Priyadarshini D.

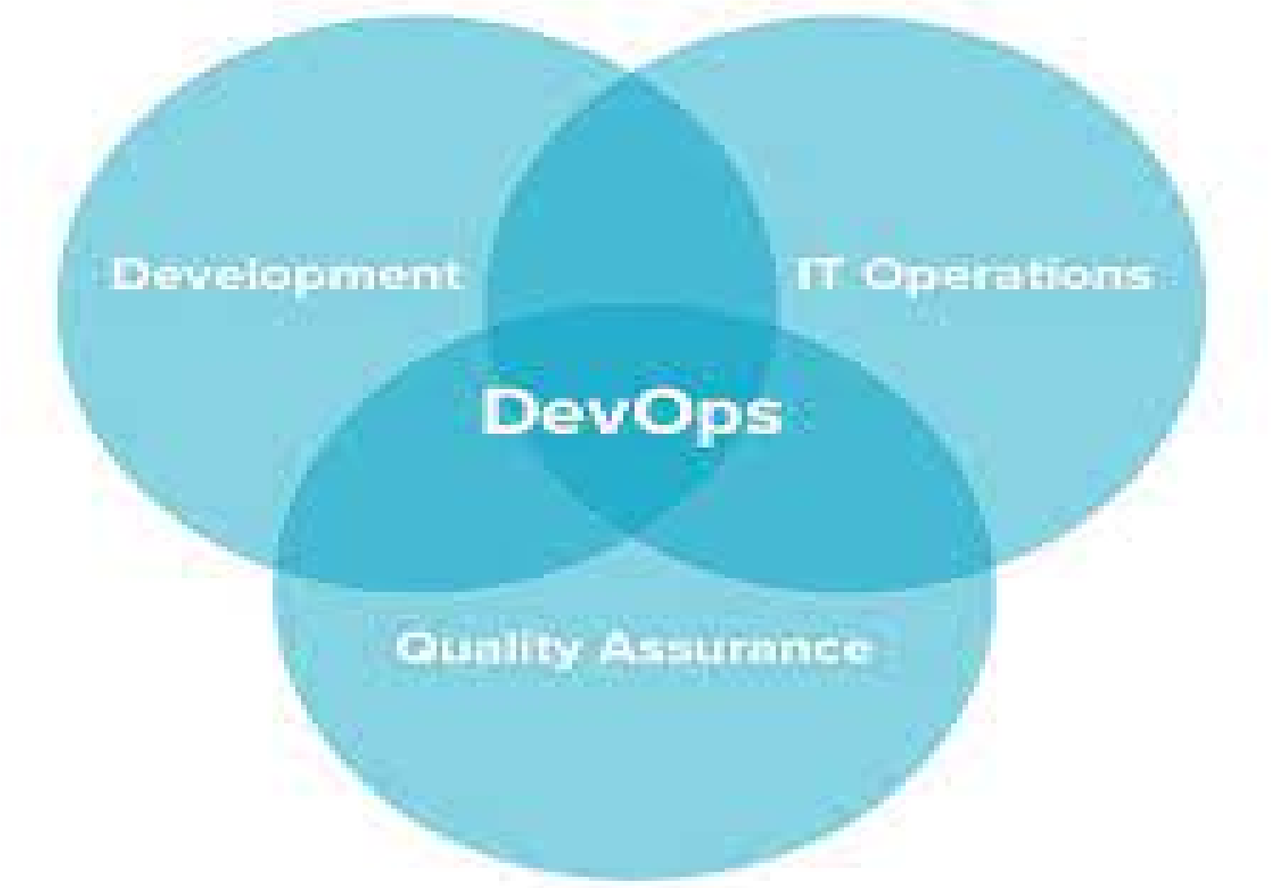
What is DevOps?

DevOps is an evolving philosophy and framework that encourages faster, better application development and faster release of new or revised software features or products to customers.

The practice of DevOps encourages smoother, continuous communication, collaboration, integration, visibility, and transparency between application development teams (Dev) and their IT operations team (Ops) counterparts.

DevOps Engineering Practices

DevOps stands for development and operations. Its pratices that aims at merging development, vality assurance and operations into a single, continuous set of process. This methodology is a natural extention of agile and continuous delivery approaches.



Configuration Management

Configuration management occurs when a configuration platform is used to automate, monitor, design and manage otherwise manual configuration processes. System-wide changes take place across servers and networks, storage, applications, and other managed system.

An important function of configuration management is defining the state of each system. By orchestrating these processes with a platform, organizations can ensure consistency across integrated systems and increase efficiency. The result is that businesses can scale more readily without hiring additional IT management staff. Companies that otherwise wouldn’ t have the resources can grow by deploying a DevOps approach.

What is continuous integration

Continuous integration refers to the build and unit testing stage of the software release process. Every revision that is committed triggers an automated build and test. With continuous delivery, code changes are automatically built, tested, and prepared for a release to production.

What is automated testing

DevOps makes testing a shared responsibility of the entire team, while test automation enables developers to ship code changes quickly with high confidence in quality.

Infrastructure as Code

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes. With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations.

Continuous Delivery

Continuous delivery lets developers automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying to customers. These tests may include UI testing, load testing, integration testing, API reliability testing, etc.

DevOps practices

DevOps practices reflect the idea of continuous improvement and automation. Many practices focus on one or more development cycle phases. These practices include:

- **Continuous development.** This practice spans the planning and coding phases of the DevOps lifecycle. Version-control mechanisms might be involved.

- Continuous testing. This practice incorporates automated, prescheduled, continued code tests as application code is being written or updated. Such tests can speed the delivery of code to production.

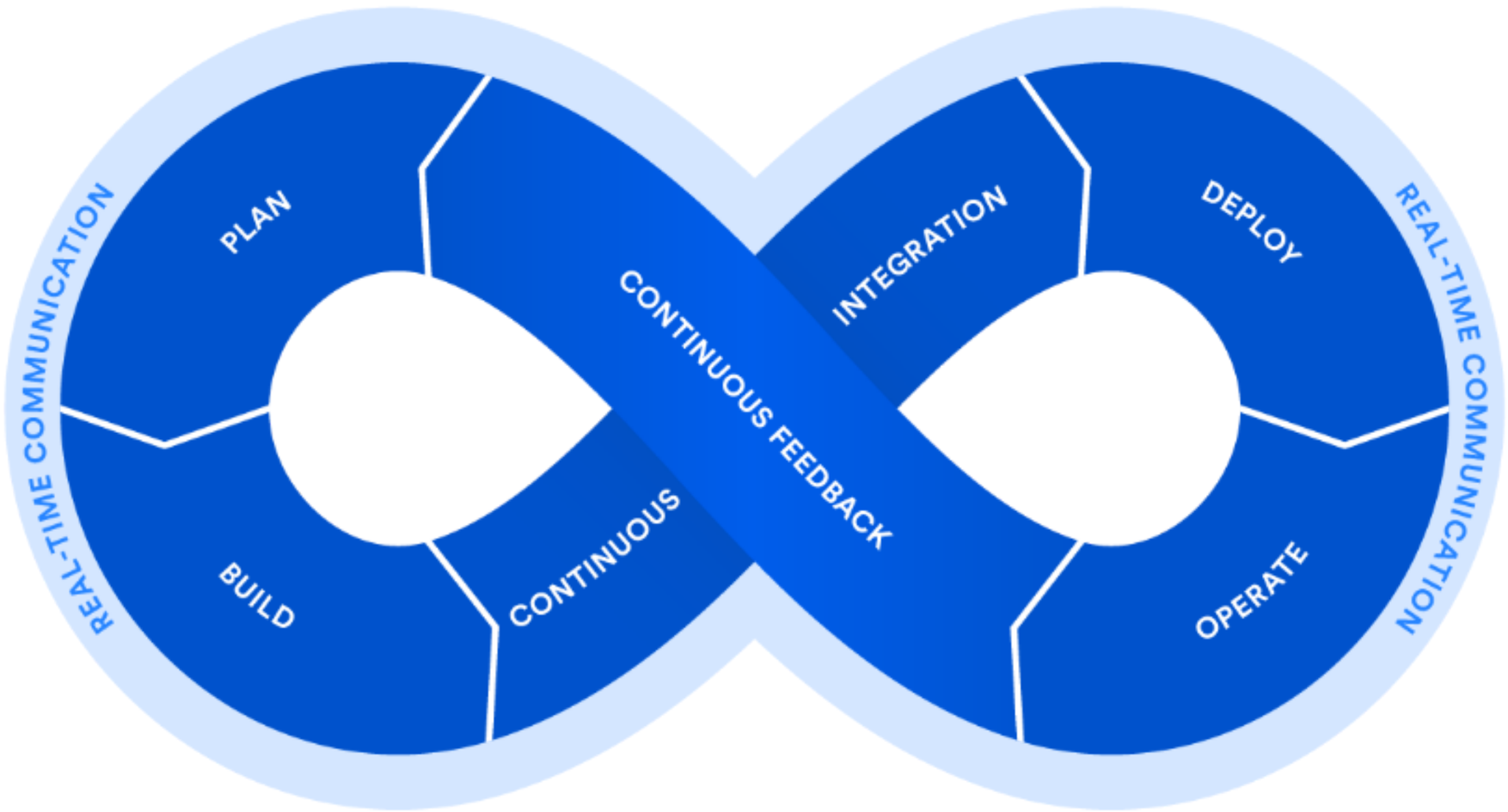
- Continuous integration (CI). This practice brings configuration management (CM) tools together with other test and development tools to track how much of the code being developed is ready for production. It involves rapid feedback between testing and development to quickly identify and resolve code issues.

