CLOUD SECURITY

A project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

Bachelor of Technology (B.Tech.)

Submitted by

KARTIK JAITLY

35714803115

Under the supervision of

Mr. Pankaj

Assistant Professor (MAIT)

DEPARTMENT OF INFORMATION TECHNOLOGY

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

SEC-22, ROHINI, DELHI –110086, INDIA

DEC, 2018

ACKNOWLDGEMENTS

I extend my sincerest gratitude to my mentor, Mr. Pankaj, for his interest, guidance, and suggestions throughout the making of this project. I feel honored and privileged to have worked under his supervisory eye. He shared his knowledge and provided me with insights, without which this project could not have been completed. His inputs helped me clear my doubts and steer through mistakes and difficulties with ease. I would also like to extend my sincere regards to Dr.M.L Sharma (Head of Department, IT).

Kartik Jaitly

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project report titled, **"CLOUD SECURITY"** submitted by me in the partial fulfillment of the requirement of the award of the degree of **Bachelor of Technology (B.Tech.)** Submitted in the Department of **Information Technology,** Maharaja Agrasen Institute of Technology is an authentic record of my project work carried out under the guidance of (Supervisors name and affiliation)

Date **Kartik Jaitly**

Place: Delhi Roll No.: 35714803115

SUPERVISOR'S CERTIFICATE

It is to certify that the Project entitled **"CLOUD SECURITY"** which is being submitted by **Mr.Kartik Jaitly** to the Maharaja Agrasen Institute of Technology, Rohini in the fulfillment of the requirement for the award of the degree of **Bachelor of Technology (B.Tech.)**, is a record of bonafide project work carried out by him/her under my/ our guidance and supervision. The matter presented in this project report has not been submitted either in part or full to any University or Institute for award of any degree.

Signature
Mr.Pankaj Garg
(Supervisor)

Signature
Dr.M.L. Sharma
(HOD, IT)

LIST OF FIGURES

TABLE OF CONTENTS

ABSTRACT

The project focuses on the implementation of cloud computing for the purposes of AI chatbot applications and also the security of the cloud system.

The platforms used for the project include

- Amazon Web Services
- Annaconda
- Tensorflow
- Pandas
- Numpy

The project lead to better understanding of the cloud computing platform Amazon Web Services (AWS). Also the techniques used to create the AI chatbot such as the Tensorflow API and python modules like Pandas, Numpy, Tensorflow and setting up the virtual environment through anaconda.

Introduction

Why Cloud Security?

Cloud security is important for both business and personal users. Everyone wants to know that their information is safe and secure and businesses have legal obligations to keep client data secure, with certain sectors having more stringent rules about data storage.

Artificial intelligence and Machine learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

Chatbots

A chatbot (also known as a smartbots, talkbot, chatterbot, Bot, IM bot, interactive agent, Conversational interface or Artificial Conversational Entity) is a computer program or an artificial intelligence which conducts a conversation via auditory or textual methods.

Cloud and chatbots

Derived from “chat robot”, "chatbots" allow for highly engaging, conversational experiences, through voice and text, that can be customized and used on mobile devices, web browsers, and on popular chat platforms such as Facebook Messenger, or Slack. With the advent of deep learning technologies such as text-to-speech, automatic speech recognition, and natural language processing, chatbots that simulate human conversation and dialogue can now be found in call center and customer service workflows, DevOps management, and as personal assistants. AI has reached a stage in which chatbots can have engaging conversations with humans. The Chatbot ecosystem is already robust, including investment from large corporations such us Facebook's Bots and AWS's LEX Built-in integration with cloud services is already on place

One of the focus area of Innovation Zone's team during H2 2017 will be to support customers and partners to validate chatbot technologies and its implications, as for now we are integrating a NAO robot with AWS LEX service in a project in common with our colleagues at Ericsson Garage Gothenburg.

Chatbots hosted on clouds is one the most used applications of cloud computing. With the rise of Artificial Intelligence and Machine Learning, it most cost effective and better to host the AI-Chatbots on the Cloud Vendors instead of getting expensive hardware resources to keep up with the high computing power required for these AI – Chatbots to Train, Test and Serve on.

It's is also made simple with the pre-built chatbot support on all the major Cloud Vendors like Amazon Web Services Lex and Microsoft Azure.

It is highly cost effective and resourceful to host these bots on these servers of cloud vendors as it provides highly flexible resources according to the requirements of the organization.

Technologies Used

Tensorflow

TensorFlow™ is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors". In June 2016, Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

Various Tensorflow Clients:

Python

Python is one of those rare languages which can claim to be both *simple* and *powerful*. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python Modules used:

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Pandas

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Math

This module is always available. It provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

OS

The OS module in Python provides a way of using operating system dependent functionality.

The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

You can find important information about your location or about the process.

In this post I will show some of these functions.

Random

This module implements pseudo-random number generators for various distributions. For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement. On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available. Almost all module functions depend on the basic function random(), which generates a random float uniformly in the semi-open range [0.0, 1.0). Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of 2**19937-1. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

data_utils

It is a vocabulary processor of the tflearn model for data science and machine learning. It is an important library to work for the project. Maps documents to sequences of word ids. TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

re

This module provides regular expression matching operations similar to those found in Perl. Both patterns and strings to be searched can be Unicode strings as well as 8-bit strings.

Regular expressions use the backslash character ('\') to indicate special forms or to allow special characters to be used without invoking their special meaning. This collides with Python's usage of the same character for the same purpose in string literals; for example, to match a literal backslash, one might have to write '\\\\' as the pattern string, because the regular expression must be \\, and each backslash must be expressed as \\ inside a regular Python string literal.

__future__

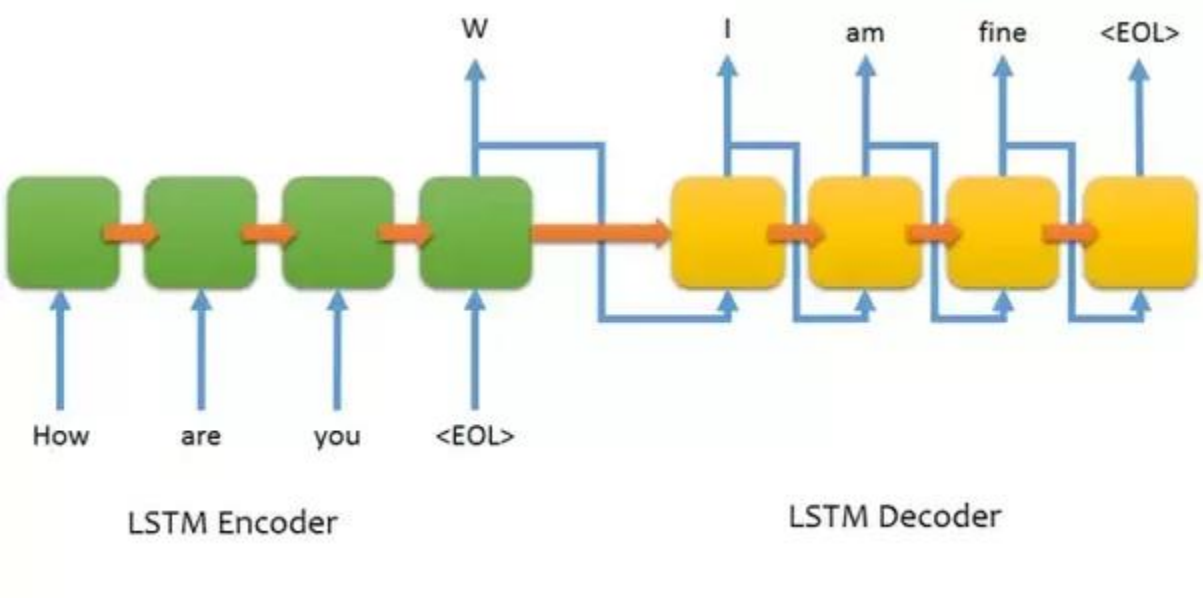
__future__ is a real module, and serves three purposes:

To avoid confusing existing tools that analyze import statements and expect to find the modules they're importing.

To ensure that future statements run under releases prior to 2.1 at least yield runtime exceptions (the import of __future__ will fail, because there was no module of that name prior to 2.1).

To document when incompatible changes were introduced, and when they will be — or were — made mandatory. This is a form of executable documentation, and can be inspected programmatically via importing __future__ and examining its contents.

seq2seq_model



Deep neural networks that are mainly feedforward fully connected neural network are powerful but not really appropriate for sequential data such as time series data or language. They are very good to map input data to discrete output or continuous variable but not sequence to sequence mapping. Seq2seq model learns from variable sequence input fixed length sequence output. It uses two LSTM model, one learns vector representation from input sequence of fixed dimensionality and another LSTM learns to decode from this input vector to target sequence. LSTM is a variant of recurrent neural network that solves problem of handling long sequences using different gates.

Seq2seq model solves a specific limitation of deep neural network. DNN requires fixed length vector representation of input and output. But machine translation where input language is converted to target language input and output sentence length can vary. In question answering problem any length of input question needs to mapped to any length of answer sequence.

Anaconda

Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called Anaconda Navigator, so it eliminates the need to learn to install each library independently.

The open source data packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together.

You can also make your own custom packages using the conda build command, and you can share them with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

Cloud Computing

Simply put, cloud computing is the delivery of computing services—servers, storage, databases, networking, software, analytics, intelligence and more—over the Internet ("the cloud") to offer faster innovation, flexible resources and economies of scale. You typically pay only for cloud services you use, helping lower your operating costs, run your infrastructure more efficiently and scale as your business needs change.

Types of cloud computing

Not all clouds are the same and not one type of cloud computing is right for everyone. Several different models, types and services have evolved to help offer the right solution for your needs.

Types of cloud deployments: public, private and hybrid

First, you need to determine the type of cloud deployment or cloud computing architecture that your cloud services will be implemented on. There are three different ways to deploy cloud services: on a public cloud, private cloud or hybrid cloud.

Public cloud

Public clouds are owned and operated by a third-party cloud service providers, which deliver their computing resources like servers and storage over the Internet. Microsoft Azure is an example of a public cloud. With a public cloud, all hardware, software and other supporting infrastructure is owned and managed by the cloud provider. You access these services and manage your account using a web browser.

