NTTF
Securing your Future with your own Hands

CP 08 SEMESTER 5 ITIL & DEVOPS

SME-Preetha S

Prepared by: Preetha S
Modified by: Preetha S
Verified by:Program cooridinator
Approved by: Program cooridinator
Rev No : 02
Released Date: 22-08-2022

INDEX

CONTENT

UNIT 1 Introduction to ITIL

1.1 ITIL History, components of ITIL and their specific functions

Definition:

ITIL, formerly an acronym for Information Technology Infrastructure Library, is a set of detailed practices for IT service management (ITSM) that focuses on aligning IT services with the needs of business.

ITIL describes processes, procedures, tasks, and checklists which are not organization-specific nor technology-specific, but can be applied by an organization towards strategy, delivering value, and maintaining a minimum level of competency. It allows the organization to establish a baseline from which it can plan, implement, and measure. It is used to demonstrate compliance and to measure improvement.

History

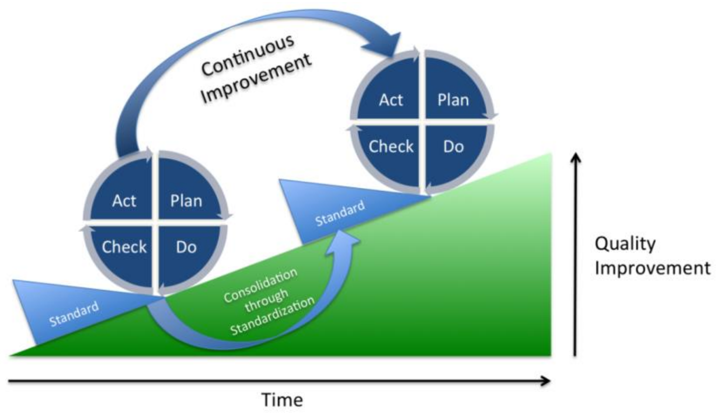
ITIL was developed by the UK Government's Central Computer and Telecommunications Agency (CCTA) in the 1980s, due to the growing influence of IT in the government and other industries to form a set of recommendations.

It recognized that, without standard practices, government agencies and private sector contracts had started independently creating their own IT management practices.

ITIL was built around a process model-based view of controlling and managing operations often credited to W. Edwards Deming and his plan-do-check-act (PDCA) cycle

PDCA Cycle:

It is also known as the Deming circle, which is a continuous feedback system for continuous improvement of any process/operation.



The IT Infrastructure Library originated as a collection of books, each covering a specific practice within IT service management. After the initial publication in 1989–96, the number of books quickly grew within ITIL Version 1 to more than 30 volumes.

ITIL Version 2:

In 2000/2001, to make ITIL more accessible, ITIL Version 2 consolidated the publications into nine logical "sets" that grouped related process-guidelines to match different aspects of IT management, applications and services.

In April 2001, the CCTA was merged into the OGC (Office of Government Commerce), an office of the UK Treasury. In 2006, the ITIL Version 2 glossary was published.

In 2009, the OGC officially announced that ITIL Version 2 certification would be withdrawn and launched a major consultation as per how to proceed.

ITIL Version 3:

In May 2007, this organization issued ITIL Version 3 (also known as the ITIL Refresh Project) consisting of 26 processes and functions, now grouped into only 5 volumes, arranged around the concept of Service lifecycle structure. ITIL Version 3 is now known as ITIL 2007 Edition.

Since 2013, ITIL has been owned by AXELOS, a joint venture between Capita (International business outsourcing company based in UK) and the UK Cabinet Office.

1.2 Service Management for IT, Technology and architecture of ITIL, HPSM and OTRS

Definition of Service Management

Service Management is a set of specialized organizational capabilities for providing value to customers in the form of services. The capabilities take the form of functions and processes for managing services over a lifecycle, with specializations in strategy, design, transition, operation and continual improvement. The capabilities represent a service organization’s capacity, competency and confidence for action. The act of transforming resources into valuable services is at the core of Service Management. Without these capabilities, a service organization is merely a bundle of resources that by itself has relatively low intrinsic value for customers.

Service management capabilities are influenced by the following challenges that distinguish services from other systems of value creation such as manufacturing, mining and agriculture Intangible nature of the output and intermediate products of service processes: difficult to measure, control, and validate (or prove).

Demand is tightly coupled with customer’s assets: users and other customer assets such as processes, applications, documents and transactions arrive with demand and
stimulate service production.

· High-level of contact for producers and consumers of services: little or no buffer between the customer, the front-office and back-office. The perishable nature of service output and service capacity: there is value for the customer in receiving assurance that the service will continue to be supplied with consistent quality. Providers need to secure a steady supply of demand from customers.

1.3 ITIL Service Strategy, service portfolio management and process objective, demand management process

Service Portfolio:

The most effective way of managing all aspects of services through their lifecycle is by using appropriate management systems and tools to support and automate efficient processes.

The Service Portfolio is the most critical management system used to support all processes and describes a provider's services in terms of business value. It articulates business needs and the provider's response to those needs. By definition, business value terms correspond to market terms, providing a means for comparing service competitiveness across alternative providers.

The Service Portfolio would therefore contain details of all services and their status with respect to the current stage within the Service Lifecycle Customers and users would only be allowed access to those services within the Service Portfolio that were of a status between 'chartered' and 'operational', those services contained within the Service Catalogue. Service Strategy and Service Design personnel would need access to all records within the Service Portfolio, as well as other important areas such as Change Management.

Other members of the service provider organization would have access to a permitted subset of the records within the Service Portfolio. Although the Service Portfolio is designed by Service Design, it is owned and managed by Service Strategy within the Service Portfolio Management process.

Unit 2 ITIL Service Lifecycle processes and functions

2.1 Service life cycle, approach to IT Service Management (ITSM), 5 Stages of the ITIL - Service Strategy -financial management, service portfolio management, demand management, strategy operations

History of ITIL

The British government commissioned a study to find out “what is the best way to align IT with business objectives, lower costs and improve quality.” Results published in 40+ books as the Information Technology Infrastructure Library (ITIL) Revisions between 2000 and 2004 resulted in Version 2 of ITIL as a consolidated framework for IT Service Management Version 3 released in 2007 establishes a lifecycle approach to IT Service Management.

Why ITIL is important?

• Lower TCO

• Redirect Operational Costs to Innovation

• Improve Service Quality

• Improve Predictability of IT Costs and Chargebacks

• Facilitates Outtasking /Outsourcing Operations

• Connect all the pieces of systems management

• Represents best practices in the industry

• Vendor neutral

• Consistent concepts and terminology ITIL services are key to understanding how well IT is being utilized and the extent to whichthe IT function is meeting its service level commitments

ITIL is focused on improving service quality to the customer. Customers who adopt the ITILframework can expect to see improved alignment of IT with the business, a long-termreduction in the cost of IT services, and better service quality and responsiveness. Customers should see a reduction in system outages as proactive planning and quality measures are implemented, and they should be able to implement IT infrastructure changesmore quickly to support new business requirements. Customer satisfaction should increasebecause service providers will understand what is expected of them, based on businessneeds. Support services should become more competitive.

Service providers will also benefit from the adoption of ITIL. Service providers should see improvements in service delivery because they will have a single definable, repeatable,scalable, and consistent set of IT processes. Roles and responsibilities will be properlyaligned and understood.

Communications between IT departments should improve because the linkages between processes will be documented and understood. Service providers should see better resource utilization and resources will be allocated based on business demands. Staff will be more professional and motivated because skill requirements and capabilities, along with job expectations, are better understood.

ITIL Implementation

Adopt ITIL as a common language and reference point for IT Service Management best practices and key concepts Adapt ITIL best practices to achieve business objectives specific to each company ITIL describes what needs to be done but not how it should be done.

ITIL does not claim to be a comprehensive description of everything within IT, but IT management “best practices”observed and accepted in the industry.

ITIL does not define:· Every role, job, or organization design· Every tool, every tool requirement, every required customization.Every process, procedure, and task required to be implemented ITIL V3 represents a significant milestone in the evolution of service management The initial five books should be available on 30 May 2007(in English –

translation work has begun).

The release of ITIL v3 should further accelerate interest in and adoption of leading practices for service management IBM strongly supports the continued advancement of service management best practice ITIL V3 represents a significant milestone in the evolution of service management.

Introduces the Service Lifecycle Bring more management disciplines to the attention of the professional service manager Adds new service management concepts to ITIL

Emphasises the need for integration.Updates and integrates ITIL v2 content

ITIL V3 Service Lifecycle

ITIL Publications Structure

Core

Introduction to the ITIL Service Lifecycle

Five books

Service Strategy (SS)

Service Design (SD)

Service Transition (ST)

Service Operation (SO)

Continual Service Improvement (CSI)

Complementary Publications

Support for particular market sector or technology

Web

Value added products, process maps, templates, studies

Service Management as a Practice

What is a Service: Services are a ‘means of delivering value to customers by facilitating outcomes customers want to achieve, without the ownership of specific costs and risks’.

Services are a means of delivering value to customers by facilitating outcomes customers want to achieve, without the ownership of specific costs and risks. Services facilitate outcomes by enhancing the performance of associated tasks and reducing the effect of constraints. The result is an increase in the From the customer's perspective, value consists of two primary elements: utility or fitness for purpose and warranty or fitness for use.

Utility is perceived by the customer from the attributes of the service that have a positive effect on the performance of tasks associated with desired outcomes. Removal or relaxation of constraints on performance is also perceived as a positive effect. Warranty is derived from the positive effect being available when needed, in sufficient capacity or magnitude, and dependably in terms of continuity and security.

Definition of Service Management

Service Management is a set of specialized organizational capabilities for providing value to customers in the form of services. The capabilities take the form of functions and processes for managing services over a lifecycle, with specializations in strategy, design, transition, operation and continual improvement. The capabilities represent a service organization's capacity, competency and confidence for action. The act of transforming resources into valuable services is at the core of Service Management. Without these capabilities, a service organization is merely a bundle of resources that by itself has relatively low intrinsic value for customers.

Service management capabilities are influenced by the following challenges that distinguish services from other systems of value creation such as manufacturing, mining and agriculture:

· Intangible nature of the output and intermediate products of service processes: difficult tomeasure, control, and validate (or prove).

Demand is tightly coupled with customer's assets: users and other customer assets such as processes, applications, documents and transactions arrive with demand and
stimulate service production.High-level of contact for producers and consumers of services:

little or no buffer between the customer, the front-office and back-office.

The perishable nature of service output and service capacity: there is value for the customer in receiving assurance that the service will continue to be supplied with consistent quality. Providers need to secure a steady supply of demand from customers.

What is a Process?

Business outcomes are produced by business processes governed by objectives, policies and constraints. The processes are supported by resources including people, knowledge, applications and infrastructure. Workflow coordinates the execution of tasks and flow of control between resources, and intervening action to ensure adequate performance and desired outcomes. Business processes are particularly important from a service management perspective. They apply the organization’s cumulative knowledge and experience to the achievement of a particular outcome.

