University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Automated Deployment of Secure Cloud based Accounting Application

Master's Thesis in Computer Science
by

Malik Bilal

Internal Supervisors

Jayachander Surbiryala

June 15, 2021

Declaration of Authorship

I, Malik Bilal, declare that this thesis titled, 'Automated Deployment of Secure Cloud based Accounting Application' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master's degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Malik Bilal

Date: June 15, 2021

"Sometimes life is going to hit you in the head with a brick. Don't lose faith."

Steve Jobs

Abstract

With the rapid growth of Companies' trust and willingness to move their on premises based Software solutions to the Cloud environments , multiple challenges and requirements arise due to these shift.First and foremost is related to the security of their resources that are hosted inside the cloud environment. As the infrastructure that is used to provide computational, storage, networking, and other resources that are required to host application, are not in full control of the customers.One requirement is related to monitor the overall working of the applications that are being shifted to the the Cloud environment. With monitoring , we refer to how the application is being used by the customers , are there any anomalies inside the application and other issues that are important to monitor.Similarly, the customer also want to automate the deployment of the application in the cloud to remove the unnecessary time that is spent in the life cycle of application from the Development environment to the Production. Also there is one more related thing with the deployment and that is versioning of the software because it is important to know which version of the application is being deployed in the cloud as application can have many test versions as well but we only want an application that is tested properly and is bug free. It is therefore important to see how we can deploy application inside a cloud environment while have a complete control of the behaviour, security, its versions, and its deployment. To address these issues, we have taken an accounting application application which is being hosted inside Microsoft Azure and develop a monitoring architecture , a version control system , proposed a security architecture and an automatic Deployment Process to deliver this application to target customers.

Acknowledgements

First and foremost, praises and thanks to The God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.I would like to thank my supervisor, Dr.Jayachander Surbiryala, for guidance and invaluable feedback throughout my work on this thesis.It is an honor to work under his supervision. He has extended his support to complete this thesis through suggestions and encouragement.I would also like to thank My Manager Mr Jon Terje Aksland who has been very helpful in providing resources, and much valuable feedback on the thesis as a whole.Finally, I would like to thank my family for all their support during my master's degree.

Contents

Abbreviations

CI	**C**ontinuous **I**ntegration
CD	**C**ontinuous **D**elivery
APM	**A**pplication **P**erformance **M**onitoring
AAD	**A**zure **A**ctive **D**irectory
AES	**A**dvanced **E**ncryption **S**tandard
ADE	**A**zure **D**isk **E**ncryption
BYOD	**B**ring **Y**our **O**wn **D**evice
ELK	**E**lastic **L**ogstash **K**ibana
DevOps	**Dev**elopment **Op**erations
RESTful	**Re**presentational **ST**ate **T**ransfer
RSA	**R**ivest **S**hamir **A**dleman
JSON	**J**ava **S**cript **O**bject **N**otation
LVC	**L**ocal **V**ersion **C**ontrol
CVC	**C**entralised **V**ersion **C**ontrol
DVC	**D**istributed **V**ersion **C**ontrol
Dev	**Dev**elopment
VHD	**V**irtual **H**ard **D**rive
TDE	**T**ransparent **D**ata **E**ncryption
SQL	**S**quential **Q**uery **L**anguage
SSO	**S**ingle **S**ign **O**n
OSI	**O**pen **S**ystem **I**nterconnect
WAF	**W**eb **A**ppliction **F**irewall
MFA	**M**ulti **F**actor **A**uthentication
VNET	**V**irtual **NET**work
IP	**I**nternet **P**rotocol

RC	**R**eleased **C**andidate

List of Figures

Chapter 1

Introduction

With the shift of hosting application from on-premises data centers to the cloud environments, multiple challenges are faced because of this transfer. These challenges range from monitoring application, its security, automating its deployment in the cloud environment. In this thesis, we are going to address these issues and find solution to them that are being faced in the deployment of an accounting application hosted in Microsoft Azure. This Accounting application includes a complete development and distribution environment in the cloud, with resources that allow for significant cost reductions for companies that want to create new accounting solutions adapted to their own business. Although, we are using Accounting application but we are not going to use Accounting terminologies is this thesis therefore, there is no need for the reader to have basic understanding of Accounting.

In the current setup for this application, there is one separate deployment of the application in the cloud for each client. Currently, this application has all the aforementioned hurdles.First, we are going to deal with the Monitoring issue , where we are going to provide a solution with which an the application can be monitored keeping in view its monitoring requirements. Next, we are going to deal with the Versioning issue. As the application is huge and it contains a lot of parts therefore there is no such mechanism to version it properly to be used in the production environment. In next step, we are going to propose a Layered Security solution for the application, using different technologies that are being offered by the Microsoft Azure. In the final phase of the thesis, we will provide a complete automation solution where we are going to automate the deployment process with CI CD tools. So based on the issues that are being faced by the application,

Chapter 2

Background and Research Questions

As discussed in the introduction we are going to work on four different areas in this thesis namely

1. Application Monitoring
2. Version control
3. Application Security
4. Automated Deployment

Let us have a brief overview of what they mean the we are going to proceed with the research questions related to them.

2.1 Application Monitoring

Also known as Application Performance Monitoring or Management (APM) is one of the core requirements in the software life cycle. It is a process that monitors whether a software is performing as per the requirements within the confined scope. It can also be defined as a “a process that ensures that a software application processes and performs in an expected manner and scope. This technique routinely identifies, measures, and evaluates the performance of an application and provides the means to isolate and rectify any abnormalities or shortcomings”[10].

2.2 Version control

Software Versioning or version control also known as revision control or source control system. "It is the process of assigning either unique version names or unique version numbers to unique states of computer Software". Within a given version number category (major, minor), these numbers are generally assigned in increasing order and correspond to new developments in the software. At a fine-grained level, revision control is often used for keeping track of incrementally different versions of information, whether or not this information is computer software". [11]

2.3 Application Security

Security is one of the most important aspects of any Software application architecture[12]. Ensuring that your business data and customer data are secure is critical. A public data breach can ruin a company's reputation as well as cause significant personal and financial harm. Once you have hosted your application inside cloud the responsibility of securing the application depends upon which type of services you are taking from the cloud to host your application. But usually , both the cloud providers and users have a shared responsibility of securing the application.

2.4 Automated Deployment

"Application Deployments define the package of software components that make up an application in a particular environment, e.g. development or production. Instances of these are deployed onto physical Technology Nodes to capture where that software is executing. "[13] As just like other living things, any application also has life cycle where it continues to grow and change. With every change, either there is something added to the application or there is some fix to the previous bugs that were discovered in the previous release. For this new application to be reflected in the production environment, where application is being used by customers, is a continuous process. Which, if it is done manually, can be error prone and time consuming. Therefore, various techniques have been developed where this process is automated in a way that whenever there is some change in the application code of the software and that you want to deploy it to the test or the production environment can be done either in few clicks or is done automatically. This automation process is also called Continuous Integration (CI) Continuous Delivery (CD). Let us define them one by one we have divided the thesis based on the four research and implementation questions. Which are:

2.5 Research Questions

2.5.1 Research Question 1

Is it possible to setup a monitoring solution outside the cloud environment?

In this thesis we are going to find out whether we can set a monitoring solution for accounting application with some characteristics that are going to be discussed in the relevant section.

2.5.2 Research Question 2

Can we Develop a Version Control System for the Application that is consistent with the needs of the organization?

We are going to explore different software versioning techniques and select one that is more suitable to our environment

2.5.3 Research Question 3

Research on the security possibilities inside the hosted environment of application and propose a security architecture.

We will explore different security tools within Microsoft Azure and will propose a security architecture that protects all the aspects of the application including data, network (all the layers), identity and other resources that are crucial for the application hosting.

2.5.4 Research Question 4

Can we automate the deployment process of the application to the target environment?

Lastly, we are going to research and architect an Continuous Integration and Continuous Delivery architecture to automate the deployment process

2.6 Contributions

The main contributions of this thesis are:

1. We have developed a Monitoring solution outside the cloud environment where the application is hosted.This solution is cost efficient,easy to set up , very versatile and have storage capabilities.
2. We have Proposed a layered Security architecture to secure the data using advance encryption techniques, secure network architecture to provide network security and identity protection mechanisms for authentication and authorisation .
3. We also have setup a Software versioning system by researching on different versioning techniques and setting a versioning solution.
4. We have also automated the process of build and deploy of application so using different CI and CD tools and a pipeline that would control the flow of deployment of the application from the development untill the production stage.

2.7 Structure of Dissertation

This thesis is organized as follows: In Chapter 3, we have presented a monitoring solution for a cloud based application. In Chapter 4 , we have presented a Versioning solution. .In Chapter 5 ,we have proposed a Security architecture to secure the application.In Chapter 6 , we have used CI CD tools to automate build and deploy application.In Chapter 7 we are going to Conclude our thesis with Research Questions ,their answers and Future work.

Chapter 3

Application Monitoring

In this chapter we are going to research on the monitoring aspect of the application. Application monitoring is achieved by enabling a special piece of software inside the application being monitored sometimes called monitoring agent[14]. This agent instruments the application and sends the collected information to the destination where you can visualize, analyse, and respond to the events. Figure 3.1 is a generalized architecture of the whole process.

3.1 Monitoring Requirements for Application Monitoring

Following are the requirements with the monitoring of our application:

- The monitoring solution must be easy to set up.
- The monitoring solution must not be very costly.
- There must be some way where the customers can also monitor the status of the application, they are using.

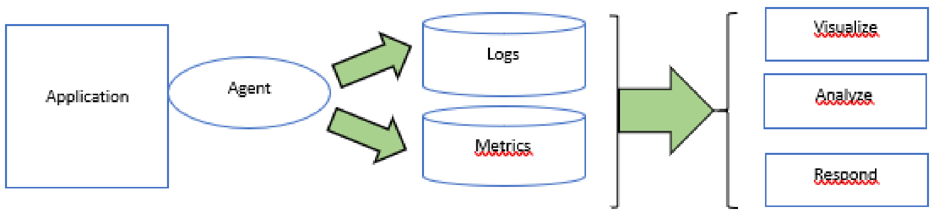


Figure 3.1: A Generic architecture of the Application Monitoring Process [1]

- There must be some way of storing the logs and metrics from the monitoring tool for future analyses and studies.
- It should monitor every aspect of the application.
- It should provide the basis to troubleshoot issues with the application and easy to pinpoint the problem with the application.

3.2 Monitoring Solutions

There are many Application Monitoring solutions available in the market for monitoring purposes but here we are only going to mention the major ones. These monitoring solutions are:

- Application Insights[15]
- Raygun[16]
- Elastic Stack[17]
- Logz.IO[18]

First, we are going to mention the monitoring solutions that we are not using and the reasons behind them, then we are going to proceed with the deployed application monitoring mechanism along with the tool that we are currently using, the issues with it and the propose and setup a solution for these issues.

3.2.1 Raygun

Raygun Application Performance Monitoring (APM) provides server-side application performance monitoring. Raygun provides both the logs and metrics of your application deployed along with the alerting mechanism in case if there is an issue with the application[19]. The unique issue creation agent of the tool prioritises the issues for the application therefore you don't need to worry about sorting them as they have already been sorted based on their criticality for the application. It also provides a wide variety of integration options as well. Applications made in node.js, .net, .net core, react etc. Also, it has a specialised plugin for the applications deployed in the Microsoft Azure. With this plugin it becomes easy for the Raygun to integrate with application. As far as pricing is concerned, it provides three different pricing models. Either you can choose Application Performance Monitoring which monitor only your

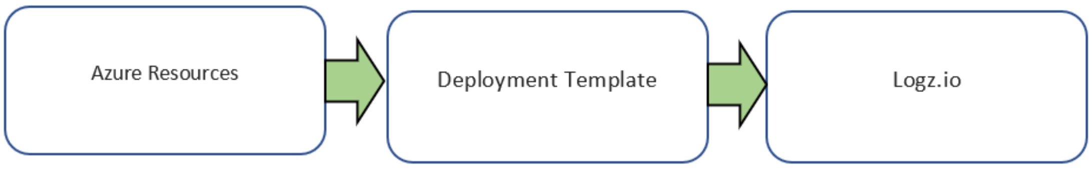


Figure 3.2: Logz.io Generic architecture

server-side performance bottlenecks, or you can choose Real User Monitoring which is used to monitor your front-end performance, or you can choose Error Monitoring and Crash Reporting Monitoring solution which pinpoints different errors and crash affecting the application. As for the monitoring requirement we need all the different monitoring pricing models. Based on the amount of data that is going to be ingested the price increases drastically. Though the price for using this tools is on the higher side but the main reason for not using it is its Data Protection Policy in which it is said clearly that all the customers data will be stored in US and will be under US jurisdiction [20]. Looking at the nature of the application, Raygun compliance to the latest Data Processing law called GDPR (General Data Protection Regulation) introduced on 25th May 2018, makes this monitoring tool unsuitable as the data is very sensitive as it contains customers banking related information and therefore cannot be exported to other countries for processing

3.2.2 Logz.io

Logz.io is a Monitoring solution based on open-source ELK (Elasticsearch, Logstash, and Kibana) stack and a combination of various other monitoring tools that provides Infrastructure monitoring, logs management, Application Performance Management, Cloud SIEM and distributed tracing. It gives you a complete visibility inside your application by integrating application from 50 plus different sources from different providers [21]. It can also easily integrate easily with the applications hosted in Microsoft Azure. This integration is based on the ready-made azure deployment templates that sets up all the necessary building blocks of the pipeline — an Events Hub namespace, two Events Hubs, an Azure Function app, two Azure Functions, two Azure Storage Blobs, and all the correct permissions and connections required . The shipment of logs and events from Azure environment to the logz.io environment can be depicted in the Figure 3.2.

Here whenever there is an event or log an azure function is triggered that sends data to either an Event Hub (Data ingestion Service) from where it is forwarded to the logz.io account for visualization and analysis purposes. The data that is streamed to the Event

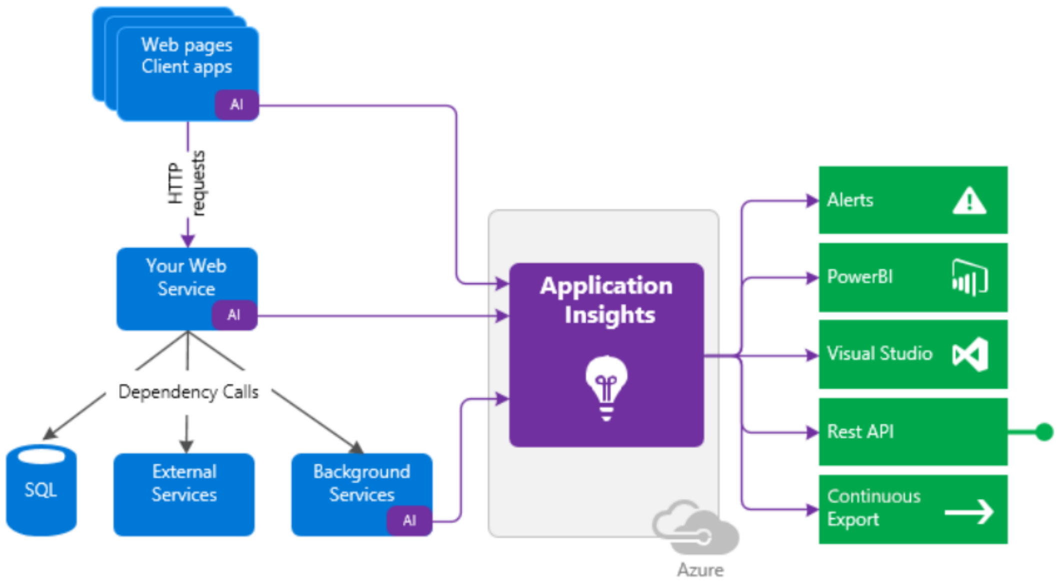


Figure 3.3: Application Insights Generic Architecture (taken from [2])

Hub can also be stored inside Azure Blobs (Azure storage Service) for future uses. As far as cost is concerned the service is based on per GB data usage ingestion. The cost of the service rises exponentially with the increase of data being ingested from the Azure. The expected data that is going to be ingested is close to 60 GB and is envisaged to grow that is why logz.io is not a suitable solution. Now the solution that we are currently using and the solution we are going to use for the application

3.2.3 Application Insights

Application Insights, a feature of Azure Monitor, is an extensible Application Performance Management (APM) service for developers and DevOps professionals [22]. It is a powerful monitoring tool that if enabled inside the application detect issues in the application and usage of it by the end users. It can integrate with the applications build on NET, Node.js, Java, and Python hosted on-premises, hybrid, or any public cloud. It can also monitor and analyse logs and metrics from mobile based applications. 1 To implement application Insights, you need to instrument the application that is going to be monitored. This can be achieved by installing an instrumentation package (SDK) in the application. This will install an agent inside your application which will start instrumenting the application. The data collected by the agent will be sent to the application insight resource based on the unique Instrumentation key. This setup can be shown in the Figure 3.3.

