IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Impact Factor: 4.819

Volume 5, Issue 2, May 2021

# Cloud Testing Perspectives with Continuous Integration

**Correa Bosco Hilary**
Department of Information Technology (MSc. IT Part I)
Sir Sitaram and Lady Shantabai Patkar College of Arts and Science, Mumbai, India

**Abstract:** *One of the most critical activities in the software development process is software testing. Depending on the ways of testing , software testing can be implemented at any time in the development process. Cloud computing has emerged as a new technology across the cooperates that impacts several different research fields, including software testing. Testing the applications has their own testing tools , challenges and testing methodologies. In this paper we provide an overview regarding cloud testing trends, technologies, challenges and the comparison of tools for testing in the cloud.*

**Keywords:** Cloud Computing, Cloud Testing, Continuous integration, Continuous delivery , STLC, TaaS, Functional Testing, Performance Testing

## I. Introduction

Software testing is one of the significant activities in the span of software development lifecycle . It determines whether the requirements are met , completeness and quality of the software product. Usually, software testing is done internally using the infrastructure that already exists within the organizations. Testing software requires costly machines, storage and network devices only for a limited time. Some computing resources are not valuable after testing, and thus incurring extra cost on budget. To ensure a reliable service, the service providers must have to test their services on all ( most off ) platforms. Over time, the software testing function has become a complex activity for enterprises due to increased challenges like rise of microservices, high costs to simulate and security. Developers are practicing new continuous integration techniques [CI] and merging their edits back to the main branch i.e. the testing or staging code as often as possible. These developer's changes are then validated by creating a build and running automated tests against the build in the cloud. By doing so, you avoid regression bugs that can happen when waiting for release day to merge changes into the release branch. Thus reducing the manual testing efforts of the QA Engineer. Continuous delivery [CD] is an addition to continuous integration since it automatically deploys all code changes in the test and production environment after the build phase.

## II. Continuous Integration

### 2.1 What is Continuous Integration?

Continuous integration [CI] is a development methodology in which developers regularly integrate their code into a shared repository, preferably several times a day. Each integration can be verified with automated builds and tests. Automated testing isn't strictly a part of CI, but one of the main benefits of regular integration is that errors can be caught quickly and easily. Each change introduced is usually small, so you can quickly identify the specific change that introduced the defect.In recent years, CI has been one of the best practices for software development and is governed by many important principles such as revision control, build automation, and test automation. You can deploy the app at any time. You can also automatically push your main code base into production as new changes are introduced. This allows you to move equipment quickly while maintaining high quality standards that can be tested automatically.

### 3.2 Why use Continuous Integration?

- Frequent mergers will reduce merger conflicts. Anyone who has ever struggled with git merge knows the pain of merge conflicts. Sometimes these problems are easy to fix, and sometimes it requires major changes.

- Developers get quick feedback, especially regarding bugs they need to fix. This means the developer won't have to fix the code he / she worked on weeks ago.
- Everyone is always working on the latest codebase. This greatly reduces the risk of problems with complex software and allows developers to easily see what other users are doing.
- All changes to the back-end environment are available to front-end developers.
- This allows for continuous deployment. Even if you're not doing this as a business, putting your code in a deployment location is a definite advantage.
- Product managers can also benefit because they get to see the latest product changes as it happens.

### 2.3 Requirements for CI

#### A. Code Repository

What is a repository ? - It provides version control options to track all changes to your code and ensure you know who did something to your code. You can also revert to the previous version of the code. Simplifies the process of unifying changes from developers' collaboration. Simplifies the process of integrating co-developer changes. It provides and promotes the principles of teamwork because multiple developers can collaborate on the same project, module, and the same lines of code. It also keeps statistics and analyzes code changes.
Some popular options for code repository hosting are:

**1. GitHub**

Seriously, this is a valid option but not open source. But there's nothing wrong with keeping your project on GitHub and taking a wait-and-see perspective. It's still the largest community website for software development, and it still has some of the best tools for issue tracking, code review, continuous integration, and general code management.

**2. GitLab**

GitLab is probably the leading contender when it comes to alternative code platforms for github. It's fully open source. You can host your code right on GitLab's site much like you would on GitHub, but you can also choose to self-host a GitLab instance of your own on your own server and have hundred percent control over who has access to everything there and how things are managed.

**3. Bitbucket**

Bitbucket has been around for many years. In some ways, it could serve as a looking glass into the future of GitHub. It's still a commercial platform like GitHub, but it's far from being a startup, and it's on pretty stable footing, organizationally speaking.
Some commonly used git terminologies are :

- Branch - A branch is a version of the repository that diverges from the main working project.
- Master - Master is a naming convention for Git branches. This is the default branch for Git.
- Merge - Merging is a process to put a forked history back together.
- Pull/Pull Request - The term pull is used to get data from a source.
- Push - The term push refers to uploading content from a local repository to a remote repository.
- Clone - It is used to make a copy of the target repository or clone it.
- Stashing - The git stash command enables you to switch branch without committing the current branch.
- Git Revert - used to revert some commits.

