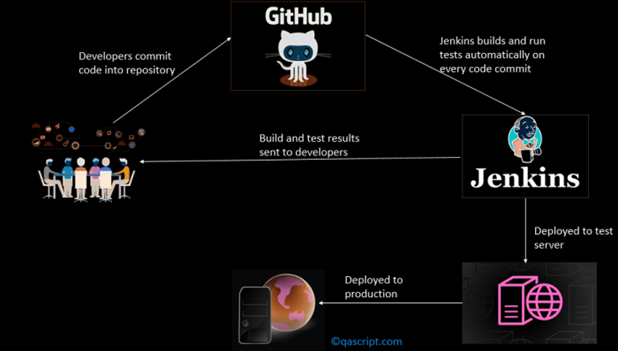
Jenkins for Noobs

**What is Jenkins?**

Jenkins is an open-source automation tool written in Java programming language that allows **continuous integration**.



Jenkins builds and tests our software projects which continuously makes it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build. It also allows us to continuously deliver our software by integrating with many testing and deployment technologies.



With the help of Jenkins, organizations can speed up the software development process through automation. Jenkins achieves CI (Continuous Integration) with the help of plugins. Plugins is used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you must install the plugins for that tool. For example, Maven 2 Project, Git, HTML Publisher, Amazon EC2 etc.

Jenkins can do the following steps (there are much more than these):

* Perform a software build using a build system like Gradle or Maven Apache
* Execute a shell script
* Archive a build result
* Running software tests

How to run Jenkins locally?

Download jenkins.war file from the website and run-

**java -jar Jenkins.war --httpPort=8000**

Jenkins UI

Well, there are a bunch of things that you can do using the UI, like installing plugins and stuff, managing users etc. This document will cover the basic ones, rest of them can be easily found on the internet.

Creating and Managing Users

All users should not have all the access rights, so creating users and assigning roles to them can be very helpful in managing projects.

Step 1: Install Role-based Authorization Strategy Plugin

Step 2: Enable Role-based Strategy in Global Security

Step 3: Create roles

Step 4: Assign roles to users

Jenkins Pipeline

A pipeline is a collection of events or jobs which are interlinked with one another in a sequence. In simple words, a Jenkins pipeline is a collection of jobs that brings software from version control into the hands of the end users by using automation tools. It is used to incorporate continuous delivery in our software development workflow.

A pipeline has an extensible automation server for creating simple or even complex delivery pipelines “as code”, via DSL (Domain-specific language).

Continuous Delivery Pipeline (CDP)

A CDP is a set of steps your code changes will go through to make their way to production. A CDP delivers, as per business needs, quality products frequently and predictably from test to staging to production in an automated fashion. A CDP works on a commit or a small incremental batch of commits which will make it to production whenever the pipeline runs successfully. Eventually, teams ship features – and ultimately – products in a secure and auditable way.

Phases in a CDP

The four most common phases in CDP are:

1. Component Phase
2. Subsystem Phase
3. System Phase
4. Production Phase

**The cost of defects is low when discovered in test, medium when discovered in staging, and high in production.**

**Shift Left** refers to validations being pulled earlier in the pipeline. The gate from test to staging has far more defensive techniques built in nowadays, and hence staging doesn’t have to look like a crime scene anymore.

CD Component Phase

The pipeline first builds components – the smallest distributable and testable units of the product.

What’s done here?

* Unit Tests
* Code Coverage
* Static Code Analysis (Linting, Memory Leaks, SAST – Static Analysis Security Testing)
* Code Reviews

