How to Find Bugs in Your App

testbytes.net/blog/how-to-find-bugs-in-your-app
August 17, 2019
How To

Saturday August 17, 2019

Have you ever wondered about how to find bugs in your app? Do you possess testcases that you think is enough to trace out bugs? In this blog we have detailed about the effective steps that can be used to find bugs in your app.

1. Save the Code at Good Place

This first step is not only useful for tracking bugs. It has many other advantages that, truthfully, would give for an entire article. All the files that make up the source code of an application are not simple files that we should have saved in Dropbox or, worse, in a hard disk.

We should not save the source code of an application in places like Dropbox, or on a hard drive. They are pretty peculiar files.

They are files that undergo different changes over time and that as a whole have different versions. To coder, these versions of the code have to be easily accessible and must have an explanation of what they represent.

To understand each other, we are talking about different versions of your code files, representing different versions of your application in the market for example, version 1, version 1.1, and version 2.0.

To all this, we have to add that these files have to be easily accessible to all the people who are working on them. The owner of the code has to be in possession of these files, but he has to allow access in a secure way to the developer or developers working on it.

Also Read : ***How to Test an Ecommerce Website: Points To Remember***

The good news? There is a storage system that allows all this and much more that was designed precisely to save code. This system is called Git.

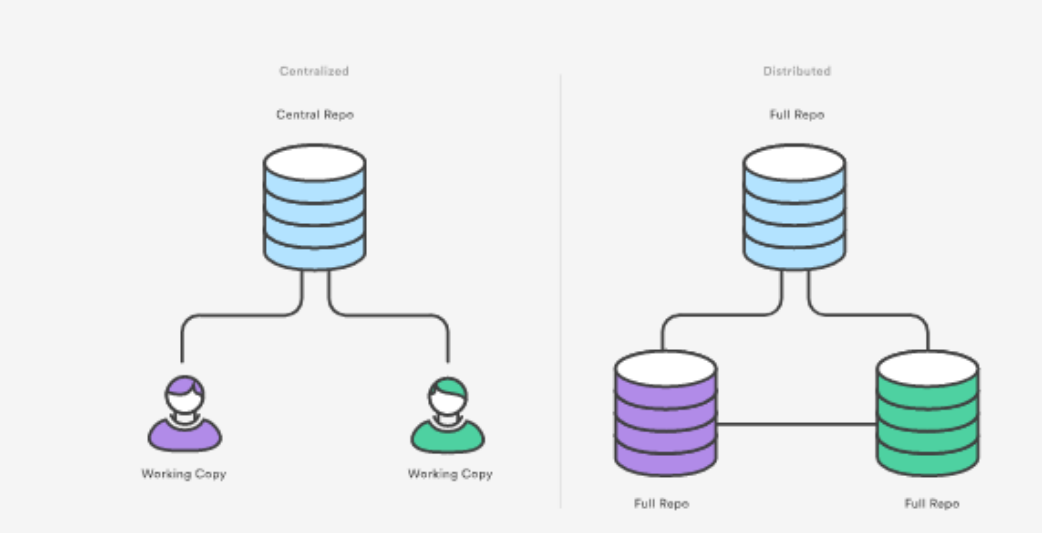
We could say that Git is not just a protocol, a standard or rules to follow, call it what you want. From here, there

are different platforms that implement it. The best known today, and the one that has more popularity, is Github.

