CHECKLIST

Spark Pipeline Encryption Readiness Checklist

A practical vendor-neutral checklist for verifying encryption coverage across Apache Spark pipelines, including transport, shuffle, storage, and secret handling.

Spark Pipeline Encryption Readiness Checklist

Use this checklist to verify that an Apache Spark pipeline has encryption coverage across all sensitive data paths. Mark each item Yes, No, or N/A and record evidence where needed.

---

1) Data flow and scope

- [ ] Data paths are mapped: input, driver, executors, shuffle, spill/temp files, output, logs, and secrets.
- [ ] Sensitive data types are identified: regulated records, credentials, customer data, security telemetry, or proprietary data.
- [ ] Trust boundaries are documented for the submitter, cluster runtime, and storage/downstream systems.
- [ ] Encryption requirements are defined by path, not just by application or storage system.
- [ ] Any exceptions are documented with a clear business justification.

Evidence / notes:

---

2) Submission and transport encryption

- [ ] Client-to-cluster traffic uses encrypted transport.
- [ ] Driver-to-executor communication is encrypted.
- [ ] Connections to external services (catalogs, queues, metadata stores, databases, object stores) use encrypted channels where supported.
- [ ] Certificate validation is enabled to prevent interception or downgrade.
- [ ] Plaintext protocols are not allowed for sensitive environments.
- [ ] Connection strings and endpoints are reviewed to ensure they do not bypass secure transport.

Verify:

- [ ] TLS or equivalent is active
- [ ] Certificates are trusted and not expired
- [ ] Weak ciphers/protocols are disabled where configurable

Evidence / notes:

---

3) Shuffle and intermediate data

- [ ] Shuffle traffic between executors is encrypted.
- [ ] Intermediate data used in joins, aggregations, and repartitioning is protected.
- [ ] Sensitive records are not exposed in plain shuffle files or network traffic.
- [ ] Performance impact has been measured after enabling shuffle protection.
- [ ] Failure behavior has been tested if shuffle encryption prerequisites are missing.

Evidence / notes:

---

4) Local disk, spill, and temporary storage

- [ ] Executor local disks are encrypted or otherwise protected by the platform.
- [ ] Spill files and temporary files are covered by encryption controls.
- [ ] Local scratch space does not retain sensitive plaintext after job completion.
- [ ] Temporary paths are reviewed for permissions and cleanup behavior.
- [ ] Multi-tenant host exposure has been assessed.

Evidence / notes:

---

5) Output and persistent storage

- [ ] Data written to object storage, filesystems, or databases is encrypted at rest.
- [ ] The layer responsible for storage encryption is clearly identified: Spark, OS, filesystem, or storage backend.
- [ ] Encryption coverage includes staging areas and backup locations.
- [ ] Key ownership for storage encryption is understood.
- [ ] Downstream sinks enforce encryption independently when required.

Evidence / notes:

---

6) Secret handling and credentials

- [ ] Credentials are not embedded in source code, notebooks, or scripts.
- [ ] Secrets are not logged in clear text.
- [ ] Environment variables do not expose sensitive values unnecessarily.
- [ ] Secrets are stored in a managed secret system or equivalent protected store.

- [ ] Access to secrets is restricted to the minimum required identities.
- [ ] Secret rotation is documented and tested.

Evidence / notes:

---

7) Key management controls

- [ ] Key source is known for each encrypted layer.
- [ ] Key rotation method is documented.
- [ ] Key access is restricted and audited.
- [ ] Certificate and key expiry is monitored.
- [ ] Recovery steps exist for missing, invalid, or rotated keys.
- [ ] Emergency access procedures are approved.

Evidence / notes:

---

8) Validation and verification

- [ ] A controlled test job was run to confirm encryption settings are active.
- [ ] Logs or configuration output confirm encryption is enabled.
- [ ] Negative testing was performed by simulating a missing or invalid certificate/key.
- [ ] Results were checked at each boundary, not just in one location.
- [ ] Verification evidence is stored with the deployment record.

Suggested verification methods:

- Inspect Spark and cluster configuration
- Review runtime logs for encryption-related settings
- Confirm storage backend encryption status
- Check connection policies and certificate trust paths

- Run a small controlled job with known test data

Evidence / notes:

---

9) Performance and operational impact

- [ ] CPU overhead from encryption has been measured.
- [ ] Job latency and throughput are acceptable after encryption is enabled.
- [ ] Shuffle-heavy workloads were tested separately if relevant.
- [ ] Operational alerts exist for certificate/key failures.
- [ ] Troubleshooting procedures include encryption-related checks.
- [ ] Rollback or fallback steps are documented.

Evidence / notes:

---

10) Production readiness

- [ ] All required paths are encrypted or formally accepted as exceptions.
- [ ] Ownership is assigned for each encryption layer.
- [ ] Monitoring covers certificates, keys, and secure transport status.
- [ ] Compliance requirements have been reviewed and met.
- [ ] Runbooks include encryption validation steps.
- [ ] The deployment has been approved for production use.

Go-live decision:

- [ ] Ready
- [ ] Ready with exceptions
- [ ] Not ready

Approver: _____ Date: _____

Quick final review

Before rollout, confirm these four questions:

- Is every sensitive data path protected?
- Do you know which layer owns each encryption control?
- Can you prove encryption is active, not just configured?
- Are keys, certificates, and secrets managed securely?

If the answer to any question is no, fix the gap before production deployment.