**ADOPT (Application Delivery Optimization Platform)**

Central Reporting dashboard, access different tools from a single login & define approval workflows.

Table of Contents

[1. Introduction 2](#_Toc453661105)

[1.1 Overview 2](#_Toc453661106)

[1.2 Scope 3](#_Toc453661107)

[2. System Features 4](#_Toc453661108)

[2.1 Architecture and explanation 4](#_Toc453661109)

[2.2 Reporting 5](#_Toc453661110)

[2.3 User and Access Management 7](#_Toc453661111)

[2.4 Application design and data flow 9](#_Toc453661112)

3. APPENDIX I: Sample Reports

4. APPENDIX II: List of to be implemented features

**Note: Following names are used synonymously in this document**

1. **ADOPT web application** or **ADOPT web portal**.
2. **Tools** or **Tabs** or **Linked Applications**.

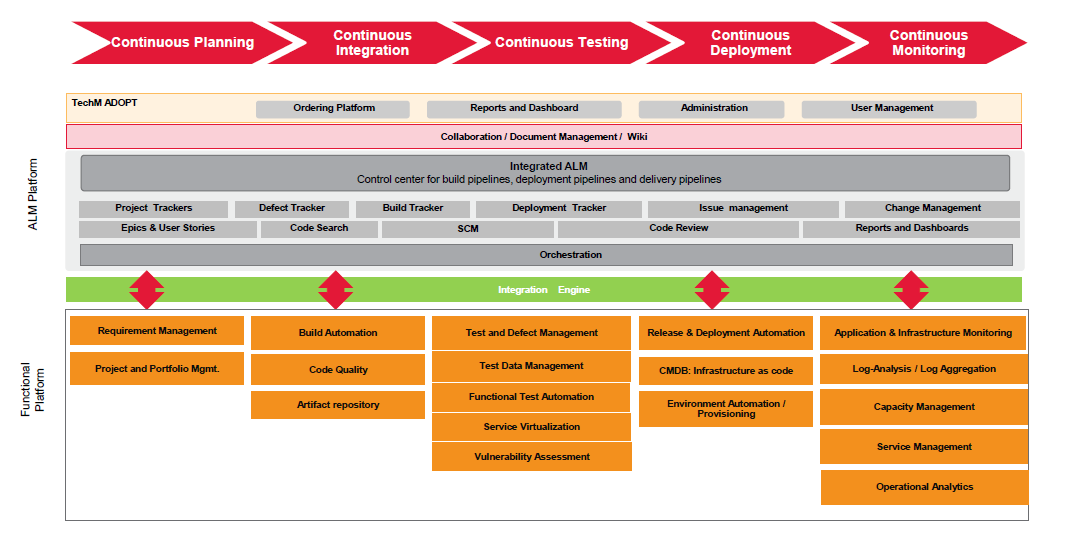
**Documented by:** Anil Kumar Mahapatra, Aditya C S

# Introduction

## Overview

DevOps in general has a very broad scope and each organization or team tries to adopt the DevOps concepts based on their application landscape and process. At TechMahindra, we do lot of R&D on DevOps and its practices. Also, we interact with lot of customers and help them implement the DevOps practice.

When the team was formed way back in 2014, we started by developing ADOPT, a tool agnostic platform where we enable reporting, traceability and induce more automation. Refer **Figure 1.0** to get a fair idea of what ADOPT is. We have had used many tools to achieve this. For ex: we used Collabnet TeamForge as ALM tool, Jenkins for orchestration, SonarQube for static code analysis, CA RA/Udeploy for deployment, Chef/Puppet for Environment provisioning and deployment. Using Jenkins as orchestration engine we were/are able to communicate two ways with our above mentioned sample tool stack. By two way communication I mean, triggering an action in the particular tool from Jenkins and getting the feedback from the tool. To get the feedback from the tools we developed plugins where we accessed APIs of the tools and pulled the data, and using APIs from ALM tool we pushed the data to ALM. By this we get a single truth of repository from where we can utilize the data and enable reporting and traceability. We tried to bring maximum automation at each stage.



**Figure 1.0**

We faced many challenges while presenting our platform (ADOPT) to customers and reasons for it are many. Main points are mentioned below.

1. We were trying to push CTF as ALM and customer was (is) not ready to invest on it.
2. We were successful in implementing CI-CD in customer POCs; however we were unable to show strong reporting as customers were using Jira/HP ALM as ALM tools. Since we were dependent on ALM for holding the feedback from various tools and do reporting, we were not able to show very good reports through Jira/HP ALM as in CTF due to some limitation in the tools.
3. Some Customers even asked us for creating Jenkins templates for CI-CD pipeline and at that point of time we didn’t have as strong solution to it.
4. Some Customers don’t want to invest in purchasing and installing the tools on-prem but want to access the ADOPT platform on cloud. Currently we don’t have this feature.

