# Assignment No 1

**Title: Implementation of Virtualization in Cloud Computing to Learn Virtualization Basics.**

**Requirement:** KVM, Linux: Any distro

**Theory:**

Kernel-based Virtual Machine (KVM) is a virtualization module in the Linux kernel that allows the kernel to function as a hypervisor. It was merged into the Linux kernel mainline in kernel version 2.6.20, which was released on February 5, 2007. KVM requires a processor with hardware virtualization extensions, such as Intel VT or AMD-V. KVM has also been ported to other operating systems such as FreeBSD and illumos in the form of loadable kernel modules.

KVM was originally designed for x86 processors and has since been ported to S/390, PowerPC, IA-64, and ARM.

KVM provides hardware-assisted virtualization for a wide variety of guest operating systems including Linux, BSD, Solaris, Windows, Haiku, ReactOS, Plan 9, AROS Research Operating System and OS X In addition, Android 2.2, GNU/Hurd (Debian K16), Minix 3.1.2a, Solaris 10 U3 and Darwin 8.0.1, together with other operating systems and some newer versions of these listed, are known to work with certain limitations.

Additionally, KVM provides paravirtualization support for Linux, OpenBSD, FreeBSD, NetBSD, Plan 9 and Windows guests using the VirtIO API. This includes a paravirtual Ethernet card, disk I/O controller, balloon device, and a VGA graphics interface using SPICE or VMware drivers.

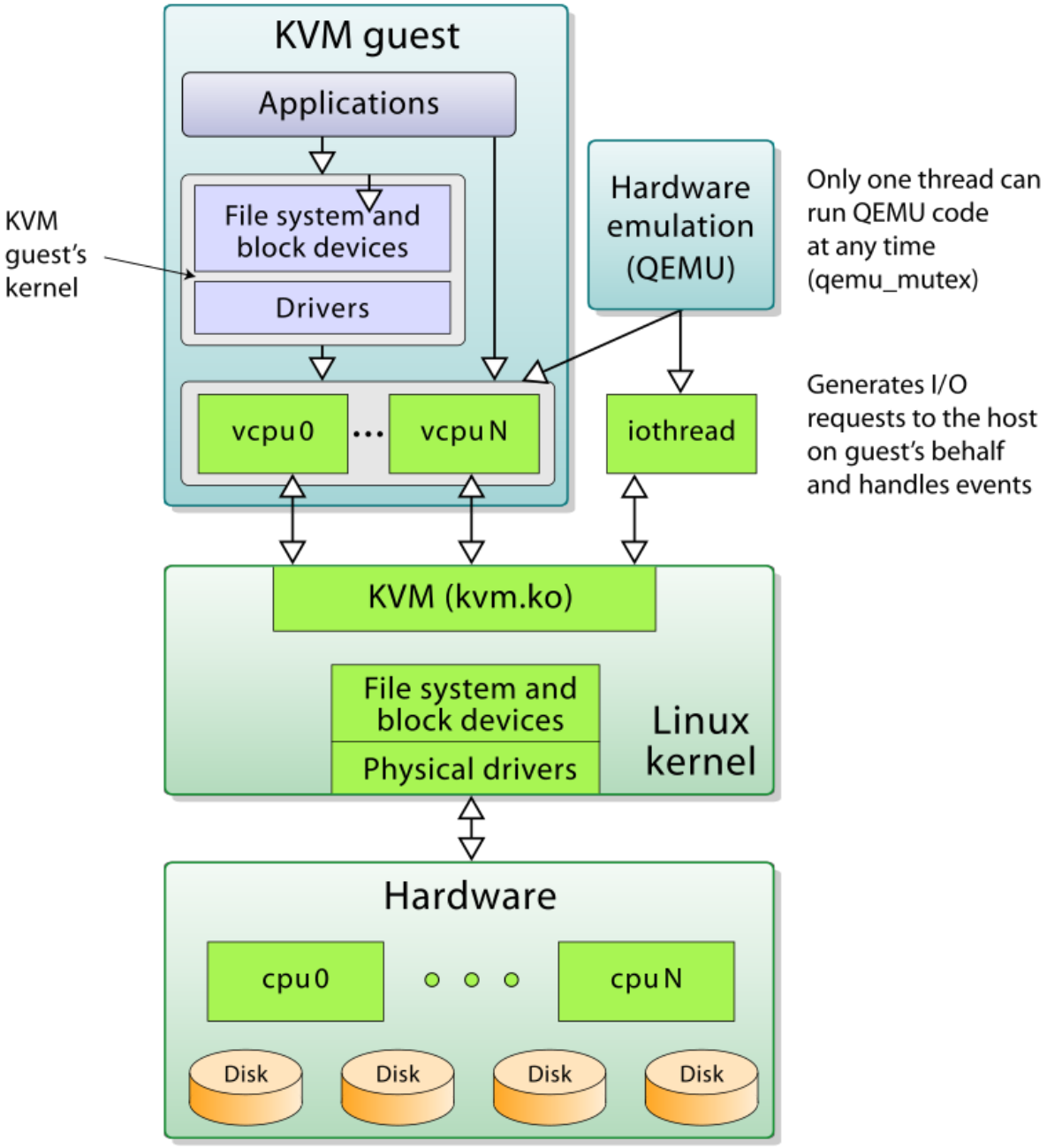
Internals
A high-level overview of the KVM/QEMU virtualization environment