- Continuous delivery. This practice automates the delivery of code changes, after testing, to a preproduction or staging environment. An staff member might then decide to promote such code changes into production.

- Continuous deployment (CD). Similar to continuous delivery, this practice automates the release of new or changed code into production. A company doing continuous deployment might release code or feature changes several times per day. The use of container technologies, such as Docker and Kubernetes, can enable continuous deployment by helping to maintain consistency of the code across different deployment platforms and environments.

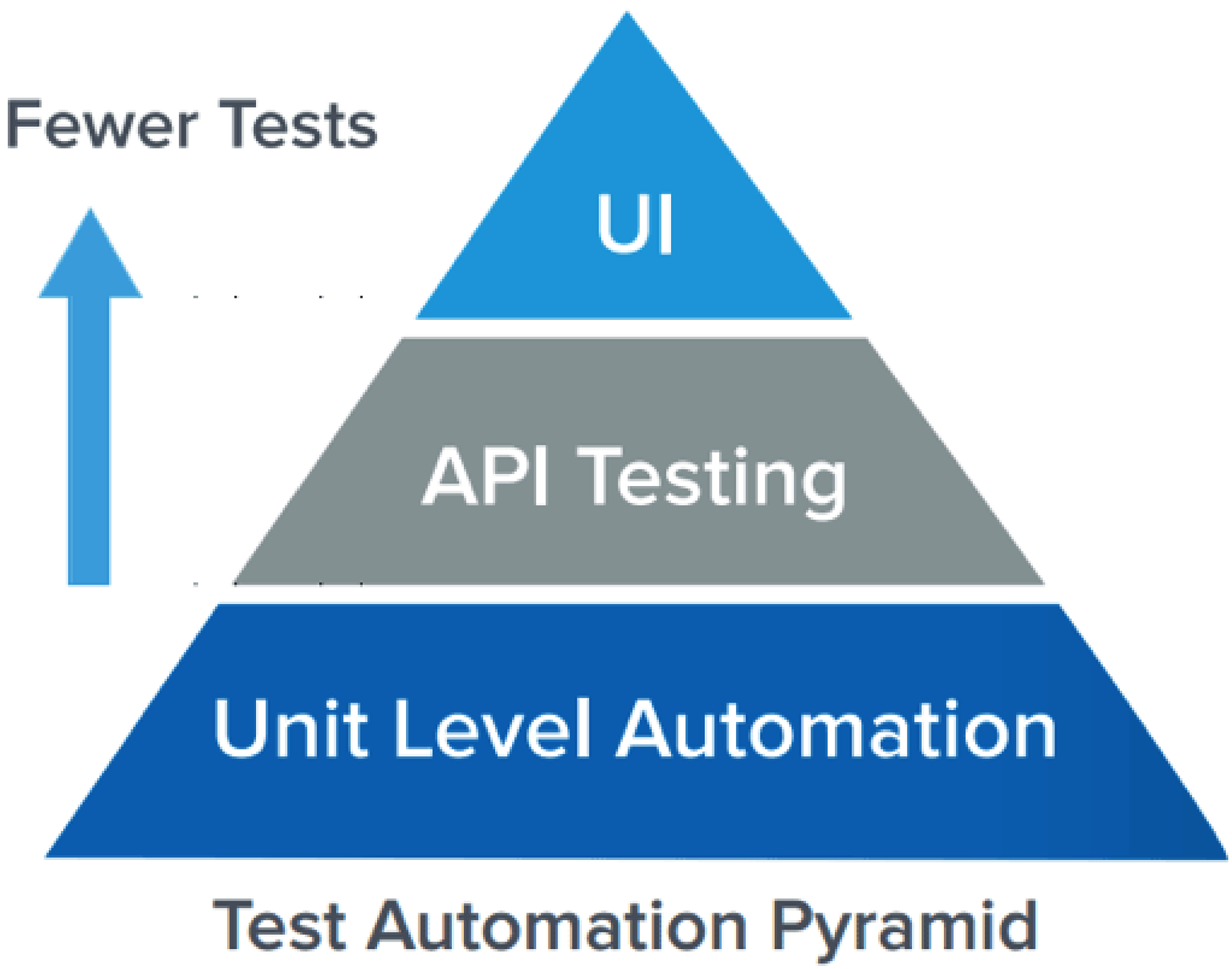
- Continuous monitoring. This practice involves ongoing monitoring of both the code in operation and the underlying infrastructure that supports it. A feedback loop that reports on bugs or issues then makes its way back to development.

- Infrastructure as code. This practice can be used during various DevOps phases to automate the provisioning of infrastructure required for a software release. Developers add infrastructure "code" from within their existing development tools. For example, developers might create a storage volume on demand from Docker, Kubernetes, or OpenShift. This practice also allows operations teams to monitor environment configurations, track changes, and simplify the rollback of configurations.

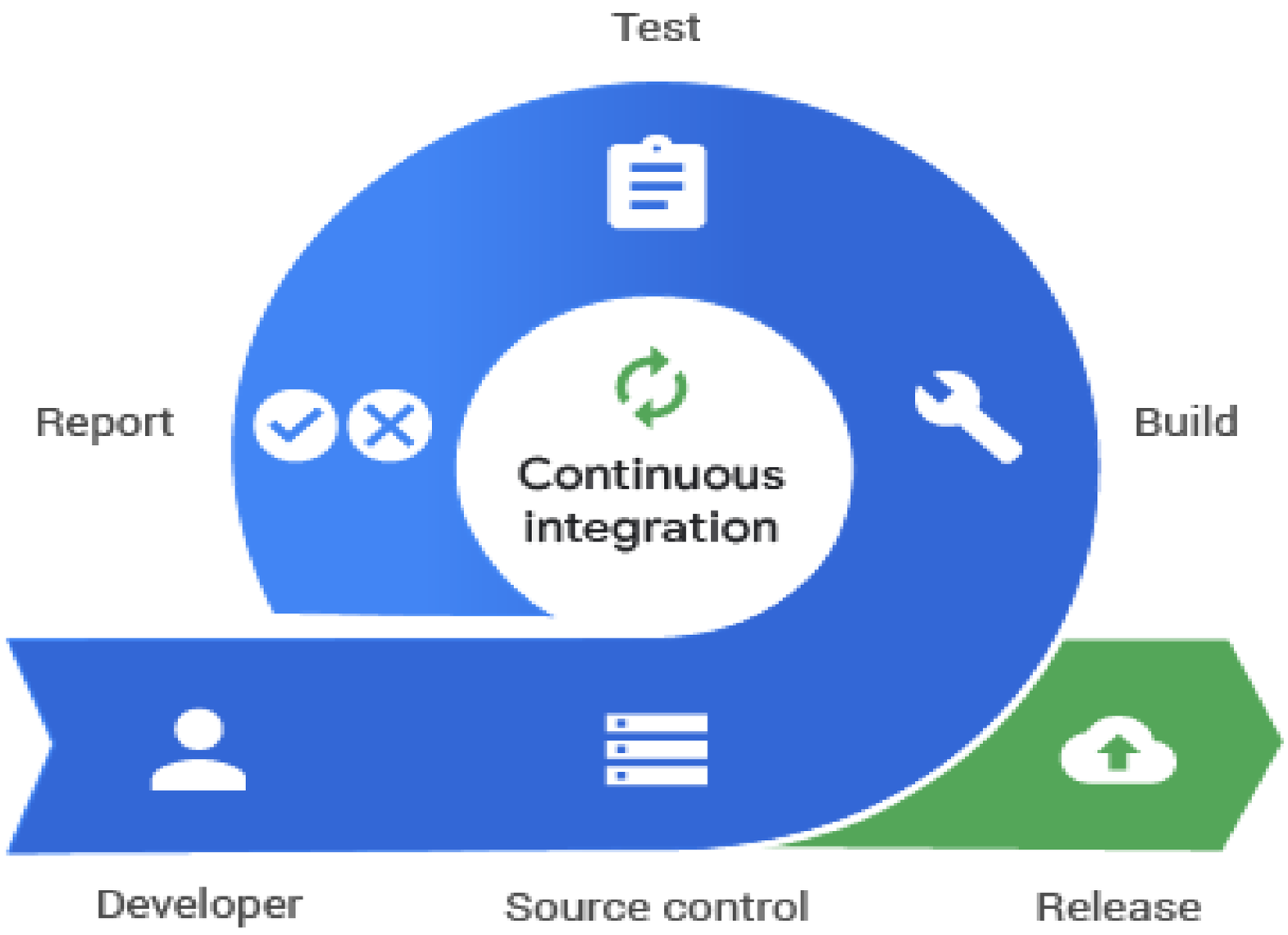


Agile planning in DevOps

- Continuous development. This practice spans the planning and coding phases of the DevOps lifecycle. Version-control mechanisms might be involved.
- Continuous testing. This practice incorporates automated, prescheduled, continued code tests as application code is being written or updated. Such tests can speed the delivery of code to production.

