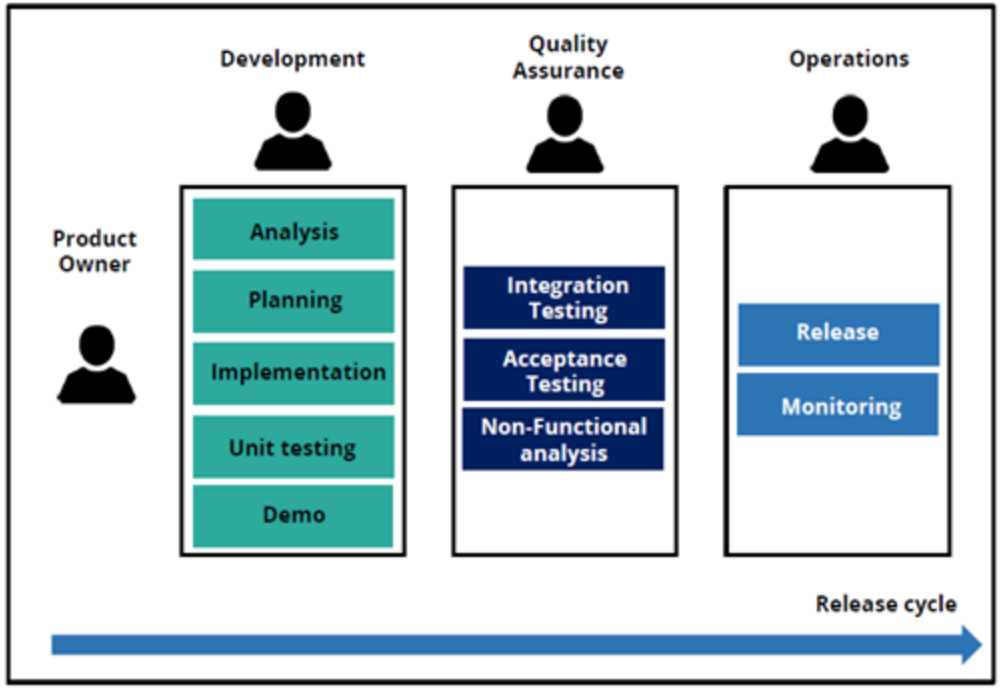
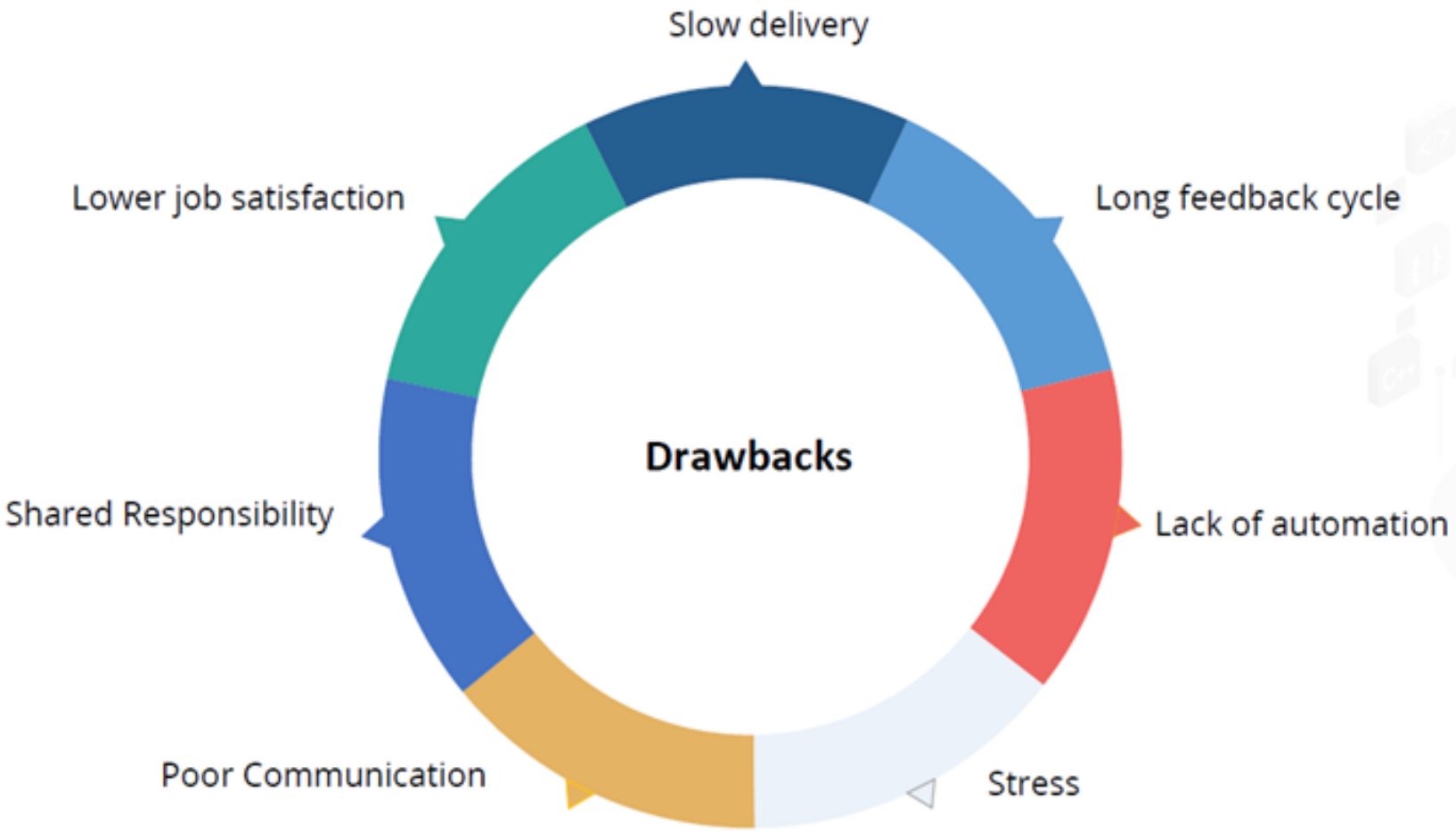
Sunitha C S

