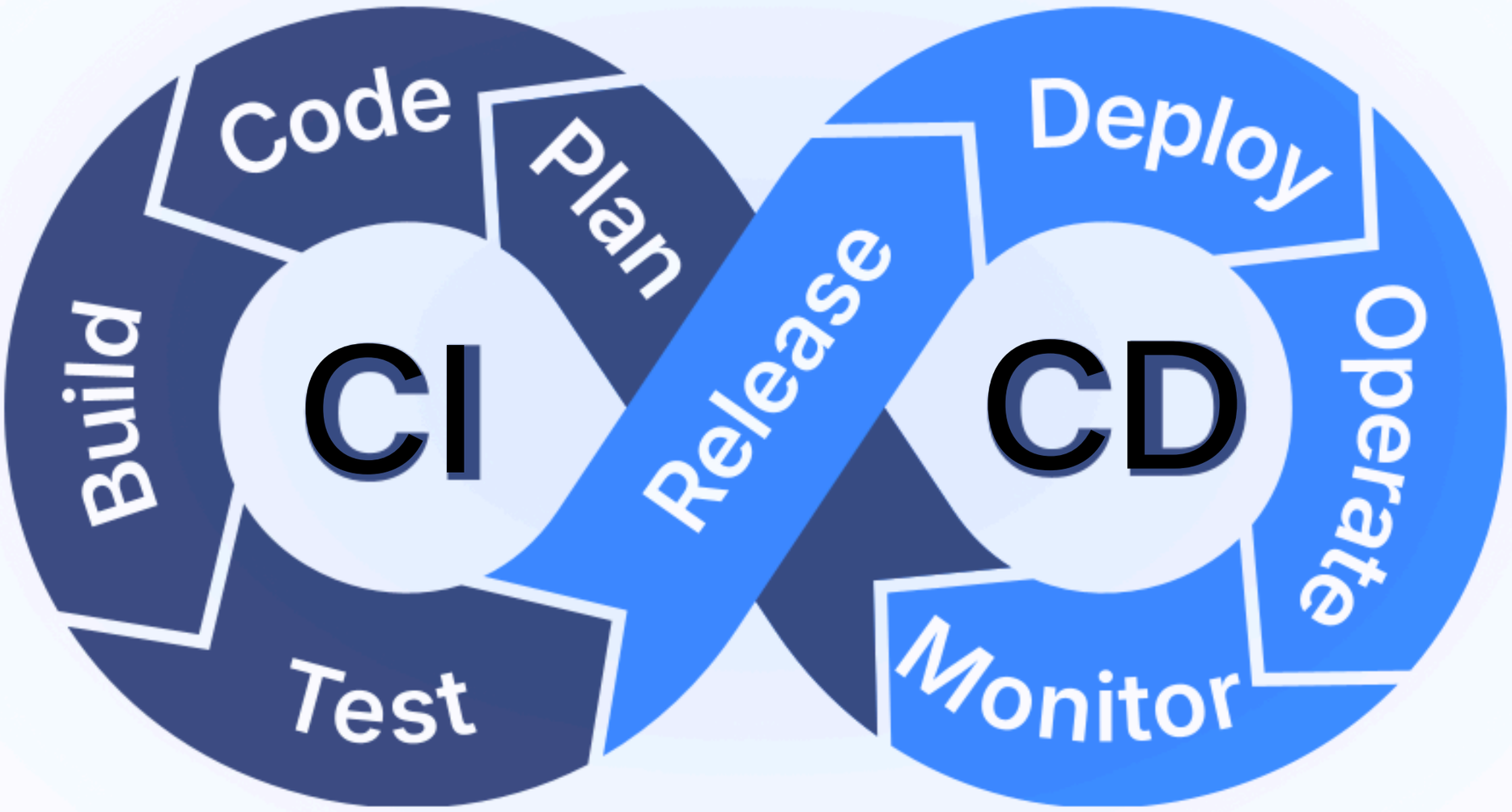
CI/CD

PIPELINE



A Data Scientist's Backbone!

