CHECKLIST

Zero Trust Inference Pipeline Readiness Checklist

A practical checklist for securing AI model inference pipelines with zero trust access, focused on identity, authorization, transport protection, observability, and production verification.

Zero Trust Inference Pipeline Readiness Checklist

Use this checklist to validate whether an AI inference pipeline is ready for production under a zero trust access model. It is designed for model-serving systems that include API gateways, internal services, feature stores, orchestration layers, logs, and downstream consumers.

How to use this checklist

- Review each item against your current architecture.
- Mark items as Yes, No, or N/A.
- Treat any No on a critical control as a production blocker.
- Revisit the checklist after major model, infrastructure, or policy changes.

---

1) Identity and authentication

- [ ] Every human operator has a unique identity for accessing inference systems.
- [ ] Every service, workload, or automation job has a distinct non-shared identity.

- [ ] No production component relies on shared static credentials.
- [ ] Authentication is required for inbound client requests.
- [ ] Authentication is required for internal service-to-service calls.
- [ ] Service identities can be traced to a specific workload, namespace, deployment, or host.
- [ ] Secrets, tokens, or certificates are rotated on a defined schedule.
- [ ] Credential revocation can be performed quickly without broad service disruption.

Verify before production

- Can you identify the caller of every inference request?
- Can you revoke one compromised workload without affecting unrelated services?
- Can you prove that authentication cannot be bypassed through an internal network path?

---

2) Authorization and policy enforcement

- [ ] Authorization is enforced independently from authentication.
- [ ] Access rules are specific to the model, route, environment, and action.
- [ ] Policies distinguish between read, invoke, admin, and debug operations.
- [ ] Tenant-specific or environment-specific restrictions are enforced where applicable.
- [ ] Canary, staging, and production models are separated by policy, not just naming.
- [ ] Internal dependencies such as feature stores or retrieval services are least-privileged.
- [ ] Authorization is evaluated close to the request, not only at login or session creation.
- [ ] Policy changes are versioned, reviewed, and auditable.

Verify before production

- Does a service have access only to the exact model endpoint it needs?
- Can a workload call one model but not another with the same credentials?
- Are permissions narrow enough to prevent cross-environment access?

---

3) Transport protection and service trust

- [ ] All inference traffic is protected in transit.
- [ ] Mutual TLS or an equivalent mechanism is enforced for service-to-service trust.
- [ ] Certificate or key rotation is documented and tested.
- [ ] Connections cannot silently downgrade to weaker transport protection.
- [ ] Internal network location is not treated as sufficient trust.
- [ ] Service mesh, sidecars, gateways, or proxies do not introduce unreviewed trust gaps.
- [ ] Dependencies do not trust the caller based only on subnet, VPC, or cluster membership.

Verify before production

- Can an internal attacker or compromised pod impersonate a trusted service?
- Is transport protection validated on every hop, including between internal components?
- Are expired or revoked certificates blocked reliably?

---

4) Request-level controls

- [ ] Requests are inspected individually before they reach the model server.
- [ ] Request context such as caller identity, route, model, environment, and risk signals is available for policy decisions.
- [ ] Rate limits or quota controls exist for sensitive endpoints.

- [ ] Replay protection is in place where repeated requests would be harmful.
- [ ] Only approved methods and routes are exposed.
- [ ] Debug, trace, or introspection endpoints are disabled or tightly restricted.
- [ ] Bulk extraction patterns can be detected and throttled.

Verify before production

- Can the system stop abnormal request volume from a single identity?
- Are policy decisions based on the specific request, not just the source network?
- Are model endpoints protected from unauthorized enumeration?

---

5) Data exposure boundaries

- [ ] Inputs, features, outputs, and intermediate artifacts are classified by sensitivity.
- [ ] Sensitive data is minimized at every step of the inference path.
- [ ] The model service receives only the inputs it needs.
- [ ] Feature store access is limited to approved fields and records.
- [ ] Post-processing services do not receive raw inputs unless required.
- [ ] Logs do not store sensitive payloads by default.
- [ ] Output handling rules prevent accidental propagation of restricted data.
- [ ] Downstream systems receive only the data required for their function.

