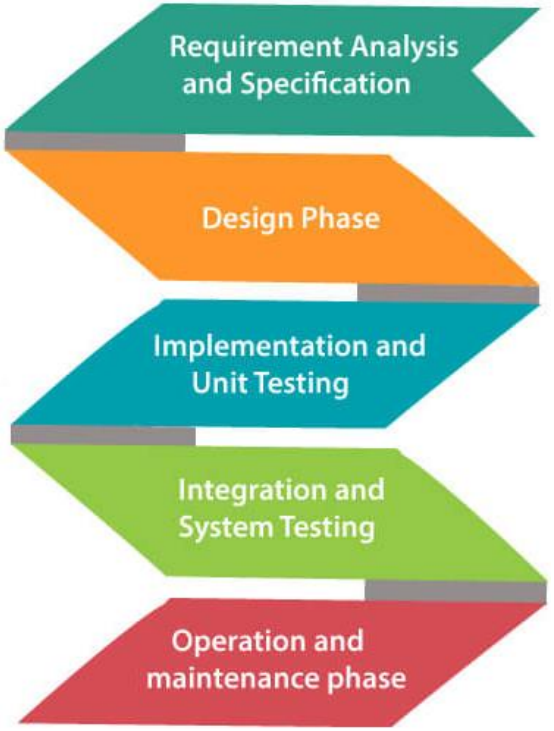
DEVOPS

Presented by

Jitendra Singh Tomar

What we had before DevOps?

- Traditionally, we had **Waterfall Model.**
- In this model, next stage will only start when the earlier stage is completed.



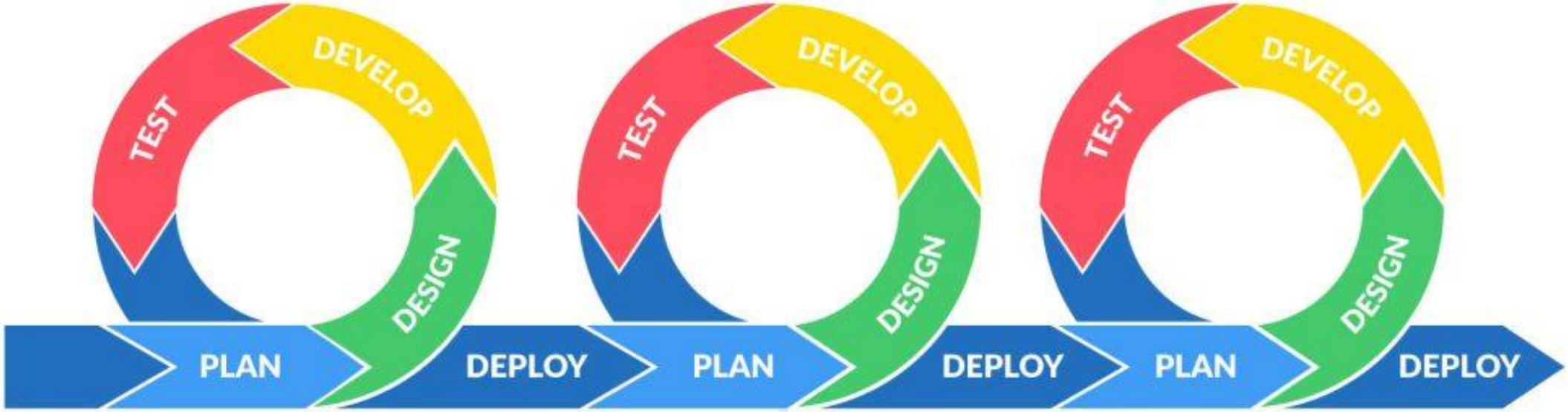
Drawbacks on Waterfall Model:

- Not suitable for **complex projects**.
- No acceptance of changes **during development**.
- It **tough to revert** back.
- Testing done at a later stage, making it hard to **identify the challenges and risks** earlier.

Agile Methodology

- The Agile methodology is a way to manage a project by breaking it up into **several phases.**
- This methodology is one of the **simplest** and **effective** processes to turn a vision for a business need into software solutions.
- Agile is a term used to describe software development approaches that employ continual planning, learning, improvement, team collaboration, evolutionary development, and early delivery.

Agile Methodology



DevOps

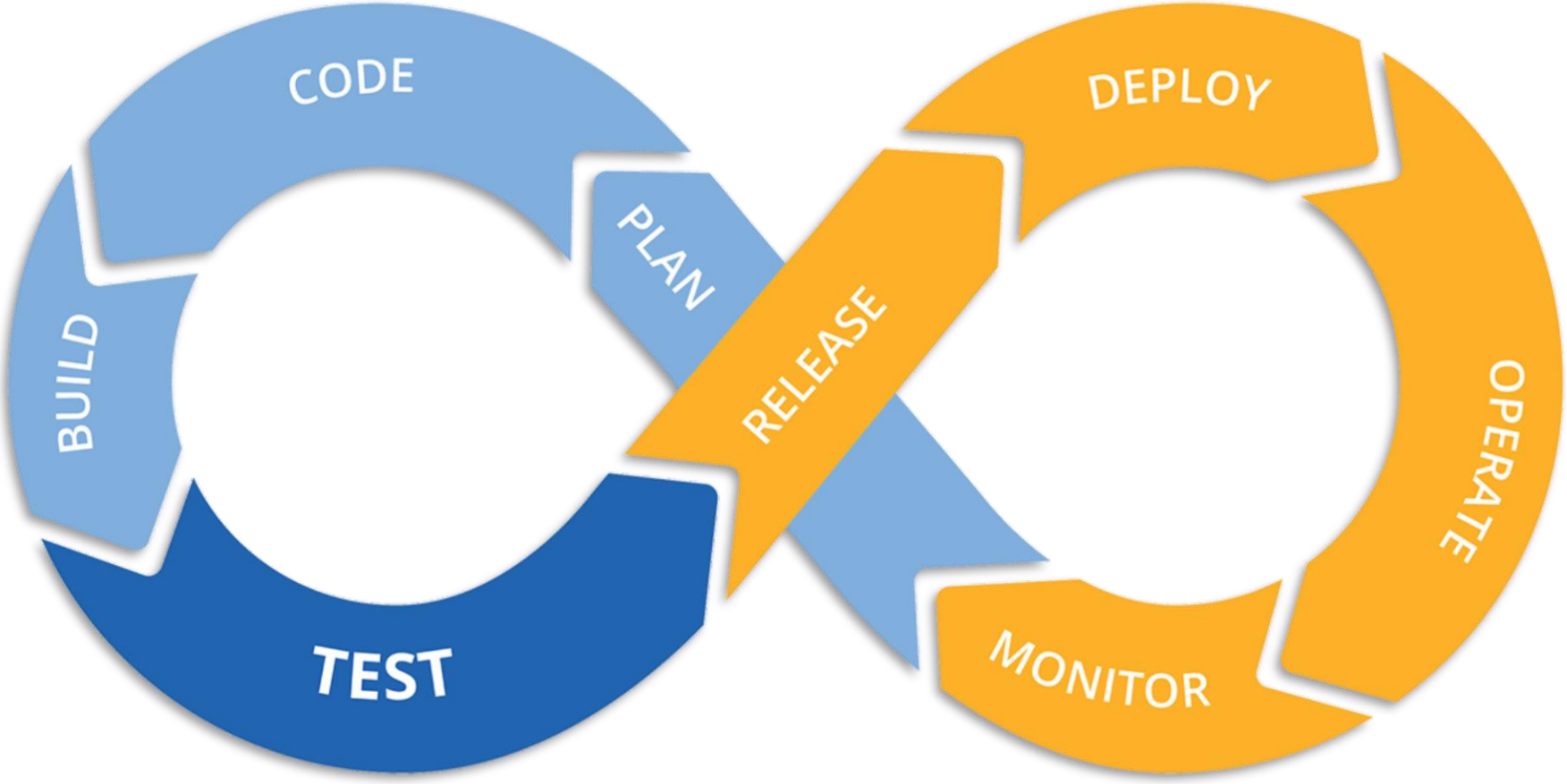
What is DevOps?

