Background

Currently, the android release process is briefly divided into three steps (as shown in the roadmap diag)

1. **Final integrated build drop** for quality assurance. It usually takes 2-3 days. Developers need to raise merge requests to the reviewers before the predecided code freeze date while following the JIRA pipeline and GITLAB nomenclature *(IND: Feature: Project Nitro INDT-4166)*. It is an entirely automated process.
2. **Quality Assurance & bug reporting:** This takes around 5-7 days. The process is partially automated with some test cases.
3. **Application publishing and monitoring**: Below data of the last **ten** releases shows how app rollout works. The process is entirely manual, from reviewer filtering bugs to assigning a developer to release an incremental version.

Last 10 release cycle information

Release version	1% rollout start	100% rollout end	Total releases	Days to finish
17.00.00	29/01/2022	02/02/2022	2	4
17.10.00	09/02/2022	11/02/2022	1	2
17.20.00	24/02/2022	01/03/2022	2	6
17.30.00	10/03/2022	13/03/2022	1	3
17.40.00	23/03/2022	26/03/2022	2	3
17.50.00	07/04/2022	17/04/2022	4	10
17.60.00	28/04/2022	05/05/2022	3	6
17.70.00	12/05/2022	19/05/2022	5	7
17.80.00	25/05/2022	30/05/2022	3	5
17.90.00	10/06/2022	14/06/2022	2	4
18.00.00	23/06/2022	01/07/2022	5	8

We can draw the following conclusions from the above table:

- Average days to finish complete rollout (last 10 releases) : 5 - 6 days
- $Days\ to\ finish\ complete\ rollout \propto (Directly\ proportional\ to)\ Total\ incremental\ releases\ per\ version$

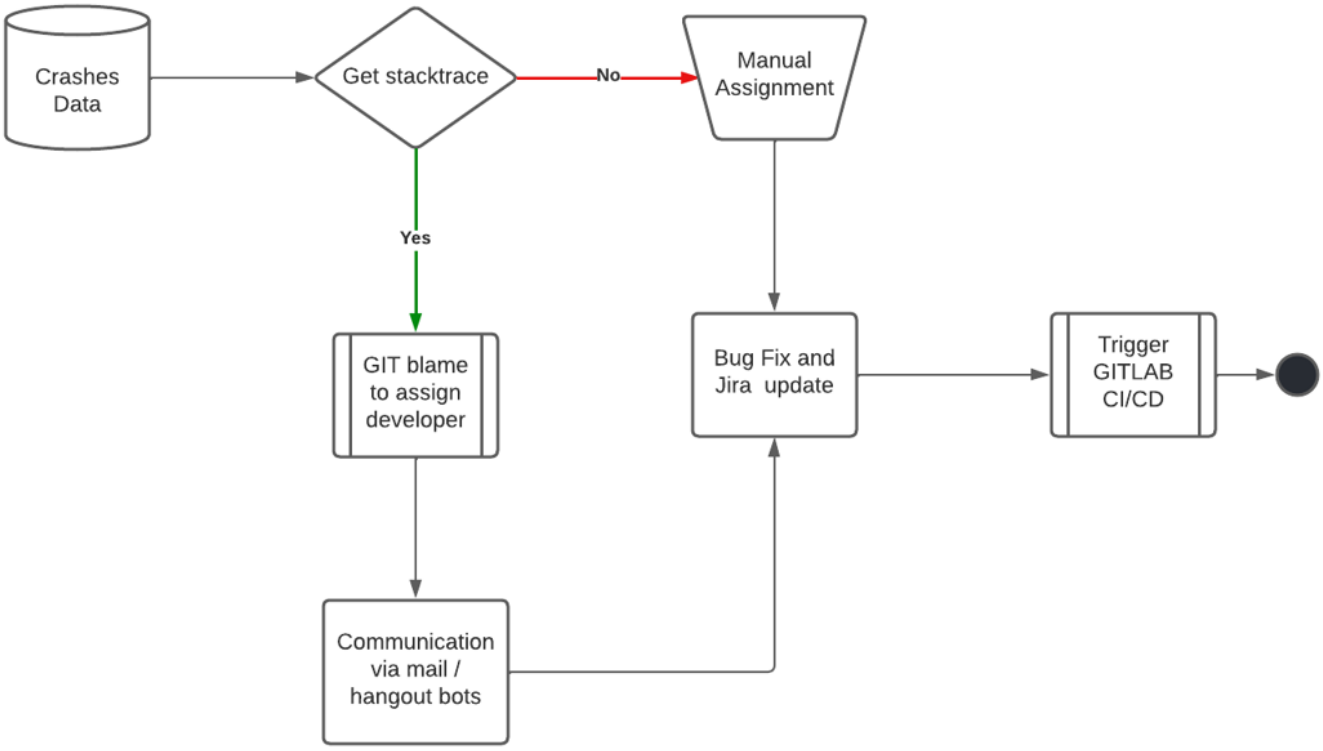
Goals / Use cases

➢ Improve the app maintenance process by automation which helps reviewer and developers to quickly release the bug fixes (Automatic bug assign) therefore reducing the *#Days to finish the complete rollout*

➢ There is no record of the bug fixes during the incremental releases. Since it usually doesn't go through the JIRA pipeline. Automation should keep track of each bug using JIRA. The developer needs to close that assigned JIRA by adding the final remarks *(ex, Added null check for programList property in search since not all services will have program feature)*

➢ It is sometimes possible to bypass crashes during the release process since it is a manual process, and these bugs may prove to be problematic in the future.

Implementation



Automation will be responsible for getting corrupt line number and file from stack trace and informing to developer by creating JIRA and communicating via hangouts.
Developer will be assigned through git annotate for that corrupted line.
It should be assigned developer's responsibility to work on that bug or reassigned that bug to the respective person.

Challenge : The biggest challenge right now is to get the crash data in the structured format and do analysis over that.

Possible Solutions:

- Pulling data from big query since both firebase or play console doesn't provide the data in a direct way
- Web crawling to generate the stack trace data

Additional information

This documentation aims to automate step 3 (refer background) of the android release process.

A few years back when CI/CD automation was completed for step 1 (Integrated build drop) the following milestones were achieved

Below is the graph which shows the percentage of the number of merge requests sent after the freeze date **(Automation started on 9.10.00)**