Part 3

# Scenario

You just joined a company with one developer who manages his code using GIT. There is also an AWS instance where the developer run tests and nothing else in terms of infrastructure. The company is growing fast and more developers will be joining the team soon and the CEO wants to ship some MVP ASAP.

# Question

Where would you begin? Please explain how you’d define what needs to be done and why, explaining the pros and cons of your approach. Assume the answer is for a technical person also familiar with software development cycle.

The objective of this question is to assess your understanding of the software development cycle. There are multiple correct answers with their own pros and cons, we just expect you to quickly present a plan and explain your it. You can also add questions you’d ask and what sort of decisions they would lead you to.

First of all, I would audit the existing code, in order to check if its structure is comprehensive and scalable. Normally, as projects grow, if the basis is not well structured and/or not using some frameworks that are commonly needed or advisable (e.g. Maven, SpringFramework, etc), they tend to get extremely complicated to manage and maintain. Moreover, the late adoption of those base frameworks turn out to be difficult to painful, normally costing a lot of refactoring and time.

After the auditing, we should decide if the code needs some refactoring and/or frameworks inclusion in order to be ready for having a lot of people working on it. Being already on GIT is nice, maybe just verify if the used service is GitHub in order to have the Pull Request functionality, which would be nice for later team code reviewing and code acceptance.

The approach I’d use following these actions would depend on what would happen first, the team growth or the MVP handing, and the MVP project complexity itself.

In case the timing for the MVP delivery happens first and the application itself is not too complex, I’d start to integrate the project specifications with a project management and tracking platform like JIRA to receive the stories with the features to implement and the bugs found. We may integrate this tool in an Agile/Scrum methodology later on when the team grows, but for now I would only use it to track the tasks needed to do, given that the team is of just 2 people and therefore easy to synchronize.

In case the team grows before the application delivery or the MVP to deliver is of some complexity, I’d also integrate JIRA platform to manage the project, but now in an Agile/Scrum methodology, using the sprints and daily meetings to account for the project progress.

With the team growth is also important to have a good test setup and a continuous integration server like Jenkins, making it easier the inclusion of the latest features and bug corrections for testing. The features would enter the main branch after passing the tests and having been reviewed by another developer, in a situation where we could use the Pull Request feature of GitHub and having the branch merges locked and limited to the product owner, in order to avoid unintended, non-reviewed merges.

The testing for the application should be taken into consideration to have the many automatic tests possible, in order to cover regression tests and therefore saving the team’s time on manual testing each time a new release is to be launched. I’d adopt an unit test framework like Junit and the policy of defining unit tests for each feature developed.