



MGM's
**Jawaharlal Nehru Engineering College
Aurangabad**

MGM University, Aurangabad

Department of Computer Science & Engineering

LAB MANUAL

Program (UG/PG) : UG

Year : Third Year

Semester : V

Course Code : 20UCS513L

Course Title : Basics of Cloud Computing Lab

Prepared By : Dr J.D.Pagare

Department of Computer Science & Engineering
2022-23

FOREWORD

It is my great pleasure to present this laboratory manual for **Third Year Computer Science and engineering** students for the subject of Basics Cloud Computing. As a student, many of you may be wondering about the subject and exactly that has been tried through this manual.

As you may be aware that MGM has already been awarded with ISO 9000 certification and it is our aim to technically equip students taking the advantage of the procedural aspects of ISO 9000 Certification.

Faculty members are also advised that covering these aspects in initial stage itself will relieve them in future as much of the load will be taken care by the enthusiastic energies of the students once they are conceptually clear.

Dr. H. H. Shinde

Principal

LABORATORY MANUAL CONTENTS

This manual is intended for Third Year COMPUTER SCIENCE & ENGINEERING students for the subject of **Basics of Cloud Computing**. This manual typically contains practical/Lab Sessions related cloud computing PaaS, SaaS, IaaS, etc covering various aspects related to the subject to enhance understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions.

Dr. V.B. Musande
HOD, CSE

Dr. J D Pagare
CSE Dept

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.
2. All the students should sit according to their roll numbers starting from their left to right.
3. All the students are supposed to enter the terminal number in the log book.
4. Do not change the terminal on which you are working.
5. All the students are expected to get at least the algorithm of the program/concept to be implemented.
6. Strictly observe the instructions given by the teacher/Lab Instructor.

Instruction for Laboratory Teachers::

1. Submission related to whatever lab work has been completed should be done during the next lab session. The immediate arrangements for printouts related to submission on the day of practical assignments.
2. Students should be taught for taking the printouts under the observation of lab teacher.
3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

MGM's



Jawaharlal Nehru Engineering College, Aurangabad

Department of Computer Science and Engineering

Vision of CSE Department

To develop computer engineers with necessary analytical ability and human values who can creatively design, implement a wide spectrum of computer systems for welfare of the society.

Mission of the CSE Department:

Preparing graduates to work on multidisciplinary platforms associated with their professional Position both independently and in a team environment. Preparing graduates for higher education and research in computer science and engineering enabling them to develop systems for society Development.

Programme Educational Objectives

Graduates will be able to

- I.** To analyze, design and provide optimal solution for Computer Science & Engineering and multidisciplinary problems.
- II.** To pursue higher studies and research by applying knowledge of mathematics and fundamentals of computer science.
- III.** To exhibit professionalism, communication skills and adapt to current trends by engaging in lifelong learning.

Programme Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the broadest context of technological change.

SUBJECT INDEX

Sr. No.	Title	Page No.
1	Introduction to cloud computing.	9
2	Creating a Warehouse Application in SalesForce.com.	15
3	Creating an Application in SalesForce.com using Apex programming Language.	17
4	Implementation of SOAP Web services in C#/JAVA Applications.	19
5	Implementation of Para-Virtualization using VM Ware's Workstation/ Oracle's Virtual Box and Guest O.S.	27
6	Install a C compiler in the virtual machine and execute a sample program.	37
7	Create a Cloud Storage bucket using Amazon Simple Storage Service (Amazon S3).	41
8	Working with AWS CLI	50
9	Creating Azure Container Instances	59
10	Case Study: PAAS(Facebook, Google App Engine)	60

LABORATORY OUTCOMES

The practical/exercises in this section are psychomotor domain Learning Outcomes (i.e., subcomponents of the COs), to be developed and assessed to lead to the attainment of the competency.

CO-1: Identify various Cloud Services.

CO2 – Implement Virtualization.

CO-3: Demonstrate the concept of Saas.

CO-4: Demonstrate the concept of PaaS.

Experiment No. 1

Aim: To study in detail about cloud computing.

Theory:

The term *cloud* has been used historically as a metaphor for the Internet. This usage was originally derived from its common depiction in network diagrams as an outline of a cloud, used to represent the transport of data across carrier backbones (which owned the cloud) to an endpoint location on the other side of the cloud. This concept dates back as early as 1961, when Professor John McCarthy suggested that computer time-sharing technology might lead to a future where computing power and even specific applications might be sold through a utility-type business model. ¹ This idea became very popular in the late 1960s, but by the mid-1970s the idea faded away when it became clear that the IT-related technologies of the day were unable to sustain such a futuristic computing model. However, since the turn of the millennium, the concept has been revitalized. It was during this time of revitalization that the term *cloud computing* began to emerge in technology circles. Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers.

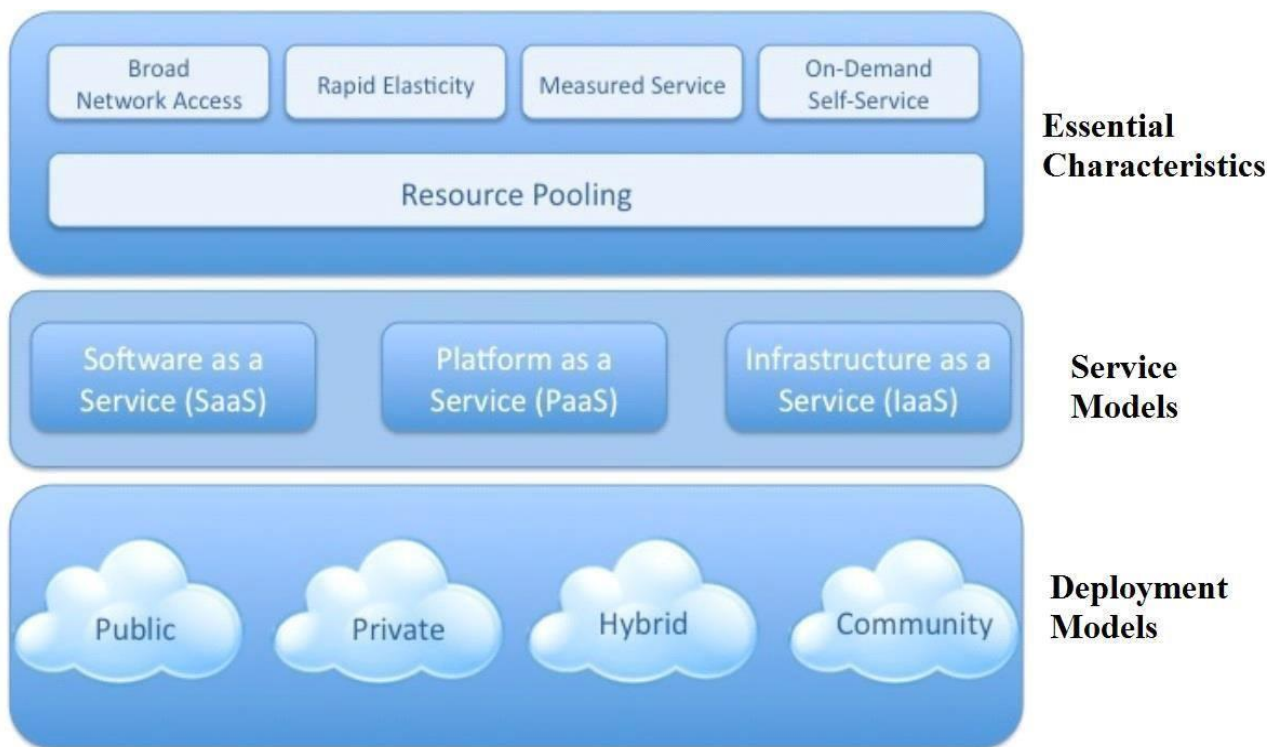
When you store your photos online instead of on your home computer, or use webmail or a social networking site, you are using a cloud computing service. If you are in an organization, and you want to use, for example, an online invoicing service instead of updating the in-house one you have been using for many years, that online invoicing service is a — cloud computing service. Cloud computing is the delivery of computing services over the Internet. Cloud services, Allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere. Cloud

computing provides a shared pool of resources, including data storage space, networks, Computer processing power, and specialized corporate and user applications.

Architecture

Cloud Service Models

- Cloud Deployment Models
- Essential Characteristics of Cloud Computing



NIST Visual Model of Cloud Computing Definition

Cloud Service Models

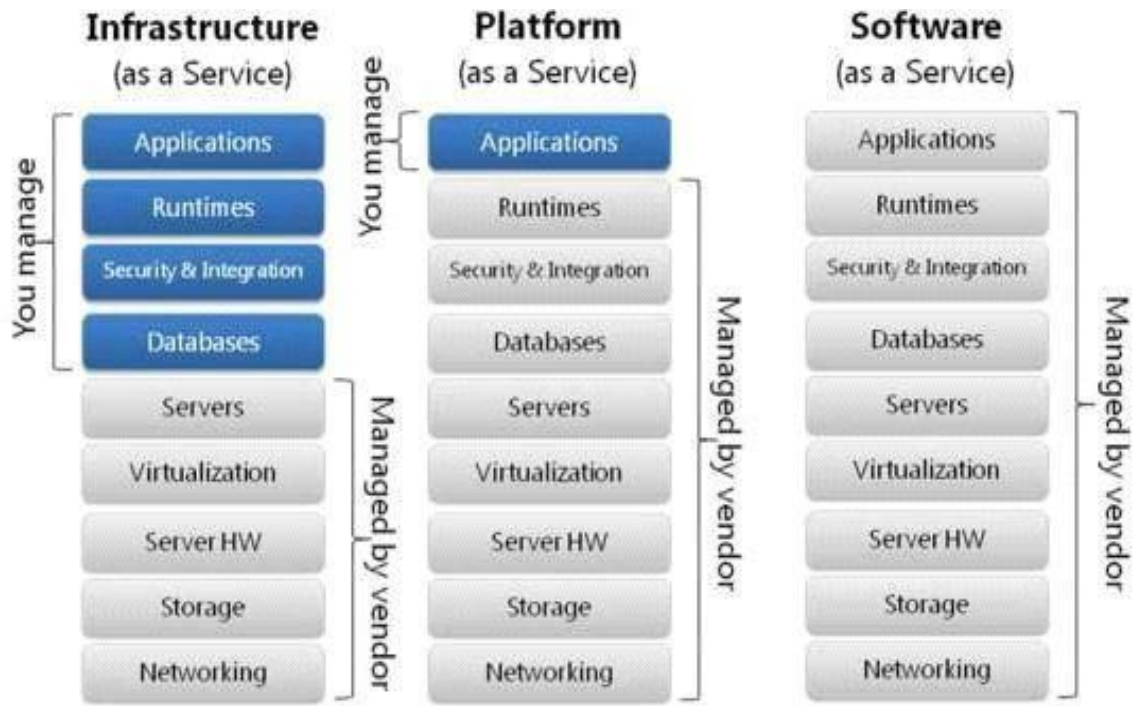
- Cloud Software as a Service (SaaS)
- Cloud Platform as a Service (PaaS)
- Cloud Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS):--

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.
- Consumer is able to deploy and run arbitrary software, which can include operating systems and applications.
- The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems; storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

Platform as a Service (PaaS):--

- The capability provided to the consumer is to deploy onto the cloud infrastructure consumer created or acquired applications created using programming languages and tools supported by the provider.
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.



Software as a Service (SaaS):--

- The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure.
- The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email).
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

	Amazon	Google	Salesforce	Customer Implications
Software as Service				+ Application logic, platform and infrastructure abstracted + Significant reduction in effort to deploy, run and manage - Apps can be configured but may not meet highly customized requirements
Platform as Service				+ Platform & infrastructure abstracted + Custom apps can be built order of magnitude more quickly and cheaply - Custom apps still need to be supported and managed
Infrastructure as Service				+ Physical infrastructure abstracted + Can be scaled up and down as needed - Needs to be provisioned/managed - Higher levels of stack still need to be managed, maintained and supported

Cloud Deployment Models:

- Public
- Private
- Community Cloud
- Hybrid Cloud

- **Public Cloud:** The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

- **Private Cloud:** The cloud infrastructure is operated solely for a single organization. It may be managed by the organization or a third party, and may exist on-premises or off-premises.

- **Community Cloud:** The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, or compliance considerations). It may be managed by the organizations or a third party and may exist on-premises or off-premises.

- **Hybrid Cloud:** The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or

Proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

ESSENTIAL CHARACTERISTICS:--

- **On-demand self-service:--** A consumer can unilaterally provision computing capabilities such as server time and network storage as needed automatically, without requiring human interaction with a service provider.
- **Broad network access:--** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client

platforms (e.g., mobile phones, laptops, and PDAs) as well as other traditional or cloud based software services.

- **Resource pooling:**--The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
- **Rapid elasticity:**--Capabilities can be rapidly and elastically provisioned in some cases automatically - to quickly scale out; and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured service:**--Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported - providing transparency for both the provider and consumer of the service.

Conclusion:

Thus we have studied in detail about overview of cloud computing

Experiment No. 2

Aim: Creating a Warehouse Application in SalesForce.com's Force.com.

Theory:

Steps to create an application in Force.com by declarative model

→ → → →
Step 1: Click on Setup → Create → Objects → New custom object
Label: MySale

Pular Label: MySales

Object Name: MySale

Record Name: MySale Description

Data Type: Text

→
Click on Save.

