1. **What problem serverless solves vs. K8s micro-services**
   Serverless removes the need to size, patch or pay for always-on infrastructure; you get per-request billing and zero-to-N autoscaling handled by the platform.
   - **Better fit**: nightly CSV-parser that runs when files land on S3—spiky, unpredictable, zero traffic most of the day.
   - **Worse fit**: real-time multiplayer game server that keeps persistent WebSockets and GPU state; cold-starts and frequent scale-ups would kill UX.
2. **Advantages of a service mesh over plain K8s networking**
   - Mutual TLS with automatic cert rotation and identity based on K8s SA.
   - Fine-grained L7 observability (latency, success-rate, retries) without code changes.
   - Policy enforcement (rate-limit, circuit-breaker, ACL) that works across clusters and languages.
   - Traffic shifting, mirroring, fault-injection for safe experimentation.
3. **Sidecar proxy (Envoy) role**
   Envoy runs alongside the app container and intercepts all ingress/egress packets via iptables. It terminates TLS, enforces authz policies, collects metrics, handles retries, and reports to the control plane. This lets the mesh add reliability & security transparently, leaving the application binary unchanged.
4. **Istio traffic-management features & examples**
   - **Weighted routing**: send 5 % of live traffic to v2 of a recommendation model to validate new embeddings.
   - **Circuit-breaking**: eject endpoints after 50 % 5xx errors, preventing cascade failures in the checkout service.
     (Also retries, timeouts, mirroring, fault-injection.)
5. **Knative Serving autoscaling**
   The Knative Pod Autoscaler (KPA) watches average concurrent requests per pod (default target = 100).
   - **Scale-up**: request volume > target concurrency.

- **Scale-down**: no requests for ~30 s (configurable), then pods are terminated to zero; the Activator component buffers new traffic until a pod is ready.

6. **Knative Eventing role**
   It supplies a CloudEvents-compatible broker/trigger model. Sources (GitHub, Kafka, S3, CronJob) inject events; triggers filter and route to Knative Services or external sinks. This decouples producers from consumers, enabling event-driven, serverless pipelines without writing queues or adapters.
7. **How Knative abstracts K8s primitives**
   Knative creates and owns standard objects under the hood:
   - `Configuration` → immutable `Deployment` (revision)
   - `Route` → `Service`/`Ingress` (Istio or K8s gateway)
   - KPA replaces `HorizontalPodAutoscaler` with request-based metrics

     Developers declare only a one-page `Service`; they never edit Deployments, Services, Ingress or HPAs, cutting boiler-plate and allowing focus on code.
8. **KServe InferenceService purpose**
   InferenceService is a CRD that packages model artifacts, server type (TFServing, TorchServe, Triton), resource/gpu needs, and optional transformer. KServe generates the Knative Service, configures Istio routing, adds Prometheus metrics, and exposes a single predictable HTTP/gRPC endpoint—one YAML deploys, versions, scales and monitors the model.
9. **End-to-end request path in a KServe production stack**
   1. Client HTTPS request → Istio Gateway (TLS, auth)
   2. Istio VirtualService routes to `kserve-model-predictor` service
   3. If scaled to zero, Knative Activator queues, KPA spins pod; K8s scheduler places it
   4. Model container pulls weights from object storage; sidecar Envoy signals ready
   5. Request hits model-server; response flows back through Envoy, Istio, Gateway

Latency bottlenecks: image/layer pull, large model load, cold-start concurrency, JSON serialization, Istio filter chain, insufficient GPU/cpu quota.

10. **Istio traffic routing for canary/A-B in Knative/KServe**

Apply a VirtualService with weight-based subsets:`http:`

```
- match:
  - headers {canary: "true"}
  route:
  - destination: kserve-predictor-canary 100 %
- route:
  - destination: kserve-predictor-stable 90 %
  - destination: kserve-predictor-canary 10 %
```

Add retries & circuit-breakers to protect the canary.

Pros: declarative, git-ops friendly, instant rollback, no custom CI scripts, metrics-driven.

Cons: extra proxy hops raise p99 latency; weights static until updated; header spoofing risk if user-controlled; needs Prometheus/Flagger for full automation vs. manual `kubectl rollout`.