

(20CSE10) DevOps

(Professional Elective – II)

Instruction per week	3 Hours
Duration of End Examination	3 Hours
Semester End Examination	60 Marks
Continuous Internal Evaluation	40 Marks
Credits	3

Pre-requisites: Database management systems, Operating systems, OOPs.

Course Coordinator: Venkata Sivarao.Alapati

Course Objectives and Outcomes

Course Objectives: The objectives of this course are,

1. To describe the agile relationship between development and IT operations.
2. To understand the skill sets and high-functioning teams involved in DevOps and related methods to reach a continuous delivery capability.
3. To implement automated system update and DevOps lifecycle.

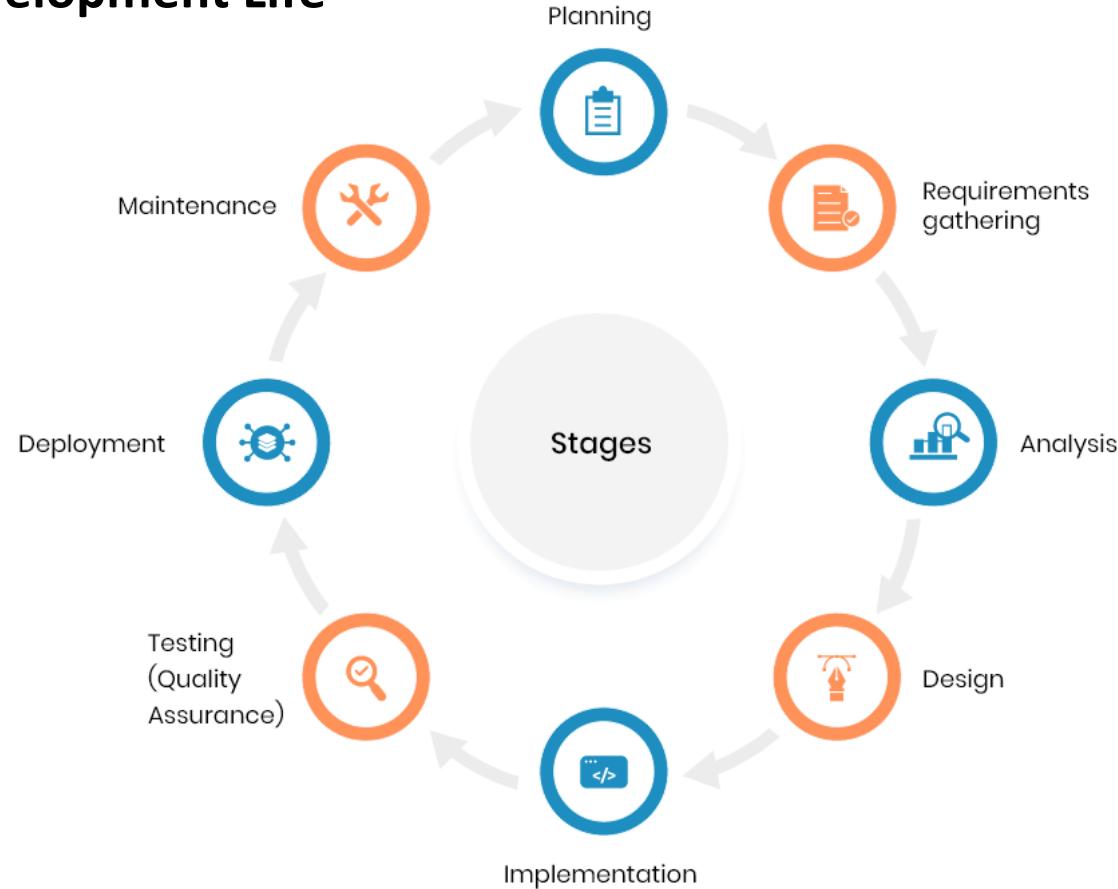
Course Outcomes: On successful completion of this course, students will be able to,

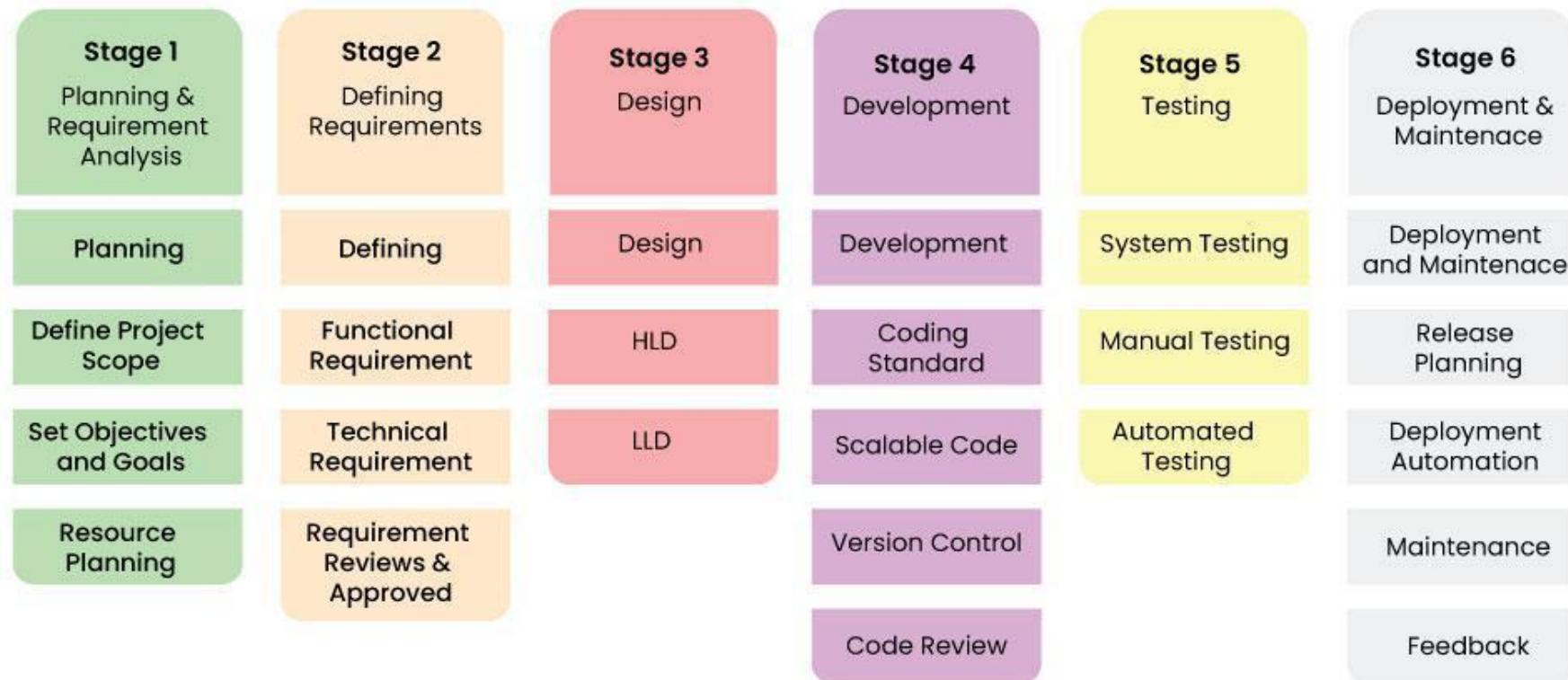
1. Identify components of Devops environment.
2. Describe Software development models and architectures of DevOps.
3. Apply different project management, integration, testing and code deployment tools.
4. Investigate different DevOps Software development models.
5. Assess various Devops practices.
6. Collaborate and adopt Devops in real-time projects.

Unit-1

- Introduction: Software development models,
- Introduction to DevOps, Why DevOps,
- DevOps process and Continuous Delivery,
- Delivery pipeline,
- Release management,
- Scrum, Kanban
- DevOps Architecture,
- DevOps Workflow
- DevOps Lifecycle for Business Agility, and Continuous Testing.

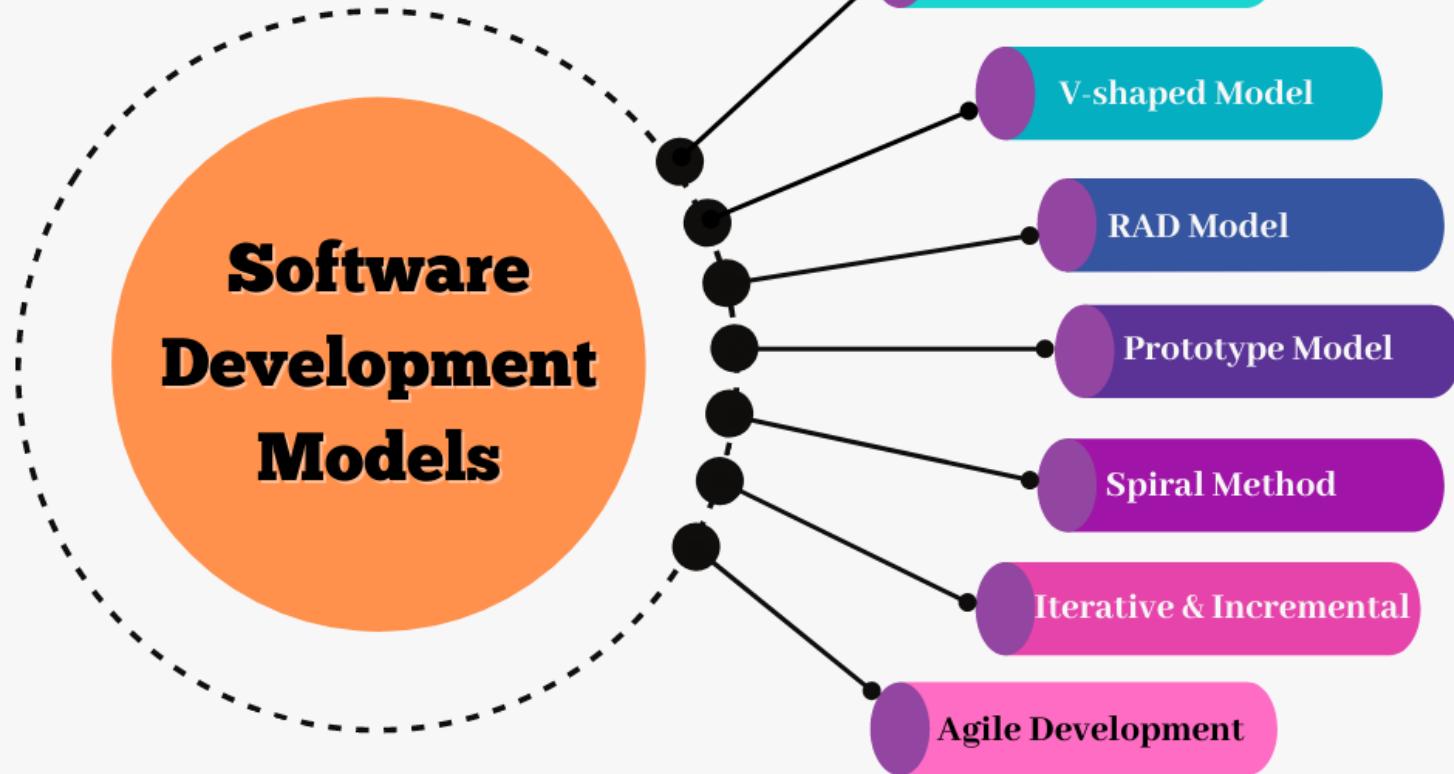
Software Development Life cycle



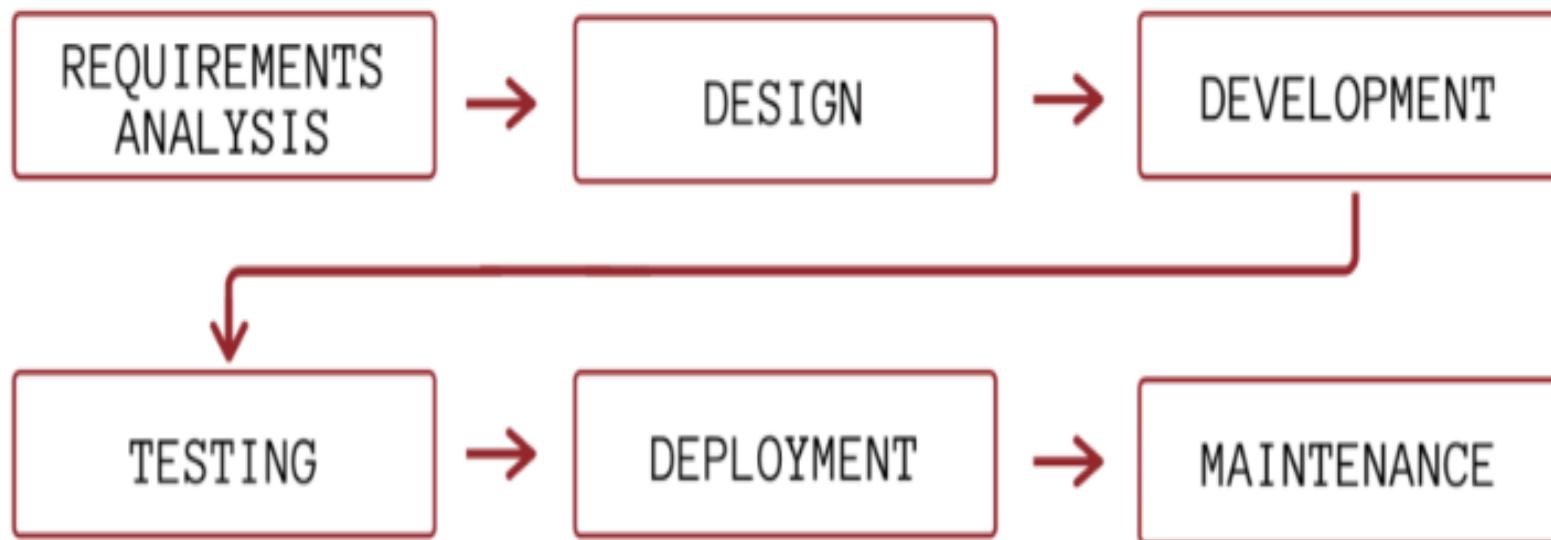


6 Stages of Software Development Life Cycle

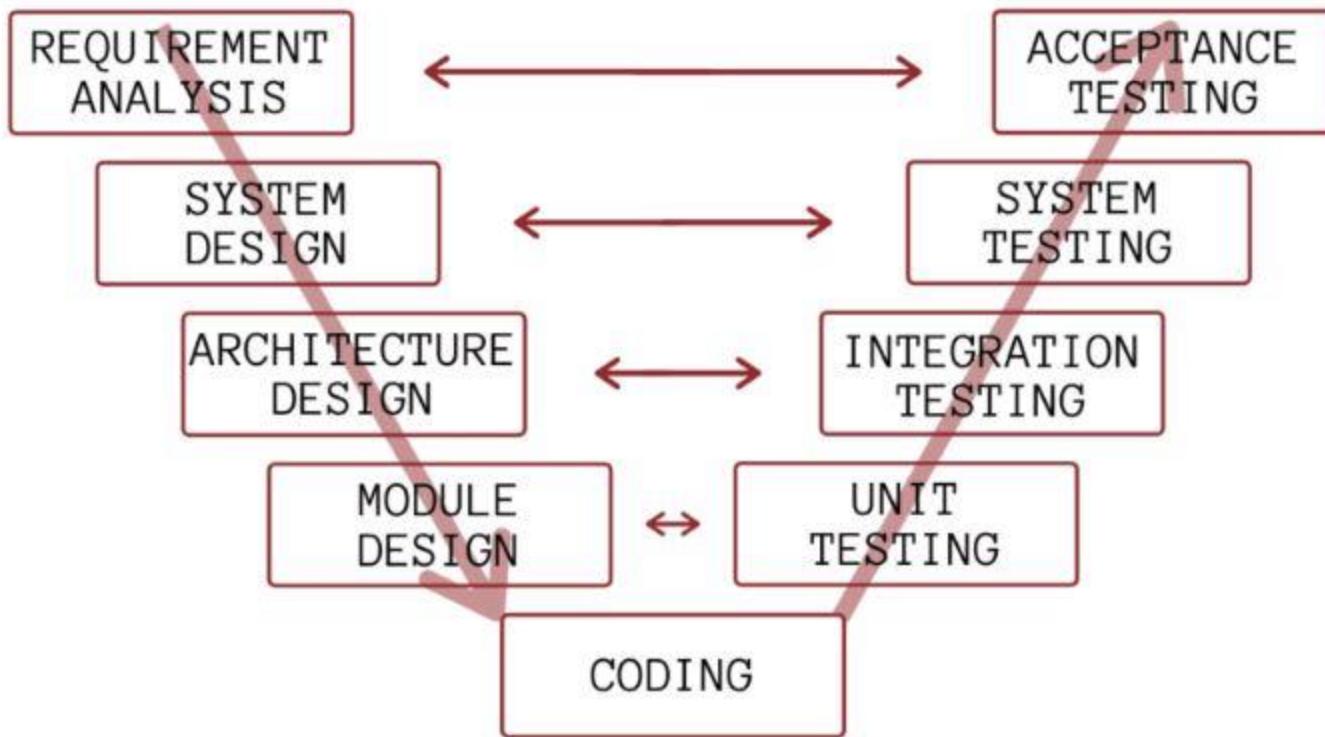




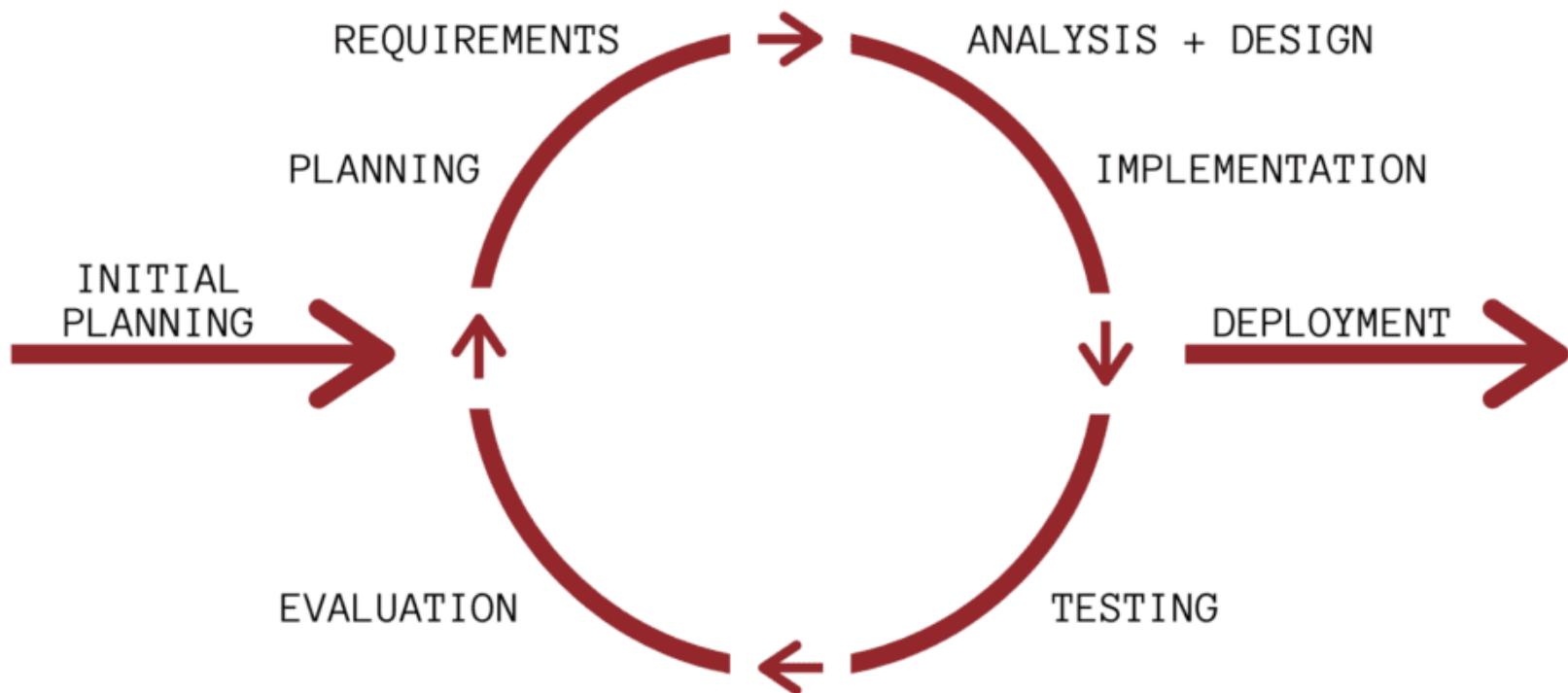
THE WATERFALL MODEL



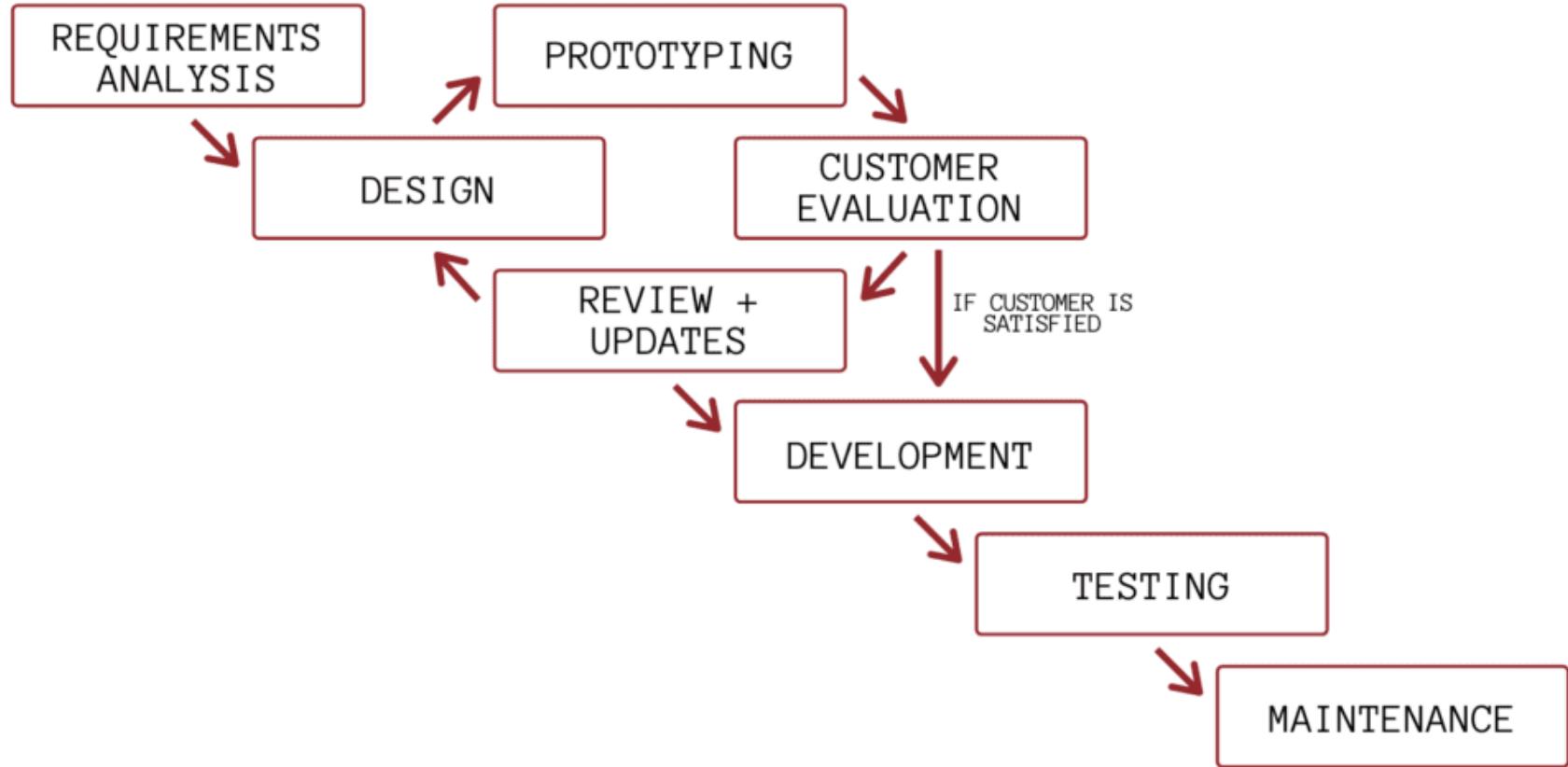
V-MODEL



ITERATIVE MODEL



PROTOTYPING MODEL



SPIRAL MODEL

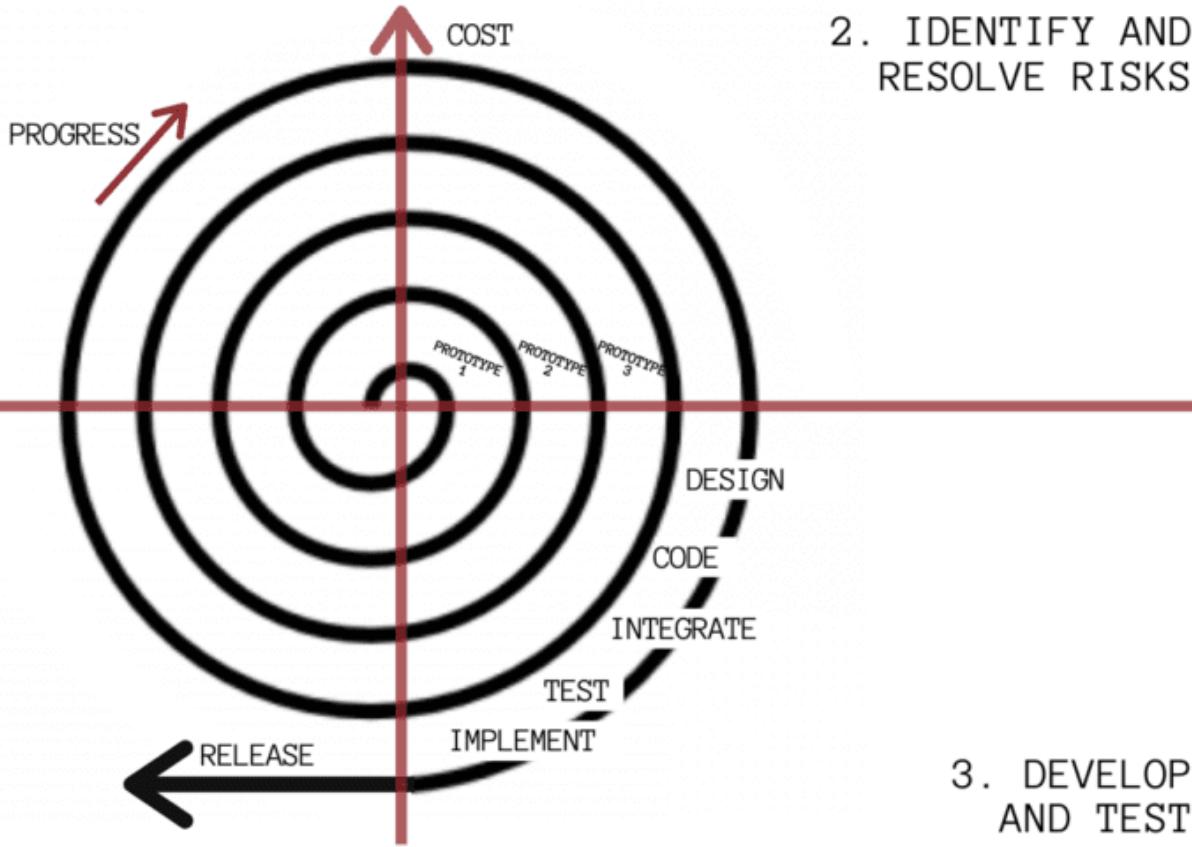
1. DETERMINE OBJECTIVES

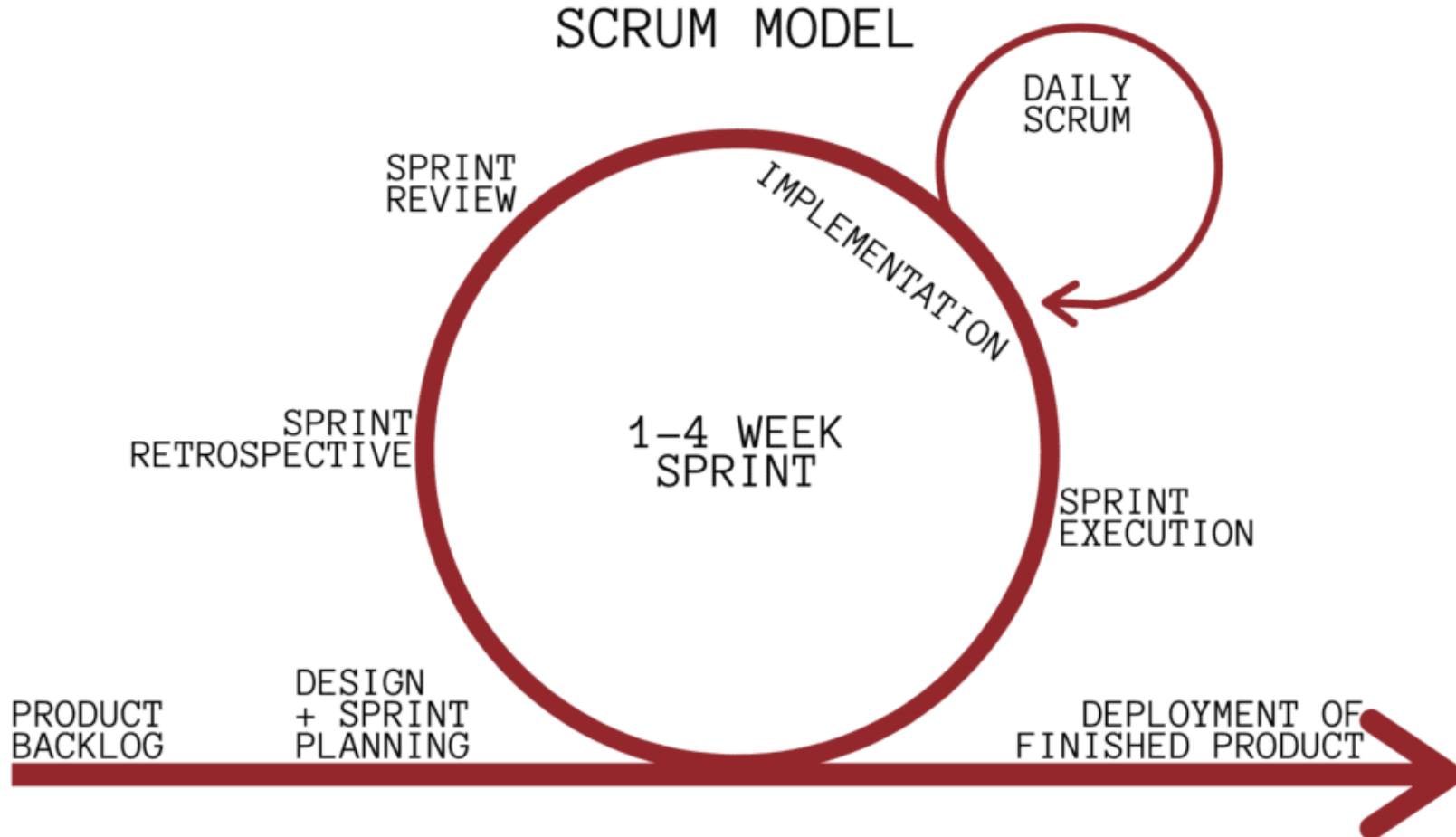
2. IDENTIFY AND RESOLVE RISKS

REVIEW

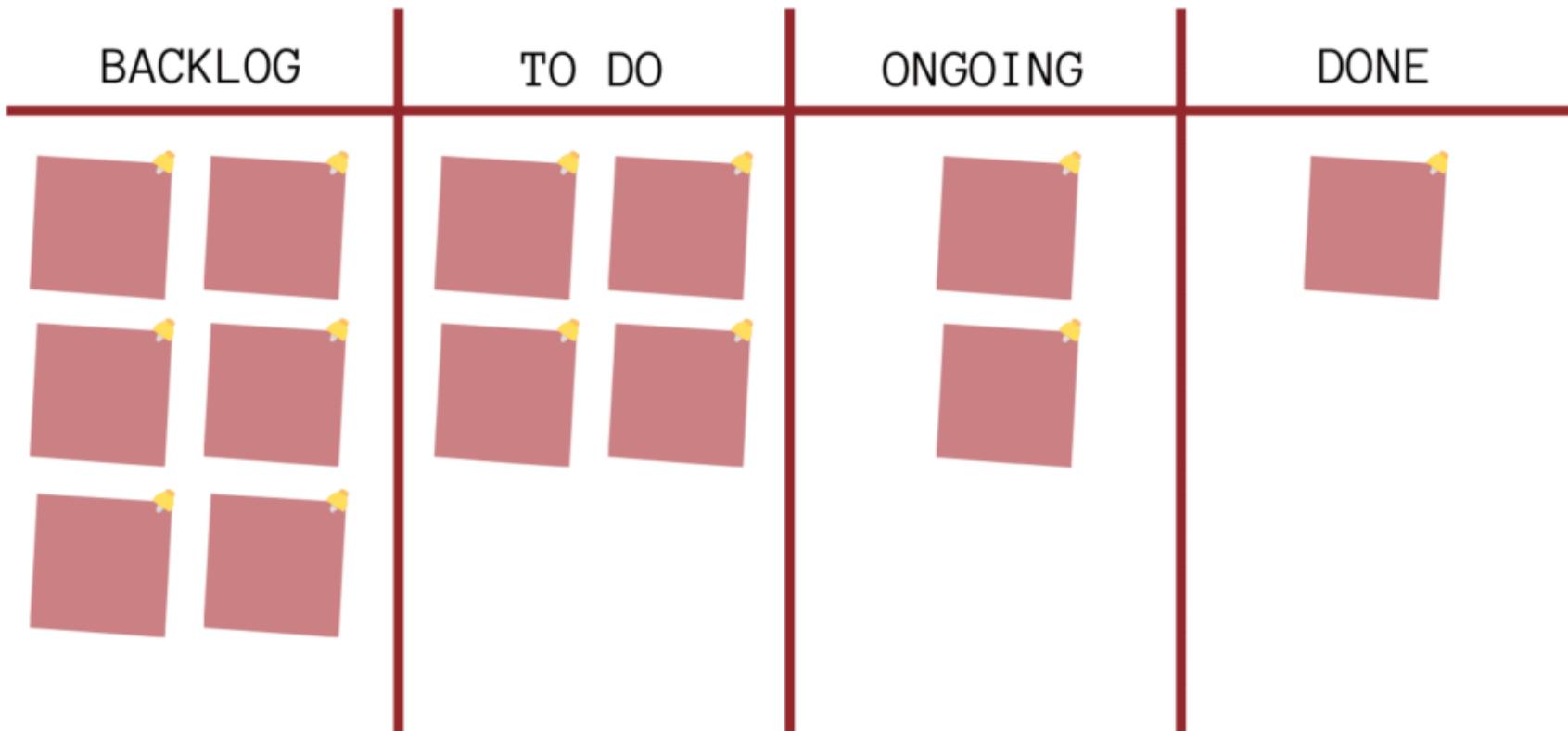
4. PLAN NEXT PHASE

3. DEVELOP AND TEST

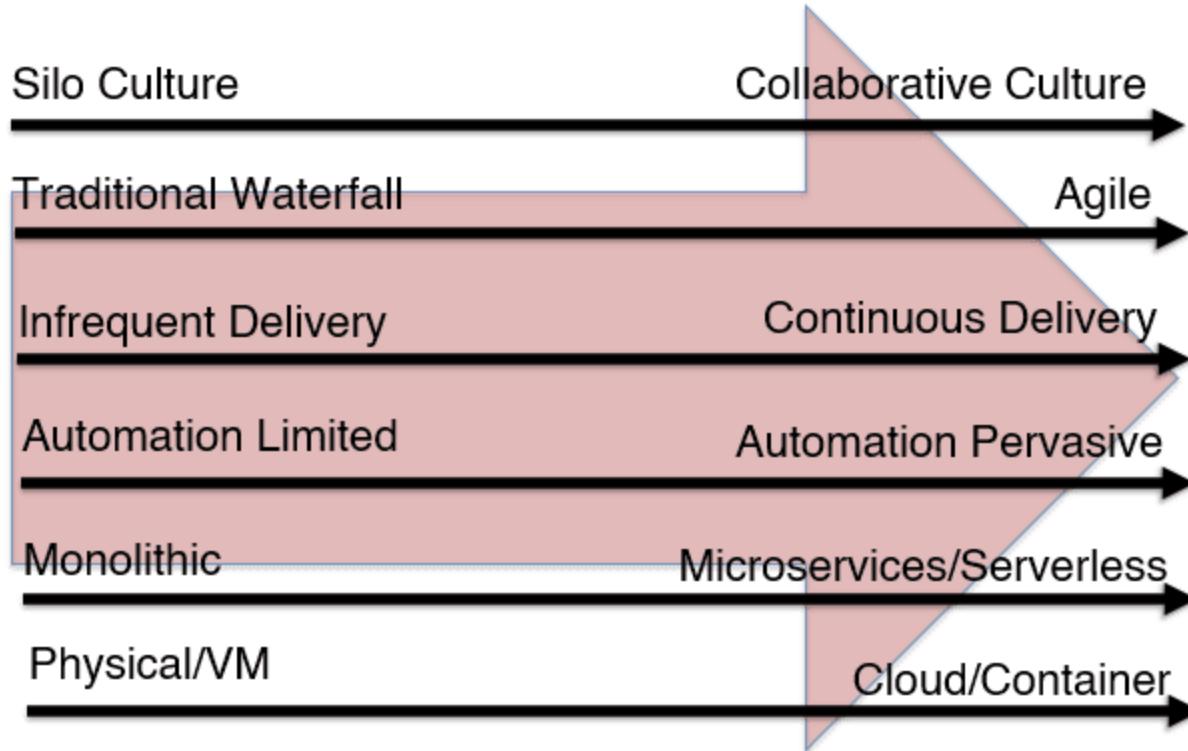




KANBAN BOARD



The Shift to Agile Software Development



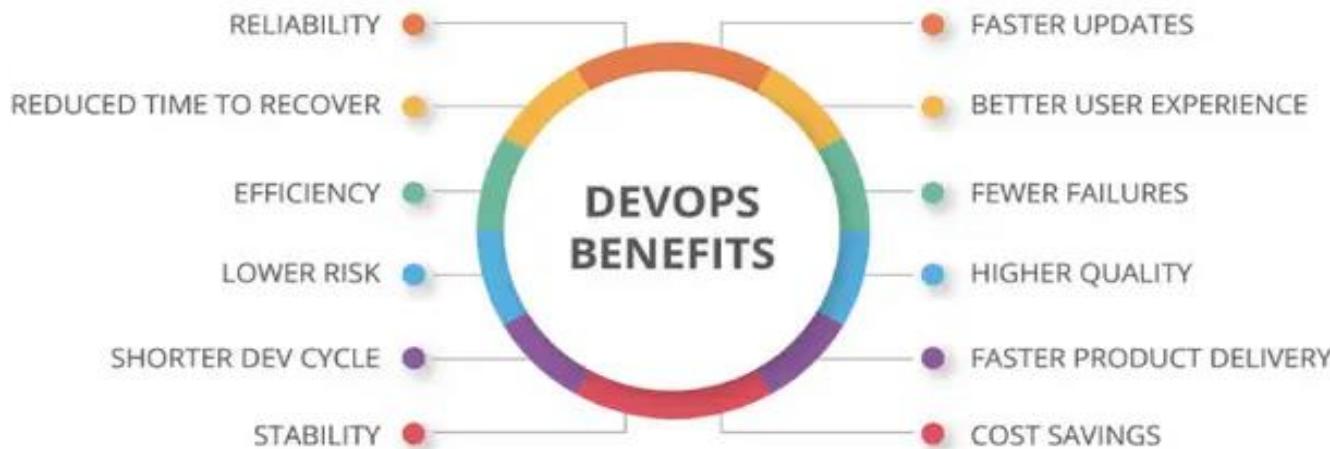
	Cost	Development Speed	Delivery Frequency	A Possible Number of Defects	Project Complexity	The Popularity
Waterfall	High (\$>150.000)	Low (12 months)	Once	Functional defects are possible	Middle complexity	Low
Spiral Method	Moderate (\$<150.000)	Medium (6 months)	Monthly	Minor defects are possible	Middle complexity	Moderate
V-model	High (\$>150.000)	Low (12 months)	Once	Minor defects are possible	Huge and Complex App	Moderate
Iterative and Incremental	Low (\$<10.000) - High (\$>150.000)	Quick (3 months) - Medium (6 months)	Weekly-Monthly	Minor defects are possible	Middle complexity - Huge complex Apps	High
The Rational Unified Process (RUP)	Moderate (\$<150.000)	Medium (6 months)	Once	Minor defects are possible	No complex Logic - Middle Complexity	Moderate
Extreme Programming (XP)	Low (\$<10.000) - Moderate (\$<150.000)	Medium (6 months)	Weekly-Monthly	Minor defects are possible	Middle complexity	Low
Scrum	Low (\$<10.000)	Quick (3 months)	Weekly	Functional defects are possible	Middle Complexity - Huge and complex applications	High
Kanban	Moderate (\$<150.000)	Continuous delivery	Weekly-Monthly	Minor defects are possible	Middle Complexity - Huge and complex applications	High
DevOps	High (\$>150.000)	Continuous delivery	Weekly-Monthly	Minor defects are possible	Huge and complex applications	Low, but it's becoming more and more popular
Shape Up Method (a black horse)	Moderate (\$<150.000)	Quick (3 months)	Weekly	Minor defects are possible	Middle complexity	Low

Dev Team





Why DevOps Is Important



DevOps Process and Continuous delivery

- **Continuous integration:** [Continuous integration](#) is the practice of automating the integration of code changes into a software project. It allows developers to frequently merge code changes into a central repository where builds and tests are executed.
- **Continuous Testing:** [Continuous Testing](#) obtains immediate feedback on the business risk associated with Software Releases. It's a challenging and essential part of the software. Software rating depends upon Testing. The test function helps the developer to balance quality and speed.
- **Continuous Delivery:** Continuous delivery expands upon continuous integration by automatically deploying code changes to a testing/production environment. It follows a [continuous delivery pipeline](#), where automated builds, tests, and deployments are orchestrated as one release workflow.

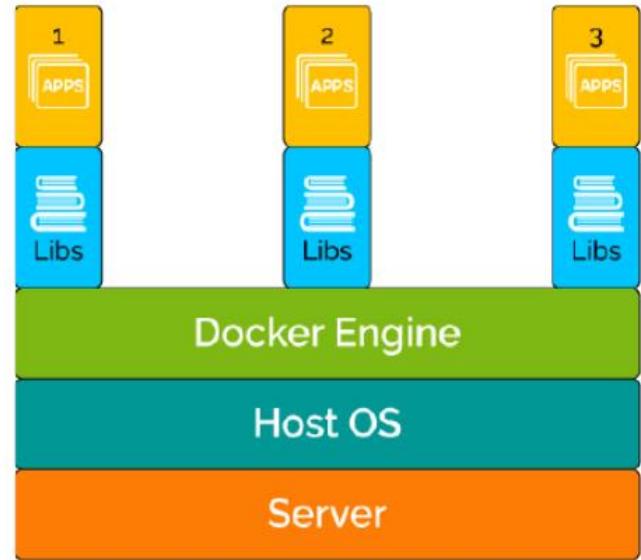
- **Continuous Deployment:** The code is automatically deployed to the production environment as it passes through all the test cases. Continuous versioning ensures that multiple code versions are available in the proper places. Here every changed code is put into production, automatically resulting in many deployments in the production environment every day.
- **Continuous Monitoring :** It is a reporting tool because which developers and testers understand the performance and availability of their application, even before it is deployed to operations? Feedback provided by continuous monitoring is essential for lowering the cost of errors and change. Nagios tool is used for continuous monitoring.
- **Continuous Delivery:** Continuous delivery expands upon continuous integration by automatically deploying code changes to a testing/production environment. It follows a continuous delivery pipeline, where automated builds, tests, and deployments are orchestrated as one release workflow.

DevOps Key Terminology



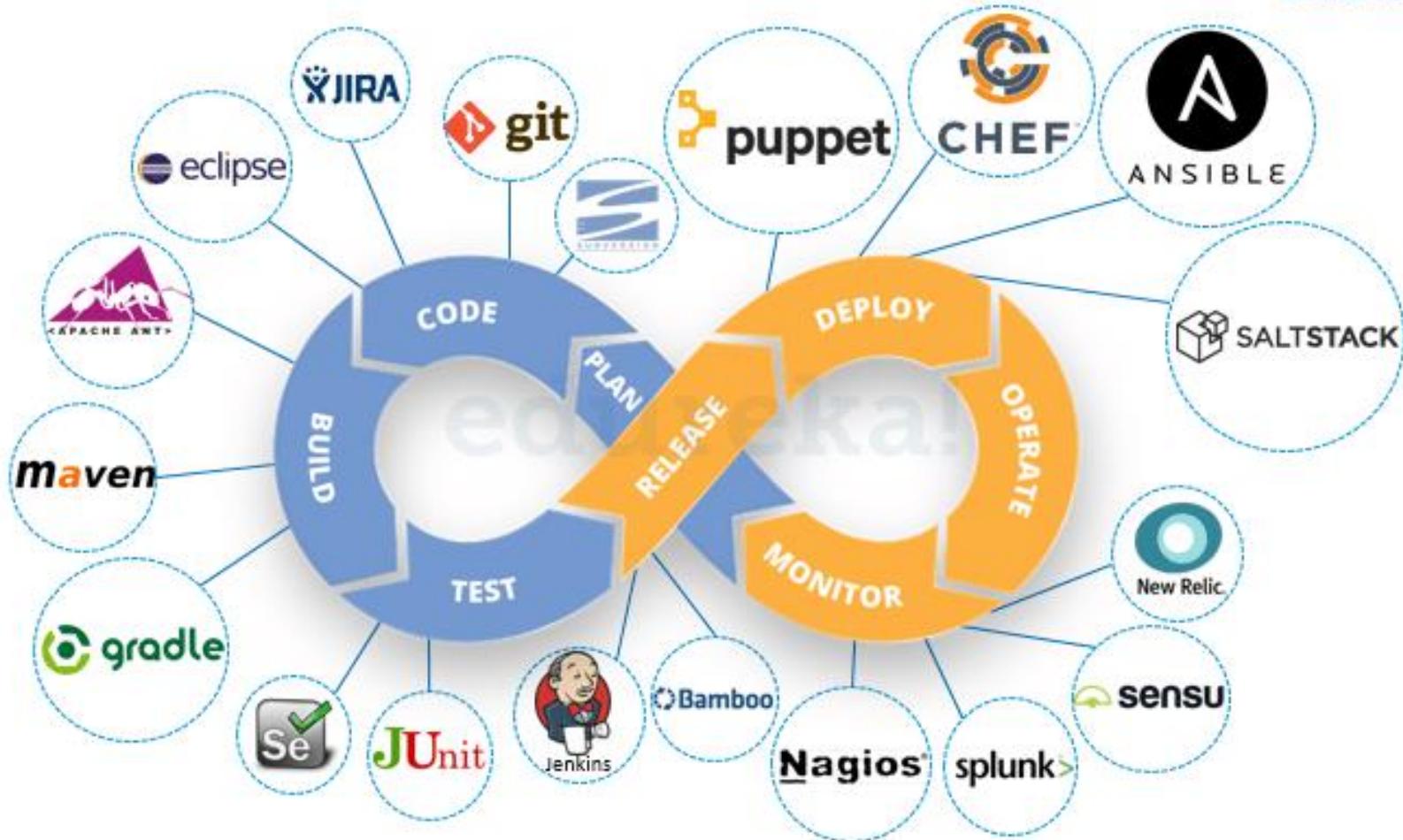
Microservices is an architectural style of developing a complex application by dividing it into smaller modules/microservices. These Microservices are loosely coupled, deployed independently, and are appropriately focused by small teams. With Microservices, developers can decide how to use, design, language to choose, a platform to run, deploy, scale, etc.

Containers and Dockers: Containers create a virtualization environment that allows us to run multiple applications without interrupting each other. With the container, we can quickly, reliably, and consistently deploy our application because containers have their CPU, memory, network resources, and block I/O that are shared with the host operating system's kernel.



Container Orchestration

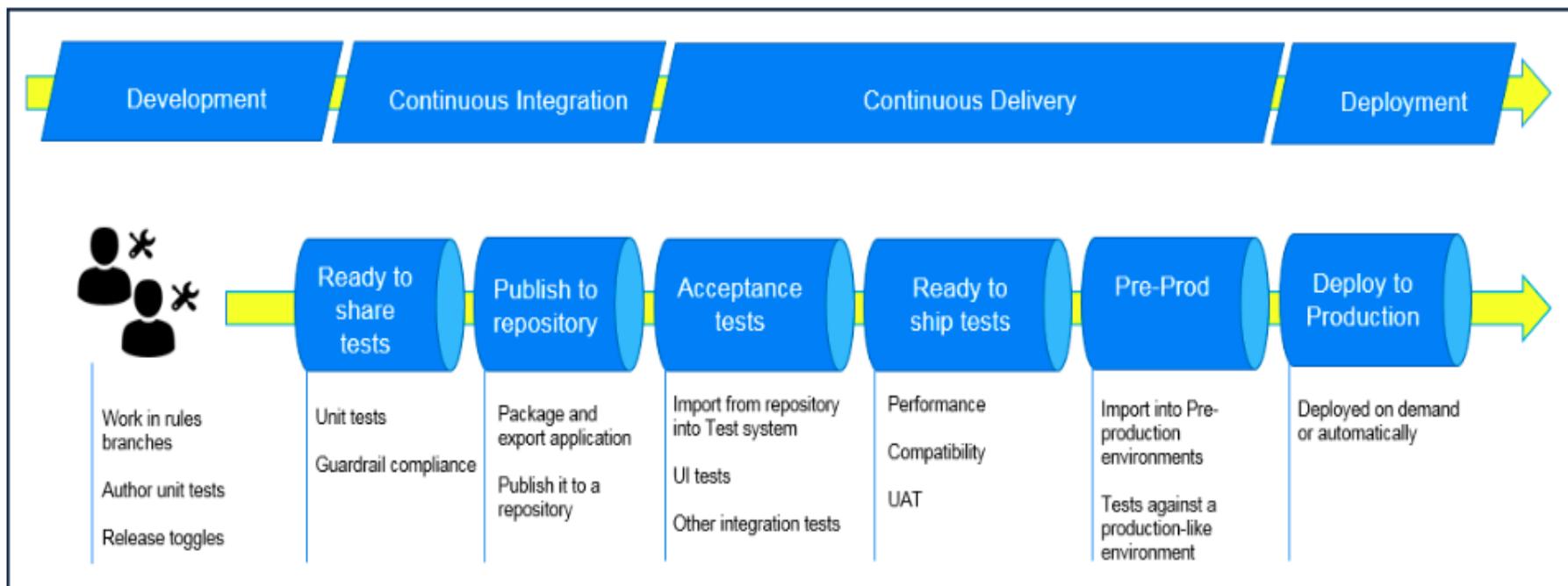
It is Automated, Arrangement, Coordination, and Management of containers and the resources they consume while deploying a multi-container packed application.



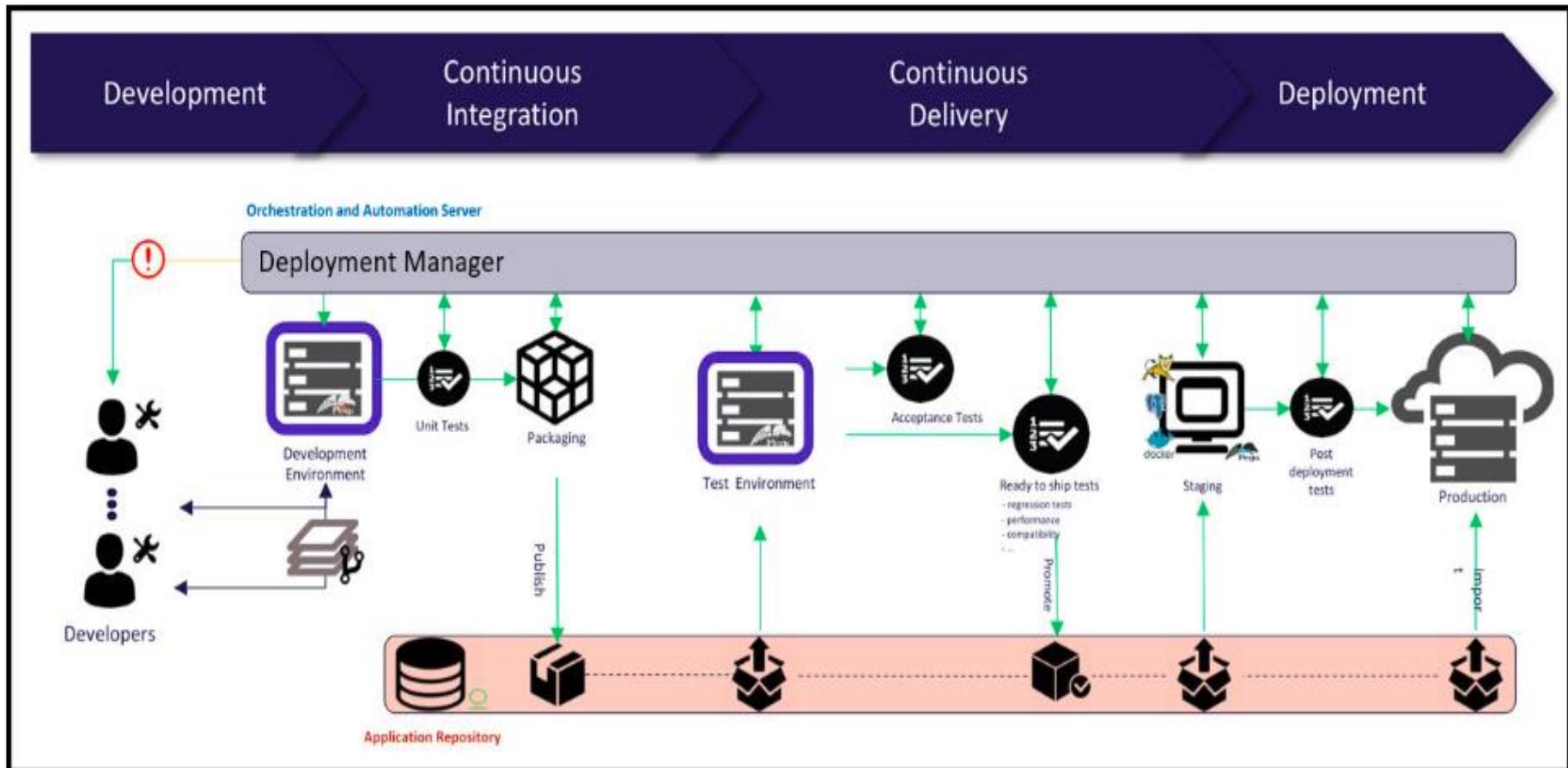
Delivery Pipeline

DevOps involves the concept of software delivery pipelines for applications. A pipeline is an automated process to quickly move applications from development through testing to deployment. The continuous integration and continuous delivery (CI/CD) phases are at the beginning of the pipelines.

