# Introduction to Cloud Testing

## Introduction to cloud computing

Cloud computing is an information technology (IT) paradigm that enables ubiquitous access to shared pools of configurable system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet.

Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a public utility.

It is the responsibility of the vendor, the Cloud Service Provider (CSP), to develop, own and maintain these resources and make them available to the consumers over the internet.

You, the consumer, need not know exactly where the resources are located and how it all works. It is just somewhere up in the "cloud" that the internet represents.

Example:  
• If you want to use an email service, you would need the hardware and software resources for an email server to send, receive and store your mails an email client to access the data and operations in your email server.  
• Instead, if you use a cloud based mail service like Gmail, Outlook, etc., all you need is a device, with an app or a browser, connected to the internet.

Each deployment model aims at addressing one or more concerns of the cloud consumer. Therefore, it is very important that consumers prioritize their concerns before opting for a particular model.

1. Public cloud(SaaS – Software as a service)

The public cloud is one that is available for use by the general public. Hence, it is the most common and popular deployment model available today.

Eg:Gmail

2. Private cloud

A private cloud is available only to users within a single organization.  
Eg: A cloud created for a banking organization

3. Community cloud(PaaS,IaaS,DbaaS etc)

A community cloud is a private cloud that is shared by two or more organizations havingshared concerns like security requirements, policy, and compliance considerations etc.  
Eg: A cloud created for a retail application which can be used for another retail application also

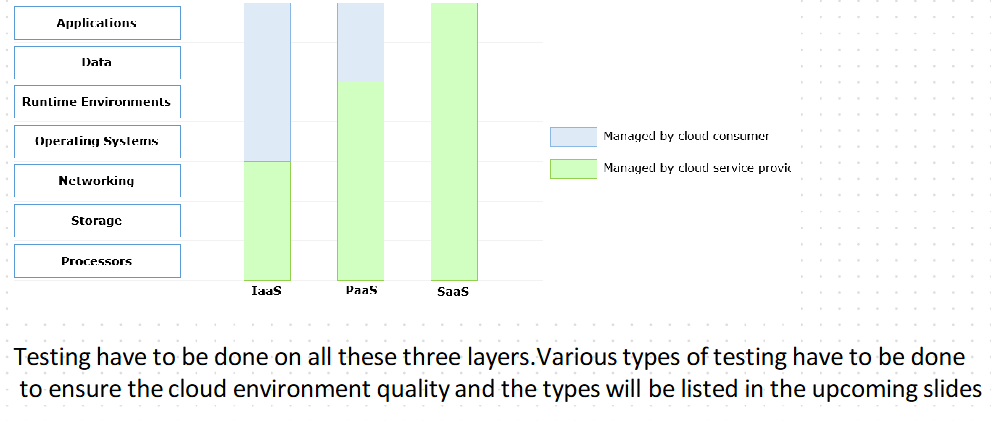
## Introduction to cloud testing

A cloud system will be having mainly three layers and testing is supposed to be done on all the three level of layers.

First layer consist of the basic high end hardware servers in which the cloud environment is build upon.This layer mostly consist of the processors,storage,network devices etc.

Second layer consists of the operating system and runtime environments which includes app creating softwares like Visual studio,JRE Engines,API services etc.. Which is used for  
creating or hosting the main application.The second layer is build upon the first layer

Third layer consist of the main application which is being hosted in the cloud.The third layer  
sustains upon the second layer and the first layer

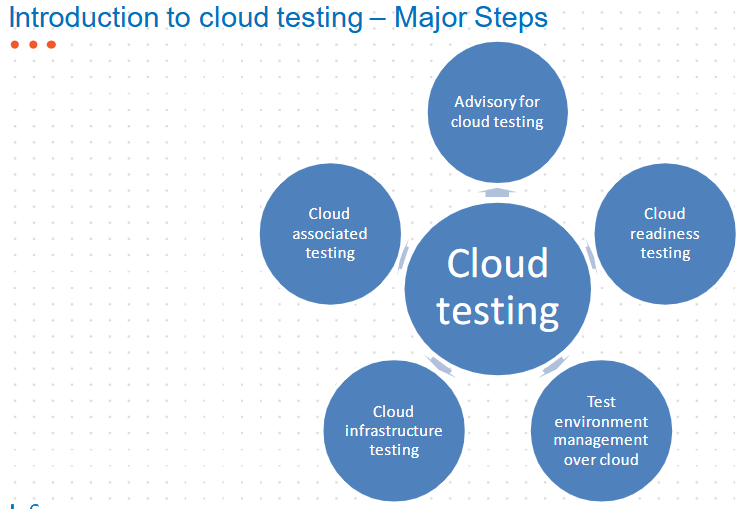


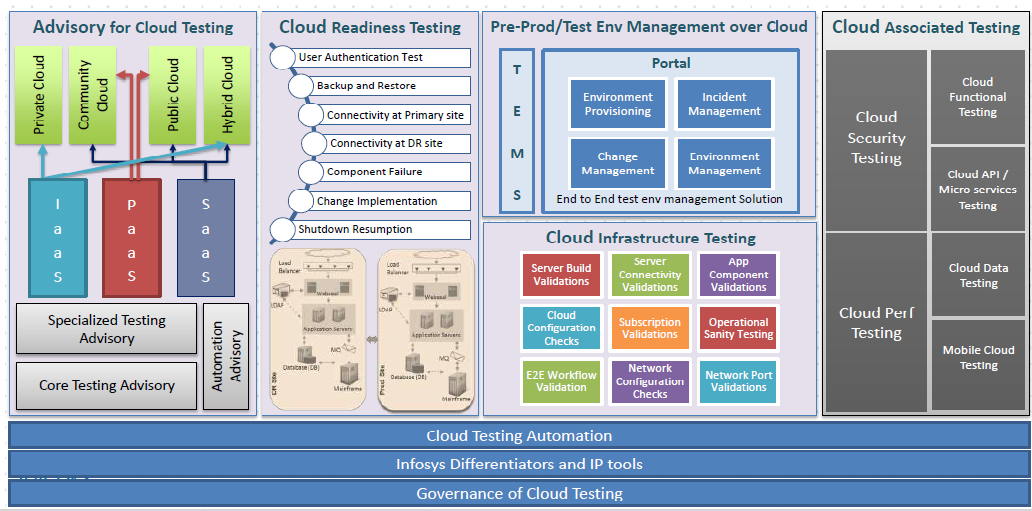
### Objectives of cloud testing:

• What to test: Test whether all the three layers of Cloud environment is functionally working according to the business requirements. Apart from functionality, performance  
security, infrastructure etc. also have to be tested.

• How to test: Testing can be done through manual or automation methods. Manual will be done on the initial build and later the whole testing process will be automated with  
the help of dedicated cloud testing tools

• Where to test: All the types of testing have to be done of cloud environment,  
middleware components and on the application being hosted on the cloud.





### Steps in Cloud testing

1. Advisory for cloud testing

Cloud Advisory provides services viz, Assessment of the current Landscape, Converting the Cloud strategy to requirements, Identify the Keys areas for Cloud readiness, Identify Confidential Business areas, Risk Assessment, Assessment of Business continuity risk during  
migration, Blue print preparation, Migration Planning, Cloud Test Plan Readiness, Performance & DR Readiness .  
This will help in deciding how to test,when to test and what to test.

2. Cloud app readiness testing

Cloud App readiness is for ensuing Application Portability validation, API validation, Container connectivity, Database Integration, Download and Upload, response time, Identity and Access Validation, Compliance validation, Usage and bills

3. Infrastructure testing

Cloud infra testing is to ensure the Server Blueprint configuration validation, Server Capacity ( SKUs /CPUs/ Storage/ Memory) validation, Network Capacity validation, Firewall Validation, DNS and Switched LAN validation, Services failure and availability, Service Integration validation, Load Balancing, Identity and Access Validation, Compliance  
validation, Usage and bills, Career SLA validation

4. Test environment management services

A centralized Test Environment Management function supports the usage of client’sintegrated end-to-end cloud based scale down version of test Environment to address many of the non-prod issues early in the life cycle thus helps arresting issues before hitting production.

5. Cloud associated testing

This consist of the major testing activites which include  
• Functional testing  
• Performance testing  
• Security testing  
• Data testing  
• Mobile testing  
• Service testing

### Functional testing

• Functional testing is done to ensure that the deployed application is meeting the functional requirements and thus by assuring the quality of the application

• For example if it is a mail application to be tested,major functionalities including the composing and sending the mail,retrieving the mail and replying to etc..

• Functional test cases can be automated using various tools which results in the faster execution of test cases

### Performance testing

Performance testing is done to ensure the performance of the application is meeting the required needs.This includes testing the below things.  
• Load  
• Stress  
• Reliability  
• Availability  
Doing performance testing through manual mode is not possible and hence we prefer automation in these scenarios. Tools like Loadrunner,Jmeter etc helps in performing  
performance testing on the application

### Security testing

Security testing is done to identify the flaws and vulnerabilities in the system to prevent the intruders from stealing or malfunctioning the system data.

Security testing covers the below areas/attacks  
• User access/roles  
• Web UI attackes like SQL injection,Password cracking  
• Data deletion confirmation  
• Vulnerabilities and attacks like Ddos attack

Automation tools like Nessus,Wireshark,Nmap etc helps in performing the security testing  
for cloud based applications.

### Data testing

Data testing is done to ensure the correctness and readiness of the data in the database.Data will be populated from user entry or from data warehousing and hence the resultant data have to be verified against the business requirements.

Below types of data need to be tested  
1. Structured – Relational tables  
2. Semi structured – xml,csv  
3. Un structured data – emails,images,videos

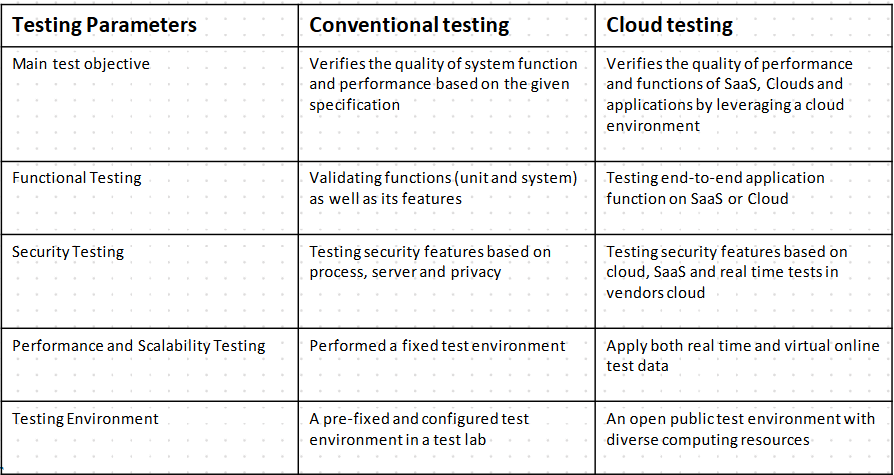
Semi structured and Un structured data falls under Bigdata technology and for that corresponding bigdata testing have to be done.Infosys have created an internal tool named  
IDTW for doing data validations including the big data testing

### Service testing

Web Services is the mechanism or the medium of communication through which two applications / machines will exchange the data irrespective of their underline architecture  
and the technology In the modern era of technology if you want to build a software application you don't need to build each and everything from scratch. There are lots of readymade services available which you can plug into your application and you can start providing those services in your application.

Web Services Testing is testing of Web services and its Protocols like SOAP & REST. To test a Webservice you can

• Test Manually  
• Create your own Automation Code  
• Use an off-the shelf automation tool like SoapUI.



### Advantages in cloud testing

• Improved asset utilization and cost control  
• For scalability it takes less time when compared with traditional testing methods  
• Datacenters are purchased as a service from cloud providers and no wastage of time in creating it  
• Better management and increased productivity.  
• Product availability will be faster to market.  
• Cleaner,green testing and save CO2 emissions as all are in virtual form in the cloud

Challenges in cloud testing  
  
• Data Security  
Since multiple users are having access to the cloud ,establishing a complete secure environment is highly difficult. Extreme measures have to be done to avoid data theft and hacking.

• Data migration  
Since the cloud consist of humongous amount of interrelated data, if migration have to be done ,it must be done with utmost care. Lot of efforts are needed to understand the  
tables, relationships etc before doing the migration process

• Performance  
Millions of users will be there for cloud app like gmail and hence the performance of the application have to be tested and maintained with great care.More effort have to be  
done on the performance engineering part to make the application the best

# CLOUD TESTING

## Introduction to Cloud computing

**Cloud computing** is an information technology (IT) paradigm that enables ubiquitous access to shared pools of configurable system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet.

Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a public utility.

It is the responsibility of the vendor, the **Cloud Service Provider (CSP),** to develop, own and maintain these resources and make them available to the consumers over the internet.

You, the consumer, need not know exactly where the resources are located and how it all works. It is just somewhere up in the "cloud" that the internet represents.

**Example:**

• If you want to use an email service, you would need the hardware and software resources for an email server to send, receive and store your mails an email client to access the data and operations in your email server.

• Instead, if you use a cloud based mail service like Gmail, Outlook, etc., all you need is a device, with an app or a browser, connected to the internet.

Each deployment model aims at addressing one or more concerns of the cloud consumer. Therefore, it is very important that consumers prioritize their concerns before opting for a particular model.

**1.Public cloud(SaaS – Software as a service)**

The public cloud is one that is available for use by the general public. Hence, it is the most common and popular deployment model available today.

Eg: Gmail

**2.Private cloud**

A private cloud is available only to users within a single organization.

Eg: A cloud created for a banking organization

**3. Community cloud(PaaS,IaaS,DbaaS etc)**

A community cloud is a private cloud that is shared by two or more organizations having shared concerns like security requirements, policy, and compliance considerations etc.

