ISSN:0975-9646

Reliable Data Tier Architecture for Job Portal using AWS

Manoj Prakash Thalagatti[1], Chaitra B[2], Mohammed Asrar Naveed[3]

[1,3]M. Tech Student, Dept. of ISE, Acharya Institute of Technology, Bengaluru, India
[2]Assistant Professor, Dept. of ISE, Acharya Institute of Technology, Bengaluru, India

Abstract - **Cloud computing is considered as one of the most notable revolution in IT world. Virtualization of computing resources provides benefits which includes rapid elasticity, resource pooling, location independency, and many more. No enterprise can afford paying for something that is unnecessary or paying twice for something. Small and medium sized enterprises will look for a way for reducing their costs and maximize their profits. Cloud computing is a way to assist in this regard by reducing huge upfront costs and number of employees being employed. Amazon Web Services (AWS) is one among the cloud computing market leaders. AWS provides a reliable, scalable, secure, and highly performing infrastructure for the most demanding web applications, an infrastructure that matches IT costs with customer traffic patterns in real time.**

Keywords - Cloud Computing, AWS, MongoDB, MariaDB

I. INTRODUCTION

People started searching for jobs over an internet rather than print media like News Paper. This leads to the development of Job Portals. Job portal is a website that helps in the recruitment to both the recruiter and the job seeking candidate. Advantage of using job portal is that the overall procedure of appointments becomes faster.

Companies adopt cloud infrastructure which helps in reducing upfront costs. Cloud computing, in addition with resource virtualization provides various computing resources in pay-as-we-use method and also its additional benefits include rapid elasticity, network access, location independent resource pooling and many more [1]. There are many public cloud services providers; Amazon Web Services (AWS) is one among them that provides Infrastructure as a Service (IaaS). Cloud Computing technology has become one of the most adopted IT paradigms of recent years. IT industry has found greatest advantage from Cloud Computing over the past decade.. With cloud computing, organizations can utilize the resources such as shared storage and computing rather than building, operating, and improving infrastructure on their own. Users store data in a cloud, and retrieve them whenever they want to use them. The speed of change in markets creates significant effect on the enterprise IT infrastructure to adapt and deliver. Cloud computing provides efficient solutions to address these changes. As defined by Gartner, “Cloud computing is a style of computing where scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies” [2]. Essential characteristics of cloud are: on demand self-service, broad network access, resource pooling, rapid elasticity and measured service.

II. LITERATURE SURVEY

A. Amazon Web Services (AWS)

AWS is referred to as a collection of computing services accessed remotely, and also called web servers that make up a cloud computing platform by *Amazon.com*. AWS provides trusted, cloud based solutions to help you meet your business needs. AWS offers broad set of global compute, analytics, database, application, and deployment services that help organizations move faster, scale applications and lower IT costs. All these services are trusted by largest enterprises and the hottest start-ups to power a wide variety of workloads including mobile and web applications, storage, data processing and warehousing, archive, and many others [3].

Amazon resources can be hosted in multiple sites world-wide. These sites are called as *Regions [3].* AWS regions have completely isolated zones, so called *Availability Zone (AZ).* Always plan to build architecture with sufficient amount of isolation between the machines (Multi Availability Zone (AZ) architecture). In AWS, distribute your EC2 instances in the availability zones in the region. You can even go multi region architecture, but deployment across regions introduces complexity and also additional cost. Usually, small and medium sized organizations will go for multi Availability Zones and larger ones can afford multi region architecture costs [4] [6].

Amazon EC2 instances work in conjunction with Amazon VPC and provides secure and robust network environment for computing resources [8].

- All EC2 instances are located in a VPC. IP addresses to be assigned to instances will be in your control. You can decide on instances to be kept private and to expose to the Internet.
- Control the network access of your instances by creating Security Groups and specify the inbound and outbound rules to control the traffic to and from instances.
- You can also have a connection between your existing corporate data center and your VPC using IPSec VPN.

B. Application Layers

The most general tiers in architecture are discussed in this section [9].

1) Load Balancing Tier: The incoming traffic is distributed to mid tiers like Web Tier/App Tier. This is done by load balancing tier. The Elastic Load Balancer (ELB) is the solution for this tier, but user also can setup their own EC2 instances to do this task by using software like Nginx, HAProxy etc. Entry point to the application is the load balancer and it is very essential aspect to have high availability to the load balancers.

2) Web Tier: The serving of static files to the user and routing of traffic to the right App Tier is done by this tier. Even though web server is intended to serve files in static very fast, the best method is to use S3 or CloudFront to deliver static files.

3) App Tier: The role of this tier is to run your app's main process. For example, if you've built a Java app, this tier is running a Java program that's listening for incoming connections.

Your App Server is sometimes where "state" is stored in your app. For example, a user logging in may have his session information on a particular EC2 instance in the App Tier. The problem with this is that this user is now "pinned" to one EC2 instance, and if it fails, your user's session dies with it. The best practice here is to make your App Tier instances stateless and store things like session state in a different tier.

4) Cache Tier: ElastiCache, which acts as cache tier and let user to choose either Memcached or Redis. When Cache Tier clenches all short-lived state like session info, this cannot be maintained by App Tier longer. This helps user to launch further EC2 instances, and as soon as instances are configured it stores short-lived data on cache tier and persistent data on the database tier as before. This is the reason for cache tier to become a vital part of Auto Scaling.

5) Data Tier: This is the tier where all confidential, determined data is maintained. First and foremost, the decision of what kind of database system to be used is a very crucial decision for the entire system architecture. The best practice is to maintain all instances in database tier privately. Proper measures have to be taken in account to handle even the worst case scenarios to the application. The solutions to handle failure scenarios are [4] [6]:

- Use of real-time data replication for high availability
- Use of real-time replication with read replicas
- Use certain amount of isolation between the database servers
- Setup automated backups

6) Ancillary Tier: The additional instances that are ancillary (supplementary), to access app server (which are placed in private subnet). Bastion Host can be used to securely connect to instances. Ideally, our infrastructure is automated enough that we rarely need to directly login to a server. But when we must login, it's too risky to open login ports (3389 for Windows Remote Desktop, 22 for Linux SSH) directly to the public Internet.

A better option is to either lock out these ports completely from the Public Internet or connect with via a VPN, or to use a Bastion Host. Network access to EC2 instances running privately can be controlled by using Bastion Host and it is the only server that permits login from the public Internet [9] [10].

C. MongoDB - A NoSQL Database

A NoSQL database (interpreted as “Not only SQL”) provides a mechanism for retrieval and storage of data that is modeled other than the tabular relations used in relational database. This approach has got advantages compared to relational databases which include simplicity of data design, finer control of data availability, and horizontal scaling. NoSQL database addresses many issues that relational databases did not address. The greatest advantage of using NoSQL databases include big data and real time web applications, and can scale out on commodity hardware.

MongoDB is a document oriented database. It stores all documents in collections. A collection is a group of documents that have a common shared set of indexes. Collections are equivalent to a table in relational databases [5]. MongoDB provides some features which include: indexing, caching, replica set, sharding [11].

D. MariaDB

MariaDB is a community, which is the developed fork of MySQL RDBMS. It aims to maintain high compatibility with MySQL and also exact match to APIs and commands of MySQL. It provides two types of multi-source replication: Master Slave replication and Master Master replication [13].

E. Big Data

Data analytics has become a most trendy part of the application to improve the business of a company. Apache Hadoop is a framework to handle very large data sets. It is an open source software that supports distributed parallel processing of large volume of data across less expensive commodity hardware.

Hadoop has many features which overcomes the challenges of RDBMS [12]:

- It is designed to process both structured and unstructured data.
- It is a platform to manage Big Data.
- It runs on commodity hardware.

F. MongoDB Connector on Hadoop

It is a plugin for Hadoop to use MongoDB as the source of input and the destination output [7].

III. IMPLEMENTATION

An overview of the system architecture along with different sub system components is described in Fig. 1. This section presents an abstract view of the components involved in the system and the interaction between each of the components satisfying the system requirements. This work mainly focuses only on the “*Data Tier*” and hence no detail components present in the other tiers of the architecture.

The architectural overview for job portal application is presented in Fig. 1. It consists of various AWS resources; Amazon Elastic Cloud Compute (EC2) is used for MongoDB and MariaDB acts as database servers. Amazon Simple Storage Service (S3) used as highly-scalable object storage typically used to store end users images, videos and resumes. All the above components are kept in Amazon Virtual Private Cloud (VPC) which is combination of Subnets, Route table, Internet Gateways etc. Auto Scaling is used to replace failed node as soon as they go down. Bastion Host is used to enable the Internet access to the EC2 instances launched in the private subnet. Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications that run on AWS. ACW is used to collect metrics and set alarms. Each of the components is explained in detail in this section.