The Continuous Integration portion of the pipeline is dominated by the development group. The Continuous Delivery portion of the pipeline is dominated by the quality assurance group.

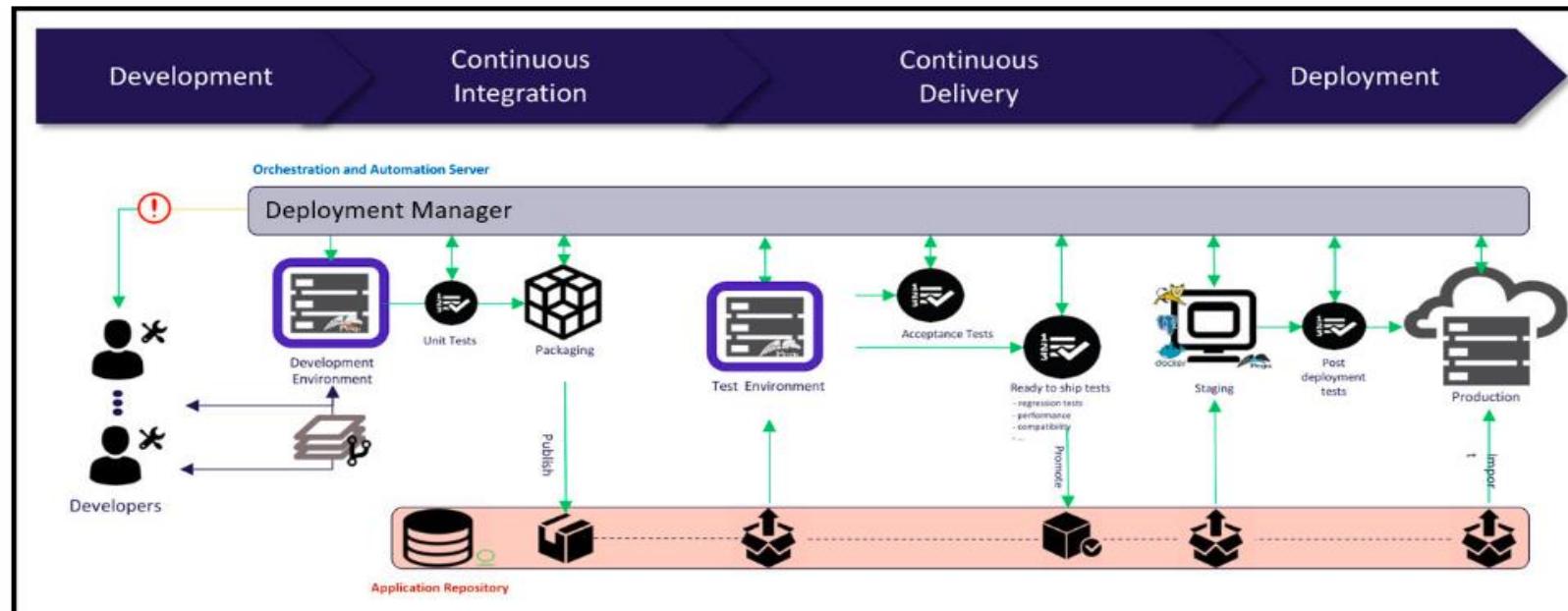


The pipeline is managed by some form of **orchestration and automation server** such as open source Jenkins. The following diagram illustrates an example of a release pipeline.



The **automation server** plays the role of an orchestrator and manages the actions that happen in continuous integration and delivery. In this example, Deployment Manager is used as the automation server.

A pipeline pushes application archives into, and pulls then from, **application repositories**. The **application repositories** are used to store the application archive for each successful build. There should be both a **development repository** and a **production repository**.



Release Management

- Release management is a structured model that refers to the process of *planning, designing, scheduling, testing, deploying, and controlling software releases*.
- It ensures that release teams efficiently deliver the applications and upgrades required by the business while maintaining the integrity of the existing production environment.

Necessity of RM:

1. Quality control
- 2 Risk mitigation
- 3 Customer satisfaction
- 4 Team collaboration

Strategies for Coordinating Releases Across Teams:

Release Calendar

The calendar should include information such as feature freeze dates, code freeze dates, testing periods, and final release dates.

Slow cadence release windows: A release window is a period of time when new software may be released.

Release Meetings: These meetings should discuss the release progress, potential roadblocks, and upcoming milestones.

Version control and Branching

Continuous Integration and Continuous Deployment (CI/CD):



Release Management Process Components



Release Pipeline

A particular release process from feature planning to delivery



Release Value Stream

The release processes that add or create value across the release pipeline

Release Management Process Components

Release Policy

Release policy consists of the definition of release types, standards and governance requirements for an organization



Release Template

Release template is a single or repeatable workflow process for release pipeline which includes human and automated activities.

Release Management Process Components

Release Plan

A structured plan of an instance of a release template developed for a specific release

Deployment Plan

Deployment Plan consists of activities to deploy a release to the production environment.



Release Management Process Components

Release Unit

The set of artifacts released together to implement a specific feature



Release Package

Release package is a combination of release units deployed together as a single release due to interdependencies, scheduling, or business priorities.

Release Management Process Components

Major Release

Infrequent release packages that include many release units that have a high or critical business impact.



Minor Releases

More frequent release packages with fewer release units that do not include critical components.

Release Management Process Components

Strategies of Release Management



Release Windows (slow cadence)

A release window is a period of time in which one or more teams release into production.

Release Train



Strategies of Release Management

Release Windows (quick cadence)

The quick cadence release are less likely to suffer from the bottleneck challenges associated with slow cadence release windows and release trains.



Continuous Release Availability

With this approach delivery teams are allowed to release their solutions into production whenever they need to.

Strategies of Release Management

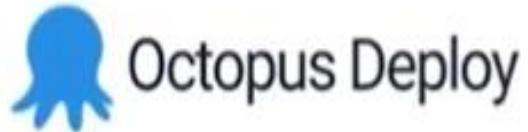
Benefits of Release Management



Release Management Tools



urban{code}



CLARIVE



Scrum is one of the leading Agile software development processes. Scrum has been recognized as one of the **best project management frameworks for handling rapidly changing or evolving projects**, especially those with technology or requirements uncertainty.

What is Scrum?

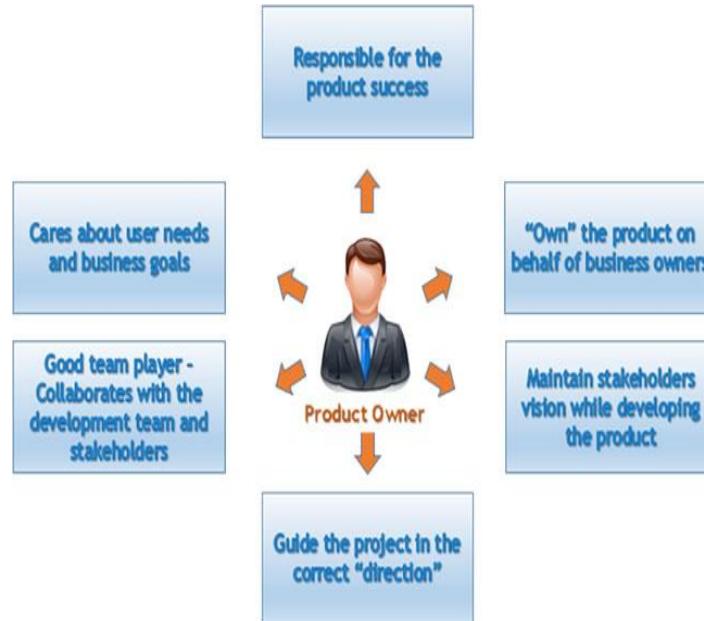
Scrum is a framework that helps agile teams to work together. Using it, the team members can deliver and sustain the complex product. It encourages the team to learn through practice, self-organize while working on the problem. Scrum is a work done through the framework and continuously shipping values to customers.

It is the most frequent software that is used by the development team. Its principle and lessons can be applied to all kinds of teamwork. Its policy and experiences is a reason of popularity of Scrum framework. **The Scrum describes a set of tools, meetings, and roles that help the teams structure. It also manages the work done by the team**

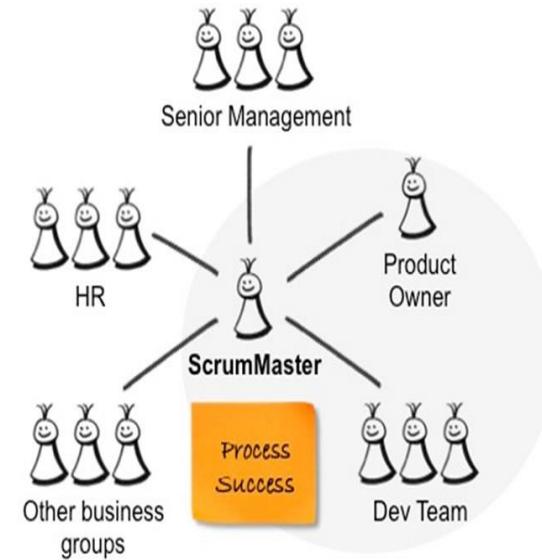
Scrum commonly used terms:

1. Product Owner
2. Scrum Master
3. Scrum Team
4. User Story
5. Product Backlog
6. Sprint plan
7. Sprint Backlog
8. Sprint Planning Meeting
9. Daily Stand-up, Sprint Review & Team Retrospective Meetings
10. Burndown Charts
11. Impediments

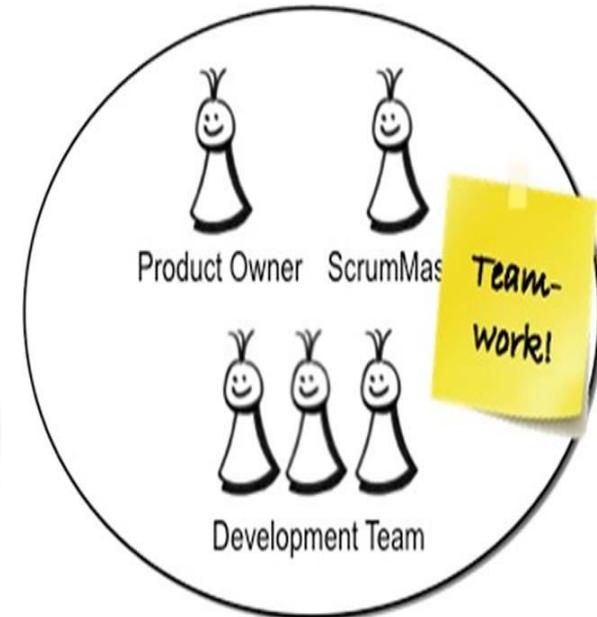
The **Scrum product owner** is typically a project's key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team.



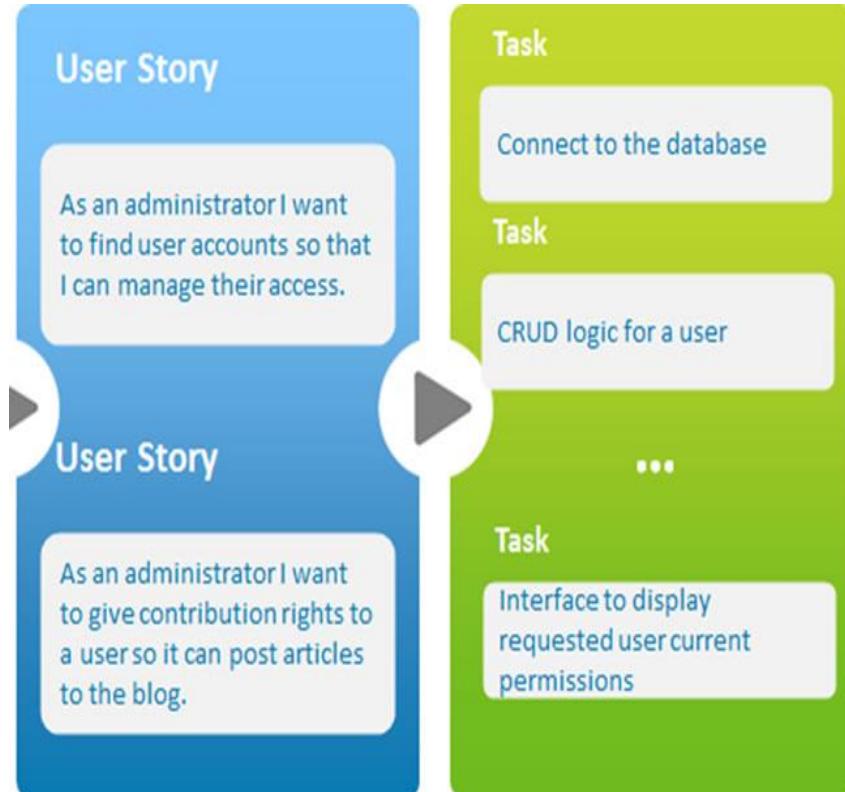
A **scrum master** is the facilitator for a product development team that uses scrum. The scrum master manages the process for how information is exchanged.



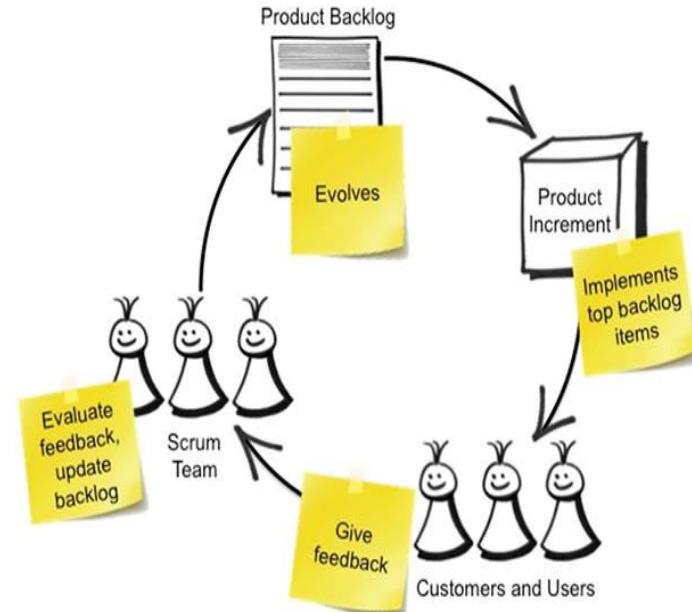
A **Scrum Team** is a collection of individuals working together to deliver the requested and committed product increments.



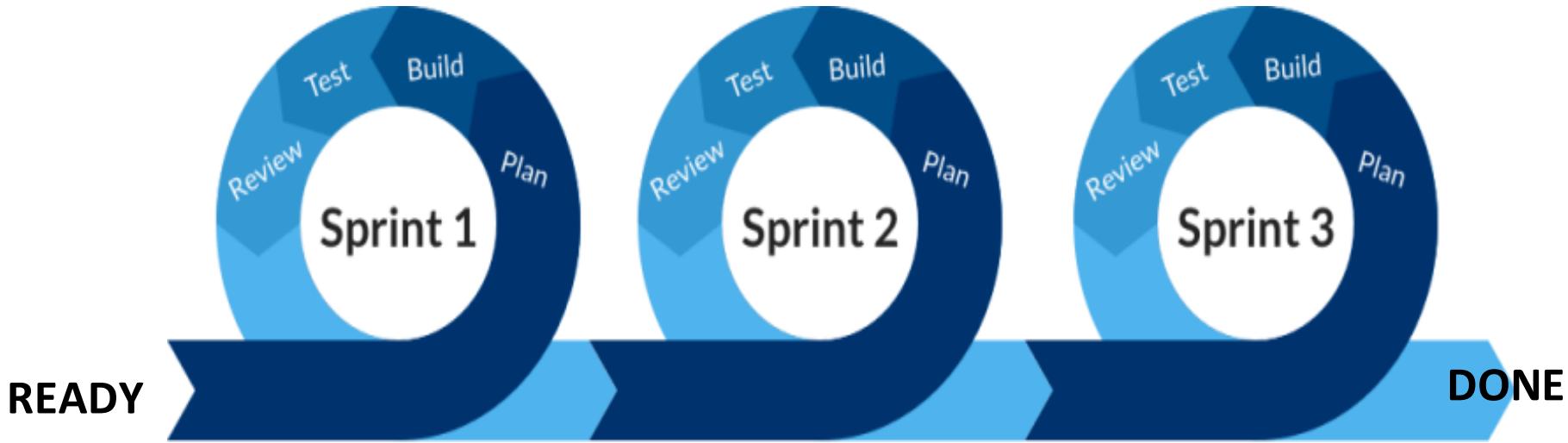
A user story is used to capture a description of a software feature from an end-user perspective. It describes the type of user, what they want and why, helps to create a simplified description of a requirement.



Product Backlog is simply a list of all things that needs to be done within the project and are usually sorted by value, risk, priority, and necessity. It is a sequence of highest to lowest priority, with each entry having a unique order.

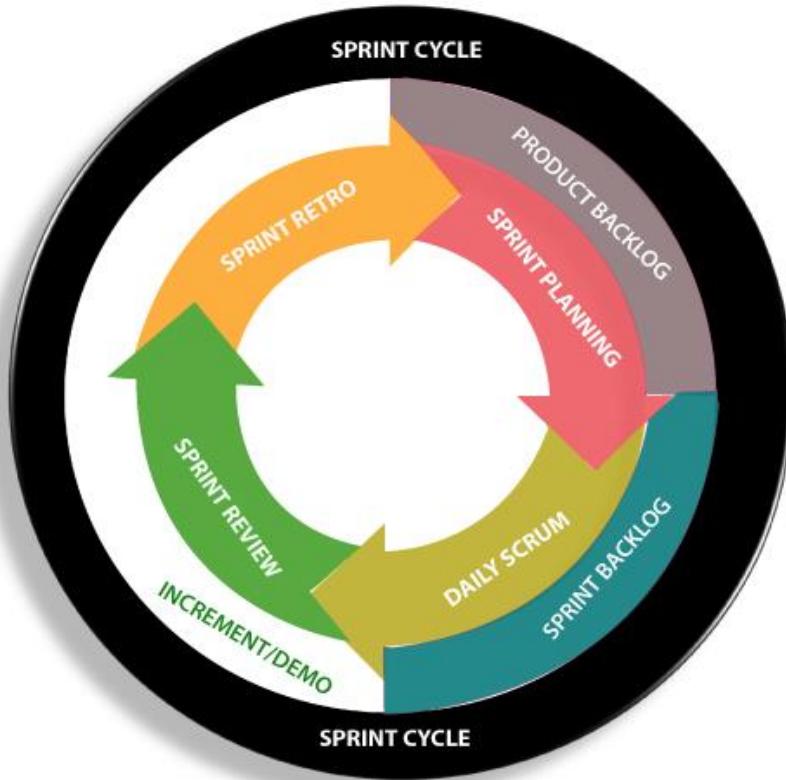


In the Scrum method of Agile software development, work is confined to a regular, repeatable work cycle, known as a **sprint or iteration**. Scrum sprints used to be 30 days long, but today many teams prefer shorter sprints, such as one-week or two-week sprints.

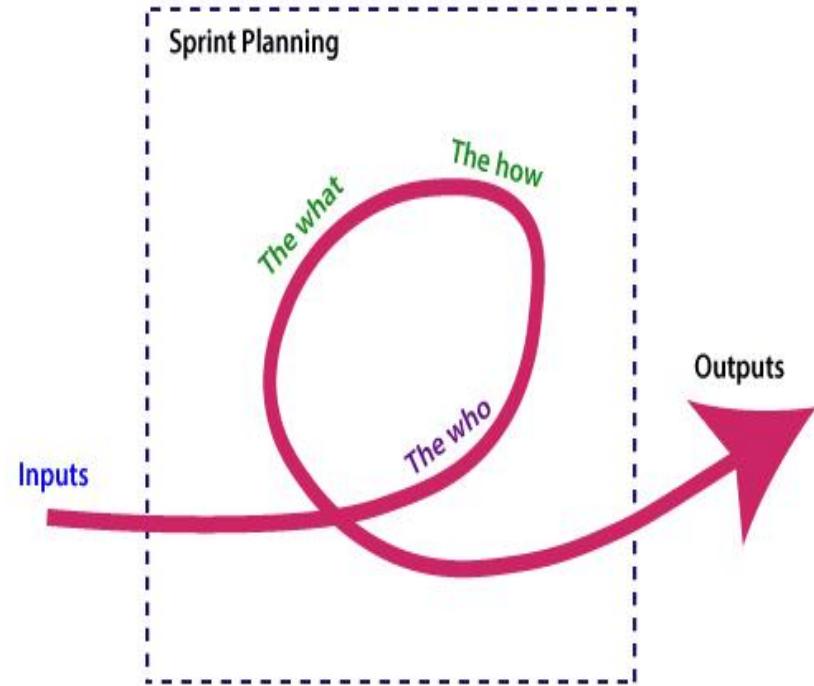


- With scrum, a product is built in a series of repetition called **sprints**. It breaks down big complex projects into bite-size pieces. It makes projects more manageable, allows teams to ship high quality, work faster, and more frequently. The sprints give them more flexibility to adapt to the changes.

- Sprint plan is an action in Scrum that kicks off the sprint. The primary purpose of sprint plan is to define what can deliver in the sprint. It also focuses on how the work will be achieved. It is done in combination with the whole Scrum team members.



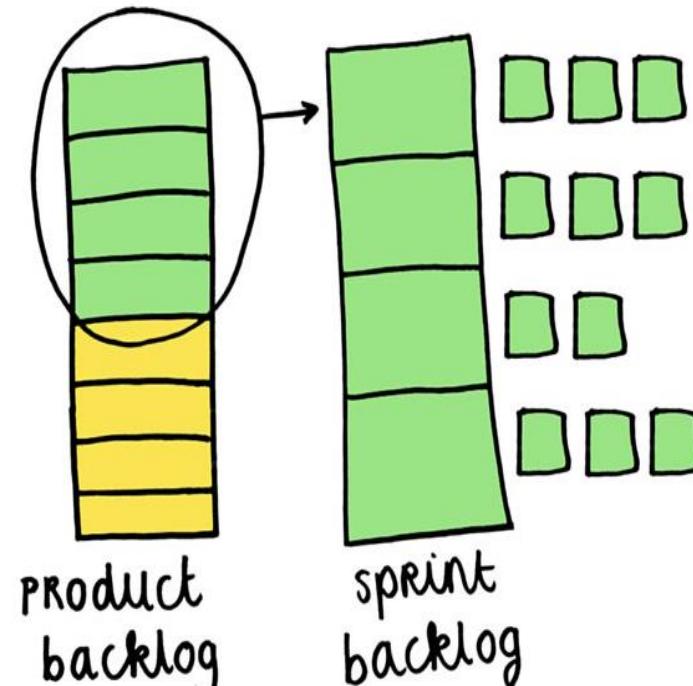
Factors affecting sprint planning



The **Sprint Planning Meeting** is the first meeting to kick off the sprint. It is attended by the ScrumMaster, Development Team and the Product Owner along with interested and invited stakeholders.



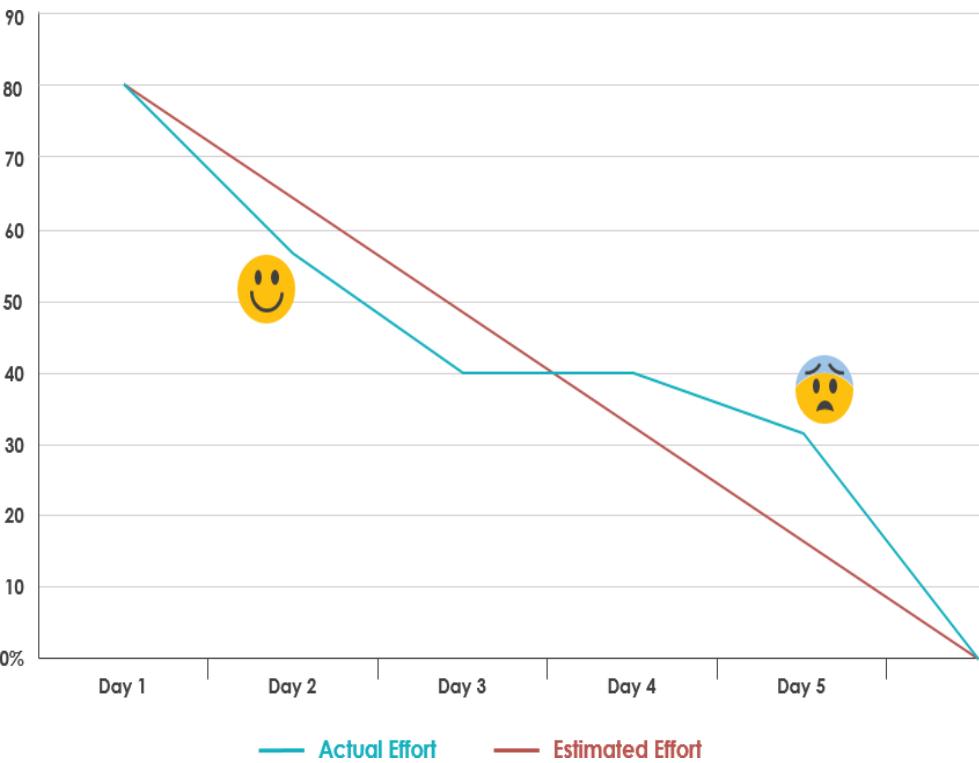
The **sprint backlog** is a list of tasks identified by the Scrum team to be completed during the sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story.



Daily Standup, Sprint Review & Team Retrospective Meetings



A **burndown** chart is a graphical representation of work left to do versus time.

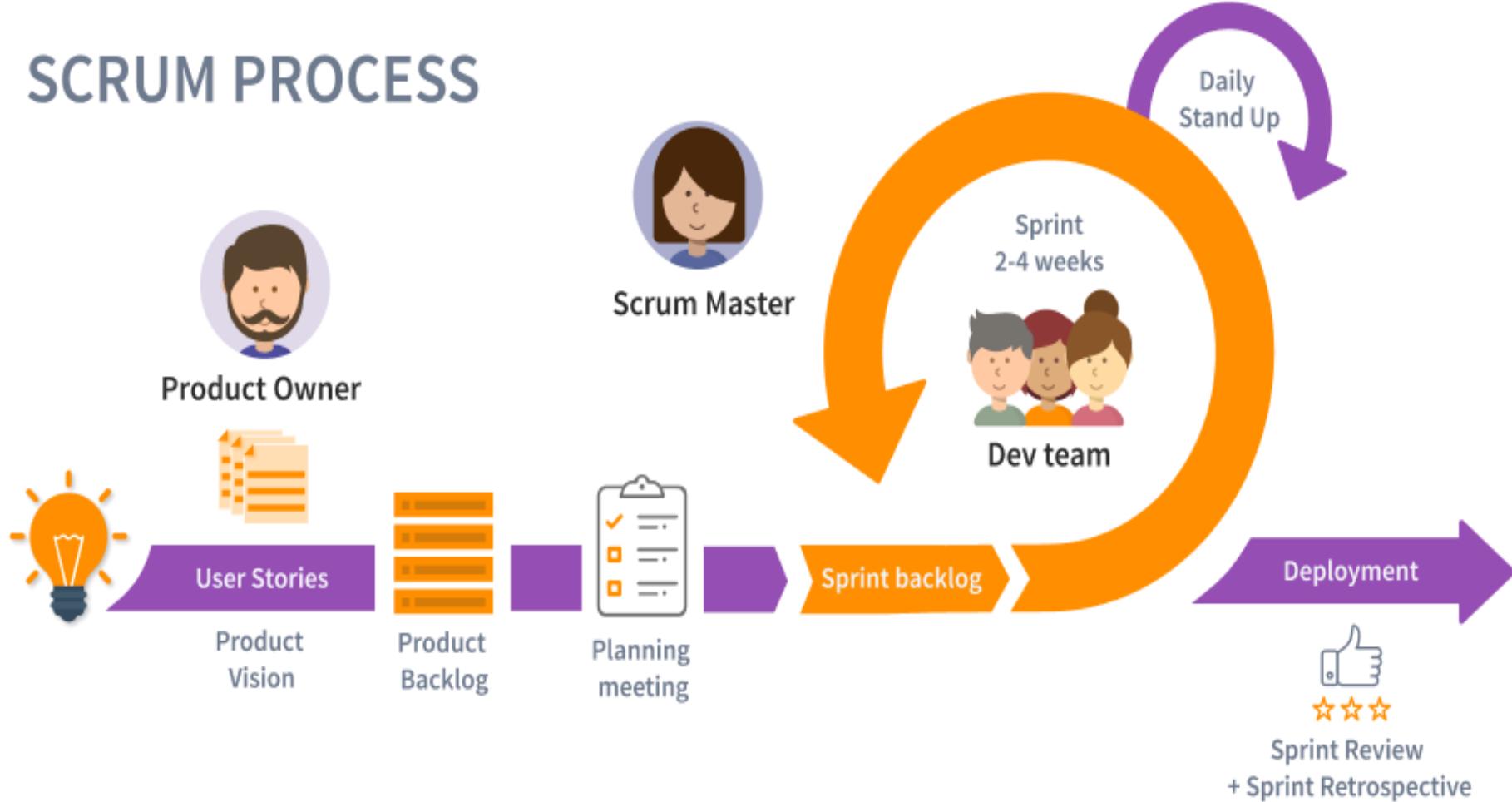


Impediments come in many forms:

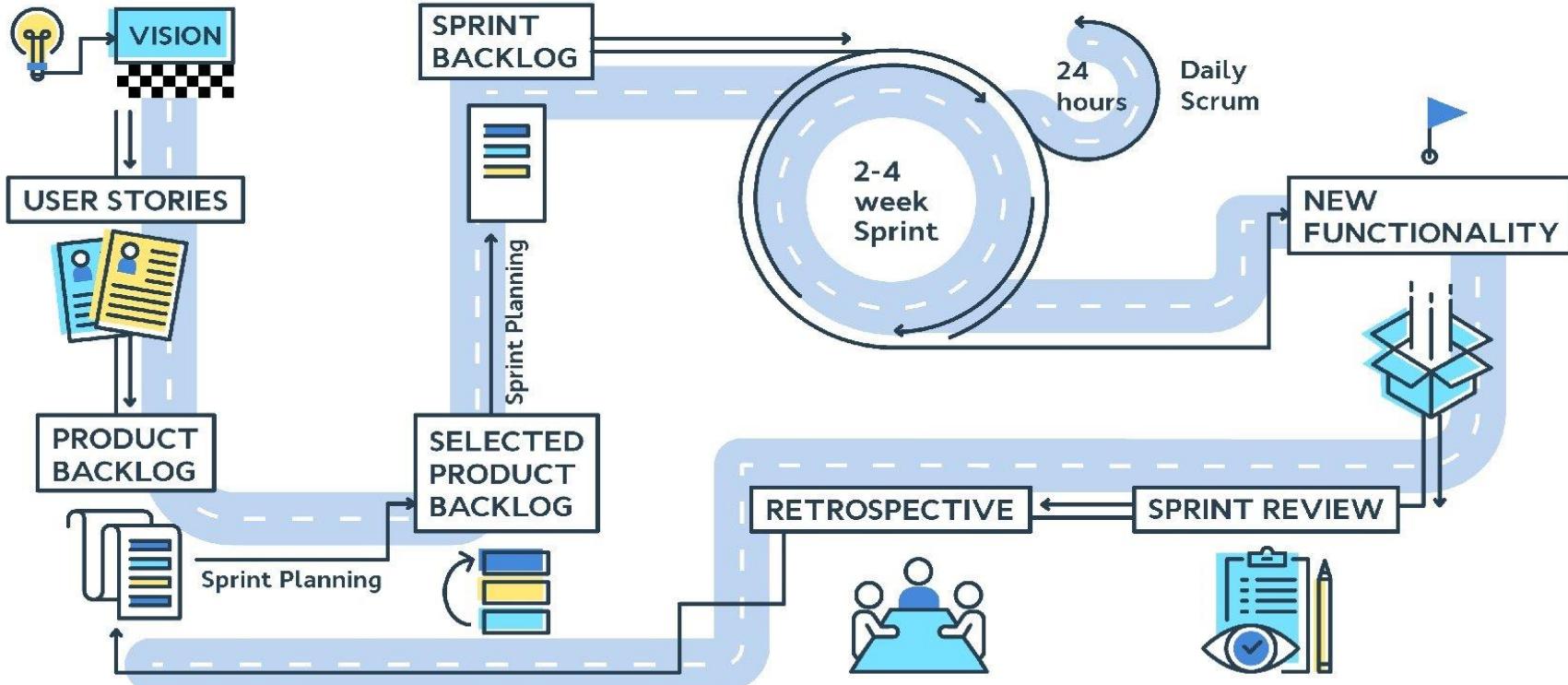
- A sick team member
- A missing resource
- Lack of management support
- Even a cold team room or Slow connection



SCRUM PROCESS



SCRUM PROCESS



The Scrum Framework

The Agile: Scrum Framework at a glance

Inputs from Executives,
Team, Stakeholders,
Customers, Users



Product Owner



The Team



Product Backlog

Team selects starting at top as much as it can commit to deliver by end of Sprint

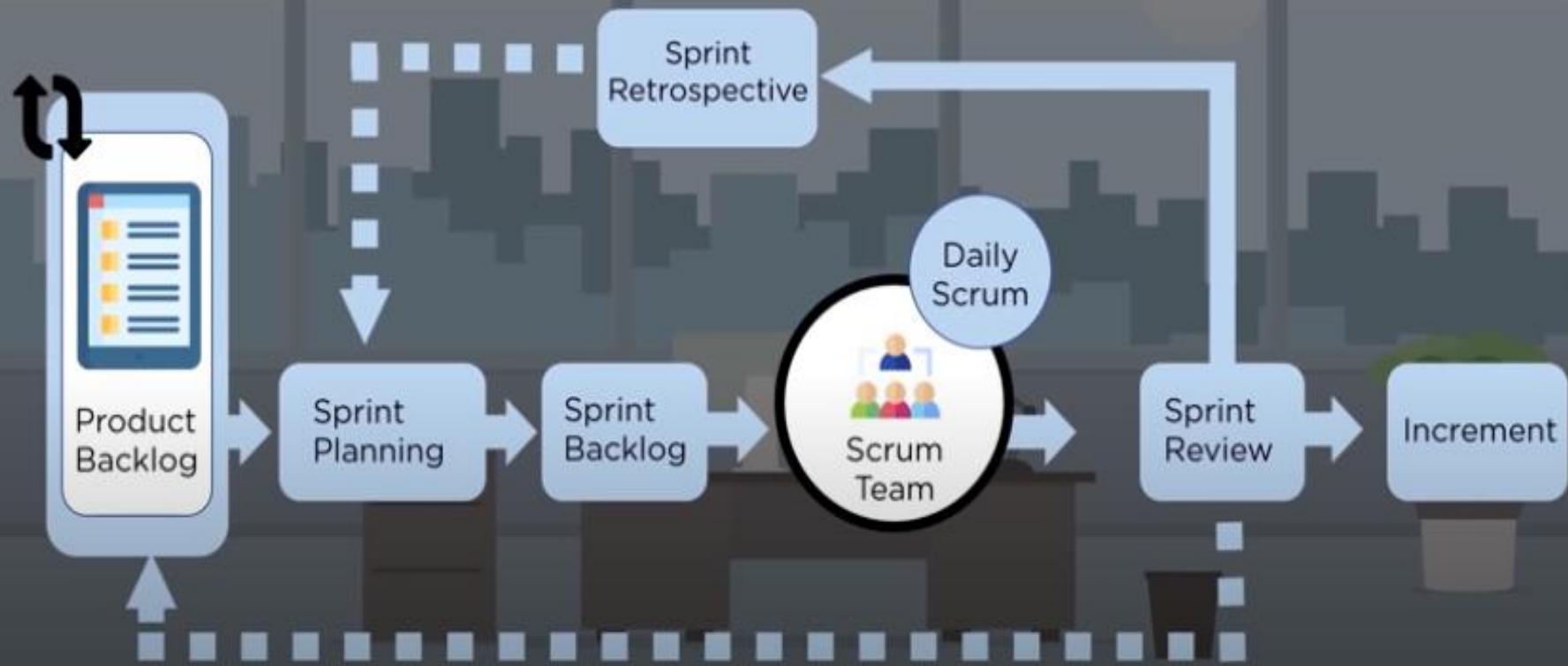
Sprint Planning Meeting



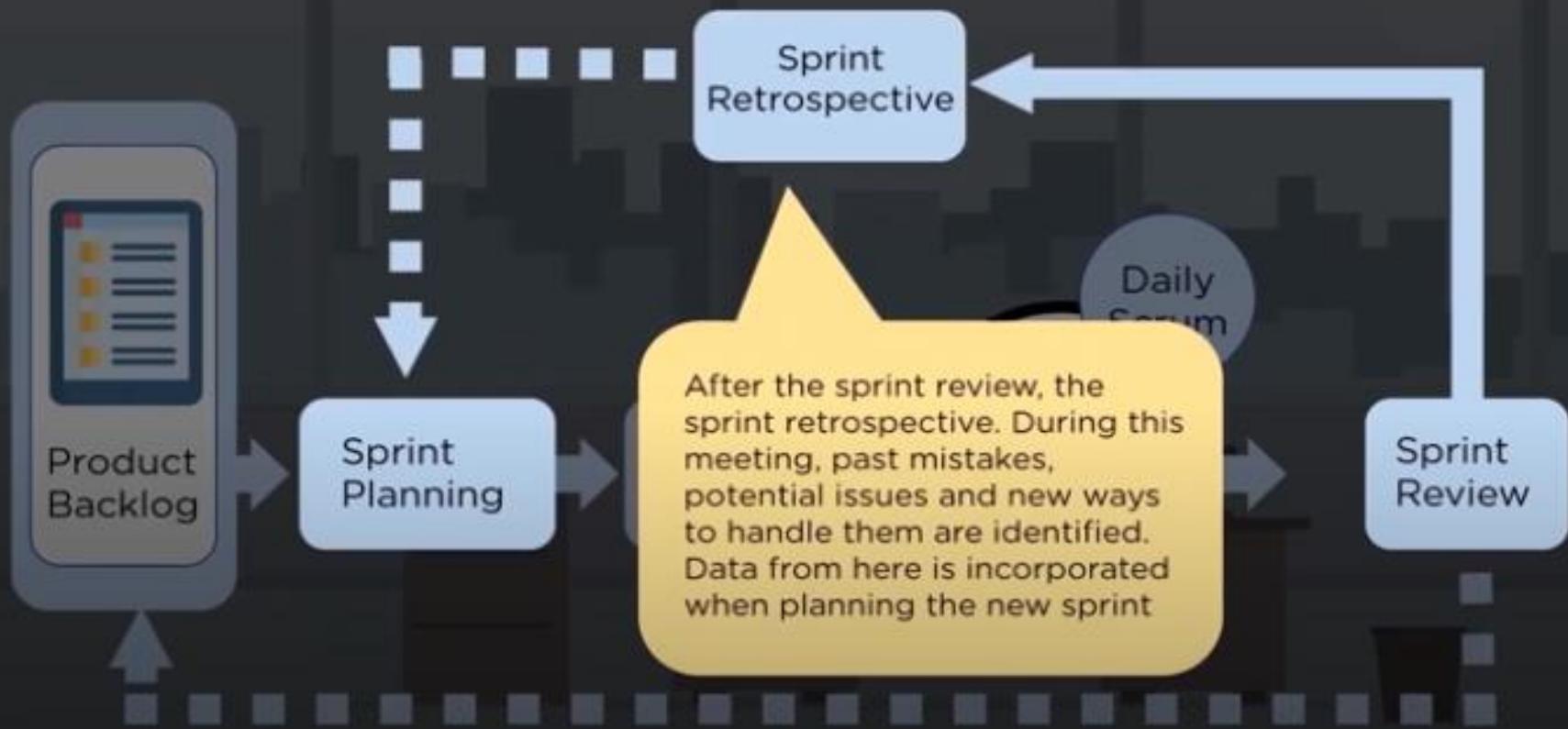
Sprint Backlog



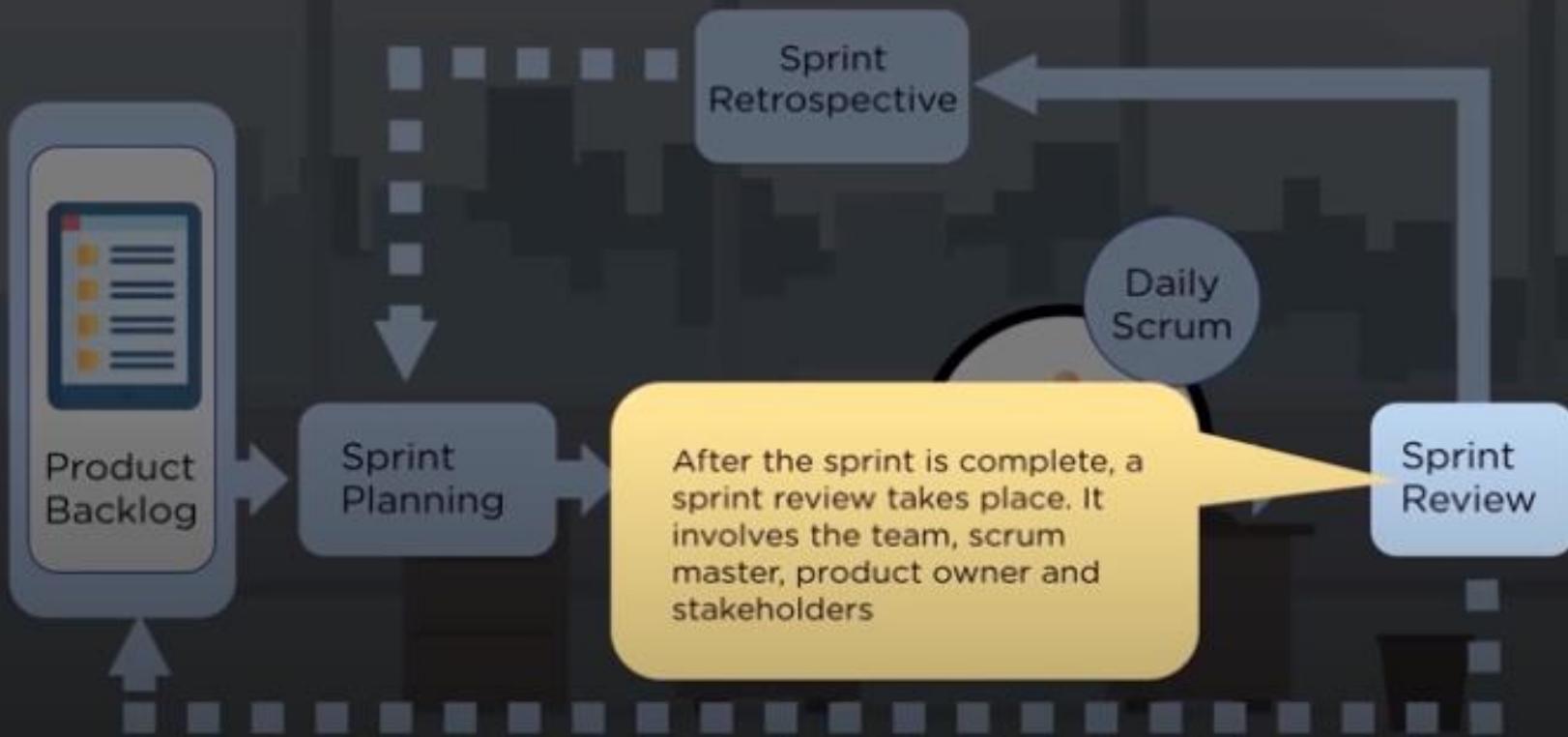
Scrum Framework



Scrum Framework



Scrum Framework



THE SCRUM FRAMEWORK



Different Scrum Artifacts

3 types of scrum artifacts include:

- Product backlog
- Sprint backlog and
- Product increments

Now we will see what these terms mean and how to create these artifacts.



Product Backlog

To put it in simple terms, a product backlog is a list of all the things that are required in the product. It's the final document to be referred to by the scrum team for anything related to the product. It's an ordered list of items which is owned by the Product Owner (PO).

The PO is responsible for creating, maintaining and prioritizing this list. The POs use this product backlog to explain the top requirements that need to be done during the sprint to the scrum teams.

The items in this list may or may not be in a technical language. It can even be a layman's language, but it should contain all the product requirements and the accompanying changes. Also, having a product backlog doesn't mean that the scrum team will only have this artifact to follow.

They can create their own detailed artifacts but those won't contradict or replace the product backlog. They will rather be in an alignment with the product backlog requirements.

Below is an Example of what a typical product backlog can look like:

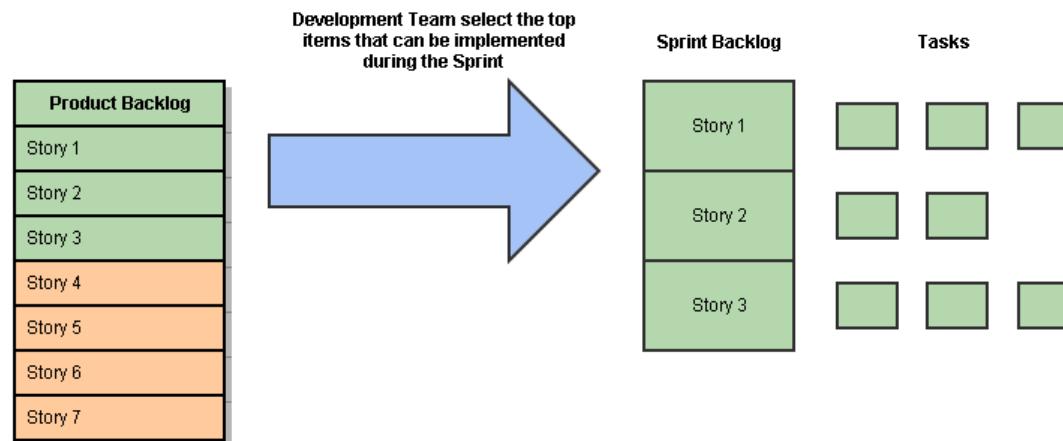
Story	Estimate	Priority
I want to login	4	1
I want to logout	2	2
I want to change password	1	3
I want to update address	3	4
I want to add a new home phone number	1	5

A product backlog should ideally follow the below rules:

- (i) It should be prioritized** – The items in the product backlog should be ordered as per their priority. This priority can be decided by the PO and the scrum team together. The prioritization factors can be any like benefit from the story point, the effort involved in the creation, complexity, customer priority etc.
It helps the team in understanding what needs to be delivered first.
- (ii) It should be estimated** – The stories should always be estimated as per the agreed definition, whatever that might be. This can be used for prioritization as well.
- (iii) It should be high level** – The stories in the product backlog are meant to be high level and should not go into the details. Creation of detailed user stories as per the requirement is up to the scrum team and not the PO.
- (iv) It should be dynamic** – The product backlog is not a final static document. It should be revisited as the PO receives inputs from the scrum team and the customer requirements become more and

Sprint Backlog

You might remember that the scrum teams work in short iterations of 2 to 4 weeks called a sprint. During these sprints, the scrum team identifies the items from the product backlog created by the PO, which they plan to deliver as a part of the next iteration. The items which the scrum team selects to work upon become a part of the sprint backlog. Thus they decide what functionalities are going to be there in the next iteration of the product. The scrum team is the one who decides what will go into the sprint backlog as they are the ones who are going to work on it.



Product Increments

A goal of the scrum process is to develop features in such a way that the product is in a completed state at the end of every sprint, so it could be released, demonstrated to clients for feedback, or used as a tool for testing. While it's not mandatory for an organization to release the product according to the schedule of scrum, this goal allows the state of the product to be part of the iterative process of development, testing, evaluation, and innovation that scrum encourages.

The Increment is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints.

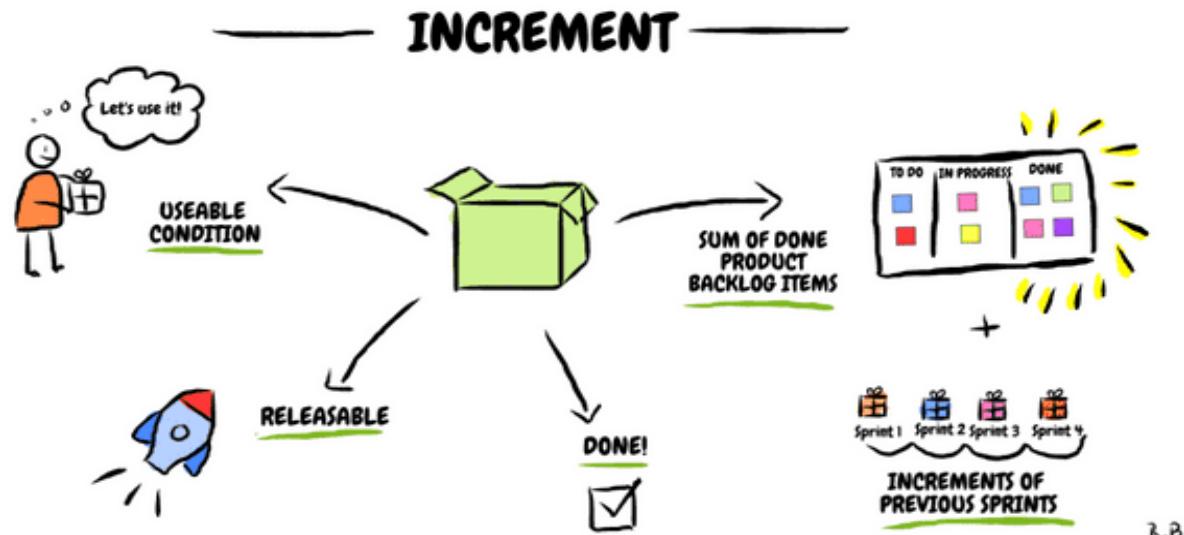
“DONE” INCREMENT

Every new Increment needs to be “Done”. It means that it needs to meet the Definition of “Done” created by the Scrum Team.

The main role of the definition of “Done” is to increase and support high quality of the increment. Definition of “Done” is created by the [Development Team’s](#) to have a common understanding about completed work and to ensure transparency.

What is more, definition of “Done” helps the [Development Team’s](#) to select the right number of Product Backlog items into [Sprint Backlog](#) during [Sprint Planning](#) session.

- Increment is sum of all items completed within the Sprint + value of increments from previous iterations
- When Increment is created, definition of “Done” needs to be met
- Decision about releasing the Increment is in the Product Owner’s hands, but it needs to be in useable/releasable state.



Increment

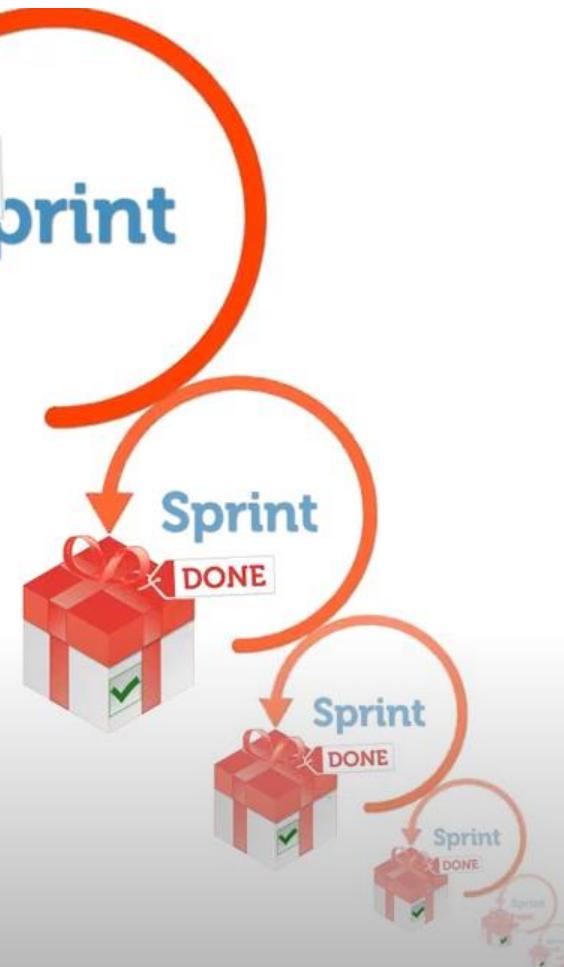


DONE Sprint



Our Scrum Team's Definition of Done

- Quality Criteria
- Quality Criteria
- Quality Criteria
- Quality Criteria
- Quality Crite



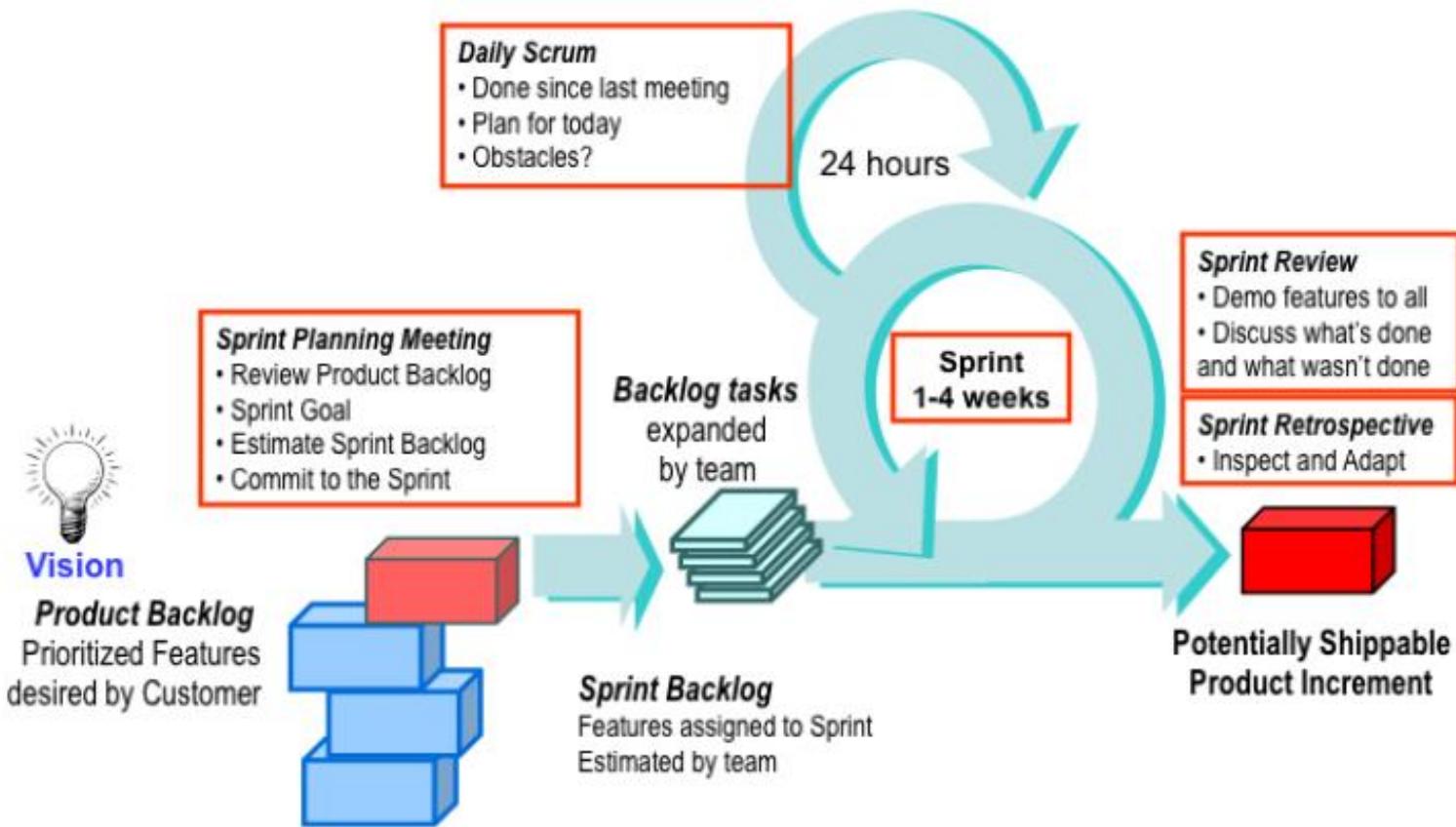


Exhibit 1. The Original Scrum Framework

Exhibit 1. The Original Scrum Framework

The project begins with a clear vision provided by the business, and a set of product features in order of importance. These features are part of the **product backlog**, which is maintained by the customer or customer representative referred to as the **Product Owner**. A time box commonly referred to as **an iteration or sprint**, is the set amount of time that the team has to complete the features selected.