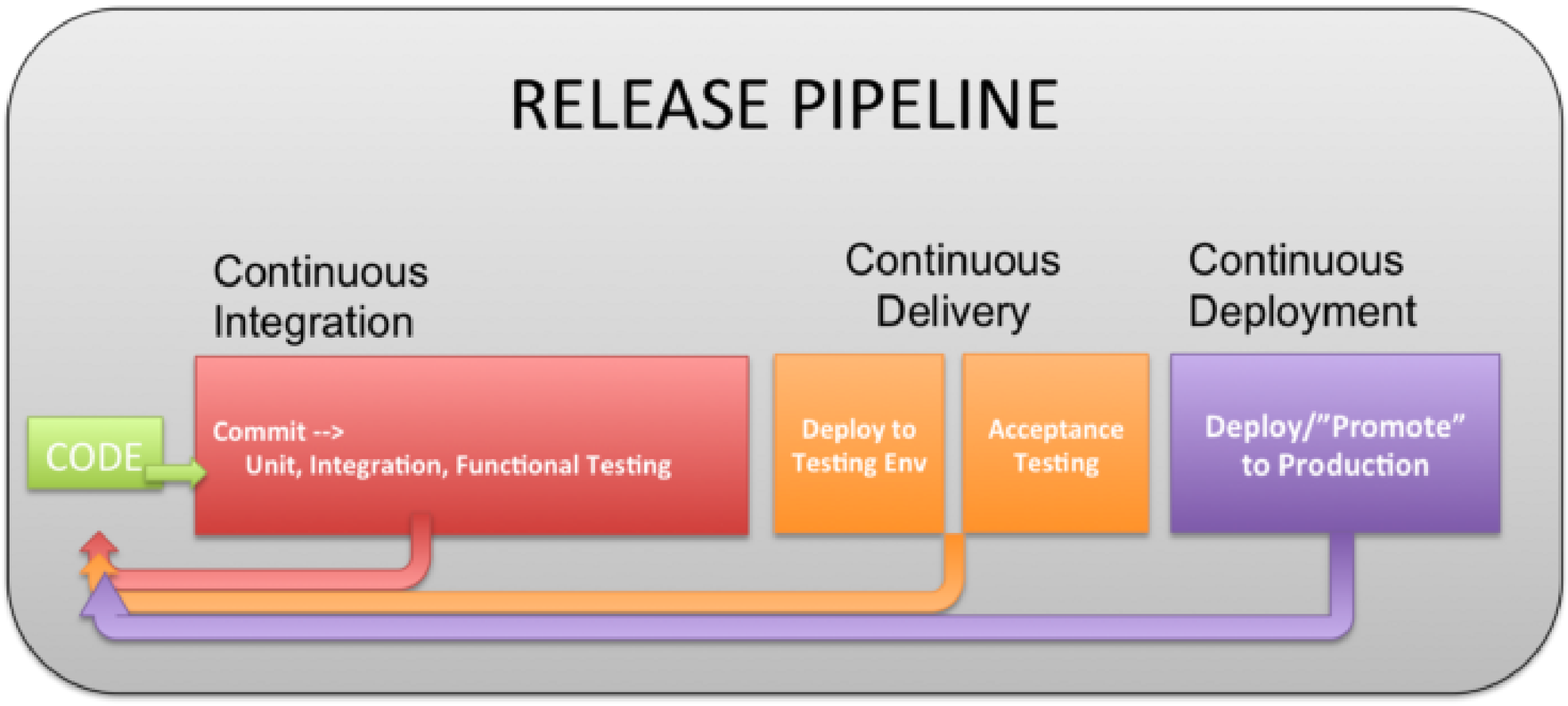


- Continuous integration (CI). This practice brings configuration management (CM) tools together with other test and development tools to track how much of the code being developed is ready for production. It involves rapid feedback between testing and development to quickly identify and resolve code issues.



- Continuous delivery. This practice automates the delivery of code changes, after testing, to a preproduction or staging environment. An staff member might then decide to promote such code changes into production.

- Continuous deployment (CD). Similar to continuous delivery, this practice automates the release of new or changed code into production. A company doing continuous deployment might release code or feature changes several times per day. The use of container technologies, such as Docker and Kubernetes, can enable continuous deployment by helping to maintain consistency of the code across different deployment platforms and environments.



- Continuous monitoring. This practice involves ongoing monitoring of both the code in operation and the underlying infrastructure that supports it. A feedback loop that reports on bugs or issues then makes its way back to development.

- Infrastructure as code. This practice can be used during various DevOps phases to automate the provisioning of infrastructure required for a software release. Developers add infrastructure "code" from within their existing development tools. For example, developers might create a storage volume on demand from Docker, Kubernetes, or OpenShift. This practice also allows operations teams to monitor environment configurations, track changes, and simplify the rollback of configurations.

Benefits of DevOps

DevOps proponents describe several business and technical benefits, many of which can result in happier customers. Some benefits of DevOps include:

- Faster, better product delivery
- Faster issue resolution and reduced complexity
- Greater scalability and availability
- More stable operating environments
- Better resource utilization
- Greater automation
- Greater visibility into system outcomes
- Greater innovation

The 7 key practices of DevOps are:

1. Configuration Management
2. Continuous Integration
3. Automated Testing
4. Infrastructure as Code
5. Continuous Delivery
6. Continuous Deployment
7. Continuous Monitoring

DevOps tools

Followers of DevOps practices often use certain DevOps-friendly tools as part of their DevOps “ toolchain.” The goal of these tools is to further streamline, shorten, and automate the various stages of the software delivery workflow (or “ pipeline”). Many such tools also promote core DevOps tenets of automation, collaboration, and integration between development and operations teams. The following shows a sample of tools used at various DevOps lifecycle stages.

- Plan. This phase helps define business value and requirements. Sample tools include Jira or Git to help track known issues and perform project management.

- Code. This phase involves software design and the creation of software code. Sample tools include GitHub, GitLab, Bitbucket, or Stash.

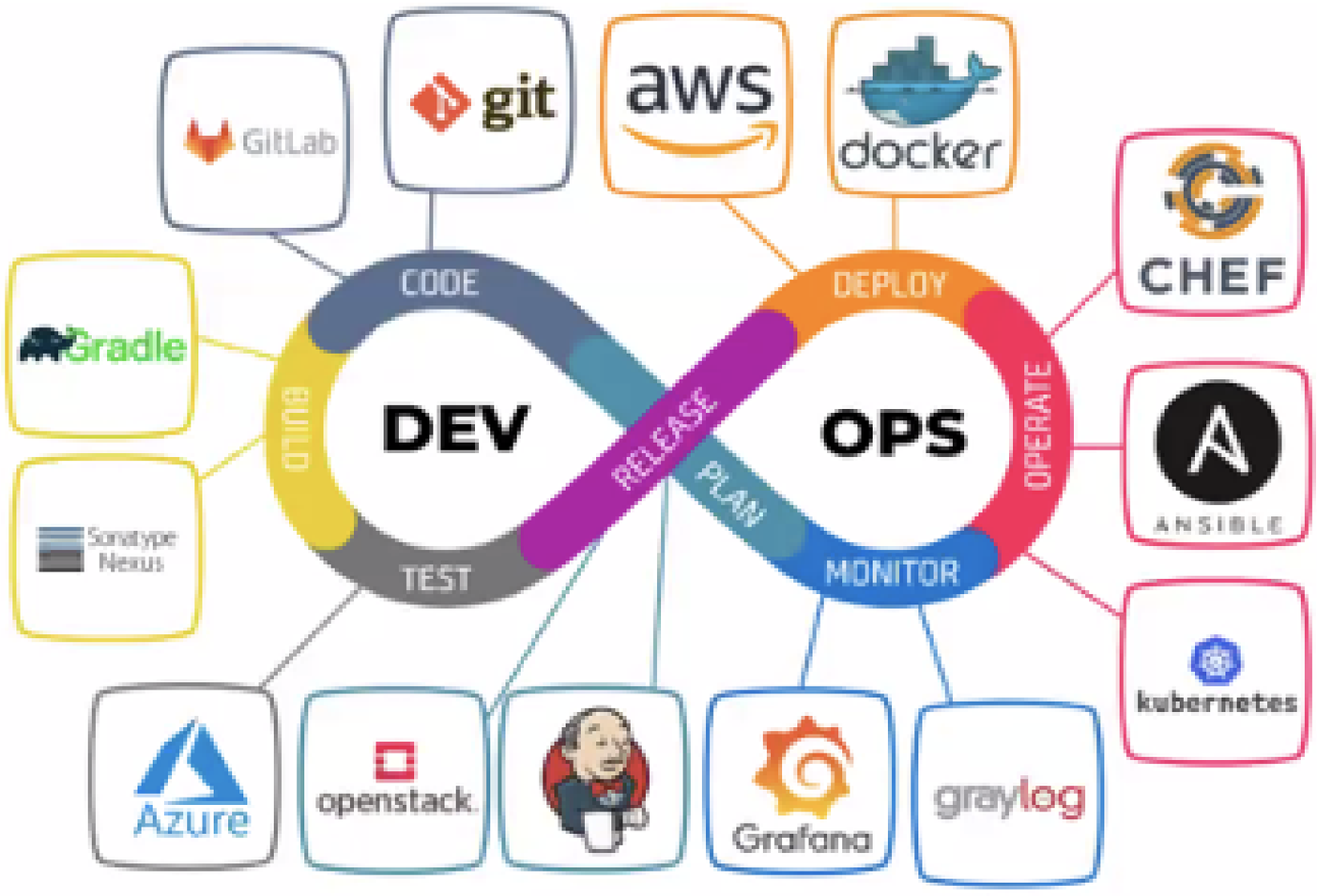
- Build. In this phase, you manage software builds and versions, and use automated tools to help compile and package code for future release to production. You use source code repositories or package repositories that also “ package” infrastructure needed for product release. Sample tools include Docker, Ansible, Puppet, Chef, Gradle, Maven, or JFrog Artifactory.

- Test. This phase involves continuous testing (manual or automated) to ensure optimal code quality. Sample tools include JUnit, Codeception, Selenium, Vagrant, TestNG, or BlazeMeter.

- Deploy. This phase can include tools that help manage, coordinate, schedule, and automate product releases into production. Sample tools include Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenStack, Docker, or Jira.

