Dr. D. Y. Patil Unitech Society's

Dr. D. Y. Patil Institute of Technology, Pimpri, Pune-411 018

(Affiliated to Savitribai Phule Pune University, Pune and Approved by AICTE, New Delhi)

http://engg.dypvp.edu.in/

Department of Computer Engineering

A Laboratory Manual

on

Laboratory Practice IV (BE Computer Engineering 2015 Pat)

For A. Y. 2018-19, Term II

About Savitribai Phule Pune University

Savitribai Phule Pune University, one of the premier universities in India, is positioned in the North-western part of Pune city. It occupies an area of about 411 acres. It was established on 10th February, 1949 under the Poona University Act. The university houses 46 academic departments. It is popularly known as the 'Oxford of the East'. It has about 307 recognized research institutes and 612 affiliated colleges offering graduate and under-graduate courses.

The university attracts many foreign students due to its excellent facilities. It offers good accommodation facility. There is a provision of hostel for the students. There is a well-stocked library containing plenty of books regarding various subjects. The university offers different scholarships to the students. The university conducts seminars and conferences for the students.

About Institute (DIT)

Dr. D. Y. Patil Institute of Technology, Pimpri, Pune, was established as Dr. D. Y. Patil Women's College of Engineering in July 1998. The college is converted into Co-education College with a changed name as Dr. D. Y. Patil Institute of Engineering & Technology, Pimpri, Pune from the academic year 2002-2003. The Institute is situated in the vicinity of Pimpri-Chinchwad industrial belt, which is one of the biggest industrial belts in Asia.

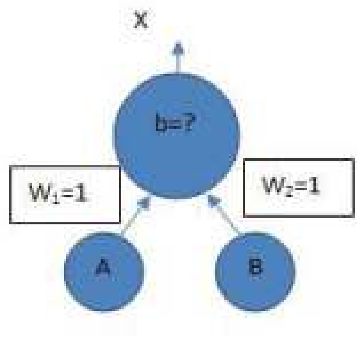
All the eligible programmes have been accredited by the National Board of Accreditation(NBA). The Institute has been awarded ‘A’ grade by the Government of Maharashtra (India). The Institute is an ISO 9001:2008 certified institute.

The alumni of this institute are conscientious in elevating the name of the institute to international levels by securing admission in reputed foreign University to pursue higher studies. DIT was the first institute to be selected by Wipro Mission 10X for Deeper Engagement. The Institute is the first Elite member and received Unified Teaching Learning Kits (UTLK) developed by Wipro Technologies. Also, the first Mission 10X Learning Resource Centre has been established in the institute.

DIT is a consortium member of Indo-US Collaboration in Engineering Education (IUCEE) and International Federation for Engineering Education (IFEES).

Lab Manual

Class: BE COMP ENGG **Term: II**
Academic Year 2018-19
Subject: Laboratory Practice IV
Teaching Scheme **Examination Scheme**
Practical's: 4 hrs/week **Term Work: 50 Marks**
Presentation: 50 Marks

Assignment Number	Assignment Name
1	Implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.
2	Implement Particle swarm optimization for benchmark function (eg. Square, Rosenbrock function). Initialize the population from the Standard Normal Distribution. Evaluate fitness of all particles. Use : c1=c2 = 2 Inertia weight is linearly varied between 0.9 to 0.4. Global best variation
3	Write a program to find the Boolean function to implement following single layer perceptron. Assume all activation functions to be the threshold function which is 1 for all input values greater than zero and 0, otherwise 
4	Implement genetic algorithm for benchmark function (eg. Square, Rosenbrock function etc) Initialize the population from the Standard Normal Distribution. Evaluate the fitness of all its individuals. Then you will do multiple generation of a genetic algorithm. A generation consists of applying selection, crossover, mutation, and replacement. Use: • Tournament selection without replacement with tournament size s • One point crossover with probability Pc • bit-flip mutation with probability Pm • use full replacement strategy
5	Installation and configuration of own Cloud

6	Write a Program to Create, Manage and groups User accounts in own Cloud by Installing Administrative Features.
7	Case study on Amazon EC2 to learn about Amazon EC2,Amazon Elastic Compute Cloud is a central part of Amazon.com's cloud computing platform, Amazon Web Services. How EC2 allows users torrent virtual computers on which to run their own computer applications.
8	Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, forbuilding, deploying and managing applications and services through a global network of Microsoft-managed datacenters. How it work, different services provided by it.
9	Study and implementation of infrastructure as Service using Open Stack.
10	Implementation of Virtualization in Cloud Computing to Learn Virtualization Basics, Benefits of Virtualization in Cloud using Open Source Operating System.
11	Assignment to install and configure Google App Engine.
12	Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store.
13	**Mini-Project 1:** Setup your own cloud for Software as a Service (SaaS) over the existing LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open source technologies **without HDFS**. Implement the basic operations may be like to upload and download file on/from cloud in encrypted form. **OR** **Mini-Project 2:** Setup your own cloud for Software as a Service (SaaS) over the existing LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open source technologies to implement **with HDFS**. Implement the basic operations may be like to divide the file in segments/blocks and upload/ download file on/from cloud in encrypted form. **OR** Any Mini Project based on either Elective III or Elective IV

Assignment No: 01

Aim: Implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.

Objectives:

1. To learn different fuzzy operation on fuzzy set.
2. To learn min-max composition on fuzzy set.

Software Requirements:

Ubuntu 18.04

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth- truth values between "completely true" and "completely false". As its name suggests, it is the logic underlying modes of reasoning which are approximate rather than exact. The importance of fuzzy logic derives from the fact that most modes of human reasoning and especially common sense reasoning are approximate in nature.

The essential characteristics of fuzzy logic as founded by Zader Lotfi are as follows.

- In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
- In fuzzy logic everything is a matter of degree.
- Any logical system can be fuzzified
- In fuzzy logic, knowledge is interpreted as a collection of elastic or, equivalently , fuzzy constraint on a collection of variables
- Inference is viewed as a process of propagation of elastic constraints.

The third statement hence, defines Boolean logic as a subset of Fuzzy logic.

Fuzzy Sets

Fuzzy Set Theory was formalised by Professor Lofti Zadeh at the University of California in 1965. What Zadeh proposed is very much a paradigm shift that first gained acceptance in the Far East and its successful application has ensured its adoption around the world.

A paradigm is a set of rules and regulations which defines boundaries and tells us what to do to be successful in solving problems within these boundaries. For example the use of transistors instead of vacuum tubes is a paradigm shift - likewise the development of Fuzzy Set Theory from conventional bivalent set theory is a paradigm shift.

Bivalent Set Theory can be somewhat limiting if we wish to describe a 'humanistic' problem mathematically. For example, Fig 1 below illustrates bivalent sets to characterise the temperature of a room.

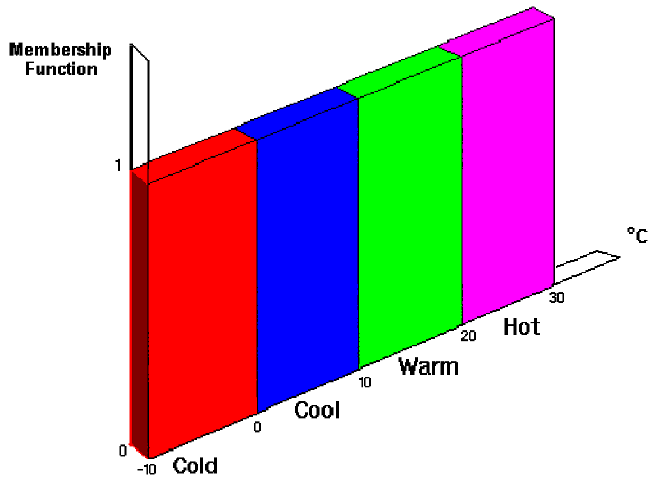


Fig. 1 : Bivalent Sets to Characterize the Temp. of a room.

The most obvious limiting feature of bivalent sets that can be seen clearly from the diagram is that they are mutually exclusive - it is not possible to have membership of more than one set (opinion would widely vary as to whether 50 degrees Fahrenheit is 'cold' or 'cool' hence the expert knowledge we need to define our system is mathematically at odds with the humanistic world). Clearly, it is not accurate to define a transiton from a quantity such as 'warm' to 'hot' by the application of one degree Fahrenheit of heat. In the real world a smooth (unnoticeable) drift from warm to hot would occur. This natural phenomenon can be described more accurately by Fuzzy Set Theory. Fig.2 below shows how fuzzy sets quantifying the same information can describe this natural drift.

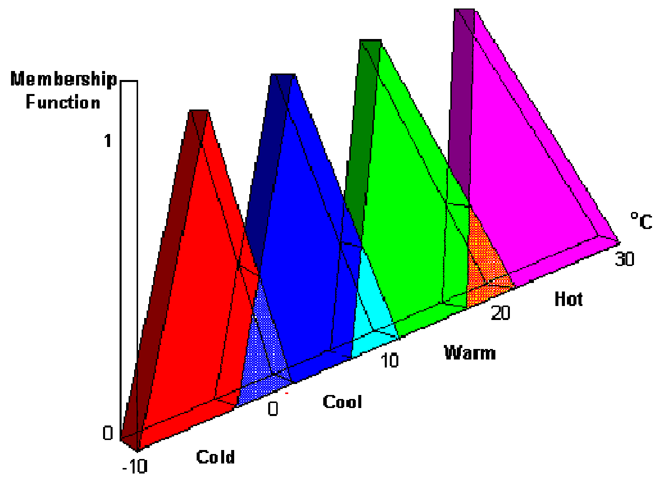


Fig. 2 - Fuzzy Sets to characterize the Temp. of a room.

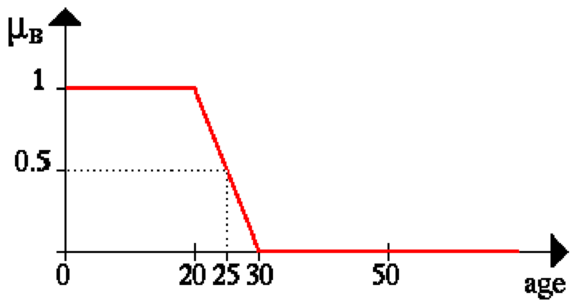
The whole concept can be illustrated with this example. Let's talk about people and "youthness". In this case the set S (the universe of discourse) is the set of people. A fuzzy subset YOUNG is also defined, which answers the question "to what degree is person x young?" To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset YOUNG. The easiest way to do this is with a membership function based on the person's age.

young(x) = { 1, if age(x) <= 20,

(30-age(x))/10, if 20 < age(x) <= 30,

0, if age(x) > 30 }

A graph of this looks like:



Given this definition, here are some example values:

Person Age degree of youth

Johan	10	1.00
Edwin	21	0.90
Parthiban	25	0.50
Arosha	26	0.40
Chin Wei	28	0.20
Rajkumar	83	0.00

So given this definition, we'd say that the degree of truth of the statement "Parthiban is YOUNG" is 0.50.

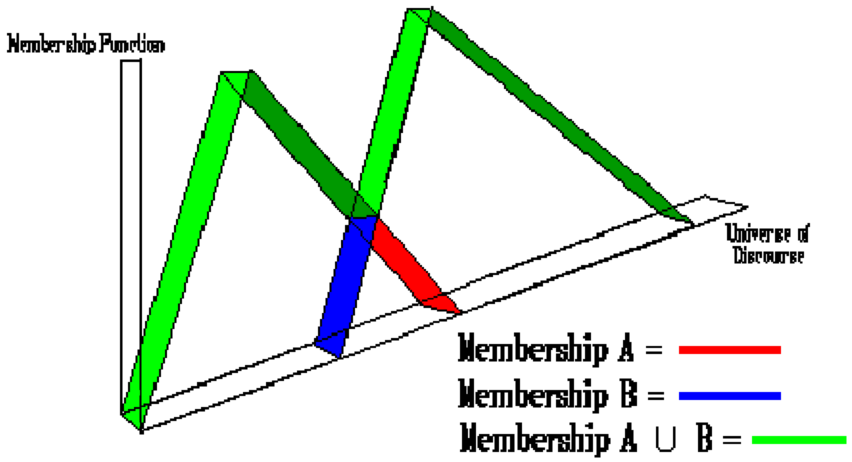
Note: Membership functions almost never have as simple a shape as age(x). They will at least tend to be triangles pointing up, and they can be much more complex than that. Furthermore, membership functions so far is discussed as if they always are based on a single criterion, but this isn't always the case, although it is the most common case. One could, for example, want to have the membership function for YOUNG depend on both a person's age and their height (Arosha's short for his age). This is perfectly legitimate, and occasionally used in practice. It's referred to as a two-dimensional membership function. It's also possible to have even more criteria, or to have the membership function depend on elements from two completely different universes of discourse.

Fuzzy Set Operations.

Union

The membership function of the Union of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the maximum of the two individual membership functions. This is called the *maximum* criterion.

$$\mu_{A \cup B} = \max(\mu_A, \mu_B)$$

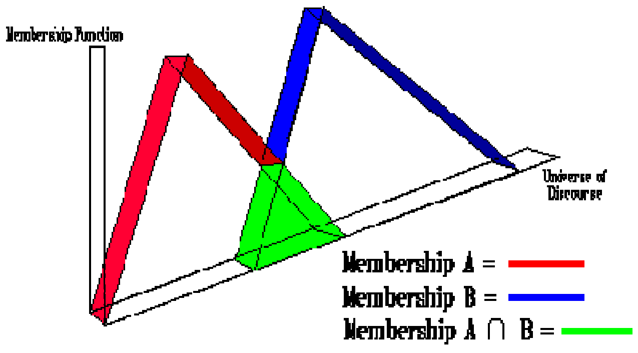


The Union operation in Fuzzy set theory is the equivalent of the **OR** operation in Boolean algebra.

Intersection

The membership function of the Intersection of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the minimum of the two individual membership functions. This is called the *minimum* criterion.

$$\mu_{A \cap B} = \min(\mu_A, \mu_B)$$

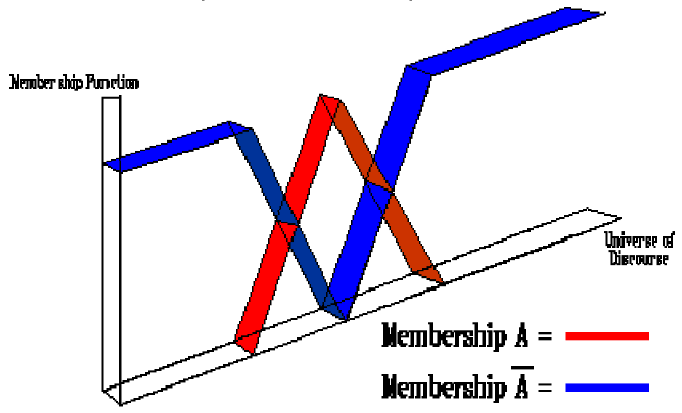


The Intersection operation in Fuzzy set theory is the equivalent of the **AND** operation in Boolean algebra.

Complement

The membership function of the Complement of a Fuzzy set A with membership function μ_A is defined as the negation of the specified membership function. This is caleed the *negation* criterion.

$$\mu_{\overline{A}} = 1 - \mu_A$$



The Complement operation in Fuzzy set theory is the equivalent of the **NOT** operation in Boolean algebra.

The following rules which are common in classical set theory also apply to Fuzzy set theory.

De Morgans law

$$\overline{(A\cap B)} = \overline{A}\cap\overline{B} \;,\; \overline{(A\cup B)} = \overline{A}\cap\overline{B}$$

Associativity

$$(A\cap B)\cap C = A\cap(B\cap C)$$

$$(A\cup B)\cup C = A\cup(B\cup C)$$

Commutativity

$$A\cap B = B\cap A \;,\; A\cup B = B\cup A$$

Distributivity

$$A\cap(B\cup C) = (A\cap B)\cup(A\cap C)$$

$$A\cup(B\cap C) = (A\cup B)\cap(A\cup C)$$

Let A1, A2, ….., An be fuzzy sets in U1, U2, …Un, respectively. The Cartesian product of A1, A2, ….., An is a fuzzy set in the space U1 x U2 x…x Un with the membership function as: µA1 x A2 x…x An (x1, x2,…, xn) = min [µA1 (x1), µA2 (x2), …. µAn (xn)]
So, the Cartesian product of A1, A2, ….., An are donated by A1 x A2 x….. x An.
Cartesian Product: Example Let A = {(3, 0.5), (5, 1), (7, 0.6)} Let B = {(3, 1), (5, 0.6)}
The product is all set of pairs from A and B with the minimum associated memberships Ax B = {[(3, 3), min (0.5, 1)], [(5, 3), min(1, 1)], [(7, 3), min(0.6, 1)], [(3, 5), min(0.5, 0.6)], [(5, 5), min(1, 0.6)], [(7, 5), min(0.6, 0.6)]} = {[(3, 3), 0.5], [(5, 3), 1], [(7, 3), 0.6], [(3, 5), 0.5], [(5, 5), 0.6], [(7, 5), 0.6]}.

Conclusion: Thus we learnt implementation of Union, Intersection, Complement and Difference operations on fuzzy sets. Also created fuzzy relation by Cartesian product of any two fuzzy sets.