Sprints are generally from **one to four weeks** in length, and that length is maintained throughout the life of the project so as to establish a cadence. The team selects items from the product backlog that it believes can be completed in the sprint, and creates a sprint backlog consisting of the features and tasks as part of the sprint-planning meeting.

Once the team has committed to a sprint backlog, the task work begins. During this time in the sprint, the team is protected from interruptions and allowed to focus on meeting the sprint goal. No changes to the sprint backlog are allowed; however, the product backlog can be changed in preparation for the next sprint.

During the sprint, the team checks in daily with each other in the form of a 15-minute meeting known as a scrum. The team stands in a circle and each member states what they did yesterday, what they plan to do today, and what is getting in their way.

At the end of the sprint, the team demos the work they have completed to the stakeholders and gathers feedback that will affect what they work on in the next sprint. They also hold a retrospective to learn how to improve. This meeting is critical, as its focus is on the three pillars of Scrum: transparency, inspection, and adaptation.

The Application of Scrum

Scrum is applied by following a set of ceremonies, or meetings. Required Scrum ceremonies include the sprint planning meeting, the daily scrum, the sprint review and the sprint retrospective. Working in time boxes called sprints is also required. Release planning meetings are optional and allow for the planning and forecasting of groups of sprints.

Sprint Planning Meeting

The sprint-planning meeting is held on the first day of every sprint. The ScrumMaster, Product Owner, and Team are all in attendance. The Product Owner presents the set of features he or she would like to see completed in the sprint (the “what”) then the team determines the tasks needed to implement these features (the “how”). Work estimates are reviewed to see if the team has the time to complete all the features requested in the sprint. If so, the team commits to the sprint. If not, the lower priority features go back into the product backlog, until the workload for the sprint is small enough to obtain the team's commitment.

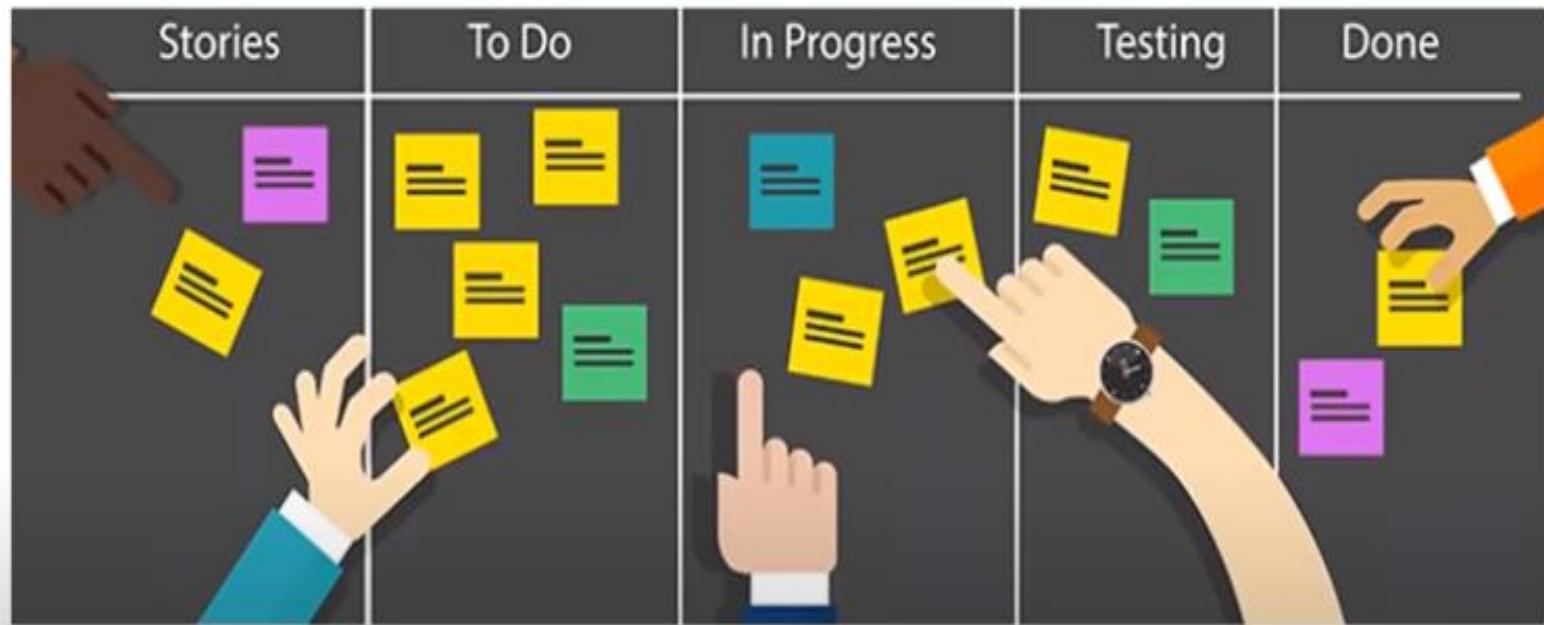
Tracking Progress

Once the sprint-planning meeting is complete and the team has made a commitment, the team begins to track its progress using highly visible information radiators. These radiators include the burndown chart and the task board.

The task board is used by the team to track the progress of the tasks for each feature. The minimum columns used are To Do, Doing, and Done. Teams will have their daily scrum meeting at the task board, and move items across the board when stating what they did yesterday, what they plan to do today, and what obstacles they are grappling with. See Exhibit 2 for an example task board for a software development project.

Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9 Code the... 2 Test the... 8	Test the... 8 Code the... DC Test the... SC	Test the... 6 Test the... SC	Code the... D Test the... SC Test the... SC Test the... SC Test the... SC Test the... SC
As a user, I... 5 points	Code the... 8 Code the... 4	Test the... 8 Code the... DC		Test the... SC Test the... SC Test the... SC Test the... SC

Exhibit 2. Scrum Task Board Example



The burndown chart shows the trend line of the amount of work left to do in the sprint. The x-axis is the number of days in the sprint, and the y-axis is the number of hours for all the tasks that were defined in the sprint-planning meeting. Over the days of the sprint, the line indicating the amount of work left to do should trend down to zero by the last day of the sprint.

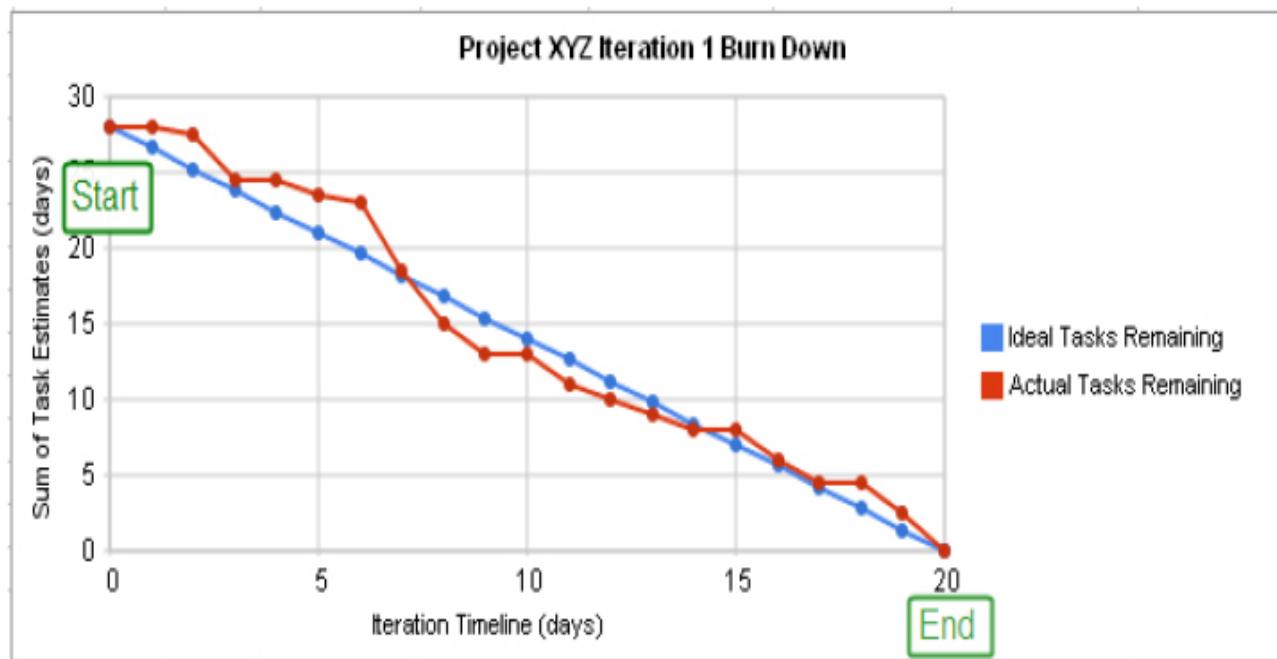


Exhibit 3. Sprint Burndown Chart Example

Sprint progress is tracked using the burndown chart, the task board, and the daily scrum. In combination, these three things can provide a clear picture of what's being worked on, what's completed, what's still to be done, whether or not it will be completed in time, and what might be preventing the team from meeting its sprint and/or release goal.

Sprint Review

At the end of the sprint, the team invites stakeholders to a sprint review meeting where the features that were completed in the sprint are demo'd and feedback is requested. The Product Owner keeps track of the feedback and incorporates it as needed into the product backlog.

Once the review is complete, the team (without the stakeholders) conducts a retrospective to determine what they did well that they wish to continue doing, what they struggled with, and what recommendations they have for change going forward. An action plan is created and these items are implemented over the course of the next sprint, and reviewed for efficacy in the next sprint retrospective.

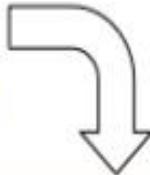
Release Planning

Release Planning is also part of Scrum, and is a way to do long-term planning for a time box that consists of multiple sprints. This is often done quarterly, and the results of the quarter do not have to be a release to the customer, but may simply be an internal release to confirm system integration and validation. Exhibit 4 shows how release planning fits in with the rest of the Scrum framework.

The entire team attends the release-planning meeting, where the Product Owner presents the features she/he would like to see completed in the quarter. The team does not task out these features however, but instead provides gross level estimates to determine what features can be done in what sprint, and how many of these features can be completed by the end of the quarter. Release planning can be feature-driven (how many sprints will it take to complete this set of features?), time-driven (how many features can we expect to have completed by this deadline?) or cost-driven (given this budget, what does our schedule look like and what features will we have done before we run out of money?).

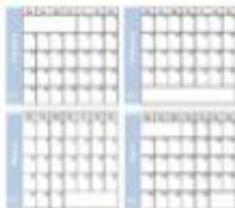


Vision



Release Planning Meeting

- Establish release goal
- and the highest priority features the release will contain
- Determine the probable delivery date, costs



Sprint Planning Meeting

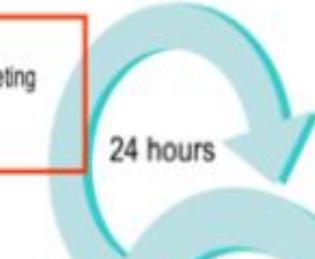
- Review Product Backlog
- Sprint Goal
- Estimate Sprint Backlog
- Commit to the Sprint



Daily Scrum

- Done since last meeting
- Plan for today
- Obstacles?

24 hours



Sprint
1-4 weeks

Backlog tasks expanded by team

Sprint Review

- Demo features to all
- Discuss what's done and what wasn't done

Sprint Retrospective

- Inspect and Adapt



Potentially Shippable Product Increment

Exhibit 4. Release Planning in Scrum

Project Name	OpenCart (Frontend)				
Client	OpenCart				
Created By	Name of the Product Owner				
Creation Date	DD-MM-YYYY				
Approval Date	DD-MM-YYYY				
Epic	User Story ID	Feature/Title	User Story	Status	Acceptance Criteria
OpenCart_Epic_001 : For a new e-commerce website to launch, the highest Business Value will be when a new user is able to buy an item from the website.	US001	Registration	As a First-time visitor to the e-commerce website, I want to register my account, So that I can login to application.	New	New user should be able to register account with valid data.
	US002	Login	As a registered user, I want to login to the website, So that I can see my account details etc..	New	System must validate user credentials and allow login if credentials are correct..
	US003	Logout	As a registered user, I want to logout from website, So that no one else can't access my account.	New	System must logout after login.
	US004	>User search products	As a user, I want to be able to search items, So that I can add them to cart and do payment.	New	User should be able to search products and add them to cart.

Project Name	OpenCart (Frontend)				
Client	OpenCart				
Created By	Name of the Scrum Master				
Attendees	Scrum Team				
Creation Date	DD-MM-YYYY				
<hr/>					
Epic	User Story ID	Feature/Title	User Story	Story Points	Sprint
OpenCart_Epic_001 : For a new e-commerce website to launch, the highest Business Value will be when a new user is able to buy an item from the website.	US001	Registration	As a First-time visitor to the e-commerce website, I want to register my account, So that I can login to application.	8	1
	US002	Login	As a registered user, I want to login to the website, So that I can see my account details etc..	5	1
	US003	Logout	As a registered user, I want to logout from website, So that no one else can't access my account.	3	1
	US004	User search products	As a user, I want to be able to search items, So that I can add them to cart and do payment.	5	3
<hr/>					
Story Points	Hours				
1	1 Hour/ Day (Depends on company)				
0,1,1,2,3,5,8	Fibonacci series				

Developer Tasks	QA Tasks
Under standing Requirements	Under standing Requirements
Desing	Writing Test Scenarios
Coding	Writing Test Cases
Unit Testing	Test Case Reviews
Integration Testing	Test Data Preparation
Code Review	Test Environment Setup
Bug Fixes	Test Execution
Team Meetings	Re-Testing Bugs
Any other...	Team Meetings
	Automation
	Any other...

✓ **Kanban**, is a Japanese term for “signboard” or “Billboard” that indicates “available capacity (to work)”. Kanban is a concept related to lean and just-in-time (JIT) production, where it is used as a scheduling system that tells you what to produce, when to produce it, and how much to produce.

✓ Kanban is a visual system for managing work as it moves through a process. Kanban visualizes both the process (the workflow) and the actual work passing through that process.

To Improved communication through visual management.

Japanese Meaning = Visual Signal or Signal Board or Billboard

Invented
at Japan

By Toyota

On 1940

Demand Based Supply

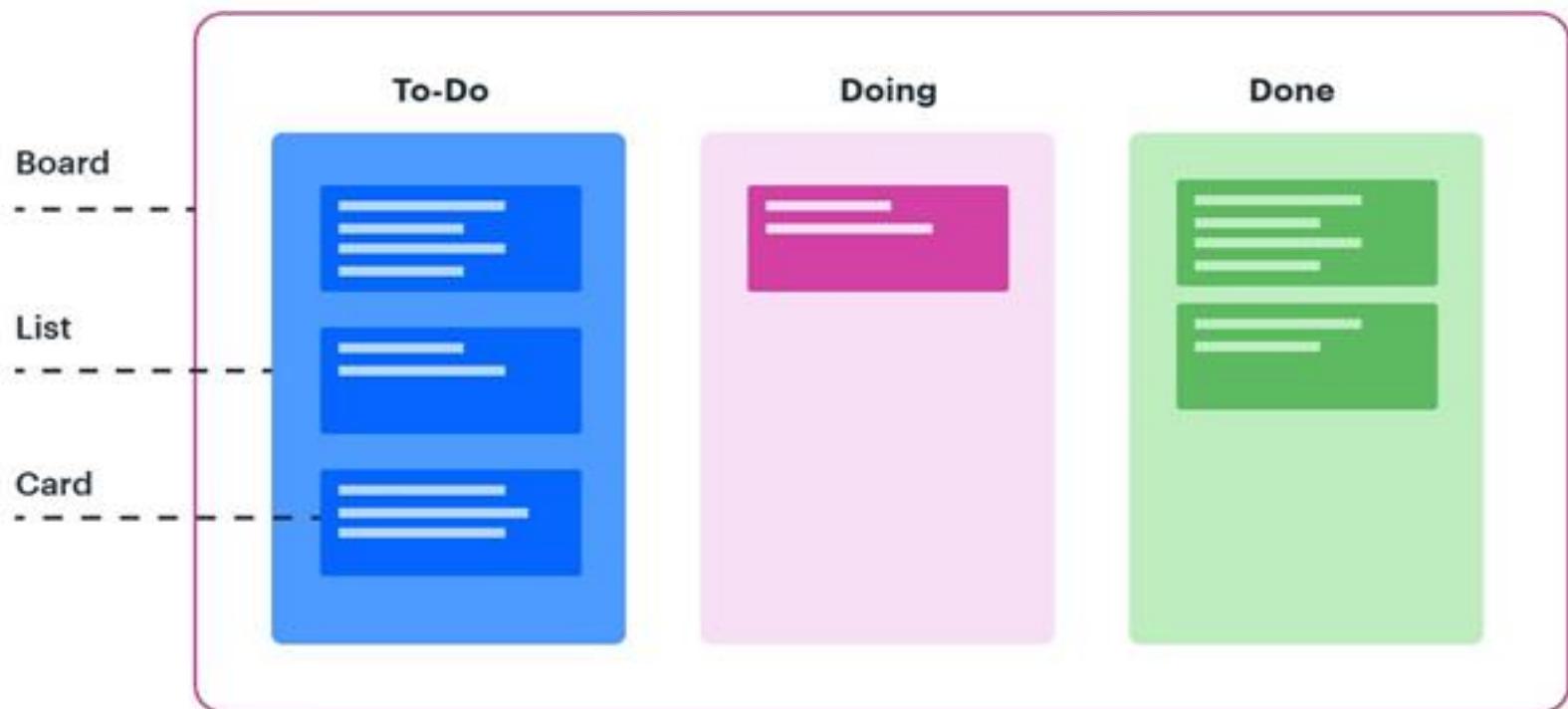
Reduce Waste

Maximize Value

Kanban
Principles



Kanban project management framework



How does Kanban work?

- Kanban method revolves around the kanban board. It is a tool that visualizes the entire project to track the flow of their project. Through this graphical approach of Kanban boards, a new member or an external entity can understand what's happening right now, tasks completed and future tasks.
- The first step in the introduction of Kanban is to visualize the workflow. This is done in the form of a Kanban board consisting of a simple whiteboard and sticky notes or cards. Each card on the board represents a task.
- **Kanban board indicates:**
 - ❖ the current tasks that are being performed
 - ❖ the tasks to do in the future
 - ❖ the tasks that are completed

Kanban cards:

- The tasks and stories are represented by Kanban cards. The cards represent the actual work.
- When we look at a card it depicts a work item/task in the work process.
- Cards are usually used to communicate progress with your team, it represents information such as status, cycle time, and impending deadlines.
- The current status of each task is known by displaying the cards in separate columns on the board.

- Kanban system measures the work cycle being completed through the principle of Work in Progress (WIP). WIP has certain limits and a pre-defined specific status.

WIP Limit:

- you can place a WIP (Work in Progress) limit on the column. ***The WIP limit means the maximum number of cards that can stay on that column.*** The label in the Doing column also contains a number, which represents the maximum number of tasks that can be in that column at any point of time. i.e., the number associated with the **Doing** column is the WIP (Work-In-Progress) Limit.
- cards <=the WIP limit (The cards represent the actual work).
- You can use positive numbers to limit work-in-progress, and this limit number can be placed on the top of the columns in both physical and digital Kanban boards. Any individual of the team can manage the state of his card, and the entire team can visualize the workflow.
- Limiting WIP in order to maintain consistent standards is one of the core principles that govern the Kanban methodology in Agile. It is extremely important for the team to complete the current tasks in the prescribed order

- As Kanban **focuses on breaking down work into small tasks, visualizing them**, and getting few items in any given work state.
- In the Kanban board, **work always moves from left to right**. And, you pick work from the column to your left when you have completed all your existing work items or when an urgent task surfaces.
- . This helps in increasing visibility teams as the teams can see the progress through every stage of development and prepare for the upcoming tasks to deliver the product “just in time”!
- As the development evolves, the information contained in the table **changes**, and when a new task comes into play, a new “card” is created.
- The Kanban method requires **communication and transparency** so that the **members of a team can know exactly at what stage the development is** and can see the status of the project at any time. visualize and examine specific parts of the workflow to identify bottlenecks in order to remove them.
- Kanban Board is updated **on a daily basis** as the team progresses through the development

Kanban Uses A Pull Approach:-

- Pull approach is used as and when a task is completed in the Doing column. Another card is pulled from the To Do column.
- THAT IS as Kanban project management uses a pull-based system, and when a developer is free, he/she can pull a card from the to-do column to the dev column to develop that.
- A key aspect of Kanban is to reduce the amount of multi-tasking that most teams and knowledge workers are prone to do and instead encourage them to “Stop Starting! And Start Finishing. THE WIP – Work-in-Progress – Limits defined at each stage of the workflow on a Kanban board encourage team members to finish work at hand and only then, take up the next piece of work.

Therefore using Kanban board we can :

- Provides easy access to everyone involved in the project.
- Facilitates communication as and when necessary.
- Progress of the tasks are visually displayed.
- Bottlenecks are visible as soon as they occur.

VISUAL MANAGEMENT



ORGANIZE
THE PROCESS



KANBAN IS A PROCESS
DESIGNED TO HELP TEAMS WORK
TOGETHER MORE EFFECTIVELY

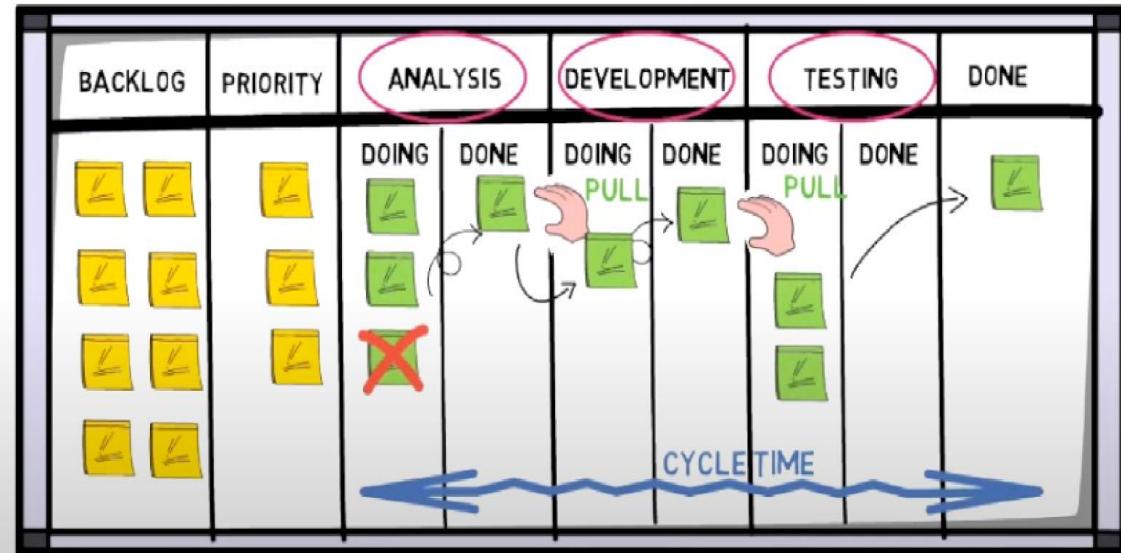
DEFINED BY THE TEAM
BASED ON THEIR CAPACITY

3

2

2

WIP LIMIT



LEAD TIME

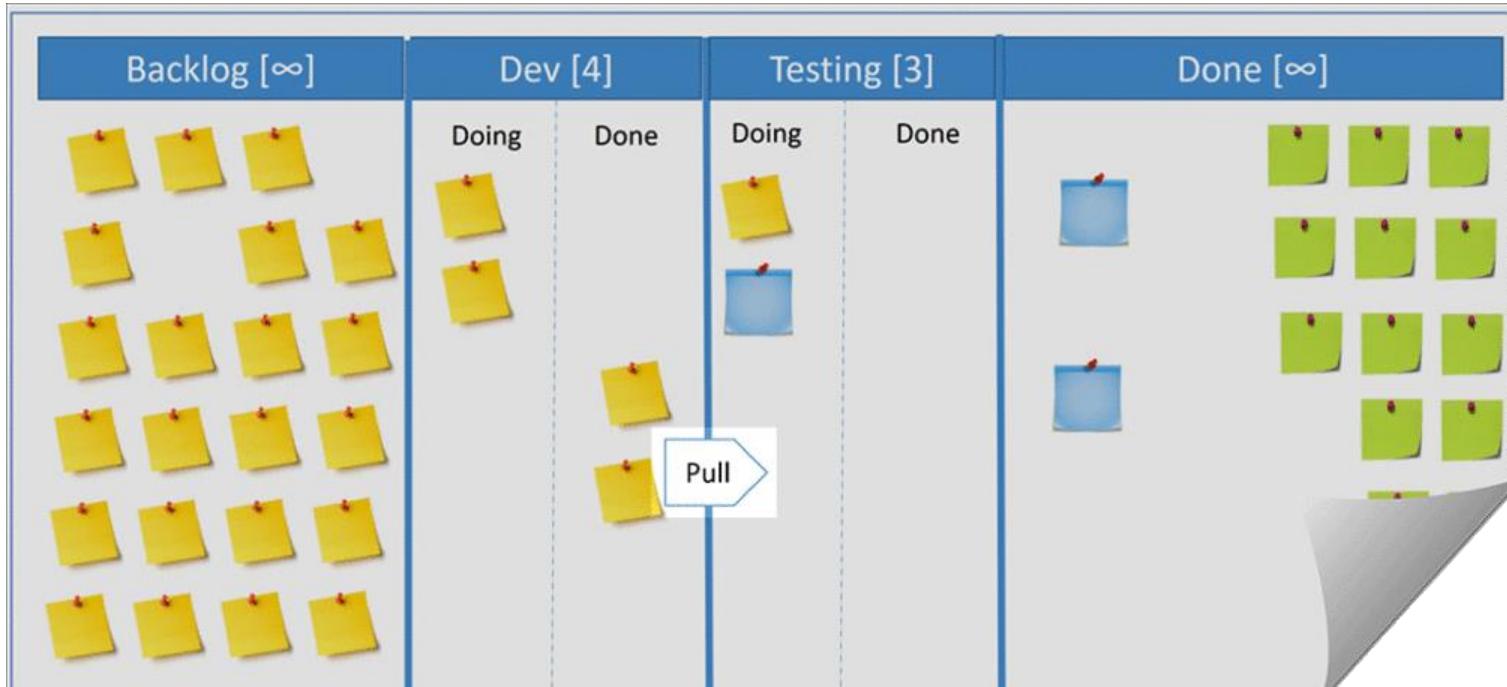


Understanding the Pull Based System

As we discussed earlier, **Kanban is a pull based System**. Implementing the WIP Limit, active the Pull based system. That means Thew team pulls a new task to its WIP column based on its current tasks and limits.

Even there are rooms in the current column, the team decide to pull a new story/task or finish the existing stories or task in the column.

The below three picture demonstrate three scenario, where the developer or tester have the opportunity can pull new story or task.



Benefits of Kanban

Kanban is useful and beneficial if you use it to cater the work items that best fit for Kanban. like Production support, Adhoc Requests , unplanned work, portfolio or program level works etc. Few of its important benefits are mentioned below.

Planning flexibility

 A kanban team is only focused on the work that's currently in progress. Once the team completes a work item, they pull the next work item off the top of the backlog. The owner/stakeholders are free to re-prioritize work in the backlog without interfering with the team. Any changes outside the current work items don't impact the team. As long as we keep the most important work items on top of the backlog, the development team is assured they are delivering maximum value back to the business.

Reduce Road blocks

 Multitasking kills efficiency. That's why a key principle of kanban is to limit the amount of work in progress (WIP). Work-in-progress limits help visualize bottlenecks. And the team unitedly jump into resolving the road blocks to get the flow enabled.

Visual Metrics

 Kanban System is known for its visual workflow, so the metrics like Cycle time, Throughput etc, gives end-to-end transparency of the current flow, performance, improvement opportunity to act on. We will talk about its different metrics in our later in this page.

Shortened cycle times

 Cycle time is one of the key metrics for kanban teams. Cycle time is the duration of time a story/work unit takes to travel through the team's workflow—from the moment work starts to the moment it is finished. By optimizing cycle time, the team can confidently forecast the delivery of future work.

Continuous delivery

 CD is the practice of producing work results to customers frequently—even daily or hourly. Kanban and Continuous Delivery complement each other because both techniques focus on the just-in-time delivery of value.

Kanban Metrics

Cycle Time

When your stories or work items travels on kanban flow, from first stage to last stage, its flows through multiple stage. Its stays with in one stage for a longer time or Shorter time.

We calculate the cycle time of a story by calculating the time between its entered the first WIP column. and leave the last WIP column in control (UAT is treated as Out of control WIP). It should be calculated based on the team spent actually working on this item, not how much time it was on the board.

In this example on right, the total time a story or work item spent in Development and Testing is the cycle time of the story.

Lead Time

This is similar to cycle time, but only difference is, its get calculated from when the work-item/story was created, or requested by the client and the time when it was delivered to the client. This time is also called as Customer lead time.

From the example on right, the lead time is getting calculated as the time between a story enters into the board and leave the UAT column (Done means delivered to Customer).

Response Time

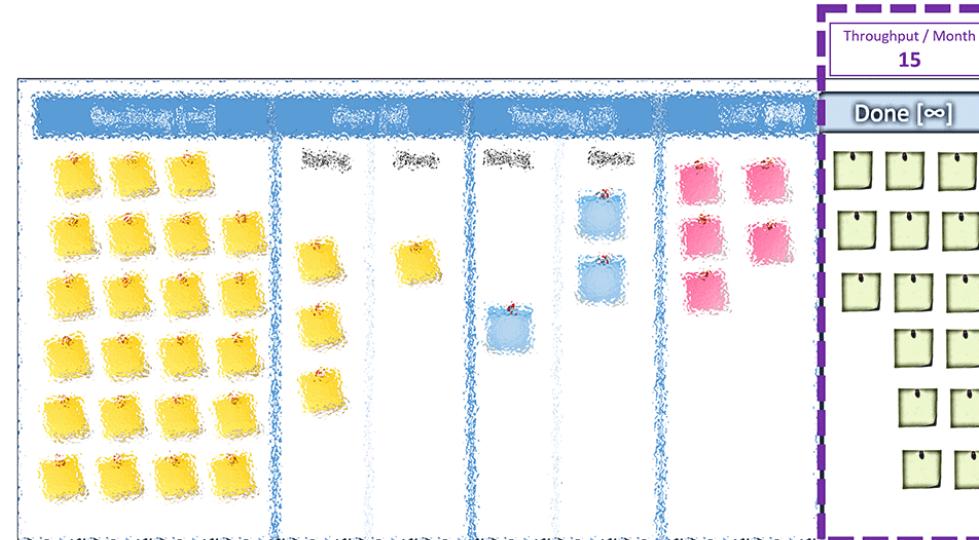
This is again calculated as the time a story/work item was waited on todo list. The time between it was created and got the first Response by moving it to in Progress.

Throughput

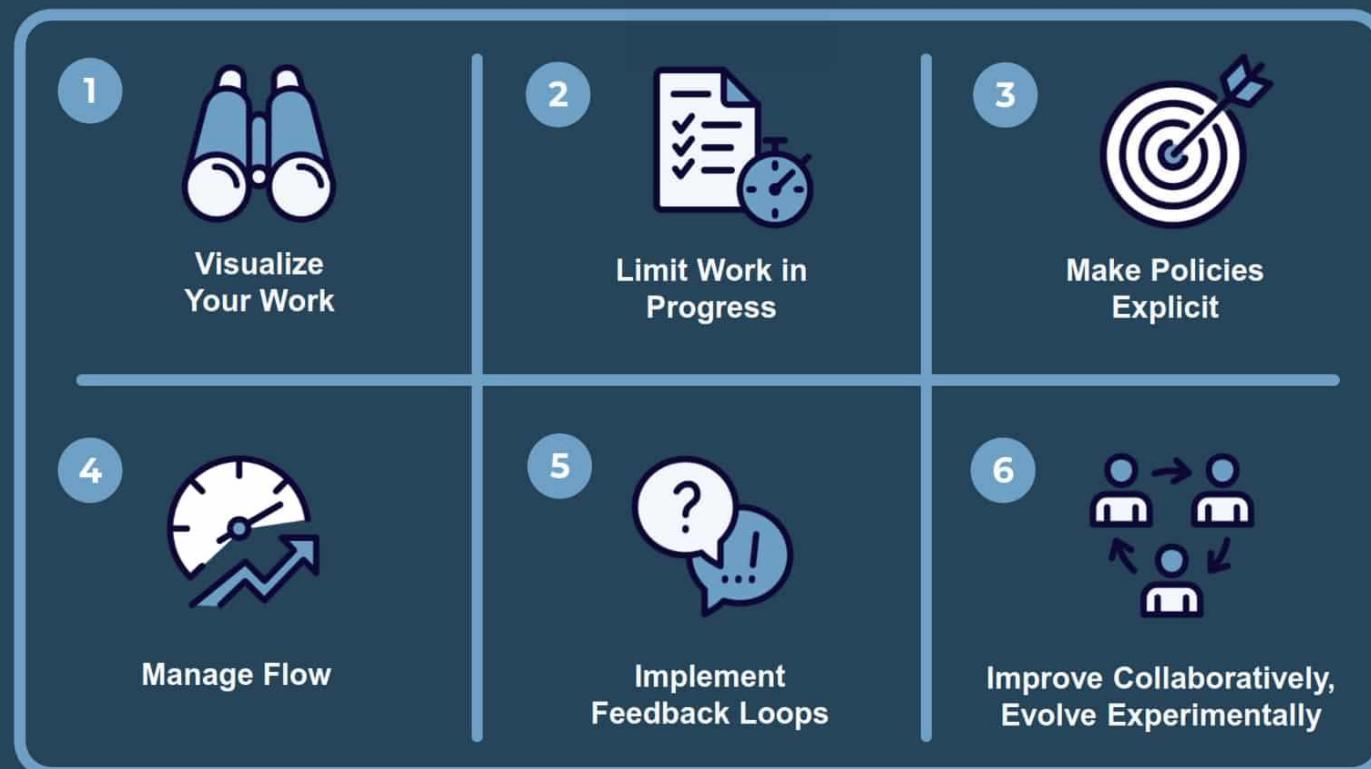
Throughput is not that known measurement metrics. This metrics is number of stories completed in a given time frame, Week, Month etc.

Completed means the stories/work-item completed or delivered to client. Its a real data, not on assumption or prediction.

Throughput is the number (count) of stories or work items that the team is capable to deliver in a given time period, e.g. week, month, provided that they keep a bearable work load.



6 GENERAL PRACTICES OF KANBAN:



1. Visualize the workflow:

- This principle suggests having a Kanban board (physical or digital) to visualize the workflow. Each individual of a team must see his card and cards of other team members. You can move your cards in different columns as per your requirement. It brings lots of transparency within the team and also makes it easier to resolve blockers

2. Limit WIP (Work in Progress):

- Limiting work-in-progress (WIP) is fundamental to implementing Kanban – a ‘Pull-system’. Setting maximum items per stage ensures that a card is only “pulled” into the next step when there is available capacity. Such constraints will quickly illuminate problem areas in your flow so you can identify and resolve them. By limiting WIP, you encourage your team to complete work at hand first before taking up new work.
- Thus, work currently in progress must be completed and marked done. This creates capacity in the system, so new work can be pulled in by the team. Initially, it may not be easy to decide what your WIP limits should be. In fact, you may start with no WIP limits.

3. Make Process Policies Explicit:

- Policies can be established in a team to reduce the rework and focus on the areas which require attention or where it is more effective.
- It is important for the project team to know what they are trying to achieve. The reason behind is quite simple, when someone has a clear goal in front of them, they'll try harder to achieve it
- By formulating explicit process guidelines, you create a common basis for all participants to understand how to do any type of work in the system. They can be a checklist of steps to be done for each work item-type, entry-exit criteria for each column, or anything at all that helps team members manage the flow of work on the board well. Examples of explicit policies include the definition of when a task is completed, the description of individual lanes or columns, who pulls when.

4. Managing the flow

- One of the main goals when implementing a Kanban system is to create a smooth, healthy work flow
- Managing the flow is about managing the work but not the people. By flow, we mean the movement of work items through the production process at a predictable and sustainable pace.
- Instead of micro-managing people and trying to keep them busy all the time, you should focus on managing the work processes and understanding how to get that work faster through the system.
- A Kanban system helps you manage flow by highlighting the various stages of the workflow and the status of work in each stage. Depending on how well the workflow is defined and WIP Limits are set, you will observe either a smooth flow within .
- If there are interruptions or blockers, they must be fixed permanently. and make adjustments to improve flow so as to reduce the time it takes to complete each piece of work

5.implement Feedback Loops:

- just delivering fast is not everything; **delivering the right things is also important**. To know this you need to know what the customers, the end users think, and how well the product contributes to your company's revenue and wellbeing. The need for getting feedback from people outside of your system. There is also a need for feedback loops within a system to make sure you deliver the expected functionality with the right quality. Here is where different kinds of tests come into the picture. Automated tests that run continuously are preferred since they make feedback loops shorter.
- In Kanban, feedback is formalised and implemented not just within the team, but also between teams and between team members and their coach or manager.
- make sure **you get sufficient feedback from them on a regular basis**. This feedback will not only help you optimize your system but also your product quality and the value you deliver.

6.Improve Collaboratively, Evolve Experimentally

- This is the core principle of the Kanban system. It states that you can always improve the process, and that will result in better efficiency.
- Any **improvements you make to your system have to be done collaboratively**. This is where your feedback loops come in handy.
- You **want to make changes to the system with your entire team. The team owns the system and has to work with it, so they have to support it**. If only part of the team supports your system, it is less likely people will stick to its flow and its policies. The **system has to be owned by the entire team**.
- The Kanban Method is an evolutionary improvement process. It **helps you adopt small changes and improve gradually at a pace and size** that your team can handle easily.

DevOps Workflow

<https://www.devopsschool.com/blog/what-is-the-devops-workflow/>

Unit-2

Introduction to project management:

- The need for source code control,
- the history of source code management,

Git - A version control tool:

- Version Control System and Types - CVCS and DVCS.

Git Essentials:

- Creating repository,
- Cloning,
- check-in and committing,
- Fetch pull and remote, Branching

What is source control?

- Source control (or version control) is the practice of tracking and managing changes to code.
- Source control management (SCM) systems provide a running history of code development and help to resolve conflicts when merging contributions from multiple sources.

A BRIEF HISTORY OF

VERSION CONTROL

Early 1960s

IEBUPDTE for IBM OS/360
(punched card system)

1982

RCS (Revision Control System) created
by Walter Tichy

2000

Subversion (SVN) created by
CollabNet

1972

SCCS (Source Code Control
System) created by Marc
Rochkind

1986

CVS (Concurrent Versions Systems)
created by Dick Grune

2005

Git created by Linus Torvalds

What Is Source Code Management?

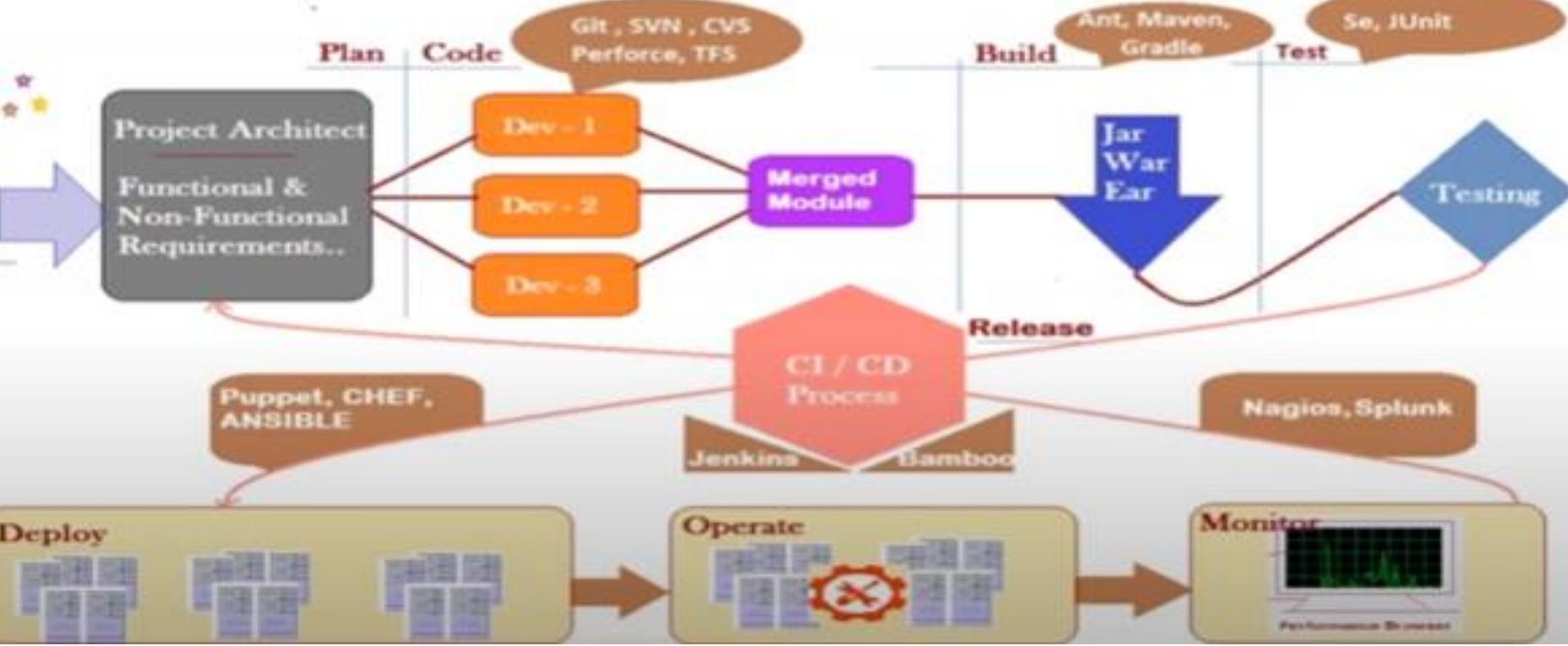
- ❖ Source code management is the concept of storing the source code for your software applications in a centralized, managed place. It's used by software development teams, who are the ones that gain the most benefit from it due to the multiple developers they have.
- ❖ The main people involved in source code management are the developers or programmers. They are the ones that are working on the code, making changes, adding features and requirements. The source code is a product of their work and it needs to be managed in some way.
- ❖ Testers are also main users of source code. When they are required to test applications or features, they benefit from a source code management system and process, as it ensures they are working with accurate code and helps the team run better.

What Are The Benefits Of Source Code Management?

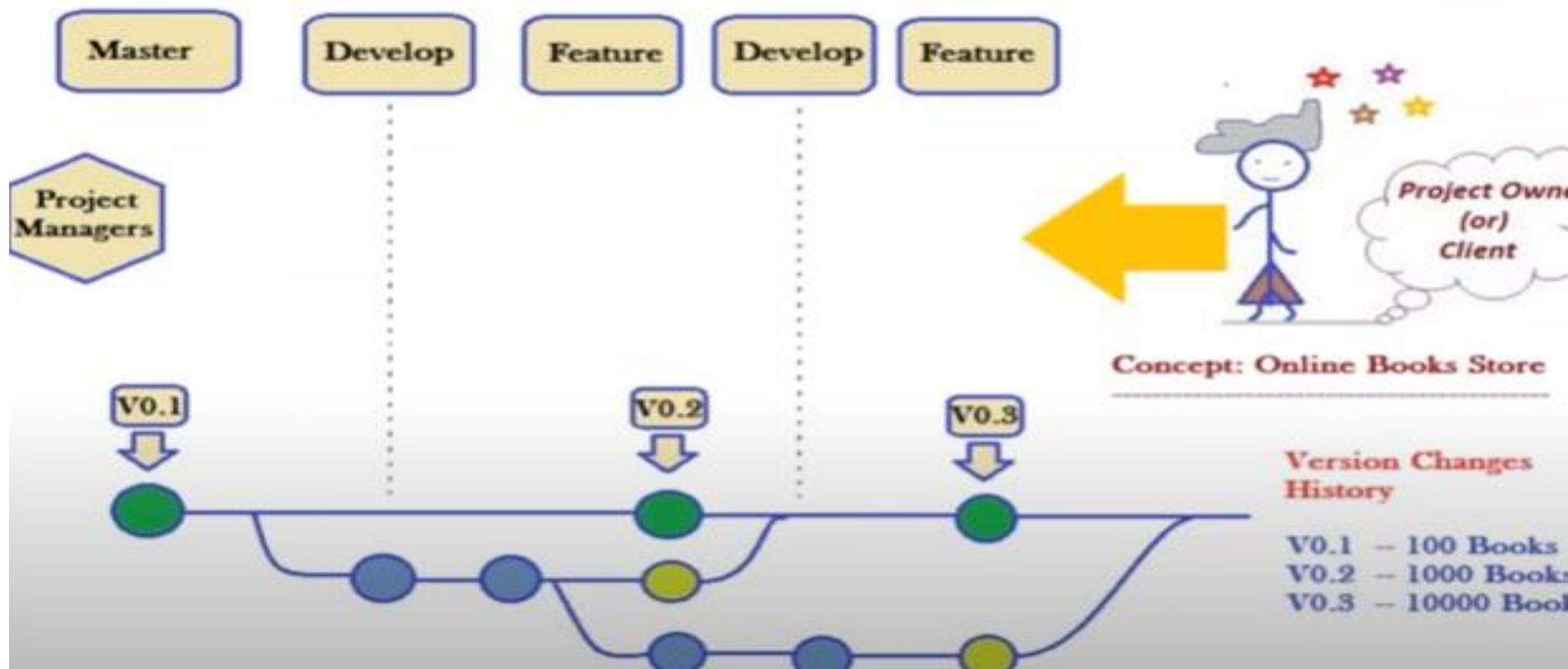
There are several reasons why we should manage the source code that we write:

- **Work together in teams.** A source code management system allows multiple developers to access and change different areas of the code, without interfering with each other's work. This is a big benefit for large teams of developers.
- **Version history.** The system will keep a history of all saved changes to the code. This not only allows you to see what has changed in a file, but allows you to go back to a previous version if needed.
- **Generate release notes.** Code can be linked to a release, and as a result, release notes can be generated. This means time is saved in generating it manually and searching for the changes.
- **Backup of code.** A centralized place on a server where the source code is stored ensures there is a main location for the code (other than the developer's machine). It also allows for easy backup of the code, depending on your server setup.

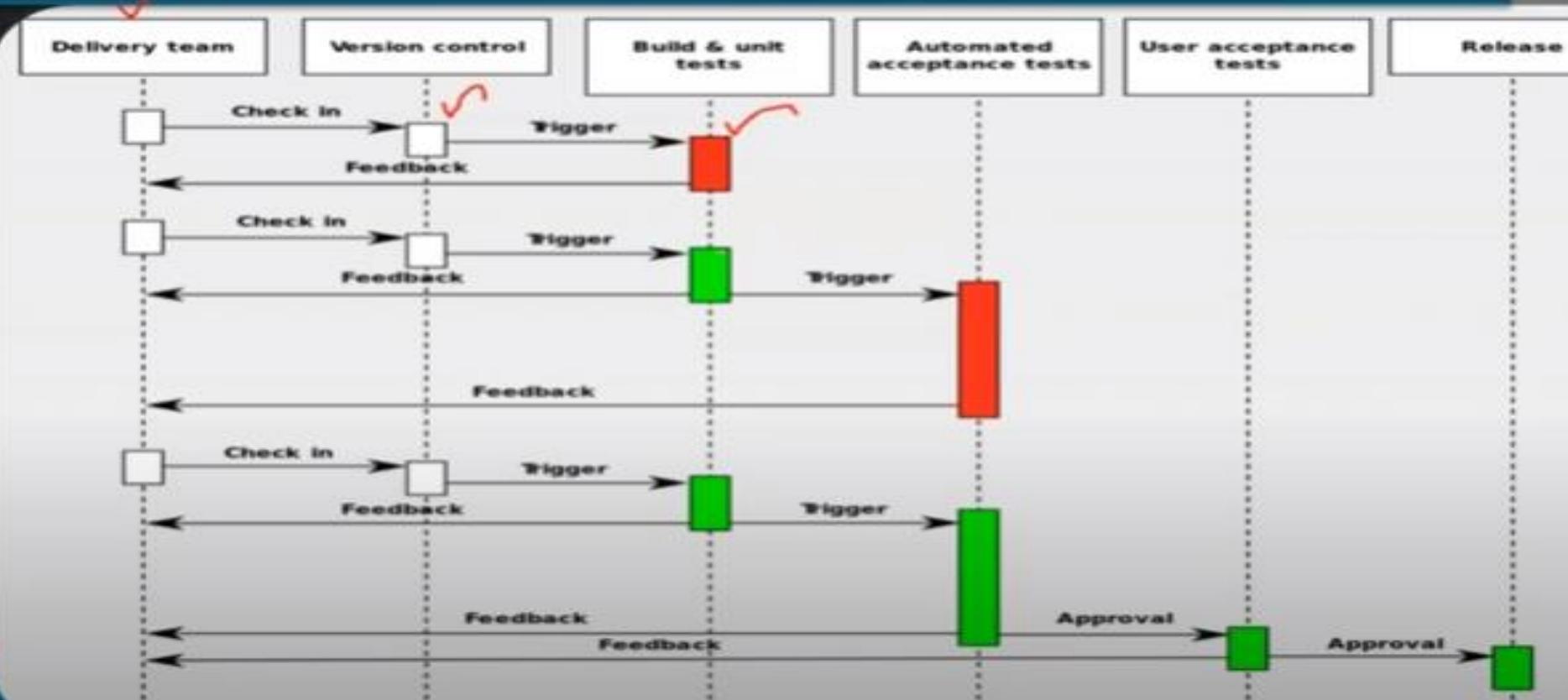
DevOps Tools



What is Version Control System ?



Version Control System Flow? ✓



How Does Source Code Management Work?

The process works by a development team or company having a centralized server that stores all of the required code. This server is called the **repository** – the storage place for code. It can run various source code management systems, which are used to manage changes and different projects.

The system stores a copy of the source code as individual files, just like the developers have written it. When you want to make changes to some code, you perform what's called a “check out” on the file. You get your own copy of the file in your environment. It also creates a “lock” on the file on the server, which means others can't check out the same piece of code. This is to minimise the chances of someone overwriting your work.

Once you've made the changes, and tested them, you can save them onto the server using a process called “commit” or “check in”. This saves your code onto the server, replacing the existing file, and incrementing the version. It also removes the lock that was placed previously. You can usually add comments to this process, which allows you to specify what was changed.

This can all be done by different developers on different files. The source code management system will store this code and manage the changes that have been made.

Importance of Source Code Management

- **Track Changes** – Changes can be tracked as someone making a change leaves a commit message about it.
- **Synchronization** – The up-to-date codes can be fetched from the repository.
- **Backup and Restore** – Files are saved at any time and restored from the last saved one.
- **Undoing** – You can undo both the last known version and the last one created a long time ago.
- **Branching and Merging** – Changes are made on a branch and after being approved, they can be merged with the master branch.
- **Identifying Conflicts and Preventing Overwrites** – Overwrites are prevented and in case of conflicts, they are identified. SCM notifies the developers so that they can review these conflicts and resolve them.

Source code management strategies



1. Planning: Deciding what features or fixes are to be developed in the next cycle.

2. Coding: Writing code in individual branches created from the main codebase.

3. Building: Compiling code into executable or deployable artifacts.

4. Testing: Automatically testing the built artifacts for errors and issues.



5. Releasing: Staging code for deployment to production.

6. Deploying: Moving the code into production environments.

7. Operating: Ongoing operation of the software in the production environment.

What is a “version control system”? (OR SCM)

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done to the code.

Why Version Control system is so Important?

As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes in some specific kind of functionality/features.

So in order to contribute to the product, they made modifications in the source code(either by adding or removing). A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what change has been made. A separate branch is created for every contributor who made the changes and the changes aren't merged into the original source code unless all are analyzed as soon as the changes are green signaled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.