#### B. Single-Action Builds

You must be able to launch your build process with a single action, such as a CLI command, mouse click or by automatically monitoring code commits. It is used to continuously build and test software projects, making it easier for developers to integrate changes to the project, and making it easier and faster for users to obtain a new build. You can also continue to deliver software by integrating with a variety of deployment and testing technologies. Automation can help organisations speed up the software development process. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.

Some popular automation build solutions are:

**1. Jenkins**

Jenkins is an open source Continuous Integration server capable of orchestrating a set of actions that help achieve continuous (as well as) integration in an automated manner.. Jenkins is free to use and is written in Java. Jenkins is a widely used application around the world that has around 3,00,000 installations and growing day by day.

**2. Atlassian Bamboo**

Bamboo is a continuous integration server that automates the management of software application releases, thus creating a continuous delivery pipeline. Bamboo covers feature building and testing, version allocation, version tagging, and deploying and activating new versions in production.

**3. Circle CI**

Circle CI lets you automate your entire process, from code creation to testing and deployment. CircleCI allows automation across the user's pipeline, from code building, testing to deployment. You can integrate Circle CI with GitHub, GitHub Enterprise, and Bitbucket to create builds as new lines of code are released. CircleCI also hosts continuous integration under the cloud-managed option or runs behind a firewall on private infrastructure.

**4. TeamCity**

TeamCity is a JetBrains's build management and continuous integration server. TeamCity is a continuous integration tool that helps you build and implement different types of projects. TeamCity runs in a Java environment and integrates with Visual Studio and IDEs. The tool can be installed on both Windows and Linux servers, supports .NET and open-stack projects.

**5. Github Actions**

GitHub Actions is a task automation system fully integrated with GitHub. It was out of beta and reached general availability in 2019. It has the potential for many applications, including continuous integration and continuous deployment.

**C. Automated Tests**

Automated testing All tests should be automated so that they can be verified as soon as the build is complete. Test automation or test automation is a software testing technique performed using special test automation software tools to run a set of test cases. Comprehensive test suite covering everything from unit tests to comprehensive user interface and regression testing. Unit tests are, by definition, tiny . So, unit tests can be done rapidly. By contrast, a full set of regression tests can easily take a lot of time to complete. This is especially the case if you have to reset the system state between each test (as is often the case). Often your developers will push code several times a day. This means there simply isn't time to run all the tests for every build.

Some popular automation testing solutions are:

**1. Selenium**

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use many programming languages such as Java, C #, and Python. Create a Selenium test script. Since Selenium is a collection of different tools, it had different developers as well. Below are the key persons who made notable contributions to the Selenium Project Primarily, Selenium was created by Jason Huggins in 2004. There are a lot of frameworks written in various language bindings. Also, Selenium has a large community.

**D. Selenium in Cloud**

**1. Setting up the UI tests and switching to RemoteWebDriver**

We typically use the WebDriver class in the code to set up parameters like the required browser, incognito mode, download directory, browser driver, etc… By default, WebDriver assumes that we want to run the UI tests inside the host machine. But, here, we want to run them inside a docker container. So, we need to connect to the container IP address. For that, we need to use the RemoteWebDriver and specify the IP address and port number for the container

**2. Setting up Selenium Docker Containers**

Selenium can run in a single / stand-alone container or in a multi-container grid, Or we can use docker compose

**3. Horizontal Pod Autoscaling in Kubernetes**

We can scale these tests with kubernetes. And can do everything in the cloud. But this is a bit overdo at this point

**4. Cypress**

It is a next generation UI testing tool designed for the latest web. This tool addresses the critical things developers, and testers used to face when testing modern applications, E.g., synchronization bugs, the inconsistency of tests due to elements not visible or available or accessible.

**D. Cypress in Cloud**

**1. Setting up GitHub Actions**

Workflow is a configurable automated process made up of one or multiple Jobs (Jobs will be explained further down this post). Workflow files must be stored in the github/workflow directory of your repository and they are defined using YAML syntax (.yml or .yaml file extension).

**2. Schedule GH Actions Jobs to Run**

We can also set up our workflow to run on a schedule using POSIX cron syntax independent of any activity on the repository. Schedule events allow you to set a schedule for running a workflow.

**3. Send test results to Cypress Dashboard for Monitoring**

Cypress Dashboard is an option that provides detailed information and visual representation of tests, reports, status and events that occur at runtime and basically provides detailed information about what will happen when running the test.

**4. Online Paid Tools**

There are a lot of testing tools that are coming up with IAAS, TAAS and PAAS. Some tools are SauceLabs, Browser stack, test Project and AWS device farm.