Traditional Delivery Process

- Any Delivery process starts with the Requirements defined by a customer and ends with the release to Production.
- The diagram shows the traditional delivery process.



Drawbacks


Slow delivery
Long feedback cycle
Lack of automation
Stress
Poor Communication
Shared Responsibility
Lower job satisfaction
Drawbacks

Continuous Integration & Continuous Delivery

CI & CD

What is continuous integration, delivery & deployment

"Continuous integration (CI) is the practice, in software engineering, of merging all developer working copies to a shared mainline several times a day."

"Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing, and releasing software faster and more frequently."

- Wikipedia

Continuous Integration, in its simplest form, involves a tool that monitors your version control system and automatically compiles and tests your application whenever a change is detected.

Continuous Integration lets you deploy the latest version of your application either automatically or as a one-click process.

Continuous Delivery is the next step of Continuous Integration. Your code is integrated and tested, and then it is ready to be deployed with one-click.

Automating your deployment eliminates the need for human intervention. Automating the deployment process lets you push every build that passes the tests into production.

The practice of automatically deploying every successful build directly into production is known as Continuous Deployment.

Advantages

- Continuous Integration automatically monitors the health of your codebase, code quality, and code coverage metrics.
- Technical debts are kept down and maintenance costs are low.
- Publicly-visible code quality metrics encourage developers to improve their code quality.
- Automated end-to-end acceptance tests provide a clear picture of the current state of development efforts.
- Continuous Integration reduces risk by providing faster feedback.
- CI tools are designed to help identify and fix integration and regression issues faster, resulting in fewer bugs and quicker delivery.
- CI helps simplify and accelerate delivery by automating the deployment process.
- Automating the deployment process helps get your software into the hands of the testers and end users faster.

- With Continuous Delivery, any successful build that has passed all the relevant automated tests and quality gates can *potentially* be deployed into production, and be in the hands of the end user within minutes.
- But this process is not **automatic**.
- It is the business, rather than IT that decides the best time to deliver the latest changes.

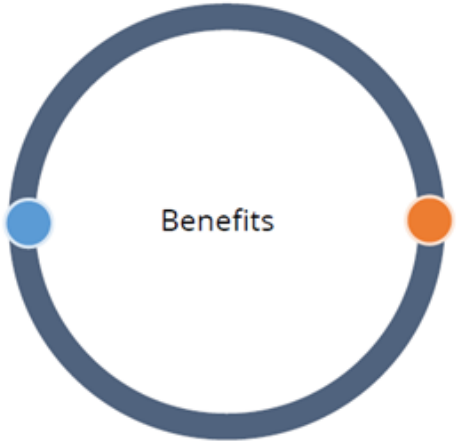
Fast delivery
Customers can use the product as soon as the development is complete.

Fast feedback cycle
Identifying bugs as soon as they are developed, combined with quick rollback strategy, keeps the production stable.

Low-risk releases
If you release on a daily basis, the process becomes repeatable and much safer.

Flexible release options
You can release the software without any additional time or cost spent in case of an immediate release.