CD Subsystem Phase

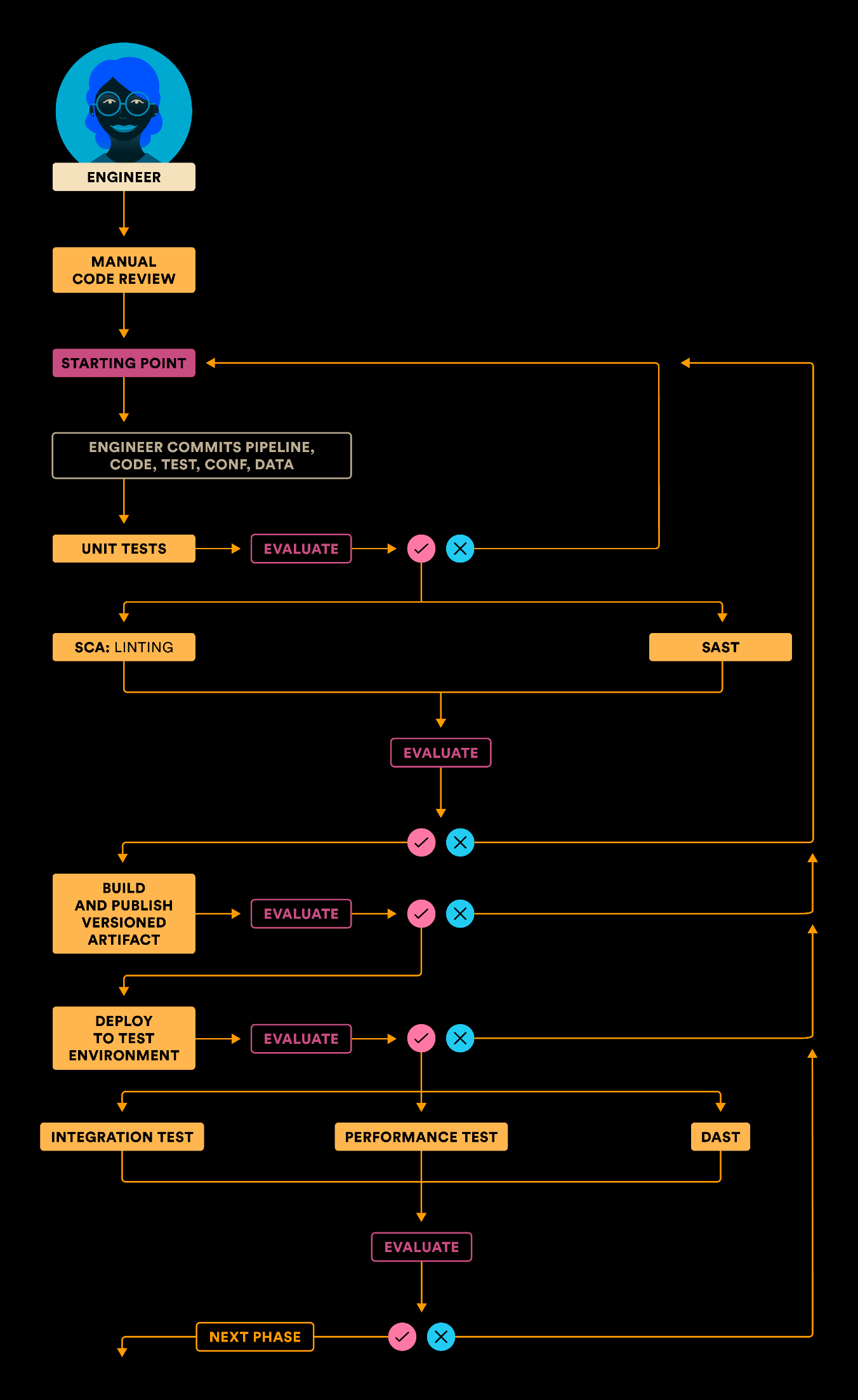
Loosely coupled components make up a subsystem – the smallest deployable and runnable units. For example, a server is a subsystem. A microservice running in a container is also a subsystem. A subsystem can be validated against customer use cases. Subsystems can be deployed and certified by functional, performance, and security tests.

Functional Tests include all customer use cases that involve internationalization (I18N), localization (L10N), data quality, accessibility, negative scenarios etc. These tests make sure that your product functions as per customer expectations, honors inclusion, and serves the market it’s built for.

Determine your performance benchmarks with your product owners.

DAST (Dynamic Analysis Security Testing) is a proven way to discover security vulnerabilities.