- Operate. This phase manages software during production. Sample tools include Ansible, Puppet, PowerShell, Chef, Salt, or Otter.

- Monitor. This phase involves identifying and collecting information about issues from a specific software release in production. Sample tools include New Relic, Datadog, Grafana, Wireshark, Splunk, Nagios, or Slack.



Configuration management

Why Do We Need a Version Control System?

Version control systems allow multiple developers, designers, and team members to work together on the same project. It helps them work smarter and faster! A version control system is critical to ensure everyone has access to the latest code and modifications are tracked. As development becomes increasing complex and teams grow, there’ s a bigger need to manage multiple versions and components of entire products.

What is GitHub?

Git is not the same as GitHub.
GitHub makes tools that use Git.

GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

.

Git Install

You can download Git for free from the following website: https://www.git-scm.com/

Fundamentals of Git

Here is a basic overview of how Git works:

1. Create a “ repository” (project) with a git hosting tool (like Bitbucket)
2. Copy (or clone) the repository to your local machine
3. Add a file to your local repo and “ commit” (save) the changes
4. “ Push” your changes to your main branch
5. Make a change to your file with a git hosting tool and commit
6. “ Pull” the changes to your local machine
7. Create a “ branch” (version), make a change, commit the change
8. Open a “ pull request” (propose changes to the main branch)
9. “ Merge” your branch to the main branch

Git Client installation and setup

Installation of Git Client

If you are using Debian base GNU/Linux distribution, then apt-get command

Will do the needful.

```
[ubuntu ~]$ sudo apt-get install git-core

[sudo] password for ubuntu:


[ubuntu ~]$ git – version

Git version 1.8.1.2
```

And if you are using RPM based GNU/Linux distribution, then use yum command

As given.

```
[CentOS ~]$

Su –

Password:

[CentOS ~]#yum -y install git-core


[CentOS ~]#git – version

Git version 1.7.1
```

Customize Git Environment

Git provides the git config tool, which allows you to set configuration variables.

Git stores all global configurations in .gitconfig file, which is located in your

Home directory. To set these configuration values as global, add the –

Global option, and if you omit – global option, then your configurations are Specific for the current Git repository.

You can also set up system wide configuration. Git stores these values in

The /etc/gitconfig file, which contains the configuration for every user and

Repository on the system. To set these values, you must have the root rights

And use the – system option.

When the above code is compiled and executed, it produces the following result:

Setting username

This information is used by Git for each commit.

```
[jerry@CentOS project]$ git config – global user.name“ Jerry Mouse”
```

Setting email id

This information is used by Git for each commit.

```
[jerry@CentOS project]$ git config – global user.email

“ jerry@tutorialspoint.c
```

Avoid merge commits for pulling

You pull the latest changes from a remote repository, and if these changes are

Divergent, then by default Git creates merge commits. We can avoid this via

Following settings.

```
jerry@CentOS project]$ git config –  global branch.autosetuprebase always
```

Color highlighting

The following commands enable color highlighting for Git in the console.

```
[jerry@CentOS project]$ git config –  global color.ui true


[jerry@CentOS project]$ git config –  global color.status auto


[jerry@CentOS project]$ git config –  global color.branch auto
```

Setting default editor

By default, Git uses the system default editor, which is taken from the VISUAL or

EDITOR environment variable. We can configure a different one by using git

Config.

Basic local Git operations

▪ creating a repository

Different ways to Create Git Repository

We can Create Git Repository using one of the three approaches enlisted below:

Create a bare repositor

Creating a Bare Git Repository for a new project is three-step process:

1) Create a New Project/Folder
2) Browse to New Project
3) Initialize Git Repository for the Project

Reference: https://www.toolsqa.com/git/create-git-repository/

Initialize repository in an existing project directory

Create Git Repository for an Existing Project

We'd also like to track an existing project by using Git. In this case, we'll initialize a Git repository in an existing project directory. There is no rocket science in creating git Repository for an existing project, it is as same as creating a git repository for a new project with the only difference of step 1 is not required:

I. Browse to Existing Project
II. Initialize Git Repository for the Project

Reference : https://www.toolsqa.com/git/create-git-repository/

Cloning a repository

i. On GitHub.com, navigate to the main page of the repository.
ii. Above the list of files, click Code.
iii. Copy the URL for the repository. To clone the repository using HTTPS, under “ HTTPS” , click . … Open .
iv. Change the current working directory to the location where you want the cloned directory

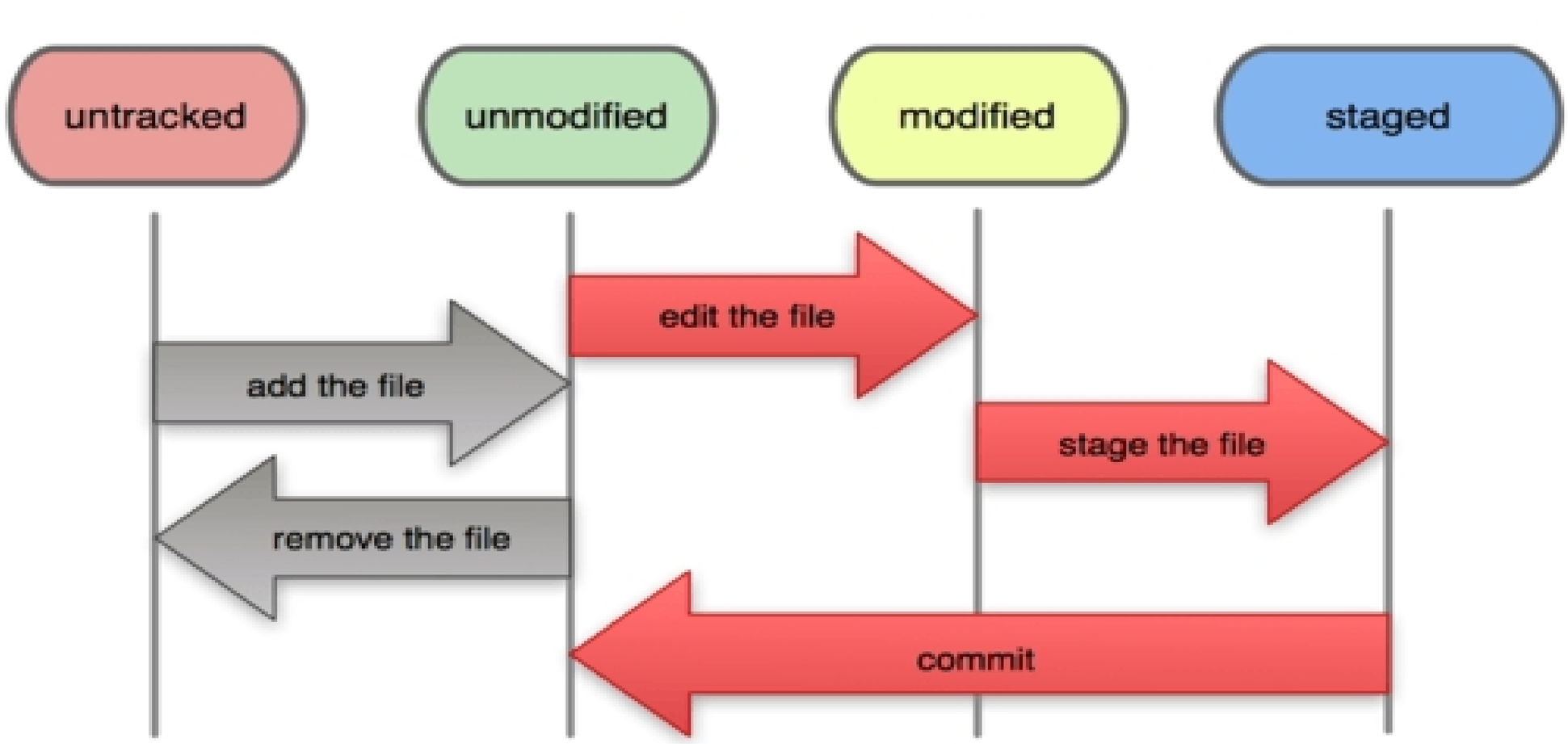
making and recording changes

You have a bona fide Git repository and a checkout or working copy of the files for that project. You need to make some changes and commit snapshots of those changes into your repository each time the project reaches a state you want to record.

Remember that each file in your working directory can be in one of two states: tracked or untracked. Tracked files are files that were in the last snapshot; they can be unmodified, modified, or staged. Untracked files are everything else — any files in your working directory that were not in your last snapshot and are not in your staging area. When you first clone a repository, all of your files will be tracked and unmodified because you just checked them out and haven’ t edited anything.