→
Step 2: Under MySale Go to Custom Field and Relationships → Click on New Custom Field

Creating 1st Field:--

→ →
select Data type as Auto Number → next
→ → →
Enter the details → Field Label: PROD_ID → Display Format: MYS-{0000}
→ → →
Starting Number: 1001 → Field Name: PRODID → Next → Save & New

Creating 2nd Field:--

→ →
select Data type as Date → next
→ → →
Enter the details → Field Label: Date of Sale → Field Name: Date_of_Sale
→ → →
Default Value: Today()-1 → Next → Save & New

Creating 3rd Field:--

→ →
select Data type as Number → next

→ Enter the details → Field Label: Quantity Sold → Length:3 → Decimal places:0
 → Default Value: Show Formulae Editor:1 → Next → Save & New

Creating 4th Field:--

→ select Data type as Currency → next
 → Enter the details → Field Label: Rate → Field Name: Rate → Length:4 → Decimal places:2
 → Default Value: 10 → Next → Save & New

Creating 5th Field:--

→ select Data type as Currency → next
 → MySale field → Quantity__Sold__c*Rate__c → next → save.

Now create an App

→ Setup → Create → App → new → MyShop → Next → Select an Image → Next → Add Object MySales.

Now create an Tab

→ Setup → Create → Tab → New Custom Tab → Choose MySales object → select tab style → save.

On the top in the tab bar you can see the tab which has been created by you click on the tab you can see your object is opened just click on new button and provide the details mentioned.

Conclusion: In this we have created a MyShop Application on Force.com using declarative model.

Experiment No. 3

Aim: Creating an Application in Salesforce.com using Apex programming Language.

Theory: Step1:

Log into your Sandbox or Developers Organization.

→ → →

Click on setup → create → objects → new custom objects.
Enter Book for label.

Enter Books for plural label.

Click Save.

Step 2:

Now let's create a custom field.

In the custom field & relationship section of the Book Object click new.

Select Number for the datatype & next.

Enter Price for the field Label.

Enter 16 in the length text box.

Enter 2 in the decimal places & Next....next.... save.

Step 3:

→ →

Click setup → Develop → Apex Classes & click new
In the class Editor enter this class

```
public class MyHelloWorld{  
  
    public static void applyDiscount(Book_c[] books)  
  
    {  
  
        for(Book_c b:books)
```

```

    {b.Price_c*=0.9;}
}

}

```

Step 4:

Add a trigger

A trigger is a piece of code that can execute objects before or after specific data manipulation language events occurred.

→ → →

Click on setup create objects click the object you have created ex:
Book Scroll down you can see Trigger Click on New

In the trigger Editor enter this class

trigger HelloWorldTrigger on Book_c(before insert)

```
{
Book_c[] books=Trigger.new;
MyHelloWorld.applyDiscount(books);
}
```

Step 5:

Click on setup → create → tabs → new custom tab → choose Book → next&.next&..save.
Click on tab Books → new → insert a name for Book → insert price for that book → click on save.

Conclusion:

Thus we have studied how to create and run an application in salesforce developers site by using APEX programming language.

Experiment No. 4

Aim: To study & Implement Web services in SOAP for JAVA Applications.

Theory:

Overview of Web Services

Web services are application components that are designed to support interoperable machine-to-machine interaction over a network. This interoperability is gained through a set of XML-based open standards, such as the Web Services Description Language (WSDL), the Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration (UDDI). These standards provide a common and interoperable approach for defining, publishing, and using web services.

Choosing a Container:

You can either deploy your web service in a web container or in an EJB container. This depends on your choice of implementation. If you are creating a Java EE application, use a web container in any case, because you can put EJBs directly in a web application. For example, if you plan to deploy to the Tomcat Web Server, which only has a web container, create a web application, not an EJB module.

- Choose File > New Project. Select Web Application from the Java Web category. Name the project `Calculator WS Application`. Select a location for the project. Click Next.
- Select your server and Java EE version and click Finish.

Creating a Web Service from a Java Class

- Right-click the `Calculator WS Application` node and choose New > Web Service.
- Name the web service `Calculator WS` and type `org.me.calculator` in Package. Leave Create Web Service from Scratch selected.

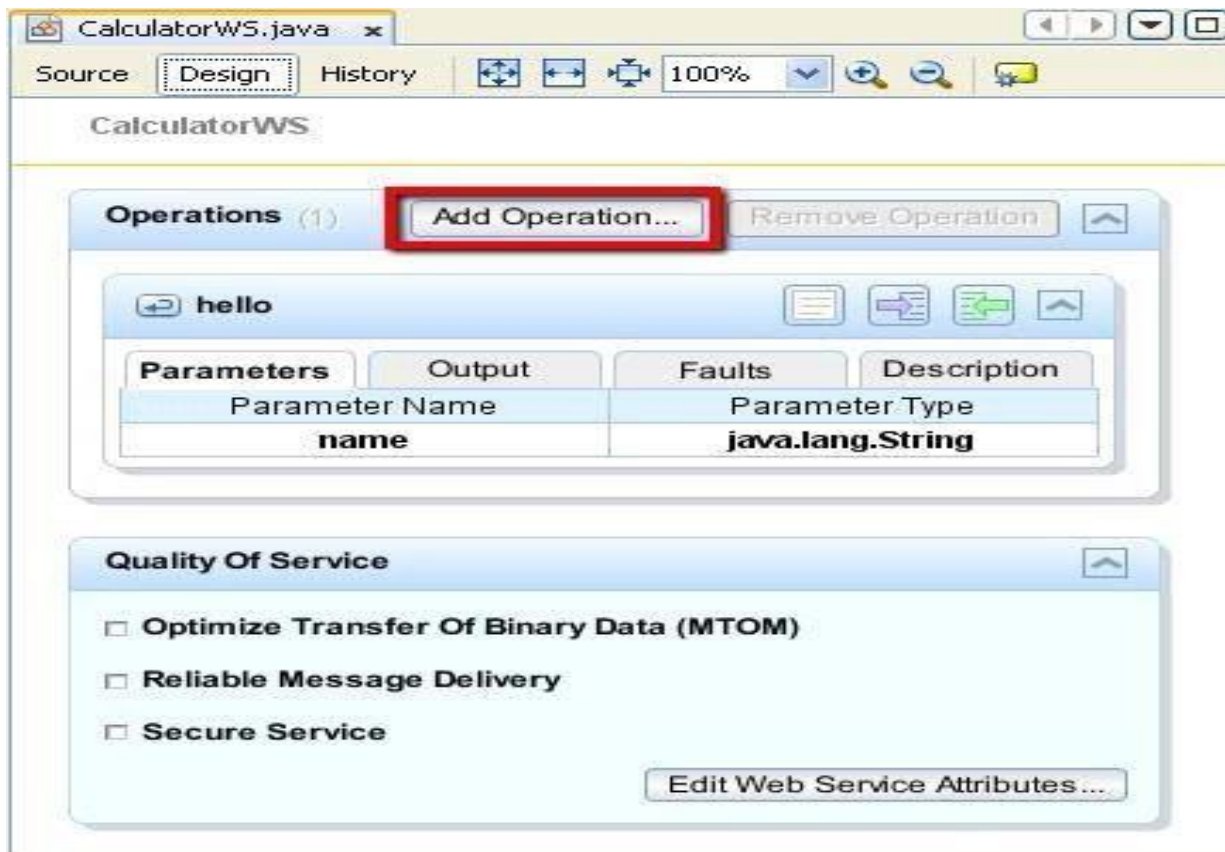
- If you are creating a Java EE project on GlassFish or WebLogic, select Implement Web Service as a Stateless Session Bean.
- Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.

Adding an Operation to the Web Service

The goal of this exercise is to add to the web service an operation that adds two numbers received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service context menu.

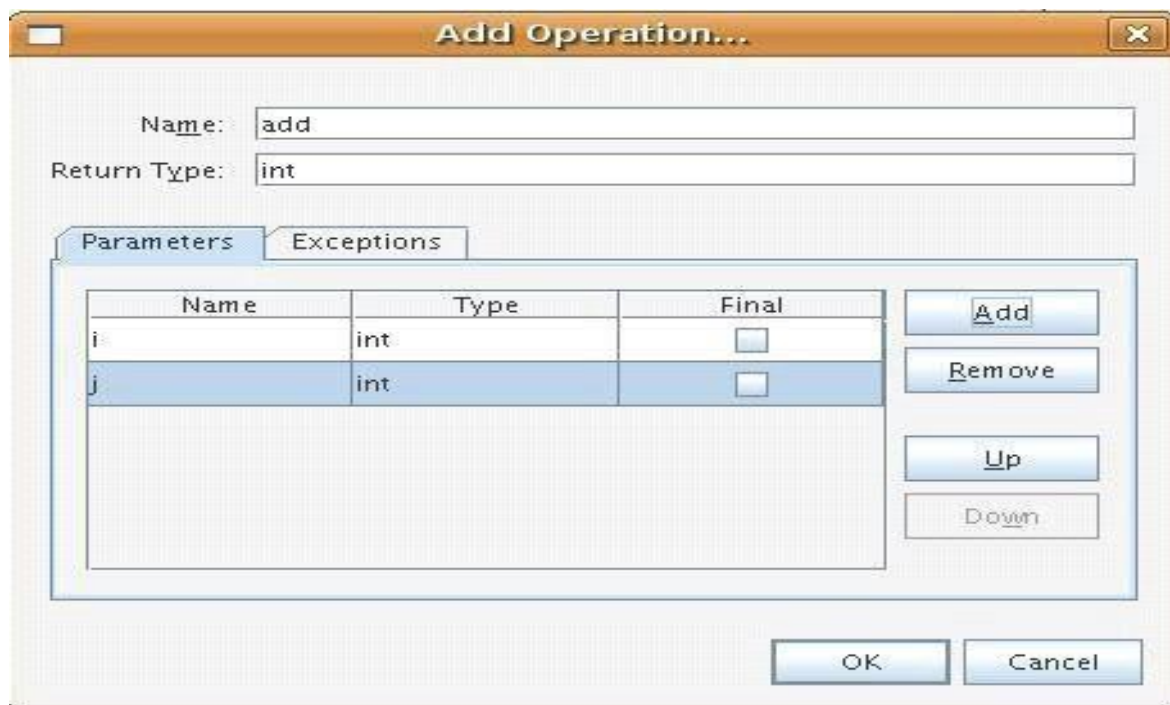
To add an operation to the web service:

- Change to the Design view in the editor.



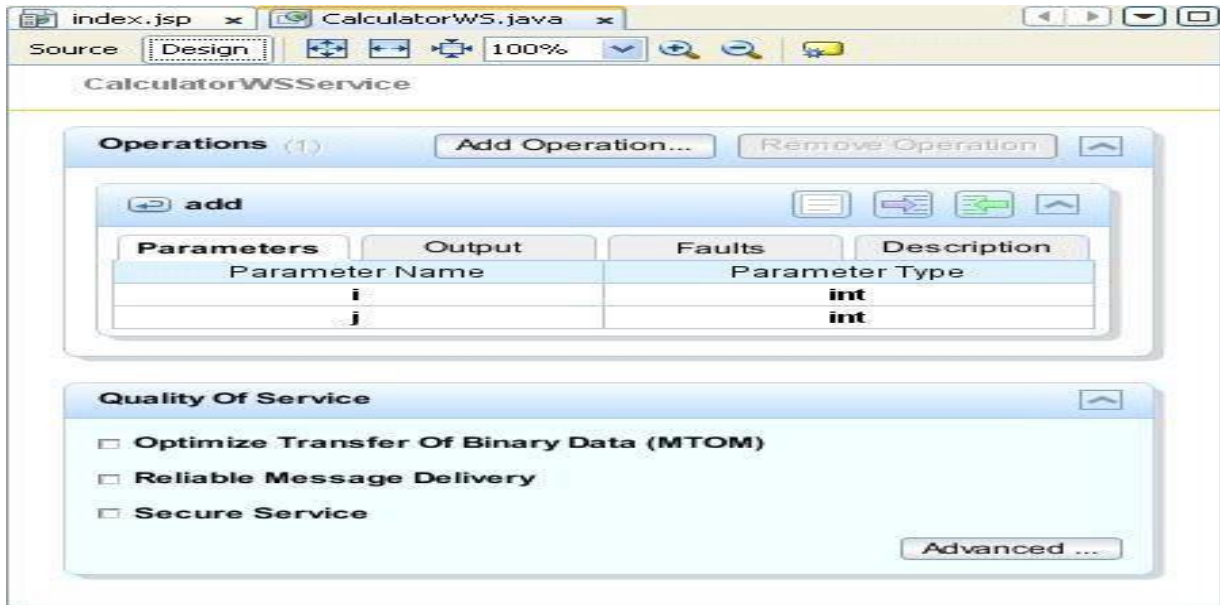
- Click Add Operation in either the visual designer or the context menu. The Add Operation dialog opens.
- In the upper part of the Add Operation dialog box, type `add` in Name and type `int` in the Return Type drop-down list.
- In the lower part of the Add Operation dialog box, click Add and create a parameter of type `int` named `i`.

Click Add again and create a parameter of type `int` called `j`.



- Click OK at the bottom of the Add Operation dialog box. You return to the editor.
- Remove the default `hello` operation, either by deleting the `hello()` method in the source code or by selecting the `hello` operation in the visual designer and clicking Remove Operation.

The visual designer now displays the following:



- Click Source and view the code that you generated in the previous steps. It differs whether you created the service as a Java EE stateless bean or not. Can you see the difference in the screenshots below? (A Java EE 6 or Java EE 7 service that is not implemented as a stateless bean resembles a Java EE 5 service.)

Note. In NetBeans IDE 7.3 and 7.4 you will notice that in the generated `@WebService` annotation the service name is specified explicitly: `@WebService(serviceName = "CalculatorWS")`.