Eg: A cloud created for a retail application which can be used for another retail application also

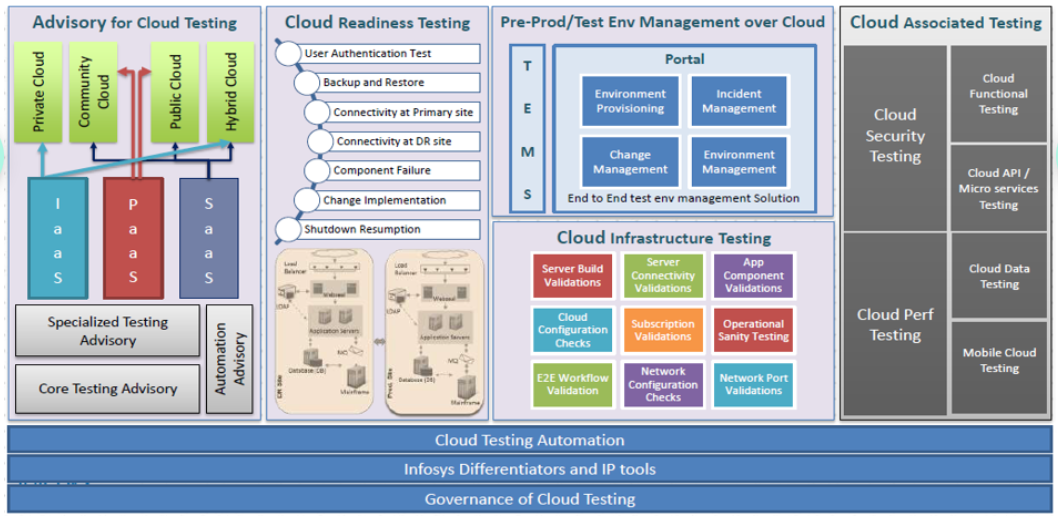
A cloud system will be having mainly three layers and testing is supposed to be done on all the three level or layers.

**First layer** consist of the basic high end hardware servers in which the cloud environment is build upon. This layer mostly consist of the processors, storage, network devices etc.

**Second layer** consists of the operating system and runtime environments which includes app creating softwares like Visual studio, JRE Engines, API services etc. This layer is used for creating or hosting the main application. The second layer is build upon the first layer.

**Third layer** consist of the main application which is being hosted in the cloud. The third layer sustains upon the second layer and the first layer.

## Introduction to Cloud testing



### Steps in Cloud Testing

**1. Advisory for cloud testing**

Cloud Advisory provides services like Assessment of the current Landscape, Conversion of the Cloud strategy to requirements, Identification of Keys areas for Cloud readiness, Identification of Confidential Business areas, Risk Assessment, Assessment of Business continuity risk during migration, Blue print preparation, Migration Planning, Cloud Test Plan Readiness, Performance & DR Readiness.

This will help in deciding how to test, when to test and what to test.

**2. Cloud app readiness testing**

Cloud App readiness is for ensuring Application Portability validation, API validation, Container connectivity, Database Integration, Download and Upload, response time, Identity and Access Validation, Compliance validation, Usage and bills.

**3. Infrastructure testing**

Cloud infra  testing is to ensure the Server Blueprint configuration validation, Server Capacity ( SKUs /CPUs/ Storage/ Memory) validation, Network Capacity validation, Firewall Validation, DNS and Switched LAN validation, Services failure and availability, Service Integration validation, Load Balancing, Identity and Access Validation, Compliance validation, Usage and bills, Career SLA validation

**4. Test environment management services**

A centralized Test Environment Management function supports the usage of client’s integrated end-to-end cloud based scale down version of test Environment to address many of the non-prod issues early in the life cycle thus helps arresting issues before hitting production.

**5. Cloud associated testing**

This consist of the major testing activities which include

•Functional testing

•Performance testing

•Security testing

•Data testing

•Mobile testing

•Service testing

Various types of testing done in the cloud world

Web Services is the mechanism or the medium of communication through which two applications / machines will exchange the data irrespective of their underline architecture and the technology.

In the modern era of technology, if you want to build a software application, you don't need build each and everything from scratch. There are lots of readymade services available. These services can be plugged into your application and they start providing those services in your application.

Web Services Testing is testing of Web services and its Protocols like SOAP & REST. To test a Webservice you can

•Test Manually

•Create your own Automation Code

•Use an off-the shelf automation tool like SoapUI.

### Advantages of Cloud Computing

Improved asset utilization and cost control

• For scalability it takes less time when compared with traditional testing methods

• Datacenters are purchased as a service from cloud providers and no wastage of time in creating it

• Better management and increased productivity

• Product availability will be faster to market

• Cleaner, green testing and save CO2 emissions as all are in virtual form in the cloud

### Challenges in Cloud Testing

**Data Security**

Multiple users are having access to the cloud. So, establishing a complete secure environment is highly difficult. Extreme measures have to be done to avoid data theft and hacking.

**• Data migration**

The cloud consists of humongous amount of interrelated data. If migration have to be done, it must be done with utmost care. Lot of efforts are needed to understand the tables, relationships etc. before doing the migration process.

**• Performance**

Millions of users  use cloud app like gmail. Hence, the performance of the application have to be tested and maintained with great care. More effort on the performance engineering part is required to make the application the best.

## Cloud app readiness testing

We are going to check for cloud app readiness in amongst the following factors.

1. Readiness Testing

2. Application Portability validation

3. API validation

4. Container connectivity

5. Database Integration

6. Response time

7. Identity and Access Validation

8. Compliance validation

9. Stability

10. BLOB Validation

### 1) Readiness Testing

Readiness testing is way to validate the first two layers of any cloud environment. In a cloud readiness testing:

* we will check for everything that is related to the network, JRE, Storage, Connectivity etc. which is going to be vital when we load app onto the cloud.
* we have to make sure of the infrastructure before loading app onto the cloud.

Like smoke testing is to software testing – cloud readiness testing is to cloud testing.

Note : More about the readiness testing will be discussed in the upcoming modules

### 2) Application Portability validation

In the context of cloud computing, Portability is about the

* ability of a customer to move and suitably adapt their applications and data between their own systems and cloud services,
* between cloud services of different cloud service providers and potentially different cloud deployment models.

Generally, portability in cloud happens in one of the following scenario:

1. **On Premise to Cloud**: It is the process of moving data, applications, or other business components from an organization’s on-premises infrastructure to the cloud.
2. **One CSP to Another**: CSP is short form for ‘Cloud Service Provider’. This is typically moving from one cloud to another. For example, from AWS to Azure or vice versa.
3. **Cloud Upgrade**: Here, cloud is upgraded for space, speed etc. In any such case, movement of data or apps in the cloud becomes unavoidable.

In all the above cases, validation has to be performed before and after the portability and also during the portability.

### 3) API validation

An Application Programming Interface (API) at its core is a formal specification that acts **as a guaranteed contract**between two separate pieces of software.

API testing involves testing the APIs directly and as part of integration testing to determine if they meet expectations for **functionality, reliability, performance, and security.**

Here is a quick list of the more common API technologies that exist in approximate chronological order:

* TCP/IP Sockets
* Remote Procedure Call (RPC)
* Common Object Request Broker Architecture (CORBA)
* Java Remote Method Invocation (RMI) and Enterprise Java Beans (EJBs)
* Microsoft Distributed Component Object Model (DCOM) –also known as ActiveX
* Web Services (SOAP then REST)

### 4) Container connectivity

The key thing to recognize with cloud containers is that they are designed to **virtualize a single application**.Two major things that we would want to validate here are

1. **Security:** A great deal of effort has been put into providing security to Containers. Containers for sensitive production applications should be treated in the same way as any other deployment when it comes to security. Inside the container is software that may have vulnerabilities. Although, this might not grant access to the underlying OS of the server, there still may be issues such as denial of service.
2. **Connectivity:** We must make sure that all the underlying components and also other cloud services are able to access the containers. Scenarios should be tested such that, even if one way to accessing the container fails, there should be another way to reach it. Also, a method has to be taken to mitigate the connectivity issues later as well.

### 5) Database Integration

* Any cloud that we use in today’s world must have a database connected to it in the backend
* These databases not only provides fetch and retrieve data but also could be used for access management as well
* These connections to cloud and the database must be checked properly and should be validated on a regular basis.
* These validations could prove vital if the cloud service depends more on the data.
* This is not a one time process of validation. As the cloud grows, there is certainty that the database could also increase in size.
* So after the scale up we need to test those integrations as well. That is, check for the connectivity of the new database with the cloud and the also the with existing databases.

### 6) Response Time

* Response time is the time taken to fulfil a response by the cloud.
* That is, if the a user is giving a request to the cloud then how fast the cloud service can be given to the user.
* This may not necessarily be outside the cloud, this response time should be validated for most of the underlying components of the cloud.
* Testing this response time will come under Cloud performance testing which will be in compliance with the software agreement.
* The response time on a cloud environment may not be similar to that for a non-cloud based infrastructure. So, care should be taken to baseline and measure the response times.
* In a public cloud, the performance tests that can impact other customers are not performed.

### 7) Identity and Access Validation

**Access Management:**

* According to Gartner, Identity and Access Management (IAM) is the security discipline that enables the right individuals to access the right resources at the right times for the right reasons.
* IAM addresses the mission-critical need to ensure appropriate access to resources across increasingly heterogeneous technology environments.
* Enterprises traditionally used on-premises IAM software to manage identity and access policies. But as the cloud service increases, the process of managing identities is getting more complex.
* Therefore, adopting cloud-based Identity-as-a-Service (IDaaS) and cloud IAM solutions becomes a logical step.

Cloud IAM typically includes the following features:

* **Single Access Control Interface:**

Cloud IAM solutions provide a clean and consistent access control interface for all cloud platform services. The same interface can be used for all cloud services.

* **Enhanced Security:**

You can define increased security for critical applications.

* **Resource-level Access Control:**

You can define roles and grant permissions to users to access resources at different granularity levels.

### 8) Compliance Validation

* Compliance validation will tell how far the SLA has been followed in the installation, operation & maintenance of cloud.
* If the cloud satisfies all the SLA then we can say that the cloud is fully compliant.
* In cloud, the data we store must be compliant to certain standards and encryption policies, these are done to protect sensitive information like health records, bank details etc. There are certain encryptions has to be used so that there is a compliance in data storage. Some examples are given below.

1. **In health sector**, few complaint standards are DICOM and HIPPA
2. **In banking sector**, Examples of industry-tested and accepted standards and algorithms for encryption include AES (128 bits and higher), TDES (minimum double-length keys), RSA (2048 bits and higher).
3. **In energy sector,** some regulations are Sarbanes-Oxley Act of 2002 (SOX), The International Traffic in Arms Regulations (ITAR) regulations, The Payment Card Industry Data Security Standard (PCI DSS) etc.

### 9) Stability

Some programming bugs directly affect the memory and cause an unexpected memory leak. This results in extensive usage of memory and a potential system crash. The best practice is to ensure that the application remain completely stable under long sustained workloads that simulates production usage patterns.

Stability test

* is performed to find out how stable a cloud system is.
* will provide useful insights on potential threats that can impact the stability of your system.
* gives us information like:

1. **Reliability**–how often does the cloud go up/down
2. **Response times** –how long does this simple test take to execute and how does that change over time
3. **Outage periods** –we know when they’re down

### 10) BLOB Validation

* BLOB (Binary Large Object) is a large object data type in the database system. BLOB could store large chunk of data, document types and even media files like audio or video files.
* In cloud, images for any website is stored in shared path and will be accessed later. As BLOB are perfect candidate for storing large values, it is ideally chosen for images in cloud as well.
* So whenever a website is loaded, all the images in the website will be fetched from a shared location.
* If we don’t validate these access/retrieval/storage for BLOB objects then, the website will have nothing but plain text.
* So validations are done for these BLOB data.
* One of the widely used tool to validate BLOB are**Azure Storage Emulator.**

As a tester we should be able to identify various components of cloud and test them according to the context.

Some of tools to perform such operations are mentioned below:

* SOASTA CloudTest.
* LoadStorm.
* CloudTestGo.
* AppPerfect.
* Jmeter.
* Cloudslueth.
* CloudTestGo.
* AppPerfect.

There are various services provided by cloud. We must make sure all those services should be provided with minimal interruption. These services can be provided only if we validate cloud app readiness. Some services are listed below

* Storage-as-a-Service
* Database-as-a-Service
* Information-as-a-Service
* Process-as-a-Service
* Application-as-a-Service
* Platform-as-a-Service
* Integration-as-a-Service
* Security-as-a-Service
* Management-as-a-service
* Infrastructure-as-a-Service

## What is Cloud readiness testing

“Cloud readiness represents the maturity of an organization, to identify appropriate Cloud-services, which generate value. Cloud-services have to be integrated efficiently and target oriented within enterprises in order to be able to manage their use and allows the agile change of cloud platform”

**Importance of Cloud readiness testing:**

• Gartner’s IT Budget report shows that healthcare companies often spend nearly **75% of their IT budgets on maintaining internal systems.**

• Cloud services allow you to pay for the resource usage you need while taking advantage of scale and reliability, two things that most companies can’t afford internally.

• But ‘Can I move onto the cloud infra?’ and ‘Should I move onto cloud infra?’ are different perspective entirely.

• All the advantages of moving onto cloud will go for a toss if don’t decide to check for proper readiness for cloud infrastructure.

**A cloud system will be having mainly three layers . Testing is supposed to be done on all the three level of layers.**

**Innermost**-First layer consist of the basic high end hardware servers in which the cloud environment is build upon. This layer mostly consist of the processors, storage, network devices etc. Here testing is done on the networking components.

**Middleware**-Second layer consists of the operating system and runtime environments which includes app creating software like Visual studio, JRE Engines, API services etc. which is used for creating or hosting the main application. The second layer is build upon the first layer. Here validation is done for the middleware.

**Outermost**-Third layer consist of the main application which is being hosted in the cloud. The third layer sustains upon the second layer and the first layer. Here validation is done on the app or on cloud services.