As you edit files, Git sees them as modified, because you’ ve changed them since your last commit. You stage these modified files and then

File Status Lifecycle



commit all your staged changes, and the cycle repeats. This lifecycle is illustrated in Figure 2-1.

staging and committing changes

Stage Files to Prepare for Commit

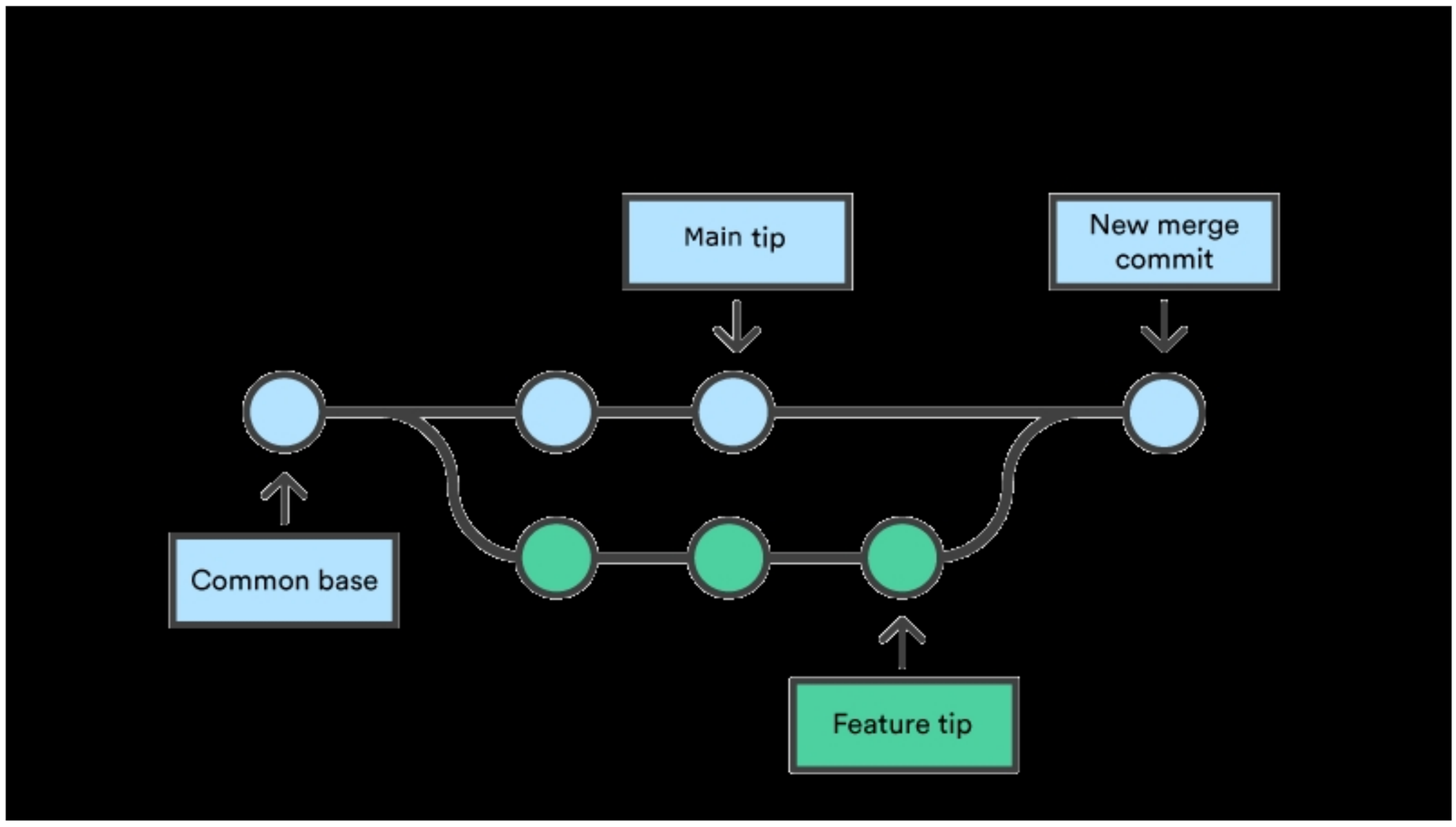
1. Enter one of the following commands, depending on what you want to do: Stage all files: git add . Stage a file: git add example. Html (replace example. …
2. Check the status again by entering the following command: git status.
3. You should see there are changes ready to be committed.

Viewing the history of all the changes Undoing changes

The git revert command is used for undoing changes to a repository’ s commit history. Other ‘ undo’ commands like, git checkout and git reset , move the HEAD and branch ref pointers to a specified commit. Git revert also takes a specified commit, however, git revert does not move ref pointers to this commit.

Git Branching and merging

Merging is Git’ s way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch. Note that all of the commands presented below merge into the current branch.



Basic

- **Creating and switching to new branches**

New Branches

The git branch command can be used to create a new branch. When you want to start a new feature, you create a new branch off main using git branch new_branch . Once created you can then use git checkout new_branch to switch to that branch.

- **Switching between branches**

You can use the git switch – command to undo any changes you make and return to your previous branch. If you instead want to keep your changes and continue from here, you can use git switch -c <new-branch-name> to create a new branch from this point.

- **Merging local branches together**

To merge branches locally, use git checkout to switch to the branch you want to merge into. This branch is typically the main branch. Next, use git

merge and specify the name of the other branch to bring into this branch. This example merges the jeff/feature1 branch into the main branch.

GitHub

- Basics of distributed git

Git is distributed version control software. Version control is a way to save changes over time without overwriting previous versions. Being distributed means that every developer working with a Git repository has a copy of that entire repository – every commit, every branch, every file.

Account creation and configuration

You typically configure your global username and email address after installing Git.

…

1.Toet your global username/email configuration:

2.Open the command line.

3.Set your username: git config – global user.name “ FIRST_NAME LAST_NAME”

4.Set your email address: git config – global user.email “ MY_NAME@example.com”

- Create and push to repositories

- **Create a new repository on GitHub.com. …**
- **Open TerminalTerminalGit Bash.**
- **Change the current working directory to your local project.**
- **Use the init command to initialize the local directory as a Git repository. …**
- **Add the files in your new local repository. …**

- **Commit the files that you’ ve staged in your local repository.**

- **Versioning**
  - **Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.**
- **Collaboration**
  - **Collaboration is the way different people can work on the same project together. It is like creating a group in GitHub just like Groups in other social media. The people added to the collaborator’ s list can be able to push, merge, and do other kinds of similar things on the project.**
- **Migration**
  - **We’ ve broken down the SVN-to-Git migration process into 5 simple steps: Prepare your environment for the migration. Convert the SVN repository to a local Git repository. Synchronize the local Git repository when the SVN repository changes. Share the Git repository with your developers via Bitbucket.**

CLOUD BASICS

Cloud infrastructure overview

Cloud computing infrastructure is the collection of hardware and software elements needed to enable cloud computing. It includes computing power, networking, and storage, as well as

an interface for users to access their virtualized resources. The virtual resources mirror a physical infrastructure, with components like servers, network switches, memory and storage clusters.]

Cloud computing architecture and its components

Cloud computing, which is one of the demanding technology of the current time and which is giving a new shape to every organization by providing on demand virtualized services/resources. Starting from small to medium and medium to large, every organization use cloud computing services for storing information and accessing it from anywhere and any time only with the help of internet. In this article, we will know more about the internal architecture of cloud computing.

Transparency, scalability, security and intelligent monitoring are some of the most important constraints which every cloud infrastructure should experience. Current research on other important constraints is helping cloud computing system to come up with new features and strategies with a great capability of providing more advanced cloud solutions.

Cloud Computing Architecture :

The cloud architecture is divided into 2 parts i.e.

1. Frontend
2. Backend

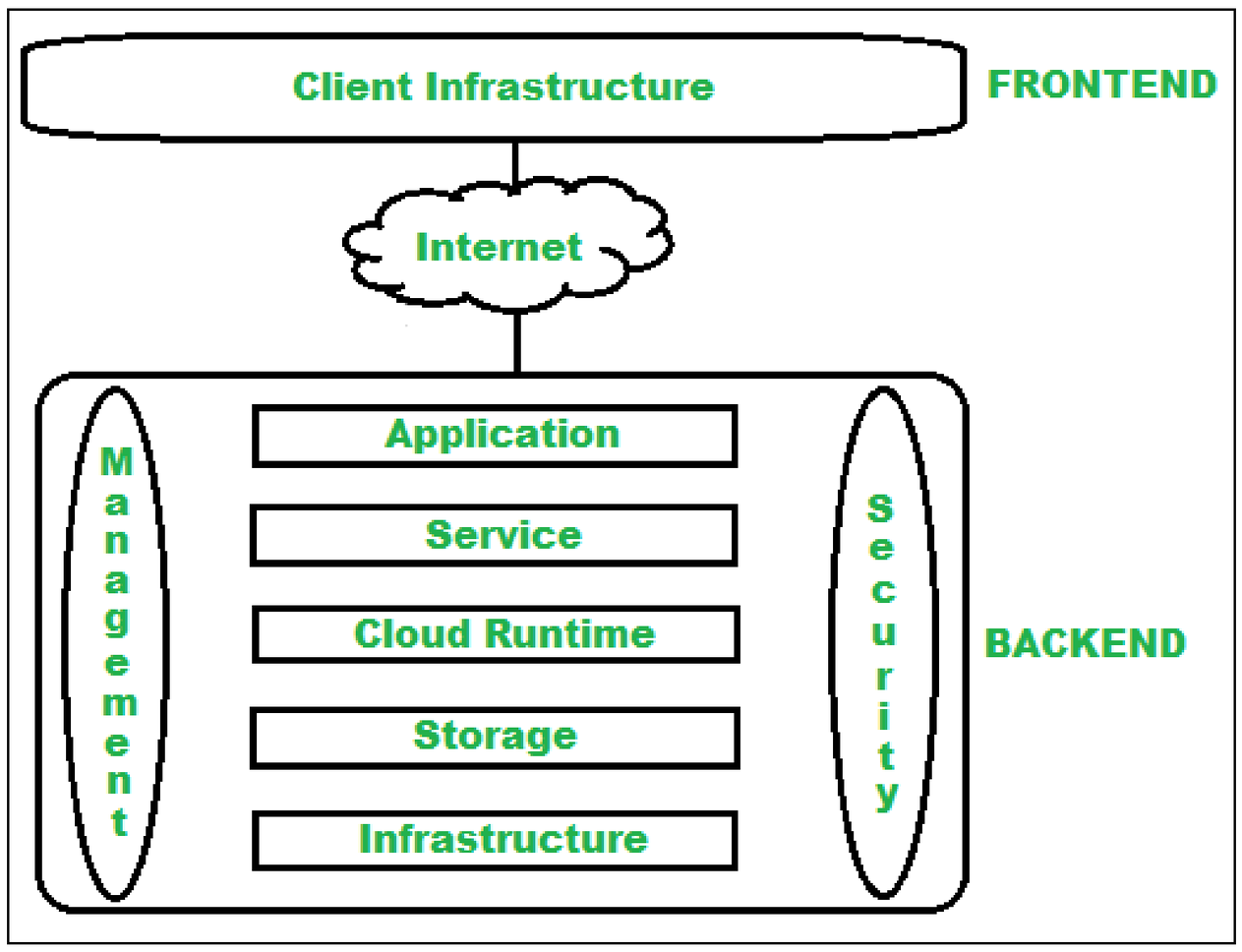
The below figure represents an internal architectural view of cloud computing.