9. In the editor, extend the skeleton `add` operation to the following (changes are in bold):

```
@WebMethod
```

```
public int add(@WebParam(name = "i") int i, @WebParam(name = "j") int j) {
```

```
    int k = i + j;
```

```
    return k;
```

```
}
```

As you can see from the preceding code, the web service simply receives two numbers and then returns their sum. In the next section, you use the IDE to test the web service.

Deploying and Testing the Web Service

After you deploy a web service to a server, you can use the IDE to open the server's test client, if the server has a test client. The GlassFish and WebLogic servers provide test clients.

If you are using the Tomcat Web Server, there is no test client. You can only run the project and see if the Tomcat Web Services page opens. In this case, before you run the project, you need to make the web service the entry point to your application. To make the web service the entry point to your application, right-click the `CalculatorWSApplication` project node and choose Properties. Open the Run properties and type `/CalculatorWS` in the Relative URL field. Click OK. To run the project, right-click the project node again and select Run.

To test successful deployment to a GlassFish or WebLogic server:

- Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server. You can follow the progress of these operations in the `CalculatorWSApplication` (run-deploy) and the GlassFish server or Tomcat tabs in the Output view.
- 2. In the IDE's Projects tab, expand the Web Services node of the `CalculatorWSApplication` project. Right-click the `CalculatorWS` node, and choose Test Web Service.

The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server. For the Tomcat Web Server and deployment of EJB modules, the situation is different:

- If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:

Consuming the Web Service

Now that you have deployed the web service, you need to create a client to make use of the web service's `add` method. Here, you create three clients— a Java class in a Java SE application, a servlet, and a JSP page in a web application.

Note: A more advanced tutorial focusing on clients is [Developing JAX-WS Web Service Clients](#).

Client 1: Java Class in Java SE Application

In this section, you create a standard Java application. The wizard that you use to create the application also creates a Java class. You then use the IDE's tools to create a client and consume the web service that you created at the start of this tutorial.

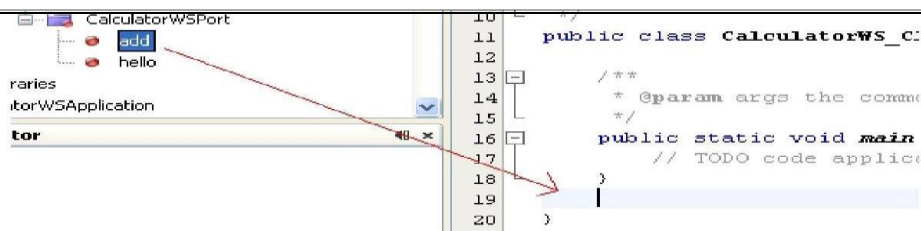
• Choose File > New Project (Ctrl-Shift-N on Linux and Windows, ⌘-Shift-N on MacOS).

- Select `Java Application` from the `Java` category. Name the project `CalculatorWS_Client_Application`. Leave `Create Main Class` selected and accept all other default settings. Click `Finish`.
- Right-click the `CalculatorWS_Client_Application` node and choose `New > Web Service Client`. The `New Web Service Client` wizard opens.
- Select `Project` as the WSDL source. Click `Browse`. Browse to the `CalculatorWS` web service in the `CalculatorWSApplication` project. When you have selected the web service, click `OK`.
- Do not select a package name. Leave this field empty.
- Leave the other settings at default and click `Finish`.

The `Projects` window displays the new web service client, with a node for the `add` method that

you created:

- Double-click your main class so that it opens in the `Source Editor`. Drag the `add` node below the `main()` method.



Note: Alternatively, instead of dragging the `add` node, you can right-click in the editor and then choose `Insert Code > Call Web Service Operation`.

8. In the `main()` method body, replace the `TODO` comment with code that initializes values for `i` and `j`, calls `add()`, and prints the result.
9.

```
public static void main(String[] args) { int i = 3;
```

```
int j = 4;

int result = add(i, j);
System.out.println("Result = " +
result);

}
```

- Right-click the project node and choose Run.

The Output window now shows
the sum: compile:

```
ru
n:
Re
sul
t =
7
```

BUILD SUCCESSFUL (total time: 1 second)

Conclusion:

Thus we have studied use of webservices using SOAP for a java application.

Experiment No. 5

Experiment Title: Implementation of Para-Virtualization using VM Ware's Workstation/ Oracle's Virtual Box and Guest O.S.

Aim: Implementation of Virtual Box for Virtualization of any OS.

Theory:

Virtual Box is a cross-platform virtualization application. What does that mean? For one thing, it installs on your existing Intel or AMD-based computers, whether they are running Windows, Mac, Linux or Solaris operating systems. Secondly, it extends the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. So, for example, you can run Windows and Linux on your Mac, run Windows Server 2008 on your Linux server, run Linux on your Windows PC, and so on, all alongside your existing applications. You can install and run as many virtual machines as you like the only practical limits are disk space and memory. Virtual Box is deceptively simple yet also very powerful. It can run everywhere from small embedded systems or desktop class machines all the way up to datacenter deployments and even Cloud environments.

The techniques and features that Virtual Box provides are useful for several scenarios:

- **Running multiple operating systems simultaneously.** Virtual Box allows you to run more than one operating system at a time. This way, you can run software written for one operating system on another (for example, Windows software on Linux or a Mac) without having to reboot to use it. Since you can configure what kinds of "virtual" hardware should be presented to each such operating system, you can install an old operating system such as DOS or OS/2 even if your real computer's hardware is no longer supported by that operating system.
- **Easier software installations.** Software vendors can use virtual machines to ship entire software configurations. For example, installing a complete mail server solution on a real machine can be a tedious task. With Virtual Box, such a complex setup (then often called an "appliance") can be packed into a virtual machine. Installing and running a mail server becomes as easy as importing such an appliance into Virtual Box.

- **Testing and disaster recovery.** Once installed, a virtual machine and its virtual hard disks can be considered a "container" that can be arbitrarily frozen, woken up, copied, backed up, and transported between hosts.
- **Infrastructure consolidation.** Virtualization can significantly reduce hardware and electricity costs. Most of the time, computers today only use a fraction of their potential power and run with low average system loads. A lot of hardware resources as well as electricity is thereby wasted. So, instead of running many such physical computers that are only partially used, one can pack many virtual machines onto a few powerful hosts and balance the loads between them.

Some Terminologies used:

When dealing with virtualization (and also for understanding the following chapters of this documentation), it helps to acquaint oneself with a bit of crucial terminology, especially the following terms:

Host operating system (host OS). This is the operating system of the physical computer on which Virtual Box was installed. There are versions of Virtual Box for Windows, Mac OS X, Linux and Solaris hosts.

Guest operating system (guest OS). This is the operating system that is running inside the virtual machine. Theoretically, Virtual Box can run any x86 operating system (DOS, Windows, OS/2, FreeBSD, Open BSD), but to achieve near-native performance of the guest code on your machine, we had to go through a lot of optimizations that are specific to certain operating systems. So while your favorite operating system *may* run as a guest, we officially support and optimize for a select few (which, however, include the most common ones).

Virtual machine (VM). This is the special environment that Virtual Box creates for your guest operating system while it is running. In other words, you run your guest operating system "in" a VM. Normally, a VM will be shown as a window on your computers desktop, but depending on which of the various frontends of VirtualBox you use, it can be displayed in full screen mode or remotely on another computer. In a more abstract way, internally, VirtualBox thinks of a VM as a set of parameters that determine its behavior. They include

hardware settings (how much memory the VM should have, what hard disks VirtualBox should virtualize through which container files, what CDs are mounted etc.) as well as state information (whether the VM is currently running, saved, its snapshots etc.). These settings are mirrored in the VirtualBox Manager window as well as the **VBoxManage** command line program;

Guest Additions. This refers to special software packages which are shipped with VirtualBox but designed to be installed *inside* a VM to improve performance of the guest OS and to add extra features.

Starting Virtual Box:

After installation, you can start VirtualBox as follows:

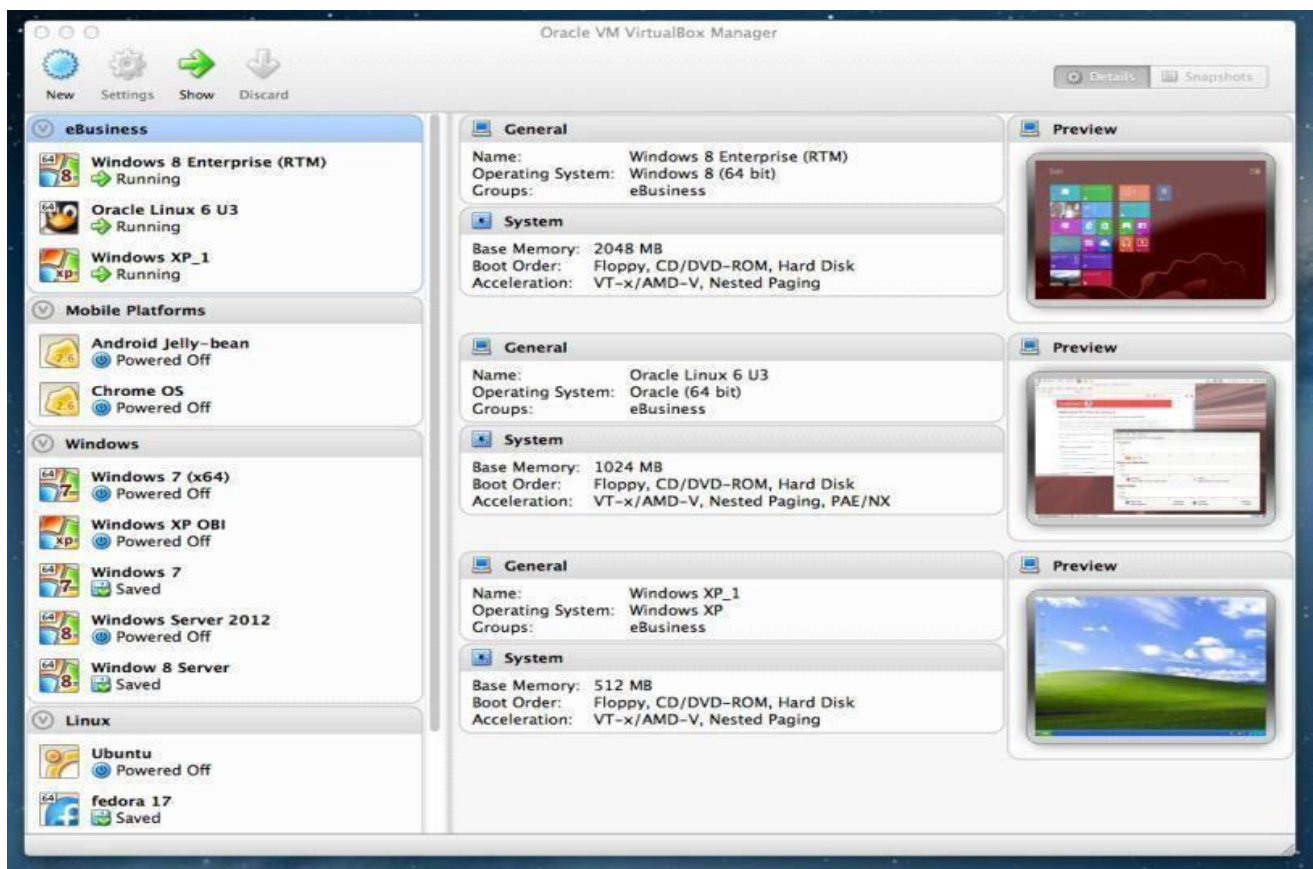
- On a Windows host, in the standard "Programs" menu, click on the item in the "VirtualBox" group. On Vista or Windows 7, you can also type "VirtualBox" in the search box of the "Start" menu.
- On a Mac OS X host, in the Finder, double-click on the "VirtualBox" item in the "Applications" folder. (You may want to drag this item onto your Dock.)
- On a Linux or Solaris host, depending on your desktop environment, a "VirtualBox" item may have been placed in either the "System" or "System Tools" group of your "Applications" menu. Alternatively, you can type VirtualBox in a terminal.

When you start VirtualBox for the first time, a window like the following should come up:



This window is called the "**VirtualBox Manager**". On the left, you can see a pane that will later list all your virtual machines. Since you have not created any, the list is empty. A row of buttons above it allows you to create new VMs and work on existing VMs, once you have some. The pane on the right displays the properties of the virtual machine currently selected, if any. Again, since you don't have any machines yet, the pane displays a welcome message.

To give you an idea what VirtualBox might look like later, after you have created many machines, here's another example:



Creating your first virtual machine:

Click on the "New" button at the top of the VirtualBox Manager window. A wizard will pop up to guide you through setting up a new virtual machine (VM)



On the following pages, the wizard will ask you for the bare minimum of information that is needed to

create a VM, in particular:

- The **VM name** will later be shown in the VM list of the VirtualBox Manager window, and it will be used for the VM's files on disk. Even though any name could be used, keep in mind that once you have created a few VMs, you will appreciate if you have given your VMs rather informative names; "My VM" would thus be less useful than "Windows XP SP2 with OpenOffice".
- For "**Operating System Type**", select the operating system that you want to install later. The supported operating systems are grouped; if you want to install something very unusual that is not listed, select "Other". Depending on your selection, Virtual Box will enable or disable certain VM settings that your guest operating system may require. This is particularly important for 64-bit guests (see [Section 3.1.2, 64-bit guests](#)). It is therefore recommended to always set it to the correct value.