## III. Testing in The Cloud

**3.1 Cloud Testing Cycle**

1. Define scope - Develop on scope and criteria, the technology that will be used.
2. Plan /design / integrate tests - The test team plans the strategy and approach for testing .
3. Execution - Real-time validation of product and finding bugs.
4. Report/ Bugs logs - reports, results are documented.

### 3.2 Types of Testing in Cloud

#### A. Unit Testing

Unit test is a type of software testing where individual units or components of a software are tested. The purpose is to verify that each unit of software code works as intended. Unit Testing is done during the development phase of an application by the developers.

#### B. Functional Testing

Functional testing of both internet and non-internet applications can be performed using cloud testing. The process of verification against specifications or system requirements is carried out in the cloud instead of on-site software testing. Functional Testing can also be divided in two parts: API and UI based testing. User interface test. A testing technique used to determine the presence of defects in the product / software under test using the graphical user interface [GUI]. API (Application Programming Interface) is a computing interface which enables communication and data exchange between two separate software systems.API testing is a type of software test that validates an application programming interface (API). The purpose of API testing is to verify the functionality, reliability, performance, and security of the programming interfaces. In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the API, get output, and note down the system's response. API testing is very different from GUI testing and does not focus on how your application looks. It mainly focuses on the business logic layer of the software architecture.

#### C. Performance testing

Performance testing is carried out to check the system's performance under varying loads. It verifies scalability, speed, reliability etc. of the system or software application. In performance tests, the load limit is lower or higher than the cut-out threshold. It ensures that the system or software application performs properly.

#### D. Stress Testing

Stress testing is carried out to check the behavior of the system under the sudden increased load. Stress testing is the variant of the performance testing. It only checks the stability of the system or software application. In stress testing, the limit of load is above the threshold of break.

## IV. Advantage And Disadvantages Of CI And Cloud Testing

### 4.1 Advantages of CI

#### A. Smaller Code Changes

One of the technical advantages of continuous integration and continuous delivery is that it allows you to integrate small pieces of code at one time. These code changes are simpler and easier to use than larger pieces of code, so there are fewer issues that need to be addressed later.

#### B. Fault Isolations

Fault isolation refers to the practice of designing systems such that when an error occurs, the negative outcomes are limited in scope. Limiting the scope of the problem reduces the possibility of failure and makes the system easier to maintain.

#### C. Faster Mean Time To Resolution of Errors

MTTR measures the maintainability of repairable features and sets the average time to repair a broken feature. Basically, it helps you track the amount of time spent to recover from a failure.

#### D. More Test Reliability

Using CI/CD, test reliability increases due to the bite-size and changes introduced to the system, allowing for more correct positive and negative tests to be held.

**E. Faster Release Rate**

Failures are detected faster and as such, can be repaired faster, leading to increasing release rates. However, constant releases are possible only if the code is developed in a continuously developing system.

**4.2 Disadvantage of CI**

**A. Budget Constraints**

CI requires a significant initial investment. However, its automated aspects will save you time and money with every new build. Saving on the initial outlay is a false economy.

**B. Time Constraints**

Because of the initial time in setting up CI, projects with short lead times are not ideal candidates. As many projects are implemented on Agile, scrum, Kanban cycles.

**C. Skill set requirements / Difficult to Maintain**

Setting up a CI process requires a specific skill-set. You should think about if you have the time and money to hire that person. On top of that, it's not enough to just implement CI/CD practices; they require constant support. This can be hard for large organizations that offer diverse services.

**D. Graphical or Picture based Applications are Difficult to Automate**

## V. Conclusion

There are a lot of tools and technologies apart from these that allow you to deploy, test and maintain your software on the cloud. Also, there are a lot of advantages and disadvantages of using the CI pipeline in software development. There is no fixed answer of whether CI is good or bad? or whether you should use CI tools in your next big app , it's mostly like picking your own situations in which CI pipelines will save you time and effort. In situations in which you are doing just front end development using plain HTML, CSS , then NO, CI is not a tool for you. Or if you are reading data from the database directly without any processing even then CI is not the process of your choice. CI pipelines are most effective when there is an application that are based on API's, multiple and complex backends systems. Every tool/ process has a place to be used, Same with CI processes and tools.

## References

**[1]** https://www.functionize.com/blog/continuous-integration-how-it-affects-testing/
**[2]** https://dev.to/iphiee_oma/continuous-integration-for-end-to-end-tests-with-cypress-and-github-actions-19a5
**[3]** https://medium.com/@HoussemDellai/run-selenium-ui-tests-in-docker-containers-f41ae2796b8d
**[4]** https://www.intellias.com/the-pros-and-cons-of-ci-cd-for-fintech/
**[5]** https://www.functionize.com/blog/continuous-integration-how-it-affects-testing/
**[6]** Conventional Software Testing Vs. Cloud
**[7]** Testing by Mrs.A.Vanitha Katherine, Dr. K. Alagarsamy,
**[8]** Cognizant reports, ―Taking Testing to the Cloud. March 2011