## Weekly Questions

**Week 1**

**[1.0] [5 points] The evolution of Cloud Computing has been compared to the evolution of electricity supply as a utility. Describe specific problems that Cloud Computing solves as compared to businesses running their own data centres.**

When businesses run their own data canters, they will face the following problems:

* Difficult to dimension – Load may vary considerably, so it will waste resources.
* It is expensive - They should invest to buy hardware, expertise and maintenance.
* Difficult to scale - Scale up and scale down is difficult.

But with cloud,

* Cheaper to run one big data center than many small ones
* High utilization
* No investment in data center
* pay-as-you-go model
* Thousands of computers available on demand; add more within seconds

**[2.0] [5 points] Describe the different categories of services (XaaS) cloud computing can provide with specific examples of each service.**

The most common types are Saas, Paas, Iaas.

Saas: Software as a service. Cloud provide an entire application and customer pay cloud provider, such as Google apps, Salesforce.com, Microsoft Office.

Pass: Platform as a service. Cloud provides middleware/infrastructure. Customer pays SaaS provider for the service and SaaS provider pays the cloud for the infrastructure, such as Windows Azure, Google App Engine.

Iaas: Infrastructure as a Service. Cloud directly provides raw computing resources, such as virtual machine, blade server and hard disk. For example, the Amazon Web Services, Rackspace Cloud, GoGridProvide are all Iass.

And there are certain services within these categories can be offered such as Dass, Caas.

**[3.0] [10 points] An established financial company is about to launch their new banking application. Give 5 reasons why the company should use their own data centre rather than cloud computing.**

- Data confidentiality. The bank data is very important and private. As a third party, cloud increase the risk of bank data leakage.

- Data auditability. It’s difficult to make sure the cloud will change the data, for example, change the balance of their own card in this bank.

- Regulation, such as Sarbanes-Oxley act. The finance information is sensitive.

- Availability. The bank application will shut down if there is an outage in the cloud.

- Internet network is poor. Afraid of having downtime because of the network.

**[4.0] [20 points] Describe the concepts of** **vertical and horizontal scale.** **Describe 2 different ways in which you could scale a web application horizontally. Describe a potential architecture to scale the database to handle the scaling out of the web servers.**

1. Concept

Vertical : Add more power to now machine, such as upgrading the CPUs, scale the memory, storage, or network speed. Data sharing is less expensive, but it is impossible to increase it indefinitely on one machine.

Horizontal: Add more machines, and it becomes a cluster or even a data center. No limit for peak handling. Redundancy is easy to create so less downtime.

2. Scale application horizontally

1) Adding more instances within a server, improve ability to deal with requests

simultaneously.

2) Deploy multiple server. Adding more machines in the pool of existing resources, choose

the closest and available machines to execute the computation.

3. use database sharding architecture

Database sharding is an architecture to split the primary database into multiple databases, including two types of sharding techniques, vertical and horizontal sharding. For example, in horizontal sharding, tables are taken out and placed on different machines, each with the same columns but different rows.

**[5.0] [20 points]** **How could a mobile device benefit from cloud computing? Explain the reasons or provide your arguments supporting the contrary. Discuss several cloud applications for mobile devices; explain which one of the three cloud computing delivery models, SaaS, PaaS, or IaaS, would be used by each one of the applications and why.**

1. Benefit and reason

An application can work on the multiple types of mobile devices by shifting their program data (which is typically saved on the mobile's servers) to a cloud-based server.

2. Cloud applications and their models

Here are two examples for cloud-based application. The first one is Dropbox, which is SaaS. Dropbox is a file-hosting service that offers cloud storage. It lets the users access their files in the ‘Dropbox’ from their devices, which can be synced to other computers or mobile devices.

The second one is Amazon Cloud Player, which is SaaS too. It is used to store and play MP3 files. Here the ‘Cloud Drive’ acts as a hard drive set in the cloud. Users can play their MP3 files via the web.