Assignment No: 02

Aim: Implement Particle swarm optimization for benchmark function (eg. Square, Rosenbrock function). Initialize the population from the Standard Normal Distribution. Evaluate fitness of all particles.

Use:

 c1=c2 = 2

 Inertia weight is linearly varied between 0.9 to 0.4.

 Global best variation

Objectives:

1. To learn swarm algorithm
2. To learn about optimization algorithm.

Software Requirements:

Ubuntu 18.04

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

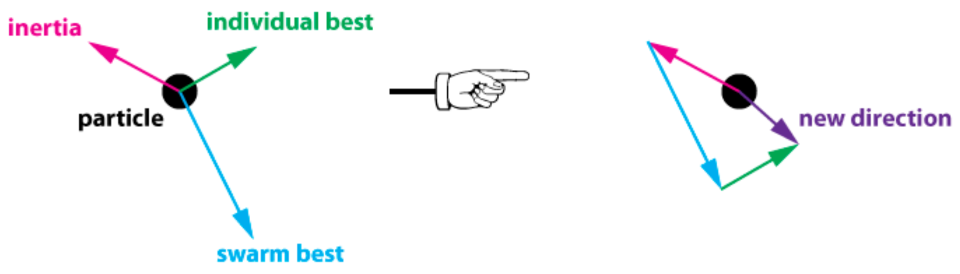
Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

PSO uses a bunch of particles called *the swarm*. These particles are allowed to move around & explore the search-space.

These particles move in a direction which is guided by —

1. The particle's own previous velocity (*Inertia*)
2. Distance from the individual particles' best known position (*Cognitive Force*)
3. Distance from the swarms best known position (*Social Force*)



Particle movement is overpowered by the swam direction

Essentially the particles collectively communicate with each other to converge faster. The swarm doesn't fully explore the search space but potentially finds a better solution.

Interestingly the overall direction of the swarm movement can be changed at any point of time when a particle's individual best is better than the swarm best. This allows a lot of *disorder* and more chances of getting close to the global minima of the cost function

Algorithm

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (a)
present[] = persent[] + v[] (b)

v[] is the particle velocity, persent[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2.

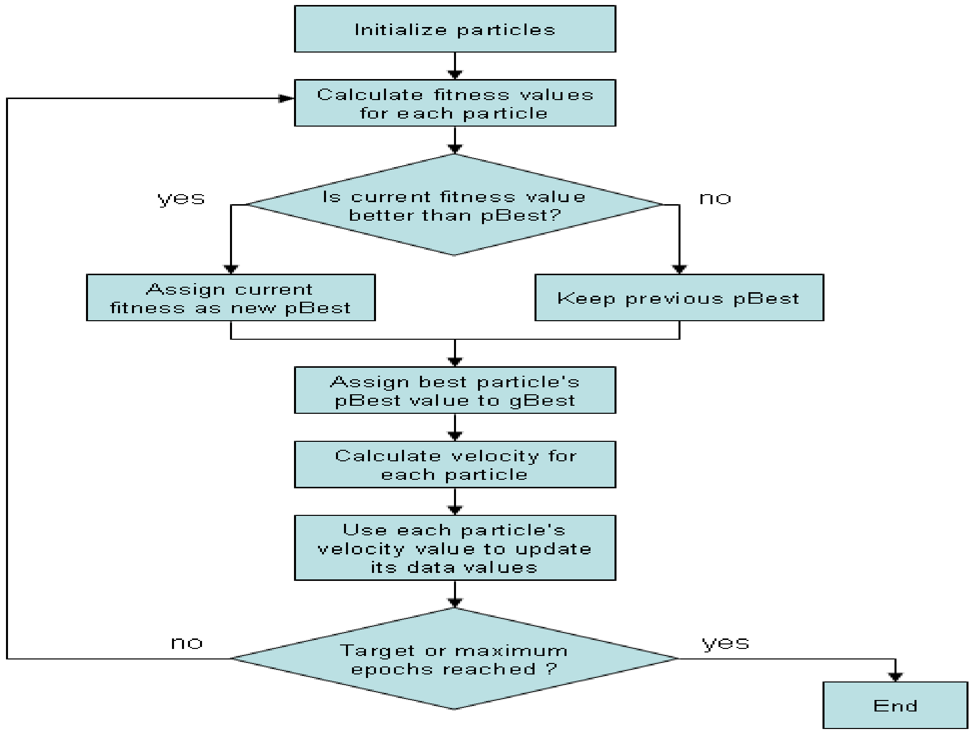
The pseudo code of the procedure is as follows

```
For each particle
   Initialize particle
END

Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in history
         set current value as the new pBest
   End

   Choose the particle with the best fitness value of all the particles as the gBest
   For each particle
      Calculate particle velocity according equation (a)
      Update particle position according equation (b)
   End
While maximum iterations or minimum error criteria is not attained
```

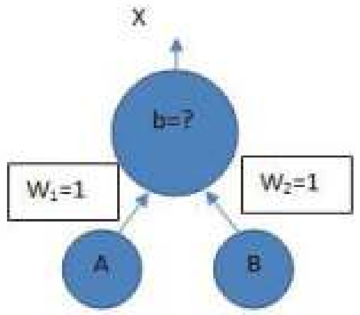
Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.


Initialize particles
Calculate fitness values for each particle
Is current fitness value better than pBest?
yes
no
Assign current fitness as new pBest
Keep previous pBest
Assign best particle's pBest value to gBest
Calculate velocity for each particle
Use each particle's velocity value to update its data values
Target or maximum epochs reached ?
no
yes
End

Conclusion: Thus we learnt implementation of particle swarm optimization for benchmark function.

Assignment No: 03

Aim: Write a program to find the Boolean function to implement following single layer perceptron. Assume all activation functions to be the threshold function which is 1 for all input values greater than zero and 0, otherwise



Objectives:

1. To learn single layer perceptron.
2. To learn Boolean logic implementation using perceptron.

Software Requirements:

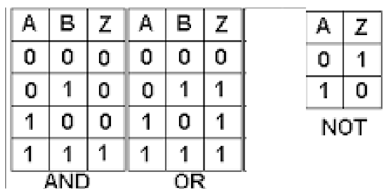
Ubuntu 18.04

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

The most common Boolean functions are AND, OR, NOT. The Boolean logic AND only returns 1 if both inputs are 1 else 0, Boolean logic OR returns 1 for all inputs with 1, and will only return 0 if both input is 0 and lastly logic NOT returns the invert of the input, if the input is 0 it returns 1, if the input is 1 it returns 0. To make it clear the image below shows the truth table for the basic Boolean Function.



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

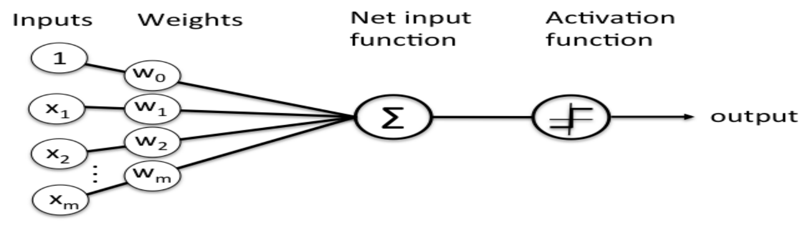
OR

A	Z
0	1
1	0

NOT

The columns A and B are the inputs and column Z is the output. So, for the inputs A = 0, B = 0 the output is Z = 0.

Perceptron



Schematic of Rosenblatt's perceptron.

A perceptron is the basic part of a neural network. A perceptron represents a single neuron on a human's brain, it is composed of the dataset (Xm) , the weights (Wm) and an activation function, that will then produce an output and a bias.

The datasets (inputs) are converted into an ndarray which is then matrix multiplied to another ndarray that holds the weights. Summing up all matrix multipy and adding a bias will create the net input function, the output would then passed into an activation function that would determine if the neuron needs to fire an output or not.

Most common activation function used for classification used is a sigmoid function, which is a great function for classification (Although sigmoid is not the leading activation function for middle layers of neural networks [ehem ReLU / Leaky ReLU] it still is widely used for final classifications.)

The perceptron is simply separating the input into 2 categories, those that cause a fire, and those that don't. It does this by looking at (in the 2-dimensional case):

$w_1I_1 + w_2I_2 < t$

If the LHS is < t, it doesn't fire, otherwise it fires. That is, it is drawing the line:

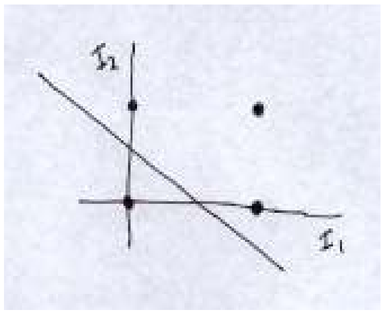
$w_1I_1 + w_2I_2 = t$

and looking at where the input point lies. Points on one side of the line fall into 1 category, points on the other side fall into the other category. And because the weights and thresholds can be anything, this is just *any line* across the 2 dimensional input space.

So what the perceptron is doing is simply drawing a line across the 2-d input space. Inputs to one side of the line are classified into one category, inputs on the other side are classified into another. e.g. the OR perceptron, $w_1=1$, $w_2=1$, $t=0.5$, draws the line:

$I_1 + I_2 = 0.5$

across the input space, thus separating the points (0,1),(1,0),(1,1) from the point (0,0):



As you might imagine, not every set of points can be divided by a line like this. Those that can be, are called *linearly separable*.

In 2 input dimensions, we draw a 1 dimensional line. In n dimensions, we are drawing the (n-1) dimensional *hyperplane*:

$w_1I_1 + .. + w_nI_n = t$

Perceptron Learning Algorithm

```
Algorithm: Perceptron Learning Algorithm
P ← inputs with label 1;
N ← inputs with label 0;
Initialize w randomly;
while !convergence do
    Pick random $x \in P \cup N$ ;
    if $x \in P$ and $w.x < 0$ then
        $w = w + x$ ;
    end
    if $x \in N$ and $w.x \geq 0$ then
        $w = w - x$ ;
    end
end
//the algorithm converges when all the
 inputs are classified correctly
```

We initialize **w** with some random vector. We then iterate over all the examples in the data, (*P* U *N*) both positive and negative examples. Now if an input **x** belongs to *P*, ideally what should the dot product **w.x** be? It would be greater than or equal to 0 because that’s the only thing what our perceptron wants at the end of the day so let's give it that. And if **x** belongs to *N*, the dot product MUST be less than 0. So if you look at the if conditions in the while loop:

```
while !convergence do
    Pick random $\mathbf{x} \in P \cup N$ ;
    if $\mathbf{x} \in P$ and $\mathbf{w.x} < 0$ then
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$ ;
    end
    if $\mathbf{x} \in N$ and $\mathbf{w.x} \geq 0$ then
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$ ;
    end
end
```

Case 1: When **x** belongs to *P* and its dot product **w.x** < 0

Case 2: When **x** belongs to *N* and its dot product **w.x** ≥ 0

Only for these cases, we are updating our randomly initialized **w**. Otherwise, we don’t touch **w** at all because Case 1 and Case 2 are violating the very rule of a perceptron. So we are adding **x** to **w** (ahem vector addition ahem) in Case 1 and subtracting **x** from **w** in Case 2.

Conclusion

Thus we learned implementation of single layer perceptron for AND ,OR and NOT Boolean function.

Assignment No: 04

Aim: Implement genetic algorithm for benchmark function (eg. Square, Rosenbrock function etc) Initialize the population from the Standard Normal Distribution. Evaluate the fitness of all its individuals. Then you will do multiple generation of a genetic algorithm. A generation consists of applying selection, crossover, mutation, and replacement.

Use:

- Tournament selection without replacement with tournament size s
- One point crossover with probability Pc
- bit-flip mutation with probability Pm
- use full replacement strategy

Objectives:

1. To learn Genetic Algorithm.
2. To learn about optimization algorithm.

Software Requirements:

Ubuntu 18.04

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Genetic Algorithms are a class of stochastic, population based optimization algorithms inspired by the biological evolution process using the concepts of "Natural Selection" and "Genetic Inheritance" (Darwin 1859) and originally developed by Holland.

GAs are now used in engineering and business optimization applications, where the search space is large and/or too complex (non-smooth) for analytic treatment. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Five phases are considered in a genetic algorithm.

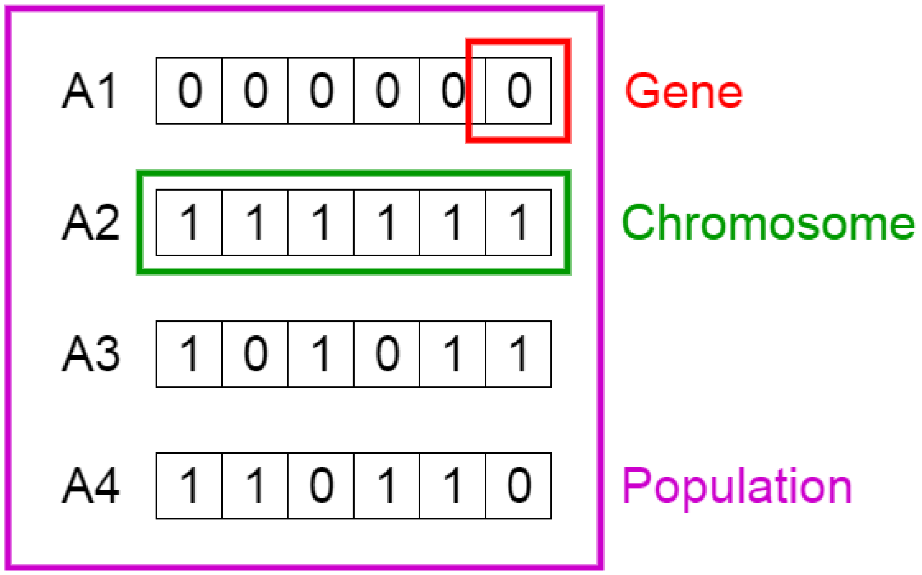
1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

Initial Population

The process begins with a set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.

An individual is characterized by a set of parameters (variables) known as **Genes**. Genes are joined into a string to form a **Chromosome** (solution).

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.



Fitness Function

The **fitness function** determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a **fitness score** to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

Selection

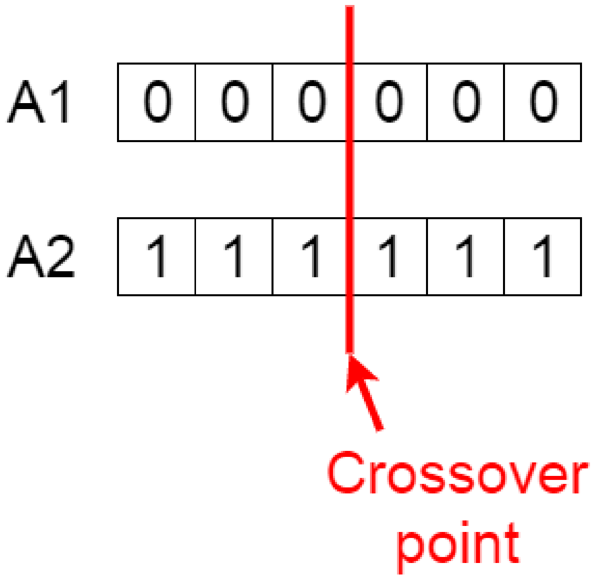
The idea of **selection** phase is to select the fittest individuals and let them pass their genes to the next generation.

Two pairs of individuals (**parents**) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

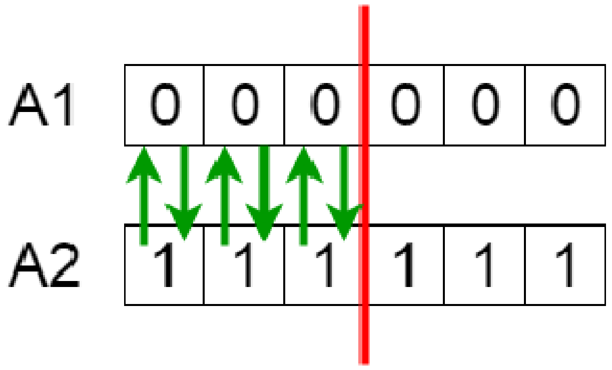
Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.

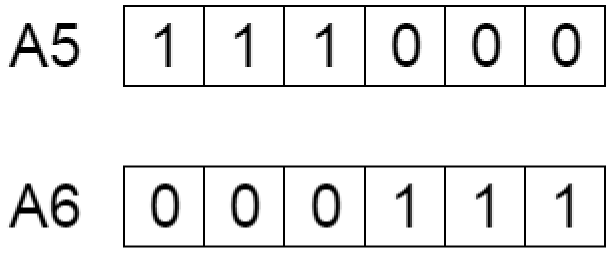
For example, consider the crossover point to be 3 as shown below.



Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

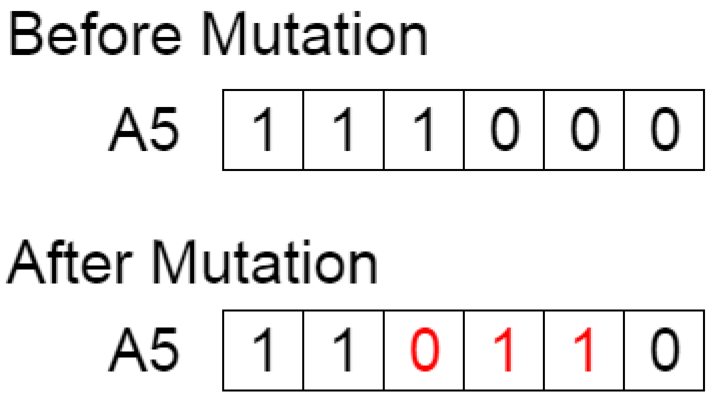


The new offspring are added to the population.



Mutation

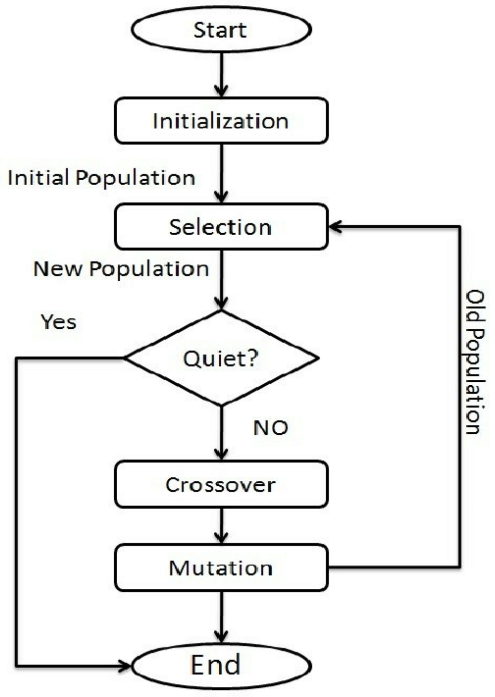
In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.



Mutation occurs to maintain diversity within the population and prevent premature convergence.

Termination

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.



Comments

The population has a fixed size. As new generations are formed, individuals with least fitness die, providing space for new offspring.
The sequence of phases is repeated to produce individuals in each new generation which are better than the previous generation.

GA Software: Python, Weka , Mendal Soft ,Matlab

Conclusion:

Thus we learnt implementation of genetic algorithm for benchmark function.

Assignment No: 05

Aim: Installation and configuration of own Cloud

Objectives:

1. To learn Cloud computing
2. To install and configure own Cloud

Software Requirements:

Ubuntu 18.04

PHP

MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Cloud Computing:

1. Disambiguation—Just What Is Cloud Computing? Cloud computing gets its name as a metaphor for the Internet. Typically, the Internet is represented in network diagrams as a cloud, as shown in Figure 1-1. The cloud icon represents "all that other stuff" that makes the network work. It's kind of like "etc." for the rest of the solution map. It also typically means an area of the diagram or solution that is someone else's concern, so why diagram it all out? It's probably this notion that is most applicable to the cloud computing concept. Cloud computing promises to cut operational and capital costs and, more importantly, let IT departments focus on strategic projects instead of keeping the

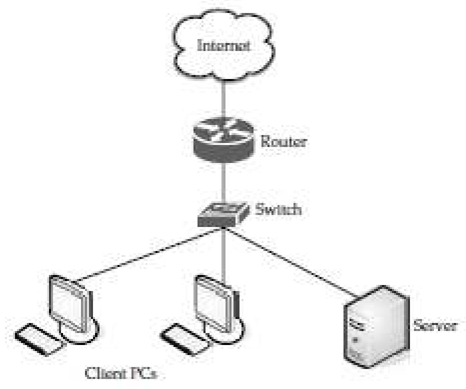


Figure 1: Cloud

datacenter running.

What Works

But there's more going on under the hood than to simply equate cloud computing to the Internet. In essence, cloud computing is a construct that allows you to access applications that actually reside at a location other than your computer or other Internet-connected device; most often, this will be a distant datacenter. There are many benefits to this. For instance, think about the last time you bought Microsoft Word and installed it on your organization's computers. Either you ran around with a CD- or DVD-ROM and installed it on all the computers, or you set up your software distribution servers to automatically install the application on your machines. And every time Microsoft issued a service pack, you had to go around and install that pack, or you had to set up your software distribution servers to distribute it. Oh, and don't forget the cost of all the licenses. Pete down the hall probably uses Word once a month, but his license cost just as much as everyone else's. The beauty of cloud computing, as shown in Figure 1-2, is that another company hosts your application (or suite of applications, for that matter). This means that they handle the costs of servers, they manage the software updates, and—depending on how you craft your contract—you pay less for the service. Don't forget the equipment that you won't need to buy—which will result in fewer capital expenditures—thereby causing the CFO to actually smile when she sees you. By having someone else host the applications, you need not buy the servers nor pay for the electricity to power and cool them. It's also convenient for telecommuters and traveling remote workers, who can simply log in and use their applications wherever they are.

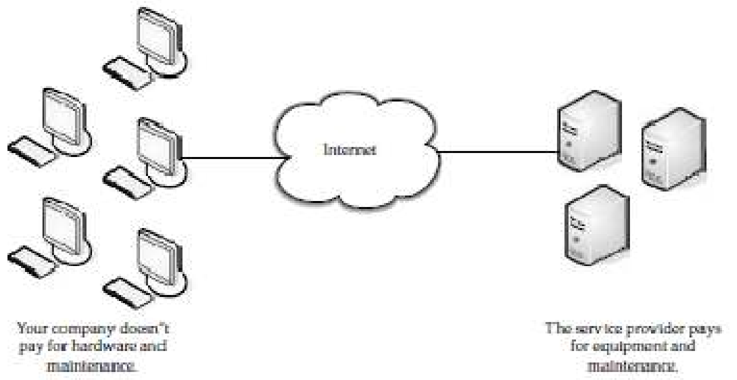

Internet
Your company doesn"t pay for hardware and maintenance.
The service provider pays for equipment and maintenance.

Figure 2 What works in Cloud

Weak Links

So it all sounds great, right? Not so fast. As with everything in IT, there are pros and cons. Cloud computing is not exempt. Let's take a quick look at a few areas of potential trouble. The following illustration shows potential points of failure.

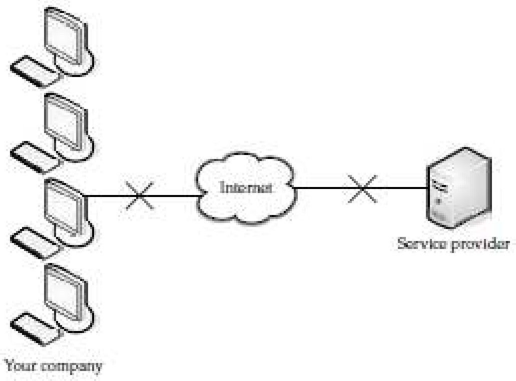


Figure 3 Weak Links of Cloud

While an Internet outage or problems with your Internet service provider (ISP) are rare, you may not be able to access your applications and do your work. Not that everyone sits in one office much anymore, but if you currently have the application on your own local servers, and all those who access it are not remote, you'd be at least somewhat assured that an Internet outage wouldn't affect your application. But it isn't your connection to the Internet that can be prone to outages. What if the site you're accessing has problems? It's happened already. In July 2008, Amazon's S3 cloud storage service went down for the second time that year. A lot of applications were hosted by the company and all those services could not be accessed until techs could fix the problem. Some applications were down for eight hours. Also, there may simply be applications or data that you want located on-site. If you have sensitive or proprietary information, your IT security group may simply mandate that you not store it on someone else's machines. Application Integration Issues You might also find that it's more difficult to integrate your applications if they are geographically dispersed. That is, it is easier to manage and access your data if it is nearby, and not under someone else's control. For instance, if you need two applications to exchange information, it's easier to do if they both reside in the same place. If you have one application in-house and it has to contact another application on the cloud, it becomes far more complicated, and more prone to failure.

Cloud Components

In a simple, topological sense, a cloud computing solution is made up of several elements: clients, the datacenter, and distributed servers. As shown in Figure 1-3, these components make up the three parts of a cloud computing solution. Each element has a purpose and plays a specific role in delivering a functional cloud based application, so

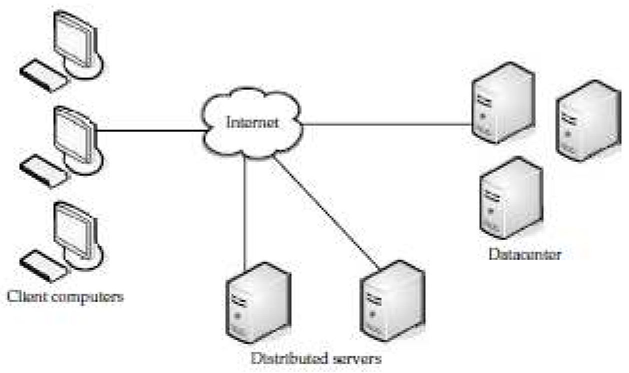


Figure 4 Cloud Components

let's take a closer look. Figure 5 Three components make up a cloud computing solution.

Clients

Clients are, in a cloud computing architecture, the exact same things that they are in a plain, old, everyday local area network (LAN). They are, typically, the computers that just sit on your desk. But they might also be laptops, tablet computers, mobile phones, or PDAs—all big drivers for cloud computing because of their mobility. Anyway, clients are the devices that the end users interact with to manage their information on the cloud. Clients generally fall into three categories:

• **Mobile** Mobile devices include PDAs or smartphones, like a Blackberry, Windows Mobile Smartphone, or an iPhone.

• **Thin** Clients are computers that do not have internal hard drives, but rather let the server do all the work, but then display the information.

• **Thick** This type of client is a regular computer, using a web browser like Firefox or Internet Explorer to connect to the cloud.

Thin clients are becoming an increasingly popular solution, because of their price and effect on the environment. Some benefits to using thin clients include

• **Lower hardware costs** Thin clients are cheaper than thick clients because they do not contain as much hardware. They also last longer before they need to be upgraded or become obsolete.

• **Lower IT costs** Thin clients are managed at the server and there are fewer points of failure.

• **Security** Since the processing takes place on the server and there is no hard drive, there's less chance of malware invading the device. Also, since thin clients don't work without a server, there's less chance of them being physically stolen.

• **Data security** Since data is stored on the server, there's less chance for data to be lost if the client computer crashes or is stolen.

• **Less power consumption** Thin clients consume less power than thick clients. This means you'll pay less to power them, and you'll also pay less to air-condition the office.

• **Ease of repair or replacement** If a thin client dies, it's easy to replace. The box is simply swapped out and the user's desktop returns exactly as it was before the failure.

• **Less noise** Without a spinning hard drive, less heat is generated and quieter fans can be used on the thin client.

Datacenter

The *datacenter* is the collection of servers where the application to which you subscribe is housed. It could be a large room in the basement of your building or a room full of servers on the other side of the world that you access via the Internet. A growing trend in the IT world is virtualizing servers. That is, software can be installed allowing multiple instances of virtual servers to be used. In this way, you can have half a dozen virtual servers running on one physical server.

Distributed Servers

But the servers don't all have to be housed in the same location. Often, servers are in geographically disparate locations. But to you, the cloud subscriber, these servers act as if they're humming away right next to each other. This gives the service provider more flexibility in options and security. For instance, Amazon has their cloud solution in servers all over the world. If something were to happen at one site, causing a failure, the service would still be accessed through another site. Also, if the cloud needs more hardware, they need not throw more servers in the safe room—they can add them at another site and simply make it part of the cloud.

Steps for Installation and configuration of owncloud:

Introduction

ownCloud is an open-source file sharing server and collaboration platform that can store your personal content, like documents and pictures, in a centralized location. This allows you to take control of your content and security by not relying on third-party content hosting services like Dropbox.

In this manual, we will install and configure an ownCloud instance on an Ubuntu 18.04 server.

Prerequisites

In order to complete the steps in this guide, you will need the following:

- **A sudo user and firewall on your server**: You can create a user with sudo privileges and set up a basic firewall by following the Ubuntu 18.04 initial server setup guide.
- **A LAMP stack**: ownCloud requires a web server, a database, and PHP to function properly. Setting up a LAMP stack (Linux, Apache, MySQL, and PHP) server fulfills all of these requirements. Follow this guide to install and configure this software.
- **An SSL certificate**: How you set this up depends on whether or not you have a domain name that resolves to your server.

- **If you have a domain name...** the easiest way to secure your site is with Let's Encrypt, which provides free, trusted certificates. Follow the Let's Encrypt guide for Apache to set this up.
- **If you do not have a domain...** and you are just using this configuration for testing or personal use, you can use a self-signed certificate instead. This provides the same type of encryption, but without the domain validation. Follow the self-signed SSL guide for Apache to get set up.

Step 1 – Installing ownCloud

The ownCloud server package does not exist within the default repositories for Ubuntu. However, ownCloud maintains a dedicated repository for the distribution that we can add to our server.

To begin, download their release key using the curl command and import it with the apt-key utility with the add command:

```
Curl https://download.owncloud.org/download/repositories/10.0/Ubuntu_18.04/Release.key | sudo apt-key add -
```

The 'Release.key' file contains a PGP (Pretty Good Privacy) public key which apt will use to verify that the ownCloud package is authentic.

In addition to importing the key, create a file called owncloud.list in the sources.list.d directory for apt. The file will contain the address to the ownCloud repository.

```
echo 'deb http://download.owncloud.org/download/repositories/10.0/Ubuntu_18.04/ /' | sudo tee /etc/apt/sources.list.d/owncloud.list
```

Now, we can use the package manager to find and install ownCloud. Along with the main package, we will also install a few additional PHP libraries that ownCloud uses to add extra functionality. Update your local package index and install everything by typing:

```
sudo apt update
sudo apt install php-bz2 php-curl php-gd php-imagick php-intl php-mbstring php-xml php-zip owncloud-files
```

Everything we need is now installed on the server, so next we can finish the configuration so we can begin using the service.

Step 2 — Adjusting the Document Root

The ownCloud package we installed copies the web files to /var/www/owncloud on the server. Currently, the Apache virtual host configuration is set up to serve files out of a different directory. We need to change the DocumentRoot setting in our configuration to point to the new directory.

You find which virtual host files reference your domain name or IP address using the apache2ctl utility with the DUMP_VHOSTS option. Filter the output by your server's domain name or IP address to find which files you need to edit in the next few commands:

```
sudo apache2ctl -t -D DUMP_VHOSTS | grep server_domain_or_IP
```

The output will probably look something like this:

Output

```
*:443                  server_domain_or_IP (/etc/apache2/sites-enabled/server_domain_or_IP-le-ssl.conf:2)
         port 80 namevhost server_domain_or_IP (/etc/apache2/sites-enabled/server_domain_or_IP.conf:1)
```

In the parentheses, you can see each of the files that reference the domain name or IP address we'll use to access ownCloud. These are the files you'll need to edit.

For each match, open the file in a text editor with sudo privileges:

```
sudo nano /etc/apache2/sites-enabled/server_domain_or_IP.conf
```

Inside, search for the DocumentRoot directive. Change the line so that it points to the /var/www/owncloud directory:

Example DocumentRoot edit

```
<VirtualHost *:80>
   . . .
   DocumentRoot /var/www/owncloud
   . . .
</VirtualHost>
```

Save and close the file when you are finished. Complete this process for each of the files that referenced your domain name (or IP address if you did not configure a domain for your server).

When you are finished, check the syntax of your Apache files to make sure there were no detectable typos in your configuration:

```
sudo apache2ctl configtest
```

Output

```
Syntax OK
```

Depending on your configuration, you may see a warning about setting ServerName globally. As long as the output ends with Syntax OK, you can ignore that warning. If you see additional errors, go back and check the files you just edited for mistakes.

If your syntax check passed, reload the Apache service to activate the new changes:

```
sudo systemctl reload apache2
```

Apache should now know how to server your ownCloud files.

Step 3 – Configuring the MySQL Database

Before we move on to the web configuration, we need to set up the database. During the web-based configuration process, we will need to provide an database name, a database username, and a database password so that ownCloud can connect and manage its information within MySQL.

Begin by logging into your database with the MySQL administrative account:

```
sudo mysql
```

If you set up password authentication for MySQL root account, you may have to use this syntax instead:

```
mysql -u root -p
```

Create a dedicated database for ownCloud to use. We will name the database owncloud for clarity:

```
CREATE DATABASE owncloud;
```

Note: Every MySQL statement must end with a semi-colon (;). Be sure to check that this is present if you are experiencing an issue.

Next, create a separate MySQL user account to manage the newly created database. Creating one-function databases and accounts is a good idea from a management and security standpoint. As with the naming of the database, choose a username that you prefer. We elected to go with the name owncloud in this guide.

```
GRANT ALL ON owncloud.* to 'owncloud'@'localhost' IDENTIFIED BY
'owncloud_database_password';
```

Warning: Be sure to put an actual password where the command states: owncloud_database_password

With the user assigned access to the database, perform the flush privileges operation to ensure that the running instance of MySQL knows about the recent privilege assignment:

```
FLUSH PRIVILEGES;
```

You can now exit the MySQL session by typing:

```
exit
```

With the ownCloud server installed and the database set up, we are ready to turn our attention to configuring the ownCloud application.

