The tools for our investigation will be Jenkins, Gitlab CI and Travis CI. We will compare them in regards to usability and suitability for different scenarios. We will purely use MacOS as our operating system and use the tools provided for the Apple platform. For version handling we will use GitHub in sync with Jenkins and Travis, and GitLab for the GitLab CI service.

Of these tools 2 will be set up or partly set up on our local machine. For GitLab CI we will have a runner installed on our local machine, and for Jenkins we will have to install the jenkins server on the local machine. Since Travis CI is purely a cloud service, no installation will be needed except configuring and linking a Travis account to our GitHub account.

For Jenkins we start by downloading the latest stable build (2.164.2) from their website https://jenkins.io/download/
After installing Jenkins we access the Jenkins server on http://127.0.0.1:8080/ and set up a administrator user, and then install the recommended plugins. We also install the Build Pipeline plugin that will be used to make similar build process as we use in GitLab Ci and Travis CI. We then create a new item in jenkins for each of our steps we want to test (clean, build, test and deploy). We link these items to our GitHub account with our sample project. We have yet to find a way to trigger automatic Jenkins build when pushing to the GitHub repo. It is possible for GitHub to sniff the Jenkins server, but since we have dynamic IP on the Jenkins server we would like it the other way around, the same way that the GitLab runner sniffs the GitLab account.

For Travis CI we create an account on their website and connect it with the GitHub account and on GitHub grant access to Travis.
Travis will now automatically execute the steps we define in the .travis.yml file on the GitHub account and we use the same steps as in Jenkins (clean, build, test and deploy).

For GitLab CI we will use a runner on the local machine. This runner we install with the help of brew, with the command brew install gitlab-runner.
This runner is configured with the provided token on the GitLab account. Now when we push changes to the GitLab repo, the local runner on the machine will execute the steps defined in the .gitlab-ci.yml, and we use the same steps (clean, build, test and deploy).

We will need to find a suitable project that can be implemented on both Jenkins, Travis and Gitlab CI. As for now we only have the steps to be executed by the CI programs but not defined any actions yet.