Processes are strategic assets when they create competitive advantage and market differentiation.

As a result, business processes define many of the challenges faced by service management. The nature and dynamics of the relationship between business processes and IT best explains this.

Processes that provide transformation towards a goal, and utilize feedback for selfreinforcing and self-corrective action, function as closed-loop systems. It is important to consider the entire process or how one process fits into another. Process definitions describe actions, dependencies and sequence.

Processes have the following characteristics:

Measurable: We are able to measure the process in a relevant manner. It is performance driven. Managers want to measure cost, quality and other variables, while practitioners are concerned with duration and productivity.

· Specific results: The reason a process exists is to deliver a specific result. This result must be individually identifiable and countable. While we can count changes, it is impossible to count how many Service Desks were completed.

· Customers: Every process delivers its primary results to a customer or stakeholder.They may be internal or external to the organization but the process must meet their expectations.

· Responds to a specific event: While a process may be ongoing or iterative, it should be traceable to a specific trigger.

ITIL Process Model:

In the ITIL process model, data enters the process, is processed, and then the output is measured and reviewed. A process is always organized around a goal. By defining which inputs are necessary and which outputs will result from the process, it is possible to work in a more efficient and effective manner. Measuring and steering the activities increases this effectiveness. To discover whether or not activities are contributing to the business goal of the process, measurements must be made on a regular basis.

Measuring allows comparison between what has actually been done and what the organization set out to do. Process improvements can then be considered and implemented.

What is a Role: A set of responsibilities defined in a Process and assigned to a person or team.One person or team may have multiple Roles, for example the Roles of Configuration Manager and Change Manager be carried out by a single person.

Roles of processes:

Process Owner Role: Responsible for the process goal and the results of the Process . The “What” . Process Manager Role: Responsible for the design, and management of the process. Reports the process results to the process owner . The “How”. Process Operatives Role: Responsible for defined activities within the process. These activities are reported to the process manager.

What is a Function?

Functions are a means of structuring organizations so as to implement the specialization principle. Functions typically define roles and the associated authority and responsibility for a specific performance and outcomes. Coordination between functions through shared processes is a common pattern in organization design. Functions tend to optimize their work methods locally, to focus on assigned outcomes. Poor coordination between functions, combined with an inward focus, leads to functional silos that hinder alignment and feedback critical to the success of the organization as a whole.

Process models help avoid this problem with functional hierarchies by improving cross-functional coordination and control. Well-defined processes can improve productivity within and across functions.

The Service Lifecycle

•Service Strategy provides guidance on how to design, develop and implement IT services as strategic assets

•Service Design provides guidance for the design and development of services and service management processes

•Service Transition provides guidance for the improvement of capabilities and transitioning services into operations

•Service Transition introduces the concept of the Service Knowledge Management System

•Service Operation provides guidance on service delivery and support so as to ensure value for the customer and service provider

•Continual Service Improvement provides guidance on creating value for customers through better service management The ITIL library is a set of books that describe internationally accepted best practices for IT service management. These include the following practices:

· A process-based approach to IT service management

· A common language for IT service management

· A framework that is independent of organizational structures, architectures, or technologies

The five volumes that makeup the ITIL Lifecycle are:

· ITIL Service Strategy

· ITIL Service Design

· ITIL Service Transition

· ITIL Service Operation

· ITIL Continual Service Improvement

Each book addresses capabilities having direct impact on the performance of a service provider. In this course, you will notice that the structure of the ITIL core is in the form of a lifecycle. This ITIL core is expected to provide structure, stability, and strength to service management capabilities with durable principles, methods, and tools.

The best practices guidance in ITIL can be adapted for changes used in various business environments and organizational strategies.

Service Lifecycle

The service management lifecycle is an approach that emphasizes coordination, integration, and control across processes, functions, and systems that is necessary to manage IT services. The service management lifecycle includes service strategy, design, transition, operation and continuous improvement stages that correspond to the 5 core ITIL publications.

Its important to note that although each of the 26 processes are identified within one book of the lifecycle, they may be applied at across the service management lifecycle. So for example: Service Level management is described in the service design book where it sets the targets that are the design points for a service. During operation service levels are monitored and responded to, and it is also applied during continual service improvement to ensure that service levels are attained The ITIL framework also describes the high-level integration of processes. For example, service management provides inputs to incident management for the prioritization of

incidents and receives information from the Service Measurement and reporting process to understand if service levels are being attained..

It is important to understand that the service management lifecycle is not quite the same as the outsourcing lifecycle. In it service management when we are in transition, we are moving a new or changed service into a production environment using change and release management. In outsourcing transition is moving control of a customer's infrastructure to a service provider. So for example, its setting up how change management will be performed rather than performing actual change management. This is a very important distinction to think about when you are talking to customers about ITIL.

IT shows the 5 books in ITIL V3 and the 26 processes and 5 functions. A function is a group of people and the the tools they use to carry our a process or activity.

Processes:

Strategy:

Strategy Generation:

Service Strategy provides guidance on how to design, develop and implement IT services as strategic assets and organizational capabilities

IT Financial Management:

IT Financial Management provides a means of understanding and managing costs and opportunities associated with services in financial terms. IT Financial Management includes three basic activities:

Service Portfolio Mgmt:

Service portfolio explains the business needs and the response from providers to those needs. The service portfolio is a combination, or we can say set of services managed by a service provider.

Demand Management:

Service portfolio explains the business needs and the response from providers to those needs. The service portfolio is a combination, or we can say set of services managed by a service provider.

Design:

Service CatalogMgmt:

The Service Catalog is the subset of the Service Portfolio which contains services currently available to customers and users. The Service Catalog is often the only portion of the Service Portfolio visible to customers. The Service Catalog commonly acts as the entry portal for all information services in the live environment

Service Level Mgmt:

The Service Catalog is the subset of the Service Portfolio which contains services currently available to customers and users. The Service Catalog is often the only portion of the Service Portfolio visible to customers. The Service Catalog commonly acts as the entry portal for all information services in the live environment

Capacity Mgmt:

The Service Catalog is the subset of the Service Portfolio which contains services currently available to customers and users. The Service Catalog is often the only portion of the Service Portfolio visible to customers. The Service Catalog commonly acts as the entry portal for all information services in the live environment

Availability Mgmt:

ensures that IT services meet agreed availability goals. It also ensures new or changed service meet availability goals and doesn’t affect the existing services

IT Service Continuity Mgmt

IT Service Continuity Management process (ITSCM) ensures that the IT Service Provider can always provide the minimum, agreed-upon levels of service, by reducing the risk from disaster events to an acceptable level and planning for the recovery of IT services.

Information Security Mgmt

IT Security Management focuses on the protection of five basic qualities of information assets:

1. Confidentiality: Assurance that the asset is only available to appropriate parties
2. Integrity: Assurance that the asset has not been modified by unauthorized parties
3. Availability: Assurance that the asset may be utilized when Required
4. Authenticity: Assurance that the transactions and the identities of parties to transactions are genuine
5. Non-Repudiation: Assurance that transactions, once completed, may not be reversed without approval

Supplier Mgmt

Supplier Management is the process charged with obtaining value for money from third-party suppliers. Supplier Management handles supplier evaluation, contract negotiations, performance reviews, renewals, and terminations.

Transition:

Transition Planning &
Support

Change Management:

The objective of this processing activity is to control the lifecycle of all the changes. The primary objective of Change Management is to enable beneficial changes to be made with minimum disruption to IT services.

Service Asset and Configuration Management:

The objective is to maintain information about Configuration Items required to deliver an IT service, including their relationships.

Release & Deployment:

It includes planning, designing, building, testing and deploying new software and hardware components in the live environment. The primary goal is to ensure that the integrity of the live environment is protected and that the correct components are released.

Service Testing and Validation:

This ensures that deployed releases and the resulting services meet customer expectations, and to verify that IT operations are able to support the new service.

Knowledge Management:

The objective is to gather, analyze, store and share knowledge and information within an organization. The primary purpose of Knowledge Management is to improve efficiency by reducing the need to rediscover knowledge.

Operations:

Service operation ensures that services are being provided efficiently and effectively as per SLAs. It includes monitoring services, resolving incidents, fulfilling requests and carrying out operational tasks.

Event Mgmt:

Event management can be described as the process that monitors all events that occur through the IT infrastructure to allow for normal operation and also to detect and escalate exception conditions.

Incident Mgmt:

The purpose of Incident Management is to restore the service to the previous stage as early as possible. The primary objective of Incident Management is to return the IT service to users as quickly as possible. Incident Management ensures that the agreed levels of service quality are maintained.

Request Fulfillment:

The purpose of Request Fulfilment process is to manage the lifecycle of all service requests from the users.

Problem Mgmt:

The process objective is to manage the lifecycle of all problems. The primary objectives of Problem Management are to prevent Incidents from happening and to minimize the impact of incidents that cannot be prevented.

Access Mgmt:

The objective is to grant authorized users the right to use a service while preventing access to unauthorized users. Access Management is something also referred to as Rights Management or Identity Management. To manage access to services based on policies and actions defined in Information Security Management

Technology Management:

Technical Management provides technical expertise and support for the management of the IT infrastructure.

IT Operations Management:

This function is responsible for managing organization's day-to-day operational activities. This includes job scheduling, backup and restoring activities, print and output management, and routine maintenance

Continual Improvement:

The objective of this component is to use methods from quality management to learn from past successes and failures. The Continual Service Improvement process aims to continually improve the effectiveness and efficiency of IT processes and services in line with the concept of continual improvement adopted in ISO 2000.