Disclaimer

Everyone learns uniquely.

What matters is developing the problem solving ability to solve new problems.

This Doc will help you with the same.

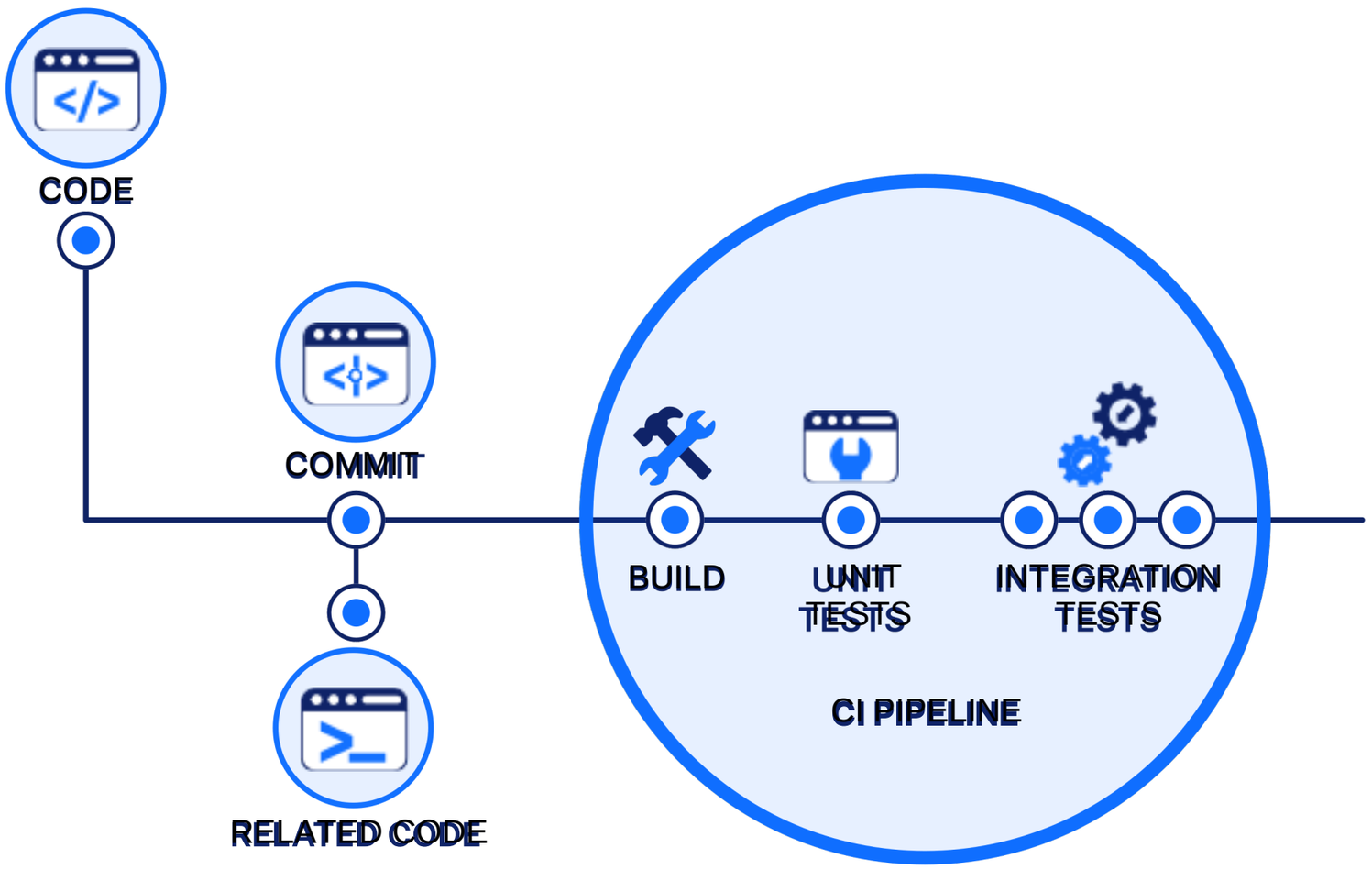
What is the CI/CD Pipeline?

CI and CD is the process in which we automate the integration of code changes from multiple developers into a single codebase.
It is actually a software development practice where the developers commit their work frequently to the central code repository-Github or Stash.

The key goals of Continuous Integration are:

1. To find and remove bugs quicker.
2. Make the process of integrating code across a team of developers easier.
3. Improving software quality.
4. Reducing the time it takes to release new feature updates.

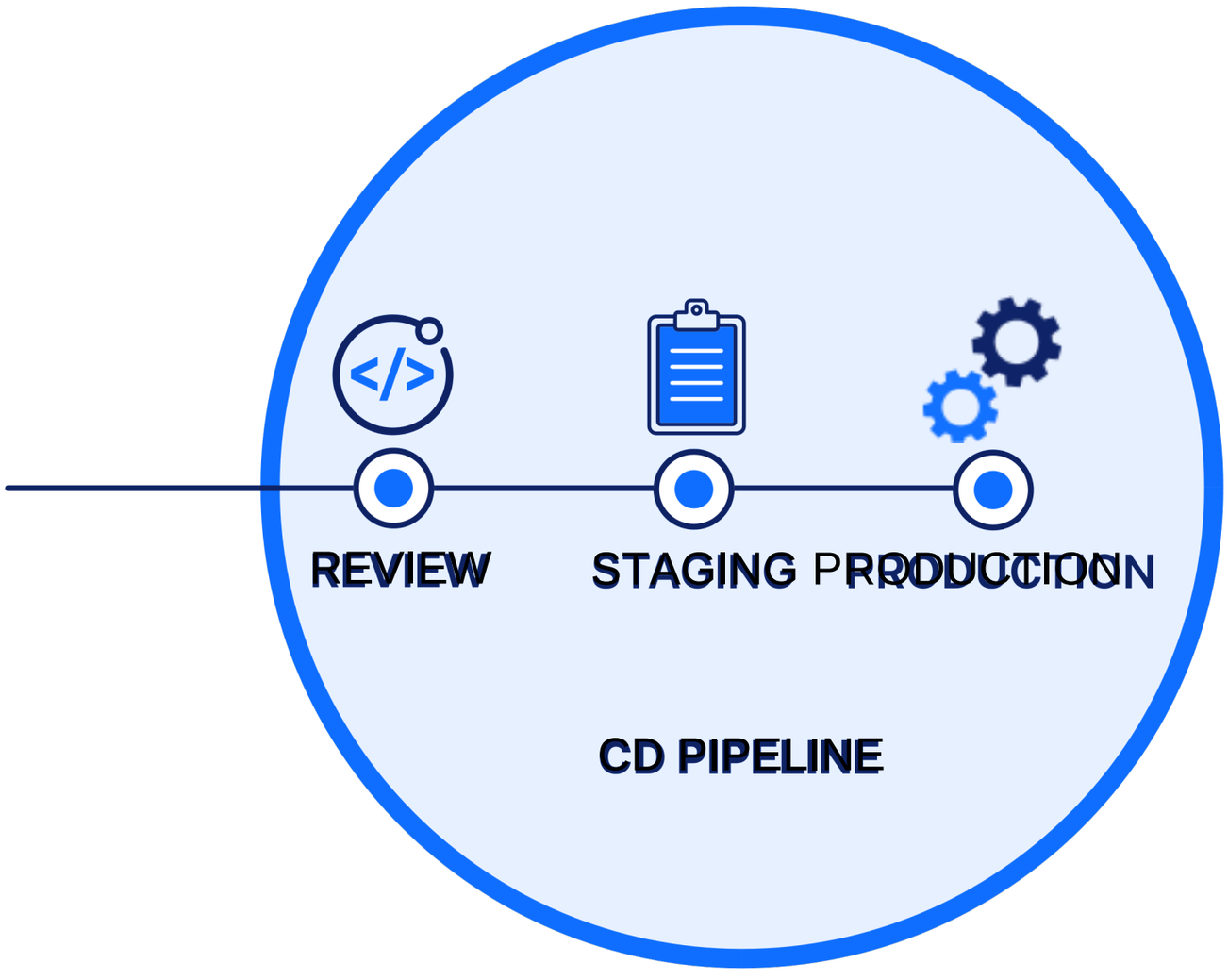
Continuous Integration



Continuous Integration (CI) is the process of merging code changes from various developers into a shared central repository often known as the main repository. The integration is then automatically tested using an automated build and test process.

- The objective of CI is to detect defects as early as possible and maintain a codebase which is consistent as well as deployable.
- Developers commit changes to code in a centralized repository often known as the main repository quite regularly.
- Automated builds and tests detect errors early and improve code quality so that it is consistent.

Continuous Deployment/Continuous Delivery



Continuous Delivery automates the process of testing and staging, followed by manual approval for release, after that using Continuous Deployment we deploy every verified change directly to production.

- **Continuous Delivery**: Here testing is automated and code is ready for manual approval before deployment.
- **Continuous Deployment**: Here the entire process is automated so that changes are released directly into production.