Below are the Key validations ensuring cloud readiness:

* Identity Access Management(Secure Access)
* API Integration Testing
* Billing Testing
* Third Party Integration
* Connectivity Testing
* Load Balancer Testing
* Failover / Resilience testing
* Shutdown & Resumption Testing
* Backup / Restore Testing
* Disaster Recovery Testing
* Logging
* Monitoring
* Archiving
* Identity management, also known as identity and access management (IAM) is the security and business discipline that "enables the right individuals to access the right resources at the right times and for the right reasons".
* Identity and access management is a critical part of any enterprise security plan, as it is inextricably linked to the security and productivity of organizations in today’s digitally enabled economy.
* The following tools are used widely for Identity Management:
  + SailPoint
  + RSA
  + IBM Security Identity Governance and Administration
  + Oracle Identity Governance Suite
  + Courion Access Assurance Suite
* API flows, or API integration, means running a sequence of API requests, where each request depends on the output of the previous requests.
* The simplest example is:

1. Perform request to authenticate and get a token barrier
2. Use the token to perform another API request

* Some readily available tools are:

1.SuperTest

2.Mocha

3.Chai

For this testing, let’s consider the following example.

A customer has a cloud service which finds monthly bill on 30th of every month. In February, the bill has to be paid by 28th itself. But here, the cloud service will look for the bill on 30th only. Without proper testing, the cloud services or CSP (Cloud Service Provider) will look for the billing which is not present or worse, finds it after due date.

One more important factor to be tested are the SLAs, which is a contract service provider promises for a defined level of service, such as response time, throughput or capacity. These SLAs might include downtime, patches, new build migration and system upgrades. Bad reasons – system crashes, security problems

Governance is essential for measuring principles and processes required for operating in the cloud. We need to test the governance of cloud here.

To avoid these overhead, billing testing has to be done.

In every cloud, the customer would want to use any third party application according to their liking. Reasons could be anything for the customer for using his/her own customized third party application onto the cloud environment.

In such cases, validating the third party application, its integration could be vital. This could also be a key business differentiator.

In such scenario, validation has to be performed on below three places:

1. API Integration: The APIs of both the service and application has to be validated.

2. Authentication:  Validation has to be performed for authenticating the application with the cloud.

3. Service Integration: The validation of service integration is also vital.

The purpose of Connectivity Testing is to ensure the following set of requirements are covered:

•Ensure the connectivity of different network interfaces involved are working as expected.

•Most often than not, the Account and the Data Center must be in different places. We must check if the data is properly retrieved and sent back.

•Connectivity between various CSP like Data Base, Storage etc. has to checked properly.

In Connectivity Testing, we have to perform all the validations that is going in or coming out of the cloud components.

•In a load-balanced environment, requests that clients send are distributed among several servers to avoid an overload.

•Depending on your load balancer configuration, the balancer may treat certain requests as either a single request in which case it is given to only one server or distribute them among various servers available.

•This way loads can be balanced between multiple servers.

•Some available Load Testing tools are:

* WebLOAD
* LoadUI NG Pro
* SmartMeter.io
* Apache JMeter
* LoadRunner

Failover/ Resilience testing:

•validates a system's ability to allocate extra resource and to move operations to back-up systems.

•verifies an IT system's ability to continue operations while the processing capability is being transferred to a back-up system.

•determines whether a system is able to allocate extra resource such as additional CPU or servers during critical failures or at the point the system reaches a predetermined performance threshold.

•To prove that the system can be shutdown and restarted properly without service disruption or within an agreed window of scheduled downtime.

•The testing should prove that:

* Each component can be shutdown and resumed successfully within the agreed time scale.
* The order of resumption of the components, if applicable, is valid and documented.

•We take backups for various reasons: hardware can fail, software has bugs, users make mistakes and delete or change data unintentionally.

•There is also the risk of deliberate and malicious attempts to destroy or encrypt data for financial gain or to “get back” at a previous employer.

•At this point, we should think about exactly what we want to recover as part of a test. There are multiple levels to consider:

* File recovery
* VM recovery
* Physical recovery
* Data Recovery
* Application Recovery

Disaster Recovery (DR) will ensure that in the event of a hardware failure such as a disk or server break down; there are processes and procedures to change to an alternative or redundant hardware such as a backup disk or standby server.

Testing involves the simulation of switching off/on hardware components during operation.

•DR is the process, policies and procedures related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster.

•DR testing will be exercised across all platforms, network, database and all eligible third party applications. Key Metrics in DR are Recovery Time Objective (RTO), Revised Operation Level (ROL) and Recovery Point Objective (RPO).

 Software should have comment in code and documentation for a product. Log has become very essential.

• Logs come in handy when there is an unforeseen shutdown/failure.

• We can have good restore/backup systems. But to find the root cause of the failure, we need to have detailed logs.

• These logs will give us details like time of error, error description, error fatality and so on.

• To test these logs, we need to manually disable certain components and check for the logs.

• Some log management tools are listed below:

1.LogRhythm

2.LogPacker

3.Splunk

4.Loggly

•Cloud monitoring is the process of evaluating, monitoring, and managing cloud-based services, applications, and infrastructure.

•You’ll want to monitor the following KPIs:

–Network KPIs will inform you about the bytes sent and received on the instance network interface.

–CPU KPIs will monitor estimated CPU utilization for the current billing period at the time of last collection.

–Billing KPIs will tell you the estimated credit balance, credit usages, and charges accrued for the current billing period at the time of last collection.

–System Integrity KPIs will let you know if the system instance, customer instance or system has failed.

–Storage KPIs will give you data on the length and number for Write Operations per collection cycle on an EBS Volume.

•The aim of archiving in cloud is to provide storage of old data that is optimized for long-term data retention, security and compliance with data regulation policies.

•Once in the cloud archive, data must be easily searchable via metadata; protected from overwrites or tampering; and provide client -- or legally specified, automatically applied data retention policies.

•In this case, we need to validate that the data is being archived at a agreed time. If so, after archival if you are able to access it.

•Then we need to check regularly for data redundancy.

•Also we have to make sure that the data is retrievable only through meta-data.

Compatibility testing:

•Validates the compatibility requirements of the application under test system

•Check browser compatibility on cloud environment

•Identify the defect that might arise while connecting to cloud

•Check that any incomplete data on cloud should not be transferred

•Verify that application works across different platform of cloud

•Test application in in-house environment and then deploy it on cloud environment

The best practice is to ensure that the application remain completely stable under long sustained workloads that simulates production usage patterns.

•This important test is performed to find out how stable a cloud system is.

• Some programming bugs directly affect the memory and cause an unexpected memory leak. This results in extensive usage of memory and a potential system crash.

•Stability tests will provide useful insights on potential threats that can impact the stability of your system.

## NFR testing in cloud - Introduction to Non functional Requirements (NFR)

According to IEEE definition NFR means:

“In software system engineering, a software requirement that describes not what the software will do, but how the software will do it”

* NFR requirements elaborate the performance and operation constraints of the system. It focuses more  on the operation of the system than its behavior to particular scenarios.

For example application performance requirements, application security requirements, external interface requirements etc.

* NFR requirements are informally contradictory, difficult to enforce during development and evaluate for the customer prior to delivery.

**Examples:**

1. If a web mail is sent to the user, he/she should receive it within 5 seconds.
2. The application should be capable of resisting any SQL injection hacking method.

In the above requirements we are not mentioning anything about the functionality of the system. In the first one, we are mentioning about the performance of the system. In the second one we are mentioning about the security feature of the system.

### NFR Testing in cloud – Challenges

he main challenges we face in NFR testing of cloud applications are

1. **Requirements are subjective** –You have to extract measurable data from the requirements before starting with the testing process.

2. **Setting Infrastructure** –For doing NFR testing like performance testing, you need the correct setup for simulating thousands of virtual users and high end load generators should be there.

3. **Test Data**–Humongous amount of test data need to be generated before starting with many of the NFR testing in cloud environment.

4. **Cost of Tool** –You have to pay bit higher for non functional testing tools when compared with functional testing tools.

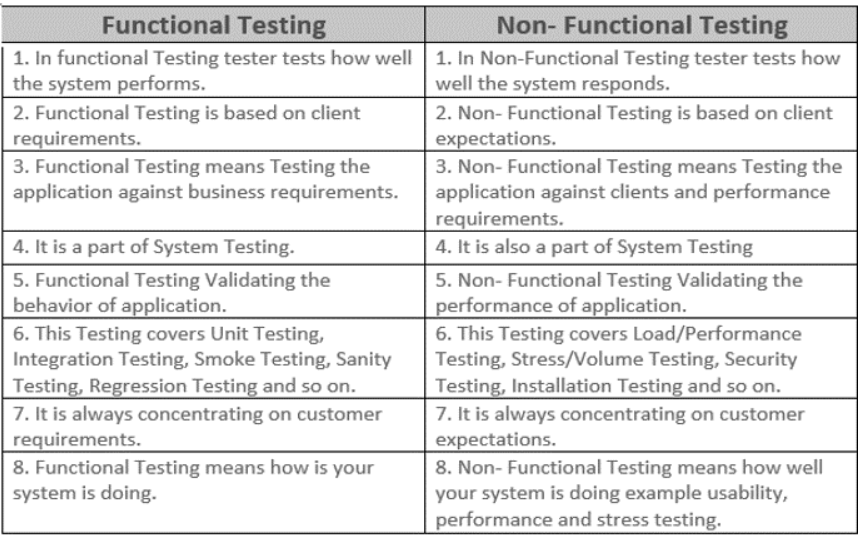
### Types of NFR Testing in Cloud

Apart from functionality testing, non-functionality testing also have to be done in cloud applications as well as in cloud environment.

•In non-functional testing of cloud the quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action such as scalability or security.

•Non functional testing ensures that a system/application meets the specified performance requirements. In non functional software testing, by performance we do not only mean response time, but several other factors such as security, scalability and usability of the application as well.

•These types of testing will be done by specially skilled testers with the help of dedicated tools since non functional requirements are too heavy to test manually.



Various types of NFR testing methods are available in cloud computing environments.

Below given are partial list of NFR testing techniques

•Performance testing

1. Load testing
2. Stress testing
3. Endurance testing
4. Availability testing
5. Error Handling

•Security testing

•Interoperability testing

•Usability testing

•Scalability testing

•Reliability testing and many more.

Lets discuss in details about different types of testing in the coming modules.

Upgrading your cloud environment, application, database etc. is a necessary component of your IT strategy. Testing throughout the upgrade process will mitigate the risk of lost, incomplete, or corrupted data, and help to ensure the success of your project.

The most important aspect of testing the upgrade process is

* to make sure that the data within the older version transfers properly to the upgraded version,
* to check that data and functional integrity is maintained after the upgrade.

This can be done with the help of tools which will check the difference in data and functionality by comparing the old version and the new version components.

Middleware is the layer that resides between the hardware layer and the application layer to provide services such as database management.

This manages the communication between a client program and a database. For example, a Web server connected to a database can be considered middleware - the Web server sits between the client program (a Web browser) and a database. The middleware allows the database to be changed without necessarily affecting the client and vice versa.

Test strategy done for middle ware testing in cloud:

* Performed early
* Performed by developers first
* Leveraged by structural test tools
* Realized by using test harnesses and stubs
* More object-oriented than procedural
* Involves testing large numbers of interfaces

Interoperability testing in cloud environment is done to validate the working of the application when it is connected with external applications, components or systems.

The main idea behind interoperability testing is to prove that functionality is working end to end between two communicating systems according to the standards demanded by the system.

The main strategy in Interoperability testing includes

* Connect two or more devices from different vendors
* Validating connectivity between devices
* Confirm whether the device can send/receives packets or frames from each other
* Validating if data is handled correctly in the network and facility layers

## Introduction to Performance testing

Cloud computing is changing the way application are deployed, monitored and used by the end-users. Cloud offers a virtually infinite pool of resources for computing, storage and networking resources where applications can be scaled as desired.

So how do you measure the performance of these applications in the cloud?

Performance is the utmost factor in testing a web application as it directly impacts the end-user experience. Performance testing in the cloud is different from that of traditional applications. Let’s find out what should be your strategy for performance testing in the cloud.

The main aim of the Performance testing is to measure the various parameters such as

* system throughput,
* latency with changing number of parallel users accessing your application,
* multitenancy(multiple instances of an application) and various other performance metrics.

In Cloud infra, this becomes a bit more typical with various virtual machines coming into play.

Moving performance and load-based application testing to the cloud brings cloud benefits to the Software testing by lowering Total Cost of Ownership (TCO). It also support for distributed development & testing teams.

Cloud-based testing is one of the cost-effective means of testing applications at large scale. It also allows you to simulate load tests constituting millions of concurrent users coming from multiple geographical locations.

Security, quality, reliability, latency, bandwidth etc. should be carefully looked upon if you are going with any external cloud service provider.

After all the setup is done, we finally need to execute tests. For performance tests, it becomes typical to vary the load with varying user’s demands and many other factors. So, this should be tested according to the defined test strategy.

Continuously monitor and analyze the test results as you would do in other setup. But, in this case it should be done in real-time to ensure you can improve it over a period of time.

Advantage of performance testing on cloud

•High on-premise cost

* Maintenance costs

•Real user experience

* Load distribution from different geographies
* High concurrency testing

•On demand provisioning

•No dependence on the infrastructure team.

### Various types of performance testing done on cloud environment

1. Load testing

2. Stress testing

3. Spike testing

4. Availability testing

5. Endurance/Reliability testing

6. Stability testing

7. Scalability testing

8. Error Handling

Load testing

Load testing is a kind of performance Testing that determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.

* Assume a scenario where an online shopping site is not able to handle 20 lakh users and getting crashed during a big offer day. This scenario arised due to the lack of proper load testing.

We have to get the load limit or maximum amount of concurrent users who uses the application for setting the limit during the load testing.

* For example if an airline application can have a maximum of 2000 concurrent users then our load to be tested is 2000. Even though the load is 2000 we will never test the cloud app directly for 2000. The first performance will be measured for limited users and slowly the count will be increased to 2000 to make sure the system is stable.

Loading testing on the cloud identifies the following problems before moving the application to market or production:

•Response time for each transaction

•Performance of System components under various loads

•Performance of Database components under different loads

•Network delay between the client and the server

•Server configuration issues like Web server, application server, database server etc.

1.Upload the script to the controller. An uploaded script looks like the one given below

2.Mention the number of virtual users for the load test. If for a scenario, 10 Virtual users should execute the script for 5 minutes. Then, scheduling has to be done accordingly.

3. Execute the test and analyze the various result components

Stress testing

Stress testing is the method used to identify the breakpoint of an application. Here, the user load is increased beyond the limit keeping the same number of resources as load testing.

