OSO Kafka Backup

Well-Architected Framework

A comprehensive guide to designing, deploying, and operating production-grade Kafka backup architectures

Version 1.0.0 — March 2026

OSO — oso.sh

Well-Architected Framework

> **DOWNLOAD AS PDF**
>
> A printable version of this framework is available for offline reference and stakeholder review. Download the Well-Architected Framework PDF

> *The OSO Kafka Backup Well-Architected Framework is modelled on the principles of the AWS Well-Architected Framework, Azure Well-Architected Framework, and Google Cloud Architecture Framework. It adapts these proven methodologies specifically to the domain of Apache Kafka backup, disaster recovery, and data protection using OSO Kafka Backup.*

Introduction

Purpose

Apache Kafka has become the backbone of modern event-driven architectures, handling mission-critical data streams across financial services, healthcare, e-commerce, and beyond. Yet backup and disaster recovery for Kafka remains one of the most under-addressed areas of platform engineering. When a cluster failure, misconfiguration, or data corruption event occurs, organisations without a robust backup strategy face permanent data loss, extended outages, and regulatory exposure.

Traditional backup approaches designed for databases and file systems fail when applied to streaming platforms. Kafka's append-only log, partitioned topic model, consumer offset semantics, and high-throughput nature demand purpose-built tooling and methodology. A nightly snapshot strategy that works for PostgreSQL is wholly inadequate for a system ingesting millions of events per second across hundreds of partitions.

The OSO Kafka Backup Well-Architected Framework provides a structured methodology for designing, implementing, and continuously improving Kafka backup architectures. It distils real-world operational experience into actionable guidance organised around six pillars, each with design principles, best practices, review questions, and anti-patterns. Whether you are deploying OSO Kafka Backup for the first time or auditing an existing installation, this framework gives you a repeatable process for achieving production-grade data protection.

Who Is This For?

Role	How to Use This Framework
Platform Engineers	Use the pillar checklists to validate your OSO Kafka Backup deployment against best practices. Reference the architecture patterns when designing new environments.
SREs / DevOps Engineers	Focus on the Operational Excellence and Reliability pillars to build runbooks, define SLOs for backup health, and automate recovery testing.
Kafka Administrators	Review the Performance Efficiency pillar to tune backup throughput and minimise impact on production clusters. Use Definitions to align on terminology with your team.
Solutions Architects	Leverage the reference architectures and pillar trade-off analysis to make informed design decisions during project planning and architecture reviews.
CTOs / Engineering Managers	Start with the General Design Principles and the Cost Optimisation pillar to understand strategic priorities, budgetary impact, and risk posture.
Compliance / Security Teams	Focus on the Security pillar for encryption, access control, and audit requirements. Use the review questions as input to compliance assessments and audit evidence.

How to Use This Framework

1. **Read the General Design Principles** below to establish a shared understanding of the foundational tenets that underpin every pillar.
2. **Review each pillar** in sequence or jump directly to the one most relevant to your current challenge. Each pillar is self-contained with its own best practices, anti-patterns, and review questions.

3. **Use the review questions** at the end of each pillar as a checklist during architecture reviews, post-incident analysis, or periodic health checks of your backup infrastructure.
4. **Refer to the reference architectures** for concrete deployment patterns that implement the guidance from multiple pillars in a cohesive design.
5. **Use the self-assessment** to score your current deployment against each pillar and identify the highest-priority improvement areas.
6. **Revisit periodically** --- as your Kafka footprint grows, as OSO Kafka Backup releases new capabilities, and after any significant incident, return to this framework to reassess your posture.

Definitions

Term	Definition
Backup	A point-in-time copy of Kafka topic data (records, headers, timestamps) and associated metadata (consumer offsets, topic configuration) stored in an external system such as object storage.
Restore	The process of replaying backed-up data into a Kafka cluster to recover topics, partitions, and consumer state to a specific point in time.
PITR (Point-in-Time Recovery)	The ability to restore Kafka data to any arbitrary timestamp within the retention window, rather than only to fixed snapshot boundaries.
RPO (Recovery Point Objective)	The maximum acceptable amount of data loss measured in time. An RPO of 5 minutes means the organisation accepts losing up to 5 minutes of Kafka data in a disaster scenario.
RTO (Recovery Time Objective)	The maximum acceptable duration to restore Kafka services after a failure. An RTO of 30 minutes means backup data must be fully restored and consumers operational within that window.
Backup Window	The period during which a backup job runs. For continuous backup with OSO Kafka Backup, the window is effectively zero as data is captured in near-real-time.

Term	Definition
Consumer Offset	A numeric position within a Kafka partition that tracks where a consumer group has read up to. Backing up and restoring offsets is essential for resuming processing without duplication or data loss.
Checkpoint	A recorded marker of backup progress that allows OSO Kafka Backup to resume from the last known good position after an interruption, avoiding full re-backup.
Incremental Backup	A backup strategy that captures only the data produced since the last backup rather than re-reading the entire topic log. OSO Kafka Backup operates incrementally by default.
Air-Gapped Backup	A backup stored in a location that is logically or physically isolated from the production environment, preventing ransomware or cascading failures from compromising both primary and backup data.
Segment	A unit of data within a backed-up topic partition, corresponding to a Kafka log segment. OSO Kafka Backup stores segments as discrete objects in the backup destination.
Manifest	A metadata file maintained by OSO Kafka Backup that records which segments, offsets, and timestamps are present in a backup, enabling fast lookup and selective restore.
Backup ID	A unique identifier assigned to each backup run or backup set, used to reference and manage backup data across operations such as list, restore, and delete.

General Design Principles

The following ten principles apply across all pillars of the framework. They represent the highest-level guidance for building and operating Kafka backup infrastructure with OSO Kafka Backup.

1. Automate Backup Operations

Manual backup processes are error-prone and do not scale. Every aspect of your Kafka backup lifecycle --- scheduling, execution, verification, retention enforcement, and alerting --- should be codified and automated. OSO Kafka Backup is designed for unattended operation via CLI flags, configuration files, and Kubernetes-native deployment; leverage these capabilities to eliminate human intervention from the critical path.

2. Test Recovery, Not Just Backup

A backup that has never been restored is a liability, not an asset. Regularly execute end-to-end restore drills in isolated environments to validate that backup data is complete, uncorrupted, and that your team can execute the recovery procedure within the defined RTO. Automated restore verification should be part of your CI/CD pipeline, not a quarterly manual exercise.

3. Design for Point-in-Time Recovery

Snapshot-only strategies leave gaps that accumulate between backup runs. OSO Kafka Backup's continuous, offset-aware backup model enables true PITR. Architect your deployment to preserve this capability by maintaining sufficient backup retention, storing consumer offsets alongside record data, and avoiding configurations that truncate backup granularity.

4. Decouple Backup from Cluster Operations

Backup processes must not become a single point of failure or a performance bottleneck for production Kafka. Deploy OSO Kafka Backup as an independent consumer that can be stopped, upgraded, or scaled without impacting producer or consumer workloads. Use dedicated consumer groups, separate monitoring, and independent resource quotas to maintain isolation.

5. Treat Backup Configuration as Code

Backup configuration should be version-controlled, reviewed, and deployed through the same pipelines as application code. Store OSO Kafka Backup configuration files, Kubernetes manifests, Helm values, and Terraform modules in source control. Apply pull-request review processes to changes that affect backup scope, retention, or destination to prevent accidental misconfiguration.

6. Plan for Cross-Region and Cross-Cloud Recovery

A backup stored in the same failure domain as the primary cluster provides limited protection. Design your backup architecture to write data to a geographically separate region or a different cloud provider entirely. OSO Kafka Backup's support for S3-compatible, GCS, and Azure Blob storage makes multi-destination backup achievable without custom tooling.

7. Secure Backup Data with the Same Rigour as Production

Backup data contains the same sensitive records as your production topics and must be protected accordingly. Apply encryption at rest and in transit, enforce least-privilege access policies on backup storage, enable audit logging, and rotate credentials on the same schedule as production systems. A security breach of backup storage is equivalent to a breach of production data.

8. Monitor Backup Health Continuously

Silent backup failures are the most dangerous kind. Instrument your OSO Kafka Backup deployment with metrics (backup lag, bytes written, error rates) and alerts that fire when backup health degrades. Integrate backup metrics into your existing observability stack alongside Kafka cluster monitoring so that backup drift is detected before it becomes a data-loss event.

9. Right-Size Retention to Business Requirements

Over-retention wastes storage budget; under-retention risks non-compliance and incomplete recovery. Work with stakeholders across engineering, legal, and compliance to define retention policies that satisfy regulatory obligations, business continuity requirements, and cost constraints. OSO Kafka Backup's retention configuration allows per-topic policies --- use them to differentiate between critical and ephemeral data.

10. Document and Drill Recovery Procedures

Recovery under pressure is not the time to consult documentation for the first time. Maintain clear, tested runbooks for every recovery scenario: single-topic restore, full-cluster rebuild, cross-region failover, and offset reset. Conduct tabletop exercises and live drills at regular intervals so that on-call engineers are confident and practised in executing recovery procedures with OSO Kafka Backup.

Framework Overview

The Well-Architected Framework is organised into six pillars and a set of supporting resources. Each pillar examines Kafka backup architecture through a specific lens and provides targeted guidance.

Pillar / Resource	Description	Link
Operational Excellence	Practices for running and improving backup operations through automation, observability, and continuous improvement.	Operational Excellence
Security	Guidance on protecting backup data through encryption, access control, network isolation, and audit logging.	Security
Reliability	Strategies for ensuring backup completeness, recovery success, and resilience to component failures.	Reliability
Performance Efficiency	Techniques for optimising backup throughput, minimising cluster impact, and tuning restore speed.	Performance Efficiency
Cost Optimisation	Approaches to managing storage costs, right-sizing retention, and selecting appropriate storage tiers.	Cost Optimisation
Sustainability	Considerations for reducing the environmental footprint of backup infrastructure through efficient resource utilisation.	Sustainability

WHERE TO START

If you are new to OSO Kafka Backup, begin with the Operational Excellence pillar to establish a solid foundation, then work through

> Security and Reliability before optimising for performance, cost, and sustainability.

Operational Excellence

> *"The ability to run and monitor Kafka backup workloads effectively, gain insight into their operations, and continuously improve supporting processes and procedures to deliver reliable data protection."*

The Operational Excellence pillar focuses on ensuring your backup and restore operations are reliable, observable, and continuously improving. It encompasses how your team organises around backup responsibilities, automates routine tasks, monitors health, responds to incidents, and evolves practices over time.

Design Principles

1. **Perform operations as code** -- Define backup schedules, retention policies, and restore procedures as version-controlled configuration. Eliminate manual, ad-hoc CLI invocations for production workloads.

2. **Make frequent, small, reversible changes** -- Roll out configuration changes incrementally (e.g., one topic pattern at a time). Use canary deployments for operator upgrades and validate each change before proceeding.

3. **Refine operations procedures frequently** -- Review runbooks and automation after every incident and at regular intervals. Update them to reflect current topology, tooling versions, and lessons learned.

4. **Anticipate failure** -- Design backup pipelines assuming that Kafka brokers, storage backends, and network connectivity will fail. Build pre-mortems into planning and run regular disaster-recovery drills.

5. **Learn from all operational events** -- Treat successful restores, slow backups, and outright failures equally as sources of insight. Conduct blameless post-incident reviews and feed findings back into automation and monitoring.

6. **Use managed services where possible** -- Leverage managed object storage (S3, GCS, Azure Blob), managed Kubernetes, and managed Kafka where appropriate to reduce undifferentiated operational burden and let your team focus on backup-specific concerns.

Best Practices

OE-01: Organisation & Team Readiness

What

Establish clear ownership, on-call procedures, and team competency for Kafka backup operations.

Why

Backup systems that lack a clear owner tend to drift into a neglected state. When a restore is needed during an outage, confusion over who is responsible and what steps to follow turns a recoverable incident into a prolonged one.

Implementation Guidance

- **Assign a backup operations owner** -- A named individual or team accountable for backup health, capacity planning, and restore readiness.
- **Define on-call procedures** -- Include backup/restore responsibilities in your existing on-call rotation. Ensure on-call engineers have the necessary access and credentials.
- **Maintain a disaster-recovery playbook** -- Document exact `kafka-backup` CLI commands for every recovery scenario. Store the playbook alongside your infrastructure code, not in a separate wiki.
- **Run quarterly DR drills** -- Execute full and partial restores against a staging environment. Record time-to-restore and compare against your RTO targets.
- **Train all platform engineers** -- Every engineer on the team should be able to execute a restore independently. Avoid single points of knowledge.
- **Define escalation paths** -- Document when to escalate from on-call to the backup owner, and from the backup owner to OSO support (for Enterprise customers).

> > **TIP**

> Store your DR playbook in the same Git repository as your backup configuration. This ensures the playbook is always versioned alongside the config it references.

Anti-patterns

> **ANTI-PATTERNS**
>
> - **No designated owner** -- Backup is "everyone's responsibility", which means it is no one's responsibility.
> - **Untested playbook** -- A restore procedure that has never been executed is not a procedure; it is a hope.
> - **Single point of knowledge** -- Only one engineer knows how to operate `kafka-backup`. When they are unavailable, the team is blocked.

OE-02: Backup Lifecycle Management

What

Define and automate the full lifecycle of backups: scheduling, validation, retention, and deletion.

Why

Without lifecycle automation, storage costs grow unchecked, stale backups give a false sense of security, and teams discover validation failures only when a restore is attempted during an incident.

Implementation Guidance

- **Define schedules with cron** -- Use Kubernetes CronJobs or the operator's built-in scheduling to run backups at predictable intervals.
- **Automate validation** -- Run `kafka-backup validate --deep` after every backup completes. Deep validation checks segment integrity, offset continuity, and header consistency.
- **Set retention policies per environment:**
  - **Development:** 7 days
  - **Production:** 90 days
  - **Compliance/Audit:** 7 years (with immutable storage locks)
- **Automate deletion** -- Use the operator's `retention.maxAge` or a scheduled job to remove expired backups. Never rely on manual cleanup.
- **Tag backups** -- Apply metadata labels (environment, team, compliance tier) to every backup for filtering, reporting, and cost allocation.

Configuration Example

```yaml
apiVersion: kafka-backup.osodevops.io/v1alpha1
kind: KafkaBackupSchedule
metadata:
  name: production-nightly
  namespace: kafka-backup
  labels:
    environment: production
    team: platform
    compliance-tier: standard
spec:
  schedule: "0 2 * * *"
  backupSpec:
    kafka:
      bootstrapServers: "kafka-0.kafka:9092,kafka-
1.kafka:9092,kafka-2.kafka:9092"
      topics:
        include:
          - "orders-*"
          - "payments-*"
    storage:
      type: s3
      s3:
        bucket: acme-kafka-backups-prod
        region: eu-west-1
        prefix: nightly/
    compression:
      algorithm: zstd
    retention:
      maxAge: 90d
    validation:
      deep: true
      onComplete: true
```

WARNING

Always enable `validation.deep: true` for production backups. Shallow validation only checks that files exist; it does not verify data integrity.

Anti-patterns

ANTI-PATTERNS

- **No automated deletion** -- Storage costs grow linearly and old backups become a liability rather than an asset.
- **No post-backup validation** -- You discover corrupt backups at the worst possible time: during a restore.
- **Same retention everywhere** -- Applying production retention to development wastes storage; applying development retention to compliance data violates regulations.

OE-03: Observability & Monitoring

What

Instrument backup and restore operations with metrics, dashboards, and alerts to maintain full visibility into pipeline health.

Why

Backups are background processes. Without observability, failures go unnoticed until a restore is needed. By then, your most recent valid backup may be hours or days old -- far outside your RPO.

Implementation Guidance

- **Enable Prometheus metrics** on port 8080 for all `kafka-backup` instances.
- **Monitor key metrics:**
  - `kafka_backup_lag_records` -- Consumer lag per partition. Rising lag indicates the backup cannot keep pace with production throughput.
  - `kafka_backup_records_total` -- Total records backed up. Use the rate to track throughput.
  - `kafka_backup_compression_ratio` -- Compression efficiency. A sudden change may indicate a shift in message format.
  - `kafka_backup_storage_write_latency_seconds` -- Storage backend latency. Elevated latency degrades backup performance and may indicate storage issues.
- **Build Grafana dashboards** for:
  - Overall backup health (active jobs, success/failure rates)
  - Per-topic backup status and lag
  - Storage growth trends and cost projection
  - Restore operation tracking and duration
- **Configure alerts** for:
  - Backup job failure (any job that does not complete successfully)
  - Consumer lag exceeding RPO threshold
  - Storage write errors or elevated latency
  - Checkpoint staleness (no checkpoint update within expected interval)

Configuration Example

```yaml
# kafka-backup config
metrics:
  enabled: true
  port: 8080
  bind_address: "0.0.0.0"
  path: "/metrics"
```

```yaml
# Prometheus scrape config
scrape_configs:
  - job_name: kafka-backup
    kubernetes_sd_configs:
      - role: pod
    relabel_configs:
      - source_labels: [__meta_kubernetes_pod_label_app]
        regex: kafka-backup
        action: keep
      - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_port]
        action: replace
        target_label: __address__
        regex: (.+)
        replacement: ${1}:8080
```

> **TIP**
>
> Set up a dedicated "Backup Health" Grafana dashboard and include it in your team's daily standup review. Catching a slow backup trend early is far cheaper than discovering a gap during an incident.

Anti-patterns

> **ANTI-PATTERNS**
>
> - **No monitoring at all** -- You have no idea whether backups are running, succeeding, or falling behind.

- **Monitoring backup jobs but not storage** -- A backup that completes but fails to write to storage is worse than a visible failure; it is a silent one.
- **No alerting** -- Dashboards that no one watches provide no value. Alerts ensure the right people are notified at the right time.

OE-04: Runbooks & Automation

What

Create detailed, executable runbooks for every backup and restore scenario. Automate routine operations and keep manual procedures as copy-paste-ready CLI commands.

Why

During an outage, engineers are under pressure. Runbooks that contain exact commands -- not prose descriptions -- dramatically reduce mean time to recovery (MTTR). Automation eliminates human error from repetitive tasks.

Implementation Guidance

- **Maintain runbooks for:**
  - Full cluster restore from backup
  - Single topic point-in-time recovery (PITR)
  - Consumer offset recovery
  - Backup failure investigation and remediation
  - Storage backend failover
  - Configuration change rollout
- **Each runbook must include:**
  - Prerequisites (access, credentials, tooling versions)
  - Step-by-step CLI commands (copy-paste ready)
  - Validation steps after each action
  - Rollback procedure
  - Estimated time to complete

Example Runbook Excerpt: Single Topic PITR Restore

```
# Step 1: List available backups for the target topic
kafka-backup list \
  --storage s3 \
  --bucket acme-kafka-backups-prod \
  --prefix nightly/ \
  --topic orders-events \
  --from "2026-03-20T00:00:00Z" \
  --to "2026-03-23T23:59:59Z"
```

```
# Step 2: Describe the selected backup to confirm contents
kafka-backup describe \
  --storage s3 \
  --bucket acme-kafka-backups-prod \
  --backup-id backup-20260322-020000
```

```
# Step 3: Create restore configuration (restore-orders.yaml)
restore:
  kafka:
    bootstrapServers: "kafka-0.kafka:9092,kafka-
1.kafka:9092,kafka-2.kafka:9092"
  storage:
    type: s3
    s3:
      bucket: acme-kafka-backups-prod
      region: eu-west-1
  topics:
    include:
      - "orders-events"
  pointInTime: "2026-03-22T14:30:00Z"
  targetTopic: "orders-events-restored"
  restoreOffsets: true
```

```
# Step 4: Execute the restore
kafka-backup restore --config restore-orders.yaml

# Step 5: Validate the restored topic
kafka-backup validate \
  --deep \
  --topic orders-events-restored \
  --kafka-bootstrap "kafka-0.kafka:9092"
```

WARNING

Always restore to a separate target topic (e.g., `orders-events-restored`) first. Validate the data before swapping consumers to the restored topic. Never overwrite a production topic directly.