By itself, KVM does not perform any emulation. Instead, it exposes the /dev/kvm interface, which a userspace host can then use to:

Set up the guest VM's address space. The host must also supply a firmware image (usually a custom BIOS when emulating PCs) that the guest can use to bootstrap into its main OS.
Feed the guest simulated I/O.
Map the guest's video display back onto the system host.

On Linux, QEMU versions 0.10.1 and later is one such userspace host. QEMU uses KVM when available to virtualize guests at near-native speeds, but otherwise falls back to software-only emulation.

Internally, KVM uses SeaBIOS as an open source implementation of a 16-bit x86 BIOS.




**Conclusion**: In this way, the basics of virtualization using KVM has completed.

# Assignment No 2

**Title: Study and implementation of infrastructure as Service using OpenStack.**

**Requirement:** Linux, Openstack Source code.

**Theory:**

OpenStack is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service (IaaS), whereby virtual servers and other resources are made available to customers, The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center. Users either manage it through a web-based dashboard, through command-line tools, or through RESTful web services. OpenStack began in 2010 as a joint project of Rackspace Hosting and NASA. As of 2016, it is managed by the OpenStack Foundation, a non-profit corporate entity established in September 2012. to promote OpenStack software and its community.More than 500 companies have joined the project.

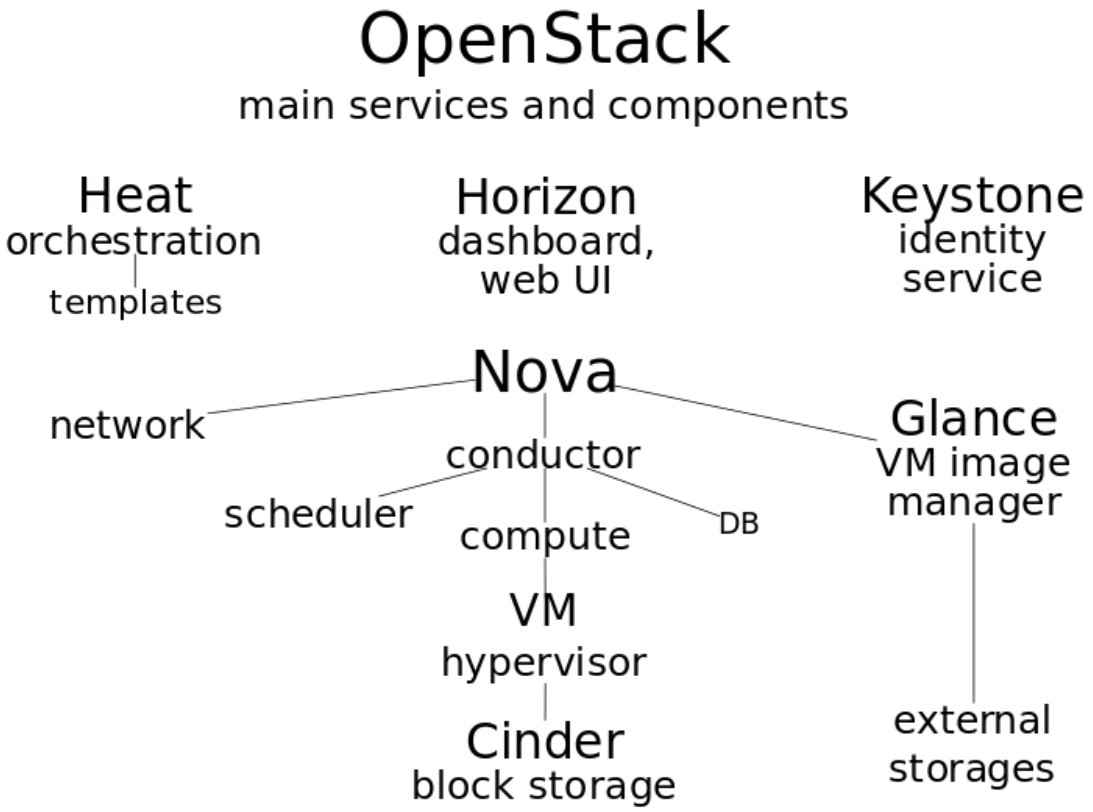
**OpenStack development**
The OpenStack community collaborates around a six-month, time-based release cycle with frequent development milestones.
During the planning phase of each release, the community would gather for an OpenStack Design Summit to facilitate developer working sessions and to assemble plans.These Design Summits would coincide with the OpenStack Summit conference.