**Week 2**

**[6.0] [20 points] Describe the steps which you would take on AWS and the decisions that would need to be made to create, configure and run a Virtual Machine Instance.**

(参考lab2 report)

Create an EC2 instance using awscli

[1] Create a security group

aws ec2 create-security-group --group-name <student number>-sg --description "security group for development environment"

Note: this will use the default VPC (you will learn about this later in the course) – if you want to specify another VPC, you would use --vpc-id vpc-xxxxxxxx

Note the security group id that is created

[2] Authorise inbound traffic for ssh

aws ec2 authorize-security-group-ingress --group-name <student number>-sg --protocol tcp --port 22 --cidr 0.0.0.0/0

[3] Create a key pair that will allow you to ssh to the EC2 instance

aws ec2 create-key-pair --key-name <student number>-key --query 'KeyMaterial' --output text > <student number>-key.pem

To use this key on Linux, copy the file to a directory ~/.ssh and change the permissions to:

chmod 400 <student number>-key.pem

[4] Create the instance and note the instance id

aws ec2 run-instances --image-id ami-d38a4ab1 --security-group-ids <student number>-sg --count 1 --instance-type t2.micro --key-name <student number>-key --query 'Instances[0].InstanceId'

Optional: Add a tag to your Instance

aws ec2 create-tags --resources i-??????? --tags Key=Name,Value=<student number>

// 18.04 ami-176aa375

[5] Get the public IP address

aws ec2 describe-instances --instance-ids i-<instance id from above> --query 'Reservations[0].Instances[0].PublicIpAddress'

[6] Connect to the instance

ssh -i <student number>-key.pem ubuntu@<IP Address>

**[7.0] [10 points] Describe EBS and what features it offers.**

AWS EBS storage is allocated in volumes. A volume is a 'virtual disk' (size: 1GB - 1TB). Iit’s a raw block device and it can be attached to an instance (but only one at a time, an instance can access multiple volumes).

Features:

1. Replicated across multiple servers - so data is not lost if a single server fails. The EBS provides maximum data protection as the failure rate is between 0.1 to 0.2% a year.

2. Automatically apply encryption - AWS EBS encryption offers encryption of EBS data volumes, boot volumes, and snapshots.

3. Snapshot – create a copy of the EBS volume(s) to recreate at any time

4. Elastic Volumes – can increase size automatically without using a snapshot. It works with no downtime or performance impact.

5. Underlying technology is SSD or HDD - two types of volumes that are SSD-Backed delivering low-latency with highest IOPS and HDD-Backed for streaming sequential access for highest throughput. SSD better for disk intensive applications.

6. EBS-Optimized – guarantees certain levels (provisioned IOPS) of throughput (500 and 10,000 Mbps) (SATA drive around 5,000 Mbps)

**[8.0] [10 points] What is CLI and Boto? What are advantages of using CLI? How does Boto function helps in AWS operation?**

1. Concept of CLI and Boto

CLI is AWS Command Line Interface, which allows you to type commands into a terminal window and write scripts interact with you AWS Cloud Account.

Boto is a software development kit (SDK) designed to improve the use of the Python programming language in Amazon Web Services, which converts application programming interface (API) responses from AWS into Python classes.

2. Advantage of using CLI

Greater control of an OS or application; faster management of many operating systems; ability to store scripts to automate regular tasks; basic command-line interface knowledge to help with troubleshooting, such as network connection issues.

3. Boto function

In the lab2, we import the boto3 to the python file and use many functions to operate with EC2. In the lab3, we import it again and do many functions about S3. (结合实验报告)

**Week 3**

**[8.0] [10 points] Describe what** **virtualisation is and describe the characteristic attributes of the different types of virtualisation (Language, Operating System and** **Hardware).**

1. Concept

It can be:

A machine that’s implemented in software, rather than hardware. A self-contained environment that acts like a computer. An abstract specification for a computing device (instruction set, etc.)

4 properties:

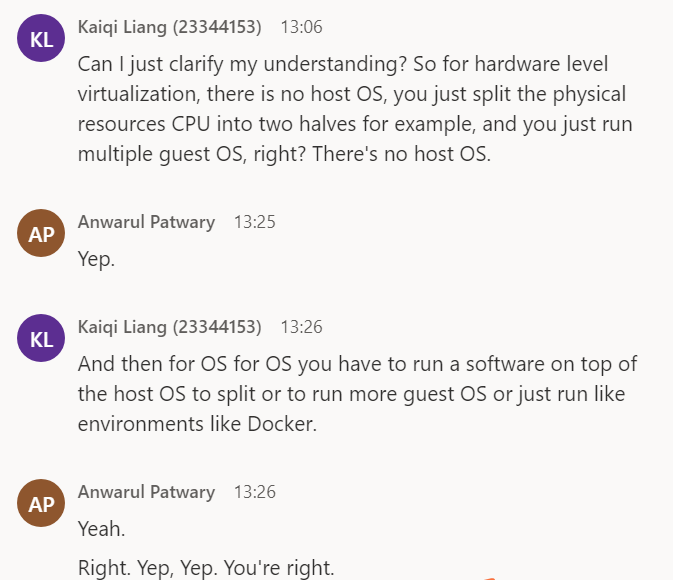
Partitioning – can run multiple OS on one physical machine and can divide resources between VMS