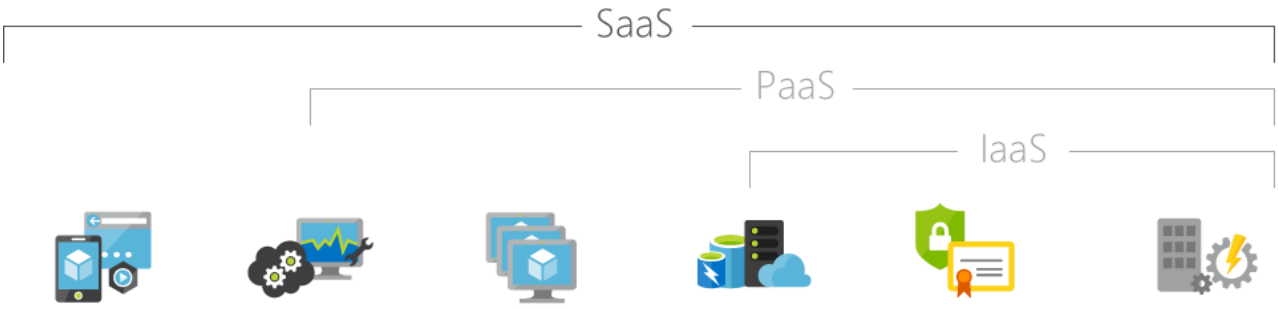
Private cloud

A private cloud refers to cloud computing resources used exclusively by a single business or organization. A private cloud can be physically located on the company's on-site datacenter. Some companies also pay third-party service providers to host their private cloud. A private cloud is one in which the services and infrastructure are maintained on a private network.

Hybrid cloud

Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. By allowing data and applications to move between private and public clouds, a hybrid cloud gives your business greater flexibility, more deployment options and helps optimize your existing infrastructure, security and compliance.

Types of cloud services: IaaS, PaaS, serverless and SaaS



Most cloud computing services fall into four broad categories: infrastructure as a service (IaaS), platform as a service (PaaS), serverless and software as a service (SaaS). These are sometimes called the cloud computing stack because they build on top of one another. Knowing what they are and how they are different makes it easier to accomplish your business goals.

Infrastructure as a service (IaaS)

The most basic category of cloud computing services. With IaaS, you rent IT infrastructure—servers and virtual machines (VMs), storage, networks, operating systems—from a cloud provider on a pay-as-you-go basis.

Platform as a service (PaaS)

Platform as a service refers to cloud computing services that supply an on-demand environment for developing, testing, delivering and managing software applications. PaaS is designed to make it easier for developers to quickly create web or mobile apps, without worrying about setting up or managing the underlying infrastructure of servers, storage, network and databases needed for development.

Serverless computing

Overlapping with PaaS, serverless computing focuses on building app functionality without spending time continually managing the servers and infrastructure required to do so. The cloud provider handles the setup, capacity planning and server management for you. Serverless architectures are highly scalable and event-driven, only using resources when a specific function or trigger occurs.

Software as a service (SaaS)

Software as a service is a method for delivering software applications over the Internet, on demand and typically on a subscription basis. With SaaS, cloud providers host and manage the software application and underlying infrastructure and handle any maintenance, like software upgrades and security patching. Users connect to the application over the Internet, usually with a web browser on their phone, tablet or PC.

Top benefits of cloud computing

Cloud computing is a big shift from the traditional way businesses think about IT resources. Here are seven common reasons organizations are turning to cloud computing services:

Cost

Cloud computing eliminates the capital expense of buying hardware and software and setting up and running on-site datacenters—the racks of servers, the round-the-clock electricity for power and cooling, the IT experts for managing the infrastructure. It adds up fast.

Speed

Most cloud computing services are provided self service and on demand, so even vast amounts of computing resources can be provisioned in minutes, typically with just a few mouse clicks, giving businesses a lot of flexibility and taking the pressure off capacity planning.

Global scale

The benefits of cloud computing services include the ability to scale elastically. In cloud speak, that means delivering the right amount of IT resources—for example, more or less computing power, storage, bandwidth—right when it is needed and from the right geographic location.

Productivity

On-site datacenters typically require a lot of "racking and stacking"—hardware set up, software patching and other time-consuming IT management chores. Cloud computing removes the need for many of these tasks, so IT teams can spend time on achieving more important business goals.

Performance

The biggest cloud computing services run on a worldwide network of secure datacenters, which are regularly upgraded to the latest generation of fast and efficient computing hardware. This

offers several benefits over a single corporate datacenter, including reduced network latency for applications and greater economies of scale.

Amazon Web Services

Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser. The browser acts as a window into the virtual computer, letting subscribers log-in, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for themselves and their customers.

Types of Instances in AWS

S3 Instance

Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers.

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

This guide explains the core concepts of Amazon S3, such as buckets and objects, and how to work with these resources using the Amazon S3 application programming interface (API).

EC-2 Instance

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Amazon EC2 provides the following features:

- Virtual computing environments, known as instances
- Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as regions and Availability Zones
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

Security Concerns In Cloud Computing

1. Data Breaches

Cloud computing and services are relatively new, yet data breaches in all forms have existed for years. The question remains: “With sensitive data being stored online rather than on premise, is the cloud inherently less safe?”

A study conducted by the Ponemon Institute entitled “Man In Cloud Attack” reports that over 50 percent of the IT and security professionals surveyed believed their organization’s security measures to protect data on cloud services are low. This study used nine scenarios, where a data breach had occurred, to determine if that belief was founded in fact.

After evaluating each scenario, the report concluded that overall data breaching was three times more likely to occur for businesses that utilize the cloud than those that don’t. The simple conclusion is that the cloud comes with a unique set of characteristics that make it more vulnerable.

2. Hijacking of Accounts

The growth and implementation of the cloud in many organizations has opened a whole new set of issues in account hijacking.

Attackers now have the ability to use your (or your employees’) login information to remotely access sensitive data stored on the cloud; additionally, attackers can falsify and manipulate information through hijacked credentials.

Other methods of hijacking include scripting bugs and reused passwords, which allow attackers to easily and often without detection steal credentials. In April 2010 Amazon faced a cross-site scripting bug that targeted customer credentials as well. Phishing, keylogging, and buffer overflow all present similar threats. However, the most notable new threat – known as the Man In Cloud Attack – involves the theft of user tokens which cloud platforms use to verify individual devices without requiring logins during each update and sync.

3. Insider Threat

An attack from inside your organization may seem unlikely, but the insider threat does exist. Employees can use their authorized access to an organization’s cloud-based services to misuse or access information such as customer accounts, financial forms, and other sensitive information.

Additionally, these insiders don't even need to have malicious intentions.

A study by Imperva, "Inside Track on Insider Threats" found that an insider threat was the misuse of information through malicious intent, accidents or malware. The study also examined four best practices companies could follow to implement a secure strategy, such as business partnerships, prioritizing initiatives, controling access, and implementing technology.

4. Malware Injection

Malware injections are scripts or code embedded into cloud services that act as "valid instances" and run as SaaS to cloud servers. This means that malicious code can be injected into cloud services and viewed as part of the software or service that is running within the cloud servers themselves.

Once an injection is executed and the cloud begins operating in tandem with it, attackers can eavesdrop, compromise the integrity of sensitive information, and steal data. Security Threats On Cloud Computing Vulnerabilities, a report by the East Carolina University, reviews the threats of malware injections on cloud computing and states that "malware injection attack has become a major security concern in cloud computing systems."

5. Abuse of Cloud Services

The expansion of cloud-based services has made it possible for both small and enterprise-level organizations to host vast amounts of data easily. However, the cloud's unprecedented storage capacity has also allowed both hackers and authorized users to easily host and spread malware, illegal software, and other digital properties.

In some cases this practice affects both the cloud service provider and its client. For example, privileged users can directly or indirectly increase the security risks and as a result infringe upon the terms of use provided by the service provider.

These risks include the sharing of pirated software, videos, music, or books, and can result in legal consequences in the forms of fines and settlements with the U.S. Copyright Law reaching up to $250,000. Depending on the damage, these fines can be even more cost prohibitive. You can reduce your exposure to risk by monitoring usage and setting guidelines for what your employees host in the cloud. Service providers and legal entities, such as CSA have defined what is abusive or inappropriate behavior along with methods of detecting such behaviors.

6. Insecure APIs

Application Programming Interfaces (API) give users the opportunity to customize their cloud experience.

However, APIs can be a threat to cloud security because of their very nature. Not only do they give companies the ability to customize features of their cloud services to fit business needs, but they also authenticate, provide access, and effect encryption.

As the infrastructure of APIs grows to provide better service, so do its security risks. APIs give programmers the tools to build their programs to integrate their applications with other job-critical software. A popular and simple example of an API is YouTube, where developers have the ability to integrate YouTube videos into their sites or applications.

The vulnerability of an API lies in the communication that takes place between applications. While this can help programmers and businesses, they also leave exploitable security risks.

7. Denial of Service Attacks

Unlike other kind of cyberattacks, which are typically launched to establish a long-term foothold and hijack sensitive information, denial of service assaults do not attempt to breach your security perimeter. Rather, they attempt to make your website and servers unavailable to legitimate users. In some cases, however, DoS is also used as a smokescreen for other malicious activities, and to take down security appliances such as web application firewalls.

8. Insufficient Due Diligence

Most of the issues we've looked at here are technical in nature, however this particular security gap occurs when an organization does not have a clear plan for its goals, resources, and policies for the cloud. In other words, it's the people factor.

Additionally, insufficient due diligence can pose a security risk when an organization migrates to the cloud quickly without properly anticipating that the services will not match customer's expectation.

This is especially important to companies whose data falls under regulatory laws like PII, PCI, PHI, and FERPA or those that handle financial data for customers.

9. Shared Vulnerabilities

Cloud security is a shared responsibility between the provider and the client.

This partnership between client and provider requires the client to take preventative actions to protect their data. While major providers like Box, Dropbox, Microsoft, and Google do have standardized procedures to secure their side, fine grain control is up to you, the client.

As Skyfence points out in its article “Office 365 Security & Share Responsibility,” this leaves key security protocols – such as the protection of user passwords, access restrictions to both files and devices, and multi-factor authentication – firmly in your hands.

The bottom line is that clients and providers have shared responsibilities, and omitting yours can result in your data being compromised.

10. Data Loss

Data on cloud services can be lost through a malicious attack, natural disaster, or a data wipe by the service provider. Losing vital information can be devastating to businesses that don’t have a recovery plan. Amazon is an example of an organization that suffered data loss by permanently destroying many of its own customers’ data in 2011.

Google was another organization that lost data when its power grid was struck by lightning four times.

Securing your data means carefully reviewing your provider’s back up procedures as they relate to physical storage locations, physical access, and physical disasters.