Anti-patterns

ANTI-PATTERNS

- **Tribal knowledge** -- Restore procedures exist only in one engineer's head. They are effectively unavailable at 3 a.m. on a Sunday.
- **Prose without commands** -- "Connect to the cluster and restore the topic" is not a runbook. Exact commands with exact flags are a runbook.
- **Referencing an external wiki during an outage** -- If your Confluence page is behind an SSO that depends on the infrastructure you are trying to recover, your runbook is inaccessible when you need it most.

OE-05: Continuous Improvement

What

Establish feedback loops that drive ongoing improvement to backup operations, configuration, and tooling.

Why

Kafka topologies evolve, throughput changes, compliance requirements shift, and new `kafka-backup` releases bring performance improvements. A backup strategy that is never revisited will silently fall behind operational needs.

Implementation Guidance

- **Conduct post-incident reviews** after every backup or restore incident. Document root cause, timeline, impact, and action items. Track action item completion.
- **Run monthly metrics reviews** covering:
  - Backup duration trends (are backups taking longer as data volume grows?)
  - Storage growth rate and cost trajectory
  - Restore success rate and time-to-restore
  - RPO/RTO compliance percentage
- **Update configurations** when the environment changes:
  - New topics or topic patterns added to Kafka
  - Significant throughput increases
  - New compliance or regulatory requirements
  - Infrastructure changes (new regions, storage tiers)
- **Benchmark against new releases** -- Test new `kafka-backup` versions in staging. Measure throughput, compression ratio, and resource usage against your current version before upgrading production.

> **TIP**
>
> Add a recurring calendar event for a monthly "Backup Operations Review". Use it to walk through metrics dashboards, review open action items, and assess whether current configurations still meet requirements.

Anti-patterns

ANTI-PATTERNS

- **Set-and-forget** -- Deploying a backup configuration once and never reviewing it. Environments change; configurations must follow.
- **No post-incident reviews** -- Repeating the same failure because the team never analysed the first occurrence.
- **Annual-only review** -- Reviewing backup strategy once a year guarantees that it is out of date for eleven months.

Review Questions

Use these questions to assess your operational maturity. For each question, rate your current state as **None**, **Basic**, **Advanced**, or **Expert**.

1. Is there a designated owner (individual or team) accountable for Kafka backup operations?
2. Are backup schedules, retention policies, and validation steps defined as version-controlled configuration?
3. Do you run automated deep validation (`kafka-backup validate --deep`) after every backup?
4. Are Prometheus metrics enabled and scraped for all `kafka-backup` instances?
5. Do you have Grafana dashboards (or equivalent) providing visibility into backup health, lag, and storage growth?
6. Are alerts configured for backup failures, RPO threshold breaches, and storage errors?
7. Do you maintain copy-paste-ready runbooks for every restore scenario (full cluster, single topic PITR, offset recovery)?
8. Have you executed a full disaster-recovery drill in the last quarter?
9. Do you conduct post-incident reviews after every backup or restore incident?
10. Do you review backup metrics and configurations at least monthly to ensure they match current requirements?

Resources

- Deployment Guide -- Infrastructure setup for all supported platforms
- CLI Reference -- Complete `kafka-backup` command reference
- Kubernetes Operator -- Operator installation, CRDs, and guides
- Metrics Reference -- Full list of Prometheus metrics
- Monitoring Setup Guide -- Step-by-step Prometheus and Grafana configuration

Security

> *Protecting backup data, configurations, and operations through identity management, encryption, access controls, and audit logging — with the same rigour applied to production Kafka data.*

Backups contain a complete copy of your Kafka data. A compromised backup is as damaging as a compromised production cluster. The Security pillar ensures that every layer of your backup architecture — from credentials and network paths to storage buckets and audit trails — is hardened, monitored, and regularly reviewed.

Design Principles

1. **Implement strong identity foundation** — Apply the principle of least privilege to every service account, IAM role, and operator that interacts with backup infrastructure.
2. **Enable traceability** — Audit-log every backup, restore, and configuration change so you can answer *who did what, when, where, and with what outcome*.
3. **Apply security at all layers** — Secure data in transit, at rest, at the access layer, and inside configuration files. No single control should be the only line of defence.
4. **Automate security best practices** — Rotate credentials on a schedule, scan configurations for drift, and enforce policies via code rather than manual checklists.
5. **Protect data at rest with encryption** — Encrypt all backup data using customer-managed keys where compliance demands it, and verify encryption status continuously.
6. **Prepare for security events** — Maintain runbooks for credential revocation, backup quarantine, and forensic investigation so the team can respond quickly to incidents.

Best Practices

SEC-01: Identity & Access Management

Effective IAM starts with dedicated service accounts scoped to the minimum permissions each operation requires.

Least-Privilege Role Separation

Role	Storage Access	Kafka Access
Backup	Write-only to storage	Read-only from source Kafka
Restore	Read-only from storage	Write to target Kafka
Validate	Read-only from storage	None

TIP

Use IAM roles (AWS), managed identities (Azure), or workload identity (GCP) instead of static credentials. These provide automatic credential rotation and eliminate the risk of leaked long-lived keys.

Enterprise: Role-Based Access Control

Enterprise editions support RBAC for multi-team environments, allowing administrators to grant granular permissions per team, topic pattern, or environment.

Configuration: AWS IAM Policy

Backup role — write-only storage with read-only Kafka:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BackupWriteOnly",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-kafka-backups",
        "arn:aws:s3:::my-kafka-backups/*"
      ]
    }
  ]
}
```

Restore role — read-only storage:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestoreReadOnly",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-kafka-backups",
        "arn:aws:s3:::my-kafka-backups/*"
      ]
    }
  ]
}
```

Anti-Patterns

ANTI-PATTERNS

- **Admin credentials for backup operations** — Over-privileged accounts increase blast radius if compromised.
- **Same IAM role for backup and restore** — Violates least privilege; a compromised backup process could overwrite or delete data.
- **Static access keys** — Long-lived keys are a top cause of cloud security incidents.

- **No environment separation** — Production and staging sharing credentials or storage accounts.

SEC-02: Data Protection at Rest

All backup data must be encrypted at rest, with key management appropriate to your compliance requirements.

Server-Side Encryption Options

Cloud	Default	Customer-Managed Key
AWS S3	SSE-S3 (AES-256)	SSE-KMS / SSE-C
Azure Blob	Microsoft-managed keys	Customer-managed keys (CMK)
GCS	Google-managed keys	Customer-managed encryption keys (CMEK)

Enterprise: Client-Side Encryption

Enterprise editions support client-side AES-256 encryption, ensuring data is encrypted before it leaves the backup process — providing defence-in-depth even if storage-layer encryption is misconfigured.

> **TIP**
>
> For regulated industries (finance, healthcare), use customer-managed keys (CMK/CMEK) with automatic key rotation. Combine with S3 Object Lock or Azure immutable storage to protect against tampering and ransomware.

Configuration: S3 with KMS Encryption

```yaml
storage:
  type: s3
  s3:
    bucket: my-kafka-backups
    region: eu-west-1
    encryption:
      type: aws:kms
      kms_key_id: arn:aws:kms:eu-west-1:123456789012:key/abcd-
1234
```

Configuration: Enterprise Client-Side Encryption

```yaml
storage:
  type: s3
  s3:
    bucket: my-kafka-backups
    region: eu-west-1
  encryption:
    enabled: true
    algorithm: AES-256
    key_provider: aws-kms
    kms_key_id: arn:aws:kms:eu-west-1:123456789012:key/abcd-
1234
```

Anti-Patterns

ANTI-PATTERNS

- **Unencrypted storage buckets** — Backup data readable by anyone with network access to the storage layer.
- **Default S3 encryption without key management** — SSE-S3 encrypts data but Amazon manages the keys; insufficient for regulatory requirements that mandate customer-controlled keys.

- **Public buckets** — S3 buckets or Azure containers with public access enabled, exposing backup data to the internet.

SEC-03: Data Protection in Transit

Encrypt all data moving between Kafka, the backup process, and cloud storage.

Kafka Connections

- Use TLS for all Kafka connections; **mTLS is preferred** for mutual authentication.
- Minimum **TLS 1.2**; prefer **TLS 1.3** where supported.
- Validate broker certificates against a trusted CA.

Storage Backend Connections

HTTPS is the default for S3, Azure Blob, and GCS. Ensure custom or on-premises storage endpoints also enforce TLS.

Private Network Paths

Cloud	Service	Benefit
AWS	S3 Gateway Endpoint	Traffic stays within VPC
Azure	Private Endpoint	Private IP for storage account
GCP	Private Google Access	No public IP required

⚠ **WARNING**

> VPC endpoints and private links prevent backup data from traversing the public internet, reducing exposure to man-in-the-middle attacks and data exfiltration.

Configuration: Kafka TLS / mTLS

```yaml
source:
  bootstrap_servers:
    - kafka:9093
  security:
    security_protocol: SASL_SSL
    ssl_ca_location: /certs/ca.crt
    ssl_certificate_location: /certs/client.crt
    ssl_key_location: /certs/client.key
```

kafka-backup supports the following security properties:

Property	Description
`security_protocol`	`SASL_SSL`, `SSL`, `SASL_PLAINTEXT`, `PLAINTEXT`
`ssl_ca_location`	Path to CA certificate
`ssl_certificate_location`	Path to client certificate
`ssl_key_location`	Path to client private key

Anti-Patterns

> **ANTI-PATTERNS**
>
> - **PLAINTEXT connections** — Data and credentials sent in the clear on the network.

- **Backup traffic over the public internet** — Without VPC endpoints, data traverses public networks unnecessarily.
- **Self-signed certificates without a trust chain** — Disabling certificate verification removes protection against MITM attacks.
- **TLS 1.0 / 1.1** — Deprecated protocols with known vulnerabilities.

SEC-04: Secrets Management

Credentials must never appear in plain text inside configuration files or source control.

Recommended Secrets Backends

Backend	Use Case
AWS Secrets Manager / SSM Parameter Store	AWS-native workloads
Azure Key Vault	Azure-native workloads
HashiCorp Vault	Multi-cloud or on-premises
Kubernetes Secrets (encrypted with KMS)	Kubernetes-native deployments

> **TIP**
>
> Use short-lived credentials wherever possible — AWS STS tokens, federated identities, or OIDC-based workload identity. These expire automatically, limiting the window of exposure if intercepted.

Environment Variable Substitution

kafka-backup supports `${VAR_NAME}` environment variable substitution in configuration files, allowing secrets to be injected at runtime from any secrets backend:

```yaml
source:
  bootstrap_servers:
    - ${KAFKA_BOOTSTRAP_SERVERS}
  security:
    security_protocol: SASL_SSL
    sasl_mechanism: ${SASL_MECHANISM}
    sasl_username: ${SASL_USERNAME}
    sasl_password: ${SASL_PASSWORD}
```

Configuration: Kubernetes Secret

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: kafka-backup-credentials
  namespace: kafka-backup
type: Opaque
stringData:
  SASL_USERNAME: backup-service-account
  SASL_PASSWORD: changeme-use-sealed-secrets
  AWS_ACCESS_KEY_ID: AKIAIOSFODNN7EXAMPLE
  AWS_SECRET_ACCESS_KEY:
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

WARNING

The example above uses `stringData` for readability. In production, use **SealedSecrets**, **External Secrets Operator**, or **Vault Agent** to inject secrets securely rather than storing them directly in Kubernetes manifests committed to Git.

Anti-Patterns

ANTI-PATTERNS

- **Plaintext credentials in YAML committed to Git** — Secrets in version control are exposed to anyone with repository access and persist in Git history.
- **Long-lived static access keys** — Keys that never rotate accumulate risk over time.
- **Shared credentials across environments** — A compromise in staging exposes production.

SEC-05: Audit & Compliance

Maintain a tamper-evident record of every backup operation to support compliance audits and incident investigations.

Enterprise: Audit Logging

Enterprise editions emit structured audit events capturing:

- **Who** — service account or user identity
- **What** — operation performed (backup, restore, validate, configure)
- **When** — timestamp with timezone
- **Where** — source cluster, target storage, topic(s)
- **Outcome** — success, failure, partial completion

Cloud Storage Access Logging

Cloud	Service	Captures
AWS	S3 Server Access Logging + CloudTrail	Object-level reads/writes, API calls

Cloud	Service	Captures
Azure	Storage Analytics Logging	Read/write/delete operations
GCP	Cloud Audit Logs	Data access and admin activity

TIP

Align log retention periods with your compliance framework (SOC 2, PCI-DSS, HIPAA). Configure alerts for anomalous operations such as unexpected restore activity, bulk deletes, or access from unfamiliar IP ranges.

Configuration: Enterprise Audit Logging

```yaml
audit:
  enabled: true
  destination:
    type: datadog
    api_key: ${DATADOG_API_KEY}
    site: datadoghq.eu
  events:
    - backup.started
    - backup.completed
    - backup.failed
    - restore.started
    - restore.completed
    - restore.failed
    - config.changed
    - credentials.rotated
  retention:
    days: 365
```

Anti-Patterns

ANTI-PATTERNS

- **No audit trail for restore operations** — Restores modify production data; without logging, you cannot trace who restored what or diagnose issues.
- **Audit logs co-located with backups** — An attacker who compromises backup storage can also tamper with logs. Store logs in a separate, append-only destination.
- **No log retention policy** — Logs that expire before an audit window closes leave gaps in compliance evidence.

SEC-06: Data Masking & Privacy

Backup data often contains personally identifiable information (PII) subject to privacy regulations such as GDPR, CCPA, and HIPAA.

Enterprise: Field-Level Data Masking

Enterprise editions support schema-aware, field-level data masking that integrates with Schema Registry to identify and redact sensitive fields before data is written to storage.

Capabilities include:

- **Right to be forgotten** — Delete or mask specific records to satisfy GDPR erasure requests.
- **Masking policies per topic and field** — Define rules such as *mask `email` in `user-events`* or *hash `ssn` in `customer-records`*.
- **Schema Registry integration** — Automatically detect new fields and apply default masking policies.

Tiered Backup Strategy

Separate backup data into tiers based on sensitivity:

Tier	Access	Retention	Content
Full (unmasked)	Restricted — security/compliance team only	Short (regulatory minimum)	Complete data for disaster recovery
Masked	Broader — development, analytics teams	Longer	PII redacted for safe use in non-production

> ⚠ **WARNING**
>
> Masking must be applied at backup time, not at restore time. Once unmasked PII is written to storage, it is subject to the same data protection obligations as the original Kafka topic.

Anti-Patterns

> **ANTI-PATTERNS**
>
> - **Backing up PII without masking** — Creates a shadow copy of sensitive data outside the controls applied to production systems.
> - **No GDPR deletion process for backups** — Failing to honour erasure requests in backup data is a compliance violation.
> - **Assuming backups are exempt from privacy regulations** — Regulators consider backups as part of the data lifecycle; all obligations apply.

Review Questions

Use the following questions during architecture reviews to assess your backup security posture:

1. Are dedicated, least-privilege service accounts used for backup, restore, and validate operations?
2. Is IAM role assumption or workload identity used instead of static access keys?
3. Is all backup data encrypted at rest with appropriate key management (SSE-KMS, CMK, CMEK)?
4. Are all Kafka connections secured with TLS 1.2+ or mTLS?
5. Does backup traffic stay within private network paths (VPC endpoints, private links)?
6. Are secrets managed through a dedicated secrets backend rather than stored in configuration files?
7. Are credentials short-lived and rotated automatically?
8. Is there an audit trail for every backup, restore, and configuration change?
9. Are audit logs stored separately from backup data with appropriate retention policies?
10. Is PII identified, masked, or encrypted before being written to backup storage?

Resources

- Security Setup Guide — Step-by-step TLS, SASL, and encryption configuration
- Encryption (Enterprise) — Client-side encryption and key management
- RBAC (Enterprise) — Role-based access control for multi-team environments
- Audit Logging (Enterprise) — Structured audit events and compliance reporting
- Schema Registry (Enterprise) — Schema-aware masking and data governance

Reliability

> *The ability to consistently perform backup and restore operations correctly, recover from failures gracefully, and meet defined RPO and RTO targets under all conditions.*

Reliability is the foundation of any backup system. A backup that cannot be restored is worse than no backup at all — it creates a false sense of safety. The Reliability pillar ensures that every backup is validated, every restore is rehearsed, and every failure scenario has a documented, tested recovery path.

Design Principles

1. **Automatically recover from failure** — Use checkpoints and incremental resume so that a failed backup picks up where it left off rather than starting from scratch.
2. **Test recovery procedures, not just backup creation** — A backup is only as good as its last successful restore. Validate regularly.
3. **Scale horizontally to handle partition growth** — As topics gain partitions or new topics are added, the backup infrastructure must scale without manual intervention.
4. **Manage change through automation** — Use GitOps workflows and Kubernetes CRDs to version, review, and roll out configuration changes predictably.
5. **Design for zero data loss** — Understand and configure your RPO targets explicitly; do not rely on defaults to meet business requirements.
6. **Implement fault isolation** — One partition failure should not affect the backup of other partitions. Isolate failure domains so blast radius is minimised.

Best Practices

REL-01: Backup Integrity & Validation

Every backup must be validated to confirm it can be used for a successful restore. Silent corruption, incomplete segments, or missing offsets can render a backup useless when it is needed most.

What to Validate

Check	Description
Manifest completeness	All expected topics and partitions are present in the backup manifest
Segment integrity	Checksums match and compressed segments can be decompressed without errors
Offset continuity	No gaps in offset sequences within each partition
Time window coverage	Backup spans the expected time range without holes

Why It Matters

Backup operations can report success (exit code 0) while producing incomplete or corrupt output. Network interruptions, storage throttling, or transient Kafka errors can result in partial writes that are only detectable through explicit validation.

Implementation

- Run `kafka-backup validate --deep` after every backup operation.
- Automate validation in your CI/CD pipeline or as a post-backup Kubernetes Job.
- Periodically perform a full restore-to-temporary-cluster validation to confirm end-to-end recoverability.

> **TIP**
>
> Schedule a weekly automated restore to a temporary cluster. This catches issues that static validation cannot detect, such as schema compatibility problems or consumer group restoration failures.

Configuration

Deep validation of a backup:

```
kafka-backup validate \
  --path s3://my-kafka-backups/production \
  --backup-id 2026-03-24T00-00-00Z \
  --deep
```

Describe backup metadata for programmatic checks:

```
kafka-backup describe \
  --path s3://my-kafka-backups/production \
  --backup-id 2026-03-24T00-00-00Z \
  --format json
```

Anti-Patterns

ANTI-PATTERNS

- **Assuming success from exit code alone** — A zero exit code means the process completed, not that the backup is complete or uncorrupted.
- **Never validating until a real restore is needed** — Discovering corruption during a production incident turns a recoverable situation into a crisis.
- **Validating only the latest backup** — Older backups may have degraded in storage; periodic re-validation catches bit rot and storage issues.

REL-02: Point-in-Time Recovery Strategy

Point-in-time recovery (PITR) allows you to restore data to a specific moment, not just the latest backup. This is critical for recovering from data corruption, accidental deletes, or application bugs that produced bad data.

What to Define

Before implementing PITR, answer these questions:

- **What granularity is needed?** — Can you tolerate restoring to the nearest hour, or do you need minute-level precision?
- **What is the maximum acceptable backup window?** — The gap between the last backup and the failure determines potential data loss.
- **Which topics need PITR?** — Not every topic requires the same recovery granularity.

Why It Matters

Backup Frequency	Achievable RPO	Use Case
Continuous	< 1 minute	Payment transactions, order streams
Hourly	< 1 hour	User events, session data
Daily	< 24 hours	Logs, analytics, non-critical streams

Implementation

Configure time windows in your restore configuration using epoch milliseconds. This allows precise recovery to the exact moment before corruption or data loss occurred.

⚠ **WARNING**

Epoch timestamps must be in **milliseconds**, not seconds. A common mistake is using a 10-digit Unix timestamp (seconds) instead of a 13-digit epoch millisecond value, resulting in restoring data from 1970.