With the setup we can monitor number of requests, their response time and failure rates, dependency in the application on other resources, number of exceptions in the applications, how the users page are being loading, performance counters and many other applications monitoring stuff. As far as cost is concerned, there cost is charged based on the amount of data that is ingested. But the bills can be paid in two ways:

- Pay as you go
- Capacity Reservation.

In the first model you pay as you use the data and there is no discount while in the second method of Payment to you can make the reservation of data that will be ingested, and you can get up to 25 Percent discount on your bills. You can also cancel the reservation after the 31st day of subscription [23] Owing to the ease of integration and the lower pricing model, we are using the Application Insights to monitor our application, but it has some draw backs as well one of the main drawback is that we cannot share the dashboards from inside the azure to outside to the customer without giving them access inside azure environment which does not seems to be feasible for the security purposes because it would expose the internal proprietary structure of the application which is not supposed to be exposed. Therefore, we must work around a way where we can create a monitoring dashboard for the customers without bringing them inside the deployed application environment. This can only be done if we can send the monitoring data outside from azure to 3rd party monitoring solution and create the monitoring dashboard for the customers. Here we have opted for Elastic Stack to accomplish this task of creating monitoring dashboard for the customers.

3.2.4 ElasticStack

Formerly called ELK stack is a group of open source software namely: Elasticsearch, Logstash, Kibana and Beats used for Searching, Observability and Security purposes. Each component of stack performs a specific job. Elasticsearch is used to store and search data , Kibana is used to visualize the data stored in the Elasticsearch in the dashboard while Beats and are data shippers and Logstash is used as a pipeline to import and filter data before they are shipped to the Elasticsearch. The stack can be visualised in Figure 3.4.

Let us have a brief overview of each component so it is easier for us to understand working of elastic stack. We will follow the bottom up approach to understand the Elastic stack. Here the data can be shipped using either Beats or Logstash depending upon the requirements and architecture of your application. For our architecture we will be using Logstash as a pipeline to import data but we will also have a look at the Beats to understand the various uses of Elastic Stack.

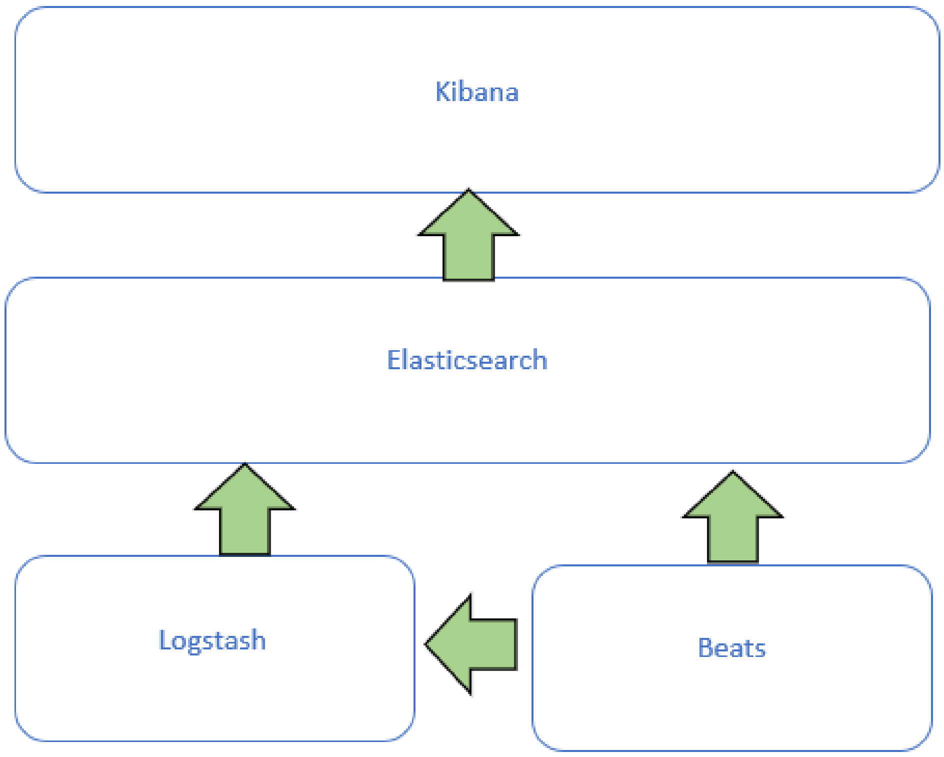


Figure 3.4: An Overview of ElasticStack [3]

3.2.4.1 Beats

"Beats is the platform for single-purpose data shippers. They install as lightweight agents and send data from hundreds or thousands of machines to Logstash or Elasticsearch"[24]. Elasticsearch provides multiple beats to ship different type of data from the data source .To name a few : Audit beat are used to ship audit data, file beat are used to ship log data, Function beat are used for Cloud data, Heartbeat are used for availability of the server being monitored. The data from beats family can be either sent directly to the Elasticsearch or to the Logstash (for further refinement) before being visualised in Kibana. Figure 3.5 shows the architecture.

Here depending upon the nature of data to be shipped one or several beats data shippers are installed inside the data source. The data is then transferred to either Elasticsearch (if there is no need of further processing) or it is pipelined to Logstash where the data is refined according to the needs and is then sent to the Elasticsearch from where it is indexed by Kibana for visualisation purposes.

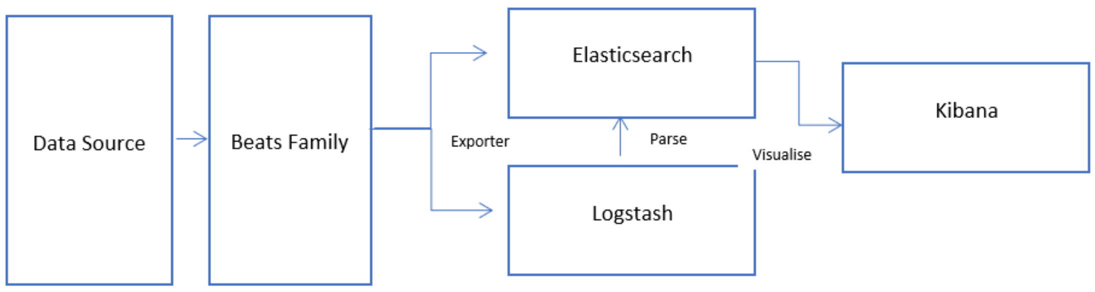


Figure 3.5: Different scenario of log shipment [4]

3.2.4.2 Logstash

"Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favourite stash."[25] Logstash has the capability of combining data from different resources and sent to the destination. After data collection it can enrich, the data can be changed and twigged with various kinds of filters and output plugins. The power of Logstash lies in its ability to scale horizontally with the increase in data flow, pluggable pipeline architecture and huge range of community-extensible plugin ecosystem. Logstash pipeline has three components: Input, filter, and Output. In input stage the events that are fed into the pipeline are queued either in memory or on disk. Then each pipeline worker takes a batch of these events runs them through the configured filters and then output them to the Elasticsearch or any other storage media of choice.

3.2.4.3 ElasticSearch

"Elasticsearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data for lightning fast search, fine tuned relevancy, and powerful analytics that scale with ease."[26] The data inside the Elasticsearch is stored in JSON document format. With multiple Elasticsearch nodes in elastic cluster, the documents are shared in these nodes making them documents available in multiple nodes. The documents search in Elasticsearch is fast and occurs within 1 second due to a use of indexing technology called "inverted index" that supports very fast full text searches. This fast searching capability is achieved by listing every unique word that appears in any document and points out to each document in which that word occurs. An index, in the Elasticsearch, consist of combination of multiple documents and each document itself consist of fields, while each field is a key-value pair that has data in it. Elasticsearch indexes each field which in turn is safe in a dedicated and opti-mized data structure. This whole architecture makes the Elasticsearch capable of fast search.

3.2.4.4 Kibana

"Kibana enables you to give shape to your data and navigate the Elastic Stack."[27] Sitting at the top of the stack, Kibana enables the user to search, observe and protect the data stored inside the Elasticsearch. With Kibana, the user can have insight of the data and make them available as charts and different visualization techniques. The user can also manage the indices and pipelines. It can also monitor the overall health of the elastic stack cluster and manage the permissions of the users.

Having discussed the main components of the elastic stack, one of the main requirements to monitor the application is to provide the monitoring dashboard for the users who are using this application. The dashboard provided by application insights cannot be shared with the users outside the azure environment until we give them some rights inside azure which is undesirable for the reasons that it will expose the internal architecture and usage to the customers Therefore, there is requirement to export the monitoring capability outside the Microsoft environment. For this we needed a solution where we can export the monitoring data outside azure and share them with the customers in the form of dashboards.

3.2.5 Architecture to export Monitoring Data outside Azure

We did a lot of research and brainstorming to put multiple pieces of diverse technologies together to export monitoring data from Azure to Elastic stack. The monitoring data for the application is residing inside the Application Insights that we are using for monitoring our application. So we requires some means where we can collect this data at some place , prepare it for ingestion , ingest the data and export it to a place where it can again be collected ,indexed and can be displayed. After a lot of research and trials we have come up with an architecture which does the following things:

- Continuous Export data from Application Insights to azure Blob storage
- Whenever there is a new log or event stored inside a blob storage account, we export that data to Azure Event Hub for ingestion
- From Azure Event Hub the ingested data is exported via Logstash pipeline to the Elasticsearch database
- From Elasticsearch database the stored data is indexed and is displayed using Kibana

Information can be shown diagrammatically in the Figure 3.6.

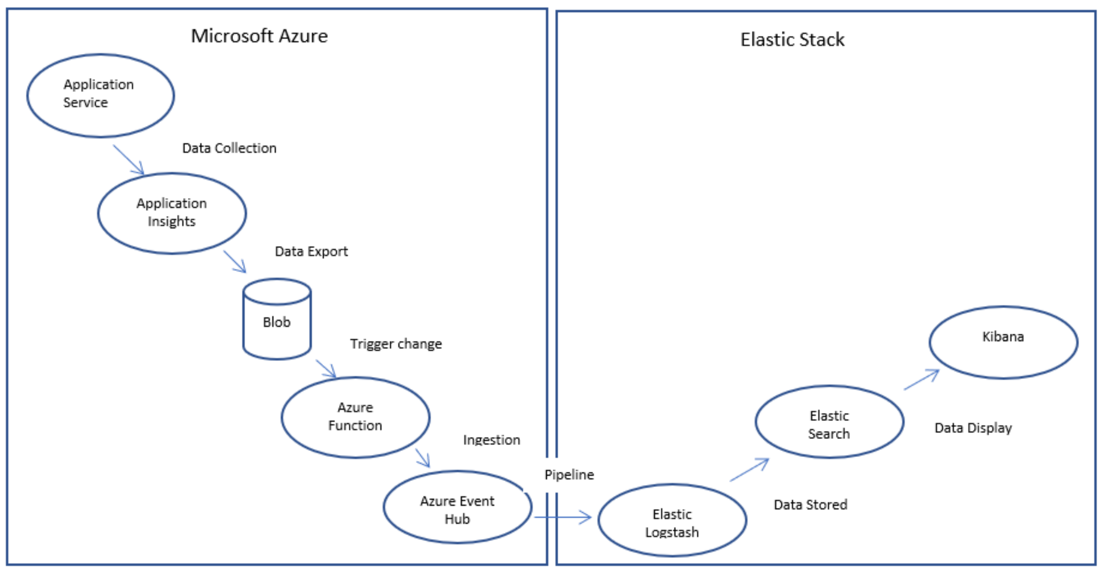


Figure 3.6: Export Monitor Architecture

3.2.5.1 Microsoft Azure setup

For preparing the Azure environment to export the monitoring data to Elastic stack following things must be considered that we must first select different from a large set of Azure technologies and connect them in such a way so as to accomplish our task of exporting data outside of Azure. For this we have selected following technologies to do these tasks.

- Application Insights to collect monitoring data from the Application environment.

- Azure Blob Storage to store the monitoring data collected from the Application Insight

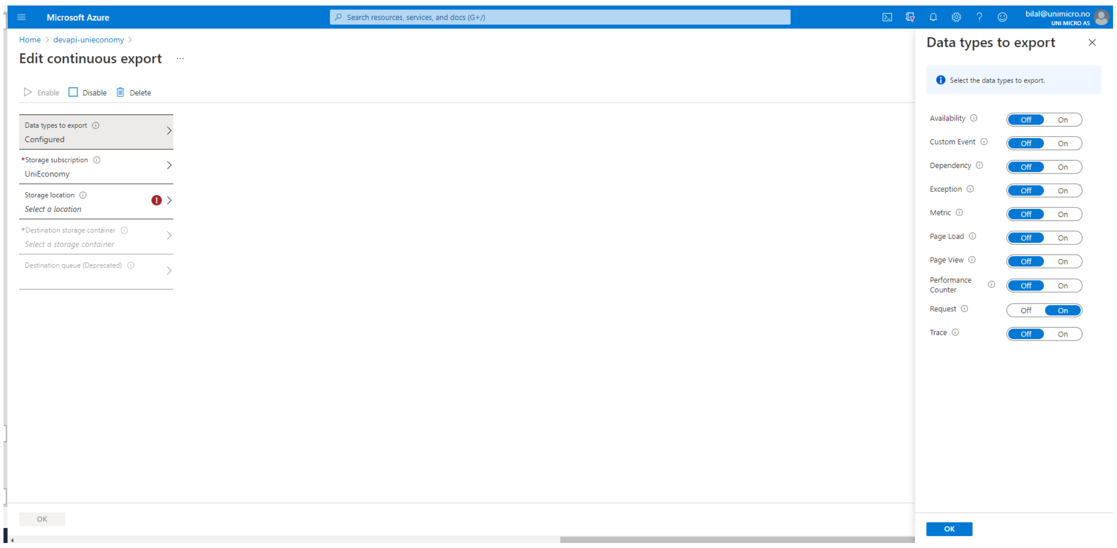
- Azure Function to monitor the Blob storage and trigger whenever there is some update in the blob storage account and transfer that to the Azure Event Hub

- Azure Event Hub to get the stream of events collected by the Azure function, prepare it be ingested outside to the destination.

Let us have a brief overview of what these technologies are, how they work and how they relate to each other to fulfil our requirements. We have already discussed Application Insights so we will skip it but we will explain how the data from Application Insights is saved to Azure Blob storage.

3.2.5.1.1 Azure Application Insight Continuous Export

As mentioned before that Application Insight is a part of Azure monitoring solution provided by the Azure that if enabled inside the Application detects the anomalies in the application and how it is being used. It collects this information in the form of various logs and metrics, but this information is not stored anywhere. If you want to use the telemetry collected by it, you must export and store it somewhere for future usage purposes. The events collected by the Application Insight can be stored using Continuous export available in the Application Insight setting. This stores the logs in the JSON format to the destination storage. There are different things you need to mention and select while configuring Continuous Export. These options are what data type to export. Whether you want to export Availability data, Custom Event, Exceptions occurring in the application, Metrics, Performance counter and other settings to choose. Then you must select where and which storage account to select. In our scenario we have only selected Request Counter because now we are only interested in total number of request and data related to them to be exported outside so as to see how the application is performing overall. These settings can be seen in the Figure 3.7.



Data types to export	Configured
*Storage subscription	UniEconomy
Storage location	*Select a location*
*Destination storage container	*Select a storage container*
Destination queue (Deprecated)	

Data types to export

Select the data types to export.

