Build and Release Management

Aman Kumar Gupta

CSE DevOps 18

R171218016-500067759

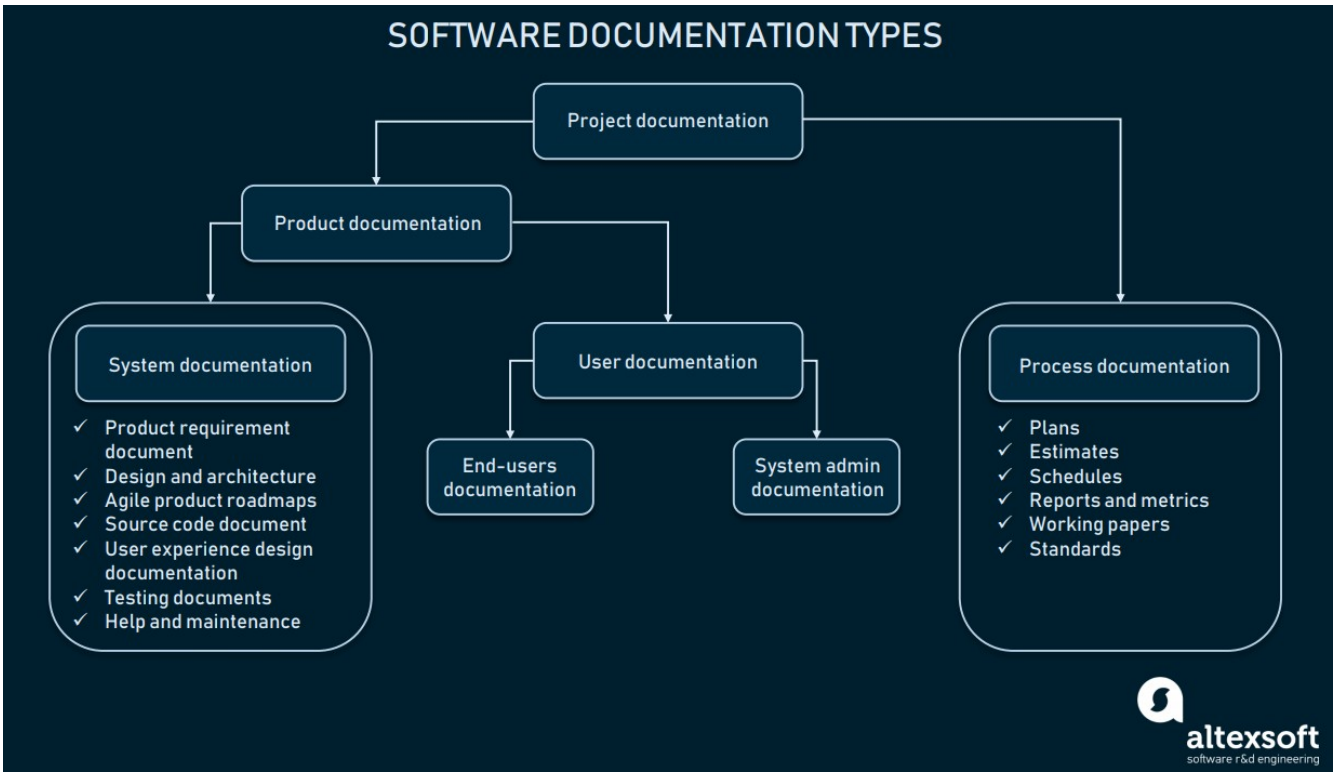
Q1. What is the difference between Product Documentation and Process Documentation?

Product documentation describes the product that is being developed and provides instructions on how to perform various tasks with it. In general, product documentation includes requirements, tech specifications, business logic, and manuals. There are two main types of product documentation:

- System documentation represents documents that describe the system itself and its parts. It includes requirements documents, design decisions, architecture descriptions, program source code, and FAQs.
- User documentation covers manuals that are mainly prepared for end-users of the product and system administrators. User documentation includes tutorials, user guides, troubleshooting manuals, installation, and reference manuals.

Process documentation represents all documents produced during development and maintenance that describe… well, the process. The common examples of process-related documents are standards, project documentation, such as project plans, test schedules, reports, meeting notes, or even business correspondence.