Faster return on investment for each feature as it gets developed



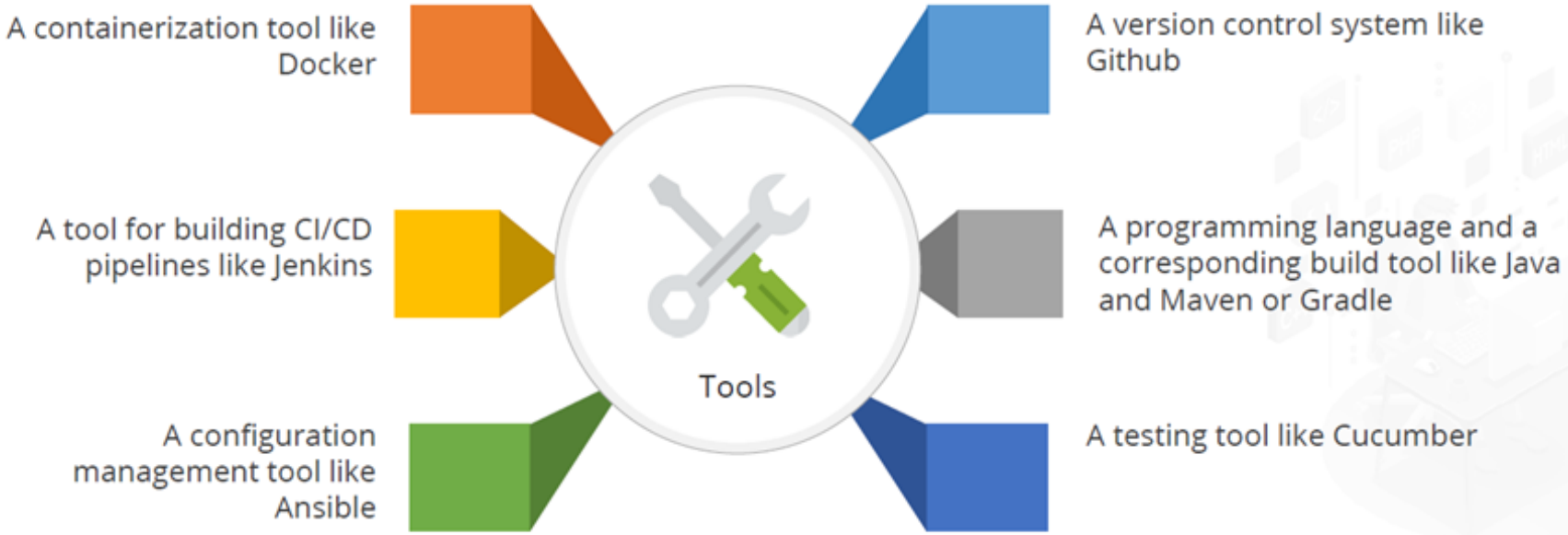
Faster feedback from end users on each new feature as it is released to production

Continuous Delivery Process Tools

There are a variety of tools available in the market for performing each of the operations involved in building a Continuous Deployment process.

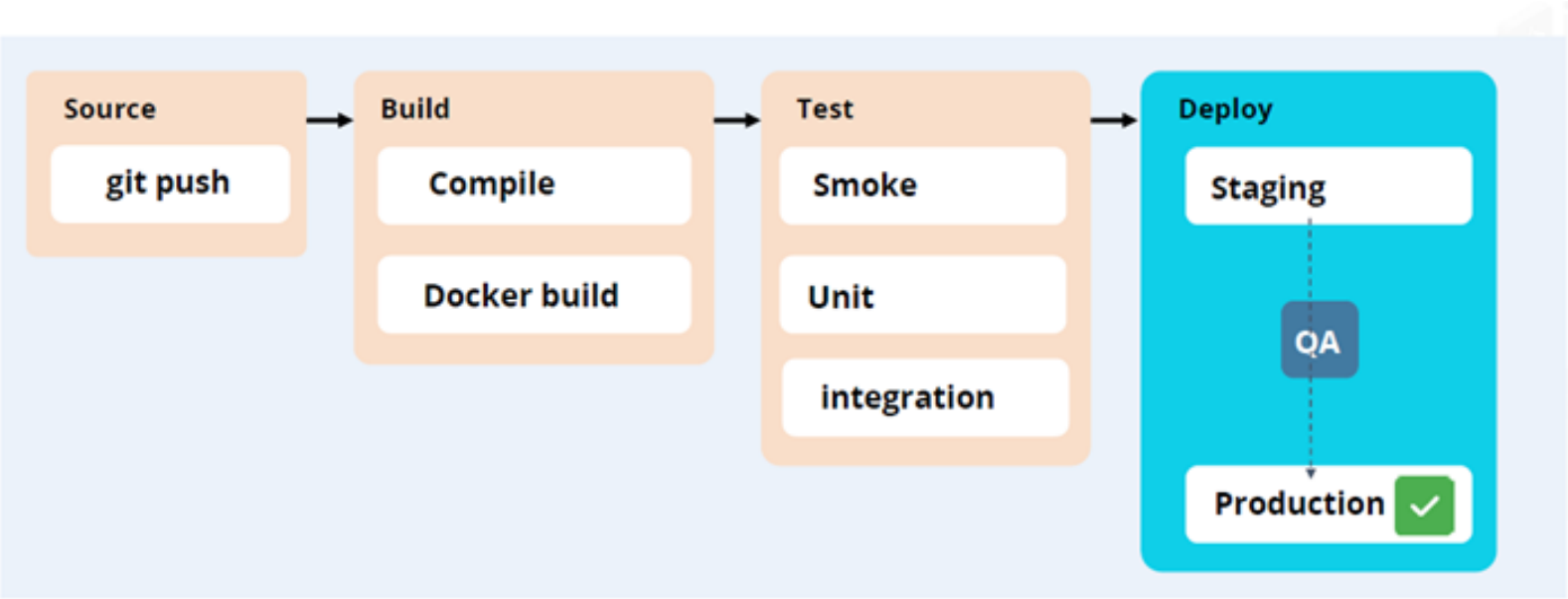
Any tool can be replaced with any other tool that plays the same role, depending on your environment.