Availability	[✓ Off] On
Custom Event	[✓ Off] On
Dependency	[✓ Off] On
Exception	[✓ Off] On
Metric	[✓ Off] On
Page Load	[✓ Off] On
Page View	[✓ Off] On
Performance Counter	[✓ Off] On
Request	Off [✓ On]
Trace	[✓ Off] On

Figure 3.7: Different scenario of log shipment

3.2.5.1.2 Azure Blob Storage

Azure Blob storage is highly scalable, secure, and durable storage service provided by Azure to store unstructured data inexpensively [28] . Selected monitoring data collected by Application Insight is now being continuously transferred by Application Insight inside Azure Blob Storage container. After the data is being continuously transferred to azure blob storage, the next step is to monitor this storage in such a way that whenever there is some update in the blob storage made by Application Insight, in the form of new logs and metrics added to, this can be detected and transferred to the Azure Event hub for ingestion outside azure. To accomplish this task, we have used Azure function.

3.2.5.1.3 Azure Function

Azure Functions is a serverless solution that allows you to write less code, maintain less infrastructure, and save on costs [29]. Instead of worrying about deploying and maintaining servers, the cloud infrastructure provides all the up-to-date resources needed to keep applications running. Azure function allows developers to implement Systems logic into block of code. These blocks are referred as Functions. Depending upon the requirement these functions can run any time based on the system need. These functions are scalable as well,meaning that they will automatically increase their size as per the requirements. Azure function consists of at least one trigger that causes it to start processing. There are wide variety of triggers available for Azure functions in the azure portal. The second part of the function consist of the processing logic which tells what we want to do with the data captured inside the trigger's third and last part is what we call bindings which binds logic with the input or output . In our scenario we want this function to detect the changes in the azure container, capture them and transfer them to the Event hub. To achieve these goals, we have set up the azure function with the blob storage trigger. This blob storage is triggered whenever there is some change inside the connected blobs defined in it. These blobs are the one which are defined in the previous step where we were storing our Application Insight data. Furthermore, this function is also attached with Event Hub with the output bindings which will connect it with the Azure Event Hub. The above-mentioned configurations are done in a JSON file. For the sake of security, I am only going to put few lines.

```

"config": {
            "bindings": [
                {
                    "name": "myBlob",
                    "type": "blobTrigger",
                    "direction": "in",
                    "path": "test/{name}",
                    "connection": "AzureWebJobsStorage"
                },
                {
                    "name": "outputEventHubMessage",
                    "direction": "out",
                    "type": "eventHub",
                    "connection": "
    testelknamespace_RootManageSharedAccessKey_EVENTHUB",
                    "eventHubName": "testevent"
                }
            ]
        }
```

Listing 3.1: Azure function configuration

In Listing 3.1, the first binding is the Azure blob storage trigger as can be seen with the type being "blob-trigger". The path is the location of the blob storage being monitored and connection is the name of the connection string which contains all the credentials and rights to write to this storage ac-count. Similarly, the second part is the output binding where the data will be sent. The settings contain the name, which tells how the EventHub will be mentioned inside the azure function. Direction tells that the data will be sent to it. Type tells what kind of binding it is, here it is EventHub. Connection contains the connection string to the Event hub. While the last parameter tells the name of the EventHub inside the EventHub namespace where the events would be sent. The third and last part of the Azure Function is the logic of the azure function which processes and captures the data from the azure blob storage trigger, reads the data from it and sent it to the Event hub. Some of the logic to do this task is given below

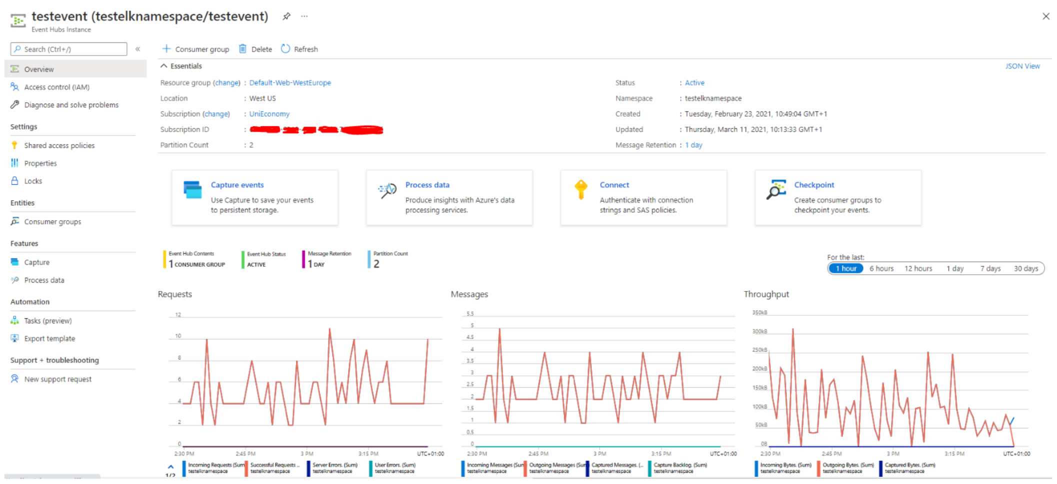
```


public static void Run(Stream myBlob, string name, out string
    outputEventHubMessage,ILogger log)
 //[return: EventHub("testevent",Connection = "testelk-
    namespace_RootManageSharedAccessKey_EVENTHUB")])
{
    log.LogInformation($"C# Blob trigger function Processed blob\n Name:{
    name} \n Size: {my-Blob.Length} Bytes");
    StreamReader reader = new StreamReader(myBlob);
    log.LogInformation($"{reader}");
    string jsonContent = reader.ReadToEnd();
    log.LogInformation($"{jsonContent}");
    outputEventHubMessage = jsonContent;
    //return ($"{jsonContent}");
    //await outputEvents.AddAsync(JsonConvert.SerializeObject(jsonContent
    ));
}
```

Listing 3.2: Azure function code

In Listing 3.2 we have defined a function which is taking stream from blob and sending it to the Event hub. First the function is reading all the data from the blob storage inside a variable called reader in line 14 and then it is passing this to the Event hub parameter called "out-putEventHubMessage". Once the messages are sent to the Event hub, that is already setup, it will start ingesting the data

3.2.5.1.4 Azure Event Hub

"Azure Event Hubs is a scalable event processing service that ingests and processes large volumes of events and data, with low latency and high reliability" [30] components Event Producers, Partitions, Consumer groups, throughput units and Event receivers. Event producers is any entity that produces some data and send it. In our case it is the azure blob storage that is producing the event and being sent to the Event hub using azure function. Partition are the place from where the readers of the events read the data ingested by the Event hub. In our case we have two partitions count. Consumer groups are the groups that reads the partitions. Here we have only one consumer group "Default". Throughput units control the capacity of the Event hub using the throughput units. In our case we have set it to "auto inflate" so it will adjust its capacity as per the amount of data being sent by the producers. After this configuration, Azure Event hub have started receiving events that can seen in the 'Messages' chart. As shown in the Figure 3.8 .


testevent (testelknamespace/testevent)
Event Hubs Instance

Essentials

Resource group (change) : Default-Web-WestEurope
Location : West US
Subscription (change) : UniEconomy
Subscription ID : [redacted]
Partition Count : 2

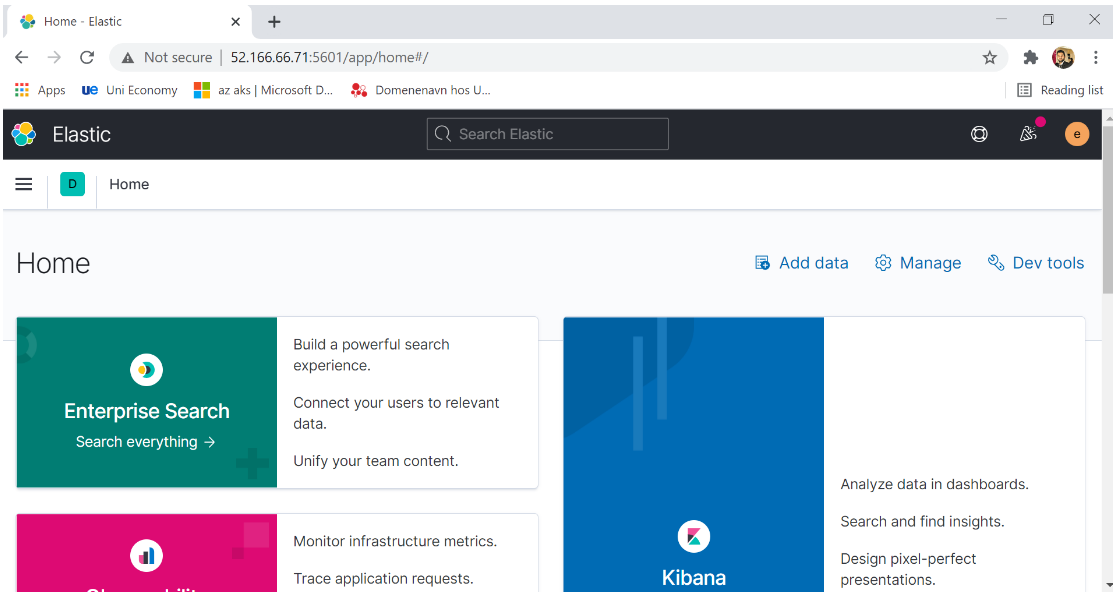
Status : Active
Namespace : testelknamespace
Created : Tuesday, February 23, 2021, 10:49:04 GMT+1
Updated : Thursday, March 11, 2021, 10:13:33 GMT+1
Message Retention : 1 day

Event Hub Contents: 1 CONSUMER GROUP
Event Hub Status: ACTIVE
Message Retention: 1 DAY
Partition Count: 2

Figure 3.8: Event Hub Input and Output flow

3.2.5.2 Setup Procedure for Elastic Stack

To set up the above architecture, first thing that I installed was an Elastic stack. For this I chose Ubuntu VM hosted inside Microsoft Azure. As Elastic stack itself comprises of three different open-source products, we must install them separately and configure them so that they can interact with each other and work as one stack. During the setup process, I face various issues related to configuring the stack as Logstash was unable to communicate with the Elasticsearch database so that it can store its output. After days of research and trying different configurations finally, the elastic stack was installed successfully and was accessible at the public IP address. Shown below is the capture from the installed elastic stack Kibana dashboard in Figure 3.9.



Build a powerful search experience.

Connect your users to relevant data.

Unify your team content.

Monitor infrastructure metrics.

Trace application requests.

Analyze data in dashboards.

Search and find insights.

Design pixel-perfect presentations.

Figure 3.9: Elastic stack User Interface

Here you can only access Kibana dashboard using UI. If you want to access Logstash and Elasticsearch you need to use any telnet software like Putty etc to ssh into the machine where the Elastic stack is installed and is ready to process the data. But to do so we will write a Logstash pipeline to take the data from the source, filter it and place it in Elasticsearch database where it will be indexed and visualized in Kibana dashboard. Each Opensource installed software is hosted on the same IP address but using the different port number for example Kibana uses TCP port 5601 and Elasticsearch uses TCP port 9200. In the start we only setup the elastic stack and did the integration step with the Azure after preparing Azure to export monitoring data outside. After setting up the Elastic stack the next step was to prepare Azure environment so that we can export the monitoring data to Elastic stack.

3.2.5.2.1 Logstash Pipeline

The next step is to configure and programme Elastic stack to start receiving event form the event hub. For this purpose, we will use the elastic stack that we have configured earlier but now we have to programme a Logstash pipeline that will start importing data from the Azure Event Hub. As men-tioned before Logstash pipeline has three components. First is the input which takes the input from the different sources, next being the filter which shapes the data according to the desired needs of the system and last being the output which throws the data to output source. The pipe-line that has been made to receive the events and do some processing and send the events to the Elasticsearch database. Let us explain the Logstash pipeline components one by one.

3.2.5.2.1.1 Logstash Input

Following is the code snippet that defines the input to the Logstash pipeline

```

input {
  azure_event_hubs {
    config_mode => "advanced"
    event_hubs => [
      { "testevent" => {
        event_hub_connection => " EventHub_connection_string"
      }}
    ]
    threads => 8
    decorate_events => true
    consumer_group => "$Default"
    storage_connection => "Storage account connection String
    type => "azure_event_hub"
    initial_position => "beginning"
```

Listing 3.3: Logstash Pipeline Input

In Listing 3.3, we have defined two things, first what would be the source of the pipeline and second it requires a storage account where it can store its processing status. In the first part we are defining the input to Logstash and that being, in our case is, Event hub. Here we are going to pass two things, then name of the EventHub and the connection string to connect with it. In the second part of the Input we are defining where would be the storage account with which Logstash will connect and store its status.

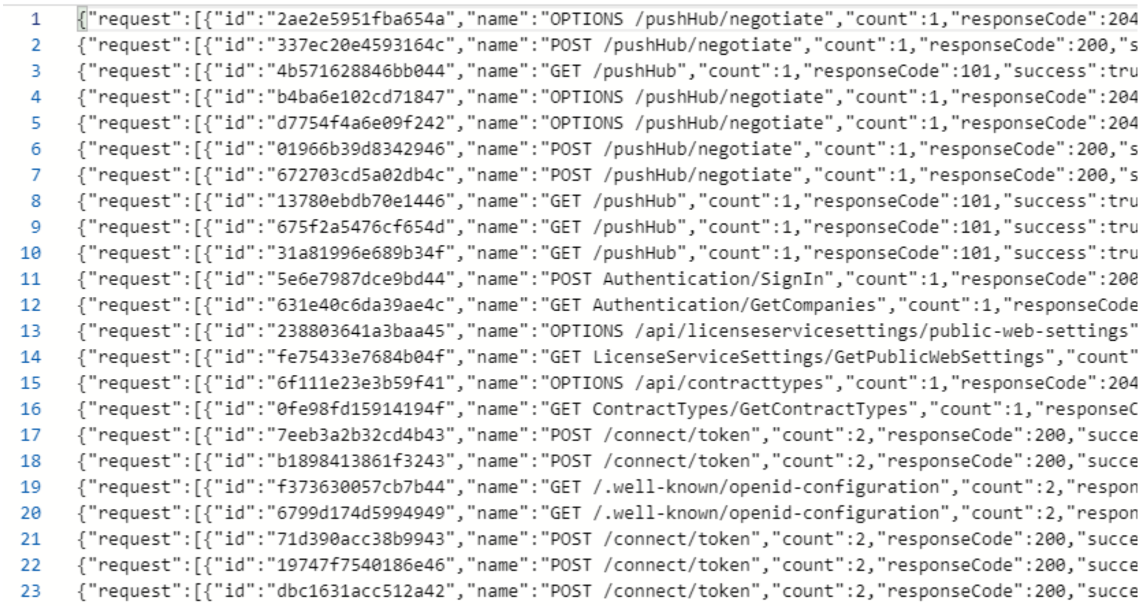
3.2.5.2.1.2 Logstash filter

In the second part we are going to filter the input received. Here we are going to define what will be the type of input and how we are going to process it. Below is the filter part of the Logstash pipeline

```

filter {
     json {
        source => "message"
        remove_field  => [ "context"]
        remove_field => [ "internal.data.documentVersion"]
        remove_field => [ "@version"]
}
      split {
        field => "message"
      }
```