- DevOps is a combination of two words **Development** and **Operations**
- The **Development team** is responsible for developing, designing, and building the application.
- The **Operation team** deals with the deployment and testing of the application.
- DevOps is a software development strategy which bridges the gap between **Dev-side** & **Ops-side.**

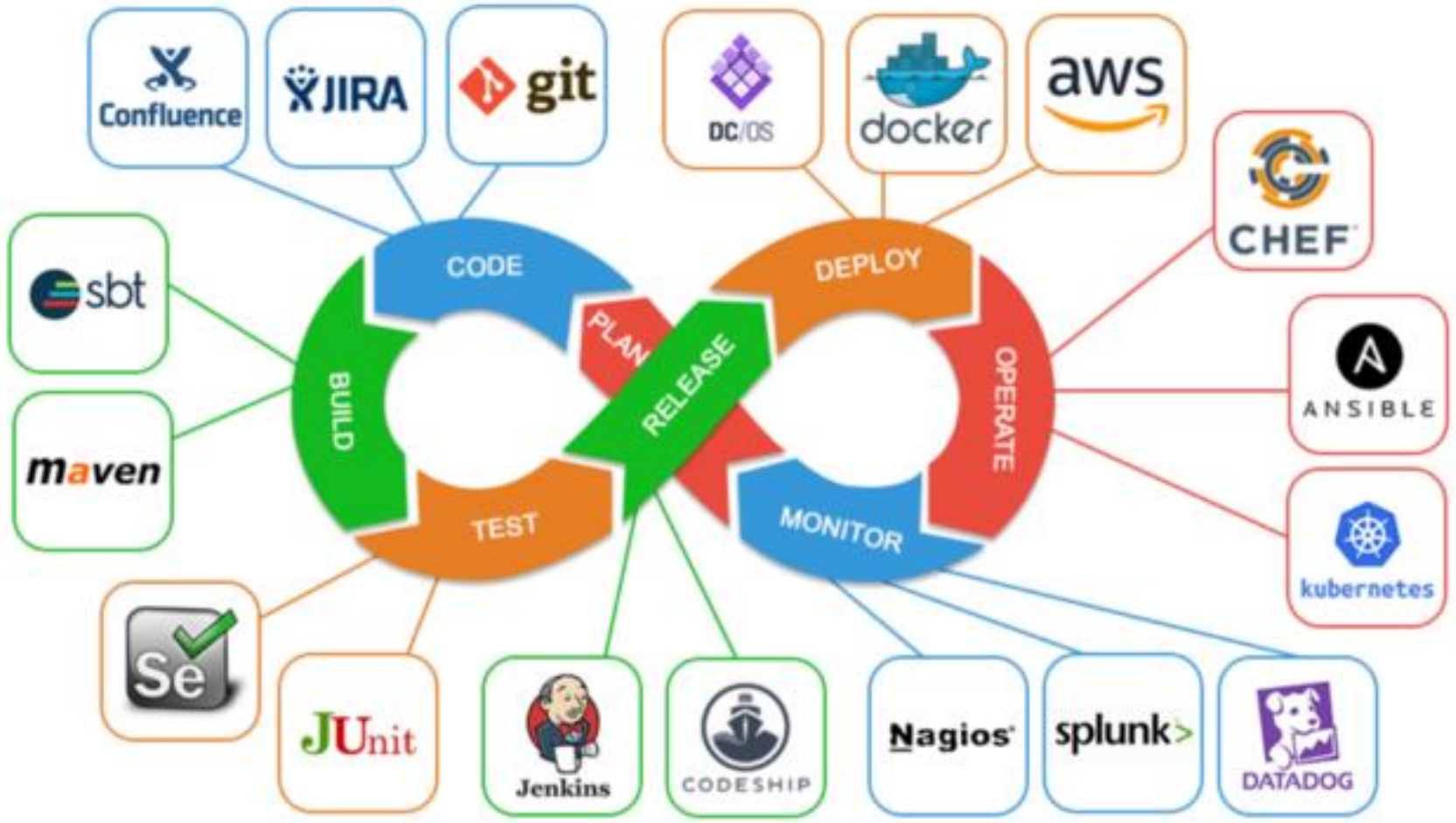
What is DevOps?

- DevOps is not a technology.
- In DevOps, teams participate together on entire software life cycle from design through the development to the production support.
- Both teams (Dev & Ops) can work on rapidly changing systems, fix bugs, and help to deliver a good quality of software in time.

How DevOps work?



Tools under DevOps



DevOps Tools

Areas	Tools
Planning & Coding	Git, Jira
Building	Apache ants, Maven, Gradle
Integration	Selenium, Junit
Testing	Bamboo, Jenkins
Deployment & Operations	Docker, Puppet, Chef, Ansible, Saltstack
Monitoring	Stack, Splunk, Nagios

CI/CD

- Jenkins
- Travis CI
- CircleCI
- GitLab CI/CD
- Azure DevOps (formerly known as Visual Studio Team Services)
- TeamCity
- Bamboo
- GitHub Actions

Version Control:

- Git
- GitHub
- GitLab
- Bitbucket
- Subversion (SVN)

Containerization and Orchestration:

- Docker
- Kubernetes
- OpenShift
- Amazon ECS
- Google Kubernetes Engine (GKE)
- Azure Kubernetes Service (AKS)

Configuration Management:

- Ansible
- Puppet
- Chef
- SaltStack
- Terraform (for infrastructure as code)

Infrastructure as Code (IaC):

- Terraform
- AWS CloudFormation
- Azure Resource Manager (ARM) Templates
- Google Cloud Deployment Manager

Monitoring and Logging:

- Prometheus
- Grafana
- ELK Stack (Elasticsearch, Logstash, Kibana)
- Splunk
- New Relic
- Datadog
- Nagios
- Zabbix

Collaboration and Communication:

- Slack
- Microsoft Teams
- Mattermost
- HipChat
- Jira
- Confluence

Artifact Repository:

- Nexus Repository
- JFrog Artifactory
- Docker Hub
- AWS Elastic Container Registry (ECR)
- Google Container Registry (GCR)

Continuous Testing:

- Selenium
- JUnit
- TestNG
- Postman
- JMeter
- Appium
- Cucumber

Security:

- SonarQube
- OWASP ZAP (Zed Attack Proxy)
- Nessus
- Qualys
- Vault (for secret management)

Deployment and Release Orchestration:

- Spinnaker
- Octopus Deploy
- DeployHub

Continuous Monitoring and Performance Optimization:

- APM (Application Performance Monitoring) tools like AppDynamics, Dynatrace
- Performance testing tools like Apache JMeter
- Load balancers and traffic management tools

Cloud Providers:

- AWS (Amazon Web Services)
- Azure (Microsoft Azure)
- Google Cloud Platform (GCP)
- IBM Cloud
- Alibaba Cloud

Serverless Computing:

- AWS Lambda
- Azure Functions
- Google Cloud Functions

Database and Data Management:

- Database migration and schema management tools
- Data backup and recovery tools
- DataOps platforms

Code Quality and Code Review:

- SonarQube
- Crucible
- Review Board

DevOps Stages

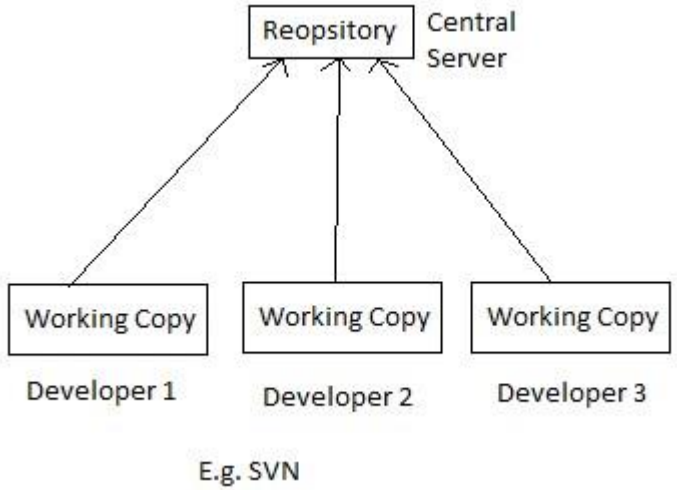
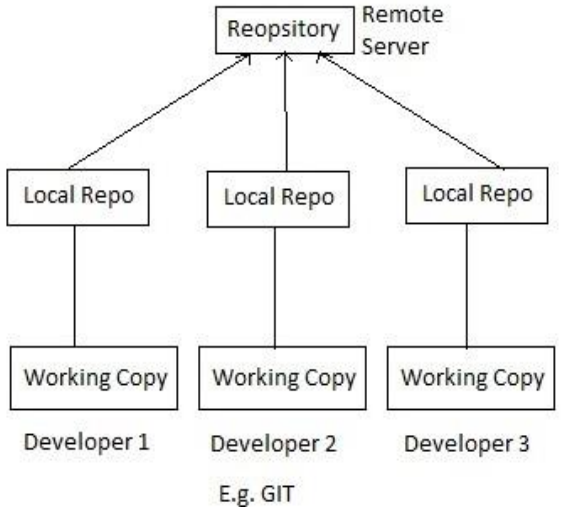
- There are 4 stages under DevOps.

1. Stage 01 – **Version Control**
2. Stage 02 – **Continuous Integration**
3. Stage 03 – **Continuous Delivery**
4. Stage 04 – **Continuous Deployment**

Git

Version Control

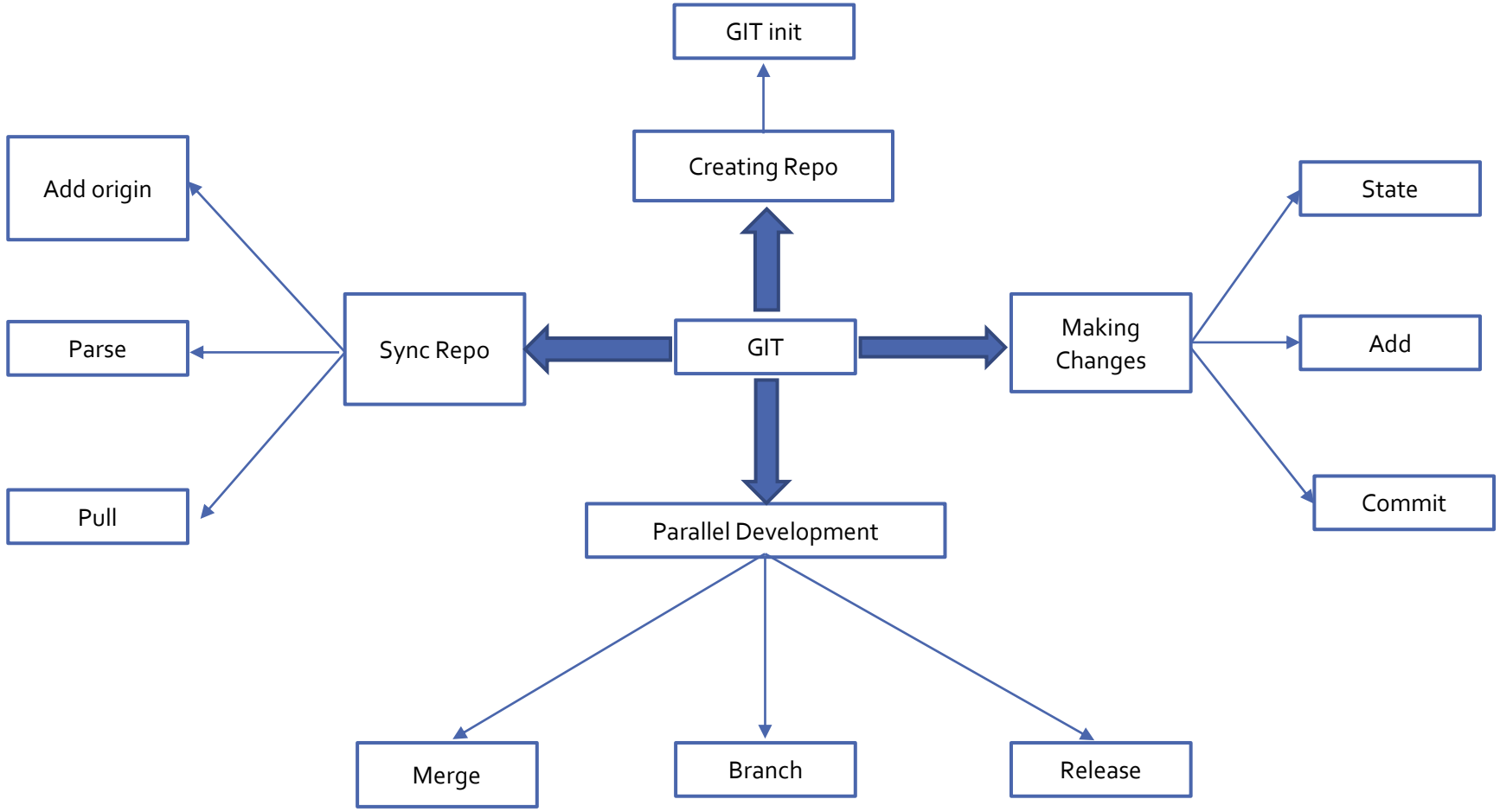
- Its the practice of tracking and managing changes to **software code**.
- Version control has two approaches.