Configuration

PITR restore configuration:

```yaml
restore:
  source:
    path: s3://my-kafka-backups/production
    backup_id: 2026-03-24T00-00-00Z
  target:
    bootstrap_servers:
      - kafka-restore:9092
  options:
    time_window_start: 1711234800000   # 2026-03-23T15:00:00Z
    time_window_end: 1711238400000     # 2026-03-23T16:00:00Z
    topic_mapping:
      - source: orders
        target: orders-restored
      - source: payments
        target: payments-restored
```

Anti-Patterns

ANTI-PATTERNS

- **Daily backups when RPO is 1 hour** — The backup frequency must match or exceed the RPO requirement. A daily backup cannot deliver an hourly RPO.
- **Not understanding epoch timestamps** — Misconfigured time windows restore the wrong data range, wasting time during an incident.
- **No topic-level RPO classification** — Treating all topics with the same backup frequency wastes resources on low-value data and under-protects high-value data.

REL-03: Consumer Offset Recovery

Restoring messages is only half the battle. If consumer offsets are not recovered correctly, applications will either reprocess data (duplicates) or skip data (loss). Offset recovery must be planned as part of every restore operation.

What to Understand

kafka-backup supports multiple offset recovery strategies:

Strategy	Description	Best For
Timestamp-based	Reset offsets to a specific timestamp	PITR restores
Offset-based (header)	Use original offset headers embedded in backup	Exact replay
Group-based	Restore committed consumer group offsets	Resuming existing consumers
Cluster-scan	Scan target cluster to determine appropriate offsets	Cross-cluster migration
Manual	Specify exact offsets per partition	Surgical recovery

Why It Matters

Incorrect offset recovery is the most common cause of post-restore issues. Applications may appear healthy but silently skip records or reprocess hours of data, causing downstream inconsistencies.

Implementation

Always use the two-phase approach: **plan** first, then **execute**.

1. Generate a plan to review before applying changes.
2. Review the plan to confirm offsets are correct.
3. Execute the plan to apply offset resets.

> **TIP**
>
> The plan phase is non-destructive and produces a reviewable output. Always inspect the plan before executing, especially during incident recovery when mistakes are costly.

Configuration

Generate an offset reset plan:

```
kafka-backup offset-reset plan \
  --config restore-config.yaml \
  --strategy timestamp \
  --timestamp 1711238400000 \
  --output offset-plan.json
```

Review the plan:

```
cat offset-plan.json | jq '.partitions[] | {topic, partition,
current_offset, new_offset}'
```

Execute the offset reset:

```
kafka-backup offset-reset execute \
  --plan offset-plan.json \
  --config restore-config.yaml
```

Anti-Patterns

ANTI-PATTERNS

- **Restoring data without considering consumer offsets** — Messages are restored but consumers either skip them entirely or reprocess old data.
- **Blindly resetting to earliest or latest** — `earliest` causes full reprocessing; `latest` skips all restored data. Neither is appropriate without understanding the restore context.
- **Skipping the plan phase** — Executing offset resets without review during a high-pressure incident leads to compounding errors.

REL-04: Disaster Recovery Planning

A disaster recovery plan defines how you will restore Kafka data when the worst happens. Without explicit RPO and RTO targets per data tier, recovery is ad-hoc and unpredictable.

What to Define

Classify topics into tiers based on business impact and assign RPO/RTO targets:

Tier	Examples	RPO	RTO
Tier 1 — Critical	Payments, orders, financial transactions	< 1 minute	< 15 minutes
Tier 2 — Important	User events, session data, notifications	< 1 hour	< 1 hour
Tier 3 — Standard	Logs, metrics, analytics streams	< 24 hours	< 4 hours

Why It Matters

Without defined tiers, teams either over-invest in protecting low-value data or under-invest in protecting critical streams. Explicit targets drive backup frequency, storage redundancy, and recovery automation decisions.

Implementation

Document and rehearse a six-phase DR procedure:

1. **Detection** — Automated alerting identifies the failure (monitoring, health checks, anomaly detection).
2. **Declaration** — On-call engineer or automated runbook declares a disaster based on predefined criteria.
3. **Communication** — Stakeholders are notified via defined channels (PagerDuty, Slack, status page).
4. **Execution** — Restore operations are initiated following the documented runbook.
5. **Validation** — Restored data is verified against integrity checks and consumer applications are confirmed healthy.
6. **Fallback** — If restoration fails, execute the fallback plan (alternative backup, manual recovery, or degraded mode).

> ⚠ **WARNING**
>
> DR documentation must be accessible during an outage. If your runbooks are stored on the same infrastructure that has failed, you cannot access them when you need them most. Keep copies in at least two independent locations.

Anti-Patterns

> **ANTI-PATTERNS**

- **No defined RPO/RTO targets** — Without targets, there is no way to measure whether your backup strategy is adequate or whether recovery was successful.
- **DR plan exists but has never been tested** — An untested plan is an assumption, not a plan.
- **DR documentation stored only on the failing system** — Wiki on the same cloud region, runbooks in the same Kubernetes cluster, or playbooks in the same Git hosting provider.
- **Single-person DR knowledge** — If only one engineer knows how to restore, your RTO depends on their availability.

REL-05: Fault Isolation & Redundancy

Backup infrastructure must be isolated from the systems it protects. A failure that takes down your Kafka cluster should not also take down your ability to restore from backup.

What to Implement

Isolation Domain	Recommendation
Storage location	Different account, region, or cloud provider from the source cluster
Compute	Dedicated backup infrastructure, not co-located on broker nodes
Network	Separate failure domain; backup process reachable even if source network is degraded
Partition-level	Per-partition fault isolation — one partition failure does not block others

Why It Matters

If backups are stored in the same region and account as the source cluster, a single cloud incident (region outage, account compromise, IAM misconfiguration) can destroy both production data and all backups simultaneously.

Implementation

- Enable **S3 cross-region replication** to maintain backup copies in a separate region.
- Run backup processes on **dedicated infrastructure**, not on Kafka broker nodes.
- Use **per-partition checkpointing** so a failure in one partition allows others to continue.
- Implement **checkpoint-based resume** so interrupted backups pick up where they left off.
- For ransomware protection, maintain **air-gapped backups** using S3 Object Lock or equivalent immutable storage.

> **TIP**
>
> Use a separate AWS account for backup storage. Even if the production account is compromised, the backup account remains isolated. Cross-account IAM roles provide secure access without shared credentials.

Configuration

Cross-region S3 storage with replication:

```yaml
storage:
  type: s3
  s3:
    bucket: my-kafka-backups-dr
    region: us-west-2          # Different region from source
cluster
    endpoint: ""
    force_path_style: false
  backup:
    checkpoint_interval: 30s   # Frequent checkpoints for
```

```
resume
    per_partition_isolation: true
```

Anti-Patterns

ANTI-PATTERNS

- **Backups in the same region and account as the source cluster** — A region-wide outage or account compromise destroys both production data and backups.
- **Running backup processes on Kafka broker nodes** — Broker failure takes down both the data source and the backup process simultaneously.
- **No redundancy for backup storage** — A single storage location is a single point of failure.
- **Single point of failure in backup infrastructure** — One backup server, one storage bucket, one network path.

REL-06: DR Testing & Chaos Engineering

A disaster recovery plan is only reliable if it is tested regularly. Chaos engineering validates that your backup infrastructure handles real-world failure scenarios, not just ideal conditions.

What to Test

Test Type	Frequency	Scope
Tabletop exercise	Monthly	Walk through DR scenarios with the team; identify gaps in runbooks
Single topic restore	Weekly (automated)	Restore one topic to a temporary cluster and validate integrity

Test Type	Frequency	Scope
Full cluster restore	Quarterly	Restore all tiered topics and measure actual RTO
Failover drill	Semi-annually	Simulate primary cluster loss and execute full DR procedure
Chaos test	Quarterly	Inject failures into backup infrastructure and observe behaviour

Why It Matters

Without regular testing, your DR plan degrades over time. Infrastructure changes, new topics, configuration drift, and team turnover all erode recovery capabilities. Testing keeps them current.

Implementation

Design chaos scenarios that exercise your failure modes:

Scenario	What It Tests
Kill backup process mid-run	Checkpoint resume — does backup continue from last checkpoint?
Storage outage (revoke S3 access)	Error handling and retry logic
Network partition (block Kafka port)	Graceful degradation and reconnection
Corrupt a backup segment	Validation detection — does `validate --deep` catch it?
Full region failure	Cross-region restore from replicated backup

> ⚠ **WARNING**
>
> Always run chaos tests in a controlled environment with clear rollback procedures. Document the blast radius of each test before executing. Never run destructive chaos tests against production backups without an isolated copy.

Implementation: Document Results

Every DR test must produce a written report including:

- **Actual RTO achieved** vs target RTO
- **Actual RPO achieved** vs target RPO
- **Issues encountered** during the test
- **Action items** with owners and deadlines
- **Pass/fail determination** against defined success criteria

> **TIP**
>
> Track RTO and RPO measurements over time. Trending data reveals whether your recovery capabilities are improving or degrading, and provides evidence for compliance audits.

Anti-Patterns

ANTI-PATTERNS

- **DR drills only after incidents** — Reactive testing means you discover problems during real outages, not before them.
- **Testing in non-production-like environments** — A DR test against a cluster with 10 partitions does not validate recovery of a production cluster with 10,000 partitions.
- **Undocumented test results** — Without written records, lessons are lost, the same issues recur, and compliance auditors have no evidence of testing.
- **Never testing full-scale restore** — Single-topic restores build confidence but do not validate that your infrastructure can handle a complete cluster recovery within the RTO window.

Review Questions

Use the following questions during architecture reviews to assess the reliability of your backup strategy:

1. Are all backups validated automatically after completion using `kafka-backup validate --deep`?
2. Is a full restore-to-temporary-cluster test performed at least quarterly?
3. Are RPO and RTO targets defined and documented for every topic tier?
4. Does the backup frequency match or exceed the RPO requirement for each tier?
5. Is consumer offset recovery planned and tested as part of every restore procedure?
6. Are backups stored in a different failure domain (region, account, or cloud provider) from the source cluster?
7. Does the backup process use per-partition fault isolation and checkpoint-based resume?

8. Is there a documented, tested DR procedure with clear escalation and communication steps?
9. Are DR drills conducted at least quarterly, with results documented and action items tracked?
10. Are chaos engineering scenarios used to validate backup infrastructure resilience?

Resources

- Point-in-Time Recovery Guide — Step-by-step PITR restore procedures
- Offset Management Guide — Consumer offset recovery strategies and workflows
- CLI Reference — Complete command reference for backup, restore, and validate operations
- PITR Architecture — Technical deep-dive into point-in-time recovery implementation
- Multi-Cluster DR Examples — Example configurations for cross-region and multi-cluster disaster recovery

Performance Efficiency

> *"Using compute, storage, and network resources efficiently to meet backup and restore throughput requirements, and maintaining that efficiency as data volumes grow."*

Kafka backup workloads must keep pace with production data rates while consuming the minimum necessary resources. The Performance Efficiency pillar ensures your backup architecture is right-sized, properly tuned, and continuously benchmarked — so that growing data volumes never outstrip your ability to protect them.

Design Principles

1. **Right-size for throughput requirements** — Match compute, memory, and network capacity to measured data rates rather than guessing. Over-provisioning wastes budget; under-provisioning causes backup lag.

2. **Use compression to reduce storage and network overhead** — Compress backup segments to shrink storage footprint and reduce network transfer time, choosing an algorithm that balances ratio against CPU cost.

3. **Benchmark before deploying to production** — Validate throughput, latency, and resource consumption under realistic load before going live. Assumptions about performance are not a substitute for measurement.

4. **Monitor performance continuously and act on trends** — Track throughput, compression ratios, and resource utilisation over time. Identify degradation early and address it before it becomes an incident.

5. **Go serverless / managed where possible** — Prefer managed object storage (S3, GCS, Azure Blob) over self-managed block storage. Managed services scale automatically and eliminate storage infrastructure overhead.

6. **Experiment with configuration in staging before production** — Test tuning changes (segment sizes, compression levels, concurrency) in a staging environment that mirrors production topology and data characteristics.

Best Practices

PE-01: Throughput Optimisation

Performance Targets

Establish clear targets for your environment and validate them through benchmarking:

Metric	Target	Notes
Throughput per partition	100+ MB/s	Depends on network, storage backend, and message size
Checkpoint latency (p99)	< 100 ms	Ensures minimal data loss window on failure
Compression ratio	3--5x	Typical for JSON/text payloads with zstd
Memory usage	< 500 MB for 4 partitions	Rust-native efficiency keeps the footprint low

Tuning Levers

- `segment_max_bytes` — Set between 128 MB and 256 MB. Larger segments reduce the number of storage API calls but increase the flush latency window.
- `fetch_max_bytes` — Increase to match partition throughput. Undersized fetch buffers cause excessive round trips to brokers.
- `compression` — Choose `zstd` for the best ratio-to-speed trade-off, or `lz4` when CPU is constrained.
- `max_concurrent_partitions` — Scale this to match available CPU cores and network bandwidth. Each partition backup runs in its own async task.

Implementation Guidance

- **Scale horizontally** — Run multiple `kafka-backup` instances, each handling a subset of partitions, rather than scaling a single instance vertically.
- **Co-locate with Kafka brokers** — Deploy backup instances in the same availability zone as the brokers they read from to minimise network latency and

cross-AZ data transfer costs.

- **Tune incrementally** — Change one parameter at a time and measure the impact. Simultaneous changes make it impossible to attribute improvements or regressions.

Configuration: High-Throughput Backup

```yaml
kafka:
  bootstrap_servers: "kafka-0:9092,kafka-1:9092,kafka-2:9092"
  fetch_max_bytes: 52428800  # 50 MB

backup:
  segment_max_bytes: 268435456  # 256 MB
  compression: zstd
  compression_level: 3
  max_concurrent_partitions: 8

storage:
  type: s3
  bucket: my-kafka-backups
  region: eu-west-1
```

> **TIP**
>
> Start with `segment_max_bytes: 134217728` (128 MB) and increase to 256 MB only after confirming that your storage backend handles larger PUT requests without timeout issues.

Anti-patterns

> **ANTI-PATTERNS**

- **Using default configuration for high-throughput topics** — Defaults are conservative. Production workloads almost always benefit from tuning segment size, fetch size, and concurrency.
- **Deploying backup in a different region from Kafka** — Cross-region data transfer adds latency, cost, and fragility.
- **Using very small segments (< 32 MB)** — Creates excessive storage API calls, increasing both latency and cost.
- **No benchmarking before production** — Performance assumptions based on development data volumes are unreliable.

PE-02: Compression Strategy

Algorithm Comparison

Algorithm	Compression Ratio	Speed	CPU Usage	Best For
zstd	3--5x	Fast	Moderate	General-purpose default
lz4	2--3x	Very fast	Low	Latency-sensitive workloads
none	1x	Fastest	None	Pre-compressed or encrypted data

Compression Ratios by Data Format

Data Format	Typical Ratio (zstd)	Notes
JSON	5--8x	Highly repetitive structure compresses well
Avro	2--4x	Binary format, moderate compressibility

Data Format	Typical Ratio (zstd)	Notes
Protobuf	2--3x	Compact binary, less headroom for compression
Already compressed	< 1.1x	No benefit — adds CPU cost for no gain

Zstd Compression Levels

Level Range	Speed	Ratio	Use Case
1--3	Fast	Moderate	Recommended for real-time backup
4--9	Balanced	Higher	Batch or off-peak backup windows
10+	Slow	Maximum	Archival, where ratio matters more than speed

TIP

Levels 1--3 offer the best throughput-to-ratio trade-off for real-time backup. Only increase beyond 3 if you have measured a meaningful improvement in your specific data and can tolerate the additional CPU overhead.

Monitoring Compression

Monitor the `kafka_backup_compression_ratio` metric to detect changes in data compressibility over time. A sudden drop in ratio may indicate a change in message format or the introduction of pre-compressed payloads.

Anti-patterns

ANTI-PATTERNS

- **Compressing already-compressed data** — Wastes CPU cycles for negligible size reduction. Disable compression for topics containing gzip, snappy, or encrypted payloads.
- **Using maximum compression levels for latency-sensitive workloads** — High zstd levels (10+) can add significant CPU time per segment, increasing flush latency.
- **Never benchmarking compression** — Different data formats yield vastly different ratios. Benchmark your actual payloads to choose the right algorithm and level.

PE-03: Storage Backend Selection

Backend Comparison

Backend	Throughput	Durability	Relative Cost	Notes
Amazon S3	High	11 9's	$$	Most mature, widest tooling ecosystem
Azure Blob Storage	High	17 9's (GRS)	$$	Native integration with Azure workloads
Google Cloud Storage	High	11 9's	$$	Strong multi-region replication
MinIO	High	Depends on deployment	$	S3-compatible, self-hosted

Backend	Throughput	Durability	Relative Cost	Notes
Filesystem	Very high	Depends on underlying storage	Free	No network overhead, limited durability

Production Recommendations

- **Use object storage** — S3, Azure Blob, or GCS provide the durability, scalability, and lifecycle management required for production backup data.
- **Enable versioning** — Protect against accidental overwrites or deletions. Versioning also supports compliance requirements for immutable backups.
- **Use Transfer Acceleration for cross-region** — When restoring data to a different region, enable S3 Transfer Acceleration or equivalent to reduce transfer time.

Development Recommendations

- **Filesystem or MinIO for local development** — Avoids cloud costs and network dependencies during development and testing.
- **Mirror production configuration** — Use the same segment sizes, compression settings, and directory structures so that performance characteristics transfer to production.

Reduce Latency with VPC Endpoints

Deploy VPC endpoints (AWS), private endpoints (Azure), or Private Service Connect (GCP) to keep backup traffic on the cloud provider's backbone network. This eliminates internet gateway latency and reduces data transfer costs.

Configuration Examples

Amazon S3:

```yaml
storage:
  type: s3
  bucket: my-kafka-backups
  region: eu-west-1
  # Use VPC endpoint for private network access
  endpoint: https://s3.eu-west-1.amazonaws.com
```

Azure Blob Storage:

```yaml
storage:
  type: azure
  container: my-kafka-backups
  account_name: mystorageaccount
```

Filesystem (development):

```yaml
storage:
  type: filesystem
  base_path: /var/lib/kafka-backup/data
```

Anti-patterns

ANTI-PATTERNS

- **Using filesystem storage for production** — Filesystem storage lacks the durability guarantees, lifecycle management, and cross-region replication of object storage.
- **No versioning on backup buckets** — A single accidental deletion or overwrite can destroy your recovery capability.
- **Storing backups on a different continent from Kafka** — Intercontinental data transfer adds significant latency and cost to every backup segment write.
- **No VPC endpoints** — Routing backup traffic through the public internet adds latency, increases cost, and exposes data to unnecessary network hops.

PE-04: Resource Sizing & Scaling

Sizing Guidelines

Partition Count	CPU	Memory	Network
1--4	1 vCPU	512 MB	1 Gbps
5--16	2 vCPU	1 GB	5 Gbps
17--64	4 vCPU	2 GB	10 Gbps
65+	Scale horizontally	—	—

> **TIP**
>
> OSO Kafka Backup is written in Rust and is designed to use less than 500 MB of memory for 4 partitions. The sizing table above includes headroom for bursts and garbage-free operation.