Listing 3.4: Logstash Pipeline filter

In listing 3.4 , we have defined that the message would be in JSON format also after receiving the data we have removed some of the unnecessary fields from the data like "context" etc. Here we were facing an issue because the pipeline was reading the whole blob and each blob was containing a lot of messages inside it and we wanted that information to be extracted from the blob and is sent to the Elasticsearch database as a distinct message.In 3.10 you can find the contents of the blob.



```
{"request":[{"id":"2ae2e5951fba654a","name":"OPTIONS /pushHub/negotiate","count":1,"responseCode":204
{"request":[{"id":"337ec20e4593164c","name":"POST /pushHub/negotiate","count":1,"responseCode":200,"s
{"request":[{"id":"4b571628846bb044","name":"GET /pushHub","count":1,"responseCode":101,"success":tru
{"request":[{"id":"b4ba6e102cd71847","name":"OPTIONS /pushHub/negotiate","count":1,"responseCode":204
{"request":[{"id":"d7754f4a6e09f242","name":"OPTIONS /pushHub/negotiate","count":1,"responseCode":204
{"request":[{"id":"01966b39d8342946","name":"POST /pushHub/negotiate","count":1,"responseCode":200,"s
{"request":[{"id":"672703cd5a02db4c","name":"POST /pushHub/negotiate","count":1,"responseCode":200,"s
{"request":[{"id":"13780ebdb70e1446","name":"GET /pushHub","count":1,"responseCode":101,"success":tru
{"request":[{"id":"675f2a5476cf654d","name":"GET /pushHub","count":1,"responseCode":101,"success":tru
{"request":[{"id":"31a81996e689b34f","name":"GET /pushHub","count":1,"responseCode":101,"success":tru
{"request":[{"id":"5e6e7987dce9bd44","name":"POST Authentication/SignIn","count":1,"responseCode":200
{"request":[{"id":"631e40c6da39ae4c","name":"GET Authentication/GetCompanies","count":1,"responseCode
{"request":[{"id":"238803641a3baa45","name":"OPTIONS /api/licenseservicesettings/public-web-settings"
{"request":[{"id":"fe75433e7684b04f","name":"GET LicenseServiceSettings/GetPublicWebSettings","count"
{"request":[{"id":"6f111e23e3b59f41","name":"OPTIONS /api/contracttypes","count":1,"responseCode":204
{"request":[{"id":"0fe98fd15914194f","name":"GET ContractTypes/GetContractTypes","count":1,"responseC
{"request":[{"id":"7eeb3a2b32cd4b43","name":"POST /connect/token","count":2,"responseCode":200,"succe
{"request":[{"id":"b1898413861f3243","name":"POST /connect/token","count":2,"responseCode":200,"succe
{"request":[{"id":"f373630057cb7b44","name":"GET /.well-known/openid-configuration","count":2,"respon
{"request":[{"id":"6799d174d5994949","name":"GET /.well-known/openid-configuration","count":2,"respon
{"request":[{"id":"71d390acc38b9943","name":"POST /connect/token","count":2,"responseCode":200,"succe
{"request":[{"id":"19747f7540186e46","name":"POST /connect/token","count":2,"responseCode":200,"succe
{"request":[{"id":"dbc1631acc512a42","name":"POST /connect/token","count":2,"responseCode":200,"succe
```

Figure 3.10: Elastic stack User Interface

In Figure 3.10 , as you can see that the blob consists of 23 total requests coming to the server. And the Logstash pipeline was reading these 23 different messages as one message rather than 23 separate messages.. But what we wanted was to split this blob into 23 distinct requests. This we achieved by splitting the Blob inside Logstash filter using “message” field.

3.2.5.2.1.3 Logstash Output

In this part of the Logstash we define where we are going to send output of the Logstash to. Below is the code snippet from the Logstash pipeline defining the output of it.

```
output {
   elasticsearch {
    hosts => ["http://x.x.x.x:9200"]
    manage_template => false
    index => "azure_event_hub"
    user => "xxxxxx"
    password => "xxxxxxxxx"
         }
    }
```

Listing 3.5: Logstash Pipeline Output

In Listing 3.5 we have defined that output would be sent to the Elasticsearch which is hosted locally, on the same machine where the Logstash is hosted, and is listening at port 9200. Second, we are de-fining that the output would be saved under the index called "Azure Event hub". This would be used by Kibana to index the Elasticsearch while visualising the data inside it. Last as the Elasticsearch is password protected we have defined what would be the user credentials to save the data inside Elastic search.

3.2.5.2.2 Kibana

In the Kibana dashboard we can index the data by searching with the index the data inside ElastiSearch. In order to view the data first we must create a visualization based on the index from the Elasticsearch database and then we are going to save change the data as per our requirements. Figure 3.11 is the result from the Kibana dashboard . where we have created a new index called "azure event hub" and visualised the data that is stored inside it.

In Figure 3.11 we have counted the total number of successful requests that were made to the server, represented in green colour, while in red we can see total number of failed requests made to the server.

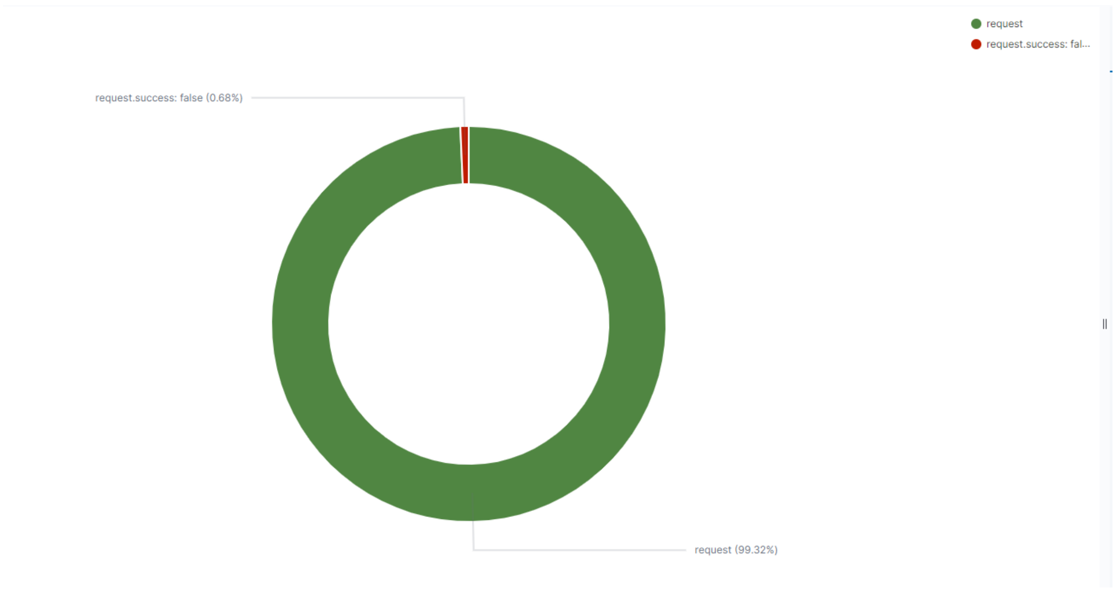


Figure 3.11: Kibana dashboard

Chapter 4

Software Versioning

In this chapter we are going to find out a version control solution for the application. As discussed , version control scheme forms a basis of any application whose source code is to be strictly monitored and numbered in order to form a tree of changes in the application life cycle.

4.1 Why do we need a Software Version?

For any software, its source code is one of the most important aspect of it as it forms the basis of the software. This source code is hosted inside source code repository like GitHub to easily manage changes to the software. This repository is a data structure that stores metadata for the set of files or directory structure. This repository keeps track of all the changes that have been made to the software and the source code itself. So that developers can keep track of the lifecycle of the software. This is achieved by using Version control system [31].

We need to version software to identify it. The developers have certain information about the software like release notes that contains what new functionality have been added to the software, what are the bugs that have been fixed from the previous release and so on. But it is difficult to identify a software based on this information. Therefore, you need a numbering convention so that it is easier to refer to the delivered software. Also, when you add a little more information when numbering the software, it becomes ore easy to recognise the state of the software. Like Alpha, Beta and Release Candidate. Adding this information tells at what stage the software is in its lifecycle or how mature it is. Also it is also important to version the software for the developers for example when users report bugs in the software, it is easier for the developers to point out which version of the software contains the bug when there are multiple versions of the software that have been released. Same is the case with our software as well, as our software consist of multiple components and there are multiple developers' teams that are working on each separate part of the software. So it becomes even more important to devise a versioning scheme where we can combine various component of the software in such a way that the final software that is deployed in the production can be identified properly by all the teams working on different parts. That is that they should know which version of their part of the software have been incorporated in the final version. So that they can keep track of the changes and the bug fixes from the previous versions. Other advantages of version control include: Conflict Resolution, Rollback and undo changes to the source code and offsite source code backup.

4.1.1 Types of Versioning Schemes

There are three different types of versioning control schemes [5]:

1. Local Version Control System (LVC)
2. Centralized Version Control System (CVC)
3. Distributed Version Control System (DVC)

Let us have a brief look at all of them

4.1.1.1 Local Version Control System

This is the simplest version Control system technique that is confined to the Local system where the source code of the software is contained. In this type of scheme, the source code is versioned as per the discretion of the people who are managing the source code. This control system can be depicted in the Figure 4.1

As can be seen in Figure 4.1, that in the Local Version Control System everything that is being on the Local system. Here every change is stored as a patch. It contains only the changes that are made since last version. So, if you want to know how the current version looks like you must add up all the previous version

4.1.1.1.1 Advantages and Disadvantages of Local Version Control System

The issue with this system is that everything is kept on the local system and if anything happens to the Local System, everything is lost. Al it is difficult to collaborate amongst

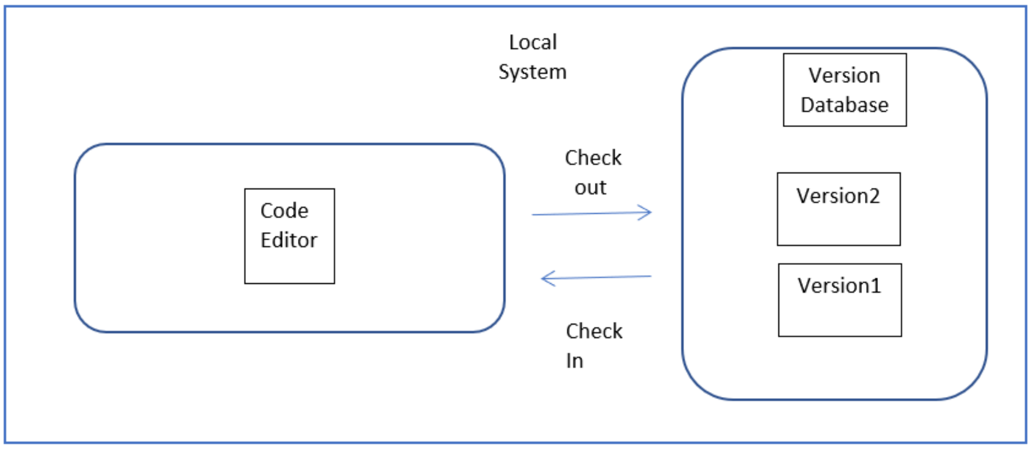


Figure 4.1: Local Version Control System [5]

the developers as about the changes that are to be made on the source code. Also, it cannot be accessed by multiple people if they want to access the code and make some changes because there is no way to synchronise changes made on the system. So, it is rarely used in the Production Systems.

4.1.1.2 Centralised Version Control System

In this form of version Control System, the source code is hosted on a centralized server. Different teams can access the system simultaneously and can make the changes because everyone can download the code and can see what changes have been made so it is easier for them to keep track of changes. This system can be shown diagrammatically in Figure 4.2

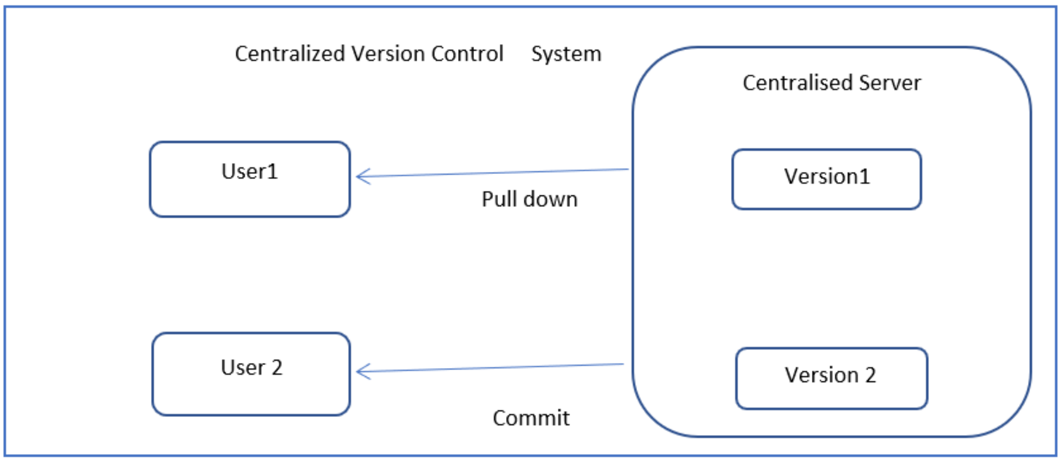


Figure 4.2: Centralised Version Control System [5]

A typical Version control system has following workflow:

- Download or pull the changes that have been made by other people represented by version on the central server
- Update /make changes to the existing code, test the changes by checking that they are working properly
- Commit the changes that you have made to the Central version control server so that other people can see them.

4.1.1.2.1 Advantages and Disadvantages of Centralized Version Control System

CVC provides following benefits:

- Enable a Central authority to control the access and privileges to the Version control System that were lacking in the LCS.
- The system is easy to set up as there are no complexities involved in it
- Changes made to the code are transparent to the other users therefore it is easier to track down who, what and what changes were made to the system.

Though the system has some advantages, but it has also some disadvantages as well. Some of them are as follow:

- If the server has no redundancy or backup enabled, then server failure can ruin the whole system.
- Changes or commits that are made remotely are slow. It has security issues as well depend-ing upon various inherent vulnerabilities in it.

4.1.1.3 Distributed Version Control System

"DVCS don't rely on Central Server. They allow developers to clone the repository and work on the version. Developers will have the entire history of the project on their own hard drives"[32]. Changes to the files are tracked between different computers that have cloned their system to the code repository. To keep all the developers to have consistent view of the code a special coordination strategy is employed. Here you can still keep all the code to the centralized Version Control System but then you can mirror the Central repository to you system giving an impression that there are various redundant copies of the Central Version database. DVCS can be represented in Figure 4.3

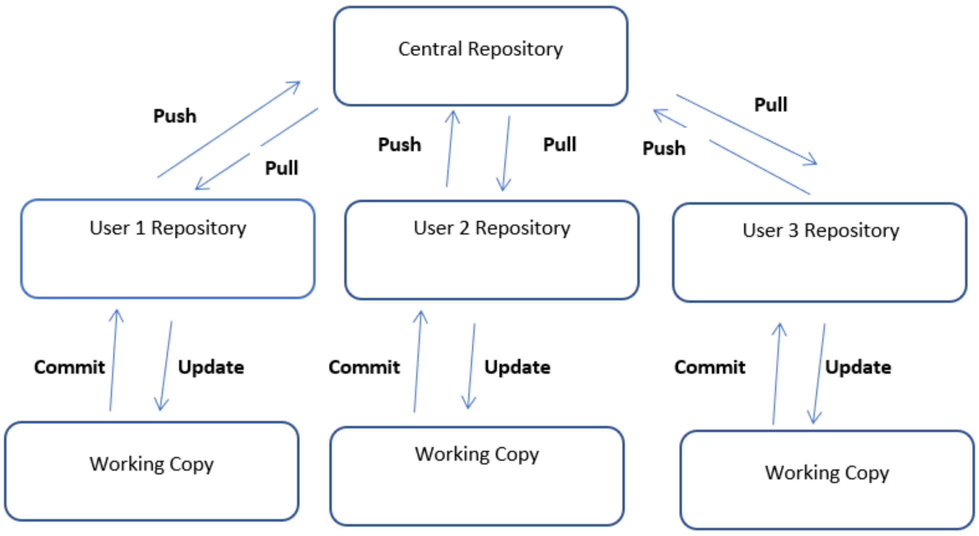


Figure 4.3: Distributed Version Control System [5]

As you can see, in the Figure 4.3,that every user has its own copy of the central server repository. First, every user downloads or clone from the central repository so that each user has the same copy of the current version or changes that are committed to the central server. This local copy is maintained at the local hard drive. At any given point of time, these local copies can be updated by either "push"or "pull" operation. With "Push" operation you upload the changes that are made to the local copies to the Central server. While in the "Pull" operation you clone with the latest version of the Central repository

4.1.1.1.3.1 Advantages and Disadvantages of Distributed Version Control System

Having discussed the functioning of the DVCS let us have a look at the advantages of this type of system. Following are the advantages of it:

- Each user has complete overview of all the changes that have been made to the system as each user have the exact copy of the repository.
- As each programmer have its own copy, committing new changes becomes easy and you don't need to worry about its repercussions to the overall system as the effects are localised to your system. Once you have tested all the changes you can push the changes back to the Central repository.
- Also, there is built in mechanism of restore with the DVCS. If the Central repository gets corrupted or went down, it can be easily recovered by one of the local repositories cloned by the users.