If load testing is a pass for 2000 users in the previous example, the number of concurrent users will be gradually increased. Say 2250, 2500, 3000 etc. for stress testing.

This will help in identifying the maximum limit of the application even though the required load is 2000. By identifying the threshold limit, necessary steps can be taken to make the application behave in the right manner once the threshold is reached.

The steps for stress testing remains the same except for the number of virtual users scheduled in the controller area.

If the limit is 1000, once load testing is completed, the virtual user count will be increased to 1250,1500 etc. and the same test will be executed again and the result will be analyzed.

Spike testing

Spike testing is done to measure the performance of cloud applications when there is a sudden increase in the number of users is there at a particular point.

For example, let's assume the normal average concurrent users of the IRCTC application is 50000. But during the time of tatkal booking, a sudden increase in users will happen and they complete the transaction within a short period of time.

We have to emulate these kinds of scenario in a cloud environment which is called spike testing.

Spike testing is implemented by scheduling a steep increase in virtual users. The test will be executed for a shorter duration and suddenly there will be a steep reduction in the virtual users

The below configuration shows how to spike testing is scheduled for a test.

High Availability testing

Availability testing is done to ensure the availability of the system and its features at the required time.

Availability testing helps to determine whether the application is up and running fine without any issues(24x7x365). Basically, its done to ensure service reliability and increased fault tolerance.

High availability testing can be used to:

•prevent outright failures of the online cloud service.

•To ensure continuous operation(increasing fault tolerance and reliability).

•High availability testing ensures designing for tolerance, recoverability and reduced occurrence of high severity faults.

•It measures business impact from faults and helps in planning for reducing impact.

Endurance testing

Endurance testing is done to test the reliability of the application. It helps in finding out how long the cloud application or the cloud environment runs without any failure.

This is measured using the parameter Mean time between failure (MTBF).

If the application crashes 2 times in 90 days then the MTBF of the app is 45 days. This also is being executed with the help of performance testing tools where the scenario will be executed for weeks.

Endurance testing is done by executing a  test for a longer duration. The below scheduler shows how endurance testing is done for a test by executing it for 5 days. Once the execution is completed, results will be analysed.

### Steps involved in Performance testing on Cloud

Cloud services are deployed in various forms – Public, Private, and Hybrid. Below mentioned are some of the important tests that development teams must consider in a Cloud set-up and as a part of the Performance Testing strategy.

**1.Get Statistics on the Load**

When the testing process begins, it is critical to get the required information from the Cloud provider on the load statistics from other customers sharing the Cloud. This will provide key inputs on the kind of response time that you can expect considering other applications hosted on the Cloud. Hence, it is recommended to run the tests when the load is high and evaluate the response time.

**2. Prepare the test**

The transactions to be tested have to be converted into test using the appropriate performance testing tools. The test consists of scripts that instruct the tool to execute which transaction has to be executed.

The generated test have to be configured to make sure the scenario is staying close to that of the real-time environment. A baseline will be created with on-premise or previous benchmark and will be compared with the metrics obtained from actual testing.

Defining the iterations, parameterization, think time etc. comes under these enhancement part. A dry run has to be done on the test before going for the actual performance execution part.

**3. Executing the test**

Once the test is created, the number of concurrent users to execute the script have to be initialized. Along with this the time duration for the test, the load generators, monitors etc. have to be initialized before starting the run.

For cloud environment load generator machine will be from the cloud itself which generate a sufficient amount of load for the test to get executed. Depending upon the type of testing, the time taken for completing the execution may vary. For example, for endurance testing the test execution time will be more compared with load  or stress testing.

**4. Monitoring**

With memory monitoring tools, volume tests can be conducted. Here, the testing team can monitor the virtual memory.

This is essential for understanding the capacity and scalability of the virtual environment for testing the application. There are various tools that can be implemented to measure the same.

For example, Application dynamic monitoring tool can be integrated with LoadRunner to measure the network, bandwidth, utilization parameters of the cloud environment.

**Below screenshot is the monitoring page of a tool called Dynatrace**

### Performance tracking and measurement

Bottlenecks refer to the slowest part of a system which can affect the performance of the system badly. Common areas of bottleneck that can appear in any system are CPU utilization, memory utilization, disk usage, network utilization, operating system limitations.

In order to determine the bottleneck in a system involving multiple tiers/components, testing each tier in isolation is required. This helps to determine its peak performance capability, Apply fixes to ensure that each tier/component is performing at its optimal level. Remember that the end-user response time is limited to the throughput of the slowest tier/component. Hence, it is not enough to have only one tier/component that is highly tuned. All tiers/components have to be similar in their throughputs.

There is no simple solution to an identified bottleneck. It is an iterative process of fixing and re-testing until the desired objective is met.

**Some of the basic measures which should be monitored are:**

**Response Time:**

See that response time doesn’t increase abruptly with an increase in load. Load Testing tools give these values which shows the time taken for a client who has made a request until the complete page is downloaded (depending on the kind of application).

**% Processor utilization:**

 % Processor Time is the percentage of time that the processor is executing a non-idle thread. The CPU Usage should not vary between maximum extremes.

**Memory utilization:**

 Memory is  also an important parameter that should be monitored. Ensure memory doesn’t fall down abruptly and check for memory leaks.

**Throughput:**

 Number of bytes per second transmitted with increasing load is the throughput.

**% Disk Time:**

 % Disk Time is the percentage of elapsed time that the selected disk drive is busy servicing read or writes requests. As disk operations are very costly make sure that the disk time is idle.

**Process:**

Monitor the ongoing processes for its CPU & Memory.

**Other parameters that are monitored**

**Connection pooling:**Connection pooling is a technique used for sharing server resources among requesting clients through connection pools.

**Number of sessions:** Too many sessions should not be opened at any point in time. Also, emphasis should be given to the maximum number of sessions that can be active at any point in time.

**Hit Ratios:** When it comes to analyzing database for bottleneck identification, the most common parameters that would be best to go for are Hit Ratios (Buffer cache, Data Dictionary etc.) which would mean that the parsed SQL Statements would be directly fetched from the memory instead of doing a costly I/O operation.

**Rollback Segment:**The DBA should consider the size of the Rollback segment carefully as this determines how much rollback can happen at any point in time (depending on the size of data).

**Locks**: Most of the database systems maintain locks at the lowest level. As the demand for concurrency increases the lock escalates to the table, tablespace, database locks etc. The transactions that cause the locks should be carefully accounted for and tuned. The kind of locking implemented should be accounted for, particularly when large inserts or updates are involved.

**Memory (Committed Bytes)**: It is the amount of committed virtual memory, in bytes.  (Committed memory is physical memory for which space has been reserved on the disk file in case if it needs to be written back to disk).

**Memory Pages/Sec**: Pages/sec is the number of pages read from or written to disk to resolve hard page faults. (Hard page faults occur when a process requires code or data that is not in its working set or elsewhere in physical memory and must be retrieved from disk).

**Memory Page faults/sec**: Page Faults/sec is the overall rate of faulted pages are handled by the processor.  It is measured in numbers of pages faulted per second.  A page fault occurs when a process requires code or data that is not in its working set (its space in physical memory).

**CPU Interrupts/sec**: Interrupts/sec is the average number of hardware interrupts the processor is receiving and servicing in each second.

**Disk Queue Length**: Average Disk Queue Length is the average number of both read and writes requests that were queued for the selected disk during the sample interval.

**Network Output Queue Length**: Output Queue Length is the length of the output packet queue (in packets).  If this is longer than 2, delays are being experienced and the bottleneck should be found and eliminated if possible.  Since the requests are queued by NDIS in this implementation, this will always be 0.

In a nutshell, the main parameters monitored in the core areas are

In order to identify the bottlenecks in a system, data regarding various resources and their performance has to be collected. The sources of data are the following:

**Profilers** – Records the time spent in different parts of a program. Profilers like SQL profiler can be used for recording the time spent on SQL queries, stored procedures, etc. Database statistics give an insight into slow running queries, the health of the db.

**Traces** – Records the occurrence of various specified events. These traces can be on the client-side (server response, communication between server and client, done by the load controller) or at the server-side (e.g. event viewer on the servers). Calls to web and app servers can be accounted for by looking into the trace logs.

**Counters**– Records the health of the servers during test execution. Load generating tools or utilities like PerfMon for windows, vmstat, iostat for unix, netstat for network monitoring can be used.

## Introduction to Mobile Testing

A mobile application, most commonly referred to as an app. This is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Mobile applications frequently serve to provide users with similar services to those accessed on PCs.

Mobile applications comes in the below mentioned formats:-

•Native mobile app (Ex:Contacts, Messaging)

•Hybrid mobile app (Ex:Facebook, Whatsapp)

•Web mobile app (Ex:Anymobile optimized app opened through browser)

### Mobile testing in Cloud

**Mobile testing in cloud**is a process by which application software developed for handheld **mobile**devices deployed in cloud are tested for its functionality, usability and consistency and many other quality characteristics.

Additionally, wearable application testing is an interesting market. Bluetooth, GPS, Sensors, Wi-Fi are some of the core technologies at play in wearables. A lot of importance is needed here for field testing, user focus, and areas where hardware and software need to be tested in unison.

It can be done either through automation or putting manual efforts. Both of them have their areas of relevance as per their need.

It can be done either through automation or putting manual efforts. Both of them have their areas of relevance as per their need. Below is a brief overview of what they are actually

**Manual testing :**

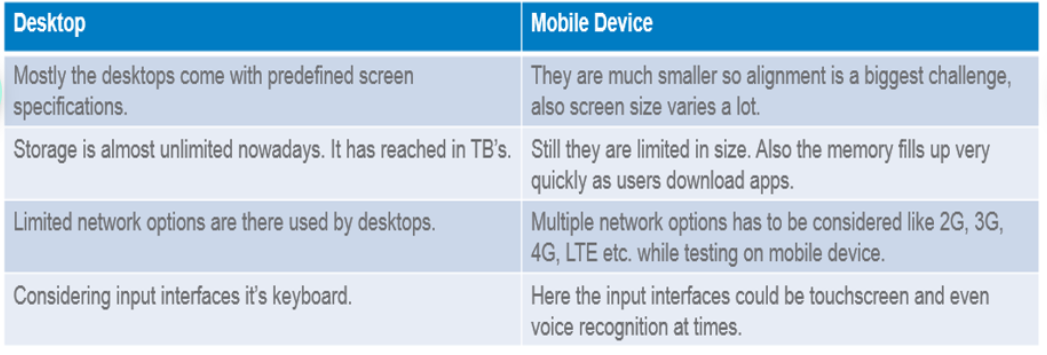
As the name says, whenever the testing is done using manual efforts like writing test cases, executing them etc. is called as manual testing. Even if we can perform most tests in an automated way, we still lack the technology to fully automate some types of tests, such as those for accessibility and usability. Manual testing also allows us to monitor user reactions, see where problems may lie, and what features users like more.

**Automation testing:**

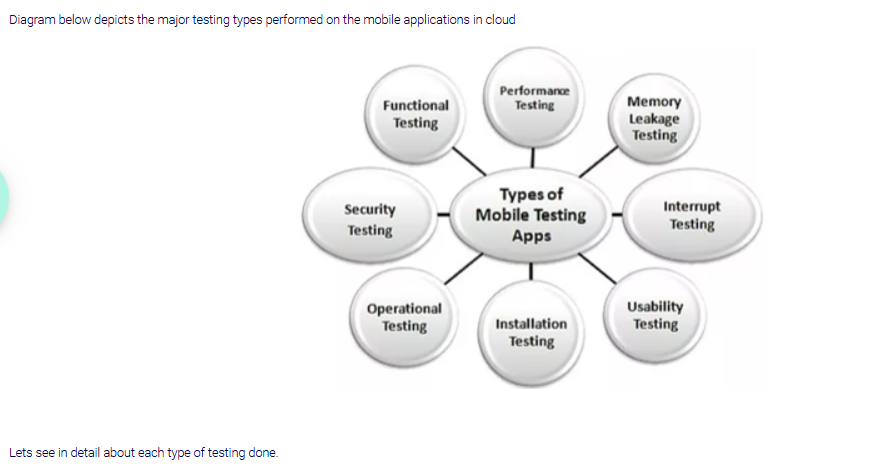
Mobile automation, as the name suggests, refers to 'automation' that is done on mobile devices. This can be done by using tools and helps in reducing the testing time cycle. Mobile automation can be done using many tools - some are licensed, while some are open source. For example:-Appium, robotium, selendroid etc.

**Difference of testing in normal system and mobile device:**

User interface looks almost same for a web application running on a desktop and a mobile device simultaneously. It has been made possible because of “Responsive Web Design” but still there are lot of differences we have to keep in mind while testing these apps on both these environments.



### Types of Mobile Testing done in cloud



**Functional testing**

Mobile Application Functional Testing involves testing of **Android Applications, iOS Applications & Web Applications** for Mobile Browsers. Mobile Application functional Testing is done to validate that each and every component of the application is working as expected.

For example:- If clicking on a button should generate a specific response by design, it should do that on the mobile device.

**Performance testing**

The testing process is carried out to test the performance and actions of the applications that pass through various mobile device. The challenges like low battery power due to heavy battery uses, network out of coverage area, poor bandwidth, changing internet connection mode (2G, 3G, or Wi-Fi), changing broadband connection, transferring heavy file, less memory, concurrent approach to the application’s server by various users, etc. are addressed.

**Usability testing**

It is basically done to assess user-friendliness, GUI consistency etc. Usability testing is majorly about verifying the ease of usage of an interface within an application. Learnability and memorability of the application are main factors in this case.

**Security testing**

It is done to test how well the system can preserve itself and the holds the data in situation of malicious attacks. Confidentiality, integrity, availability, authentication and authorization are the main areas that are tested when security testing is considered. Also, network security, system security and application security are other areas that will be tested in this case.

**Installation testing**

Mobile devices hold two types of applications: the one which automatically comes with mobile OS (while installing OS, it automatically get installed), and another one you have to install specially from the store to use the particular application. Installation testing is used to test the particular application is installing, uninstalling, and updating properly without any interruption (user is smoothly and flexibly installing the application).