**Certifying Components And / Or Subsystems in the test environment**

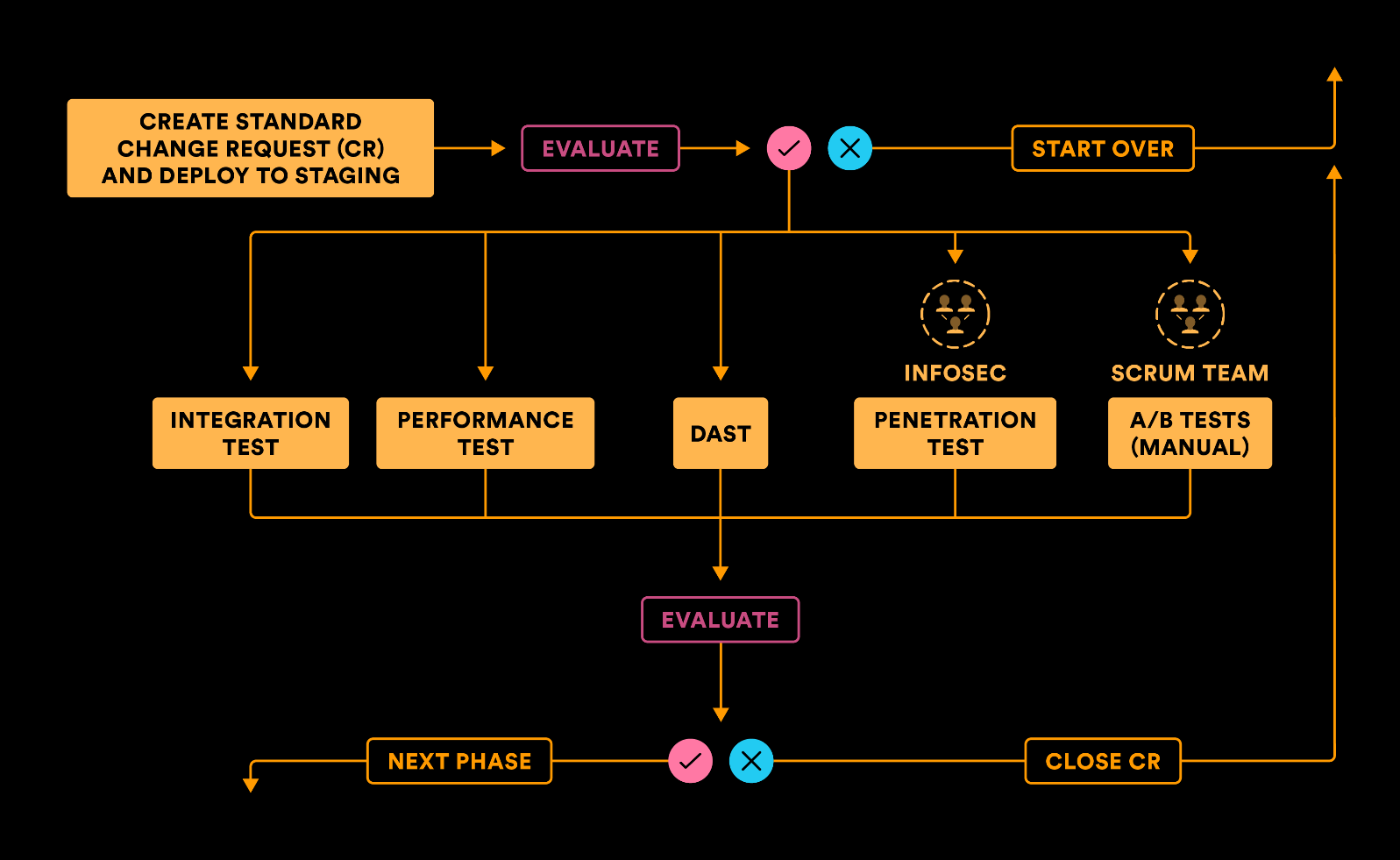
****

CD System Phase

Where systems need to be validated as a whole, they can be certified by integration, performance, and security tests. Unlike subsystem phase, do not use mocks or stubs during testing in this phase. Also, focus on testing interfaces and network more than anything else.

The pipeline can automatically file **Change Requests (CR)** to leave an audit trail.

**Certifying subsystems / system in the staging environment**

****

Once the assembled system is certified, leave the assembly unchanged and promote it to production.