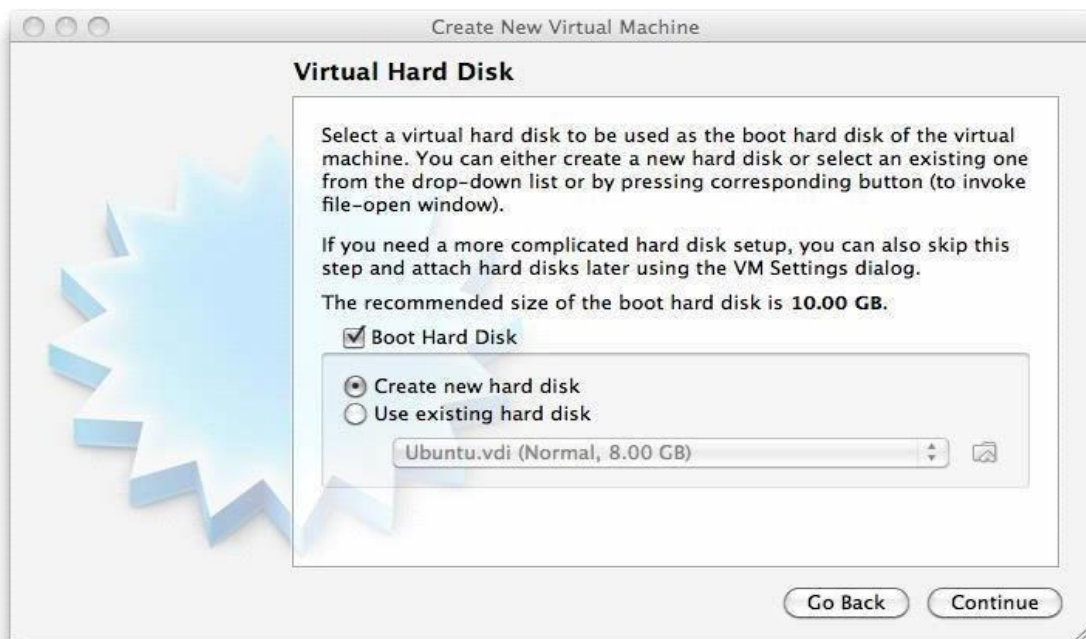
- On the next page, select the **memory (RAM)** that Virtual Box should allocate every time the virtual machine is started. The amount of memory given here will be taken away from your host machine and presented to the guest operating system, which will report this size as the (virtual) computer's installed RAM.

A Windows XP guest will require at least a few hundred MB RAM to run properly, and Windows Vista will even refuse to install with less than 512 MB. Of course, if you want to run graphics-intensive applications in your VM, you may require even more RAM.

So, as a rule of thumb, if you have 1 GB of RAM or more in your host computer, it is usually safe to allocate 512 MB to each VM. But, in any case, make sure you always have at least 256 to 512 MB of RAM left on your host operating system. Otherwise you may cause your host OS to excessively swap out memory to your hard disk, effectively bringing your host system to a standstill. As with the other settings, you can change this setting later, after you have created the VM.

4. Next, you must specify a **virtual hard disk** for your VM. There are many and potentially complicated ways in which VirtualBox can provide hard disk space to a VM (see [Chapter 5, *Virtual storage*](#) for details), but the most common way is to use a large image file on your "real" hard disk, whose contents VirtualBox presents to your VM as if it were a complete hard disk. This file represents an entire hard disk then, so you can even copy it to another host and use it with another VirtualBox installation.

The wizard shows you the following window:



Here you have the following options:

- To create a new, empty virtual hard disk, press the **"New"** button.
- You can pick an **existing** disk image file. The **drop-down list** presented in the window contains all disk images which are currently remembered by VirtualBox, probably because they are currently attached to a virtual machine (or have been in the past). Alternatively, you can click on the small **folder button** next to the drop-down list to bring up a standard file dialog, which allows you to pick any disk image file on your host disk.

Most probably, if you are using VirtualBox for the first time, you will want to create a new disk image. Hence, press the "New" button. This brings up another window, the **"Create New Virtual Disk Wizard"**, which helps you create a new disk image file in the new virtual machine's folder.

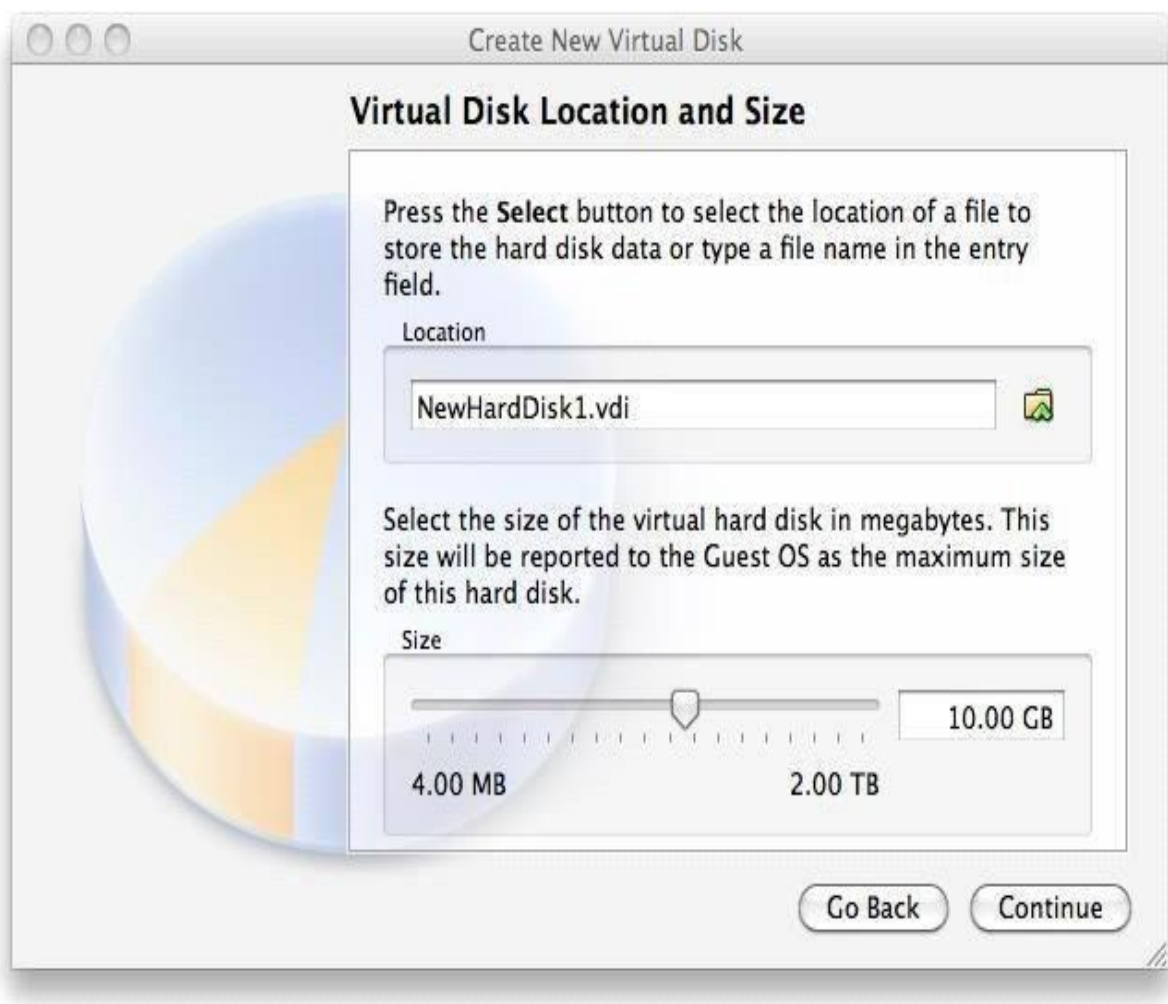
VirtualBox supports two types of image files:

- A **dynamically allocated file** will only grow in size when the guest actually stores data on its virtual hard disk. It will therefore initially be small on the host hard drive and only later grow to the size specified as it is filled with data.
- A **fixed-size file** will immediately occupy the file specified, even if only a fraction of the virtual hard disk space is actually in use. While occupying much more space, a fixed-size file incurs less overhead and is therefore slightly faster than a dynamically allocated file.

For details about the differences, please refer to [Section 5.2, Disk image files \(VDI, VMDK, VHD, HDD\)](#).

After having selected or created your image file, again press **"Next"** to go to the next page.

- After clicking on **"Finish"**, your new virtual machine will be created. You will then see it in the list on the left side of the Manager window, with the name you entered initially.

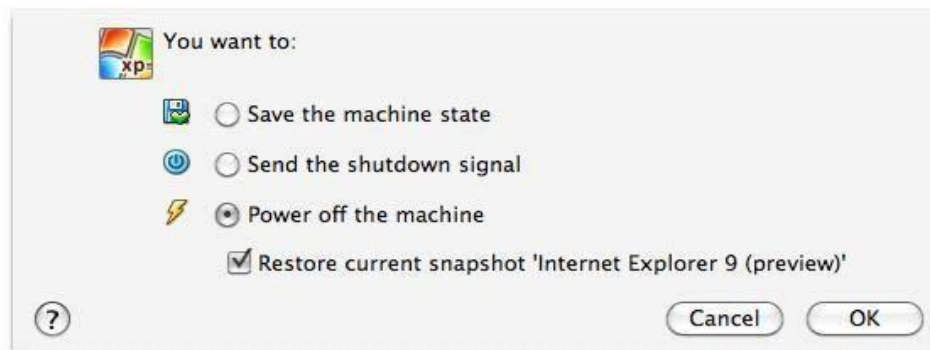


Running your virtual machine: To start a virtual machine, you have several options:

- Double-click on its entry in the list within the Manager window or
- select its entry in the list in the Manager window it and press the "Start" button at the top or
- for virtual machines created with VirtualBox 4.0 or later, navigate to the "VirtualBox VMs" folder in your system user's home directory, find the subdirectory of the machine you want to start and double-click on the machine settings file (with a .vbox file extension). This opens up a new window, and the virtual machine which you selected will boot up. Everything which would normally be seen on the virtual system's monitor is shown in the window. In general, you can use the virtual machine much like you would use a real computer. There are couple of points worth mentioning however.

Saving the state of the machine: When you click on the "Close" button of your virtual machine window (at the top right of the window, just like you would close any other window on your

system), VirtualBox asks you whether you want to "save" or "power off" the VM. (As a shortcut, you can also press the Host key together with "Q".)



The difference between these three options is crucial. They mean:

- **Save the machine state:** With this option, VirtualBox "freezes" the virtual machine by completely saving its state to your local disk. When you start the VM again later, you will find that the VM continues exactly where it was left off. All your programs will still be open, and your computer resumes operation. Saving the state of a virtual machine is thus in some ways similar to suspending a laptop computer (e.g. by closing its lid).
- **Send the shutdown signal.** This will send an ACPI shutdown signal to the virtual machine, which has the same effect as if you had pressed the power button on a real computer. So long as the VM is running a fairly modern operating system, this should trigger a proper shutdown mechanism from within the VM.
- **Power off the machine:** With this option, VirtualBox also stops running the virtual machine, but *without* saving its state. As an exception, if your virtual machine has any snapshots (see the next chapter), you can use this option to quickly **restore the current snapshot** of the virtual

machine. In that case, powering off the machine will not disrupt its state, but any changes made since that snapshot was taken will be lost. The "**Discard**" button in the VirtualBox

Manager window discards a virtual machine's saved state. This has the same effect as powering it off, and the same warnings apply.

Importing and exporting virtual machines

VirtualBox can import and export virtual machines in the industry-standard Open Virtualization Format (OVF). OVF is a cross-platform standard supported by many virtualization products which allows for creating ready-made virtual machines that can then be imported into a virtualizer such as VirtualBox. VirtualBox makes OVF import and export easy to access and supports it from the Manager window as well as its command-line interface. This allows for packaging so-called **virtual appliances**: disk images together with configuration settings that can be distributed easily. This way one can offer complete ready-to-use software packages (operating systems with applications) that need no configuration or installation except for importing into VirtualBox.

Appliances in OVF format can appear in two variants:

- They can come in several files, as one or several disk images, typically in the widely-used VMDK format (see [Section 5.2, Disk image files \(VDI, VMDK, VHD, HDD\) ||](#)) and a textual description file in an XML dialect with an .ovf extension. These files must then reside in the same directory for Virtual Box to be able to import them.
- Alternatively, the above files can be packed together into a single archive file, typically with an .ova extension. (Such archive files use a variant of the TAR archive format and can therefore be unpacked outside of Virtual Box with any utility that can unpack standard TAR files.)

Select "File" -> "Export appliance". A different dialog window shows up that allows you to combine several virtual machines into an OVF appliance. Then, select the target location where the target files should be stored, and the conversion process begins. This can again take a while.

Conclusion:

Thus we have studied use of Multiple OS using Virtual Box by virtualizing.

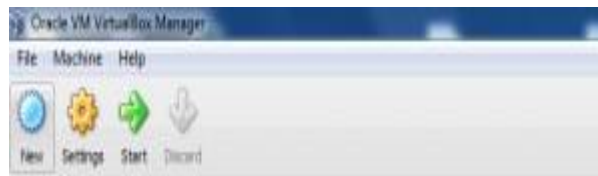
Experiment No. 6

Aim: Install a C compiler in the virtual machine and execute a sample program.