Step 4 – Configuring ownCloud

To access the ownCloud web interface, open a web browser and navigate to the following address:

https://server_domain_or_IP

Note: If you are using a self-signed SSL certificate, you will likely be presented with a warning because the certificate is not signed by one of your browser's trusted authorities. This is expected and normal. Click the appropriate button or link to proceed to the ownCloud admin page.

You should see the ownCloud web configuration page in your browser.

Create an admin account by choosing a username and a password. For security purposes it is not recommended to use something like "admin" for the username:

Next, leave the **Data folder** setting as-is and scroll down to the database configuration section.

Fill out the details of the database name, database username, and database password you created in the previous section. If you used the settings from this guide, both the database name and username will be owncloud. Leave the database host as localhost:

Click the **Finish setup** button to finish configuring ownCloud using the information you've provided. You will be taken to a login screen where you can sign in using your new account:

On your first login, a screen will appear where you can download applications to sync your files on various devices. You can download and configure these now or do it at a later time. When you are finished, click the **x** in the top-right corner of the splash screen to access the main interface:

Here, you can create or upload files to your personal cloud.

Conclusion

ownCloud can replicate the capabilities of popular third-party cloud storage services. Content can be shared between users or externally with public URLs. The advantage of ownCloud is that the information is stored in a place that you control and manage without a third party.

Assignment No: 06

Aim: Write a Program to Create, Manage and groups User accounts in ownCloud by Installing Administrative Features.

Objectives:

1. To learn Cloud computing administration
2. To install and configure ownCloud administrative features

Software Requirements:

Ubuntu 18.04

PHP

MySQL

Hardware Requirements:

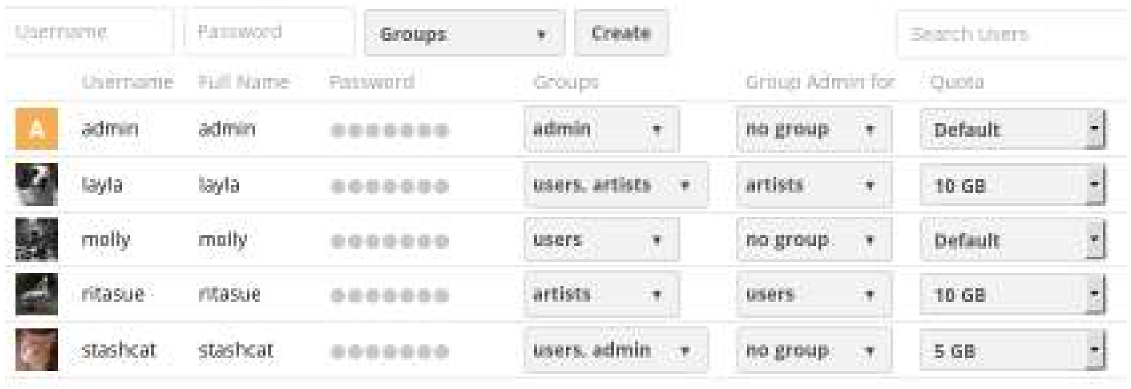
Pentium IV system with latest configuration

Theory:

On the User management page of your ownCloud Web UI you can:

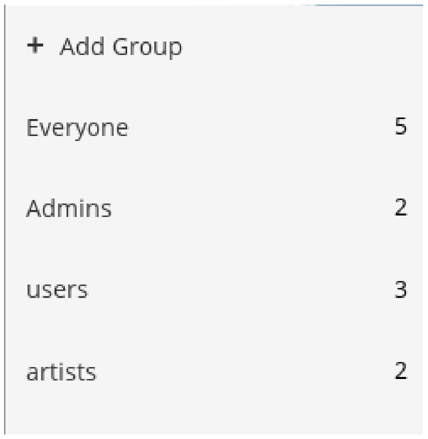
- Create new users
- View all of your users in a single scrolling window
- Filter users by group
- See what groups they belong to
- Edit their full names and passwords
- See their data storage locations
- View and set quotas
- Create and edit their email addresses
- Send an automatic email notification to new users
- Delete them with a single click

The default view displays basic information about your users.



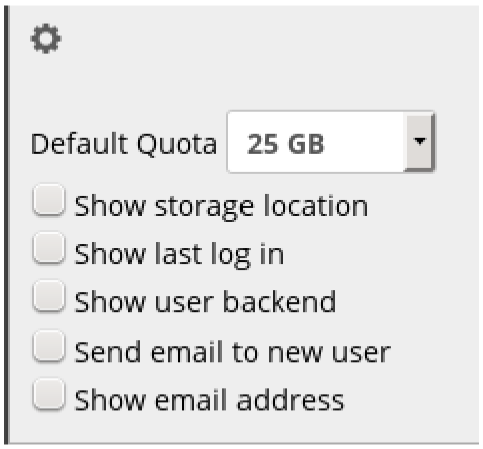
	Username	Full Name	Password	Groups	Group Admin for	Quota
A	admin	admin	●●●●●●●	admin	no group	Default
	layla	layla	●●●●●●●	users, artists	artists	10 GB
	molly	molly	●●●●●●●	users	no group	Default
	ritasue	ritasue	●●●●●●●	artists	users	10 GB
	stashcat	stashcat	●●●●●●●	users, admin	no group	5 GB

The Group filters on the left sidebar lets you quickly filter users by their group memberships, and create new groups.



Everyone	5
Admins	2
users	3
artists	2

Click the gear icon on the lower left sidebar to set a default storage quota, and to display additional fields: **Show storage location, Show last log in, Show user backend, Send email to new users,** and **Show email address**.



Default Quota 25 GB

- [ ] Show storage location
- [ ] Show last log in
- [ ] Show user backend
- [ ] Send email to new user
- [ ] Show email address

User accounts have the following properties:

Login Name (Username)

The unique ID of an ownCloud user, and it cannot be changed.

Full Name

The user's display name that appears on file shares, the ownCloud Web interface, and emails. Admins and users may change the Full Name anytime. If the Full Name is not set it defaults to the login name.

Password

The admin sets the new user's first password. Both the user and the admin can change the user's password at anytime.

Groups

You may create groups, and assign group memberships to users. By default new users are not assigned to any groups.

Group Admin

Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.

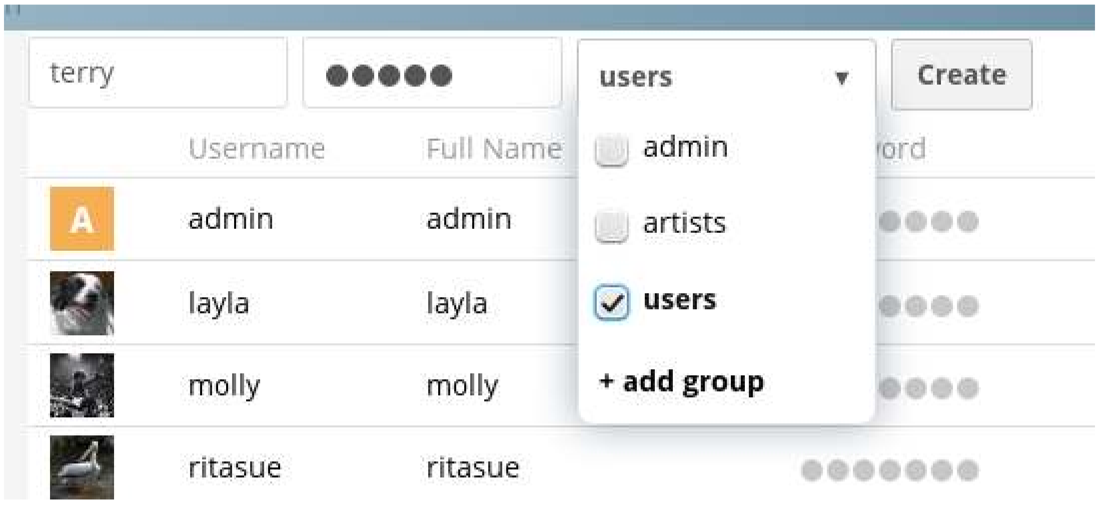
Quota

The maximum disk space assigned to each user. Any user that exceeds the quota cannot upload or sync data. You have the the option to include external storage in user quotas.

Creating a New User

To create a user account:

- Enter the new user's **Login Name** and their initial **Password**
- Optionally, assign **Groups** memberships
- Click the **Create** button



	Username	Full Name	ord
A	admin	admin	●●●●
	layla	layla	●●●●
	molly	molly	●●●●
	ritasue	ritasue	●●●●●●●

Login names may contain letters (a-z, A-Z), numbers (0-9), dashes (-), underscores (_), periods (.) and at signs (@). After creating the user, you may fill in their **Full Name** if it is different than the login name, or leave it for the user to complete.

If you have checked **Send email to new user** in the control panel on the lower left sidebar, you may also enter the new user's email address, and ownCloud will automatically send them a notification with their new login information. You may edit this email using the email template editor on your Admin page.

Reset a User's Password

You cannot recover a user's password, but you can set a new one:

- Hover your cursor over the user's **Password** field
- Click on the **pencil icon**
- Enter the user's new password in the password field, and remember to provide the user with their password

If you have encryption enabled, there are special considerations for user password resets.

Renaming a User

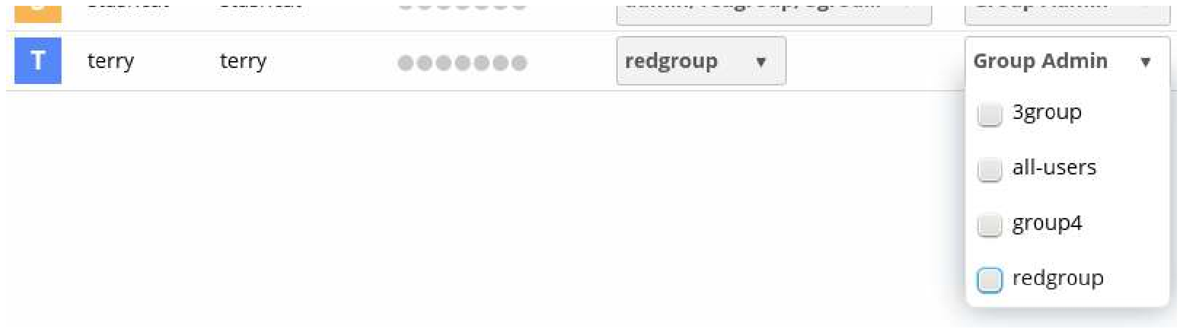
Each ownCloud user has two names: a unique **Login Name** used for authentication, and a **Full Name**, which is their display name. You can edit the display name of a user, but you cannot change the login name of any user.

To set or change a user's display name:

- Hover your cursor over the user's **Full Name** field
- Click on the **Pencil icon**
- Enter the user's new display name

Granting Administrator Privileges to a User

ownCloud has two types of administrators: **Super Administrators** and **Group Administrators**. Group administrators have the rights to create, edit and delete users in their assigned groups. Group administrators cannot access system settings, or add or modify users in the groups that they are not **Group Administrators** for. Use the dropdown menus in the **Group Admin** column to assign group admin privileges.



Super Administrators have full rights on your ownCloud server, and can access and modify all settings. To assign the **Super Administrators** role to a user, simply add them to the admin group.

Managing Groups

You can assign new users to groups when you create them, and create new groups when you create new users. You may also use the **Add Group** button at the top of the left pane to create new groups. New group members will immediately have access to file shares that belong to their new groups.

Setting Storage Quotas

Click the gear on the lower left pane to set a default storage quota. This is automatically applied to new users. You may assign a different quota to any user by selecting from the **Quota** dropdown, selecting either a preset value or entering a custom value. When you create custom quotas, use the normal abbreviations for your storage values such as 500 MB, 5 GB, 5 TB, and so on.

You now have a configurable option in config.php that controls whether external storage is counted against user's quotas. This is still experimental, and may not work as expected. The default is to not count external storage as part of user storage quotas. If you prefer to include it, then change the default false to true.:

```
'quota_include_external_storage' => false,
```

Metadata (such as thumbnails, temporary files, and encryption keys) takes up about 10% of disk space, but is not counted against user quotas. Users can check their used and available space on their Personal pages. Only files that originate with users count against their quotas, and not files shared with them that originate from other users. For example, if you upload files to a different user's share, those files count against your quota. If you re-share a file that another user shared with you, that file does not count against your quota, but the originating user's.

Encrypted files are a little larger than unencrypted files; the unencrypted size is calculated against the user's quota.

Deleted files that are still in the trash bin do not count against quotas. The trash bin is set at 50% of quota. Deleted file aging is set at 30 days. When deleted files exceed 50% of quota then the oldest files are removed until the total is below 50%.

When version control is enabled, the older file versions are not counted against quotas.

When a user creates a public share via URL, and allows uploads, any uploaded files count against that user's quota.

Deleting users

Deleting a user is easy: hover your cursor over their name on the **Users** page until a trashcan icon appears at the far right. Click the trashcan, and they're gone. You'll see an undo button at the top of the page, which remains until you refresh the page. When the undo button is gone you cannot recover the deleted user.

All of the files owned by the user are deleted as well, including all files they have shared. If you need to preserve the user's files and shares, you must first download them from your ownCloud Files page, which compresses them into a zip file, or use a sync client to copy them to your local computer.

Conclusion

As a cloud service provider, we are managing user accounts and groups by using Administrative features.

Assignment No: 07

Aim: Case study on Amazon EC2 to learn about Amazon EC2,Amazon Elastic Compute Cloud is a central part of Amazon.com's cloud computing platform, Amazon Web Services. How EC2 allows users torrent virtual computers on which to run their own computer applications.

Objectives:

1. To learn Amazon Web Services.
2. To case study the Amazon EC2.

Software Requirements:

Ubuntu 18.04

PHP

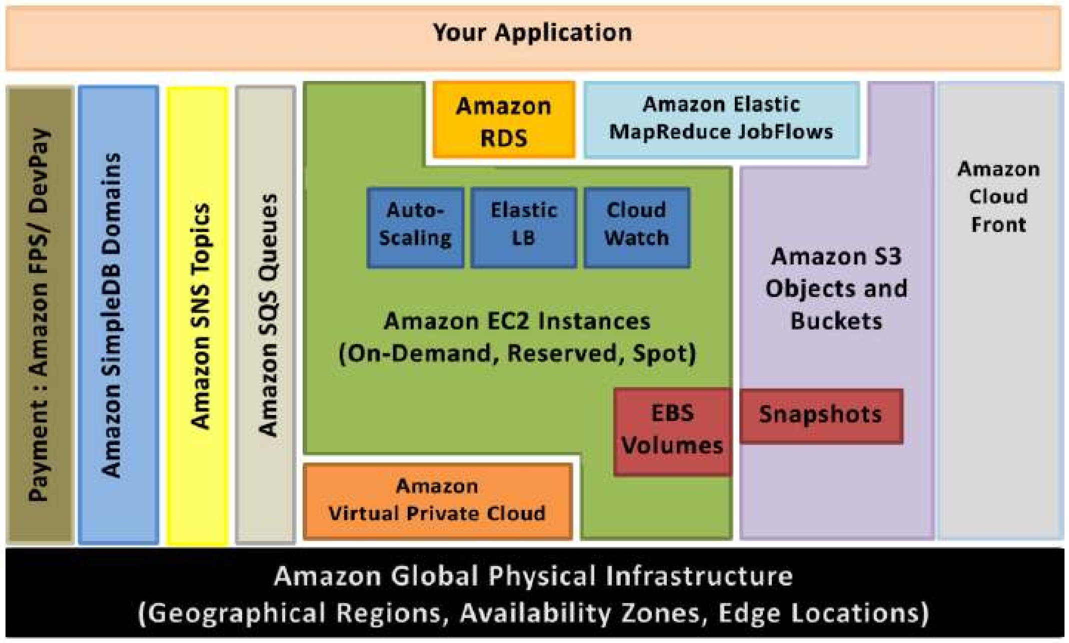
MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Amazon Elastic Compute Cloud (EC2)


Your Application
Payment : Amazon FPS/ DevPay
Amazon SimpleDB Domains
Amazon SNS Topics
Amazon SQS Queues
Amazon RDS
Amazon Elastic MapReduce JobFlows
Auto-Scaling
Elastic LB
Cloud Watch
Amazon EC2 Instances (On-Demand, Reserved, Spot)
EBS Volumes
Amazon S3 Objects and Buckets
Snapshots
Amazon Cloud Front
Amazon Virtual Private Cloud
Amazon Global Physical Infrastructure (Geographical Regions, Availability Zones, Edge Locations)

Elastic IP addresses allow you to allocate a static IP address and programmatically assign it to an instance. You can enable monitoring on an Amazon EC2 instance using Amazon CloudWatch2 in order to gain visibility into resource utilization, operational performance, and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic). You can create Auto-scaling Group using the Auto-scaling feature3 to automatically scale your capacity on certain conditions based on metric that Amazon CloudWatch collects. You can also distribute incoming traffic by creating an elastic load balancer using the Elastic Load Balancing4 service. Amazon Elastic Block Storage (EBS)5 volumes provide network-attached persistent storage to Amazon EC2 instances. Point-in-time consistent snapshots of EBS volumes can be created and stored on Amazon Simple Storage Service (Amazon S3)6.
Amazon S3 is highly durable and distributed data store. With a simple web services interface, you can store and retrieve large amounts of data as objects in buckets (containers) at any time, from anywhere on the web using standard HTTP verbs. Copies of objects can be distributed and cached at 14 edge locations around the world by creating a distribution using Amazon CloudFront7 service – a web service for content delivery (static or streaming content). Amazon SimpleDB8 is a web service that provides the core functionality of a database- real-time lookup and simple querying of structured data – without the operational complexity. You can organize the dataset into domains and can run queries across all of the

data stored in a particular domain. Domains are collections of items that are described by attribute-value pairs.