7-step Improvement Process

Service Measurement

Service Reporting

The following are functions:

Service Desk

Technology Management

Application Management

IT Operation Management

Facilities Management

The following are ITIL V2 processes incorporated in ITIL V3:

IT Financial Management

Service Level Mgmt

Capacity Mgmt

Availability Mgmt

IT Service Continuity Mgmt

Change Management

Service Asset and Configuration Management

Release & Deployment

Service Desk

Incident Mgmt

Problem Mgmt

Governance Processes:

Mainly belongs to CSI:

Service Management

Service Reporting

Service Improvement

Mainly belongs to Service Strategy:

Demand Management - spreads across service design, service transition and service operation

Strategy Generation

Service Portfolio Management

IT Financial Management - spreads across service design, service transition and service operation

Operational Processes:

Mainly belongs to Service Design: Service Catalogue Management - spreads across service transition and service operation Service Level Management - spreads across service transition and service operation Capacity Management - spreads across service transition and service operation Availablity Management - spreads across service transition and service operation Service Continuity Management - spreads across service transition and service operation Information Security Management - spreads across service strategy, service transition and service operation.Supplier Management - spreads across service strategy, service transition and service operation

Mainly belongs to Service Transition:

Transition planning and support Change Management - spreads across Service strategy, Service Design and service operation Service Asset and Configuration Management - spreads across Service strategy, Service Design and service operation

Release and Deployment Management - spreads across Service Design and Service Operation

Service Validation and Testing Evaluation

Knowledge Management - spreads across Service strategy, Service Design and service operation Mainly belongs to Service Operations:Event Management

Incident Mangement - spreads across Service Transition Request Fulfillment

Problem Management - spreads across Service Transition Operation Management

2.2 Service Design- Service level management, capacity management, continuity management, information security management, service catalogue management, supplier management

Service Design

Service Design Goal: The main goal of the Service Design stage of the lifecycle is the design of new or changed services for introduction into the live environment. A holistic approach should be adopted for all Service The main goals and objectives of Service Design are to:· Design services to satisfy business objectives, based on the quality, compliance, riskand security requirements, delivering more effective and efficient IT and business solutions and services aligned to business needs by coordinating all design activities for IT services to ensure consistency and business focus

Service Design Processes:

Service Design Process

Service Catalogue Management

Service Level Management

Capacity Management

Avaliability Management

IT Service Continuity Management

Information Security Management

Supplier Management

Service Design Overview

· The design of the services, including all of the functional requirements, resources and capabilities needed and agreed The design of Service Management systems and tools, especially the Service Portfolio, for the management and control of services through their lifecycle· The design of the technology architectures and management systems required to provide the services.

· The design of the processes needed to design, transition, operate and improve the services, the architectures and the processes themselves

Service Design Tools:

There are many tools and techniques that can be used to assist with the design of services and their associated components. These tools and techniques enable:

· Hardware design

· Software design

· Environmental design

· Process design

· Data design.

Service Catalogue Management:

The purpose of Service Catalogue Management is to provide a single source of consistent information on all of the agreed services, and ensure that it is widely available to those who are approved to access it.

The goal of the Service Catalogue Management process is to ensure that a Service Catalogue is produced and maintained, containing accurate information on all operational services and those being prepared to be run operationally.

Service Level Management:

Define, document, agree, monitor, measure, report and review the level of IT services provided. Provide and improve the relationship and communication with the business and customers.

Ensure that specific and measurable targets are developed for all IT services.

Monitor and improve customer satisfaction with the quality of service delivered.

Ensure that IT and the customers have a clear and unambiguous expectation of the level of service to be delivered. Service Level Agreement (SLA): Is a written agreement between an IT Service provider and the IT customer, defining the key Service targets and responsibilities of both parties.

Service Level Manager:

The Service Level Manager has responsibility for ensuring that the aims of Service Level Management are met. This includes responsibilities such as:

· Keeping aware of changing business needs. Ensuring that the current and future service requirements of customers are identified, understood and documented in SLA and SLR documents.

Capacity Management

The goal of the Capacity Management process is to ensure that costjustifiable IT capacity in all areas of IT always exists and is matched to the current and future agreed needs of the business, in a timely manner’. To improve the quality and performance of services according the business requirements.

Availability Management

The goal of the Availability Management process is to ensure that the level of service availability delivered in all services is matched to or exceeds the current and future agreed needs of the business, in a cost-effective manner.

IT Service Continuity Management.

The goal of ITSCM is to support the overall Business Continuity Management process by ensuring that the required IT technical and service facilities (including computer systems, networks, applications, data repositories, telecommunications, environment, technical support and Service Desk) can be resumed within required, and agreed, business timescales.

Information Security Management.

The goal of the ISM process is to align IT security with business security and ensure that information security is effectively managed in all service and Service Management activities.

Supplier Management.

The goal of the Supplier Management process is to manage suppliers andthe services they supply, to provide seamless quality of IT service to the business, ensuring value for money is obtained.

The purpose of the Supplier Management process is to obtain value for money from suppliers and to ensure that suppliers perform to the targets contained within their contracts and agreements, while conforming to all of the terms and conditions.

2.3 Service transition – change management, asset management and configuration management, release and deployment, transition planning and support, service validation and testing, evaluation, knowledge management

Service Transition Goal.

How to introduce new Services (or Change the existing Services) with

appropriate balance of:

– Speed.

– Cost.

– Safety.

– Focus on Customer expectations and requirements.

The first group are whole service lifecycle processes that are critical during the transition stage but influence and support all lifecycle stages. These comprise:

· Change Management.

· Service Asset and Configuration Management.

· Knowledge Management.

Processes within Service Transition.

The following processes are strongly focused within the Service Transition stage:

· Transition Planning and Support.

· Release and Deployment Management.

· Service Testing and Validation.

· Evaluation.

Change Management.

The goals of Change Management are to:

· Respond to the customer's changing business requirements while maximizing value and reducing incidents, disruption and re-work. Respond to the business and IT requests for change that will align the services with the business needs.

The objective of the Change Management process is to ensure that changes are recorded and then evaluated, authorized, prioritized, planned, tested, implemented, documented and reviewed in a controlled manner.

Service change: 'The addition, modification or removal of authorized, planned or supported service or service component and its associated documentation .'Change

Request: is a formal communication seeking an alteration to one or more configuration items. This could take several forms, e.g. 'Request for Change' document,

service desk call, Project Initiation Document. Different types of change may require different types of change request. An organization needs to ensure that appropriate procedures and forms are available to cover the anticipated requests. Avoiding a bureaucratic approach to documenting a minor change removes some of the Cultural.

Request for Change (RFC): A formal proposal for a Change to be made . An RFC includes details of the proposed Change, and may be recorded on paper or electronically.

The term RFC is often misused to mean a Change Record, or the Change itself.

Change Proposal: For a major change with significant organizational and/or financial implications, a change proposal may be required, which will contain a full description of the change together with a business and financial justification for the proposed change.

Service Asset and Configuration Management.

The objective is to define and control the components of services and infrastructure and maintain accurate configuration information on the historical, planned and current state of the services and infrastructure.

Provide Information.

Define and Control Configuration Items (CI's).

Protect and ensure integrity of CI's.

The Configuration Model.

Configuration Management delivers a model of the services, assets and the infrastructure by recording the relationships between configuration items. This enables other processes to access valuable information The real power of Configuration Management's logical model of the services and infrastructure is that it is THE model – a single common representation used by all parts of IT. Service Management, and beyond, such as HR, finance, supplier and customers .

Configuration Management System .

To manage large and complex IT services and infrastructures, Service Asset

and

Configuration Management requires the use of a supporting system known as the Configuration Management System (CMS). The CMS holds all the information for CIs within the designated scope. Some of these items will have related specifications or files that contain the contents of the item, e.g. software, document or photograph. For example, a Service CI will include the details such as supplier, cost, purchase date and renewal date for licences and maintenance contracts and the related documentation such as SLAs and underpinning contracts.

Configuration Management Database (CMDB).

A database used to store Configuration Records throughout their Lifecycle. The Configuration Management System maintains one or more CMDBs, and each CMDB stores Attributes of CIs, and Relationships with other CIs.

Management and planning,

There is no standard template for determining the optimum approach for SACM. The management team and Configuration Management should decide what level of Configuration Management is required for the selected service or project that is delivering changes and how this level will be achieved. This is documented in a Configuration Management Plan. Often there will be a Configuration Management Plan for a project, service or groups of services, e.g. network services. These plans define the specific Configuration Management activities within the context of the overarching Service Asset and Configuration Management strategy.

Configuration identification.

· Define and document criteria for selecting configuration items and the components that compose them.· Select the configuration items and the components that compose them based on documented criteria.

· Assign unique identifiers to configuration items Specify the relevant attributes of each configuration item.·Specify when each configuration item is placed under Configuration Management. Identify the owner responsible for each configuration item.

Configuration Control.

Configuration control ensures that there are adequate control mechanisms over CIs while maintaining a record of changes to CIs, versions, location and custodianship/ownership. Without control of the physical or electronic assets and components, the configuration data and information there will be a mismatch with the physical world. No CI should be added, modified, replaced or removed without an appropriate controlling documentation or procedure being followed. Status accounting and reporting.

Each asset or CI will have one or more discrete states through which it can progress. The significance of each state should be defined in terms of what use can be made of the asset or CI in that state. There will typically be a range of states relevant to the individual asset or CIs.

Verification and audit.;-

The activities include a series of reviews or audits to:· Ensure there is conformity between the documented baselines (e.g. agreements, interface control documents) and the actual business environment to which they refer.

· Verify the physical existence of CIs in the organization or in the DML and spares stores, the functional and operational characteristics of CIs and to check that the records in the CMS match the physical infrastructure.

· Check that release and configuration documentation is present before making a release.

Roles and Responsibilities.

Service Asset Manager.

The Service Asset Manager has the following responsibilities:

· Works to the overall objectives agreed with the IT Services Manager; implements the organization's Service Asset Management policy and standards. Evaluates existing Asset Management Systems, plans, implements and manages Changed systems, and provides reporting on progress against plan.

· Agrees to the scope of the Asset Management processes. Develops and manages Asset Management standards, plans and procedures and manages implementation.

Mounts an awareness campaign on procedures, controls Changes to Asset Management methods and processes and communicates these to staff before being implemented. The Service Asset Manager also oversees implementation of new Asset Management Systems.

· Manages the evaluation of proprietary Asset Management tools and recommends suitable tools.· Agrees to which assets will be uniquely identified with naming conventions and compliance.

Configuration Manager.

The Service Configuration Manager has the following responsibilities:

Works to the overall objectives agreed with the IT Services Manager; implements the organization's Configuration Management policy and standards. Evaluates existing Configuration Management Systems, plan, implement and manage Changed systems, and reporting on progress against plan. Agrees scope of the Configuration Management processes. Develops and manages Configuration Management standards, plans and procedures and manage implementation.

· Mounts an awareness campaign on procedures, controls Changes to Configuration Management methods and processes and communicates these to staff before being implemented. The Configuration Manager also oversees implementation of new Configuration Management Systems.

· Manages the evaluation of proprietary Configuration Management tools and recommends suitable tools.· Agrees assets to be uniquely identified with naming conventions and compliance.· Plans population of the CMS. Manages CMS, central libraries, tools, common codes and data . Ensures regular housekeeping of the CMS .

· Provides reports, including management reports, impact analysis reports and Configuration status reports .Release and Deployment Management. Release and

Deployment Management aims to build, test and deliver the capability to provide the services specified by Service Design and that will accomplish the stakeholders' requirements and deliver the intended objectives.

Release:

A collection of hardware, software, documentation, processes or other components required to implement one or more approved Changes to IT Services.

Release Unit

A 'release unit' describes the portion of a service or IT infrastructure that is normally released together according to the organization's release policy. The unit may vary, depending on the type(s) or item(s) of service asset or service component such as software and hardware.

The general aim is to decide the most appropriate release unit level for each service asset or component. An organization may, for example, decide that the release unit for business critical applications is the complete application in order to ensure that testing is comprehensive. The same organization may decide that a more appropriate release unit for a website is at the page level.