4.1.2 Version Control Solution

Looking at all the pros and cons of different types of version control system, it is obvious to go for the DVC. One of the tools that uses it is well known Git. Therefore, it is best to use Git to host the source code for our software. But there are certain changes that are made in the general working of it. In our case we can create three different repositories in the Git with the name of DEV, RC (Released Candidate) and Prod (Production). Each repository is further divided into their respective part of the source code software. There is a new version of the software after every successful fix of the bugs and issues that were discovered in the previous versions. This whole system can be represented in the Figure 4.4

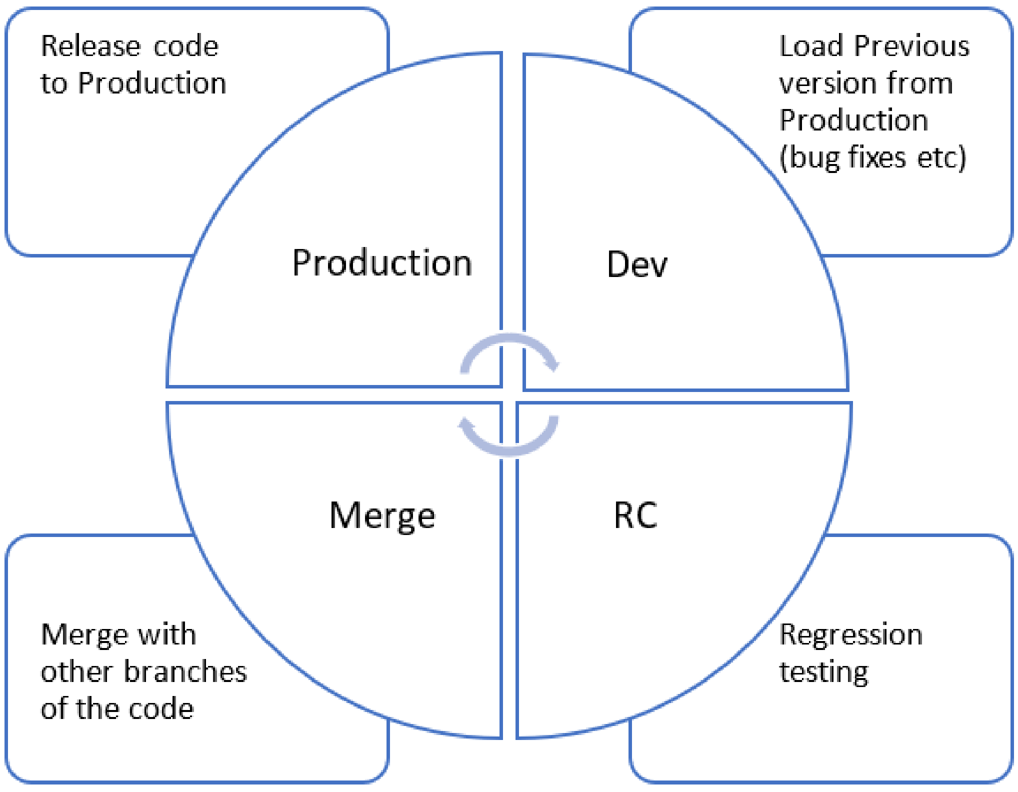


Figure 4.4: Version Control Solution

Here the system starts with the Previous version of the code which has issues or in which we want to add another functionality. The code is first put inside the DEV environment from where changes to the code happen. Here developers that are working on the various parts of the code make changes to the code by first pulling code to their respective local repositories and after making changes the code is then pushed to the DEV repository by doing Push. This process continues for a fixed duration where developers of different part of the code have made and reviewed the changes. After that period, the code is then pushed to the RC repository where the code is tested solely keeping in view all the use cases. After doing tests the all the code is merged with all the other parts of the code. The code is then ready to be versioned and ready to be deployed to the production environment. The main reason for dividing the code into different repositories is that developers can continue making changes in the DEV branch of the code while the testers can continue testing and fixing the code in the RC branch. Therefore, the developers don not have to wait for the testers to end their testing so that they can start making changes to the code. This also creates a cushion in between DEV and Production branch as the code is not directly pushed to the production because the code from DEV Repository may contain errors.

Chapter 5

Security

Application security is one of the most important concern of any organization because on it lies the success or failure of an organization. The secure the application developed by the company the more chances that they will success because the application has to gain the trust of the user that their data would be secure only then they are going to use this application. In cloud , securing the application is a shared responsibility of both the cloud providers and the companies who host their application inside the cloud.

5.1 Shared responsibilities

Once you have taken cloud services to host your application, securing application depends upon two factors: what kind of information being secured and which kind of service you have taken. Things like Information and data, devices and Accounts and Identities are always the customers responsibility and is independent upon which kind of cloud deployment model you adopt. This can be seen in the first half of the matrix. Similarly, if you are hosting on premises the responsibility of securing application falls on customer. For other services like identity and directory infrastructure, applications, network controls and operating system, the responsibility depends upon which kind of deployment model you are adopting. For example, if you are using SAS model then it is Cloud provider responsibility to secure these components. In Figure 5.1 is the Shared responsibility matrix between customer and Microsoft

As far as the security of our application is concerned, our application is being hosted inside Microsoft Azure. Therefore, the security of the application is responsibility of both us (the customer) and Microsoft. To secure our application, we are going to adopt some Security models and strategy of achieving that model.

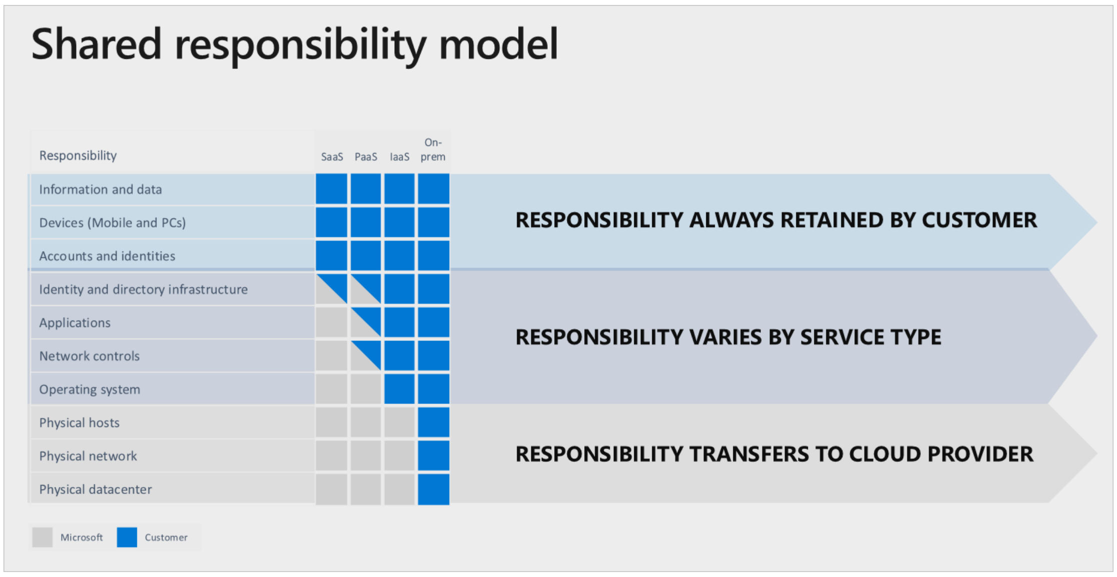


Figure 5.1: Shared Responsibilities Architecture (taken from [6])

5.2 Security Models and Strategy

With the technology being evolving, the security measures that were taken a decade ago now seems to be inadequate. Ten years ago, the main concern of the security people was to protect it from outside attacks as everything from inside was deemed to be trustful. But now with the growth of BYOD (Bring your own devices) and use of 3rd Party application has caused this security ideology to be insufficient. Now a day's security paradigm is trust nobody and authenticate every-body for them to access services. This model is called Zero Trust model. Also, one layer of security also does not fulfil the criteria of protection as different kind of information requires different type of security. Therefore, we are going to adopt a layered security approach which will not only provide us with the proper kind of security for each kind of thing that we are protecting but also it will give us what we call Defence in Depth. Let us a brief look at both and then we are going to dive into each of these models.

5.2.1 Zero Trust model

As discussed above, now the new security approach is to use Zero Trust Model. This approach was introduced by the analyst firm "Forrester Research" introduced the Zero Trust model [33]. According to this model, you should never assume trust but validate trust. This model is useful for fighting not only outside malicious attackers but also inside one as well. If we do not use this model, then we are assuming trust which can easily allow a lateral movement in privileges. This will also address the security concerns that are raised due to the BYOD because it will authenticate and verify the user before giving access to the important information and services. This model has forced security researchers, engineers, and architects to rethink the approach applied to security and use a layered strategy to protect their resources.

5.2.2 Defence in depth

It is a layered security approach that employs various security measures in layers to thwart the advancement of an attack whose aim is to get unauthorised access to information. Here each layer of security tries to prevent the security breach in a way that if one of the security layers is breach then the layer below it tries to prevent the further authorised access[34]. This Defence in Depth strategy is based upon three principles namely [35]: Confidentiality, Integrity and Availability.

• Confidentiality: With confidentiality we mean that only the intended person has access to the information. The information includes passwords, certificates, and other data of importance. This is achieved by employing different encryption techniques.

• Integrity: This principle defines that only the lawful person can make changes any unauthorised person cannot make changes to the data that is either in rest or in transit. Integrity of the data is preserved using different hashing techniques.

• Availability: Apparent from its name the main aim of this strategy is to make sure that critical services are available and protect them against different Denial of Services attack namely DOS (Denial of Services) and DDOS (Distributed Denial of Services). Availability of the services is ensured using different Denial of Services mechanisms.

5.3 Security layers

Defence in Depth, as mentioned above, is a layered approach to protect the system. Each layer having its own defence mechanism in place to protect a specific kind of service or data. This model also prevents a single point of vulnerability where the whole security structure lies within different layers where it become difficult to violate the security rules of all the layers. These layered approach model can be represented in Figure 5.2

Each layer can implement one or more of the CIA concerns We will follow bottom up approach and have a brief overview of about the nature of the target being protected and then in their respective section we are going to discuss in detail about the techniques we are going to use to protect those targets. But let us first discuss the targets first

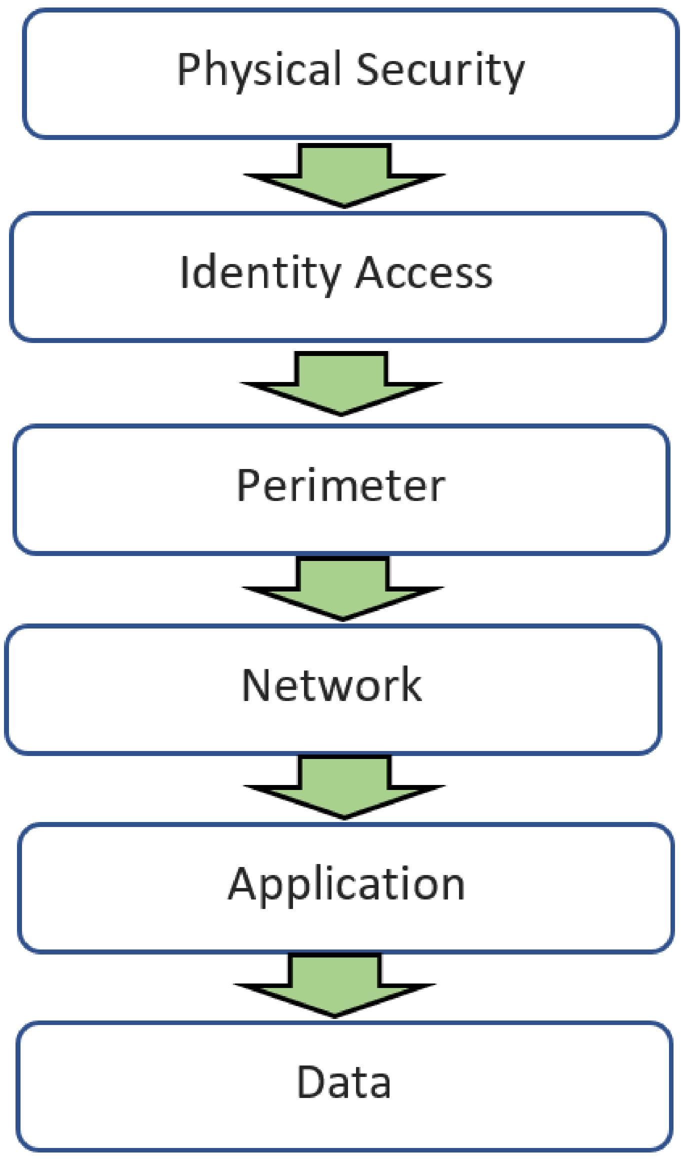


Figure 5.2: Layered Security Architecture [7]

5.3.1 Data

Data is the most important asset of an organization, especially if it is confidential data. We are going to discuss more about the sensitivity of the data and how important it is to protect the data in the upcoming section where we are going to delve deeper when we are going to discuss about the protection of the data both at rest and in transit. These data can be stored in various places inside the cloud. It can be either in the databases, or inside the virtual machines. Here the main responsibility lies on the customer about how they configure access to these resources. Here we are talking about the virtual access.

If the data is stored inside cloud, then it is cloud provider responsibility to secure the physical access to the storage.

5.3.2 Applications

With applications, we mean the piece of software. Here we are more concerned about the vulnerabilities to these codes of software inherit owing to their coding, designing and other techniques. Also, here we are talking about the sensitive secrets that are used to access these applications. Here the responsibility of the securing the application is shared between the customers and the cloud provider. If the piece of application is being taken by the customer as an SAAS then updating the application is incumbent upon cloud provider but the secrets if they are not hosted inside cloud must be protected by the customer. There are also many other considerations that must be considered while designing the security of the application.

5.3.3 Networking

Here we are going to discuss the communication themselves. We want to restrict the communication to only intended audiences. These communications involve from both internally and externally from the internet.

5.3.4 Perimeter

With perimeter protection we are talking about protecting our system form DOS and DDOS attacks. Also, we want some alerting mechanism where we want to know about these attacks so that we can design our security in such a way to curb these kinds of incidents in future.

5.3.5 Identity and Access

Identity means anything that can be authenticated to give access it can be in different forms. It can be the username; it can be biometric data or any other form of information that can be used to authenticated before giving access to the resources of the company. Not only we want to give access to only authorised person but also, we only want to give limited amount of access privileges that are required to accomplish the task.

5.3.6 Infrastructure security

Physical security means by securing the access to the datacentre.

Let us take a deep dive in each of the Security measures that are required to secure the application. As we are going to discuss different security strategies so some of them might overlap in different areas of security. Therefore, we are going to discuss the overlapping areas in pair.

Here we are not going to discuss some of the layers of the security as these are the responsibility of the cloud service provider. These layers are Perimeter and Physical security.

5.3.1 Data Security

Data either it is in the form of information or it is in the form of application code or credentials that are required to access the system, is one of the most important assets of any organization. To protect these resources, people have been using different techniques the most notable amongst them is encryption. If the malicious user gets access to the data that is stored inside the system, it is very important for the security professionals to use such techniques to preserve the confidenti-ality of the data. This encryption renders the data useless for the intruders if they are unable to decrypt the data.

