**✅ Goal Recap**

* You get **one file per day**, size varies: 100 to 2 million lines.
* Need to **scale compute (CPU/memory)** based on **file size**.
* Processing must be **sequential**, so only **one pod per file**.
* You want to **auto-scale up/down** dynamically.

**✅ Short Answer First**

✅ Yes, you will create **3 separate Kubernetes deployments**:

* file-processor-small
* file-processor-medium
* file-processor-large

Each deployment has:

* Its own **resource config** (CPU/memory).
* Its own **Service Bus queue**.
* Its own **KEDA ScaledObject** watching that queue.

🔁 **Only one of them will be active at a time** — depending on which queue receives the message for today’s file.

**🎯 Step-by-Step: How KEDA + Queues + Pods Work**

**🔹 1. File Uploads to Blob Storage**

* A daily file is uploaded (e.g., 1.6M lines).
* Trigger a **preprocessing step** (Azure Function or Spring Boot service) to:
  + Count/estimate file size.
  + Based on size → route message to:
    - file-small-queue (if <50k lines)
    - file-medium-queue (50k–500k)
    - file-large-queue (>500k lines)

**🔹 2. Kubernetes Deployments (3 Types)**

Each pod image is identical (same Spring Boot + Camel code), but the deployments differ by:

|  |  |  |
| --- | --- | --- |
| **Deployment Name** | **CPU / Memory** | **Service Bus Queue Watched** |
| file-processor-small | 0.25 / 512MB | file-small-queue |
| file-processor-medium | 1 CPU / 2GB | file-medium-queue |
| file-processor-large | 2 CPU / 4GB | file-large-queue |

Each has a corresponding **ScaledObject**.

**🔹 3. KEDA + ScaledObject (per deployment)**

Each ScaledObject is configured like this (example: large tier):

apiVersion: keda.sh/v1alpha1

kind: ScaledObject

metadata:

name: file-processor-large-scaler

spec:

scaleTargetRef:

name: file-processor-large

pollingInterval: 30 # seconds

cooldownPeriod: 300 # scale down after 5 mins of idle

minReplicaCount: 0

maxReplicaCount: 1

triggers:

- type: azure-servicebus

metadata:

queueName: file-large-queue

messageCount: "1"

connectionFromEnv: SERVICEBUS\_CONN

Repeat the above for small and medium, adjusting names and queue.

**🔹 4. At Runtime: Which Pod is Active?**

* Only the **deployment corresponding to the queue with a message** will scale **from 0 → 1 pod**.
* **Other 2 deployments remain scaled to 0.**
* Once processing completes (file done, queue empty), KEDA will **scale back down to 0**.

**✅ Benefits of This Approach**

|  |  |
| --- | --- |
| **Feature** | **Advantage** |
| Efficient resource use | Big files → big pods, small files → tiny pods |
| Fully auto-scaled | No compute waste, 0 → 1 → 0 pod lifecycle |
| Easy isolation | One pod = one file = traceable |
| Single shared logic | Same Spring Boot + Camel image for all |

**❗ Important Notes**

* Only one pod active per day.
* No overlap/duplication — job routing ensures **only one queue has a message**.
* KEDA uses **event-driven polling**, no manual pod management needed.