The following factors should be taken into account when deciding the appropriate level for release units:

The ease and amount of change necessary to release and deploy a release unit.

The amount of resources and time needed to build, test, distribute and implement a release unit.· The complexity of interfaces between the proposed unit and the rest of the services and IT infrastructure.

· The storage available in the build, test, distribution and live environments. Release packages: Release packages are planned and designed to be built, tested, delivered, distributed and deployed into the live environment in a manner that provides the agreed levels of traceability, in a cost-effective and efficient way.

Deployment Approach.

‘Big bang’ vs phased.

‘Big bang’ option – the new or changed service is deployed to all user areas in one operation. This will often be used when introducing an application change and consistency of service across the organization is considered important. Phased approach – the service is deployed to a part of the user base initially, and then this
operation is repeated for subsequent parts of the user base via a scheduled rollout plan. This will be the case in many scenarios such as in retail organizations for new services being introduced into the stores’ environment in manageable phases.

Push and Pull. A push approach is used where the service component is deployed from the centre and pushed out to the target locations. In terms of service deployment, delivering updated service components to all users – either in bigbang or phased form – constitutes ‘push’, since the new or changed service is delivered into the users’ environment at a time not of their choosing.

A pull approach is used for software releases where the software is made available in a central location but users are free to pull the software down to their own location at a time of their choosing or when a user workstation restarts.

Automation vs manual.

Whether by automation or other means, the mechanisms to release and deploy the correctly configured service components should be established in the release design phase and tested in the build and test stages.

Automation will help to ensure repeatability and consistency. The time required to provide a well-designed and efficient automated mechanism may not always be available or viable. If a manual mechanism is used it is important to monitor and measure the impact of many repeated manual activities as they are likely to be inefficient and error-prone. Too many manual activities will slow down the release team and create resource or capacity issues that affect the service levels. Service V Model.

The V-model approach is traditionally associated with the waterfall lifecycle, but is, in fact, just as applicable to other lifecycles, including iterative lifecycles, such as prototyping, RAD

approaches. Within each cycle of the iterative development, the V-model
concepts of establishing acceptance requirements against the requirements and design can apply, with each iterative design being considered for the degree of integrity and competence that would justify release to the customer for trial and assessment. The test strategy defines the overall approach to validation and testing. It includes the organization of validation and testing activities and resources and can apply to the whole organization, a set of services or an individual service Various controlled environments will need to be built or made available for the different types
and levels of testing as well as to support other transition activities such as training. Existing deployment processes and procedures can be used to build the controlled test
environments. The environments will need to be secure to ensure there is no unauthorized access and that any segregation of duty requirements are met.

knowledge Management.

The purpose of Knowledge Management is to ensure that the right
information is delivered to the appropriate place or competent person at the
right time to enable informed decision. The goal of Knowledge Management is to enable organizations to improve the quality of management decision making by ensuring that reliable and secure information and data is available throughout the service lifecycle. The objectives of Knowledge Management include: · Enabling the service provider to be more efficient and improve quality of service, increase satisfaction and reduce the cost of service.· Ensuring staff have a clear and common understanding of the value that their services provide to customers and the ways in which benefits are realized from the use of those services.

· Ensuring that, at a given time and location, service provider staff have adequate information on: Who is currently using their services.The current states of consumption. Service delivery constraints Difficulties faced by the customer in fully realizing the benefits expected from the service.

2.4 Service Operations –incident management, problem management, access management, event management, request fulfilment , technical management, application management , IT operations management

Service Operation.

Service Operation Goal.

The purpose of Service Operation is to coordinate and carry out the activities and processes required to deliver and manage services at agreed levels to business users and customers. Service Operation is also responsible for the ongoing management of the technology that is used to deliver and support services. Well-designed and well-implemented processes will be of little value if the day-to-day operation of those processes is not properly conducted, controlled and managed. Nor will service improvements be possible if day-to-day activities to monitor performance, assess metrics and gather data are not systematically conducted during Service Operation.

The scope of Service Operation includes: The services themselves. Any activity that forms part of a service is included in Service Operation, whether it is performed by the Service Provider, an external supplier or the user or customer of that service.

Service Management processes. The ongoing management and execution of many

Service Management processes are performed in Service Operation, even though a number of ITIL processes (such as Change and Capacity Management) originate at the Service Design or Service Transition stage of the Service Lifecycle, they are in use continually in Service Operation.

Some processes are not included specifically in Service Operation, such as Strategy Definition, the actual design process itself. These processes focus more on longer-term planning and improvement activities, which are outside the direct scope of Service Operation; however, Service Operation provides input and influences these regularly as part of the lifecycle of Service Management.

Technology. All services require some form of technology to deliver them. Managing this technology is not a separate issue, but an integral part of the management of the services themselves.

Value to Business.

Actual delivery of Services.

Relationship with and satisfaction of Customers is improved. Provision of what is required by the Business is provided and planned for. From Customer viewpoint, it is where actual value is seen. Each stage in the ITIL Service Lifecycle provides value to business. For example, service value is modelled in Service Strategy; the cost of the service is designed, predicted and validated in Service Design and Service Transition; and measures for optimization are identified in Continual Service Improvement.

The operation of service is where these plans, designs and optimizations are executed and measured. From a customer viewpoint, Service Operation is where actual value is seen. There is a down side to this, though: Once a service has been designed and tested, it is expected to run within the budgetary and Return on Investment targets established earlier in the lifecycle. In reality, however, very few organizations plan effectively for the costs of ongoing management of services. It is very easy to quantify the costs of a project, but very difficult to quantify what the service will cost

after three years of operation. It is difficult to obtain funding during the operational phase, to fix design flaws or unforeseen requirements – since this was not part of the original value proposition. In many cases it is only after some time in operation that these problems surface.

Most organizations do not have a formal mechanism to review operational services

for design and value. This is left to Incident and Problem Management to resolve – as if it is purely an operational issue. It is difficult to obtain additional funding for tools or actions (including training) aimed at improving the efficiency of Service Operation. This is partly because they are not directly linked to the functionality of a specific service and partly

because there is an expectation from the customer that these costs should
have been built into the cost of the service from the beginning. Unfortunately, the rate of technology change is very high. Shortly after a solution has been deployed that will efficiently manage a set of services, new technology becomes available that can do it faster, cheaper and more effectively.

Once a service has been operational for some time, it becomes part of the baseline of whatthe business expects from the IT services. Attempts to optimize the service or to use new tools to manage it more effectively are seen as successful only if the service has been very problematic in the past. In other words, some services are taken for granted and any action to optimize them is perceived as 'fixing services that are not broken'.Communication.
Good communication is needed with other IT teams and departments, with users and internal customers, and between the Service Operation teams and departments themselves. Issues can often be prevented or mitigated with appropriate communication.

An important principle is that all communication must have an intended purpose or a
resultant action. Information should not be communicated unless there is a clear audience. In addition, that audience should have been actively involved in determining the need for that communication and what they will do with the information.Meetings.

The purpose of meetings is to communicate effectively to a group of people about a common set of objectives or activities. Meetings should be well controlled and brief, and the focus should be on facilitating action. A good rule is not to hold a meeting if the information can be communicated effectively by automated means.

The Operations meeting.

Operations meetings are normally held between the managers of the IT operational
departments, teams or groups, at the beginning of each business day or week. The purpose of this type of meeting is to make staff aware of any issue relevant to Operations (such as change schedules, business events, maintenance schedules, etc.) and to provide an opportunity for staff to raise any issues of which they are aware. This is an opportunity to ensure that all departments in a data centre are synchronized.

Department, group or team meetings These meetings are essentially the same as the Operations meeting, but are aimed at a single IT department, group or team. Each manager or supervisor relays the information from the Operations meeting that is relevant to their team. Additionally, these meetings will also cover the following:

A more detailed discussion of incidents, problems and changes that are still being workedon, with information about:· Progress to date.

· Confirmation of what still needs to be done.

· Estimated completion times.

· Request for additional resources, if required.

· Discussion of potential problems or concerns.

· Confirmation of staff availability for roster duties.

· Confirmation of vacation schedules.

Customer meetings.

From time to time it will be necessary to hold meetings with customers, apart from the regular Service Level Review meetings. Examples include: Follow-up after serious incidents. The purpose of these meetings is to repair the relationship with the customers, but also to ensure that IT has all the information required to prevent recurrence. Customers also have the opportunity to provide information about unforeseen business impacts. These meetings are helpful in agreeing actions for similar types of incident that may occur in future.

A customer forum, which can be used for a range of purposes, including testing ideas for new services or solutions, or gathering requirements for new or revised services or procedures. A customer forum is generally a regular meeting with customers to discuss areas of common concern.

Service Operation and Continual Service Improvement.

· The ability to be able to prioritize technical faults as well as business demands. This needs to be done during Service Operation, but the mechanisms need to be put in place during Service Strategy and Design. These mechanisms could include incident categorization systems, escalation procedures and tools to facilitate impact assessment for changes.· Data from Configuration and Asset Management to provide

data where required, saving projects time and making decisions more accurate.·Ongoing involvement of SLM in Service Operation.

Quality of Service versus Cost of Service.

Service Operation is required to consistently deliver the agreed upon level of service.

Service Operation must keep costs and resource utilization at an optimal level. An increase in the level of quality usually results in an increase in the cost of service. The relationship between quality and cost of service is not always directly proportional.

For example, improving service availability from 55% to 75% could be straightforward and might not require a huge investment. However, improving availability of the same service from 96% to 99.9% might require large investments in high-availability technology and support staff and tools.

While this may seem straightforward, many organizations are under severe pressure to increase the quality of service while reducing their costs. In slide, the relationship between cost and quality is sometimes inverse. It is possible (usually inside the range of optimization) to increase quality while reducing costs. This is normally initiated within Service Operation and carried forward by Continual Service Improvement. Some costs can be reduced incrementally over time, but most cost savings can be made only once. For example, once a duplicate software tool has been eliminated, it cannot be eliminated again for further cost savings.

Achieving an optimal balance between cost and quality is a key role of Service Management. There is no industry standard for what this range should be, since each service will have a different range of optimization, depending on the nature of the service and the type of business objective being met. For example, the business may be prepared to spend more to achieve high availability on a mission-critical service, while it is prepared to live with the lower quality of an administrative tool.

Determining the appropriate balance of cost and quality should be done during the Service Strategy and Service Design Lifecycle phases, although in many organizations it is left to the

Service Operation teams – many of whom do not generally have all the facts or authority to be able to make this type of decision.

Alert: A warning that a threshold has been reached, something has changed, or a Failure has occurred.

Event: An event can be defined as any detectable or discernible occurrence that has significance for the management of the IT Infrastructure or the delivery of IT service and evaluation of the impact a deviation might cause to the services Incident: An unplanned interruption to an IT service or reduction in the quality of an IT service. Failure of a configuration item that has not yet impacted service is also an incident.

