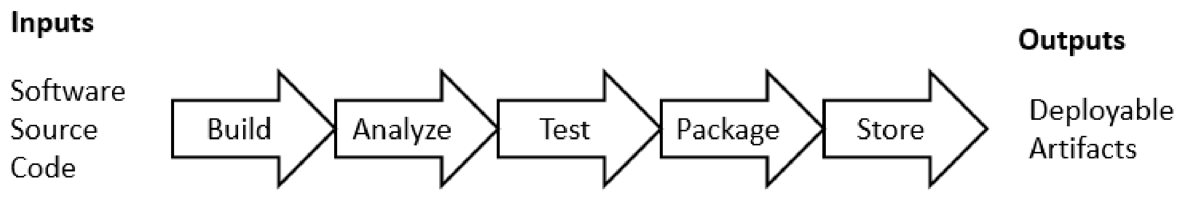
ACIT 4850 – Enterprise System Integration – Winter 2024

Lesson 5 – CI Pipelines with Jenkins

A CI pipeline consists of a series of stages that takes your source code and transforms it into a deployable arifact or set of artifacts. It will typically be an automated process so that it is repeatable and minimizes human error. A typical CI Pipeline may include the following stages:



The following describes each stage:

Build – For a statically typed language, this will include loading dependencies and compiling the software. For a dynamically typed language like Python, this may only include loading dependencies (i.e., 3rd party packages and modules) and running syntax checks on the code.

Analyze – This may include static code analysis, which is reviewing the code without actually running it. So it is like an automated code review in that it looks for common coding errors and alignment with coding conventions (i.e., naming, structure, etc.). We will be installing and integrating a static code analysis tool in a later lab.

Test – This is where automated unit tests are run to verify the code behaves as expected.

Package – This is where the built and verified source code is transformed into a deployable artifact (or multiple artifacts). A deployable artifact may be a Docker image, a Java JAR or WAR file, an EXE file, a zip file, etc.

Store – This is where the deployable artifacts are stored in a location where they can be versioned (i.e., given a unique number or timestamp) and easily accessed for either manual or automated deployment. This could simply be a file folder on a VM or network drive or a specialized artifact repository. We will be installing and integrating an artifact management tool in a later lab.

Each of the pipeline stages consists of one or steps. The steps may consist of shell commands or code with logic to perform the required tasks of the stage.

Please read the following for Lesson 5 to learn more about building pipelines in Jenkins:

- Jenkins Pipeline: https://jenkins.io/doc/book/pipeline/
- Pipeline Syntax: https://jenkins.io/doc/book/pipeline/syntax/

A basic Jenkins pipeline definition corresponding to our typically stages above, defined in a file called *Jenkinsfile*, would look like this:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                Some Steps Here
            }
        }
        stage('Analyze') {
            steps {
                Some Steps Here
             }
        }
        stage('Test') {
            steps {
                Some Steps Here
             }
        }
        stage('Package') {
            steps {
                Some Steps Here
             }
        }
        stage('Store') {
            steps {
                Some Steps Here
             }
        }
    }
}
```

With the Jenkins pipeline keywords:

- **pipeline** – The overall pipeline definition.
- **agent** – Specification for running the pipeline (i.e., Windows, Linux, etc).
- **stages** – Contains one or more stages in the pipeline.
- **stage** – Specific stage in the pipeline.
- **steps** – One or more actions to perform in the stage. Typically, these are shell commands or short Groovy scripts.

Unlike Python code, our Jenkinsfile based pipeline definitions use curly braces rather than indentation to define block of related code. So make sure you start each block with a { and end it with a }.