Use of Version Control System:

- **A repository:** It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.
- **Copy of Work (sometimes called as checkout):** It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.
- **Types of Version Control Systems:**

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems

Local Version Control Systems: It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools.

It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.

The Basic Terminology of Version Control System:

Working Directory:

Where developers are required to create/modify files.

Here version control is not applicable. Here we won't use the work like version-1, version-2 etc

Repository:

Where we have to store files and metadata.

Here version control is applicable.

Here we can talk about versions like version-1, version-2 etc

Commit:

The process of sending files from working directory to the repository.

Checkout:

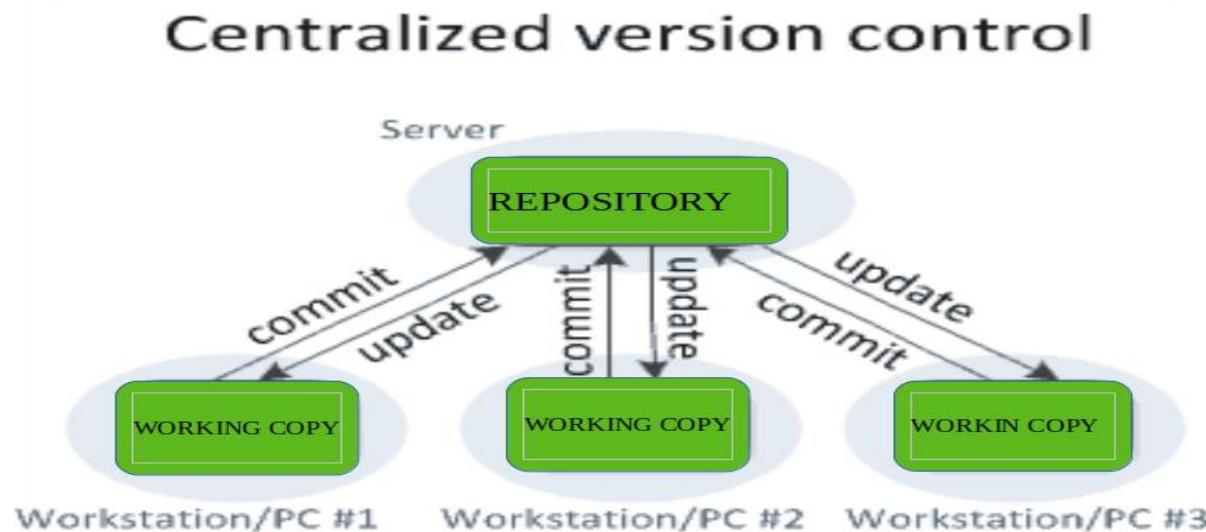
The process of sending files from repository to working directory.

Centralized Version Control Systems: Centralized version control systems contain just one repository and each user gets their own working copy. You need to commit to reflecting your changes in the repository.

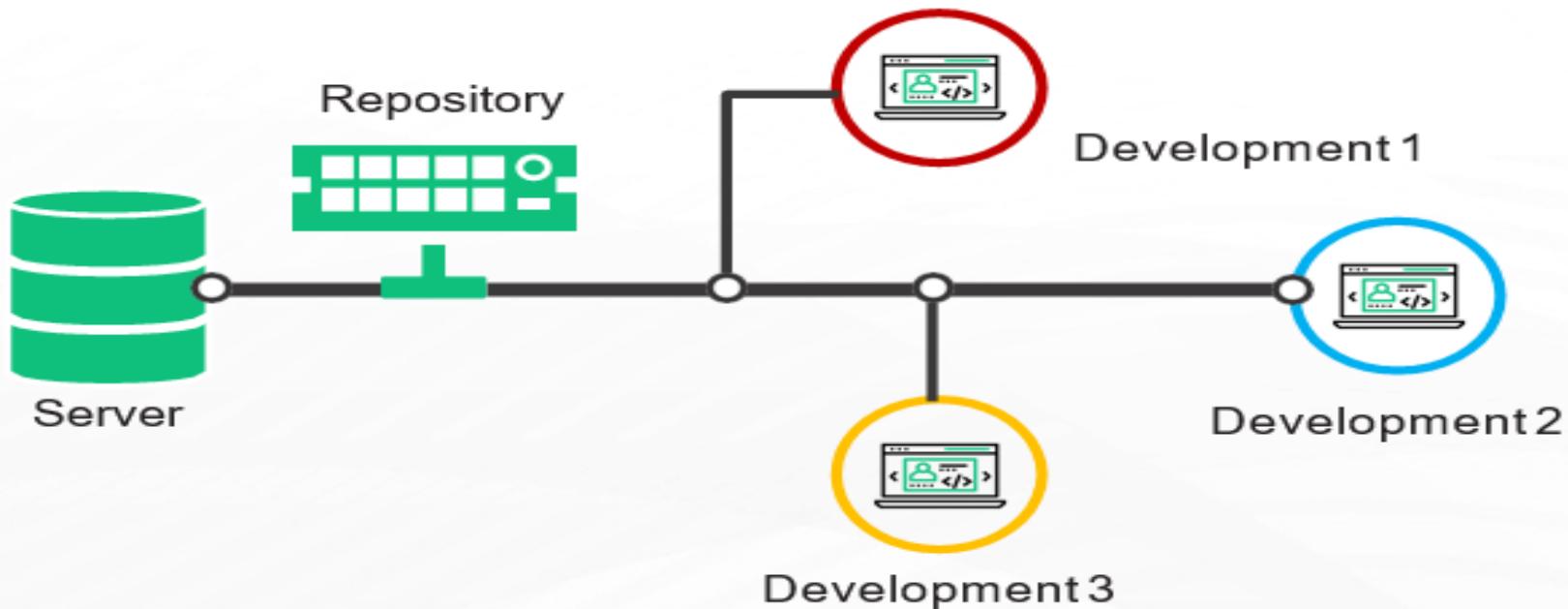
It is possible for others to see your changes by updating.

Two things are required to make your changes visible to others which are:

- You commit
- They update



Centralized Version Control System3



The **benefit** of CVCS (Centralized Version Control Systems) makes collaboration amongst developers along with providing an insight to a certain extent on what everyone else is doing on the project. It allows administrators to fine-grained control over who can do what.

It has some **downsides** as well which led to the development of DVCS. The most obvious is the **single point of failure** that the centralized repository represents if it goes down during that period collaboration and saving versioned changes is not possible.

What if the hard disk of the central database becomes corrupted, and proper backups haven't been kept? You lose absolutely everything.

Problems with Centralized VCSs:

- 1) Central Repository is the only place where everything is stored, which causes **single point of failure**. If something goes wrong to the central repository then recovery is very difficult.
- 2) All commit and checkout operations should be performed by connecting to the central repository via network. If **network outage**, then no version control to the developer. i.e in this type, developer work space and remote repository server should be connected always.
- 3) All commit and checkout operations should be performed by connecting to the central repository via network and hence these operations will become slow, which causes **performance issues**. No local operations and every version control operation should be remote operation.
- 4) **Organization of central repository is very complex** if number of developers and files increases.
etc

Distributed Version Control Systems: Distributed version control systems contain multiple repositories. Each user has their own repository and working copy. Just committing your changes will not give others access to your changes.

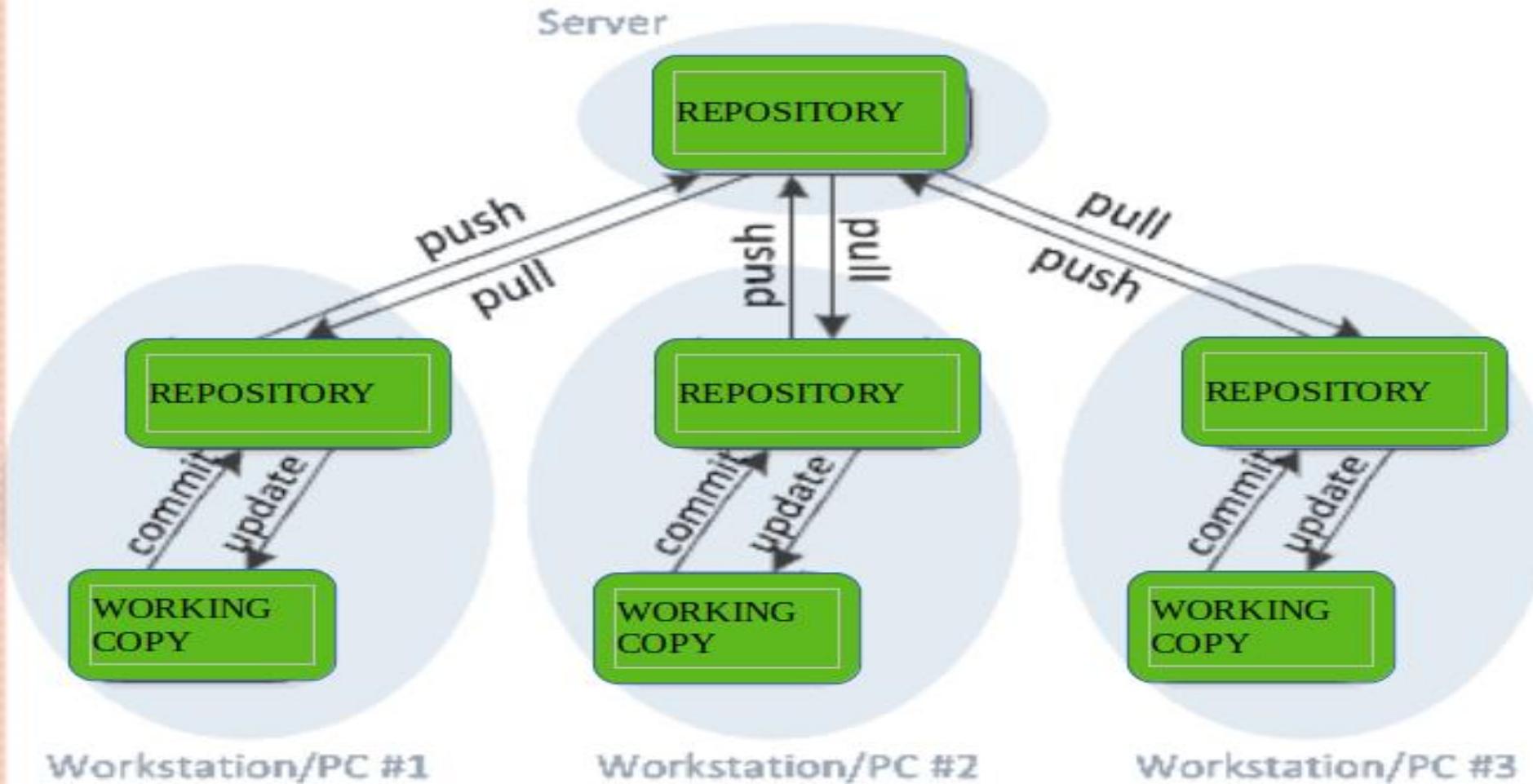
This is because commit will reflect those changes in your local repository and you need to push them in order to make them visible on the central repository. Similarly, When you update, you do not get other's changes unless you have first pulled those changes into your repository.

To make your changes visible to others, 4 things are required:

- You commit
- You push
- They pull
- They update

The most popular distributed version control systems are Git, Mercurial. They help us overcome the problem of single point of failure.

Distributed version control



Note:

1) commit and checkout operations will be performed between workspace and repository.

work space(Local) – **commit (checkin)**- Repository(Remote server) //***Push based***
Repository – checkout - workspace // ***pull based***

2) push and pull operations will be performed between repositories.

one repository ---push - other repository

one repository - pull----other repository

Benefits of Version Control System:

We can maintain different versions and we can choose any version based on client requirement.

With every version/commit we can maintain metadata like

- commit message
- who did changes
- when he did the change
- what changes he did

3) Developers can share the code to the peer developers in very easy way.

4) Multiple developers can work in collaborative way

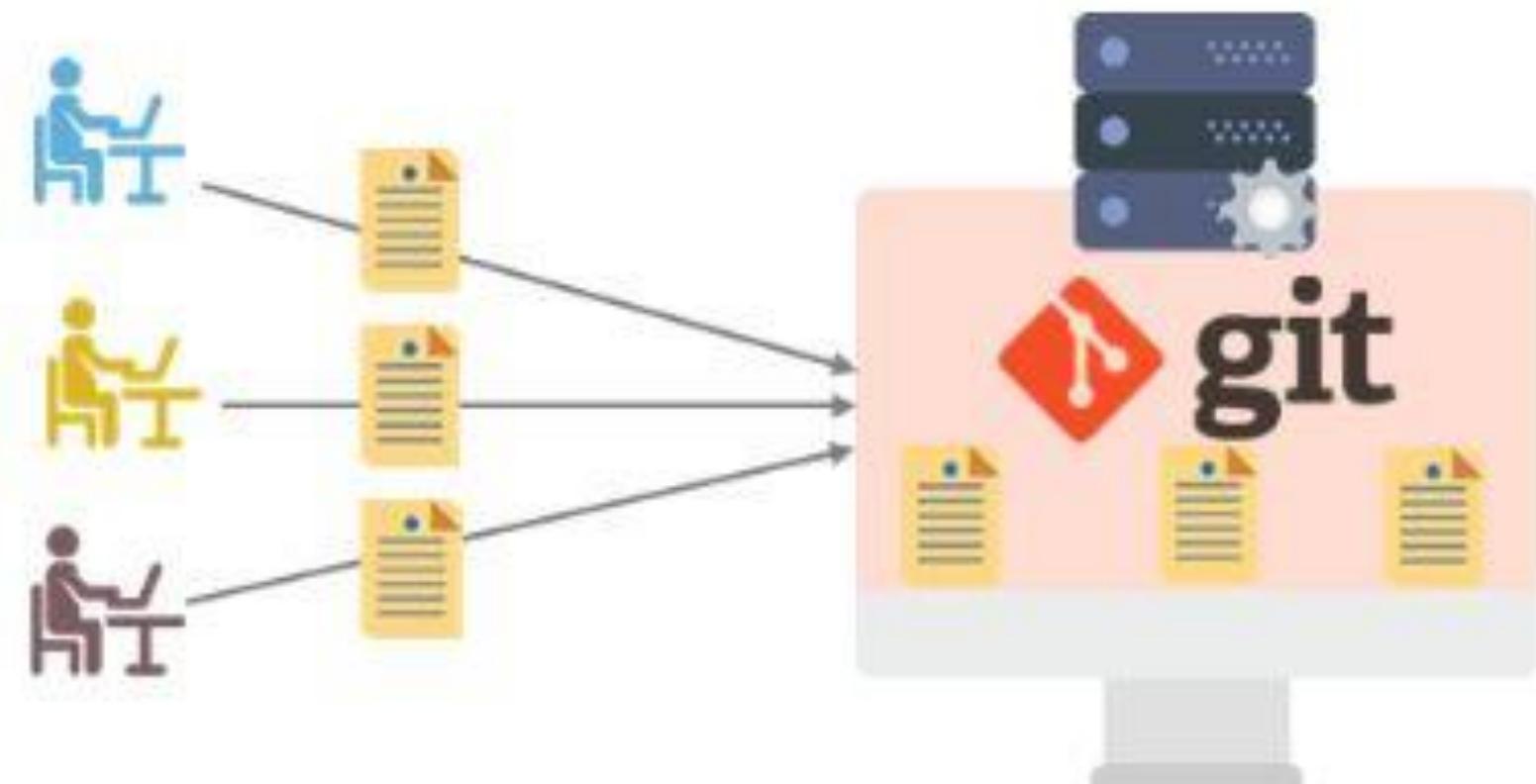
5) Parallel development.

6) We can provide access control like

- who can read code
- who can modify code

What is GIT?

- * Git is Distributed Version Control System Tool.
- * Git is not acronym and hence no expansion. But most of the people abbreviated as "Global Information Tracker".
- * GIT is developed by Linus Torvalds(Finnish software engineer), who also developed Linux Kernel.
- * Most of the companies like Microsoft, Facebook, Yahoo, LinkedIn, Intel using Git as Version Control System Tool.



Developers

Git Server

Features of Git:

Git is very popular because of the following features:

- Tracks history
- Free and open source
- Supports non-linear development
- Creates backups
- Scalable
- Supports collaboration
- Branching is easier
- Distributed development

Git is very popular because of the following features:

Distributed

Git is developed based on Distributed Version Control System Architecture.

Because of Distributed Architecture it has several advantages:

- A) Every Developer has his own local repository. All the operations can be performed locally. Hence local repo and remote repo need not be connected always.
- B) All operations will be performed locally, and hence performance is high when compared with other VCSs.
i.e. it is very speed
- C) Most of operations are local. Hence we can work offline most of the times.
- D) There is no single point failure as Every Developer has his own local repository.
- E) It enables parallel development & automatic-backups.

2) Staging Area:

It is also known as index area.

There is logical layer/virtual layer in git between working directory and local repository.

Working Directory

- Staging Area
- Local Repository

We cannot commit the files of working directory directly. First we have to add to the staging area and then we have to commit.

This staging area is helpful to double check/cross-check our changes before commit.

This type of layer is not available in other Version Control System Tools like CVS, SVN etc.

Git stores files in repository in some hash form, which saves space.

GIT will use internally snapshot mechanism for this. All these conversions and taking snapshots of our data will be happened in staging area before commit.

Eg: If a sample repository takes around 12 GB space in SVN where as in GIT it takes hardly 420 MB.

Hence Staging Area is the most important Strength of GIT.

3) Branching and Merging:

We can create and work on multiple branches simultaneously and all these branches are isolated from each other.

It enables multiple work flows.

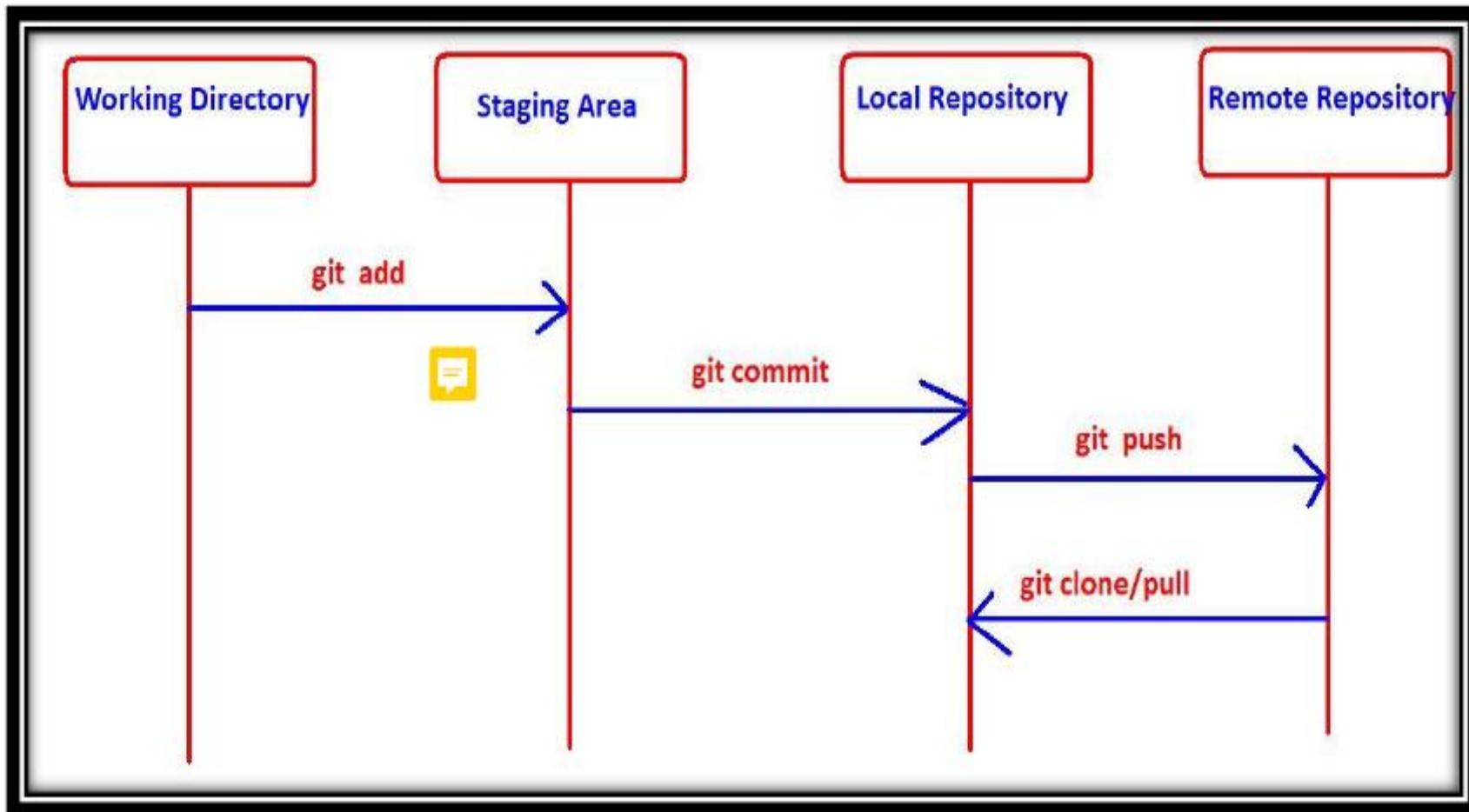
We can merge multiple branches into a single branch. We can commit branch wise also.

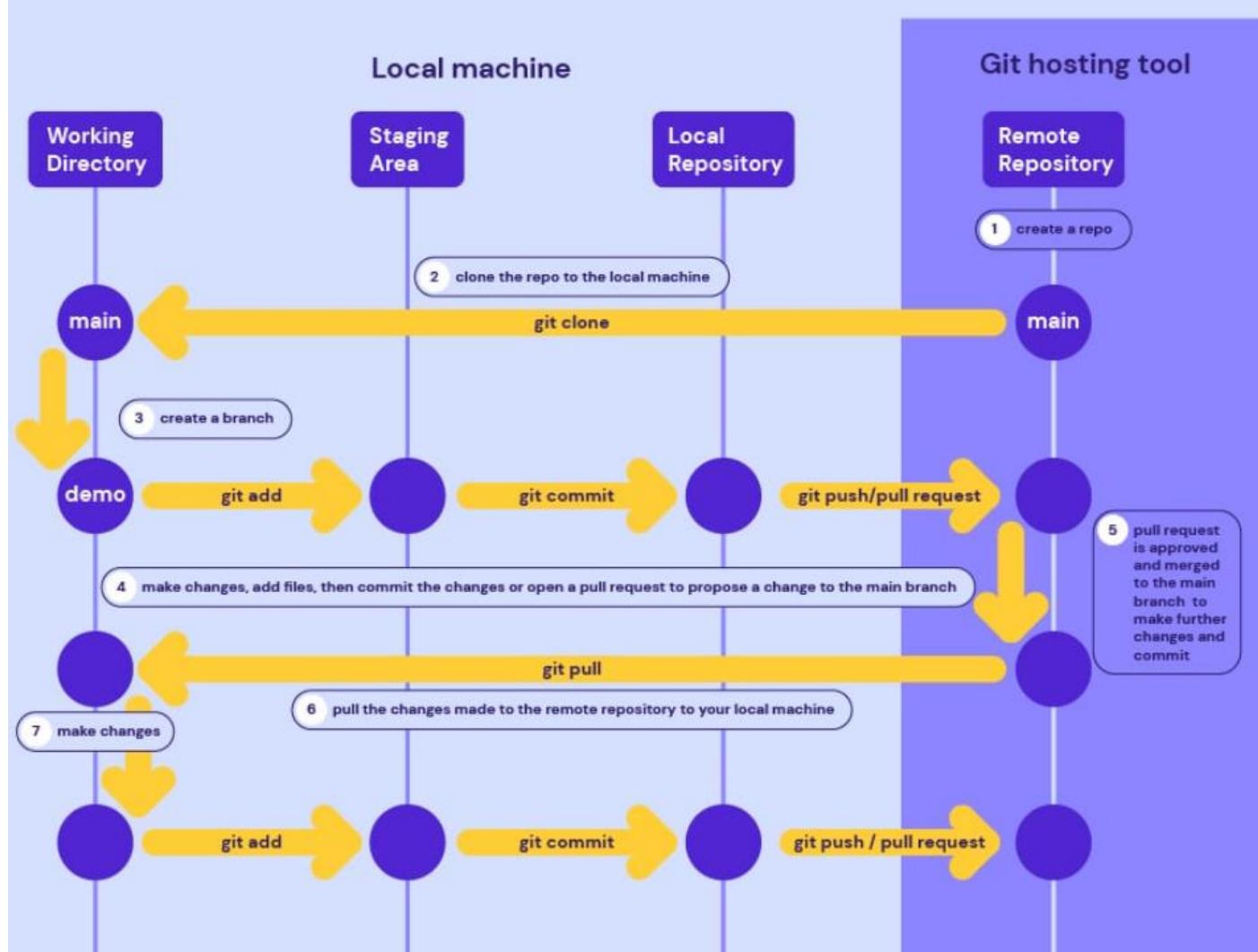
4. Moving files in GIT is very easy as GIT automatically tracks the moves. Whereas in other VCS we need to create a new file & then delete the old one.

5. Freeware and Open Source

6. It provides support for multiple platforms.

GIT Architecture:





Most of the version control systems have a two-tier architecture. However, Git has a layer more, making it a three-tier architecture. But, why are there three layers of Git? Let's find out.

The three layers are:

- **Working directory:** This is created when a Git project is initialized onto your local machine and allows you to edit the source code copied.
- **Staging area:** Post the edits, the code is staged in the staging area by applying the command, **git add**. This displays a preview for the next stage. In case further modifications are made in the working directory, the snapshots for these two layers will be different. However, these can be synced by using the same 'git add' command.
- **Local repository:** If no further edits are required to be done, then you can go ahead and apply the **git commit** command. This replicates the latest snapshots in all three stages, making them in sync with each other.

After reading about the architecture and getting a better knowledge of Git, this 'What is Git?' blog will help you understand how you can make use of Git.

How to use Git?

Much of Git's popularity is attributed to its user-friendly nature. You can check out these simple concepts to perform certain basic operations and get acquainted with [Git basics](#).

- **Commit:** This is an object that takes the current state of the repository.
- **Pull:** This operation copies the changes made to a project to the local repository from the remote one.
- **Push:** This operation copies the changes made to a project to the remote repository from the local one.

Now, let's take a look at some of the basic and commonly used commands in Git.

Now, let's take a look at some of the basic and commonly used commands in Git.

Git Commands

Based on what you work with, be it remote or local repositories, the Git commands change. Let's take a look at the various [commands in Git](#).

Git Commands When Working with Local Repositories

- **git init:** This Git command converts a directory into an empty repository. This is the initial step you need to take to build a repository. Once you run git init, you will be able to add and commit files and directories.
- **git add:** This command allows you to add files in the Git staging area. The file must be added to the index of Git before being available to commit to any particular repository. You can use this command to add directories, files, etc.
- **git commit:** The commit command in Git allows you to track the changes in the files in a local repository. Each commit has its own unique ID for reference.
- **git status:** The git status command returns the present state of a repository, like if the file is in the staging area but has not been committed.

- **git config:** There are numerous configurations and settings possible in Git, and this command allows you to assign these settings. User.name and user.Email are the two significant settings that set the name and email address of a user.
- **git branch:** This command determines the branch of the local repository and allows you to add or delete a branch.
- **git checkout:** You can use this command to switch to another branch.
- **git merge:** The merge command allows you to integrate two or more branches together. It combines the changes made in the branches.

Git Commands When Working with Remote Repositories

1.git remote: This Git command allows you to connect a remote repository to a local repository.

1.git clone: You can use the clone command to create a local copy of an already existing remote repository. This allows you to copy and download the required repository to the system. It is similar to the init command while working with remote repositories as it allows you to build a local directory, consisting of all the necessary files and history of the repository.

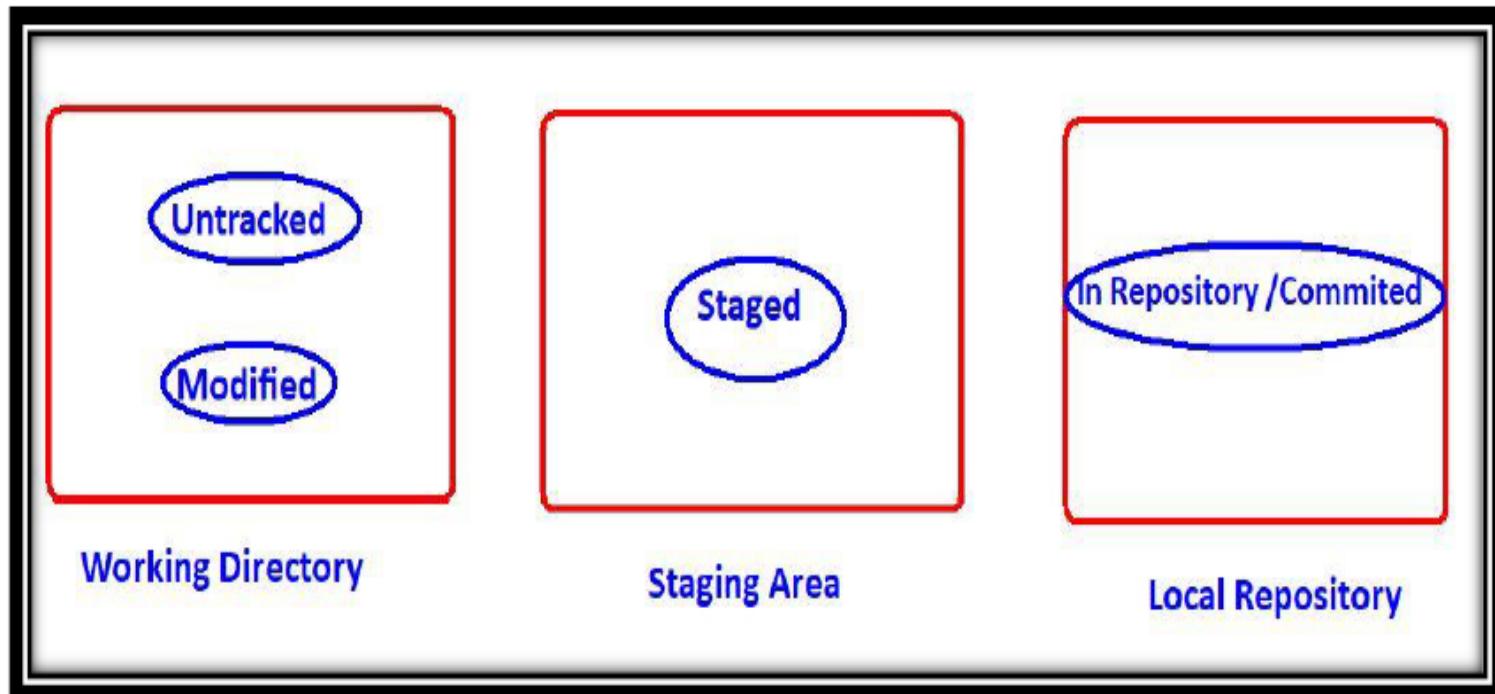
3.git pull: The pull command is used to run the latest version of any repository. This pulls all the changes made from the remote to the local repository.

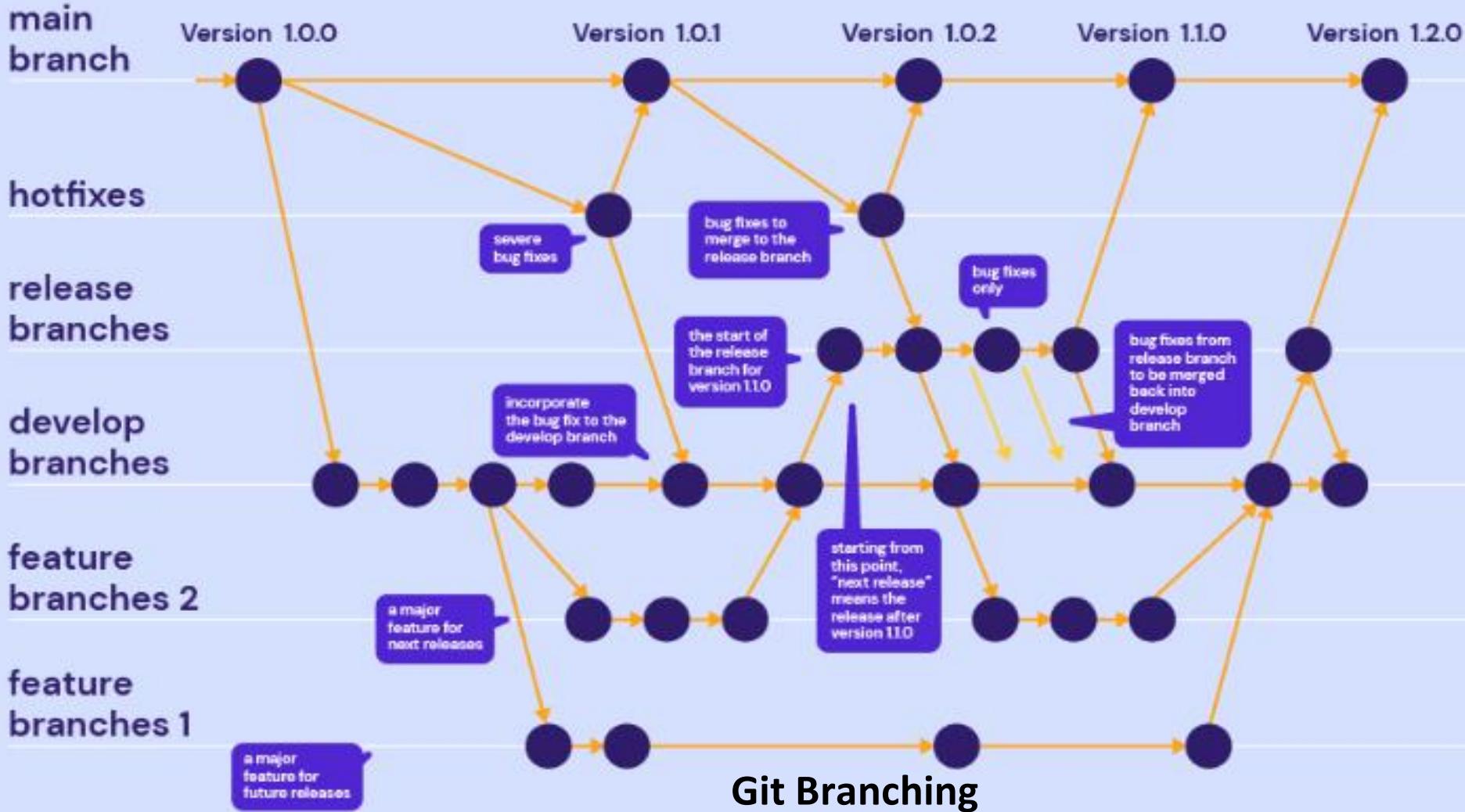
4.git push: This command sends local commits to the respective remote repository. It needs two parameters, i.e., the remote repository and the specific branch where it needs to be pushed.

There are numerous other Git commands that are of more advanced level, such as git stash, git log, git rm, etc.

Now, you will learn about the use of Git in DevOps.

Life Cycle of File in GIT





Unit-3

Continuous Integration



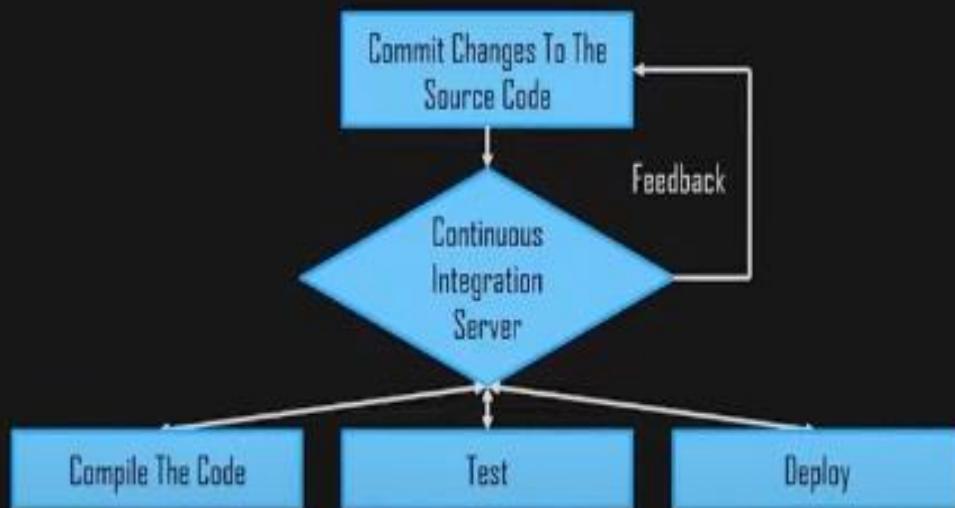
Jenkins

Continuous Integration

- What is Continuous Integration?
- Why do we need it?
- Different phases of adopting Continuous Integration

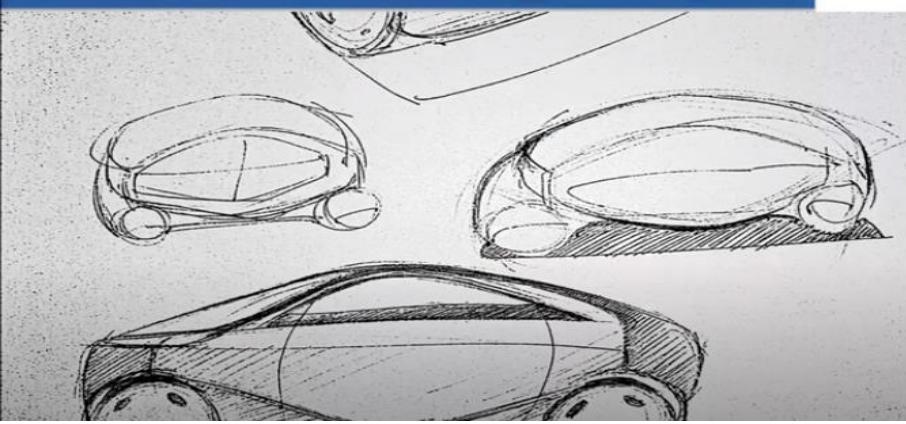


WHAT IS CONTINUOUS INTEGRATION?



Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequent!

Car manufacturing Analysis



Car Parts

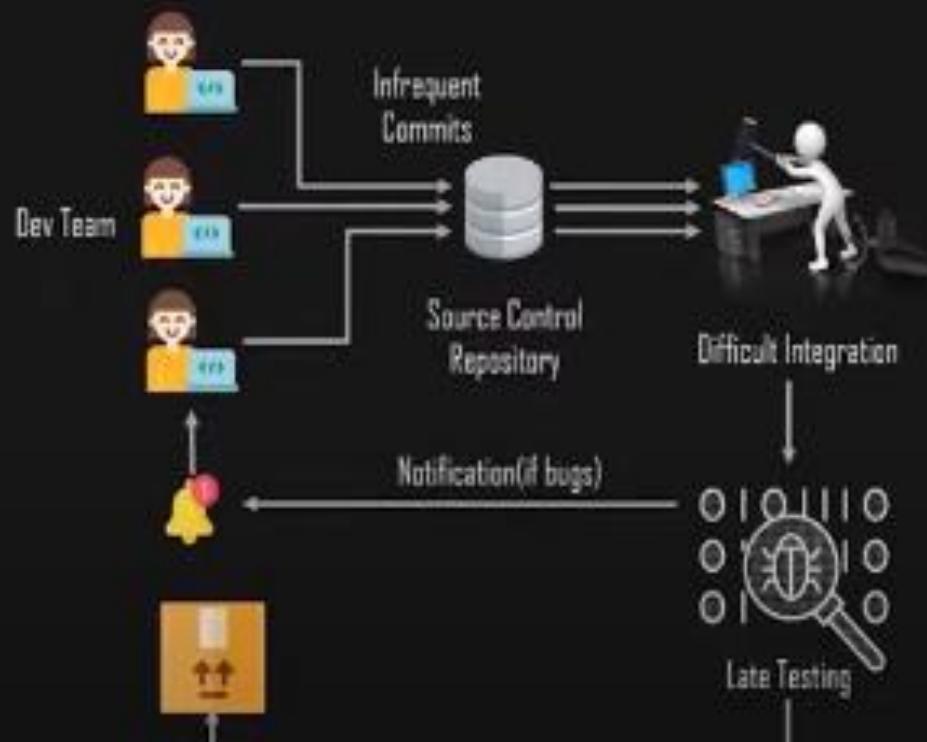


Car Parts Integration



WHY CONTINUOUS INTEGRATION

Before Continuous Integration

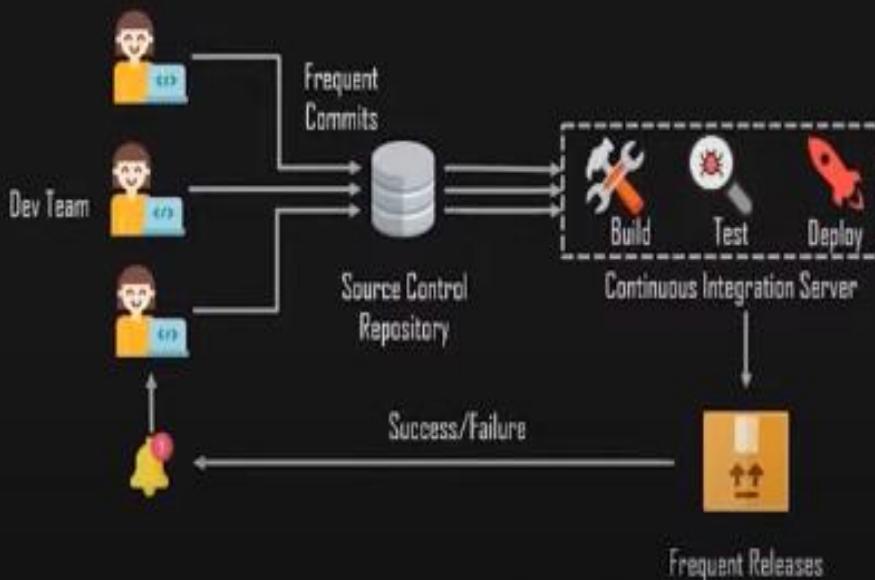


Issues faced before CI:

- ✓ Long waits to test code
- ✓ Difficult to Debug
- ✓ Slow Software Delivery Process
- ✓ Infrequent Feedback Cycles

WHY CONTINUOUS INTEGRATION?

After Continuous Integration



A build server, also called a continuous integration server (CI server), is a **centralized, stable and reliable environment for building distributed development projects**. ... A build server can also speed the development process by freeing up resources on developers' local machines.

Continuous Integration fixes the feedback and time issues!

Continuous Integration Example: Nokia

I am pretty sure you all have used **Nokia** phones at some point in your life. In a software product development project at Nokia, there was a process called **Nightly builds**.

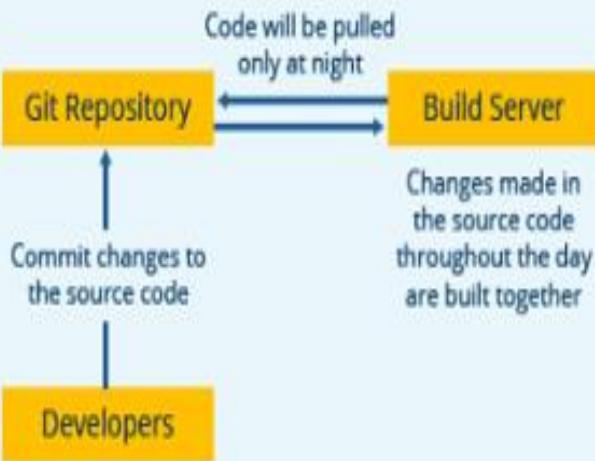
Nightly builds can be thought of as a predecessor to Continuous Integration.

It means that every night an automated system pulls the code added to the shared repository throughout the day and builds that code. The idea is quite similar to Continuous Integration, but since the code that was built at night was quite large, locating and fixing of bugs was a real pain.

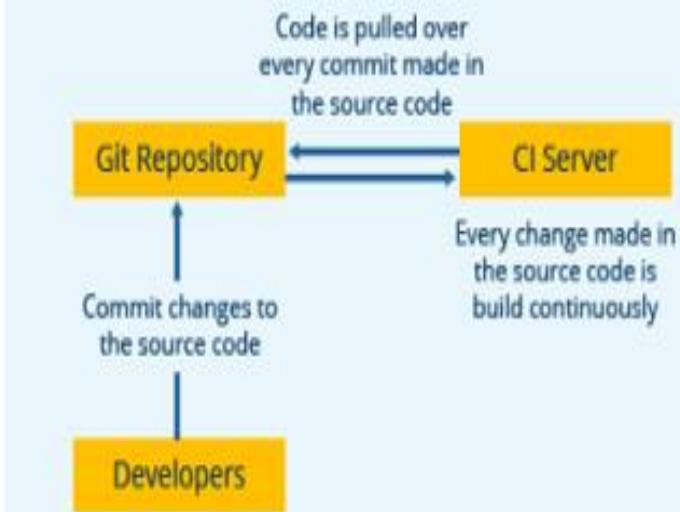
Due to this, Nokia adopted Continuous Integration (CI). As a result, every commit made to the source code in the repository was built. If the build result shows that there is a bug in the code, then the developers only need to check that particular commit. This significantly reduced the time required to release new software.



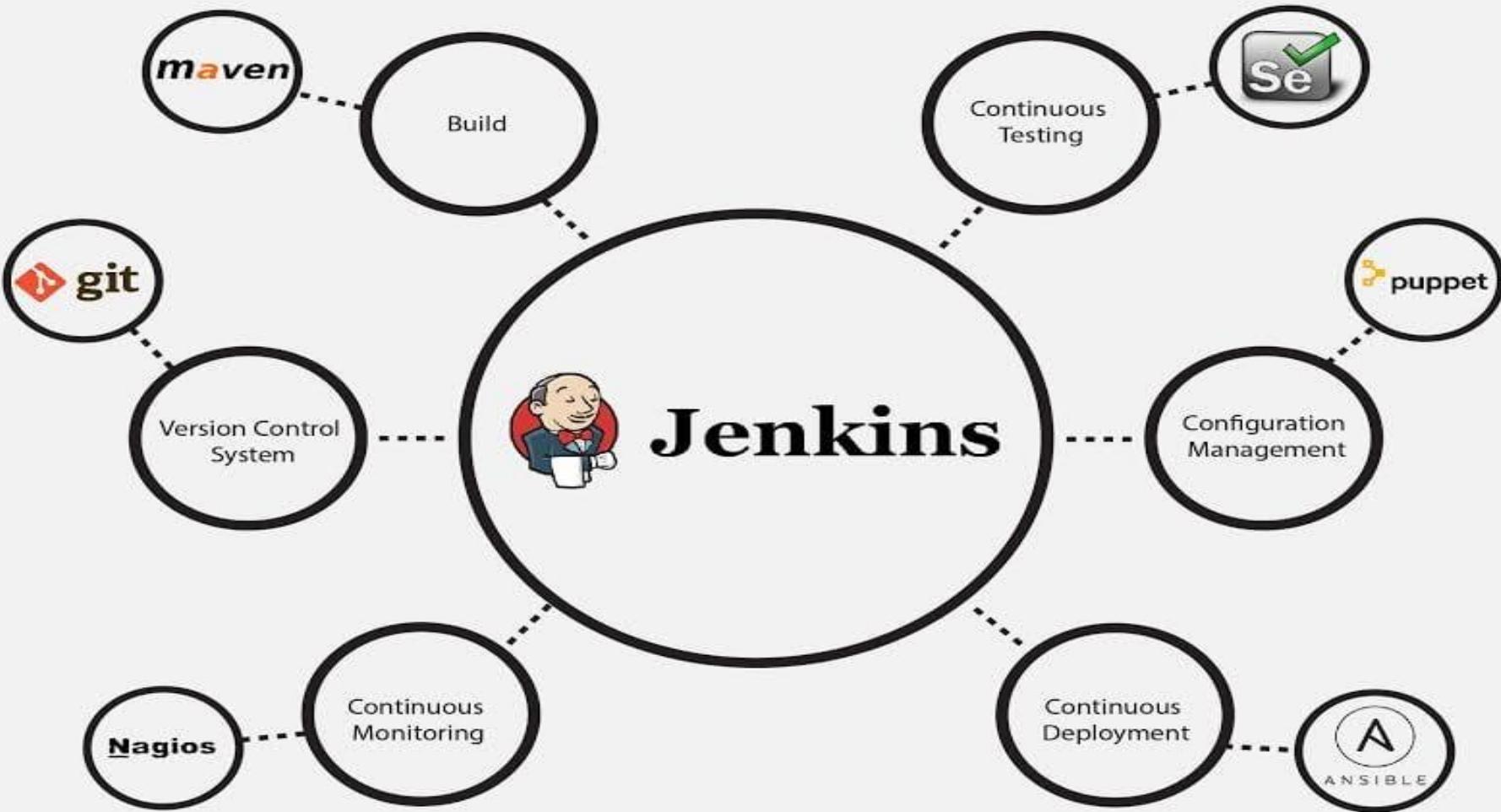
Nightly build



Continuous Integration



Now is the correct time to understand how Jenkins achieves Continuous Integration.



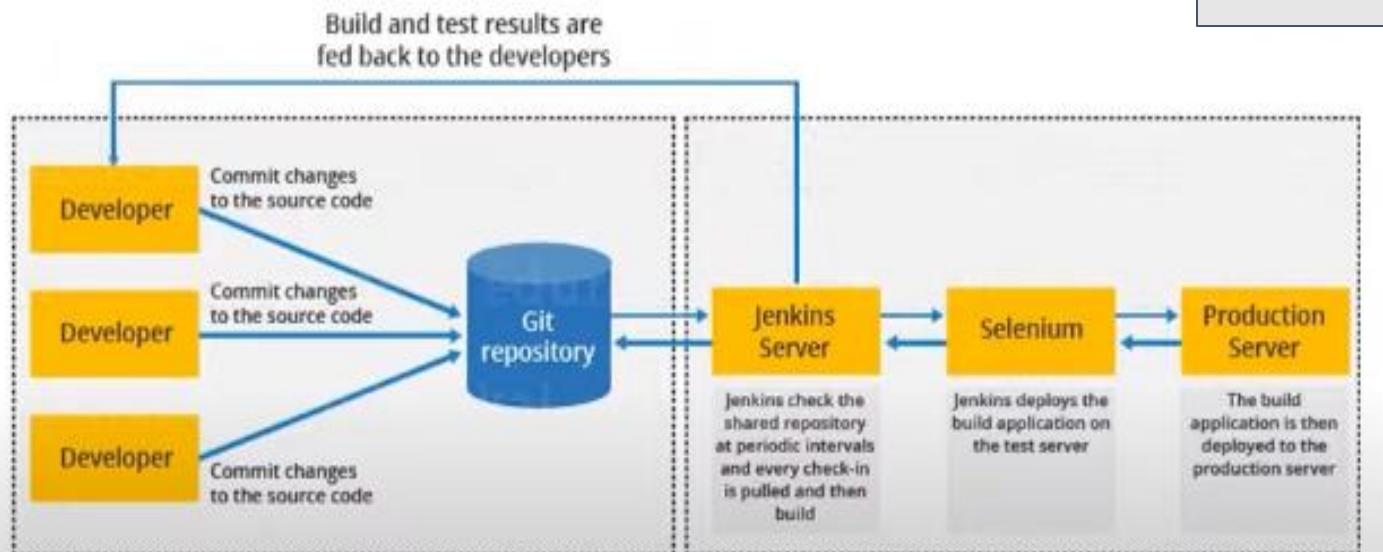
Jenkins Features

The following are some facts about Jenkins that makes it better than other Continuous Integration tools:

- **Adoption:** Jenkins is widespread, with more than 147,000 active installations and over 1 million users around the world.
- **Plugins:** Jenkins is interconnected with well over 1,000 plugins that allow it to integrate with most of the development, testing and deployment tools.

It is evident from the above points that Jenkins has a very high demand globally. Before we dive into Jenkins it is important to know what is Continuous Integration and why it was introduced.

Jenkins Architecture



This single Jenkins server was not enough to meet certain requirements like:

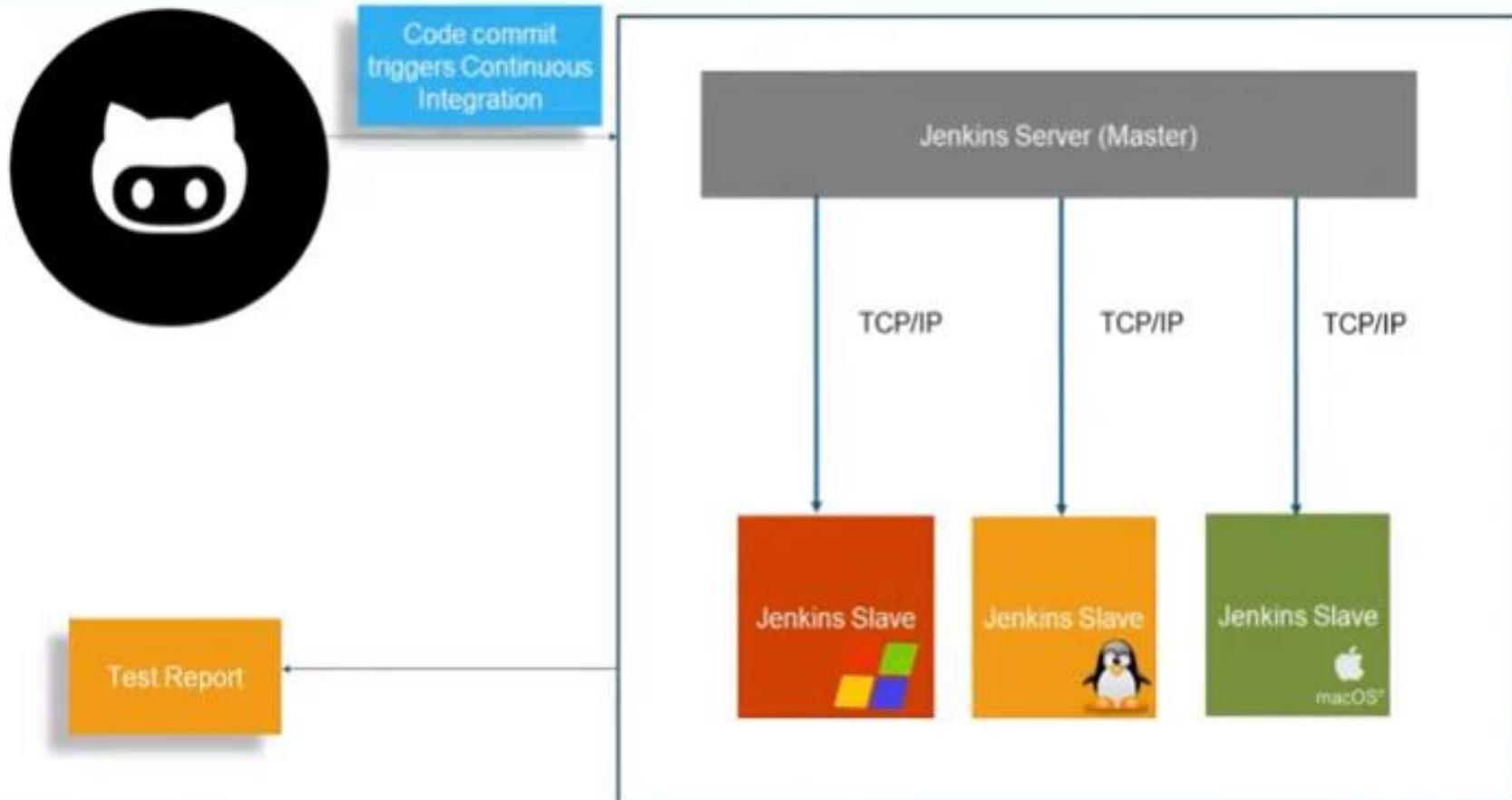
- Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.
- If larger and heavier projects get built on a regular basis then a single Jenkins server cannot simply handle the entire load.