Starting with the Pike development cycle the design meetup activity has been separated out into a separate Project Teams Gathering (PTG) event.This was done to avoid the developer distractions caused by presentations and customer meetings that were happening at the OpenStack Summit and to allow the design discussions to happen ahead of the start of the next cycle.

**Components**

OpenStack has a modular architecture with various code names for its components.

**Compute (Nova)**
OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations. KVM, VMware, and Xen are available choices for hypervisor technology (virtual machine monitor), together with Hyper-V and Linux container technology such as LXC.




**OpenStack main services**

It is written in Python and uses many external libraries such as Eventlet (for concurrent programming), Kombu (for AMQP communication), and SQLAlchemy (for database access).[61] Compute's architecture is designed to scale horizontally on standard hardware with no proprietary hardware or software requirements and provide the ability to integrate with legacy systems and third-party technologies.
Due to its widespread integration into enterprise-level infrastructures, monitoring OpenStack performance in general, and Nova performance in particular, scaling has become an increasingly important issue. Monitoring end-to-end performance requires tracking metrics from Nova, Keystone, Neutron, Cinder, Swift and other services, in addition to monitoring RabbitMQ which is used by OpenStack services for message passing.All these services generate their own log files, which, especially in enterprise-level infrastructures, also should be monitored.

**Networking (Neutron)**
OpenStack Networking (Neutron) is a system for managing networks and IP addresses. OpenStack Networking ensures the network is not a bottleneck or limiting factor in a cloud deployment,[citation needed] and gives users self-service ability, even over network configurations.

OpenStack Networking provides networking models for different applications or user groups. Standard models include flat networks or VLANs that separate servers and traffic. OpenStack Networking manages IP addresses, allowing for dedicated static IP addresses or DHCP. Floating IP addresses let traffic be dynamically rerouted to any resources in the IT infrastructure, so users can redirect traffic during maintenance or in case of a failure.

Users can create their own networks, control traffic, and connect servers and devices to one or more networks. Administrators can use software-defined networking (SDN) technologies like OpenFlow to support high levels of multi-tenancy and massive scale. OpenStack networking provides an extension framework that can deploy and manage additional network services—such as intrusion detection systems (IDS), load balancing, firewalls, and virtual private networks (VPN).

**Block storage (Cinder)**

OpenStack Block Storage (Cinder) provides persistent block-level storage devices for use with OpenStack compute instances. The block storage system manages the creation, attaching and detaching of the block devices to servers. Block storage volumes are fully integrated into OpenStack Compute and the Dashboard allowing for cloud users to manage their own storage needs. In addition to local Linux server storage, it can use storage platforms including Ceph, CloudByte, Coraid, EMC (ScaleIO, VMAX, VNX and XtremIO), GlusterFS, Hitachi Data Systems, IBM Storage (IBM DS8000, Storwize family, SAN Volume Controller, XIV Storage System, and GPFS), Linux LIO, NetApp, Nexenta, Nimble Storage, Scality, SolidFire, HP (StoreVirtual and 3PAR StoreServ families), INFINIDAT (InfiniBox) and Pure Storage. Block storage is appropriate for performance sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block level storage. Snapshot management provides powerful functionality for backing up data stored on block storage volumes. Snapshots can be restored or used to create a new block storage volume.

**Identity (Keystone)**

OpenStack Identity (Keystone) provides a central directory of users mapped to the OpenStack services they can access. It acts as a common authentication system across the cloud operating system and can integrate with existing backend directory services like LDAP. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style (i.e. Amazon Web Services) logins. Additionally, the catalog provides a queryable list of all of the services deployed in an OpenStack cloud in a single registry. Users and third-party tools can programmatically determine which resources they can access.

**Image (Glance)**

OpenStack Image (Glance) provides discovery, registration, and delivery services for disk and server images. Stored images can be used as a template. It can also be used to store and catalog an unlimited number of backups. The Image Service can store disk and server images in a variety of back-ends, including Swift. The Image Service API provides a standard REST interface for querying information about disk images and lets clients stream the images to new servers. Glance adds many enhancements to existing legacy infrastructures. For example, if integrated with VMware, Glance introduces advanced features to the vSphere family such as vMotion, high availability and dynamic resource scheduling (DRS). vMotion is the live migration of a running VM, from one physical server to another, without service interruption. Thus, it enables a dynamic and automated self-optimizing datacenter, allowing

hardware maintenance for the underperforming servers without downtimes. Other OpenStack modules that need to interact with Images, for example Heat, must communicate with the images metadata through Glance. Also, Nova can present information about the images, and configure a variation on an image to produce an instance. However, Glance is the only module that can add, delete, share, or duplicate images.

