# Stash Proposal

# Problem statement

* No central code repository
* No sufficient code quality check points
* No tracking of changes

# Problem Resolutions

1. No central code repository
   1. Stash provides a central code repository where user can commit and push there changes to a remote location as well as local Machine.
   2. For Each commit user needs to push the changes to remote or user can make its local commit
   3. Central code is not affected until developer merge the code and the code is not merged until review.
2. No sufficient code quality check points
   1. While merging the code, Code review can be made mandatory.
3. No tracking of changes
   1. Tracking of changes can be done using either stash online versioning or can use any other tool like Source tree.
   2. Each commit contains a meaningful Message so tracking can be made easy.

09ju8

# Apttus Scenarios

* 90% case 1 developer at a time per project
* 80% case developer shuffling (multiple developer involved at different time for same job)
* 80% case different developer co-located

# Why Tool?

At present we don’t use any standard Code management tool. This leads to many business problems. Following table shows issues we are facing with manual process and benefit of using tool. After evaluation of various tools, we are recommending Bitbucket as code repository tool.

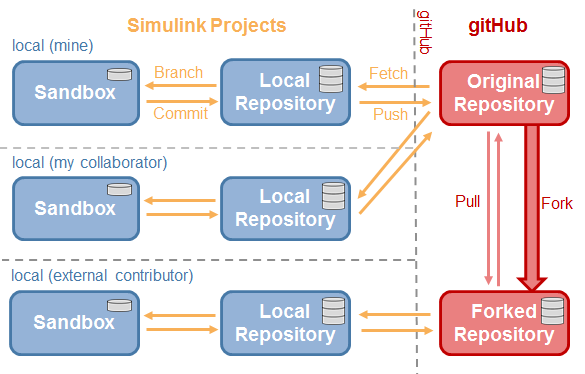
|  |  |  |
| --- | --- | --- |
|  | **Current Issues (without Tool)** | **Benefits of Tool (Bitbucket)** |
| **Code Reusability** | * Huge duplication of code across projects | * Shared Code repository on cloud where one can search & reuse existing code |
| **Code Quality** | * Controlling code quality is very difficult * Code review is too manual | * Code review by seeing only modified code * Can control Partner’s code quality as well * Future possibility of integrating Static code analysis tools |
| **History and Check points** | * All changes are overridden so no track of history of changes * In multiple people on same code scenario, good chances of overriding one others work | * Code history can be maintained & tracked * At any point of time can go to stable points * Due to version history, each changes can contains metadata like purpose of change, Issue ID, code reviewer, committed by etc. |
| **Parallel developments** | * In ‘multiple people working on same code’ scenario, good chances of overriding each other’s work | * Due to version history feature, parallel development is absolute possible |

# Cost details

|  |  |  |  |
| --- | --- | --- | --- |
| **Options** | **Option A** | **Option B** | **Option C** |
| Total Cost **/ Year** | **2400** $ | **1200** $ | **600** $ |
| Number of licenses (= No. of users) | Unlimited | 100 | 50 |
| Who will be able to use it in Apttus?  (Based on current headcounts) | Anyone | All Devs + All TAs + 50% SAs + 50% CEs | All Devs + All TAs |

# How it works

Following diagram shows how flow works among SalesForce environment, local machine (client) & Bitbucket server



# Rollout plan

Below are major components for rollout of this tool

## Pilot Implementation

## Training Plan

Following table shows training plan

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | Training | Details | Duration |
| 1 | Bitbucket tool | Basic demo of Bitbucket tool | 30 mins |
| 2 | Eclipse IDE | Training to show usage of Eclipse IDE | 30 mins |
| 3 | Plug-ins | Demo of Plug-in | 30 mins |
| 4 | Overall System | Basic guideline how one should work with tool (Dos & DONTs) | 30 mins |
|  |  |  |  |

# What Next

Integration with JIRA, once JIRA is rolled out

# Issue with Tools

* Training overheads – How much time?
* Process overheads (push at two places) – how much additional time required?
* Sync between SFDC & Code Repository may be challenge & risk