5.3.1.1 What is encryption?

“Encryption is the process of converting data to an unrecognizable or "encrypted" form. It is commonly used to protect sensitive information so that only authorized parties can view it. This includes files and storage devices, as well as data transferred over wireless networks and the Internet"[36]

5.3.1.2 Types of Encryption

There are two types of encryption [37]:

a. Symmetric Encryption

b. Asymmetric Encryption

a. **Symmetric Encryption** In asymmetric encryption we use the same key both encrypt and decrypt the data. Here the focus is how to protect and share the symmetric key to both the sender and receiver so that it remains confidential between the parties. Here first the data is encrypted to unrecognisable form using the symmetric key and is then sent to the recipient. The recipient uses the same key to decrypt the message. The key that is used to encrypt or decrypt the message can be a secret password or it can be a random string of letters or numbers generated by Random Number Generator (RNG). Some of the algorithms that are based on the symmetric encryption techniques are Advanced Encryption Algorithm (AES), Data Encryption Algorithm (DES)

b. **Asymmetric Encryption** In asymmetric algorithm, there are a pair of keys are used to encrypt and decrypt the message. These pair of keys are called public and private key. Either of these keys can be used to encrypt or decrypt the message. Usually public key is used to encrypt the message while the private key is used for decryption purposes. Asymmetric encryption is used for things like TLS (used in HTTPS) and data signing. Here when you visit any website that uses HTTPS or TLS, an asymmetric connection is established with the website. The public key is automatically detected by the browser that is installed on the certificate installed on the website. The user browser uses this key to encrypt the connection towards the website. Some of the algorithms that are built on asymmetric encryption principle are Diffie Hellman, RSA (Rivest–Shamir–Adleman) algorithms.

5.3.1.3 Encryption Basis in Cloud

Encryption in azure is based on two different state of the information. I-e whether the information is at rest or is it in transit

5.3.1.3.1 Encryption at rest

Data at rest can be residing in any of the storage services provided by the cloud. The data can be either in the hard drives of the VM, it can be within the databases services provided by the cloud or any other storage medium. The data stored must be in the encrypted form so that even if the malicious actor gets the access of the hard drive or the databases , he cannot make any sense of the data without having the key that is used to encrypt the data. In these situations, the attacker must use different combinations of passwords until it gets the right password to decrypt the data. This process is very resource intensive and can take many decades. The strength of the encryption technique is dependent upon the length of the key that is used to encrypt the data. Longer the key, longer it takes to find the key

5.3.1.3.2 Encryption in Transit

Not only the data that is at rest needs to be protected but also the data that is moving must also be protected. The data in transit can be protected in two ways:

- Create a secure tunnel within the public internet or create a dedicated connection between the sender and the receiver

- Or encrypt the data that is to be sent across the internet Here the main aim of the encryption is to protect it from any unintended observer of the data.

5.3.1.4 Encryption Services in Microsoft Azure

As discussed above the data can be stored in any storage medium or service provided by the Microsoft. It can be either in the Virtual machine hard disk, in database or any other storage medium. Microsoft Azure provides different kinds of encryption techniques depending upon where the data is stored in the cloud. Let us see each of them. We will be discussing two things here, the cloud service and the encryption technique being used in it and the benefits of using them.

5.3.1.4.1 Encrypting Raw Data Inside Azure

Data can be any form stored in the cloud . It can be stored as a table,image, file or any other form. The data can have a specific type or it can be type less. The data that has no specif type is called raw data. Microsoft Azure provides services where you can store raw data.

5.3.1.4.1.1 Cloud Services

The services here we are talking are the Azure storage services including Azure Managed Disks, Azure Blob Storage, Azure Files, Azure Queue Storage, and Azure Table Storage

5.3.1.4.1.2 Encryption Techniques

The Azure Storage platform encrypts data with 256-bit Advanced Encryption Standard (AES) encryption before persisting it to disk and then decrypts the data during retrieval. This handling of encryption, encryption at rest, decryption, and key management in Azure Storage is transparent to applications that use the service.

5.3.1.4.1.3 Benefits

For any organization, Azure Storage encryption means that whenever someone is using services that support Azure Storage encryption, their data is encrypted on the physical medium of storage. In the unlikely event that someone gets access to the physical disk, the data will be unreadable.

5.3.1.5 Encrypting Data inside Virtual machines

The data can be stored independently or it can be a part of any PAAS (Platform As A Service) like virtual machine. Microsoft Azure also has a service to encrypt data that is inside virtual machine.

5.3.1.5.1 Cloud Services

Apparent from its name, the service we are talking about is the Virtual machine service being offered by the Microsoft Azure. Here the data that is being encrypted is the data that is being written to the physical disk attached to the virtual machine.

5.3.1.5.2 Encryption Techniques

Microsoft Azure uses Azure Disk Encryption (ADE) to encrypt the data that is stored inside the Vir-tual Hard Drives (VHD). ADE uses different encryption standards for both Windows and Linux ma-chines. For Window machines, it uses standard BitLocker feature of Windows and the DM-Crypt feature of Linux to provide volume encryption for the OS and data disks. This solution is integrated with the Azure Key Vault (that is used to store the secrets and passwords.)

5.3.1.5.3 Benefits

If a malicious attacker gained access to your Azure subscription and ex filtrated the VHDs of your virtual machines he cannot read the data from it until it has the keys that are stored inside Azure Key vault to decrypt the data. Here, one thing is worth noting is that you can set up the disaster recovery inside Microsoft Azure to any region available inside Azure so that if the virtual machine fails inside one region due to an outage in that region , it can fail over to the secondary region. [38].This must be setup separately

5.3.1.6 Encrypting databases

The data can also be stored inside databases.Microsoft has a service to encrypt data stored inside databases.

5.3.1.6.1 Cloud Services

Here the focus is on the Databases services provided by the Microsoft Azure. There are a vast range of database services provided by Microsoft Azure. These services[39] are :

- Azure SQL Database
- Azure SQL Managed Instance
- SQL Server on Virtual Machines
- Azure Database or PostgreSQL
- Azure Database for MariaDB
- Azure Cosmos DB
- Azure Cache for Redis
- Azure Managed Instance for Apache Cassandra

5.3.1.6.2 Encryption Techniques

Azure uses Transparent data encryption (TDE) to encrypt the data that is stored in its database services. TDE uses Symmetric encryption methodology to encrypt the data. “Transparent data encryption encrypts the storage of an entire database by using a symmetric key called the database encryption key. By default, Azure provides a unique encryption key per logical SQL Server instance and handles all the details. Bring your own key is also supported with keys stored in Azure Key Vault. Transparent data encryption helps protect Azure SQL Database and Azure Data Warehouse against the threat of malicious activity. It performs real-time encryption and decryption of the database, associated backups, and transaction log files at rest without requiring changes to the application. By default, transparent data encryption is enabled for all newly deployed Azure SQL databases”.[40]

5.3.1.6.3 Benefits

With TDE you can protect the information that is stored inside the databases. This information that is stored inside databases can be of confidential nature like security numbers, data of birth, or personal identities. This feature helps protect column data at rest and in transit by having the client application handle the encryption and decryption outside the SQL Server database through an installed driver. This allows your organization to minimize exposure of data, because the database never works with unencrypted data. "Here as well by default (depending upon the kind of database service you are taking), the database is backed up for 7 days for basic accounts. This can be extended to 35 days for the standard accounts. This backup service is called "Point in Time Retention. The databases can be backed up for longer period (Long Time retention) up to 10 years".[41]

5.3.1.7 Encrypting secrets (Azure key vault)

To use all the encryption services, we require keys that will be used to encrypt or decrypt the data. These keys form integral part of any encryption techniques. If these keys are compromised, whole encryption standards can be easily violated. So, it is of utmost importance to protect these keys. Microsoft Azure provides a service called Azure key vault to store and protect these keys. "Azure Key Vault is a secret store: a centralized cloud service for storing app secrets - configuration values like passwords and connection strings that must remain secure at all times. Key Vault helps you control your apps' secrets by keeping them in a single central location and providing secure access, permissions control, and access logging" [42]. Azure secret key vault gives the ability to store your password in such a way where that you can configure who can access these secrets and what actions or privileges they can perform on stored secrets. These accesses can be set, list, and get depending upon what the agent or an application wants to do on the application. Azure Key Vault features multiple layers of redundancy to make sure that your keys and secrets remain available to your application even if individual components of the service fail. he contents of your key vault are replicated within the region and to a secondary region at least 150 miles away, but within the same geography to maintain high durability of your keys and secrets. If individual components within the key vault service fail, alternate components within the region step in to serve your request to make sure that there is no degradation of functionality. You don't need to take any action to start this process, it happens automatically and will be transparent to you. In the rare event that an entire Azure region is unavailable, the requests that you make of Azure Key Vault in that region are automatically routed (failed over) to a secondary region "[43]

5.3.2 DevSecOps

Hosting application inside the cloud gives you one edge that you do not have to worry about the physical security of the equipment where your services are being hosted because it is the responsibility of the cloud provider. The measures taken by the cloud are so stringent that any attack which is aimed to compromise the platform will see no results. Then the only way to attack is to attack vulnerabilities introduced due to application architecture. Therefore, it is important to create application in keeping view security of the application. To inject security measures within the application, there is a lifecycle model provided by Microsoft called Microsoft Security Development Lifecycle (SDL) or DevSecOps that does that. SDL ensures that security is incorporated within the application while it is being developed. If the application is designed fol-lowing this model, then it is easy to cope up with the security concerns that appear afterwards. A software project can use this typical sequence of SDL steps in Figure 5.3

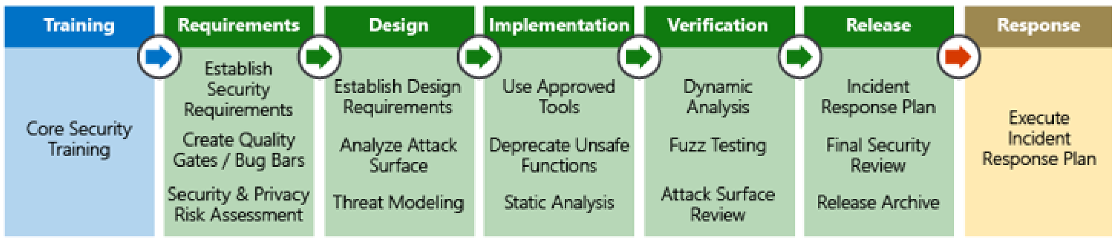


Figure 5.3: Microsoft Development Life cycle(taken from[8])

Following this process, the organization assure itself hat the security is of the main concern of any organization and this will assure that application is developed in such a way that leaves less space for the attacker.

5.3.3 Network Security

Securing your network from attacks and unauthorized access is an important part of any architecture

5.3.3.1 Layered approach to network security

Here we are going to follow a layered approach to protect the network by layered approach we mean,

- Reducing the application surface area outside so that we can reduce the surface area of the external attacks. This is what we call Internet protection.
- Reducing the communication within the application only to those parts where it is necessary and avoiding any unnecessary communication within the application

- Providing least amount of privileges wherever are possible so that ono one should have more access than what is required.

Let us first start with the first layer and that is protecting the architecture from outside attacks.

5.3.3.1.1 External protection

We will divide Internet Protection further into two further layers:

- Layer 3 (OSI (Open System Interconnect) Transport Layer) and Layer 4 (Network Layer)
- Layer 7 (Application layer)

5.3.3.1.1.1 Layer 3 and Layer 4 Protection

Here we are speaking about protecting the application from external threats or attacks from the internet. To protect the application, you to do the following things for layer 3 and layer 4 of the OSI model:

- Which parts of the applications are facing the internet?
- Is communication is coming from public IP or coming from within the organization?
- Which ports should be opened to have successful communication?

  After finding the answer of these questions we are going to do the following things
- Restrict the communication to only those parts of the application which is required for the application to communicate externally or to the internet
- If the application is not created for audiences outside to the general community but is for the organization itself. Try to reduce the range of Ip addresses to the one which are reserved for the application
- Shut down all the unused opened ports that are not required for the communication

This reduction in surface area is achieved by using VNET Security groups. This would be explained in detail later in the document.

5.3.3.1.1.2 Layer 7 Protection

For Application layer, there are other kinds of attacks that are more sophisticated than the layer 3 and 4 attacks. For this layer, you need a comprehensive security solution that keeps on evolving with the advancement of the kind of attacks that are made at the application layer. Microsoft Azure Provides Layer 7 protection to the application that are deployed within the azure or outside it is using Azure Application Gateway. "Azure Application Gateway is a Layer 7 load balancer that also includes a web application firewall (WAF) to provide advanced security for your HTTP-based services. The WAF is based on rules from the OWASP 3.0 or 2.2.9 core rule sets. It provides protection from commonly known vulnerabilities such as cross-site scripting and SQL injection" [44].

In the diagram given below hat the application firewall inspects every request coming from outside internet and based on the rules set inside the firewall allow or drops the request. At the same time, it also works as a load balancer, balancing traffic to multiple instances of the application. One worth noting thing about this firewall is that it updates itself with the latest kind of layer 7 threats knowledge and how to detect them. Therefore, there is no need to worry about the lat-est form of threats. The discussed layer 3, layer 4 and layer 7 security measures, if taken, can protect the deployed application from

- Internet attacks by reducing the surface area of application by limiting the internet access to only those IP addresses and ports and to only those part of the application where they are required

- It also inspects all the internet traffic that is coming towards the application from the internet and finds out the layer 7 attacks such as Cross site scripting and SQL injection and other layer 7 attacks.Figure 5.4 shows this scenario.

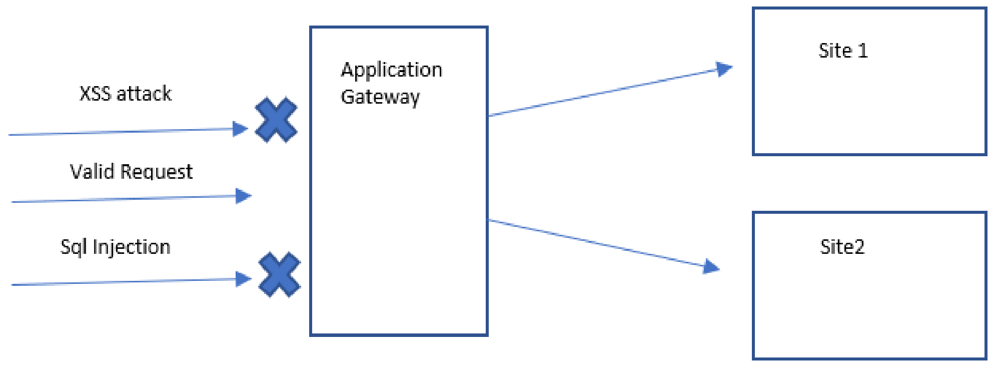


Figure 5.4: DOS Attack Prevention

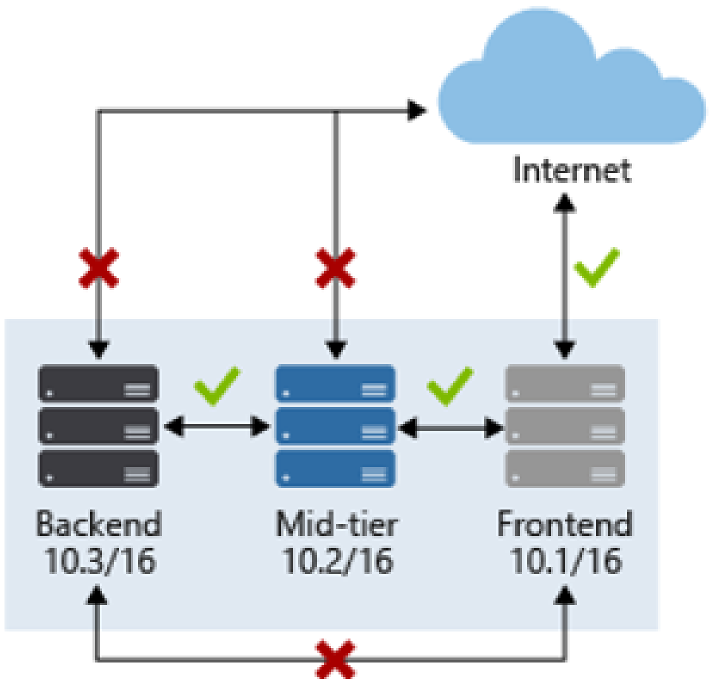


Figure 5.5: Internal Communication Protection (taken from[9])

5.3.3.1.2 Eliminating the unnecessary Communication within the Application

In addition to curb external threats, it is also important to eliminate the unnecessary communication within the application because these can lead to vulnerabilities that can be a security threat. In Microsoft Azure, application and its components are deployed within the VNET (Virtual Networks) and these VNET have built in capabilities through which we can select and allow the communication that we want between different application components. These reduction in communication is achieved by using Network Security groups. Network security groups operate at layers 3 and 4. It is a kind of firewall that is the combination of allow and deny statements which allows which IP address and which ports are allowed to and from internet and within the application different components . Therefore, it provides both internal and external layer 3 and 4 protection to the application deployed inside Microsoft Azure. Figure 5.5 depicts how a surface area of communication is being restricted to only frontend of the application using NSG allow and deny capabilities. Along with it, it is also reducing the communication within the application by only allowing necessary communication for example (Backend to Middle tier to front End) and denying unnecessary communication (Backened to Frontend tier) between components of the application

5.3.5 Identity and Access Management

Last but not the least the identities that form an integral part of the organisation must also be secured. These identities form the basis of any security architecture of an organization because they are the key to get hold of the resources provided by the organization. For any organization, following are the main concerns regarding the identities:

- Use state of the art authentication mechanism for the identities
- Allow the identities to be managed by the employees to reduce the overhead.