Advantages of Git includes,

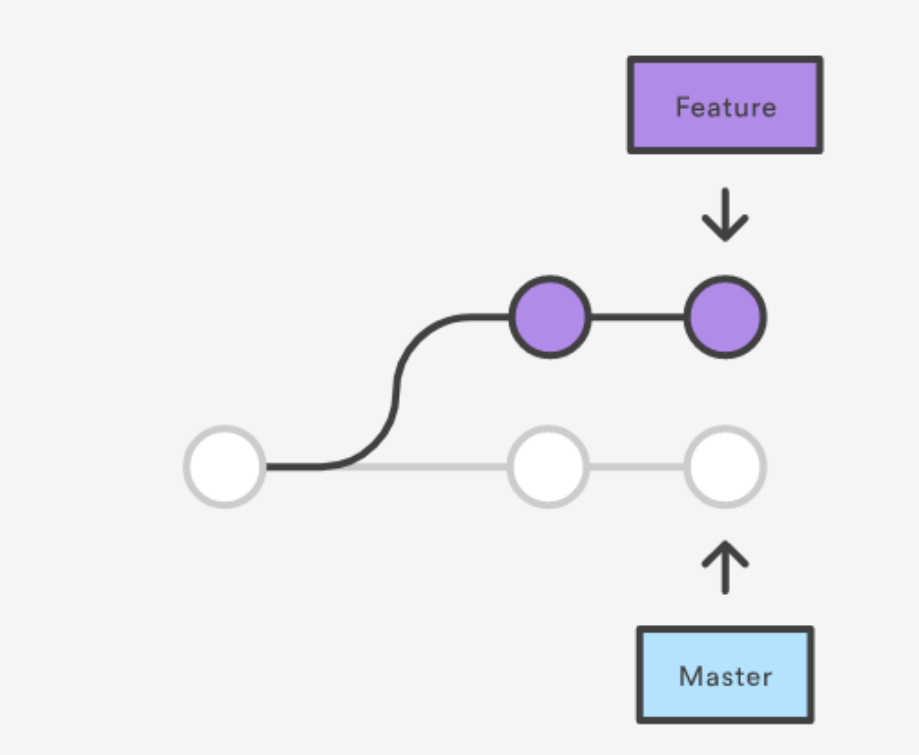
- **Distributed Development**

Each developer gets a local repository connected to a central repository does not require a network connection for commit creates a reliable environment



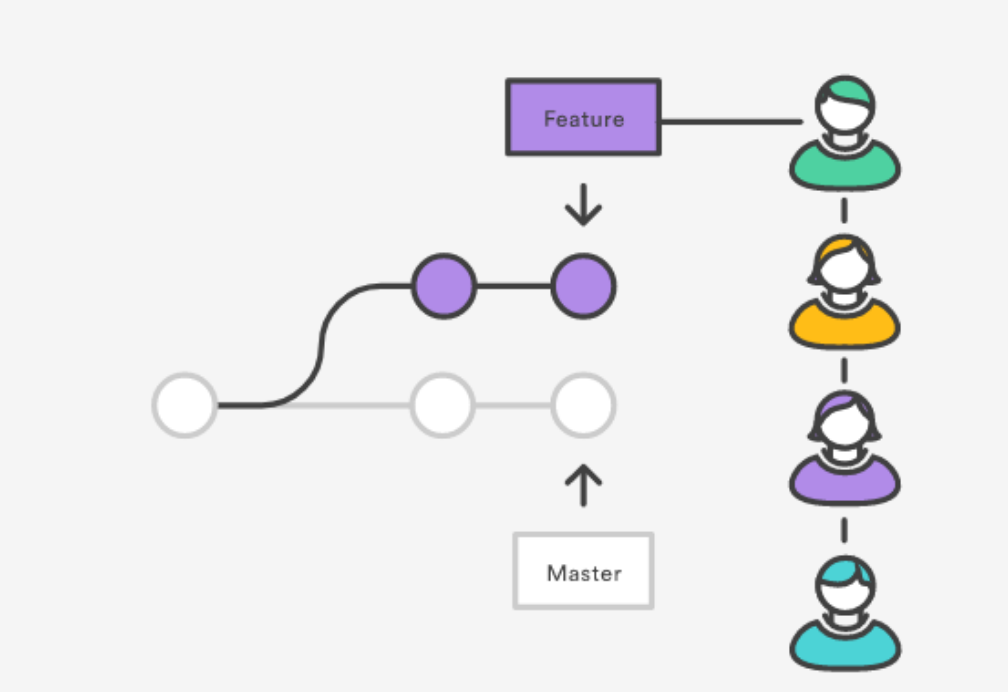
- **Workflow Branching Capability**

Easy to manage branched system that provides an isolated environment for development. Create a new branch if you wish to work on something new. Ensures master branch have production quality code helps to work as detailed as agile backlog