Service Request: A request from a User for information, or advice, or for a Standard Change or for Access to an IT Service. Problem: A cause of one or more Incidents.

Workaround: A temporary way of overcoming the difficulties. Technical Management.

It is the custodian of technical knowledge and expertise related to managing the IT Infrastructure. In this role, Technical Management ensures that the knowledge required to design, test, manage and improve IT services is identified, developed and refined. IT Operations Management IT Operations Management is responsible for executing the activities and performance standards defined during Service Design and tested during Service Transition.

In business, the term ‘Operations Management’ is used to mean the department, group or team of people responsible for performing the organization’s day- to-day operational activities – such as running the production line in a manufacturing environment or managing the distribution centres and fleet movements within a logistics organization. The role of Operations Management is to execute the ongoing activities and procedures required to manage and maintain the IT Infrastructure so as to deliver and support IT Services at the agreed levels, which are summarized here:

Operations Control, which oversees the execution and monitoring of the operational activities and events in the IT Infrastructure. This can be done with the assistance of an Operations Bridge or Network Operations Centre. In addition to executing routine tasks from all technical areas, Operations Control also performs the following specific

tasks: · Console Management, which refers to defining central observation and monitoring capability and then using those consoles to exercise monitoring and control

activities.

Application Management.

Application Management is responsible for managing applications throughout their lifecycle. The Application Management function is performed by any department, group or team involved in managing and supporting operational applications. Application Management also plays an important role in the design, testing and improvement of applications that form part of IT services. As such, it may be involved in development projects, but is not usually the same as the Applications Development teams Service Operations Processes.

Event Management.

Incident Management.

Problem Management.

Request Fulfilment.

Access Management.

Event Management.

The ability to detect events, make sense of them and determine the appropriate control action is provided by Event Management. Event Management can be applied to any aspect of Service Management that needs to be controlled and which can be automated. Effective Service Operation is dependent on knowing the status of the infrastructure and detecting any deviation from normal or expected operation. This is provided by good monitoring and control systems, which are based on two types of tools:· active monitoring tools that poll key CIs to determine their status and availability.

IT Operations Management.

Event Monitoring, Provide Initial Response.

Incident Management.

The goal of the Incident Management process is to restore normal service operation as quickly as possible and minimize the adverse impact on business operations, thus ensuring that the best possible levels of service quality and availability are maintained.

Resolution and Recovery.

When a potential resolution has been identified, this should be applied and tested. The specific actions to be undertaken and the people who will be involved in taking the recovery actions may vary, depending upon the nature of the fault.

Incident Closure.The Service Desk should check that the incident is fully resolved and that the users are satisfied and willing to agree the incident can be closed.

Incident Management Roles.

An Incident Manager has the responsibility for:· Driving the efficiency and effectiveness of the Incident Management process.· Producing management information.· Managing the work of incident support staff (first- and second-line).

· Monitoring the effectiveness of Incident Management and making recommendations for improvement.· Developing and maintaining the Incident Management systems.

· Managing Major Incidents.· Developing and maintaining the Incident Management process and procedures.

In many organizations the role of Incident Manager is assigned to the Service Desk.

Supervisor – though in larger organizations with high volumes a separate role may be necessary. In either case it is important that the Incident Manager is given the authority to manage incidents effectively through first, second and third line.

Second line. Many organizations will choose to have a second-line support group, made up of staff with greater (though still general) technical skills than the Service Desk – and with additional time to devote to incident diagnosis and resolution without interference from telephone interruptions.

Third line.Third-line support will be provided by a number of internal technical groups and/or third-party suppliers/maintainers.

Metrics.

The metrics that should be monitored and reported upon to judge the efficiency and effectiveness of the Incident Management process, and its operation, will include:

· Total numbers of Incidents (as a control measure).· Breakdown of incidents at each stage (e.g. logged, work in progress, closed etc).· Size of current incident backlog.

· Number and percentage of major incidents.

· Mean elapsed time to achieve incident resolution or circumvention, broken down by impact code.· Percentage of incidents handled within agreed response time (incident response-time targets may be specified in SLAs, for example, by impact and urgency codes).· Average cost per incident.

· Number of incidents reopened and as a percentage of the total.

· Number and percentage of incidents incorrectly assigned.

· Number and percentage of incidents incorrectly categorized.

· Percentage of Incidents closed by the Service Desk without reference to other levels of support (often referred to as 'first point of contact').

· Number and percentage of incidents processed per Service Desk agent.

· Number and percentage of incidents resolved remotely, without the need for a visit.

· Number of incidents handled by each Incident Model.

· Breakdown of incidents by time of day, to help pinpoint peaks and ensure matching of resources.

Reports should be produced under the authority of the Incident Manager, who should draw up a schedule and distribution list, in collaboration with the Service Desk and support groups handling incidents. Distribution lists should at least include IT Services Management and specialist support groups. Consider also making the data available to users and customers, for example via SLA reports.

Request Fulfilment.

Request Fulfilment is the processes of dealing with Service Requests from the users. The term ‘Service Request’ is used as a generic description

for many varying types of demands that are placed upon the IT Department by the users. Many of these are actually small changes – low risk, frequently occurring, low cost, etc. (e.g. a request to change a password, a request to install an additional software application onto a particular workstation, a request to relocate some items of desktop equipment) or maybe just a question requesting information – but their scale and frequent, low-risk nature means that they are better handled by a separate process, rather than being allowed to congest and obstruct the normal Incident and Change Management processes. Request Fulfilment is the processes of dealing with Service Requests from the users. The objectives of the Request Fulfilment process include:

· To provide a channel for users to request and receive standard services for which a predefined approval and qualification process exists.

· To provide information to users and customers about the availability of services and the procedure for obtaining them.· To source and deliver the components of requested standard services (e.g. licences and software media).· To assist with general information, complaints or comments.Many Service Requests will be frequently recurring, so a predefined process flow (a model) can be devised to include the stages needed to fulfil the request, the individuals or support groups involved, target timescales and escalation paths. Service Requests will usually be satisfied by implementing a Standard Change. The ownership of Service Requests resides with the Service Desk, which monitors, escalates, dispatches and often fulfils the user request.

Request Models. Some Service Requests will occur frequently and will require handling in a consistent manner in order to meet agreed service levels. To assist this, many organizations will wish to create pre-defined Request Models (which typically include some form of pre-approval by Change Management). This is similar in concept to the idea of Incident Models, but applied to Service Requests.

Request Fulfilment Roles.

Initial handling of Service Requests will be undertaken by the Service Desk and Incident

Management staff. Eventual fulfilment of the request will be undertaken by the appropriate Service Operation team(s) or departments and/or by external suppliers, as appropriate. Often, Facilities Management, Procurement and other business areas aid in the fulfilment of the Service Request. In most cases there will be no need for additional roles or posts to be created. In exceptional cases where a very high number of Service Requests are handled, or where the requests are of critical importance to the organization, it may be appropriate to have one or more of the Incident Management team dedicated to handling and managing Service Request

Problem Management.

The primary objectives of Problem Management are to prevent problems and resulting incidents from happening, to eliminate recurring incidents and to minimize the impact of incidents that cannot be prevented. Problem Management includes the activities required to diagnose the root cause of incidents and to determine the resolution to those problems. It is also responsible for ensuring that the resolution is implemented through the appropriate control procedures, especially Change Management and Release Management. Problem Management will also maintain information about problems and the appropriate workarounds and resolutions, so that the organization is able to reduce the number and impact of incidents over time.

In this respect,Problem Management has a strong interface with Knowledge Management, and tools such

as the Known Error Database will be used for both.

Although Incident and Problem Management are separate processes, they are closely related and will typically use the same tools, and may use similar categorization, impact and priority coding systems. This will ensure effective communication when dealing with related incidents and problems. Value to business. Problem Management works together with Incident Management and Change Management to ensure that IT service availability and quality are increased. When incidents are resolved, information about the resolution is recorded. Over time, this information is used to speed up the resolution time and identify permanent solutions, reducing the number and resolution time of incidents.

This results in less downtime and less disruption to business critical systems.

Problem: “The unknown root cause of one or more Incidents”. Workaround: “A temporary way of overcoming technical difficulties (i.e.,Incidents or Problems”). Known Error: “Problem that has a documented Root Cause and a Workaround” . Known Error Database (KEDB): “Database containing all Known Error Records“.

Access Management.

Access Management provides the right for users to be able to use a service or group of services. It is therefore the execution of policies and actions defined in Security and Availability Management. Access Management is effectively the execution of both Availability and Information Security Management, in that it enables the organization to manage the confidentiality, availability and integrity of the organization’s data and intellectual property.

2.5 Continual Service Improvement – 7 step process improvement

Continual Service Improvement.

Continual Service Improvement Goal.

The primary purpose of CSI is to continually align and realign IT services to the changing business needs by identifying and implementing improvements to IT services that support business processes. These improvement activities support the lifecycle approach through Service Strategy, Service Design, Service Transition and Service Operation. In effect, CSI is about looking for ways to improve process effectiveness, efficiency as well as cost effectiveness Consider the following saying about measurements and management:· You cannot manage what you cannot control.

· You cannot control what you cannot measure.

· You cannot measure what you cannot define. If ITSM processes are not implemented, managed and supported using clearly defined goals, objectives and relevant measurements that lead to actionable improvements, the business will suffer. Depending upon the criticality of a specific IT service to the business, the organization could lose productive hours, experience higher costs, loss of reputaton or, perhaps, even a business failure. That is why it is critically important to understand what to measure, why it is being measured and carefully define the successful outcome.

The objectives of CSI are:

· Review, analyse and make recommendations on improvement opportunities in each lifecycle phase: Service Strategy, Service Design, Service Transition and Service Operation. · Review and analyse Service Level Achievement results. · Identify and implement individual activities to improve IT service quality and improve the efficiency and effectiveness of enabling ITSM processes. · Improve cost effectiveness of delivering IT services without sacrificing customer satisfaction.

· Ensure applicable quality management methods are used to support continual improvement activities.

\Deming Cycle (PDCA).

W. Edwards Deming is best known for his management philosophy leading to higher quality, increased productivity, and a more competitive position. As part of this philosophy he formulated 14 points of attention for managers. Some of these points are more appropriate to service management than others. For quality improvement he proposed the Deming Cycle or Circle. This cycle is particularly applicable in CSI. The four key stages of the cycle are Plan, Do, Check and Act, after which a phase of consolidation prevents the circle from rolling back down the hill. Our goal in using the Deming Cycle is steady, ongoing improvement. It is a fundamental tenet of Continual Service Improvement. 7-Step Improvement Process . Fundamental to CSI is the concept of measurement. CSI uses the 7-Step Improvement Process. Which steps support CSI?

It is obvious that all the activities of the improvement process will assist CSI in some way. It is relatively simple to identify what takes places but the difficulty lies in understanding exactly how this will happen. The improvement process spans not only the management organization but the entire service lifecycle. This is a cornerstone of CSI.