Centralized version control system	Distributed version control system
	

Version Control Tools

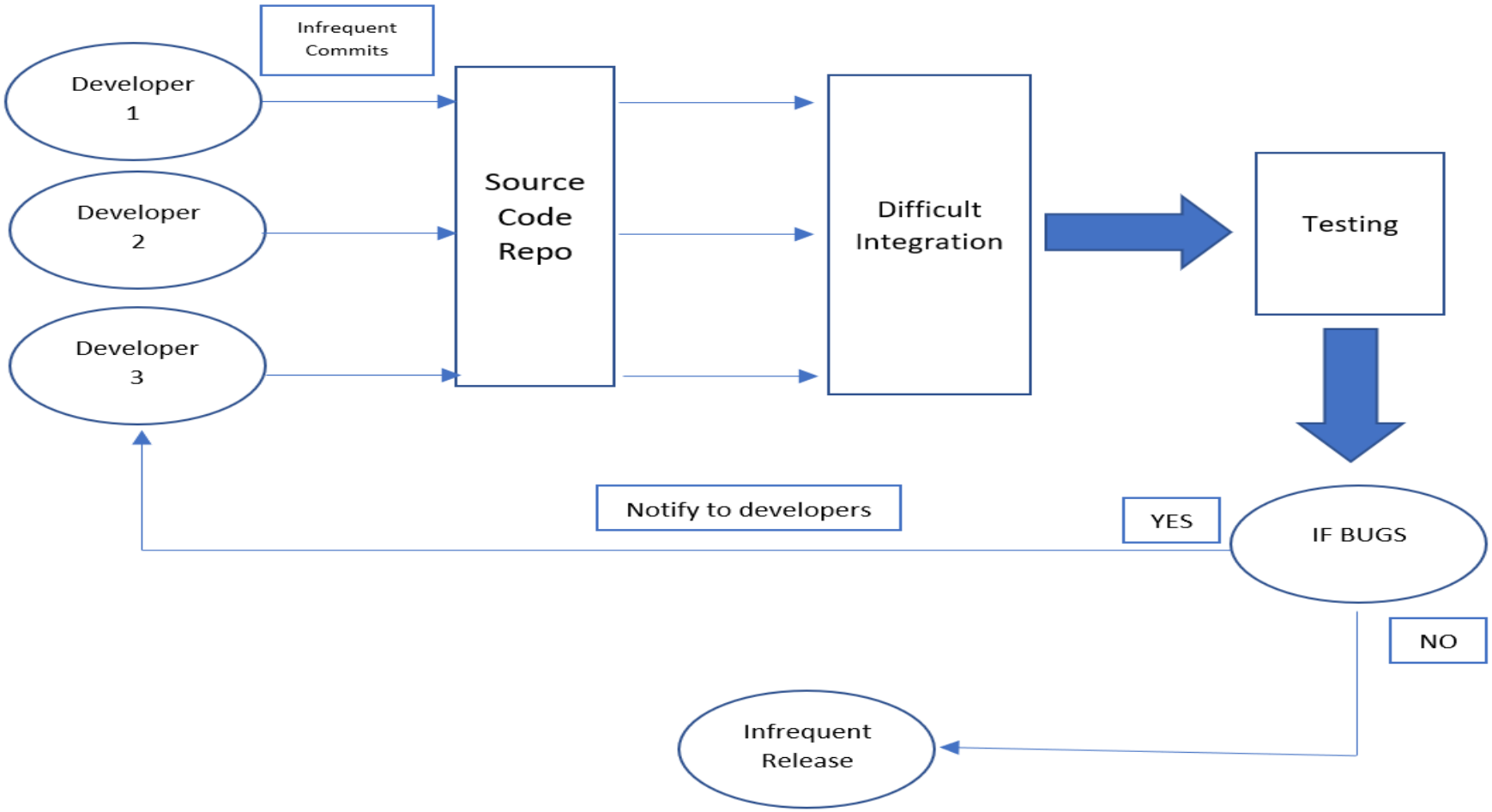
- Git
- Subversion
- CVS
- Mercurial

Git Operations & Basic commands



Continuous Integration (CI)

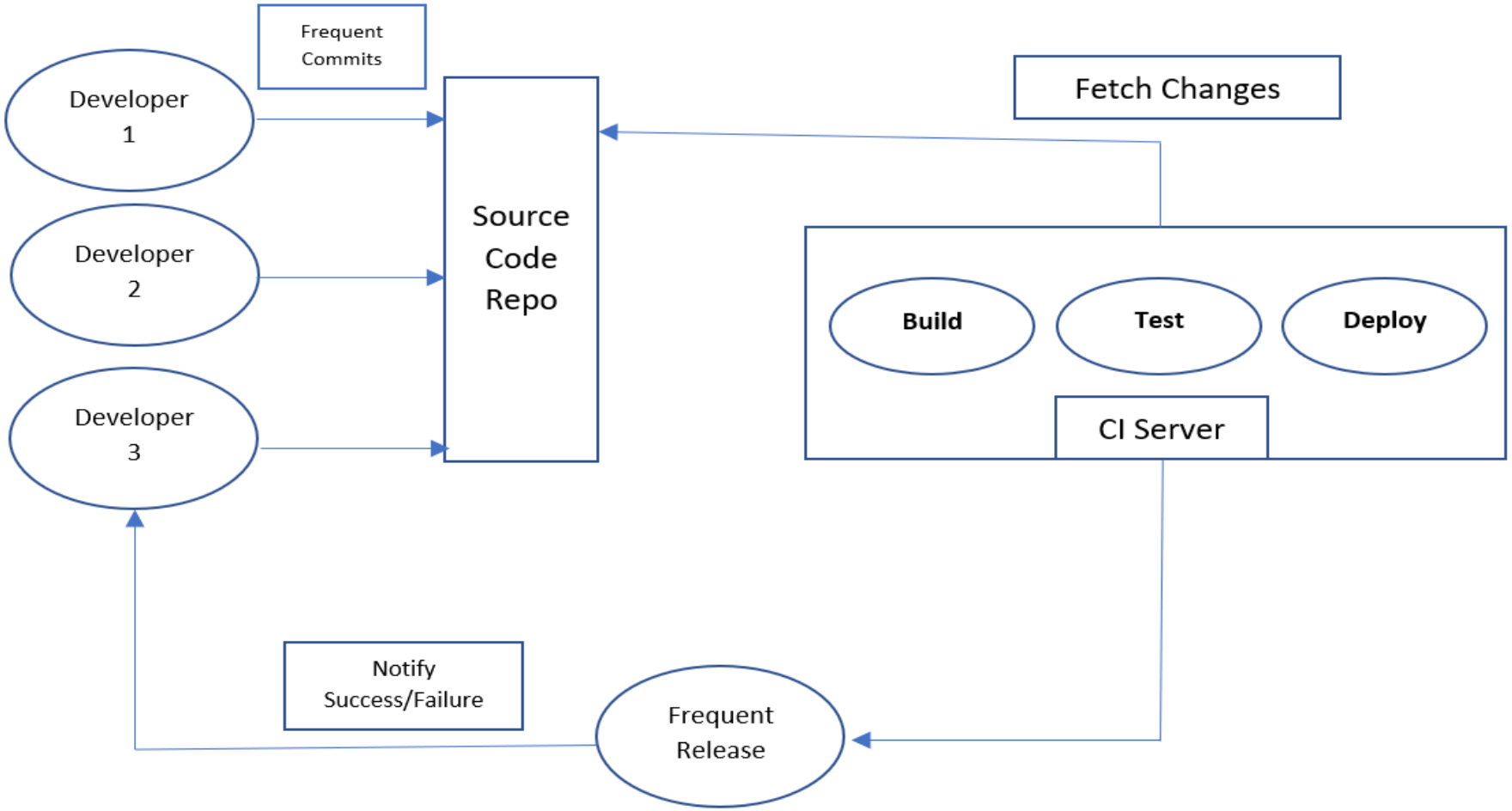
Without Continuous Integration



Continuous Integration

- Continuous integration (CI) is the practice of **automating the integration of code changes** from multiple contributors into a single software project.
- A source code version control system is the **crux** of the CI process.
- In this, developers make changes/commits **frequently**.
- Every commit is then "**build**", which allows to **detect problems/bugs** within the code at early stage.

With Continuous Integration



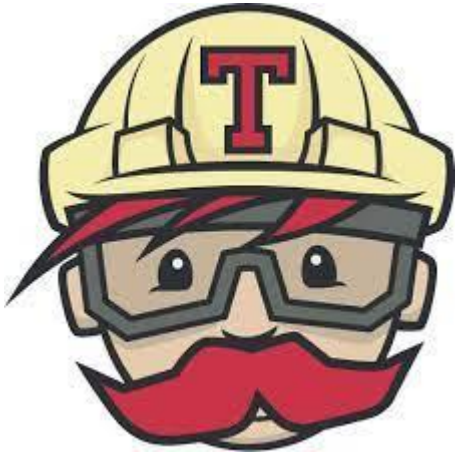
Continuous Integration Tools

- Jenkins
- Buildbot
- Travis CI
- Bamboo

Bamboo

Buildbot

The Continuous Integration Framework



Continuous Delivery (CD)

What is Continuous Delivery?