These and many other questions are addressed by the Azure Active Directory (AAD), which is an identity management service Microsoft Azure. Let us see which services are provided by AAD that can enhance Security of the application.

5.3.5.1 Single sign-on (SSO)

Sign-on is one of the important services provided by the AAD [45]. It takes the burden from the user to manage multiple identities and integrate them in one and then use it for multiple applications or services simultaneously. This service is very important in terms of security because with the pas-sage of time the complexity and length of the password for the identities are changed such that it is becoming difficult for the user to manage them. Also, if at the same time the user must use to log in to multiple applications then they must create multiple identities at the same time for them. AAD provides a service where you use one identity to be used for multiple application. Also, with the AAD, you can manage the identities in such a way that if an employee leaves the organization, there identity is automatically deleted

5.3.5.2 Authentication and access

Another important service that is provided by the AAD is Multifactor Authentication and Conditional Access policies. With, capabilities, an organization can set policies regarding the sign in process. Let us have a look at these capabilities one by one.

5.3.5.3 Conditional access policies

In addition to Multifactor authentication, AAD also provides the Condition access [46] before granting the access based on other factors. For example, if the login attempt is

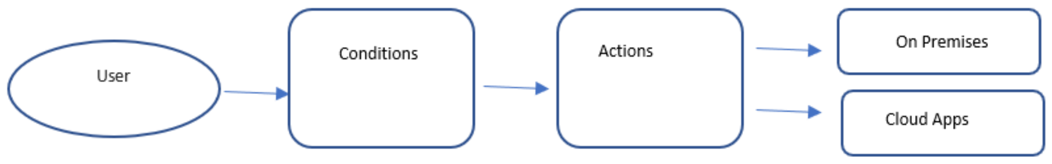


Figure 5.6: Conditional Access

being made by the unknown IP address or some unknown location or denying access from devices without malware protection can be blocked by AAD conditional access. With this you can provide any additional layer of security along with the multifactor authentication. This conditional access can be explained with the example , suppose an employee was logged in from west Europe in morning and suddenly in the evening the login attempt is made from North Africa or any other part of the world that is too far away from the first place of login. This can be thought of as a malicious login attempt and therefore must be blocked. Therefore, with the help of Condition Access provided by the AAD an organization can create such conditional access policies that can further decrease the chances of giving a malicious identity access to the company resources. This conditional policy and its implementation can be depicted in Figure 5.6 .

5.3.5.4 Multifactor authentication

"MFA, sometimes referred to as two-factor authentication or 2FA, is a security enhancement that allows you to present two pieces of evidence – your credentials – when logging in to an account. Your credentials fall into any of these three categories: something you know (like a password or PIN), something you have (like a smart card), or something you are (like your fingerprint). Your credentials must come from two different categories to enhance security – so entering two different passwords would not be considered multi-factor ". [47].

As mentioned before that the security architecture that we are going to discuss is a proposed solution for securing the application inside Microsoft Azure. Taking leverage of all the security technologies at different layers of security can help us protecting the application not only from external threats but also from internal one. But we need to implement these suggested security solution and test them with different peneteration testing tools to judge there strength.

Chapter 6

Automation (Application Deployment)

In this chapter of the thesis, we are going to research and find out an automation solution for the application where all the steps from the development to the production are automated in such a way in order to reduce the human intervention and prevent any errors that can occur during this whole process.This is achieved using CI and CD tools.

6.1 Continuous Integration (CI)

“Continuous integration is a coding philosophy and set of practices that drive development teams to implement small changes and check in code to version control repositories frequently. Because most modern applications require developing code in different platforms and tools, the team needs a mechanism to integrate and validate its changes”.[48]The main focus of CI is to translate the code changes that have been made inside the application to be build and packaged in such a way that it can be used by the CD for the deployment on the target machines. As discussed, this process must be consistent and automated to save time and reduce errors.

6.1.1 Importance of CI

CI is very important where multiple teams of developers are working at the same time making different changes to the application code. Without the CI tool, the developers must communicate with each other what changes they are making to introduce new features in the application or fix previous issues introduced in the application. This manual communication is not restricted to only these developers' team, but it also extends to rest of the organization as well. This overhead of the communication not only slows down the application production process but if there is some miscommunication between the developers can also introduce errors in the application and these anomalies and longer delivery time can increase with the increase in the code base size and developers team. With the introduction of CI in the application deployment can help tackle all these challenges. It can help developers to work independently on different features of the application until they are ready to be merged in the form of product

6.2 Continuous Delivery (CD)

"Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way" [48] . As apparent from its name, it takes the packages that are built by the CI, it deploys or delivers them to the target environments. These target environments can be Development, Test or Production environments. The main goal of CD is to the code is always in the deployable to these target environments. With CD you get low risk in deploying the software, swift time to the production by avoiding phases in traditional deployments model like integration and test phases that can consume a lot of time. Higher quality of the software because the deployment of the software now is an automated process. Lower costs as the time spent by the teams is reduced on how to deploy which can then be reverted on to focusing on improving software quality.

6.3 CI CD Process

Having explained CI CD, it is now time to see how we can we use different tools that we can use to set up a CI CD pipeline. For this we have used following tool to set up a CI CD in our scenario.

1. Git Hub (Code and Version control Repository)

2. Jenkins (CI tool)

3. Octopus (CD tool)

We have already explained about GitHub in the version control section, now we are going to discuss Jenkins and Octopus and then proceed to explain the CI CD pipeline process

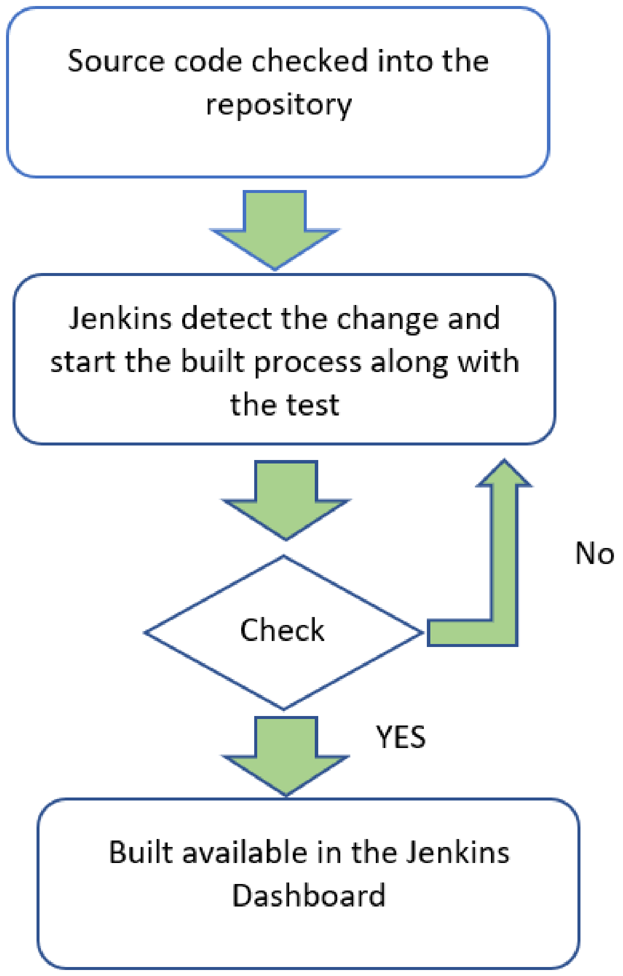

Source code checked into the repository
Jenkins detect the change and start the built process along with the test
Check
No
YES
Built available in the Jenkins Dashboard

Figure 6.1: Jenkins build process

6.3.1 Jenkins (CI tool)

“Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with several testing and deployment technologies. In this tutorial, we would explain how you can use Jenkins to build and test your software projects continuously.”[49] Here we install Jenkins on a server where the code from the source code repository is built. This built process can be accompanied with various test based on which Jenkins can decide whether the built process is successful or not. Successful built software is available in the Jenkins dashboard to be used by the CD tool for the deployment purposes. This process can be shown diagrammatically in Figure 6.1.

Once the code is built successfully the artifacts are available in the Jenkins dashboard to be used for the deployment.

6.3.2 Octopus (CD)

"Octopus Deploy is a deployment tool. It takes the packages and artifacts generated by your build server and deploys them to various targets, be it Windows, Linux, Azure, AWS, or Kubernetes, in a safe and consistent process." [50] With Octopus Server you do the following three things:

1. Define the Deployment process for the target environments
2. Configure environment variables that are going to be used for the deployment purposes
3. Deploy to the target environment

6.3.2.1 Define the Deployment process for the target environments

In the first instance you define how the software is going to be deployed on the target environment. These are high level steps that are run to perform the deployment. There are already more than 450 different templates present inside the Octopus that can form a starting point to define your deployment steps. These steps can also create alerts and notifications for humans to intervene and make any changes to the process if required.

6.3.2.2 Configure environment variables that are going to be used for the deployment purposes

In this step you define different variables that are going to be used in the deployment process. These variables include password or secrets etc. These variables can also be shared across different target environments, or even to individual targets.

6.3.2.3 Deploy to the target environment

In the third and last step Octopus uses the Deployment process and the environment variables defined in the previous steps to deploy the software to the target environments. Octopus deployment can also be viewed

6.3.3 CI CD Pipeline

In our case we have defined the CI CD pipeline process we have used the above three technologies and have created a pipeline. We will discuss the stages of the pipeline one by one.

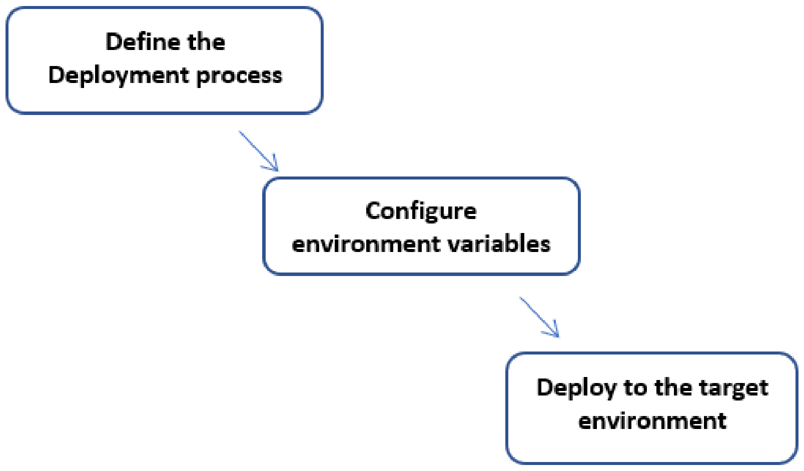

Define the Deployment process
Configure environment variables
Deploy to the target environment

Figure 6.2: Octopus process

6.3.4 Setup

We have set up following things for this automation to work 1. Jenkins Server (where Jenkins pipelines are going to run)

2. Azure Blob Storage (where build artifacts are going to be stored produced by Jenkins)

3. Octopus Server (where the artifacts downloaded from the azure blob storage will be delivered to the target environment)

We are not going to explain the setup as we have followed the documentation from the respective technologies to set them up rather we are going to explain the pipeline itself which is used .

6.3.5 Pipeline

Let us explain different steps that are used in the pipeline one by one. Each Pipeline consist of different stages with each stage itself consist of different steps.

6.3.5.1 Prepare Stage

In Listing 6.1 we are going to create different variables along with the different URL that are going to be used during the pipeline . These URL include a URL from where the code is going to be down-loaded, azure storage URL etc. Following is a code snippet of this stage. Keywords such as pipeline , stages tells us where we are . Like the keyword “pipeline ” tells that we are defining a pipeline.

```

// Define variables that are going to be used in the pipeline

pipeline
 {
     //Define agents where the code will be deployed or else you can
    store the code
options {
      disableConcurrentBuilds() //allow only one build
      timestamps()
    }
stages {
    stage('Prepare') {// prepare build steps
        steps {
          script {
            YARN_BUILD = "build.${BUILD_ENV}"


            //Define variables for use later
            UTIL_ARCHIVE = load "${pwd()}\\Pipelines\\Utility\\ "
            CODE_DIR = "${pwd()}\\thecode"
            OUT_FILE = "dist.zip"
            UPLOAD_DIR = "thecode/dist/**"
         ENV_PATH = "set PATH=%PATH%;%WORKSPACE%\\node_modules\\.bin"
                  ENV_NODE = "set NODE_PATH=%WORKSPACE%\\node_modules"
            DOWNLOAD_CHECK = true
          }
        }
      }
```

Listing 6.1: Logstash Pipeline Prepare Stage

6.3.5.2 Download Archive stage

In Listing 6.2 stage we are going to download any archive stored inside the azure so that if there is none we can store one there or we can update an existing one. This step make the build process consistent with the previous build versions of the code as it checks for the previous build versions and updates them keeping in view the changes that have been made to the code.

```

stage('Download Archive') {
    steps {
      script {
        try {
```

```
        //Always build (in case we need a different look)
      DOWNLOAD_CHECK = false
  // UTIL_ARCHIVE.downloadArchiveFromAzure('App-' + SCM_INFO.GIT_COMMIT,
    'azurestorage', "${pwd()}", true)
  // unzip zipFile: 'archive.zip', dir: "${pwd()}\\theCode", quiet: true
            }
    catch (Exception e) { DOWNLOAD_CHECK = false }
          }
        }
      }
```

Listing 6.2: Logstash Pipeline Archive stage

6.3.5.3 Build Stage

In Listing 6.3 we are going to build the code downloaded from the GIT repository. It first checks that whether the code that has been downloaded has already build or not . If it is not build , this stage builds the code and proceed to the next stage of the pipeline .If the code has already been build then it downloads that build artifact and it skips this step. The build artifacts are not yet ready because they needed to be in proper format to be saved inside the azure.

```

stage('YarnBuild') {
  steps {
     script {
       dir(CODE_DIR) {
     if (!DOWNLOAD_CHECK) {
      withEnv(([ENV_PATH, ENV_NODE])) {
                       bat "yarn ${YARN_BUILD}"
                }
              }
    else echo "Downloaded archive and skipping Build stage"
            }
          }
        }
```

Listing 6.3: Logstash Pipeline Build stage

6.3.5.4 Compression Stage

In Listing 6.4 stage we are going to zip the result so that they can be stored in the azure blob storage. Once in the previous stage, either the code is built or the artifact is downloaded the next step is to compress it so that it can be stored in a storage account where it can be used by the CD tool to pick and deploy to the target environment. This stage has its significance as it compresses the evergrowing code in order to save the memory.

```

stage('ZipResult') {
  steps {
    script {
      if (!DOWNLOAD_CHECK) {
        dir(CODE_DIR) {
        withEnv(([ENV_PATH, ENV_NODE])) {
              bat "yarn zip-dist"
                }
              }
            }
    else echo "Downloaded arhive and skipping ZipResult stage"
          }
        }
      }
```

Listing 6.4: Logstash Pipeline build compression stage

6.3.5.5 Upload Stage

In this stage we are going to upload the zip file inside the azure blob storage.From the previous step of the pipeline , we have a compressed build artifact. Now this artifact must be placed in a place from where the Octopus server can pick it and deploy them to the target environment. As the application is hosted inside Microsoft Azure, it was natural to look for the storage services provided by it. Therefore, we have chose azure blob storage service that is not only cheap but is easy to use as well as you only have to configure the access key inside your code to access the storage account.

```

stage('Upload Archive') {
  steps {
    script {
    if (!DOWNLOAD_CHECK) {
        dir(CODE_DIR) {
      try {
  UTIL_ARCHIVE.uploadZipArchiveToAzure('AppFrontend-' + SCM_INFO.
    GIT_COMMIT + YARN_BUILD.replace(".","-"), "azure storage")
                }
        catch (Exception e) {e.getMessage()}
              }
```

```
            }
    else echo "Skipping upload of archive"
```

Listing 6.5: Logstash Pipeline build compression stage

6.3.6 Integrate Octopus Server with Azure Blob Storage

In the final step, we have integrated Octopus Server with the Azure Blob Storage so that Octopus server can download the artifacts that were built by the Jenkins and stored inside Azure Blob storage. Here we require an authentication and authorization solution where Octopus Server can authenticate itself to Microsoft Storage to access the blob storage contents. For this we have used the string that is stored inside the blob storage to access the account. Once Octopus Server has access to the blob storage account, it can download the artifacts and use them deploy them to the target environments. The whole Pipeline in Figure 6.3.