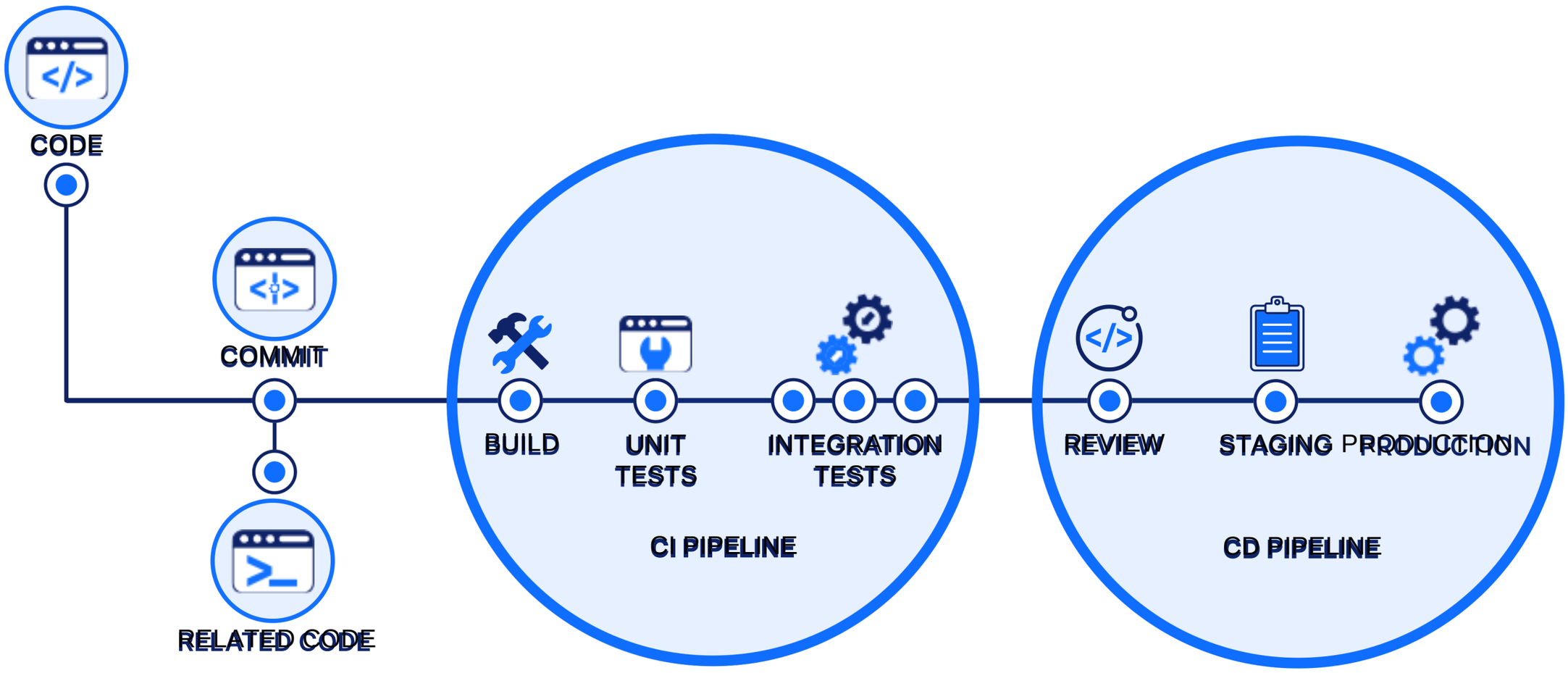
Advantages of CI/CD

CI/CD offer us multiple benefits some of them are as following:

1. **Faster Releases** – Since the repetitive tasks are automated therefore it enables quick and rapid deployment.
2. **Better Code Quality** – Testing process is automated which detects issues early in development.
3. **Enhanced Collaboration** – Developers commit change frequently, reducing conflicts.
4. **Lower Deployment Risks** – Continuous monitoring and automated tests minimize failures.
5. **Quick Rollbacks** – Ability to restore previous versions which are at some saved checkpoints if an issue occurs.
6. **Time Efficient** –Since the time spent on debugging and testing is automated , it is highly time efficient.

- **Scalability** – It also supports multiple teams working on complex applications at the same time.

STEPS INVOLVED IN CI/CD PIPELINING:-



1. Continuous Integration (CI)

CI focuses on automating code integration, building, and testing. These are the steps involved:

STEP 1: Code Commit

- Firstly, Developers push their code changes to a version control system for example *GitHub*, GitLab, Bitbucket etc.
- Code is then merged into a shared central repository which is often known as the main or develop branch.

STEP 2: Automated Build

- The CI server for example Jenkins, GitHub Actions, *GitLab* CI/CD pulls the latest code.
- Dependencies are installed, and the application is compiled (if necessary).

STEP 3: Static Code Analysis & Security Checks

- Tools like SonarQube, ESLint, or Bandit analyze code quality.
- Security scanning tools check for vulnerabilities (e.g., Snyk, OWASP Dependency Check).

STEP 4: Automated Testing

- Unit tests, integration tests, and API tests are executed using frameworks like JUnit, Jest, or PyTest.
- If tests fail, the pipeline stops and notifies the developers.

2. Continuous Deployment/Delivery (CD)

CD focuses on automating the release and deployment process.

STEP 5: Artifact Creation & Storage

- Successful builds generate artifacts (JAR, Docker images, etc.).
- Artifacts are stored in a registry (e.g., Docker Hub, Nexus, AWS S3).

STEP 6: Staging Deployment

- The application is deployed to a staging environment for further testing.
- Performance, security, and user acceptance testing (UAT) may be performed.