Kubernetes Resource Configuration

Set both requests and limits to ensure predictable scheduling and prevent noisy-neighbour effects:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kafka-backup
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kafka-backup
  template:
    metadata:
      labels:
        app: kafka-backup
    spec:
      containers:
        - name: kafka-backup
          image: ghcr.io/osodevops/kafka-backup:latest
          resources:
            requests:
              cpu: "1"
              memory: "512Mi"
            limits:
              cpu: "2"
              memory: "1Gi"
          volumeMounts:
            - name: config
              mountPath: /etc/kafka-backup
      volumes:
        - name: config
          configMap:
            name: kafka-backup-config
```

Horizontal Pod Autoscaling

Scale based on backup lag to ensure partitions are covered as throughput increases:

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: kafka-backup-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: kafka-backup
  minReplicas: 1
  maxReplicas: 8
  metrics:
    - type: Pods
      pods:
        metric:
          name: kafka_backup_lag_bytes
        target:
          type: AverageValue
          averageValue: "104857600"  # 100 MB average lag
```

Anti-patterns

ANTI-PATTERNS

- **No resource limits** — Without limits, a misbehaving backup instance can consume all node resources and affect other workloads.
- **No resource requests** — Without requests, the Kubernetes scheduler cannot make informed placement decisions, leading to resource contention.
- **Vertical scaling only** — A single large instance has a failure blast radius that affects all partitions. Horizontal scaling isolates failures.
- **Running backup on Kafka broker nodes** — Backup I/O competes with broker I/O for disk and network bandwidth, degrading both.

PE-05: Network Optimisation

Co-location

Deploy backup instances in the **same availability zone** as the Kafka brokers they read from. This provides the lowest latency and eliminates cross-AZ data transfer charges.

WARNING

Cross-AZ data transfer in AWS costs $0.01/GB in each direction. For a topic producing 1 TB/day, this adds approximately $600/month in transfer costs alone.

Private Network Paths

- **VPC endpoints** — Use gateway or interface VPC endpoints for S3 and other storage services to keep traffic off the public internet.
- **Rack awareness** — Configure `client.rack` so that Kafka directs fetch requests to the closest replica, reducing cross-rack and cross-AZ traffic.

Fetch Size Optimisation

Increase `fetch_max_bytes` to reduce the number of fetch round trips:

```
kafka:
  fetch_max_bytes: 52428800  # 50 MB
  fetch_max_wait_ms: 500
```

> **TIP**
>
> Larger fetch sizes amortise the per-request overhead but increase memory usage. Monitor memory consumption when tuning fetch sizes upward.

Cross-Region Restores

When restoring data to a different region:

- Enable **S3 Transfer Acceleration** or equivalent service to optimise long-distance transfers.
- Consider pre-staging backup data to the target region using storage replication before initiating the restore.

Anti-patterns

ANTI-PATTERNS

- **Backup in a different region from Kafka** — Adds latency to every fetch request and segment upload, reducing throughput and increasing costs.
- **Small fetch sizes with high-throughput topics** — Causes excessive round trips to brokers, wasting network capacity on request overhead.
- **Shared network without QoS** — Backup traffic can saturate shared links, affecting production Kafka clients.
- **No consideration for cross-AZ costs** — Multi-AZ deployments are resilient but expensive. Understand the cost trade-off and optimise placement accordingly.

PE-06: Benchmarking & Load Testing

Benchmark Suite

Use the kafka-backup-demos benchmark suite to measure performance under controlled conditions.

```
# Clone the benchmark suite
git clone https://github.com/osodevops/kafka-backup-demos.git
cd kafka-backup-demos

# Run the throughput benchmark
./benchmarks/run-throughput-test.sh --partitions 4 --message-
size 1024 --duration 300
```

Benchmark Scenarios

Run the following scenarios to build a complete performance profile:

Scenario	Purpose
Max throughput per partition	Establish single-partition ceiling
Multi-partition scaling	Validate linear scaling with partition count
Large messages (> 1 MB)	Identify buffer and timeout issues
Restore speed	Measure time-to-recovery for capacity planning
WAN latency simulation	Understand cross-region performance impact

Record Baselines

For each scenario, record:

- **Throughput** — MB/s sustained over the test duration
- **Duration** — Total time to back up or restore the test dataset
- **Resource utilisation** — CPU, memory, network, and disk I/O during the test

Store baselines in version control alongside your backup configuration so they can be compared over time.

When to Re-run Benchmarks

- After **version upgrades** of `kafka-backup`
- After **configuration changes** to segment size, compression, or concurrency
- After **infrastructure changes** such as instance type, network topology, or storage backend
- **Quarterly** as a routine check against baseline drift

> ⚠ **WARNING**
>
> Benchmarks run on small data volumes (< 1 GB) do not represent production performance. Use datasets that are at least 10x the segment

size and run for a minimum of 5 minutes to capture steady-state behaviour.

Anti-patterns

ANTI-PATTERNS

- **No benchmarking before production** — Deploying without performance validation is deploying blind. Every environment has unique characteristics that affect throughput.
- **Benchmarking with unrealistically small data volumes** — Small datasets fit entirely in OS page cache, producing misleadingly high throughput numbers.
- **Benchmarking backup only, not restore** — Restore performance is equally critical and often has different bottlenecks (e.g., Kafka producer throughput to the target cluster).
- **Benchmarking on different hardware than production** — Results from a developer laptop do not predict production performance.

Review Questions

Use the following questions during architecture reviews to assess performance efficiency:

1. Have you established throughput targets (MB/s per partition) and validated them through benchmarking?
2. Is compression enabled, and have you chosen the algorithm and level based on your data format?
3. Is the storage backend appropriate for your durability, throughput, and cost requirements?

4. Are resource requests and limits set for all backup workloads running in Kubernetes?
5. Are backup instances co-located with Kafka brokers in the same availability zone?
6. Are VPC endpoints or private network paths used for storage access?
7. Have you benchmarked restore performance in addition to backup performance?
8. Do you re-run benchmarks after configuration changes, version upgrades, and on a quarterly schedule?
9. Is horizontal scaling used instead of vertical scaling for high partition counts?
10. Are you monitoring `kafka_backup_compression_ratio`, throughput, and resource utilisation continuously?

Resources

- Performance Tuning Guide — Step-by-step tuning for throughput and latency
- Configuration Reference — Complete YAML configuration options
- Metrics Reference — Full list of Prometheus metrics for performance monitoring
- Compression Architecture — How compression is implemented in the backup pipeline
- Zero-Copy Optimisation — Rust zero-copy design for minimal overhead

Cost Optimisation

> *"Managing and reducing the total cost of Kafka backup operations through efficient resource utilisation, smart storage tiering, and cost-aware architecture decisions."*

Backup infrastructure can become a significant and often invisible cost centre. Without active cost management, storage grows unbounded, oversized compute runs around the clock, and network transfer charges accumulate unnoticed. The Cost Optimisation pillar ensures you achieve reliable data protection at the lowest reasonable cost by continuously measuring, right-sizing, and governing every component of your backup architecture.

Design Principles

1. **Implement cloud financial management** — Tag every backup resource, track spending against budgets, and make cost data visible to the teams that control it.

2. **Adopt a consumption model** — Pay only for what you use. Scale compute to match backup windows, use lifecycle policies to tier storage, and avoid paying for idle capacity.

3. **Measure overall efficiency** — Define a unit cost metric such as cost per GB backed up. Track it over time and use it to evaluate architecture changes.

4. **Stop spending on undifferentiated heavy lifting** — Use managed object storage (S3, GCS, Azure Blob) rather than self-hosted storage. Let the cloud provider handle durability, availability, and scaling.

5. **Analyse and attribute expenditure** — Break down backup costs by team, environment, and topic. Attribution drives accountability and surfaces optimisation opportunities.

6. **Right-size retention to actual business need** — Not all data needs the same retention period. Match retention to compliance, operational, and business requirements rather than applying a single blanket policy.

Best Practices

CO-01: Storage Cost Management

What

Minimise storage costs through lifecycle policies, compression, deduplication, and continuous monitoring — without compromising data durability or restore capability.

Why

Storage is typically the largest component of backup cost. A single unmanaged S3 bucket can grow from manageable to expensive within months. Lifecycle policies alone can reduce long-term storage costs by 95% compared to keeping everything in the default storage class.

Implementation Guidance

AWS S3 Lifecycle Tiers

Age	Storage Class	Approx. Cost (USD/GB/mo)	Use Case
0–30 days	S3 Standard	$0.023	Active backups, frequent restores
31–90 days	S3 Standard-IA	$0.0125	Infrequent access, still fast retrieval
91–365 days	S3 Glacier Instant Retrieval	$0.004	Archival with millisecond access
1+ years	S3 Glacier Deep Archive	$0.00099	Long-term compliance, rare access

Azure Blob Storage Tiers

Age	Access Tier	Use Case
0–30 days	Hot	Active backups

Age	Access Tier	Use Case
31–90 days	Cool	Infrequent access
91–365 days	Cold	Archival with moderate retrieval time
1+ years	Archive	Long-term compliance

GCS Storage Classes

Age	Storage Class	Use Case
0–30 days	Standard	Active backups
31–90 days	Nearline	Monthly access pattern
91–365 days	Coldline	Quarterly access pattern
1+ years	Archive	Annual access or compliance

Compression and deduplication:

- Enable compression in `kafka-backup` for a typical **3–5x reduction** in stored data size
- Use deduplication to eliminate redundant segments across incremental backups
- Automate deletion of backups beyond retention policy

Understand total cost components:

- Storage at rest (the largest component)
- API calls (PUT, GET, LIST operations)
- Data transfer (retrieval and cross-region)
- Compute (the backup process itself)

TIP

> Enable S3 Intelligent-Tiering for buckets where access patterns are unpredictable. It automatically moves objects between tiers based on access frequency, with no retrieval fees for the frequent and infrequent access tiers.

Configuration

S3 Lifecycle Policy:

```json
{
  "Rules": [
    {
      "ID": "kafka-backup-lifecycle",
      "Status": "Enabled",
      "Filter": {
        "Prefix": "backups/"
      },
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 90,
          "StorageClass": "GLACIER_IR"
        },
        {
          "Days": 365,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "Expiration": {
        "Days": 2555
      }
    }
  ]
}
```

kafka-backup compression configuration:

```yaml
backup:
  compression:
    enabled: true
    algorithm: zstd
    level: 3
  storage:
    s3:
      bucket: my-kafka-backups
      region: eu-west-1
      storage-class: STANDARD
```

Anti-patterns

ANTI-PATTERNS

- **All Standard, forever** — Storing every backup in S3 Standard with no lifecycle policy. A 10 TB dataset costs ~$230/month in Standard vs ~$10/month in Deep Archive.
- **No lifecycle policies** — Relying on manual cleanup that never happens. Storage grows linearly and silently.
- **No cost monitoring** — Discovering a $5,000/month storage bill during quarterly budget review instead of catching it at $500.
- **No compression** — Storing uncompressed Kafka segments when 3–5x compression is available with minimal CPU overhead.

CO-02: Compute Right-Sizing

What

Match compute resources to actual backup workload requirements, scaling up for backup windows and down during idle periods.

Why

Backup workloads are inherently bursty. Running large instances 24/7 for a workload that peaks during a two-hour backup window wastes 90% of the compute spend.

Implementation Guidance

- **Start with PE-04 sizing guidance** — Use the performance efficiency pillar's sizing recommendations as a baseline, then refine based on observed utilisation.
- **Monitor actual utilisation** — Track CPU, memory, and network usage during backup runs. If peak utilisation is below 50%, you are over-provisioned.
- **Right-size with 20–30% headroom** — Allow enough capacity to handle spikes without throttling, but no more.

Kubernetes requests and limits:

```yaml
resources:
  requests:
    cpu: "500m"
    memory: "512Mi"
  limits:
    cpu: "2000m"
    memory: "2Gi"
```

- **Use Spot/Preemptible instances for non-critical workloads** — Validation jobs and development backups tolerate interruption. Enable checkpointing so interrupted jobs resume rather than restart.

```yaml
# Kubernetes node affinity for Spot instances
affinity:
  nodeAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 80
        preference:
          matchExpressions:
            - key: node.kubernetes.io/instance-type
              operator: In
              values:
                - spot
```

- **Schedule scale-up during backup windows** — Use Kubernetes CronJobs or cluster autoscaler profiles to add capacity before backup runs and release it afterwards.
- **Leverage Rust efficiency** — `kafka-backup` is written in Rust, which delivers significantly lower CPU and memory consumption compared to JVM-based

alternatives. This translates directly into smaller instance sizes and lower compute costs.

TIP

Combine Spot instances with checkpointing for development and validation workloads. If a Spot instance is reclaimed, the job resumes from the last checkpoint rather than restarting from scratch.

Anti-patterns

ANTI-PATTERNS

- **Large instances 24/7 for a 2-hour daily backup** — Running an `m5.4xlarge` around the clock when a `m5.xlarge` during the backup window would suffice.
- **No utilisation monitoring** — Provisioning based on initial estimates and never revisiting. Workloads change; compute allocation should too.
- **Ignoring Spot/Preemptible** — Paying full on-demand price for fault-tolerant workloads that can run on Spot at 60–90% discount.

CO-03: Network Transfer Costs

What

Minimise data transfer charges by co-locating backup components, using private endpoints, and compressing data before transmission.

Why

Cloud providers charge for data that crosses availability zone, region, or internet boundaries. For high-throughput Kafka clusters, transfer costs can rival or exceed storage costs if not managed carefully.

Implementation Guidance

Typical transfer pricing (AWS):

Path	Approx. Cost (USD/GB)
Same AZ (private IP)	Free
Cross-AZ	$0.01
Cross-region	$0.02–$0.09
Internet egress	$0.05–$0.12

- **Co-locate in the same AZ** — Run `kafka-backup` in the same availability zone as your Kafka brokers. This eliminates cross-AZ charges for the data read path.

```yaml
# Pod topology constraint for same-AZ placement
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: DoNotSchedule
    labelSelector:
      matchLabels:
        app: kafka-backup
```

- **Use VPC endpoints** — Access S3, GCS, or Azure Blob via private endpoints rather than public internet. This eliminates NAT gateway charges and reduces latency.

```bash
# Create S3 VPC Gateway Endpoint
aws ec2 create-vpc-endpoint \
  --vpc-id vpc-abc123 \
  --service-name com.amazonaws.eu-west-1.s3 \
  --route-table-ids rtb-abc123
```

- **Backup locally, then replicate** — Write backups to a storage bucket in the same region as Kafka. Use storage-native cross-region replication (e.g., S3 CRR) for DR copies — this is cheaper and more reliable than backing up directly to a remote region.
- **Compression reduces transfer cost proportionally** — A 4x compression ratio means 75% less data traverses the network, reducing transfer charges by the same proportion.

Anti-patterns

ANTI-PATTERNS

- **Direct cross-region backup** — Streaming backup data from `eu-west-1` Kafka to a `us-east-1` bucket, paying cross-region transfer on every byte.
- **No VPC endpoints** — Routing S3 traffic through a NAT gateway, paying $0.045/GB for NAT processing on top of transfer charges.
- **No transfer cost accounting** — Tracking storage costs but ignoring the transfer charges that can exceed them.

CO-04: Backup Retention Policies

What

Define retention periods per topic or data classification, automating the deletion of backups that are no longer needed.

Why

Keeping all backups indefinitely is the single largest driver of runaway storage costs. A well-designed retention policy reduces storage by 60–80% while still meeting every compliance and operational requirement.

Implementation Guidance

Retention tiers by data classification:

Tier	Example Topics	Retention	Justification
Compliance	`financial-transactions`, `audit-log`	7 years	Regulatory requirement (e.g., SOX, MiFID II)
Critical	`orders`, `payments`, `customer-updates`	90 days	Operational recovery and dispute resolution
Standard	`user-events`, `page-views`, `click-stream`	30 days	Analytics replay, short-term debugging
Ephemeral	`logs`, `metrics`, `health-checks`	7 days	Troubleshooting only, easily regenerated

Automate with CRD configuration:

```yaml
apiVersion: kafka-backup.io/v1
kind: BackupPolicy
metadata:
  name: tiered-retention
spec:
  policies:
    - topicPattern: "financial-*"
      retention:
        maxAge: 2555d
    - topicPattern: "orders|payments|customer-*"
      retention:
        maxAge: 90d
    - topicPattern: "user-events|click-*"
      retention:
        maxAge: 30d
        maxCount: 30
    - topicPattern: "logs|metrics|health-*"
      retention:
        maxAge: 7d
        maxCount: 7
```

- **Use S3 lifecycle as a safety net** — Even with application-level retention, set a bucket-level expiration policy as a backstop to catch anything the application misses.

- **Audit retention quarterly** — Review which topics are being backed up, how much storage each consumes, and whether the retention period is still appropriate.
- **Document justification** — Record why each retention period was chosen. Compliance requirements change; undocumented policies cannot be reviewed.
- **Support legal hold** — Ensure your retention automation can be overridden for specific backups when legal hold is required (e.g., litigation or regulatory investigation).

> **⚠ WARNING**
>
> Automated deletion is irreversible. Before enabling retention policies, verify that your compliance team has signed off on the retention periods for regulated data.

Anti-patterns

> **ANTI-PATTERNS**
>
> - **Keep everything forever** — The most expensive and least compliant approach. Indefinite retention increases both cost and risk surface.
> - **Manual cleanup** — Relying on an engineer to periodically delete old backups. This never happens consistently.
> - **Same retention for all topics** — Applying a 7-year retention to ephemeral logs because "it's easier than classifying topics".

CO-05: Cost Visibility & Governance

What

Implement tagging, dashboards, budgets, and review processes that make backup costs transparent and actionable.

Why

You cannot optimise what you cannot see. Without visibility, backup costs are absorbed into general cloud spend, optimisation opportunities go unnoticed, and there is no accountability for cost growth.

Implementation Guidance

Tag all resources consistently:

```
# Standard tagging schema
tags:
  team: platform-engineering
  environment: production
  project: kafka-backup
  cost-centre: CC-4521
  managed-by: terraform
```

Build cost dashboards that show:

- Total backup cost per month (trend over 6+ months)
- Breakdown by category: storage, compute, network
- Cost per GB backed up (efficiency metric)
- Cost by environment (production vs staging vs development)
- Month-over-month growth rate

Set alerts and budgets:

```
# AWS Budget example
aws budgets create-budget \
  --account-id 123456789012 \
  --budget '{
    "BudgetName": "kafka-backup-monthly",
    "BudgetLimit": {"Amount": "500", "Unit": "USD"},
    "TimeUnit": "MONTHLY",
    "BudgetType": "COST",
    "CostFilters": {
      "TagKeyValue": ["user:project$kafka-backup"]
    }
  }' \
  --notifications-with-subscribers '[{
    "Notification": {
      "NotificationType": "ACTUAL",
      "ComparisonOperator": "GREATER_THAN",
      "Threshold": 80
    },
    "Subscribers": [{
      "SubscriptionType": "EMAIL",
      "Address": "platform-team@example.com"
    }]
  }]'
```

- **Monthly cost review** — Include backup costs in your team's monthly operational review. Compare actual spend against budget and investigate variances.
- **Cost per topic** — Where possible, attribute storage costs to individual topics. This surfaces topics that are disproportionately expensive and drives conversations about retention and compression.

> **TIP**
>
> Use AWS Cost Explorer tag filtering, Azure Cost Management scopes, or GCP billing labels to create dedicated backup cost views without building custom dashboards.

Anti-patterns

> **ANTI-PATTERNS**

- **No tagging** — Backup resources are untagged, making it impossible to separate backup costs from general infrastructure spend.
- **Lumped into general spend** — Backup costs are not broken out, so no one knows whether the $2,000/month increase is from backups, compute, or something else entirely.
- **No budget or alerts** — The team discovers cost overruns during quarterly business review instead of when they happen.

Review Questions

Use these questions to evaluate the cost optimisation of your Kafka backup architecture:

1. Do you have lifecycle policies configured on all backup storage buckets?
2. Can you state the cost per GB of backed-up data for each environment?
3. Are compute resources right-sized to actual backup workload utilisation, with no more than 30% idle headroom?
4. Are you using Spot or Preemptible instances for non-critical backup workloads?
5. Is `kafka-backup` co-located in the same availability zone as the Kafka brokers it reads from?
6. Are VPC endpoints configured for all storage access paths?
7. Do you have documented and automated retention policies for every backed-up topic?
8. Are all backup resources tagged with a consistent schema (team, environment, project, cost-centre)?
9. Do you have budget alerts that fire before costs exceed your planned spend?
10. Is backup cost reviewed as a standing agenda item in your monthly operational review?