**Object storage (Swift)**

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used. In August 2009, Rackspace started the development of the precursor to OpenStack Object Storage, as a complete replacement for the Cloud Files product. The initial development team consisted of nine developers. SwiftStack, an object storage software company, is currently the leading developer for Swift with significant contributions from HP, Red Hat, NTT, NEC, IBM and more.

**Conclusion**: In this way, study of implementation of Openstack has completed.

# Assignment No 3

**Title: 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.**

**Requirement:** AWS Account Access.

**Theory:**

What Is Amazon EC2?

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

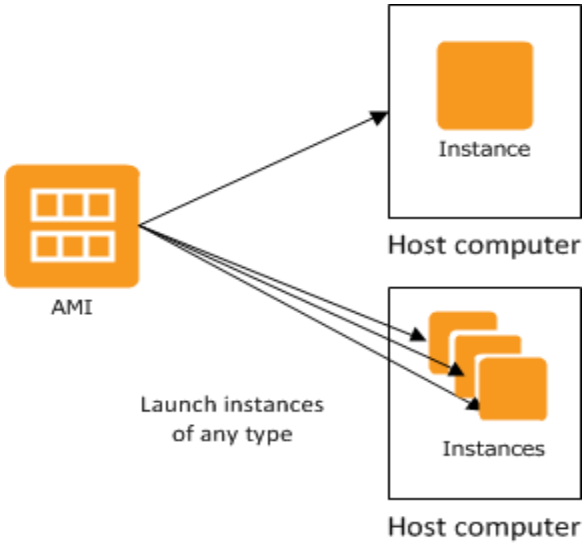
**Features of Amazon EC2**

Amazon EC2 provides the following features:

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

## Instances and AMIs

An *Amazon Machine Image (AMI)* is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, you launch an *instance*, which is a copy of the AMI running as a virtual server in the cloud. You can launch multiple instances of an AMI, as shown in the following figure.Your instances keep running until you stop or terminate them, or until they fail. If an instance fails, you can launch a new one from the AMI.




## Instances

An instance is a virtual server in the cloud. It's configuration at launch is a copy of the AMI that you specified when you launched the instance. You can launch different types of instances from a single AMI. An *instance type* essentially determines the hardware of the host computer used for your instance. Each instance type offers different compute and memory capabilities. Select an instance type based on the amount of memory and computing power that you need for the application or software that you plan to run on the instance. After you launch an instance, it looks like a traditional host, and you can interact with it as you would any computer. You have complete control of your instances; you can use sudo to run commands that require root privileges. Your AWS account has a limit on the number of instances that you can have running.

## Storage for Your Instance

The root device for your instance contains the image used to boot the instance. Your instance may include local storage volumes, known as instance store volumes, which you can configure at launch time with block device mapping. For more information, see Block Device Mapping. After these volumes have been added to and mapped on your instance, they are available for you to mount and use. If your instance fails, or if your instance is stopped or terminated, the data on these volumes is lost; therefore, these volumes are best used for temporary data. To keep important data safe, you should use a replication strategy across multiple instances, or store your persistent data in Amazon S3 or Amazon EBS volumes.

**Conclusion**: In this way, case study of AWS cloud has completed.

# Assignment No 4

**Title: Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications**

**Requirement:** MS Azure Account access

**Theory:**

Microsoft Azure, formerly Windows Azure, is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems. Azure was announced in October 2008, started with codename "Project Red Dog" and released on February 1, 2010, as "Windows Azure" before being renamed "Microsoft Azure".

**Services:**

Microsoft lists over 600 Azure services, some of which some are covered below:

**Compute**

- Virtual machines, infrastructure as a service (IaaS) allowing users to launch general-purpose Microsoft Windows and Linux virtual machines, as well as preconfigured machine images for popular software packages.
- App services, platform as a service (PaaS) environment letting developers easily publish and manage websites.
- Websites, high density hosting[non sequitur] of websites allows developers to build sites using ASP.NET, PHP, Node.js, or Python and can be deployed using FTP, Git, Mercurial, Team Foundation Server or uploaded through the user portal. This feature was announced in preview form in June 2012 at the Meet Microsoft Azure event.
- Customers can create websites in PHP, ASP.NET, Node.js, or Python, or select from several open source applications from a gallery to deploy. This comprises one aspect of the platform as a service (PaaS) offerings for the Microsoft Azure Platform. It was renamed to Web Apps in April 2015.
- WebJobs, applications that can be deployed to an App Service environment to implement background processing that can be invoked on a schedule, on demand, or run continuously. The Blob, Table and Queue services can be used to communicate between WebApps and WebJobs and to provide state.