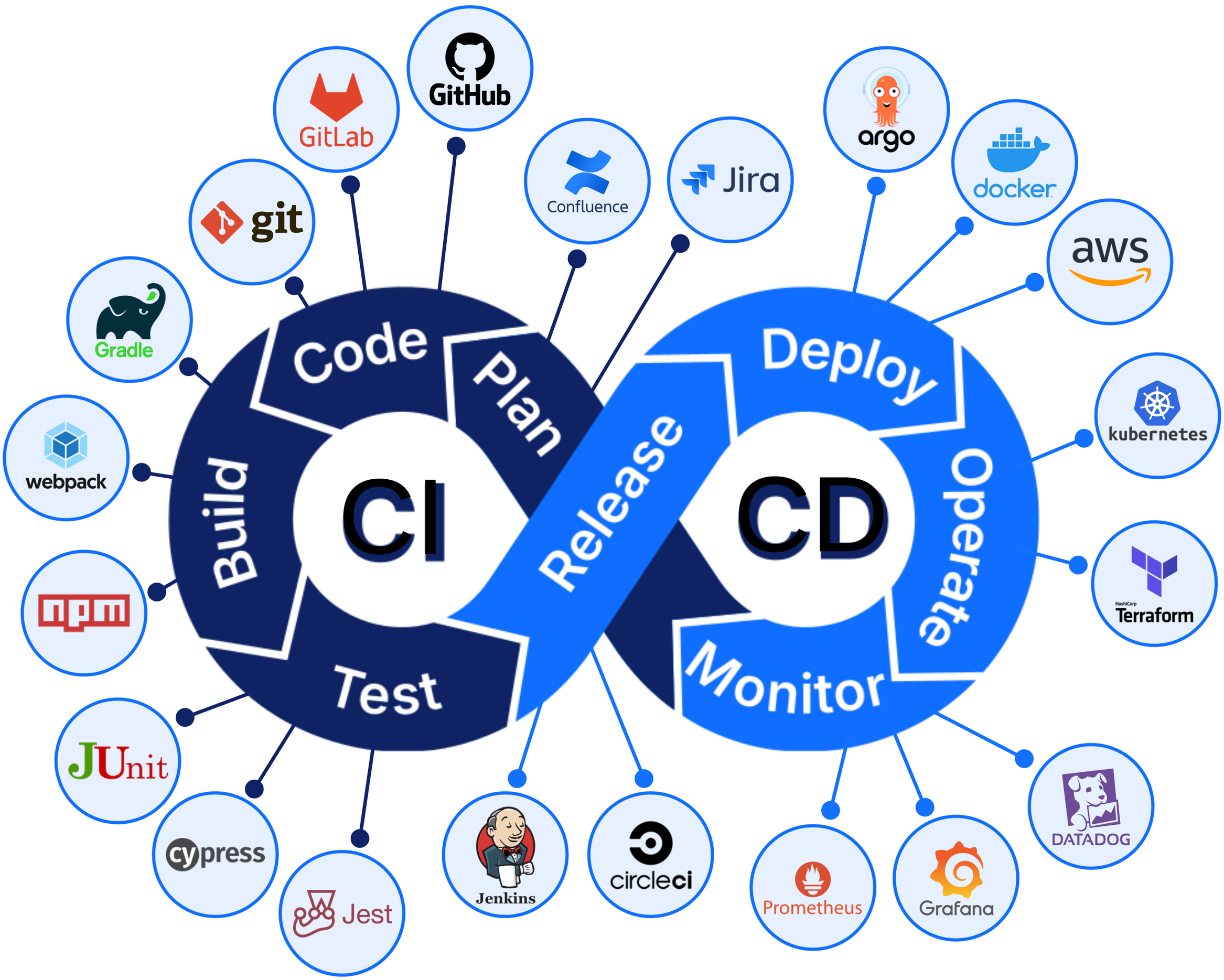
STEP 7: Continuous Deployment (Optional)

- If fully automated, the pipeline deploys the application to production after passing all tests.
- Canary releases or blue-green deployment strategies minimize risk.

STEP 8: Monitoring & Rollbackt

- Tools like Prometheus, *Datadog*, and New Relic monitor application performance.
- Rollback mechanisms allow reverting to a previous stable version if issues arise.

CI/CD Tools:


GitLab
GitHub
Confluence
Jira
argo
docker
git
aws
Gradle
Code
Plan
Build
Release
Deploy
Operate
Monitor
CI
CD
kubernetes
webpack
npm
HashiCorp
Terraform
JUnit
cypress
Jest
Jenkins
circleci
Prometheus
Grafana
DATADOG

Some commonly used CI/CD tools include:

- **CI Tools**: Jenkins, GitHub Actions, GitLab CI/CD, CircleCI
- **Testing**: Selenium, JUnit, Jest, PyTest
- **Containerization & Orchestration**: Docker, Kubernetes
- **Monitoring & Logging**: Prometheus, ELK Stack, Datadog
- **Rollback & Recovery**: Kubernetes Rollbacks ,Argo Rollouts, Spinnaker Rollbacks
- **Deployment Tools**: ArgoCD, Spinnaker, FluxCD, Octopus Deploy

Top 5 interview questions based on CI/CD Pipeline

Question- 1

State difference between docker and container.