**Memory leakage testing**

Memory leakage is one of the bad issues of the mobile application testing that directly affect on performance of the mobile devices. Due to memory leakage, process might slow down while transferring the file or in-between accessing any application mobile device might switch off automatically.

**Interrupt testing**

Interrupt testing is a process of testing a mobile application whose functions may get interrupted while using the application. Those interruptions can be incoming and outgoing SMS/MMS/calls, incoming notifications, battery/cable insertion and removal for better uses, network outage and recovery, switch off/switch on of the media player and other connecting devices, Low memory warning, and device power cycle(like low battery notification). An application should be capable to hold these interruptions by going into a suspended state and restarting afterwards.

**Operational testing**

Operational testing is used to test that the particular back-up and recovery process is working properly and responding as per the requirement.

**Cloud based solution**:-

It is an array of Mobile phones / Smartphones / Tablets connected to a remote server, but still accessed by user across different physical locations.

Various cloud based mobile testing platforms available are

1. Google Firebase Test Lab
2. Saucelabs
3. Amazon Device Farms
4. Xamarin Test Cloud

Since cloud based mobile farms provide services to a variety of consumers, they can effectively allocate and manage a wide variety of devices which is not usually cost effective if done by a single entity. Hence 1000+ types of devices can be made available at the tester’s disposal.

These platforms provide cloud-based infrastructure for testing Android apps. These tests can be initiated by a single operation and are run over multiple real devices of varying configuration hosted at their corresponding data centers.

These tests can be run on a public cloud or a dedicated private cloud.

The testers are not in direct contact with the device, a lot of user perspectives such as touchscreen responses, tests based network bandwidth etc. cannot be effectively validated.

 Amazon is a big player in cloud services and has one of the most robust and powerful cloud infrastructures for automated mobile testing on real devices. Amazon Device Farm is a service to test native apps, hybrid apps and mobile web apps on multiple devices at the same time. It provides test logs, crash logs, videos and screenshots as test results.

### Advantages of having cloud platform for mobile

Various emulators and real devices are available for testing. Test results include logs, videos and screenshots.

•Even if no test code is written for the app, Test Lab can exercise the app automatically, looking for crashes.

•Can be used with Continuous Integration systems.

•Run tests on a large number of real devices with a configured member, CPU usage and appropriate firmware and software configurations.

•Simulate real-world environments by configuring location, network connections, and language.

•Manually use remote integration to reproduce end user device issues.

### Process of Mobile Testing in Cloud

**1.Identify the type of testing to be done.**

Whether it is app testing or hardware testing, it has to be confirmed along with the finalization of the test scenarios. Details about OS, Browsers, Versions etc. needed for doing the inter operability testing have to be decided well in advance before starting the testing process.

It is a mandatory requirement that the application has to work in all iOS and Android devices as the end consumers can have diverse devices. To ensure that the app worked in all the devices, we can select a  combination of manual testing, automation testing and testing in cloud simulator.

**2.Perform manual and automation testing.**

* The app or device to be tested should be available in the dedicated cloud farm for doing the testing. These act as emulators as a good amount of testing is done on these rather than the real devices.
* Automation tools have to be decided accordingly and test cases have to be automated by creating the appropriate test scripts. User interface test cases will be mostly done through manual testing.
* Automation testing aims to reduce the intensive manual effort required for executing set of test cases on multiple devices and platforms.

**Cloud Testing Tools:** Remote Testing by acquiring devices on the cloud

  Example - Device Anywhere, See Test, Perfecto Mobile, Device Connect.

The main challenge in Mobile app development is to test and deliver the execution results for many frequent releases, which make the testing team to test the entire application faster and more often. This tedious challenge can be resolved by automating the manual testing process, with the help of the right AUTOMATION Framework

E.g. : An automation framework can be designed by integrating Cucumber, JVM, Appium and Selenium

    Appium is an Open source mobile automation framework available in the market.

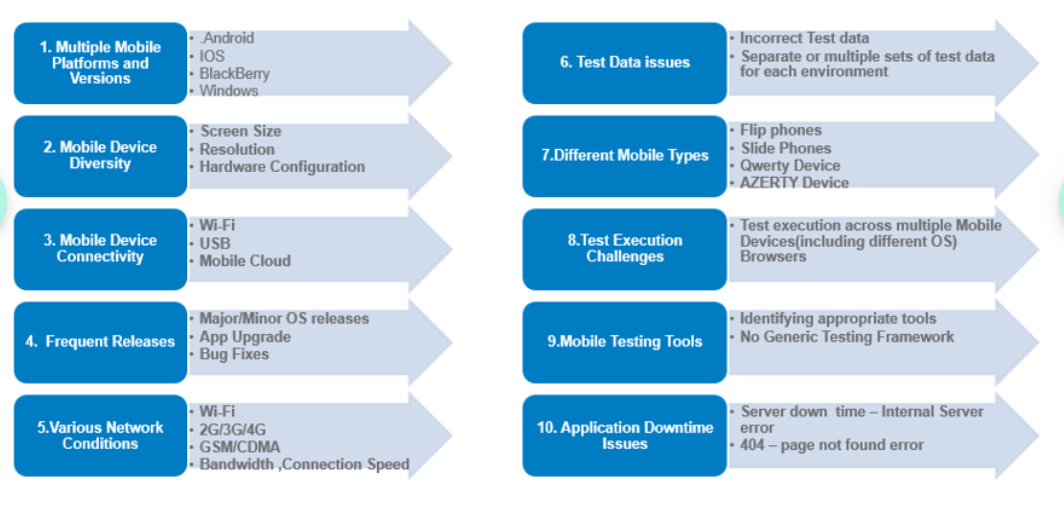
    Defect Management tools like Jira can be used for tracking the defects.

**3.Generate test summary report**

This is a management report which provides details of any important information uncovered by the tests conducted, includes assessments of the quality of the testing effort, the quality of the software system under test and statistics derived from incident reports.

The report also records different types of testing performed and how long did it take to complete the testing. This helps to improve any future test planning.

This final document indicates whether the software system under test is fit for use and has met acceptance criteria defined by project stakeholders.



## Various security threats in cloud environment

Let us see some of the major security threats in detail

• Injection attacks

• Broken Authentication attacks

• Cross-site scripting

• Sensitive data exposure

**SQL Injection** attack involves the injection of SQL commands into user input in order to affect the execution of predefined SQL commands in the application. On successful SQL injection, the attacker can

* Read and modify sensitive data from the database
* Execute administration operations like shutting down the database
* Recover file content from the DBMS file system
* Can even issue commands to the operating system

Example:

A login page has username as 106 and password as ‘hello’. Once a user logins to the application, the below query will be fired.

                        SELECT \* FROM Users WHERE UserId = 106 and password=‘hello’;

           A hacker can inject the below piece of code into the SQL query for any other user

                       SELECT \* FROM Users WHERE UserId = 107 OR 1=1;

           The above query will return the details of 107 without the need for the password.

**XML injection** attack tries to inject various XML tags in the SOAP message aiming at modifying the XML structure.

* On successful XML injection, the attacker can execute restricted operations.
* The injection of unintended XML content and/or structures into an XML message can alter the intended logic of the application.
* Further, XML injection can cause the insertion of malicious content into the resulting message/document.

For example, consider the below web page

* Online shopping payment page

                        Cart value:600

                        Card number:123456

                        Card expiry date:121224

The below xml data will be generated for the SOAP message for above GUI scenario

                       <transaction>

                              <cartvalue>600</cartvalue>

                             <cardnumber>123456</cardnumber>

                             <carddate>121224</carddate>

                       </transaction>

* Attackers can inject the below piece of XML data to the actual XML data which results in the altering of the cart value

                       <transaction>

                             <cartvalue>600</cartvalue>

                              <cardnumber>123456</cardnumber>

                              <cartvalue>6</cartvalue>

                              <cardnumber>123456</cardnumber>

                             <carddate>121224</carddate>

                       </transaction>

The cart value will be overwritten by 6 instead of 600 resulting in the loss of 594 rupees.

**Cloud testing done for injection attacks**

The following parameters can be tested for preventing the injection attacks

* Proper input validation
* Use of parameterized stored procedures instead of dynamic SQL queries
* Use of prepared statements
* Grant only necessary privileges to the accounts that are connected with the DB

Testers find an SQL injection vulnerability if the application uses user input to create SQL queries without proper input validation. Successful exploitation of this class of vulnerability allows an unauthorized user to access or manipulate data in the database.

Broken authentication is a technique used by an attacker to impersonate a session or user by predicting the user credentials or session value.

**Broken authentication can be successful to hackers mainly due to the below reasons**

1. Improper or no restriction on authentication attempts: The system should block the user after three consecutive attempts otherwise it would result in brute force attacks

2. Weak passwords: Passwords must be a minimum of 8 characters and should be a combination of numbers, special characters and alphabets. One should never follow a pattern when creating a new password with reference to the old password.

**Session related issues:**

Broken authentication issues can occur due to the vulnerability in session ids also. These issues can happen due to the below reasons.

•Session identifier passed in the URL parameter

•Weak session identifier

     Sequential Session id

     Short length random session id

     Predictable session ID’s

**Examples for session-related issue**

1. Mark was using the mail application and he forgot to logout in a cyber café. Another user Emily logged in to the system and saw Mark's mail inbox and she was able to use it. This issue will happen if proper session-id time out is not implemented

2. Mark was booking a flight. He logged in to the system. Emily needs the URL for the booking page and Mark copied and sent the URL along with session-id and authenticate code.

http://example.com/flightbook/book?sessionid=268544541&authcode=6787

If a duplicate session-id issue is not taken care, Emily can use the URL. She can even book a ticket from Mark's account as the session id is still active.

**Cloud testing done for Broken authentication attacks**

The following things have to be tested for preventing these types of attacks

•Session id’s should be sufficiently long enough to safeguard from brute force attacks: Use of strong algorithms to generate session id which has the following properties:

1. It must look random and unpredictable
2. Should not be reliably reproduced

•Appropriate session time outs

•Multiple sessions should not be allowed for critical transactions involved web applications like banking applications

**Cross site scripting** (**XSS**) technique is employed by hackers to execute embedded malicious script on sites that echo back users input without validation. Script executed could have the capabilities of reading, modifying or transmitting sensitive data.

In an application, assume after clicking on submit button, the value 2 is still reflecting in the quantity text box. This scenario can be a possible caused because of a XSS attack.

Instead of regular value, provide <script>alert(“Hello”);</script> and click submit button, you will get an alert box saying “Hello” which is a clear vulnerability.

Cloud testing for XSS vulnerability includes

•Verifying whether scripting in web browser and email clients are disabled

•Checking the denial of echoing user input without proper sanitation

•Verifying the encoding of all non alpha-numeric characters in input

The best way to test XSS vulnerabilities is to verify whether an application or web server will respond to requests containing simple scripts with an HTTP response that could be executed by a browser.

In **Sensitive data exposure** threat, hackers gain access to your sensitive data and any backups of that data. This includes the data at rest, in transit, and even in your customers’ browsers. Include both external and internal threats.

Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive data such as health records, credentials, personal data, credit cards, etc.

**Example:**

An application stores the credit card numbers in an encrypted format in a database. Upon retrieval, they are decrypted allowing the hacker to perform a SQL injection attack to retrieve all sensitive info in a clear text. This can be avoided by encrypting the credit card numbers using a public key and allowed back-end applications to decrypt them with the private key.

**Cloud testing done for sensitive data exposure includes the below verifications.**

•Don’t store sensitive data unnecessarily and should be scrapped as soon as possible if it is no more required.

•Ensuring that we incorporate strong and standard encryption algorithms and proper key management is in place.

•Disable autocomplete on forms that collect sensitive data such as password and disable caching for pages that contain sensitive data.

•Strong encryption techniques like Secure Socket Layer (SSL).

•Usage of robust mechanisms to protect and manage keys.

## AWS way of cloud security testing and Security testing challenges

Amazon Web Services (AWS) cloud system is having a monitoring tool named AWS Cloudwatch