Implementation

Setting up EC-2 Instance on AWS

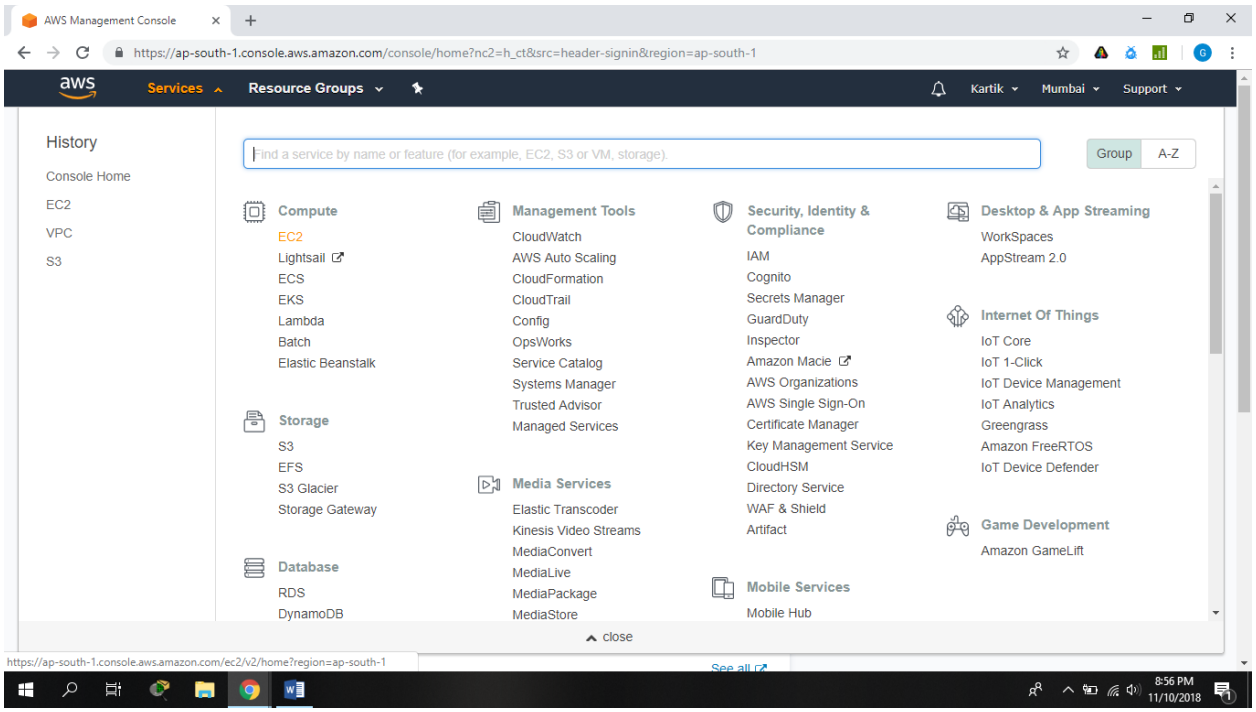


Fig – 1.1 select type of instance (EC2) to set up

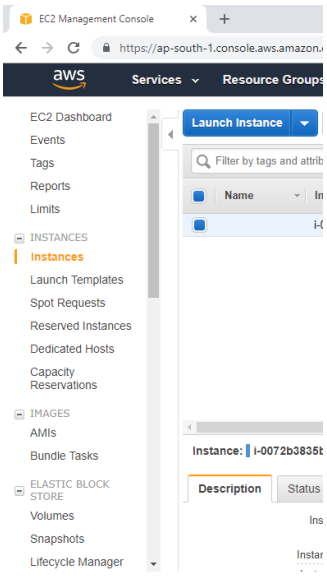


Fig 1.2 go and select instances and launch instance

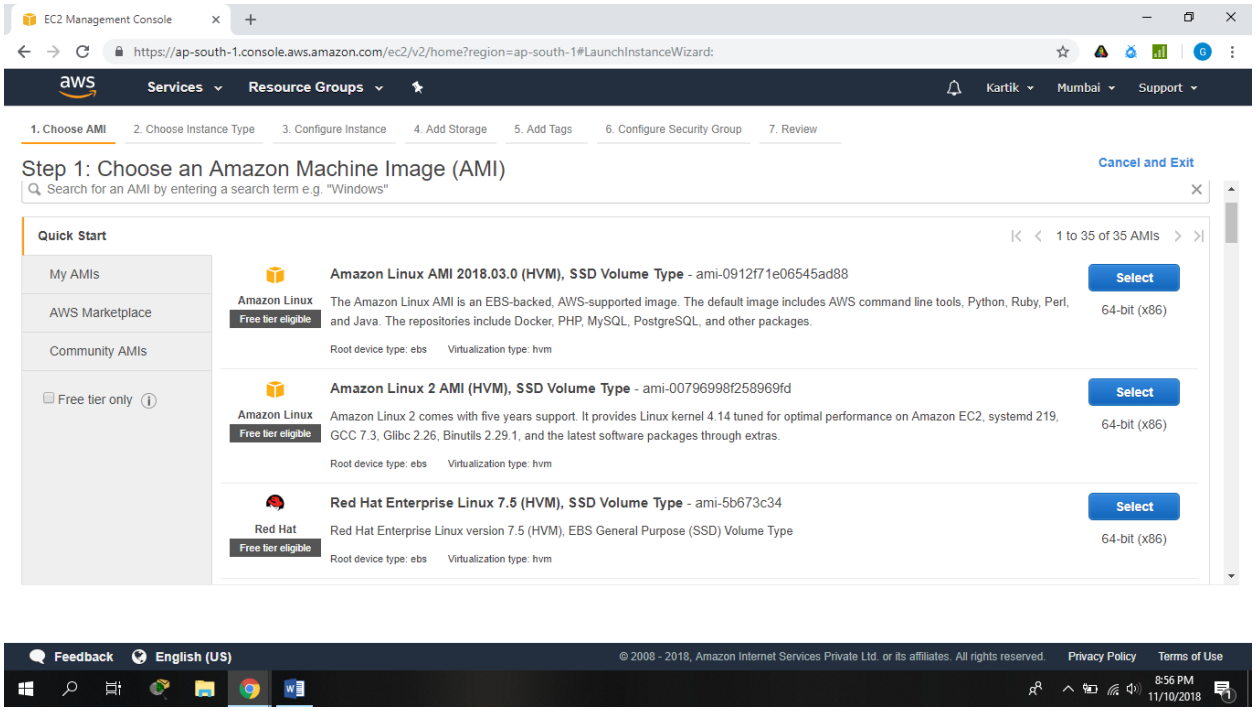


Fig – 1.3 Set up the type of AMI you need

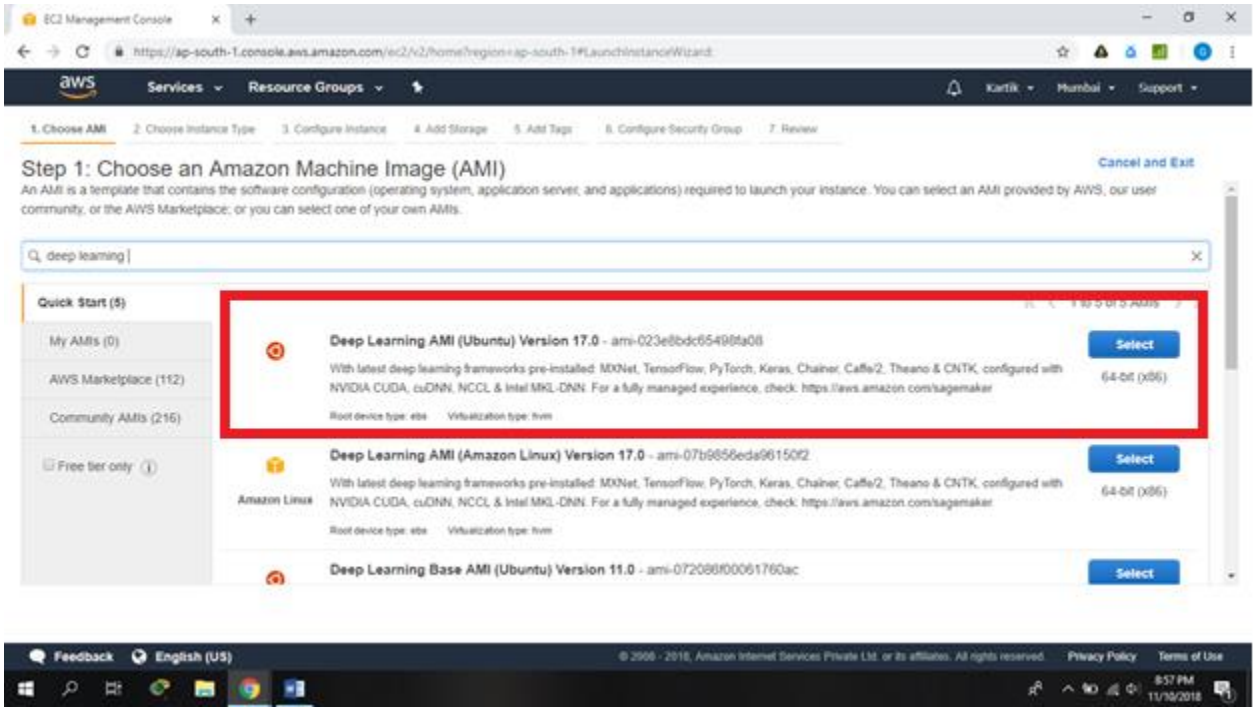


Fig – 1.4 Select Deep Learning AMI

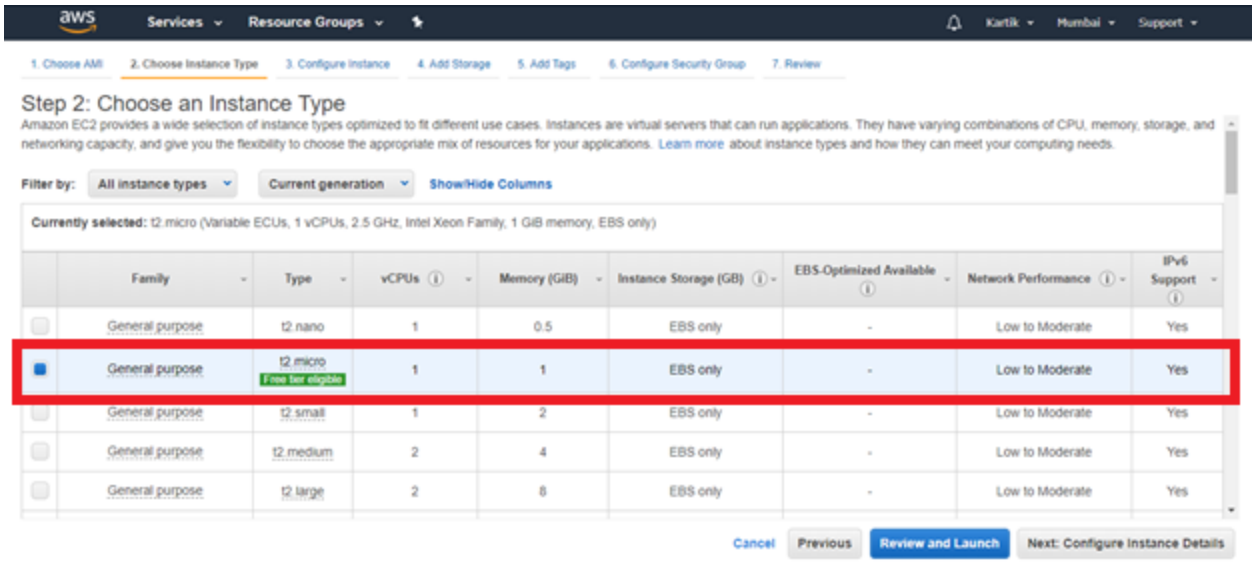


Fig – 1.5 Set up t2.micro instance

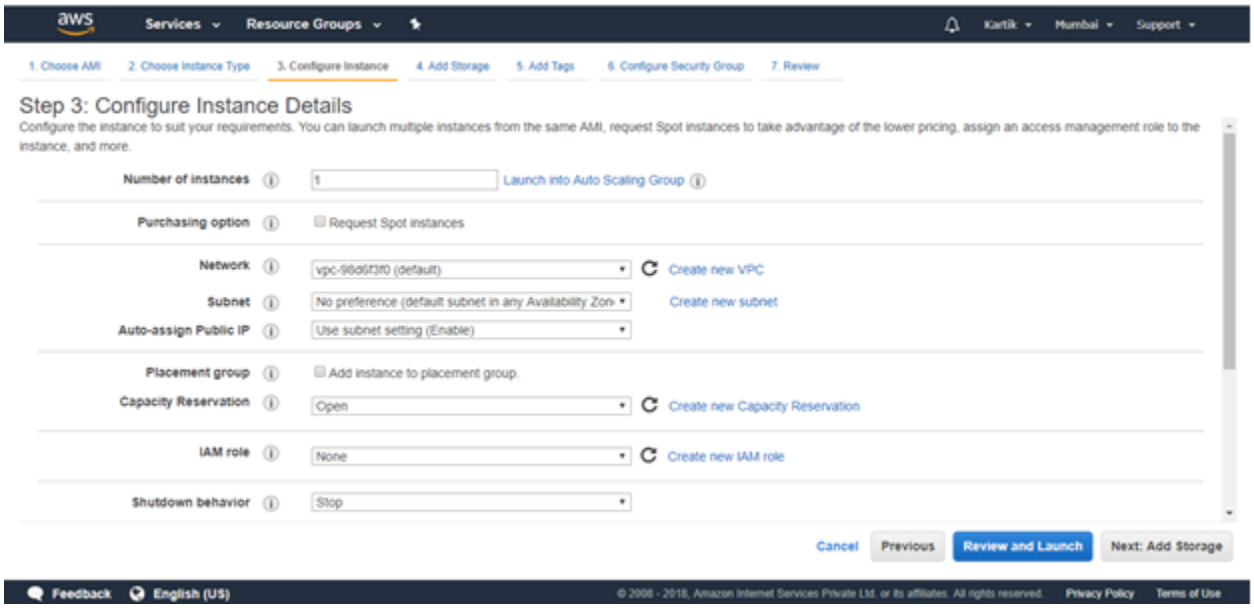


Fig – 1.6 Set up instance

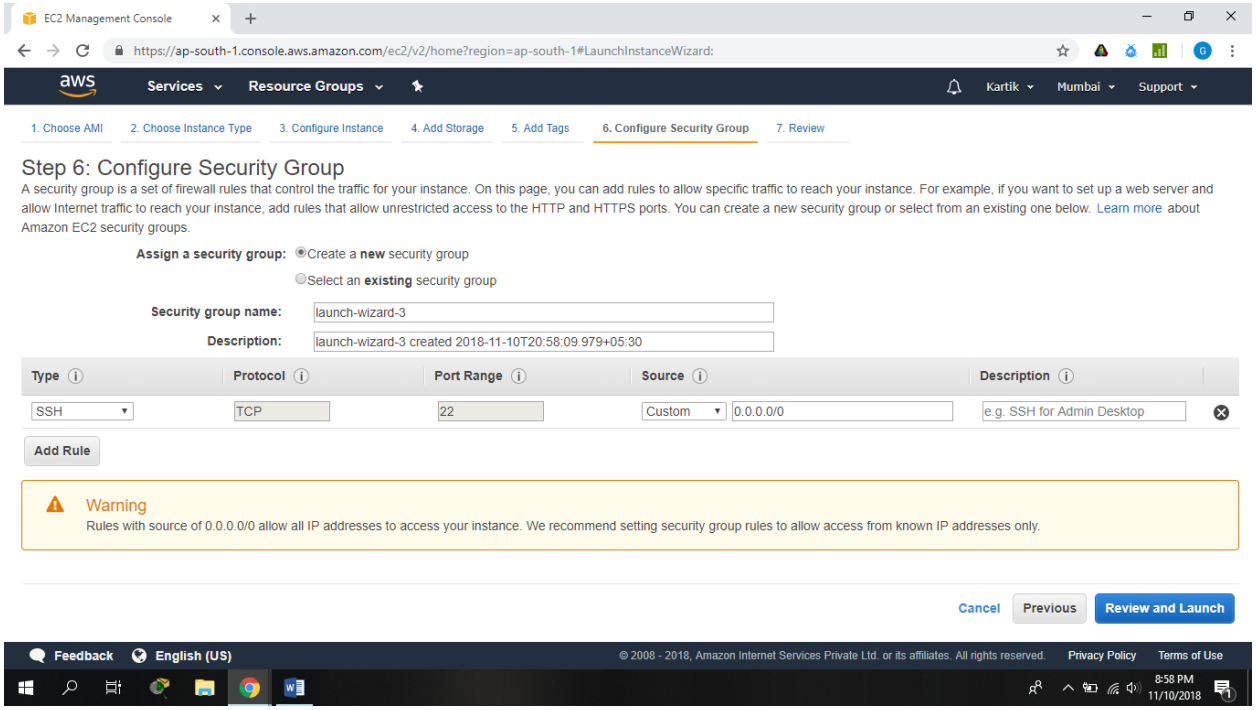


Fig – 1.7 Configuring Security

Connecting to instance on AWS

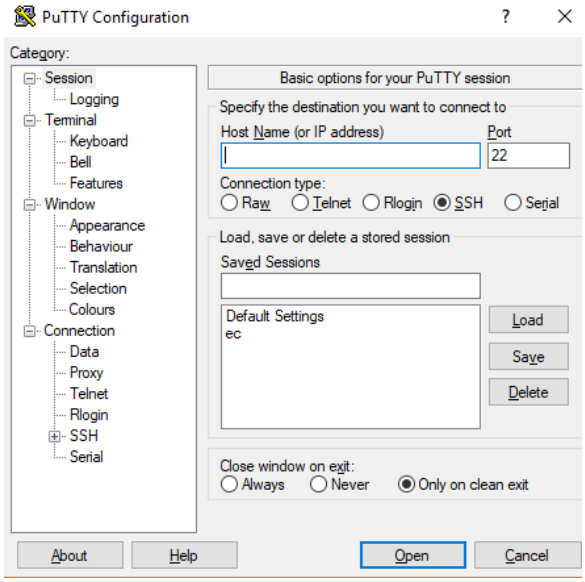


Fig – 2.1 Installing PuTTy for SSH to Instance

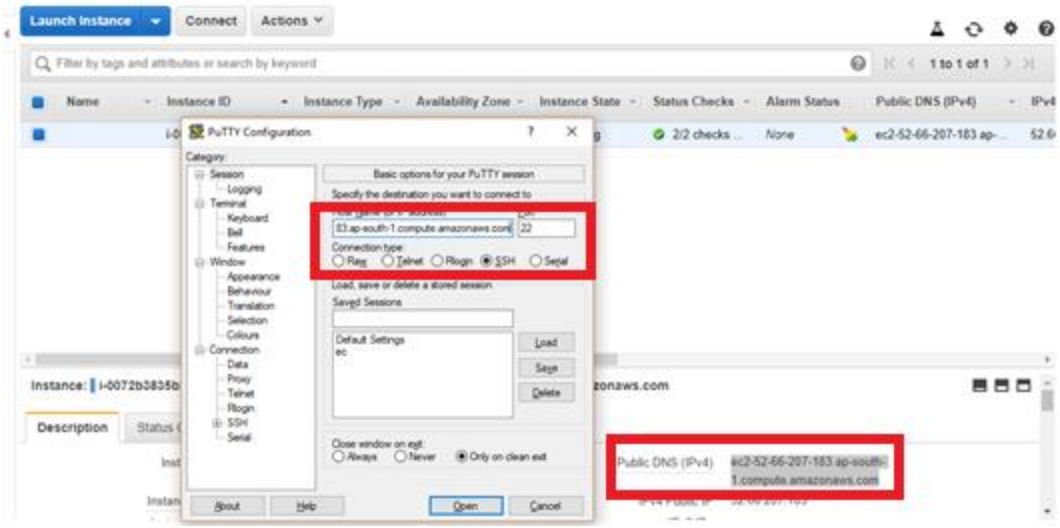


Fig – 2.2 PuTTy for SSH to Instance

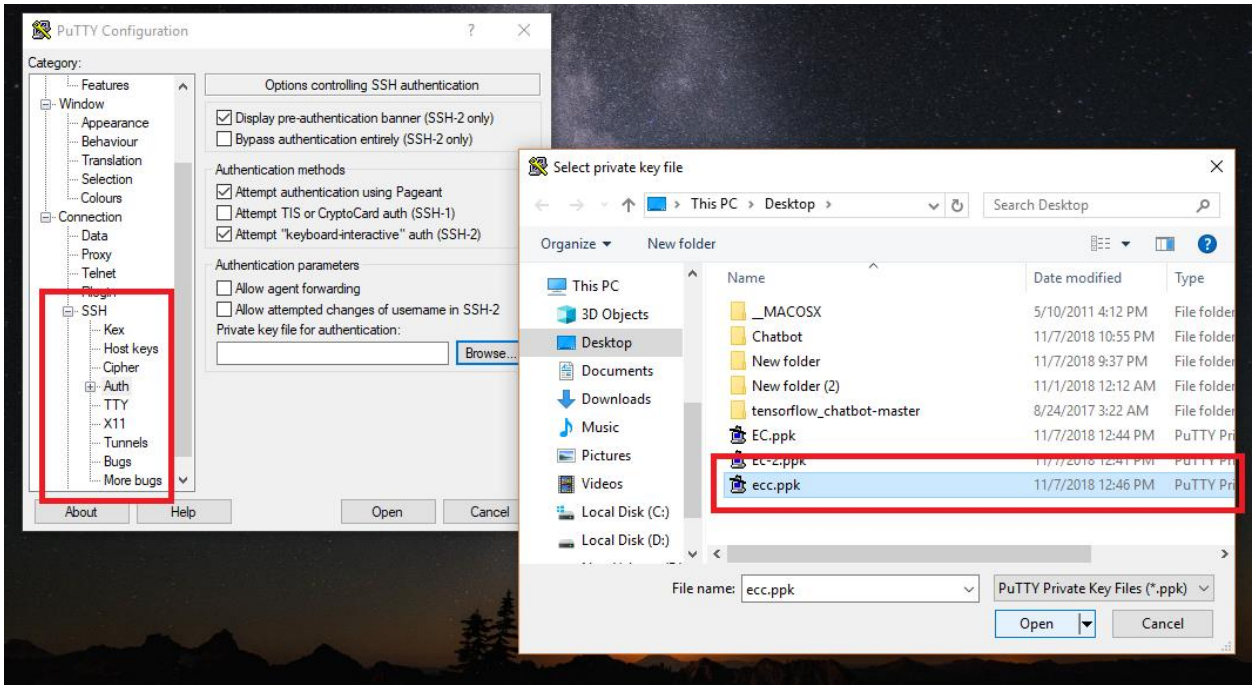


Fig – 2.3 authenticate using KEY PAIR SECURITY

Fig – 3.1 use **conda create –n tensor** for creating Virtual Environment

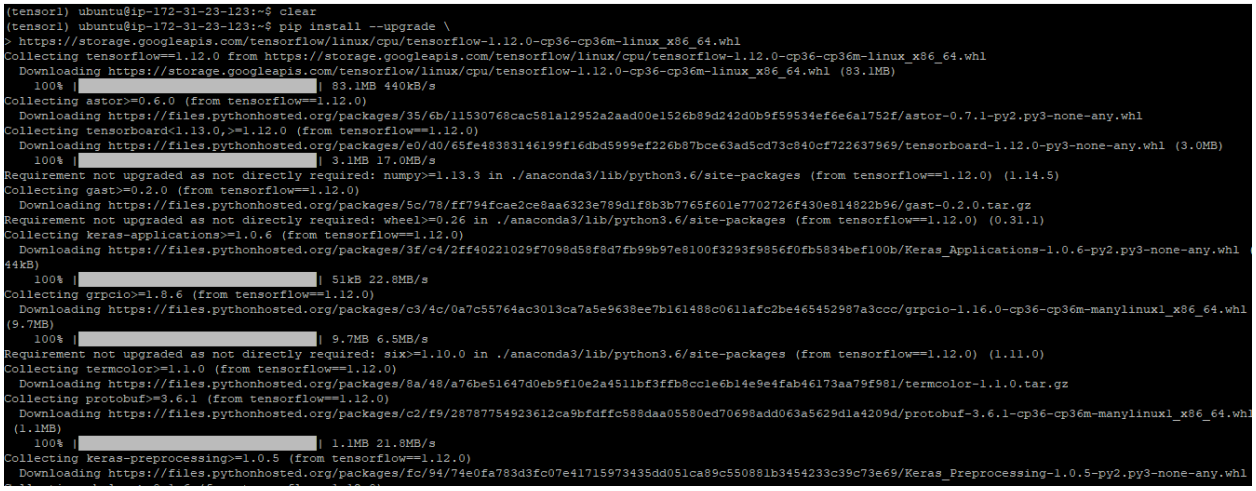


Fig – 3.2 use **pip install –upgrade \tensor_flow_URL** for installing required version

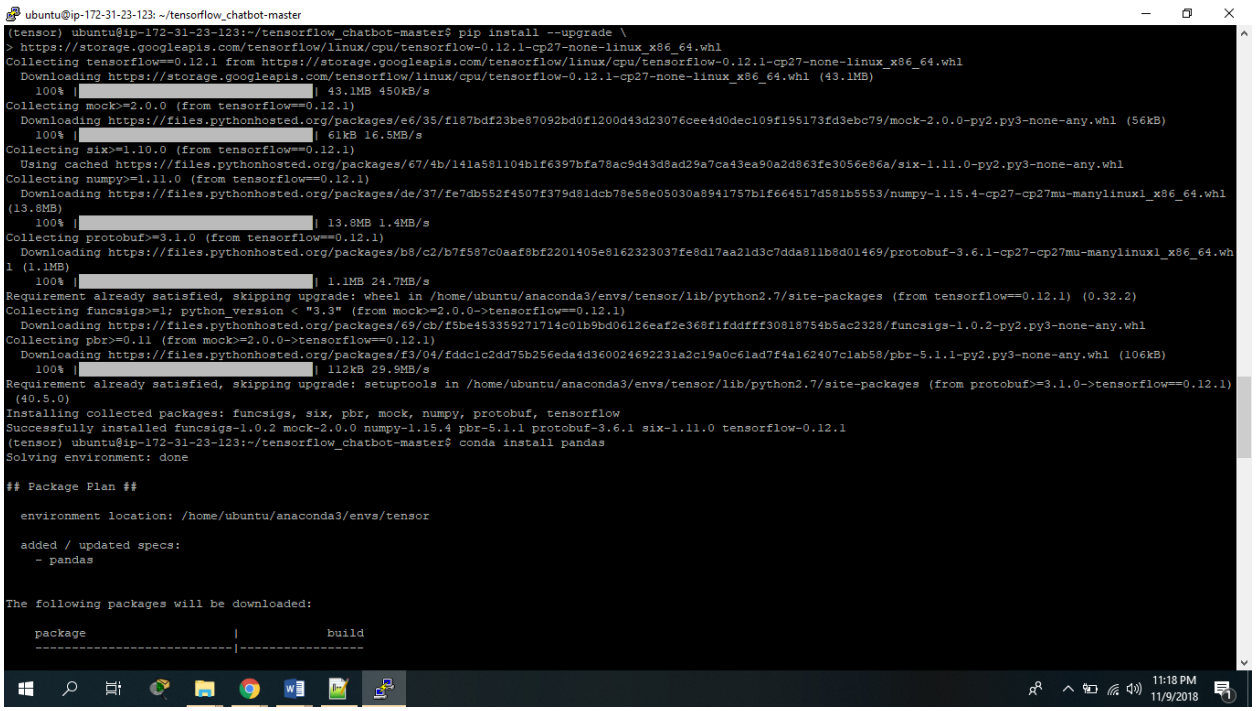


Fig – 3.3 use **pip install** to install all dependencies

Source code

```
  data_set = [[] for _ in _buckets]
  with tf.gfile.GFile(source_path, mode="r") as source_file:
    with tf.gfile.GFile(target_path, mode="r") as target_file:
      source, target = source_file.readline(), target_file.readline()
      counter = 0
      while source and target and (not max_size or counter < max_size):
        counter += 1
        if counter % 100000 == 0:
          print("  reading data line %d" % counter)
          sys.stdout.flush()
        source_ids = [int(x) for x in source.split()]
        target_ids = [int(x) for x in target.split()]
        target_ids.append(data_utils.EOS_ID)
        for bucket_id, (source_size, target_size) in enumerate(_buckets):
          if len(source_ids) < source_size and len(target_ids) < target_size:
            data_set[bucket_id].append([source_ids, target_ids])
            break
        source, target = source_file.readline(), target_file.readline()
  return data_set


def create_model(session, forward_only):

  """Create model and initialize or load parameters"""
  model = seq2seq_model.Seq2SeqModel( gConfig['enc_vocab_size'], gConfig['dec_vocab_size'], _buckets, gConfig['layer_size'], gConfig['num_layers'], gConfig['max
```

Fig – 4.1 using Tensorflow to create seq2seq model

```
def train():
  # prepare dataset
  print("Preparing data in %s" % gConfig['working_directory'])
  enc_train, dec_train, enc_dev, dec_dev, _, _ = data_utils.prepare_custom_data(gConfig['working_directory'],gConfig['train_enc

  # Only allocate 2/3 of the gpu memory to allow for running gpu-based predictions while training:
  gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.666)
  config = tf.ConfigProto(gpu_options=gpu_options)
  config.gpu_options.allocator_type = 'BFC'

  with tf.Session(config=config) as sess:
    # Create model.
    print("Creating %d layers of %d units." % (gConfig['num_layers'], gConfig['layer_size']))
    model = create_model(sess, False)

    # Read data into buckets and compute their sizes.
    print ("Reading development and training data (limit: %d)."
           % gConfig['max_train_data_size'])
    dev_set = read_data(enc_dev, dec_dev)
    train_set = read_data(enc_train, dec_train, gConfig['max_train_data_size'])
    train_bucket_sizes = [len(train_set[b]) for b in xrange(len(_buckets))]
    train_total_size = float(sum(train_bucket_sizes))

    # A bucket scale is a list of increasing numbers from 0 to 1 that we'll use
    # to select a bucket. Length of [scale[i], scale[i+1]] is proportional to
    # the size if i-th training bucket, as used later.
    train_buckets_scale = [sum(train_bucket_sizes[:i + 1]) / train_total_size
                           for i in xrange(len(train_bucket_sizes))]
```