To address the above-stated needs, Jenkins distributed architecture came into the picture.

Jenkins Distributed Architecture

Jenkins uses a Master-Slave architecture to manage distributed builds. In this architecture, Master and Slave communicate through TCP/IP protocol.

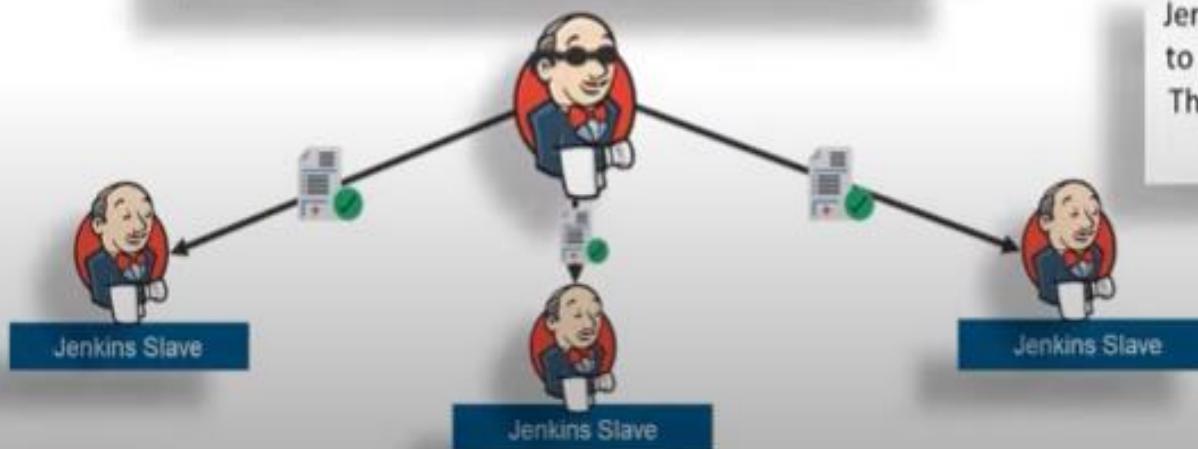
Jenkins Distributed Build Architecture



What is Distributed Jenkins Build

- To run heavy projects which gets built on a regular basis on a single machine is not a good option
- In such a scenario, need to off load load from Master by configuring more slaves
- Distributed builds are used to absorb extra load, or to run specialized build jobs in specific operating systems or environments

Jenkins Master will distribute its workload to the slaves



Jenkins Slaves are generally required to provide the desired environment. They work on the basis of requests received from Jenkins Master

Jenkins Distributed Build Architecture: Master

- To manage distributed builds, Jenkins uses a **Master-Slave** architecture
- Master and Slave communicate through TCP/IP protocol
- Master represents main Jenkins server and it handles all tasks for build system



Jenkins Distributed Build Architecture: Slave

- A slave is a remote machine that is set up to offload build projects from the Master and listens for requests from the Jenkins Master
- Each slave runs a “slave agent”
- Full Jenkins installation on a slave is not required
- There are various ways to start slave agents, but there should be a bi-directional communication between the “slave agent” and “Jenkins Master” in order to operate



Jenkins Scalability

Parallel Build
The ability to run builds in parallel



Load Distribution
Distributing the load across different physical machines while keeping required resources available for each specific build plan

Replacing corrupted instances
Automatically replacing the corrupted Jenkins instances

Cost Effective
Automatically adding and removing slaves based on the need, this saves cost too

Unit-3/Part-2:

- Build Tools-Overview of Maven.
- Virtualization, and Virtualization in DevOps.
- Understand Containers,
- Dockers- A containerization technology.

Scripts

CI/CD Tools

Build Tools

Source Code Management Tools

DevOps

Tools



Team Foundation Server



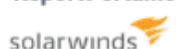
Deploy Tools



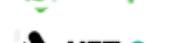
Virtualization & Containerization



Reports, Statistics & Analytics



Test Automation



Static Code Analysis Tools



Why we need build tool?

- Challenges with manual process:

- **Scalability Challenges:** Manual process management becomes more time-consuming & error-prone as the scope of the project expands.
- **Human errors:** Misconfigurations, overlooked stages, and the omission of crucial files
- **Inefficiency:** Doing the same things repeatedly
- **Inconsistency:** Inconsistency arises when different developers use their individual manual techniques to complete a build, making it difficult to work together and keep up with the project.

What is a Build Tool?

Build tools are commonly known as programs that automate the process of building an executable application from source code. This building process includes activities like compiling, linking and packaging the code into an executable form. Build automation involves scripting or automating a wide range of tasks that software developers perform in their daily activities.

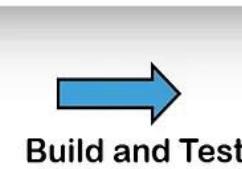
The activities include:

- Downloading the dependencies.
- Compiling source code to the form of binary code.
- Packaging that binary code.
- Running the tests.
- Deploying them to the production systems.

Maven is a widely used build automation and project management tool in the software development ecosystem. Apache Maven is two tools made into one – the Dependency Manager and also a powerful build tool. Like Apache ANT, it is an XML-based build file, but at the same time, it outlines very rigid standards for itself. Here are some key aspects and features of Maven:

- **Build Lifecycle:** Maven defines a standard build lifecycle consisting of phases such as compile, test, package, install, and deploy. These phases facilitate the build process in a standardized way.
- **Project Object Model (POM):** Maven projects are defined by a Project Object Model (POM) file, usually named pom.xml. This XML file contains project configurations, dependencies, plugins, and other information required for building the project.
- **Dependency Management:** Maven simplifies dependency management by allowing you to declare dependencies in the POM file. It can automatically download the required dependencies from repositories and manage their versions.
- **Plugin Architecture:** Maven uses a plugin-based architecture to extend its functionality. Various plugins are available for tasks such as compiling code, running tests, creating JARs, deploying artifacts, and more.

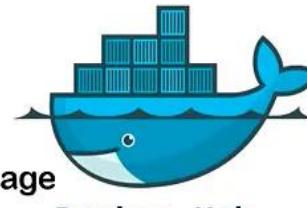
- **Convention Over Configuration:** Maven follows the principle of “convention over configuration,” which means that a project adhering to standard conventions requires minimal configuration. This standardization simplifies the build process.
- **Centralized Repository:** Maven Central Repository serves as a centralized repository for sharing and downloading dependencies. Additionally, organizations can set up their repositories to manage internal dependencies.
- **Goals and Phases:** Maven build process is organized into goals and phases. A goal represents a specific task (e.g., compile, test), while a phase is a collection of goals executed in sequence.
- **Transitive Dependency Resolution:** Maven supports transitive dependency resolution, meaning that if a project depends on a library, Maven will automatically download that library’s dependencies.
- **Integration with IDEs:** Maven integrates seamlessly with Integrated Development Environments (IDEs) such as Eclipse, IntelliJ IDEA, and NetBeans, making it easy for developers to work on Maven-based projects.



Manifest File Repo

Update Build Number

Scan Docker Image



Build & Push Docker Image



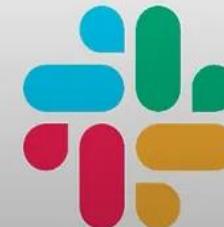
Static Code Analysis



Deploy on EKS



Send Notification



[Image source\(slide 157\)](#)

[Datasource\(slide 154-156\)](#)

Virtualization is the technique by which teams can generate a useful virtual or simulated version instance of the software on a single server, simulating its varied hardware and software settings. These resources are then partitioned into several virtual systems known as Virtual Machines, which are emulations of hardware and software setups. Even though they run on a piece of the physical device, each virtual machine has its operating system and acts as separate software.[Animated video](#)

Types:

**Application
Virtualization**

**Network
Virtualization**

**Desktop
virtualization**

**Storage
Virtualization**

**Server
Virtualization**

**Data
Virtualization**

Virtualization in DevOps

Virtualization can increase scalability while reducing costs. Some of the benefits that enabling virtualization can bring to an organization are:

- **Cost Savings** – When you enable virtualization, a single physical server is replaced by multiple virtual machines thereby reducing idle compute time and promoting optimum utilization of resources.
- **Downtime Reduction and Flexibility during Disaster Recovery** – When a physical server is affected, replacement or repair is often time-consuming. With virtualization, the virtual machine can be cloned or replicated, significantly speeding up the recovery process and enhancing business continuity.
- **Improved Productivity and Efficiency** – Given the nature of the virtual environment, virtual machines are easier to install, maintain and update.
- **Development and Testing** – Simulation of virtual machines helps the developer to run tests without altering the production environment. Software updates can be rolled out faster since the environment offers agility for development & testing.
- **Enhanced Security** – Since virtual servers are isolated, security is more adaptive in such an environment protecting the virtual machines from malware and vicious attacks as they move from one host to another.

Understanding Containers:

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Dockers- A containerization technology:

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

[Further reading](#)

Unit - IV

Testing Tools and automation

DevOps Testing Tools

Unit Testing



TypeMock™

EMMA

PARASOFT

SimpleTest
Unit Testing for PHP

Performance Testing



predator



TestComplete

Automated Testing



TestProject

Selenium⁴

Leapwork

testsigma

TRICENTIS
Tosca

Continuous Testing



appium

appVerify

Bamboo

docker

Jenkins

Unit tests are very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your software.

Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected.

Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.

End-to-end testing replicates a user behavior with the software in a complete application environment. It verifies that various user flows work as expected and can be as simple as loading a web page or logging in or much more complex scenarios verifying email notifications, online payments, etc...

Acceptance tests are formal tests that verify if a system satisfies business requirements. They require the entire application to be running while testing and focus on replicating user behaviors.

Performance tests evaluate how a system performs under a particular workload. These tests help to measure the reliability, speed, scalability, and responsiveness of an application. For instance, a performance test can observe response times when executing a high number of requests, or determine how a system behaves with a significant amount of data.

Smoke tests are basic tests that check the basic functionality of an application. They are meant to be quick to execute, and their goal is to give you the assurance that the major features of your system are working as expected.

[Further reading please click here](#)

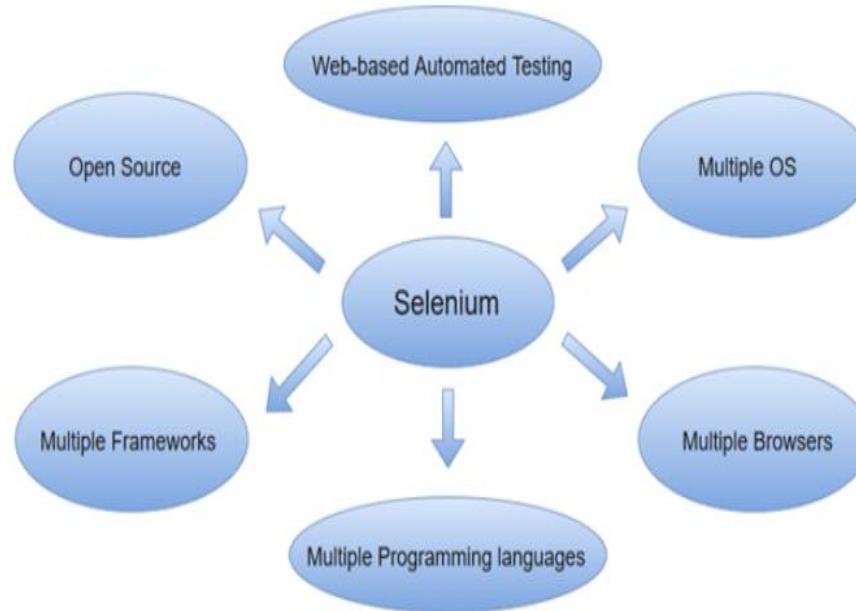
Advantages

Quantitative	Qualitative
Reduced costs	More Accurate, Consistent Results
Generates quicker Return On Investment	Considerably faster than manual testing
Improves overall organizational efficiency;	Increased and broader test coverage
Faster feedback	Faster testing cycles and faster script generation

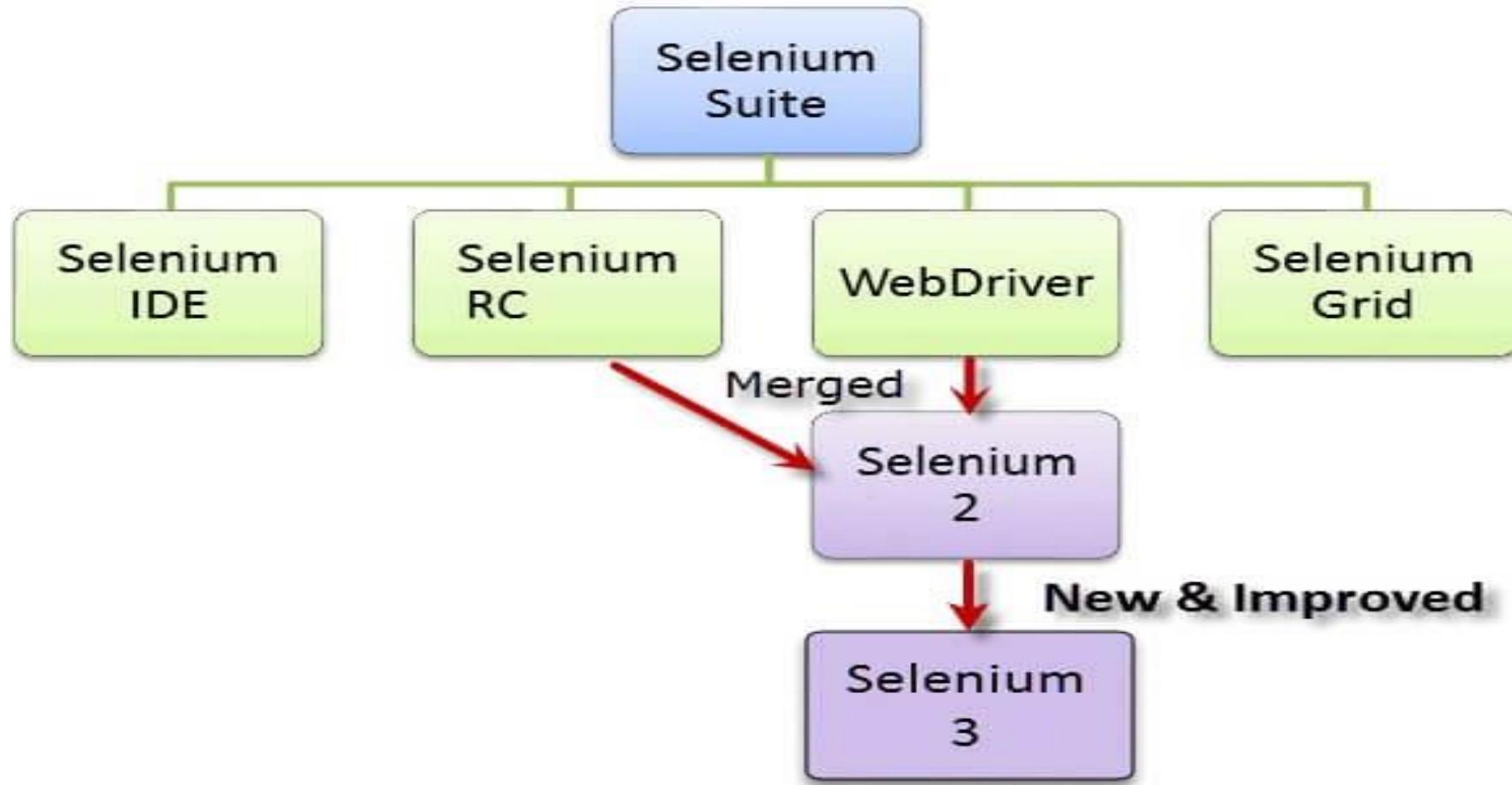
Disadvantages

- **developing time needed:** The first time developing the testing software is a time-taking process.
- **debugging script is challenging:** Not only is debugging tricky, but the bigger issue is that sometimes the errors missed in the script can lead to serious consequences.
- **expensive tools and labor:** Specialized automated software tools come at a premium. Additionally, automated QA engineers are more expensive to hire than their manual counterparts.
- **test program needs support and maintenance:** As and when conditions change, the program's coding needs to be modified and re-tested to verify it is working per requirements. This is a costly process.

Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite. It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.



Selenium can be used to automate functional tests and can be integrated with automation test tools such as **Maven**, **Jenkins**, & **Docker** to achieve continuous testing.



Note:

1. Selenium 1 version consisted of the combination of Selenium IDE, Selenium Remote Control, and Selenium Grid.
2. Version of Selenium 2 = Selenium IDE + Selenium Remote Control + Selenium WebDriver + Selenium Grid
3. Since Selenium RC was completely removed from the version of Selenium 3, therefore, this version consisted of IDE, WebDriver, and Grid.
4. Selenium 4.

Selenium WebDriver is a web framework that permits you to execute cross-browser tests.

This tool is used for automating web-based application testing to verify that it performs expectedly.

Selenium WebDriver allows you to choose a programming language to create test scripts. As discussed earlier, it is an advancement over Selenium RC to overcome a few limitations.

In WebDriver, test scripts can be developed using any of the supported programming languages and can be run directly in most modern web browsers.

Languages supported by WebDriver include C#, Java, Perl, PHP, Python and Ruby.

- Selenium is an **open source and portable** Web testing Framework.
- Selenium IDE provides a **playback and record feature** for authoring tests without the need to learn a test scripting language.
- It can be considered as the **leading cloud-based testing platform** which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
- Selenium **supports various operating systems, browsers and programming languages**. Following is the list:
 - Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
 - Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
 - Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It also supports **parallel test execution** which reduces time and increases the efficiency of tests.
- Selenium can be **integrated with frameworks like Ant and Maven for source code compilation**.
- Selenium requires **fewer resources** as compared to other automation test tools.
- **WebDriver API** has been indulged in selenium which is one of the most important modifications done to selenium.
- Selenium web driver does not require server installation, **test scripts interact directly with the browser**.

Testing backend integration points

- Most system testing **efforts go through GUI**. This is because testing validates if the software is ‘fit for use’ by the end-user or not.
- But, software has a lot of other elements too that aren’t **directly visible or available to the user for direct interaction**. It does not make these elements any less important and they must too undergo thorough testing.
- The combination of all these well-functioning elements makes a **fully formed software application**. We can combine everything we do not directly see as ‘Back-end’.
- The backend testing ensures that your application **works fine at the server-side** and it assures the **quality of the database**.
- Some of the Backend Testing elements are:

- ✓ Database
- ✓ APIs
- ✓ Servers

Database Testing?

Database Testing is a type of software testing that checks the **schema(the skeleton structure that represents the logical view of the entire database.)**, **tables**, **triggers(a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.)**, etc. of the Database under test.

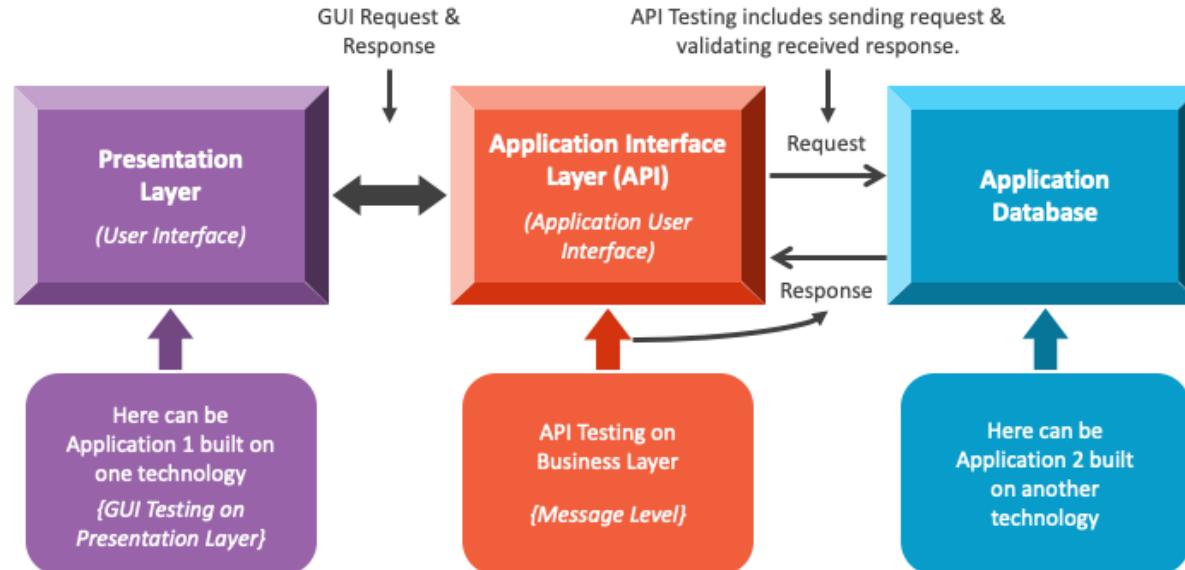
- It also checks data **integrity(accuracy) and consistency**. It may involve creating complex queries to load/stress test the Database and check its responsiveness.
- **Database Testing is Important** in software testing because it ensures **data values and information received and stored into database are valid or not.**
- Database testing helps to **save data loss**, **saves aborted transaction data** and **no unauthorized access to the information.**

Let us consider a Banking application wherein a user makes transactions. Now from Database Testing or DB Testing viewpoint following, things are important:

- The application stores the transaction information in the application database and displays them correctly to the user.
- No information is lost in the process.
- No unauthorized individual is allowed to access the user's information.
- To ensure all these above objectives, we need to use data validation or data testing.

API TESTING

Overview of API Testing



❖ API testing

- API stands for **Application Program Interface** and this is basically where all the programming logic resides. It does not have a UI which is one of the biggest challenges when it comes to testing it.
- The purpose of **API Testing** is to check the functionality, reliability, performance, and security of the programming interfaces.
- In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the **API**, get output, and note down the system's response.
- API tests are very different from GUI Tests and won't concentrate on the look and feel of an application. It mainly **concentrates on the business logic layer of the software architecture**.

What is Server Testing:

Server testing typically involves running a series of script-based tests on the server tests where the app or site is stored.

Server testing allows your development team to identify and fix flaws like downtime or run-time errors, for example.

Here are three good reasons to perform server tests:

- **Find bugs:** You can find bugs and other errors that might affect the app or website performance. Thanks to server testing, you can determine whether any new feature will affect how a project works.

Determine concurrent usage: When you conduct the tests, you will understand the maximum number of concurrent users a site or an app can handle at any given time.

You will also have a good idea of the amount of bandwidth, storage, or processing capacity you need to acquire if your targets' current capacity falls short.

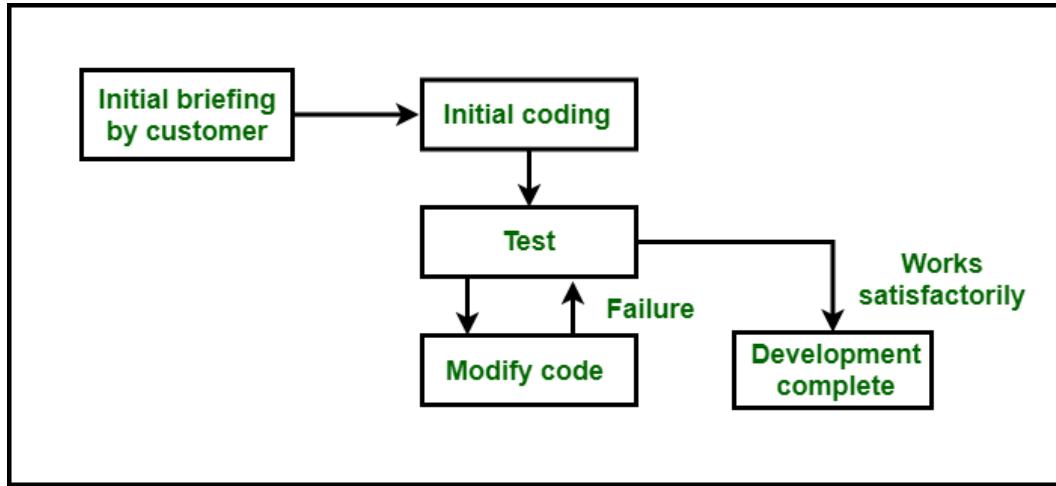
Improve website or app performance: Server testing can help you reduce the time required to request code and database optimization.

You can reduce the occurrence of memory leaks(**Memory leak** occurs when programmers create a memory in heap and forget to delete it.) and system crashes.

REPL-driven development

- The term “**REPL**” is an acronym for Read, Evaluate, Print and Loop because that’s precisely what the computer does..!
- **REPL (READ, EVAL, PRINT, LOOP)** is a computer environment similar to Shell (Unix/Linux) and command prompt.
- A Read-Eval-Print Loop, or **REPL**, is a computer environment where user inputs are read and evaluated, and then the results are returned to the user.
- The REPL is an instant feedback workflow that continually runs your code without the need to manually run an explicit compile-build-run cycle.

REPLs facilitate **exploratory programming** and debugging because the programmer can inspect the printed result before deciding what expression to provide for the next read.



The read–eval–print loop involves the programmer more frequently than the classic edit–compile–run–debug cycle.

- REPLs provide an interactive environment to explore tools available in **specific environments or programming languages**.
- Some examples include the **Node.js console**, **IPython**, the **Bash shell**, and the developer console found in most web browsers.
- REPL driven development **is the foundation of working with Clojure effectively**.
- Clojure **is a high level, dynamic functional programming language**. Clojure is **designed based on the LISP programming language** and **has compilers which makes it run on both Java and .Net runtime environment**.

- Clojure is a powerful, fun and highly productive language for developing applications and services. The clear language design is supported by a powerful development environment known as the REPL (read, evaluate, print, loop).
- The REPL gives you **instant feedback** on what your code does and enables you to test either a single expression or run the whole application (including tests).
- Clojure uses repl based development
- The REPL is the environment in which all Clojure code runs, whether that be during development, testing and production systems.
- As we are designing the code, we evaluate expressions and print the result under the expression as a comment.

<https://clojure.org/guides/repl/introduction>

❖ clojure.main :

The clojure.main namespace provides functions that allow Clojure programs and interactive sessions to be launched via Java's application launcher tool java

□ clojure.main -help

runs an interactive Read-Eval-Print Loop

main options:

-r, --repl Run a repl ,path Run a script from a file or resource , - Run a script from standard input, -e, --eval string Evaluate expressions in string; print non-nil value

Most Clojure programmers, they use a REPL integration in their editor, which enables them to write expressions directly in their editor buffer and have them evaluated in the REPL with one hotkey.

Clojure.testlibrary is part of the Clojure standard library that provides a simple way to start writing unit tests.

What is Test Driven Development (TDD)?

- Test Driven Development (TDD) is a software development practice that focuses on creating unit test cases before developing the actual code. It is an iterative approach that combines programming, the creation of unit tests, and refactoring.
- **Test Driven Development** is the process in which test cases are written before the code that validates those cases.i.e., test cases are developed to specify and validate what the code will do.
- In simple terms, Test-Driven Development starts with designing and developing tests for every small functionality of an application.
- In TDD, developers start creating small test cases for every feature based on their initial understanding.

The primary intention of this technique is to modify or write new code only if the tests fail. The simple concept of TDD is to write and correct the failed tests before writing new code (before development). It is a process of modifying the code in order to pass a test designed previously.

test cases for each functionality are created and tested first and if the test fails then the new code is written in order to pass the test and making code simple and bug-free. The test might fail as the tests are developed even before the development. Development team then develops and refactors the code to pass the test.

TDD framework instructs developers to write new code only if an automated test has failed. This helps to avoid duplication of code as we write a small amount of code at a time in order to pass tests.

As Test-Driven development is a process of developing and running automated test before actual development of the application. Hence, TDD sometimes also called as **Test First Development**.

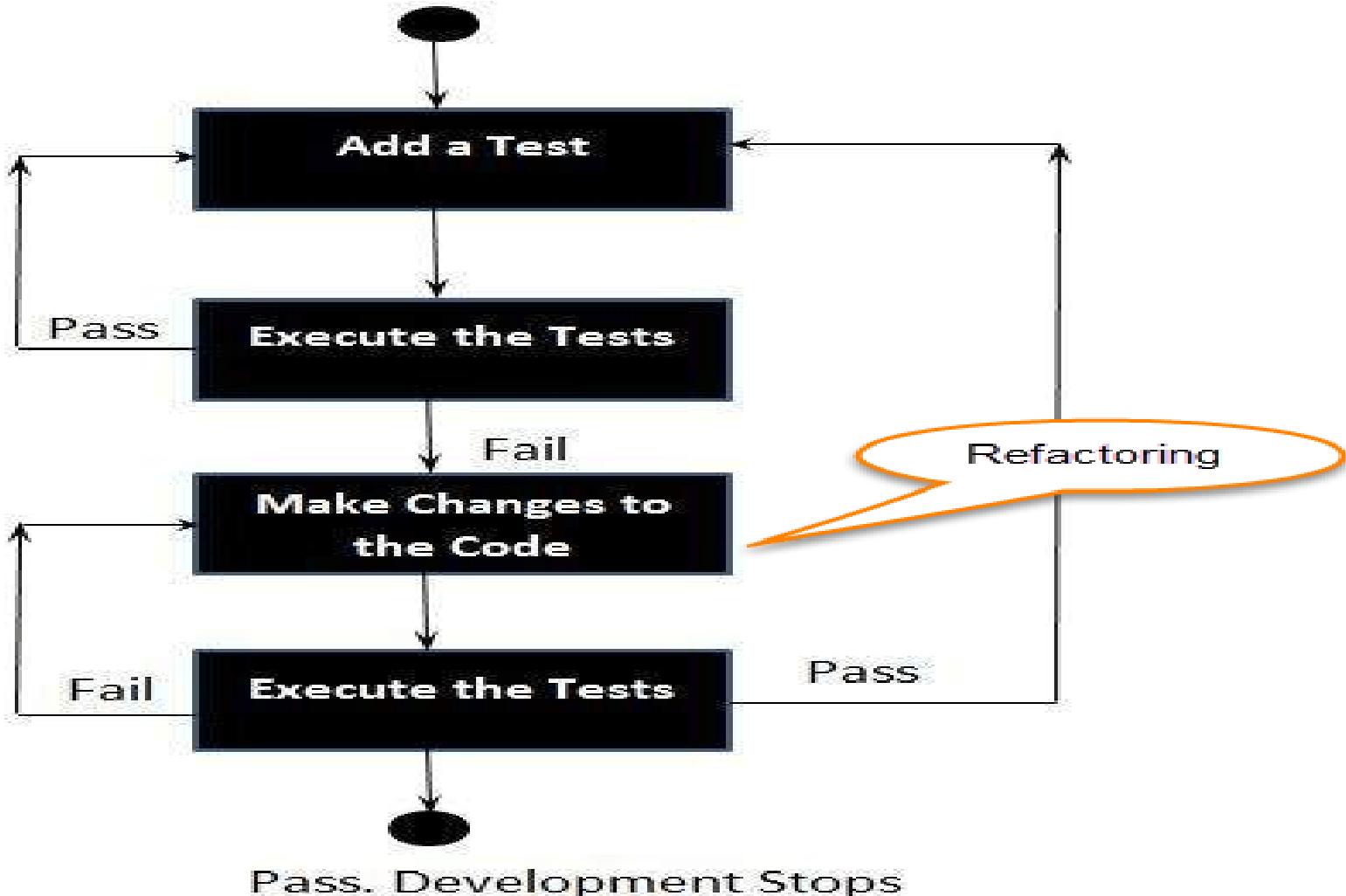
- The TDD approach derives its roots from the **Agile manifesto principles and Extreme programming**. As the name suggests, the **test process drives software development**. Moreover, it's a structuring practice that enables developers and testers to obtain optimized code that proves to be resilient in the long term.
- It more **emphasis on production code** rather than test case design.

The following sequence of steps is generally followed:

- Add a test – Write a test case that describe the function completely. In order to make the test cases the developer must understand the features and requirements using user stories and use cases.
- Run all the test cases and **make sure that the new test case fails**.
- **Write the code** that passes the test case
- Run the test cases
- **Refactor code** – This is done to remove duplication of code.
- Repeat the above mentioned steps again and again

TDD is usually described as a sequence of events, as follows:

- **Implement the test:** As the name implies, **you start out by writing the test and write the code afterwards.** To be able to write the test, the developer must find all relevant requirement specifications, use cases, and user stories. The shift in focus from coding to understanding the requirements can be beneficial for implementing them correctly.
- **Verify that the new test fails:** The newly added test should fail **because there is nothing to implement the behaviour properly yet**, only the stubs and interfaces needed to write the test. Run the test and verify that it fails.
- **Write code that implements the tested feature:** The code we write doesn't yet have to be particularly elegant or efficient. Initially, we just want to make the new test pass.
- **Verify that the new test passes together with the old tests:** When the new test passes, we know that we have implemented the new feature correctly. Since the old tests also pass, we haven't broken existing functionality.
- **Refactor the code:** The word "refactor" has mathematical roots. In programming, it means cleaning up the code and, among other things, making it easier to understand and maintain. We need to refactor since we cheated a bit earlier in the development.
- TDD is a style of development that fits well with DevOps, but it's not necessarily the only one. The primary benefit is that you get good test suites that can be used in Continuous Integration tests.



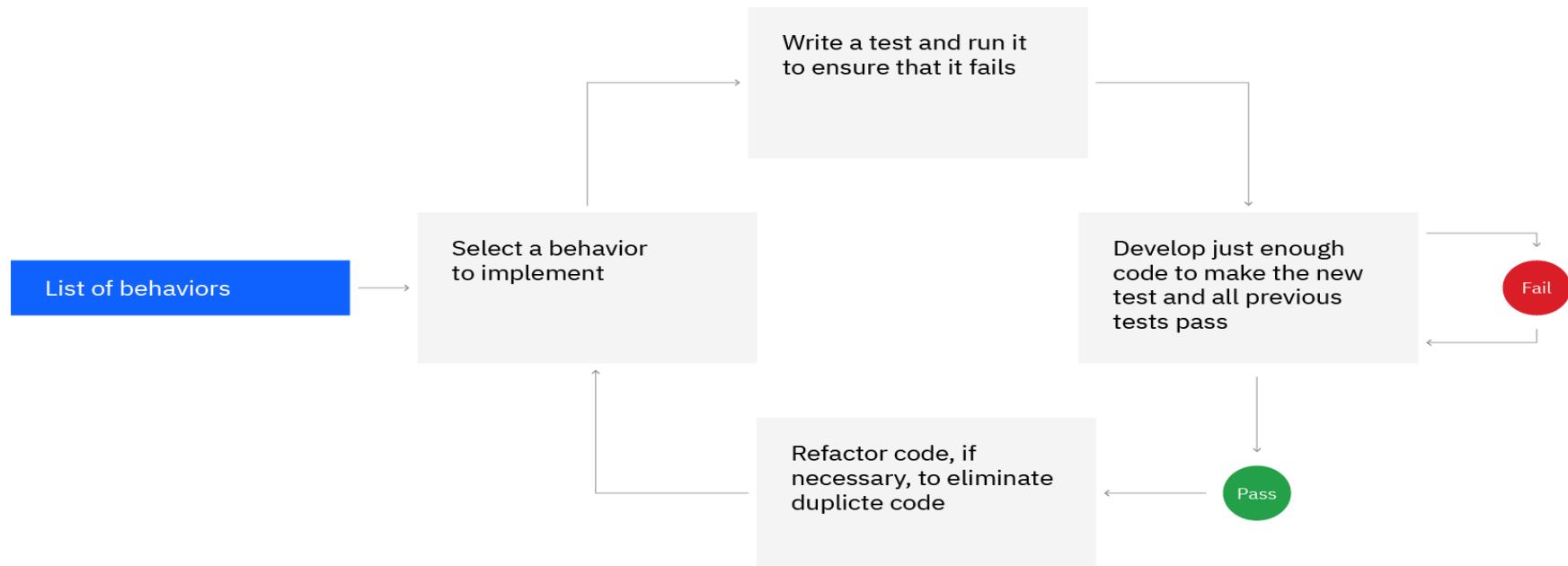
Pass. Development Stops

Red/Green/Refactor cycle

red – Create a test case and make it fail

Green – Make the test case pass by any means.

Refactor – Change the code to remove duplicate/redundancy



- Much less debug time
 - Code proven to meet requirements
 - Mostly zero defects
 - Shorter development cycle
-
- Following are the main advantages of Test Driven Development in Software Engineering:

Early bug notification.

- Developers test their code but in the database world, this often consists of manual tests or one-off scripts. Using TDD you build up, over time, a suite of automated tests that you and any other developer can rerun at will.

Better Designed, cleaner and more extensible code.

- It helps to understand how the code will be used and how it interacts with other modules.
- It results in better design decision and more maintainable code.

Confidence to Refactor

- If you refactor code, **there can be possibilities of breaks in the code**. So having a set of automated tests you can fix those breaks before release. Proper warning will be given if breaks found when automated tests are used.
- Using TDD, should results in faster, more extensible code with fewer bugs that can be updated with minimal risks.

Good for teamwork

- **In the absence of any team member, other team members can easily pick up and work** on the code. It also aids knowledge sharing, thereby making the team more effective overall.

Good for Developers

- Though developers have to spend more time in writing TDD test cases, it takes a lot less time for debugging and developing new features. You will write cleaner, less complicated code.

It can enable **faster innovation and continuous delivery** because the code is robust.

It makes **your code flexible and extensible**. The **code can be refactored or moved** with minimal risk of breaking code.

The requirements are implemented with little to no wasted effort because only the function that is needed is written.

PUPPET

Consider a system administrator working with multiple servers. If one of the servers has an issue, they can easily fix it. The situation becomes problematic, however, when multiple servers are down, This is where **Puppet** can help

What is Puppet?

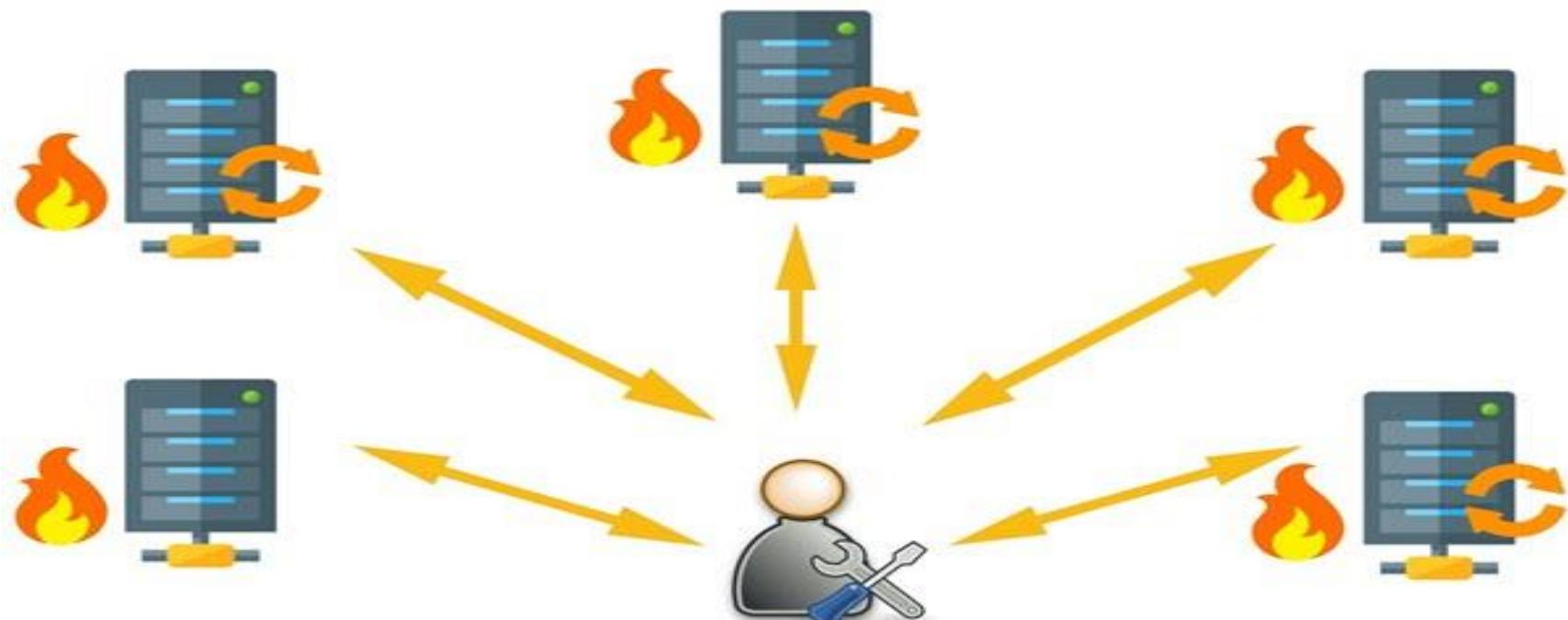
- **Puppet** is a system management tool for centralizing and automating the configuration management process. **Puppet** is also used as a software deployment tool.
- It is an **open-source configuration management software** widely used for server **configuration, management, deployment, and orchestration**(Orchestration is **the automated configuration.**) of various applications and services across the whole infrastructure of an organization.
- Puppet is specially designed to manage the configuration of Linux and Windows systems. It is written in Ruby.

What is Configuration Management?

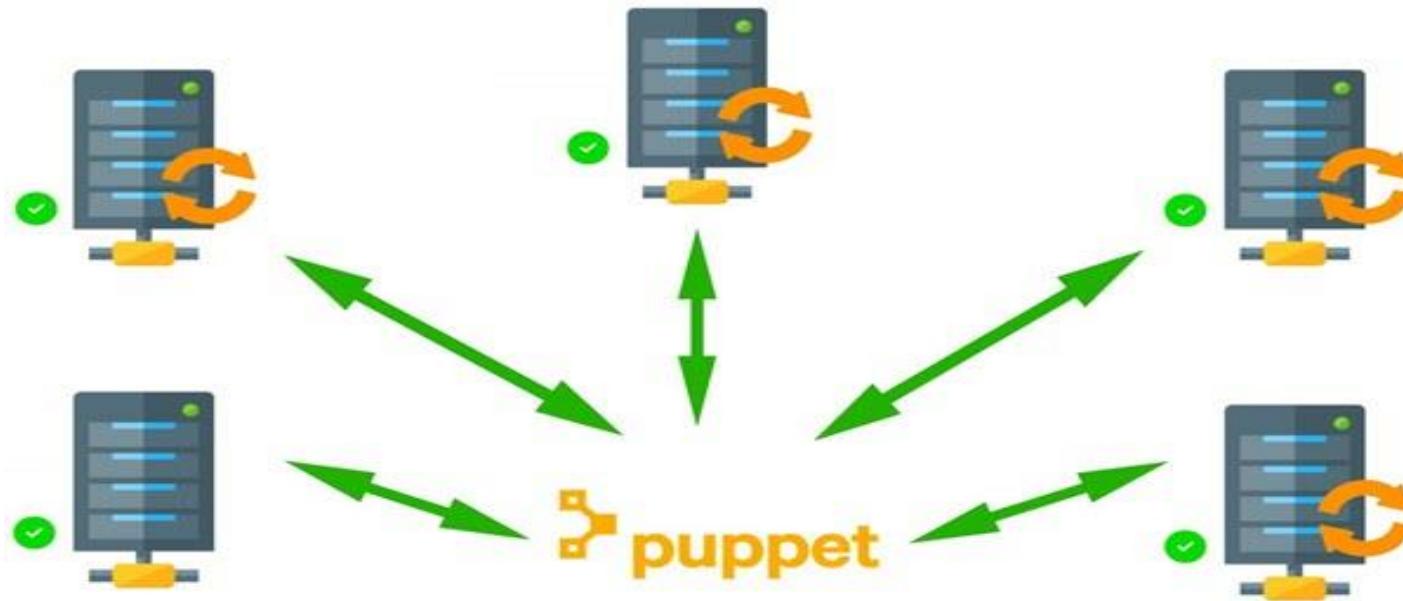
- Configuration management is the process of maintaining software and computer systems (for example servers, storage, networks) in a known, desired and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes.
- System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate this task, by writing scripts.
- However, it is a difficult job when they are working on a massive infrastructure. The Configuration Management tool like a Puppet was introduced to resolve such issues.

What Puppet can do?

For example, you have an infrastructure with about 100 servers. As a system admin, it's your role to ensure that all these servers are always up to date and running with full functionality.



To do this, you can use Puppet, which allows you to write a simple code which can be deployed automatically on these servers. This reduces the human effort and makes the development process fast and effective.



Puppet performs the following functions:

- Puppet allows you to define **distinct configurations** for every host.
- The tool allows you to **continuously monitor servers to confirm whether the required configuration exists** or not and it is not altered. If the config is changed, Puppet tool will **revert to the pre-defined configuration on the host.**
- It also provides control over all the configured system, so a **centralized change** gets automatically effected.
- It is also used as a deployment tool as it automatically deploys software to the system. It implements the **infrastructure as a code** because policies and configurations are written as **code.**

Deployment models of configuration management tools

There are two deployment models for configuration management tools :

- Push-based deployment model: initiated by a master node.
- Pull-based deployment model: initiated by agents.

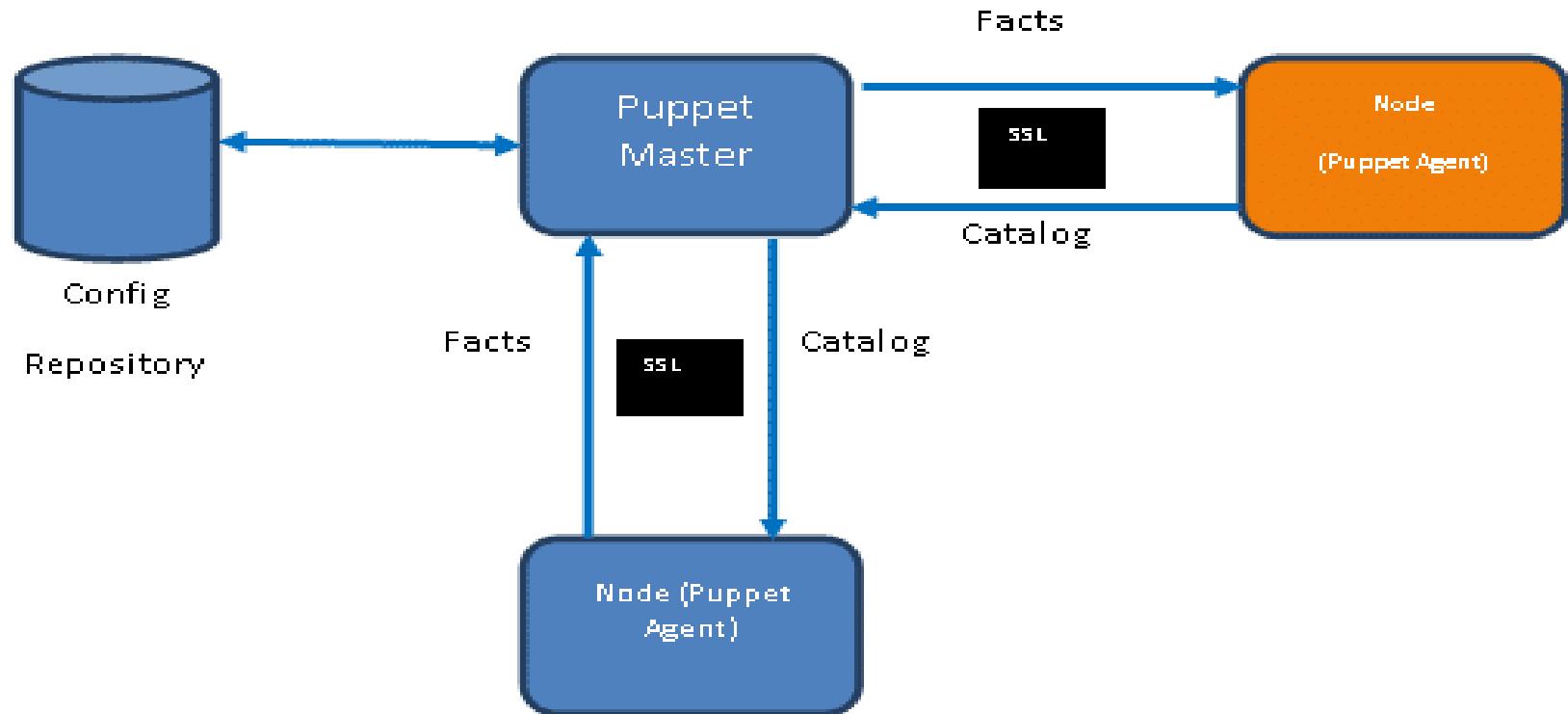
Push-based deployment model:

In this deployment model master server pushes the configurations and software to the individual agents. After verifying a secure connection, the master runs commands remotely on the agents. For example, Ansible and Salt Stack.

Pull-based deployment model.

In this deployment model, individual servers contact a master server, verify and establish a secure connection, download their configurations and software and then configure themselves accordingly — for example, Puppet and Chef.

Understanding the Puppet Architecture and Puppet Components



Client-Server Architecture

How Puppet works?

Puppet is based on a **Pull deployment model**, where the agent nodes check in regularly after every **1800 seconds** with the master node to see if anything needs to be updated in the agent. If anything needs to be updated the **agent pulls the necessary puppet codes** from the master and performs required actions.

Example: Master – Agent Setup:

The Master:

A Linux based machine with Puppet master software installed on it. **It is responsible for maintaining configurations in the form of puppet codes.** The master node can only be Linux.

The Agents:

The target machines managed by a puppet with the **puppet agent software installed** on them.

The agent can be configured on any supported operating system such as Linux or Windows or Solaris or Mac OS.

The communication between master and agent is established through **secure certificates**.

- **Config repository:** Config repository is where the entire server-related configurations and nodes are stored. They can be pulled at any time as required.

Catalog

A catalog is a document that describes the desired system state for one specific server. It lists all of the resources that need to be managed, as well as any dependencies between those resources.

Manifests

Manifests are files with extension ".pp", where we declare all resources to be checked or to be changed. Resources may be files, packages, services and so on.

Puppet Master Agent Communication

Secure Sockets Layer

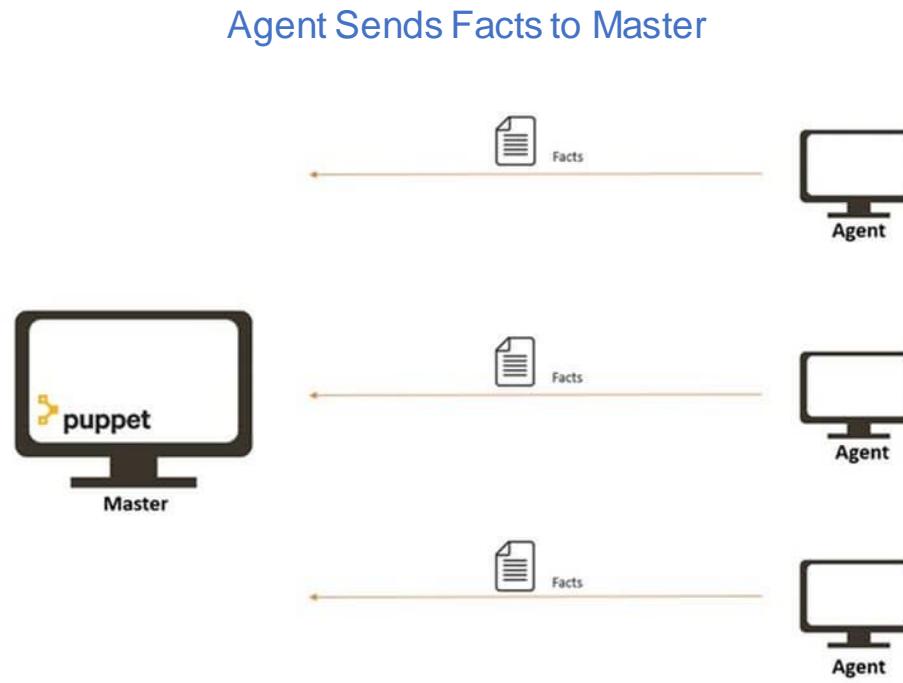


Master



Communication between the Master and the Agent:

Step 1) Once the connectivity is established between the agent and the master, the Puppet agent sends the data about its state to the Puppet master server. **These are called Facts**: This information includes the hostname, kernel details, IP address, file name details, etc....



Step 2) Puppet Master uses this data and compiles a list with the configuration to be applied to the agent. This list of configuration to be performed on an agent is known as a **catalog**. This could be changed such as package installation, upgrades or removals, File System creation, user creation or deletion, server reboot, IP configuration changes, etc.

Master sends a catalog to Agent



Step 3) The agent uses this list of configuration to apply any required configuration changes on the node.

In case there are no drifts in the configuration, Agent does not perform any configuration changes and leaves the node to run with the same configuration.

Agent applies configuration



Step 4) Once it is done the node reports back to puppet master indicating that the configuration has been applied and completed.

The Connection Between Puppet Master Server and Puppet Agent Nodes

As we can see, the Puppet master and the Puppet agents have to communicate with each other in order to let Puppet work seamlessly, but how exactly do they communicate?

Let's understand the workflow of Puppet step by step:

- The nodes that the Puppet master controls have Puppet agents installed on them. The agents collect all the configuration information about their particular nodes **using facts**. Agents then send facts to the Puppet master.