**Mobile services**

- Mobile Engagement collects real-time analytics that highlight users' behavior. It also provides push notifications to mobile devices.
- HockeyApp can be used to develop, distribute, and beta-test mobile apps.

**Storage services**

- Storage Services provides REST and SDK APIs for storing and accessing data on the cloud.
- Table Service lets programs store structured text in partitioned collections of entities that are accessed by partition key and primary key. It's a NoSQL non-relational database.
- Blob Service allows programs to store unstructured text and binary data as blobs that can be accessed by a HTTP(S) path. Blob service also provides security mechanisms to control access to data.
- Queue Service lets programs communicate asynchronously by message using queues.
- File Service allows storing and access of data on the cloud using the REST APIs or the SMB protocol.

**Data management:**

- Azure Search provides text search and a subset of OData's structured filters using REST or SDK APIs.
- Cosmos DB is a NoSQL database service that implements a subset of the SQL SELECT statement on JSON documents.
- Redis Cache is a managed implementation of Redis.
- StorSimple manages storage tasks between on-premises devices and cloud storage.
- SQL Database, formerly known as SQL Azure Database, works to create, scale and extend applications into the cloud using Microsoft SQL Server technology. It also integrates with Active Directory and Microsoft System Center and Hadoop.
- SQL Data Warehouse is a data warehousing service designed to handle computational and data intensive queries on datasets exceeding 1TB.
- Azure Data Factory, is a data integration service that allows creation of data-driven workflows in the cloud for orchestrating and automating data movement and data transformation.
- Azure Data Lake is a scalable data storage and analytic service for big-data analytics workloads that require developers to run massively parallel queries.
- Azure HDInsight is a big data relevant service, that deploys Hortonworks Hadoop on Microsoft Azure, and supports the creation of Hadoop clusters using Linux with Ubuntu.
- Azure Stream Analytics is a serverless scalable event processing engine that enables users to develop and run real-time analytics on multiple streams of data from sources such as devices, sensors, web sites, social media, and other applications.

**Design**

Microsoft Azure uses a specialized operating system, called Microsoft Azure, to run its "fabric layer": a cluster hosted at Microsoft's data centers that manages computing and storage resources of the computers and provisions the resources (or a subset of them) to applications running on top of Microsoft Azure. Microsoft Azure has been described as a "cloud layer" on top of a number of Windows Server systems, which use Windows Server 2008 and a customized version of Hyper-V, known as the Microsoft Azure Hypervisor to provide virtualization of services. Scaling and reliability are controlled by the Microsoft Azure Fabric Controller, which ensures the services and environment do not fail if one or more of the servers fails within the Microsoft data center, and which also provides the management of the user's Web application such as memory allocation and load balancing.

**Conclusion**: In this way, case study of Microsoft Azure cloud has completed.

# Assignment No 5

**Write a program for Webfeed using PHP and HTML.**

**Requirement:** HTTP Web Server, PHP, HTML

**Theory:**

RSS (Rich Site Summary) is a format which is used in many websites which allow web publisher to syndicates their latest posts or data automatically. It also benefits people who want to receive latest posts updates from their favorite websites. There is another method which allows the user to stay updated is bookmarking. But they need to manually go to websites on the timely basis and check what new have been added.

RSS (originally RDF Site Summary; later, two competing approaches emerged, which used the backronyms Rich Site Summary and Really Simple Syndication respectively) is a type of web feed which allows users and applications to access updates to online content in a standardized, computer-readable format. These feeds can, for example, allow a user to keep track of many different websites in a single news aggregator. The news aggregator will automatically check the RSS feed for new content, allowing the content to be automatically passed from website to website or from website to user. This passing of content is called web syndication. Websites usually use RSS feeds to publish frequently updated information, such as blog entries, news headlines, or episodes of audio and video series. RSS is also used to distribute podcasts. An RSS document (called "feed", "web feed",or "channel") includes full or summarized text, and metadata, like publishing date and author's name.

A standard XML file format ensures compatibility with many different machines/programs. RSS feeds also benefit users who want to receive timely updates from favourite websites or to aggregate data from many sites.

Subscribing to a website RSS removes the need for the user to manually check the website for new content. Instead, their browser constantly monitors the site and informs the user of any updates. The browser can also be commanded to automatically download the new data for the user.