Amazon Relational Database Service9 (Amazon RDS) provides an easy way to setup, operate and scale a relational database in the cloud. You can launch a DB Instance and get access to a full-featured MySQL database and not worry about common database administration tasks like backups, patch management etc.

Amazon Simple Queue Service (Amazon SQS)10 is a reliable, highly scalable, hosted distributed queue for storing messages as they travel between computers and application components.

Amazon Simple Notifications Service (Amazon SNS) provides a simple way to notify applications or people from the cloud by creating Topics and using a publish-subscribe protocol.

Amazon Elastic MapReduce provides a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) and allows you to create customized JobFlows. JobFlow is a sequence of MapReduce steps.

Amazon Virtual Private Cloud (Amazon VPC) allows you to extend your corporate network into a private cloud contained within AWS. Amazon VPC uses IPSec tunnel mode that enables you to create a secure connection between a gateway in your data center and a gateway in AWS.

Amazon Route53 is a highly scalable DNS service that allows you manage your DNS records by creating a HostedZone for every domain you would like to manage.

AWS Identity and Access Management (IAM) enable you to create multiple Users with unique security credentials and manage the permissions for each of these Users within your AWS Account. IAM is natively integrated into AWS Services. No service APIs have changed to support IAM, and exiting applications and tools built on top of the AWS service APIs will continue to work when using IAM.

AWS also offers various payment and billing services that leverages Amazon's payment infrastructure. All AWS infrastructure services offer utility-style pricing that require no long-term commitments or contracts. For example, you pay by the hour for Amazon EC2 instance usage and pay by the gigabyte for storage and data transfer in the case of Amazon S3. More information about each of these services and their pay-as-you-go pricing is available on the

AWS Website.

Note that using the AWS cloud doesn't require sacrificing the flexibility and control you've grown accustomed to:

You are free to use the programming model, language, or operating system (Windows, OpenSolaris or any flavor of Linux) of your choice.

You are free to pick and choose the AWS products that best satisfy your requirements—you can use any of the services individually or in any combination.

Because AWS provides resizable (storage, bandwidth and computing) resources, you are free to consume as much or as little and only pay for what you consume.

You are free to use the system management tools you've used in the past and extend your datacenter into the cloud.

Conclusion

Performed case study of Amazon web services: Amazon EC2.

Assignment No: 08

Aim: Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, forbuilding, deploying and managing applications and services through a global network of Microsoft-managed datacenters. How it work, different services provided by it.

Objectives:

1. To learn Microsoft Azure Cloud computing platform.
2. To case study the Microsoft Azure cloud services.

Software Requirements:

Ubuntu 18.04

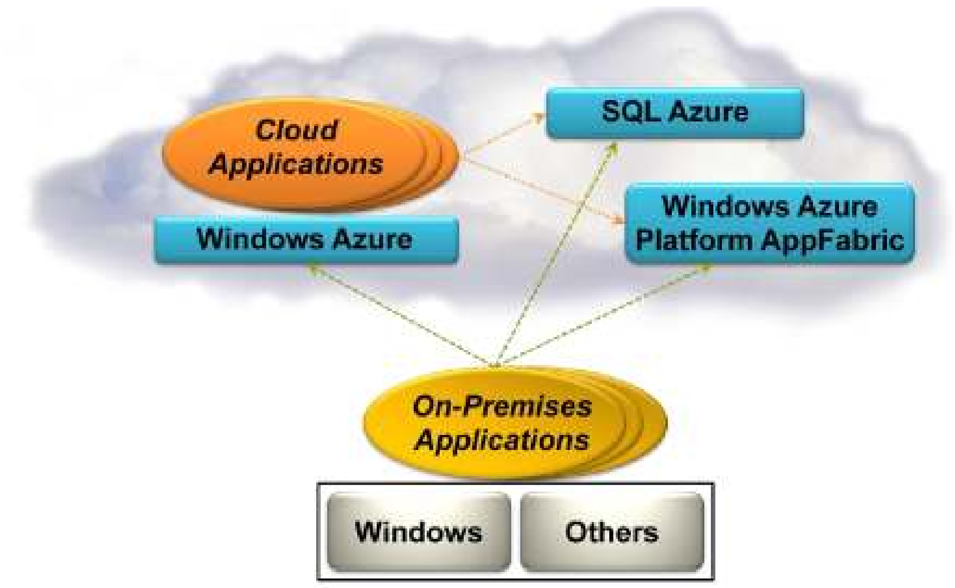
PHP

MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:



Execution Environment

The Windows Azure execution environment consists of a platform for applications and services hosted within one or more roles. The types of roles you can implement in Windows Azure are:

- **Azure Compute (Web and Worker Roles)**. A Windows Azure application consists of one or more hosted roles running within the Azure data centers. Typically there will be at least one Web role that is exposed for access by users of the application. The application may contain additional roles, including Worker roles that are typically used to perform background processing and support tasks for Web roles. For more detailed information see "Overview of Creating a Hosted Service for Windows Azure" athttp://technet.microsoft.com/en-au/library/gg432976.aspx and "Building an Application that Runs in a Hosted Service" at http://technet.microsoft.com/en-au/library/hh180152.aspx.
- **Virtual Machine (VM role)**. This role allows you to host your own custom instance of the Windows Server 2008 R2 Enterprise or Windows Server 2008 R2 Standard operating system within a Windows Azure data center. For more detailed information see "Creating Applications by Using a VM Role in Windows Azure" athttp://technet.microsoft.com/en-au/library/gg465398.aspx.

Data Management

Windows Azure, SQL Azure, and the associated services provide opportunities for storing and managing data in a range of ways. The following data management services and features are available:

- **Azure Storage:** This provides four core services for persistent and durable data storage in the cloud. The services support a REST interface that can be accessed from within Azure-hosted or on-premises (remote) applications. For information about the REST API, see "Windows Azure Storage Services REST API Reference" athttp://msdn.microsoft.com/en-us/library/dd179355.aspx. The four storage services are:
    - **The Azure Table Service** provides a table-structured storage mechanism based on the familiar rows and columns format, and supports queries for managing the data. It is primarily aimed at scenarios where large volumes of data must be stored, while being easy to access and update. For more detailed information see "Table Service Concepts" at http://msdn.microsoft.com/en-us/library/dd179463.aspx and "Table Service API" at http://msdn.microsoft.com/en-us/library/dd179423.aspx.

- **The Binary Large Object (BLOB) Service** provides a series of containers aimed at storing text or binary data. It provides both Block BLOB containers for streaming data, and Page BLOB containers for random read/write operations. For more detailed information see "Understanding Block Blobs and Page Blobs" athttp://msdn.microsoft.com/en-us/library/ee691964.aspx and "Blob Service API" at http://msdn.microsoft.com/en-us/library/dd135733.aspx.
- **The Queue Service** provides a mechanism for reliable, persistent messaging between role instances, such as between a Web role and a Worker role. For more detailed information see "Queue Service Concepts" at http://msdn.microsoft.com/en-us/library/dd179353.aspx and "Queue Service API" athttp://msdn.microsoft.com/en-us/library/dd179363.aspx.
- Windows Azure Drives provide a mechanism for applications to mount a single volume NTFS VHD as a Page BLOB, and upload and download VHDs via the BLOB. For more detailed information see "Windows Azure Drive" (PDF) at http://go.microsoft.com/?linkid=9710117.

- **SQL Azure Database:** This is a highly available and scalable cloud database service built on SQL Server technologies, and supports the familiar T-SQL based relational database model. It can be used with applications hosted in Windows Azure, and with other applications running on-premises or hosted elsewhere. For more detailed information see "SQL Azure Database" at http://msdn.microsoft.com/en-us/library/ee336279.aspx.
- **Data Synchronization:** SQL Azure Data Sync is a cloud-based data synchronization service built on Microsoft Sync Framework technologies. It provides bi-directional data synchronization and data management capabilities allowing data to be easily shared between multiple SQL Azure databases and between on-premises and SQL Azure databases. For more detailed information see "Microsoft Sync Framework Developer Center" at http://msdn.microsoft.com/en-us/sync.
- **Caching:** This service provides a distributed, in-memory, low latency and high throughput application cache service that requires no installation or management, and dynamically increases and decreases the cache size automatically as required. It can be used to cache application data, ASP.NET session state information, and for ASP.NET page output caching. For more detailed information see "Caching Service (Windows Azure AppFabric)" at http://msdn.microsoft.com/en-us/library/gg278356.aspx.

Networking Services

Windows Azure provides several networking services that you can take advantage of to maximize performance, implement authentication, and improve manageability of your hosted applications. These services include the following:

- **Content Delivery Network (CDN).** The CDN allows you to cache publicly available static data for applications at strategic locations that are closer (in network delivery terms) to end users. The CDN uses a number of data centers at many locations around the world, which store the data in BLOB storage that has anonymous access. These do not need to be locations where the application is actually running. For more detailed information see "Delivering High-Bandwidth Content with the Windows Azure CDN" athttp://msdn.microsoft.com/en-us/library/ee795176.aspx.
- **Virtual Network Connect.** This service allows you to configure roles of an application running in Windows Azure and computers on your on-premises network so that they appear to be on the same network. It uses a software agent running on the on-premises computer to establish an IPsec-protected connection to the Windows Azure roles in the cloud, and provides the capability to administer, manage, monitor, and debug the roles directly. For more detailed information see "Connecting Local Computers to Windows Azure Roles" at http://msdn.microsoft.com/en-us/library/gg433122.aspx.
- **Virtual Network Traffic Manager.** This is a service that allows you to set up request redirection and load balancing based on three different methods. Typically you will use Traffic Manager to maximize performance by redirecting requests from users to the instance in the closest data center using the Performance method. Alternative load balancing methods available are Failover and Round Robin. For more detailed information see "Windows Azure Traffic Manager" at http://msdn.microsoft.com/en-us/WAZPlatformTrainingCourse_WindowsAzureTrafficManager.
- **Access Control.** This is a standards-based service for identity and access control that makes use of a range of identity providers (IdPs) that can authenticate users. ACS acts as a Security Token Service (STS), or token issuer, and makes it easier to take advantage of federation authentication techniques where user identity is validated in a realm or domain other than that in which the application resides. An example is controlling user access based on an identity verified by an identity provider such as Windows Live ID or Google. For more detailed information see "Access Control Service 2.0" at http://msdn.microsoft.com/en-

us/library/gg429786.aspx and “Claims Based Identity & Access Control Guide” at http://claimsid.codeplex.com/.

- **Service Bus.** This provides a secure messaging and data flow capability for distributed and hybrid applications, such as communication between Windows Azure hosted applications and on-premises applications and services, without requiring complex firewall and security infrastructures. It can use a range of communication and messaging protocols and patterns to provide delivery assurance, reliable messaging; can scale to accommodate varying loads; and can be integrated with on-premises BizTalk Server artifacts. For more detailed information see “AppFabric Service Bus” at http://msdn.microsoft.com/en-us/library/ee732537.aspx

Conclusion

Performed case study of Microsoft Azure Cloud computing platform and services.

Assignment No: 09

Aim: Study and implementation of infrastructure as Service using Open Stack.

Objectives:

1. To learn Infrastructure as a Service.
2. To implement IaaS using Open Stack.

Software Requirements:

Ubuntu 18.04

PHP

MySQL

Hardware Requirements:

Pentium IV system with latest configuration

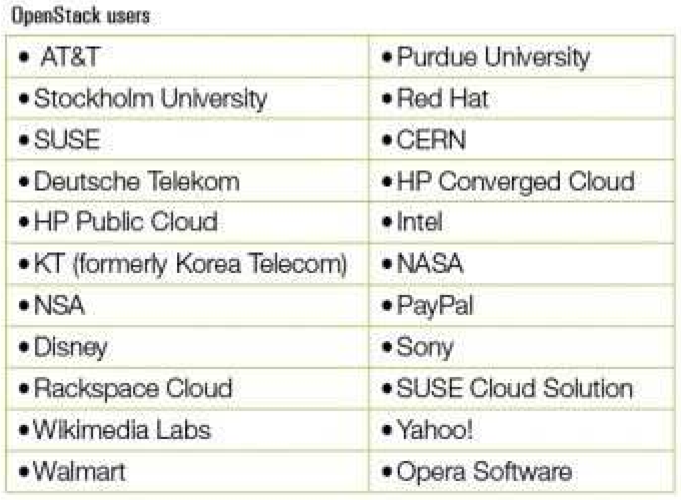
Theory:

Infrastructure as-a-service (IaaS)

IaaS includes the delivery of computing infrastructure such as a virtual machine, disk image library, raw block storage, object storage, firewalls, load balancers, IP addresses, virtual local area networks and other features on-demand from a large pool of resources installed in data centres. Cloud providers bill for the IaaS services on a utility computing basis; the cost is based on the amount of resources allocated and consumed.

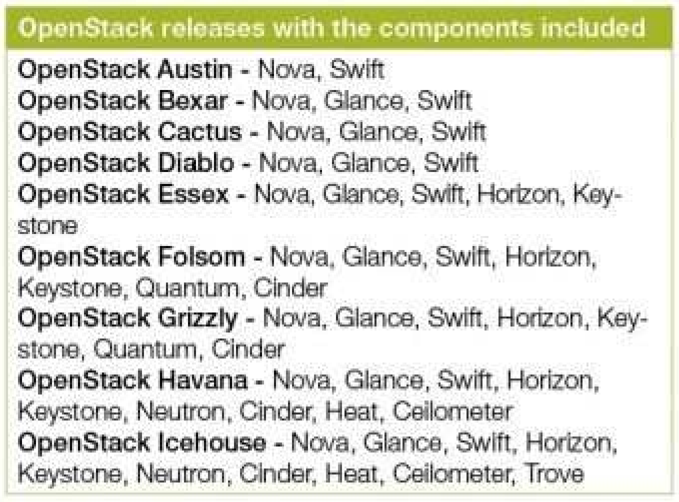
OpenStack: a free and open source cloud computing platform

OpenStack is a free and open source, cloud computing software platform that is widely used in the deployment of infrastructure-as-a-Service (IaaS) solutions. The core technology with OpenStack comprises a set of interrelated projects that control the overall layers of processing, storage and networking resources through a data centre that is managed by the users using a Web-based dashboard, command-line tools, or by using the RESTful API. Currently, OpenStack is maintained by the OpenStack Foundation, which is a non-profit corporate organisation established in September 2012 to promote OpenStack software as well as its community. Many corporate giants have joined the project, including GoDaddy, Hewlett Packard, IBM, Intel, Mellanox, Mirantis, NEC, NetApp, Nexenta, Oracle, Red Hat, SUSE Linux, VMware, Arista Networks, AT&T, AMD, Avaya, Canonical, Cisco, Dell, EMC, Ericsson, Yahoo!, etc.



OpenStack users

• AT&T	• Purdue University
• Stockholm University	• Red Hat
• SUSE	• CERN
• Deutsche Telekom	• HP Converged Cloud
• HP Public Cloud	• Intel
• KT (formerly Korea Telecom)	• NASA
• NSA	• PayPal
• Disney	• Sony
• Rackspace Cloud	• SUSE Cloud Solution
• Wikimedia Labs	• Yahoo!
• Walmart	• Opera Software



OpenStack releases with the components included

OpenStack Austin - Nova, Swift
OpenStack Bexar - Nova, Glance, Swift
OpenStack Cactus - Nova, Glance, Swift
OpenStack Diablo - Nova, Glance, Swift
OpenStack Essex - Nova, Glance, Swift, Horizon, Keystone
OpenStack Folsom - Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
OpenStack Grizzly - Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
OpenStack Havana - Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer
OpenStack Icehouse - Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove

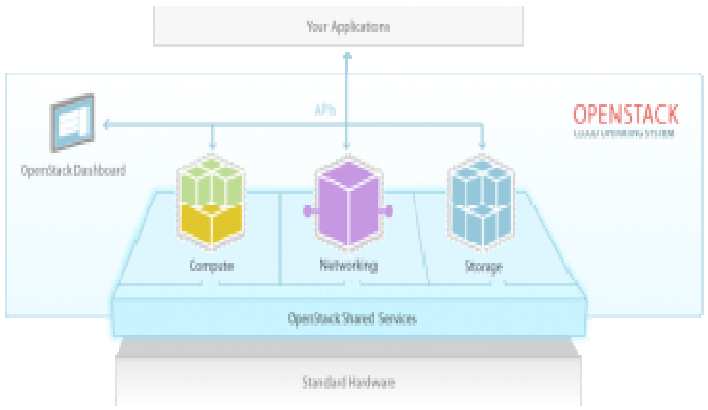
OpenStack computing components

OpenStack has a modular architecture that controls large pools of compute, storage and networking resources.