1. Define what you should measure

At the onset of the service lifecycle, Service Strategy and Service Design should have
identified this information. CSI can then start its cycle all over again at 'Where are we now?' This identifies the ideal situation for both the Business and IT.

2. Define what you can measure

This activity related to the CSI activities of 'Where do we want to be?' By identifying the new service level requirements of the business, the IT capabilities (identified through Service Design and implemented via Service Transition) and the available budgets, CSI can conduct a gap analysis to identify the opportunities for improvement as well as answering the question 'How will we get there?'

3. Gathering the data

In order to properly answer the 'Did we get there?' question, data must first be gathered (usually through Service Operations). Data is gathered based on goals and objectives identified. At this point the data is raw and no conclusions are drawn.

4. Processing the data

Here the data is processed in alignment with the CSFs and KPIs specified. This means that timeframes are coordinated, unaligned data is rationalized and made consistent, and gaps in data are identified. The simple goal of this step is to process data from multiple disparate sources into an 'apples to apples' comparison. Once we have rationalized the data we can then begin analysis.

5. Analysing the data

Here the data becomes information as it is analysed to identify service gaps, trends and the impact on business. It is the analysing step that is most often overlooked or forgotten in the rush to present data to management.

6. Presenting and using the information

Here the answer to 'Did we get there?' is formatted and communicated in whatever way necessary to present to the various stakeholders an accurate picture of the results of the improvement efforts. Knowledge is presented to the business in a form and manner that reflects their needs and assists them in determining the next steps.

7. Implementing corrective action

The knowledge gained is used to optimize, improve and correct services. Managers identify issues and present solutions. The corrective actions that need to be taken to improve the service are communicated and explained to the organization.

Risk Management.

Risk is defined as uncertainty of outcome, whether positive opportunity or negative threat. Managing risks requires the identification and control of the exposure to risk, which may have an impact on the achievement of an organization’s business objectives.

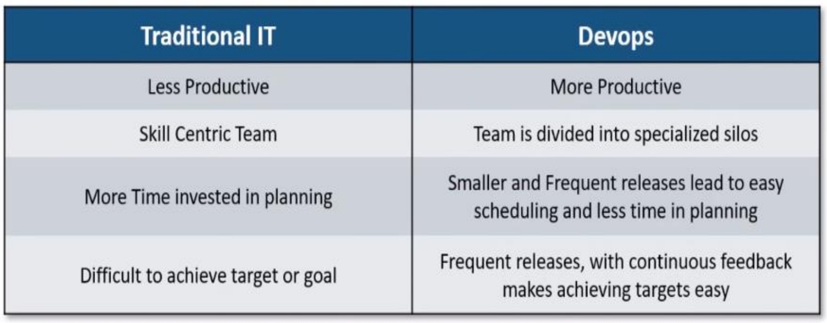
UNIT 3 Introduction to Devops

3.1 Define Devops, What is Devops, SDLC models, Lean, ITIL, Agile, Why Devops?, History of Devops

What is Devops?

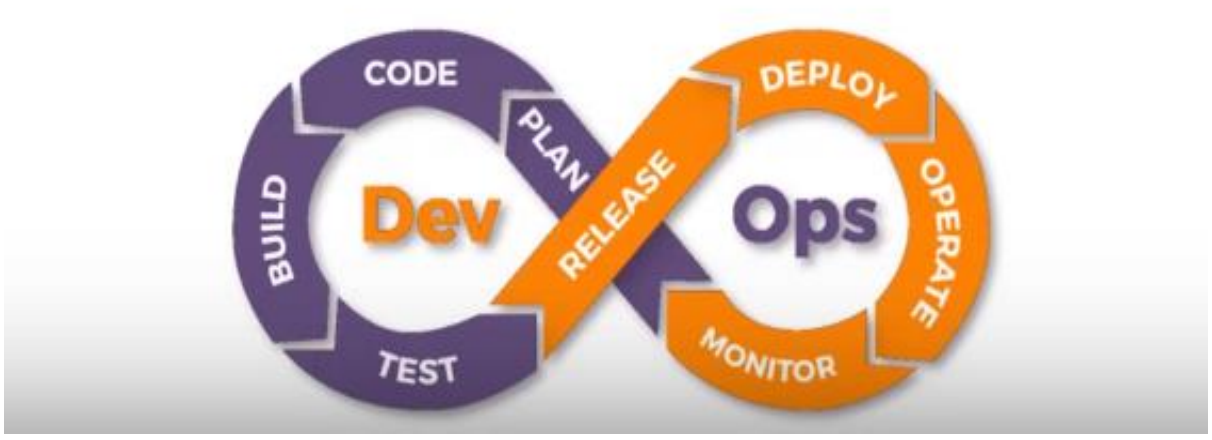
- DevOps is a methodology not a technology.
- Is a culture which promotes collaboration between Development and Operations Team to deploy code to production faster in an automated.
- The word 'DevOps' is a combination of two words 'development' and 'operations.’
- DevOps helps to increases an organization's speed to deliver applications and services. It allows organizations to serve their customers better and compete more strongly in the market.

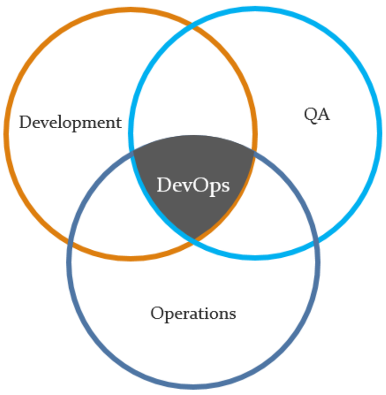
How is DevOps different from traditional IT?

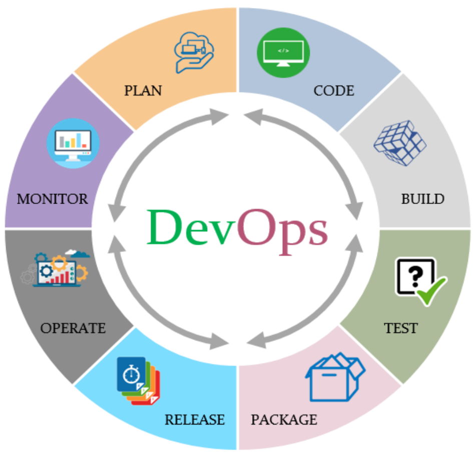


Traditional IT	Devops
Less Productive	More Productive
Skill Centric Team	Team is divided into specialized silos
More Time invested in planning	Smaller and Frequent releases lead to easy scheduling and less time in planning
Difficult to achieve target or goal	Frequent releases, with continuous feedback makes achieving targets easy

Devops Lifecycle

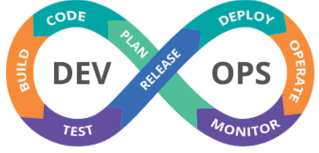


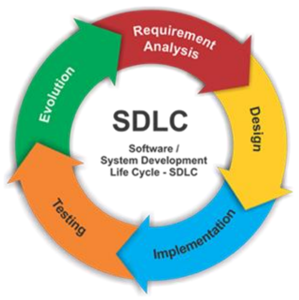




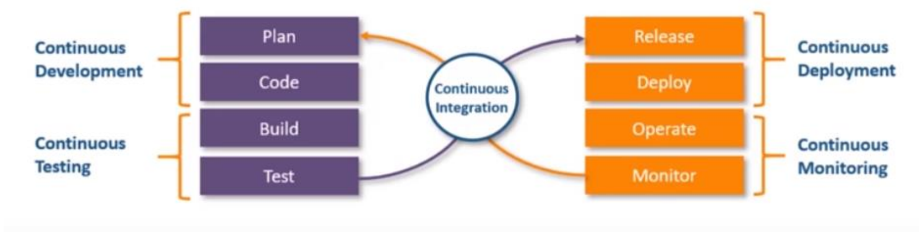
SDLC

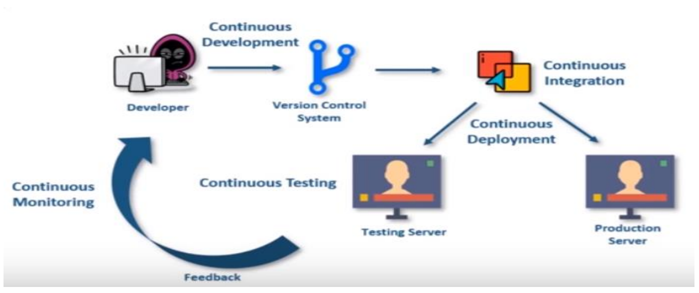
- SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.





How DevOps works ?



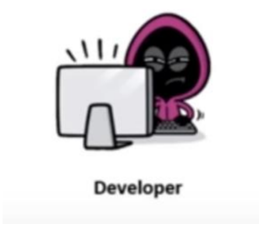


Continuous Development

- This stage involves committing code to version control tools such as **Git** for maintaining the different version of code.

Benefits of CD:

- ➢ Faster delivery of new features
- ➢ Better Quality Product
- ➢ Less Resource Requirement
- ➢ Increased Productivity



Faster delivery of new features

- With continuous development, new features are developed and deployed quickly with automated processes in place.
- As a result, the speed that companies are able to test new version of it, what they developed.
- If any , bugs can be identified and addressed almost immediately.
- By releasing new features rapidly, companies are able to generate a return on investment considerably faster than they would using other deployment methods.

Better Quality Product

- Product quality means to incorporate features that have a capacity to meet client needs.
- Gives customer satisfaction by improving products and making them free from any deficiencies or defects.
- While methods like the waterfall approach require software developers to assume which features are important to the user.
- If we see in DevOps – where in we can deliver the features or bug fix frequency.

Less Resource Requirement

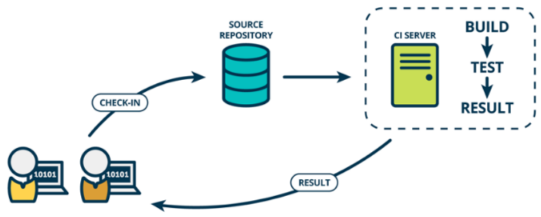
- Large updates can require a considerable amount of Dev in coding, deploying and testing.
- Increase overall efficiency and decrease the amount of time and Dev that is necessary to properly deploy new software features.

Increased Productivity

- Since the feedback loop is expedited, developers are given a full development workflow.
- By receiving continuous feedback from various stakeholders throughout the process.

- Developers are always provided with further tasks to better optimize the software.

Continuous Development



Continuous integration

- This stage is a critical point in whole DevOps lifecycle.
- It deals with integrating the different stages of the DevOps lifecycle.
- Means **CI** is acting like a gateway in whole DevOps process.

Continuous Deployment

- Continuous delivery is a software development practice where code changes are automatically prepared for a release.
- Deploying the source code to particular server like UAT,PROD etc..
- Its involves to ensure software are configured , pushing the build on the environment , where the application can works perfectly.
- We do have few DevOps tools to take care of configuration management like **Ansible, chef** etc.