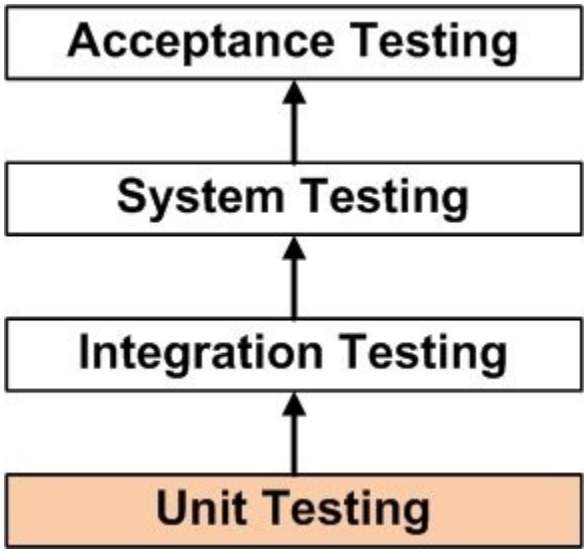
The main difference between process and product documentation is that the first one record the process of development and the second one describes the product that is being developed.



Q2. Write a short note on following :

a) Unit Testing

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/fake objects are used to assist in unit testing.



It is performed by using the White box testing methods.

There are several automated tools available to assist with unit testing. We will provide a few examples below:

- JUnit
- NUnit
- JMockit
- EMMA
- PHPUnit

Unit Testing Advantage

•Developers looking to learn what functionality is provided by a unit and how to use it can look at the unit tests to gain a basic understanding of the unit API.

•Unit testing allows the programmer to refactor code at a later date, and make sure the module still works correctly (i.e. Regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified and fixed.

•Due to the modular nature of the unit testing, we can test parts of the project without waiting for others to be completed.

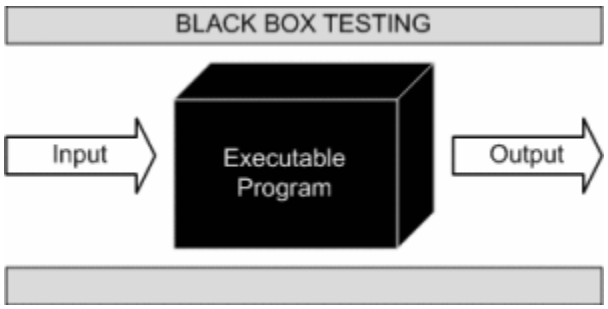
Unit Testing Disadvantages

•Unit testing can't be expected to catch every error in a program. It is not possible to evaluate all execution paths even in the most trivial programs

•Unit testing by its very nature focuses on a unit of code. Hence it can't catch integration errors or broad system level errors.

b) Black Box Testing

Also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

•Incorrect or missing functions

•Interface errors

•Errors in data structures or external database access

•Behavior or performance errors

•Initialization and termination errors

Here are the generic steps followed to carry out any type of Black Box Testing:

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones:

•**Functional testing**- This black box testing type is related to the functional requirements of a system; it is done by software testers.

•**Non-functional testing**- This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.

•**Regression testing**- Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Tools used for Black Box Testing:

Tools used for Black box testing largely depends on the type of black box testing you are doing.

I. For Functional/ Regression Tests you can use- Selenium,QTP

II. For Non-Functional Tests, you can use- LoadRunner, Jmeter

Black Box Testing method is applicable to the following levels of software testing:

I. Integration

II. System Testing

III. Acceptance Testing

The higher the level, and hence the bigger and more complex the box, the more black-box testing method comes into use.

Advantages

•Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.

•Tester need not know programming languages or how the software has been implemented.

•Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.

•Test cases can be designed as soon as the specifications are complete.

Disadvantages

•Only a small number of possible inputs can be tested and many program paths will be left untested.

•Without clear specifications, which is the situation in many projects, test cases will be difficult to design.

•Tests can be redundant if the software designer/developer has already run a test case.

•Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

c) Code Coverage

In simple words, Code coverage is a metric that can help you understand how much of your source is tested.

Following are major code coverage methods:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Toggle Coverage
- FSM Coverage

You might find several options to create coverage reports depending on the language(s) you use. Some of the popular tools are listed below:

- **Java:** Altassian Clover, Cobertura, JaCoCo, DevPartner
- **Javascript:** Istanbul, Blanket.js
- **PHP:** PHPUnit
- **Python:** Coverage.py
- **Ruby:** SimpleCov
- **C++:** BullsEye

Some tools like istanbul will output the results straight into your terminal while others can generate a full HTML report that lets you explore which part of the code are lacking coverage.

