Patch Update of application servers in .NET environment

[1]Sanjaya Shankar Hegde, [2]Dr Sharvani G S,
[1]student, [2]associate professor
[1]Department of Computer Science Engineering,
[1]R V College of Engineering, Bengaluru

Abstract— Users often do not install security-related and application software updates, leaving their devices open to exploitation by attackers. We are beginning to understand what factors affect this software updating behavior but the question of how to improve current software updating interfaces however remains unanswered. In this paper, we review one of the methods to implement the steps in software update to applications in .NET environment.

***IndexTerms* – .NET, Azure Blob, software update,PowerShell,JBoss**

I. INTRODUCTION

The user interface of an application is typically found on a user's desktop, while database management functions are typically found on a server. The application logic is located in the middle-tier in three tier architecture. The data and the user interface are segregated from the business logic or core functionality. The update system which incorporate three-tier architecture are more scalable, resilient, and adaptable than single-tier systems. Client Layer, Business Layer, and Data Layer are the three tiers of a three-tier architecture. These solutions are quicker and more secure since the client will not be able to interface directly with the update mechanism.

Vulnerabilities in client-side applications that run on user devices are on the rise. Typically, software vendors roll out software updates or "patches" to protect users by fixing these vulnerabilities and making changes to the software—such as adding new features, enhanced performance, or bug fixes. For this reason, the United States (US) government, various security agencies, and security experts advise end-users to download and install updates in a timely fashion to keep their systems secure.[2]

Patches to the servers need them to follow the software update protocols and installed on all the relevant servers in the network. Download, backup, install, rollback are some of the modules to be implemented in the patching process adopted here.

II. METHODOLOGY

A. Software used:

Azure SDK for .NET:
Azure SDK for .NET helps to connect to the cloud container and interact with the contents(blobs) in C# language. It is the interface between .NET application and Azure cloud storage.

Visual Studio IDE:
Visual Studio is an integrated development environment for computer programming (IDE) in .NET. It comes with a pre-configured workspace and a plug-in framework for further customization.

.NET MVP:

The MVP framework provides a Model-View-Presenter (MVP) architecture with preconfigured components that may be used to develop flexible and loosely coupled windows applications. The MVP pattern divides an application's multiple components (input logic, business logic, and user interface logic) while keeping a loose relationship between them.

- In most situations, the Model will be built up of POCO and will hold the application data.
- The View displays the model data and, in most situations, outputs windows forms to be displayed.
- The Presenter is in responsibility of receiving user requests and constructing an appropriate model, which is subsequently provided to the view to be shown.

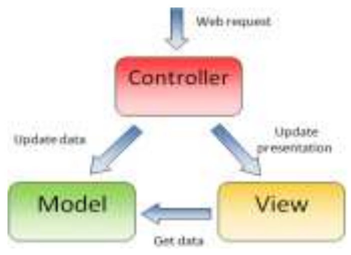

Fig. 1

MS-SQL server:
MS-SQL is a relational database management system that is available as Microsoft product. REST APIs developed fetch and save the data into the database.

JBoss
JBoss is a server that runs Java applications. The JBoss application server from Red Hat is an open-source platform for producing Java applications and a variety of other software applications. You can create and deploy Java services that can be scaled to match your company's needs.

PowerShell
Powershell is the windows command line tool designed for system administration.Its scripts can be used to automate most of the download and install steps of an update process.

B. Implementation:

Three phase of the software update process [1] are discussed in detail below.

- Downloading the Update
- Installing and delivering the Update
- Using the System Post-Update

We begin with download of the update step. Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data.The update contents are hosted in the blob storage account. They can be securely accessed by the users via azure authentication. The methods BlobContainerClient.GetBlobsAsync and BlobBaseClient.DownloadTo are used to list the contents of the container and download them respectively.[3]

```
// List all blobs in the container

await foreach (BlobItem blobItem in containerClient.GetBlobs())

{

   Console.WriteLine("\t" + blobItem.Name);

}

// Download the blob's contents and save it to a file

blobClient.DownloadTo(downloadFilePath);
```

For user interface, we use C# programming language. The UI is developed on .NET framework,specifically Windows forms. Buttons and datagrids ae used to provide input and display the responses from the cloud container. In this way , appropriate software patches are downloaded securely to the user’s computer. This completes the downloading the update phase of the software patching.