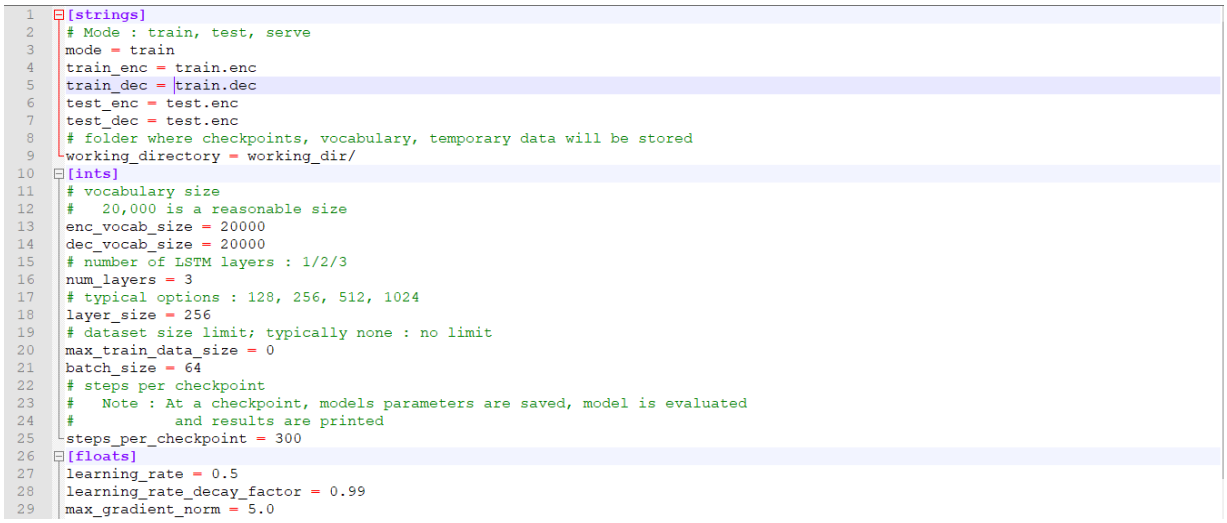
Fig – 4.2 function to train on the dataset

```
def self_test():
  """Test the translation model."""
  with tf.Session() as sess:
    print("Self-test for neural translation model.")
    # Create model with vocabularies of 10, 2 small buckets, 2 layers of 32.
    model = seq2seq_model.Seq2SeqModel(10, 10, [(3, 3), (6, 6)], 32, 2,
                                       5.0, 32, 0.3, 0.99, num_samples=8)
    sess.run(tf.initialize_all_variables())

    # Fake data set for both the (3, 3) and (6, 6) bucket.
    data_set = ([([1, 1], [2, 2]), ([3, 3], [4]), ([5], [6])],
                [([1, 1, 1, 1, 1], [2, 2, 2, 2, 2]), ([3, 3, 3], [5, 6])])
    for _ in xrange(5):  # Train the fake model for 5 steps.
      bucket_id = random.choice([0, 1])
      encoder_inputs, decoder_inputs, target_weights = model.get_batch(
          data_set, bucket_id)
      model.step(sess, encoder_inputs, decoder_inputs, target_weights,
                 bucket_id, False)


def init_session(sess, conf='seq2seq.ini'):
    global gConfig
    gConfig = get_config(conf)

    # Create model and load parameters.
    model = create_model(sess, True)
    model.batch_size = 1  # We decode one sentence at a time.

    # Load vocabularies.
    enc_vocab_path = os.path.join(gConfig['working_dir'],"vocab%d.enc" % gConfig['enc_vocab_size'])
    dec_vocab_path = os.path.join(gConfig['working_dir'],"vocab%d.dec" % gConfig['dec_vocab_size'])
```

Fig – 4.3 function to test and initialize the model to serve

```
def decode_line(sess, model, enc_vocab, rev_dec_vocab, sentence):
    # Get token-ids for the input sentence.
    token_ids = data_utils.sentence_to_token_ids(tf.compat.as_bytes(sentence), enc_vocab)

    # Which bucket does it belong to?
    bucket_id = min([b for b in xrange(len(_buckets)) if _buckets[b][0] > len(token_ids)])

    # Get a 1-element batch to feed the sentence to the model.
    encoder_inputs, decoder_inputs, target_weights = model.get_batch({bucket_id: [(token_ids, [])]}, bucket_id)

    # Get output logits for the sentence.
    _, _, output_logits = model.step(sess, encoder_inputs, decoder_inputs, target_weights, bucket_id, True)

    # This is a greedy decoder - outputs are just argmaxes of output_logits.
    outputs = [int(np.argmax(logit, axis=1)) for logit in output_logits]

    # If there is an EOS symbol in outputs, cut them at that point.
    if data_utils.EOS_ID in outputs:
        outputs = outputs[:outputs.index(data_utils.EOS_ID)]

    return " ".join([tf.compat.as_str(rev_dec_vocab[output]) for output in outputs])

if __name__ == '__main__':
    if len(sys.argv) - 1:
        gConfig = get_config(sys.argv[1])
    else:
        # get configuration from seq2seq.ini
        gConfig = get_config()

    print('\n>> Mode : %s\n' %(gConfig['mode']))
```

Fig – 4.4 function to decode the encoded data



```
[strings]
# Mode : train, test, serve
mode = train
train_enc = train.enc
train_dec = train.dec
test_enc = test.enc
test_dec = test.enc
# folder where checkpoints, vocabulary, temporary data will be stored
working_directory = working_dir/
[ints]
# vocabulary size
#   20,000 is a reasonable size
enc_vocab_size = 20000
dec_vocab_size = 20000
# number of LSTM layers : 1/2/3
num_layers = 3
# typical options : 128, 256, 512, 1024
layer_size = 256
# dataset size limit; typically none : no limit
max_train_data_size = 0
batch_size = 64
# steps per checkpoint
#   Note : At a checkpoint, models parameters are saved, model is evaluated
#          and results are printed
steps_per_checkpoint = 300
[floats]
learning_rate = 0.5
learning_rate_decay_factor = 0.99
max_gradient_norm = 5.0
```

Fig – 4.5 setting the model to train/test/serve in seq2seq.ini

Executing Chatbot on AWS

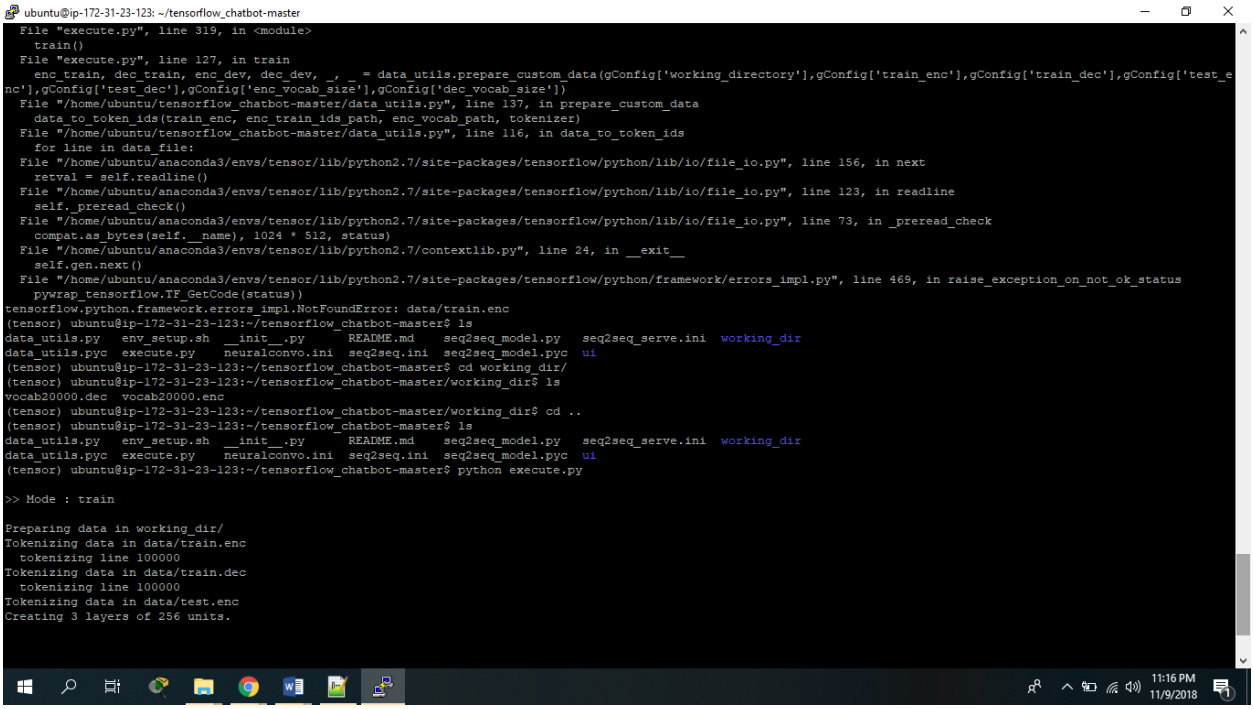


Fig – 5.1 setting the model to train/test/serve in seq2seq.ini

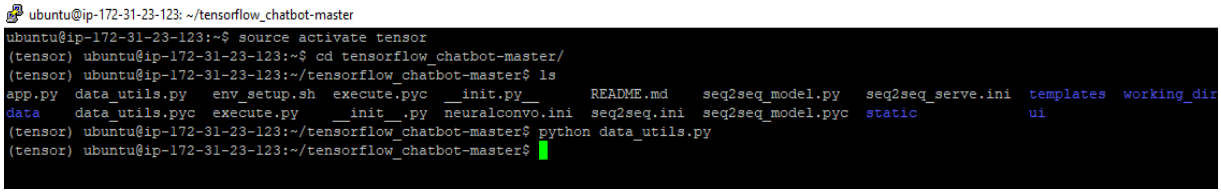


Fig – 5.2 Enter tensor and execute **data_utils.py**

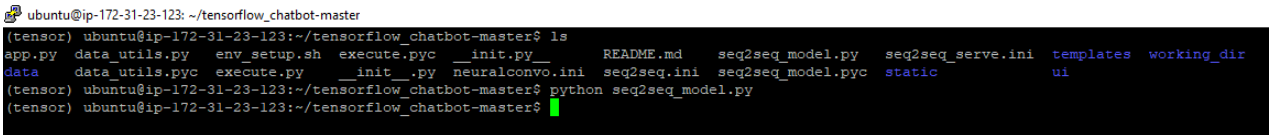


Fig – 5.3 execute seq2seq model using **python se2seq_model.py**

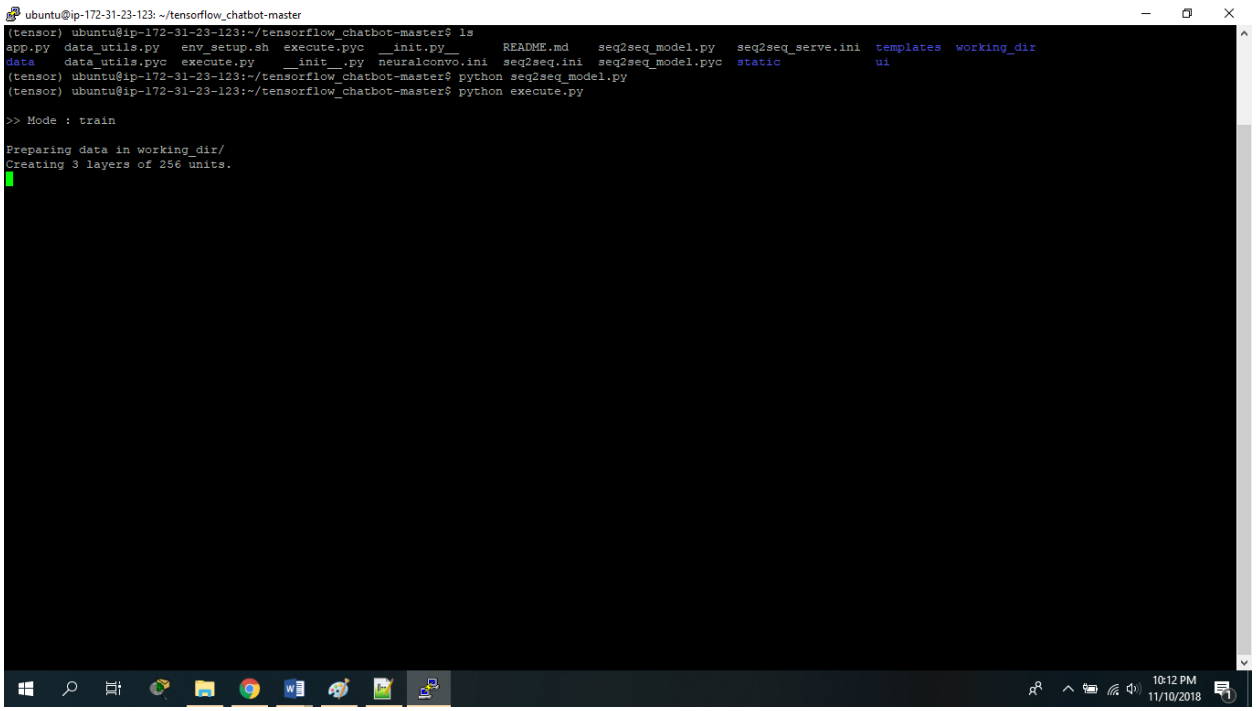


Fig – 5.4 execute execute.py using **python execute.py**

(Leave for several hours of training)