Docker is a platform to build, ship and run applications inside container.It provides tools and APIs to easily manage containers easily.
Container is a lightweight, standalone executable unit that contains application code and dependencies.Isolated runtime environment for applications.

Question- 2

State difference between CI/CD vs DevOps.

CI/CD is the methodology of automation for software development and deployment. It focuses on automation of code integration, testing, and deployment.
Devops is a culture as well as set of practices, integrating Dev and Ops teams.It covers the full software development lifecycle (SDLC).

Question- 3

What is CI/CD pipeline?

A CI/CD pipeline is the process that automates and streamlines the software development process, such as testing, deployment, etc.

It consists of the following parts:`

1. **Continuous Integration(CI)**: It automates the integration of code, building, and testing.
2. **Continuous Delivery(CD)**: It automates software deployment up to staging/testing.
3. **Continuous Deployment(CD)**: It will fully automate deploying into production.

Question- 4

What is the importance of DevOps?

DevOps is essential because it:

1. **Increases the Speed of Deployment** –It automates the software release.
2. **Improves Software Quality** –It includes testing in the pipeline.
3. **Improves Collaboration** –It involves developers and operations teams working together.
4. **It ensures reliability** -It utilizes strategies of monitoring and rollback.
5. **Scalability increases** –It also supports cloud-native and microservices architectures.

Question- 5

Describe the build stage.

The Build Stage in a CI/CD pipeline is responsible for:

1. **Compiling Source Code** – This involves converting code (Java, Python, C++) into executable binaries.
2. **Managing Dependencies** –This involves Installing required libraries and frameworks.
3. **Code Analysis** – Static code analysis for security and quality checks.
4. **Creating Artifacts** –This involves Generating deployable packages (JAR, WAR, Docker images).
5. **Storing Build Output** – This involves uploading built artifacts to a repository (e.g., JFrog Artifactory, Nexus).

Some projects based on CI/CD Pipeline

GitLab CI/CD Examples

GitLab offers a set of CI/CD examples for several use cases, such as deployment with Dpl, GitLab Pages, multi-project pipelines, and more. These examples will help you understand how to implement GitLab CI/CD for your specific needs.

Example Here

2. CI/CD Pipeline with GitHub Actions for a .NET 7.0 Durable Function

In this example, it is discussed how to follow the implementation process of a CI/CD pipeline with GitHub Actions for a Durable Function App of.NET 7.0. Included are building up the application and publishing artifacts towards deploying to Azure Functions.

Example Here

Real-World CI/CD Pipeline Project

It illustrates a complete CI/CD pipeline with Jenkins, Maven, Ansible, SonarQube, Nexus, Prometheus, Grafana, and Slack for sending notifications.

Example Here

4. CI/CD Pipeline Using GitHub Actions

This is a project where a CI/CD pipeline can be demonstrated using GitHub Actions that shows how one can build, test, and deploy a sample application automatically.

Example Here

5. CI/CD Pipeline with CircleCI

This is a project where a CI/CD pipeline can be demonstrated using GitHub Actions that shows how one can build, test, and deploy a sample application automatically.

Example Here

Here are some certification courses for CI/CD Pipelining

aws-amazon-certification-certified-devops-engineer-professional

aws-amazon-ci-cd

microsoft-certifications-devops-engineer

university.gitlab-certifications

Conclusion:

This documentation will help working professionals gather essential information about **CI/CD** and its significance in modern software development. CI/CD is a continuously evolving technology that plays a crucial role in automating software delivery, improving code quality, and accelerating development cycles. As organizations increasingly adopt DevOps practices, mastering CI/CD becomes essential for staying competitive in the industry. By upskilling in this field, professionals can enhance their technical expertise, streamline development workflows, and contribute to building robust, scalable, and efficient software solutions.