Theory:

Step1: Create a Virtual Machine

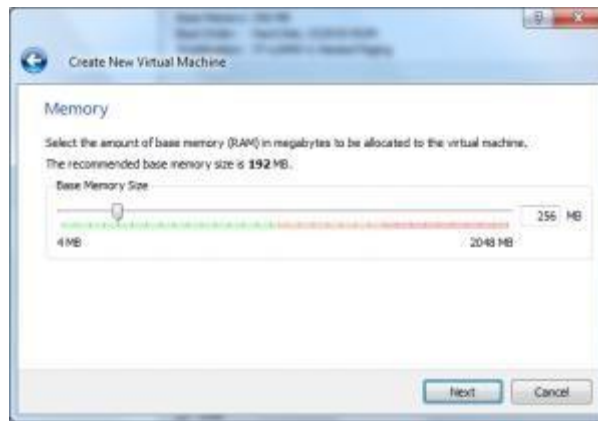
First job is to create a virtual machine, to do so open VirtualBox and click -New from the toolbar.



- Select "New" from the toolbar to create a new Virtual Machine
- Enter any name for your Virtual Machine and select the Operating System you're going to use as your GUEST OS.



- Select the OS you are going to use inside your Virtual Machine
- Select the RAM size to be allocated to your guest OS, choose this wisely it should be equal to or more than the minimum system requirement of your guest OS at the same time if it goes more than 50% of your physical machine's RAM it'll slow down your host OS.



- Select the amount of RAM to be allocated to the Virtual Machine
You need to create a Virtual Hard disk for your Virtual Machine, this is just a file with a .vdi extension which will contain all files stored inside that virtual machine. Choose a size suited for your guest OS and select the -Dynamically expanding disk if you want to save disk creation time and file size. If you choose -Fixed size disk it will take up the entire size specified but the performance of your Virtual Machine will be better. After clicking finish move on to the next step.

Step 2: Change the boot order of the VM

Right click the newly created Virtual machine and go to settings.

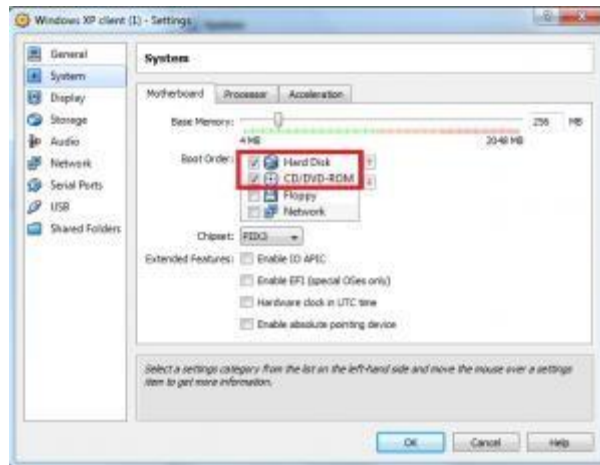


Go to the settings of your Virtual Machine

- Select storage from the left side list, select the CD icon and from the right select -Choose a virtual CD/DVD disk file, navigate to the ISO image file of the OS on your computer. Now the OS image is mounted to your Virtual machine.



The ISO image selected will be mounted in the Optical drive of your Virtual Machine To change the boot device order of your virtual machine go to the –System– option from the left side list, select Hard Disk and click the up arrow to bring it to the top of the list. Make the CD/DVD-ROM the second device and uncheck the other devices.



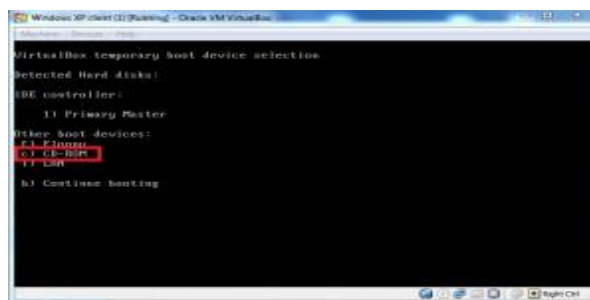
Set the boot device priority for the virtual machine

Step 3: Start the Virtual machine

To start the virtual machine double-click the VM, a window opens here you can press F12 if you want to select a device of your choice to boot.



Press F12 to select the boot device of your choice
To boot from the CD-ROM press c.



Press c to boot the VM from a CD image
Your VM will now boot from your Operating System image file and the OS installation will start as usual.
Some important shortcuts
The following keyboard shortcuts can be used inside the VM to perform certain actions that might conflict with your physical machine

[Right Ctrl] + Del – Equivalent to pressing [ctrl] + [alt] + [delete] inside the Virtual Machine. If you press ctrl+alt+del the physical machine will also be affected.

[Right Ctrl] + H – (Halt) Equivalent to pressing the power button on the physical machine.

[Right Ctrl] + R – (Reset) Equivalent to pressing the reset button, will reset your virtual machine.

[Right CTRL] + F – (Fullscreen) Toggles fullscreen

Experiment No. 7

Experiment Title: Create a Cloud Storage bucket using Amazon Simple Storage Service (Amazon S3).

Description

Amazon Simple Storage Service (S3) provides secure, durable, and highly scalable object storage. To upload data such as photos, videos and static documents, you must first create a logical storage bucket in one of the AWS regions. Then you can upload any number of objects to it. Buckets and objects are resources, and Amazon S3 provides both APIs and a web console to manage them.

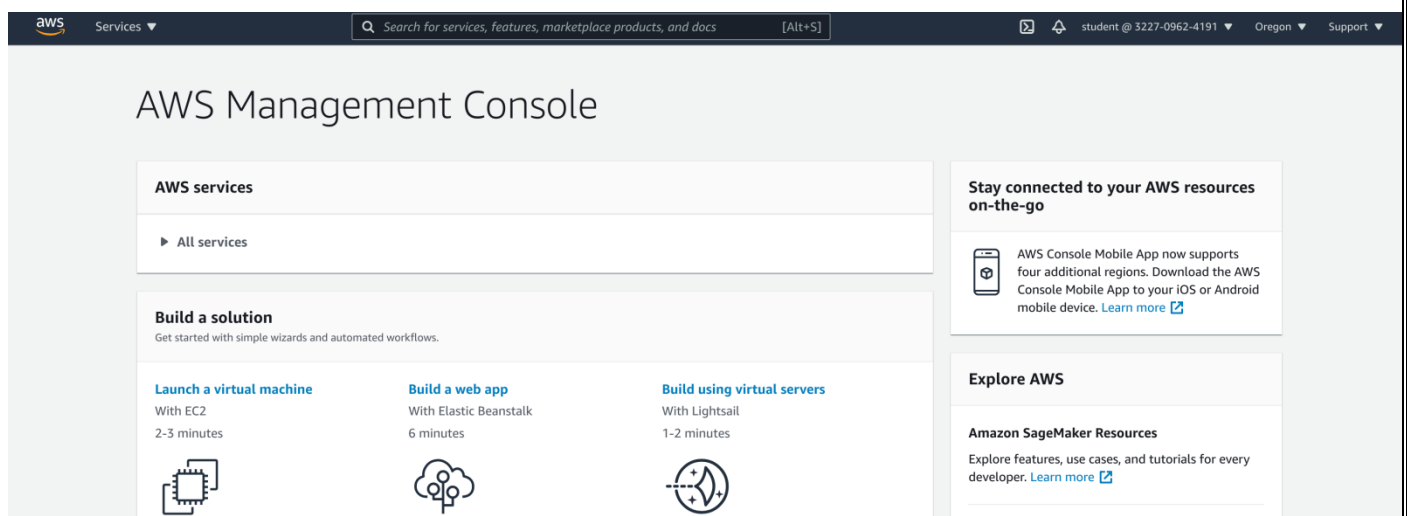
Amazon S3 can be used alone or together with other AWS services such as Amazon EC2, Amazon Elastic Block Store (Amazon EBS), and Amazon Glacier, as well as third-party storage repositories and gateways. Amazon S3 provides cost-effective object storage for a wide variety of use cases including web applications, content distribution, backup and archiving, disaster recovery, and big data analytics.

This Lab guides you through the bucket creation process and its first usage. The Cloud Academy Labs Engine constantly checks the Lab environment, giving you instant feedback on your work.

Step 1: Logging into Amazon Web Services Console

Introduction

This lab experience involves Amazon Web Services (AWS), and you will use the AWS Management Console to complete all the lab steps. Please note that you will have a space storage limit of 100GB for this lab, which will be more than sufficient to complete it.



The AWS Management Console is a web control panel for managing all your AWS resources, from EC2

instances to SNS topics. The console enables cloud management for all aspects of the AWS account, including managing security credentials, and even setting up new IAM Users.

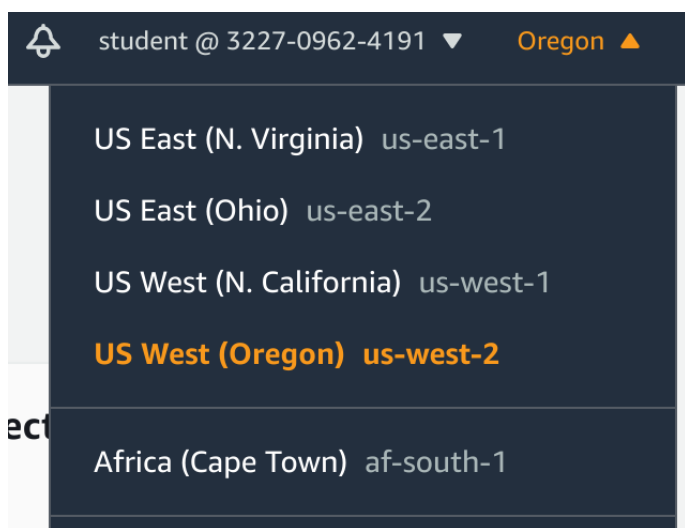
Instructions

1. Open the Amazon Console by clicking this button:

[Open Console](#)

2. Enter the credentials and click **Sign in**:

3. Select the **US West (Oregon) us-west-2** region using the upper-right drop-down menu on the AWS Management Console:



Amazon Web Services is available in different regions all over the world, and the console lets you provision resources across multiple regions. You usually choose a region that best suits your business needs to optimize your customer's experience, but you must use the **US West 2** for this Lab.

Step2: Creating an S3 Bucket

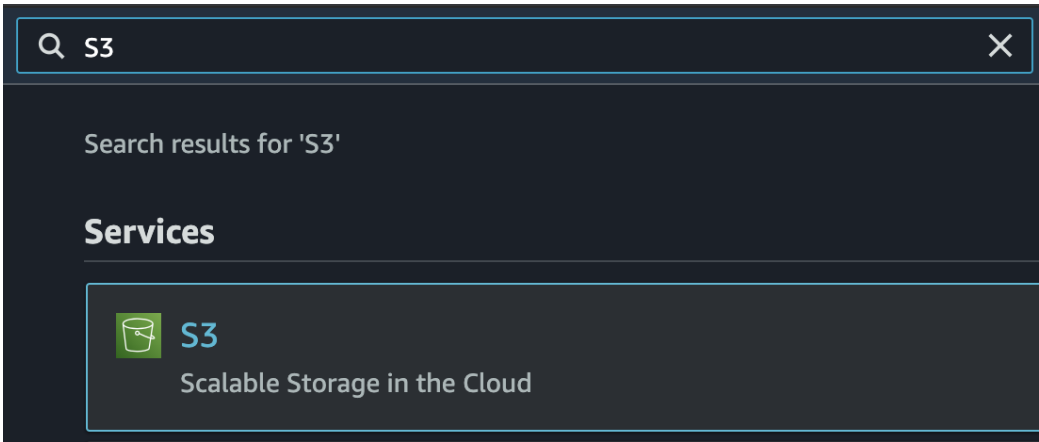
Introduction

You can create an S3 bucket using the AWS Management Console. As with many other AWS services, you can use the AWS API or CLI (command-line interface) as well.

In this step, you will create a new S3 bucket.

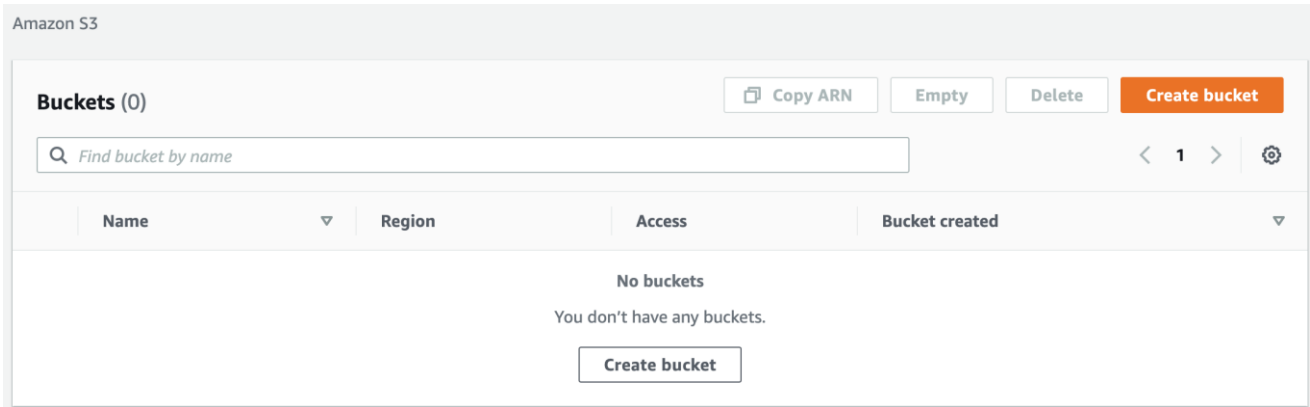
Instructions

1. In the AWS Management Console search bar, enter *S3*, and click the **S3** result under **Services**:



You will be placed in the S3 console.

2. From the S3 console, click the orange **Create Bucket** button:



3. Enter a unique **Bucket name** on the **Name and region** screen of the wizard:

General configuration

Bucket name

myawsbucket

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#) 

Region

US West (Oregon) us-west-2 ▼

- **Region:** US West (Oregon) (This should be set for you. If not, please select this region.)