Compute (Nova): OpenStack Compute (Nova) is the fabric controller, a major component of Infrastructure as a Service (IaaS), and has been developed to manage and automate pools of computer resources. It works in association with a range of virtualisation technologies. It is written in Python and uses many external libraries such as Eventlet, Kombu and SQLAlchemy.

Object storage (Swift): It is a scalable redundant storage system, using which objects and files are placed on multiple disks throughout servers in the data centre, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. OpenStack Swift replicates the content from other active nodes to new locations in the cluster in case of server or disk failure.
Block storage (Cinder): OpenStack block storage (Cinder) is used to incorporate continual block-level storage devices for usage with OpenStack compute instances. The block storage system of OpenStack is used to manage the creation, mounting and unmounting of the block devices to servers. Block storage is integrated for performance-aware scenarios including database storage, expandable file systems or providing a server with access to raw block level storage. Snapshot management in OpenStack provides the authoritative functions and modules for the back-up of data on block storage volumes. The snapshots can be restored and used again to create a new block storage volume.
Networking (Neutron): Formerly known as Quantum, Neutron is a specialised component of OpenStack for managing networks as well as network IP addresses. OpenStack networking makes sure that the network does not face bottlenecks or any complexity issues in cloud deployment. It provides the users continuous self-service capabilities in the network s infrastructure. The floating IP addresses allow traffic to be dynamically routed again to any resources in the IT infrastructure, and therefore the users can redirect traffic during maintenance or in case of any failure. Cloud users can create their own networks and control traffic along with the connection of servers and devices to one or more networks. With this component, OpenStack delivers the extension framework that can be implemented for managing additional network services including intrusion detection systems (IDS), load balancing, firewalls, virtual private networks (VPN) and many others.
Dashboard (Horizon): The OpenStack dashboard (Horizon) provides the GUI (Graphical User Interface) for the access, provision and automation of cloud-based resources. It embeds various third party products and services including advance monitoring, billing and various management tools.
Identity services (Keystone): Keystone provides a central directory of the users, which is mapped to the OpenStack services they are allowed to access. It refers and acts as the centralised authentication system across the cloud operating system and can be integrated with directory services like LDAP. Keystone supports various authentication types including classical username and password credentials, token-based systems and other log-in management systems.
Image services (Glance): OpenStack Image Service (Glance) integrates the registration, discovery and delivery services for disk and server images. These stored images can be used as templates. It can also be used to store and catalogue an unlimited number of backups. Glance can store disk and

server images in different types and varieties of back-ends, including Object Storage.
Telemetry (Ceilometer): OpenStack telemetry services (Ceilometer) include a single point of contact for the billing systems. These provide all the counters needed to integrate customer billing across all current and future OpenStack components.
Orchestration (Heat): Heat organises a number of cloud applications using templates with the help of the OpenStack-native REST API and a CloudFormation-compatible Query API.
Database (Trove): Trove is used as database-as-a-service (DaaS), which integrates and provisions relational and non-relational database engines.
Elastic Map Reduce (Sahara): Sahara is the specialised service that enables data processing on OpenStack-managed resources, including the processing with Apache Hadoop.



Deployment of OpenStack using DevStack

DevStack is used to quickly create an OpenStack development environment. It is also used to demonstrate the starting and running of OpenStack services, and provide examples of using them from the command line. DevStack has evolved to support a large number of configuration options and alternative platforms and support services. It can be considered as the set of scripts which install all the essential OpenStack services in the computer without any additional software or configuration. To implement DevStack, first download all the essential packages, pull in the OpenStack code from various OpenStack projects, and set everything for the deployment. To install OpenStack using DevStack, any Linux-based distribution with 2GB RAM can be used to start the implementation of IaaS.
Here are the steps that need to be followed for the installation.

1. Install Git

```
$ sudo apt-get install git
```

2. Clone the DevStack repository and change the directory. The code will set up the cloud infrastructure.

```
$ git clone http://github.com/openstack-dev/devstack
$ cd devstack/

/devstack$ ls

accrc exercises HACKING.rst rejoin-stack.sh tests
AUTHORS exercise.sh lib run_tests.sh tools
clean.sh extras.d LICENSE samples unstack.sh
driver_certs files localrc stackrc
eucarc functions openrc stack-screenrc
exerciserc functions-common README.md stack.sh
```

stack.sh, unstack.sh and *rejoin-stack.sh* are the most important files. *stack.sh* script is used to set up DevStack.unstack.sh is used to destroy the DevStack setup.

If you are on the earlier execution of *./stack.sh*, the environment can be brought up by executing the rejoin_*stack.sh* script.

3. Execute the stack.sh script:

```
/devstack$ ./stack.sh
```

Here, the MySQL database password is entered. There is no need to worry about the installation of MySQL separately on this system. We have to specify a password and this script will install MySQL, and use this password there.

Finally, we will have the script ending as follows:

```
+ merge_config_group /home/r/devstack/local.conf post-extra
+ local localfile=/home/r/devstack/local.conf
+ shift
+ local matchgroups=post-extra
+ [[ -r /home/r/devstack/local.conf ]]
+ return 0
+ [[ -x /home/r/devstack/local.sh ]]
+ service_check
+ local service
+ local failures
+ SCREEN_NAME=stack
+ SERVICE_DIR=/opt/stack/status
+ [[ ! -d /opt/stack/status/stack ]]
++ ls /opt/stack/status/stack/*.failure
++ /bin/true
+ failures=
+  [  -n     ]
+ set +o xtrace
```

- Horizon is now available at *http://1.1.1.1/*
- Keystone is serving at *http://1.1.1.1:5000/v2.0/*
- Examples on using the *novaclient* command line are in *exercise.sh*
- The default users are: admin and demo
- The password: nova

- This is your host IP: *1.1.1.1*

After all these steps, the machine becomes the cloud service providing platform. Here, 1.1.1.1 is the IP of my first network interface.

We can type the host IP provided by the script into a browser, in order to access the dashboard Horizon . We can log in with the username admin or demo and the password admin. You can view all the process logs inside the screen, by typing the following command:

```
$ screen -x
```

Executing the following will kill all the services, but it should be noted that it will not delete any of the code.

To bring down all the services manually, type:

```
$ sudo killall screen
```

localrc configurations

localrc is the file in which all the local configurations (local machine parameters) are maintained. After the first successful *stack.sh* run, you will see that a *localrc* file gets created with the configuration values you specified while running that script.

The following fields are specified in the *localrc* file:

DATABASE_PASSWORD

RABBIT_PASSWORD

SERVICE_TOKEN

SERVICE_PASSWORD

ADMIN_PASSWORD

If we specify the option *OFFLINE=True* in the *localrc* file inside DevStack directory, and if after specifying this, we run *stack.sh*, it will not check any parameter over the Internet. It will set up DevStack using all the packages and code residing in the local system. In the phase of code development, there is need to commit the local changes in the */opt/stack/nova* repository before restack (re-running stack.sh) with the *RECLONE=yes option.* Otherwise, the changes will not be committed.

To use more than one interface, there is a need to specify which one to use for the external IP using this configuration:

```
HOST_IP=xxx.xxx.xxx.xxx
```

Cinder on DevStack

Cinder is a block storage service for OpenStack that is designed to allow the use of a reference implementation (LVM) to present storage resources to end users that can be consumed by the OpenStack Compute Project (Nova). Cinder is used to virtualise the pools of block storage devices. It delivers end users with a self-service API to request and use the resources, without requiring any specific complex knowledge of the location and configuration of the storage where it is actually deployed.

All the Cinder operations can be performed via any of the following:

1. CLI (Cinder s *python-cinderclient* command line module)
2. GUI (Using OpenStack s GUI project *horizon*)
3. Direct calling of Cinder APIs

Creation and deletion of volumes: To create a 1 GB Cinder volume with no name, run the following command:

```
$ cinder create 1
```

To see more information about the command, just type cinder help <*command*>

```
$ cinder help create

usage: cinder create [--snapshot-id <snapshot-id>]
[--source-volid <source-volid>] [--image-id <image-id>]
[--display-name <display-name>]
[--display-description <display-description>]
[--volume-type <volume-type>]
[--availability-zone <availability-zone>]
[--metadata [<key=value> [<key=value> ...]]]
<size>
Add a new volume.
Positional arguments:
<size> Size of volume in GB
Optional arguments:
--snapshot-id <snapshot-id>
Create volume from snapshot id (Optional,
Default=None)
--source-volid <source-volid>
Create volume from volume id (Optional, Default=None)
--image-id <image-id>
Create volume from image id (Optional, Default=None)
--display-name <display-name>
Volume name (Optional, Default=None)
--display-description <display-description>
Volume description (Optional, Default=None)
--volume-type <volume-type>
Volume type (Optional, Default=None)
--availability-zone <availability-zone>
Availability zone for volume (Optional, Default=None)
--metadata [<key=value> [<key=value> ...]]
Metadata key=value pairs (Optional, Default=None)
```

To create a Cinder volume of size 1GB with a name, using *cinder create –display-name myvolume:*

```
$ cinder create --display-name myvolume 1
+-------------------+------------------------------------------+
| Property | Value |
+-------------------+----------------------------------------- -+
| attachments | [] |
| availability_zone | nova |
| bootable | false |
| created_at | time | | display_description | None |
| display_name | myvolume | | id | id |
| metadata | {} |
| size | 1 |
| snapshot_id | None |
| source_volid | None |
| status | creating |
| volume_type | None |
+------------------------+-----------------------------------+
```

To list all the Cinder volumes, using *cinder list:*

```
$ cinder list
ID Status Display Name Size Volume type Bootable Attached To
id1 Available Myvolume 1 None False
id2 Available None 1 None False
```

To delete the first volume (the one without a name), use the *cinder delete <volume_id>* command. If we execute *cinder* list really quickly, the status of the volume going to deleting can be seen, and after some time, the volume will be deleted:

```
$ cinder delete id2

$ cinder list
ID Status Display Name Size Volume type Bootable Attached To
id1 Available Myvolume 1 None False
id2 Deleting None 1 None False
```

Volume snapshots can be created as follows:

```
$ cinder snapshot-create id2
+---------------------+----------------------------------+
| Property | Value |
+---------------------+------------------------------------+
| created_at | TimeStamp |
| display_description | None |
| display_name | None |
| id | snapshot2 |
| metadata | {} |
| size | 1 |
| status | creating |
| volume_id | id2 |
+---------------------+-----------------------------------+
```

All the snapshots can be listed as follows:

```
$ cinder snapshot-list

ID Volume ID Status Display Name Size
Snapshotid1 id2 Available None 1
```

You can also create a new volume of 1GB from the snapshot, as follows:

```
$ cinder create --snapshot-id snapshotid1 1
+---------------------+--------------------------------------
| Property | Value
+---------------------+--------------------------------------
| attachments | [] |
| availability_zone | nova |
| bootable | false |
| created_at | creationtime |
| display_description | None |
| display_name | None |
| id | v1 |
| metadata | {} |
| size | 1 |
| snapshot_id | snapshotid1 |
| source_volid | None |
| status | creating |
| volume_type | None |
+---------------------+--------------------------------------+
```

There are lots of functions and features available with OpenStack related to cloud deployment. Depending upon the type of implementation, including load balancing, energy optimisation, security and others, the cloud computing framework OpenStack can be explored a lot.

Conclusion

We learnt about configuring and implementing the Infrastructure as a Service using Open stack.

Assignment No: 10

Aim: Implementation of Virtualization in Cloud Computing to Learn Virtualization Basics, Benefits of Virtualization in Cloud using Open Source Operating System.

Objectives:

1. To learn Virtualization basics.
2. To implement basic OS virtualization using VMware.

Software Requirements:

Ubuntu 18.04

PHP

MySQL

Vmware station

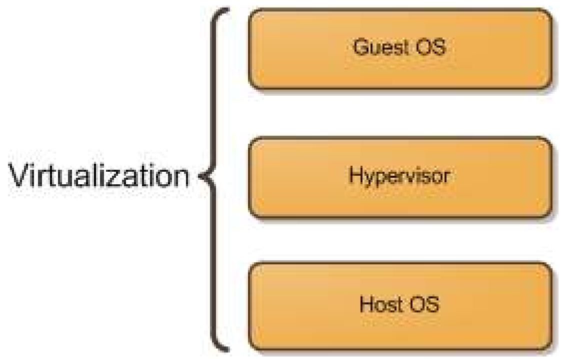
Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Virtualization is not a new concept, but its complexity has been growing, and a number of new paradigms are rising. I will try to demystify some of the concepts behind virtualization, briefly explain some of its basics, and finally look at some of the products and solutions out there.

To begin, let me introduce three very simple concepts regarding virtualization: the host operating system, the hypervisor, and the guest operating system.



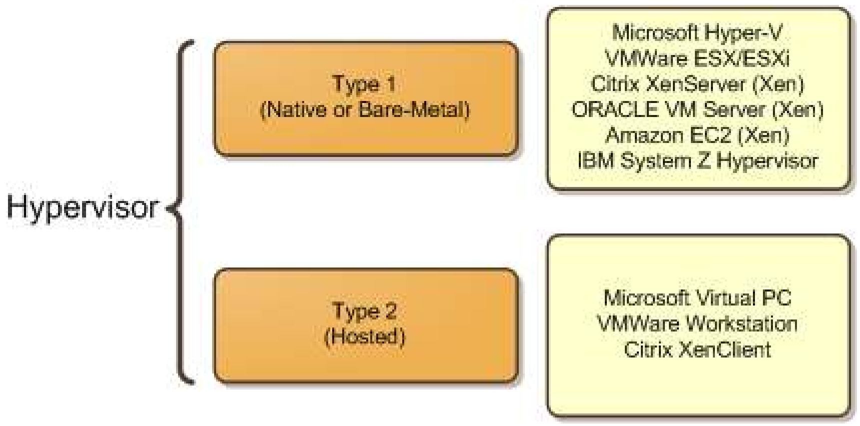
The host operating system provides a host to one or more virtual machines (or partitions) and shares physical resources with them. It’s where the virtualization product or the partitioning product is installed.

The guest operating system is the operating system installed inside a virtual machine (or a partition). In a virtualization solution the guest OS can be completely different from the host OS. In a partitioning solution the guest OS must be identical to the host OS.

A hypervisor, also called a virtual machine manager (VMM), is a program that allows multiple operating systems to share a single hardware host. Each operating system appears to have the host’s processor, memory, and other resources all to itself. The task of this hypervisor is to handle resource and memory allocation for the virtual machines, ensuring they cannot disrupt each other, in addition to providing interfaces for higher level administration and monitoring tools.

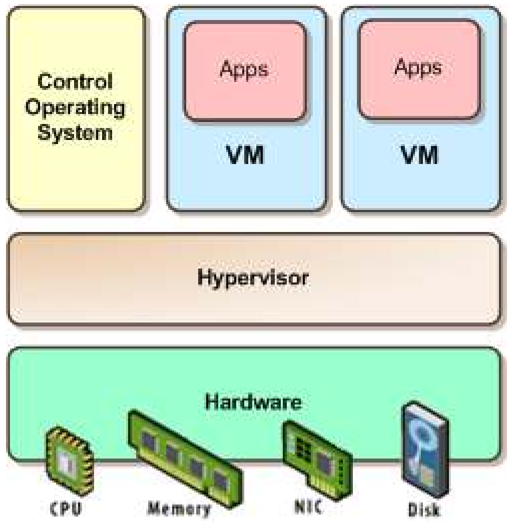
The Hypervisor

There are two types of hypervisors as depicted below:

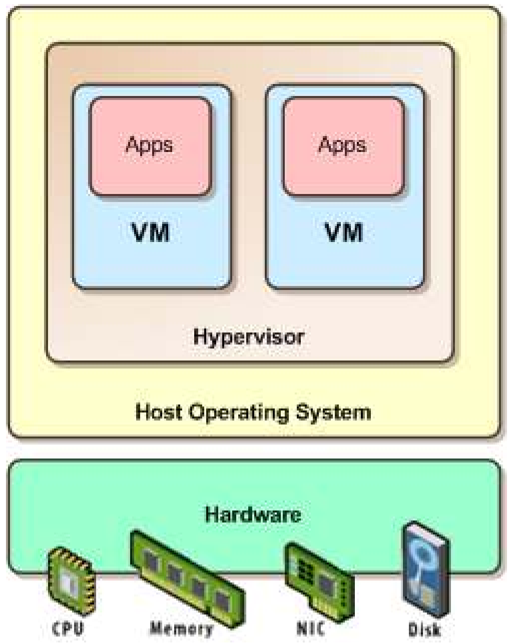


Note: Xen is an open-source virtualization software used by several companies to implement their virtualization solution; companies like, ORACLE, Citrix, Sun, and Virtual Iron, to name a few.

Type 1 hypervisors, also known as bare-metal, are software systems that run directly on the host’s hardware as a hardware control and guest operating system monitor. Bare-metal virtualization is the current enterprise data center leader. VMware ESX is easily the market leader in enterprise virtualization at the moment, and it utilizes bare-metal virtualization architecture. What is immediately apparent about this architecture, is the lack of an existing OS; the hypervisor sits directly on top of the hardware, hence the term “bare-metal virtualization”. The reason so many data centers implement bare-metal products, such as ESX, Xen, and Hyper-V, is because of the speed it provides due to the decreased overhead from the OS that hosted virtualization uses.