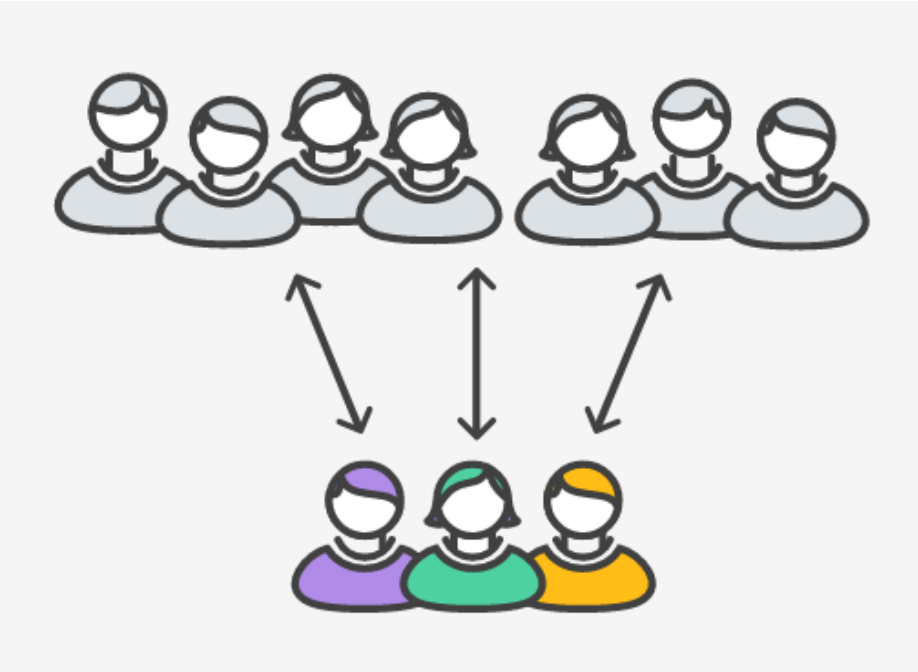
- **Merge One Branch with Another**

Merge one branch with another helps to pull request from one branch to entirely different one owing to that, changes can be tracked easily opens up the chances for discussion regarding their work before integrating with codebase inexperienced developers ruining entire project is minimal since pull request can be created as formal code review



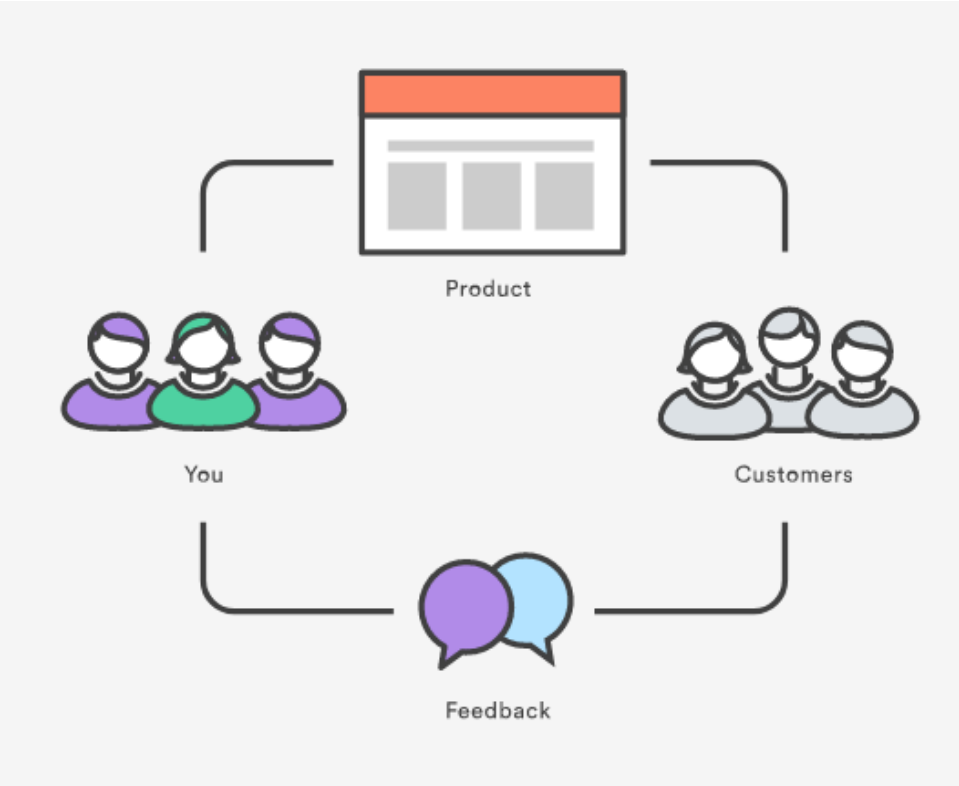
- **Community Support**

New addition to the team can get easily used to distributed development Easy to leverage 3rd party libraries others can fork their own source code



- **Faster Release Cycle**

Developers can share smaller changes frequently helps well with continuous delivery and integration environment deployment can be automated. You can build and deploy code to servers from the branch of your like



2. Bug Tracking

At this point I am going to assume that we have the code of application in a Github repository.

If we had everything well configured, we, as owners of the application, would have to be the owners of the Github repository and developers, would have to have access to it in order to make the necessary modifications.