Important! Bucket names must be globally unique, regardless of the AWS region in which you create the bucket. Buckets must also be DNS-compliant.

The rules for DNS-compliant bucket names are:

- Bucket names must be at least 3 and no more than 63 characters long.
- Bucket names can contain lowercase letters, numbers, periods, and/or hyphens. Each label must start and end with a lowercase letter or a number.
- Bucket names must not be formatted as an IP address (for example, 192.168.1.1).

The following examples are valid bucket names: calabs-bucket-1, ca-labs-bucket.

Troubleshooting Tip: If you receive an error because your bucket name is not unique, append a unique number to the bucket name in order to guarantee its uniqueness:




Error

The requested bucket name is not available. The bucket namespace is shared by all users of the system. Please select a different name and try again.

For example, change "calabs-bucket" to "calabs-bucket-1" (or a unique number/character string) and try again.

4. Leave the **Block public access (bucket settings)** at the default values:

Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#) 

☒ **Block *all* public access**

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☒ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☒ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**

S3 will ignore all ACLs that grant public access to buckets and objects.

☒ **Block public access to buckets and objects granted through *new* public bucket or access point policies**

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☒ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

No changes are needed. This is where you can set public access permissions.

5. Click on **Create bucket**.

6. Click on the new bucket and then click **Properties** to switch to the **Properties** tab for your new bucket:

Objects	Properties	Permissions	Metrics	Management	Access Points
---------	-------------------	-------------	---------	------------	---------------

The **Properties** tab allows you to configure several options after bucket creation. Similarly for the **Permissions** and **Management** tabs. No changes are needed at this time. It's helpful to know that post-creation configuration options are available, however.

Summary

In this step, you created a new S3 bucket.

Step 3: Creating a Folder inside an S3 Bucket

Introduction

The AWS S3 console allows you to create folders for grouping objects. This can be a very helpful organizational tool. However, in Amazon S3, buckets and objects are the primary resources. A folder simply becomes a prefix for object key names that are virtually archived into it.

Instructions

1. Select the calabs-bucket you created earlier. (*Reminder:* Your bucket name will differ slightly.)

2. Click + **Create Folder**:

Objects (0)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you



- Enter *cloudfolder* in the **Folder name** field
- Click **Create folder** when ready

The folder is created inside your S3 bucket:

Summary

In this Step you created a folder that can help with organizing objects within an S3 bucket. A folder within a bucket can be helpful programmatically as well. For example, you can configure your application to output certain file types (such as PNG) to a specific folder name (image-output).

Step 4: Uploading a File to S3

Introduction

When you upload a folder from your local system or another machine, Amazon S3 uploads all the of files and subfolders from the specified folder to your bucket. It then assigns a key value that is a combination of the uploaded file name and the folder name. In this Lab Step you will upload a file to your bucket. The process is similar for uploading a single file, multiple files, or a folder with files in it.

In order to complete this Step, you have to upload the any image file from your local file storage into an S3 folder you created earlier.

Instructions

1. Click on the *cloudfolder* folder. You are placed within the empty folder in your S3 bucket.

Note: Click the folder name itself, not the checkbox for the folder name. If you select the folder checkbox then upload a file, it will be placed above the folder (not inside it).

2. Click the **Upload** button.

3. Click **Add Files**.

4. Check the file and click on **Remove**.

5. This time, rather than browsing to a file, drag and drop the image file onto the wizard. The wizard adds it to the list of files to upload.

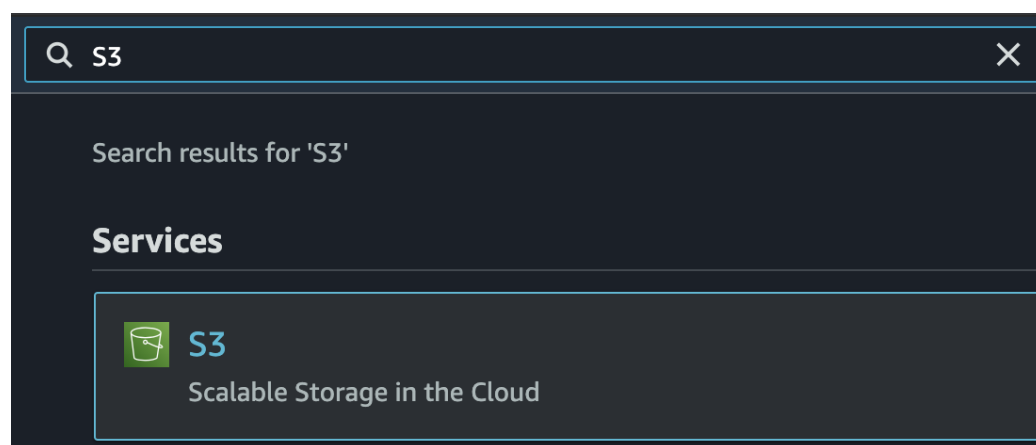
6. Click **Upload** to upload the file. It is placed in the folder in your bucket:

Deleting an S3 Bucket

You can delete an S3 bucket using the S3 console. You will delete all objects within the bucket as well.

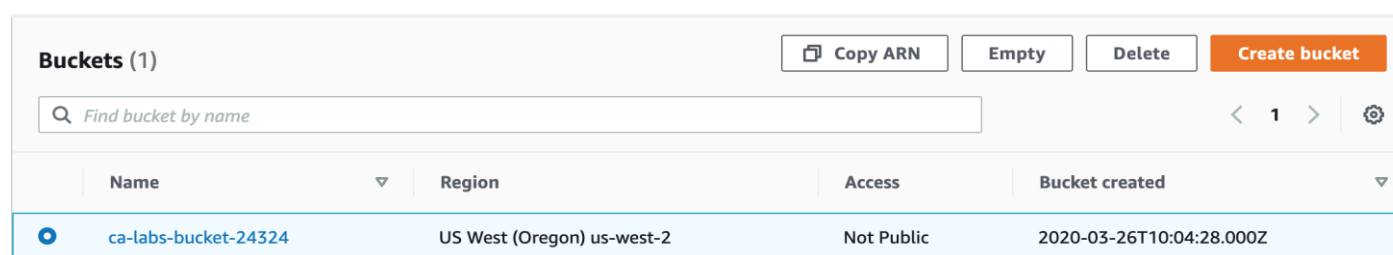
Instructions:

1. In the AWS Management Console search bar, enter *S3*, and click the **S3** result under **Services**:



From the top level of the S3 console, notice the **Delete** button is not actionable.

2. Check the the calabs-bucket. The bucket is selected:



3. Click **Delete**. Once a bucket is selected, the **Delete** button becomes actionable.

Warning: Make sure to delete all the items inside the bucket, otherwise AWS won't let you delete the bucket.

4. You must enter the entire name (for example, *calabs-bucket-123*) of the bucket to confirm deletion:

Delete bucket



- Deleting a bucket cannot be undone.
- Bucket names are unique. If you delete a bucket, another AWS user can use the name.

[Learn more](#) 

Delete bucket ca-labs-bucket-24324

To confirm deletion, type the name of the bucket.

Cancel

Delete bucket

5. Click **Confirm** to delete the bucket.

Important! Notice the message from AWS: "Amazon S3 buckets are unique. If you delete this bucket, you may lose the bucket name to another AWS user."

If retaining the bucket name is important to you, consider using the **Empty bucket** feature and not actually deleting the bucket.

Experiment No. 8

Aim: Working with AWS CLI

Theory:

Introduction

Amazon provides a lot of excellent documentation for the AWS CLI. You will likely find yourself using one of the following to help with your learning curve:

- aws help (from the command line)
- [AWS CLI Reference documentation](#)

This step will focus on the command line help. (Bookmarking the reference information is helpful when you need to look something up but don't have the AWS CLI installed on your local machine or any EC2 instances.)

Instructions

1. From your SSH shell, enter the following command:

Copy code

```
1  
aws help
```

```
AWS () AWS ()

NAME
    aws -

DESCRIPTION
    The AWS Command Line Interface is a unified tool to manage your AWS
    services.

SYNOPSIS
    aws [options] <command> <subcommand> [parameters]

    Use aws command help for information on a specific command. Use aws
    help topics to view a list of available help topics. The synopsis for
    each command shows its parameters and their usage. Optional parameters
    are shown in square brackets.

OPTIONS
    --debug (boolean)

    Turn on debug logging.
```

This is the AWS CLI help page. You can press *enter* to scroll down or *q* to quit.

2. Press *enter* until you can see the **AVAILABLE SERVICES** section:

```
AVAILABLE SERVICES
  o acm
  o apigateway
  o application-autoscaling
  o appstream
  o autoscaling
  o batch
  o budgets
  o cloudformation
  o cloudfront
  o cloudhsm
  o cloudsearch
  o cloudsearchdomain
  o cloudtrail
  o cloudwatch
  o codebuild
  o codecommit
```

As you can see, in the `aws help` command there are many services available. Thus, the AWS CLI makes use of a common pattern for working with each service: `aws <service> <command>`. For example, `aws ec2 describe-regions`.

3. Follow the AWS CLI service pattern to get help on commands available for the EC2 service:

[Copy code](#)

```
1
aws ec2 help
```

EC2 ()

EC2 ()

NAME

ec2 -

DESCRIPTION

Amazon Elastic Compute Cloud (Amazon EC2) provides resizable 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.

AVAILABLE COMMANDS

- o accept-reserved-instances-exchange-quote
- o accept-vpc-peering-connection
- o allocate-address
- o allocate-hosts
- o assign-ipv6-addresses
- o assign-private-ip-addresses
- o associate-address

The service help pages simply give a description and then list all of the available commands for the service.

4. Get the help page for a specific command:

Copy code

1

```
aws ec2 describe-regions help
```

```
DESCRIBE-REGIONS {}
```

```
DESCRIBE-REGIONS {}
```

NAME

```
describe-regions -
```

DESCRIPTION

Describes one or more regions that are currently available to you.

For a list of the regions supported by Amazon EC2, see [Regions and Endpoints](#) .

SYNOPSIS

```
    describe-regions
    [--dry-run | --no-dry-run]
    [--region-names <value>]
    [--filters <value>]
    [--cli-input-json <value>]
    [--generate-cli-skeleton <value>]
```

OPTIONS

```
--dry-run | --no-dry-run (boolean)
    Checks whether you have the required permissions for the action,
    without actually making the request, and provides an error response.
    If you have the required permissions, the error response is DryRun-
    Operation . Otherwise, it is UnauthorizedOperation .
```

Help pages for service commands provide additional sections for how to use the command:

- **SYNOPSIS:** Shows the usage of the command
- **OPTIONS:** Describes the options/arguments you can provide on the command-line to modify the behavior of the command
- **EXAMPLES:** Provides sample commands along with a text description of what the command accomplishes and the output the command produces
- **OUTPUT:** Describes the format of the output of the command

5. Specify JSON output format appending `--output json`:

Copy code

```
1
aws ec2 describe-regions --output json
```

```
{
  "Regions": [
    {
      "Endpoint": "ec2.ap-south-1.amazonaws.com",
      "RegionName": "ap-south-1"
    },
    {
      "Endpoint": "ec2.eu-west-2.amazonaws.com",
      "RegionName": "eu-west-2"
    },
    {
      "Endpoint": "ec2.eu-west-1.amazonaws.com",
      "RegionName": "eu-west-1"
    },
    {
      "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
      "RegionName": "ap-northeast-2"
    },
  ],
}
```

Recall earlier that you defined **text** as your default output, so all the commands that support text output should return text. However, you can specify the desired output when entering a command by appending `--output <type>`.

6. Display the output of the command in table format:

Copy code

```
1
aws ec2 describe-regions --output table
```

DescribeRegions	
Regions	
Endpoint	RegionName
ec2.ap-south-1.amazonaws.com	ap-south-1
ec2.eu-west-2.amazonaws.com	eu-west-2
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-northeast-2.amazonaws.com	ap-northeast-2
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.ca-central-1.amazonaws.com	ca-central-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.us-east-1.amazonaws.com	us-east-1
ec2.us-east-2.amazonaws.com	us-east-2
ec2.us-west-1.amazonaws.com	us-west-1
ec2.us-west-2.amazonaws.com	us-west-2

7. Display the output in text format:

Copy code

```
1
aws ec2 describe-regions --output text
```

```
REGIONS ec2.ap-south-1.amazonaws.com ap-south-1
REGIONS ec2.eu-west-2.amazonaws.com eu-west-2
REGIONS ec2.eu-west-1.amazonaws.com eu-west-1
REGIONS ec2.ap-northeast-2.amazonaws.com ap-northeast-2
REGIONS ec2.ap-northeast-1.amazonaws.com ap-northeast-1
REGIONS ec2.sa-east-1.amazonaws.com sa-east-1
REGIONS ec2.ca-central-1.amazonaws.com ca-central-1
REGIONS ec2.ap-southeast-1.amazonaws.com ap-southeast-1
REGIONS ec2.ap-southeast-2.amazonaws.com ap-southeast-2
REGIONS ec2.eu-central-1.amazonaws.com eu-central-1
REGIONS ec2.us-east-1.amazonaws.com us-east-1
REGIONS ec2.us-east-2.amazonaws.com us-east-2
REGIONS ec2.us-west-1.amazonaws.com us-west-1
REGIONS ec2.us-west-2.amazonaws.com us-west-2
```

Because you configured text as the default output format, you can omit the `--output` option and achieve the same result.

