EMERGING TECHNOLOGIES

Project Report

Group Members:

Kriti (C0776212)
Jatin Kumar Bhola (C0778001)
Emar Morrison (C0782758)
Gurpreet Singh (C0784329)
Gurpreet Kaur (C0780760)

Table of Contents

Describe how this application can be deployed in a cloud environment.

Cloud Computing can be defined as the novel style of computing where virtualized resources are provided as services on internet which are dynamically scalable. Cloud computing represents a different way to architect and remotely managing computing resources. It refers to both applications delivered as the service over the internet and system software in the datacenters that provide those services. Thee

data center hardware and software is called cloud.

Cloud Computing is a major paradigm shift. Most of the enterprises shifting their applications on to the cloud owing to its speed of implementation and deployment, improved customer experience, scalability, and cost control. Reliability, availability and security are the three greatest concerns for moving on to the cloud. Businesses are running all kinds of applications in the cloud, like customer relationship management (CRM), HR, accounting, and much more. Some of the world's largest companies moved their applications to the cloud with salesforce.com after rigorously testing the security and reliability of infrastructure.

A Cloud is essentially a class of systems that deliver IT resources to remote users as a service. The resources encompass hardware, programming environments and applications. The services provided through cloud systems can be classified into Infrastructure as a service (IaaS), Platform as a Service (PaaS) and Software as a service (SaaS).

A. Infrastructure as a Service

Infrastructure as a Service (IaaS) is categorized into:

1) Computation as a Service (CaaS), in which virtual machine-based servers are rented and charged per hour based on the virtual machine capacity – mainly CPU and RAM size, features of the virtual machine, OS and deployed software.

2) Data as a Service (DaaS), in which unlimited storage space is used to store the user's data regardless of its type, charged per GB for data size and data transfer.

Amazon has provided a popular universal and comprehensive solution to Cloud Computing, called the Amazon Elastic Compute Cloud (EC2). EC2 provides many useful features for customers, including a mature and inexpensive billing system able to charge for computing at a very fine-grained level (memory usage, CPU usage, data transfer, etc.), deployment between multiple locations, elastic IP addresses, connection to a customer's existing infrastructure through a Virtual Private Network, monitoring services by Amazon Cloud Watch, and elastic load balancing. EC2 has deployed such fine granularity and precision that it has become a benchmark and model in cloud computing.

Go Grid also provides Hybrid Hosting, which is a distinguishing feature. Many applications simply don't run well in a pure multi-tenant server environment. Databases perform better on a dedicated server where they don't have to compete for input/output resources, and the situation is similar with web server applications. Go Grid provides these special applications with dedicated servers that also have high security assurance.

B. Platform as a Service

Platform as a Service (PaaS) cloud systems provide an execution environment that application services can run on. The environment is not just a pre-installed operating system but is also

integrated with a programming-language-level platform, which users can be used to develop and build applications for the platform.

Microsoft’s cloud strategy is to construct a cloud platform that users can move their applications to in a seamless way, and ensure its managed resources are accessible to both cloud services and on-premises applications. To achieve this, Microsoft introduced the Windows Azure Platform (WAP), which is composed of a cloud operating system named Windows Azure, and a set of supporting services. Windows Azure is the main part of the WAP. It employs virtual machines as its runtime environments.

C. Software as a Service Software-as-a-Service (SaaS) is based on licensing software use on demand, which is already installed and running on a cloud platform. These on-demand applications may have been developed and deployed on the PaaS or IaaS layer of a cloud platform. SaaS replaces traditional software usage with a Subscribe/Rent model, reducing the user’s physical equipment deployment and management costs. The SaaS clouds may also allow users to compose existing services to meet their requirements.

Cloud Computing Deployment Models

Cloud deployment models indicate how the cloud services are made available to users. The four deployment models associated with cloud computing are as follows:

- Private Cloud
- Public Cloud
- Community Cloud
- Hybrid Cloud

Public Cloud

The cloud model we would use is the public cloud. This is the most common form of cloud computing, in which services are made available to the general public in a pay-as-you-go manner. Customers – individual users or enterprises – access these services over the internet from a third-party provider who may share computing resources with many customers. The public cloud model is widely accepted and adopted by many enterprises because, the leading public cloud vendors as Amazon, Microsoft and Google, have equipped their infrastructure with a vast amount of data centers, enabling users to freely scale and shrink their rented resources with low cost and little management burden.

Steps for deployment

The following steps would be used to deployment of the application:

- A load balancer, Web server, and database server appliances should be selected from a library of preconfigured virtual machine images.
- Configuring each component to make a custom image will be made. Load balancer is configured accordingly; The application would be migrated to a web server running python with all the relevant components installed. The database would them be migrated to an Amazon EC2 running Microsoft Windows server
- Repoint the application to communicate with the database server.
- We would then feed the custom code in to the new architecture making components meet their specific requirements.

Describe how you would implement security features for your app in the cloud.

App security isn’t a feature or a benefit – it is a bare necessity. One breach could cost our company not just millions of dollars but a lifetime of trust. That is why security should be a priority from the moment we start writing the first line of code.

With one break-in, criminals could know our name, age and even our current location precise to a few meters. Enterprise applications exchange exceedingly sensitive information that attackers are constantly on the prowl for.