Front End:

The client uses the front end, which contains a client-side interface and application. Both of these components are important to access the Cloud computing platform. The front end includes web servers (Chrome, Firefox, Opera, etc.), clients, and mobile devices.

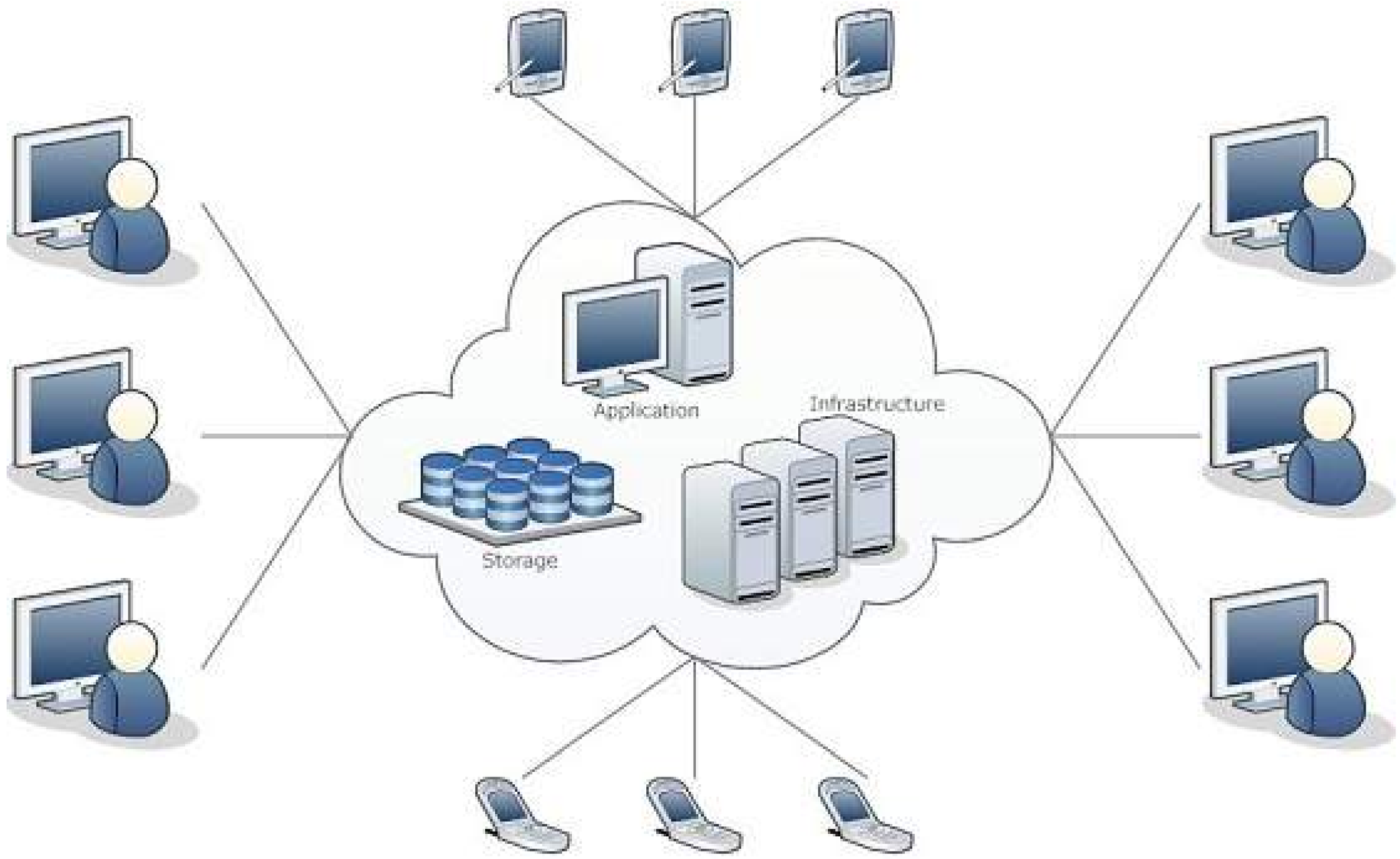
Back End:

The backend part helps you manage all the resources needed to provide Cloud computing services. This Cloud architecture part includes a security mechanism, a large amount of data storage, servers, virtual machines, traffic control mechanisms, etc.



What is Cloud Computing?

Cloud Computing refers to **manipulating, configuring,** and **accessing** the hardware and software resources remotely. It offers online data storage, infrastructure, and application.



Cloud computing offers **platform independency,** as the software is not required to be installed locally on the PC. Hence, the Cloud Computing is making our business applications **mobile** and **collaborative.**

Basic Concepts

There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

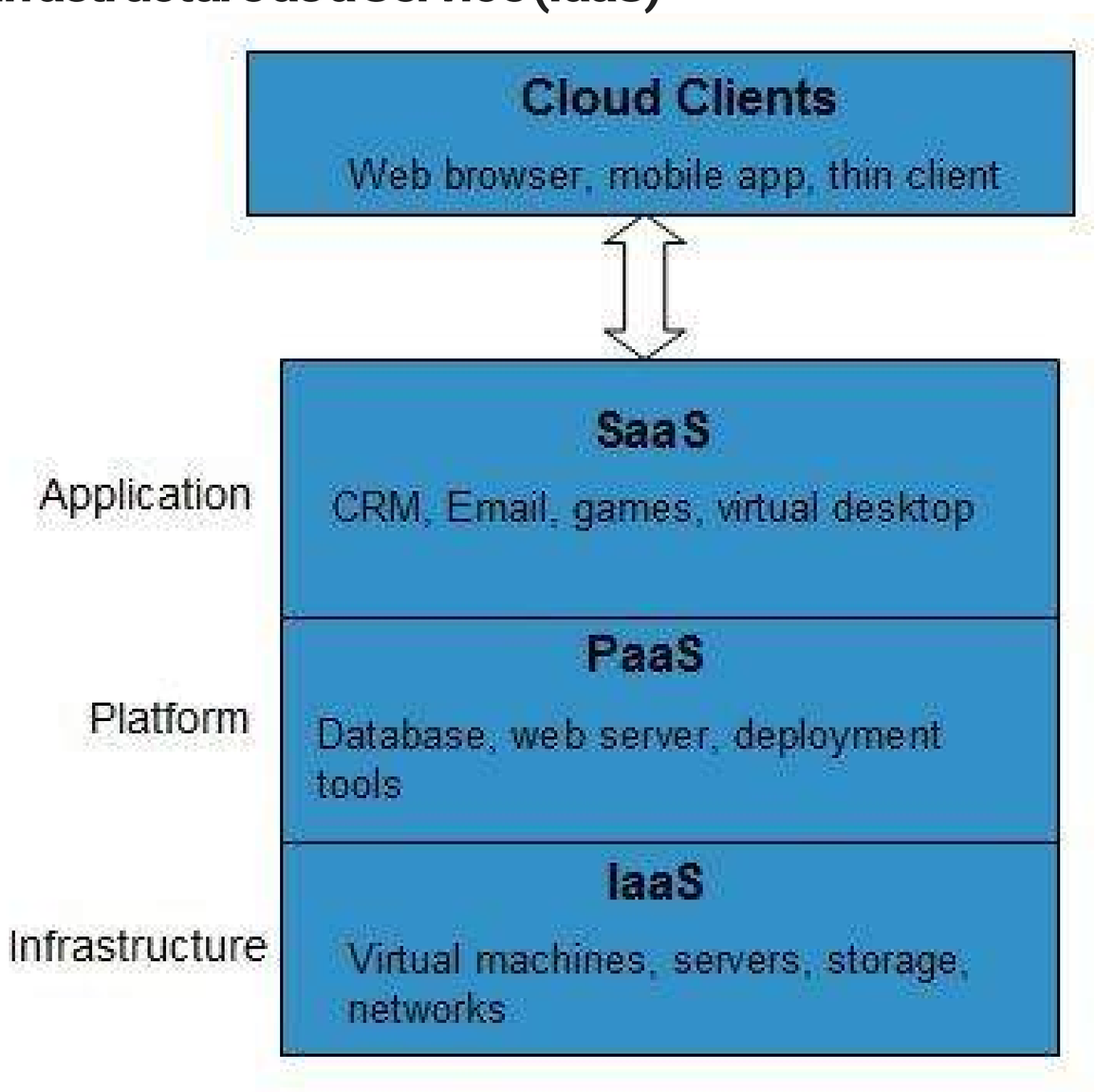
Deployment Models

Service Models

Services models

The three major Cloud Computing Offerings are

Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)



Different business use some or all of these components according to their requirement.