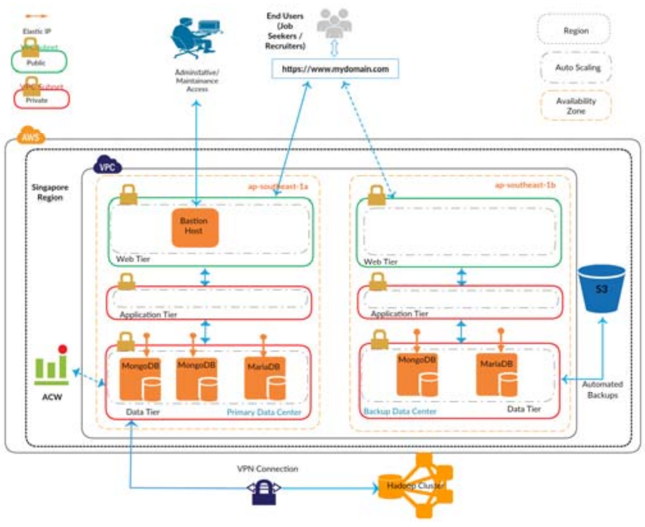


Fig. 1 Overview of Job Portal Architecture

A. Region and Availability Zones

The region selected is Asia Pacific (Singapore) identified by ap-southeast-1. Two zones are available in this region are ap-southeast-1a and ap-southeast-1b. Multi availability zone architecture is adopted to ensure fault tolerance.

B. Amazon Virtual Private Cloud (VPC)

VPC is used to create a private cloud environment where application specific architecture can be defined. A virtual network environment is created which includes IP addresses, security groups, route tables, subnets of our own choice. The VPC consists of 2 private subnets (one each in two availability zones). A subnet should be created in the VPC, associate a route table to it and then launch EC2 instances. Private IP addresses to EC2 instances will be allocated in the range of Classless Inter Domain Routing (CIDR) block of the subnet. Since these AZs are isolated from each other, distributing instances on two AZs ensure high availability and fault tolerance of databases servers.

C. Amazon EC2 Key Pairs

Amazon EC2 Instances use public-key cryptography that requires two separate keys to encrypt and decrypt login information. The private and public keys are referred to as key pair. A key pair is created using AWS EC2. The key used by Amazon EC2 are 2048-bit SSH-2 RSA keys. This key pair will be used when EC2 instances are launched and when remote connection has to be established to EC2 instances using PuTTY. This allows login to instances remotely and executing the commands. Amazon EC2 stores the public key and save the private key in a secure place.

D. Elastic IP Address (EIP)

Public DNS and public IP address allocated to EC2 instances are not static. A new DNS and public IP address will be assigned on every restart. In effect to this, the connection between the application and database will break. In order to keep the application live even on failure of any EC2 instances, static Elastic IP addresses are used for all EC2 instances.

Application will be configured with IP addresses of database servers and it is not right approach to change the IP addresses every time in case of failures. The best practice is to use static Elastic IP addresses for database servers. Now the Public DNS will also be static. Amazon internally resolves Public DNS of the instance to private IP address associated to it. Public DNS are used in the following cases:

- MongoDB Replica Set
- MariaDB Master-Master Replication

E. Amazon Machine Image (AMI)

An AMI is a master image for launching EC2 Instance. Amazon provides machine images that are configured with an operating system and any other software. Amazon *Community AMIs* is used to launch EC2 Instances. The AMIs created by user will be *My AMIs* for AWS account.

Launch EC2 instance and set up customized environment. It is time consuming every time to set up customized environment on EC2 instances. MongoDB and MariaDB are setup on separate EC2 instances and then AMIs are created of these two which are *My AMIs* for AWS account. Two AMIs are created – one is of MongoDB and the other is of MariaDB; launch two more instances for replica set from MongoDB AMI and one instance for replication from MariaDB AMI.

F. Amazon Elastic Compute Cloud (EC2)

Elastic Compute Cloud (EC2) is considered as the central part of the Amazon.com’s cloud computing platform. AWS allows cloud consumers to rent virtual computers to host their applications. Amazon provides AMIs of different underlying environments; Amazon calls the virtual machine as “*EC2 instance*”. Ubuntu is used as the underlying operating environment for database servers. Launch two EC2 instances of required specifications and install MongoDB and MariaDB on top of those instances.

A total of five EC2 instances are required in the database tier.

- 3 for MongoDB Replica Set
- 2 for MariaDB Master Master Replication

G. Route Table

A Route Table is a set of rules used to determine where the network traffic is directed. Route Table determines whether a subnet is exposed to the Internet or bound to internal traffic.