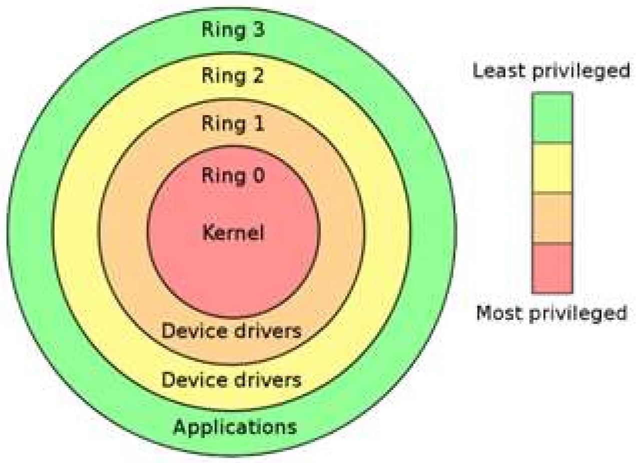


Type 2 hypervisors, also known as hosted, are software applications running within a conventional operating system environment. This type of hypervisor is typically used in client side virtualization solutions such as Microsoft´s Virtual PC, and VMWare´s Workstation.



The Protection Rings

Another important concept is the protection rings. x86 CPUs provide a range of protection levels, also known as rings, in which code can execute. Ring 0 has the highest level privilege and is where the operating system kernel normally runs. Code executing in Ring 0 is said to be running in system space, kernel mode or supervisor mode. All other code, such as applications running on the operating system, operate in less privileged rings, typically Ring 3.



The hypervisor runs directly on the hardware of the host system in ring 0. Clearly, with the hypervisor occupying ring 0 of the CPU, the kernels for any guest operating systems running on the system must run in less privileged CPU rings. Unfortunately, most operating system kernels are written explicitly to run in ring 0, for the simple reason that they need to perform tasks that are only available in that ring, such as the ability to execute privileged CPU instructions and directly manipulate memory.

The AMD-V and Intel-VT CPUs use a new privilege level called Ring -1 for the VMM to reside, allowing for better performance as the VMM no longer needs to fool the Guest OS that it is running in Ring 0. Solutions like VMWare ESX, Xen (Citrix, ORACLE, IBM, etc.), and Microsoft Hyper-V take advantage of the hardware virtualization capabilities inherent to the new Intel and AMD CPUs.

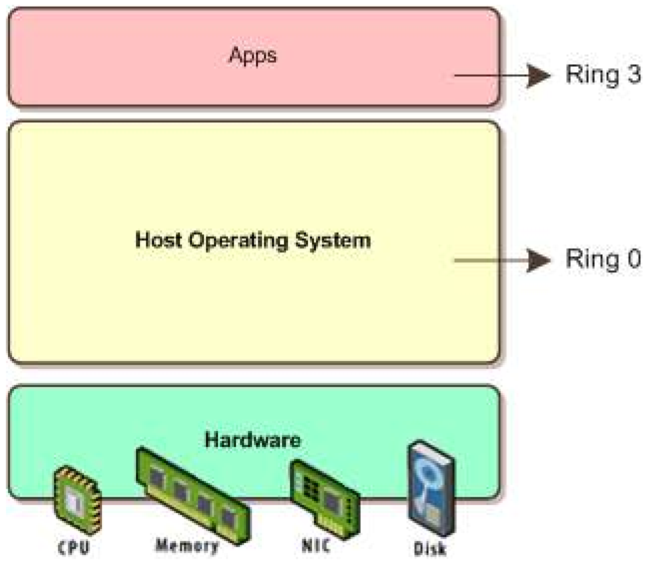
Virtualization Landscape

After this brief introduction, let´s now take a look at the global virtualization landscape available out there. The following diagram shows how virtualization architectures are organized, as well as some of the solutions that implement them.

The following sections will briefly introduce some of the most important types of virtualization.

Traditional

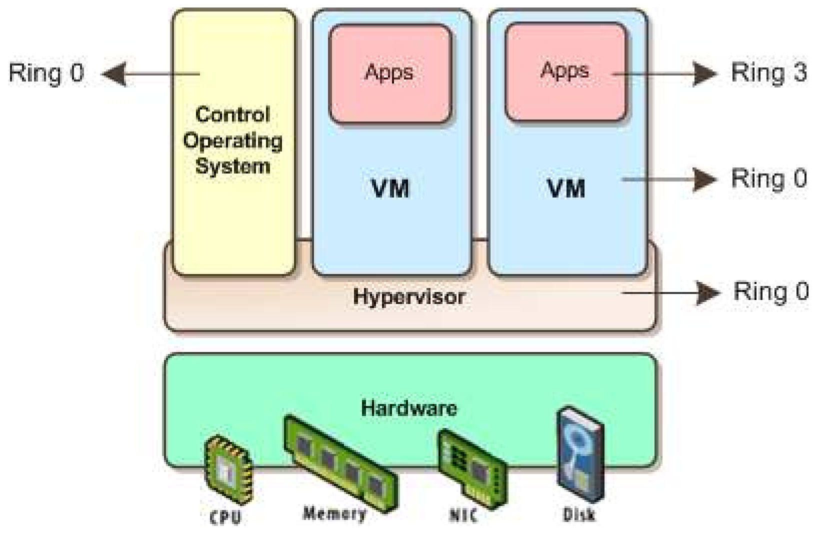
This is not a virtualization scenario; it´s here solely for comparison purposes. Here we see that the OS sits directly above the hardware executing in the ring 0.



Paravirtualization

Under paravirtualization, the kernel of the guest operating system is modified specifically to run on the hypervisor. This typically involves replacing any privileged operations that will only run in ring 0 of the CPU with calls to the hypervisor (known as hypercalls). The hypervisor in turn performs the task on behalf of the guest kernel.

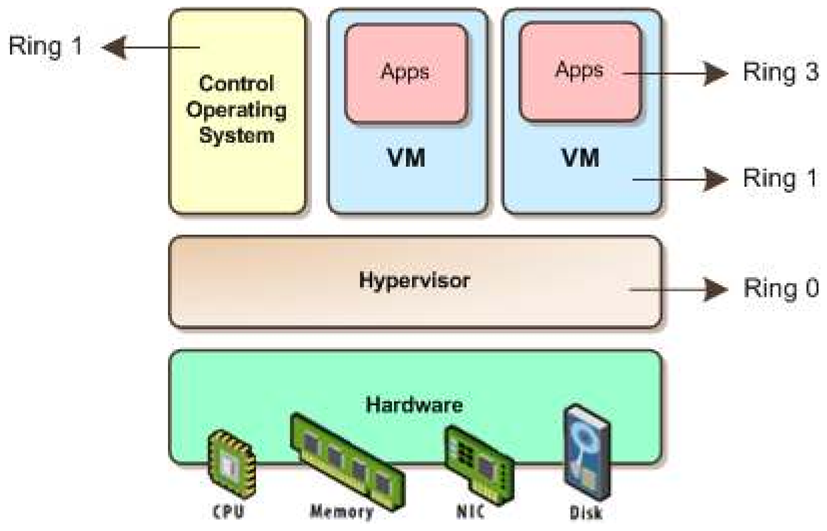
This typically limits support to open source operating systems, such as Linux, which may be freely altered, and proprietary operating systems where the owners have agreed to make the necessary code modifications to target a specific hypervisor. This results in the ability of the guest kernel to communicate directly with the hypervisor, resulting in greater performance levels than other virtualization approaches.



Full Virtualization without Hardware Assist

Full virtualization provides support for unmodified guest operating systems. The term unmodified refers to operating system kernels which have not been altered to run on a hypervisor and, therefore, still execute privileged operations as though running in ring 0 of the CPU.

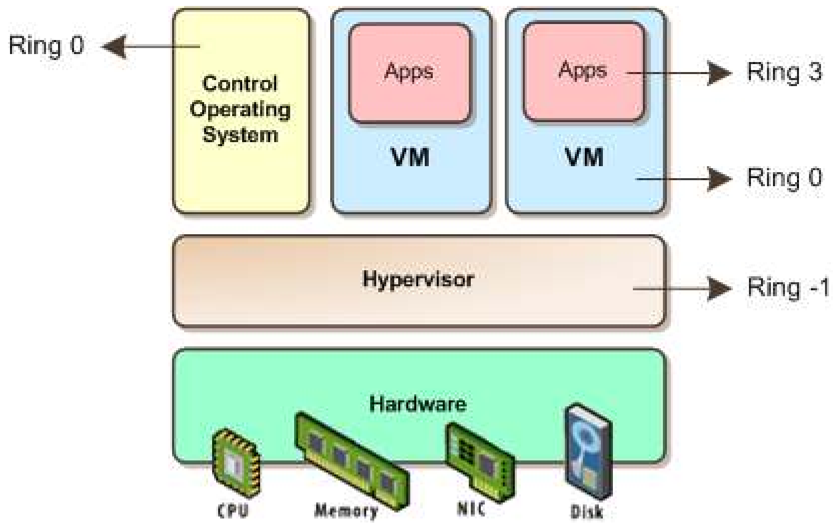
In this scenario, the hypervisor provides CPU emulation to handle and modify privileged and protected CPU operations made by unmodified guest operating system kernels. Unfortunately, this emulation process requires both time and system resources to operate, resulting in inferior performance levels when compared to those provided by paravirtualization.



Full Virtualization with Hardware Assist

Hardware virtualization leverages virtualization features built into the latest generations of CPUs from both Intel and AMD. These technologies, known as Intel VT and AMD-V, respectively, provide extensions necessary to run unmodified guest virtual machines without the overheads inherent in full virtualization CPU emulation.

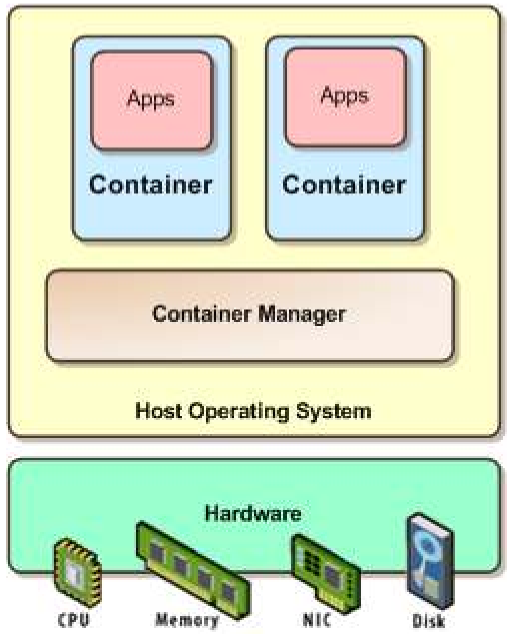
In very simplistic terms, these new processors provide an additional privilege mode below ring 0 in which the hypervisor can operate essentially, leaving ring 0 available for unmodified guest operating systems.



OS virtualization

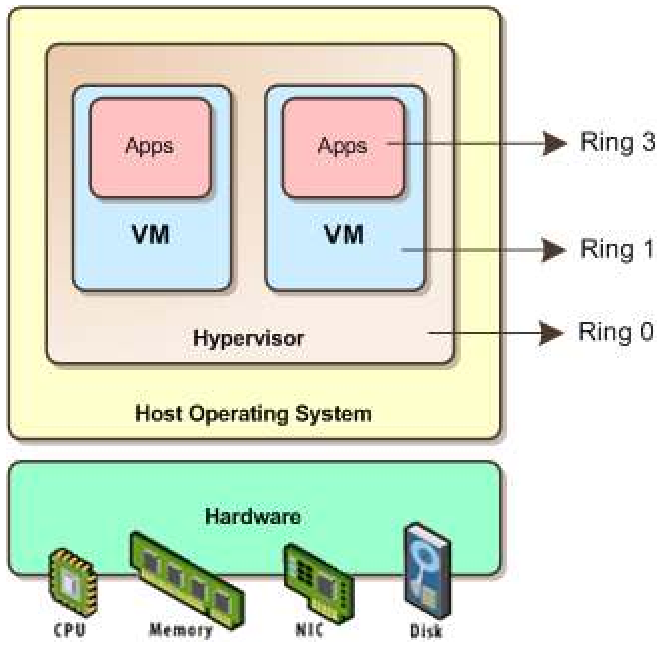
Compared with hypervisor based virtualization, container based virtualization offers a completely different approach to virtualization. Instead of virtualizing with a system in which there is a complete operating system installation, container based virtualization isolates containers work from within a single OS. In cases where only one operating system is needed, the main benefits of container based virtualization are that it doesn’t duplicate functionality and improves performance. OS virtualization has been making waves lately because Microsoft is rumored to be in the market for an OS virtualization technology. The most well-known products that use OS virtualization are Parallels Virtuozzo and Solaris Containers. This virtualization architecture has many benefits, speedy performance being the foremost. Another benefit is reduced disk space requirements. Many containers can use the same files, resulting in lowered disk space requirements.

The big caveat with OS virtualization is the OS requirement. Container OSs must be the same OS as the host OS. This means that if you are utilizing Solaris containers then all containers must run Solaris. If you are implementing Virtuozzo containers on Windows 2003 Standard Edition, then all its containers must also be running Windows 2003 Standard Edition.



Hosted virtualization

This is the type of virtualization with which most users are familiar with. All of the desktop virtualization products, such as VMware Workstation, VMware Fusion, and Parallels Desktop for the Mac, and Microsoft Virtual PC implement hosted virtualization architecture. There are many benefits to this type of virtualization. Users can install a virtualization product onto their desktop just as any other application, and continue to use their desktop OS. Hosted virtualization products also take advantage of the host OS's device drivers, resulting in the virtualization product supporting whatever hardware the host does.



Conclusion

We learnt Virtualization and implementation using different applications and tools called as hypervisors, basic OS virtualization using Vmware.

Assignment No: 11

Aim: Assignment to install and configure Google App Engine.

Objectives:

1. To learn basics of Google App Engine.
2. To install and configure Google App Engine.

Software Requirements:

Ubuntu 18.04

Python

MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Google App Engine is Google's platform as a service offering that allows developers and businesses to build and run applications using Google's advanced infrastructure. These applications are required to be written in one of a few supported languages, namely: Java, Python, PHP and Go. It also requires the use of Google query language and that the database used is Google Big Table. Applications must abide by these standards, so applications either must be developed with GAE in mind or else modified to meet the requirements.

GAE is a platform, so it provides all of the required elements to run and host Web applications, be it on mobile or Web. Without this all-in feature, developers would have to source their own servers, database software and the APIs that would make all of them work properly together, not to mention the entire configuration that must be done. GAE takes this burden off the developers so they can concentrate on the app front end and functionality, driving better user experience.

Advantages of GAE include:

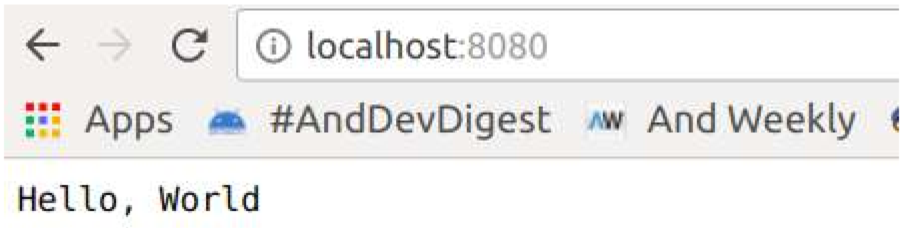
- Readily available servers with no configuration requirement
- Power scaling function all the way down to "free" when resource usage is minimal

- Automated cloud computing tools

1. Make sure you have python installed in your ubuntu system. run the command "*python -V*" and most probably you will get "Python 2.7.6" or above.
2. Crul https://sdk.cloud.google.com and use bash to run the commands by typing this command curl https://sdk.cloud.google.com | bash
3. Whenever you get to choose directories just hit enter, "YEAH IT WILL BE FINE".
4. Follow the instructions in the installation process.
5. Then run gcloud init
6. Follow the installation instructions as they are very straight forward.
7. Choose the account you want to use for google app engine.
8. Choose the project with numeric choice (don't use textual, you might make mistake). If you do not already have a google app engine project create a app engine project by following this link. https://console.cloud.google.com/start
9. Enable google api by pressing Y in the command line prompt.

Now as we have finished installing appengine, now it's time to create and upload an app. In this case we will be taking example of a "HELLO WORLD" app in python.

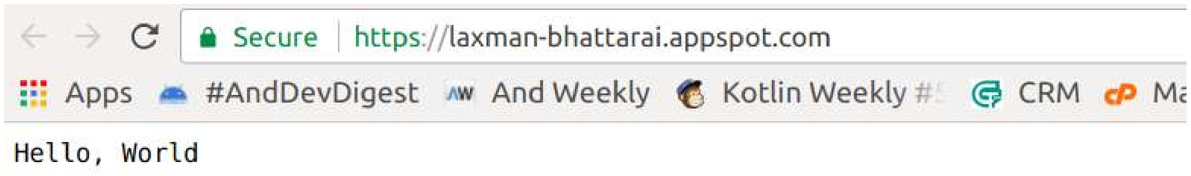
1. As we already have made sure that we have python installed in our system, It will be easier for us to clone existing code and deploy it rather than creating our own so we will use python-docs-sample. Run the command "git clone https://github.com/GoogleCloudPlatform/python-docs-samples".
2. cd to hello world sample by typing the command " cd python-docs-samples/appengine/standard/hello_world".
3. Then run the command "dev_appserver.py app.yml". It will run and give you the url of default and admin. If you go to the link of default you see the text hello world like this.



```
Hello, World
```

This is how you run the python app in your local server. But what we have to do is hosting the app in google app engine. To do so Now let's follow the following instructions.

