CHECKLIST

Secure ML Model Deployment Checklist

A practical checklist for hardening the MLOps pipeline so model artifacts, dependencies, approvals, and runtime controls stay trustworthy from training to release.

Secure ML Model Deployment Checklist

Use this checklist to verify that a model release is trustworthy from source intake through production rollback. Mark each item Yes, No, or N/A.

---

1) Inputs and source control

- [ ] Source code is reviewed before training or packaging
- [ ] Training data sources are approved and documented
- [ ] Feature definitions and preprocessing logic are version-controlled
- [ ] Configuration files are stored in source control or a controlled config system
- [ ] Branch protection prevents direct changes to protected branches
- [ ] Secrets are excluded from notebooks, code, and manifests
- [ ] Secret scanning is enabled in repositories and CI pipelines
- [ ] Only authorized users can trigger training or release workflows

Evidence to capture: commit hash, data version, config version, reviewer names, access policy references.

---

2) Build and training environment

- [ ] Training and packaging jobs run in isolated environments
- [ ] Jobs use least-privilege service accounts or identities
- [ ] Training jobs do not have standing access to production systems
- [ ] Packaging jobs have only the permissions they need
- [ ] Build images or base environments are pinned to approved versions
- [ ] Dependency versions are locked or otherwise reproducible
- [ ] Shared caches and mutable host state are minimized
- [ ] Environment changes are tracked and auditable

Evidence to capture: job identity, container image digest, dependency lockfile, environment manifest, access scope.

---

3) Model artifact integrity

- [ ] Artifacts are stored in a controlled registry or artifact store
- [ ] Artifact names are not the only identifier used for promotion
- [ ] Each model artifact has a unique digest or checksum
- [ ] Artifacts are immutable after publication
- [ ] Artifact overwrite or silent retagging is prevented
- [ ] Artifact signing or integrity verification is enabled
- [ ] Build metadata is retained with the artifact
- [ ] Retention policies preserve release candidates long enough for audit and rollback

Evidence to capture: artifact digest, registry location, signature status, metadata record, retention policy.

---

4) Provenance and traceability

- [ ] The release record links model to source commit
- [ ] The release record links model to training data version
- [ ] The release record links model to environment and dependency versions
- [ ] The release record links model to training parameters or pipeline run ID
- [ ] The release record shows who initiated and who approved promotion
- [ ] Provenance is retrievable without manual reconstruction
- [ ] A reviewer can identify exactly which inputs produced the deployed model

Evidence to capture: pipeline run ID, source commit, data lineage, parameter set, approval log.

---

5) Validation gates

- [ ] Model quality checks run before promotion
- [ ] Validation thresholds are defined in advance
- [ ] Relevant slice or segment checks run where needed
- [ ] Security checks are included where applicable
- [ ] Policy checks confirm the model meets deployment criteria
- [ ] Validation results are stored and attached to the release record
- [ ] Promotion is blocked when required checks fail
- [ ] Exceptions require documented approval and justification

Evidence to capture: validation report, threshold definitions, exception record, test summary, policy outcome.

---

6) Approval and promotion control

- [ ] Promotion requires explicit approval
- [ ] Approvers can see validation evidence before approving
- [ ] Approvers can see artifact identity and target environment
- [ ] Approval is tied to the specific release candidate
- [ ] ?Latest successful run? is not the only promotion rule
- [ ] Manual approvals are meaningful, not ceremonial
- [ ] Promotion paths are documented and enforced
- [ ] Unauthorized promotion is blocked

Evidence to capture: approval log, approver identity, release candidate ID, target environment, promotion policy.

---

7) Deployment and runtime controls

- [ ] Deployment uses a controlled release channel
- [ ] Production access is limited to approved automation and operators
- [ ] Request authentication is enabled where the model is exposed through an API
- [ ] Runtime telemetry is collected for requests, errors, and latency
- [ ] Anomaly detection or alerting is configured for suspicious behavior
- [ ] Traffic shaping, canarying, or staged rollout is used where appropriate
- [ ] Production secrets are stored outside the model artifact
- [ ] Runtime configuration is separate from training configuration

Evidence to capture: deployment record, service account, telemetry dashboard, alert policy, rollout strategy.

---

8) Rollback and recovery

- [ ] A known-good previous version is available for rollback
- [ ] Rollback steps are documented and tested
- [ ] Rollback does not require manual artifact reconstruction
- [ ] Deployment metadata makes reversion fast and unambiguous
- [ ] The team knows who can initiate emergency rollback
- [ ] Rollback criteria are defined before release
- [ ] Incident response steps include model release revocation when needed
- [ ] Post-rollback review captures what changed and why

Evidence to capture: rollback procedure, last known-good artifact, deployment history, emergency contacts, incident playbook.

---

9) Operational readiness review

Answer the following before production release:

- Can we prove this exact model was built from approved inputs?
- [ ] Yes
- [ ] No
- Can we show what validations ran and what their results were?
- [ ] Yes
- [ ] No
- Can we identify who approved the release and why?
- [ ] Yes
- [ ] No
- Can we roll back quickly if evidence changes?

- [ ] Yes
- [ ] No
- Is runtime monitoring in place to detect unexpected behavior?
- [ ] Yes
- [ ] No

If any answer is No, treat the deployment as not production-ready until the gap is addressed.

---

10) Production-readiness decision

Deploy if all of the following are true:

- [ ] Inputs are controlled and reviewed
- [ ] Training and packaging environments are isolated and least-privileged
- [ ] Artifacts are immutable and identifiable by digest
- [ ] Provenance is recorded and retrievable
- [ ] Validation gates passed with documented evidence
- [ ] Approval was explicit and based on evidence
- [ ] Runtime monitoring and rollback are ready

Do not deploy if any of the following are true:

- [ ] Artifact identity is unclear
- [ ] Inputs are unreviewed or untrusted
- [ ] Validation evidence is missing
- [ ] Approval was informal or undocumented

- [ ] Dependencies or environments are not reproducible
- [ ] Rollback is untested or too slow for the risk level

---

11) Release record template

Use this short template to document each release:

- Model name:
- Version or digest:
- Source commit:
- Training data version:
- Environment or image digest:
- Key parameters:
- Validation checks passed:
- Approver:
- Approval date:
- Target environment:
- Rollback version:
- Monitoring owner:

---

12) Final release gate

Before go-live, confirm:

- [ ] The model artifact is the exact one that was validated
- [ ] The promotion path is documented and approved
- [ ] The runtime environment is ready and monitored
- [ ] The rollback plan is available and tested

- [ ] Security, quality, and policy checks are complete

If all boxes are checked, the deployment is ready for production review.

Notes

- This checklist is vendor-neutral and can be applied to notebooks, batch training jobs, CI/CD pipelines, model registries, and inference services.
- For higher-risk or regulated use cases, require stronger evidence and tighter approvals.
- Pair pipeline hardening with runtime controls so you reduce both the chance of a bad model reaching production and the impact if it does.