/

98.36% Statements 1859/1890 **93.92%** Branches 726/773 **100%** Functions 266/266 **99.73%** Lines 1816/1821 **37 statements, 21 branches** Ignored

File		Statements		Branches		Functions		Lines	
express/		100%	1/1	100%	0/0	100%	0/0	100%	1/1
express/examples/auth/		93.75%	75/80	80.77%	21/26	100%	17/17	100%	71/71
express/examples/content-negotiation/		100%	32/32	100%	2/2	100%	13/13	100%	32/32
express/examples/cookie-sessions/		100%	12/12	100%	4/4	100%	1/1	100%	12/12
express/examples/cookies/		95.83%	23/24	87.5%	7/8	100%	3/3	100%	22/22
express/examples/downloads/		94.12%	16/17	87.5%	7/8	100%	3/3	100%	15/15
express/examples/ejs/		100%	11/11	100%	2/2	100%	1/1	100%	11/11
express/examples/error-pages/		97.14%	34/35	83.33%	10/12	100%	6/6	100%	34/34
express/examples/error/		90%	18/20	66.67%	4/6	100%	4/4	100%	18/18
express/examples/markdown/		95.24%	20/21	66.67%	4/6	100%	5/5	100%	20/20
express/examples/multi-router/		100%	9/9	100%	2/2	100%	1/1	100%	9/9
express/examples/multi-router/controllers/		100%	14/14	100%	0/0	100%	4/4	100%	14/14
express/examples/mvc/		95.45%	42/44	80%	8/10	100%	4/4	100%	42/42
express/examples/mvc/controllers/main/		100%	2/2	100%	0/0	100%	1/1	100%	2/2
express/examples/mvc/controllers/pet/		94.12%	16/17	50%	1/2	100%	4/4	100%	16/16
express/examples/mvc/controllers/user-pet/		92.86%	13/14	50%	1/2	100%	1/1	100%	13/13
express/examples/mvc/controllers/user/		100%	21/21	100%	4/4	100%	6/6	100%	19/19
express/examples/mvc/lib/		97.96%	48/49	80%	20/25	100%	2/2	100%	46/46

Advantages:

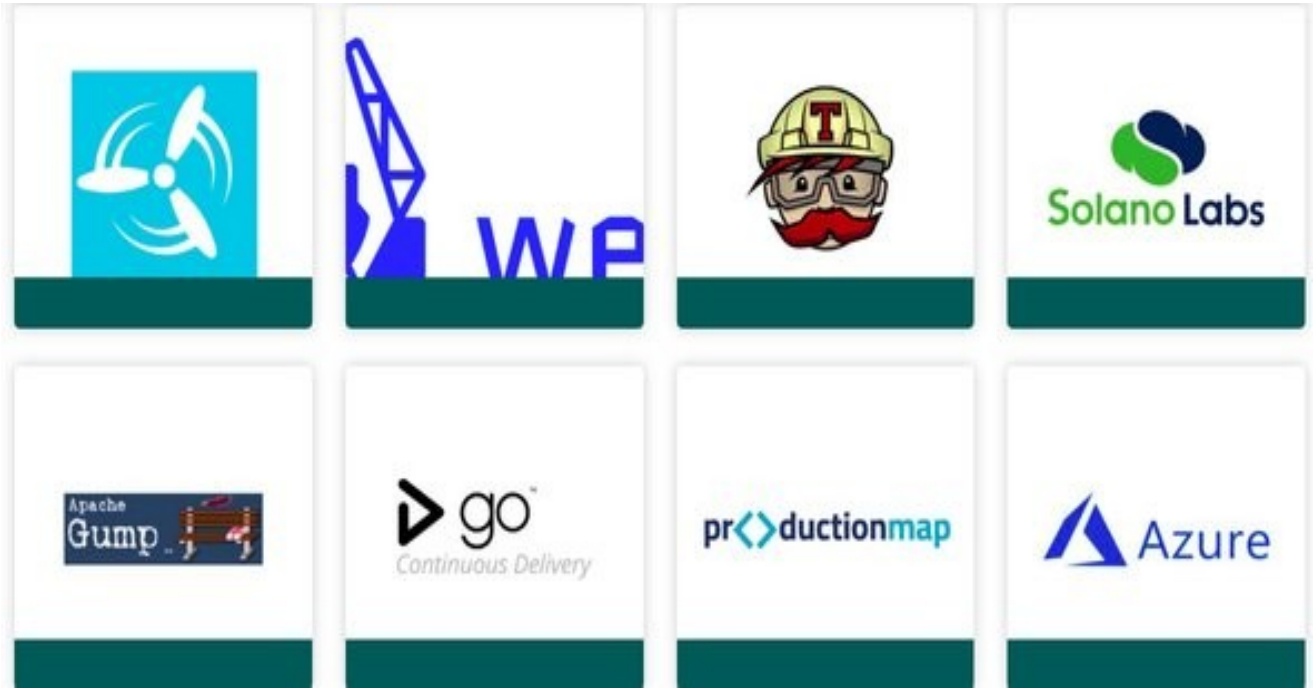
•Helpful to evaluate a quantitative measure of code coverage
•It allows you to create extra test cases to increase coverage
•It allows you to find the areas of a program which is not exercised by a set of test cases