SaaS (Software as a Service)

SaaS or software as a service is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network (internet). SaaS is becoming an increasingly prevalent delivery model as underlying technologies that supports **Service Oriented Architecture (SOA) or Web Services**. Through internet this service is available to users anywhere in the world.

Traditionaly, software application needed to be purchased upfront &then installed it onto your computer. SaaS users on the other hand, instead of purchasing the software subscribes to it, usually on monthly basisvia internet.

Anyone who needs an access to a particular piece of software can be subscribe as a user, whether it is one or two people or every thousands of employees in a corporation. SaaS is compatible with all internet enabled devices.

Many important tasks like accounting, sales, invoicing and planning all can be performed using SaaS.

PaaS (Platform as a Service)

Platform as a service, is referred as PaaS, it provides a platform and environment to allow developers to build applications and services. This service is hosted in the cloud and accessed by the users via internet.

To understand in a simple terms, let compare this with painting a picture, where you are provided with paint colors, different paint brushes and paper by your school teacher and you just have to draw a beautiful picture using those tools.

PaaS services are constantly updated & new features added. Software developers, web developers and business can benefit from PaaS. It provides platform to support application development. It includes software support and management services, storage, networking, deploying, testing, collaborating, hosting and maintaining applications.

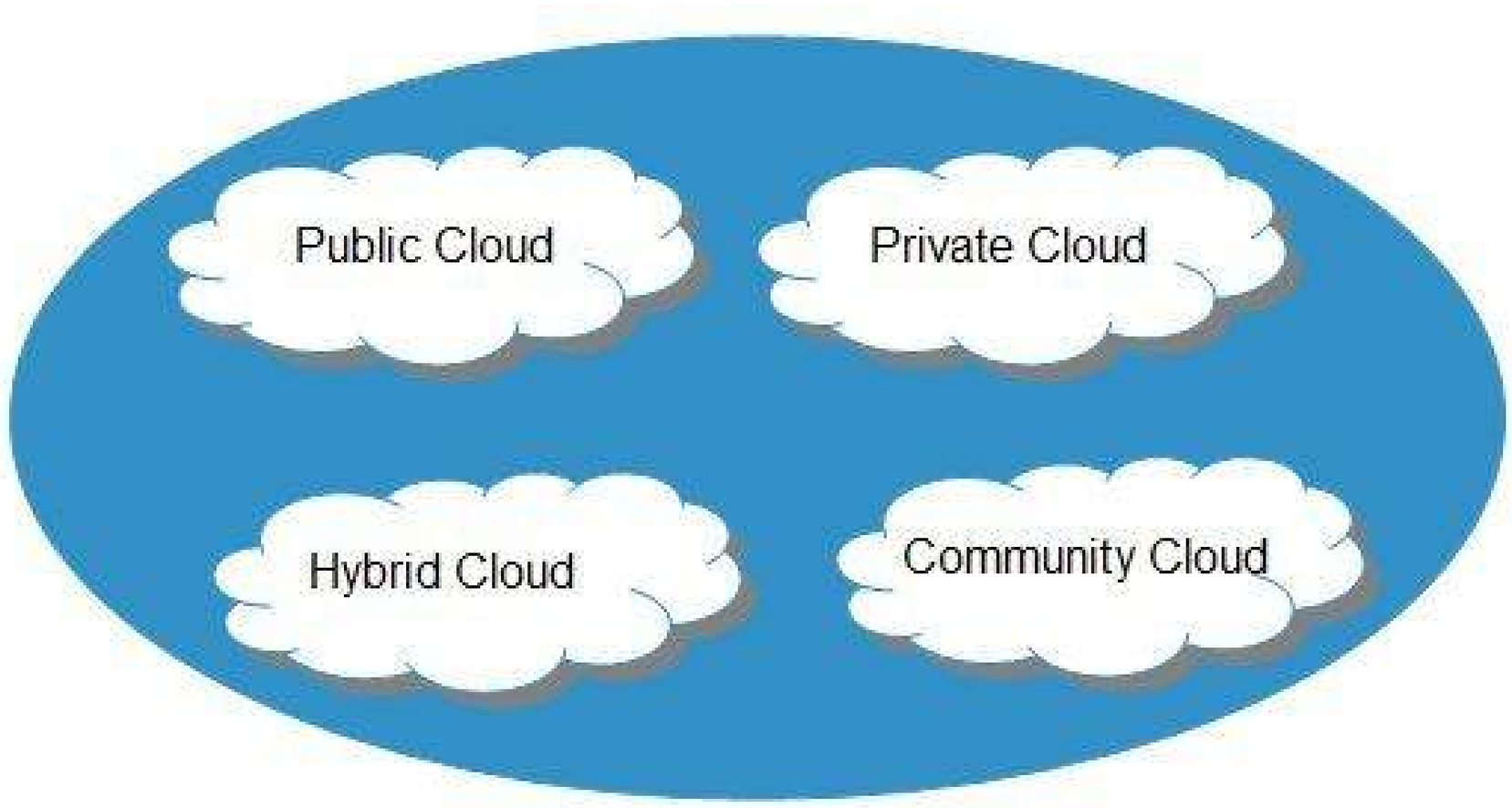
IaaS (Infrastructure as a Service)

IaaS (Infrastructure As A Service) is one of the fundamental service model of cloud computing alongside PaaS(Platform as a Service). It provides access to computing resources in a virtualized environment “ the cloud” on internet. It provides computing infrastructure like virtual server space, network connections, bandwidth, load balancers and IP addresses. The pool of hardware resource is extracted from multiple servers and networks usually distributed across numerous data centers. This provides redundancy and reliability to IaaS.

IaaS(Infrastructure as a service) is a complete package for computing. For small scale businesses who are looking for cutting cost on IT infrastructure, IaaS is one of the solutions. Annually a lot of money is spent in maintenance and buying new components like hard-drives, network connections, external storage device etc. which a business owner could have saved for other expenses by using IaaS.

Deployment Models

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid, and Community.



Public Cloud

The **public cloud** allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness.

Private Cloud

The **private cloud** allows systems and services to be accessible within an organization. It is more secured because of its private nature.

Community Cloud

The **community cloud** allows systems and services to be accessible by a group of organizations.

Hybrid Cloud

The **hybrid cloud** is a mixture of public and private cloud, in which the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

Important Components of Cloud Computing Architecture

Here are some important components of Cloud computing architecture:

1. Client Infrastructure:

Client Infrastructure is a front-end component that provides a GUI. It helps users to interact with the Cloud.

2. Application:

The application can be any software or platform which a client wants to access.

3. Service:

The service component manages which type of service you can access according to the client’ s requirements.

Three Cloud computing services are:

Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)

4. Runtime Cloud:

Runtime cloud offers the execution and runtime environment to the virtual machines.

5. Storage:

Storage is another important Cloud computing architecture component. It provides a large amount of storage capacity in the Cloud to store and manage data.

6. Infrastructure:

It offers services on the host level, network level, and application level. Cloud infrastructure includes hardware and software components like servers,

storage, network devices, virtualization software, and various other storage resources that are needed to support the cloud computing model.

7. Management:

This component manages components like application, service, runtime cloud, storage, infrastructure, and other security matters in the backend. It also establishes coordination between them.

8. Security:

Security in the backend refers to implementing different security mechanisms for secure Cloud systems, resources, files, and infrastructure to the end-user.

9. Internet:

Internet connection acts as the bridge or medium between frontend and backend. It allows you to establish the interaction and communication between the frontend and backend.

Benefits of Cloud Computing Architecture :

Makes overall cloud computing system simpler.
Improves data processing requirements.
Helps in providing high security.
Makes it more modularized.
Results in better disaster recovery.
Gives good user accessibility.
Reduces IT operating cost.

Virtualization and Cloud Computing

The main enabling technology for Cloud Computing is Virtualization. Virtualization is the partitioning of a single physical server into multiple logical servers. Once the physical server is divided, each logical server behaves like a physical server and can run an operating system and applications independently. Many popular companies like VMware and

Microsoft provide virtualization services. Instead of using your PC for storage and computation, you can use their virtual servers. They are fast, cost-effective, and less time-consuming.

Network Virtualization: It is a method of combining the available resources in a network by splitting up the available bandwidth into channels. Each channel is independent of others and can be assigned to a specific server or device in real time.

Storage Virtualization: It is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console. Storage virtualization is commonly used in storage area networks (SANs).

Server Virtualization: Server virtualization is the masking of server resources like processors, RAM, operating system, etc., from server users. Server virtualization intends to increase resource sharing and reduce the burden and complexity of computation from users.