CD Production Phase

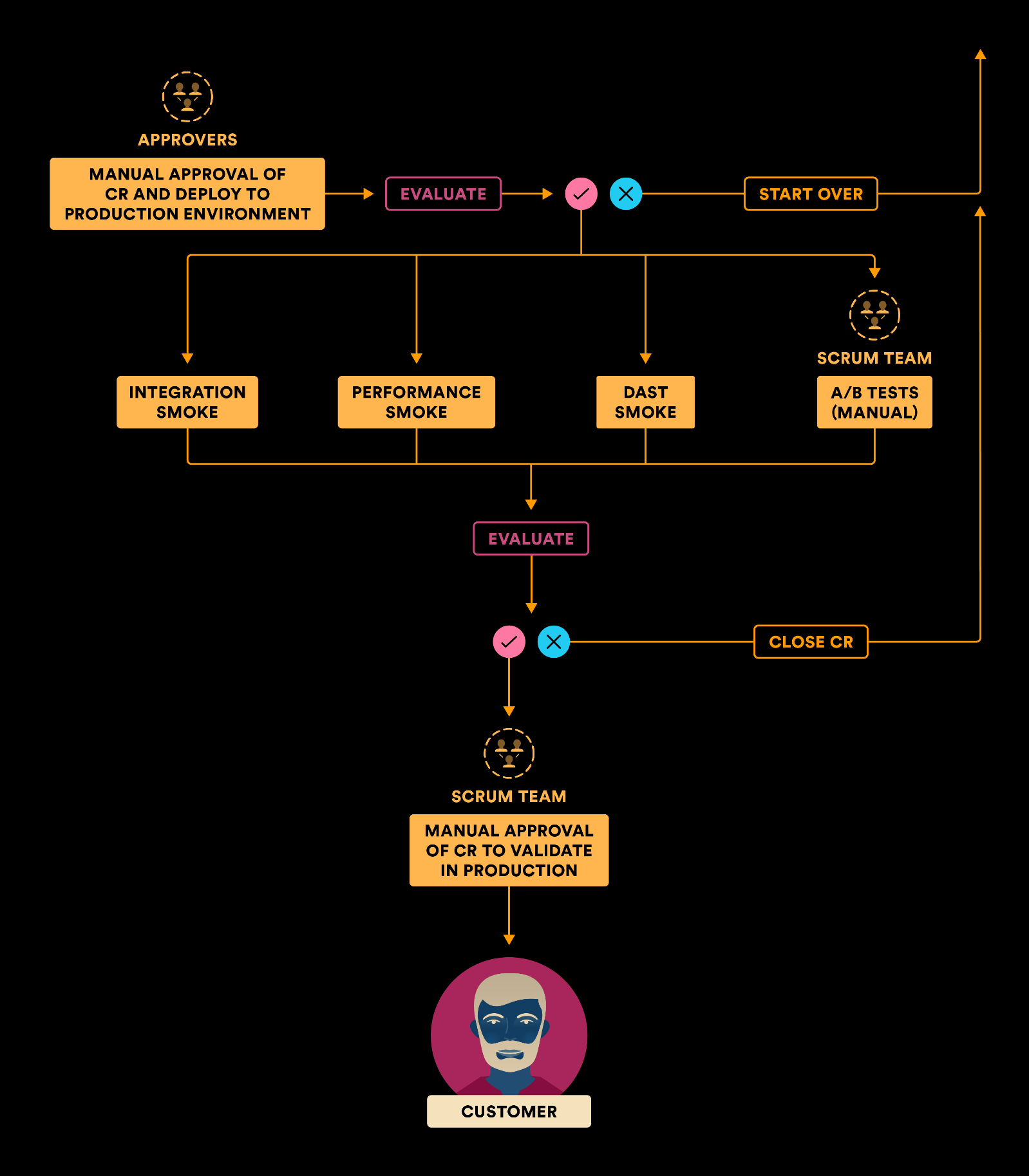
Zero Downtime Deployment (ZDD) is a must to prevent downtime for customers and should be practiced all the way from test to staging to production. Blue-green deployment is a popular technique.

There is a difference between Continuous Delivery and Continuous Deployment.

Continuous Delivery allows manual gates whereas Continuous Deployment doesn’t. While both are referred to as CD, continuous deployment requires more discipline and rigor as there is no human intervention in the pipeline.

Run smoke tests in production, which are a subset of the integration, performance, and security test suites. Once the smoke tests pass, turn the bits on, and the product goes live in the hands of the customers.

**Certifying subsystem / system in production environment**



Jenkins CLI

To get help for CLI commands, just append “/cli” to your Jenkins instance url. To get started, download the Jenkins-cli.jar file, and run a simple command:

java -jar jenkins-cli.jar -s <http://localhost:8080/> -webSocket help

This is the prefix to every command:

java -jar jenkins-cli.jar -s <http://localhost:8080/> -webSocket