Resources

- AWS S3 Pricing
- AWS Data Transfer Pricing

- Azure Blob Storage Pricing
- Google Cloud Storage Pricing
- AWS Cost Explorer
- Azure Cost Management
- GCP Billing Reports
- kafka-backup Configuration Reference

Sustainability

> *"Minimising the environmental impact of Kafka backup operations through efficient resource utilisation, data lifecycle management, and considered infrastructure choices."*

Every compute cycle, stored byte, and network transfer has an environmental cost. While individual backup workloads are modest in isolation, the cumulative impact across environments, regions, and retention periods adds up. The Sustainability pillar helps you reduce this footprint by making deliberate choices about how, where, and how long you store and process backup data.

Design Principles

1. **Understand your impact** — Measure the carbon footprint of your backup infrastructure using cloud-provider tools. You cannot reduce what you do not measure.

2. **Maximise utilisation** — Right-size compute resources, use efficient compression algorithms, and eliminate idle capacity. Higher utilisation means less wasted energy per unit of useful work.

3. **Adopt more efficient technology** — `kafka-backup` is written in Rust, which delivers significantly lower CPU and memory consumption compared to JVM-based alternatives. Choosing inherently efficient tooling reduces energy consumption at the source.

4. **Reduce downstream impact** — Efficient storage tiering moves data to lower-energy cold storage over time, reducing the ongoing energy required to maintain backups.

Best Practices

SUS-01: Efficient Resource Utilisation

What

Minimise the compute, memory, and energy consumed per unit of backed-up data by right-sizing resources, using efficient algorithms, and eliminating waste.

Why

Over-provisioned infrastructure consumes energy whether it is doing useful work or not. A right-sized, efficiently compressed backup pipeline can deliver the same data protection with a fraction of the environmental footprint.

Implementation Guidance

- **Rust is inherently efficient** — `kafka-backup` is written in Rust, which compiles to native machine code with no garbage collection overhead. This means significantly lower CPU and memory consumption compared to JVM-based backup tools, translating directly into smaller instances and lower energy usage.

- **Right-size compute resources** — Follow the guidance in PE-04 to size instances based on actual workload. Monitor utilisation and downsize when headroom exceeds 30%.

```yaml
# Right-sized resource allocation
resources:
  requests:
    cpu: "500m"
    memory: "512Mi"
  limits:
    cpu: "2000m"
    memory: "2Gi"
```

- **Enable autoscaling** — Scale compute to match backup windows rather than running at peak capacity 24/7. Use Kubernetes Horizontal Pod Autoscaler or cluster autoscaler to add and remove capacity dynamically.

- **Use efficient compression** — `zstd` compression reduces stored data by 3–5x with minimal CPU overhead, reducing both storage energy and transfer energy.

```yaml
backup:
  compression:
    enabled: true
    algorithm: zstd
    level: 3
```

- **Choose regions with lower carbon intensity** — Where latency and data residency requirements allow, prefer regions powered by renewable energy.

> **TIP**
>
> The combination of Rust's efficiency, right-sized compute, and zstd compression can reduce the energy footprint of your backup pipeline by 5–10x compared to a naively provisioned JVM-based alternative.

Anti-patterns

> **ANTI-PATTERNS**
>
> - **Large instances running 24/7** — Running oversized compute around the clock for a workload that runs for two hours per day wastes 90% of the energy consumed.
> - **No compression** — Storing and transferring uncompressed data when efficient compression is available at negligible CPU cost.
> - **No region consideration** — Deploying backup infrastructure based solely on latency without considering the carbon intensity of the region's energy grid.

SUS-02: Data Lifecycle & Minimisation

What

Back up only the data you need, retain it only as long as required, and match recovery granularity to actual business requirements.

Why

Every byte stored consumes energy — for storage media, cooling, and redundancy. Reducing the volume of stored data through selective backup, appropriate retention, and right-sized granularity directly reduces the ongoing energy cost of your backup infrastructure.

Implementation Guidance

- **Implement retention policies** — Follow the tiered retention guidance in CO-04 to ensure data is deleted when it is no longer needed. Every deleted backup is energy that no longer needs to be spent on storage.

- **Use topic filtering** — Back up only the topics that require protection. Exclude ephemeral topics, internal Kafka topics, and any topic whose data can be trivially regenerated.

```yaml
backup:
  topics:
    include:
      - "orders.*"
      - "payments.*"
      - "customer.*"
    exclude:
      - ".*\\.internal"
      - "logs\\.debug.*"
      - "__consumer_offsets"
```

- **Match PITR granularity to actual needs** — Point-in-time recovery with minute-level granularity generates far more backup data than hourly granularity. Choose the granularity your RTO and RPO actually require, not the finest granularity available.

- **Archive to cold storage tiers** — Move older backups to cold storage (Glacier, Archive, Coldline). Cold storage tiers consume less energy per byte than hot storage because they use denser, less frequently accessed media.

> ⚠ **WARNING**

> Before excluding topics from backup, confirm with application owners that the data is genuinely ephemeral or regenerable. Excluding a topic that turns out to be critical is not recoverable.

Anti-patterns

> **ANTI-PATTERNS**
>
> - **Backing up all topics indiscriminately** — Including internal Kafka topics, debug logs, and ephemeral streams that have no recovery value.
> - **No retention policies** — Storing backups indefinitely, consuming energy for data that will never be accessed again.
> - **Over-specifying granularity** — Configuring minute-level PITR for a workload where hourly recovery is perfectly acceptable, generating 60x more backup data than necessary.

SUS-03: Region & Storage Tier Selection

What

Choose cloud regions and storage tiers that minimise the carbon intensity of your backup infrastructure.

Why

Cloud regions differ significantly in carbon intensity depending on the local energy grid. A backup stored in a region powered primarily by renewables has a materially lower carbon footprint than the same backup in a coal-heavy region — with identical durability and availability.

Implementation Guidance

Prefer regions with renewable energy:

Provider	Lower-Carbon Regions
AWS	`eu-west-1` (Ireland), `eu-north-1` (Stockholm)
Azure	North Europe (Ireland), Sweden Central
GCP	`europe-north1` (Finland), `us-central1` (Iowa)

> **TIP**
>
> AWS, Azure, and GCP all publish sustainability commitments and region-level carbon data. Use this information when choosing where to deploy backup infrastructure, especially for DR copies that are latency-insensitive.

- **Use cold storage for long-term backups** — Cold and archive storage tiers use denser storage media that consumes less energy per byte. For backups older than 90 days that are rarely accessed, cold storage is both cheaper and more sustainable.

- **Track your carbon footprint** — Use cloud-provider tools to measure the emissions attributable to your backup infrastructure:

```
# AWS: View Carbon Footprint in the Billing Console
# Navigate to: AWS Billing > Carbon Footprint

# Azure: View emissions via Emissions Dashboard
# Navigate to: Azure Portal > Carbon Optimization

# GCP: View Carbon Footprint in the Console
# Navigate to: GCP Console > Carbon Footprint
```

- **Set sustainability targets** — Track carbon emissions per GB of backed-up data as a sustainability KPI. Review quarterly and set reduction targets aligned with your organisation's sustainability commitments.

Anti-patterns

ANTI-PATTERNS

- **No carbon consideration in region selection** — Choosing regions based solely on cost or latency without evaluating carbon intensity.
- **All hot storage, all the time** — Keeping every backup in hot/standard storage tiers when the vast majority will never be accessed after 30 days.
- **No emissions tracking** — Operating backup infrastructure without any visibility into its environmental impact.

Review Questions

Use these questions to evaluate the sustainability of your Kafka backup architecture:

1. Do you know the carbon intensity of the regions where your backup infrastructure runs?
2. Are compute resources right-sized to actual workload utilisation, avoiding persistent over-provisioning?
3. Is compression enabled to reduce both storage volume and transfer energy?
4. Do you have retention policies that automatically delete backups when they are no longer needed?
5. Are you using cloud-provider carbon footprint tools to track and report on the emissions of your backup infrastructure?

Resources

- AWS Customer Carbon Footprint Tool
- Azure Emissions Dashboard
- Google Carbon Footprint
- The Green Web Foundation
- Cloud Carbon Footprint (open source)

Cross-Cutting Concerns

> *Architecture considerations that span multiple pillars of the Well-Architected Framework — topics that affect security, reliability, cost, operational excellence, and performance simultaneously.*

Some architectural decisions do not fit neatly into a single pillar. They cut across every dimension of the Well-Architected Framework and must be addressed holistically. This page covers the most important cross-cutting concerns for organisations running OSO Kafka Backup in production.

Multi-Cloud & Hybrid Deployments

Many organisations operate Kafka across multiple clouds or hybrid on-prem/cloud environments. Your backup strategy must account for heterogeneous infrastructure, differing credential models, and the realities of cross-cloud networking.

Key Considerations

- **Storage portability** — kafka-backup supports S3, Azure Blob, GCS, and local filesystem. Backups created in one cloud can be restored in another.
- **Credential management across clouds** — Each cloud has its own identity model (IAM roles, managed identities, workload identity). Backup configs must handle these differences.
- **Network connectivity and latency** — Cross-cloud restores introduce network hops and potential bandwidth constraints.
- **Data sovereignty and residency requirements** — Regulations may restrict where backup data can be stored or transferred.
- **Cost of cross-cloud data transfer** — Egress charges can be significant when replicating backups between providers.

Recommended Patterns

- **Back up locally, replicate cross-cloud for DR** — Perform primary backups to the same cloud as the source cluster, then replicate to a secondary cloud for disaster recovery.
- **Use S3-compatible storage (MinIO) as a universal intermediate format** — MinIO provides an S3-compatible API that runs on any cloud or on-prem, giving

you a consistent storage interface.
- **Consistent config across clouds using GitOps** — Store backup configurations in Git and deploy them identically across environments to reduce drift.

> **TIP**
>
> Start with local backups to minimise latency and cost, then add cross-cloud replication as a second stage. This avoids paying egress fees on every backup cycle.

Configuration Example

Multi-cloud backup with primary S3 and secondary Azure Blob for DR:

```yaml
# Primary backup — S3 in AWS (same region as source Kafka)
backup:
  name: production-primary
  source:
    bootstrap-servers: "${KAFKA_BOOTSTRAP_SERVERS}"
    security-protocol: SASL_SSL
    sasl-mechanism: SCRAM-SHA-512
    sasl-username: "${KAFKA_USERNAME}"
    sasl-password: "${KAFKA_PASSWORD}"
  storage:
    type: s3
    bucket: "${AWS_BACKUP_BUCKET}"
    region: "${AWS_REGION}"
    prefix: kafka-backup/production
  topics:
    include:
      - ".*"

---
# Secondary backup — Azure Blob for cross-cloud DR
backup:
  name: production-dr
  source:
    bootstrap-servers: "${KAFKA_BOOTSTRAP_SERVERS}"
    security-protocol: SASL_SSL
    sasl-mechanism: SCRAM-SHA-512
    sasl-username: "${KAFKA_USERNAME}"
    sasl-password: "${KAFKA_PASSWORD}"
  storage:
    type: azure-blob
    container: "${AZURE_BACKUP_CONTAINER}"
    account-name: "${AZURE_STORAGE_ACCOUNT}"
    account-key: "${AZURE_STORAGE_KEY}"
    prefix: kafka-backup/production
  topics:
    include:
      - ".*"
```

> ⚠ **WARNING**
>
> Cross-cloud credential environment variables must be managed carefully. Use a secrets manager (Vault, AWS Secrets Manager, Azure Key Vault) rather than storing credentials directly in config files or CI/CD pipelines.

Kubernetes-Native Operations

kafka-backup provides a Kubernetes operator with Custom Resource Definitions (CRDs) for GitOps-native backup management. This allows you to declare backup and restore jobs as Kubernetes resources, managed alongside your application manifests.

Key Concepts

CRD	Purpose
KafkaBackup	Defines a backup job — source cluster, storage target, schedule, and topic filters
KafkaRestore	Defines a restore job — source backup, target cluster, and restore parameters
KafkaOffsetReset	Manages consumer offset recovery after a restore operation
KafkaOffsetRollback	Rolls back offset changes if a reset produces unexpected results

Best Practices

- **Store CRDs in Git alongside application manifests** — Backup definitions should live in the same repository as the services that produce and consume the data.
- **Use ArgoCD or Flux for GitOps deployment** — Automate CRD deployment through your existing GitOps pipeline.
- **Define resource requests and limits** — Prevent backup pods from starving other workloads or being OOM-killed during large backups.
- **Use Kubernetes RBAC to control who can create restore CRDs** — Restores are destructive operations; limit access to authorised personnel.
- **Monitor CRD status with kubectl and Prometheus** — The operator exposes metrics and CRD status conditions for observability.

INFO

The Kubernetes operator watches for CRD changes and reconciles the desired state automatically. This means you can trigger a backup or restore simply by applying a manifest — no imperative commands required.

Configuration Examples

KafkaBackup CRD:

```yaml
apiVersion: kafka.oso.dev/v1alpha1
kind: KafkaBackup
metadata:
  name: production-daily
  namespace: kafka-backup
spec:
  schedule: "0 2 * * *"
  source:
    bootstrapServers: kafka-cluster-kafka-bootstrap:9093
    securityProtocol: SASL_SSL
    saslMechanism: SCRAM-SHA-512
    credentialsSecret:
      name: kafka-backup-credentials
  storage:
    type: s3
    bucket: my-backup-bucket
    region: eu-west-1
    prefix: production/daily
  topics:
    include:
      - ".*"
    exclude:
      - "__.*"
  resources:
    requests:
      cpu: 500m
      memory: 1Gi
    limits:
      cpu: "2"
      memory: 4Gi
```

KafkaRestore CRD:

```yaml
apiVersion: kafka.oso.dev/v1alpha1
kind: KafkaRestore
metadata:
  name: restore-production-20260324
  namespace: kafka-backup
spec:
  backup:
    name: production-daily
    snapshot: "2026-03-24T02:00:00Z"
  target:
    bootstrapServers: kafka-cluster-kafka-bootstrap:9093
    securityProtocol: SASL_SSL
    saslMechanism: SCRAM-SHA-512
    credentialsSecret:
      name: kafka-restore-credentials
  topics:
    include:
      - "orders.*"
      - "payments.*"
  restoreOffsets: true
  resources:
    requests:
      cpu: "1"
      memory: 2Gi
    limits:
      cpu: "4"
      memory: 8Gi
```

> **TIP**
>
> Use `kubectl get kafkabackup` and `kubectl get kafkarestore` to check the status of backup and restore operations. The operator sets status conditions such as `Ready`, `Running`, `Completed`, and `Failed`.

See also: Operator Overview, KafkaBackup CRD, KafkaRestore CRD, GitOps Guide

---

Schema Registry Integration

For clusters using a Schema Registry (Confluent, Apicurio), backup must include schemas alongside topic data. Without schemas, consumers cannot deserialise restored messages, and producers cannot validate new messages against the expected format.

Key Considerations

- **Schema IDs may not be preserved across clusters** — Schema IDs are auto-incremented integers assigned by the registry. A restore to a different cluster will likely produce different IDs.
- **Schema evolution history should be backed up** — Consumers may depend on older schema versions for backward compatibility.
- **Restore must handle schema ID remapping** — Messages reference schema IDs in their headers. After restore, these IDs must map to the correct schemas in the target registry.

> ⚠ **WARNING**
>
> Restoring topic data without its associated schemas will result in deserialisation failures for all Avro, Protobuf, or JSON Schema consumers. Always include schema backup in your DR plan.

Enterprise Feature

The Enterprise edition provides integrated Schema Registry backup and restore with:

- **Automatic ID remapping** — Schema IDs in restored messages are updated to match the target registry.
- **Compatibility validation** — Schemas are validated against the target registry's compatibility settings before restore.
- **Full evolution history** — All schema versions and their metadata are preserved.

Recommended Patterns

- **Back up Schema Registry independently** — Export schemas via the Schema Registry REST API as a supplementary backup.
- **Use Enterprise for integrated schema backup/restore** — The Enterprise edition handles the complexity of ID remapping and compatibility checks automatically.

- **Test schema compatibility after restore** — Verify that consumers can deserialise messages and producers can register new schemas.

See also: Schema Registry (Enterprise)

Kafka Streams & Stateful Applications

Kafka Streams applications maintain local state stores backed by changelog topics. Backup and restore of a Streams application requires special consideration to ensure the application can recover its state correctly.

Key Considerations

- **Changelog topics must be included in backup** — These topics are the source of truth for Streams state stores. Without them, the application must reprocess all input data from scratch.
- **State store rebuild time after restore** — Even with changelog topics restored, state stores must be rebuilt locally. Factor this time into your RTO calculations.
- **Repartition topics may need to be excluded** — Repartition topics are intermediate topics generated by Streams. They can be regenerated from input data and do not need to be backed up.
- **Consumer offset recovery is critical for Streams apps** — Streams applications use consumer offsets to track processing progress. Incorrect offsets can cause duplicate processing or data loss.

Recommended Patterns

- **Include all changelog topics in backup scope** — Use topic name patterns to capture changelog topics (typically suffixed with `-changelog`).
- **Exclude repartition topics** — Repartition topics (typically suffixed with `-repartition`) regenerate automatically and waste storage if backed up.
- **Test Streams app recovery as part of DR drills** — Streams recovery is more complex than simple consumer recovery. Validate it regularly.
- **Use PITR to restore to a consistent state across all related topics** — Point-in-time recovery ensures that input topics, changelog topics, and output topics are restored to the same logical point.

> **INFO**
>
> Kafka Streams state store rebuild time depends on the volume of data in the changelog topics. For large state stores, this can take minutes to hours. Plan accordingly and consider standby replicas to reduce recovery time.

Configuration Example

Topic filtering for Kafka Streams applications — include changelogs, exclude repartition topics:

```yaml
backup:
  name: streams-app-backup
  source:
    bootstrap-servers: kafka-cluster:9092
  storage:
    type: s3
    bucket: my-backup-bucket
    prefix: streams-app
  topics:
    include:
      # Input topics
      - "orders\\..*"
      - "payments\\..*"
      # Output topics
      - "enriched-orders"
      - "order-summaries"
      # Changelog topics (state stores)
      - "streams-app-.*-changelog"
    exclude:
      # Repartition topics (will regenerate)
      - "streams-app-.*-repartition"
      # Internal Streams topics
      - "__consumer_offsets"
      - "__transaction_state"
```

> **TIP**
>
> Use a naming convention for your Streams application ID (e.g., `streams-app-*`) so that changelog and repartition topics can be easily identified with wildcard patterns.

See also: Kafka Streams Example

Regulatory & Compliance Scenarios

Industries such as finance, healthcare, and retail have specific regulatory requirements for data backup, retention, and protection. kafka-backup can be configured to meet these requirements, with Enterprise features providing additional compliance capabilities.

Compliance Mapping

Regulation	Requirement	kafka-backup Feature
GDPR	Right to be forgotten, data minimisation	Data masking, field-level redaction (Enterprise)
SOX	Financial data retention (7 years)	Long-term retention with lifecycle policies
HIPAA	PHI protection, access logging	Encryption at rest, audit logging (Enterprise)
PCI DSS	Cardholder data protection	Field-level encryption, RBAC (Enterprise)
DORA	IT system resilience testing	DR testing framework, RTO/RPO tracking

Recommended Patterns

- **Define compliance requirements per topic** — Not all topics carry regulated data. Tag topics with their compliance classification and apply appropriate backup policies.
- **Use Enterprise features for regulated industries** — Field-level encryption, data masking, RBAC, and audit logging are essential for GDPR, HIPAA, and PCI DSS compliance.
- **Implement audit logging for all operations** — Every backup, restore, and configuration change should be logged with the operator identity, timestamp, and outcome.
- **Conduct regular compliance audits** — Periodically review backup configurations, retention policies, and access controls against regulatory requirements.
- **Maintain evidence of DR testing for auditors** — Regulators such as those enforcing DORA require documented evidence that disaster recovery procedures have been tested.