- After gaining all the information, **the Puppet master compiles a catalog based on how the nodes should be configured**. The Puppet master then sends back the catalog to the agents.
- Each agent uses these catalogs and the information in them to make necessary configuration updates on their nodes and then reports back to the Puppet master.
- The Puppet master can also share the reports with a third-party tool if needed.
- The Puppet master communicates with a Puppet agent via HTTPS (HyperText Transfer Protocol Secure) with client verification.
- The Puppet master provides an HTTP interface. Whenever the Puppet agent has to make a request or submission to the Puppet master,
 - it just makes an HTTPS request to one of the endpoints available in the HTTP interface provided by the Puppet master.

Puppet
Agent

Puppet
Master

Request for Master Certification

Send Master Certificate

Request for slave certificate

Send Slave Certificate

Request for Data

Send Data



ANSIBLE

What is Ansible?(Push based)

- Ansible is an open source DevOps tool which can help the business in configuration management, deployment, provisioning(**the process of setting up IT infrastructure**), etc.
- It leverages SSH(**Secure Shell is a network communication protocol that enables two computers to communicate..** and share data to communicate between servers.
- Ansible provides reliability, consistency, and scalability to your IT infrastructure. You can automate configurations of databases, storage, networks, firewalls using Ansible.
- It makes sure that all the necessary packages and all other software are consistent on the server to run the application.

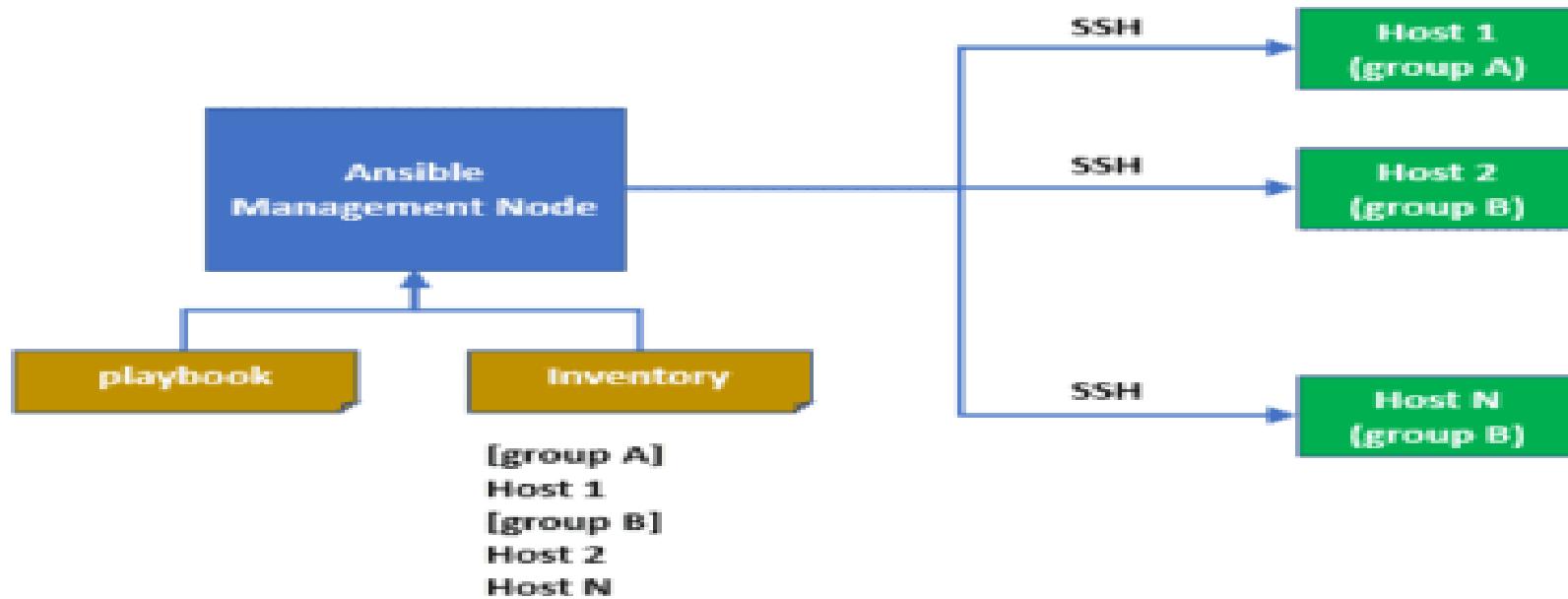
Let's take an example; you've got a debug version of an application that is built on visual C++.

Now if you want to run that application on a computer, you would need to meet some prerequisites like Microsoft Visual C++ library DLLs, and you would need visual C++ installed in your computer.

So, this is the part where Ansible will make sure that all these basic packages and all the software's are installed in your computer so that your application can run smoothly on all the environments, may it be test or production environment.

It also holds all the historical data of your application, so if at any time you want to roll back to the previous version, or you want to upgrade it, you can easily do that.

How Ansible Works?



What are playbook tasks?

A task is the smallest unit of action you can automate using an Ansible playbook. Playbooks typically contain a series of tasks that serve a goal, such as to set up a web server, or to deploy an application to remote environments. Ansible executes tasks in the same order they are defined inside a playbook.

Ansible Workflow

Ansible works by connecting to your nodes and **pushing out a small program called Ansible modules** to them.

Then Ansible executed these **modules** and removed them after finished. The library of modules can reside on any machine.

In the above image, the **Management Node** is the controlling node that controls the entire execution of the playbook.

The **inventory** file provides the list of hosts where the Ansible modules need to be run.

The **Management Node** makes an **SSH** connection and executes the small modules on the host's machine and install the software.

Ansible removes the modules once those are installed so expertly. It connects to the host machine executes the instructions, and if it is successfully installed, **then remove that code in which one was copied on the host machine.**

Let's take a look at some of the following features.

Agentless – Which means there is no kind of software or any **agent managing the node like other solution such as puppet and chef**.

Python – Built on top of **python**, which is fast and one of the robust programming languages in today's world.

SSH – Very simple password less network authentication protocol which is secure. So, your responsibility is to copy this key to the client

Push architecture – Push the necessary configurations to them, clients. All you have to do is, write down those configurations (**playbook**) and **push them all at once to the nodes**. You see how powerful it can be to push the changes to thousands of servers in minutes.

Setup – a minimal requirement and configuration needed to get it to work.

Benefits of Ansible

- **Free:** Ansible is an open-source tool.
- **Very simple to set up and use:** No special coding skills are necessary to use Ansible's playbooks (more on playbooks later).
- **Powerful:** Ansible lets you model even highly complex IT workflows.
- **Flexible:** You can orchestrate the entire application environment no matter where it's deployed. You can also customize it based on your needs.
- **Agentless:** You don't need to install any other software or firewall ports on the client systems you want to automate. You also don't have to set up a separate management structure.
- **Efficient:** Because you don't need to install any extra software, there's more room for application resources on your server.

Next, in our path to understanding what ansible is, let us find out the features and capabilities of Ansible.

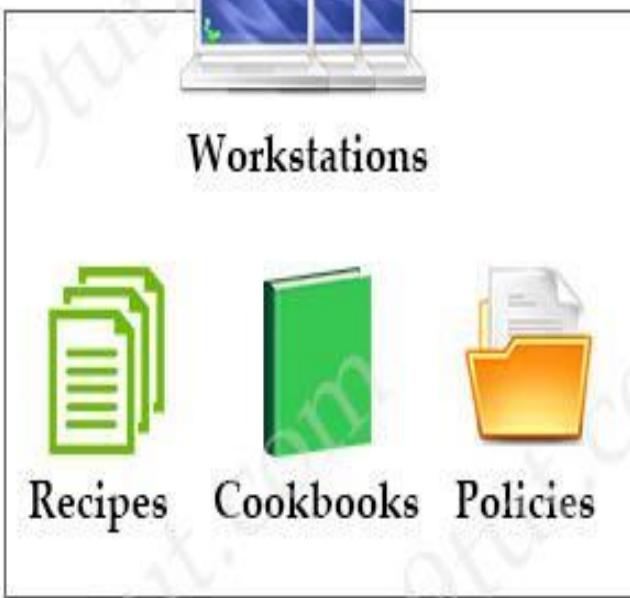
Chef is an automation platform that configures and manages your network

infrastructure.(PULL BASED) Chef transforms infrastructure into code. “Infrastructure into code” here means “deploy your code/application/configuration and policy” on many machines or instances automatically via your code.

Chef is a tool used for Configuration Management which closely competes with Puppet. Chef is an automation tool that provides a way to define infrastructure as code.

As shown in the diagram below, there are three major Chef components:

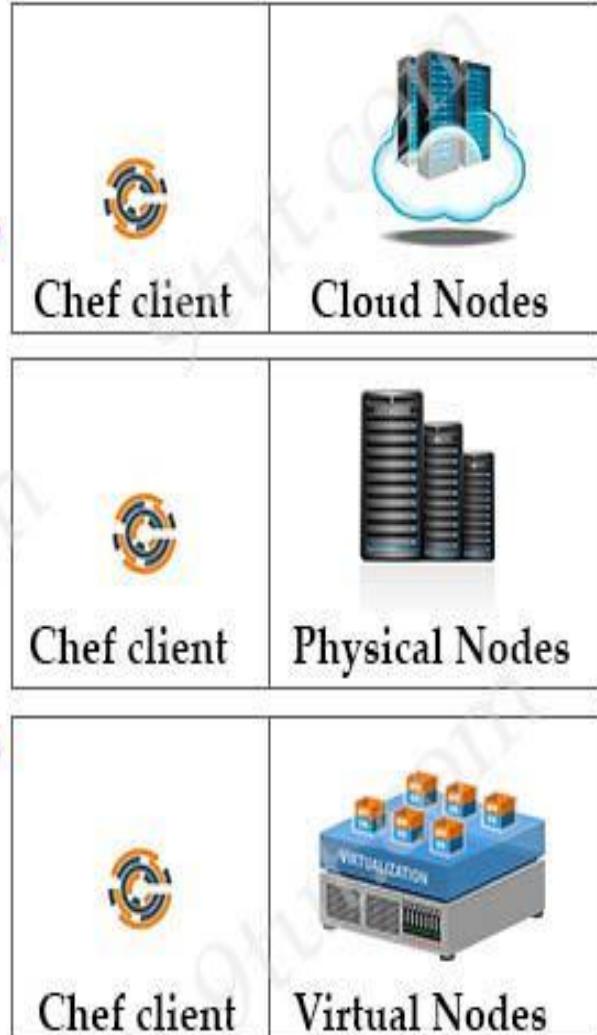
- Workstation
- Server
- Nodes



Knife
upload
config



Chef Server



Workstations: simply personal computers where all development configuration code is created, tested, and changed before uploading to the Chef Server.

Each Chef workstation also has **a command line tool called “Knife”**, which will be used to upload configuration changes to the Chef Server.

Workstations are the place to write **Recipes** and **Cookbooks**:

- **Recipes:** A Recipe is a collection of resources that **describes a particular configuration ,how to implement,code.**
- **Cookbooks:** Multiple Recipes can be grouped together to form a **Cookbook**. A Cookbook defines a scenario and contains everything that is required to support that scenario.
- **policy.** It describes everything that is required to configure part of a system and in which order it is to be used. The user writes Recipes that describe
 - how Chef manages applications and utilities (such as Apache HTTP Server, MySQL, or

- Cookbooks: Multiple Recipes can be grouped together to form a Cookbook. A Cookbook defines a scenario and contains everything that is required to support that scenario.
 - A Cookbook also includes **attributes, libraries, metadata, and other files** that are necessary for supporting each configuration.
 - Cookbooks are created using **Ruby language** is used for specific resources.
-
- **Writing Cookbooks and Recipes that will later be pushed to the central Chef Server**

- **Chef Server:** The centralized store of our infrastructure's configuration. The Chef server stores, manages and provides configuration to all nodes that make up the infrastructure.
- **Nodes:** are the servers **where your code needs to run**. Chef server manages Nodes by **Chef client**, which is a software installed on each Node.
- Chef client retrieving configuration information from the Chef Server. **Nodes can be a cloud-based/virtual/physical server in your own data center.**

Any changes made to your infrastructure code must pass through the Chef server in order to be applied to nodes. **Prior to accepting or pushing changes, the Chef server authenticates all communication via its REST API using public key encryption.**

Chef client periodically pulls Chef server to see if there are any changes in cookbooks or settings. If there are changes then Chef server sends the latest configuration information to Chef client. Chef client applies these changes to nodes.

Why Saltstack?

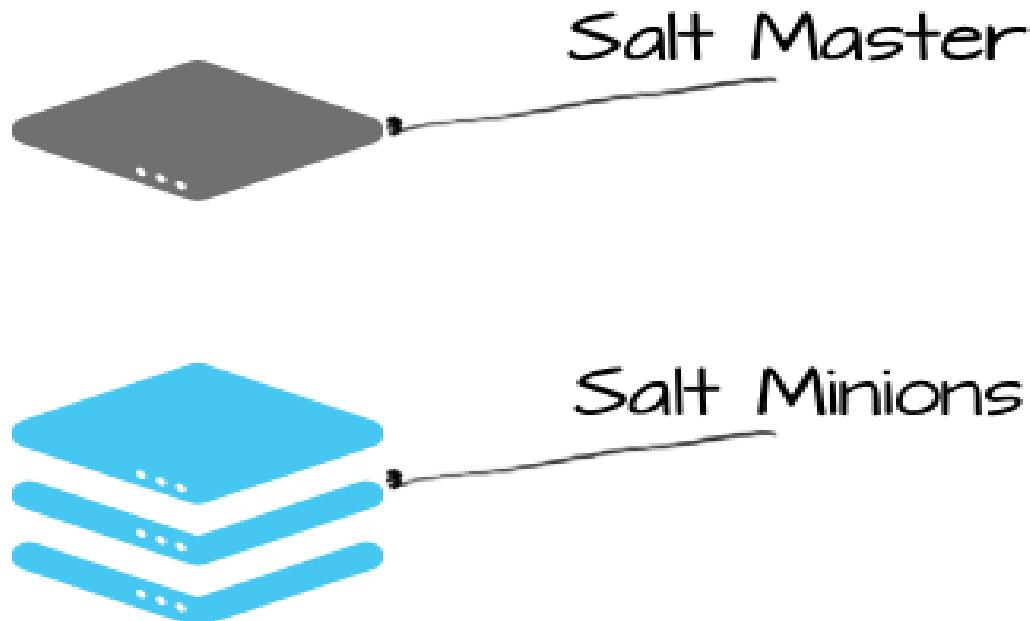
SaltStack Architecture is developed for providing speed and scale.

This is the reason why this architecture is used for managing the thousands of servers at Google, Linked In, and many more companies.

For example - If a user has thousands of servers and the user wants to perform functions on every server. A user would be required to log in to every server and perform functions one at a time. This process will be very complicated and time-consuming as functions like software installation and configuration based on particular criteria may consume a lot of time.

To overcome this problem, SaltStack Architecture has been introduced as one can perform different functions just by typing one command. Thus, SaltStack is a single solution to overcome all such issues

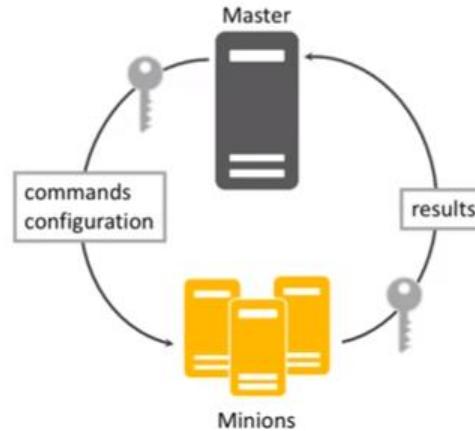
Salt uses a server-agent communication model. The server component is called the Salt master, and the agent is called the Salt minion.



The Salt master is responsible for sending commands to Salt minions, and then aggregating and displaying the results of those commands. A single Salt master can manage thousands of systems.

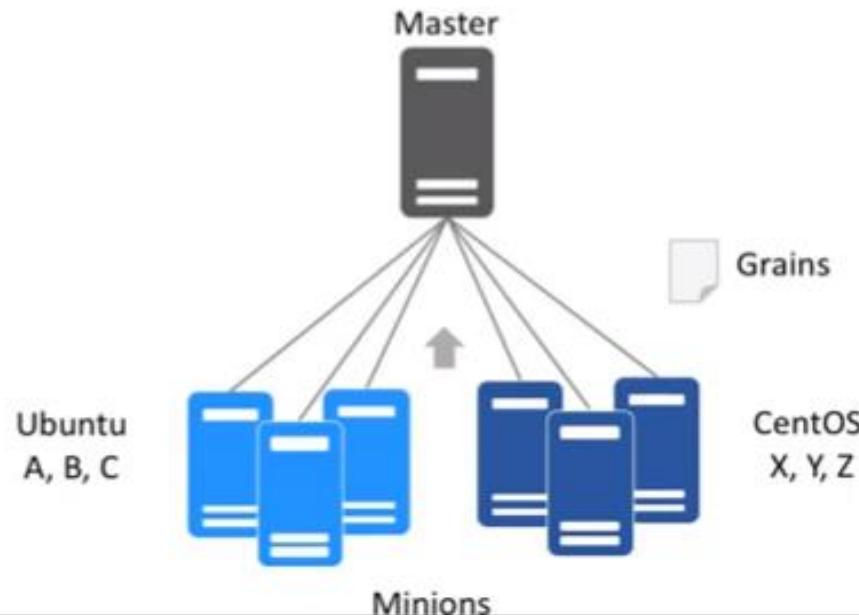
SALTSTACK ARCHITECTURE

- Client server model.



SALT GRAINS

- Static information about the minion
- Eg: OS, memory, model, serial number etc.



Salt is a **simple IT automation framework** that automates provisioning(Providing), configuration management, application deployment, service arrangements and many other IT needs.

- ✓ Remote execution of commands is the important highlight of using Salt.
- ✓ Unlike other competitors, Salt uses **ZeroMQ message bus** for the communication.
{ ZeroMQ API provides sockets , each of which can represent a many-to-many connection between endpoints. }
- ✓ It offers high speed effective communication between remote machines.
- ✓ Additionally, the remote execution can target a group of machines simultaneously.

The relevance of Salt and its applications

- ✓ Salt (commonly known as SaltStack) is a widely accepted tool for Configuration management.
- ✓ Consider that hundreds of machines need to be configured identically - install software, run some services, and make it more awesome.
- ✓ Rather than performing these tasks manually, scripts can be used to simplify it. Since the same task needs to be executed in hundreds of machines, if the script is placed in a location that can be send to all these machines will further simplify the repetitive task.
- ✓ With the help of Salt, we can manage this activity from a single system (named Salt master) and complete this in a swift manner. Just write the task once, specify the machines to be modified and execute the script remotely. The result from all machines will be send back to Salt master.

Features of Salt

- 1. Simplicity** - The Salt setup and usage is **very simple process**. Still it is powerful enough to manage hundreds of distributed systems in quick and efficient manner
- 2. Parallel Execution** - The Salt remote execution is effected in nodes **parallelly, not serially**. Hence, the time taken will be very less
- 3. Open Source** - It is a open source tool
- 4. Fast, Flexible, Scalable** - By making use of **ZeroMQ message bus**, the configuration management will be fast and it supports various types of models like **master and minion, master-only, minion-only or combination of these models**. The support to include ten thousands of minions per single master displays the effectiveness of scalability
- 5. Simple Interfaces** - Salt can be accessed from the command line, as well as from any simple Python API

Salt Architecture:

Salt remote execution is built on top of an event bus unlike other configuration management tools. Event bus makes communication between senders and receivers simple and fast.

It is specifically optimized for high performance. In Salt , the salt master is a server where the salt-master service is running. It issues commands to salt-minions, that are servers where the salt-minion service is running.

Public keys are used for authentication with master daemon, for communication it uses faster AES (Advanced Encryption Standard) encryption.

After the execution of job in minion, the job return data is sent back to master.

By default two ports are used by Salt, for the minions to communicate with master.

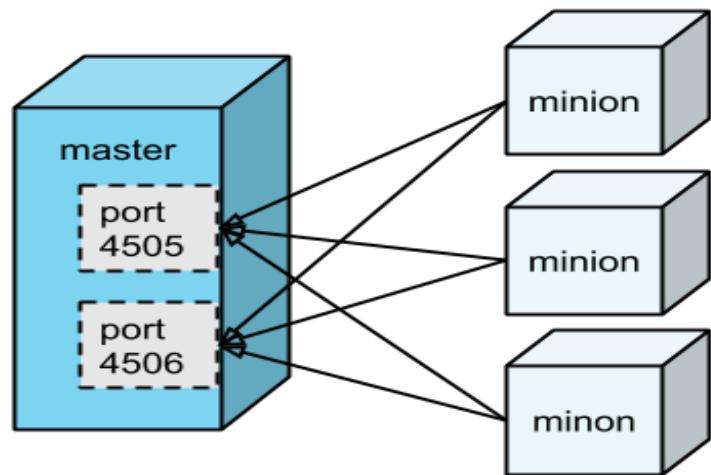
These ports work together to receive and send data to the [Message Bus](#).

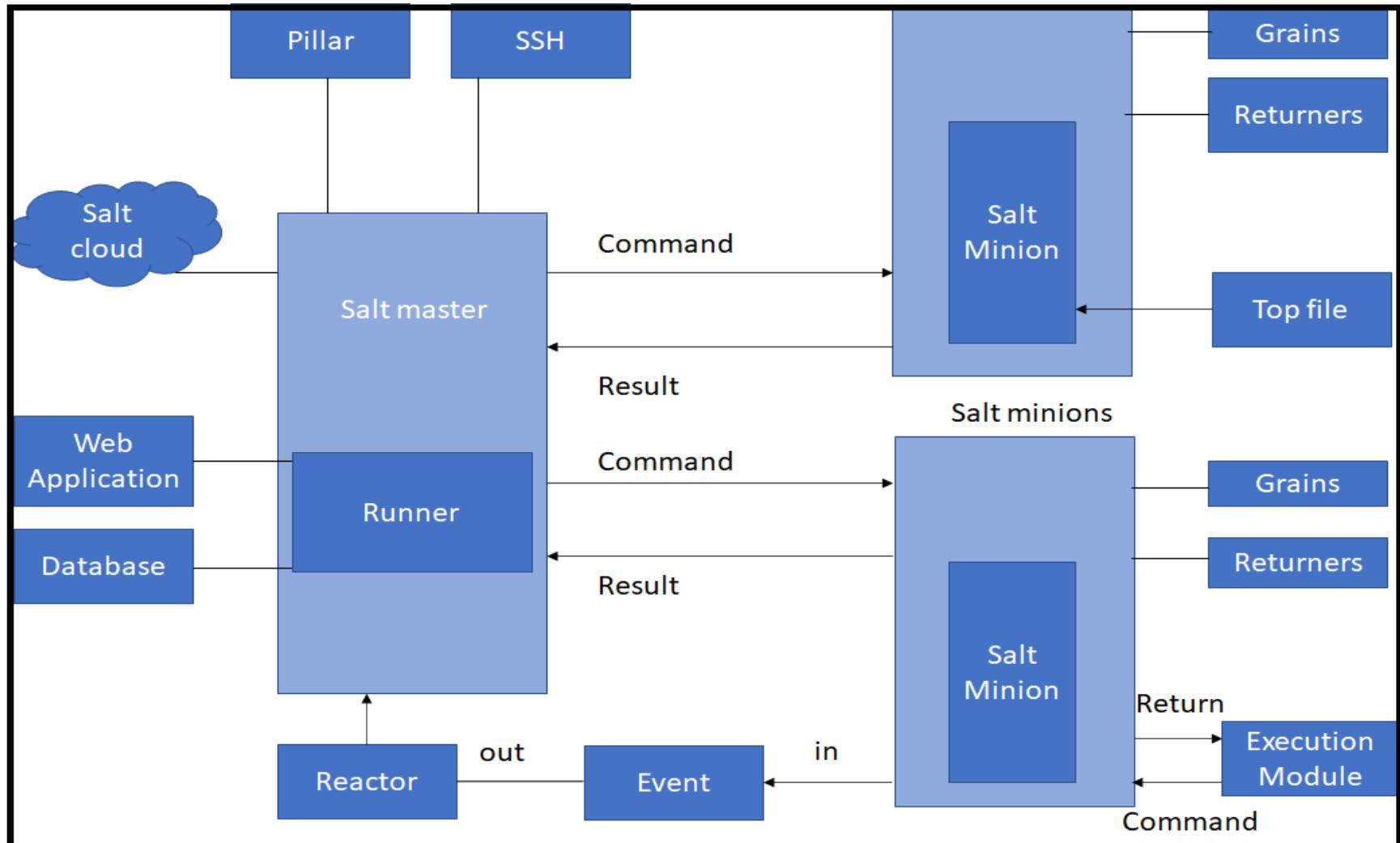
Salt uses ZeroMQ message bus. ZeroMQ provides the fastest communication possible.

Salt minions are responsible for connecting to the Salt master. Minions receive *jobs* (i.e., instructions) via the open connection to port 4505, and send results (i.e., status updates) via the open connection to master 4506.

That is, Salt masters publish jobs to all connected minions, and it is the responsibility of each minion itself to determine whether or not it is an intended *target* (i.e., whether or not to run the job).

Communication between master and minions is done using [ZeroMQ](#), an asynchronous messaging library.





Salt-master - The central management system. It is used to send commands and configurations to the salt-minion that is running on managed nodes.

Salt-minion - The managed system which receives commands and configuration from the salt-master.

Execution Modules - Ad hoc commands executed from the master's command line that reflect on the minions. It performs real-time monitoring.

Formulas - A declarative or imperative representation of a system configuration. They are used for tasks such as installing a package, setting up users or permissions, configuring and starting a service and other common tasks.

Grains - Grains provides information specific to a minion. It includes information like operating system, memory, and other system properties. Grains get loaded when the salt-minion starts. Grains are system variables.

Pillar - They are user-defined variables. These variables store highly sensitive data specific to a particular minion, such as passwords and cryptographic keys. They are stored on salt-master and then

Top file - It contains the mapping between groups of machines on the network and the configuration roles.

Runners – It is a module within salt-master that executes to perform supporting tasks. Salt runners reports job status read data from external APIs, connection status, query connected salt-minions, and much more.

Returners – Returns data from salt-minions to another system like a database. It can run on salt-minion or salt-master.

Reactor – It triggers reactions when events occur in SaltStack environment.

SaltCloud – Provision systems on cloud providers / hypervisors and bring them under management of salt-master.

SaltSSH – Run Salt commands over SSH on the systems not having salt-minion.

Virtualization stacks



What is Virtual Machine?

Applications



Windows OS



Hardware



What is Virtual Machine?

RUN APPLICATIONS
ON LINUX?

? !



Buy a new machine to run Linux?

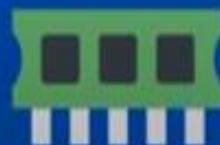


Linux OS



Virtualization

Windows



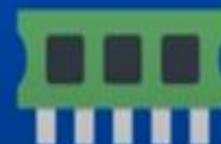
Hardware

Windows



Virtualization

Linux



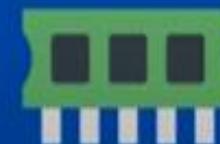
Hardware

Windows



Virtualization

Mac OS



Hardware



Hypervisor

It is software that creates and runs virtual machines (VMs).



HOST OS



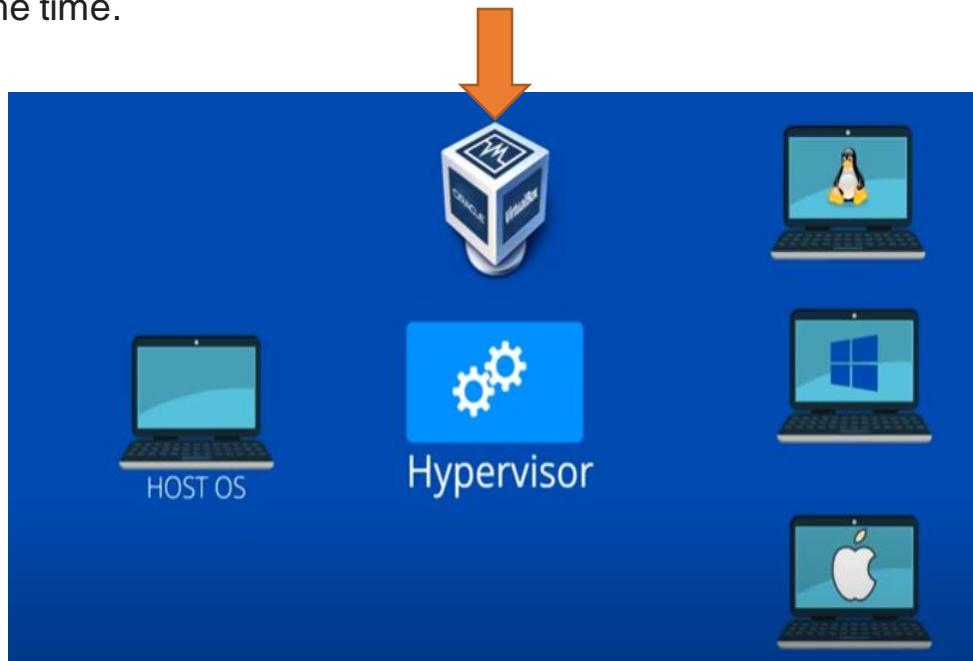
Hypervisor



Example :

Oracle VM VirtualBox is **a free and open-source hosted hypervisor for x86 virtualization**, developed by Oracle Corporation.

Oracle VM VirtualBox is **cross-platform virtualization software**. It allows users to extend their existing computer to run multiple operating systems including Microsoft Windows, Mac OS X, Linux, and Oracle Solaris, at the same time.





8GB RAM
100GB HDD



2GB RAM
20 GB HDD

VM1



2GB RAM
10 GB HDD

VM2



Benefits of VMs

- We don't need new resources to use different OS
- No risk of any issues with your primary OS
- Testing any app on different OS



- Virtualization is the process of creating **several virtual systems on a single server**. This practice maximizes a physical machine's capacity by distributing its resources between multiple users and environments.
- A single digital container is usually designed to operate one operating system, with designated spaces for each of the computer's resources. **The traditional container bears one central processing unit (CPU), a designated area for storage, and a fixed space for memory.**
- Virtualization treats all these items as a pool, so they can be **expanded or rearranged** as necessary.
- DevOps teams use virtualization to create **virtual machines (VMs)**, copying of hardware and software configurations. Each VM has an operating system and acts as an independent computer even though it runs on a portion of the physical device.

A virtual machine mimics all components of a computer, including:

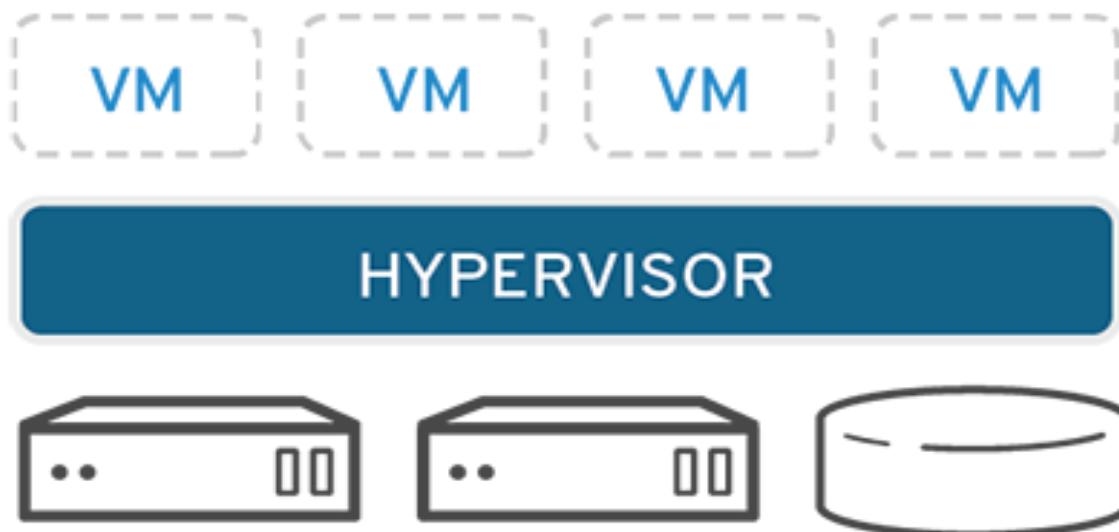
- CPU.
- RAM.
- Storage.
- Networks.

With virtualization, a piece of hardware can host numerous VM configurations simultaneously without performance issues. The main benefits of virtualization are:

- More computing capabilities with fewer resources.
- Running multiple independent systems on a single hardware.
- Consistent environments throughout the continuous integration and continuous delivery (CI/CD) process.

How does virtualization work?

- Software called hypervisors separate the physical resources from the virtual environments—the things that need those resources. Hypervisors can sit on top of an operating system (like on a laptop) or be installed directly onto hardware (like a server), which is how most enterprises virtualize. Hypervisors take your physical resources and divide them up so that virtual environments can use them.



Different Types of Virtualization

There are four primary types of virtualization:

- [Server virtualization.](#)
- Network virtualization.
- Desktop virtualization.
- Operating system virtualization.

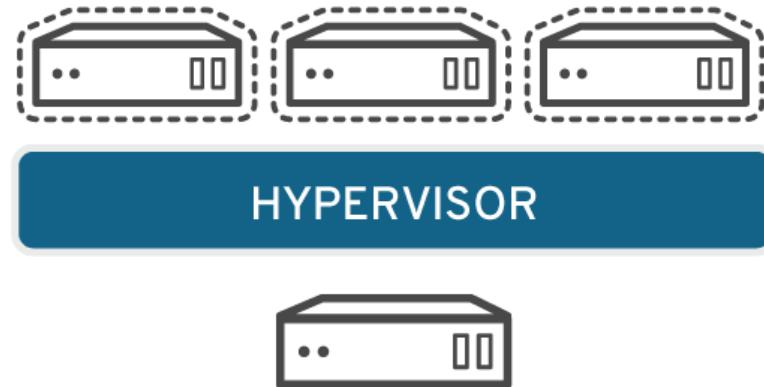
Server Virtualization

Server virtualization enables a single physical server to perform multiple independent functions. This form of virtualization leads to:

- Reduced operating costs.
- Increased server performance.
- Faster workload capacity.

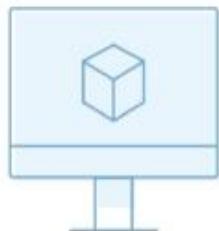
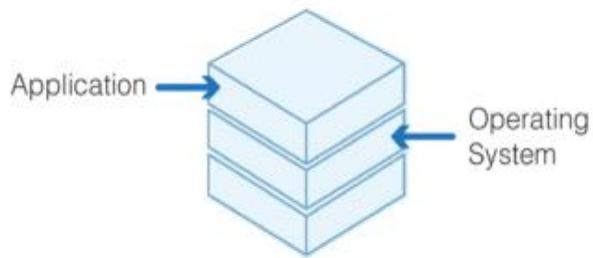
Server virtualization also lowers managing costs and physical server complexity.

Servers are computers designed to process a high volume of specific tasks really well so other computers—like laptops and desktops—can do a variety of other tasks. Virtualizing a server lets it do more of those specific functions and involves partitioning it so that the components can be used to serve multiple functions.

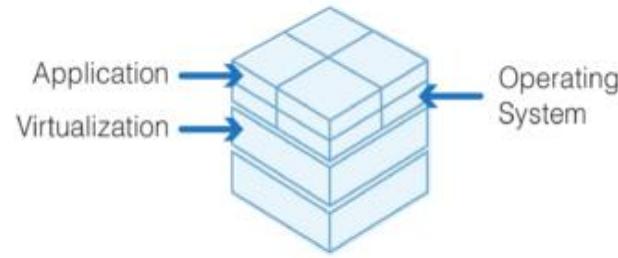


What is Server Virtualization?

Traditional server architecture

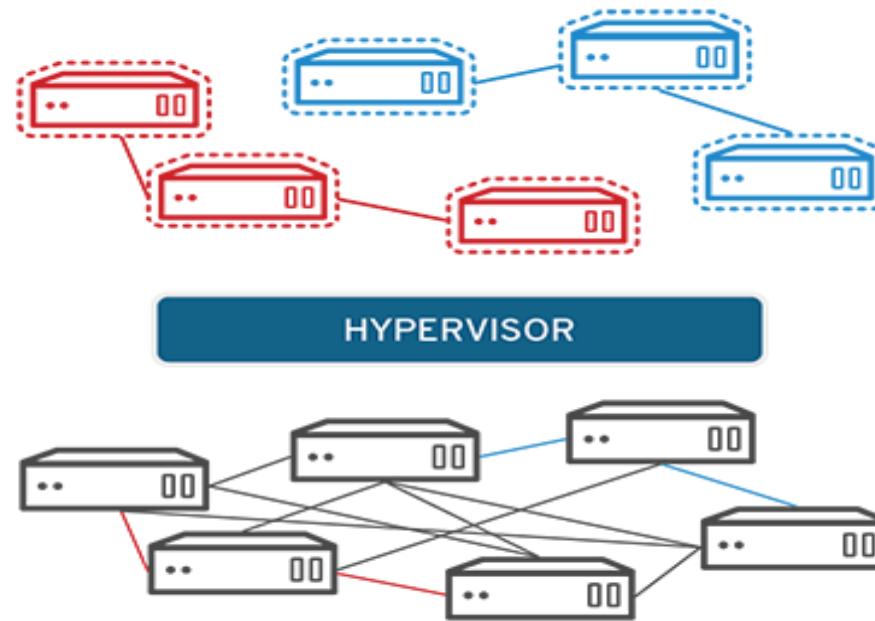


Virtualized server architecture



Network Virtualization

Network virtualization mimics a network in a virtual environment. This virtualization process distributes network functions (directory services, file sharing, IP configurations) among virtual environments.

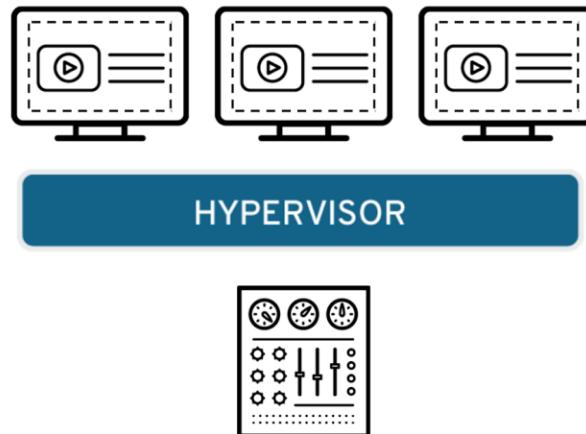


Desktop Virtualization

Desktop virtualization creates a virtual environment that imitates the settings and applications of a desktop device.

This virtualization form allows an administrator (either a person or a tool) to deploy desktop environments to multiple physical machines. Admins can perform mass configurations, updates, and security checks on all virtual desktops simultaneously.

[Virtual desktop infrastructure](#) is ideal for providing a secure, centralized work environment accessible from any device.

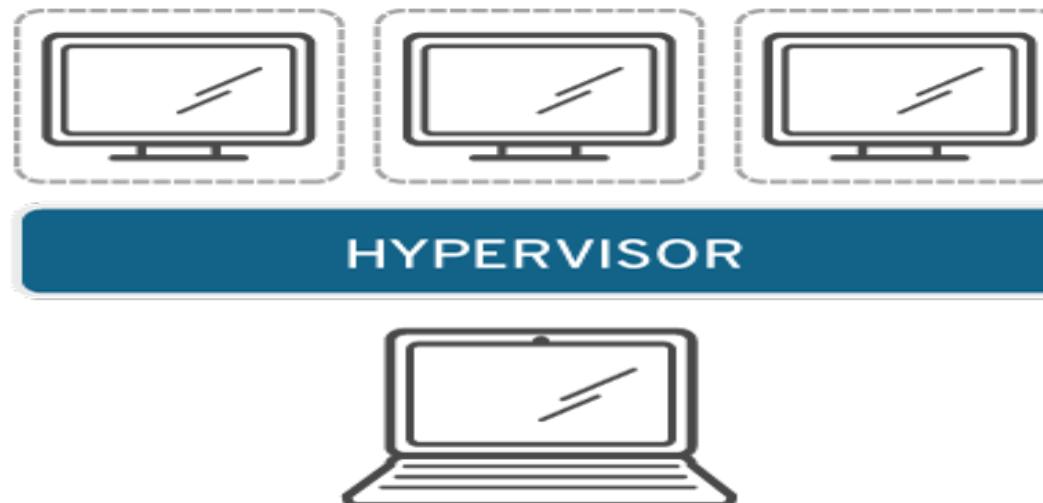


Operating System Virtualization

Operating system virtualization allows a developer to deploy multiple operating systems on a single machine. This virtualization type helps a team to:

- Reduce bulk hardware costs.
- Test an app in multiple operating systems on a single machine.
- Increase security due to the isolation of virtual instances (e.g., test unapproved software solutions).
- Speed up IT services.

Operating system virtualization is a popular option when a team needs to run Linux and Windows environments on a single device.



virtualization stack

A virtualization stack is a group of software components used to support a virtual environment. Items include the management console, virtual machine processes, emulated devices, management services and the user interface combined with the hypervisor.

- **Hardware Virtualization:**
- **Network Virtualization:**
- **Desktop Virtualization:**
- **Storage Virtualization:**
- **Application Virtualization:**
- **Server Virtualization:**

Hardware Virtualization

It may be considered as the most common type of virtualization these days. **The best example of hardware virtualization is a Virtual Machine.** A virtual machine works and looks like a real system with the same or a different operating system.

- **Network Virtualization:**

Network virtualization is a process in which **a combination of software and hardware network resources form a single software network**, which is commonly known as Virtual Network. Also, the available bandwidth is divided into several independent channels, which can be used by real devices and servers.

- **Desktop Virtualization:**

In the case of desktop virtualization, the logical or virtual desktop is separate from the physical desktop. Here, instead of accessing the desktop using the computer hardware like keyboard, mouse of the system, the desktop is located remotely from another system by using a network connection.

- The network can be a wired/wireless LAN or the internet. So, the user can access their files from any system without physically operating the order that contains the data.

- **Storage Virtualization:**

In this case, a combination of several storage disks forms a storage pool or group. These groups are virtual storage units. These can then be assigned to servers for use. Logical volumes are one of the examples of storage virtualization, which represent the storage as a coherent unit rather than a physical unit.

- **Application Virtualization:**

- In application virtualization, applications are virtualized and encapsulated. Virtual applications are not installed like traditional applications but are used as they are installed.

- **Server Virtualization:**

This type of virtualization comes in handy when we need to run a single physical server on multiple operating systems simultaneously. With this process, the performance, capacity and efficiency of the server are increased, while managing costs and complexity are reduced.

The value of virtualization in DevOps.

(how virtualization enables engineers to set up flexible and consistent systems throughout the software development life cycle (SDLC).)

The goal of DevOps is to improve the speed and quality of software development. Traditional teams must evolve to reach faster deployments, and embracing virtualization is a major part of the DevOps transition.

Virtualization offers the consistency and agility a team requires to make the most out of modern development.

Role of Virtualization in DevOps

- Virtualization plays a vital role in devops.
- DevOps, automates various software development processes, including **testing and delivery**. With the help of virtualization, the devops teams can develop and test within **virtual and simulated** environments using similar devices and systems to the end-users.
- This way, the development and testing become **more efficient and less time-consuming**.
- Virtual live environments can also be provided to test the software at the deployment level.
- This helps in real-time testing, as the team can check the effect of every new change made to the software. By doing these tasks in virtualized environments, the amount of computing resources is reduced.
- This real-time testing helps in increasing the quality of the product. Working with a virtual environment reduces the time for **retesting and rebuilding the software for production**.
- Thus, the virtualization reduces the extra efforts for the devops team, while ensuring **faster and reliable delivery**.

Advantages of Virtualization in Devops

- There are many perks of virtualization in Devops; some of these are:

- The workload is reduced**

The providers of virtualization continuously update the hardware and software used for virtualization, so there is no need to do these updates locally. The IT staff of a company can focus on other important things and save time and cost for the organization.

- Testing Environment**

With the help of virtualization, we can set up a local testing environment. This environment can be used for various kinds of testing for software. Even if a server crashes, there won't be any data loss. So, the reliability is increased, and the software can be tested on this virtual environment until it is ready for live deployment.

- Energy-saving**

Virtualization saves energy as instead of using local software or servers; the virtualization takes place with the help of virtual machines, which lowers the power or energy utilization. By saving this energy, the cost is reduced, and this saved money can be used for other useful operations. **Improving Hardware utilization**

With virtualization, the need for physical systems decreases. Thus, maintenance costs and power utilization is reduced. The use of CPU and memory is improved.

- Improving Hardware utilization**

With virtualization, the need for physical systems decreases. Thus, maintenance costs and power utilization is reduced. The use of CPU and memory is improved.

Deployment systems

What is Software Deployment?

Software Deployment Meaning: Software deployment includes all of the steps, processes, and activities that are required to make a software system or update available to its intended users.

- Today, most IT organizations and software developers deploy software updates, patches and new applications with a combination of manual and automated processes.
- Some of the most common activities of software deployment include software release, installation, testing, deployment, and performance monitoring.
- **Software deployment** refers to the process of running an application on a server or device. A software update or application may be deployed to a test server, a testing machine, or into the live environment, and it may be deployed several times during the development process to verify its proper functioning and check for errors.
- Another example of software deployment could be when a user downloads a mobile application from the App Store and installs it onto their mobile device.
- To summarize, a software release is a specific version of a code and its dependencies that are made available for deployment. Software deployment refers to the process of making the application work on a target device, whether it be a test server, production environment or a user's computer or mobile device.

Why is Software Deployment Important?

- Software deployment is one of the most important aspects of the software development process.
- Deployment is the mechanism through which applications, modules, updates, and patches are delivered from developers to users.
- The methods used by developers to build, test and deploy new code will impact how fast a product can respond to changes in customer preferences or requirements and the quality of each change.
- Software development teams that streamline the process of building, testing and deploying new code can respond more quickly to customer demand with new updates and deliver new features more frequently to drive customer satisfaction, satisfy user needs and take advantage of economic opportunities.

What is the Software Deployment Process?

Every organization must develop its own process for software deployment, either basing it on an existing framework of best practices or customizing a process that meets relevant business objectives. **Software deployment can be summarized in three general phases: preparation, testing and the deployment itself.**

Preparation

In the preparation stage, developers must gather all of the code that will be deployed along with any other libraries, configuration files, or resources needed for the application to function. Together, these items can be packaged as a single software release. Developers should also verify that the host server is correctly configured and running smoothly.

Testing

Before an update can be pushed to the live environment, it should be deployed to a test server where it can be subjected to a pre-configured set of automated tests. Developers should review the results and correct any bugs or errors before deploying the update to the live environment.

Deployment

Once an update has been fully tested, it can be deployed to the live environment. Developers may run a set of scripts to update relevant databases before changes can go live. The final step is to check for bugs or errors that occur on the live server to ensure the best possible customer experience for users interacting with the new update.

UNIT-5

Code monitoring and Issue Tracking

Code monitoring tools:

What is Continuous code Monitoring?

- Continuous code Monitoring is all about the ability of an organization to detect, react, report, respond, contain and mitigate(**reducing risk of loss from the occurrence of any undesirable event**) the attacks that occur, in its infrastructure.
- Once the application is deployed into the server, the role of continuous monitoring comes in to play. The entire process is all about taking care of the company's infrastructure and respond appropriately.

Why We Need Continuous Monitoring?

Continuous Monitoring Tools resolve any system errors (low memory, unreachable server etc.) before they have any negative impact on your business productivity.

- ✓ It detects any network or server problems
- ✓ It determines the root cause of any issues
- ✓ It maintains the security and availability of the service .It monitors and troubleshoot server performance issues
- ✓ It allows us to plan for infrastructure upgrades before outdated systems cause failures
- ✓ It can respond to issues at the first sign of a problem
- ✓ It can be used to automatically fix problems when they are detected
- ✓ It ensures IT infrastructure outages have a minimal effect on your organization's bottom line.
It can monitor your entire infrastructure and business processes

✓ From past years our security professionals are performing static analysis from – system log, firewall logs, IDS logs, IPS logs etc. But, it did not provide proper analysis and response. Today's Continuous Monitoring approaches which are automated tools gives us the ability to aggregate all of the events discussed above, co-relate them, compare them and then estimate the organization's risk posture.

system log

Console (443 messages)

Now Activities Clear Reload Info Share MESSAGE TYPE ▾ error MESSAGE TYPE

All Messages Errors and Faults

Devices	Type	Time	Process	Message
Chris's MacBoo...	Yellow	14:29:17.446211	Script...	Couldn't set launchd job stati...
Reports	Yellow	14:29:17.446381	Script...	Couldn't update launchd job: ...
Mac Analytics...	Red	14:29:18.738595	quickl...	Failed to generate image with ...
System Reports	Red	14:29:18.843148	quickl...	Failed to generate image with ...
User Reports	Yellow	14:29:26.557057	Window...	CGXRegisterForPortNotification

quicklookd (IconServices)
Subsystem: com.apple.iconservices Category: default Details 2018-01-11 14:29:18 +0000

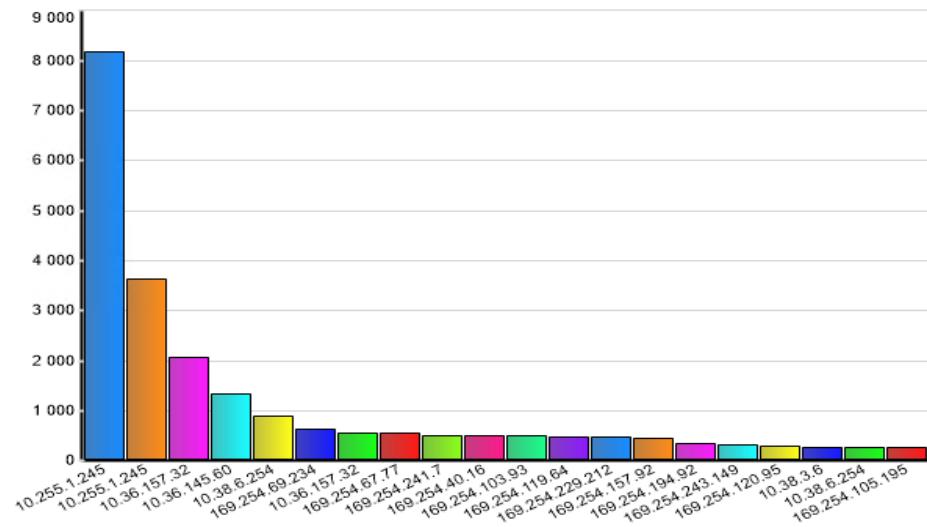
Windows Firewall Logs

Contents

- Windows Firewall Network Analysis
 - > Top 20 IP Address Communications - Overview
 - > Top 5 Destination Subnets (By Size) - Overview
 - Top 5 Destination Subnets (By Size) - (Top 5 by Size)
 - + External
 - + Internal
 - > Top 5 Source Subnets (By Size) - Overview
 - Top 5 Source Subnets (By Size)
 - Internal
 - > Source Addresses (Top 20) - Overview
 - External
 - > Source Addresses (Top 20) - Overview

1. Top 20 IP Address Communications - Overview

Chart of Source Address, Destination Address, Destination Port versus Hits:



#	Source Address	Destination Address	Destination Port	Hits
1	10.255.1.245	239.255.255.250	Unknown	8 182
2	10.255.1.245	224.0.0.252	hostmon	3 629
3	10.36.157.32	239.255.255.250	Unknown	2 066
4	10.36.145.60	10.38.25.45	Unknown	1 334
5	10.38.6.254	224.0.0.252	hostmon	880
6	169.254.69.234	224.0.0.252	hostmon	606
7	10.36.157.32	224.0.0.252	hostmon	541
8	169.254.67.77	224.0.0.252	hostmon	534
9	169.254.241.7	224.0.0.252	hostmon	495
10	169.254.40.16	224.0.0.252	hostmon	477
11	169.254.103.93	224.0.0.252	hostmon	471
12	169.254.119.64	224.0.0.252	hostmon	462
13	169.254.229.212	224.0.0.252	hostmon	455
14	169.254.157.92	224.0.0.252	hostmon	436

IDS logs (Intrusion Detection System logs)

 **Security Event Manager** Events Nodes Rules SEM CONSOLE  

Rules

REFINE RESULTS    

 **Name**  Search... 

Availability

-  Enabled
-  Disabled

Test Mode

- On
- Off

Created

Last Modified

Tag category

-  Activity Types
-  Authentication
-  Change Management
-  Compliance
-  Devices
-  Endpoint Monitoring

Rule Name	Description	Tags	Score	Status	Actions
Critical Account Logon Failures	Logon Failures to Administrative Accounts	General B..., General B..., Server O...	+13		
Detach Unauthorized USB Device	You will first want to populate the Authorized USB Devices user defined group with your approv...	USB Devi..., Workstat..., Server O...	+10		
File Audit Failure with Restricted Information Inferen	HINT: File auditing must be turned on for the file in question (it is not turned on by default).	File Auditi..., General B..., General B...	+4		
IPSec Failure	Monitors failed certificate negotiations (usually VPN related)	Network, Remote A..., VPN and ...	+5		
MSSQL Critical Account Logon Failure	NOTE: You may also use Directory Services integration and an appropriate group within your do...	General B..., General B..., Database...	+13		
MSSQL Database Change Attempt	Note: You must be using a TriGeo database auditing utility (such as MSSQL Auditor) in order to c...	Database..., Device Ch..., Administr...	+11		
MSSQL Service Shutdown	Note: You must be using a TriGeo database auditing utility (such as MSSQL Auditor) in order to c...	Database..., Database..., Windows	+1		

- ✓ Continuous code Monitoring, sometimes called Continuous Control Monitoring (CCM), is an automated process by which DevOps personnel can observe and detect compliance issues and security threats during each phase of the DevOps pipeline
- ✓ So, Continuous code Monitoring is an automated process that leverages specialized software tools to empower DevOps teams with enhanced visibility of application performance, security threats, and compliance concerns across the entire DevOps pipeline.
- ✓ Continuous code Monitoring comes in at the end of the DevOps pipeline. Once the software is released into production, Continuous Monitoring will notify dev and QA teams in the event of specific issues arising in the prod environment. It provides feedback on what is going wrong, which allows the relevant people to work on necessary fixes as soon as possible.
- ✓ Continuous monitoring (CM) tools are a critical component of the DevOps pipeline, providing automated capabilities that allow developers to effectively monitor applications, infrastructure, and network components in the production environment.

- ✓ It also helps teams or organizations monitor, detect, study key relevant metrics, and find ways to resolve said issues in real-time.
- ✓ We can say Continuous Monitoring basically assists IT organizations, DevOps teams in particular, with procuring real-time data from public and hybrid environments.
- ✓ This is especially helpful with implementing and fortifying various security measures –
incident response, threat assessment, computers, and database forensics, and root cause analysis.
- ✓ It also helps provide general feedback on the overall health of the IT setup, including deployed software.

