Continuous Delivery & Pipeline Service Tools Navigator

DISSERTATION

Submitted in partial fulfillment of the requirements of the

M. Tech. Software Engineering Degree Programme

By

Rowllah T Basheer

2015HS70041

Under the supervision of

Sumeet Raj (Senior Developer)

Dissertation work carried out at

SAP Labs, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

(April 2019)

SE SAP ZG629T DISSERTATION

Continuous Delivery & Pipeline Service Tools Navigator

Submitted in partial fulfillment of the requirements of the

M. Tech. Software Engineering Degree Programme

By

Rowllah T Basheer

2015HS70041

Under the supervision of

Sumeet Raj (Senior Developer)

Dissertation work carried out at

SAP, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

(April 2019)

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**SECOND SEMESTER 2018-19**

**SESAP ZG629T DISSERTATION**

Dissertation Title: Continuous Delivery & Pipeline Service Tools Navigator

Name of Supervisor: Sumeet Raj (Senior Developer)

Name of Student: Rowllah T Basheer

ID No. of Student: 2015HS70041

## Abstract

Cloud application and product development teams need a continuous production line for their software application. This production line is called the continuous integration and delivery pipeline. A continuous delivery (CD) pipeline is an implementation of the continuous paradigm, where automated builds, tests and deployments are orchestrated as one release workflow. Put more plainly, a CD pipeline is a set of steps your code changes will go through to make their way to production. A CD pipeline delivers, as per business needs, quality products frequently and predictably from test to staging to production in an automated fashion. SAP being a cloud company, focuses on creating products which are CNCF (Cloud Native Computing Foundation). By building this application, the above the problems will be addressed, and will help increase development productivity for teams at SAP.

In this project, I am targeting to provide product development teams with the knowledge of Cloud Delivery Pipeline tools available with relevant and updated information about each tool in detail. This way, my application will act as an interface to the Production Services teams and development teams. The application will help in virtually building a pipeline for the product by providing choices among relevant tools for each stage in the pipeline. The end user will be able to download the relevant information for further reference. Having achieved the above I would be targeting to solve problems like getting the right information on pipeline tools, analyzing what pipeline tool fits where becomes a critical problem for the development teams. Additionally, it will help in identifying product standards is and what tools is are compliant in the wide range of available tools. Development team will also be able to get relevant contacts of the service teams for each tool. This would then make my application a single go-to place for all their Cloud Delivery services.

# Problem Statement

Cloud application and product development teams need a continuous production line for their software application. This production line is called the continuous integration and delivery pipeline. A continuous delivery (CD) pipeline is an implementation of the continuous paradigm, where automated builds, tests and deployments are orchestrated as one release workflow. Put more plainly, a CD pipeline is a set of steps your code changes will go through to make their way to production. A CD pipeline delivers, as per business needs, quality products frequently and predictably from test to staging to production in an automated fashion. Some of the major concerns that exist in the CI/CD domain are:

1. To get the right information on the pipeline tools used and to ensure that information is consistently updated
2. In the everchanging business environment, analyzing what pipeline tool fits where becomes a critical problem for the development teams
3. Additionally, in the wide range of available tools it is difficult to know what the product standards is and what tools is are compliant.
4. Development teams also find it difficult to know who-is-who in the Software Production domain, for example whom to contact in case of a Github or a piper issue, etc.

SAP being a cloud company, focuses on creating products which are CNCF (Cloud Native Computing Foundation). By building this application, the above the problems will be addressed, and will help increase development productivity for teams at SAP.

# Business Process Flow

The business flow would have the following:

* Providing a web interface to end user using which end user can explore various tools in offering
* Based on information, user can continue building pipeline for their software by providing basic or advanced filters on the screen
* The user is then taken to a guided UI which accepts user inputs and creates possible pipelines for their respective use cases.
* The pipelines can also be saved in the application for future reference and usage
* Further options are provided like to download the pipeline or create comparison charts, etc.

# Objectives

The Tools Navigator application aims at providing development teams with:

* The knowledge of Cloud Delivery Pipeline tools available with relevant and updated information about each tool in detail.
* An application interface to the Production Services consultants and experts.
* The ability to virtually build a pipeline for their product by choosing relevant tools for each stage in the pipeline and download the relevant information for further reference
* A single go-to place for all their Cloud Delivery services including storing various pipelines and viewing them

# Uniqueness of the project

As of today, there exists no tool that solves the above problems [This statement is as per the information given to me by my manager and service owner]. This is also the prime reason for the project to be monitored by the Production Services teams, the project would also be productized at SAP if successfully implemented.