> **WARNING**
>
> Regulatory non-compliance can result in significant fines and reputational damage. Treat compliance requirements as hard constraints, not aspirational goals. If in doubt, consult your compliance or legal team before finalising backup configurations.

> **TIP**

> Use the Enterprise audit logging feature to generate compliance reports automatically. These reports can be exported in formats suitable for external auditors and regulators.

See also: Audit Logging (Enterprise), Encryption (Enterprise), RBAC (Enterprise), Compliance Audit Use Case

Reference Architectures

> *Proven deployment patterns that combine the principles from every Well-Architected pillar into end-to-end, production-ready configurations you can adopt or adapt.*

Each reference architecture below includes a complete topology, configuration, cost estimate, and known limitations so you can evaluate trade-offs before committing to a design. Pick the architecture closest to your constraints, then adjust RPO, RTO, and storage tiers to match your specific requirements.

Architecture Comparison

Architecture	RPO	RTO	Complexity	Est. Monthly Cost	Best For
1. Single-Region S3	< 1 hr	< 4 hr	Low	~$105	Single-region workloads, dev/staging
2. Cross-Region DR	< 15 min	< 1 hr	Medium	~$175	Multi-region availability, production DR
3. Multi-Cloud Active-Passive	< 1 hr	< 2 hr	High	~$265	Cloud-provider failure protection
4. Air-Gapped Compliance	< 24 hr	< 8 hr	High	~$195	Ransomware protection, regulatory compliance
5. Kubernetes GitOps Pipeline	< 1 hr	< 2 hr	Medium	~$105	K8s-native teams, declarative operations

> **TIP**
>
> Start with **Architecture 1** to validate your backup strategy, then evolve toward cross-region or multi-cloud patterns as your availability requirements grow. Each architecture builds on the configuration patterns established in the simpler designs.

Architecture 1: Single-Region Backup to S3

Overview

The simplest production-ready pattern. A single kafka-backup deployment runs continuously inside the same region as your Kafka cluster, streaming data to an S3 bucket with versioning enabled. Prometheus scrapes the built-in metrics endpoint for alerting and dashboards.

When to Use

- Single-region Kafka deployment
- **RPO < 1 hour** is acceptable
- **RTO < 4 hours** is acceptable
- You want the lowest operational overhead and cost
- Cross-region protection is not yet a requirement

Architecture Diagram

```
┌──────────────────────────────────────────────────────────────
│                        AWS Region (us-east-1)
│
│  ┌───────────────────────┐        ┌───────────────────────────┐  │
│  │   Kafka Cluster        │        │   Kubernetes Cluster       │
│
```

```
│  │                        │       │                            │
│
│  │  ┌─────┐  ┌─────┐     │       │  ┌──────────────────────┐ │
│  │  │ b-1 │  │ b-2 │     │ ───── │  │ kafka-backup         │ │
│  │  └─────┘  └─────┘     │       │  │ (Deployment, 1 pod)  │ │
│  │  ┌─────┐               │       │  └──────────┬───────────┘ │
│
│  │  │ b-3 │               │       │             │              │
│
│  │  └─────┘               │       │  ┌──────────┴───────────┐ │
│
│  └───────────────────────┘       │  │ Prometheus + Grafana │ │
│
│                                   │  └──────────────────────┘ │
│
│                                   └────────────────────────────┘
│                                                │
│
│                                                ▼
│
│                                   ┌───────────────────┐
│                                   │  S3 Bucket        │
│
│                                   │  (versioning on)  │
│
│                                   │  kafka-backup/    │
│
│                                   └───────────────────┘
│
└──────────────────────────────────────────────────────────────────
```

Components

Component	Purpose
Kafka cluster (3 brokers)	Source data
kafka-backup (K8s Deployment)	Continuous backup, 1 replica
S3 bucket (versioning enabled)	Backup storage, same region
Prometheus + Grafana	Metrics scraping, alerting, dashboards

Configuration

backup.yaml

```yaml
source:
  bootstrap_servers:
    - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
  topic:
    include:
      - ".*"    # back up all topics

storage:
  type: s3
  s3:
    bucket: my-org-kafka-backup
    region: us-east-1
    prefix: prod/

backup:
  compression: zstd
  segment_max_bytes: 134217728   # 128 MB
  continuous: true
  checkpoint_interval_secs: 60

metrics:
  enabled: true
  port: 9090
```

Kubernetes Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kafka-backup
  namespace: kafka-backup
  labels:
    app: kafka-backup
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kafka-backup
  template:
    metadata:
      labels:
        app: kafka-backup
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9090"
    spec:
      serviceAccountName: kafka-backup
      containers:
        - name: kafka-backup
          image: osodevops/kafka-backup:latest
          args: ["backup", "--config", "/etc/kafka-
backup/backup.yaml"]
          ports:
            - name: metrics
              containerPort: 9090
          resources:
            requests:
              cpu: 500m
              memory: 512Mi
            limits:
              cpu: "2"
              memory: 2Gi
          volumeMounts:
            - name: config
              mountPath: /etc/kafka-backup
              readOnly: true
      volumes:
        - name: config
          configMap:
            name: kafka-backup-config
```

IAM

Refer to SEC-01: Identity & Access Management for the least-privilege IAM policy. The backup role requires **write-only** access to S3 and **read-only** access to Kafka.

Cost Estimate

Item	Monthly Cost
S3 storage (~1 TB/day, 30-day retention, zstd compression)	~$70
Compute (1 pod, 2 vCPU / 2 GB)	~$30
Monitoring (Prometheus + Grafana)	~$5

Item	Monthly Cost
Total	**~$105**

Limitations

- No cross-region protection — a regional outage affects both source and backup
- Single storage backend — no redundancy if S3 experiences an availability event
- Restore must occur from the same region to avoid data transfer costs

---

Architecture 2: Cross-Region Disaster Recovery

Overview

Extends Architecture 1 with S3 Cross-Region Replication (CRR) to maintain a replica of all backup data in a secondary region. A standby kafka-backup instance in the DR region can restore data to a pre-provisioned DR Kafka cluster, achieving a significantly lower RTO than a cold-start approach.

When to Use

- Multi-region availability is required
- **RPO < 15 minutes** is required
- **RTO < 1 hour** is required
- You need protection against a full regional outage
- Regulatory requirements mandate geographically separated copies

Architecture Diagram

```
┌───────────────────────────────────┐
┌───────────────────────────────────┐
│      Primary Region                 │          │     DR Region
│
│         (us-east-1)                 │          │       (us-west-2)
│
│                                     │          │
│
```

```
│  ┌──────────┐  ┌────────────┐ │         │ ┌────────────┐
┌──────────┐ │
│  │  Kafka   │  │  kafka-    │ │         │ │  kafka-    │  │
DR     │ │
│  │  Cluster │──│  backup    │ │         │ │  backup    │──│
Kafka  │ │
│  │  (prod)  │  │  (backup)  │ │         │ │  (restore) │  │
Cluster │ │
│  └──────────┘  └─────┬──────┘ │         │ └─────┬──────┘
└──────────┘ │
│                      │        │         │       │
│
│               ┌──────▼───────┐ │  S3 CRR │ ┌─────▼───────┐
│
│               │  S3 Bucket   │─┼────────►│ │  S3 Bucket │
│
│               │  (primary)   │ │         │ │  (replica)  │
│
│               └──────────────┘ │         │ └─────────────┘
│
└───────────────────────────────┘
└───────────────────────────────┘
```

Components

Component	Purpose
Primary Kafka cluster	Production source data
kafka-backup (primary region)	Continuous backup to S3
S3 bucket (primary)	Primary backup storage with versioning
S3 Cross-Region Replication	Asynchronous replication to DR region
S3 bucket (DR region)	Replica backup storage
kafka-backup (DR region)	Standby restore instance
DR Kafka cluster	Pre-provisioned restore target

Configuration

Primary Region — backup.yaml

```yaml
source:
  bootstrap_servers:
    - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
  topic:
    include:
      - ".*"

storage:
  type: s3
  s3:
    bucket: my-org-kafka-backup-primary
    region: us-east-1
    prefix: prod/

backup:
  compression: zstd
  segment_max_bytes: 134217728
  continuous: true
  checkpoint_interval_secs: 60

metrics:
  enabled: true
  port: 9090
```

S3 Cross-Region Replication

```json
{
  "Role": "arn:aws:iam::123456789012:role/s3-crr-role",
  "Rules": [
    {
      "ID": "kafka-backup-crr",
      "Status": "Enabled",
      "Priority": 1,
      "Filter": {
        "Prefix": "prod/"
      },
      "Destination": {
        "Bucket": "arn:aws:s3:::my-org-kafka-backup-dr",
        "StorageClass": "STANDARD_IA"
      },
      "DeleteMarkerReplication": {
        "Status": "Disabled"
      }
    }
  ]
}
```

DR Region — restore.yaml

```yaml
source:
  type: s3
  s3:
    bucket: my-org-kafka-backup-dr
    region: us-west-2
    prefix: prod/

target:
  bootstrap_servers:
    - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
  topic:
    include:
      - ".*"

restore:
  from_latest: true
```

Cost Estimate

Item	Monthly Cost
Primary region (Architecture 1)	~$105
S3 Cross-Region Replication (transfer + storage)	~$50
DR standby compute	~$20
Total	**~$175**

Limitations

- S3 CRR replication lag (typically seconds to minutes) adds to effective RPO
- DR Kafka cluster incurs cost even when idle
- Manual or scripted failover — not automatic unless combined with health-check automation
- Cross-region data transfer costs increase with data volume

Architecture 3: Multi-Cloud Active-Passive DR

Overview

Protects against an entire cloud provider outage by maintaining backup data on a secondary cloud platform. The primary backup runs on AWS with S3 storage, while a cross-cloud sync process keeps an Azure Blob Storage copy up to date. A standby kafka-backup instance on Azure can restore to an Azure-hosted Kafka cluster.

When to Use

- Cloud provider failure protection is a business requirement
- **RPO < 1 hour** is acceptable
- **RTO < 2 hours** is acceptable
- Regulatory or contractual requirements mandate multi-cloud data residency
- Your organisation already operates infrastructure on multiple cloud providers

Architecture Diagram

```
┌──────────────────────────────┐
┌───────────────────────────────┐
│     AWS (us-east-1)           │          │     Azure (East US)
│
│                               │          │
│
│  ┌──────────┐ ┌─────────────┐ │         │ ┌─────────────┐
┌──────────┐│
│  │  Kafka   │ │  kafka-     │ │          │ │  kafka-     │ │
Azure    ││
│  │  Cluster │─│  backup     │ │          │ │  backup     │─│
Kafka    ││
│  │  (prod)  │ │  (backup)   │ │          │ │  (restore)  │ │
Cluster  ││
```

```
│  └──────────┘ └────────┬─────┘ │          │ └──────┬──────┘
└────────────┘ │
│                    │        │          │       │
│
│              ┌─────▼──────┐ │  rclone  │ ┌──────▼──────────┐
│
│              │ S3 Bucket  │─┼─────────►│ │  Blob Storage   │
│
│              │            │ │  sync    │ │  Container      │
│
│              └────────────┘ │          │ └─────────────────┘
│
└─────────────────────────────┘
└──────────────────────────────┘
```

Components

Component	Purpose
AWS Kafka cluster	Production source data
kafka-backup (AWS)	Continuous backup to S3
S3 bucket	Primary backup storage
Cross-cloud sync (rclone)	Scheduled sync from S3 to Azure Blob
Azure Blob Storage	Secondary backup storage
kafka-backup (Azure)	Standby restore instance
Azure Kafka cluster	DR restore target

Configuration

AWS — backup.yaml

```yaml
source:
  bootstrap_servers:
    - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
  topic:
    include:
      - ".*"

storage:
  type: s3
  s3:
    bucket: my-org-kafka-backup
    region: us-east-1
    prefix: prod/

backup:
  compression: zstd
  segment_max_bytes: 134217728
  continuous: true
  checkpoint_interval_secs: 60

metrics:
  enabled: true
  port: 9090
```

Azure — restore.yaml

```yaml
source:
  type: azure
  azure:
    storage_account: myorgkafkabackupdr
    container: kafka-backup
    prefix: prod/

target:
  bootstrap_servers:
    - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
  topic:
    include:
      - ".*"

restore:
  from_latest: true
```

Cross-Cloud Sync Script

```bash
#!/usr/bin/env bash
# sync-to-azure.sh — runs on a schedule (e.g., every 15 minutes
via cron or K8s CronJob)

set -euo pipefail

RCLONE_CONFIG="/etc/rclone/rclone.conf"
SOURCE="aws-s3:my-org-kafka-backup/prod/"
DEST="azure-blob:kafka-backup/prod/"

echo "[$(date -u)] Starting cross-cloud sync..."
rclone sync "$SOURCE" "$DEST" \
  --config "$RCLONE_CONFIG" \
  --transfers 16 \
  --checkers 32 \
  --fast-list \
  --log-level INFO

echo "[$(date -u)] Sync complete."
```

Cost Estimate

Item	Monthly Cost
Primary AWS (Architecture 1)	~$105
Azure Blob Storage	~$80
Cross-cloud sync (rclone compute + egress)	~$50
DR standby compute (Azure)	~$30
Total	**~$265**

Limitations

- Cross-cloud sync introduces complexity and a potential failure point
- Network egress costs (AWS to Azure) scale linearly with data volume
- Separate credential management for each cloud provider
- Sync lag adds to effective RPO — monitor rclone metrics closely

- Requires expertise in both AWS and Azure infrastructure

Architecture 4: Air-Gapped Compliance Backup

Overview

Provides ransomware-proof, tamper-proof backup storage for regulated industries. Backup data is written to a primary S3 bucket, then transferred to a completely isolated AWS account with S3 Object Lock (WORM — Write Once, Read Many). The air-gapped account has no VPC peering or network connectivity to the production environment, ensuring that a compromised production account cannot modify or delete backup data.

When to Use

- Ransomware protection is a top priority
- **RPO < 24 hours** is acceptable
- **RTO < 8 hours** is acceptable
- Regulatory requirements mandate immutable, tamper-proof backups (financial services, healthcare, government)
- Compliance frameworks require geographically or logically separated backup copies
- You need to demonstrate chain-of-custody for audit purposes

Architecture Diagram

```
┌──────────────────────────────────┐
┌───────────────────────────────────┐
│   Production Account              │         │   Air–Gapped
Account                  │
│                                   │         │   (no VPC
peering, no network)     │
│  ┌─────────┐  ┌───────────────┐  │         │
│
│  │  Kafka   │  │  kafka–backup │ │         │
┌───────────────────────────┐        │
│  │  Cluster │──│  (continuous) │ │         │  │  S3 Bucket
│    │
│  └─────────┘  └──────┬────────┘  │         │  │  (Object Lock /
WORM)    │    │
│                       │         │        │  │  (Glacier for
archive)    │    │
│               ┌───────▼───────┐ │ S3     │
└───────────────────────────┘    │
│               │  S3 Bucket    │─┼─Batch──│
│
│               │  (primary)    │ │ or     │
┌───────────────────────────┐    │
│               └───────────────┘ │ DataSync │  │  IAM: deny
all deletes   │    │
│                                  │        │  │  MFA-protected
root only  │    │
│  ┌──────────────────┐           │        │
└───────────────────────────┘    │
│  │  Prometheus +     │           │        │
│
│  │  Grafana          │           │        │
┌───────────────────────────┐    │
│  └──────────────────┘           │        │  │  CloudTrail
audit logging │    │
│                                  │        │
└───────────────────────────┘    │
└─────────────────────────────────┘
└──────────────────────────────────┘
```

Components

Component	Purpose
Kafka cluster	Production source data
kafka-backup (production account)	Continuous backup to primary S3
S3 bucket (primary)	Initial backup storage
AWS S3 Batch / DataSync	Scheduled transfer to air-gapped account
S3 bucket (air-gapped, Object Lock)	Immutable WORM storage
Glacier transition	Long-term archive for cost optimisation
CloudTrail (air-gapped account)	Audit logging for compliance

Configuration

Production Account — backup.yaml

```yaml
source:
  bootstrap_servers:
    - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
    - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
  topic:
    include:
      - ".*"

storage:
  type: s3
  s3:
    bucket: my-org-kafka-backup-prod
    region: us-east-1
    prefix: prod/

backup:
  compression: zstd
  segment_max_bytes: 134217728
  continuous: true
  checkpoint_interval_secs: 60

metrics:
  enabled: true
  port: 9090
```

S3 Object Lock Configuration (Air-Gapped Account)

```json
{
  "ObjectLockEnabled": "Enabled",
  "Rule": {
    "DefaultRetention": {
      "Mode": "COMPLIANCE",
      "Days": 365
    }
  }
}
```

> **INFO**
>
> **COMPLIANCE mode** prevents anyone — including the root user — from deleting or overwriting objects before the retention period expires. Use **GOVERNANCE mode** if you need the ability to override with special permissions during testing.

Air-Gapped Account IAM Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllDeleteOperations",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutBucketPolicy",
        "s3:DeleteBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::my-org-kafka-backup-airgap",
        "arn:aws:s3:::my-org-kafka-backup-airgap/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::111111111111:root"
        }
      }
    },
    {
      "Sid": "AllowWriteFromProductionAccount",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:role/kafka-backup-
transfer-role"
```

```
        },
        "Action": [
          "s3:PutObject",
          "s3:ListBucket"
        ],
        "Resource": [
          "arn:aws:s3:::my-org-kafka-backup-airgap",
          "arn:aws:s3:::my-org-kafka-backup-airgap/*"
        ]
      }
    ]
  }
```

Cost Estimate

Item	Monthly Cost
Primary backup (Architecture 1)	~$105
Air-gapped S3 storage (Glacier + Object Lock)	~$90
Total	**~$195**

TIP

Use S3 Intelligent-Tiering or lifecycle policies to transition older backups to Glacier Deep Archive after 90 days. This can reduce air-gapped storage costs by up to 70% for long-retention requirements.

Limitations

- Higher RTO due to the air gap — restoring requires transferring data back from the isolated account
- Transfer scheduling adds complexity (S3 Batch operations, DataSync jobs)

- MFA-protected root account access is required for emergency operations in the air-gapped account
- Object Lock retention cannot be shortened once set in COMPLIANCE mode
- Testing restores from the air-gapped account requires careful planning to avoid violating the air gap

---

Architecture 5: Kubernetes GitOps Backup Pipeline

Overview

A fully declarative, Kubernetes-native approach where backup and restore operations are managed through Custom Resource Definitions (CRDs) and reconciled by a GitOps controller such as ArgoCD or Flux. All configuration lives in a Git repository, providing version history, peer review, and automated rollout for every change.

