AWS architecture for deployment of the trained TF model

Ivan Ćorić
4. October 2020.

Summary

In this document the Amazon Web Services (AWS) architecture and service required for the deployment of the trained Tensorflow (TF) model will be described, while keeping in mind data privacy and the protection of the intellectual property.

1 Architecture

The model will be served using containerized Tensorflow ModelServer on Kubernetes with Amazon EC2 Spot Instances.

Kubernetes clusters are managed using Amazon Elastic Kubernetes Service (EKS) and the Amazon EC2 Spot Instances are chosen to cost optimize the TF serving workloads. Spot instances are somewhat cheaper than On-Demand instances, but they can be interrupted by the EC2, so we will handle the interruptions using AWS Node Termination Handler, which detects the interruptions and automatically drains the nodes.

To ensure the high throughput we will use Application Load Balancer (ALB) created by Ingress Controller, which is deployed on the On-Demand Instance.

Data Protection in the AWS conforms to the AWS shared responsibility model, where Security and Compliance is a shared responsibility between AWS and the customer - us. AWS is responsible for the "Security *of* the Cloud" - infrastructure (hardware, software, networking and facilities) that runs all of the services offered in the AWS Cloud. On the other hand, we are responsible for the "Security *in* the Cloud" - client-side and server-side data, networking, identity and access management, OS, firewall...

The model will be stored in the Amazon S3 bucket, where we will use the default encryption to ensure the protection of the model. To ensure encryption in transit we will use HTTPS protocol for the communication.

Elasticity, the degree to which a system is able to automatically adapt to workload changes, is ensured using Horizontal Pod Autoscaler (HPA), Cluster Autoscaler (CA) and EC2 Auto Scaling group.

Horizontal Pod Autoscaler monitors the metrics (CPU / RAM) and launches Replicas (pods) if needed. Cluster Autoscaler, which also runs on On-Demand Instances, scales EC2 instances automatically according to pods running in the cluster, while Auto Scaling group provisions and maintains EC2 instance capacity.

Scheme of the proposed architecture and the relations between objects can be seen on the figure 1 and the following steps describe the process which ensures elasticity:

1. HTTPS requests flow in through the ALB and Ingress object.

2. HPA monitors the metrics (CPU / RAM) and once the threshold is breached a Replica (pod) is launched.

3. If there are sufficient cluster resources, the pod starts running, else it goes into the pending state.

4. If one or more pods are in pending state, the Cluster Autoscaler (CA) triggers a scale up request to Auto Scaling group. If HPA tries to schedule pods more than the current size of what the cluster can support, CA can add capacity to support that.

5. Auto Scaling group provision a new node and the application scales up

6. A scale down happens in the reverse fashion when requests start tapering down.

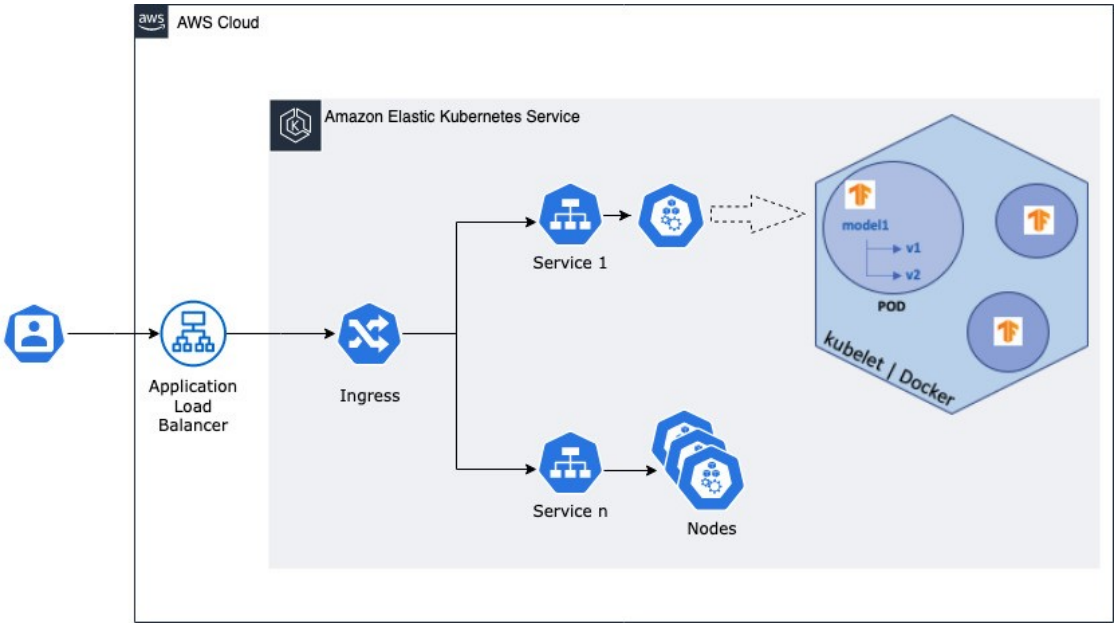

Fig. 1: Architecture graph. Downloaded from the [1]

References

[1] TensorFlow Serving on Kubernetes with Amazon EC2 Spot Instances https://aws.amazon.com/blogs/compute/tensorflow-serving-on-kubernetes-spot-instances