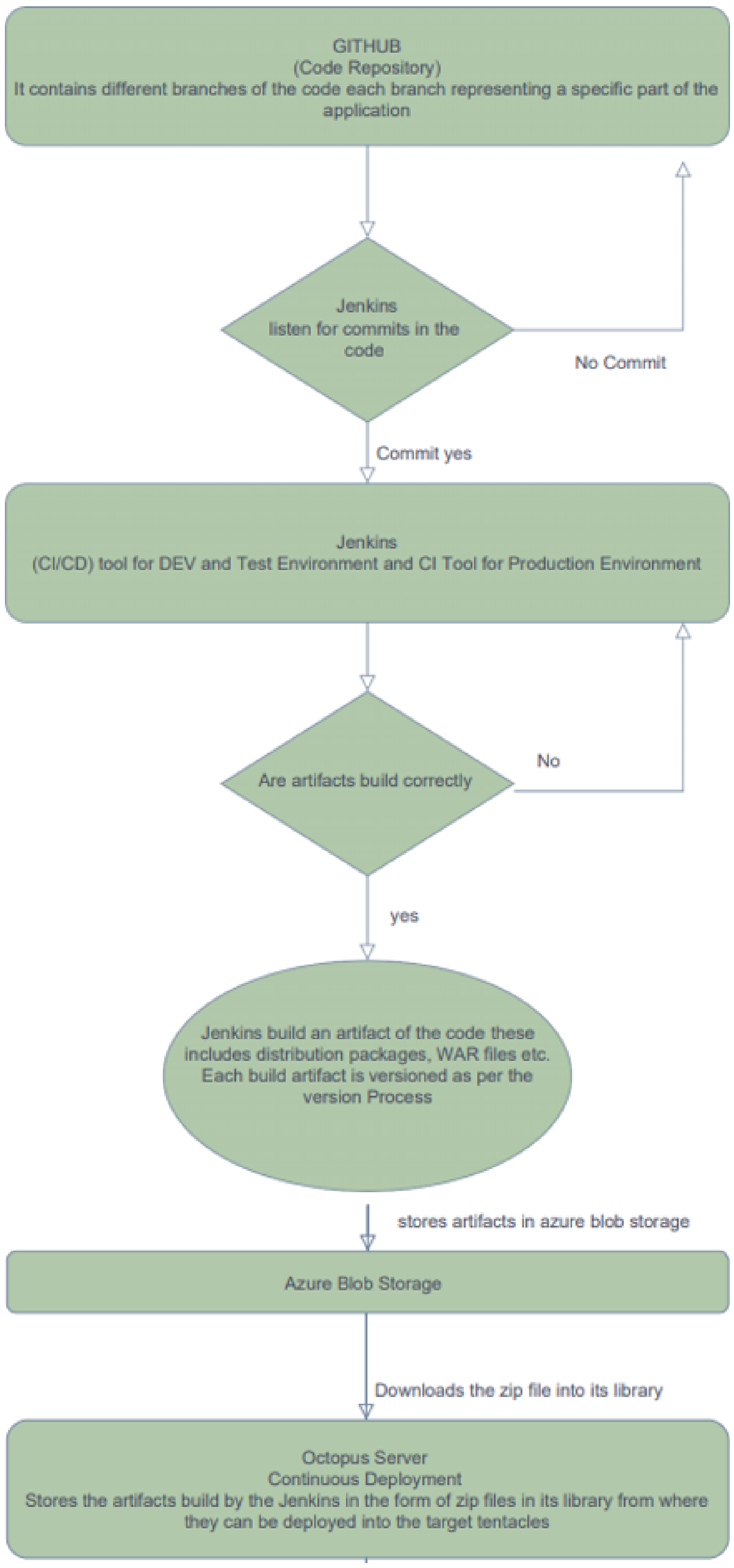

GITHUB
(Code Repository)
It contains different branches of the code each branch representing a specific part of the application
Jenkins
listen for commits in the code
No Commit
Commit yes
Jenkins
(CI/CD) tool for DEV and Test Environment and CI Tool for Production Environment
Are artifacts build correctly
No
yes
Jenkins build an artifact of the code these includes distribution packages, WAR files etc. Each build artifact is versioned as per the version Process
stores artifacts in azure blob storage
Azure Blob Storage
Downloads the zip file into its library
Octopus Server
Continuous Deployment
Stores the artifacts build by the Jenkins in the form of zip files in its library from where they can be deployed into the target tentacles

Figure 6.3: CI CD pipeline

Chapter 7

Conclusion and Future Work

In this chapter we are going to conclude our thesis with Research Questions and their answers. We will then discuss possibilities of future work within this thesis.

7.1 Conclusion

In this thesis we have worked on four different aspects of an application: Monitor application status, Its versioning mechanism, its security and its continuous integration and continuous delivery. First, we have developed a solution to Monitoring application, keeping in view its monitoring requirements. Next, we went on to provide a Versioning issue. As the application is huge and it contains a lot of parts therefore there is no such mechanism to version it properly to be used in the production environment. Next, we have proposed a Layered Security solution for the application, using different technologies that are being offered by the Microsoft Azure. In the final phase of the thesis, we have automated a solution with CI CD tools.Also let us have a brief overview of research questions and the result of the research on those questions.

Research Questions1:

Is it possible to setup a monitoring solution outside the cloud environment?

We have developed a monitoring architecture keeping in view of all the requirements. Presently, we were able to transfer monitoring data of different components of the application. Also the solution is cost effective as we have used open source elastic stack package which if free of cost.The data is also stored inside the elastic search database and can be used in future for further analysis purposes.

Research Questions2:

Can we Develop a Version Control System for the Application that is consistent with the needs of the organization ?

We have developed a version control system for the source code of the application using Distributed Version Control System. Here each software release of the code is versioned so that it is easier for the developers to identify the software that has been delivered.

Research Questions3:

Research on the security possibilities inside the hosted environment of application and propose a security architecture

We have proposed a Securing the different assets of the the application using different services provided by the cloud. We have used a layered approach where each component of the application will be protected by a specific service. This approach not only protects the application from malicious attacks but also segregates the security solution.

Research Questions4:

Can we automate the deployment process of the application to the target environment?

We have made a CI CD process for the application using CI CD tools to automate the deployment of the application to the target environment.

7.2 Future Work

Though we have proposed different solutions of the issues that the current application setup had but the as the application always evolve therefore there are always new things that are discovered and needs to be fixed. Here we would like to mention few things as far as the future work is concerned. First, for the monitoring of application, though we were able to export logs of the application to monitoring tool, but we have not carried out any analysis of the data. We can use different data analytic and machine learning techniques to analyse and visualise the data to extract meaningful information from it. Second, for the versioning of the application is concerned, the needs to version of the application keeps on changing therefore we must keep on changing the versioning system as well. For the security of the application, it is a huge topic. There is a still a need to prepare and deploy the proposed architecture and see how it performs. One test that can be done is that we can do, is to use different intrusion techniques to check the resilience of the security architecture. Last but not the least the tools that we have used to automate the deployment of the application like Jenkins. Here we have used it only for the CI purposes. But it can also be used for the CD. We have avoided it and have selected Octopus for the sake of simplicity. We can use Jenkins for both CI and CD and test it performance with the CI CD architecture that we have proposed.

Bibliography

[1] ApplicationMonitoring,What is Application Performance Monitoring,(Accessed 02 Feb,2021). URL https://www.aternity.com/faq/what-is-application-monitoring/).

[2] Application Insights Architecture (Accessed 11 Mar,2021. URL https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview.

[3] Elastic Stack ,The Elastic Stack,(Accessed 04 Feb,2021). URL https://www.elastic.co/elastic-stack/.

[4] ElasticStack Diagram (Accessed 05 Mar,2021, . URL https://www.elastic.co/guide/index.html.

[5] Types of Version Control , (Accessed 18 Feb,2021). URL https://serengetitech.com/tech/introduction-to-git-and-types-of-version-control-systems/.

[6] Azure Shared Responsibility Model (Accessed 23 Mar,2021). URL ImageSource:https://docs.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility.

[7] LayeredSecurityArchitecturel,Introduction to Azure Security,(Accessed 7 Feb,2021).

[8] Azure appsecurity lifecycle (Accessed 05 Apr,2021), . URL https://docs.microsoft.com/en-us/learn/modules/azure-well-architected-security/7-application-security.

[9] Denial of Service attack prevention (Accessed 24 Mar,2021). URL https://docs.microsoft.com/en-us/learn/modules/azure-well-architected-security/6-network-secur.

[10] Application Monitoring Definition (Accessed 08 Mar,2021, . URL https://www.techopedia.com/definition/29133/application-monitoring.

[11] Softwareversioning (Accessed 18 Mar,2021). URL https://en.wikipedia.org/wiki/Software_versioning#:

~:text=Software20versioning20is20the20process,unique20states20of20computer20software.&text=At20a20fine2Dgrained20level,this20information20is20computer20software.

[12] ApplicationSecurity,VMWARE,(Accessed 01 Feb,2021). URL https://www.vmware.com/topics/glossary/content/application-security.

[13] Azure ActiveDirectorry (Accessed 03 Apr,2021). URL https://www.nist.gov/itl/applied-cybersecurity/tig/back-basics-multi-factor-authentication.

[14] Azure Monitor,(Accessed 01 Mar,2021). URL https://docs.microsoft.com/en-us/azure/azure-monitor/agents/agents-overview.

[15] Application Insights, (Accessed 08 Feb,2021), . URL https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overvieww.

[16] Raygun, (Accessed 09 Feb,2021). URL https://raygun.com/platform/apm.

[17] ElasticStack , (Accessed 18 Feb,2021) , . URL https://www.elastic.co/guide/index.html.

[18] Logs.io (accessed 07 mar,2021, . URL https://logz.io.

[19] Raygun Documentation (Accessed 06 Mar,2021. URL https://raygun.com/documentation.

[20] Rayygun GDPR compliance, (Accessed 10 Feb,2021). URL https://raygun.com/gdpr.

[21] Logz.io Solutions (Accessed 06 Mar,2021, . URL https://logz.io/solutions/fully-managed-elk/.

[22] Application Insights Overview (Accessed 04 Mar,2021, . URL https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview.

[23] Azure Pricing Details (Accessed 18 Mar,2021. URL https://azure.microsoft.com/en-us/pricing/details/monitor/.

[24] Elastic stack:beats (accessed 13 mar,2021). URL https://www.elastic.co/search?q=beats&size=n_20_n.

[25] Elastic Stack:Logstash (Accessed 14 Mar,2021), . URL https://www.elastic.co/search?q=logstash&size=n_20_n&filters%5B0%5D%5Bvalues%5D%5B0%5D=documentation&filters%5B0%5D%5Bfield%5D=website_area&filters%5B0%5D%5Btype%5D=all&filters%5B1%5D%5Bfield%5D=product_name&filters%5B1%5D%5Bvalues%5D%5B0%5D=libbeat&filters%5B1%5D%5Btype%5D=all&filters%5B2%

5D%5Bfield%5D=product_version&filters%5B2%5D%5Bvalues%5D%5B0%5D=7.11&filters%5B2%5D%5Btype%5D=all.

[26] ElasticSearch (Accessed 15 Mar,2021). URL https://www.elastic.co/elasticsearch/.

[27] Elastic Stack:Kibana (Accessed 16 Mar,2021). URL https://www.elastic.co/kibana/.

[28] Azure Blob storage services , (Accessed 28 Feb,2021). URL https://azure.microsoft.com/en-us/services/storage/blobs/.

[29] Storage Services , (Accessed 01 Mar,2021). URL https://azure.microsoft.com/en-us/services/storage/blobs/.

[30] Azure Event Hub (Accessed 17 Mar,2021). URL https://https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-features.

[31] version Control,What is Version Control,(Accessed 05 Feb,2021).

[32] DistributedVersionSystem (Accessed 19 Mar,2021). URL https://content.intland.com/blog/sdlc/the-needs-that-version-control-systems-serve.

[33] Zero Trust Model, (Accessed 12 Feb,2021). URL https://www.microsoft.com/en-ww/security/business/zero-trust.

[34] Defense in Depth , (Accessed 14 Feb,2021). URL https://en.wikipedia.org/wiki/Defense_in_depth_(computing).

[35] CIA Triads , (Accessed 13 Feb,2021). URL https://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA.

[36] Encryption Definition(Accessed 20 Mar,2021). URL https://techterms.com/definition/encryption.

[37] Types of Encryption , (Accessed 12 Feb,2021). URL https://cyware.com/news/exploring-the-differences-between-symmetric-and-asymmetric-encryption-8de86e8a.

[38] Azure failover Scenario (Accessed 21 Mar,2021). URL https://docs.microsoft.com/en-us/azure/site-recovery/azure-to-azure-tutorial-failover-failback.

[39] Azure Databases (Accessed 22 Mar,2021), . URL https://azure.microsoft.com/en-us/product-categories/databases/#:~:text=Azure%20offers%20a%20choice%20of,saving%20you%20time%20and%20money.

[40] Databases Encryption (Accessed 25 Mar,2021), . URL https://docs.microsoft.com/en-us/azure/azure-sql/database/transparent-data-encryption-tde-overview?tabs=azure-portal.

[41] Azure LongTermRetention (Accessed 26 Mar,2021). URL https://docs.microsoft.com/en-us/azure/azure-sql/database/long-term-retention-overview.

[42] Azure KeyVault (Accessed 28 Mar,2021). URL https://docs.microsoft.com/en-us/learn/modules/manage-secrets-with-azure-key-vault/2-what-is-key-vault.

[43] Disaster KeyVault Recovery (accessed 30 mar,2021). URL https://docs.microsoft.com/en-us/azure/key-vault/general/disaster-recovery-guidance.

[44] Azure Layer7protection (Accessed 01 Apr,2021). URL https://docs.microsoft.com/en-us/azure/web-application-firewall/ag/ag-overview.

[45] Azure Single Sign On Service (Accessed 03 Mar,2021. URL https://azure.microsoft.com/en-us/free/active-directory/search/?&ef_id=CjwKCAjw_JuGBhBkEiwA1xmbRTikLLpaXsUZzuZgQb6lGeCWaLW4CjLsMA7fOoIrobbIMFk4LMsYDxoCsi8QAvD_BwE:G:s&OCID=AID2100088_SEM_CjwKCAjw_JuGBhBkEiwA1xmbRTikLLpaXsUZzuZgQb6lGeCWaLW4CjLsMA7f
BwE:G:s&gclid=CjwKCAjw_JuGBhBkEiwA1xmbRTikLLpaXsUZzuZgQb6lGeCWaLW4CjLsMA7fOoIrobbIMFk4LM
BwE.

[46] Azure Conditional Access (Accessed 02 Mar,2021. URL https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/.

[47] Azure MultifactorAuthentication (Accessed 02 Apr,2021). URL https://www.nist.gov/itl/applied-cybersecurity/tig/back-basics-multi-factor-authentication.

[48] ContinousIntegeration (Accessed 08 Apr,2021). URL WhatisCI/CD?Continuousintegrationandcontinuousdeliveryexplained|InfoWorld.

[49] Jenkins. URL https://www.tutorialspoint.com/jenkins/index.htm.

[50] Octopus. URL https://octopus.com/docs/getting-started.