When to Use

- Your team already operates a Kubernetes platform with GitOps tooling
- **RPO < 1 hour** is acceptable
- **RTO < 2 hours** is acceptable
- You want all backup configuration versioned, reviewed, and auditable in Git
- You need to manage backup across multiple environments (dev, staging, prod) consistently

Architecture Diagram

```
┌─────────────────┐
┌────────────────────────────────────────────────────────┐
│  Git Repo       │     │  Kubernetes Cluster
│
│                 │     │
│
│  envs/          │     │  ┌───────────┐
┌────────────────────────────────┐  │
│  └─ prod/       │───►│  │  ArgoCD   │──►│  kafka-backup
Operator          │  │
│     ├─ app.yaml     │  └───────────┘    │
│  │
│     ├─ backup.yaml  │                  │
┌──────────────────────────┐  │  │
│     ├─ monitor.yaml │                  │  │  KafkaBackup CR
│  │  │
│     └─ restore.yaml │                  │  │  (reconciles
backup     │  │  │
│               │     │                  │  │   pods
automatically)   │  │  │
└──────────────┘     │                  │
└───────────┬──────────────┘  │  │
                      │
└──────────────┼─────────────────┘  │
                      │                                  │
│
                      │
┌──────▼───────┐          │
                      │                          │  kafka-
backup│            │
                      │                          │  pods
│            │
                      │
└──────┬───────┘          │
                      │                                  │
│
                      │  ┌───────────────────┐
┌──────▼───────┐          │
                      │  │  Prometheus +     │    │  S3 Bucket
│            │
                      │  │  Grafana          │    │
│            │
                      │  │  (ServiceMonitor) │
└──────────────┘          │
                      │  └───────────────────┘
│

└──────────────────────────────────────────────────────┘
```

Components

Component	Purpose
Git repository	Single source of truth for all backup configuration
ArgoCD / Flux	GitOps controller, reconciles desired state

Component	Purpose
kafka-backup Operator	Watches KafkaBackup/KafkaRestore CRDs, manages pods
KafkaBackup CRD	Declarative backup specification
KafkaRestore CRD	Declarative restore specification
Prometheus ServiceMonitor	Auto-discovered metrics scraping
S3 bucket	Backup storage

Configuration

Git Repository Structure

```
environments/
└── prod/
    ├── kustomization.yaml
    ├── argocd-application.yaml
    ├── kafka-backup-crd.yaml
    ├── kafka-restore-crd.yaml
    └── service-monitor.yaml
```

ArgoCD Application

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: kafka-backup-prod
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/my-org/kafka-backup-config.git
    targetRevision: main
    path: environments/prod
  destination:
    server: https://kubernetes.default.svc
    namespace: kafka-backup
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

KafkaBackup Custom Resource

```yaml
apiVersion: kafka-backup.osodevops.io/v1alpha1
kind: KafkaBackup
metadata:
  name: prod-backup
  namespace: kafka-backup
spec:
  source:
    bootstrapServers:
      - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
      - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
      - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
    topicSelector:
      include:
        - ".*"
  storage:
    type: s3
    s3:
      bucket: my-org-kafka-backup
      region: us-east-1
      prefix: prod/
  backup:
    compression: zstd
    segmentMaxBytes: 134217728
    continuous: true
    checkpointIntervalSecs: 60
  resources:
    requests:
      cpu: 500m
      memory: 512Mi
    limits:
      cpu: "2"
      memory: 2Gi
```

KafkaRestore Custom Resource

```yaml
apiVersion: kafka-backup.osodevops.io/v1alpha1
kind: KafkaRestore
metadata:
  name: prod-restore
  namespace: kafka-backup
spec:
  source:
    type: s3
    s3:
      bucket: my-org-kafka-backup
      region: us-east-1
      prefix: prod/
  target:
    bootstrapServers:
      - kafka-0.kafka-headless.kafka.svc.cluster.local:9092
      - kafka-1.kafka-headless.kafka.svc.cluster.local:9092
      - kafka-2.kafka-headless.kafka.svc.cluster.local:9092
    topicSelector:
      include:
        - ".*"
  restore:
    fromLatest: true
  # Set to 'paused: true' until a restore is needed
  paused: true
```

Prometheus ServiceMonitor

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: kafka-backup
  namespace: kafka-backup
  labels:
    release: prometheus
spec:
  selector:
    matchLabels:
      app: kafka-backup
  endpoints:
    - port: metrics
      interval: 30s
      path: /metrics
```

> **INFO**
>
> With GitOps, every configuration change goes through a pull request. This gives you a full audit trail, peer review, and the ability to roll back any change by reverting a commit.

Cost Estimate

Item	Monthly Cost
Base backup infrastructure (Architecture 1)	~$105
GitOps tooling (ArgoCD/Flux — typically already deployed)	~$0
Total	**~$105**

Limitations

- Requires Kubernetes and GitOps expertise on the team
- Operator learning curve — custom resources add an abstraction layer
- CRD schema changes require careful upgrade planning
- ArgoCD/Flux must be operational for configuration changes to propagate (backup continues running if GitOps is temporarily down)

Choosing an Architecture

> **TIP**

> Use the comparison table at the top of this page as a starting point. Then consider these questions:
>
> 1. **What is your RPO/RTO budget?** If < 15 min RPO is required, start with Architecture 2 (Cross-Region DR).
> 2. **Do you need multi-cloud protection?** Architecture 3 is the only option that survives a full cloud provider outage.
> 3. **Are you in a regulated industry?** Architecture 4 (Air-Gapped) provides the immutability guarantees auditors look for.
> 4. **Is your team already running GitOps?** Architecture 5 adds minimal overhead and maximum auditability.
> 5. **Just getting started?** Architecture 1 is the fastest path to a working, production-grade backup.

All architectures can be combined. For example, you can run Architecture 5 (GitOps) as your deployment model while using Architecture 2 (Cross-Region) as your storage topology and Architecture 4 (Air-Gapped) as an additional compliance layer.

Self-Assessment Checklist

Use this checklist to evaluate the maturity of your Kafka backup architecture across all six pillars. Score each item honestly — the goal is to identify improvement areas, not to achieve a perfect score on day one.

How to Score

Rate each item on a 0–3 scale:

Score	Level	Description
0	Not implemented	No action taken
1	Basic	Partially implemented, manual processes
2	Advanced	Fully implemented, mostly automated
3	Expert	Fully automated, continuously improved, measured

Scoring Thresholds

Total Score	Maturity	Action
0–25	Critical gaps	Address immediately — your backup infrastructure has significant risk
26–50	Developing	Create a prioritised improvement plan targeting the lowest-scoring pillars
51–70	Mature	Focus on optimisation and automation of remaining manual processes
71–87	Well-Architected	Maintain through continuous improvement and regular reassessment

REASSESS REGULARLY

Re-run this assessment quarterly, or after significant changes to your Kafka environment (new topics, increased throughput, new compliance requirements). Track your score over time to measure improvement.

Operational Excellence

#	Check	Score (0–3)
1	Backup operations have a designated owner with clear escalation paths	
2	Backup schedules are fully automated (no manual runs required)	
3	Monitoring and alerting covers all key backup metrics (lag, throughput, errors, checkpoint age)	
4	DR runbooks exist with exact `kafka-backup` CLI commands and have been tested	
5	All backup configuration is version-controlled and deployed via GitOps or CI/CD	

Pillar subtotal: ___ / 15

Security

#	Check	Score (0–3)
6	Least-privilege IAM policies are enforced for backup and restore processes separately	
7	All backup data is encrypted at rest (SSE or client-side encryption)	
8	All connections are encrypted in transit (TLS 1.2+ for Kafka, HTTPS for storage)	
9	No hardcoded credentials — all secrets managed via a secrets manager or environment variables	
10	Audit logging is enabled for all backup and restore operations	

Pillar subtotal: ___ / 15

Reliability

#	Check	Score (0–3)
11	Backup integrity is validated automatically after every run (`kafka-backup validate --deep`)	
12	RPO and RTO targets are defined per topic tier and documented	
13	Consumer offset recovery has been tested and is part of the restore procedure	
14	DR drills are conducted at least quarterly with documented results	
15	Backup storage is geographically separated from the primary Kafka cluster	

Pillar subtotal: ___ / 15

Performance Efficiency

#	Check	Score (0–3)
16	Backup throughput has been benchmarked and meets RPO requirements	
17	Compression algorithm and level have been optimised for your data formats	
18	kafka-backup is co-located with Kafka brokers (same AZ/region)	
19	Compute resources are right-sized based on measured utilisation	
20	Restore performance has been benchmarked and meets RTO requirements	

Pillar subtotal: ___ / 15

Cost Optimisation

#	Check	Score (0–3)
21	Storage lifecycle policies are active (tiering from Standard → IA → Glacier)	
22	Retention policies are defined per topic tier and enforced automatically	
23	Backup costs are tracked, tagged, and attributed to teams or projects	
24	VPC endpoints are used for storage access (no public internet transfer costs)	

#	Check	Score (0–3)
25	Compute is right-sized and scales down when not actively backing up	

Pillar subtotal: ___ / 15

Sustainability

#	Check	Score (0–3)
26	Compute resources scale down or terminate when not in use	
27	Topic filtering excludes unnecessary topics from backup	
28	Cold storage tiers are used for long-term retention	
29	Compression is enabled to reduce storage and network resource consumption	

Pillar subtotal: ___ / 12

Total Score

Pillar	Score
Operational Excellence	___ / 15
Security	___ / 15
Reliability	___ / 15
Performance Efficiency	___ / 15

Pillar	Score
Cost Optimisation	___ / 15
Sustainability	___ / 12
Total	**___ / 87**

Next Steps

Based on your score, prioritise improvements in the lowest-scoring pillars:

1. **Identify the pillar with the lowest score** — this is your highest-risk area
2. **Review the corresponding pillar page** for detailed best practices and implementation guidance
3. **Start with the highest-impact, lowest-effort items** — typically monitoring (OE-03), encryption at rest (SEC-02), and backup validation (REL-01)
4. **Set a target score** for your next quarterly assessment
5. **Track progress** over time and celebrate improvements

> **NEED HELP?**
>
> If your assessment reveals critical gaps, the Reference Architectures provide proven deployment patterns you can adopt. For Enterprise features like encryption, RBAC, and audit logging, contact OSO for a consultation.

Frequently Asked Questions

Common questions about OSO Kafka Backup, organized by category.

General

What is OSO Kafka Backup?

OSO Kafka Backup is an open-source, high-performance backup and restore tool for Apache Kafka, written in Rust. It provides point-in-time recovery (PITR) for Kafka topics and consumer group offsets, supports multi-cloud storage backends (S3, Azure Blob Storage, GCS), and ships as a single static binary. The project is licensed under the MIT License.

How does OSO Kafka Backup differ from MirrorMaker 2?

MirrorMaker 2 is a **replication** tool designed to mirror data between live Kafka clusters in real time. OSO Kafka Backup is a **backup and recovery** tool designed to create durable, versioned copies of your Kafka data in external object storage. Key differences:

Capability	MirrorMaker 2	OSO Kafka Backup
Primary purpose	Cross-cluster replication	Backup and restore
Storage target	Another Kafka cluster	Object storage (S3, GCS, Azure Blob)
Point-in-time recovery	No	Yes
Offset recovery	Limited	Full consumer group offset restore
Independent of Kafka	No (requires target cluster)	Yes (stores to object storage)

Use MirrorMaker 2 for active-active or active-passive cluster topologies. Use OSO Kafka Backup for disaster recovery, compliance archival, and point-in-time restore scenarios.

How does OSO Kafka Backup compare to Confluent Replicator?

Unlike Confluent Replicator, OSO Kafka Backup:

- **Stores backups in external object storage** rather than requiring a destination Kafka cluster
- **Supports point-in-time recovery (PITR)** to restore data to any arbitrary timestamp
- **Recovers consumer group offsets** so applications resume from the correct position after restore
- **Ships as a single binary** with no dependencies on the Confluent Platform or Connect framework
- **Is open source** under the MIT License, with no per-broker licensing costs

Is OSO Kafka Backup production-ready?

Yes. OSO Kafka Backup is built in Rust for memory safety and high performance. In production environments it achieves throughput exceeding 100 MB/s and operates with less than 500 MB of memory. It includes built-in checkpointing for crash resilience, Prometheus metrics for observability, and has been validated across enterprise workloads.

What Kafka versions are supported?

OSO Kafka Backup supports any Kafka cluster that implements the Kafka protocol version 0.10 or later. This includes clusters running in both ZooKeeper mode and KRaft mode. The tool uses the standard Kafka consumer and producer APIs, so it is compatible with all Kafka distributions that adhere to the protocol.

What managed Kafka services are supported?

OSO Kafka Backup works with all major managed Kafka services, including:

- **Amazon MSK** (both provisioned and serverless)
- **Confluent Cloud**
- **Aiven for Apache Kafka**

- **Redpanda** (Kafka API-compatible)
- **Azure Event Hubs for Kafka** (Kafka protocol endpoint)

Any service that exposes a standard Kafka protocol endpoint is supported.

What is the difference between the OSS and Enterprise editions?

OSS edition includes:

- Full backup and restore functionality
- Point-in-time recovery (PITR)
- Compression (zstd, gzip, snappy, lz4)
- Prometheus metrics and monitoring
- Consumer group offset backup and restore

Enterprise edition adds:

- Client-side AES-256 encryption
- Role-based access control (RBAC)
- Audit logging
- GDPR compliance tools (data masking, right to be forgotten, field-level redaction)
- Schema Registry backup and restore
- Priority support with SLAs

Backup Operations

How do I schedule automated backups?

There are two primary approaches:

Kubernetes CronJob -- Run `kafka-backup backup` with `stop_at_current_offsets: true` on a schedule:

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: kafka-backup-scheduled
```

```
spec:
  schedule: "0 */6 * * *"  # Every 6 hours
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: kafka-backup
              image: ghcr.io/osodevops/kafka-backup:latest
              args: ["backup", "--config", "/etc/kafka-
backup/config.yaml"]
          restartPolicy: OnFailure
```

Kafka Backup Operator -- Use the `KafkaBackupSchedule` CRD to define schedules declaratively:

```
apiVersion: kafkabackup.oso.sh/v1alpha1
kind: KafkaBackupSchedule
metadata:
  name: daily-backup
spec:
  schedule: "0 2 * * *"
  backupSpec:
    configRef:
      name: backup-config
```

Can I back up specific topics?

Yes. Use `topics.include` and `topics.exclude` with wildcard patterns:

```
topics:
  include:
    - "orders.*"
    - "payments.*"
    - "inventory.updates"
  exclude:
    - "*.test"
    - "*.staging"
```

Patterns use glob-style matching. If `include` is not specified, all topics are backed up. The `exclude` list takes precedence over `include`.

How does incremental backup work?

OSO Kafka Backup uses checkpoint-based incremental backups. A local SQLite database tracks the last committed offset for each topic-partition. On each backup run, the tool resumes consuming from the last checkpointed offset, so only new messages are read and stored. This makes subsequent backup runs significantly faster and reduces storage costs.

What happens if a backup fails mid-run?

The checkpoint mechanism ensures crash resilience. If a backup run fails or is interrupted, the checkpoint database retains the last successfully committed offset for each partition. The next backup run automatically resumes from that point. No data is lost and no duplicate data is written to storage.

How are consumer group offsets backed up?

Consumer group offsets are stored as `x-original-offset` headers within the backed-up messages. During restore, a three-phase process recovers them:

1. **Restore messages** to the target cluster
2. **Plan offset reset** using `kafka-backup offset-reset plan` to compute the mapping between original and new offsets
3. **Execute offset reset** using `kafka-backup offset-reset execute` to commit the mapped offsets to the target cluster's consumer groups

Can I run multiple backup instances simultaneously?

Yes. You can run multiple instances of OSO Kafka Backup concurrently, provided each instance is configured to back up a different set of topics. Use non-overlapping `topics.include` patterns to partition the workload. Do not configure multiple instances to back up the same topic-partition, as this will result in duplicate data in storage.

How do I verify a backup?

Use the built-in validation command:

```
kafka-backup validate --deep --config /path/to/config.yaml
```

The `--deep` flag performs a full integrity check, verifying that all segments are present, checksums are valid, and the manifest is consistent with the stored data.

What is the maximum supported message size?

The maximum message size is governed by the Kafka cluster's `max.message.bytes` configuration, which defaults to 1 MB. OSO Kafka Backup has been tested with messages up to 10 MB. If your cluster uses a non-default maximum, ensure the backup tool's consumer configuration matches (via `message.max.bytes` in the consumer properties).

Restore & Recovery

How do I restore to a specific point in time?

Use the `time_window_start` and `time_window_end` parameters in your restore configuration, specified in epoch milliseconds:

```
restore:
  time_window_start: 1742817600000  # 2026-03-24 12:00:00 UTC
  time_window_end:   1742846400000  # 2026-03-24 20:00:00 UTC
  source:
    storage:
      type: s3
      bucket: my-kafka-backups
  target:
    bootstrap_servers: "target-kafka:9092"
```

Only messages with timestamps within the specified window will be restored.

How do I convert a date to epoch milliseconds?

Bash:

```
date -d "2026-03-24 12:00:00 UTC" +%s%3N
# Output: 1742817600000
```

Python:

```python
from datetime import datetime
int(datetime(2026, 3, 24, 12).timestamp() * 1000)
# Output: 1742817600000
```

macOS (BSD date):

```bash
date -j -u -f "%Y-%m-%d %H:%M:%S" "2026-03-24 12:00:00" +%s000
```

Can I restore to a different cluster?

Yes. Specify the target cluster's `bootstrap_servers` in your restore configuration. The source and target clusters are completely independent. This is a core use case for disaster recovery -- restoring data to a standby cluster in a different region or cloud provider.

Can I restore to a different topic name?

Yes. Use the `topic_mapping` configuration to remap topic names during restore:

```yaml
restore:
  topic_mapping:
    "orders.production": "orders.restored"
    "payments.production": "payments.restored"
```

How do I recover consumer offsets after a restore?

Use the two-step offset reset workflow:

```bash
# Step 1: Generate the offset mapping plan
kafka-backup offset-reset plan \
  --config /path/to/config.yaml \
  --output offset-plan.json

# Step 2: Review and execute the plan
kafka-backup offset-reset execute \
  --plan offset-plan.json \
  --target-bootstrap-servers target-kafka:9092
```

The plan maps original offsets to the corresponding offsets in the restored topic, accounting for any gaps or reordering.

How long does a restore take?

Restore duration depends on the data volume, storage backend read throughput, network bandwidth, and target cluster write capacity. Under optimal conditions, OSO Kafka Backup achieves approximately 100 MB/s restore throughput. For example, restoring 1 TB of data takes roughly 2.5 to 3 hours.

Can I restore a subset of partitions?

Yes. Use the `source_partitions` configuration to specify which partitions to restore:

```yaml
restore:
  source_partitions: [0, 1, 2, 5]
```

Only the specified partitions will be restored from the backup.

What happens if the target topic already has data?

OSO Kafka Backup **appends** data to the target topic; it does not overwrite or truncate existing data. If you need a clean restore, create a new topic (or use `topic_mapping` to restore to a different topic name) to avoid mixing existing and restored data.

Storage

What storage backends are supported?

OSO Kafka Backup supports:

- **Amazon S3**
- **Azure Blob Storage**
- **Google Cloud Storage (GCS)**
- **S3-compatible storage** (MinIO, Ceph, Wasabi, DigitalOcean Spaces)
- **Local filesystem** (for testing and development)

How much storage will my backups consume?

Estimate storage as:

```
storage_required = raw_data_size / compression_ratio
```

Compression ratios vary by data type:

Data Type	Compression (zstd)	Example
JSON	5:1 to 7:1	1 TB raw ≈ 200-300 GB compressed
Avro	2:1 to 3:1	1 TB raw ≈ 350-500 GB compressed
Protobuf	2:1 to 3:1	1 TB raw ≈ 350-500 GB compressed
Already compressed	~1:1	No significant reduction

What is the backup storage format?

Backups are organized as follows:

```
backup-root/
├── manifest.json
├── state/
│   └── offsets.db
└── topics/
    └── {topic-name}/
        └── partition={id}/
            ├── segment-000000000000.zst
            ├── segment-000000001000.zst
            └── ...
```

- `manifest.json` -- Metadata about the backup (topics, partitions, offsets, timestamps)
- `state/offsets.db` -- SQLite checkpoint database tracking committed offsets
- `topics/{topic}/partition={id}/segment-NNNN.zst` -- Compressed data segments

Can I access backup data without restoring?

Yes. Use the `describe` command to inspect backup metadata:

```
kafka-backup describe --config /path/to/config.yaml
```

You can also directly access objects in S3 (or other storage) using standard tools such as the AWS CLI, `gsutil`, or `az storage blob`. Segment files are compressed with the configured algorithm (e.g., zstd) and contain Kafka records in a binary format.

Can I migrate backups between storage backends?

Yes. Since backups are stored as standard objects, you can copy them between backends using tools like `aws s3 sync`, `gsutil rsync`, `azcopy`, or `rclone`. After copying, update your restore configuration to point to the new storage location.

Does OSO Kafka Backup work with S3-compatible storage?

Yes. Configure the `endpoint` URL to point to your S3-compatible service:

```yaml
storage:
  type: s3
  bucket: my-backups
  region: us-east-1
  endpoint: "https://minio.internal:9000"
  force_path_style: true
```

This works with MinIO, Ceph Object Gateway, Wasabi, DigitalOcean Spaces, and other S3-compatible services.