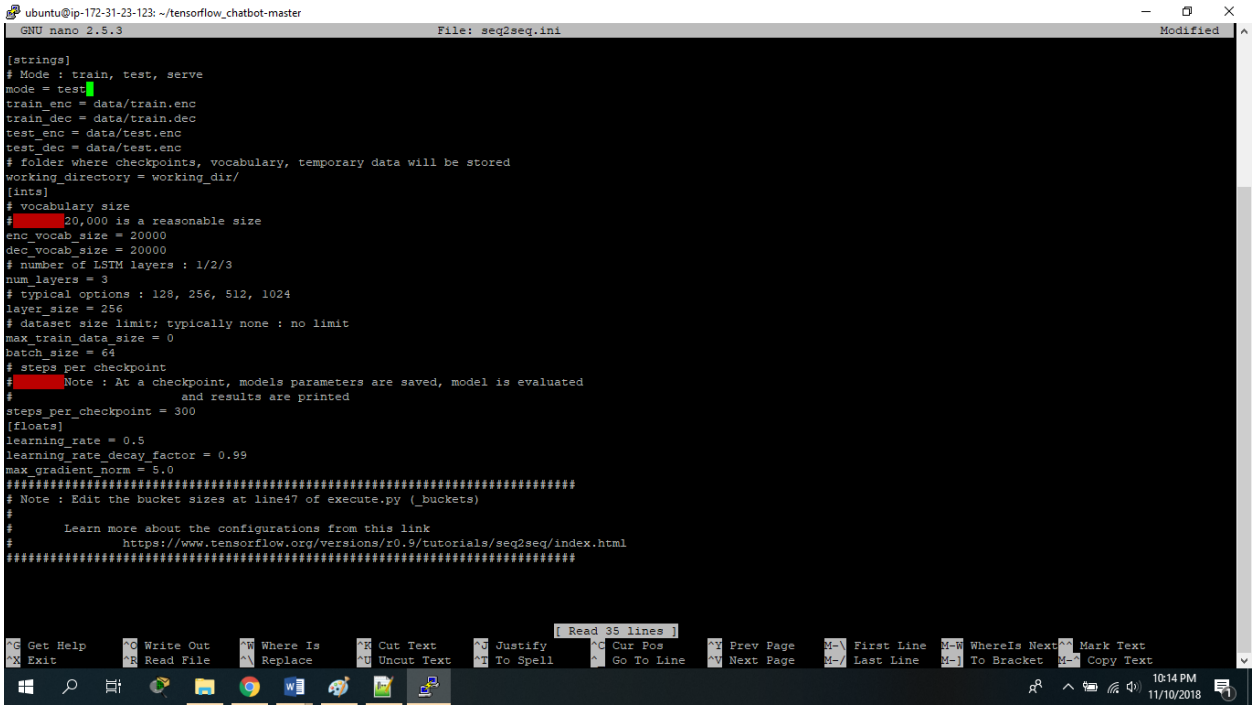


Fig – 5.5 change the mode to **test**

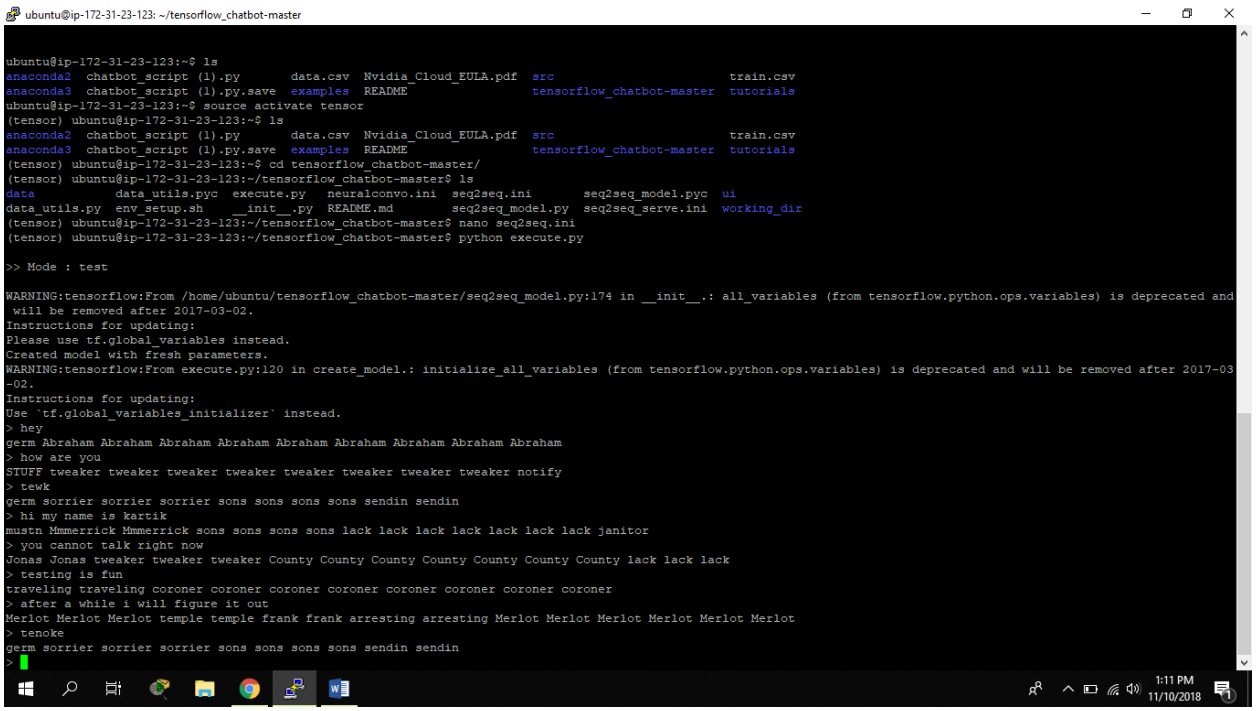


Fig – 5.6 test the model by entering random inputs

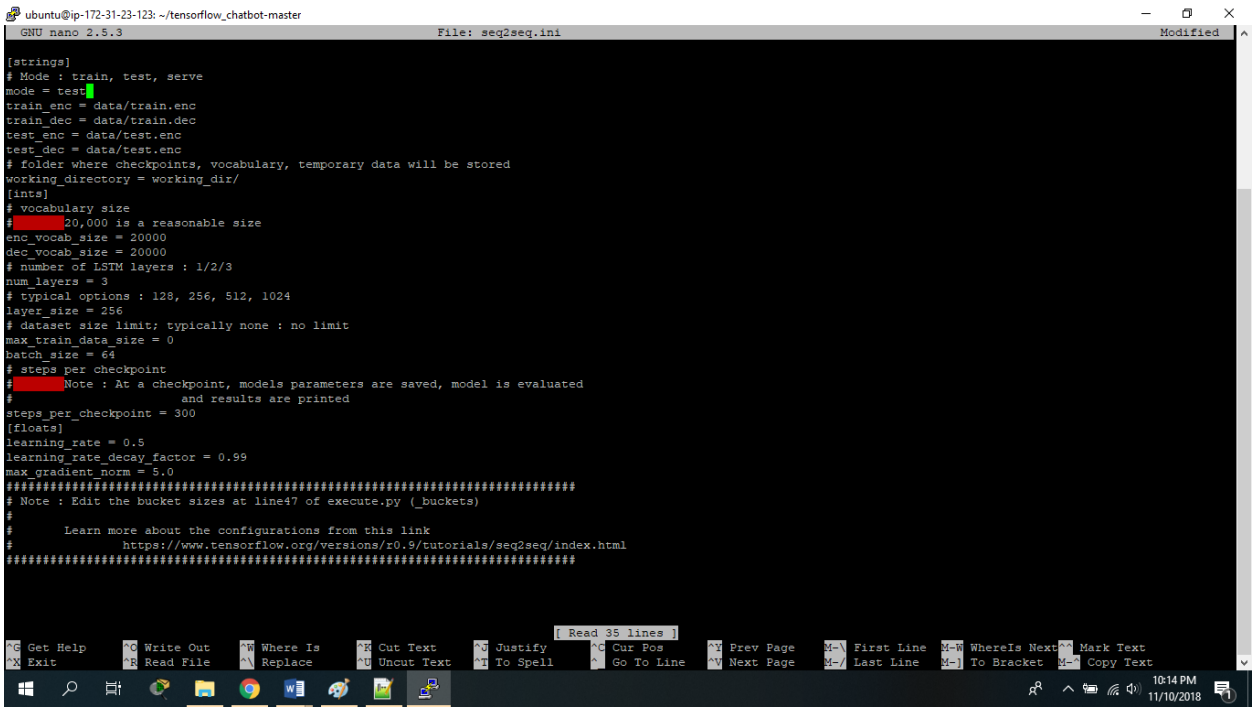


Fig – 5.7 change the mode to **serve**

```
Hi, What is your name
>Kartik
Heyyy Kartik, How are you ?
>I'am good, How are you?
i'am fine too
>Tell me something about politics
i haven't learned anything about that
>Tell me about me
i can learn about you if you want
>
```

