



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Agenda

Continuous Integration

- Continuous Integration
- Prerequisites for Continuous Integration Version Control
- Continuous Integration Practices

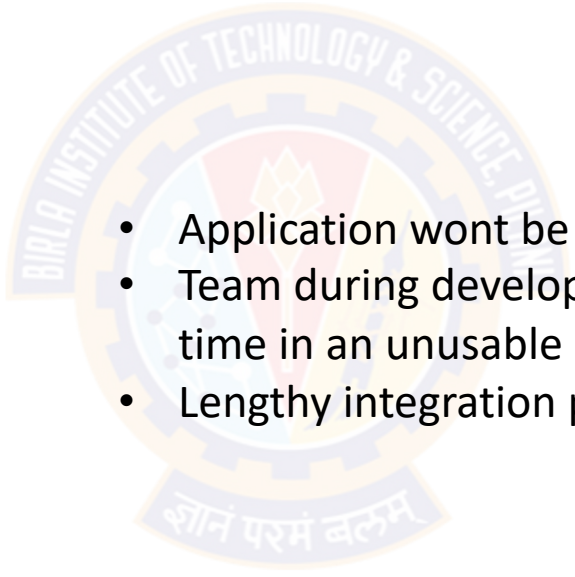


Continuous Integration

Continuous integration (CI)

- Process of integrating new code written by developers with a mainline or “master” branch frequently throughout the day

“Nobody is interested in trying to run the whole application until it is finished”

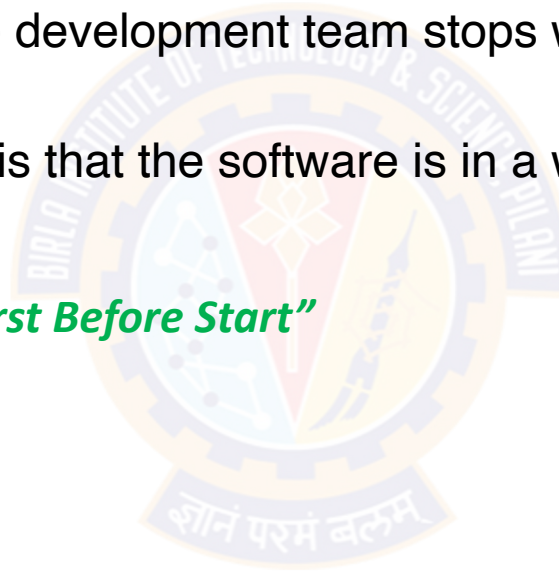
- 
- Application won't be in a working state
 - Team during development spends significant proportion of time in an unusable state
 - Lengthy integration phases at the end of development

Continuous Integration

Continuous integration requires

- Every time somebody commits any change, the entire application is built and a comprehensive set of automated tests is run against it
- If the build or test process fails, the development team stops whatever they are doing and fixes the problem immediately
- The goal of continuous integration is that the software is in a working state all the time

In simple words we can say “Finish First Before Start”



Implementing Continuous Integration

Pre-requisites

1. Version Control



2. An Automated Build



3. Agreement of the Team



Continuous Integration Pre-requisite

Agreement of the Team

- This is more about People & Culture
- Continuous integration is a practice, not a tool
- It requires a degree of commitment and discipline from your development team or people involved
- As said “Fix first before Proceed”: Need everyone to check in small incremental changes frequently to mainline and agree that the highest priority task on the project is to fix any change that breaks the application
- If people don't adopt the discipline necessary for it to work, attempts at continuous integration will not lead to the improvement in quality that you hope for



Continuous Integration

CI Tools

- Open Source :
 - Jenkins
 - Cruise Control
 - GitLab CI
 - GitHub Actions
- Commercial:
 - ThoughtWorks Studios
 - TeamCity by JetBrains
 - Bamboo by Atlassian
 - BuildForge by IBM



How it was before Continuous Integration

Just a glance !!!