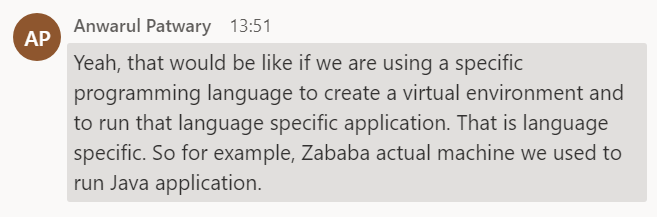
Isolation – provide fault and security isolation at the hardware level and preserve performance with advanced resource controls

Encapsulation - Save the entire state of a virtual machine to files and Move and copy virtual machines as easily as moving and copying files.

Hardware Independence - Provision or migrate any virtual machine to any physical server.

2. Common distinction:





(language-based) virtual machines

Instruction set usually does not resemble any existing architecture

Java VM, .Net CLR, many others

Virtual Machine Monitors (VMM) or Hypervisor

instruction set fully or partially taken from a real architecture

Other types of virtualization

Web Server

Virtual hosts using a different hostname, configuration file

Virtual Environments

Java, Python and Ruby amongst others have the ability to configure separate versions of language and libraries

Containers

Like VMs but run in user space and packages binaries and libraries

Docker, Kubernetes, AWS Containers (Docker)

Serverless Environments

Code is executed in response to an event, including HTTP

**[9.0] [10 points] Describe what containers are with reference to Docker and discuss their similarities and differences from operating system virtualisation perspective as provided by VirtualBox.**

（\*）

Docker is Originally based on Linux Containers (LXC) but now on runC

In traditional virtualization, a hypervisor virtualizes physical hardware. The result is that each virtual machine contains a guest OS, a virtual copy of the hardware that the OS requires to run and an application and its associated libraries and dependencies. VMs with different operating systems can be run on the same physical server. For example, a VMware VM can run next to a Linux VM, which runs next to a Microsoft VM, etc.

Instead of virtualizing the underlying hardware, containers virtualize the operating system (typically Linux or Windows) so each individual container contains only the application and its libraries and dependencies. Containers are small, fast, and portable because, unlike a virtual machine, containers do not need to include a guest OS in every instance and can, instead, simply leverage the features and resources of the host OS.

Just like virtual machines, containers allow developers to improve CPU and memory utilization of physical machines. Containers go even further, however, because they also enable microservice architectures, where application components can be deployed and scaled more granularly. This is an attractive alternative to having to scale up an entire monolithic application because a single component is struggling with load.

**[10.0] [10 points] There was an evolution of operating system during the half century from 1960 to 2010. Identify the virtualisation milestones in this above evolution and explain them briefly.**

First VM architected by IBM in 1972 VM/370 to provide full VM of mainframe machine

1997 Virtual PC for Mac by Connectix

1999 VMware’s VMware Virtual Platform

2003 Open Source hypervisor Xen

2005 VMware Player – free VM player

2007 VirtualBox

（缺explain）

**Week 4**

**[10.0] [5 points] You are asked to store data about music albums in a DynamoDB table. For each album, you need to record the title of the album and the artist name. Describe the commands you would use to create a table to store such information and write an entry to that table in DynamoDB.**