Fig – 5.8 have a fun conversation with it

Enabling Security Options

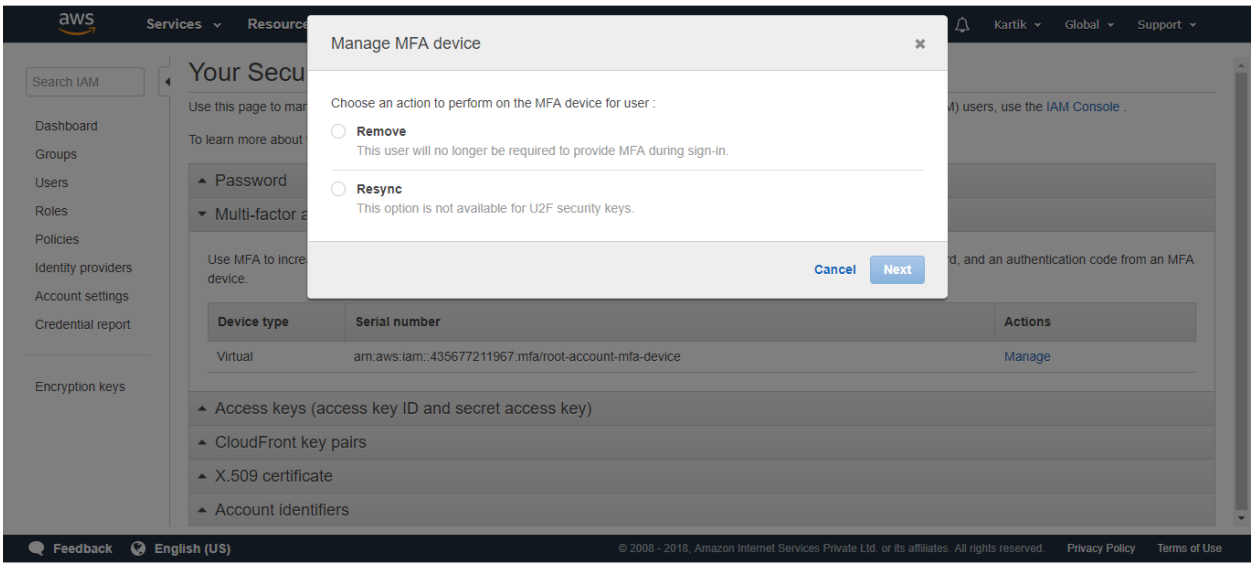


Fig – 6.1 enabling Multi Factor Authentication

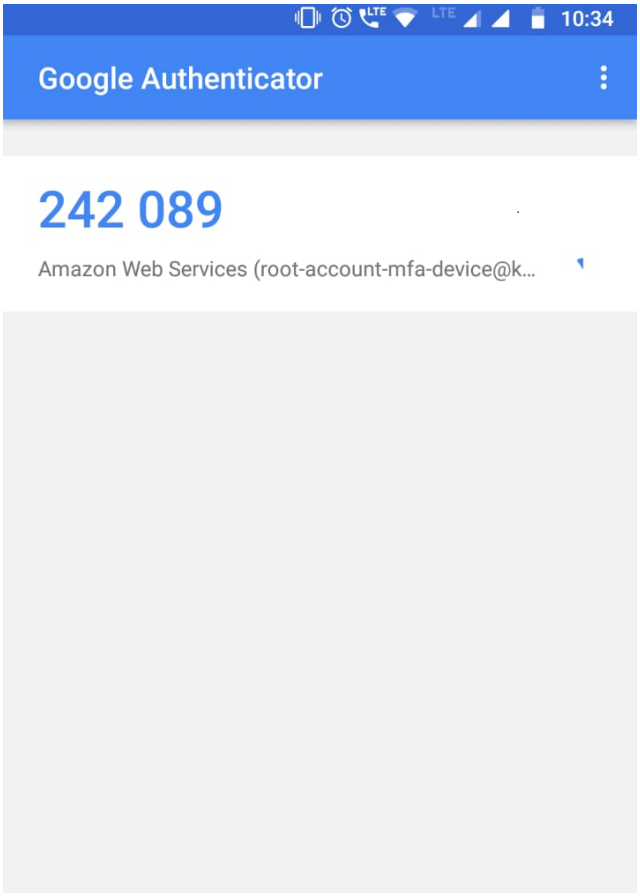


Fig – 6.2 login using Multi Factor Authentication

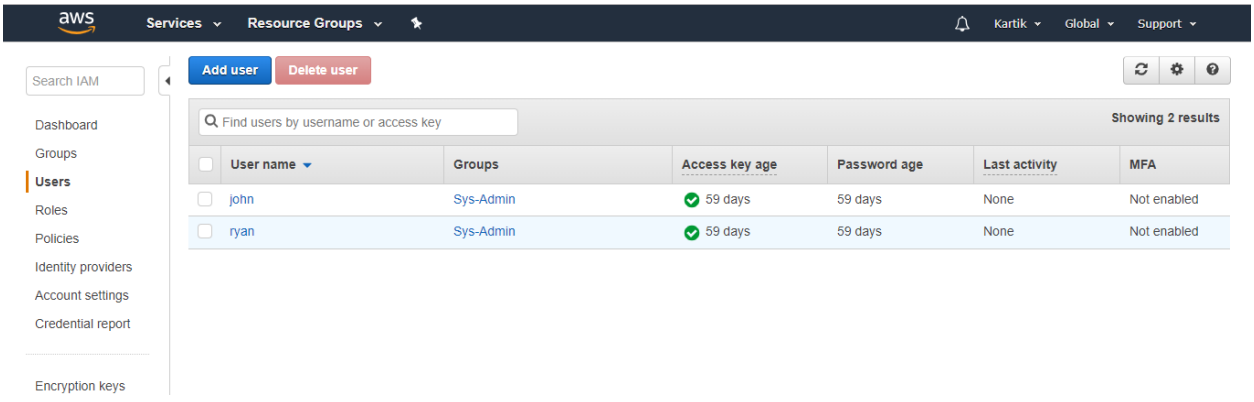


Fig – 6.3 creating users and adding permissions

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

Fig – 6.4 writing JSON file for customized permissions

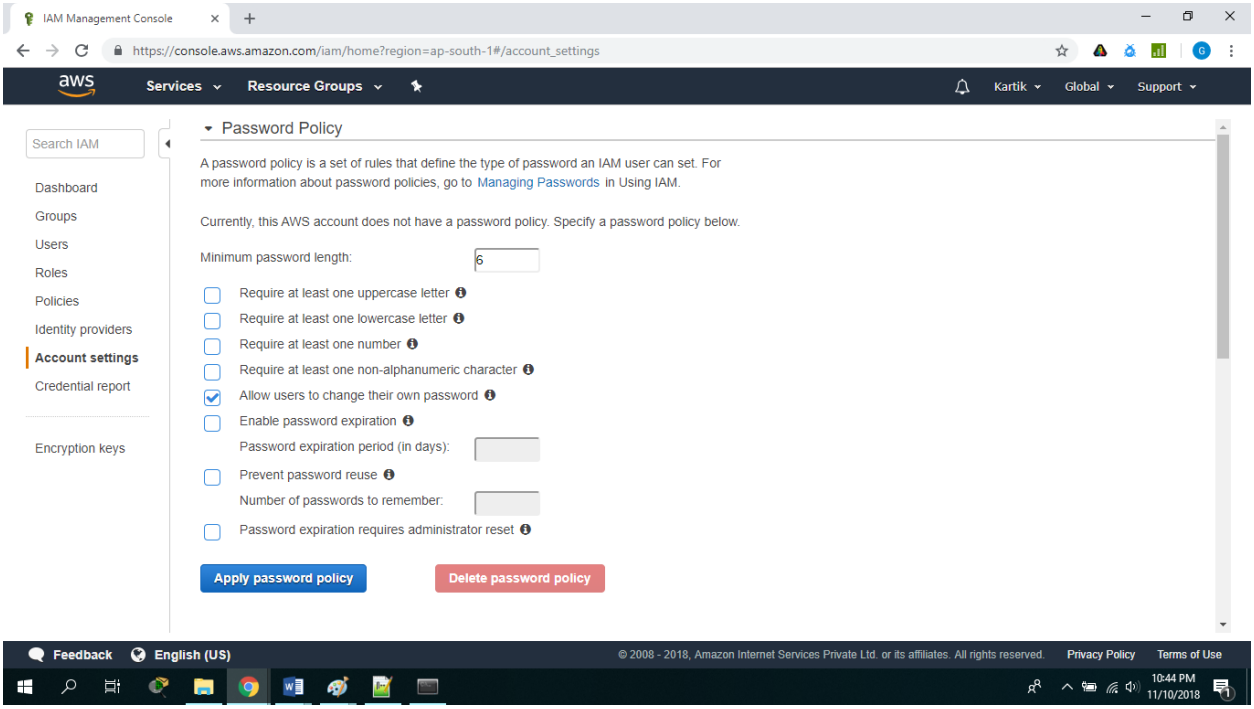


Fig – 6.5 implementing Password Policy according to users or groups

CONCLUSIONS AND FUTURE SCOPE

This project concludes on the above shown experimental results of both the AI Chatbot and Cloud Security implementations done. Though AWS provides various kinds of securities options but it is not enough and hence, the cloud is still not safe.

The future extensions of this project could be the introduction of security features such as Intrusion Prevention Systems for better security over the cloud

According to the 2017 Cost of Data Breach Study: Global Overview (Ponemon Institute, June 2017), the average total cost of a data breach is $3.62 million. The average cost for each lost or stolen record containing sensitive and confidential information is $141. While these costs decreased overall from 2016 to 2017, the numbers remain astronomical, particularly to small businesses who may be unable to recover from data breach liability. No industry is safe from cyberattacks and cyberattacks continue to grow, year after year.

Cloud security must grow and evolve to face these threats and provide a bulwark of defence for the consumers that leverage the efficiencies and advantages cloud services provide. In addition to offsetting the fear highlighted above through good security practices by the cloud security vendor, cloud services can take security one step further. Cloud services can not only secure data within the cloud, but can leverage the transformative cloud industry to secure the endpoint users that use the service.

REFERENCES

- https://www.gigabitmagazine.com/cloud-computing/future-cybersecurity-cloud
- https://www.incapsula.com/blog/top-10-cloud-security-concerns.html
- https://www.skyhighnetworks.com/cloud-security-blog/6-cloud-security-issues-that-businesses-experience/
- https://www.tensorflow.org/
- https://anaconda.org/
- https://pandas.pydata.org/
- http://www.numpy.org/
- https://aws.amazon.com/
- https://aws.amazon.com/about-aws/
- https://searchaws.techtarget.com/definition/Amazon-Web-Services
- https://towardsdatascience.com/seq2seq-model-in-tensorflow-ec0c557e560f
- https://datajobs.com/what-is-data-science
- https://www.expertsystem.com/chatbot/
- https://en.wikipedia.org/wiki/Chatbot
- https://ieeexplore.ieee.org/document/6202020
- https://arxiv.org/ftp/arxiv/papers/1403/1403.5627.pdf
- http://www.engpaper.net/security-in-cloud-computing-research-papers.htm
- https://www.researchgate.net/publication/324757057_Learning_System_Customer_Service_Chatbot
- https://www.researchgate.net/publication/323314352_Case_Study_Building_a_Serverless_Messenger_Chatbot