In the case of multiple servers, the downloaded patch needs to be delivered to their application locations. The list of servers is maintained in the database(MS-SQL) and retrieved to be copied to them. In the code below, dbConnString contains the dbname,username, password necessary to make the connection to database.[6]

```
using (SqlConnection connection = new SqlConnection(dbConnString)){

using (SqlDataAdapter dataAdapter = new SqlDataAdapter()){

const string cmdText =

"SELECT servers FROM serverTable";

SqlCommand command = new SqlCommand(cmdText, connection);

dataAdapter.SelectCommand = command;

command.CommandType = CommandType.Text;

using (DataSet dsDataSet = new DataSet()){

                dataAdapter.Fill(dsDataSet);

              }}}
```

The patch content is copied to each server on the table and further steps necessary executed from PowerShell object of .NET framework. PowerShell instance can be used to execute individual commands or complete scripts on the remote machines from the local computer that has the downloaded patch. Eventhandlers are setup to handle the different responses from the PowerShell instance execution.[4]

```
using (Runspace myRunSpace = RunspaceFactory.CreateRunspace())
      {
         myRunSpace.Open();
         using (PowerShell powershell = PowerShell.Create())
         {
            // Create a pipeline with any powershell commands
            powershell.AddScript("Set-ExecutionPolicy -Scope Process -ExecutionPolicy Unrestricted");
           //execute any specific scripts
            powershell.AddScript(@"C:\Users\you\Desktop\a.ps1");

            // execute the script
            var results = powershell.Invoke();

            powershell.Streams.ClearStreams();
            powershell.Commands.Clear();

            // convert the script result into a single string
            StringBuilder stringBuilder = new StringBuilder();
            foreach (PSObject obj in results)
            {
               stringBuilder.AppendLine(obj.ToString());
            }

         }
      }
```

For remote execution, powershell cmdlets to be used:

```
Invoke-Command -ComputerName Server1, Server2 -FilePath c:\Scripts\anyscript.ps1
```

These steps are used to deliver the patches to the multiple servers in the network. Powershell commands in scripts automate the various steps in installing the patch to the application.All the steps in the process can be appropriately logged for future reference.

Modern servers in production need to have high availability for the critical applications. Servers like JBoss provide mechanisms to reflect the change in application without stopping the complete server. The filesystem scanner checks the deployment folder at defined intervals. It auto deploys the changed content as set in the server configuration. Deployment scanner can be used with manual changes as well by creating the appropriate marker files for the changed applications.[5]

```
<deployment-scanner path="deployment" relative-to="jboss.server.base.dir" scan-interval="5000" auto-deploy-zipped="true" auto-deploy-exploded="false"/>
```

The above lines in standalone.xml of Jboss set the attributes for the deployment scanner. Some of the relevant marker file types for manual deployment are:

.dodeploy -- indicate that the given content should be deployed into the runtime (or redeployed if already deployed in the runtime.)

.deployed -- indicate that the application content has been deployed into the runtime. If an end user deletes this file, the content will be undeployed.

.failed -- indicates the failure of application deployment. The content of the file will include some information about the cause of the failure.

.isdeploying -- indicate that it has noticed a .dodeploy file or new or updated content and is in the process of deploying the content. This marker file will be deleted when the deployment process completes.

The hot deployment modes of the server restart specific content on servers without restarting all applications on server concurrently.This reduces the time spent in normal restart which take longer due to the cache of all the applications involved.The statistics of the users who download the updates and install them can be maintained with the Azure storage setup or specific monitor logs of the application. They help to provide better post update services to the users and maintain the number of updated users in the user group.This concludes the Post update steps in the patching process.

III. ANALYSIS OF ADVANTAGES

The above method is one among the many approaches to update any software system in the windows environment. Some of it advantages are:

- Security: it is able to provide secure access to application server artifacts stored in the cloud using Identity and Access management.
- Time taken for workflow: the time taken to run the workflow decreases as the servers are not restarted and all scripts for the networked computers are run by windows remoting.
- Ease of execution: the manual steps of copying patches, deploying the servers are automated simulating a single click installation feature.
- Telemetry: Azure storage logs the points of failure in cloud, enabling a faster resolution of situation. Monitor logs can enable event log instances of success in upgrade.

IV. CONCLUSION

The paper presents a way to implement software update delivery in windows applications using .NET framework and it provides a multi step implementation using three steps. The response time for all the web services is less than 200 milli seconds. This method of implementation can be used for enterprise applications development. The actual operating of the system is very steady and trustworthy. This has been demonstrated to be an excellent lightweight J2EE application update solution.

V. REFERENCES

1. Mathur, Arunesh, Josefine Engel, Sonam Sobti, Victoria Chang, and Marshini Chetty. "" They Keep Coming Back Like Zombies": Improving Software Updating Interfaces." In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*, pp. 43-58. 2016.
2. Li, Frank, Lisa Rogers, Arunesh Mathur, Nathan Malkin, and Marshini Chetty. "Keepers of the machines: Examining how system administrators manage software updates for multiple machines." In Fifteenth Symposium on Usable Privacy and Security ({SOUPS} 2019), pp. 273-288. 2019.
3. https://docs.microsoft.com/en-us/dotnet/api/overview/azure/storage
4. https://docs.microsoft.com/en-us/dotnet/api/system.management.automation.powershell?view=powershellsdk-1.1.0
5. https://docs.jboss.org/author/display/AS7/Application%20deployment.html
6. https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=netframework-4.8