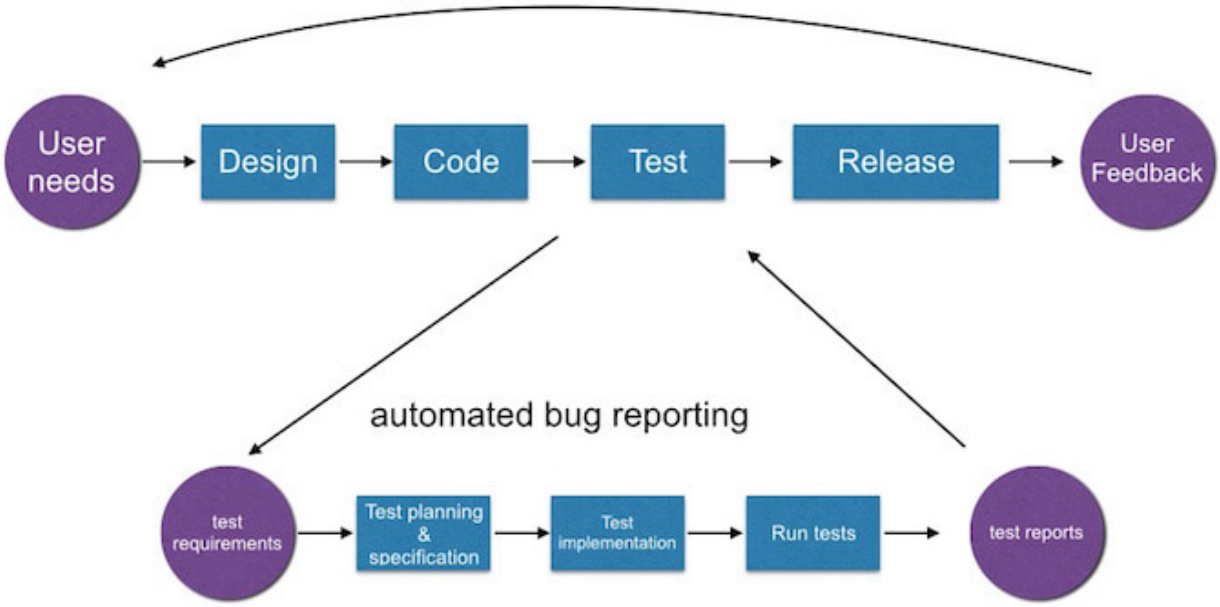
So far we have not seen any reason why Github can help us in tracking bugs, but peace of mind, we start now. Every repository in Github has a tab called bugs (bugs), and yes, we will use it for precisely to register the different bugs that happen in your application.

We can create a new bug by clicking on the New Issue button. From there we can add a title and a description. Now this bug will be registered in the repository until someone does something with it.

3. Automating the Bug Log

So far we have solved a small part of the problem. Now we have a registry of bugs to which, both we as owners of the code, and the developer that will be the person in charge of solving them, we can access.

However there is a great disadvantage. At this time the bug log is something manual. To track a bug, we would have to experience it ourselves and then go to the repository to register it and write down the details of it.



Does not seem like a good idea, right? Above all we have to take into account that there will be bugs that will only appear in certain specific cases, with which it is possible that, yes, they will be experienced by other users and not by ourselves.

It is also not necessary to mention that the process of going to create the bug manually to the repository is not an especially productive method.

Another tool to the rescue! In this case it is about Rollbar. Rollbar is a platform that is responsible for tracking bugs in any type of software. We can use it on websites, desktop programs, and programs running on a server and, of course, mobile applications.

4. Configuring Rollbar and Connecting with Github

How can we configure both tools to automate bug logging?

First of all we must create a project in Rollbar and configure it so that it is linked to the Github account. Specifically we will have to link new project to a repository. This repository will obviously be the one that contains all the source code of your application.

Also Read : ***How to Test a Bank ERP System***

Next, we will have a slightly more technical part that the developer must complete. You will have to implement the Rollbar library within the source code of the application. It's a quick job that can be ready in less than 1 day depending on the level of registration you want to follow.

What will happen once these steps are completed? Any bug, which happens to any user in your application, will be created automatically in the Issues section of Github repository. Automated bug logging!

5. Obtaining More information of Each Bug

Another of the problems we had without Rollbar that was the little information we had about a bug.

For anyone, with or without technical knowledge, it is very difficult to find out at a glance what is causing an application to fail. It does not matter if you have technical knowledge. Maybe you could know where the shots are going, but it would be almost impossible to know the cause at 100%.

6. Solving Bugs and Communication with the Developer

Well, now that we have seen the different tools, the configuration of the same and what benefits we will obtain, we will see what the day to day would be like to track the bugs of application.

In the first place, we should not have any person in charge of recording the bugs. Automatically Rollbar would create them as they happened.

The developer every X time would have to go consulting if there are new bugs and solves them. Once a bug has been solved and the code has been updated in the repository, the bug would be closed.

These are the standard factors that one asks for testing, for each of these clients, more information is required, such as on which network provider the bug was detected or the size of the screen where the bug was, etc.

Finding out bugs can be fun. Correct documentation along with corner cases will help you maximize the result.

Conclusion

These are the basic steps in finding bugs in the developmental phase of software/app once the development has been done, it's always better to do professional QA (Quality Analysis) with the help of a QA company to make the app as stable as it can get.