•Monitor Elastic Compute Cloud (EC2) and other AWS resources

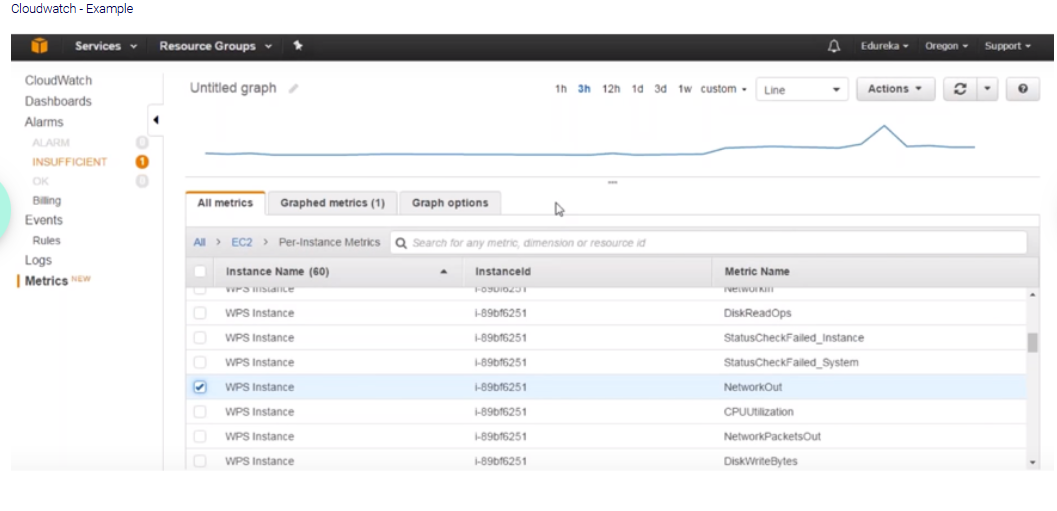
•Ability to monitor custom metrics

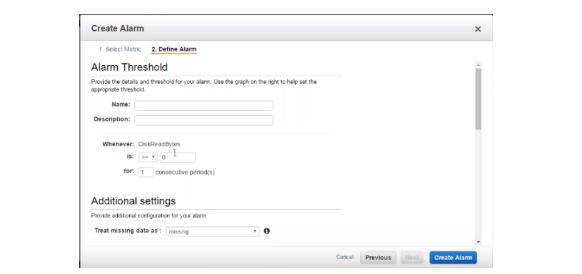
•Monitor and store logs

•Set Alarms

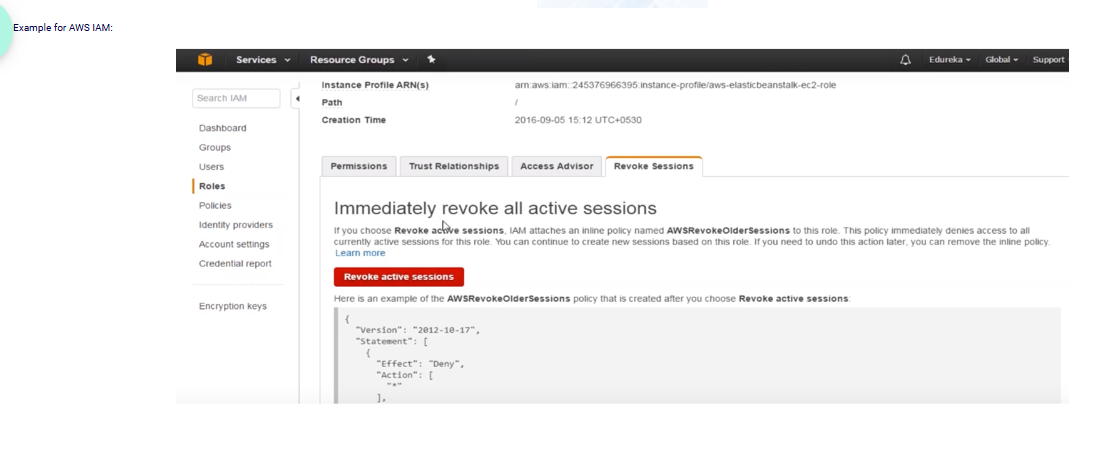
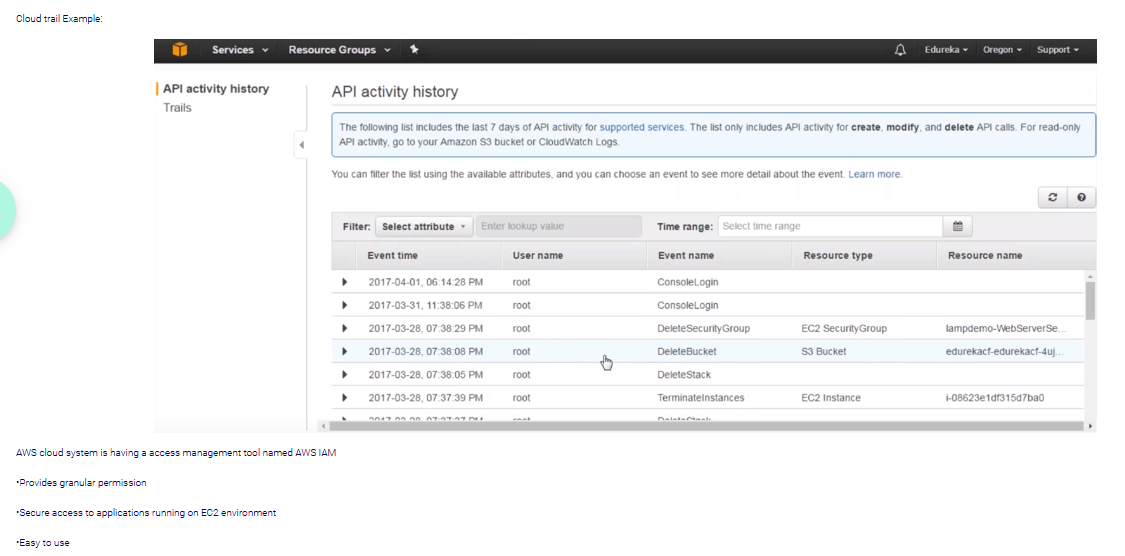
•View graphics and statistics

•Monitor and react to resource changes









**Automation tools used for security testing in the cloud**

•IBM Appscan

•HP Fortify Security Centre

•ZAP-Paros Proxy

•BurpSuite

•Wireshark-Network Packet Analyzer

Challenges in security testing:

•Increase in sophisticated threats and vulnerabilities

•Increasing complexity of computer infrastructure administration and management

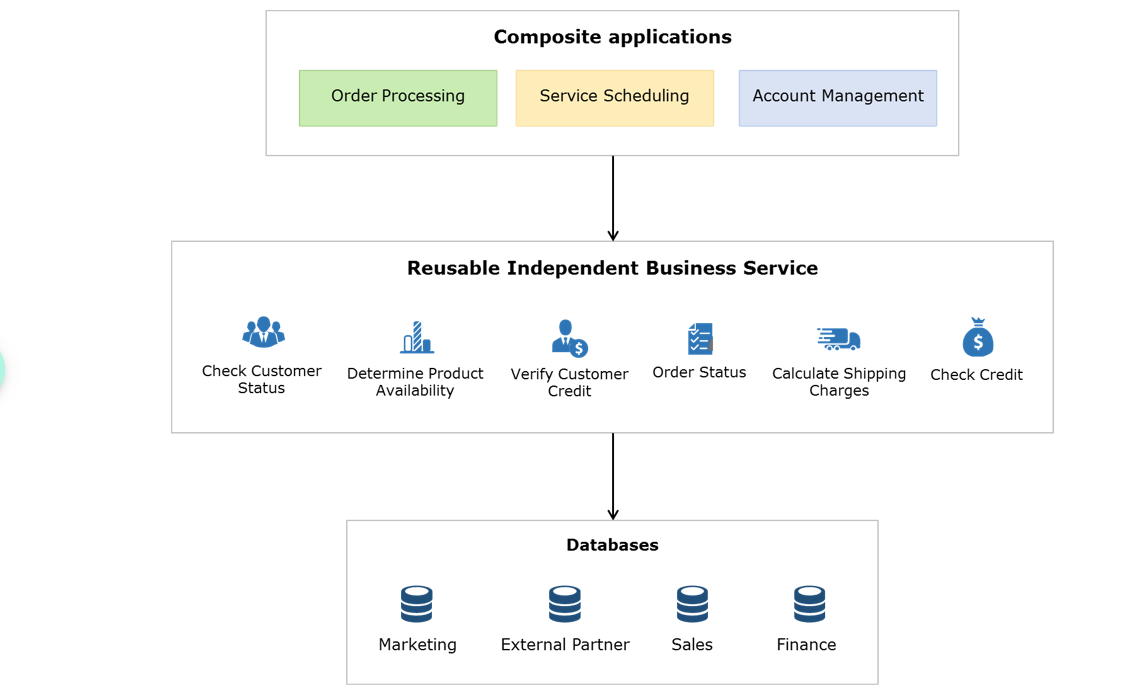
•Continuous up-gradation of the tools and technology used in the marke

## Introduction to SOA and Web Services

•In Service Oriented Architecture (SOA), the large application is made up of functionally, technologically and architecturally independent modules called 'services’.

•From a user's point of view, they would still interact with the application User Interface (UI) without noticing what's happening underneath. The UIs, in turn, pass on the user's intent to the layer of services. A combination of these independent services then gets executed to realize the user's intent. Now it is these services that interact with the databases, again independent of other services or application components.

•Below is an example of SOA architecture for an e-commerce website.



**What is a Service?**

•Consider that you want to add 'interest calculation' functionality in a Banking application that you are developing. Then, there is no need to code each line of it from scratch. Instead, you can reuse similar functional modules that are already available:

•With other banking applications hosted on different servers within your company's LAN.

•With any other 3rd party developer who is willing to share it with you over the internet.

•If your program sends a request to a specific 'interest calculator' functional module along with the required data such as principal amount, interest percentage and duration, it would calculate the interest and send it back as a response to your program.

•Such a functional module that services requests from any other program is called a Service. The application or system, which has placed the request, is called the service consumer. The application or system, which is hosting the service, is called the service provider.

•It is not necessary that all services that you use should reside in your local machine. Services can be hosted by a service provider, over a local area network or even the internet, for a service consumer. Such services are called web services.

•The following image shows the protocol stack for such request/response architecture.

Need for SOA testing

• With the market trends shifting towards SOA architecture, it becomes necessary for you to test them. As a modern-day tester, you have to get a fair idea of the architectural layers of SOA.

•This is because the strategies used and phases in SOA testing are tightly coupled with the layers in the architecture.

• In the next pages, let’s look into the phases which are there for SOA testing.

 There are five phases of SOA Testing. They are:

1. Service Level Testing

2. Integration Testing

3. Orchestration Testing

4. End-to-end Testing

5. Regression Testing

• Now let us learn these phases one by one. To understand better let's look at a ‘Loan Broker’ Application

• Working on the 'loan application' process in 'loan broker' application:

• In service-level testing, each service involved in the business process is tested individually. This phase is significant and mandatory in order to proceed with other testing processes.

•The services involved - loan broker service, credit agency service and bank services - are tested individually.

**Each service is tested for all the data conditions.**

–Test cases to check whether the credit score retrieved from credit agency service is greater than, lesser than or equal to a specific limit.

–In order to check lender service, we have to create test cases for checking which banks to request quotes based on the credit score.

–Check the loan quotation received from different banks.

**Each service is tested in all possible ways.**

–Here are some of the major testing activities done during this phase.

•Functional Testing

•Security Testing

•Performance Testing

**What is integration level testing?**

•In this phase, the following entities are combined and tested as a group to check how they work together.

•Services in the service layer

•components of application and data layer

•components of enterprise components layer

•The inter-layer as well as intra-layer data transfer between the modules are tested thoroughly.

•Here, we should understand that integration testing does not happen at the end of the cycle. It is conducted simultaneously with the development phase.

•One of the significant challenges in this phase is that most of the times all the modules are not actually available to test as they are still in their development phase. This is addressed by creating representations of the actual services using service virtualization techniques. It is similar to the use of stubs and drivers in manual testing.

**Functional testing:**

•Validates the communications between the services within every pair or cluster.

•Aims to assure that information sharing between the services are working fine.

**Security testing:**

•Validates whether the interfaces are vulnerable to malicious attacks and ensure that only authorized communications can happen between services.

**Performance testing:**

•Ensures that overall system-level performance after each component's integration is still within acceptable limits to the end-user.

**What is orchestration testing?**

•This phase of testing would cover business logic, sequencing, exception handling and process decomposition (including service and process reuse).

•To get a better understanding of orchestration level testing, let’s consider the following business flows of our loan broker:

**Business Flow 1**

•Customer makes a request to loan broker service.

•Customer’s information is sent to the credit agency which tries to get the credit score report.

•If the credit score is above a certain limit, then a loan request is prepared based on the two pieces of information (customer request and credit profile).

•The quote collection process collects all the loan offers from different banks.

•The quote collection process responds to the customer with the best loan offer.

**Business Flow 2**

•Customer makes a request to loan broker service.

•Customer’s information is sent to the credit agency which tries to get the credit score report.

•If credit score is not satisfying the limit then a reply, with the ineligibility status, is sent to the customer

•These business flows, as you have already seen, are defined using Business Process Execution Language (BPEL).

**Functional testing:**

•Ensure that all the business flows configured by their respective BPEL files are doing what they are supposed to do. They are not doing anything unexpected.

**Performance testing:**

•Load and endurance testing is conducted to measure business process throughput over specific time period.

**What is end-to-end testing?**

•This phase is meant to validate the business usability in day to day scenarios. It is similar to user acceptance tests which are performed towards the end of the development cycle.

•The test cases are usually written down as steps in a complete end-to-end business process and their validations are performed at the presentation layer.

•For example, a simplified end-to-end test flow of an email application might involve:

•Logging in to the application

•Accessing the inbox

•Opening and closing the mailbox

•Composing, forwarding or replying to the email

•Checking the sent items

•Logging out of the application

•The below items are validated during the end to end testing:

–UI of the application

–The business process involved

–The end-to-end data flow

–Handshakes and integration between services

**Functional testing:**

•Presentation layer validation or the verification of user interfaces.

•Only critical business flows are chosen for end-to-end testing as we are dealing with a system that has already undergone various stages of validation.

**Security testing:**

•Security vulnerabilities of individual apps in the presentation layer, that use the underlying architecture, is assessed.

**•Performance testing:**

•Load, stress and failover testing (in clustered environments) is performed to benchmark the performance and failover capability of the application.

**What is regression testing?**

•Regression testing is a full or partial re-execution of previously passed test cases (in the current or previous release cycles) to ensure that the existing functionalities  are working fine.

•In a distributed architecture like SOA, it also ensures that the changes to any interface/service will not affect the functionalities of other modules/services which is are interfacing with it.

•Regression Testing is required when there is a:

•Code change in any components of any SOA layer.

•New feature or functionality is added to the SOA system.

•Depending on the type of change that initiates a regression test requirement, performance tests might also have to be included. (E.g. Change in the database schema, installation of new routers, upgrade of Enterprise Service Bus (ESB) version etc.)

## Microservices and its testing

•The microservice architectural style involves developing single applications that can work together as a suite of small services, each running in its individual process and communicating with lightweight mechanisms such as an HTTP resource API.

•These services require bare minimum centralized management, use different data storage technologies, and can be written in different programming languages.

•These services, built around business capabilities, can also be deployed independently by machinery that supports fully automated deployment.

•The typical SOA model usually has dependent enterprise service buses (ESBs), with microservices using faster-messaging mechanisms.

•While SOA focuses on imperative programming, the microservices architecture uses a programming style that focuses on a responsive-actor as its base.

•While SOA models usually have an outsized Relational Database Managem (RDBMS), microservices frequently use databases such as NoSQL or micro-SQL that can be connected to conventional databases.

•Said that, the real difference lies in the architecture methods that are used for creating an integrated set of services.

•A microservices architecture consists of focused, small services that together create a complete application or task. Every instance of a microservice represents a single responsibility within your application.

•The real advantage is that these services are independent of one another, which makes them independently deployable and testable.

•Let’s look at some approaches below:

1. Unit Testing

2. Contract Testing

3. Integration Testing

4. End-To-End Testing

5. UI/Functional Testing

**1. Unit Testing:**

The scope of unit testing is internal to the service. In terms of volume of tests, they are the largest in number. Unit tests should ideally be automated, depending on the development language and the framework within the service.