- Continuous delivery is a software engineering approach in which teams produce software in **short cycles**.
- This ensures that the software can be reliably released at any time.
- It aims at **building, testing**, and **releasing software** with greater speed and frequency.
- The approach helps **reduce the cost** & **time.**

Continuous Delivery

- Continuous delivery (CD) has become a mandatory requirement for organizations.
- A release pipeline can create multiple testing or staging environments to automate infrastructure creation and deploy new builds.
- Successive environments support progressively longer-running integration, load, and user acceptance testing activities.

Continuous Delivery Tools

- Gradle
- Jenkins
- BuildBot
- Buddy
- Ant

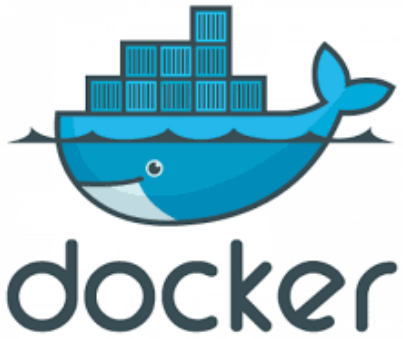
Configuration Management

What is Configuration Management?

- Configuration management (CM) is a process for establishing and maintaining **consistency of a product's performance, functional, and physical attributes** with its requirements, design, and operational information throughout its life.
- Software configuration management is a systems engineering process that **tracks** and **monitors changes** to a software systems configuration metadata.
- In software development, configuration management is commonly used alongside **version control** and **CI/CD infrastructure**.

Configuration management tools

- Terraform
- Git
- Chef
- Ansible
- Saltstack
- Puppet
- Docker



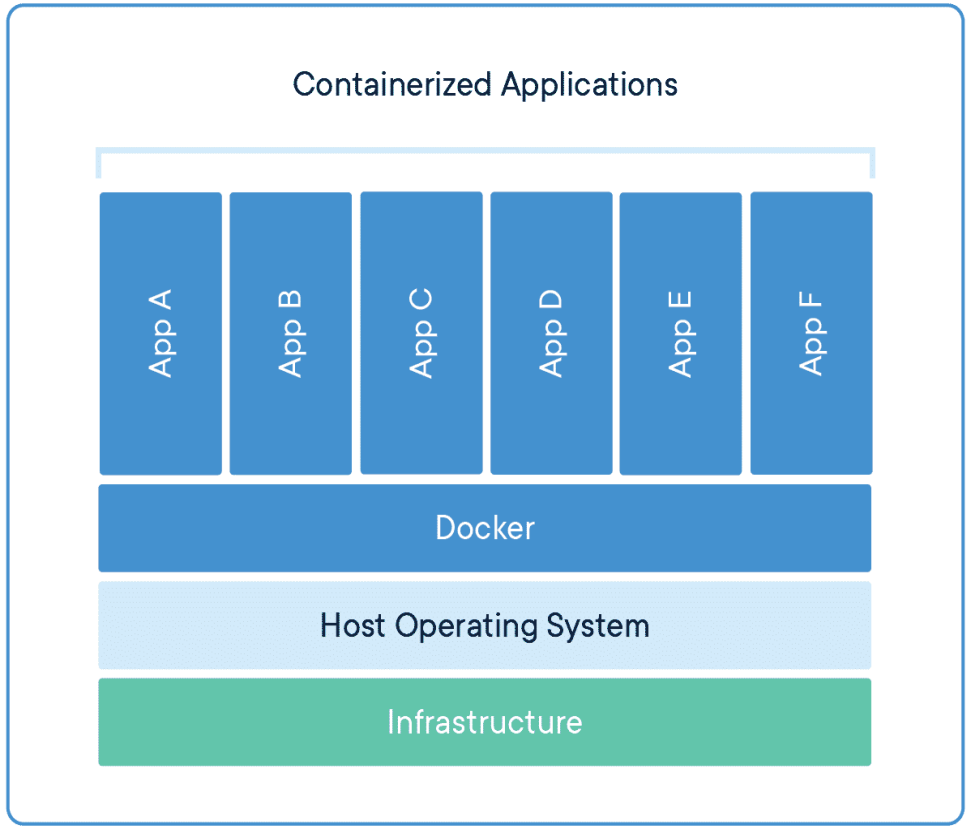
Outcomes of Properly Managed Configurations

- Automation of the infrastructure environment provides standardization
- Setups are free of human error
- Collaboration is enhanced between operations and development
- Keeps configurations from drifting
- Makes infrastructure more flexible, ready to scale
- Each step is consistent across all resources
- Version control is a given