Verify before production

- Could a compromised downstream service exfiltrate raw inputs or features?
- Are model outputs filtered before being written to shared systems?
- Do logs, traces, or analytics pipelines overexpose inference content?

---

6) Logging, auditability, and monitoring

- [ ] Every access decision is logged.
- [ ] Logs include caller identity, target service, policy outcome, and request context.
- [ ] Logs are sufficient to reconstruct who accessed which model and when.
- [ ] Unauthorized requests are distinguishable from successful requests.
- [ ] Logs are protected from tampering and unauthorized access.
- [ ] Monitoring exists for unusual access patterns and policy denials.
- [ ] Alerts are tuned for both abuse and misconfiguration.
- [ ] Access logs are retained according to operational and compliance requirements.

Verify before production

- Can you investigate a suspicious inference request end to end?
- Can you correlate access events across gateway, model service, and downstream systems?
- Can you detect repeated denials that indicate probing or misuse?

---

7) Break-glass and incident response

- [ ] Break-glass access exists for emergency operations.
- [ ] Break-glass access is time-bound.
- [ ] Break-glass access is logged and monitored.
- [ ] Elevated access requires an explicit approval or justification flow.
- [ ] Revocation procedures are tested before an incident occurs.
- [ ] The team knows how to isolate a compromised identity or workload.

- [ ] Incident runbooks include service-to-service trust failures and credential compromise.
- [ ] Recovery steps do not depend on disabling all security controls.

Verify before production

- Is there a safe emergency path that does not create permanent privilege creep?
- Can you contain a compromised inference service quickly?
- Do operators know which components to disable first during an active incident?

---

8) Architecture and boundary review

- [ ] The public edge is separated from internal service trust.
- [ ] External client access and internal workload access use different controls.
- [ ] The model server does not inherit broad trust from the network path.
- [ ] Shared infrastructure does not create unintended cross-team access.
- [ ] Multi-tenant or multi-environment deployments are isolated by policy.
- [ ] Canary and staged models are protected from accidental production access.
- [ ] Dependency boundaries are documented and reviewed.

Verify before production

- Where exactly is the trust boundary for the model endpoint?
- Which services are allowed to cross it, and why?
- What prevents a compromised internal workload from moving laterally?

---

9) Operational readiness checks

- [ ] Identity, policy, and transport controls are covered in deployment documentation.
- [ ] Configuration changes are reviewed before release.
- [ ] Rollback steps preserve access control integrity.
- [ ] Access reviews occur on a regular schedule.
- [ ] Security ownership is clear across platform, ML, and application teams.
- [ ] New models or services cannot bypass baseline controls.
- [ ] Production verification is repeated after significant topology changes.

Verify before production

- Is zero trust enforced consistently across all environments?
- Do new deployments inherit secure defaults automatically?
- Can the team explain the control model during an audit or incident review?

---

10) Production go-live decision

Approve production release only if all of the following are true:

- [ ] Every critical component has a unique identity.
- [ ] Authentication and authorization are enforced on every relevant hop.
- [ ] Transport protection is verified end to end.
- [ ] Sensitive inputs and outputs are classified and protected.
- [ ] Logging and audit trails are sufficient for reconstruction.
- [ ] Break-glass access is controlled and monitored.
- [ ] Revocation and incident containment procedures are tested.
- [ ] The team has reviewed trade-offs such as latency, operability, and monitoring overhead.

Final review questions

- What is the smallest trust boundary that still allows the system to function?
- Which control would most likely fail first under real-world load?
- If one workload is compromised, how far can the attacker move?

Quick red flags

If you see any of these, treat the pipeline as not ready for production:

- Shared credentials across multiple inference services.
- Internal access allowed by subnet or cluster membership alone.
- No policy enforcement at request time.
- Logging that omits caller identity or target model.
- Unrestricted feature store access.
- Debug endpoints exposed to broad audiences.
- No tested revocation path for compromised identities.
- Raw inputs or outputs written into shared logs or analytics systems.

Operational summary

Zero trust for inference pipelines is about making every access decision explicit. The safest production pattern is not to assume the model is hidden behind a private network, but to verify identity, authorization, transport, and data handling at every boundary.

Use this checklist as a launch gate, a review template, and a recurring control audit for your ML serving stack.