1.Create a table:

aws dynamodb create-table --table-name Music

--attribute-definitions \

AttributeName=Artist,AttributeType=S \

AttributeName=SongTitle,AttributeType=S

--key-schema AttributeName=Artist,KeyType=HASH \

AttributeName=SongTitle,KeyType=RANGE \

--provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1 \

--endpoint-url=http://localhost:8000

aws dynamodb scan --table-name Music \

--endpoint-url=http://localhost:8000

2. Write an entry:

aws dynamodb put-item \

--table-name Music \

--item \ '{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"},

"AlbumTitle": {"S": "Somewhat Famous"}}’ \

--return-consumed-capacity TOTAL --endpoint-url=http://localhost:8000

3. Validate

Scan:

aws dynamodb scan --table-name Music \

--endpoint-url=http://localhost:8000

Query:

{ "Artist": { "AttributeValueList": [{ "S": "No One You Know" } ], "ComparisonOperator": "EQ" },

"SongTitle": { "AttributeValueList": [{ "S": "Call Me Today" } ], "ComparisonOperator": "EQ"}

}

aws dynamodb query --table-name Music --key-conditions= file://key-conditions.json \

--endpoint-url=http://localhost:8000

**[11.0] [5 points] Describe how S3 handles consistency of objects and how this approach affects the state of objects when they are read using a GET.**

S3 handles consistency through versioning rather than locking

The idea: every bucket + key maps to a list of versions

[bucket+key]  [object v1] [object v2] [object v3] …

Each time we PUT an object, it gets a new version

The last-received PUT overwrites any previous ones!

When we GET:

An unversioned request likely receives the last version – but this is not guaranteed depending on propagation delays

A request for bucket + key + version uniquely maps to a single object

**[12.0] [5 points] What are the core components of DynamoDB**

1.Tables

Collection of data (e.g. representing entities): People

Schemaless

2.Items

Group of attributes that is uniquely identifiable

Person in People table

Physically a JSON document

3.Atrributes

Fundamental data element

Limited types of data: Scalar, Document, Sets

LastName in Person item

One or more attributes make a primary key (unique)

**[13.0] [5 points] When a Bucket is created, AWS allows the specification of a number of features that can be managed. What are the key properties and features?**

1.Properties

Users with special permissions of Read and Write

Grant public read access to the bucket

2.Features

Lifecycle - Transition objects that are infrequently accessed (or after fixed time) to cold storage

Replication - Automatically copy objects to another bucket in a different region

Analytics - Suggest how to manage objects based on access patterns

Metrics - Stats on operations on objects in the bucket

Inventory - Provide a regular snapshot of contents of bucket

**[14.0] [5 points] We can leave S3 buckets open to public. Is this suitable for a specific application? Why and why not? Justify your answer**

Not suitable. Because S3 buckets have left open to public, however:

Nov 2017 Contractor exposes personally identifiable data from 50,000 Australians (AMP, UGL, Rabobank)

Nov 2017 Accenture leaked corporate information

Alteryx exposes data on 120 million US households

March 2018 Medical Data of 33,000 patients

When this important personal information is leaked, it may raise some information security as well as personal privacy issues. It can also lead to a crisis of trust for the companies involved.

So, we should Access to S3 buckets and objects controlled by user policies, bucket policy, bucket ACL (Access Control List) and object ACL.

**Week 5**

**[14] [10 points] An organisation has 5 departments and has separated out each of the IAM users into separate groups using paths following the pattern companybucket/department1/\*, companybucket /department2/\*, companybucket /department3/\* etc.  
  
Their IAM account names follow the pattern [user@department1.company.com](mailto:user@department1.company.com" \t "https://lms.uwa.edu.au/webapps/blackboard/content/_blank), [user@department2.company.com](mailto:user@department2.company.com" \t "https://lms.uwa.edu.au/webapps/blackboard/content/_blank) etc.  
  
You are tasked with securing a bucket that contains a folder for each of 5 departments in an organisation. Only people within a department can write to their own folder. Everyone can read from all folders.    
  
Discuss the principles that you would use to create a policy that would achieve this objective.  
  
Write the policy as a JSON file that you would use.  
  