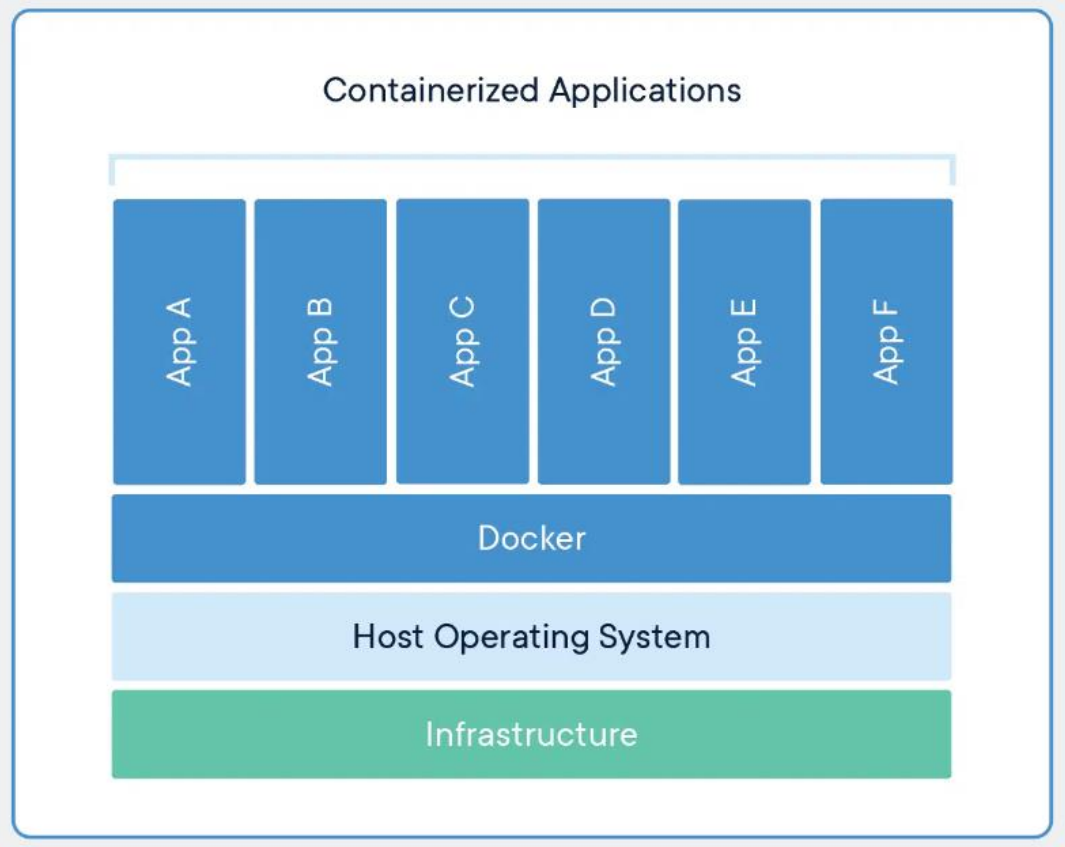
Containerization

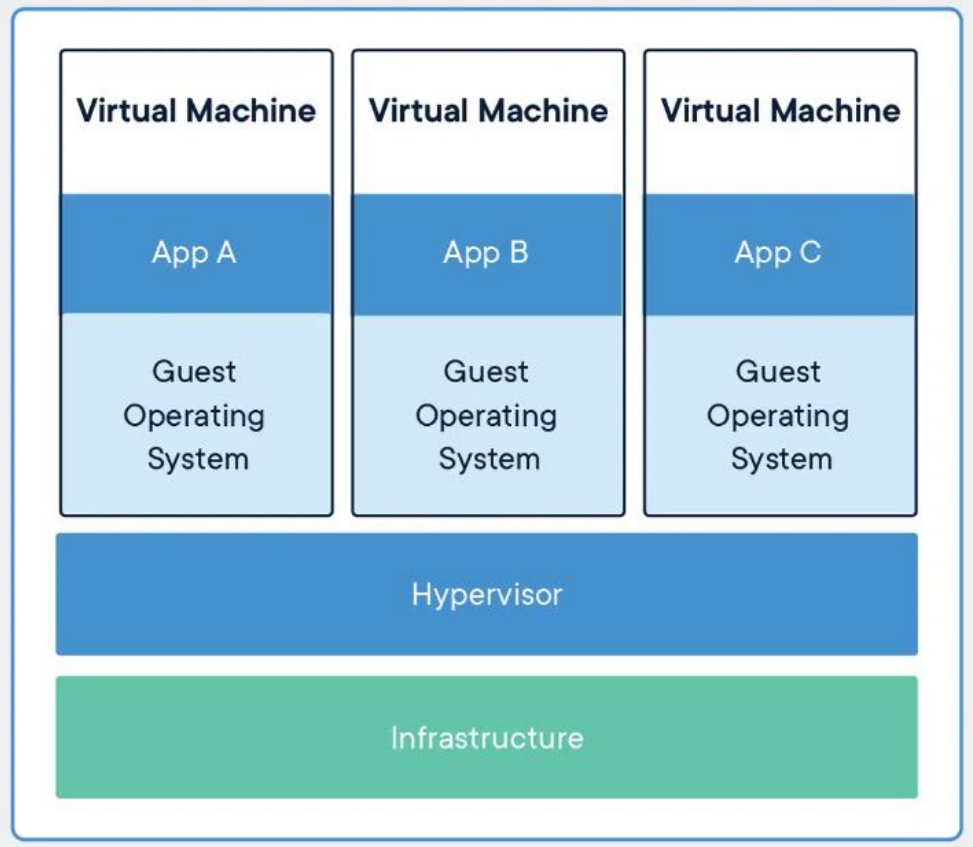
What is Containerization?

- Containerization is a form of **virtualization.**
- Containers are **lightweight, portable**, and highly conducive to automation.
- Containers solve application conflicts between different environments.



Containers vs Virtual Machines





Benefits of Containers

- Greater consistency
- Cost savings
- Security
- Agility

Continuous Monitoring (CM)

What is Continuous Monitoring?

- Continuous monitoring is the process of **identifying threats** to the security and compliance rules of a software development cycle and architecture.
- It is an automated procedure that can be extended to detect similar inconsistencies in IT infrastructures.
- Continuous monitoring (CM) is a step towards the end of the DevOps process.

Types of Continuous Monitoring

- **Infrastructure Monitoring**
  - IT infrastructures typically include components like storage, software and hardware units, data centers, servers, networks, and so on.
- **Application Monitoring**
  - provides details about an application, everything from application uptime, security to performance and log-time.
- **Network Monitoring**
  - This tool facilitates the evaluation of switches, servers, virtual machines, firewalls, and routers.

Tools for Continuous Monitoring

Infrastructure Monitoring

Nagios®

Prometheus

Application Monitoring

splunk>

DATADOG

Network Monitoring

WIRESHARK

NMAP

Docker

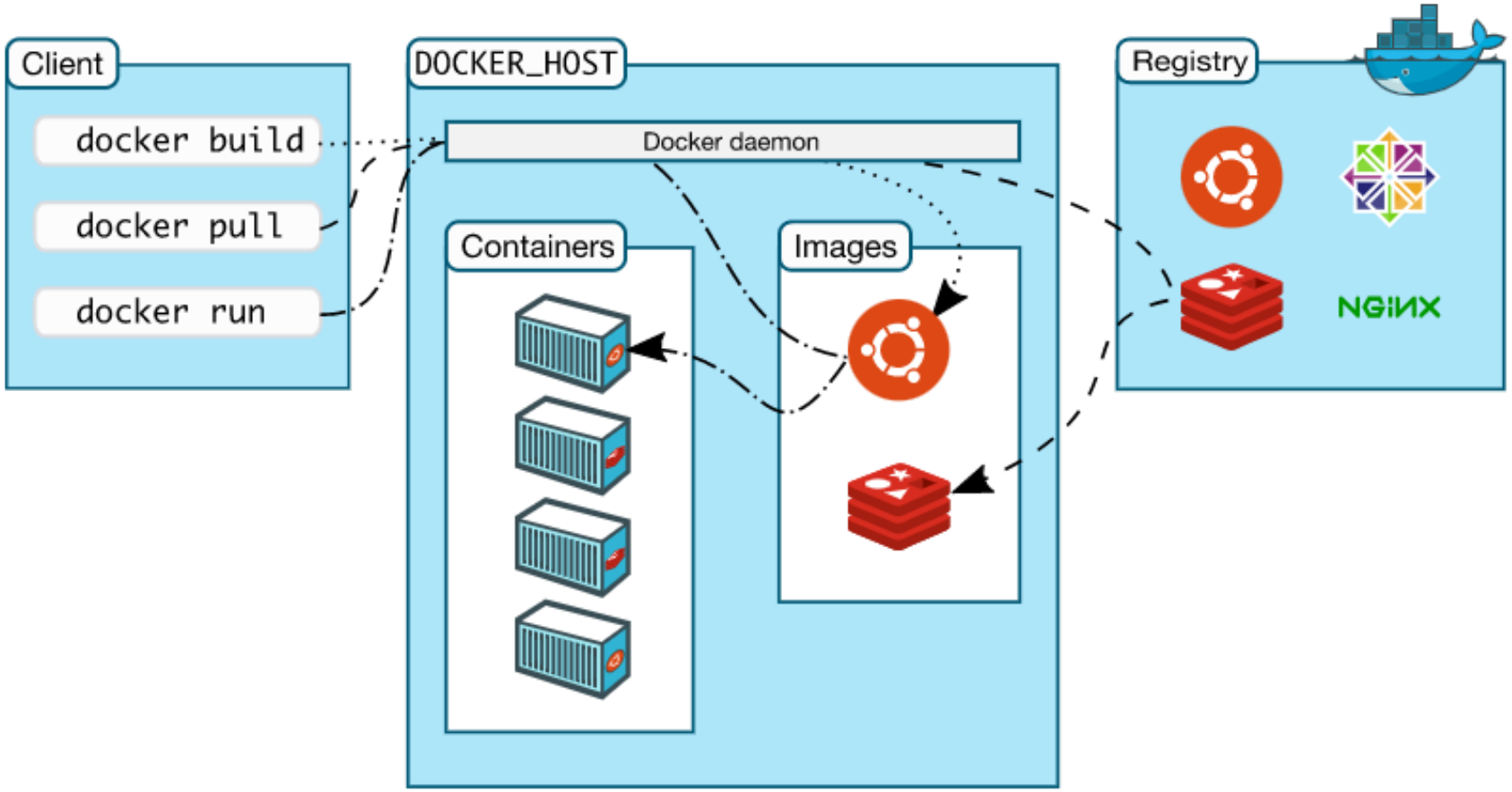
What is Docker?

- Docker is a set of **platforms as a service** (PaaS) product that use the Operating system level visualization to deliver software in packages called containers.
- **Containers are isolated** from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels.
- Docker is an open platform for **developing, shipping, and running applications**.