8. Describe the VPCs in the default region us-west-2:

Copy code

```
1
aws ec2 describe-vpcs --output json
```

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-41e2db27",
      "InstanceTenancy": "default",
      "State": "available",
      "DhcpOptionsId": "dopt-52c58934",
      "CidrBlock": "172.31.0.0/16",
      "IsDefault": true
    },
    {
      "VpcId": "vpc-ca2e2bac",
      "InstanceTenancy": "default",
      "Tags": [
        {
          "Value": "Lab VPC",
          "Key": "Name"
        },
        {
          "Value": "cloudacademylabs",
          "Key": "Lab"
        },
        {
          "Value": "VPC",
          "Key": "aws:cloudformation:logical-id"
        },
        {
          "Value": "arn:aws:cloudformation:us-west-2:053166311128:stack/cloudacademylabs/734828e0-ca9b-11e7-a7d1-50a686be73f2",
          "Key": "aws:cloudformation:stack-id"
        },
        {
          "Value": "cloudacademylabs",
          "Key": "aws:cloudformation:stack-name"
        }
      ],
      "State": "available",
      "DhcpOptionsId": "dopt-52c58934",
      "CidrBlock": "10.0.0.0/20",
      "IsDefault": false
    }
  ]
}
```

Notice there are two **VPCs** in this region.

9. Specify a different region by appending `--region <region name>`:

Copy code

```
1
aws ec2 describe-vpcs --region us-west-2 --output json
```

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-f0e53797",
      "InstanceTenancy": "default",
      "State": "available",
      "DhcpOptionsId": "dopt-a06695c4",
      "CidrBlock": "172.31.0.0/16",
      "IsDefault": true
    }
  ]
}
```

This time, only one VPC is shown, the **default** VPC for the region. If you don't specify a region the AWS CLI will use the default region previously configured.

10. Change your default format to JSON by entering `aws configure` by entering *json* for the format prompt and pressing *enter* to keep the existing value for the other options:

```
AWS Access Key ID [*****QGXA]:
AWS Secret Access Key [*****m+cV]:
Default region name [us-west-2]:
Default output format [text]: json
```

Conclusion

You learned the basics of entering AWS CLI commands, and how to get more information on available commands.

Spend a few minutes and play around a bit more with the help tool. You will learn it is a very powerful tool to help you find useful commands. After using the AWS CLI and the help tool for a while, you will discover the majority of the commands follow the same pattern. Thus it is possible to guess correct commands based on the action you want to perform.

For example, using the command line help, try to find the command to describe all the instances in a particular region or list all S3 buckets that you have in your account.

Experiment No. 9

Aim: Creating Azure Container Instances

In Process

Experiment No. 10

Aim: Case Study of Google App Engine.

Theory:

Platform-as-a-Service (PaaS):

Cloud computing has evolved to include platforms for building and running custom web-based applications, a concept known as Platform-as-a- Service. PaaS is an outgrowth of the SaaS application delivery model. The PaaS model makes all of the facilities required to support the complete life cycle of building and delivering web applications and services entirely available from the Internet, all with no software downloads or installation for developers, IT managers, or end users. Unlike the IaaS model, where developers may create a specific operating system instance with homegrown applications running, PaaS developers are concerned only with webbased development and generally do not care what operating system is used. PaaS services allow users to focus on innovation rather than complex infrastructure. Organizations can redirect a significant portion of their budgets to creating applications that provide real business value instead of worrying about all the infrastructure issues in a roll-your-own delivery model. The PaaS model is thus driving a new era of mass innovation. Now, developers around the world can access unlimited computing power. Anyone with an Internet connection can build powerful applications and easily deploy them to users globally.

Google App Engine:

Architecture :

The Google App Engine (GAE) is Google`s answer to the ongoing trend of Cloud Computing offerings within the industry. In the traditional sense, GAE is a web application hosting service, allowing for development and deployment of web-based applications within a pre-defined runtime environment. Unlike other cloud-based hosting offerings such as Amazon Web Services that operate on an IaaS level, the GAE already provides an application infrastructure on the PaaS level. This means that the GAE

abstracts from the underlying hardware and operating system layers by providing the hosted application with a set of application-oriented services. While this approach is very convenient for

developers of such applications, the rationale behind the GAE is its focus on scalability and usage-based infrastructure as well as payment.

Costs :

Developing and deploying applications for the GAE is generally free of charge but restricted to a certain amount of traffic generated by the deployed application. Once this limit is reached within a certain time period, the application stops working. However, this limit can be waived when switching to a billable quota where the developer can enter a maximum budget that can be spent on an application per day. Depending on the traffic, once the free quota is reached the application will continue to work until the maximum budget for this day is reached. Table 1 summarizes some of the in our opinion most important quotas and corresponding amount per unit that is charged when free resources are depleted and additional, billable quota is desired.

Features :

With a Runtime Environment, the Data store and the App Engine services, the GAE can be divided into three parts.

Runtime Environment

The GAE runtime environment presents itself as the place where the actual application is executed. However, the application is only invoked once an HTTP request is processed to the GAE via a web browser or some other interface, meaning that the application is not constantly running if no invocation or processing has been done. In case of such an HTTP request, the request handler forwards the request and the GAE selects one out of many possible Google servers where the application is then instantly deployed and executed for a certain amount of time (8). The application may then do some computing and return the result back to the GAE request handler which forwards an HTTP response to the client. It is important to understand that the application runs completely embedded in this described sandbox environment but only as long as requests are still coming in or some processing is done within the application. The reason for this is simple: Applications should only run when they are actually computing, otherwise they would allocate precious computing power and memory without need. This paradigm shows already the GAE's potential in terms of scalability. Being able to run multiple instances of one application independently on different servers guarantees for a decent level of scalability. However, this highly flexible and stateless application execution paradigm has its limitations. Requests

are processed no longer than 30 seconds after which the response has to be returned to the client and the application is removed from the runtime environment again (8). Obviously this method

accepts that for deploying and starting an application each time a request is processed, an additional lead time is needed until the application is finally up and running. The GAE tries to encounter this problem by caching the application in the server memory as long as possible, optimizing for several subsequent requests to the same application. The type of runtime environment on the Google servers is dependent on the programming language used. For Java or other languages that have support for Java-based compilers (such as JRuby, Rhino and Groovy) a Java-based Java Virtual Machine (JVM) is provided. Also, GAE fully supports the Google Web Toolkit (GWT), a framework for rich web applications. For Python and related frameworks a Python-based environment is used.

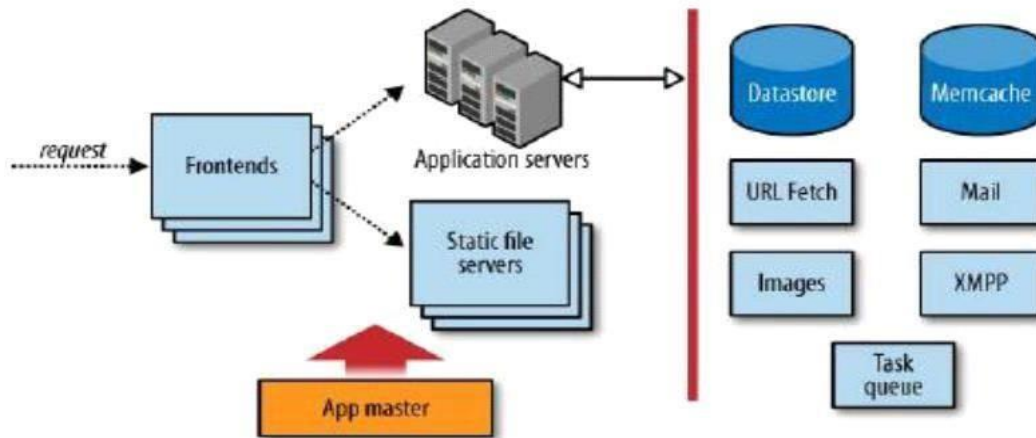


FIGURE 4: STRUCTURE OF GOOGLE APP ENGINE (13)

Persistence and the datastore

As previously discussed, the stateless execution of applications creates the need for a datastore that provides a proper way for persistence. Traditionally, the most popular way of persisting data in web applications has been the use of relational databases. However, setting the focus on high flexibility and scalability, the GAE uses a different approach for data persistence, called *Bigtable* (14). Instead of rows found in a relational database, in Google's *Bigtable* data is stored in *entities*. Entities are always associated with a certain *kind*. These entities have *properties*, resembling columns in relational database schemes. But in contrast to relational databases, entities are actually schemaless, as two entities of the same kind not necessarily have to have the same properties or even the same type of value for a certain property.

The most important difference to relational databases is however the querying of entities within a Bigtable datastore. In relational databases queries are processed and executed against a database at application runtime. GAE uses a different approach here. Instead of processing a query at application runtime, queries are pre-processed during compilation time when a corresponding index is created. This index is later used at application runtime when the actual query is executed. Thanks to the index, each query is only a simple table scan where only the exact filter value is searched. This method makes queries very fast compared to relational databases while updating entities is a lot more expensive.

Transactions are similar to those in relational databases. Each transaction is atomic, meaning that it either fully succeeds or fails. As described above, one of the advantages of the GAE is its scalability through concurrent instances of the same application. But what happens when two instances try to start transactions trying to alter the same entity? The answer to this is quite simple: Only the first instance gets access to the entity and keeps it until the transaction is completed or eventually failed. In this case the second instance will receive a concurrency failure exception. The GAE uses a method of handling such parallel transactions called optimistic concurrency control. It simply denies more than one altering transaction on an entity and implicates that an application running within the GAE should have a mechanism trying to get write access to an entity multiple times before finally giving up.

Heavily relying on indexes and optimistic concurrency control, the GAE allows performing queries very fast even at higher scales while assuring data consistency.

Services

As mentioned earlier, the GAE serves as an abstraction of the underlying hardware and operating system layers. These abstractions are implemented as services that can be directly called from the actual application. In fact, the datastore itself is as well a service that is controlled by the runtime environment of the application.

MEM CACHE

The platform innate memory cache service serves as a short-term storage. As its name suggests, it stores data in a server's memory allowing for faster access compared to the datastore. Memcache is a non-persistent data store that should only be used to store temporary data within a series of computations. Probably the most common use case for Memcache is to store session specific data (15). Persisting session information in the datastore and executing queries on every page interaction is highly inefficient over the application lifetime, since session-owner instances are unique per session (16). Moreover, Memcache is well suited to speed up common datastore queries (8). To interact with the Memcache

GAE supports JCache, a proposed interface standard for memory caches (17).

URL FETCH

Because the GAE restrictions do not allow opening sockets (18), a URL Fetch service can be used to send HTTP or HTTPS requests to other servers on the Internet. This service works asynchronously, giving the remote server some time to respond while the request handler can do

other things in the meantime. After the server has answered, the URL Fetch service returns response code as well as header and body. Using the Google Secure Data Connector an application can even access servers behind a company's firewall (8).

MAIL

The GAE also offers a mail service that allows sending and receiving email messages. Mails can be sent out directly from the application either on behalf of the application's administrator or on behalf of users with Google Accounts. Moreover, an application can receive emails in the form of HTTP requests initiated by the App Engine and posted to the app at multiple addresses. In contrast to incoming emails, outgoing messages may also have an attachment up to 1 MB (8).

XMPP

In analogy to the mail service a similar service exists for instant messaging, allowing an application to send and receive instant messages when deployed to the GAE. The service allows communication to and from any instant messaging service compatible to XMPP (8), a set of open technologies for instant messaging and related tasks (19).

IMAGES

Google also integrated a dedicated image manipulation service into the App Engine. Using this service images can be resized, rotated, flipped or cropped (18). Additionally it is able to combine several images into a single one, convert between several image formats and enhance photographs. Of course the API also provides information about format, dimensions and a histogram of color values (8).

USERS

User authentication with GAE comes in two flavors. Developers can roll their own authentication service using custom classes, tables and Memcache or simply plug into Google's Accounts service.

Since for most applications the time and effort of creating a sign-up page and store user passwords is

not worth the trouble (18), the User service is a very convenient functionality which gives an easy method for authenticating users within applications. As byproduct thousands of Google Accounts are leveraged. The User service detects if a user has signed in and otherwise redirect the user to a sign-in page. Furthermore, it can detect whether the current user is an administrator, which facilitates implementing admin-only areas within the application (8).

OAUTH

The general idea behind OAuth is to allow a user to grant a third party limited permission to access protected data without sharing username and password with the third party. The OAuth specification separates between a consumer, which is the application that seeks permission on accessing protected data, and the service provider who is storing protected data on his users' behalf (20). Using Google Accounts and the GAE API, applications can be an OAuth service provider (8).

SCHEDULED TASKS AND TASK QUEUES

Because background processing is restricted on the GAE platform, Google introduced task queues as another built-in functionality (18). When a client requests an application to do certain steps, the application might not be able to process them right away. This is where the task queues come into play. Requests that cannot be executed right away are saved in a task queue that controls the correct sequence of execution. This way, the client gets a response to its request right away, possibly with the indication that the request will be executed later (13). Similar to the concept of task queues are cron jobs. Borrowed from the UNIX world, a GAE cron job is a scheduled job that can invoke a request handler at a pre-specified time (8).

BLOBSTORE

The general idea behind the blobstore is to allow applications to handle objects that are much larger than the size allowed for objects in the datastore service. Blob is short for binary large object and is designed to serve large files, such as video or high quality images. Although blobs can have up to 2 GB they have to be processed in portions, one MB at a time. This restriction was introduced to smooth the curve of datastore traffic. To enable queries for blobs, each has a corresponding blob info record which is persisted in the datastore (8), e. g. for creating an image database.