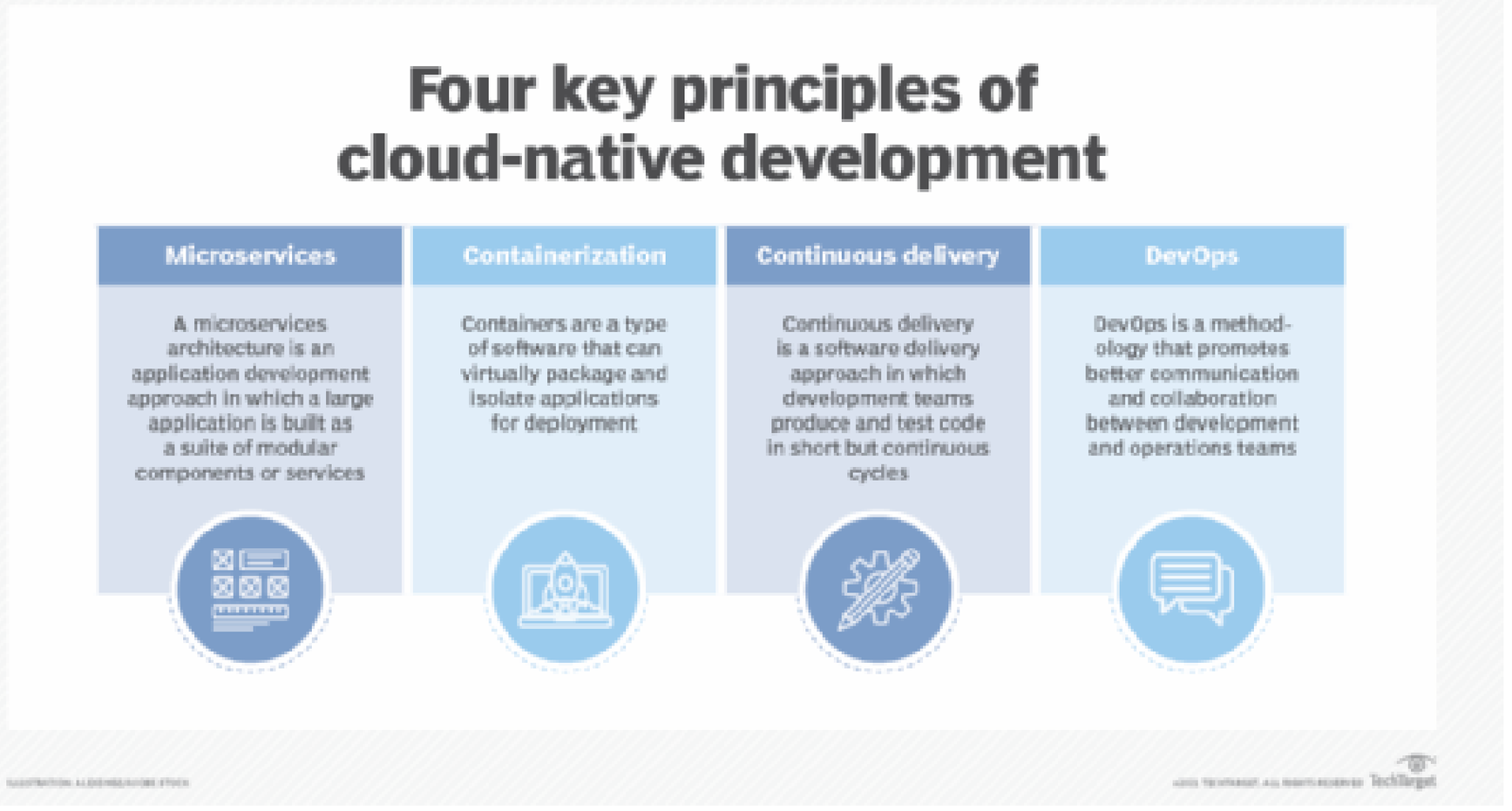
cloud native application deveploment

The term cloud native refers to the concept of building and running applications to take advantage of the distributed computing offered by the cloud delivery model. Cloud native apps are designed and built to exploit the scale, elasticity, resiliency, and flexibility the cloud provides.

As defined by the Cloud Native Computing Foundation (CNCF), Cloud native technologies empower organizations to build and run scalable applications in public, private, and hybrid clouds. Features such as containers, service meshes, microservices,

immutable infrastructure, and declarative application programming interfaces (APIs) best illustrate this approach.

These features enable loosely coupled systems that are resilient, manageable, and observable. They allow engineers to make high-impact changes frequently and with minimal effort.



Continuous integration

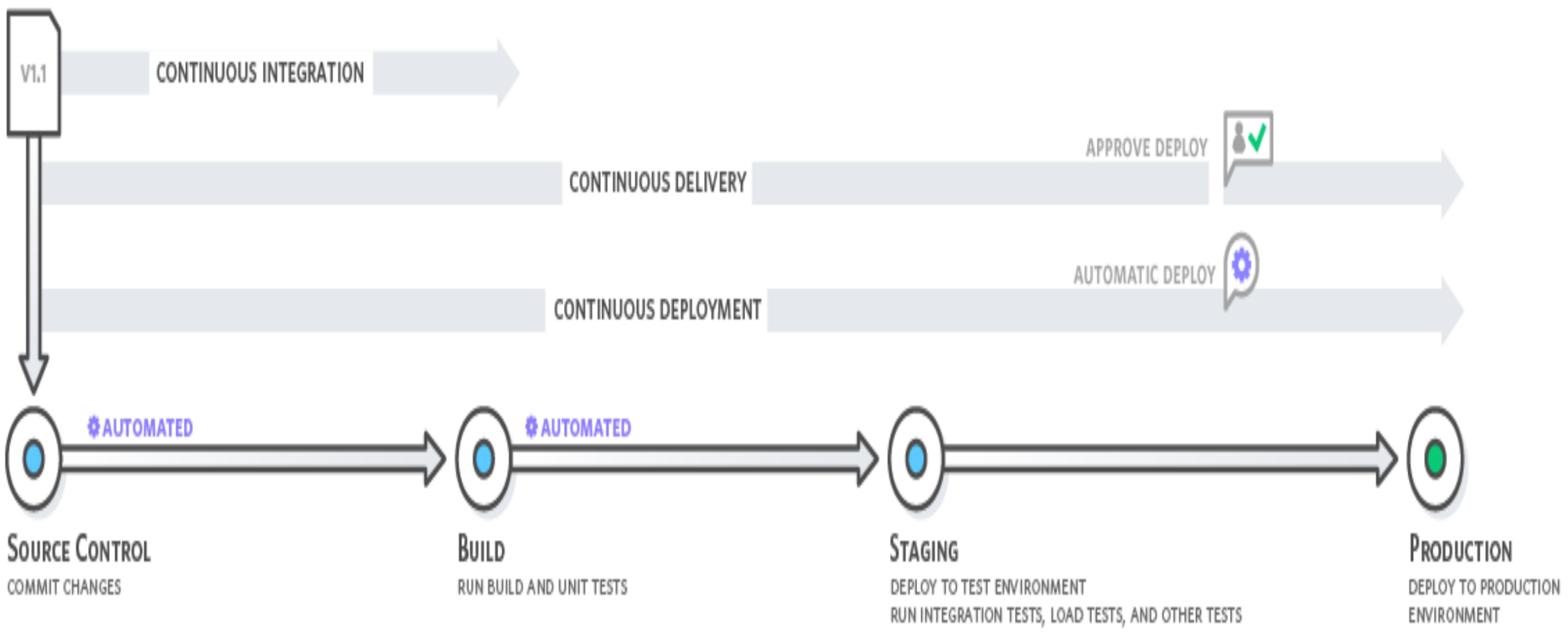
Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often

refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

CI/CD tool

CI/CD tools can help a team automate their development, deployment, and testing. Some tools specifically handle the integration (CI) side, some manage development and deployment (CD), while others specialize in continuous testing or related functions.

One of the best known open source tools for CI/CD is the automation server Jenkins. Jenkins is designed to handle anything from a simple CI server to a complete CD hub.



Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test.

With continuous delivery, code changes are automatically built, tested, and prepared for a release to production. Continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

https://aws.amazon.com/devops/continuous-integration/

What is Jenkins?

Jenkins is an open-source Continuous Integration server written in Java for orchestrating a chain of actions to achieve the Continuous Integration process in an automated fashion. Jenkins supports the complete development life cycle of software from building, testing, documenting the software, deploying, and other stages of the software development life cycle.

How does Jenkins work?

Jenkins is a server-based application and requires a web server like Apache Tomcat to run on various platforms like Windows, Linux, macOS, Unix, etc. To use Jenkins, you need to create pipelines which are a series of steps that a Jenkins server will take. Jenkins Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to host, monitor, compile and test code, or code changes, like:

Continuous Integration Server (Jenkins, Bamboo, CruiseControl, TeamCity, and others)
Source Control Tool (e.g., CVS, SVN, GIT, Mercurial, Perforce, ClearCase and others)
Build tool (Make, ANT, Maven, Ivy, Gradle, and others)
Automation testing framework (Selenium, Appium, TestComplete, UFT, and others)

Why use Continuous Integration with Jenkins?

Some people might think that the old-fashioned way of developing the software is the better way. Let’ s understand the advantages of CI with Jenkins with the following example

Let us imagine, that there are around 10 developers who are working on a shared repository. Some developer completes their task in 25 days while others take 30 days to complete.

Before Jenkins	After Jenkins

Once all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.	The code is built and test as soon as Developer commits code Jenkin will build and test code many times during the day If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team. If the build fails, then Jenkins will notify the errors to the developer team.
Code commit built, and test cycle was very infrequent, and a single build was done after many days.	
Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build	The code is built immediately after any of the Developer commits.
It is not an easy task to isolate, detect, and fix errors for multiple commits.	Since the code is built after each commit of a single developer, it’ s easy to detect whose code caused the built to fail
Code build and test process are entirely manual, so there are a lot of chances for failure.	Automated build and test process saving timing and reducing defects.
The code is deployed once all the errors are fixed and tested.	The code is deployed after every successful build and test.
Development Cycle is slow	The development cycle is fast. New features are more readily available to users. Increases profits.

Comparison of cloud service

Full Stack Developers and DevOps

Both are, in a sense, two sides of the same coin. It's difficult to decide which side to play on. Both are highly able experts. So, they strive to deliver software on time and with zero errors. What separates them from one another are the ideas and tactics them to meet their aims.

They are both giving contributors to the growing firm. So, they desire more flexibility and agility. Let's look at the following points to better grasp the parallels between both Coders.

Before fixing which one to hire. So, it's a good idea to figure out what your firm's current needs are. The following are the critical factors that must conclude.