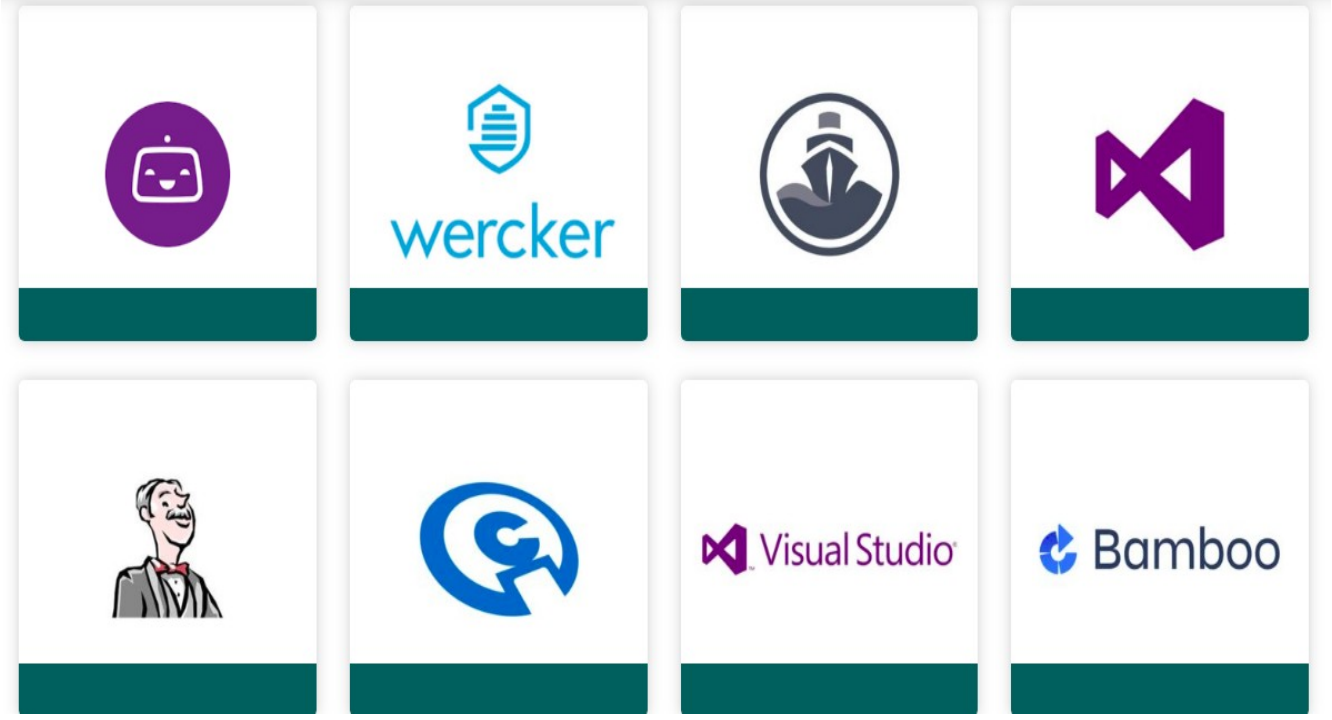
Disadvantages:

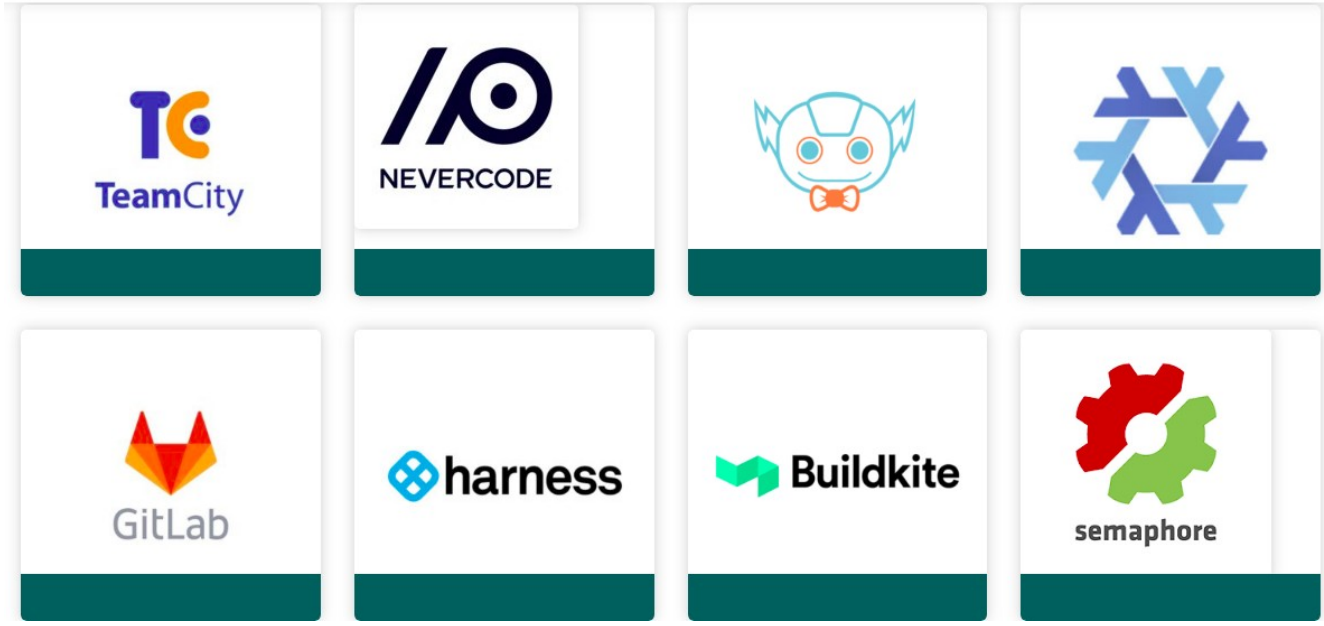
•Even when any specific feature is not implemented in design, code coverage still report 100% coverage.
•It is not possible to determine whether we tested all possible values of a feature with the help of code coverage
•Code coverage is also not telling how much and how well you have covered your logic
•In the case when the specified function hasn't implemented, or a not included from the specification, then structure-based techniques cannot find that issue.

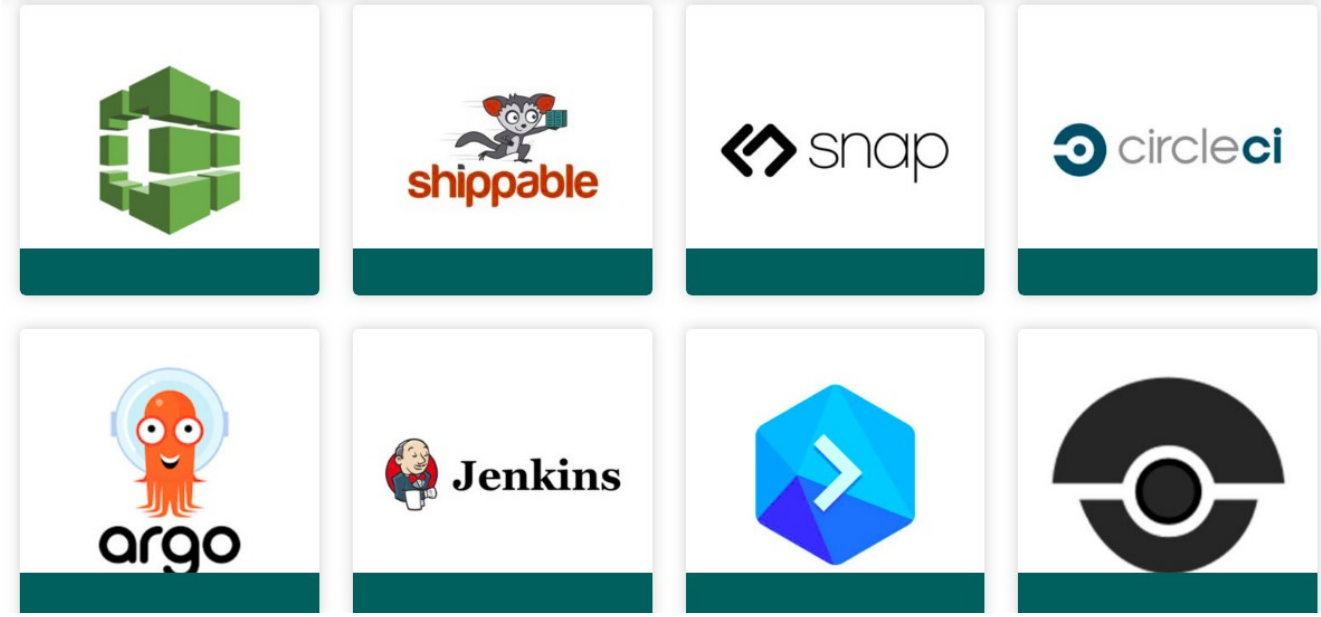
Q3. Write any Three majorly used open source tools for CI/CD.