ADMINISTRATION CONSOLE

The administration console acts as a management cockpit for GAE applications. It gives the developer real-time data and information about the current performance of the deployed application and is used to upload new versions of the source code. At this juncture it is possible to test new versions of the

application and switch the versions presented to the user. Furthermore, access data and logfiles can be viewed. It also enables analysis of traffic so that quota can be adapted when needed. Also

the status of scheduled tasks can be checked and the administrator is able to browse the applications datastore and manage indices (8).

App Engine for Business

While the GAE is more targeted towards independent developers in need for a hosting platform for their medium-sized applications, Google's recently launched App Engine for Business tries to target the corporate market. Although technically mostly relying on the described GAE, Google added some enterprise features and a new pricing scheme to make their cloud computing platform more attractive for enterprise customers (21). Regarding the features, App Engine for Business includes a central development manager that allows a central administration of all applications deployed within one company including access control lists. In addition to that Google now offers a 99.9% service level agreement as well as premium developer support. Google also adjusted the pricing scheme for their corporate customers by offering a fixed price of \$8 per user per application, up to a maximum of \$1000, per month. Interestingly, unlike the pricing scheme for the GAE, this offer includes unlimited processing power for a fixed price of \$8 per user, application and month. From a technical point of view, Google tries to accommodate for established industry standards, by now offering SQL database support in addition to the existing Bigtable datastore described above (8).

APPLICATION DEVELOPMENT USING GOOGLE APP ENGINE

General Idea

In order to evaluate the flexibility and scalability of the GAE we tried to come up with an application that relies heavily on scalability, i.e. collects large amounts of data from external sources. That way we hoped to be able to test both persistency and the gathering of data from external sources at large scale. Therefore our idea has been to develop an application that connects people's delicious bookmarks with their respective Facebook accounts. People using our application should be able to see what their

Facebook friends' delicious bookmarks are, provided their Facebook friends have such a delicious account. This way a user can get a visualization of his friends' latest topics by looking at a generated tag cloud giving him a clue about the most common and shared interests.

PLATFORM AS A SERVICE: GOOGLE APP ENGINE:--

The Google cloud, called Google App Engine, is a 'platform as a service' (PaaS) offering. In contrast with the Amazon infrastructure as a service cloud, where users explicitly provision virtual machines and control them fully, including installing, compiling and running software on

them, a PaaS offering hides the actual execution environment from users. Instead, a software platform is provided along with an SDK, using which users develop applications and deploy them on the cloud. The PaaS platform is responsible for executing the applications, including servicing external service requests, as well as running scheduled jobs included in the application. By making the actual execution servers transparent to the user, a PaaS platform is able to share *application* servers across users who need lower capacities, as well as automatically scale resources allocated to applications that experience heavy loads. Figure 5.2 depicts a user view of Google App Engine. Users upload code, in either Java or Python, along with related files, which are stored on the Google File System, a very large scale fault tolerant and redundant storage system. It is important to note that an application is immediately available on the internet as soon as it is successfully uploaded (no virtual servers need to be explicitly provisioned as in IaaS).

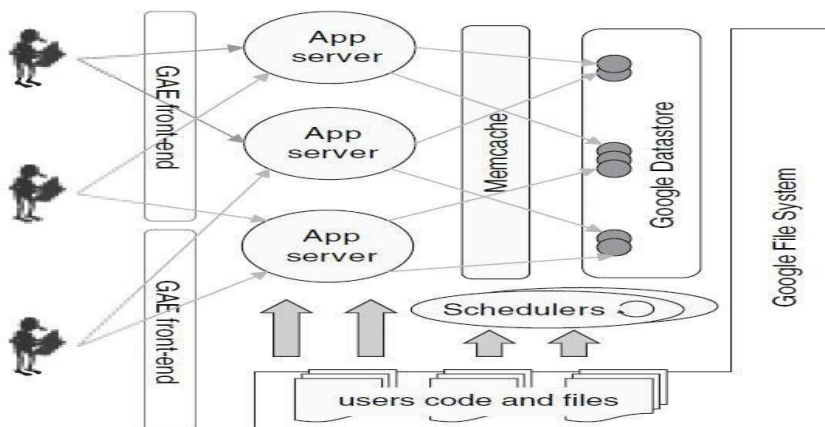


FIGURE 5.2. Google App Engine

Resource usage for an application is metered in terms of web requests served and CPU-hours actually spent executing requests or batch jobs. Note that this is very different from the IaaS model: A PaaS application can be deployed and made globally available 24×7, but charged only when *accessed*

(or if batch jobs run); in contrast, in an IaaS model merely making an application continuously available incurs the full cost of keeping at least some of the servers running all the time. Further, deploying applications in Google App Engine is free, within usage limits; thus applications can be developed and tried out free and begin to incur cost only when actually accessed by a sufficient volume of requests. The PaaS model enables Google to provide such a free service because applications do not run in

dedicated virtual machines; a deployed application that is not accessed merely consumes storage for its code and data and expends no CPU cycles.

GAE applications are served by a large number of web servers in Google's data centers that execute requests from end-users across the globe. The web servers load code from the GFS into memory and serve these requests. Each request to a particular application is served by any one of GAE's web servers; there is no guarantee that the same server will serve requests to any two requests, even from the same HTTP session. Applications can also specify some functions to be executed as batch jobs which are run by a scheduler.

Google Datastore:--

Applications persist data in the Google Datastore, which is also (like Amazon SimpleDB) a non-relational database. The Datastore allows applications to define structured types (called `_kinds`) and store their instances (called `_entities`) in a distributed manner on the GFS file system. While one can view Datastore `_kinds` as table structures and entities as records, there are important differences between a relational model and the Datastore, some of which are also illustrated in Figure 5.3.

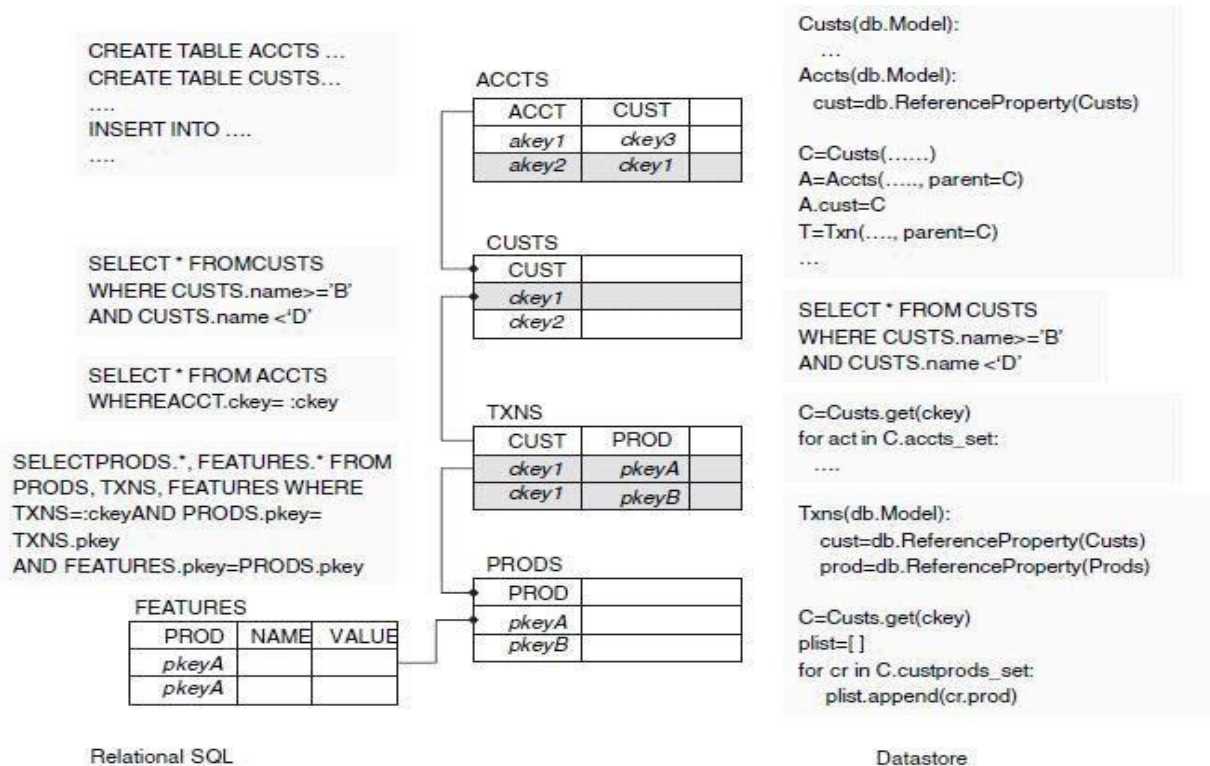


FIGURE 5.3. Google Datastore

Unlike a relational schema where all rows in a table have the same set of columns, all entities of a `_kind` need not have the same properties. Instead, additional properties can be added to any entity. This feature is particularly useful in situations where one cannot foresee all the potential properties in

a model, especially those that occur occasionally for only a small subset of records. For example, a model storing

`_products` of different types (shows, books, etc.) would need to allow each product to have a different set of features. In a relational model, this would probably be implemented using a separate FEATURES table, as shown on the bottom left of Figure 5.3. Using the Datastore, this table (`_kind`) is not required; instead, each product entity can be assigned a different set of properties at runtime. The Datastore allows simple queries with conditions, such as the first query shown in Figure 5.3 to retrieve all customers having names in some lexicographic range. The query syntax (called GQL) is essentially the same as SQL, but with some restrictions. For example, all inequality conditions in a query must be on a single property; so a query that also filtered customers on, say, their `_type`, would be illegal in GQL but allowed in SQL.

Relationships between tables in a relational model are modeled using foreign keys. Thus, each account in the ACCTS table has a pointer *ckey* to the customer in the CUSTS table that it belongs to. Relationships are traversed via queries using foreign keys, such as retrieving all accounts for a particular customer, as shown. The Datastore provides a more object-oriented approach to relationships in persistent data. Model definitions can include references to other models; thus each entity of the Accts

`_kind` includes a reference to its customer, which is an entity of the Custs `_kind`. Further, relationships defined by such references can be traversed in *both* directions, so not only can one directly access the customer of an account, but also *all* accounts of a given customer, without executing any query operation, as shown in the figure.

GQL queries *cannot* execute joins between models. Joins are critical when using SQL to efficiently retrieve data from multiple tables. For example, the query shown in the figure retrieves details of all products bought by a particular customer, for which it needs to join data from the transactions (TXNS), products (PRODS) and product features (FEATURES) tables. Even though GQL does not allow joins, its ability to traverse associations between entities often enables joins to be avoided, as shown in the figure for the above example: By storing references to customers and products in the Txns model, it is possible to retrieve all transactions for a given customer through a reverse traversal of the customer reference. The product references in each transaction then yield all products and their features (as discussed earlier, a separate Features model is not required because of schema

flexibility). It is important to note that while object relationship traversal can be used as an alternative to joins, this is not always possible, and when required joins may need to be explicitly executed by application code.

The Google Datastore is a distributed object store where objects (entities) of all GAE applications are maintained using a large number of servers and the GFS distributed file system. From a user perspective, it is important to ensure that in spite of sharing a distributed storage scheme with many other users, application data is (a) retrieved efficiently and (b) atomically updated. The Datastore provides a mechanism to group entities from different `_kinds` in a hierarchy that is used for both these purposes. Notice that in Figure 5.3 entities of the `Accts` and `Txns` `_kinds` are instantiated with a parameter `_parent` that specifies a particular customer entity, thereby linking these three entities in an `_entity group`. The Datastore ensures that all entities belonging to a particular group are stored close together in the distributed file system (we shall see how in Chapter 10). The Datastore allows processing steps to be grouped into transactions wherein updates to data are guaranteed to be

atomic; however this also requires that each transaction only manipulates entities belonging to the same entity group. While this transaction model suffices for most on line applications, complex batch updates that update many unrelated entities cannot execute atomically, unlike in a relational database where there are no such restrictions.

Amazon SimpleDB:--

Amazon SimpleDB is also a nonrelational database, in many ways similar to the Google Datastore.

SimpleDB `_domains` correspond to `_kinds`, and `_items` to entities; each item can have a number of attribute-value pairs, and different items in a domain can have different sets of attributes, similar to Datastore entities. Queries on SimpleDB domains can include conditions, including inequality conditions, on any number of attributes. Further, just as in the Google Datastore, joins are not permitted. However, SimpleDB does not support object relationships as in Google Datastore, nor does it support transactions. It is important to note that all data in SimpleDB is replicated for redundancy, just as in

GFS. Because of replication, SimpleDB features an `_eventual consistency` model, wherein data is guaranteed to be propagated to at least one replica and will eventually reach all replicas, albeit with some delay. This can result in perceived inconsistency, since an immediate read following a write may not always yield the result written. In the case of Google Datastore on the other hand, writes succeed only when all replicas are updated; this avoids inconsistency but also makes writes slower.

CONCLUSION :

Cloud Computing remains the number one hype topic within the IT industry at present. Our evaluation of the Google App Engine has shown both functionality and limitations of the platform. Developing and deploying an application within the GAE is in fact quite easy and in a way shows the progress that software development and deployment has made. Within our application we were able to use the abstractions provided by the GAE without problems, although the concept of Bigtable requires a big change in mindset when developing. Our scalability testing showed the limitations of the GAE at this point in time. Although being an extremely helpful feature and a great USP for the GAE, the built-in scalability of the GAE suffers from both purposely-set as well as technical restrictions at the moment. Coming back to our motivation of evaluating the GAE in terms of its sufficiency for serious large-scale applications in a professional environment, we have to conclude that the GAE not (yet) fulfills business needs for enterprise applications at present.