RSS feed data is presented to users using software called a news aggregator. This aggregator can be built into a website, installed on a desktop computer, or installed on a mobile device. Users subscribe to feeds either by entering a feed's URI into the reader or by clicking on the browser's feed icon. The RSS reader checks the user's feeds regularly for new information and can automatically download it, if that function is enabled. The reader also provides a user interface.

**Program:**

**HTML Program**

```
<html>
  <head>

        <script>
        function showRSS(str) {
        if (str.length == 0) {
        document.getElementById("output").innerHTML = "";
        return;
        }

        if (window.XMLHttpRequest) {
        xmlhttp = new XMLHttpRequest();
        }else {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                document.getElementById("output").innerHTML = xmlhttp.responseText;
        }
        }

        xmlhttp.open("GET","rss.php?q="+str,true);
        xmlhttp.send();
        }
        </script>

  </head>

  <body>
        <p>Please Select an option to get RSS:</p>

        <form>
        <select onchange = "showRSS(this.value)">
        <option value = "">Select an RSS-feed:</option>
        <option value = "cmd">ComedyFlavors.com</option>
        <option value = "bbc">BBC News</option>
        <option value = "pc">TryCatch Classes</option>
        </select>
        </form>
        <br>
        <div id = "output">RSS-feeds</div>
```

```
  </body>
</html>
```

**PHP Code:**

```
<?php
error_reporting(0);
  $q = $_GET["q"];

  if($q == "cmd") {
        $xml = ("http://www.comedyflavors.com/feed/");
  }elseif($q == "bbc") {
        $xml = ("http://newsrss.bbc.co.uk/rss/newsonline_world_edition/americas/rss.xml");
  }elseif($q = "pcw"){
        $xml = ("http://www.trycatchclasses.com/feed/");
  }

  $xmlDoc = new DOMDocument();
  $xmlDoc->load($xml);

  $channel = $xmlDoc->getElementsByTagName('channel')->item(0);

  $channel_title = $channel->getElementsByTagName('title')
  ->item(0)->childNodes->item(0)->nodeValue;

  $channel_link = $channel->getElementsByTagName('link')
  ->item(0)->childNodes->item(0)->nodeValue;

  $channel_desc = $channel->getElementsByTagName('description')
  ->item(0)->childNodes->item(0)->nodeValue;

  echo("<p><a href = '" . $channel_link . "'>" .
        $channel_title . "</a>");
  echo("<br>");
  echo($channel_desc . "</p>");

  $x = $xmlDoc->getElementsByTagName('item');

  for ($i = 0; $i<=2; $i++) {
        $item_title = $x->item($i)->getElementsByTagName('title')
        ->item(0)->childNodes->item(0)->nodeValue;

        $item_link = $x->item($i)->getElementsByTagName('link')
        ->item(0)->childNodes->item(0)->nodeValue;
```

```
        $item_desc = $x->item($i)->getElementsByTagName('description')
        ->item(0)->childNodes->item(0)->nodeValue;

        echo ("<p><a href = '" . $item_link . "'>" .
        $item_title . "</a>");
        echo ("<br>");
        echo ($item_desc . "</p>");
    }
?>
```

**Conclusion**: In this way, we have developed webfeed using PHP and HTML.

.

# Assignment No 6

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

**Requirement:** ownCloud on linux

## Theory:

ownCloud is a suite of client–server software for creating and using file hosting services. ownCloud functionally has similarities to the widely used Dropbox. The primary functional difference between ownCloud and Dropbox is that ownCloud does not offer data centre capacity to host stored files. The Server Edition of ownCloud is free and open-source, thereby allowing anyone to install and operate it without charge on their own private server.
ownCloud supports extensions that allow it to work like Google Drive, with online document editing, calendar and contact synchronization, and more. Its openness avoids enforced quotas on storage space or the number of connected clients, instead having hard limits (like on storage space or number of users) defined only by the physical capabilities of the server.

**Overview**
**Design**

For desktop machines to synchronize files with their ownCloud server, desktop clients are available for PCs running Windows, macOS, FreeBSD or Linux. Mobile clients exist for iOS and Android devices. Files and other data (such as calendars, contacts or bookmarks) can also be accessed, managed, and uploaded using a web browser without any additional software. Any updates to the file system are pushed to all computers and mobile devices connected to a user's account. Encryption of files may be enforced by the server administrator. The ownCloud server is written in the PHP and JavaScript scripting languages. For remote access, it employs sabre/dav, an open-source WebDAV server. ownCloud is designed to work with several database management systems, including SQLite, MariaDB, MySQL, Oracle Database, and PostgreSQL