Goals of Continuous Monitoring in DevOps

- Enhance transparency and visibility of IT and network operations, especially those that can trigger a security breach, and resolve it with a well-timed alert system.
- Help monitor software operation, especially **performance issues**, identify the cause of the error, and apply appropriate solutions before significant damage to uptime and revenue.
- Help **track user behavior**, especially right after an update to a particular site or app has been pushed to prod. This monitors if the update has a positive, negative, or neutral effect on user experience.

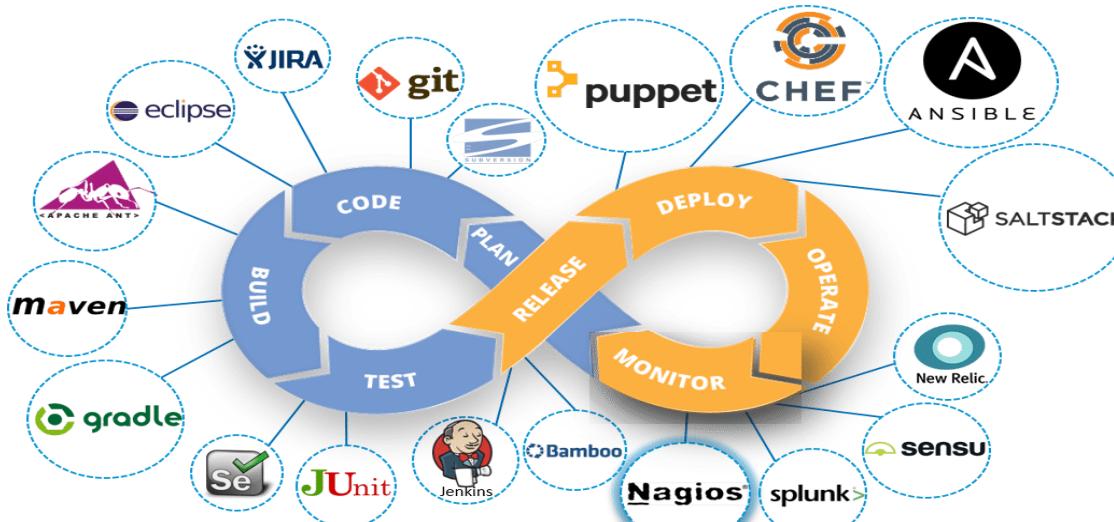
Types of Continuous Monitoring

- **Infrastructure Monitoring:** Monitors and manages the IT infrastructure required to deliver products and services. This includes data centres, networks, hardware, software, servers, storage, and the like. Infrastructure Monitoring collates and examines data from the IT ecosystem to improve product performance as far as possible.

Infrastructure Monitoring: Tool must monitor

- ✓ Server Availability
- ✓ CPU & Disk Usage
- ✓ Server & System Uptime
- ✓ Response Time to Errors
- ✓ Storage
- ✓ Database Health
- ✓ Storage
- ✓ Security
- ✓ User permissions
- ✓ Network switches
- ✓ Process level usage
- ✓ Relevant performance trends

- **Application Monitoring:** Monitors the performance of released software based on metrics like uptime, transaction time and volume, system responses, API responses, and general stability of the back-end and front-end.
- **Network Monitoring:** Monitors and tracks network activity, including the status and functioning of firewalls, routers, switches, servers, Virtual Machines, etc. Network Monitoring detects possible and present issues and alerts the relevant personnel. Its primary goal is to prevent network downtime and crashes.



Application Monitoring: Tool must monitor:

- ✓ availability
- ✓ error rate
- ✓ throughput
- ✓ user response time
- ✓ pages with low load speed
- ✓ third-party resource speed
- ✓ browser speed
- ✓ end-user transactions
- ✓ SLA status

Network Monitoring: Tool must monitor:

- ✓ Latency
- ✓ Multiple port level metrics
- ✓ Server bandwidth
- ✓ CPU use of hosts
- ✓ Network packets flow

Apart from these, following things also important to monitor in Continues monitoring...

- Server status and health
- Application performance log
- System vulnerabilities
- Development milestones
- User activity/behavior

❑ What is Nagios?

- Nagios is an open source software for continuous monitoring of systems, , applications. networks, infrastructures, services, and business processes etc , in a DevOps culture.
- In the event of a failure, Nagios can alert technical staff of the problem, allowing them to begin remediation processes before outages affect business processes, end-users, or customers. With Nagios, you don't have to explain why an unseen infrastructure outage affect your organization's bottom line.
- It runs plugins stored on a server which is connected with a host or another server on your network or the Internet. In case of any failure, Nagios alerts about the issues so that the technical team can perform recovery process immediately.

Nagios software runs periodic checks on critical parameters of application, network and server resources.

For example, Nagios can monitor memory usage, disk usage, microprocessor load, the number of currently running processes and log files.

Nagios also can monitor services, such as Simple Mail Transfer Protocol (SMTP), Post Office Protocol 3 (POP3), Hypertext Transfer Protocol (HTTP) and other common network protocols.

SMTP, which stands for **Simple Mail Transfer Protocol**, is an email protocol used for sending email messages from one email account to another via the internet.

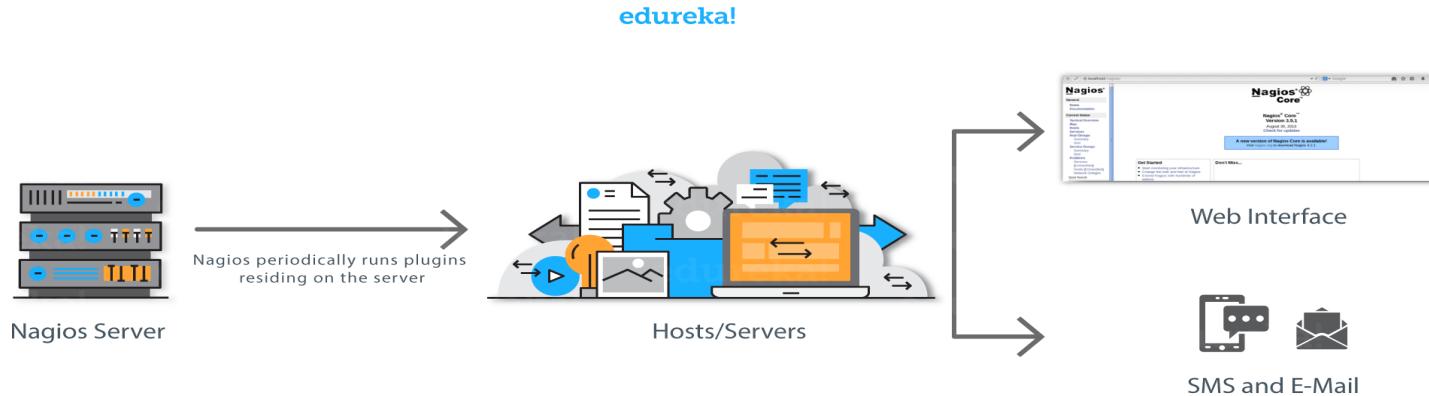
Post Office Protocol 3, or **POP3**, is the most commonly used protocol for receiving email over the internet.

HTTP is the standard 'language' used to communicate between web browsers and website servers.

Why We Need Nagios tool?

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

□ How Nagios works: a basic workflow diagram

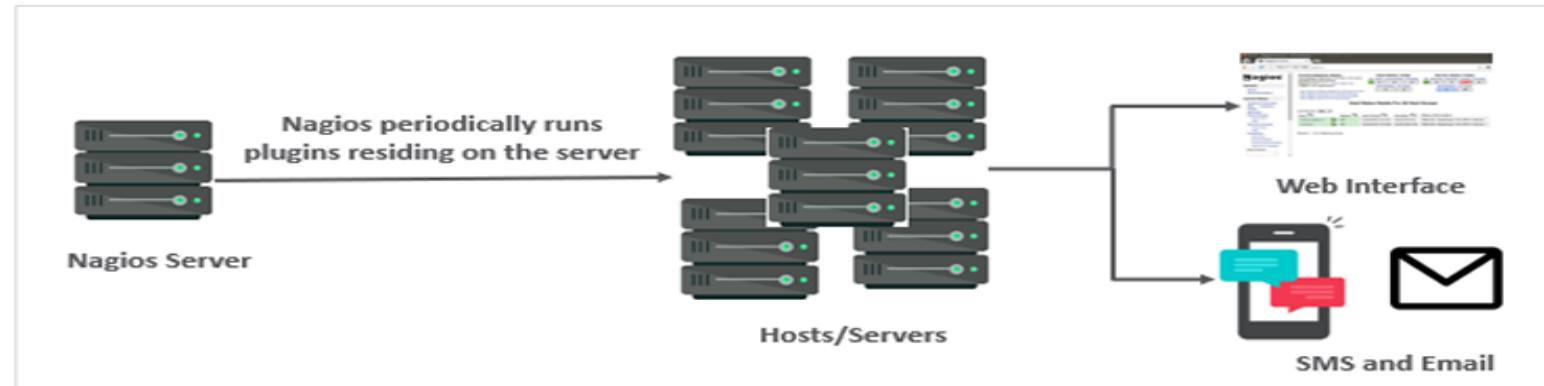


- Nagios runs on a server, **usually as a daemon or a service**.
- It periodically runs plugins residing on the same server, they contact hosts or servers on your network or on the internet. One can view the status information using the web interface. You can also receive email or SMS notifications if something happens.
- The **Nagios daemon behaves like a scheduler that runs certain scripts at certain moments. It stores the results of those scripts and will run other scripts if these results change.**

Nagios Architecture

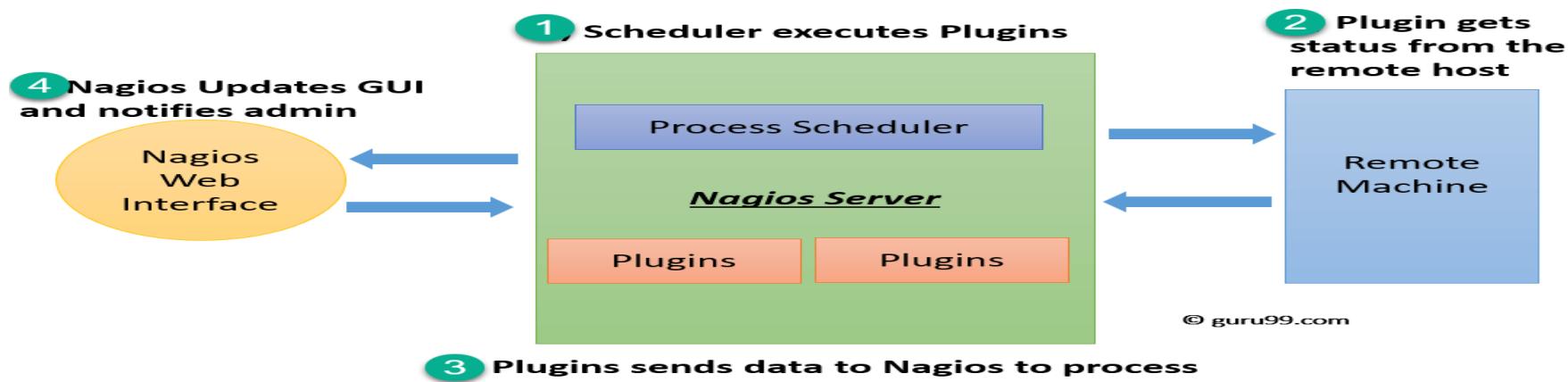
The following points are worth notable about Nagios architecture:

- Nagios has server-agent architecture.
- Nagios server is installed on the host and plugins are installed on the remote hosts/servers which are to be monitored.
- Nagios sends a signal through a process scheduler to run the plugins on the local/remote hosts/servers.
- Plugins collect the data (CPU usage, memory usage etc.) and sends it back to the scheduler.
- Then the process schedules send the notifications to the admin/s and updates Nagios GUI.



The following figure shows Nagios Server Architecture in detail:

- ❑ **Nagios process scheduler**:-The scheduler is a component of server part of Nagios. It sends a signal to execute the plugins at the remote host.
- ❑ **Plugins**: These are compiled executables or scripts (Perl scripts, shell scripts, etc.) that can be run from a command line to check the status of a host or service. Nagios uses the results from the plugins to determine the current status of the hosts and services on your network.
 - The plugin gets the status from the remote host
 - The plugin sends the data to the process scheduler
 - The process scheduler updates the GUI and notifications are sent to admins



- Nagios offers the following benefits for the users:

- ✓ It helps in getting rid of **periodic testing**.
- ✓ It detects **instantaneous failures**
- ✓ It reduces maintenance cost **without sacrificing performance**.
- ✓ It provides **timely notification** to the management of **control and breakdown**.

munin

- You are a system administrator, managing a set of servers. As a part of your job, you need to ensure that all the servers are up and running 24X7, they are resources are utilized optimally. And that no single server is overloaded or consuming huge memory or some similar abnormal behaviour over a time period. Obviously, you will be needing a monitoring tool. Munin is one such tool, which helps system administrators to keep track of system resources.
- **Munin** is a open-source ,web based computer system monitoring, network monitoring and infrastructure monitoring tool.
- It is designed around a client-server architecture and can be configured to monitor the machine it's installed on (the *Munin master*) and any number of client machines, which in Munin parlance, are called *Munin nodes*.
- Munin is written in Perl. Its emphasis is on plug and play capabilities. About 500 monitoring plugins are currently available.

Munin is a system, network, and infrastructure monitoring application that provides monitoring information in the graphical from with the help of graphs.

Munin uses RRD tool(*round-robin database tool*) to create graphs.

aims to handle time series data such
as network bandwidth, temperatures or CPU load.

- munin shows network usage of servers and services in graphical form using **RRDtool**. With the help of Munin you can monitor the performance of your systems, networks, SANs's(SANS stands for **SysAdmin, Audit, Network, and Security**) and applications.
- With Munin, real-time information about the servers, applications, services, SANs, network, and many more resources is available in graphs through a web interface.
- And Munin is easy to set up and deploy with its plug-and-play capabilities(A user can add and remove devices without having to do manual configuration, and without knowledge of computer hardware.) and a popular monitoring tool for web servers and hosts.

❖ How does it work?

Munin the monitoring tool surveys all your computers and remembers what it saw.

It presents all the information in graphs through a web interface. Its emphasis is on plug and play capabilities.

After completing an installation, a high number of monitoring plugins will be playing with no more effort.

Using Munin you can easily monitor the performance of your computers, networks, SANs, applications.

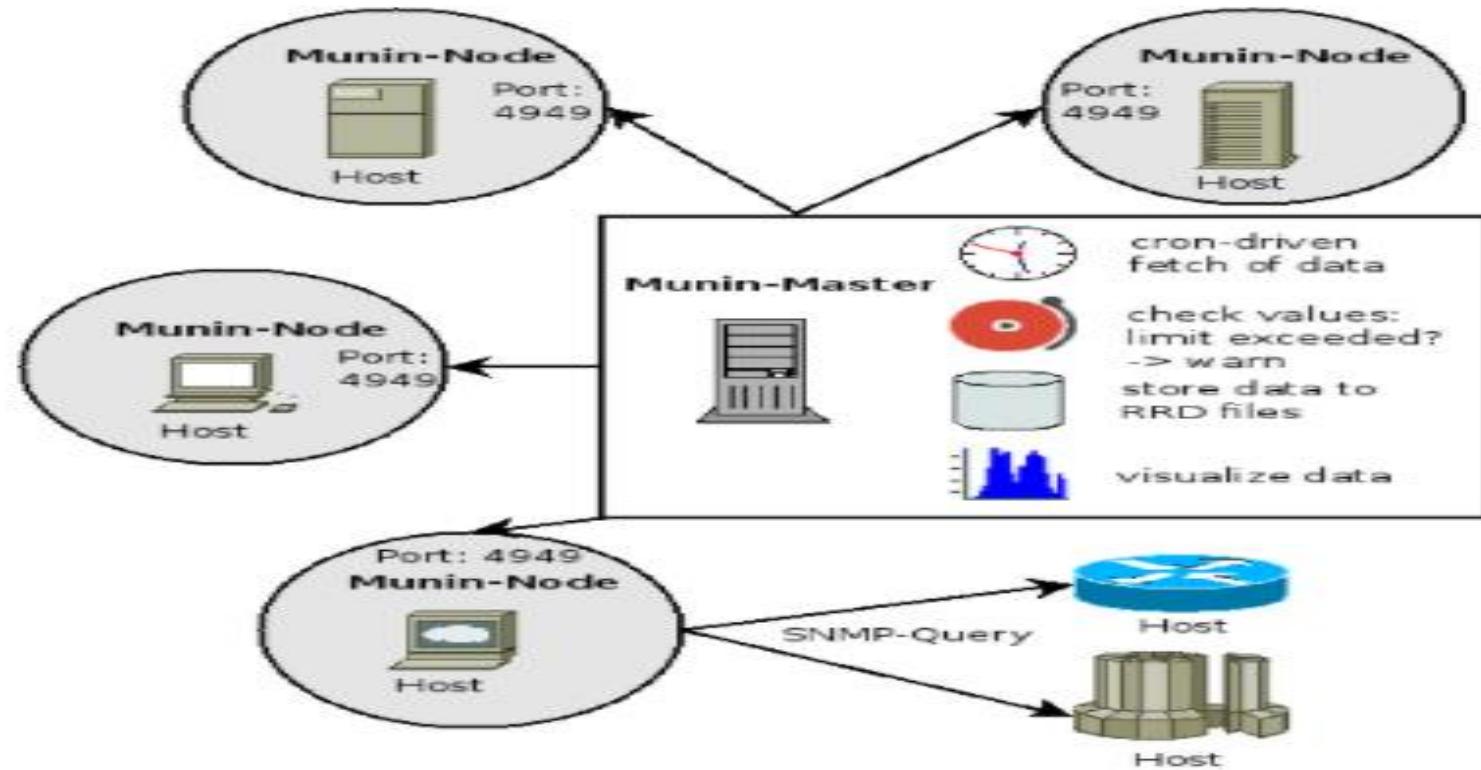
It makes it easy to determine "what's different today" when a performance problem crops up. It makes it easy to see how you're doing capacity-wise on any resources.

Munin uses the excellent RRD Tool and the framework is written in Perl, while plugins may be written in any language.

Munin has a master/node architecture in which the master connects to all the nodes at regular intervals and asks them for data.

It then stores the data in RRD files, and (if needed) updates the graphs.

- This monitoring tool uses Master-node architecture. Master periodically pulls data from all the configured nodes and processes it to create graphical dashboards.



Work flow

Munin derives its name from **mythology**. It means **memory**. This monitoring software memorizes **what it sees** and hence the name.

This monitoring tool uses **Master-node architecture**. Master periodically pulls data from all the configured nodes and processes it to create graphical dashboards.

One of the advantages of Munin is its vast plugin eco-system. There are several hundred plugins available, which make it a powerful monitoring system.

Munin itself uses **RRDTool**, which is written in Perl, whereas plugins could be written in any language.

Once data is collected from nodes, **it is processed by the master and various graphs** are made available on the dashboard.

One of the components is – **Munin-limits** – which monitors the values between configured “ok”, “warn” and “crit” levels and can create a notification for the administrators.

Munin actually consists of two parts, namely :

- 1.Munin Master and
2. Munin node.

1.Munin master or Server

- Munin master or Server is the data gatherer and grapher which also hosts the software website.
The munin master is responsible for gathering data from Munin nodes
- master connects to each node regularly and pulls the data from them. i.e.,The Munin Server will periodically poll all the nodes in your network, it's aware of for data, which is, in turn, will use to create graphs and HTML pages for viewing via a web browser.
- It then uses RRDtool to log and generate updated graphs.
- It stores this data in RRD, files, and graphs them on request. It also checks whether the fetched values fell below or go over **specific thresholds (warning, critical)** and will send alerts if this happens and the administrator configured it to do so.

2.Munin node.

With the Munin master server, we can now configure the Munin nodes.

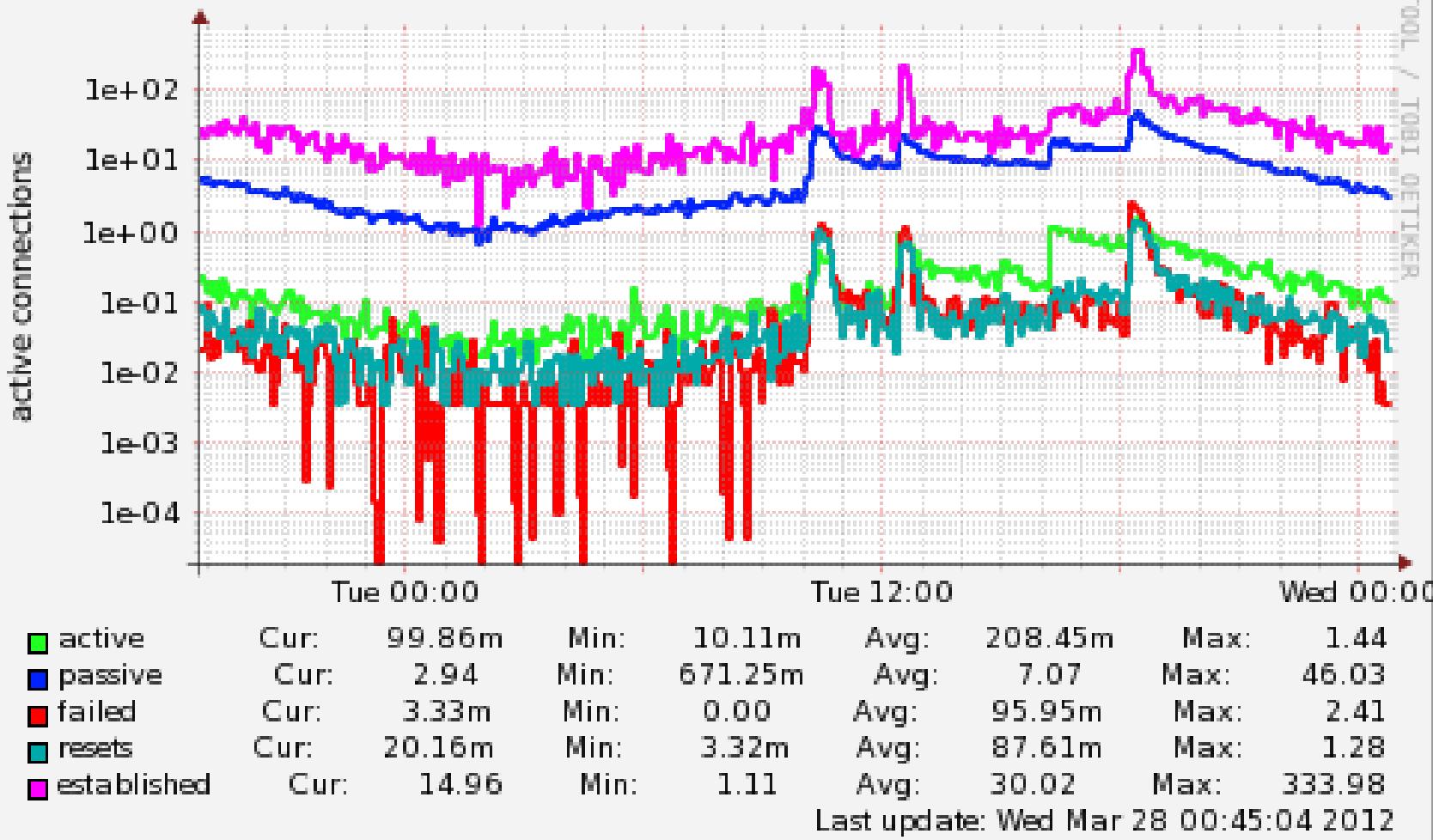
These have an agent on them, called **munin-node**.

Munin Node is the agent that monitors and extracts data from the node it runs on and then passes the data to the server for further processing.so, munin node can be called as a data collector for munin. Then the master uses this data to gather the information and present to the user.

3.Plugins

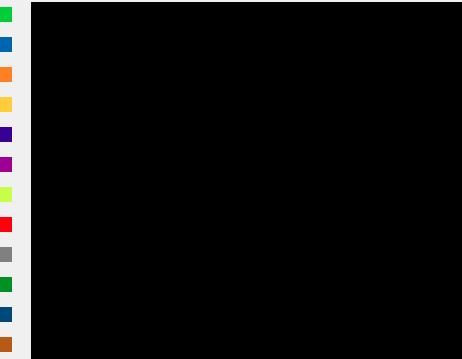
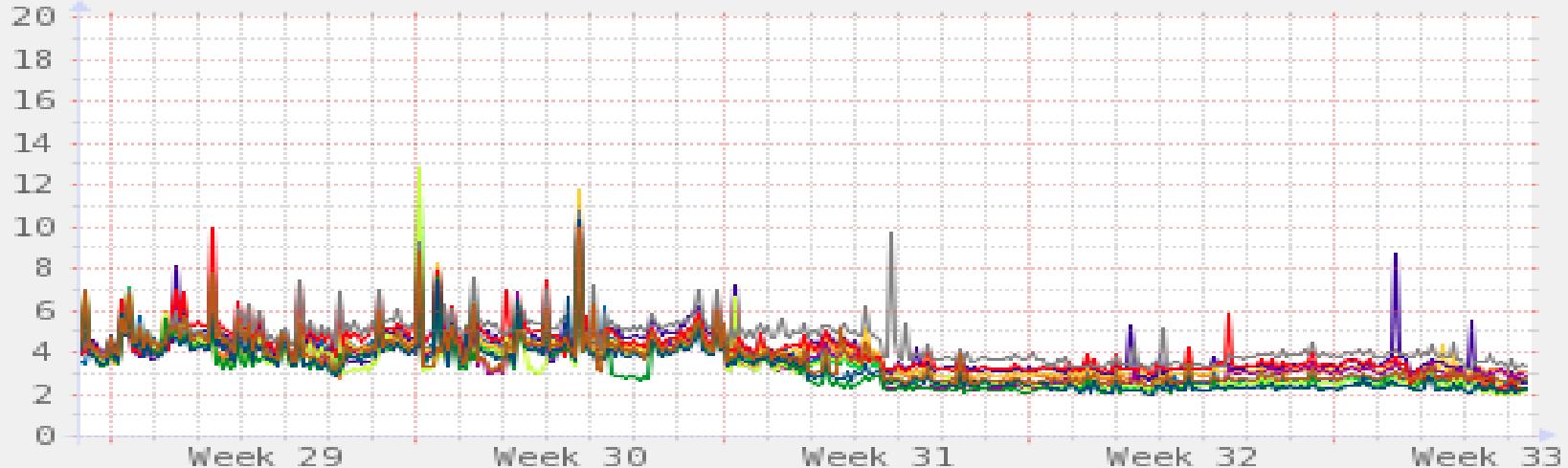
Plugins are small, standalone executables. They are easy to write, and can be written in any language.

Netstat - by day



wget loadtime of webpages - by month

Load time in seconds

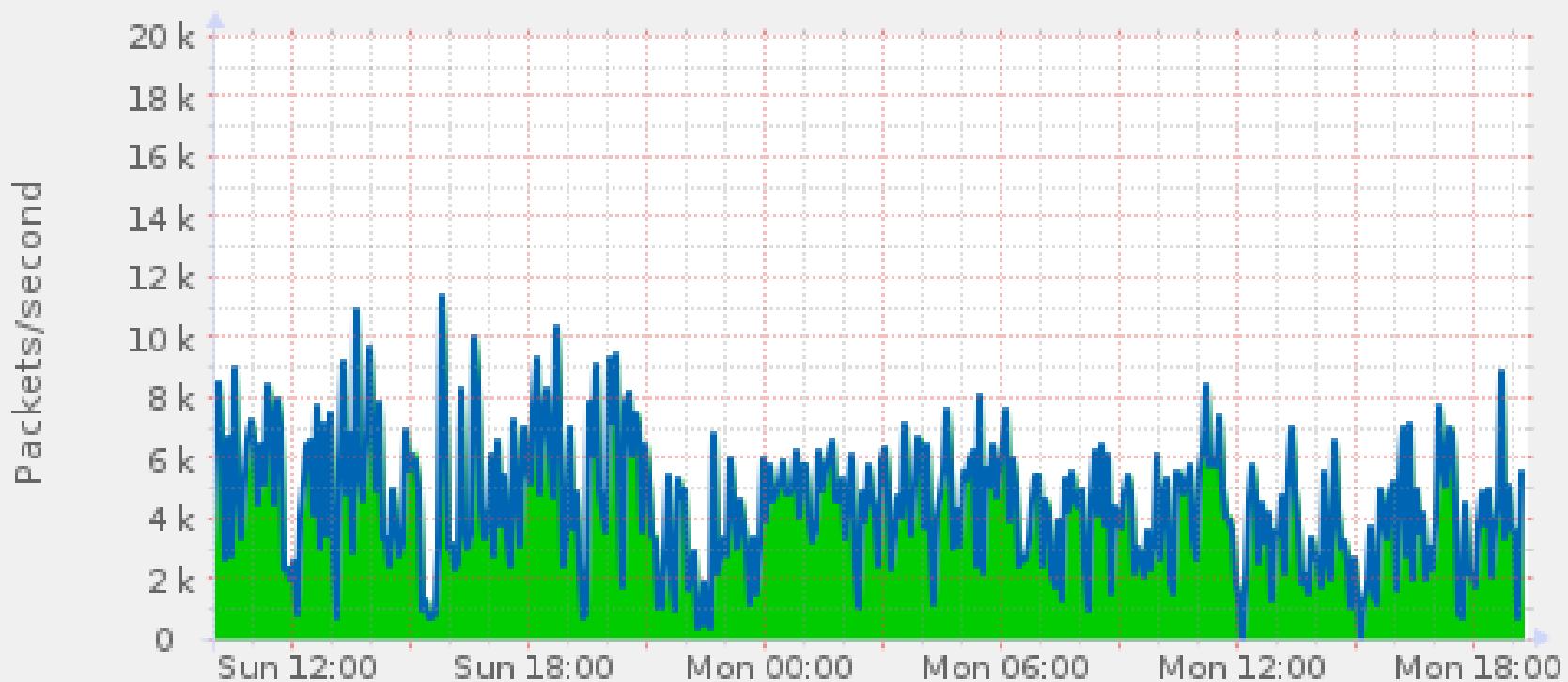


Cur:	Min:	Avg:	Max:
2.62	60.83m	3.58	95.34
2.60	131.78m	3.71	58.98
3.08	60.56m	3.79	80.95
2.53	122.32m	3.86	58.90
2.50	60.72m	4.01	74.59
2.88	61.21m	3.55	81.41
2.13	60.62m	3.37	116.63
2.25	123.07m	4.12	63.42
3.33	137.45m	4.54	103.52
2.22	60.70m	3.16	104.06
2.34	113.48m	3.29	63.48
2.69	60.65m	3.60	70.57

Munin 2.0.18

Last update: Fri Aug 16 12:45:07 2013

Firewall Throughput - by day



■ Received
■ Forwarded

	Cur:	Min:	Avg:	Max:
Received	5.68k	85.07	4.52k	11.45k
Forwarded	5.64k	53.65	4.49k	11.40k

Last update: Mon Nov 8 19:15:03 2021

- With the size of an organisation's infrastructure increasing, monitoring is becoming a challenge.
- Monitoring multiple data centres is becoming a difficult task for sys admins, and it certainly starts to become tiresome(impatient) as the number of nodes increase rapidly. System administrators would typically want to check systems statistics like the disk, CPU and network utilisation, etc.
- They would be interested in knowing whether systems are performing within thresholds, where the bottlenecks are occurring, and how applications are performing. Now, when managing a few systems, you could probably log in to each system and check —it becomes a problem as infrastructure grows.
- Ganglia presents itself as a very good solution when it comes to cluster-based monitoring, and analysing the available data.

ganglia is an open-source scalable distributed monitoring system for high-performance computing systems such as clusters and Grids.

Ganglia allows you to gather important information about machines where it is applied.

It allows the user to remotely view live or historical statistics (such as CPU load averages or network utilization) for all machines that are being monitored.

Ganglia is currently in use on thousands of clusters around the world and can scale to handle clusters with several thousand of nodes.

- Here is a small glossary of terms used in Ganglia:
 - Host/node: Typically, a machine.
 - Cluster: A group of nodes.
 - Grid: A group of clusters.
 - Metrics: The graphs that are displayed.
 - RRDs or Round Robin Databases: RRDtool (the Round Robin Database tool) is a system to store and display time-series data (e.g., network bandwidth, machine-room temperature, server load average, etc). It stores the data very compactly to maintain a manageable archive size. RRDtool presents useful graphs by processing the data to enforce a certain data density.
- It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualisation.”
- It is carefully engineered to achieve very low per-node overheads and high concurrency.

Ganglia?

Ganglia - A real-time cluster monitoring tool that collects information from each computer in the cluster and provides an interactive way to view the performance of the computers and cluster as a whole.

Ganglia like other monitoring tools only provide a way to view but not control the performance of each computer.

Cluster?

A cluster is a collection of computers which work together in accomplishing a task.

Cluster computing has become a practical choice for high-performance computing (HPC) deployment.

❖ Ganglia architecture

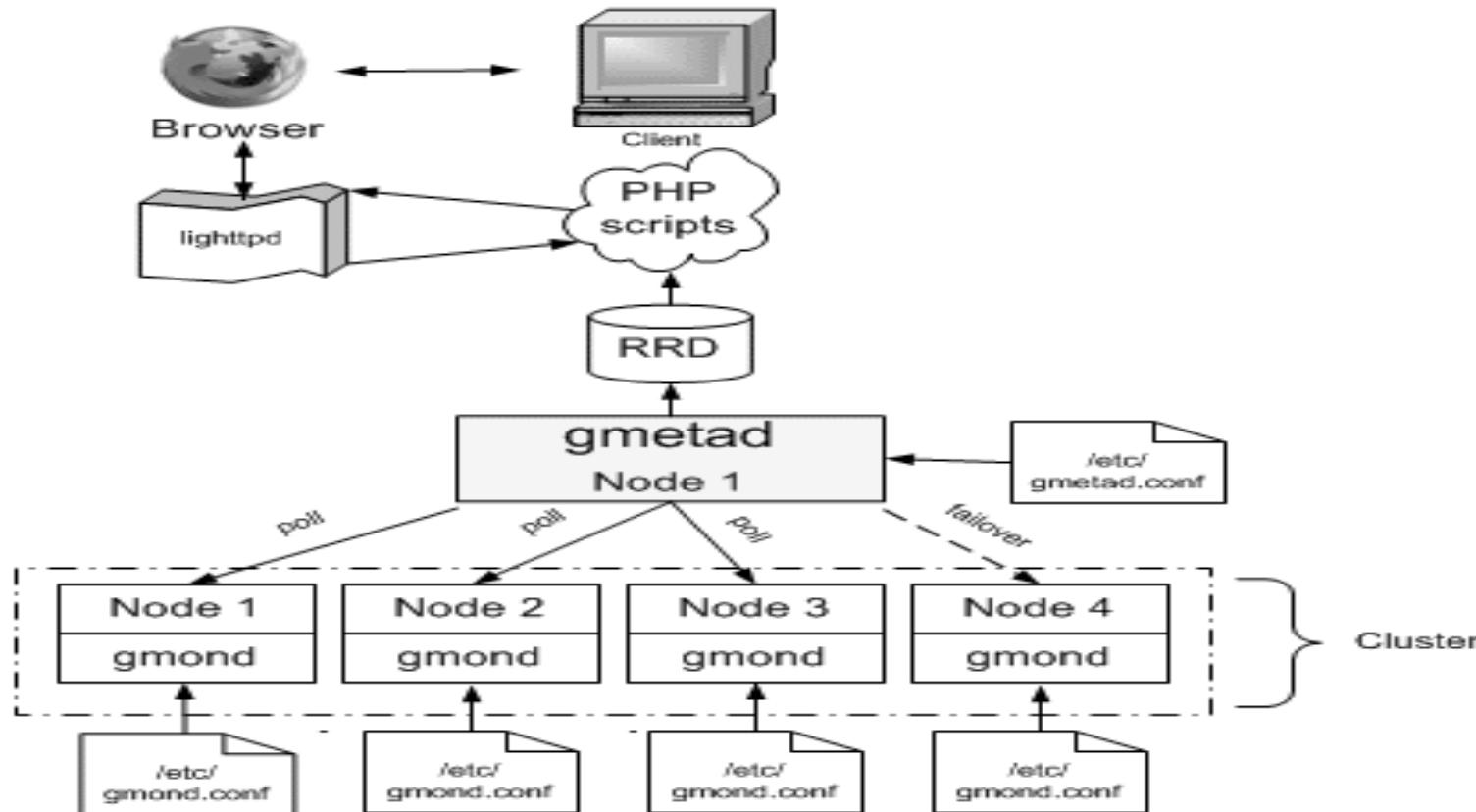


Figure 2: Ganglia architecture

❑ (gmond)Ganglia monitoring daemon

- gmond is installed on all nodes that need to be monitored.
- Runs on every node of the cluster and collects data about the node like CPU load, free memory, disk usage, network traffic etc.
- It monitors the changes in host state, listens to other gmond instances over multi-cast/unicast, and is responsible for sending XML, over a TCP connection, to gmetad.

❑ Gmetad

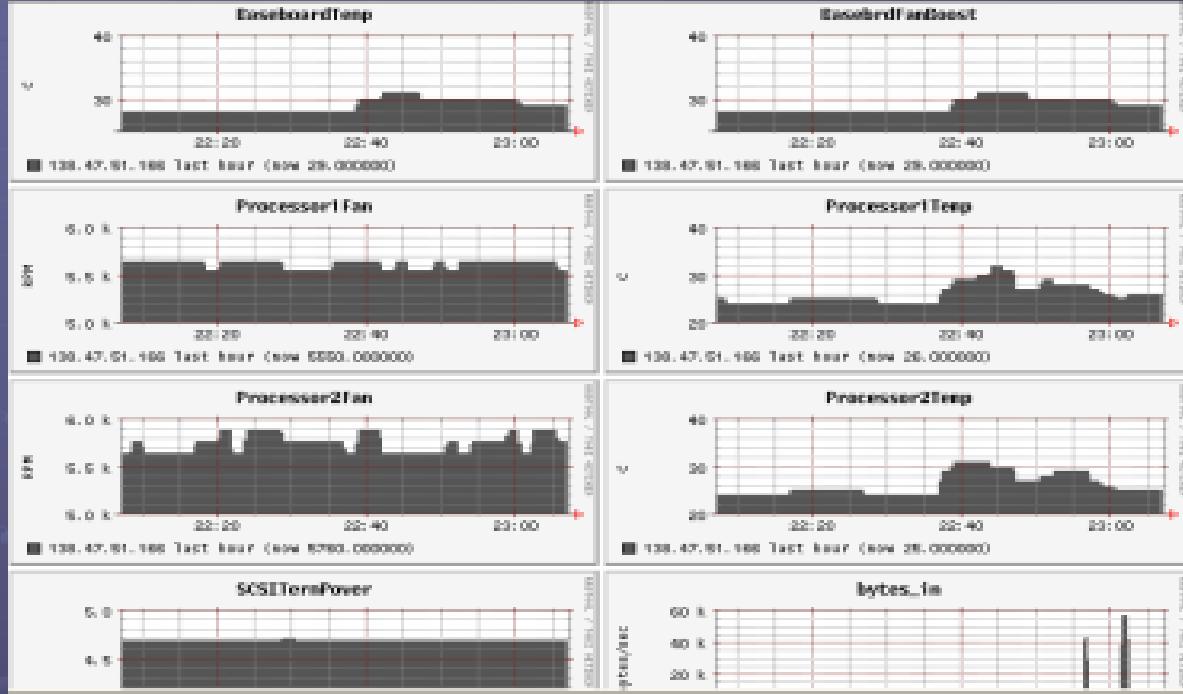
- Runs on a head node, gathers the data from all the nodes, and displays it.
- gmetad collects information from multiple gmond or gmetad sources. It saves the information to a local round-robin database, and exports (in XML) a concatenation of all the data sources.

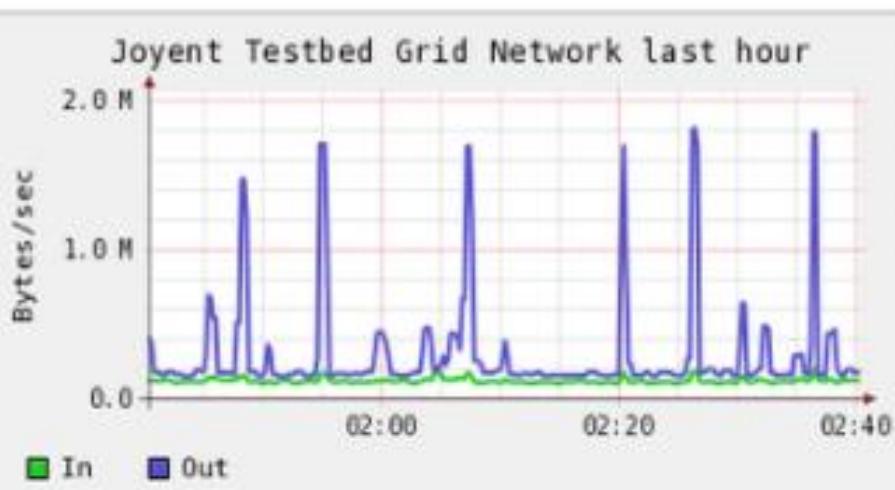
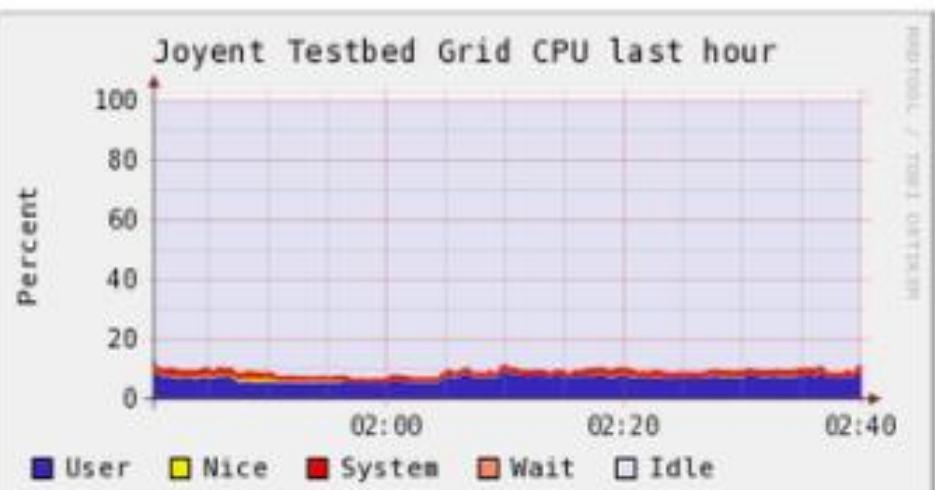
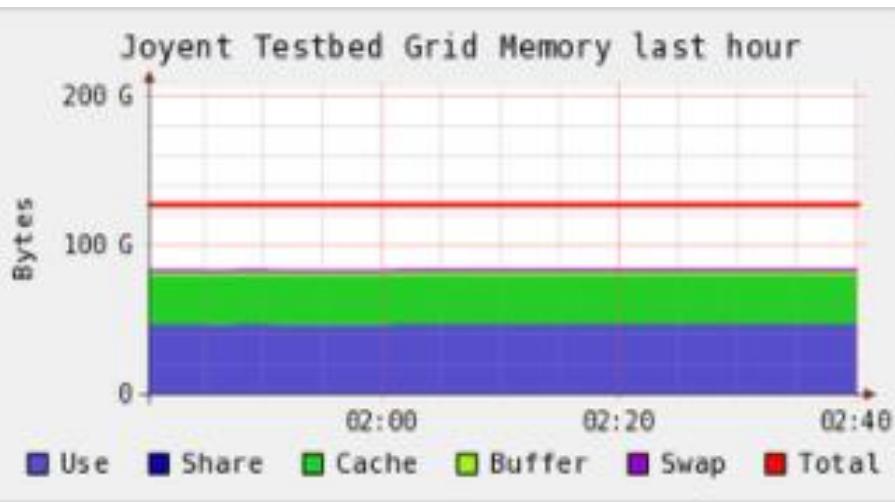
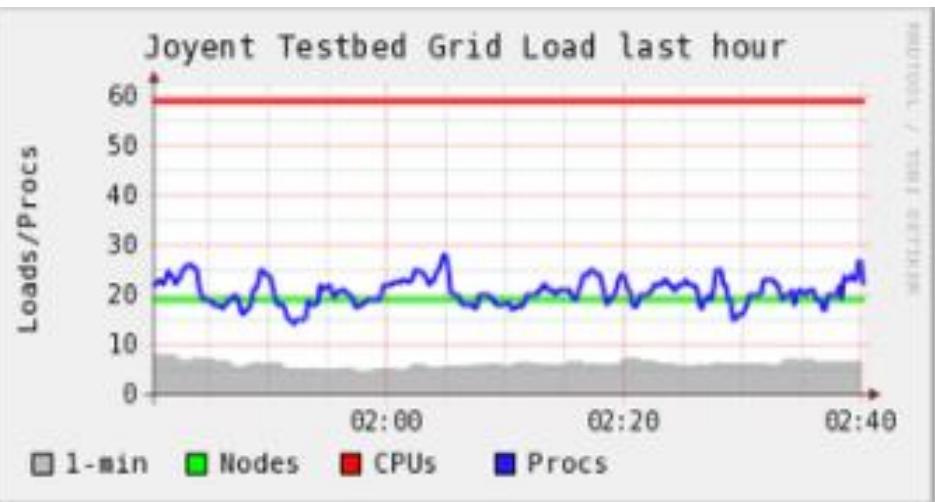
❑ Ganglia PHP Web frontend

- The frontend presents the data collected by gmond and gmetad in a meaningful form. Ganglia has been around for many years without much change to the UI — but since October 2010, a few contributors began contributing to a new UI, which looks awesome

❑ **RRDs are Round-Robin Databases.** The data is archived after each hour is stored in an average form. You can demand the exact value of a particular instance of a particular month and hour.

A snapshot of our enhanced Ganglia





Log Handling

- It's important for IT teams to continue having complete visibility of every component related to their applications and the underlying infrastructure. This is where real-time log monitoring and analysis come in.

□What Are Logs?

- All applications, networking devices, workstations, and servers create logs—or records of events—which are written to files on local disks by default. These logs contain crucial information. For example, an event log from a web server can contain information like a user's IP address, date, time, request type, and more. Logs provide administrators with an audit trail to find the root cause of problems and troubleshoot errors. Because there's a wide range of logs, they often vary in format.
- Logs are typically recorded in one or more log files. Log management allows you to gather the data in one place and look at it as part of a whole instead of separate entities. As such, you can analyze the collected log data, identify issues and patterns so that you can paint a clear and visual picture of how all your systems perform at any given moment.

□ Log file

- Log file is a file that records either messages or events that occur in an operating system/application. These messages are written to log file.

❑ Log management :-

- Log management is the process of handling log events generated by all software applications and infrastructure on which they run. It involves log collection, aggregation, parsing, storage, analysis, search, archiving, and disposal, with the ultimate goal of using the data for troubleshooting and gaining business insights, while also ensuring the compliance and security of applications and infrastructure.

❑ What Is Log Monitoring?

- Log monitoring refers to the set of practices involved in log management and analysis to help IT teams manage their infrastructure and operations

❑ Why is Log Analysis Becoming More Important?

- In today's competitive world, organizations cannot afford one second of downtime or slow performance of their applications. Performance issues can damage a brand and in some cases translate into a direct revenue loss.

❖**Log Monitoring:-**

Web Server Monitoring : You can track traffic volume, server errors, failed services, and more with server logs.

Network Monitoring : Devices like routers, firewalls, and load balancers form the backbone of enterprise networks.

Application Monitoring: While metrics give aggregated information about the health of different services over time

Database Monitoring: Database logs from MongoDB, MySQL, PostgreSQL, etc., can help in proactive monitoring and troubleshooting database errors.

ELK Stack

ELK Stack is the leading **open-source IT log management solution** for companies who want the benefits of a centralized logging solution without the enterprise software price.

Elasticsearch, Logstash, and Kibana when used together, form an end-to-end stack (ELK Stack) and real-time data analytics tool that provides actionable insights from almost any type of structured and unstructured data source.

Each of these products plays a different role in delivering one seamless stack:



- **Elasticsearch** provides the storage and analytics engine
- **Logstash** acts as a collection and transformation agent
- **Kibana** helps you visualize the data that you have

Introduction to issue trackers

- An issue in Issue Tracker is a bug report, feature request, change request or process workflow item that a user wants to track or expects another user or team to track.
- We can also say, An issue is any problem, question, grievance, or criticism offered by a user. When your customers experience a challenge while using your product, then they have encountered an issue.
- Each issue has a set of associated fields that describe it and its current state. This includes the type of issue, its importance in terms of severity and priority, and the record of activity on the issue

Issue tracker:- an Issue tracker records issues customers have experienced with a software product, and it enables support agents, engineers, and managers to track those problems until they have been successfully resolved. Often referred to as a bug tracking system, these tools present an organized way—via tickets—for customer service and product development teams to address issues efficiently while also providing users with timely status updates.

Issue tracking

- Issue tracking is the process of monitoring problems that users are experiencing with a software product. While issue tracking primarily came about as a solution for tracking problems in software.

Issue Tracking System

- An issue tracking system (also **ITS**, trouble ticket system, support ticket, request management or incident ticket system) is a computer software package or application that manages and maintains lists of issues, and also allows you to record and follow the progress of every customer ticket or "issue" in your inbox until the problem is resolved.
- In a typical issue tracking system, a customer service agent can assign a priority level to a ticket (for example, P1 for the most severe of issues to P4 for the least). Those priority levels will often be adjusted by development teams' project managers based on team resources, development roadmaps, and feasibility.

How are issue tracking tools used?

- Issue tracking systems are most commonly used for handling bugs. For example, when a user attempts to perform an action in a software product—such as logging in, exporting data, or creating a dashboard for analytics—and cannot complete the task, he or she will likely contact the company for help. That customer will contact the business via one of the available support channels. When the user makes that initial contact, the issue tracking system creates a ticket, which is assigned to a customer support agent. That agent might also choose the ticket rather than be assigned it, depending on how the support teams' workflow has been set up.

❖ Issue tracking used in development and testing:-

- Issue tracking is the process of logging and monitoring bugs or errors during software testing. It is also referred to as defect tracking or issue tracking.
- In IT, a bug refers to an **error, fault, defects or flaw** in any computer program or a hardware system. A bug produces unexpected results or causes a system to behave unexpectedly. **In short it is any behavior or result that a program or system gets but it was not designed to do.**
- Large systems may have hundreds or thousands of defects.
- Each needs to be evaluated, **monitored and prioritized for debugging**. In some cases, bugs may need to be tracked over a long period of time.
- “Defect tracking is an important process in software engineering, as complex and business critical systems have hundreds of defects,”.
- One of the challenging factors is **managing, evaluating and prioritizing** these defects. The number of defects gets multiplied over a period of time and to effectively manage them, a defect tracking system is used to make the job easier.”

❖ How bug tracking or defect tracking works

- A software bug occurs **when an application or program doesn't work the way it is designed to function.** Most errors are faults or mistakes made by system architects, designers or developers.
- Testing teams use **bug tracking** to monitor and report on errors that occur as an application is developed and tested.
- “A major component of a bug tracking system is a database that records facts about known bugs.”.
- Facts may include the time a bug was **reported, its severity, the erroneous program behavior and details on how to reproduce the bug;** as well as the identity of the person who reported it and any programmers who may be fixing it.”

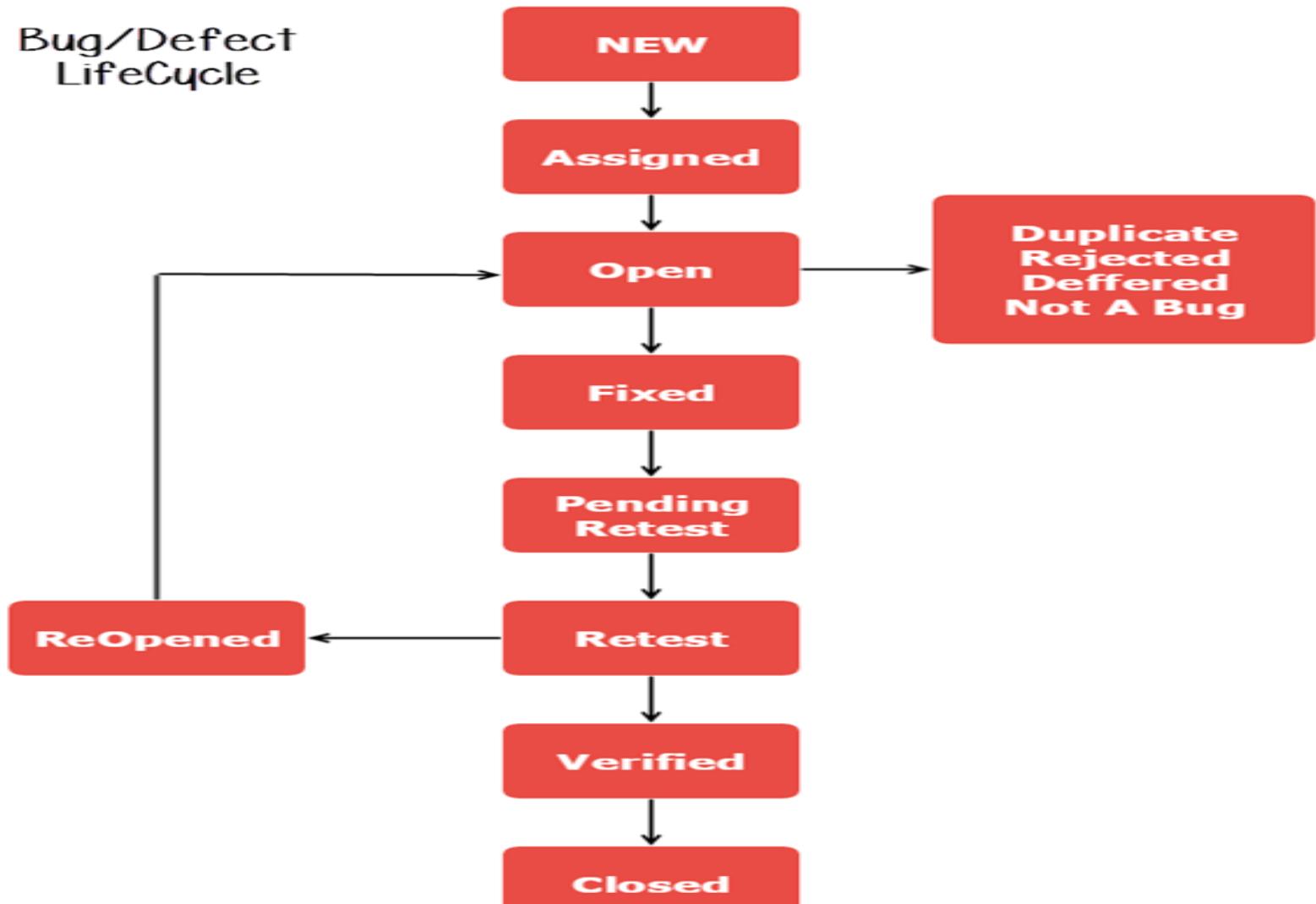
What is Defect Life Cycle or a Bug life cycle?