- **Continuoustesting**

  This stage deals with automated testing of the application pushed by developer. if there is an error , the message is sent back to the integration tool, then tool turns the notification the developer about the error.

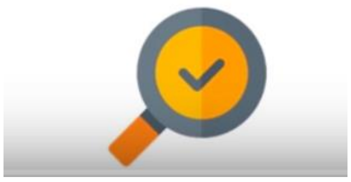
- If the test is success , then the tested code pushes to PROD server or any.

- Continuous Testing needs automated end-to-end testing solutions that integrate existing development processes while excluding errors and enabling continuity throughout SDLC.

Continuous monitoring

Application monitoring is a process that ensures that a software application processes and performs in an expected manner and scope.

This technique routinely identifies, measures and evaluates performance of an application and provides the means to isolate and rectify any abnormalities or shortcomings.



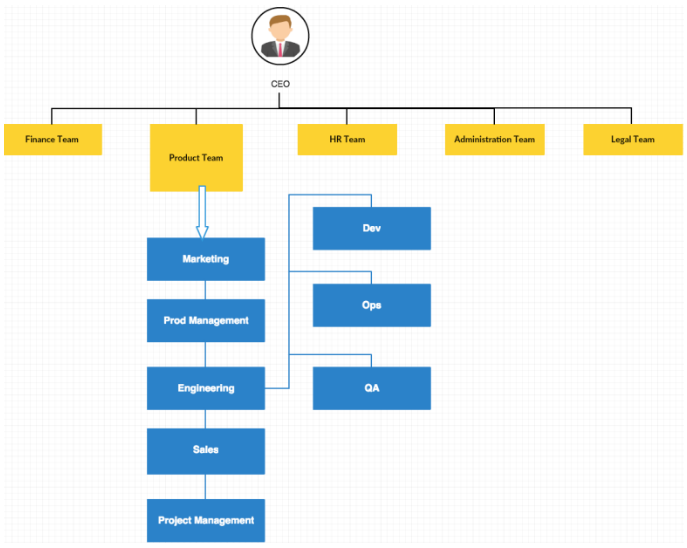
History of Devops

- The idea began in 2008 with a discussion between Patrick Debois and Andrew Clay Shafer concerning the concept of agile infrastructure. However, the idea only started to spread in 2009 with the advent of the first DevOpsDays event held in Belgium.
- DevOps is such an idea. But, it didn't look that way at first. Yet, DevOps has taken off in only a few years to become a major influence on software development and deployment.

3.2 Devops Stakeholders, Devops Goals, Important terminology, Devops perspective, Devops and Agile, Devops Tools

Stakeholders for implementing DevOps

The Blue blocks are the departments focused on the Product. These are the people who I would say are the stakeholders in the DevOps implementation.

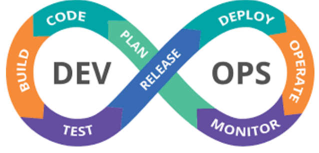


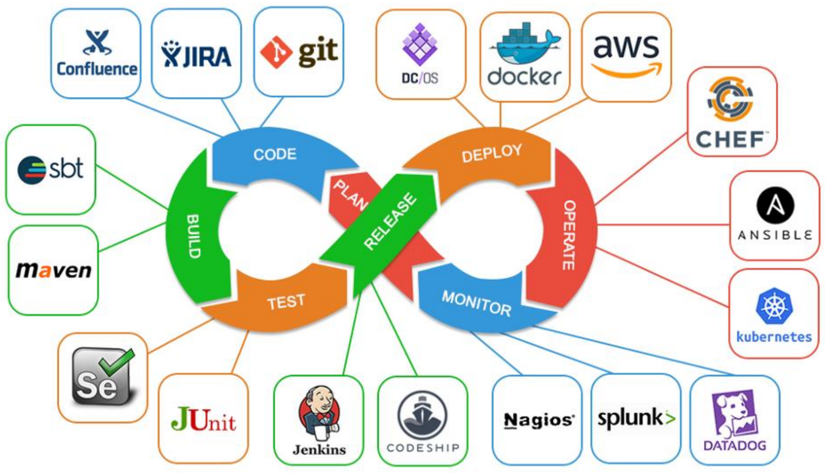
Why do they care about devops?

- **Product Management** : They are interested in testing the different variants of the product and evolve the best of it, in order to do that they need a good devops support so that they can see the changes right away in prod with all the bells and whistles (tracking, monitoring, alerting).
- **Project Management** : Time is of essence for them. When they know that if devops mindset is in place they can smile as they know that code is shipped from the developers laptop to production on a shuttle.
- **Marketing** : They need to know that the product evolution in this shop is on steroids. We can overhaul the product and ship it overnight.
- **Engineering :** Is the technical team of developers and managers who write the code and create the front end, so the clearer the guidance they get upfront, the better. That doesn't mean micromanaging from the product **team.**
- **Customer Success** : ETA’s for resolving issues in minutes, amazing monitoring and ability to change prod in minutes and maintain the stable state sounds like a recipe to delight the customers, hence devops is important for the customer success teams.

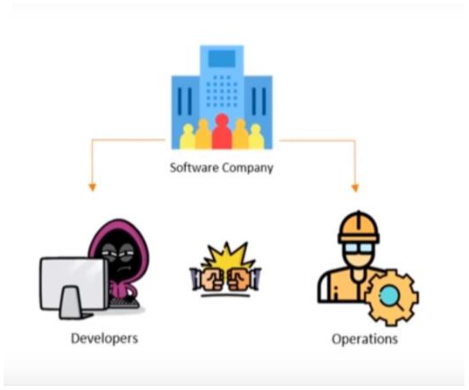
DevOps Tools

- Git and GitHub – Source code management (Version Control System)
- Jenkins – Automation server, with plugins built for developing CI/ CD pipelines
- Chef – Configuration Management and Deployment
- Ansible – Configuration Management and Deployment
- Nagios – Continuous Monitoring
- Maven - Build automation tool





Why DevOps?



- Although, the software quality was improved.
- We still had a lack of efficiency among Dev team.
- Dev and Ops teams are apart of development team.

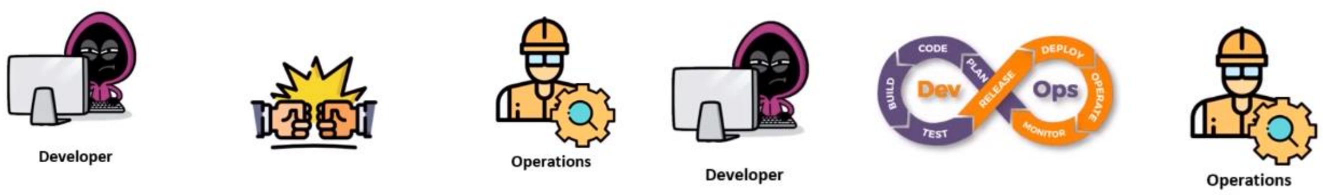
Who is Developer and Ops?

- A developer is someone who writes and debugs code to applications.
- Once the development activity is completed, then developer will pass the code to Ops team.

- The operations team role to test the code , and provide feedback to developers in case of any bugs – The feedback would be passed to developer, who developed the code.
- The Operations team has quite a significant role in the DevOps environment. A huge chunk of DevOps involves configuring, managing, provisioning and monitoring servers.

Problems with Dev and Ops.

Solved using DevOps!



Metrics

- Frequency of deployments.
- Lead time to change
- Mean time to recover
- Change failure rates

Devops perspective

- **Who is a DevOps Engineer?**
    - DevOps Engineer is somebody who understands the Software Development Lifecycle and has the outright understanding of various automation tools for developing digital pipelines (CI/ CD pipelines).
    - DevOps Engineer works with developers and the IT staff to oversee the code releases. They are either developers who get interested in deployment and network operations or sysadmins who have a passion for scripting and coding and move into the development side where they can improve the planning of test and deployment.

Devops Goals

- Fast Development,Quality Assurance, Deployment .

- Faster time to market
- Iteration & Continuous Feedback (strong and continuous communication between stakeholders — the end users and customers, product owners, development, quality assurance, and production engineers)

Lean vs Agile

- "Lean" and "Agile" are terms references to software development methodologies, project management, or
- styles.

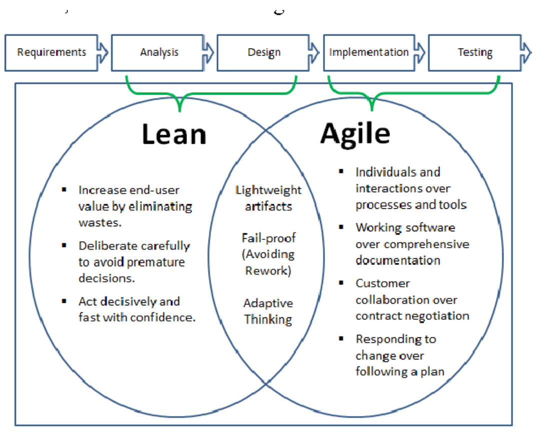
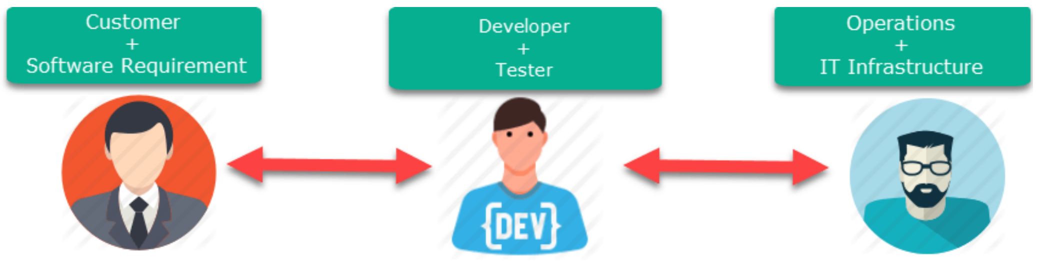

Requirements
Analysis
Design
Implementation
Testing
Lean
Agile
Increase end-user value by eliminating wastes.
Deliberate carefully to avoid premature decisions.
Act decisively and fast with confidence.
Lightweight artifacts
Fail-proof (Avoiding Rework)
Adaptive Thinking
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

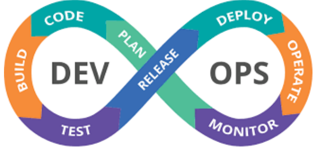
Figure 1. Lean and agile landscape

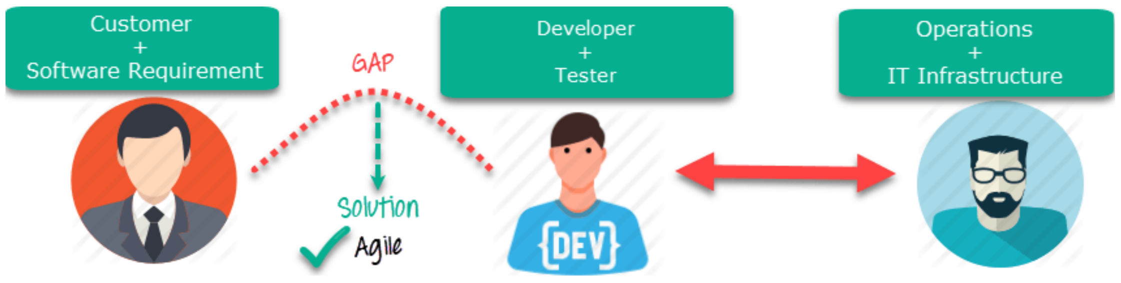
Agile Vs. DevOps

- Stakeholders and communication chain in a typical IT process.