- For example: Jenkins can be replaced with Atlassian Bamboo and Chef can be used instead of Ansible.



Stages of CI/CD Pipeline

A CI/CD pipeline is essentially a runnable specification of the steps that need to be performed in order to deliver a new version of a software product. A CI/CD pipeline usually has the following stages:



A pipeline run is usually triggered by a **source code repository**.
A change in code triggers a notification to the CI/CD tool that runs the corresponding pipeline. Other common triggers include:

Build Stage

In the build stage, the source code and its dependencies are combined to build a runnable instance of your product that you can potentially ship to end users.

Programs written in languages such as Java, C/C++, or Go need to be compiled, whereas Ruby, Python, and JavaScript programs work without this step.

If the software is deployed using Docker, this stage of the CI/CD pipeline builds the Docker images.

Failure to pass the build stage is an indicator of a fundamental problem in the configuration of the project.

Test Stage

- In test phase, automated tests run to validate the correctness of the code and the behavior of the product.
- The test stage acts as a safety net that prevents easily reproducible bugs from reaching the end users.
- The responsibility of writing tests falls on the developers, and is best done while writing new code in the process of test- or behavior-driven development.
- Depending on the size and complexity of the project, this phase can last from seconds to hours.

Deploy Stage

Once a runnable instance of the code is built and it passes all predefined tests, it is ready to be deployed.

There are usually multiple deploy environments, like a beta or a staging environment to be used internally by the product team and a production environment for end users.

Teams that use the Agile model of development usually deploy work-in-progress manually to a staging environment for additional manual testing and review.