# Benefits to the organization

The application will help in increasing the development productivity for all teams developing cloud applications at SAP. Further, this also gives the knowledge of Cloud Delivery Pipeline tools available with relevant and updated information about each tool in detail to the teams. For users who do not have knowledge on whom to contact for what, the application provides an interface to the Production Services team at SAP. The teams will then be able to virtually build a pipeline for their product by choosing relevant tools for each stage in the pipeline and download the relevant information for further reference. It will also provide a single go-to place for all their Cloud Delivery services including storing various pipelines and viewing them.

# Scope of Work

The scope of the project is limited to the different tools and stages with relevance to the cloud delivery pipeline for teams developing cloud application and their respective service teams.

# Resources Needed

For developing the application, we would need:

* Visual Studio Enterprise 2017 [for application development]
* SQL Server Enterprise (ntx64) and SQL Server Management Studio [for DB]
* Microsoft Windows 2012 Standard with IIS [For hosting web application]
* Know-how of languages – SQL, C#, VB.NET, Angularjs and HTML5
* Authentication Software (SSO)
* Git [Code Repository]
* Microsoft TFS [Software Build Environment]
* Guidance from Production Services team at SAP in steering the project [Rinku Dubey, Torsten Stefan, Adithya K Bhagavan, Rohit Meharia, Sofia Krause, Martin Riesterer]
* Guidance and support for application development from Mentor [Sumeet Raj]

# Potential Challenges and Risks

1. Formulation of Decision Matrix – which becomes the core brain of the application. This decision matrix will have to take every possible combination of DB entities and serve as a basis to take decisions on the application frontend
2. To compare the chosen solution by end user with different available frameworks (Piper and CodePipes) and create a comparison chart, using which the development team can take better decision

# Background of Previous Work Done in the chosen area

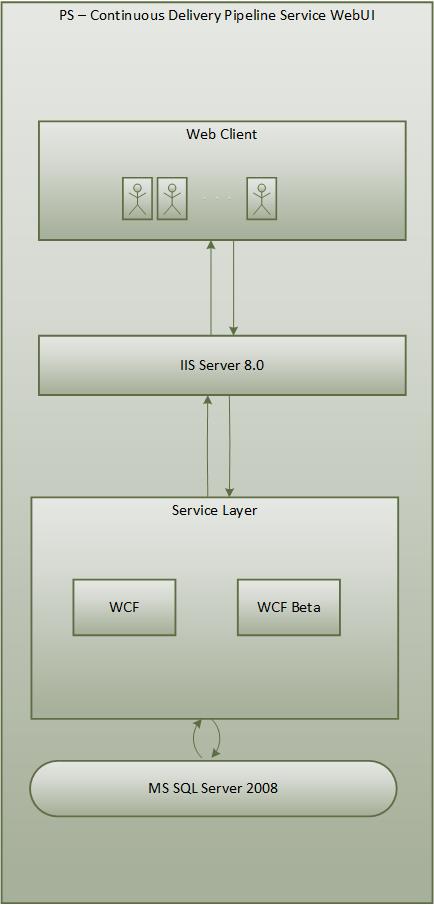
An effort was put in by members of Production Services team to theoretically propose a model that may serve in solving these problems. For references:

* <https://jam4.sapjam.com/blogs/show/beJTdYUKxMhxdbUo9Rtf65?_lightbox=true>
* <https://github.wdf.sap.corp/PS-CDPipelineServices/PSCD-Documents/blob/master/Tool_Matrix/Tool_Matrix.pdf>

It was later proposed by the management team that further research and investment on this topic should be done, which is where my mentor and I joined the project to take the research and implementation further.

# Solution Architecture

TAM diagram of the application:



# Detailed Plan of Work

|  |  |  |  |
| --- | --- | --- | --- |
| # | Task Name | Expected date of completion | Deliverables |
| 1 | Create a UI mock up | 18 Feb | UI Screens for all the given requirements |
| 2 | Create DB design | 25 Feb | DB which supports the requirement |
| 3 | Create Technical Design | 4 Mar | Design the classes and interfaces for the application |
| 4 | Documentation | 11 Mar | Create initial versions of end user and developer documentation |
| 5 | Steering Meeting | 11 Mar | Request approval from stakeholders to exit from design phase and enter into implementation phase |
| 6 | Implement feedback from Stakeholders | 18 Mar | Perform above tasks and request feedback again, the activity continues till stakeholders give a go ahead |
| 7 | Implement DB Schema | 25 Mar | Create tables, triggers and stored procedures for each entity |
| 8 | Implement the classes | 1 April | Create classes in application layer for each entity in DB |
| 9 | Implement services | 4 April | Create backend interfaces and its functionality for service |
| 10 | Implement Web UI | 8 April | Implement UI screens and controllers |
| 11 | Integration of Web UI and Backend | 12 April | Integrate Web UI controllers with the backend service |
| 12 | Developer Testing | 18 April | Perform end to end tests and provide bug fixes |
| 13 | Steering Meeting | 18 April | Show version 1 of the application to stakeholders |
| 14 | Acceptance Testing | 17 Jun | Invite early adopters, internal members and stakeholders to use the application and request for feedback |
| 15 | Provide Bug fixes | 17 Jun | Fix issues reported during acceptance tests, continue till zero bugs found |
| 16 | Steering Meeting | 17 Jun | Request approval from stakeholders to exit from implementation phase and enter into phased rollout |
| 17 | Roll out | 18 Jun | Execute phased roll out to production team locations – Bangalore (India), Rot (Germany), Vancouver (Canada), Moscow (Russia) and request for volunteers who would act as knowledge multipliers |
| 18 | Enable Knowledge Multipliers, other developers | 26 Jun | Give detailed info session to knowledge multipliers on application usage and developer info session to remaining members of development team |
| 19 | Documentation | 26 Jun | Update end user and developer documentation for missing information (if any). |
| 20 | Steering Meeting (Project Closure) | 27 Jun | Update stakeholders about the project |

**Completed Work**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Task Name | Expected date of completion | Deliverables |
| 1 | Create a UI mock up | 18 Feb | UI Screens for all the given requirements |
| 2 | Create DB design | 25 Feb | DB which supports the requirement |
| 3 | Create Technical Design | 4 Mar | Design the classes and interfaces for the application |
| 4 | Documentation | 11 Mar | Create initial versions of end user and developer documentation |
| 5 | Steering Meeting | 11 Mar | Request approval from stakeholders to exit from design phase and enter into implementation phase |
| 6 | Implement feedback from Stakeholders | 18 Mar | Perform above tasks and request feedback again, the activity continues till stakeholders give a go ahead |
| 7 | Implement DB Schema | 25 Mar | Create tables, triggers and stored procedures for each entity |
| 8 | Implement the classes | 1 April | Create classes in application layer for each entity in DB |
| 9 | Implement services | 4 April | Create backend interfaces and its functionality for service |
| 10 | Implement Web UI | 8 April | Implement UI screens and controllers |
| 11 | Integration of Web UI and Backend | 12 April | Integrate Web UI controllers with the backend service |
| 12 | Developer Testing | 18 April | Perform end to end tests and provide bug fixes |
| 13 | Steering Meeting | 18 April | Show version 1 of the application to stakeholders |

# Plan for Remainder of the project

|  |  |  |  |
| --- | --- | --- | --- |
| # | Task Name | Expected date of completion | Deliverables |
| 1 | Acceptance Testing | 17 Jun | Invite early adopters, internal members and stakeholders to use the application and request for feedback |
| 2 | Provide Bug fixes | 17 Jun | Fix issues reported during acceptance tests, continue till zero bugs found |
| 3 | Steering Meeting | 17 Jun | Request approval from stakeholders to exit from implementation phase and enter into phased rollout |
| 4 | Roll out | 18 Jun | Execute phased roll out to production team locations – Bangalore (India), Rot (Germany), Vancouver (Canada), Moscow (Russia) and request for volunteers who would act as knowledge multipliers |
| 5 | Enable Knowledge Multipliers, other developers | 26 Jun | Give detailed info session to knowledge multipliers on application usage and developer info session to remaining members of development team |
| 6 | Documentation | 26 Jun | Update end user and developer documentation for missing information (if any). |
| 7 | Steering Meeting (Project Closure) | 27 Jun | Update stakeholders about the project |

**Date: 4th April 2019**

**References**

* Frontend AngularJS - <https://angularjs.org/> (last cited on 4th April 2019)
* Design Material Ui - <https://material.angularjs.org/latest/> (last cited on 4th April 2019)
* Backend SQL Server - <https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017> (last cited on 4th April 2019)
* Controller C# - <https://docs.microsoft.com/en-us/dotnet/csharp/> (last cited on 4th April 2019)
* Backend VB classes - <https://docs.microsoft.com/en-us/dotnet/> (last cited on 4th April 2019)
* Cloud Native Computing Foundation - <https://www.cncf.io/> (last cited on 4th April 2019)