This list continues as we face more customers and it’s good for us to improve our solution.

With all this background and experience we feel the need to come up with a solution which will answer all the customer needs and provides many benefits to the teams/organizations who adopts it. For this reason we are developing an application in which we try to address many items like reporting, traceability, cloud enabled platform, encourage more automation and this features list expands.

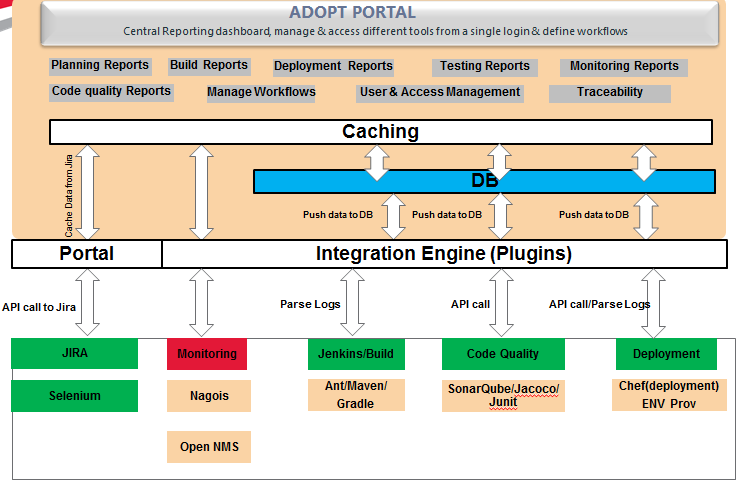
## Scope

In the previous section we spoke lot about DevOps, its practice and what we have done all these years. With our past experience and also by foreseeing into the future, we are developing a solution which should address most of the problems which customers face today. This solution not only solves the customer problems in implementing DevOps practice but also take the customer to the better world by introducing maximum automation into his systems. The present scope of this solution is developing a web application which will include the following components.

1. Reporting engine and traceability.
2. SSO & LDAP integration.
3. Cloud enabled.
4. Support integration of all industry standard tools and customer specific tool.
5. Templates for automated set up and configuration of new projects.
6. Templates for changes in Orchestration layer like change in Jenkins CI pipeline, configuration change in jobs etc…

# System Features

## Architecture and explanation



**Figure 2.0**

We have many areas in DevOps to address and most important requirement of any DevOps implementation would be reporting, traceability and to induce more automation. The ADOPT application which we are building is to address all the above said requirements. The ADOPT portal has many features, some of which are listed below.

1. Very Vast Reporting capability, right from planning till monitoring.
2. End to End Traceability.
3. Promotes Automation in most of SDLC phase

The architecture has three main components

1. Integration Engine
2. DB and Caching Layer
3. UI and business logic layer

**Integration Engine:**

Plugins developed/to be developed by DevOps team for getting feedback from different tools and inserting the data to DB

**DB and Caching:**

DB is designed in such a way to capture variety of data which is coming from different tools and also to complement reporting. Caching layer is used for caching DB data for fast response.

**UI and business logic:**

To implement functionalities like Reporting, Role based linked application access, Role based Report access, create project setup templates etc…

Please refer the next sections of this document which explains about each and every feature of ADOPT and Adopt Portal.

## Reporting

Reporting is the means by which you can measure your project(s) performance. Most of the tools we use today will have its own reporting capability and we use it in different ways to get the proper metrics. However this has limitations, though we get various metrics from the tools, there is no single dashboard from where you can view these reports. In ADOPT we not only address the above said limitation instead we address both reporting and traceability from a central dashboard. The following section explains how we achieve centralized dashboard and what tools we use.

**How we achieve centralized Reporting?**

Jenkins is the orchestration layer from which most of the tools is connected. Triggering any action in a particular tool would be handled through Jenkins. So, once the tool has completed its action we have to pull the data from that tool and push the data to our DB. To get more clarity on this, please look at the below example. Let’s assume that we are using SonarQube for code analysis and we have to fetch code quality metrics like, technical debt, complexity, and blocker, major and minor issues etc. for reporting.