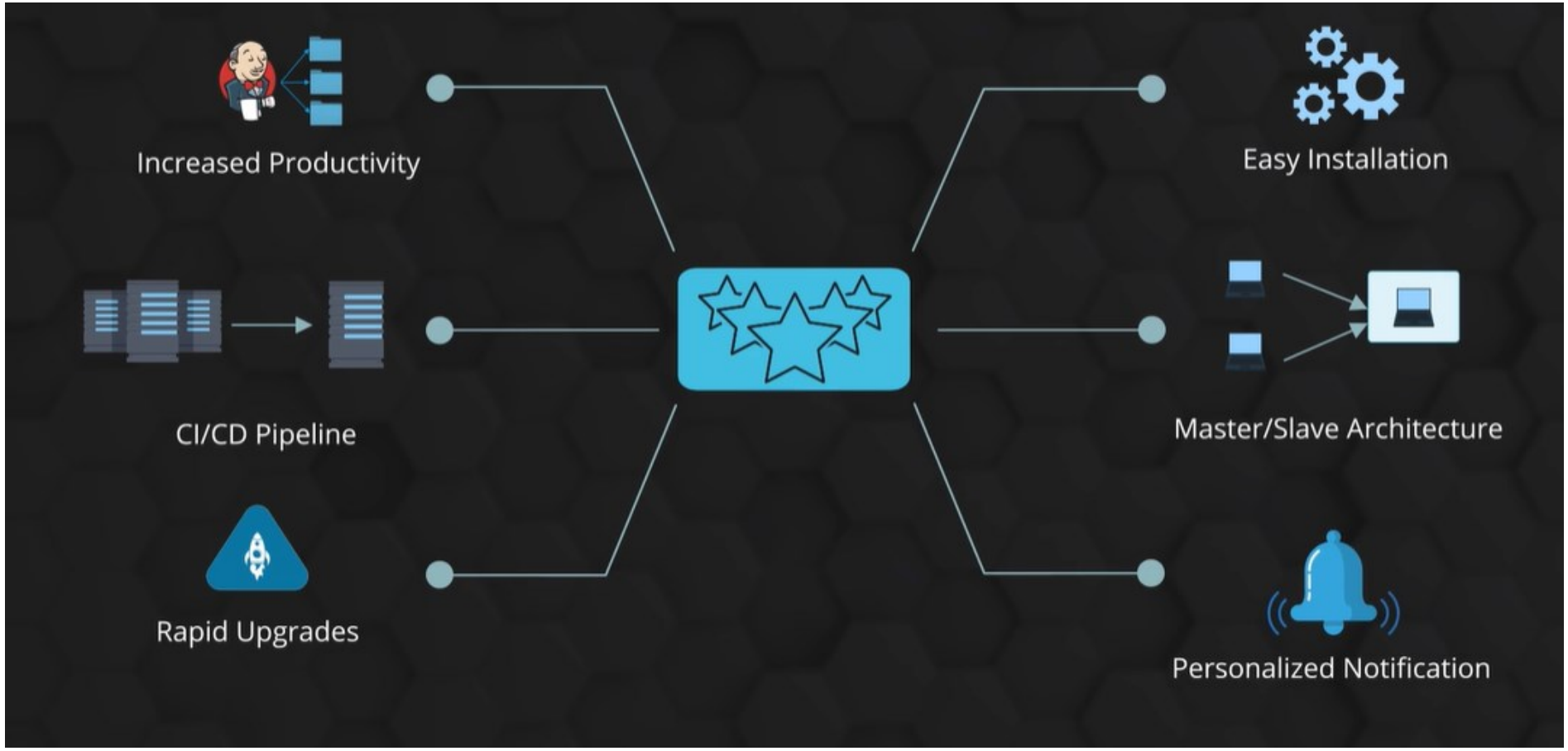
The approved changes are automatically deployed from the master branch to production.

Introduction to Jenkins

Jenkins

- Jenkins is an open-source project written in Java.
- It supports Windows, macOS and other Unix-like operating systems.
- It's free, community-supported, and is a popular first-choice tool for Continuous Integration.
- Jenkins is primarily deployed on-premises, but it can also run on cloud servers.

Features of Jenkins



What is Jenkins

Definition by Jenkins.io: Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

- One can install Jenkins as a standalone software, Docker, and also as a package on the machine that has JRE pre-installed.

Prerequisites:

1. A machine with 256 MB of RAM, although more than 512MB is recommended.

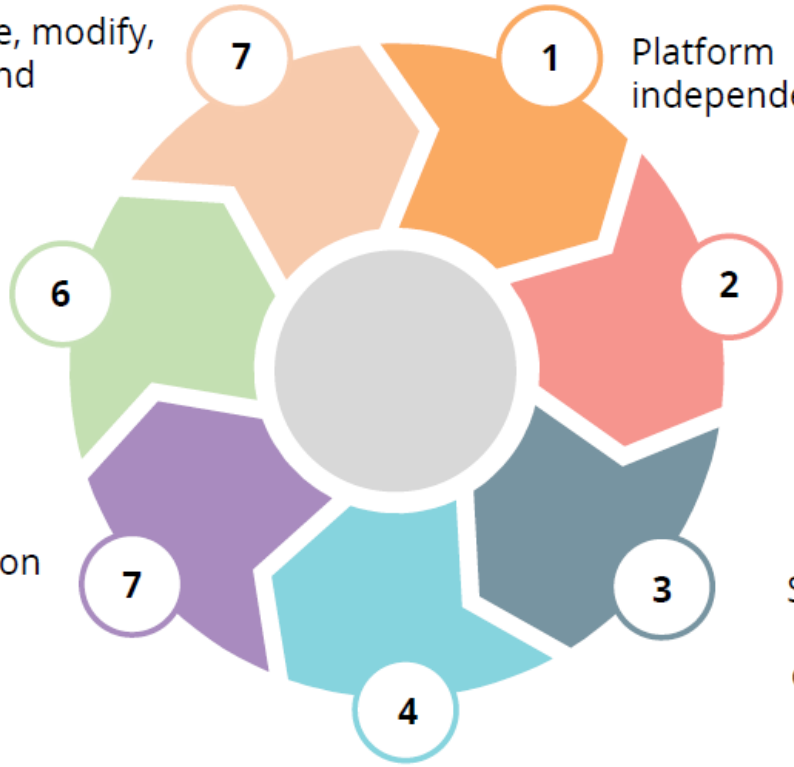
1. 10 GB of drive space (for Jenkins and your Docker image)