With that kind of information at stake we will to do everything we can to protect users our users. The methods outline below can be used to protect out application in the cloud:

App Engine firewall

The App Engine firewall enables us to control access to our App Engine app through a set of rules that can either allow or deny requests from the specified ranges of IP addresses. We will not be billed for traffic or bandwidth that is blocked by the firewall. We will create a firewall to:

Allow only traffic from within a specific network

> Ensure that only a certain range of IP addresses from specific networks can access your app. For example, create rules to allow only the range of IP addresses from within our company's private network during your app's testing phase. We can then create and modify our firewall rules to control the scope of access throughout your release process, allowing only certain organizations, either within your company or externally, to access your app as it makes its way to public availability.

Allow only traffic from a specific service

Ensure that all the traffic to our App Engine app is first proxied through a specific service. For example, if we use a third-party Web Application Firewall (WAF) to proxy requests directed at your app, we can create firewall rules to deny all requests except those that are forwarded from your WAF.

Block abusive IP addresses

While cloud providers as many mechanisms in place to prevent attacks, we can use the App Engine firewall to block traffic to your app from IP addresses that present malicious intent or shield your app from denial-of-service attacks and similar forms of abuse. We can add IP addresses or subnetworks to a denylist, so that requests routed from those addresses and subnetworks are denied before they reach your App Engine app.

Access control determines who has permission to access services and resources in our Cloud project. We can use the following use cases for setting up access control:

- Granting team members access to your Cloud project so they can set up services and deploy apps.
- Granting our app access to Cloud services, such as Cloud Storage. All Cloud services require authentication and authorization for every API call, including calls from our App Engine app.
- Granting our users access to resources in a Cloud project. While this use case isn't common, there may be cases in which your app needs to request access to a Cloud resource on behalf of a user.

Encrypt All Data

Every single unit of data that is exchanged over our app must be encrypted. Encryption is the way of scrambling plain text until it is just a vague alphabet soup with no meaning to anyone except those who have the key. This means that even if data is stolen, there's nothing criminals can read and misuse.

Use Authorized APIs Only

APIs that aren't authorized and are loosely coded can unintentionally grant a hacker privileges that can be misused gravely. For example, caching authorization information locally helps programmers easily reuse that information when making API calls. Also, it makes coders' life easier by making it easier to use the APIs. However, it also gives attackers a loophole through which they can hijack privileges. Experts recommend that APIs be authorized centrally for maximum security.

Use High-Level Authentication

In the wake of the fact that the some of the biggest security breaches happen due to weak authentication, it is becoming increasingly important to use stronger authentication.

We can design our application to only accept strong alphanumeric passwords that must be renewed every three or six months. Multi-factor authentication is gaining prominence, which involves a combination of static password and dynamic OTP. In case of overly sensitive apps, biometric authentication like retina scan and fingerprints can be used too.

Deploy Tamper-Detection Technologies

There are techniques to set off alerts when someone tries to tamper with your code or inject malicious code. Active tamper-detection can be deployed to make sure that the code will not function at all if modified.

Use the Principle of Least Privilege

The principle of least privilege dictates that a code should run with only the permissions it absolutely needs and no more. Our app shouldn't request for any more privileges than the minimum required for it to function. If we don't need access to the user's contacts, we won't ask for it. We won't make unnecessary network connections.

Deploy Proper Session Handling

"Sessions" on mobile last much longer than on desktops. This makes session handling harder for the server. We will use tokens instead of device identifiers to identify a session. Tokens can be revoked at any time, making them more secure in case of lost and stolen devices.

Database Partitioning and Sharding To Limit Data Exposure

We will use data partitioning and sharding to divide our database into smaller more manageable parts. This combined with limiting privileges and account access makes sure that the cloud is less likely to become compromised. Even the strictest security efforts may not always be sufficient to prevent a data breach or hacking attack from happening, but limiting access and partitioning may help to make the devastating effect much less.

If you need to make your application serverless how it can be done?

Serverless Computing

Serverless computing allows you to build and run applications and services without thinking about servers. With serverless computing, your application still runs on servers, but all the server management is done by our cloud provider Amazon Web Services (AWS). Using AWS and its Serverless Platform, we can build and deploy our applications on cost-effective services that provide built-in application availability and flexible scaling capabilities.

Why build a serverless application?

Building a serverless application allows us to focus on your application code instead of managing and operating infrastructure. We do not have to think about provisioning or configuring servers since AWS handles all of this for us. This reduces infrastructure management burden and helps us get faster time-to-market.

Building a serverless application offers you four main benefits:

No server management

There is no need to provision or maintain any servers. There is no software or runtime to install, maintain, or administer.

Flexible scaling

Your application can be scaled automatically or by adjusting its capacity through toggling the units of consumption (e.g. throughput, memory) rather than units of individual servers.

High availability

Serverless applications have built-in availability and fault tolerance. You do not need to architect for these capabilities since the services running the application provide them by default.

No idle capacity

You do not have to pay for idle capacity. There is no need to pre- or over-provision capacity for things like compute and storage. For example, there is no charge when your code is not running.

How to make our application serverless

AWS introduced Lambda Services, a platform that enables developers to simply have their code executed in a particular runtime environment. To make the platform easy to use, many communities have come up with some really good frameworks around it in order to make the serverless apps a working solution. For our application we would use the Chalice framework.

What is Chalice?

Chalice, a Python Serverless Microframework developed by AWS, enables us to quickly spin up and deploy a working serverless app that scales up and down on its own as required using AWS Lambda.

Why Chalice?

For Python developers accustomed to the Flask web framework, Chalice should be a breeze in terms of building and shipping your first app. Highly inspired by Flask, Chalice keeps it pretty minimalist in terms of defining what the service should be like and finally making an executable package of the same.

High level Setup

1. Create a virtualenv
2. Install chalice
3. Create a new chalice project
4. Deploy our application using API Gateway and Lamda function

5. Migrate existing application code.