## Features

owncloud is a software only product. owncloud does not offer off-premises storage capacity. This is in contrast to the likes of Dropbox that offers both software and off-premises storage capacity. owncloud storage capacity has to be provided on user owned devices. ownCloud files are stored in conventional directory structures, and can be accessed via WebDAV if necessary. User files are encrypted both at rest and during transit. ownCloud can synchronise with local clients running Windows (Windows XP, Vista, 7 and 8), macOS (10.6 or later), or various Linux distributions. ownCloud users can manage calendars (CalDAV), contacts (CardDAV) scheduled tasks and streaming media (Ampache) from within the platform. From the administration perspective, ownCloud permits user and group administration (via OpenID or LDAP). Content can be shared by defining granular read/write permissions between users and/or groups. Alternatively, ownCloud users can create public URLs when sharing files. Logging of

file-related actions is available in the Enterprise and Education service offerings. Furthermore, users can interact with the browser-based ODF-format word processor, bookmarking service, URL shortening suite, gallery, RSS feed reader and document viewer tools from within ownCloud. For additional extensibility, ownCloud can be augmented with "one-click" applications and connection to Dropbox, Google Drive and Amazon S3. All ownCloud clients (Desktop, iOS, Android) support the OAuth 2 standard for Client Authentication.

**Program:**

```
<?php

// This works only with MySQL as backend DB

// Define user.txt file
// Field order:  USERNAME | FULL NAME | EMAIL ADDRESS | GROUPS | GROUPADMIN
| QUOTA | LANGUAGE
define('FIELDS_',       ';'); // Delimiter of fields in the .txt file
define('GROUPS_',      ':'); // Delimiter of the groups a user is member of

// Define curl login options

define('DOMAIN_',            'http://your.domain.com/owncloud_path');  //
Give the whole basepath to your OC installation, e.g.
http://domain.com/owncloud

// Set some default values for the user. They are applied only if the
user.txt file doesn't provide any other option. Set value to '-1' to
completely ignore.
define('LANGUAGE_',       'en');

// Set Email Parameters
define('SENDER_',      'myemail@mydomain.com'); // The sender of the email
define('SUBJECT_',      'ownCloud Login Data');
define('MESSAGE_',      'Hi there

Your account for the ' . DOMAIN_ . ' ownCloud service has been setup.

You can access it with the following data:

Adress: ' . DOMAIN_ . '
Username: [username]
Password: [password]


Best regards

Webmaster');

// Require necessary files
require_once('config/config.php');
```

```php
// Get more Constants
define('DB_HOST_',    $CONFIG['dbhost']);
define('DB_NAME_',    $CONFIG['dbname']);
define('DB_USER_',    $CONFIG['dbuser']);
define('DB_PASSWD_',    $CONFIG['dbpassword']);
define('DB_PREFIX_',    $CONFIG['dbtableprefix']);


// Set DSN
define('DSN_', 'mysql:host=' . DB_HOST_ . ';dbname=' . DB_NAME_);


// Initialize objects
$db = new DB;

// Read text file into array
$users = file('user.txt');


// Loop through arrey
foreach($users as $val) {
   // Unset $err variable
   unset($err);

   // Explode user line into fields
   $data = explode(FIELDS_, $val);
   $username =   trim($data[0]);
   $fullname =   trim($data[1]);
   $email =   trim($data[2]);
   $group =   $data[3];
   $groupadmin =   trim($data[4]);
   $quota = trim($data[5]);
   $language =   trim($data[6]);

   // Check if username is valid
   $length = strlen($username);
   if($length >= 1) {
      $usercheck = validChars($username);
      if( $usercheck == 0 ) {
         $err .= ' Invalid characters in username.';
      }
   } else {
      $err .= ' Username is too short.';
   }
```

```
   // Check if username already exists
   $stmt = $db->prepare('SELECT uid FROM ' . DB_PREFIX_ .'users WHERE uid =
?');
   $stmt->execute(array($username));
   $count = $stmt->rowCount();
   if( $count >= 1 ) {
      $err .= ' User already exists.';
   }

   // Check if group names are valid. Also trim and put them into new array
   $length = strlen($group);
   if($length >= 1) {
         $groups = explode(GROUPS_, $group);
      unset($group);
      foreach ($groups as $tmp) {
         $tmp = trim($tmp);
         $groupcheck = validChars($tmp);
         if ( $groupcheck == 0 ) {
            $err .= ' Invalid characters in group name.';
         }
         $group[] = $tmp;
      }
   }

   // Check if email is valid
   if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $err .= ' Invalid email address.';
   }

   // Check if there was an error, if not, add data
   if(!isset($err)) {