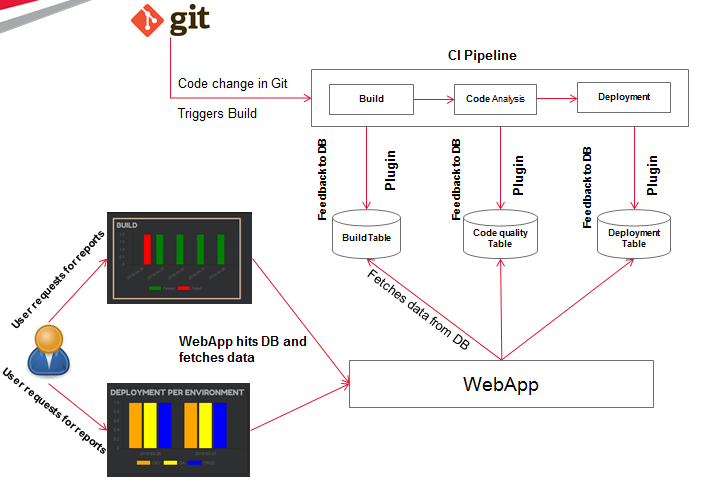
1. Use SonarQube for code analysis.
2. Trigger sonar analysis from Jenkins in CI pipeline.
3. Once analyses are complete, access SonarQube APIs and fetch the data.
4. Push the data to ADOPT DB.

The same logic applies for most of the tools. However there may be scenarios where we might not be able to access APIs or the tool itself is not exposing any APIs. In such cases we will parse Jenkins logs to fetch the data and insert it in to ADOPT DB.

We have to develop plugins to pull data from different tools and push it to ADOPT DB. Once we have the data residing in ADOPT DB, we can pull the data from DB and present it in the UI layer for reporting. Please refer **APPENDIX I** section which briefs about some of the sample reports available in ADOPT portal. The following is the Front-End technology stack we are using to achieve reporting.

1. HTML, Bootstrap & Css.
2. ChartJs (Open source JavaScript library for Graphical reporting)

**Figure 3.0** explains how data flow happens for displaying report to user.



**Figure 3.0**

Following section explains the data flow:

1. CI pipeline is configured for jobs like build, code analysis, deployment etc.
2. At the end of completion of each job we run the plugin which fetches data from the particular tool and inserts it into DB.
3. Whenever the user requests for report, webapp makes an Ajax call to the servlet which in turn calls service and DB layer to fetch the data.
4. Data fetched is parsed and fed to Chart Js in case of graphical reporting or to Html tables in case of tabular reports.
5. Html or Chart Js renders the report to the user.

## User and Access Management

1. **User Management**

This is one of the modules in the Administration page of the portal. It deals with addition, deletion and modification of users and user roles.

1. **Roles**

Create various roles for a Project.

1. **Projects**

On board, different projects to the portal.

1. **Tabs Configuration**

We have different tools in the different tabs of the portal. Each tab holds an unique tool that serves different purpose in a project lifecycle e.g. planning tab deals with project planning, build and deployment tabs deal with build process and deployment process respectively.

Tab Configuration is one of the modules in the Administration page, which helps user to configure a new tab that will hold a frame for new tool.

1. **Role Report Mapping**

For a project there will be number of users and each user will be mapped to different roles. In this module permissions can be set as to which role can access to what reports.

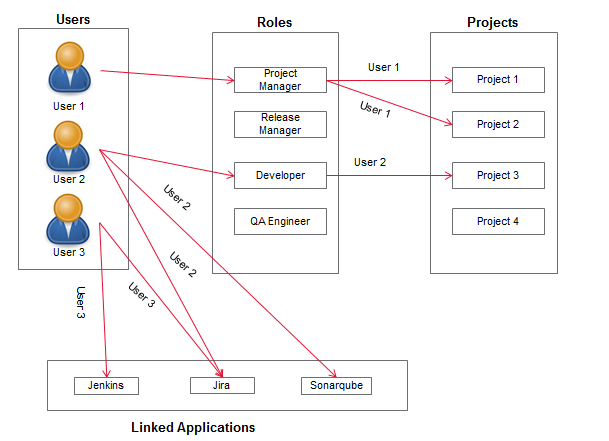
1. **Role User Mapping**

Map which user belongs to what role in a project

1. **Role Linked application Mapping**

Map which user mapped what role will be able to see which linked application.

Refer **Figure 4.0**



**Figure 4.0**

A Little explanation of what **Figure 4.0** means:

1. User 1 has a role or Project Manager in Project 1 and Project 2.
2. User 2 has a role of developer in Project 2.
3. User 2 has access to Jira and SonarQube in Project 3
4. User 3 has access to Jenkins and Jira and he might be mapped to a different project.

