In my pipeline, there will be four stages[[1]](#footnote-1). These are:

* Source
* Build
* Test
* Deploy

First, the ‘Source’ stage refers to tracking changes made to the code base. Since I am using GitHub, the pipeline will be activated when I push a new commit, and subsequent phases will be executed.

The ‘Build’ stage deals with building the artifact and making sure that the correct plugins are implemented and that there are no build errors prior to test execution. For this task, I will use Jenkins as my tool for the following reasons: it is portable to all major platforms, errors are reported and can be fixed immediately, and “carries thousands of plugins which make CI/CD operations much simpler”[[2]](#footnote-2). My system uses Maven to build its tools, thus Jenkins can take advantage of this as it is able to integrate with Maven.

The ’Test’ stage includes the unit, integration, and system level tests that I have laid out in previous documents. The tool that will be used here is JUnit, as Jenkins can integrate with it to run tests automatically. JUnit is appropriate for this pipeline because not only is it supported by almost all IDEs, but it also can efficiently detect errors as well as being easy to read.

Finally, the ‘Deploy’ stage signals that the altered codebase has successfully built its plugins, passed all the given tests, and is ready to be deployed into a real-life environment. A deployment strategy that could be useful here is the ‘Blue-green deployment strategy’. This is a model used to transfer “traffic from a previous version to a new version”[[3]](#footnote-3), and “helps to reduce downtimes and risks”[[4]](#footnote-4). I thought that this particular model was advantageous to the design as my system deals with streams of users making requests and the system producing responses to them, both of which are essential and should not be lost. Since the handover from a ‘Blue’ environment to a ‘Green’ one is seamless, users will not experience any downtime and even if there is an error in the new environment, it is easy to rollback to a previous version and work on a fix.

Issues with the pipeline:

Although Jenkins is widely used, it is not the easiest to maintain and requires expert knowledge to operate it. This falls under personnel risk, as if an employee who is tasked with maintaining the server leaves, then it will be difficult for anyone else to do their job. This could be remedied by creating a document detailing the job in detail as well as having multiple employees undergo training to avoid an absence of talent.

A disadvantage to the Blue-Green deployment strategy is that due to the instant transfer from one environment to the other, user requests that occur at that exact time could be lost, which is something that definitely should be prevented. One potential fix for this issue is to put the new environment into a read-only mode first to ensure all requests are safely read, and then switching to read-write.

Levels of testing:

For this pipeline, the levels of testing required are as follows:

* Unit Testing
  + LngLat
  + Card
  + Model class
* Integration Testing:
  + Polygon
  + Graph
  + PathFinder
  + OrderChecker
* Performance Testing
* System Testing
  + App

Furthermore, some tests should be executed before others, i.e., there should be dependencies between tests. ‘x -> y’ signifies that test suite x should be run before y.

* LngLat -> Polygon, Graph, PathFinder -> App
* Card -> OrderChecker -> App

1. How to set up a Continuous Integration & Delivery Pipeline, Shanika Wickramasinghe, bmc blogs, <https://www.bmc.com/blogs/ci-cd-pipeline-setup/> [↑](#footnote-ref-1)
2. Maven vs Jenkins: Key differences, Shreya Bose, BrowserStack, <https://www.browserstack.com/guide/maven-vs-jenkins> [↑](#footnote-ref-2)
3. Blue-Green Deployment: An introduction, Stephen Watts, bmc blogs, <https://www.bmc.com/blogs/blue-green-deployment/> [↑](#footnote-ref-3)
4. Blue-Green Deployment: An introduction, Stephen Watts, bmc blogs, <https://www.bmc.com/blogs/blue-green-deployment/> [↑](#footnote-ref-4)