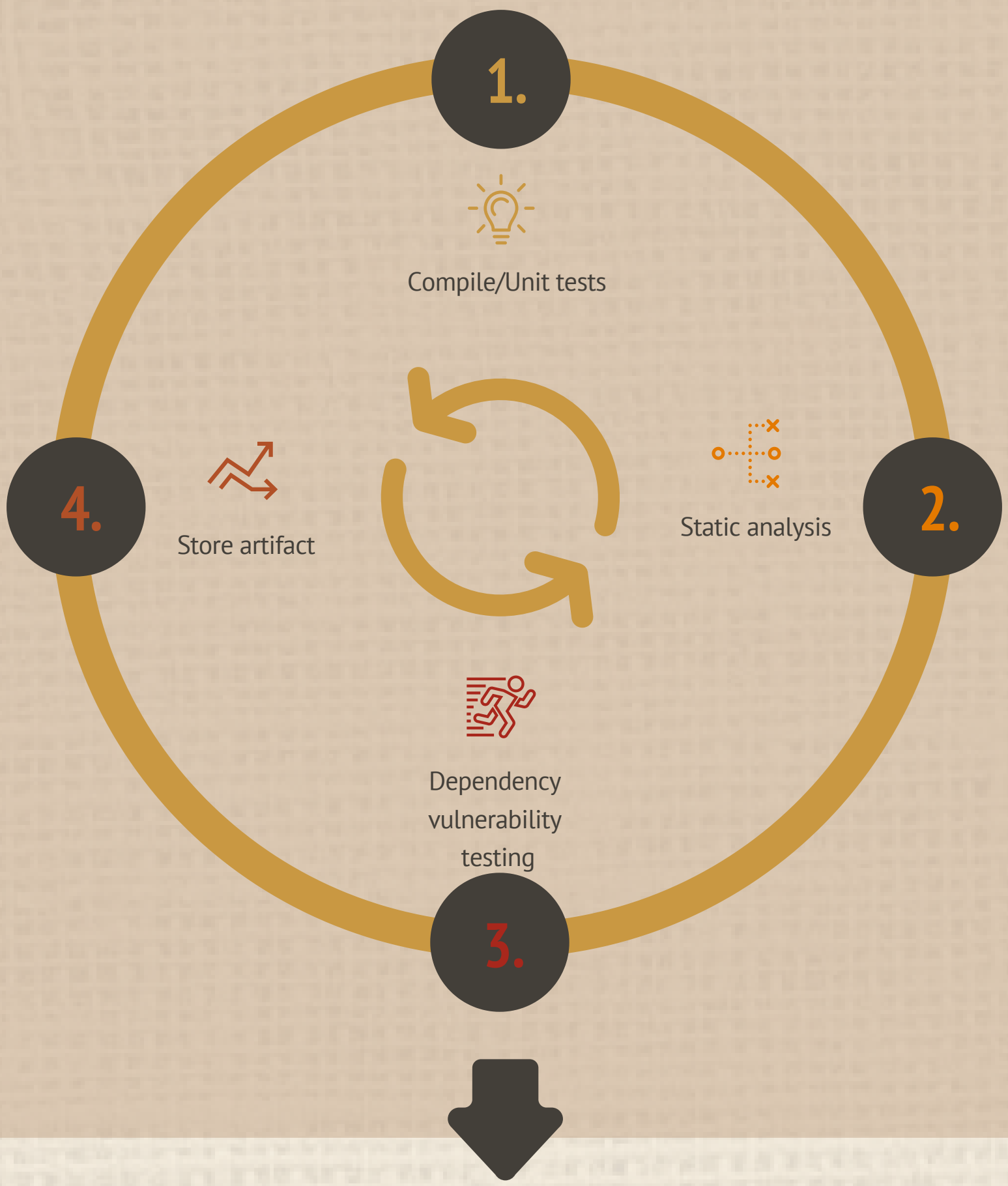
Continuous integration - Journey

The practice of merging all developers' working copies to a shared mainline several times a day. It's the process of "Making". Everything related to the code fits here, and it all culminates in the ultimate goal of CI: a high quality, deployable artifact! Some common CI-related phases might include:



Development Cost:

Writing automated tests for each feature or bug code and merging cost as often as possible.

Infraestructure Cost:

Configuring continuous integration server to monitor main repository in order to run tests every time code is pushed.

Development Gain:

Less bugs as code is tested constantly by automated tests and less context switches as devs are informed about broken builds, issues come up earlier.

Testing Costs Gain:

Running hundreds if tests by CI server just in seconds and decreasing testing time spent by QA team allowing them to focus on improvements and quality.

Continuous delivery/deployment - Journey

A software engineering approach in which the value is delivered frequently through automated deployments. Everything related to deploying the artifact fits here. It's the process of "Moving" the artifact from the shelf to the spotlight. Some common CD-related phases might include:

Delivery Cost:

Test suite should have with good coverage and deployments have to be automatic.

Deployment Cost:

Time spent in good quality suite tests, updated documentation process about deployments and coordination in releases with other departments.

Delivery Gain:

Less complex, faster and more often releases, that encourages to iterate faster.

Deployment Gain:

Faster development as deployment is triggered automatically in every change, less risky releases and easier to fix as deployments are done in small batches. Customers are able to see continuous improvements and changes in the application.