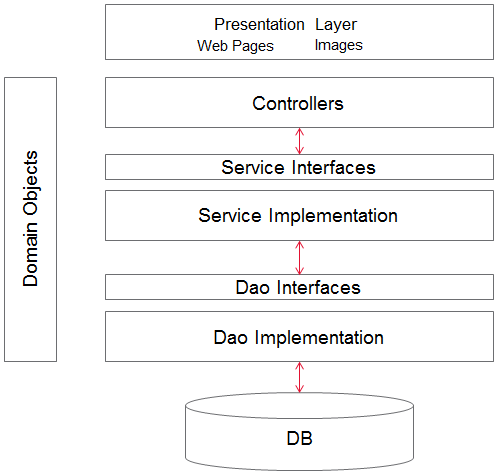
**In brief:**

* A user might be playing different roles across projects.
* A user might have access to different set of tools (linked applications) across projects**.**

## Application design and data flow

The following is the typical data flow for Reporting

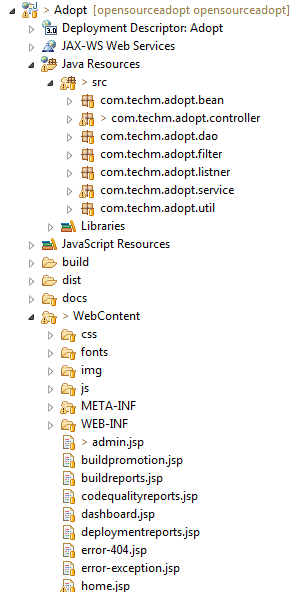
1. From plugins, data from different tools is stored in ADOPT DB.
2. Request coming from the UI is handled by controller.
3. Then the service layer takes care of the business logic required.
4. Service layer accesses the Dao layer for DB transactions.
5. After the proper data is fetched the data is presented in UI.



**Image 4.0**

**Following is the folder structure and naming convention used for development**

1. All package name starts with com.techm.adopt. For ex: package name for bean classes in **com.techm.adopt.bean**, for controller **com.techm.adopt.controller**.Refer **Image 5.0.**
2. **JSP** pages: WebContent folder.
3. **Css** files: WebContent/css folder
4. **JS** files: WebContent/JS folder
5. **Images**: WebContent/img folder
6. **Fonts**: WebContent/fonts folder
7. **DB** connection configuration: WebContent/META-INF/context.xml
8. Dependent **libs**: WebContent/WEB-INF/lib
9. **Web.xml**: : WebContent/WEB-INF/web.xml
10. **Log4j.properties** file: WebContent/WEB-INF/log4j.properties