**2. Contract Testing:**

•Contract testing should treat each service as a black box and all the services must be called independently and their responses must be verified.

•Any dependencies of the service must be stubs that allow the service to function but do not interact with any other services.

•This helps avoid any complicated behavior that may be caused by external calls and turn the focus on performing the tests on a single service.

**3. Integration Testing:**

•Verification of the services that have been individually tested must be performed.

•This critical part of microservice testing relies on the proper functioning of inter-service communications. Integration testing thus validates if the system is working together seamlessly.

**4. End-To-End Testing:**

•End-to-end testing verifies that the entire process flows work correctly, including all service and DB integration.

•Thorough testing of operations that affect multiple services ensures that the system works together as a whole and satisfies all requirements.

**5. UI/Functional Testing:**

•User interface testing is the testing of the highest order as it tests the system as an end-user would use it.

•Testing of this level must feel like a user trying to interact with the system. All the databases, interfaces, internal and third-party services must work together seamlessly to produce the expected results.

## Data Migration Testing - Data Comparison

**•Data Migration Testing:**

–Test the quality of data after migration involves a lot of techniques and methods.

–Below mentioned methods are the best way to ensure the coverage and authenticity of the data migrated. These methods are widely combined as three types:

1. Data comparison
2. Data quality analysis
3. BI report validation

**Data Comparison:**

–Here we execute a query to check if the result fetched by source and target are same.

e.g.

 Business rules states that any employee drawing salary less than 10000 should not be present in the target table. Then, the query must omit rows with salary less than 10000 from source table and select all the rows from target table.

* Source Query :  Select \* from source\_table where salary>10000
* Target Query  :  Select \* from target\_table

–Ideally, both the query should return the same number of rows.

–The query might be simple enough but as the business rule increases we cannot check for migrated in a single query but through series of queries which is discussed next.

•The below mentioned techniques are the ways to check for maximum coverage and authenticity

1. Count Testing
2. CheckSum Testing
3. Match Testing
4. Mismatch Testing
5. Business rule Testing

**1) Count Testing:**

–In Count testing we will check if the number of rows in both source and target are equal.

E.g.

If in source table, after applying all the business conditions the number of rows are equal to ‘n’ then the number of rows present in target table must also be ‘n’

•Source Query : Select count(\*) from source\_table where salary>10000

•Target Query :  Select count(\*) from target\_table

–The above mentioned query must return the same result.

–Here we have applied business rule only for salary column, we can also apply to various columns.

**2) Checksum Testing:**

–Although we can check number of rows, we can’t be sure if rows present in the rows are as same as in source and target. So we perform checksum on columns which is distinct and additive in nature.

E.g.

If in source table, after applying all the business conditions the value of the checksum must be same as the value of checksum in target.

•Source Query :  Select SUM(EMPID) from source\_table where salary>10000

•Target Query  :  Select SUM(EMPID) from target\_table

–In the above result, if the output is same, then we can be sure that the rows are same as well.

**3) Match Testing:**

–Even after checking for checksum, we can’t be sure if the rows are going to match in source and target.

–So we are going to check for the values for each column in target to each column in source.

E.g.

• Sample Query :  Select EMPID,Salary from source\_table where salary>10000

   intersect

   Select EMPID,Salary from target\_table

–This intersect query will give us the matching rows from both source and target.

–Ideally, the number of rows returned here must be same as the number rows from count testing.

**4) Mismatch Testing:**

–It is as same as match testing, but here we are going to check for mismatching rows

E.g.

• Sample Query  :  Select EMPID,Salary from source\_table where salary>10000

   minus

  Select EMPID,Salary from target\_table

–This minus query will give us the mismatching rows from both source and target.

–Ideally, the number of rows returned here must be ‘0’.

**5)  Business rule Testing:**

–In this type, wildcard operations are performed.

–That is, if the business rule states that only employee with salary greater than 10000 must be present, then we will check for the rows in target which has salary less than 10000.

–This testing is done mostly in the target table.

• Target query:

–SELECT \* from target\_table where salary<10000

–In the above scenario, number of rows in the table should be returned. Since the rows with less than 10000 as salary should not go into target.

**What is Data Migration?**

–Data migration is the process of transporting data between computers, storage devices or formats. It is a key consideration for any system implementation, upgrade or consolidation. During data migration, software programs or scripts are used to map system data for automated migration.

**Why Data Migration?**

–Data migration occurs for a variety of reasons, including:

•Server or storage equipment replacements or upgrades

•Website consolidation

•Server maintenance

•Data center relocation

**What is Data Migration in Cloud?**

–Data migration in cloud or cloud migration is the process of moving data, applications or other business elements from an organization's onsite computers to the cloud, or moving them from one cloud environment to another.

**Cloud Migration Tools:**

1.AWS Database Migration Service

2.Cloudscape

3.ScienceLogic

4.AppDynamics

5.DynaTrace

**Data Migration Testing:**

–Migration Testing is a verification process of migration of the legacy system to the new system with minimal disruption/downtime, with data integrity and no loss of data, while ensuring that all the specified functional and non-functional aspects of the application are met post-migration.

**Why Data Migration Testing?**

–Data migration is used to ensure the below mentioned points.

•Any kind of disruption/inconvenience caused to the user due to migration needs to be avoided/minimized. Eg: downtime, loss of data

•Need to ensure if the user can continue to use all the features of the software by causing minimal or no damage during migration. Eg: change in the functionality, removal of a particular functionality

•It is also important to anticipate and rule out, all the possible glitches/hindrances that might occur during the actual migration of the live system.

When do we require Data Migration Testing?

* Testing has to be performed both before and after migration.
* The different phases of Migration test to be carried out at the Test Lab can be classified as below.

1. Pre-Migration Testing
2. Migration Testing
3. Post Migration Testing

* Through data migration testing we can ensure the following to our customers:

1. Lower business downtime
2. Optimized effort/cost
3. Business Predictability
4. Consistent Data Quality

Let's discuss each and every techinque in detail

**1. Pre-Migration Testing**

•If the destination system has a mandatory field, ensure that the appropriate source is not null. If the destination system field has a list of valid values, test to ensure that the appropriate source fields contain these valid values.

**2.Migration Testing**

•In this phase metrics/process like definition of the source systems, the source system’s data sets and queries, the mappings between the source system fields and the destination system, number of source records, number of source systems records created per unit time, identification of supplementary sources, data cleansing requirements, performance requirements, testing requirements.  has to be validated.

**3.Post-Migration Testing**

•In this phase metrics/process like testing the throughput of the migration process, comparing Migrated Records to Records Generated by the destination dystem, Summary Verification, Compare Migrated Records to Sources has to be performed.

Data Migration Testing - Data Quality Analysis

**Data Quality Analysis:**

–Not all tests in a Data WareHouse Testing (DWT) project are going to involve data comparison. There are a lot of tests that are done on the data warehouse after Extract, Transform and Load (ETL)/migration to ensure that the data quality on the target systems meet the high standards required by downstream processes (Business Intelligence reports, legal compliance reports, data mining, etc.)

–Five major ways to perform the above operations are:

1. Metadata analysis
2. Statistical analysis
3. Relationship analysis
4. Pattern analysis
5. Business rule analysis

**1) Meta Data Analysis:**

•Metadata is data about data.

•In database/data warehouse terms, it translates to the characteristics/properties/constraints of a table as described below.

1. Which column/ column combinations form the primary key
2. Which column/ column combinations form the foreign key
3. What is the data type of each column in the table
4. Can a specific column allow null values

•This information will be available in the metadata file of a table.

•Using the metadata file, we can check whether the data in that table conform to all constraints set by the metadata definition

**2) Statistical Analysis :**

•Tests based on counts, ranges, sums, averages, etc. of the data in the Data Warehouse.

•The following are various ways to check for Statistical Analysis:

1. How many records in the price column of pricelist table do not have a valid, non-null, non-zero price (column statistics checks)
2. Does all dates in the DateOfBirth column in the current employee table lie within the accepted range (Age must be greater than 21 and less than 65 as of today). If not, what percentage of records are outside the range? (Outlier checks)
3. Are there any duplicates in the EmailID column and social security number column of the customer table? If yes, how many? (Uniqueness checks)

**3) Relationship Analysis :**

•Tests based on the parent-child relationship between two tables.

•For example, consider the below scenario:

1. Does every employee record in the employee table have a unit ID that corresponds to a unit ID present in the unit table? Are there any employees whose unit ID is not present in the unit table?
2. There cannot be a unit in the organization without any employees. Is there any unit ID value in the unit table for which there are no employee records in the employee table?
3. Each unit in the company can have multiple employees. Is that relationship (Cardinality of 1:M) reflected in the data, where each unit ID in the unit table is used by multiple employee records in the employee table?

**4) Pattern Analysis :**

•In today’s world, apart from uniquely identifying resources, data is made intelligent enough to contain a lot of information by defining data patterns.

For example:

–Vehicle registration numbers have information about the state where road tax has been paid, city/town where it was purchased/registered and the year of registration.

–Bank account numbers contain information about the account holder’s branch apart from identifying him/her uniquely.

•Testing data’s conformity to such patterns is indispensable to ensure data quality and data sanity.

**5) Business rule Analysis**

•Business rules are rules/requirements that a particular data value must conform to, as per the organizations’ guidelines, for smooth day-to-day operations.

For example

–Customer data must always have an email ID associated with each customer.

–The first three characters of an employee ID must be alphabets, representing his/her department

–The transaction ID of all debit transactions should start with the letter ‘D’ and credit transactions should start with the letter ‘C’

## Data Quality Analysis - BI Report Validation

**BI Report Validation :**

•Report generations are one of the final processes in a data warehouse system and the core reason for which data warehouse systems are built.

•Testing of a data warehouse is incomplete without testing of reports generated from it.

•In Perfaware, the BI report validation tool can check the data accuracy of the reports and provide a detailed report of the anomalies, if any.

•BI report validation can be performed by comparing the report-under-test with either data in a database or another base lined BI report.

**Big Data testing:**

Now let us see how big data systems are tested by understanding:

–The types of tests done on big data systems:

* Functional Testing

1. Data Ingestion
2. Data Quality Analysis
3. Data Processing / Data Mining Testing
4. Reports and Visualization Testing

* Non functional Testing

1. Infrastructure testing
2. Big Data Appliance Setup Testing
3. Performance Testing
4. Data Security Testing
5. Failover Recovery Testing

–How to choose the types of tests for your big data testing project

There are four different big data testing assignments that you may come across in your current or future projects. They are:

1. Migration testing
2. End-to-end testing:
3. Report testing
4. Data archival testing

**Data Ingestion:**

–During the data ingestion process, if the source data is in the exact format as needed in the big data store for analysis, then it can be loaded as-is. If the source data cannot be used as-is in the big data store, then we need to perform more actions.

–In order to verify the above actions, we perform data ingestion testing by validating:

* Whether the right data is getting extracted from source systems and loaded into the correct Hadoop Distributed File System (HDFS) location
* Whether the transformation rules are correctly applied
* Whether data integrity is maintained
* There is no missed/additional data by comparing source data with data ingested

**2) Data Quality Analysis:**

•After the data has been loaded into the big data store, we need to check the quality of this data

•Some of the standard big data quality analysis checks performed are:

* Metadata analysis
* Pattern analysis
* Statistical analysis
* Relationship analysis
* Business rule validation

**3) Data processing / data mining testing:**

•You need to process the data stored in HDFS or NoSQL DBs in order to derive meaningful insights out of it. This processing can be as simple as aggregation and filtering of the input data or as complex as using algorithms to identify patterns, co-relations or clustering information.

•You can perform data processing tasks by extracting the data using MapReduce, Spark, HiveQL or any query languages specific to the NoSQL DBs.

•You can store the results of these queries or pass it on to other applications for analysis. As part of data processing/data mining tests, we validate:

–Correctness of the values.

Eg. If the calculations in sales summary are calculated and displayed correctly

–If the result is in the required format.

Eg. The report needs the dates to be diplayed as "yyyy-mmm-dd" format or the numeric values are to be rounded to 2 decimal places.

**4) Data visualization and report testing:**

–Information is best represented using pictorial or infographic reports. These reports are created by reporting tools like Tableu, Cognos etc., using the data present in big data or Enterprise Data WareHouse (EDW). Mulitple related reports are combined in a single page to create “Dashboards”.

–Reports are often used as a key basis for management's decisions. Users of these reports assume that the presented information is representative of the true source data. Incorrect report data might lead to incorrect decisions.

–These reports are vital for Big Data’s outcome. So testing these reports are also part of Big Data Testing. Some available tools are:

1. IBM Cognos
2. IDTW
3. Automated Business Intelligence Report Testing

## Introduction to automation testing and various tools used in cloud automation testing

Test automation has become a vital part of the software development process. It enables you to test the functionality, performance and many other types of requirements in a fast and efficient manner.

**Test automation**

* The process consists of converting manual tests into automated ones with the help of appropriate testing tools
* serves better if implemented on an existing manual process
* scripts are designed, coded and unit tested before the actual test execution
* is mainly used during the regression testing phase, where the same test cases are repeatedly executed
* involves repeatedly executing tasks/jobs with high accuracy in minimal time and almost no human intervention.
* Owing to its high efficiency, test automation helps in achieving maximum test coverage and thereby enables you to deliver products to clients on time, without compromising on the quality

**Listed below are some of the key activities which are mandatory for achieving test automation.**

•Automation feasibility analysis

•Identifying the right automation tool(s)

•Test data with expected results

•Setting up the test environment

**Various tools used in cloud automation testing**

•Loadrunner

•Jmeter

•Appdynamics

•Jenkins

•Jira

•Confluence

•Dynatrace

•SOAPUI

•Postman

•Dummytest

**Loadrunner** is an enterprise-class application performance testing platform owned by Microfocus.

•It provides a complete, on-demand optimized solution for the application performance process, integrating load testing with diagnostics and capacity planning applications.

• It is a heterogeneous computing environment.

**Performance Center** includes an integrated application and business dashboard for key application performance activities.