      // Generate random password and hash it
      $passwd = randString( 8 );
      //$hash = $hasher->HashPassword($passwd);

      //create user
      createUser($db, $username, $fullname, $passwd);

      // add groups
      if(is_array($group)) {
         // Check if group already exists
         foreach ($group as $tmpg) {
```

```
            if(!checkGroupexist($db, $tmpg)) {
               $stmt = $db->prepare('INSERT INTO ' . DB_PREFIX_ .'groups
SET gid = ?');
              $stmt->execute(array($tmpg));
            }
            addGroups($db, $tmpg, $username);
         }
      } else {
         addGroups($db, $group, $username);
      }

      // Set groupAdmin
      if(!empty($groupadmin)) {
            if(!checkGroupexist($db, $groupadmin)) {
               $err .= ' Invalid Group Admin.';
            }
            $stmt = $db->prepare('INSERT INTO ' . DB_PREFIX_ .'group_admin
SET gid = ?, uid=? ');
            $stmt->execute(array($groupadmin,$username));
      }

      // Set Email
      setConfig ( $db, $username, 'settings', 'email', $email );

      // Set Quota
      if(!empty($quota)) {
         setConfig ( $db, $username, 'files', 'quota', $quota );
      } else if (strcasecmp($quota,'unlimited')==0) {
         setConfig ( $db, $username, 'files', 'quota', 'none' );
      }

      // Set Language
      if( LANGUAGE_ != '-1' ) {
         if( $language == '') {
            $language = LANGUAGE_;
         }
         setConfig ( $db, $username, 'core', 'lang', $language );
      }

      // Send Email
      $message = MESSAGE_;
      $message = str_replace('[username]', $username, $message);
      $message = str_replace('[password]', $passwd, $message);
      mail_utf8($email, SUBJECT_, $message, 'FROM: ' . SENDER_);
```

```
    }

    // Output report
    if(isset($err)) {
       $fail .= $username . ' - ' . $err . "<br>\n";
    } else {
       $success .= $username . "<br>\n";
    }
}

if (!empty($fail)) {
echo "FAILED:<br>\n" . $fail;
echo "<br>\n";
} else {
echo "ADDED:<br>\n" . $success;
}

function validChars( $check )
{
    if( preg_match( '/[^a-zA-Z0-9 _\.@\-]/', $check ))
    {
       // There are illegal chars
       $result = '0';
    } else {
       // All chars are valid
       $result = '1';
    }
    return($result);
}

function randString( $length ) {
    $chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-!,.;:*()";
    $size = strlen( $chars );
    $result = '';
    for( $i = 0; $i < $length; $i++ ) {
       $result .= $chars[ rand( 0, $size - 1 ) ];
    }
    return $result;
}

function setConfig( $db, $userid, $appid, $configkey, $configvalue ) {
    $stmt = $db->prepare('INSERT INTO ' . DB_PREFIX_ .'preferences SET
          userid = ?,
```

```
          appid = ?,
          configkey = ?,
          configvalue = ?
       ');
   $stmt->execute(array(
          $userid,
          $appid,
          $configkey,
          $configvalue
       ));
}


function mail_utf8($to, $subject = '(No subject)', $message = '', $header =
'') {
        $message = utf8_encode($message);
        $header_ = 'MIME-Version: 1.0' . "\r\n" . 'Content-type:
text/plain; charset=UTF-8' . "\r\n";
        $subject = mb_encode_mimeheader( $subject, "UTF-8", "Q" );
        mail($to, $subject, $message, $header_ . $header);
}


function createUser($db, $username, $fullname, $passwd) {
   $stmt = $db->prepare('INSERT INTO ' . DB_PREFIX_ .'users SET
          uid = ?,
          displayname = ?,
          password = sha1(?)
       ');
   $stmt->execute(array(
          $username,
          $fullname,
          $passwd
       ));
}

// Check if group already exists
function checkGroupexist($db, $group) {
       $r = false;
      $stmt = $db->prepare('SELECT gid FROM ' . DB_PREFIX_ .'groups WHERE
gid = ? ');
      $stmt->execute(array($group));
      $count = $stmt->rowCount();
      if( $count >= 1 ) {
```

```
            $r = true;
        }
        return $r;
}

function addGroups($db, $group,$username) {
    $stmt = $db->prepare('INSERT INTO ' . DB_PREFIX_ .'group_user SET
            gid = ?,
            uid = ?
        ');
    $stmt->execute(array(
            $group,
            $username
        ));
}

class DB extends PDO {
    private static $_self;

    public function __construct() {
        self::$_self = $this;
        parent::__construct(DSN_, DB_USER_, DB_PASSWD_);
        $this->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY,TRUE);
        $this->fetch_mode = 'FETCH_ASSOC';
    }

    public static function getInstance() {
        if(is_null(self::$_self))
            new DB;
        return self::$_self;
    }
}
?>
```

**Conclusion:** In this way, users can be add, remove and managed in owncloud.