Benefits of using Docker

- Fast, consistent delivery of your applications
- Responsive deployment and scaling
- Running more workloads on the same hardware

Docker architecture

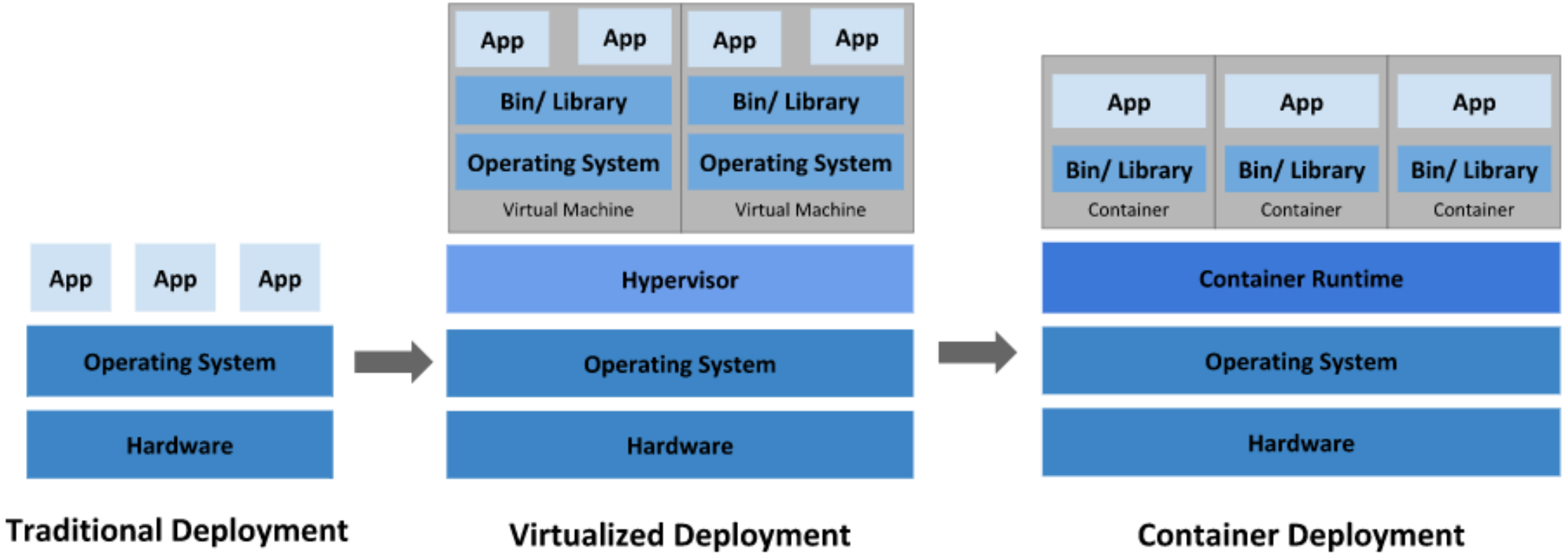


Kubernetes

What is Kubernetes?

- Kubernetes is an open-source container orchestration system for **automating software deployment, scaling, and management.**
- Kubernetes automates operational tasks of container management
  - Deploying applications
  - Rolling out changes to your applications
  - Scaling your applications up and down.
  - Monitoring your applications.

Why Kubernetes?



Benefits of Kubernetes

- Automated day-to-day operations
- Kubernetes handles complete compute, networking & storage workloads.
- Monitors every time, every container.

IaC

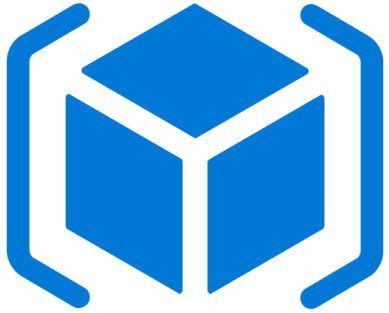
What is IaC?

- Infrastructure as Code (IaC) is an **approach to managing** data center server, storage, and networking infrastructure.
- IaC is meant to significantly **simplify large-scale configuration and management**.
- With IaC, infrastructure configuration **information is housed in standardized files**, which can be read by software that maintains the state of the infrastructure.
- IaC can **improve productivity and reliability** by eliminating manual configuration steps.

IaC vs. Automation

- Infrastructure as code is concerned with **maintaining the configuration** or state of the data center infrastructure in a known way.
- Automation deals more with the process for **automatically pushing** that state into the infrastructure and maintaining it.

Tools for Infrastructure as Code



Tools for Infrastructure as Code

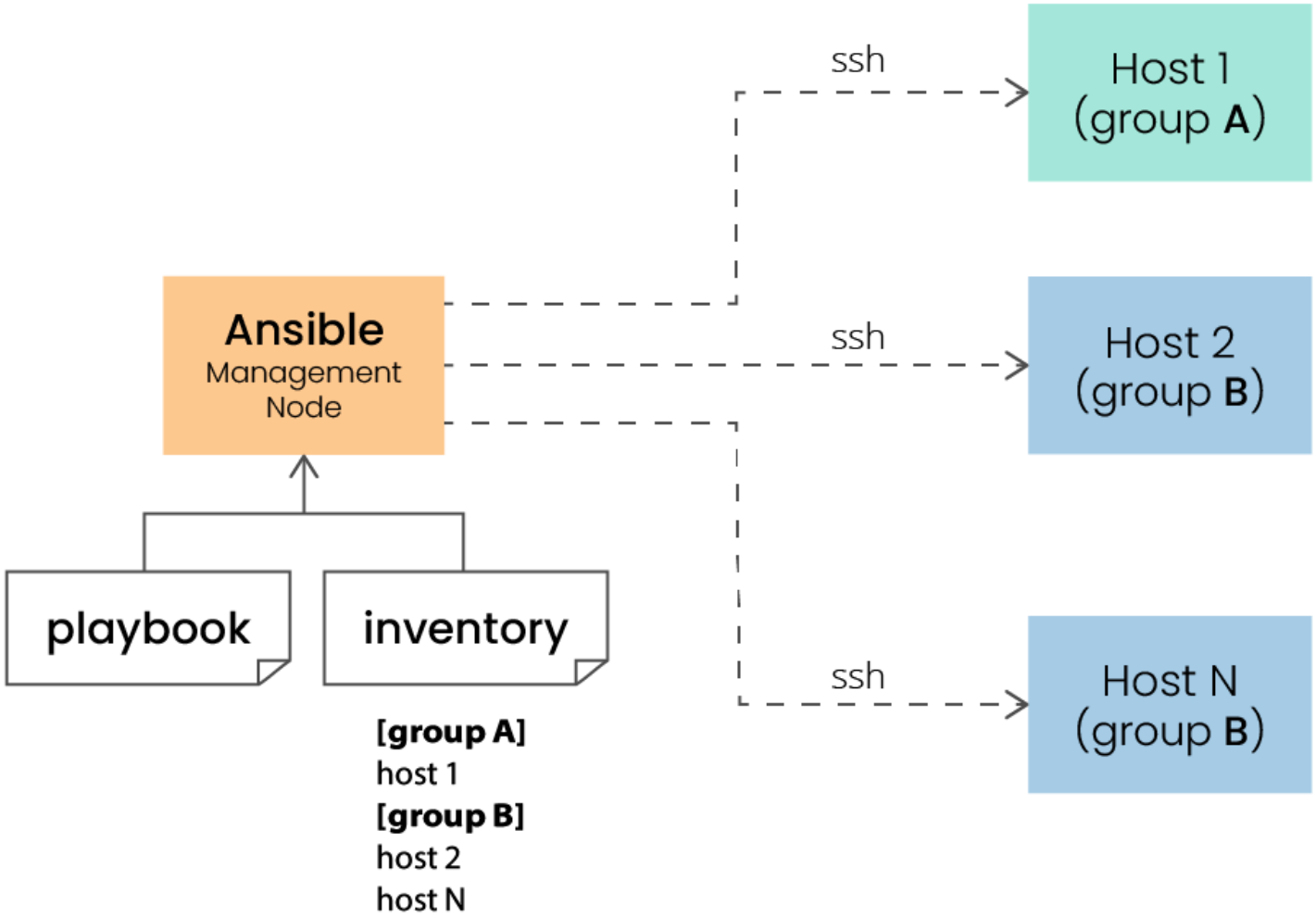
- Terraform
- AWS CloudFormation
- Azure Resource Manager (Templates)
- Google Cloud Deployment Manager
- Ansible
- Chef
- Puppet
- Vagrant

Ansible

What is Ansible?

