COMPX527 Assignment 1 Report

Glenn Cumming
Department of Computer Science
University of Waikato
Hamilton, New Zealand
glenn@hif.nz

Mitchell Grout
Department of Computer Science
University of Waikato
Hamilton, New Zealand
mjg44@students.waikato.ac.nz

Shufen Li
Department of Computer Science
University of Waikato
Hamilton, New Zealand
sl302@students.waikato.ac.nz

YingJun Huang
Department of Computer Science
University of Waikato
Hamilton, New Zealand
yh320@students.waikato.ac.nz

***Abstract*—Abstract Text**

***Index Terms*—AWS, COMPX527**

I. Solution Summary

Solution Summary

II. Motivation

We decided on object detection in the cloud in order to learn and appricate the challenges of providing a stateless, scalbale service that could be used for a variey of other projects. Examples included

- Contexutal threat analysis, such as humans detected in an area which should only include livestock.
- Numbercial anyalisis, such as the number of a species of wildlife in an area over time.
- Absence detection, indetifying objects that should be in an image but are not

III. Proposed Solution

In order to

IV. Solution Architecture

Solution Architecture

A. The Image Processing Container

The project uses the Darknet neural net fork by AlexeyAB [1] pre-trained on the COCO AWS images in context set. A Dockerfile generated the container template. This container uses the Darknet web server provided by komorin0521 [2] to create a web service that can be run on EC2 instances.

[1]https://github.com/AlexeyAB/darknet
[2]https://github.com/komorin0521/darknet_server

B. The Load Balancer

The object detection being inhernetly slow forces the use of laod balancing and scaling techniques. Using the AWS Elastic Load Balancer allowed us balance the HTTP requests over two or more EC2 servers running the Darknet servers. We could create as many Darknet servers as we wished, although of course this lent itself to waisted compute resources. Though we had great sucess using AWS AutoScaling Groups, which allows a lower and upper limit of EC2 instances running our Darknet server to be defined, this functionality was not avalible in the student accounts.

C. The Web User Interface

In order to have a better demo, it was decided to go ahead and produce a web site interface that would allow uploading

D. The Web API

The intended standard way to interact with the Darknet Object Detection Cluster is to access it via HTTP calls using the url http://<load_balancer_fqdn>/detect. As the system is inteneded for the use of non-private images no ssl was implemented on the load balancer, though it is supported. The code submitted for this project includes *simple_upload.py* and *forked_upload.py*, which are Python 3 examples for using the HTTP API.

V. Development

A. Technology

a) Darknet: Darknet is an open source object detection neural net. This technology was chosen for the following reasons.

- Open Source
- Features both Nvidia GPU and AVX2 compliation options. The AVX2 instruction set is avalible on all EC2 instances, so we compiled for it. If we wished to use the Keplar GPU P2 instances, then we could compile for it.
- Provides the Yolo3 data set. Yolo 3 is a pretrained on the COCO object detection, segmentation, and captioning

dataset avaliable on the Registry of Open Data on AWS [3]

b) Terraform: Terraform was given in as an example of the provisioning service to use in this assignment. AWS CloudFormation was another option suggested. The decision to go with terraform rather than CloudFormation was based on the following

- CloudFormation is prioritory and is specific to AWS cloud offierings TODOREF. Though this assigmnet is on a small and temporty scale, we still not want to use a technology that would create vendor lockin
- Terraform supports provisioning many different platforms, both open source stand priopritary, such as Azure, Goolge Cloud, Kubernetes and OpenStack [4]. Developng expericne in deployments with terraform therefore was demied to be more useful.

c) Ansible: Though Terraform is capable of running commands post-install in order to install services and other software needed for our cluster, our cluster used Ansible playbooks instead. Though it meant learning another technology, we demeed a good use of our time since:

- Terraform can only run scripts, which would have to be created.
- Ansible's langauge is very flxible and is created specifally for the purpose of delpoymnet.
- This is a recommended approach by HashiCorp, the devlopers of Terraform. [5]

d) Docker:

B. Future Improvements

C. Compairsion with existing object detection soltutions

Amazon Rekognition

VI. Security Assessment

Security Assessment

A. Data Security

Data Security

B. Access Security

Access Security

C. Network Security

Network Security

D. Vulnerability Assessment

E. Monitoring

AWS EC2 instance monitoring was enabled during the terraform creation of the cluster. Future development of the cluster would likely use additional monitoring of performance, costs and uptimes using dedicated EC2 instances, and such existing monitoring systems as Icinga, Promethues and Munin. Uptime moniroing would of nessescity be run etxernally, such as on internal machines or another cloud providers offering.

[3]https://registry.opendata.aws/fast-ai-coco/
[4]https://aws.amazon.com/cloudformation/
[5]https://www.hashicorp.com/resources/ansible-terraform-better-together

VII. Actual AWS Expenditure

TODO

VIII. Team Members Contributions

IX. Assignment Requirements Completion

X. Acknowledgment

This report was created in LaTex using the IEEEtran document class provided by IEEE template avaliable at [6] , and generated into PDF by Gnome LaTeX [7] .

[6]https://www.ieee.org/conferences/publishing/templates.html
[7]https://wiki.gnome.org/Apps/GNOME-LaTeX