Two route tables are created; one for internal traffic and another accessible over Internet and associate to the corresponding subnets. (i) A route table is created for only internal traffic (do not attach Internet Gateway) and associate it to the database subnets making it private. Since data is the critical aspect of the application, all the database instances are kept in private subnets not exposing to the Internet. Using this kind of configuration improves security. (ii) Another route table is created, attach it to Internet Gateway and associate it to the subnet making it publicly accessible. Bastion Host is setup in the pubic subnet.

H. Subnet

A subnet (short for “subnetwork”) is an identifiable subdivision of a larger network. A subnet can determine the number of participating machines in a LAN, building or geographic location. Two private subnets are created to implement multi AZ architecture. One Private DB Subnet is created in each of the availability zones. A total of five EC2 instances are distributed in the two availability zones. These two subnets are associated with the same private route table which is configured to route internal traffic only. A public subnet is created and is associated with public route table which is configured to be accessed over Internet. Private IP addresses are allocated to EC2 instances launched in a subnet depends on the range of CIDR block of the subnet.

I. Security Groups

Security Groups acts as the virtual firewall for AWS resources. It controls inbound and outbound traffic of the instances. Security Groups can be modified easily and will be automatically reflected on the EC2 instances with that security group without restarting instances. Security Group for database servers is configured as follows:

- Network access of database servers is only within AWS resources
- Database servers are not exposed to the world and allow app servers access to it.
- Bastion Host can access the database servers for administrative/maintenance purpose.

J. Bastion Host

A Bastion Host is a special purpose EC2 instance with access to the Internet and acts as a proxy to private instances. This system is launched in Public Subnet and is externally facing to the Internet. Administrative tasks on database servers grow as the number of instances grows. All database EC2 instances are launched in Private Subnets, henceforth no means of access to those instances.

Network access to EC2 instances running privately can be controlled by using Bastion Host. Linux instances use key files for authenticating the access to it instead of username and password. Trying to guess the password to gain access to it can be reduced by the usage of SSH key files. SSH agent forwarding allows to securely connecting to the database servers. Private key used to authenticate the access to the instance need not be stored on the bastion.

Bastion instance is used only for administrative or maintenance purpose. Security Group for this is configured to allow only SSH (TCP/22) connections. Connect to Bastion Host using PuTTY (enable Allow agent forwarding) from Windows and then SSH using private IP address of the instances. Administrative tasks like creating indexes, creating users for the database can be performed securely.

K. MongoDB Replica Set

High availability of data can be achieved using replica set in MongoDB. Usually high availability of data refers to keeping multiple copies of the data. *Replica Set* is the term used by MongoDB – maintaining multiple copies of data. Primary, Secondary, Secondary are the three members in replica set and are distributed across AZs. Replica Set members distributed in multiple AZs are insulated from single point of failure (i.e., failure of a data center). Replica Set members are configured as follows:

- Primary, Secondary in one AZ
- Secondary in other AZ

Automatic Failover is also achieved by this replica set. Hence there is no impact on the availability of the database servers and also eliminates manual intervention of database administrators. Replica Set members can be added and also be removed seamlessly with no downtime of the application.

L. MariaDB Master Master Replication

High availability of data can be achieved using Master Master Replication in MariaDB. Two nodes are used for replication and both are master nodes (both have read, write access). These two nodes are distributed across AZs to achieve multi AZ architecture:

- One Master in one AZ.
- The other Master in the other AZ.

M. Indexing

Indexing is a vital mechanism for optimizing system scalability and performance which provides flexible access to your data. The MongoDB database supports different types of indexes that can be stated on any of the fields in the document. Indexing is the efficient way of executing queries. Creating indexes on the fields in the query improves the read operations and hence reducing the amount of data need to process to fulfill a query. All the indexes created will sit in RAM, hence query execution becomes faster.

N. Caching

MongoDB has native support for caching and hence no explicit configuration is needed for caching. It provides in-memory performance; there is no need for a separate caching layer to scale your database. It keeps indexes and working set in RAM, hence all the queries will be served from the memory. As the data grows, indexes and working set cannot fit in RAM. Deployments should have sufficient memory to hold indexes in RAM and it is not always possible for complete working set to fit in RAM. If indexes and working set exceeds the memory, least recently used pages will be flushed and leaves only most recently used data. Ideally, both indexes and working set should fit in RAM to achieve best performance.

IV. RESULTS & SCREENSHOTS

A. Indexing

The example for MongoDB indexing illustrated is: Search a company by name in a collection of 100 documents. Query Analyses is performed for this search. Index is created for the field “COMPANY_NAME”.

