CTW ML assignment

Solution：

1. Utilize "ray" to support the backend model inference computing which can be extended to multi-core /distributed computing
2. It is easy to set the num_replicas/nums_cpu/nums_gpu by ray serve in order to utilize the CPU/GPU resource.
3. Split the translate service through 'ray serve' , like this "serve run -h 0.0.0.0 -p 9527 translate_service:translator"

Delivery：

- **app/Dockerfile**: ml-assignment/app/Dockerfile

Containers can be launched by following commands.

- docker build . -t ensalty/ml-assignment -f ./ml-assignment/app/Dockerfile
- docker run -p 127.0.0.1:8265:8265 -p 127.0.0.1:9527:9527 --cpus=4 --shm-size=2.47gb -it ensalty/ml-assignment:latest /bin/bash

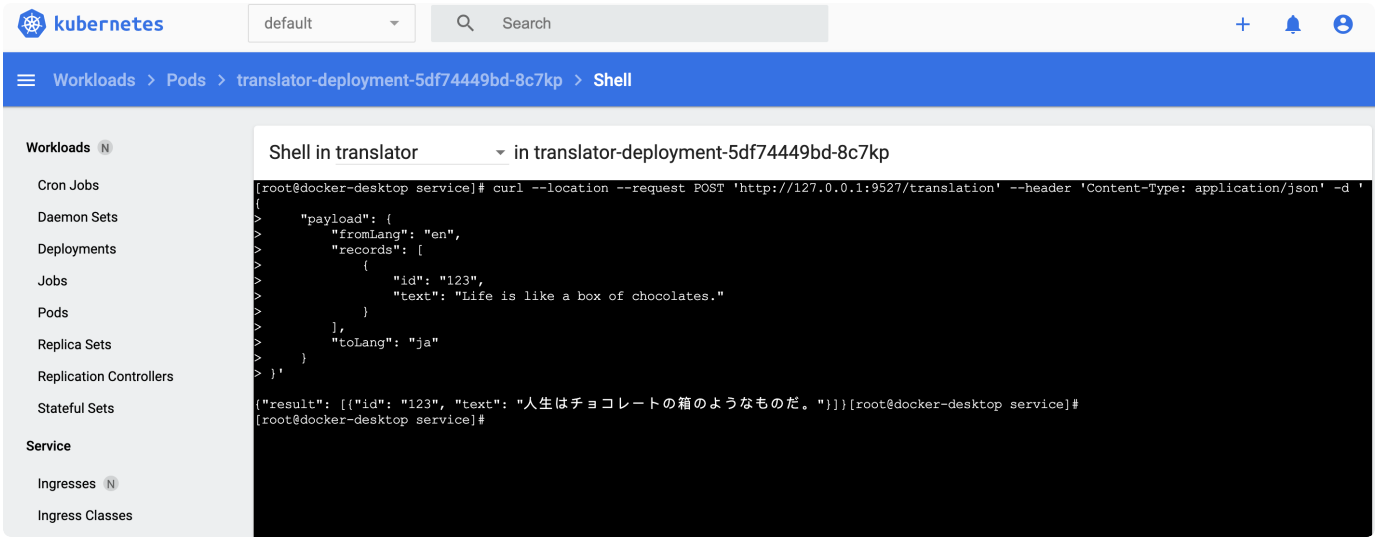
- **k8s/deployment.yaml**: ml-assignment/k8s/deployment.yaml

container can be deployed on k8s by following command

- kubectl create -f ml-assignment/k8s/deployment.yaml

- Other necessary code

Input/Output：



Scalable architecture

1. Backend service can be extended to multiple-cores or multiple hosts by configuring params.
2. Ray cluster can be extended by distributed nodes.
3. the number of CPUs/GPUs can be exactly assigned to the specific function.

```
@serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1, "num_gpus": 0}, route_prefix='/translation')
class Translator:
    """
    num_replicas determines how many copies of our deployment process run in Ray.
    Requests are load balanced across these replicas, allowing you to scale your deployments horizontally.
    """
```

Inference performance

1. LightSeq can be considered to optimize performance
2. self.accelerator = Accelerator(split_batches=True, dispatch_batches=False)

Efficient CPU/GPU utilization

CPU resource can be fully used by setting the nums_cpu.