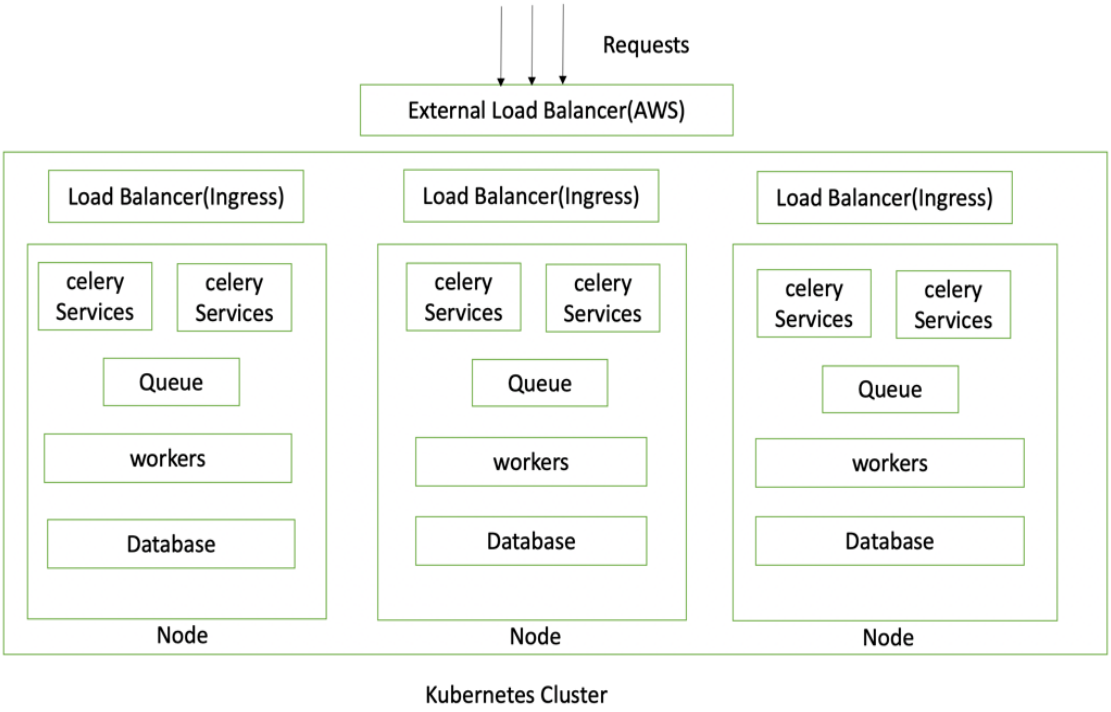
Xeneta – Task 2
Batch Processing



The above diagram describe a possible approach that can be taken to addresses a system that needs to receive and update batches of tens of thousands of new prices.

The above architecture consists of 3 main components:

- Request-Receiver end.
- Request traffic management.
- Service APIs.

In this approach we use an External Load Balancer from a Cloud Provider, Cloud Orchestration technology, Task queue, Services and Database.

The technology identified to achieve the scenario:

- External Load Balancer - AWS
- Ingress Load Balancer and Container Orchestration – Kubernetes
- Task Queue – Celery with RabbitMQ
- API – Python Flask
- Database - PostgreSQL

The architecture works as below:

- The External Load balancer on a Cloud Provider (like AWS) manages incoming bulk request and distributes application traffic to Ingress Load Balancers in Kubernetes Cluster.
- Ingress further manages the application traffic and exposes requests to Celery Services inside the cluster.
- Celery generates asynchronous tasks and allocates to a Queue (like RabbitMQ, Amazon SQS).
- Workers are allocated to consume tasks from Queue. Tasks are processed and executed.
- The Task consists of Python Flask API that reads and writes prices into PostgreSQL.
- Response statuses can be stored and processed for analysis.
- Kubernetes is used as it provides native capabilities to scale resources vertically and horizontally with improved stability, lower cost and self-healing.