1. Run the command *Ctrl + C* .
2. Being in the same working directory hello-world runt he command
   gcloud app deploy
3. Select the project you want to deploy the app , press Y and enter to continue. after that you will get the console output "Deployed service[default] to [Your web url for appengine] "
4. If you copy and paste the url, you will see the hello world in the browser too.



```
Hello, World
```

Web output

Now you have successfully uploaded your web app into app engine.

Conclusion

Hence we learnt to install and configure Google App Engine.

Assignment No: 12

Aim: Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store.

Objectives:

1. To learn basics of Google App Engine.
2. To install and configure Google App Engine.

Software Requirements:

Ubuntu 18.04

Python

MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Authenticating Users on App Engine Using Firebase

This assignment shows how to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Datastore.

The document walks you through a simple note-taking application called Firenotes that stores users' notes in their own personal notebooks. Notebooks are stored per user, and identified by each user's unique Firebase Authentication ID. The application has the following components:

- The frontend configures the sign-in user interface and retrieves the Firebase Authentication ID. It also handles authentication state changes and lets users see their notes.

- FirebaseUI is an open-source, drop-in solution that handles user login, linking multiple providers to one account, recovering passwords, and more. It implements authentication best practices for a smooth and secure sign-in experience.

- The backend verifies the user's authentication state and returns user profile information as well as the user's notes.

The application stores user credentials in Cloud Datastore by using the NDB client library, but you can store the credentials in a database of your choice.
The following diagram shows how the frontend and backend communicate with each other and how user credentials travel from Firebase to the database.

Firenotes is based on the Flask web application framework. The sample app uses Flask because of its simplicity and ease of use, but the concepts and technologies explored are applicable regardless of which framework you use.

Objectives

- Configure the Firebase Authentication user interface.
- Obtain a Firebase ID token and verify it using server-side authentication.
- Store user credentials and associated data in Cloud Datastore.
- Query a database using the NDB client library.
- Deploy an app to App Engine.

Costs
This assignment uses billable components of Cloud Platform, including:
- Google Cloud Datastore

Use the Pricing Calculator to generate a cost estimate based on your projected usage. New Cloud Platform users might be eligible for a free trial.
Before you begin
1. Install Git, Python 2.7, and virtualenv. For more information on setting up your Python development environment, such as installing the latest version of Python, refer to Setting Up a Python Development Environment for Google Cloud Platform.

2. Select or create a Google Cloud Platform project.

   Note: If you don't plan to keep the resources you create in this tutorial, create a new project instead of selecting an existing project. After you finish, you can delete the project, removing all resources associated with the project and tutorial.

   Go to the Manage resources page
3. Install and initialize the Cloud SDK.

If you have already installed and initialized the SDK to a different project, set the gcloud project to the App Engine project ID you're using for Firenotes. See Managing Cloud SDK Configurations for specific commands to update a project with the gcloud tool.

Cloning the sample app

To download the sample to your local machine:

1. Clone the sample application repository to your local machine:

   git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git

   Alternatively, you can download the sample as a zip file and extract it.
2. Navigate to the directory that contains the sample code:

   cd python-docs-samples/appengine/standard/firebase/firenotes

Adding the Firebase Authentication user interface

To configure FirebaseUI and enable identity providers:

1. Add Firebase to your app by following these steps:
   1. Create a Firebase project in the Firebase console.
      - If you don't have an existing Firebase project, click **Add project** and enter either an existing Google Cloud Platform project name or a new project name.
      - If you have an existing Firebase project that you'd like to use, select that project from the console.
   2. From the project overview page, click **Add Firebase to your web app**. If your project already has an app, select **Add App** from the project overview page.
   3. Use the Initialize Firebase section of your project's customized code snippet to fill out the following section of the frontend/main.js file:

      appengine/standard/firebase/firenotes/frontend/main.js

      View on GitHub

```
// Obtain the following from the "Add Firebase to your web app" dialogue
// Initialize Firebase
var config = {
  apiKey: "<API_KEY>",
  authDomain: "<PROJECT_ID>.firebaseapp.com",
  databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
  projectId: "<PROJECT_ID>",
  storageBucket: "<BUCKET>.appspot.com",
  messagingSenderId: "<MESSAGING_SENDER_ID>"
};
```

2. Edit the backend/app.yaml file and enter your Firebase project ID in the environment variables:

appengine/standard/firebase/firenotes/backend/app.yaml

View on GitHub

```
runtime: python27
api_version: 1
threadsafe: true
service: backend
handlers:
- url: /.*
  script: main.app

env_variables:
  # Replace with your Firebase project ID.
  FIREBASE_PROJECT_ID: '<PROJECT_ID>'
```

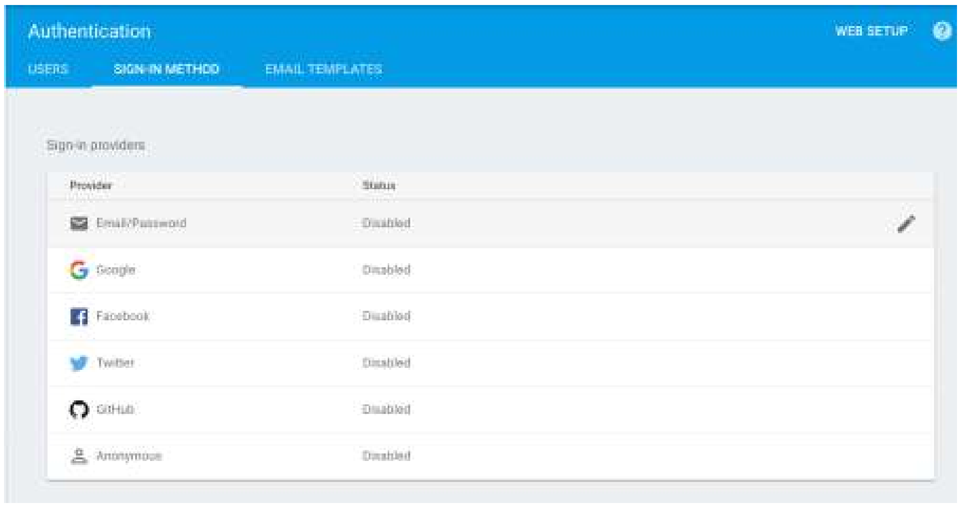
3. In the frontend/main.js file, configure the FirebaseUI login widget by selecting which providers you want to offer your users.

appengine/standard/firebase/firenotes/frontend/main.js

View on GitHub

```
// Firebase log-in widget
function configureFirebaseLoginWidget() {
  var uiConfig = {
    'signInSuccessUrl': '/',
    'signInOptions': [
      // Leave the lines as is for the providers you want to offer your users.
      firebase.auth.GoogleAuthProvider.PROVIDER_ID,
      firebase.auth.FacebookAuthProvider.PROVIDER_ID,
      firebase.auth.TwitterAuthProvider.PROVIDER_ID,
      firebase.auth.GithubAuthProvider.PROVIDER_ID,
      firebase.auth.EmailAuthProvider.PROVIDER_ID
    ],
    // Terms of service url
    'tosUrl': '<your-tos-url>',
  };
  var ui = new firebaseui.auth.AuthUI(firebase.auth());
  ui.start('#firebaseui-auth-container', uiConfig);
}
```

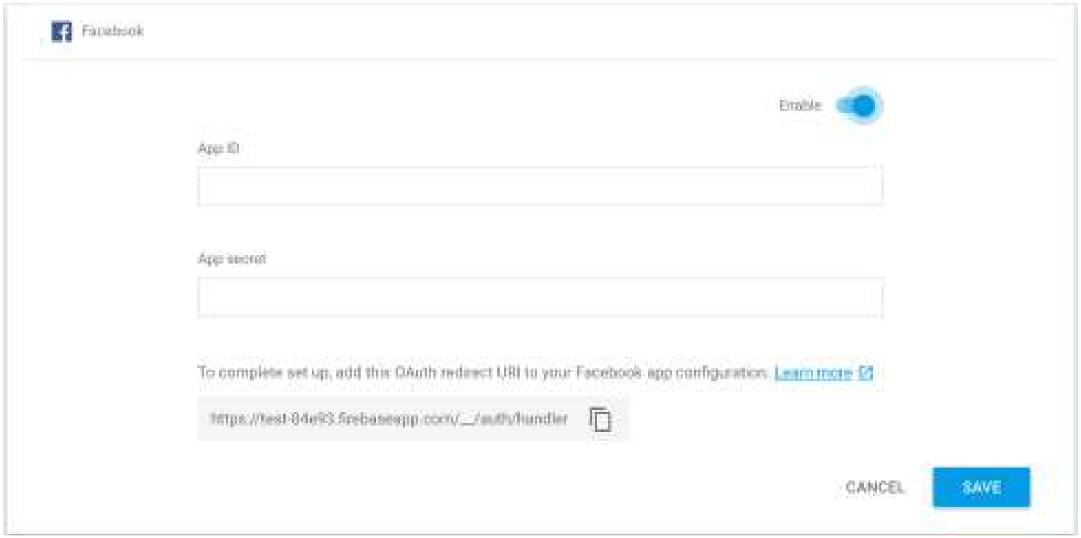
4. Enable the providers you have chosen to keep in the Firebase console by clicking **Authentication** > **Sign-in method**. Then, under **Sign-in providers**, hover the cursor over a provider and click the pencil icon.



Sign-in providers

Provider	Status
Email/Password	Disabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

1. Toggle the **Enable** button and, for third-party identity providers, enter the provider ID and secret from the provider's developer site. The Firebase docs give specific instructions in the "Before you begin" sections of the Facebook, Twitter, and GitHub guides. After enabling a provider, click **Save**.



App ID

App secret

To complete set up, add this OAuth redirect URI to your Facebook app configuration. Learn more

https://test-84e93.firebaseapp.com/__/auth/handler

2. In the Firebase console, under **Authorized Domains**, click **Add Domain** and enter the domain of your app on App Engine in the following format:
   [PROJECT_ID].appspot.com
   Do not include http:// before the domain name.

Installing dependencies

Navigate to the backend directory and complete the application setup:

Mac OS / Linux

Windows

1. Create an isolated Python environment in a directory external to your project and activate it:

   ```
   virtualenv env
   source env/bin/activate
   ```

2. Navigate to your project directory and install dependencies:

```
cd YOUR_PROJECT
pip install -t lib -r requirements.txt
```

In appengine_config.py, the vendor.add() method registers the libraries in the lib directory.

Running the application locally

To run the application locally, use the App Engine local development server:

1. Add the following URL as the backendHostURL in main.js:

   http://localhost:8081

2. Navigate to the root directory of the application. Then, start the development server:

   ```
   dev_appserver.py frontend/app.yaml backend/app.yaml
   ```

3. Visit http://localhost:8080/ in a web browser.

Authenticating users on the server

Now that you have set up a project and initialized an application for development, you can walk through the code to understand how to retrieve and verify Firebase ID tokens on the server.

Getting an ID token from Firebase

The first step in server-side authentication is retrieving an access token to verify. Authentication requests are handled with the onAuthStateChanged() listener from Firebase:

appengine/standard/firebase/firenotes/frontend/main.js

View on GitHub

```
firebase.auth().onAuthStateChanged(function(user) {
  if (user) {
    $('#logged-out').hide();
    var name = user.displayName;
```

```
    /* If the provider gives a display name, use the name for the
    personal welcome message. Otherwise, use the user's email. */
    var welcomeName = name ? name : user.email;
    user.getToken().then(function(idToken) {
      userIdToken = idToken;
      /* Now that the user is authenicated, fetch the notes. */
      fetchNotes();
      $('#user').text(welcomeName);
      $('#logged-in').show();
    });
  } else {
    $('#logged-in').hide();
    $('#logged-out').show();
  }
```

When a user is signed in, the Firebase getToken() method in the callback returns a Firebase ID token in the form of a JSON Web Token (JWT).

Verifying tokens on the server

After a user signs in, the frontend service fetches any existing notes in the user's notebook through an AJAX GET request. This requires authorization to access the user's data, so the JWT is sent in the Authorization header of the request using the Bearer schema:

appengine/standard/firebase/firenotes/frontend/main.js

View on GitHub

```
// Fetch notes from the backend.
function fetchNotes() {
  $.ajax(backendHostUrl + '/notes', {
    /* Set header for the XMLHttpRequest to get data from the web server
    associated with userIdToken */
    headers: {
      'Authorization': 'Bearer ' + userIdToken
    }
  })
```

Before the client can access server data, your server must verify the token is signed by Firebase. You can verify this token using the Google Authentication Library for Python. Use the authentication library's verify_firebase_token function to verify the bearer token and extract the claims:

appengine/standard/firebase/firenotes/backend/main.py

View on GitHub

```
id_token = request.headers['Authorization'].split(' ').pop()
claims = google.oauth2.id_token.verify_firebase_token(
   id_token, HTTP_REQUEST)
if not claims:
   return 'Unauthorized', 401
```

Each identity provider sends a different set of claims, but each has at least a sub claim with a unique user ID and a claim that provides some profile information, such as name or email, that you can use to personalize the user experience on your app.

Managing user data in Cloud Datastore

After authenticating a user, you need to store their data for it to persist after a signed-in session has ended. The following sections explain how to store a note as a Cloud Datastore *entity* and segregate entities by user ID.

Creating entities to store user data

You can create an entity in Cloud Datastore by declaring an NDB model class with certain properties such as integers or strings. Cloud Datastore indexes entities by *kind*; in the case of Firenotes, the kind of each entity is Note. For querying purposes, each Note is stored with a *key name*, which is the user ID obtained from the sub claim in the previous section.

The following code demonstrates how to set properties of an entity, both with the constructor method for the model class when the entity is created and through assignment of individual properties after creation:

appengine/standard/firebase/firenotes/backend/main.py

View on GitHub

```
data = request.get_json()
# Populates note properties according to the model,
# with the user ID as the key name.
note = Note(
   parent=ndb.Key(Note, claims['sub']),
    message=data['message'])
# Some providers do not provide one of these so either can be used.
note.friendly_id = claims.get('name', claims.get('email', 'Unknown'))
```

To write the newly created Note to Cloud Datastore, call the put() method on the note object.

Retrieving user data

To retrieve user data associated with a particular user ID, use the NDB query() method to search the database for notes in the same entity group. Entities in the same group, or_ancestor path_, share a common key name, which in this case is the user ID.

appengine/standard/firebase/firenotes/backend/main.py

View on GitHub

```
def query_database(user_id):
    """Fetches all notes associated with user_id.
    Notes are ordered them by date created, with most recent note added
    first."""
    ancestor_key = ndb.Key(Note, user_id)
    query = Note.query(ancestor=ancestor_key).order(-Note.created)
    notes = query.fetch()
    note_messages = []
    for note in notes:
        note_messages.append({
            'friendly_id': note.friendly_id,
            'message': note.message,
            'created': note.created
        })

    return note_messages
```

You can then fetch the query data and display the notes in the client:

appengine/standard/firebase/firenotes/frontend/main.js

View on GitHub

```
// Fetch notes from the backend.
function fetchNotes() {
  $.ajax(backendHostUrl + '/notes', {
    /* Set header for the XMLHttpRequest to get data from the web server
    associated with userIdToken */
    headers: {
      'Authorization': 'Bearer ' + userIdToken
    }
  }).then(function(data){
    $('#notes-container').empty();
    // Iterate over user data to display user's notes from database.
    data.forEach(function(note){
      $('#notes-container').append($('<p>').text(note.message));
    });
  });
}
```

Deploying your app

You have successfully integrated Firebase Authentication with your App Engine application. To see your application running in a live production environment:

1. Change the backend host URL in main.js to https://backend-dot-[PROJECT_ID].appspot.com. Replace [PROJECT_ID] with your project ID.
2. Deploy the application using the Cloud SDK command-line interface:
   ```
   gcloud app deploy backend/index.yaml frontend/app.yaml backend/app.yaml
   ```
3. View the application live at https://[PROJECT_ID].appspot.com.
   Note: Cloud Datastore indexes take a few minutes to update, so the application might not be fully functional immediately after deployment.

Cleaning up

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial, delete your App Engine project:

Deleting the project

The easiest way to eliminate billing is to delete the project you created for the tutorial.

To delete the project:

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an appspot.com URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

2. In the GCP Console, go to the **Projects** page.

   Go to the Projects page

3. In the project list, select the project you want to delete and click **Delete** *delete*.
4. In the dialog, type the project ID, and then click **Shut down** to delete the project.

What's next

- Set up Eclipse for development on App Engine.
- If you already have an existing user database for your app, check out how to create custom authentication tokens on Firebase.
- Try out other Google Cloud Platform features for yourself.

Conclusion

Here we learnt to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store.