- Nightly Build 😞



Note: this strategy will not be a good Idea when you have a geographically dispersed team working on a common codebase from different time zones

Continuous Integration

Pre-requisite & Best Practices

Prerequisite

Check In Regularly -> frequent check-ins

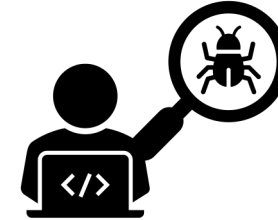


Create a Comprehensive Automated Test Suite

Unit Test

Component Test

Acceptance Test



Will provide extremely high level of confidence that any introduced change has not broken existing functionality

Keep the Build and Test Process Short

Standard Recommendation:

10 min is Good

5 min is Better

90 Sec is IDEAL

Managing Your Development Workspace

local Development Workspace must be replica of Production

Continuous Integration

Pre-requisite & Best Practices

Best Practices

Don't Check In on a Broken Build

What if we do Check In on Broken Build:

If any new check-in or build trigger during broken state will take much longer time to fix

Frequent broken build will encourage team not to care much about working condition

Commit locally than direct to Production

Wait for Commit Tests to Pass before Moving On

At time of Check-in, you should monitor the build progress
Never go home with broken build

Always Be Prepared to Revert to the Previous Revision

The previous revision was good because; you don't check in on a broken build

Continuous Integration

Scenario 1

Friday at 5.30 PM; your build is fail.

- You will be leaving late, and try to fix it
- You can revert changes
- You can leave the build broken

What Option you will opt here?



✗ Stay late to fix the build after working hours

Check in early enough so you have helping hands around
If it is late to Check In then Save your Check in for Next Day

Make rule of not checking in less than an hour before the end of work

Best Solution if you are alone then:

Revert it from your Source Control

Leaving Broken Build:

- On Monday your memory will no longer be fresh
- It will take you significantly longer to understand the problem and fix it
- Everyone at work will yell at you
- If you are late on Monday; be ready to answer a number of calls
- Not the least your name will be mud

Continuous Integration

Scenario 2

If you try to revert every time then; how can you make progress?

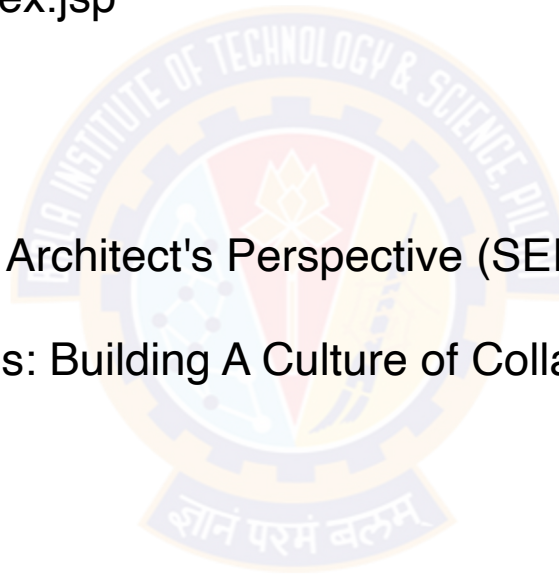


Time Boxing

- Establish a team rule
- When the build breaks on check-in, try to fix it for ten minutes or any approximate time your environment can bare
- However it should not be so long
- If, after ten minutes, you aren't finished with the solution, revert to the previous version from your version control system

CS 8 and 9

- Chapter 4, from DevOps: A Software Architect's Perspective (SEI Series in Software Engineering) by Len Bass, Ingo Weber, Liming Zhu ,
- <https://www.seleniumhq.org/docs/index.jsp>
- <https://www.sonarsource.com>
- <https://docs.sonarqube.org>
- Chapter 5, from DevOps: A Software Architect's Perspective (SEI Series in Software Engineering) by Len Bass, Ingo Weber, Liming Zhu ,
- Chapter 11,12, from Effective DevOps: Building A Culture of Collaboration, Affinity, and Tooling at Scale by Jennifer Davis





Q&A



Thank You!

In our next session: