System Design

# Specifications

Management Summary

The project here is to design and implement a containerized process for deploying projects on a regular, automatic and in a secure way to a respective environment.

The development efforts till date

* + 1. Applications are rewritten to support containerization
    2. Server have been setup to support the initial deployment
    3. Build and deploy scripts have been created and tested.

The efforts till date have been on track to the initial proposed timeline and the project is making great progress. The project cost was initially estimated for a total cost of $100,000 with the following resources developers, systems analysts, project manager, server administrator and application administrator.

The benefits of the new system are Automatic deployment, auditing and logging of what is deployed where, secure migration of code and quality code deployment. Other than these one major advantage is the ability to scale the application when required.

No issues that would need to be addressed at this time for the project completion. The scheduled for what was planned is still going to be accurate and there are no delays in the planned work.

## System Components

There are a number of components that are required for the project to be marked as complete. The first being a user interface for the management to look at the metrics based on weekly build completions, success, failures and the number of lines of code that has been changed.

This interface will require several databases with relevant information about the developer, build information, reporting manager and project information and description.

Apart from this interface there will also be servers and build and deploy scripts that will be developed. A quick reference of the dataflow and the context diagram explaining the process at a high level is illustrated below.



Containerizing software is new and open sourced. No additional purchase is required. One note to make is that all the containers need to be started when the server reboots. The steps to do this is documented in the support documentation which the server administrators own.

The inputs for the process will the code that is checked in and the output to this will be the reports being generated and application deployed to an environment.

## System Environment

Few notable constraints are the number of builds that are being triggered at once. Currently the servers that are in place are 4 core machines and there are two of the build servers, so no more than 8 builds can be processed at a given time. The other builds will be queued until the processor thread is available. This may cause some lags initially but, after there is a better estimate of the number of builds per day, the resources can be increased accurately.

The deployment of the code will follow coding standards and will be scanned for any know vulnerabilities to make sure the code is secure to deploy.

The servers in place to deploy the containers are not intense in resources as the only application that is being deployed is the member registration. If there needs to be additional applications that are to be deployed the resources of the machines have to be increased.

The containerization software is to be updated on regular basis for reduced any security vulnerabilities and the application can be scaled to have 5 running instances as per the load test.

The data in the database is backed up every week and the data is retained for 12 weeks as per the company standard. The number of databases required for the project will be 8. These databases are crucial for the display of the reports and the build statistics.

## Implementation Requirements

Developers and administrators have to be trained on how to use rewrite and deploy applications using containerization and understand the concept of how the artifacts or packages that are created from the code deploys automatically. The server administrators have to understand the resource utilization and adjust the memory and the processing power accordingly.

## Time and Cost Estimates

The project is scheduled to be done in 10 week timeframe. Developers and system analysts are required mostly throughout the project and the administrators are required full time initially and for support purposes at the end of the project timeline.

Total cost to date is close to $ 75,000 and we are very near to fully implement the project. This cost also involves the purchase of new server blades and implementation of the infrastructure.

# Data Design

The following details the data structure on how the data is stored and accessed. This structure is for the reporting interface for the management to refer.

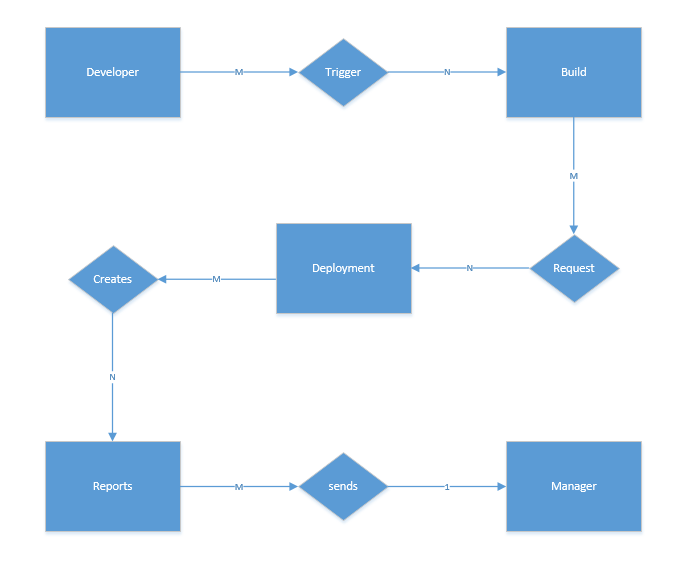
## ERD

List of entities for the entity relationship diagram

* Developer
* Build
* Deploy
* Manager
* Reports

The type of relationship between these entities is described below.

Developer can trigger multiple instances of a build and a build can have multiple developers. So the relationship between these two entities is M:N. The same way one instance build can have multiple deployments in various environments and a deployment can have multiple builds which also makes it a M:N relationship. Each deploy will create multiple reports so another M:N and Manager has access to all the reports so this will be 1:M. The following diagram shows how this relationship can be defined in diagram.



## Normalization

The above relationship between entities can be converted into data tables. The reporting table layout will contain the following columns

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Build ID | Developer Name | Developer ID | Project Name | Project Description | Environment Deployed | Manager Name | Timestamp |
|  |  |  |  |  |  |  |  |

The table above can be converted to a 2NF model by making the primary key a combination of the Build ID and the Timestamp columns. So the table layout will look like the following

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Build ID | Developer Name | Developer ID | Project Name | Project Description | Environment Deployed | Manager Name | Timestamp |
| pk |  |  |  |  |  |  | pk |

This layout can further be converted into 3NF model by having individual tables as follows

|  |  |  |  |
| --- | --- | --- | --- |
| Build ID | Developer ID | Project ID | Timestamp |
| Pk | Fk | Fk | pk |

The Build ID and the timestamp are still the primary keys and the developer ID and the project ID are the foreign keys which map information in different tables. The complete layout can be acquired by joining the master table and the other two tables which are developer and project. The layout for the other two tables is shown below.

|  |  |  |
| --- | --- | --- |
| Developer ID | Developer Name | Manager Name |
| pk |  |  |

|  |  |  |
| --- | --- | --- |
| Project ID | Project Name | Project Description |
| pk |  |  |

Designing tables in the 3NF model will reduce duplication and make maintaining the data in the tables easier. Pk – primary key.

# User Interface Design

## HCI (Human Computer interactions)

The report page will be authorized and authenticated using LDAP (Lightweight directory access protocol). This is done to ensure that only management has access to the reports that have been generated using the build and the deploy process. The management will presented a form page to enter credentials like username and password and click on the login button to enter the report dashboard. The user is expected to use keyboard and mouse for the first form. Validation rules are used to check if the username and the password are entered correctly. If incorrect credentials are entered, an error message is displayed on the screen and the user will be asked to enter the credentials again. When the user enters the password additional rules are applied on the page to mask the information. Another rule that will be applied is to make sure the password length is more than 8 characters.

The subsequent pages can only be accessed after a successful login and can be interacted by mouse but, if required keyboard can be used to get or filter any of the information which is not readily available.

Information is delivered will be internet based information delivery.

## GUI (Graphical User interface)

The user interface has the following important features

* All the pages will have the same layout. The header, footer and the main section will look and feel the same. The content may be different in each page but not the style, font and the layout.
* The GUI will have three pages mainly- The first page is to login, second is to look at the weekly reports and the third page is to look at the monthly reports
* Multiple sections of the page like the code analysis and the difference in the number of lines of code from the last build can be accessed by using the menu bar to highlight the section of the page or, user can also scroll down the page to get the information.
* A search bar is provided at the top to navigate to a section of choice quickly.
* A filter is provided so that the user can get the result of the build based either developer information and/or project information. This feature is enabled to save time for the users to get quick access to the data. This enhances the productivity of users. Users can wither type the developer name or click on the drop down list to access the same information. This list is sorted alphabetically
* Additional fields are also available to filter the data. Users can enter the information for the build timestamp before or after which the builds occurred.
* Validation rules are added to make sure the search fields follow rules required to get the appropriate data
* The layout is designed in a way that data being displayed has a text that is readable and the font size is constant in all the pages.