Search	nReturned	totalDocsExamined
Before creating index	1	100
After creating index	1	1

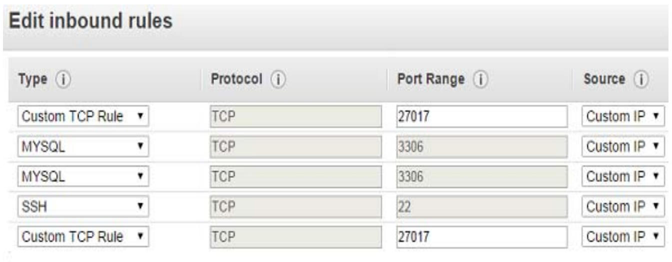
Table 1 Query Analyses

“nReturned”: The factor that determines the number of documents resulting for a query.

“totalDocsExamined”: The factor that determines the total number of documents a query examined in a collection to fetch the result.

The results prove that the indexing is the efficient way of executing queries and hence faster read operations. Indexing improves the performance of the application.

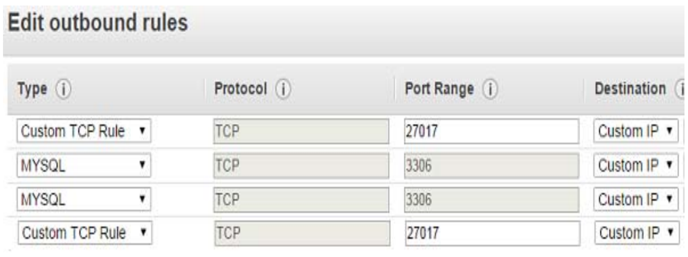
B. Security Group


Edit inbound rules

Type	Protocol	Port Range	Source
Custom TCP Rule ▾	TCP	27017	Custom IP ▾
MYSQL ▾	TCP	3306	Custom IP ▾
MYSQL ▾	TCP	3306	Custom IP ▾
SSH ▾	TCP	22	Custom IP ▾
Custom TCP Rule ▾	TCP	27017	Custom IP ▾

Fig. 2 Inbound rules for data tier

Allow inbound access (Fig. 2) from app tier for both MongoDB (Custom TCP Rule) and MariaDB (MYSQL). Another Custom TCP Rule and MYSQL for internal MongoDB and MariaDB replication respectively. SSH access to secure remote login from bastion host.


Edit outbound rules

Type	Protocol	Port Range	Destination
Custom TCP Rule ▾	TCP	27017	Custom IP ▾
MYSQL ▾	TCP	3306	Custom IP ▾
MYSQL ▾	TCP	3306	Custom IP ▾
Custom TCP Rule ▾	TCP	27017	Custom IP ▾

Fig. 3 Outbound rules for data tier

Allow outbound access (Fig. 3) to app tier from both MongoDB (Custom TCP Rule) and MariaDB (MYSQL). Another Custom TCP Rule and MYSQL for internal MongoDB and MariaDB replication respectively.

V. CONCLUSION & FUTURE WORK

The data tier architecture for job portal application designed addresses high availability, fault tolerance, security, and performance of the system. This is achieved by customizing AWS resources and necessary software.

The system architecture should be efficient enough to handle greater needs as the demand grows for this job portal application and also data grows very rapidly. As a future work, scalability, disaster recovery, automatic failed node replacement, Hadoop for data analytics parameters can be addressed.

REFERENECES

[1] Peter Mell and Timothy Grance, “NIST Definition of Cloud Computing”, September 2011

[2] Rajkumar Buyya, “Introduction to IEEE Transactions on Cloud Computing”, Jan-Jun 2013

[3] Jinesh Varia/Sajee Mathew, “Overview of Amazon Web Services”, January 2014

[4] Andreas Chatzakis, “WordPress: Best Practices on AWS”, December 2014

[5] A MongoDB White Paper, “MongoDB Architecture Guide MongoDB 3.0”

[6] Rahul Bhartia, “MongoDB on AWS – Guidelines and Best Practices”, May 2015

[7] SGT Innovation Center, “Deploying MongoDB and Hadoop to Amazon Web Services – White Paper”, 2015

[8] “Amazon Web Services: Overview of Security Processes”, June 2014

[9] Matt Tavis, Philip Fitzsimons, “Web Application Hosting in the AWS Cloud”, May 2010

[10] Amazon Web Services, “Getting Started with AWS: Web Application Hosting on Linux”, 2014

[11] White Paper, “MongoDB Operations Best Practices”, February 2015

[12] Tom Wheeler, “Inroduction to Apache Hadoop”, OSCON 2013

[13] Derek Downey, “Setting up Multi-Source Replication in MariaDB”, November 2014