Performance

What throughput can I expect?

Under optimal conditions, OSO Kafka Backup achieves 100+ MB/s for both backup and restore operations. Actual throughput depends on:

- Network bandwidth between Kafka, the backup tool, and storage
- Storage backend write/read latency
- Compression algorithm and level
- Message size (larger messages achieve higher throughput)
- Number of partitions being processed concurrently

How much memory does OSO Kafka Backup use?

Typical memory usage is under 500 MB when processing 4 partitions concurrently. Memory consumption scales with the number of concurrent partitions and the configured segment size. For high-concurrency workloads, monitor RSS via the `process_resident_memory_bytes` Prometheus metric and adjust `segment_max_bytes` or concurrency settings accordingly.

How do I tune for maximum throughput?

Refer to **PE-01: Throughput Optimisation** in the Performance Efficiency pillar. Key tuning parameters:

- **Segment size:** Increase `segment_max_bytes` to reduce the number of storage write operations
- **Fetch size:** Increase `fetch.max.bytes` and `max.partition.fetch.bytes` in the consumer config
- **Compression level:** Use a lower zstd compression level (e.g., 1-3) for faster compression at the cost of slightly larger files
- **Co-location:** Deploy the backup tool in the same region and availability zone as the Kafka cluster and storage backend

What impact does backup have on the Kafka cluster?

Minimal. OSO Kafka Backup operates as a standard Kafka consumer. It does not require any broker restarts, plugins, or configuration changes. The impact is equivalent to adding another consumer to the cluster. For latency-sensitive workloads, consider configuring a dedicated consumer group and using rack-aware replica fetching.

How do I benchmark performance?

The kafka-backup-demos repository includes a benchmark suite that generates synthetic workloads and measures backup/restore throughput under various configurations. Use it to establish baselines for your environment before deploying to production.

Security & Compliance

Is backup data encrypted?

Server-side encryption: All major cloud storage providers offer server-side encryption (SSE-S3, SSE-KMS, Azure Storage Service Encryption, GCS default encryption). Enable this on your storage bucket for encryption at rest.

Client-side encryption (Enterprise): The Enterprise edition supports client-side AES-256 encryption, where data is encrypted before it leaves the backup tool. This ensures data is encrypted in transit to storage and at rest, regardless of the storage provider's encryption settings.

How do I configure TLS or mTLS?

Set the security protocol and certificate paths in your configuration:

```yaml
kafka:
  bootstrap_servers: "kafka:9093"
  security_protocol: "SSL"  # or "SASL_SSL" for SASL + TLS
  ssl_ca_location: "/certs/ca.pem"
  ssl_certificate_location: "/certs/client.pem"
  ssl_key_location: "/certs/client-key.pem"
```

For mTLS, provide both the client certificate and key. The CA certificate is used to verify the broker's identity.

What SASL authentication mechanisms are supported?

OSO Kafka Backup supports the following SASL mechanisms:

- **PLAIN** -- Username and password (use with TLS)
- **SCRAM-SHA-256** -- Salted Challenge Response Authentication
- **SCRAM-SHA-512** -- Salted Challenge Response Authentication (stronger hash)

```yaml
kafka:
  security_protocol: "SASL_SSL"
  sasl_mechanism: "SCRAM-SHA-512"
  sasl_username: "backup-user"
  sasl_password: "${KAFKA_SASL_PASSWORD}"
```

How does OSO Kafka Backup support GDPR compliance?

The **Enterprise edition** provides GDPR compliance tools:

- **Data masking:** Redact or mask personally identifiable information (PII) during backup
- **Right to be forgotten:** Delete specific records from backups by key
- **Field-level redaction:** Selectively redact fields within messages while preserving the rest of the record

These features enable compliance with data protection regulations without sacrificing backup completeness.

How do I restrict who can perform restore operations?

Multiple layers of access control are available:

- **Enterprise RBAC:** Define roles (backup-operator, restore-operator, admin) with fine-grained permissions
- **IAM policies:** Restrict access to storage buckets using AWS IAM, Azure RBAC, or GCP IAM
- **Kubernetes RBAC:** Limit which service accounts can create `KafkaRestore` custom resources

---

Kubernetes & Deployment

How do I deploy on Kubernetes?

Install the Kafka Backup Operator via Helm:

```
helm repo add oso https://charts.oso.sh
helm repo update
helm install kafka-backup-operator oso/kafka-backup-operator \
  --namespace kafka-backup \
  --create-namespace
```

Then create backup and restore resources using CRDs:

```
apiVersion: kafkabackup.oso.sh/v1alpha1
kind: KafkaBackup
metadata:
  name: production-backup
spec:
  configRef:
    name: backup-config
```

What Kubernetes versions are supported?

OSO Kafka Backup Operator requires Kubernetes 1.24 or later. It is tested against the latest three minor versions of Kubernetes.

Can I use ArgoCD or Flux for GitOps deployments?

Yes. Store your `KafkaBackup`, `KafkaRestore`, and `KafkaBackupSchedule` CRD manifests in a Git repository. Configure an ArgoCD `Application` or Flux `Kustomization` pointing to the manifests directory:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: kafka-backup
spec:
  source:
    repoURL: https://github.com/myorg/k8s-manifests
    path: kafka-backup/
    targetRevision: main
  destination:
    server: https://kubernetes.default.svc
    namespace: kafka-backup
```

Can I deploy outside of Kubernetes?

Yes. OSO Kafka Backup ships as a standalone static binary that runs on bare metal, virtual machines, and Docker containers. No Kubernetes or container orchestration is required:

```
# Download the binary
curl -LO https://github.com/osodevops/kafka-
backup/releases/latest/download/kafka-backup-linux-amd64

# Run directly
./kafka-backup-linux-amd64 backup --config /etc/kafka-
backup/config.yaml
```

How do I monitor OSO Kafka Backup in Kubernetes?

The operator exposes Prometheus metrics on port 8080. Create a `ServiceMonitor` to scrape them:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: kafka-backup-metrics
spec:
  selector:
    matchLabels:
      app: kafka-backup
  endpoints:
    - port: metrics
      interval: 15s
```

Pair this with the provided Grafana dashboards from the kafka-backup-demos repository for comprehensive visibility.

Enterprise

What features are included in the Enterprise edition?

The Enterprise edition extends the OSS version with:

- **AES-256 client-side encryption** for backup data
- **Role-based access control (RBAC)** for backup and restore operations
- **Audit logging** for all operations with tamper-proof log storage
- **GDPR compliance tools** including data masking, right to be forgotten, and field-level redaction
- **Schema Registry backup and restore** for Avro, Protobuf, and JSON Schema
- **Priority support** with defined SLAs

How do I get an Enterprise licence?

Contact the OSO sales team at oso.sh to discuss your requirements and obtain a licence key.

Is there a trial available?

Yes. A 30-day evaluation licence is available that provides full access to all Enterprise features. Contact the sales team to request a trial.

What support is included with Enterprise?

Enterprise support includes:

- **Critical issues (P1):** 24/7 response with a 1-hour initial response time
- **Standard issues (P2-P4):** Business hours support with response times based on severity
- **Dedicated Slack channel** for direct communication with the engineering team
- **Quarterly architecture reviews** to ensure your deployment follows best practices

---

Troubleshooting

How do I enable debug logging?

Use the `-v` flag for debug-level logging or `-vv` for trace-level:

```
# Debug logging
kafka-backup -v backup --config /path/to/config.yaml
```

```
# Trace logging (very verbose)
kafka-backup -vv backup --config /path/to/config.yaml
```

Alternatively, set the `RUST_LOG` environment variable:

```
RUST_LOG=debug kafka-backup backup --config
/path/to/config.yaml
```

My backup is running slowly. How do I diagnose this?

Check the following, in order:

1. **Network latency:** Measure latency between the backup tool and both the Kafka cluster and storage backend
2. **Storage write latency:** Monitor the `kafka_backup_storage_write_duration_seconds` Prometheus metric
3. **Compression overhead:** Try a faster compression level or algorithm (e.g., lz4 instead of zstd)
4. **Resource utilisation:** Check CPU and memory usage on the host running the backup
5. **Consumer lag:** Monitor `kafka_backup_consumer_lag` to see if the tool is keeping up with producers

I am getting a connection error. What should I check?

Verify the following:

1. **Bootstrap servers:** Ensure the `bootstrap_servers` address is correct and resolvable
2. **TLS certificates:** Verify certificates are valid, not expired, and the CA chain is complete
3. **Network connectivity:** Confirm the backup tool can reach the Kafka brokers on the configured port (e.g., `telnet kafka-broker 9093`)
4. **Firewall rules:** Check that security groups, NACLs, or firewall rules allow traffic on the Kafka port
5. **Kafka ACLs:** Ensure the backup user has `READ` and `DESCRIBE` permissions on the target topics and consumer group

My restore is failing. How do I troubleshoot?

Follow these steps:

1. **Validate the backup** first: `kafka-backup validate --deep --config /path/to/config.yaml`
2. **Check target connectivity:** Verify the restore tool can reach the target Kafka cluster
3. **Verify IAM/storage permissions:** Ensure the restore process has read access to the backup storage location
4. **Check disk space:** Ensure sufficient local disk space for temporary decompression buffers
5. **Review error logs:** Enable debug logging (`-v`) and check for specific error messages

How do I report a bug or get community support?

- **Bug reports:** Open an issue on GitHub at github.com/osodevops/kafka-backup/issues
- **Community support:** Start a discussion at GitHub Discussions
- **Enterprise support:** Use your dedicated Slack channel or contact the support team directly

Glossary

Definitions of key terms used throughout the OSO Kafka Backup Well-Architected Framework documentation.

Term	Definition
ACL	Access Control List. A set of rules in Apache Kafka that define which users or service accounts are permitted to perform specific operations (read, write, describe) on topics, consumer groups, and other resources.
Air-Gapped Backup	A backup stored in a location that is physically or logically isolated from the primary environment, preventing compromise of both primary data and backups in a single incident.
Audit Log	A chronological record of operations performed by OSO Kafka Backup, including who initiated each action, what was affected, and the outcome. Available in the Enterprise edition.
Backup	A durable copy of Kafka topic data and metadata stored in external object storage, created by OSO Kafka Backup for disaster recovery and compliance purposes.
Backup ID	A unique identifier assigned to each backup run, used to reference and manage specific backup snapshots.
Backup Window	The time period during which a backup operation runs. Shorter backup windows reduce the risk of data loss but may require more resources.
Bootstrap Servers	A comma-separated list of Kafka broker addresses (host:port) used by clients to establish an initial connection to the Kafka cluster and discover the full cluster topology.
Broker	A Kafka server that stores topic partitions and serves client requests. A Kafka cluster consists of one or more brokers.
Chaos Engineering	The discipline of experimenting on a system to build confidence in its ability to withstand turbulent conditions in production, such as simulating broker failures or network partitions.

Term	Definition
Checkpoint	A record of the last successfully committed offset for each topic-partition, stored in a local SQLite database. Checkpoints enable incremental backups and crash-resilient resume.
Consumer Group	A named group of Kafka consumers that coordinate to consume messages from one or more topics, with each partition assigned to exactly one consumer in the group.
CRD	Custom Resource Definition. A Kubernetes extension mechanism used by the OSO Kafka Backup Operator to define custom resources such as `KafkaBackup`, `KafkaRestore`, and `KafkaBackupSchedule`.
Customer-Managed Key (CMK)	An encryption key owned and managed by the customer (rather than the cloud provider) used for encrypting backup data at rest, providing full control over key lifecycle and access.
Data Masking	The process of obfuscating or redacting sensitive fields within Kafka messages during backup, ensuring that personally identifiable information (PII) is not stored in plain text. Available in the Enterprise edition.
DR Drill	Disaster Recovery Drill. A planned exercise that tests the end-to-end restore process, validating that backups are viable and that the team can meet RTO and RPO targets.
Encryption at Rest	Protection of stored data using encryption algorithms (e.g., AES-256) so that data on disk or in object storage is unreadable without the decryption key.
Encryption in Transit	Protection of data as it moves between systems using TLS, ensuring that data exchanged between Kafka brokers, backup tools, and storage backends cannot be intercepted.
Full Backup	A backup that captures all messages in the configured topics from the earliest available offset through to the current offset. Contrast with incremental backup.

Term	Definition
GCS	Google Cloud Storage. An object storage service from Google Cloud Platform, supported as a backup storage backend by OSO Kafka Backup.
GitOps	An operational model where the desired state of infrastructure and applications is declared in Git repositories, with automated tooling (e.g., ArgoCD, Flux) reconciling the live state to match.
Grafana	An open-source observability platform used to visualise Prometheus metrics from OSO Kafka Backup through pre-built dashboards.
IAM	Identity and Access Management. Cloud provider services (AWS IAM, Azure RBAC, GCP IAM) that control which identities can access storage buckets, encryption keys, and other resources.
Incremental Backup	A backup that captures only messages produced since the last checkpoint, reducing backup duration and storage consumption compared to a full backup.
ISR (In-Sync Replicas)	The set of partition replicas that are fully caught up with the leader replica. A message is considered committed only when all ISR members have acknowledged it.
KRaft	Kafka Raft. The consensus protocol that replaces ZooKeeper for Kafka cluster metadata management, available from Kafka 3.3 and the default from Kafka 4.0.
Kubernetes Operator	A software extension to Kubernetes that uses CRDs and custom controllers to manage the lifecycle of OSO Kafka Backup resources, including scheduling, monitoring, and reconciliation.
Lifecycle Policy	A storage backend rule that automatically transitions or deletes objects based on age. Used to manage backup retention by moving older backups to cheaper storage tiers or expiring them.
Manifest	A JSON file (`manifest.json`) stored at the root of a backup that contains metadata about the backup, including topics, partitions, offset ranges, and timestamps.

Term	Definition
MinIO	An open-source, S3-compatible object storage system that can serve as a self-hosted backup storage backend for OSO Kafka Backup.
mTLS	Mutual TLS. A TLS configuration where both the client and server present certificates and verify each other's identity, providing stronger authentication than one-way TLS.
Object Storage	A storage architecture that manages data as objects (with metadata and a unique identifier) rather than as files in a hierarchy. Examples include S3, GCS, and Azure Blob Storage.
Offset	A sequential integer assigned to each message within a Kafka partition, uniquely identifying the message's position. Offsets are used to track consumer progress and enable point-in-time recovery.
Partition	A subdivision of a Kafka topic that provides parallelism. Each partition is an ordered, immutable sequence of messages, and each message within a partition has a unique offset.
Point-in-Time Recovery (PITR)	The ability to restore Kafka topic data to any arbitrary timestamp by filtering backed-up messages based on their timestamps.
Prometheus	An open-source monitoring and alerting toolkit used to collect and query metrics exposed by OSO Kafka Backup on its metrics endpoint (default port 8080).
RBAC	Role-Based Access Control. A security model that restricts operations based on the roles assigned to users or service accounts. Available in the Enterprise edition.
Recovery Point Objective (RPO)	The maximum acceptable amount of data loss measured in time. An RPO of 1 hour means that up to 1 hour of data may be lost in a disaster.
Recovery Time Objective (RTO)	The maximum acceptable time to restore service after a disaster. An RTO of 4 hours means the system must be operational within 4 hours of an incident.

Term	Definition
Replication Factor	The number of copies of each partition maintained across Kafka brokers. A replication factor of 3 means each partition has three replicas, providing fault tolerance.
Restore	The process of reading backed-up data from object storage and producing it to a target Kafka cluster, optionally filtered by time window, topic, or partition.
Runbook	A documented procedure for performing a specific operational task, such as restoring a Kafka topic from backup or responding to a backup failure alert.
S3	Amazon Simple Storage Service. An object storage service from AWS, and the most commonly used backup storage backend for OSO Kafka Backup.
SASL	Simple Authentication and Security Layer. A framework for authentication used by Kafka clients, supporting mechanisms such as PLAIN, SCRAM-SHA-256, and SCRAM-SHA-512.
Segment	A compressed file within a backup that contains a range of Kafka messages for a specific topic-partition. Segments are named by their starting offset (e.g., `segment-000000001000.zst`).
Server-Side Encryption (SSE)	Encryption performed by the storage provider (e.g., S3, GCS, Azure Blob) at rest, transparently encrypting and decrypting objects without changes to the client.
SLA	Service Level Agreement. A formal commitment defining the expected availability, performance, and support response times for a service.
SLI	Service Level Indicator. A quantitative metric used to measure system behaviour, such as backup success rate, restore latency, or storage write throughput.
SLO	Service Level Objective. A target value or range for an SLI, such as "99.9% backup success rate" or "restore completes within 4 hours."

Term	Definition
Storage Tier	A class of storage with specific cost and performance characteristics. For example, S3 Standard for active backups and S3 Glacier for long-term archival.
TLS	Transport Layer Security. A cryptographic protocol that provides encrypted communication between Kafka clients and brokers, and between the backup tool and storage backends.
Topic	A named category or feed in Apache Kafka to which messages are published. Topics are divided into partitions for scalability and parallelism.
ZooKeeper	A centralised coordination service historically used by Apache Kafka for cluster metadata management, being replaced by KRaft in modern Kafka deployments.

Further Reading

Curated resources for deepening your understanding of Kafka backup, disaster recovery, and operational best practices.

OSO Kafka Backup Documentation

- **Official Documentation:** kafkabackup.com -- Comprehensive guides covering installation, configuration, backup, restore, and monitoring.
- **GitHub Repository:** github.com/osodevops/kafka-backup -- Source code, issue tracker, and contribution guidelines.
- **Demo Repository:** github.com/osodevops/kafka-backup-demos -- End-to-end examples, benchmark suites, and reference architectures for testing OSO Kafka Backup in various environments.
- **Changelog:** CHANGELOG.md -- Release notes and version history for all OSO Kafka Backup releases.

Well-Architected Frameworks

These cloud provider frameworks informed the structure and principles of this Well-Architected guide:

- **AWS Well-Architected Framework:** docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html -- Amazon's framework covering operational excellence, security, reliability, performance efficiency, cost optimisation, and sustainability.
- **Azure Well-Architected Framework:** learn.microsoft.com/en-us/azure/well-architected/ -- Microsoft's guidance for designing and operating reliable, secure, and efficient workloads on Azure.
- **Google Cloud Architecture Framework:** cloud.google.com/architecture/framework -- Google's best practices for building well-architected systems on Google Cloud Platform.

Kafka Disaster Recovery

- **Apache Kafka Documentation:** kafka.apache.org/documentation/ -- The official Apache Kafka documentation, including broker configuration, client APIs, security, and operations guides.
- **KRaft Mode:** kafka.apache.org/documentation/#kraft -- Documentation for Kafka's ZooKeeper-free consensus mode, which simplifies cluster management and is the default from Kafka 4.0.

Industry Standards

The following standards and frameworks are relevant to organisations implementing Kafka backup and disaster recovery in regulated environments:

- **NIST Cybersecurity Framework** -- A voluntary framework from the U.S. National Institute of Standards and Technology providing guidelines for managing and reducing cybersecurity risk. Relevant to backup encryption, access controls, and incident response planning.
- **SOC 2 Type II** -- An auditing standard from the American Institute of CPAs (AICPA) that evaluates the effectiveness of an organisation's controls over security, availability, processing integrity, confidentiality, and privacy. Backup and restore procedures are a key component of SOC 2 compliance.
- **ISO 27001** -- An international standard for information security management systems (ISMS). It provides a systematic approach to managing sensitive data, including requirements for backup procedures, access control, and disaster recovery.
- **GDPR (General Data Protection Regulation)** -- The European Union's data protection regulation that governs how personal data is collected, processed, and stored. Relevant to backup data retention policies, right to erasure, and data masking requirements. The OSO Kafka Backup Enterprise edition provides specific GDPR compliance tools.
- **PCI DSS (Payment Card Industry Data Security Standard)** -- A set of security standards for organisations handling credit card data. Relevant to encryption of backup data, access control, audit logging, and secure storage of cardholder information.