by themselves.

Diagnostic category	Meaning
Deadline or timeout	Collection deadline or an individual storage call timed out.
Quota	A checked storage allowance rejected publication.
Integrity	Source bytes, uploaded bytes or descriptors did not match.
Ownership or durability	Publication authority or durable confirmation was lost.
Database	A database operation could not be completed.
Permission, connection or provider	A storage access or provider operation failed.
Checkpoint	A background storage checkpoint could not be confirmed.
Unknown	Inspect the operation stage and controller state without inferring a provider cause.

Review cleanup against the recorded attempt and its original storage destination. An uncertain deletion retains a conservative storage charge until the applicable cleanup/accounting policy permits release. A reservation is not a measurement of saved scientific output. A successful new attempt does not prove that every older cleanup record has settled. Do not remove backend objects or accounting records manually to make the displayed status or allowance change.

Validation scope

Software validation receipts record their executed storage integrity, fault, worker lifecycle, progress-interface and release-consistency checks against exact source identities. Local workload timings and emulated provider-operation counts are not live HF measurements or a new scientific qualification. Historical receipts retain their original scope; consult the current release record for freshly executed checks. Live deployment, native Windows and provider acceptance remain separate recorded qualifications.