- Ansible is a radically **simple IT automation** system.
- It handles **configuration management, application deployment**, **cloud provisioning, ad-hoc task execution, network automation,** and multi-node orchestration.
- Ansible makes **complex changes like zero-downtime rolling updates** with load balancers easy.

How ansible works?



Ansible playbooks

- A playbook is a configuration file written in **YAML** that provides instructions for what needs to be done in order to bring a managed node into the desired state.
- Playbooks are meant to be **simple, human-readable, and self-documenting**.
- A playbook can be very **simple** or it can be **very complex**.
- Example
  - Create a user with elevated permissions.
  - Patching servers, hosts etc.

Puppet

What is Puppet?

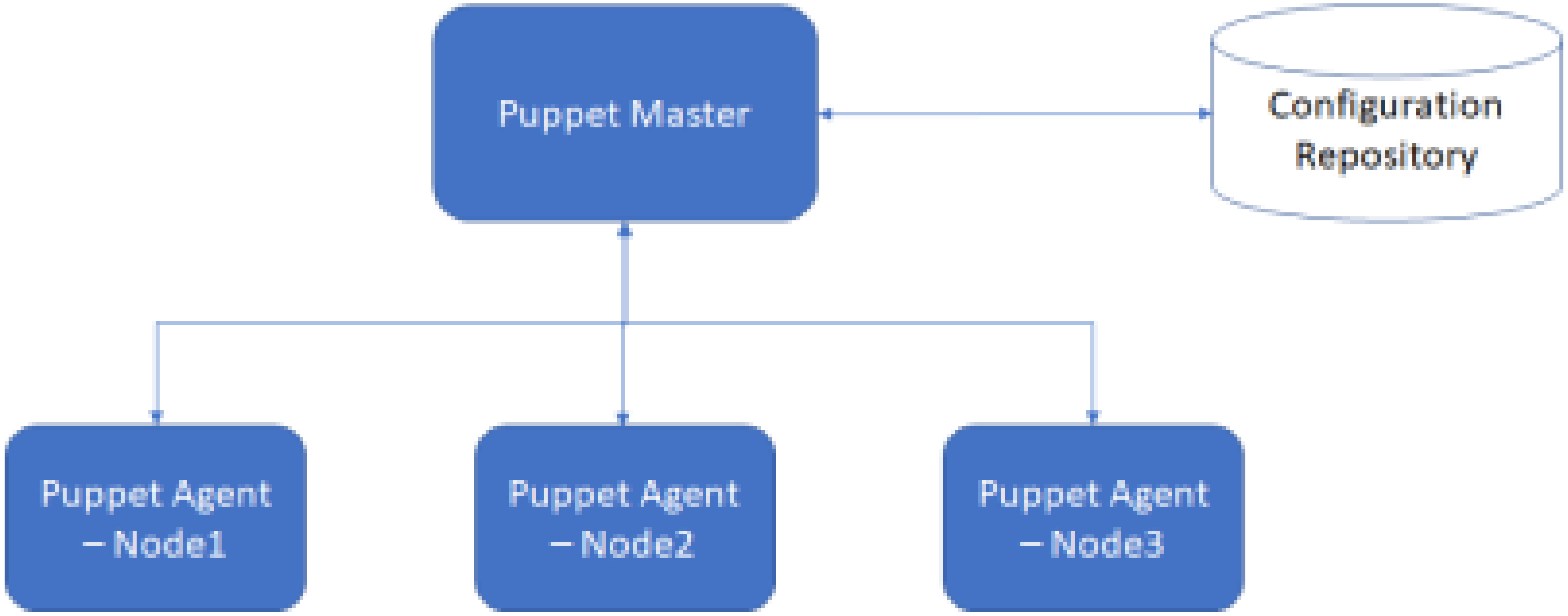
- Puppet is a **configuration management** technology to manage the infrastructure on physical or virtual machines.
- Puppet follows the **client-server model**.
- Puppet has the **capability to manage** any system from scratch, starting from initial configuration till the end-of-life of any particular machine.

Puppet versions

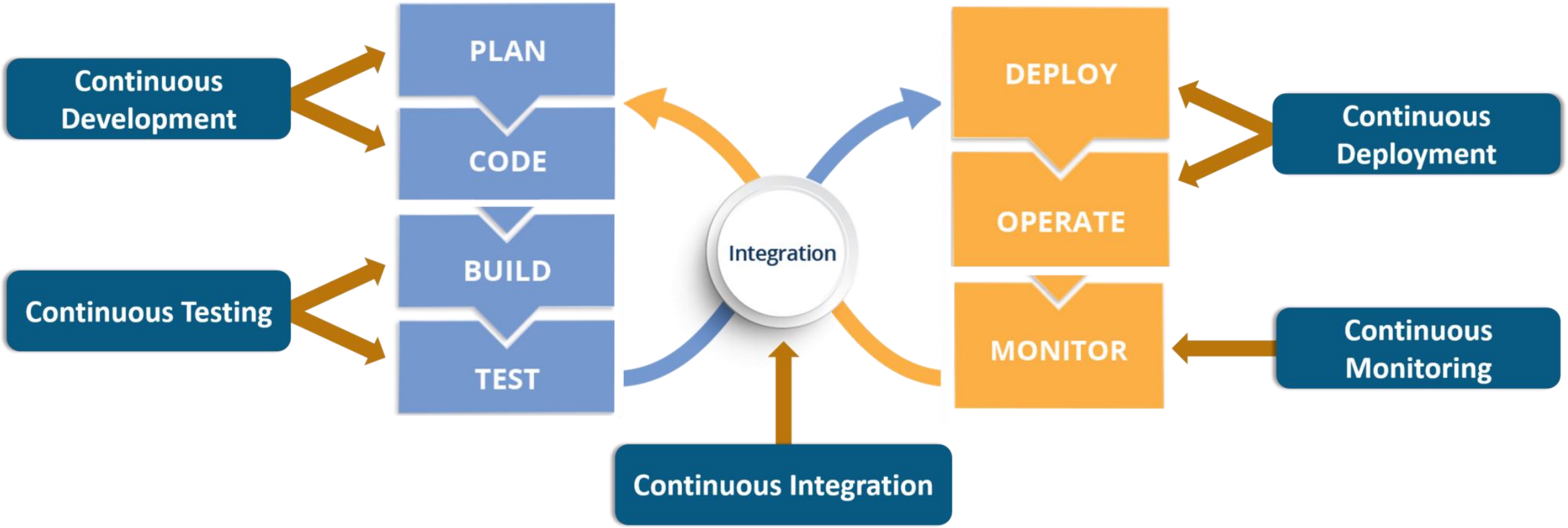
Puppet comes in two versions:

- Open Source Puppet
- Puppet Enterprise

Puppet Architecture



How everything fits together?




That's all Folks!

Any Question?