The following software installed:

1. Java 8 (either a JRE or Java Development Kit (JDK) is fine)

1. Docker suitable for your system.

Features of Jenkins



1. Platform independent
2. Rich plugin ecosystem
3. Support from large communities
4. Scaling of large error-ridden integrations
7. Immediate detection and resolution of issues
6. Open source and user-friendly
7. Easy to configure, modify, and extend

Configure Jenkins

- Add Plugins
  - Manage Jenkins
    - Global Tool Configuration
      - JDK
      - Maven
    - Manage Plugins (add)
      - Pipeline
      - Build Pipeline
      - Copy Artifact
      - Bitbucket
      - Git
      - SSH
      - Publish Over SSH
      - Pipeline: Stage View

CI & CD Pipeline With
Jenkins And Maven

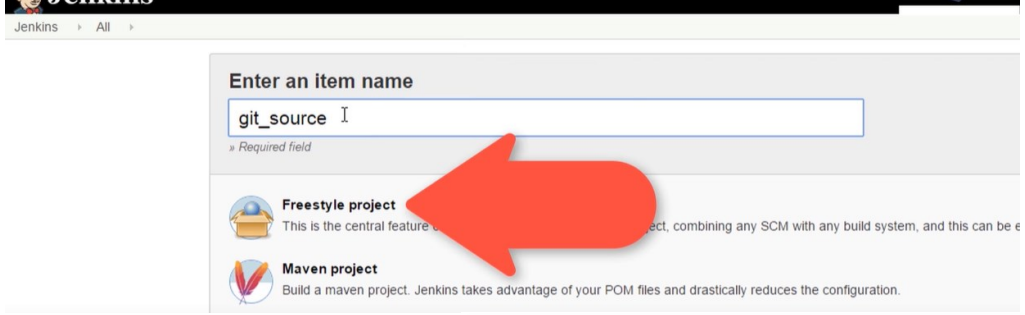
Deployment - Apache Tomcat

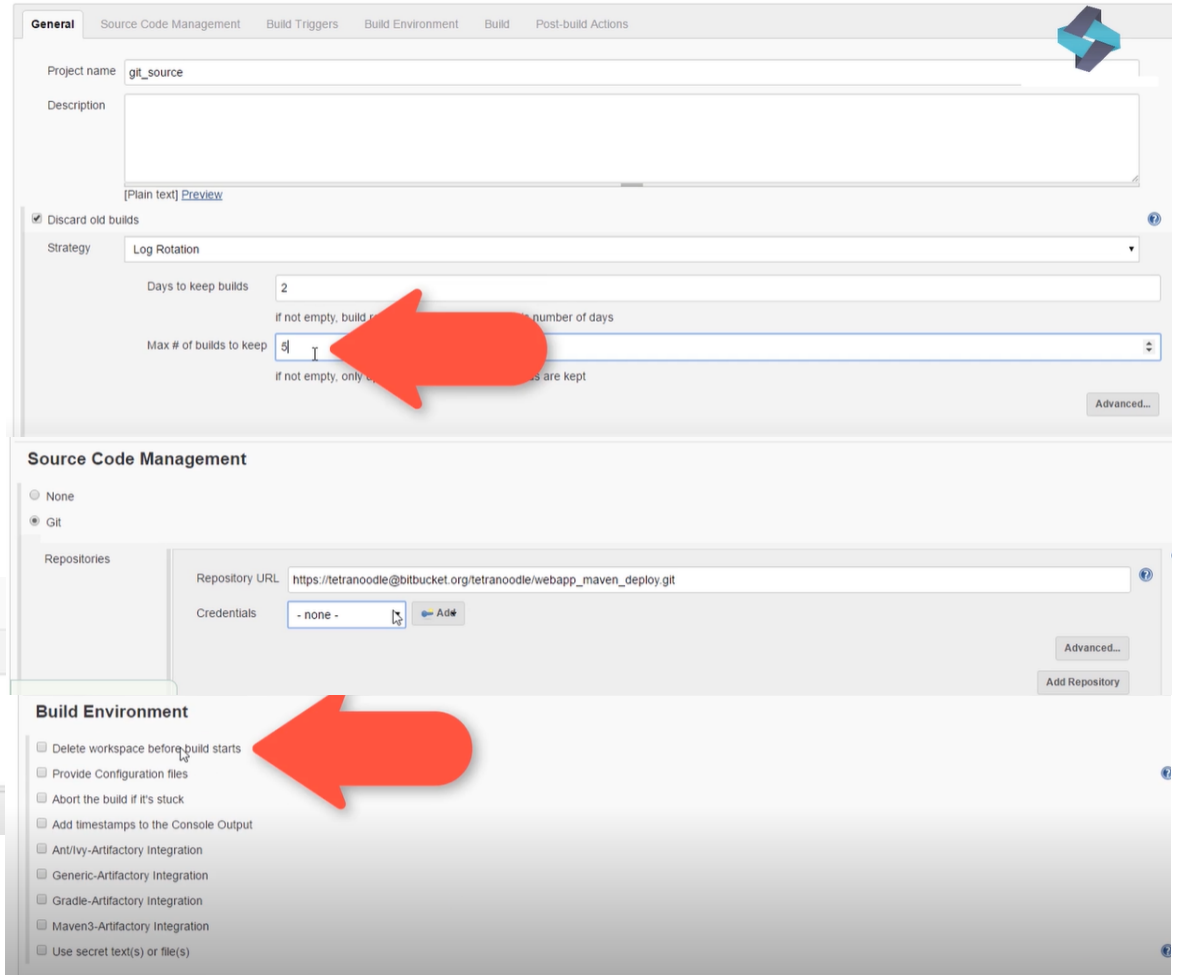
Build System - Maven

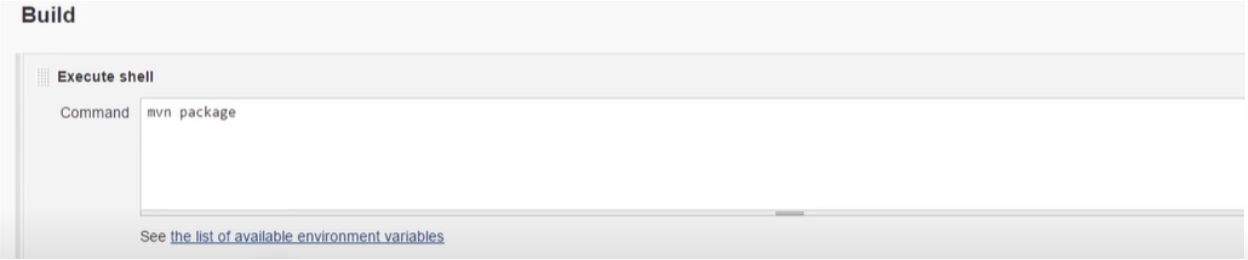
Git SCM

Simple Java Web Application

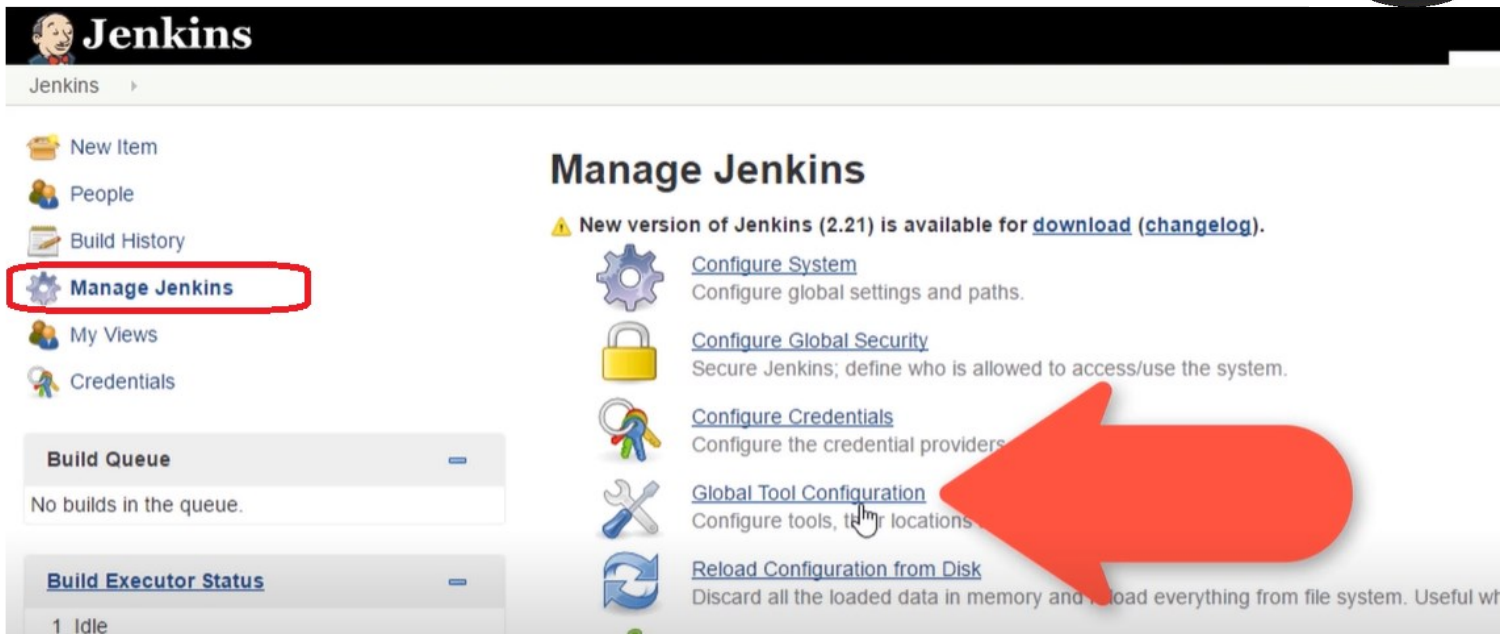
- Create a new item in Jenkins
- Customize the Project
  - Discard Old Builds
  - Source Code management
    - Provide the URL
  - Build Environment
  - Build Steps
    - Execute Shell