**Image 5.0**

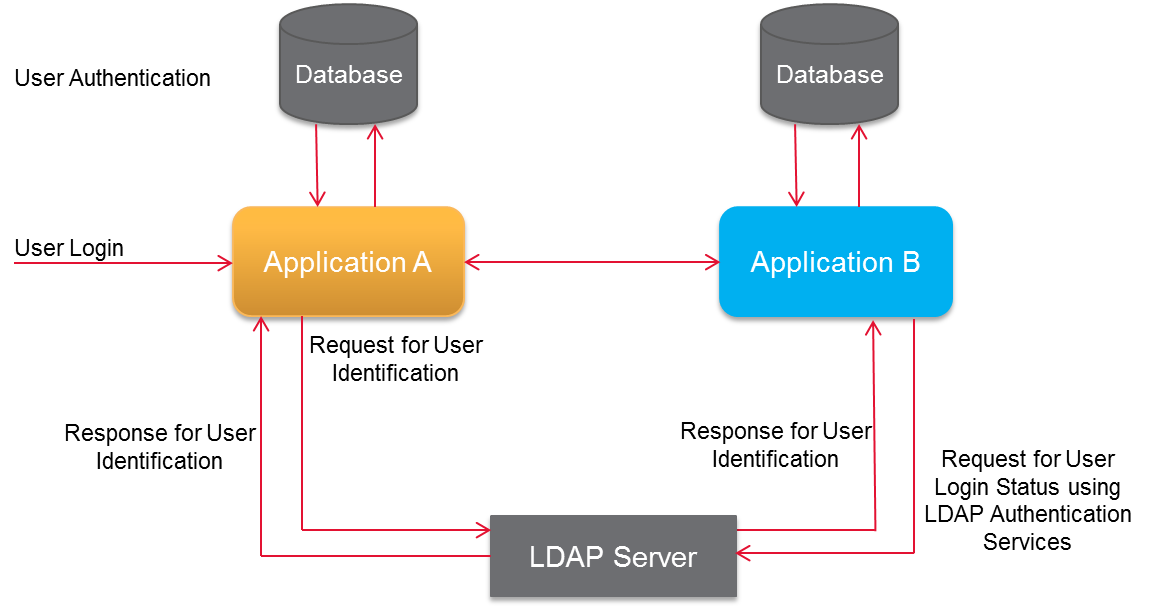
**Single Sign-on (To Be Cont.…)**

Using Single Sign-on (SSO) properties we can access multiple but independent software systems. Using SSO property a user logs in with a single user ID and password to gain access to a series of connected systems or applications over the web or in other words a user doesn’t have to use multiple credentials to gain access to multiple applications or systems that are dependent or connected within a single cluster.

This can be achieved by using the Lightweight Directory Access Protocol or (LDAP) and stored LDAP databases on servers also called directory servers.

We can achieve a basic version of single sign-on over the IP networks using cookies provided that all the sites are using the same parent DNS domain.

**Single Sign-on using LDAP Server (Current considered Solution) (To Be Cont.…)**



* The user logs in to ***Application A*** through browser.
* ***Application A*** validates the user information and generates an authorization identifier.
* ***Application A*** will route the request along with authorization identifier to the LDAP server.
* The routed request is validated against the user information at LDAP Server; if user identification is successful the application will populate the user credentials along with the authorization identifier.
* When user clicks on the ***Application A*** for SSO site link, application will send the user information to ***Application B*** along with the with authorization identifier.
* ***Application B*** will route the request to LDAP Server for user identification.
* User will be authenticated and identified against the user information at LDAP store and will automatically log on to ***Application B*** in case of correct credentials are passed.

**Handling Sessions**

* User login to the ***Application A*** and initialize the session.
* ***Application A*** sends session ID to ***Application B***.
* ***Application B*** will always check the user session with the ***Application A*** by passing the session-ID with the request before performing any process.

**Benefits**

* LDAP supports all flavors of UNIX (Linux, Solaris etc.) and Windows.
* LDAP schema validates the credentials of end user hence maintaining the user information in encrypted form. This enhances the security of applications.
* The component communications are through LDAP services providing credential verification always on the component level as well.

**Automated setup of Project and its associated components (To Be Cont.…)**

Every new project on boarding requires basic setup like creating project in ALM, setting up CI pipeline, Configuring SonarQube etc… This setup would involve lot of manual intervention and would consume time. To overcome this, we can automate the whole process of on boarding the project. Following is the currently planned way of implementing this.

1. User requests for project on boarding through mail.
2. Concerned approves the request
3. Once the approval is done, the automated process of project setup kick start.
4. We can access APIs from different tools to accomplish this

**APPENDIX I**

1. **Planning Reports**

On the *Project Summary* tab of the dashboard we can see summarize reports of sprints and deployments report and the project burn down chart.

The sprint report and project burn down chart has been fetched from the ALM’s database using the APIs provided by the ALM tool.

The deployment summary report is taken from the deployment tools using its APIs and has been saved to the ADOPT portal’s database and has been displayed in a tabular form.

In the deployment summary report we can click on the build numbers to see the different tasks associated with it. This report is a cross-tool report.

We are using an adaptor to establish the relationship between the deployment and ALM tool to generate this report.

1. **Build Reports**

On the *Build* tab of the dashboard we can see the Build Summary report and Day wise build count graph.

Both the reports are parsed from the Jenkins build logs and the necessary details for preparing build report has been saved to portal’s database from the logs.

Then the build data is taken from the database and has been populated in tabular and graphical format to showcase build summary and day wise build count report respectively.

In build summary report we can click on build numbers and see the console log of that respective build in Jenkins.

1. **Deployment Reports**

The *Deployment* tab in the dashboard displays Day wise Deployment Count and Day wise Deployment Count on each Environment.

The data for these reports are taken from the deployment tool with the help of its API. Then those deployment data were saved into the portal’s database and made into graphical reports.

1. **Code Analysis Reports**

The *Code Analysis* tab in the dashboard showcases the Code Analysis Summary tabular report and Code Analysis Issue Summary bar chart report.

We took the data for code analysis from SonarQube by using its APIs and saved it to the portal’s database and using those data we rendered the code analysis reports.

1. **Test Trend Report**

The *Testing* tab in the dashboard displays the test trend reports. That renders the data from the ALM’s database which is fetched using its API.

**APPENDIX II**

**To Be Done:**

1. High Availability Architecture of ADOPT Portal.
2. Cloud Enablement.