- **Defect Life Cycle** or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life.
- The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

❖ Defect Status

- **Defect Status** or Bug Status in defect life cycle is the present state from which the defect or a bug is currently undergoing.
- The goal of defect status is **to precisely convey the current state or progress of a defect or bug in order to better track and understand the actual progress of the defect life cycle.**
- The **number of states that a defect goes through** varies from project to project. Below lifecycle diagram, covers all possible states

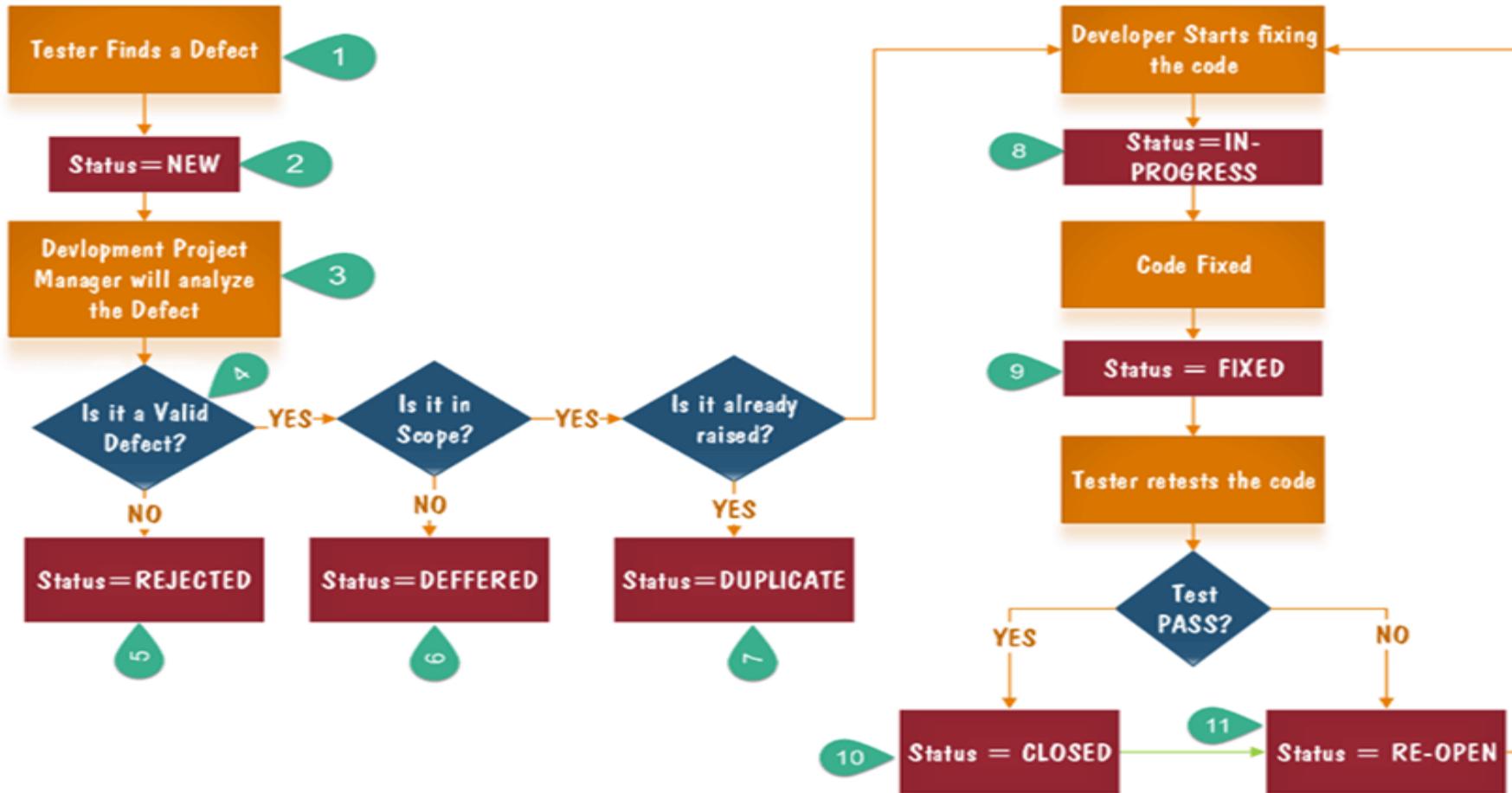
Bug/Defect LifeCycle



- **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
- **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
- **Open:** The developer starts analyzing and works on the defect fix
- **Fixed:** When developer makes necessary code change and verifies change, he or she can make bug status as ‘Fixed.’
- **Pending retest:** Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”
- **Retest:** Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
- **Verified:** The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
- **Reopen:** If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.

- **Closed**: If the bug is no longer exists then tester assigns the status “Closed.”
- **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate”
- **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
- **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
- **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

❖ Bug Life Cycle with Flow chart:-



- 1.Tester finds the defect
- 2.Status assigned to defect- New
- 3.A defect is forwarded to Project Manager for analyze
- 4.Project Manager decides whether a defect is valid
- 5.Here the defect is not valid- a status is given “Rejected.”
- 6.So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred** status.
- 7.Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.

8.If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress**.

9.Once the code is fixed. A defect is assigned a status **fixed**

10.Next, the tester will re-test the code. In case, the Test Case passes the defect is **closed**. If the test cases fail again, the defect is **re-opened** and assigned to the developer.

11.Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened**.

Need of issue tracker

- An issue tracker is software that allows developers, managers, and customer support agents to organize, prioritize, and track these issues and bugs until they are resolved. **Issue tracking systems**, commonly referred to as ITS
- Software testing is essential for isolating and mitigating errors. A good QA process can uncover hundreds or even thousands of defects, and testing teams need to manage all of them.
- Integrating bug tracking or issue tracking into the testing workflow improves efficiency by helping testers prioritize, monitor and report on the status of each error.
- “Defect tracking helps ensure that bugs found in the system actually get fixed,”
- Ideally, testing should be done as soon as possible .finding bugs in early stages are easier and far less costly to fix. That defects found post-production or after release can cost 15 times more to fix compared to errors resolved early in development.
- Many teams are now using a methodology known as continuous testing. In this case, quality testing and feedback are conducted at all stages of development.

❖ Why is an issue tracker important?

Issue trackers have several key benefits:-

- **Centralized issue management:** when all existing and solved bugs are stored in one central database, it's easier for developers, project managers, and other team members **to find relevant solutions quickly.**
- **Issue history tracking:** an issue tracker enables teams to **keep track of repetitive issues and effortlessly find answers to them.**
- **Appropriate delegation of work:** a central issue tracker improves delegation. For example, **if there's a problem with your app's UI, you can instantly assign the issue to the design team.**
- **Monitor actions related to every issue:** relevant teams can get real-time updates about every step taken to solve an issue, **so everyone is up-to-date on ongoing and solved bugs.**
- **Improved information sharing:** when every issue can be tracked at a single glance, **client-facing teams can give users timely updates to show you're working on a solution and, consequently, boost customer satisfaction.**
- **Automated reminders:** you can't take forever to fix issues. Customers and clients have limited patience. A bug reporting system with automated reminders ensures you never miss a deadline.

- **Increased efficiency:** a centralized issue tracker allows you to identify more effective methods used by other teams and implement these best practices in your team's processes.
- **Better collaboration:** software developers aren't the only ones involved in fixing bugs. Customer service reps, the QA team, and any other relevant teams all have to pitch in. You can streamline this collaboration with dynamic bug tracking software.
- **You can keep track of every issue, regardless of the department or the nature of the problem:-** An issue tracking system allows you to rapidly access answers and information and maintain a dashboard for tracking the history of issues and solutions. This makes creating reports, monitoring issues, and sharing information far easier.
- **You can monitor any action that has been taken on an issue:-** Once an issue has been submitted, it's important to stay track of who's responsible for resolving it.
If it needs to be escalated or dealt with by another department, the issue tracking software will show the status of the query and the project 'ownership.' A good issue tracking system will **allow you to track any actions taken on an issue with a quick glance on a dashboard.** Automated reminders and status updates keep everyone in the loop.
- **You can keep people informed about what has happened with each issue.** This is key for customer satisfaction and business efficiency. The more issues there are, the more staff who could interact with the issue – all these factors mean that issues can easily become lost or confused.

- An issue tracking system streamlines management for even the most complex issues, complaints, or special requests. People who know their requests and submissions have been understood and answered are far more likely to remain loyal customers or efficient, satisfied employees.
- Issue tracking software systems are easy and intuitive – and affordable. Any organization can improve its bottom line, staff satisfaction and customer retention using issue tracking software.

ISSUE TRACKER TEMPLATE		For unique Excel templates, click → here						someka		
DASHBOARD								Excel Solutions		
Project Name			High	Medium	Low	Total	%			
Delivery Process Improvement			1	1	0	2	16,7%			
Department			1	3	3	7	58,3%			
Proje office 11			1	1	1	3	25,0%			
Project Owner			3	5	4	12	100,0%			
Tom Cruise			%	25,0%	41,7%	33,3%	100,0%			
#	ISSUE	DEPARTMENT	PRIORITY	INITIATOR	STATUS	OPENED ON	CLOSED ON	DESCRIPTIONS		
1	Issue 1	IT	High	John Wicks	In Progress	13-Mar-16	14-Mar-16	This is a note about this project		
2	Issue 2	Marketing	Low	William Wicks	In Progress	02-May-16	14-Jun-16	This is a note about this project		
3	Issue 3	Operations	Low	Sarah Wicks	Not Started			This is a note about this project		
4	Issue 4	Marketing	Low	William Wicks	In Progress	02-Jun-16		This is a note about this project		
5	Issue 5	Operations	Medium	Sarah Wicks	In Progress	02-Jun-16		This is a note about this project		
6	Issue 6	Marketing	Medium	John Wicks	Closed	12-Jun-16	21-Jul-16	This is a note about this project		
7	Issue 7	Operations	Medium	John Wicks	In Progress	02-Jun-16		This is a note about this project		
8	Issue 8	IT	Low	William Wicks	In Progress	02-Jun-16		This is a note about this project		
9	Issue 9	Operations	Medium	Sarah Wicks	Not Started			This is a note about this project		
10	Issue 10	Marketing	High	William Wicks	Not Started			This is a note about this project		
11	Issue 11	Operations	Medium	Sarah Wicks	In Progress	02-Jun-16		This is a note about this project		

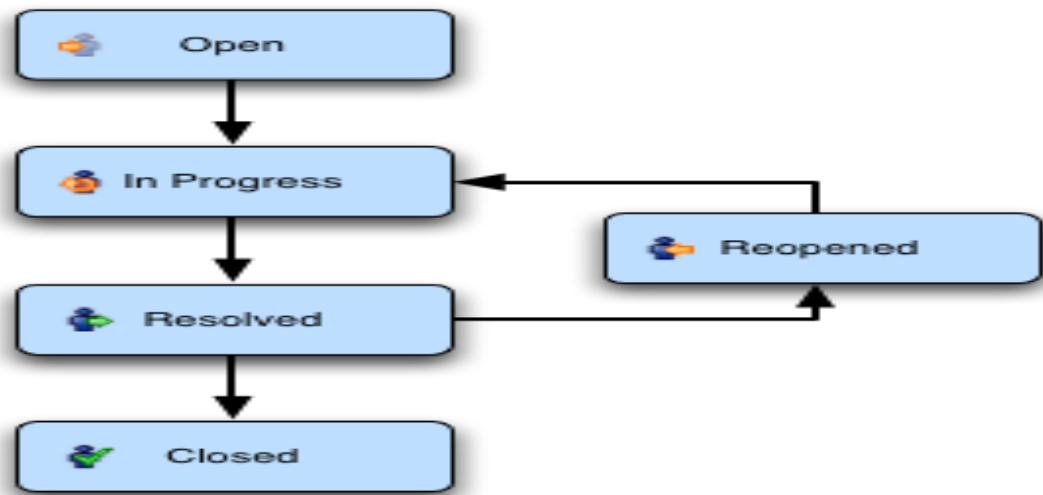
Some examples of workflows and issues

❖ What is an “Issue”?

- Any task that requires an action from a person

❖ What is “Workflow”?

- A workflow is some series of steps that need to be Carried out :



❖ What are fields?

- The attributes of an issue (Standard & Custom)

- An issue is any query or occurrence that might have an impact on a project
 - could be a software bug
 - could be a request for a change to something
 - could be something that just needs done
 - an action from a meeting
 - a task
- Whatever it is, following a simple process will help keep track of things

- The terms "ticket" or "bug". Technically, they are the same thing. An issue might also represent a to-do item, enhancement request, or any other type of work item.
- An issue has various metadata associated with it, depending on what it represents and what the issue tracker supports. The most basic type of issue is very simple, yet useful.
- It has the following basic attributes:
- **Description:** This is a free-form textual description of the issue
- **Reporter:** This represents the person who opened the issue
- **Assigned:** This is the person who should work on the item.

States of an issue representing the workflow:-

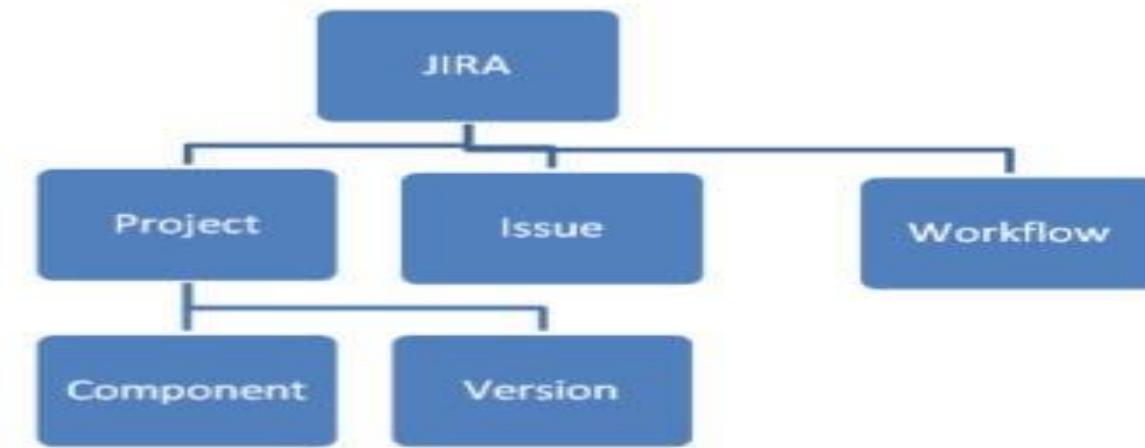
- **Open:** Here, the developer starts the process of analyzing the defect and works on fixing it, if required.
- **Closed:** When the defect does not exist any longer, then the tester changes the status of the defect to “Closed”.
- **Rejected:** If the defect is not considered a genuine defect by the developer then it is marked as “Rejected” by the developer.
- **Duplicate:** If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect then the status of the defect is changed to ‘Duplicate’ by the developer.
- **Deferred:** If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as ‘Deferred’.
- **Not a Bug:** If the defect does not have an impact on the functionality of the application, then the status of the defect gets changed to “Not a Bug”.

❖ Other attributes of an issue:-

- **Due date:** The date on which the **issue** is expected to be resolved.
- **Milestone:** A milestone is a way to group issues together in useful work packages that are larger than a single issue. A milestone can represent the output from a Scrum sprint, for example. A milestone **usually also has a due date**, and if you use the milestone feature, the due dates on individual issues are normally not used.
- **Attachments:** It might be convenient to be **able to attach screenshots and documents to an issue**, which might be **useful for the developer working on the issue or the tester verifying it**.
- **Work estimates:** It can be useful to have an estimate of the expected work expenditure required to resolve the issue. This can be used for planning purposes. Likewise, a field with the actual time spent on the issue can also be useful for various calculations.

- **JIRA** is a tool or a software used for **bug tracking, issue tracking, and project management**.
- The basic use of this tool is to **track issue and bugs related to your software and Mobile apps**. It is widely used as an **issue-tracking tool for all types of testing**.
- JIRA is **also a project management tool**. JIRA is **mainly used by agile development teams to customize your workflows, team collaboration, and release software with confidence**.
- But today, Jira **has evolved into a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development**.
- The JIRA **dashboard consists of many useful functions and features** which make **handling of issues easy**.
- Jira allows teams to break down their work into easily manageable portions and assign them to the right people. Teams can also **create a customized workflow for these projects to ensure they are done on time**.
- Its primary purpose is to help you with the following:
 - Manage Agile and Scrum teams
 - Organize your project tasks
 - Capture and record software bugs

- JIRA in its entirety is based on 3 concepts.



Issue: Every task, bug, enhancement request; basically anything to be created and tracked is considered an Issue.

Project: A collection of activities or tasks.

Workflow: A workflow is simply the series of steps an issue goes through starting from creation to completion.

Say the issue first gets created, goes to being worked on and when complete gets closed. **The workflow in this case is:**



❖ JIRA Issues and Issue types:

□ **What is JIRA Issue?**

- JIRA issue **would track bug or issue that underlies the project**. Once you have imported project then you can create issues.
- Under Issues, you will find other useful features like
- Issue Types
- Workflow's
- Screens
- Fields
- Issue Attributes

i) Jira Issue Types

- Issue Type displays all types of items that can be created and tracked via Jira testing tool. JIRA Issues are classified under various forms like new feature, sub-task, bug, etc. as shown in the screen shot.

The screenshot shows the Jira interface with the 'Issues' tab selected in the top navigation bar. On the left, a sidebar lists various management sections: ISSUE TYPES (with 'Issue Types' highlighted), WORKFLOWS, SCREENS, FIELDS, and ISSUE ATTRIBUTES. The 'Issue Types' section contains links for 'Issue Type Schemes' and 'Sub-Tasks'. The main content area is titled 'Issue Types' and displays a table of issue types:

Name	Type	Related Schemes
Sub-task The sub-task of the issue	Sub-Task	• Default Issue Type Scheme
Technical task Created by JIRA Agile - do not edit or delete. Issue type for a technical task.	Sub-Task	• Default Issue Type Scheme • Agile Scrum Issue Type Scheme
Bug A problem which impairs or prevents the functions of the product.	Standard	• Default Issue Type Scheme • Agile Scrum Issue Type Scheme
Epic Created by JIRA Agile - do not edit or delete. Issue type for a big user story that needs to be broken down.	Standard	• Default Issue Type Scheme • Agile Scrum Issue Type Scheme
Improvement An improvement or enhancement to an existing feature or task.	Standard	• Default Issue Type Scheme • Agile Scrum Issue Type Scheme
New Feature A new feature of the product, which has yet to be developed.	Standard	• Default Issue Type Scheme
Story Created by JIRA Agile - do not edit or delete. Issue type for a user story.	Standard	• Default Issue Type Scheme • Agile Scrum Issue Type Scheme
Task A task that needs to be done.	Standard	• Default Issue Type Scheme

- There are two types of Issue types schemes in Jira project management tool, one is
- **Default Issue Type Scheme:** In default issue type scheme all newly created issues will be added automatically to this scheme
- **Agile Scrum Issue Type Scheme:** Issues and project associated with Agile Scrum will use this scheme

ISSUE TYPES

- Issue Types
- Issue Type Schemes**
- Sub-Tasks

WORKFLOWS

- Workflows
- Workflow Schemes

SCREENS

- Screens
- Screen Schemes
- Issue Type Screen Schemes

FIELDS

- Custom Fields
- Field Configurations
- Field Configuration Schemes

ISSUE ATTRIBUTES

Issue Type Schemes

① An issue type scheme determines which issue types will be available to a set of projects. It also allows to specify the user interface.

Name	Options	Projects
Default Issue Type Scheme Default issue type scheme is the list of global issue types. All newly created issue types will automatically be added to this scheme.	<input checked="" type="checkbox"/> Bug (Default) <input checked="" type="checkbox"/> New Feature <input checked="" type="checkbox"/> Task <input checked="" type="checkbox"/> Improvement <input checked="" type="checkbox"/> Sub-task <input checked="" type="checkbox"/> Epic <input checked="" type="checkbox"/> Story <input checked="" type="checkbox"/> Technical task	Global (all unconfigured projects)
Agile Scrum Issue Type Scheme This issue type scheme is used by JIRA Agile's Scrum project template. Projects associated with the Scrum template will be associated to this scheme. You can modify this scheme.	<input checked="" type="checkbox"/> Epic <input checked="" type="checkbox"/> Story (Default) <input checked="" type="checkbox"/> Technical task <input checked="" type="checkbox"/> Bug <input checked="" type="checkbox"/> Improvement	No projects

ii) Issue fields

- Each issue has a set of associated fields that describe it and its current state. This includes the type of issue, its importance in terms of severity and priority, and the record of activity on the issue.

iii) JIRA Components

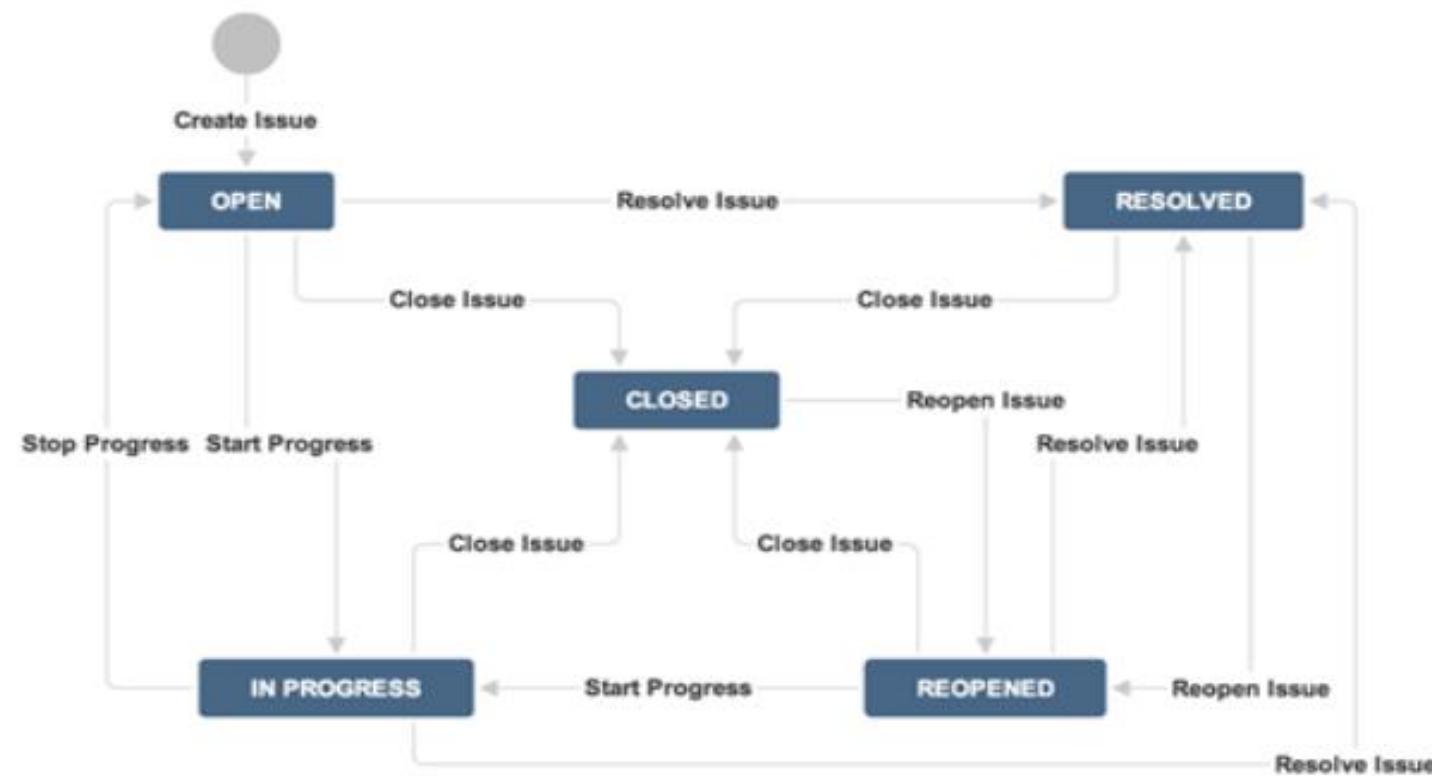
- **Jira Components** are sub-sections of a project; they are used to group issues within a project into smaller parts. Components add some structures to the projects, breaking it up into features, teams, modules, subprojects and more. Using components you can generate reports, collect statistics, and display it on dashboards and so on.

The screenshot shows the 'Components' section of a JIRA project. At the top, there is a header with the title 'Components' and a small icon consisting of three squares and a diamond. Below the header, a descriptive text states: 'Projects can be broken down into components, e.g. "Database", "User Interface". Issues can then be categorised against different components.' A table follows, listing components with columns for Name, Description, Component Lead, and Default Assignee. The table includes a header row and a data row for 'SAP Testing'. The 'Delete' button for the 'SAP Testing' row is highlighted with a red oval.

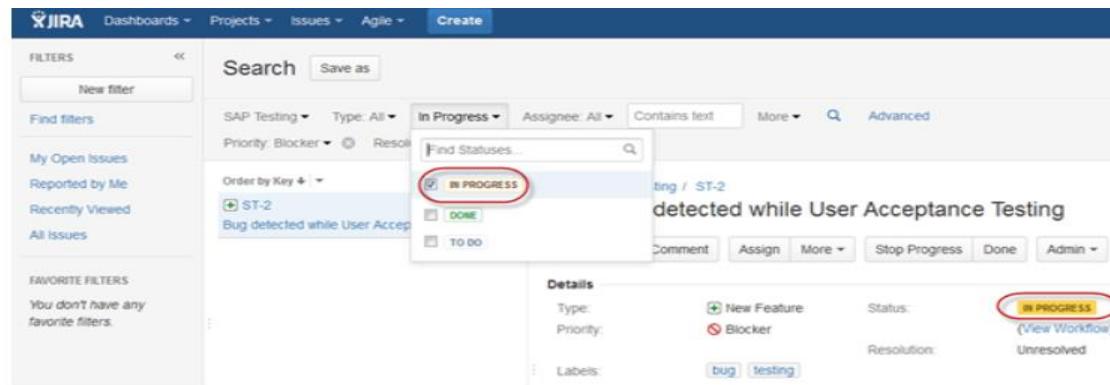
Name	Description	Component Lead	Default Assignee
SAP Testing	Bug detected while User Acceptance testing	Krishna Rungta [Administrator]	Project Lead

iv) WorkFlows

- A JIRA workflow is a set of statuses and transitions that an issue goes through during its lifecycle. JIRA workflow encompasses five main stages once the issue is created.
- Open Issue
- Resolved Issue
- InProgress Issue
- ReOpened Issue
- Close Issue



- While workflow in JIRA comprises of **Statuses, assignee, resolution, conditions, validators, post-function's and properties**
- **Statuses:** It represents the positions of the issues within a workflow
- **Transitions:** Transitions are the bridges between statuses, the way a particular issue moves from one status to another
- **Assignee:** The assignee dictates the responsible party for any given issue and determines how the task would be executed
- **Resolution:** It explains why an issue transitions from an open status to a closed one
- **Conditions:** Conditions control who can perform a transition
- **Validators:** It can ensure that the transition can happen given the state of the issue
- You can assign the status of the issue from the window itself, when you click on the check box for **IN Progress** status as shown in screen shot below, it will reflect the status in the issue panel highlighted in yellow.





FILTERS



New filter

Find filters

My Open Issues

Reported by Me

Recently Viewed

All Issues

FAVORITE FILTERS

You don't have any favorite filters.

Search

Save as

SAP Testing ▾

Type: All ▾

In Progress ▾

Assignee: All ▾

Contains text

More ▾



Advanced

Priority: Blocker ▾



Resolve

Order by Key ▾



ST-2

Bug detected while User Accep

Find Statuses...

 IN PROGRESS DONE T0 00

ting / ST-2

detected while User Acceptance Testing

Comment

Assign

More ▾

Stop Progress

Done

Admin ▾

Details

Type:

New Feature

Status:

IN PROGRESS

(View Workflow)

Priority:

Blocker

Resolution:

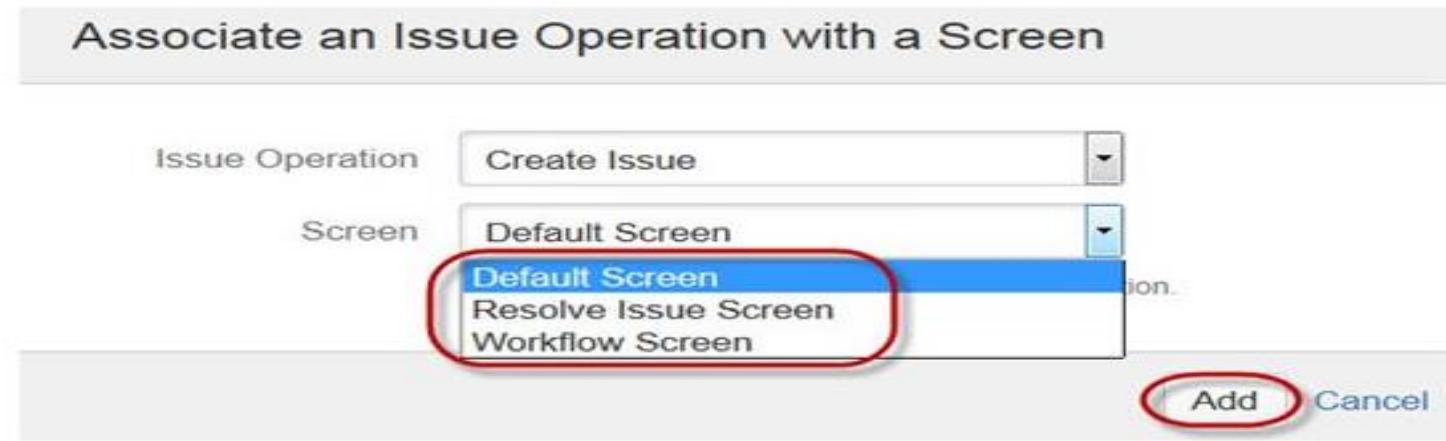
Unresolved

Labels:

bug testing

v) JIRA Screen

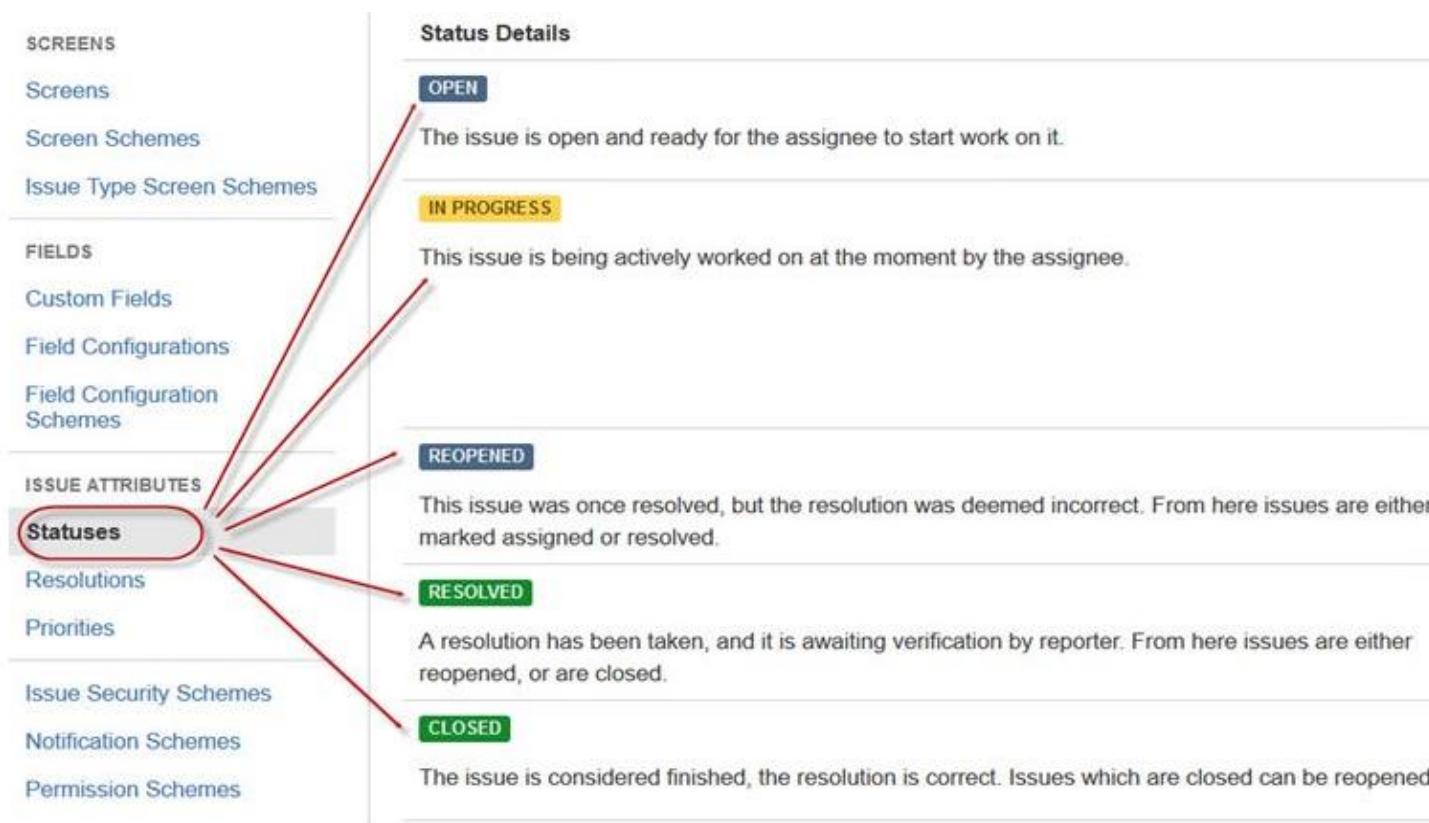
- When issue is created in JIRA, it will be arranged and represented into different fields, this display of field in JIRA is known as a screen. This field can be transitioned and edited through workflow. For each issue, you can assign the screen type as shown in the screen-shot. To add or associate an issue operation with a screen you have to go in main menu and click on Issues then click on Screen Schemes and then click on “Associate an issue operation with a screen” and add the screen according to the requirement.



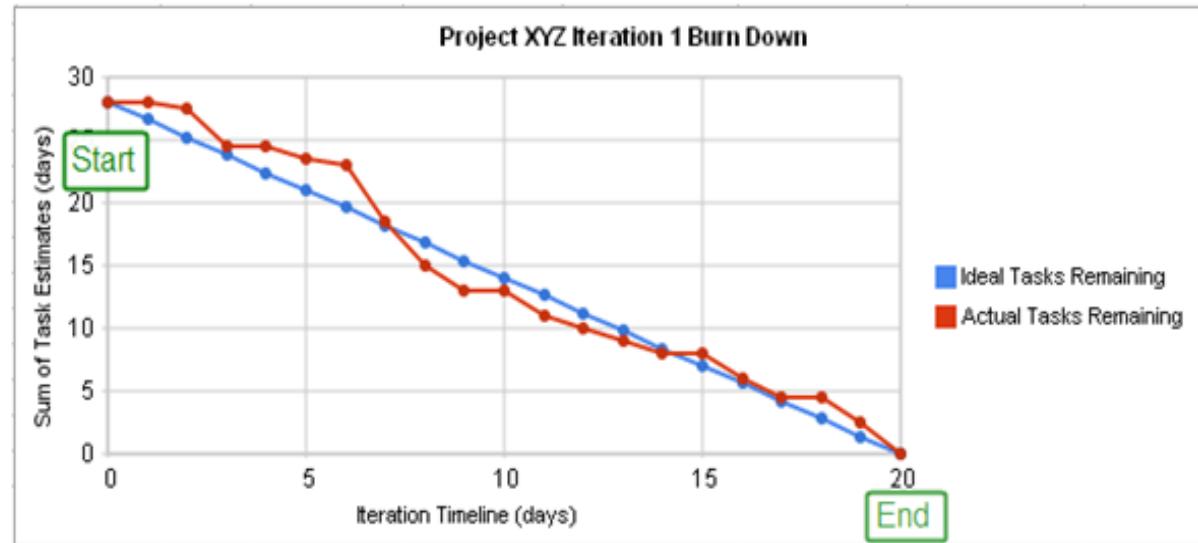
vi) Jira Issue Attributes

- Issue Attributes encompasses
 - Statuses
 - Resolutions
 - Priorities

- **Statuses:** Different statuses are used to indicate the progress of a project like **To do**, **InProgress**, **Open**, **Closed**, **ReOpened**, and **Resolved**. Likewise, you have resolutions and priorities, in resolution it again tells about the progress of issue like **Fixed**, **Won't fix**, **Duplicate**, **Incomplete**, **Cannot reproduce**, **Done** also you can set the priorities of the issue whether an issue is **critical**, **major**, **minor**, **blocker** and **Trivial**.



- To track the progress in Agile, a **Burndown Chart** shows the actual and estimated amount of work to be done in the sprint. A typical burndown chart will look somewhat like this, where the red line indicates the actual task remaining while the blue line indicates ideal task remaining during the scrum cycle.
- Apart from Burn down chart there are other options available in JIRA automation like **Sprint Report, Epic Report, Velocity Chart, Control Chart**.



- The following table explains some of the most important and commonly used features in detail for better understanding.

□ Boards

- JIRA supports Scrum and Kanban boards. These boards provide an immediate snapshot of the project to the team.
- Helps to quickly review the progress of the project and see the status of the individual tasks. Board workflow can be customized to fulfil the way a team wants to proceed.

□ Jira Dashboard

- The dashboard is the first thing when we login in the jira software. Here the admin can customise the dashboard's view and the things displayed on it.

□ Business Project Template

- JIRA supports **n** number of business templates to manage simple tasks and complex tasks like workflow.
- Template **can be customized based on the team and their approach**. Ex: Workflow can be customized based on each team's approach. Every step is accounted and team can move to achieve their goals.

□ Task Details

- Tasks can be defined at the **individual level** to track the progress.
- **Status of every task, comment, attachment and due dates are stored in one place.**

Notifications

- An [email](#) can be sent for a particular task to the users.
- [Voting](#) and [watching](#) features to keep an eye on the progress for the stakeholders.
- Use [@mention](#) to get the attention of a specific team member at [Comments](#)/[Description](#). User will instantly notify if something is assigned or if any feedback is required.

Power Search

- JIRA supports a powerful search functionality with Basic, Quick and Advanced features.
- Use the search tool to find answers like due date, when a task was last updated, what items a team member still needs to finish. Project information at one place, search within a project.

Reports

- JIRA supports more than a [dozen reports](#) to track progress over a specific timeframe, [deadlines](#), individual's contribution, etc.
- Easy to understand and generate different reports those help to analyse how the team is going on. Easy to configure these reports and display the matrices to the stakeholders.

Scale with Team Growth

- JIRA [supports](#) any business team and [any project](#) irrespective of size and complexity.

Multilingual

- JIRA supports more than 10 languages those are widely used as English (US, UK, India), French, German, Portuguese, Spanish, Korean, Japanese and Russian.

Jira security

- The security settings of jira bug tracking software restricts the access a certain bug only to those people who are allowed to work on a team member of the given security level. There is security feature like default permission scheme.

Custom workflows

- In a Jira workflow, colored blocks represent the processes, and arrows show the transitions. These workflows help you keep track of how your software development projects and tasks are progressing.
- What makes this feature more powerful is that you can start from scratch and create your own custom workflows too. You can even import workflow templates.

Problems with issue tracker proliferation

- The Issue tracker proliferation problem is that:
 - Sometimes large organization might standardize a type of tracker that few actually like to use.
 - typical reason is that only the needs of one type of team, such as the quality assurance team or the operations team, is being considered while deciding on an issue tracker.
 - Another reason is that only a limited amount of licenses were bought for the issue tracker, and the rest of the team will have to make do on their own.

- It is also quite common that the developers use one kind of tracker, the quality assurance team another, and the operations team yet another completely different and incompatible system.
- single tool that supports both Development and Operation workflows and issue types is ideal to reduce friction and promote reusability and shared understanding
- Teams using different tools to track work or defects often have difficulty working together.

❖ Examples:-

1. Production incidents originating in code or architecture flaws but stored in an issue tracking tool not accessible by Development often end up with incomplete fixes, as context and history is missing.
1. If these incidents are stored in an issue tracking tool accessible by Operations only, then Development will miss the context and history behind them. It's not uncommon that Development provides a fix that turns out incomplete because they were only told “what” failed, not the “why” and “how”.
3. On the other hand, Development work items stored in a tool not accessible by IT Operations often end up as rushed deployments that do not meet any kind of operational criteria.

BUGZILLA

- Bugzilla is an **open-source issue/bug tracking system** that allows developers to keep track of outstanding problems with their product.
- **Bugzilla is free**, open source software that you can install and use without having to pay any license fee.
- Bugzilla is a **powerful, Web-based bug or defect-tracking tool** originally developed by **Mozilla** .
- It is widely used by many organizations across the globe for reporting and managing bugs.
- Defect tracking systems **allow developers and testers to track all the outstanding defects**. Bugzilla can be linked to other testing tools like JIRA, QC or ALM, etc.
- This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data-management chain.
- Bugzilla helps users to **report bugs, assign bugs, prioritize, and arrange bugs by product and component**

Why Bugzilla?

- In earlier days, companies relied on shared lists and e-mails to track and update the status of defects reported.
- This was, however, an error-prone practice and caused lapse and confusion amongst developers to resolve or ignore significant bugs.
- Bugzilla provides an easy way to detect ,manage bugs and report them without much effort.

What does Bugzilla do?

- It keeps **track of both bugs and changes in code**.
- It enables **easy communication with teammates**.
- It **submits and reviews patches**.
- It is **responsible for quality assurance**.
- It can **dramatically increase the productivity and accountability of an individual by providing a documented workflow** and positive feedback for good performance.
- Most commercial and defect-tracking software vendors charged enormous licensing fees in the

Features of Bugzilla:-

Advanced search capabilities Bugzilla offers a simple search feature also called as a full-text search that spans **comments, summaries, and substrings of bugs.**

E-mail notification by user preferences Bugzilla gives flexibility to configure e-mail alerts as per their preferences. You can choose to or not to receive e-mails regarding the changes made to bugs. By default, an e-mail notification includes bug ID and summary of the bug report along with a list of the changes

Reports and Charts With Bugzilla, you can generate reports to view the current state of the bug database. Bugzilla also supports a charting system, which can **create graphs that track changes in the system over time.**

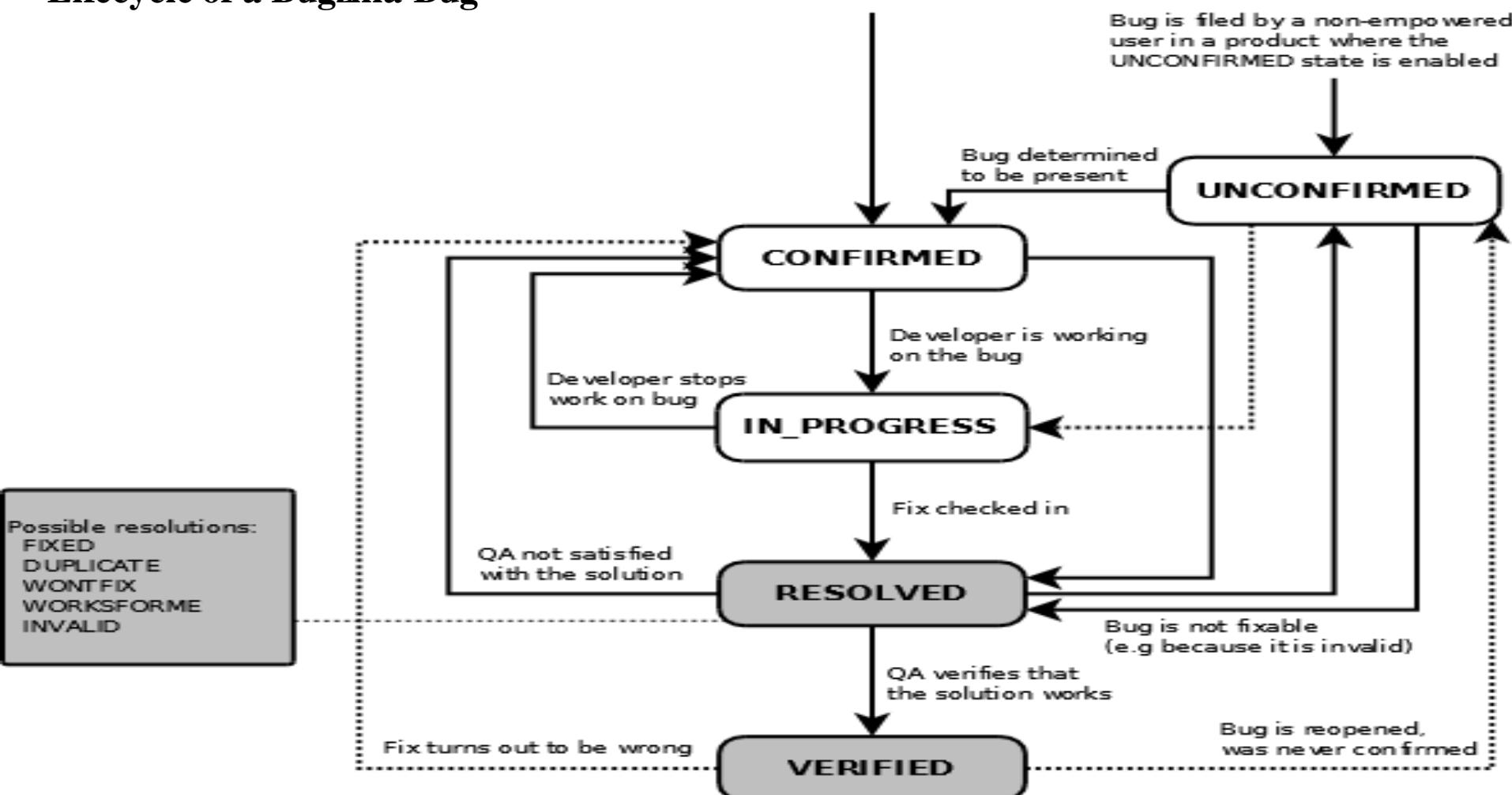
Automatic Duplicate Bug Detection With auto detection feature, you can identify if the bug you are going to file for a product in Bugzilla already exists in the database.

- Time tracking With time tracking feature, you can estimate the time a bug will take to fix and the time spent on that bug. You can also set a deadline for the completion of a bug.
- Bugzilla displays the complete bug change history. Bugzilla provides inter bug dependency track and graphic representation.
- Bugzilla allows users to attach Bug supportive files and manage it.

❖Who benefits most from using the Bugzilla bug tracking tool?

- Organizations involved in multiple software development projects -- especially organizations with large teams of developers and testers -- would find Bugzilla very helpful in their efforts, as it provides them with a convenient centralized location for tracking ongoing initiatives and helps cut down on duplicate and unnecessary work.
- All members of the DevOps team, no matter what their involvement is during development, will find the software simple to use.

Lifecycle of a Bugzilla Bug



This workflow is configurable, but the default states are as follows:

- **Unconfirmed:** A new bug by a non-EMpowered user begins in the unconfirmed state
- **Confirmed:** If the bug is confirmed to be worthy of investigation, it is put in the confirmed state
- **In progress:** When a developer starts working on resolving the bug, the in progress state is used
- **Resolved:** When developers believe that they have finished fixing the bug, they put it in the resolved state
- **Verified:** When the quality assurance team agrees that the bug is fixed, it is put in the verified state

Advantages of Bugzilla

- Improves quality
- Improves communication
- Increase productivity
- Improve customer satisfaction
- Reduce costs

GitLab tracker:

- The GitLab Issue Tracker is an advanced and complete tool for tracking the issues.
- Client constantly **pushing out new features and improvements to your product**, but with those updates and changes comes the **inevitable risk of bugs**.
- The **Issue Tracker** is the collection of all issues opened and closed that are created in a project. It is **available for all projects**, from the moment the project is created.
- Gitlab tracker uses gitlab issue boards to tackle and manage issues.
- When you access your project's issues, GitLab **will present them in a list**, and you can use the tabs available to quickly filter by open and closed issues.

❖GitLab Issues Board:-

- The GitLab Issue Board is **a software project management tool used to plan, organize, and visualize a workflow** for issues and feature or product release.
- It can be used as a Kanban or a Scrum board.
- It pairs **issue tracking and project management**, keeping everything together, so that you don't need to jump between different platforms to organize your workflow.
- Issue boards build on the gitlab's existing issue tracking functionality and leverage the power of GitLab labels.
- GitLab issue labels can be utilized as a lists on a Kanban board. Your issues appear as cards in vertical lists, organized by their assigned labels, milestones, or assignees.



In dev

0 80 174

Read git data via gitaly

In dev Plan backend

Work for the Plan team. Covers Issues, Labels, Milestones, Boards, and more. See https://about.gitlab.com/handbook/product/categories/. Ping @victorwu for questions and comments.

P1
Product vision 2019 UX ready backend
code review devops:create direction
feature proposal frontend merge requests

gitlab-org/gitlab-ce#18008 5



Multiple blocking merge request approval rules

Create Deliverable GitLab Premium In dev
P1 UX approvals backend customer
customer+ devops:create direction
missed-deliverable missed:11.5 project

In review

0 34 95

'ExpireBuildArtifactsWorker' is broken

In review P3 S3 Verify database

devops:verify missed-deliverable performance

gitlab-org/gitlab-ce#41057



Ensure that all CI/CD queries take less than 15 seconds to complete

Accepting merge requests In review Stretch

Verify database devops:verify meta

missed-deliverable performance

gitlab-org/gitlab-ce#40524

Further improvements to Project overview UI

Deliverable In review Manage P2
UX ready devops:manage direction frontend
missed-deliverable missed:11.5 project

Closed

0 47352 19838

include brand ai styles

To Do

gitlab-org/design.gitlab.com#5

static page example

To Do

gitlab-org/design.gitlab.com#4

welcome page

To Do

gitlab-org/design.gitlab.com#3



An Issue Board shows you what issues your team is working on, who is assigned to each, and where in the workflow those issues are

Issue boards help you to [visualize and manage your entire process in GitLab](#). You add your labels, and then create the corresponding list for your existing issues. When you're ready, you can drag your issue cards from one step to another one.

To let your team members organize their own workflows, use multiple issue boards. [This allows creating multiple issue boards in the same project](#).

❖ Components of gitlab Issue Board:-

□ Create a New Issue

- Issues help you summarize your ideas and align the team on a common goal.

□ Label your issue

- Labels make it easy to categorize issues based on descriptive titles. They help teams **quickly understand the context of an issue**. Labels can be used to describe the issue type, like "**new feature**," or describe the issue stage, like "**QA**."

□ Navigate to your Issue Board

- Issues appear as cards on the Issue Board.
They can be arranged across a number of columns using lists.

Create a new list

- Lists are based on the labels already created for your project.
Each list contains the issues that are assigned to the corresponding label.

Move issues between lists:- Communicate progress on your release and easily adjust your plans right from the Issue Board

Multiple Project Issue Boards :-Create multiple Issue Boards for each of your projects.

It's simple to use:- navigate to your Issue Board, click to open a dropdown menu that will give you the option to create new boards, edit the existing ones, or delete the boards you don't need anymore. Use the same dropdown menu to navigate between your boards.

Group Issue Boards

- Create Issue Boards at the group level and manage all issues of the underlying projects in a single and concentrated view.
- Lists, labels, and milestones are all managed at the group level, allowing you to focus on the group. This means a team may naturally be working across multiple projects, making it easier to track progress and coordinate efforts.

What is GitLab board?



One tool, endless functionality

Everything is in one place. Track issues and communicate progress without switching between products. One interface to follow your issues from backlog to done.



Beautiful and comprehensive

It's more than a visual representation. It's your issues, your labels, and all the metadata that comes with them. And you still have all of the same sorting and filtering tools that you use across GitLab.



Flexible setup

You define your process and we organize it. You create the label. We create the corresponding column and pull it in your existing issues. When you're ready you drag and drop from one step to the next.

- To share reports that are helpful for your developers.
- The process is long, complicated, and tracking down the crucial technical information isn't always easy.

In most teams, reporting bugs into GitLab looks like this:

1. Find the bug.
2. Open screenshot tool, capture bug.
3. Open software to annotate screenshot, add comments.
4. Open and log into GitLab.
5. Select the correct project.
6. Create new issue.
7. Document the bug.
8. Add technical information. Attach screenshots.
9. And then finally: submit report.

Development

Search or filter results...

Show labels



Group by

None

Edit board

Create list



> Open

56 10 +

Milestones swimlanes

Doing dev plan

gitlab-org/gitlab-test#80 Jun 17 4

Assign issue to epic

To Do backend

gitlab-org/gitlab-test#45

Create group

To Do

gitlab-org/gitlab-test#78 3

Remove issue from board

To Do

gitlab-org/gitlab-shell#38

Add lists for assignees and milestones

To Do

gitlab-org/gitlab-shell#2

> Deliverable

32 8 +

Update issue due date from sidebar

To Do frontend

gitlab-org/gitlab-test#54

Update issue's labels from sidebar

To Do frontend

gitlab-org/gitlab-test#55

Update issue labels

Doing

gitlab-org/gitlab-test#75

Drag and drop issue between epics

To Do frontend

gitlab-org/gitlab-test#33

Paginate issues in Swimlanes

To Do

gitlab-org/gitlab-test#39

> Closed

59 2 +

Persist collapsed state of Swimlanes

Deliverable test test

Dec 30, 2020 1w 2

gitlab-org/gitlab-test#32

Remove list from board

Caliber Colorado Doing feature proposal

gitlab-org/gitlab-test#44

Remove issue from Swimlane

To Do frontend

gitlab-org/gitlab-test#36



Expand diff to entire file

Premium To Do dev manage frontend

gitlab-org/gitlab-test#25

Laboriosam commodi ab in eum qui suscipit necessitatibus modi fuga.

Deliverable frontend