Some of the popular CD tools are the following:

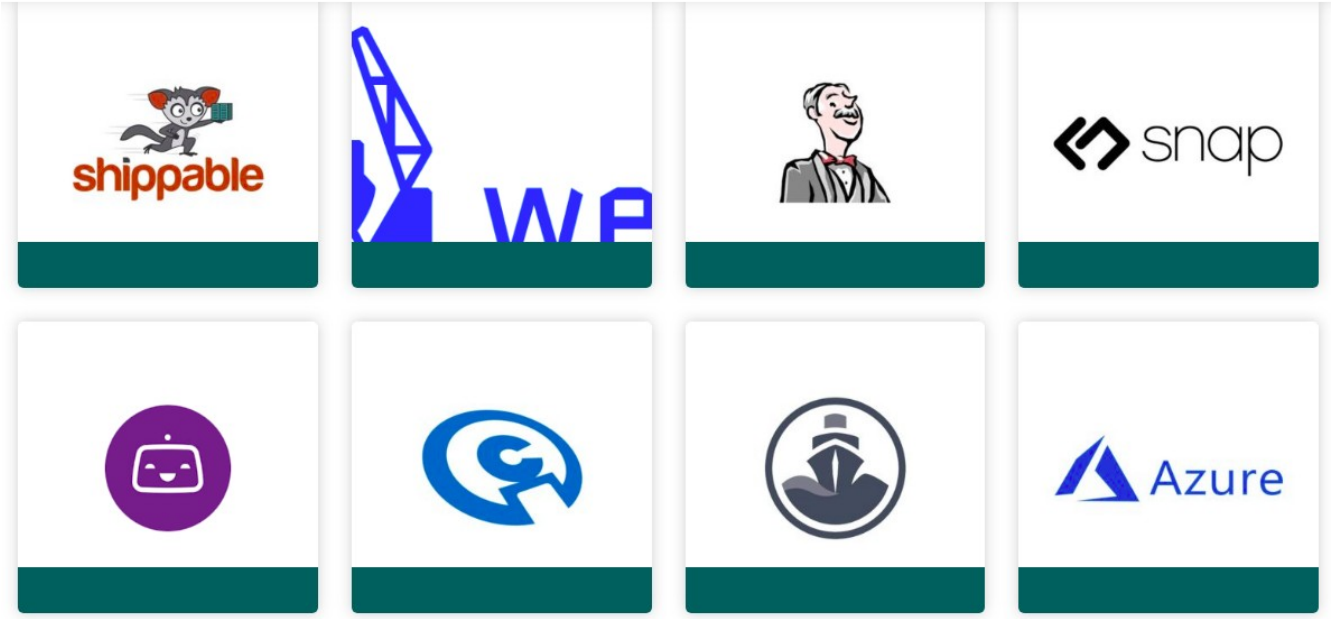


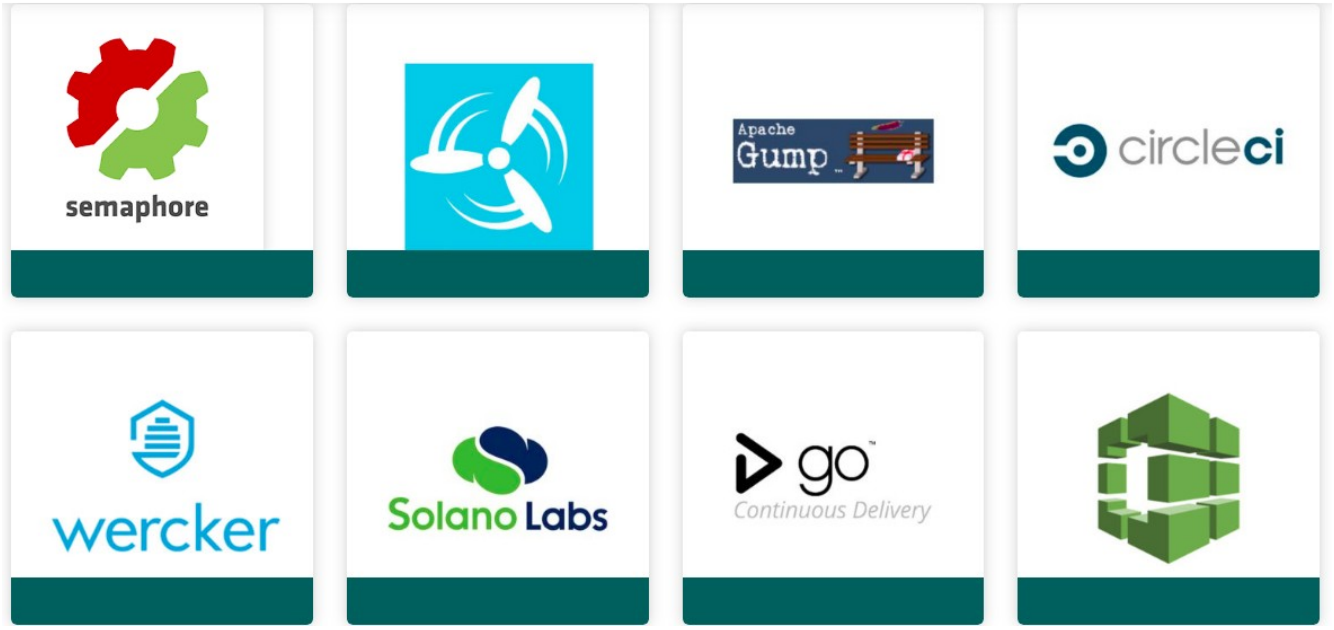


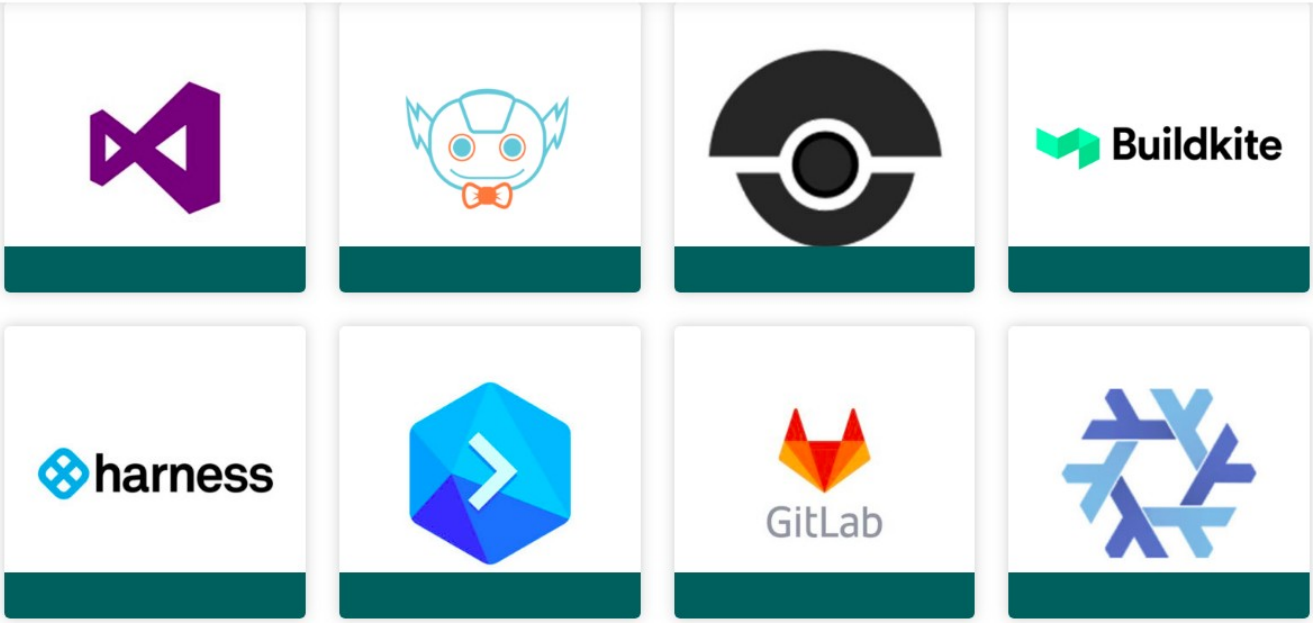

TeamCity
NEVERCODE
GitLab
harness
Buildkite
semaphore

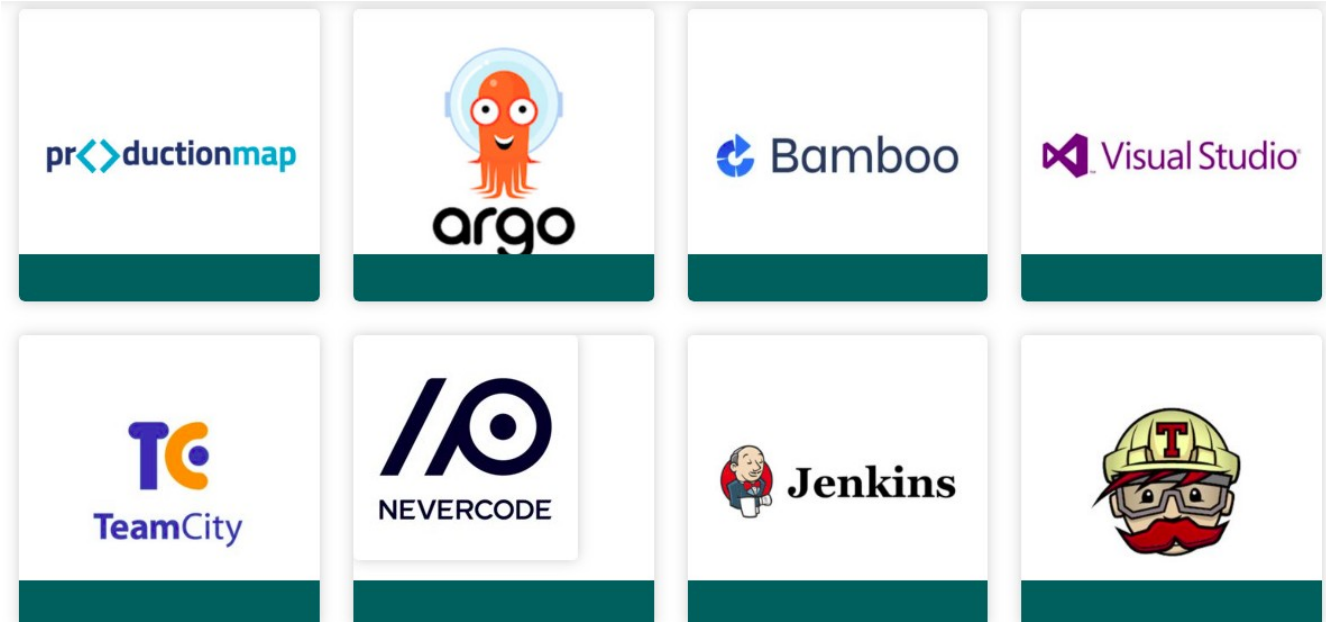

shippable
snap
circleci
argo
Jenkins

Some of the popular CI tools are the following:






Buildkite
harness
GitLab


productionmap
argo
Bamboo
Visual Studio
TeamCity
NEVERCODE
Jenkins

Q4. Write Two Majorly used Source Code Management tools and their small description.

Mercurial

It is a popular distributed version control system, that offers way to archive as well as to save older versions of source code. Mercurial came into existence in 2005 as an open-source version control system, as an alternative to the closed source Bitkeeper and was developed by Matt Mackall. Unlike SVN, which is a centalized version control system, Mercurial is a distributed version control system. That is, when you push changes to the repository, it will go to the local machine. Because of this, the process becomes very faster, since you're not constantly pushing to a remote server (although it can be set up that way).

Mercurial is built primarily in python, which makes it cross-platform compatible. This is also one of the reasons that Mercurial is most used as a command-line tool, though there are GUI tools available. Mercurial has been the version control system used by big brands like Adium, Mozilla, Netbeans, Vim, Growl and so forth. Apart from these, a lot of individual developers use Mercurial to manage their code.

Git

It is a distributed version control system similar to Mercurial. Both Git and Mercurial were created with a goal of replacing BitKeeper, the version control system used by the Linux kernel group then. Git was created almost single-handedly by Linus Torvalds, the creator of Linux. Linux group had the following goals in mind while developing the alternative version control system.

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux Kernel efficiently (speed and data size)

Since its creation in 2005,Git has been the most popular tool for source code management and version control. The truly distributed nature of Git has attracted many developers, who themselves use this product for their individual source code management. Git works well with most of the operating systems and IDEs.