Note: you can have individual statements for each department.**

1. Principles include:

1) use “s3:ListBucket” action to allow 5 users to use the [GET Bucket(List Objects)](https://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketGET.html) operation

2) use “s3:GetObject” action to allow 5 users to read the resource

3) use “s3:PutObject” action to allow department member to write their resource

2. JSON code:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "ObjectListRead",

"Effect": "Allow",

"Action": ["s3:ListBucket"],

"Resource": ["arn:aws:s3:::companybucket/department1","arn:aws:s3:::companybucket/department2","arn:aws:s3:::companybucket/department3","arn:aws:s3:::companybucket/department4","arn:aws:s3:::companybucket/department5"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department1.company.com”,

“aws:username”: “user@department2.company.com”,

“aws:username”: “user@department3.company.com”,

“aws:username”: “user@department4.company.com”,

“aws:username”: “user@department5.company.com”

}

}

},

{

"Sid": "ObjectRead",

"Effect": "Allow",

"Action": ["s3:GetObject"],

"Resource": ["arn:aws:s3:::companybucket/department1/\*","arn:aws:s3:::companybucket/department2/\*","arn:aws:s3:::companybucket/department3/\*","arn:aws:s3:::companybucket/department4/\*","arn:aws:s3:::companybucket/department5/\*"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department1.company.com”,

“aws:username”: “user@department2.company.com”,

“aws:username”: “user@department3.company.com”,

“aws:username”: “user@department4.company.com”,

“aws:username”: “user@department5.company.com”

}

}

},

{

"Sid": "Write1",

"Effect": "Allow",

"Action": ["s3:PutObject"],

"Resource": ["arn:aws:s3:::companybucket/department1/\*"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department1.company.com”

}

}

},

{

"Sid": "Write2",

"Effect": "Allow",

"Action": ["s3:PutObject"],

"Resource": ["arn:aws:s3:::companybucket/department2/\*"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department2.company.com”

}

}

},

{

"Sid": "Write3",

"Effect": "Allow",

"Action": ["s3:PutObject"],

"Resource": ["arn:aws:s3:::companybucket/department3/\*"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department3.company.com”

}

}

},

{

"Sid": "Write4",

"Effect": "Allow",

"Action": ["s3:PutObject"],

"Resource": ["arn:aws:s3:::companybucket/department4/\*"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department4.company.com”

}

}

},

{

"Sid": "Write5",

"Effect": "Allow",

"Action": ["s3:PutObject"],

"Resource": ["arn:aws:s3:::companybucket/department5/\*"],

”Condition”: {

“StringLike”: {

“aws:username”: “user@department5.company.com”

}

}

}

]

}

1. **[5 points]  What aspects of security does the OSI Security Architecture X.800 standard cover? Which particular components of this standard does AWS Identity and Access Management deal with?**

Authentication - assurance that communicating entity is the one claimed - for example, AWS will use multi-factor authentication (MFA) with each account

Access Control - prevention of the unauthorized use of a resource - for example, AWS use the Attribute-based access control (ABAC) , which is an authorization strategy that lets you create fine-grained permissions based on user attributes.

Data Confidentiality –protection of data from unauthorized disclosure

Data Integrity - assurance that data received is as sent by an authorized entity

Non-Repudiation - protection against denial by one of the parties in a communication

- for example, AWS use an HSM from AWS CloudHSM, which can manage cryptographic keys, encrypt and decrypt data, use cryptographic hash functions, cryptographically sign data and verify signatures.

Availability – resource accessible/usable - for example, aws can create policies by json file

1. **[5 points] Name 3 of the keys that you would find in a Policy. Explain their role. An example of a key is “Version” that specifies the version of the policy syntax and is normally “Version”: “2012-10-17”**

Sid

(Optional) The statement identifier (Sid) an arbitrary string you can use to describe the statement. The Sid in a key policy can include spaces. (You can't include spaces in an IAM policy Sid element.)

Effect

(Required) Determines whether to allow or deny the permissions in the policy statement. Valid values are Allow or Deny. If you don't explicitly allow access to a KMS key, access is implicitly denied. You can also explicitly deny access to a KMS key. You might do this to make sure that a user cannot access it, even when a different policy allows access.

Action

(Required) Specify the API operations to allow or deny. For example, the kms:Encrypt action corresponds to the AWS KMS [Encrypt](https://docs.aws.amazon.com/kms/latest/APIReference/API_Encrypt.html) operation. You can list more than one action in a policy statement. For more information, see [Permissions reference](https://docs.aws.amazon.com/kms/latest/developerguide/kms-api-permissions-reference.html).

Resource

(Required) In a key policy, the value of the Resource element is "\*", which means "this KMS key." The asterisk ("\*") identifies the KMS key to which the key policy is attached.