# System Architecture

## Organization culture

Considering the business functionality and the developer preferences and knowing that organization follows a centralized architecture, the design proposed is in line with all the facts and recommendations.

This proposal follows the architecture which makes maintaining the system easier and efficient. Operational feasibility is one of the main considerations for the architecture of this project. Extensive research and fact based analysis is done to determine what type of architecture will best suit this project implementation.

## Resource planning

Enterprise resource planning software is considered to establish a standard on how data modeling and interface design for the graphical user interface. Network and processes also will follow the standards set by the ERP software.

The concept of supply chain management will assist when a developer triggers the build and when the artifact is built, this in turn can cause the artifact to be deployed to other environments and after the deployment is done the reports are updated and sent to the management.

## Total cost of the ownership

There are two types of cost analysis, the first being the hard cost for the server blades, additional network interface cards and other such infrastructure related costs and the second is called a soft cost which is required for the support and product maintenance.

The following information has been considered for the total cost analysis

* In house development has been considered as the development cost to convert an existing application with in house developer is comparatively less when compared to out sourcing
* There are other products enable containerization and have a better model but, the software A is chosen as it is open source, secure and has extensive documentation for trouble shooting issues.
* No events based on economic, regulatory or government have occurred that changed the cost of the project.
* No major decision changes have been made by the company that would affect the cost associated.
* With the information available, the total cost has been updated but there is no significant difference to what was proposed initially.

## Scalability

Application will be scaled as necessary. Reactive model will be followed to scale the application as necessary. Another process will be in place to make the application elastic based on the number of requests. The application instances will be increased if the number of requests are more than a certain number and will be slimmed down if the number of requests are below that certain number.

## Integration

Integration of the application is the next important aspect that needs to be considered next. Since multiple instances of the application are deployed there is a need to have a load balancer on top of the all the instances so that the requests can be routed in a way that the end user will not have to worry about what instance to hit and a constant performance of the application is maintained.

The load balancer will also have a reverse proxy so that the end users will have a constant URL that they can keep requesting data from. This URL will be publically hosted and all the necessary public cert that are required to make the data transfer between the user’s browser and the application will be put in place.

## Interface requirements

Few requirements for the interface,

* Users need to be authenticated.
* The response time of the interface should be below 5 seconds at all time
* Crash reports should be collected and associated incidents should be generated to the support team
* The interface modeling should follow standards set by ERP software

## Security

Code quality and security is a crucial piece of this effort. All the code that is built will be scanned for known vulnerabilities. In addition to this, all the servers that this code is being deployed to will be hardened so that the surface area for the server to be compromised is minimum. The URL’s have public certificates which will ensure that the data is transferred from the browser to the application following SSL protocol. The interface also has LDAP authentication, which allows the data to be only visible to authenticated users. Only minimum necessary data will be displayed on the interface, this is another security strategy to avoid sending data that is not requested.

# Feasibility Analysis

The proposed solution will not only help with the immediate problem of how to scale an application effectively but also help in providing a common way for building artifacts and deploying them securely to respective environment. The cost of scaling other applications will be considerably small.

The proposed solution will also improve the code quality and deploy applications that are secure and reduce the risk of exposing the organization to cyber threats. The whole process is audited and can be presented during inspection. This will help in the organization reputation increase if the builds are audited and there is an accountability on how the code was migrated. As there will be multiple instances of an application deployed there will be zero down time of any application and this will increase the user experience.

**References:**

Tilley, S. & Rosenblatt, H. (2017). Systems Analysis and Design, Eleventh Edition. Boston, MA. Cengage Learning. pp. 237