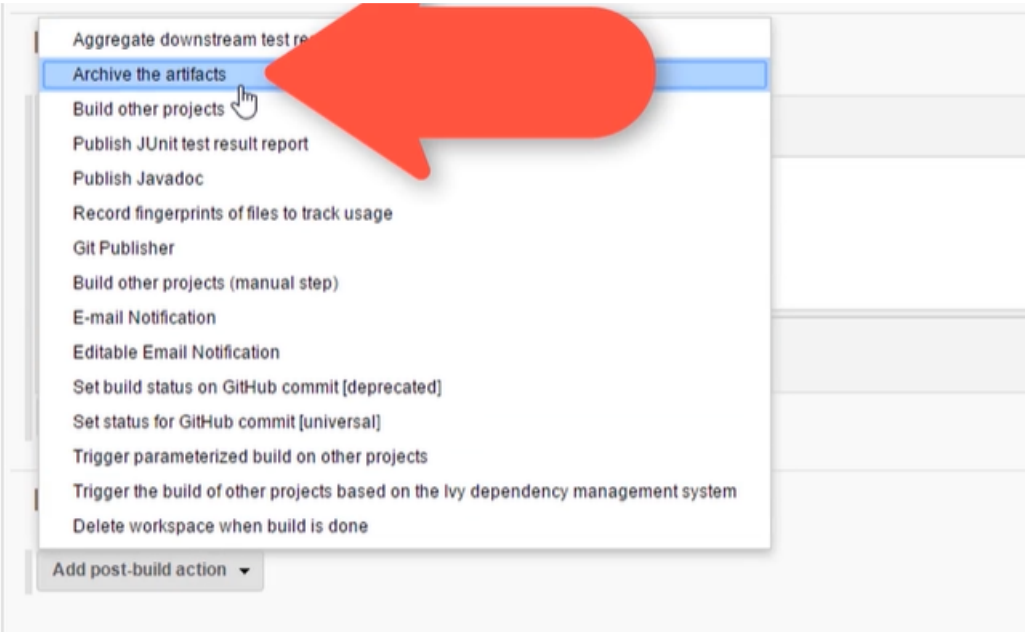


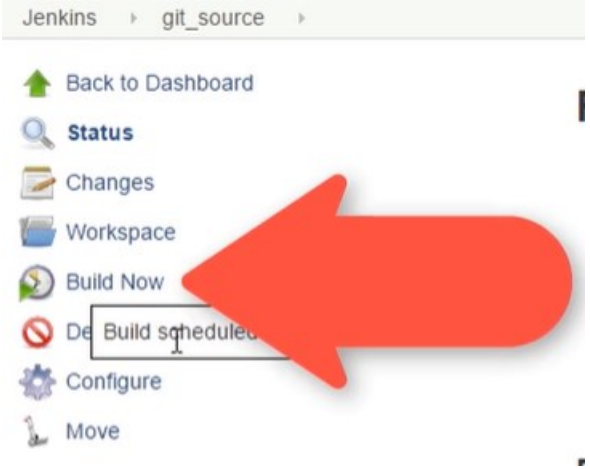
- Manage Jenkins
  - Global Tool Configuration
    - Add Maven



Pipelining

- Add Post-Build action
  - Archive the artifacts
    - multi3/target/*.war
- Apply and Save
- Build Now





Questions?