HP Performance Center

* helps you analyze and validate the performance of your applications against business requirements and mitigate the risk associated with application deployment and upgrades.
* brings together the capabilities you need to optimize your application performance

**The various components of Loadrunner architecture are:**

**1.Virtual User Generator**

•Loadrunner is based on the concept of creating virtual users who take the place of real users working on client software.

•VuGen is used to craft scripts that are in the C language.

•The scripts are generated by recording the activities between the client and the server.

•During a performance test, these scripts are executed by virtual users to emulate the actions of a human user.

• VuGen also enables the replay/run of the scripts for debugging.

•Virtual User simulates human user actions by executing the test scenario. During the Performance test, human users are replaced with virtual users (Vuser).

**2.Load Generator**

•Load Generators are systems that can be set up to generate the actual load on the AUT.

•There could be multiple load generators that can be used for a project.

•Every load generator has the capability of running multiple Vusers.

•On each load generator, multiple virtual users execute the script(s) which will be designed in the VuGen.

Consider a case wherein in order to run a test we need 300 virtual users. In this case, let’s assume we have set-up 3 systems as load generators having load generator installed in them. Each load generators can run 100 virtual users.

**3.Controller**

HP PC Controller has the ability to control multiple load generators and collate the results.

The controller is used to configure each run with scenarios that describe

•the load generators in use and the number of virtual users on each load generator

•the scripts to be run by each virtual user

•Duration of test execution

•Service Level Agreements (SLA)

ALM Server can be used to configure the Controller and Load Generator. It is used to manage Performance Tests and to upload scripts generated by VuGen.The scripts are transferred via ALM Server onto the selected Controller. The performance test results are stored in ALM Server, which can be downloaded for analysis

**4.Analyzer**

The performance test results downloaded from ALM Server is used by the analyzer to generate graphs and reports. These help in identifying:

•the bottlenecks that affect the performance of the AUT

•sources of failures

•whether the test expectations were met

•the average transaction response time

•if the SLA (Service Level Agreement) was met

Sample Response time graph obtained from analysis after performance testing

Sample Throughput graph obtained from analysis after performance testing

•JMeter is an Apache Jakarta project that can be used as a load testing tool for analyzing and measuring the performance of a variety of services, with a focus on web applications.

•Jmeter can be used as a unit test tool for JDBC database connection, FTP, LDAP, web services, JMS, HTTP and generic TCP connections.

•JMeter can also be configured as a monitor, although this is typically considered an ad-hoc solution in lieu of advanced monitoring solutions

The various steps involved in performance testing using Jmeter are

1. Preparing tests

      •Proxy server

* It is used to record HTTP requests run by users.
* Stick to the exact HTTP request a normal user is creating.
* Record only what is meaningful.

     •Organization

* Thread Groups- It decides how many users will concurrently run the tests and long between 2 launches of the test. Thread group can be also used to configure how many times the tests will be run.
* Loop controllers-It decides the duration between 2 launch of the same sample and how many times the set of tests will be run.
* Throughput controller-Make variable pause during the test run to simulate better client behaviour. Because the thread group doesn’t make any pause in between the execution of the scenario

      •Genericity

        1. Assertions-It is used to match the presence of an element like a response, XPath, size of the response etc.

        2. Regular expression-It is used to deal with dynamic data like session id and how many times the set of tests will be run.

2. Running tests

       Once the test is created it can be executed in GUI mode and distributed testing. The first one generates less stress and the later generate high stress

• Non-GUI mode

•Distributed testing

3. Analyzing Test

After the execution analysis graph will be generated on various parameters and can be used to identify the performance issues.

Sample response time graph obtained from Jmeter after executing a test

•AppDynamics continuously discovers and monitors all processing in your application environment using the advanced tag, trace, and learn technology across your distributed transactions.

•With this information, AppDynamics provides a simple intuitive view of live application traffic and you can see where bottlenecks exist.

•Health indicators are based on configurable thresholds and they update based on live traffic. When new services are added to the system, AppDynamics discovers them and adds them to the dashboards and flow maps.

**Main features in Appdynamic**

**1. Real-Time Business Transaction Monitoring**

An AppDynamics business transaction represents a distinct logical user activity such as logging in, searching for items, buying an item, etc.

Organizing application traffic into business transactions aligns the traffic with the primary functions of a web business. This approach focuses on how your users are experiencing the site and provides real-time performance monitoring.

**2. End User Monitoring**

End-user monitoring (EUM) provides information about your end users' experience starting from the users' web browsers and their native mobile applications. It gives you visibility across geographies and browser types, answering questions such as:

•Where are the heaviest loads?

•Where are the slowest end-user response times?

•Jenkins is the most popular open-source tool for a continuous integration (CI), continuous testing (CT) and continuous delivery (CD) solution developed by Hudson lab.

•Jenkins has become the open-source standard for managing the dev side of DevOps, from source code management to testing to delivering code to production.

•CI means merging codes of different developers into a build, multiple times per day. The build is tested continuously (CT) to avoid defects in a later stage. Continuous Deployment (CD) takes the next step to ensure that the successful build is always in a production-ready state.

•Jenkins serves as the workflow engine to manage this CI/CT/CD pipeline from source to delivery, along the way many different tools may be called upon to perform different functions.

**The basic workflow of Jenkins:**

**Main features**

Cross-platform: it can be used on Windows, Linux, Mac OS and Solaris environments.

Plug-ins: Jenkins support more than 1400 plugins for the automation of all kinds of development, testing and deployment tasks.

Change Support: Jenkins generates the list of all changes done in repositories like Subversion (SVN).

Project build: Jenkins documents the details of the jar, version of jar and mapping of build and jar numbers.

Email integration: Jenkins can be configured to email the content of the status of the build.

TestNG test: Jenkins can be configured to run the automation test build on TestNG after each build of SVN.

Distributed Builds: Jenkins supports the "master/slave" mode, where the workload of building projects are delegated to multiple "slave" nodes, allowing a single Jenkins installation to host a large number of projects or to provide different environments needed for builds/tests.

Projects can be interconnected by creating a pipeline connection and these projects can be executed in a flow.

**Demo Screenshots in creating a pipeline**

Once executed the result of each project can be seen in the console output.

**A sample output shot is given below**

JIRA is a tool developed by Australian Company Atlassian. It is used for bug tracking, issue tracking, and project management.

The basic use of this tool is to track issue and bugs related to your software and[Mobile](https://www.guru99.com/mobile-testing.html)apps. It is also used for project management.

The JIRA dashboard consists of many useful functions and features which make handling of issues easy.

**Features of Jira**

•Because you've got issues

JIRA lets you prioritise, assign, track, report and audit your 'issues,' whatever they may be from software bugs and help-desk tickets to project tasks and change requests.

•Reporting and statistics

Customisable reporting allows you to monitor the progress of your issues with detailed graphs and charts.

•Workflow your way

  Map your business process with a custom workflow.

•An extensible platform

  Integrate JIRA into your systems with our open API and 100+ free plugins.

Jira can be used extensively for Agile methodology which helps in defining the sprints, burn down chart, backlogs etc.

Confluence software, developed by Atlassian, is an effective team collaboration software that provides a common platform for teams to work together and share information efficiently.

This is also a great tool to centralize the knowledge repository. Confluence can be thought of almost as a wiki with advanced content creation tools.

All information is stored in one central location, which makes it easily accessible to all users, so no more scanning of email chains or meeting notes to look for the required information.

Any updates made to the information triggers a notification to other users, which makes it easy for users to remain up to date about the latest discussions.

**Real-time use of Confluence**

**Knowledgebase** – A knowledge base is basically an information repository. It usually contains documentation about how to do certain things and maybe even information about how to troubleshoot products. An example of this can be for the QA team to manage and share information about processes, how to test documents, troubleshooting tips etc.

**As your own Intranet** – Intranet refers to an internal network of any organization and is a hub for displaying and sharing information. An example of this can be a space created by the Human Resources department to share company policies, vacation policies, upcoming events, user guides for common tools like Time off request tool etc. Information can be easily shared and the access is restricted to Confluence users within your company. So it is a secure platform.

**For Software teams** – For software teams, this tool can be used to write and manage product requirements, build and share release notes, collaborate on and record team decisions, create technical documentation, create blogs to share teams progress etc.

Spaces are the main component of Confluence which can be created for saving the contents. Multiple Spaces can be created and can be organized accordingly.

Spaces in this tool are a way to organize the content. Spaces can be thought of as individual file containers where the content can be categorized and organized in a meaningful way. The user can create any number of spaces with their own specific purposes to facilitate collaboration within teams.

Dynatrace is a single platform to analyze application performance throughout your application’s full-stack, down to each individual transaction across all layers and technologies.

Full-stack monitoring provides full insights into customer experience, application performance management, and infrastructure monitoring including servers, containers and cloud end-to-end with no blind spots.

Artificial intelligence continuously auto-detects dependencies, learns application behaviour, detects anomalies, and proactively pinpoints the root causes of issues.

**What Dynatrace traces?**

**Sample monitoring page using Dynatrace**

Unlike other providers, Dynatrace is offering a full-stack solution that contains way more than application performance monitoring.

AI-powered, unified, automated performance management is the goal of Dynatrace which leads to the ability to see every user and every app everywhere.

Auto-deployment and configuration in the cloud offer a maximum amount of usability.

Especially since it does not matter which cloud technologies are used. Optimizing an entire IT environment has never been so easy.

SOAPUI is a free and open-source tool for testing Web Services (Service Oriented Architecture (SOA) applications). It has a commercial companion –SOAPUI Pro. It has extra features for mission-critical Web Services. It is built in Java using Java swings as GUI.

We can do functional, Load and compliance testing of web services using SOAPUI. SOAPUI is entirely built on the Java Platform.

Different types of testing can be done on web services using the tool:

* Functional Testing
* Performance Testing
* Interoperability Testing
* Regression Testing etc.

•SOAPUI is used for testing Web Services. It is not used just for deploying Web Services. But it is also used to monitor any external service which is to be tested. Due to this feature, we can test our Web Service independently from any external service.

•SOAPUI has a very simple graphical interface which makes it easy to use for both technical and non-technical people. It's simpler interface makes it simple to work with Web Services Description Language (WSDL) and SOAP-based Web Services. It automatically generates Web service requests and tests using the Web service client. Even though the User interface is simple, it’s very powerful. SOAPUI has various security features like Web Service authentication and WS-Security and using it we can work with XML.

•SOAP Monitor Support is also a good feature of SOAPUI. Using this SOAP Monitor Support we can monitor and analyze the traffic.

•SOAPUI also provides a command-line option using which we can run our tests using the command line which in turn can be automated in batch files.

WSDL files can be loaded into SOAPUI and the tool will list all the available services in the corresponding URL. Needed input can be given and responses can be verified through assertions.

Service virtualization also can be implemented using SoapUI. Service Virtualization/Service mocking is a concept wherein the working/actual service is cloned to create a mock service that simulates the actual service. WSDL file of the service is used to create the mock service.

Groovy script and Javascript are supporting scripting languages used to enhance SoapUI tests.

In SOAP UI assertion functionality is used to validate the response of request received by the Test Steps at the time of execution.  It is used to compare a part of the message to some expected value.

Performance test of web services also can be done using SoapUI.

Sample load test configuration page for doing performance testing for a web service.

Postman is an application that is used to fire requests to an API.

It is very lightweight and fast. Requests can be organised in groups. Also, tests can be created with verifications for certain conditions on the response.

With its features, it is a very good and convenient API tool. It is possible to make different kinds of HTTP requests – GET, POST, PUT, PATCH and DELETE. It is possible to add headers in the requests.

Postman is very convenient when it comes to executing APIs, since once you’ve entered and saved them you can simply use them over and over again, without having to remember the exact endpoint, headers, API key etc.

After the response is received Postman has the functionality to make verifications on it. This is done in the “Tests” tab. Below given is a sample page in which the test is being done.

Once the tests are successfully executed, result page will look like the one given below.

•Mock service, Collections, Environments etc. are some of the main features of Postman.

•The Postman free app was created as a side project, and first introduced in October 2012.

•It grew quickly to become one of the most popular apps on the Chrome store, and expanding to Mac, Windows, and Linux native apps.

•Today they have 3+ million developers using our apps, worldwide.

•The Postman toolchain can help to build software that is useful across various use cases ranging from a single developer to an enterprise.

Dummynet is a live network emulation tool, originally designed for testing networking protocols. It is used for a variety of applications including bandwidth management.

It simulates/enforces queue and bandwidth limitations, delays, packet losses and multipath effects.

It also implements various scheduling algorithms.

Dummynet can be used on the machine running the user's application, or on external boxes acting as routers or bridges.

•intercepts packets in various points of the protocol stack;

•passes packets through a classifier and then to pipes, which model communication links;

•on exit, packets are reinjected in the protocol stack or in the classifier.

Below given is sample code for tuning the bandwidth speed

To summarize, the key advantages of automation testing are

* It is very helpful if a set of tests have to be executed repeatedly for a large number of users.
* Automation facilitates to execute regression tests in mainstream scenarios on time. For example, execution can be set to happen overnight without any manual intervention and you will get the results the next morning.
* It helps a lot in cross-browser testing as the same test script with very little modification can be run on multiple browsers.
* Unlike manual testing where the tests are run sequentially, automated tests can be run at the same time on different machines.

Some of the common disadvantages of automation testing are

* The initial cost of creating test scripts and configuring the automation framework is more than the cost of executing the tests manually.
* Automating visual references is not possible. For example, a font or a colour cannot be input via code or automation tools and thus it becomes more or less a manual testing process.
* Licensed automation tools are expensive even though they provide good support in test automation. Since companies are investing a lot of money on the tool licenses, they have to use the tool where ever possible to get the maximum utilization.
* Proper training on automation tools and knowledge of various scripting languages are a must for working with automation testing tools.

The main challenges faced by companies trying to adopt test automation are

* Non-availability of a test tool which is a one-point solution for testing of applications developed in different technologies.
* Huge upfront investment in automation tools and training and continuous maintenance costs.
* Tool vendors downplay the limitations of test automation.
* The inability of tool vendors to provide appropriate solutions/support for third party controls in the applications.
* Non-availability of personnel who are experienced in test automation concepts.
* Non-availability of personnel who are experienced in testing as well as programming skills.