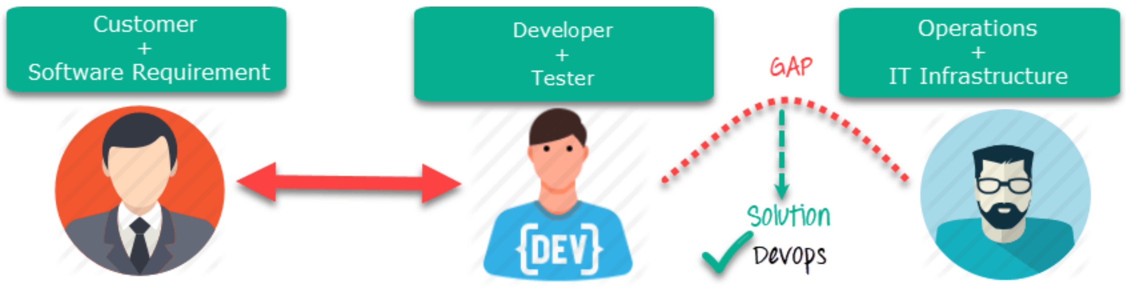


- Agile addresses gaps in Customer and Developer communications.



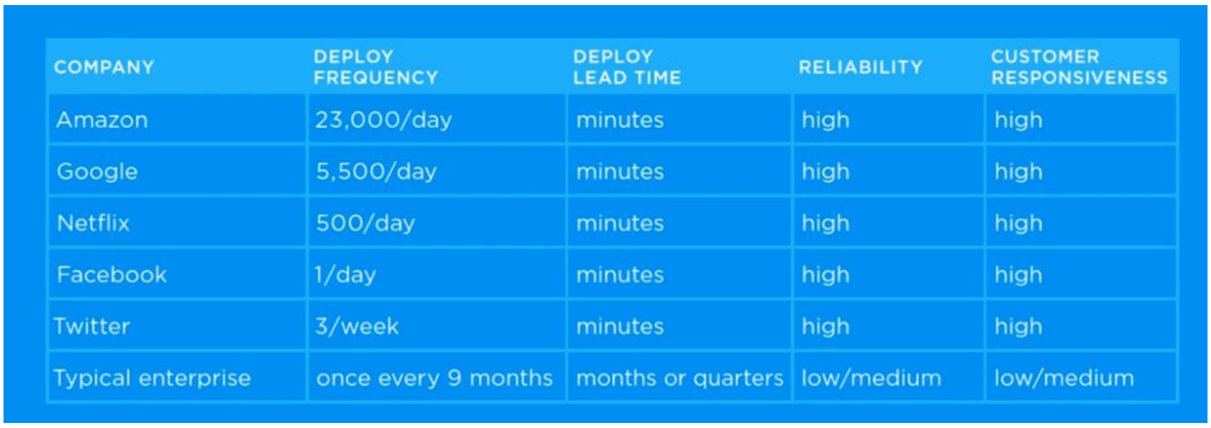

Customer + Software Requirement
GAP
Solution
Agile
Developer + Tester
{DEV}
Operations + IT Infrastructure

- DevOps addresses gaps in Developer and IT Operations communications


Customer + Software Requirement
Developer + Tester
{DEV}
GAP
Solution
Devops
Operations + IT Infrastructure

3.3 GIT: Version Control- Introduction, What is Git, About Version Control System and Types, Difference between CVCS and DVCS, A short history of GIT, GIT Basics, GIT Command Line

Revision control system

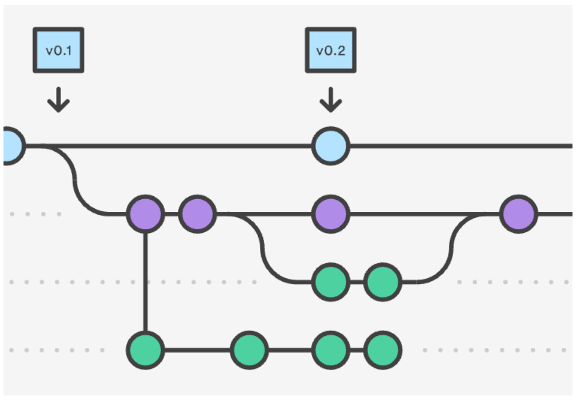


COMPANY	DEPLOY FREQUENCY	DEPLOY LEAD TIME	RELIABILITY	CUSTOMER RESPONSIVENESS
Amazon	23,000/day	minutes	high	high
Google	5,500/day	minutes	high	high
Netflix	500/day	minutes	high	high
Facebook	1/day	minutes	high	high
Twitter	3/week	minutes	high	high
Typical enterprise	once every 9 months	months or quarters	low/medium	low/medium

VCS

- Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.
- Functions of a VCS –

- Allows developers to work simultaneously.
- Does not allow overwriting each other’s changes.
- Maintains a history of every version.

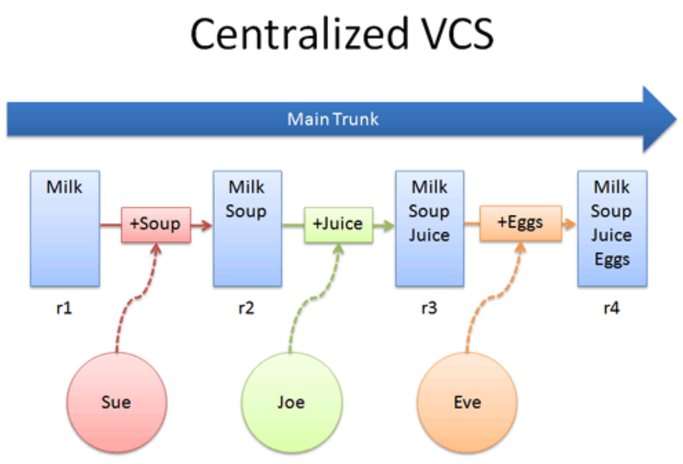


Types of VCS

- Centralized version control system (CVCS).
- Distributed/Decentralized version control system (DVCS).

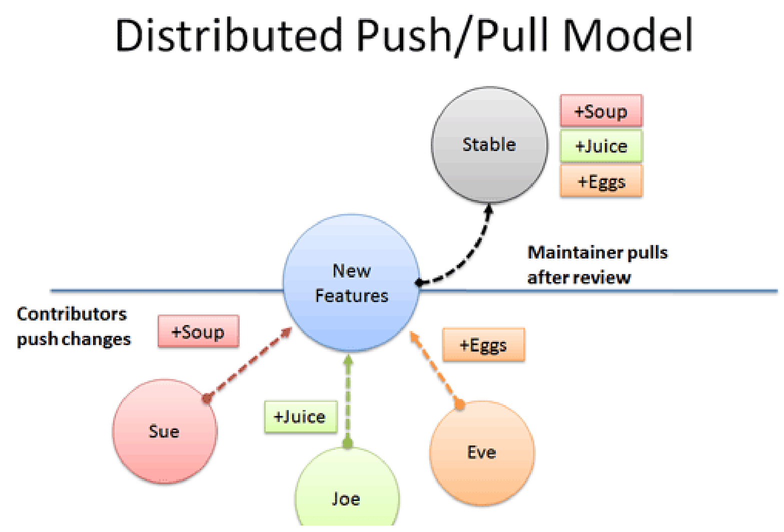
CVCS

- Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. But the major drawback of CVCS is its single point of failure, i.e., failure of the central server. Unfortunately, if the central server goes down Developers wont “Check in or Check out". Since the CVCS server is down and not accessible.



DVCS

- DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the server goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository.
- Git does not rely on the central server and that is why you can perform many operations when you are offline.
- You can commit changes, create branches, view logs, and perform other operations when you are offline. You require network connection only to publish your changes and take the latest changes.


Distributed Push/Pull Model
Stable
+Soup
+Juice
+Eggs
Maintainer pulls after review
New Features
Contributors push changes
+Soup
+Eggs
+Juice
Sue
Joe
Eve

3.4 Git Essentials, repository, Cloning, check-in and committing, Fetch pull and remote, Branching

What is repository?

- In software development, a repository is a central file storage location. It is used by version control systems to store multiple versions of files. While a repository can be configured on a local machine for a single user, it is often stored on a server, which can be accessed by multiple users.
- A repository contains three primary elements — a **trunk, branches, and tags**.
- The **trunk** contains the current version of a software project.
- **Branches** are used to store new versions of the program. A developer may create a new branch whenever he makes substantial revisions to the program. If a branch contains unwanted changes, it can be discontinued. Otherwise, it can be merged back into the trunk as the latest version.
- **Tags** are used to save versions of a project, but are not meant for active development. For example, a developer may create a "release tag" each time a new version of the software is released.

- Git is created by Linus Torvald.
- Git handles merge different source code from different people and its maintaining the version of source code.
- Its called as **distributed version control system**.

Short history of GIT

- The Linux kernel is an open source software project of fairly large scope. For most of the lifetime of the Linux kernel maintenance (1991–2002), changes to the software were passed around as patches and archived files.
- In 2002, the Linux kernel project began using a proprietary DVCS called BitKeeper.

- It's important to note that the terms "checkin" and "checkout" have different meanings depending on the type of SCM system.
- **Checkin** code means to upload code to mentioned/required branch repository so that its administrator/reviewer can review the code and finally update/merge the project version.
- **Checkout** code is the exactly opposite which means to download a copy of code from the repository.
- Centralized systems like TFVC, Subversion, and Clearcase use "exclusive" checkouts.

- For example, on GitHub, cloning means (similar to) checking-out code and pull request means (similar to) checking-in code.
- Distributed systems like git have a "checkout" command, but it means something completely different. git checkout is used to switch between branches when working with a local repository.
- In Git terms, a "checkout" is the act of switching between different versions of a target entity. The git checkout command lets you navigate between the branches created by git branch. Checking out a branch updates the files in the working directory to match
- There is no such gitcheckin command in git

Advantages of Git

- Free and open source
- Fast and small
- Security
- No need of powerful hardware
- Easier branching

DVCS Terminologies

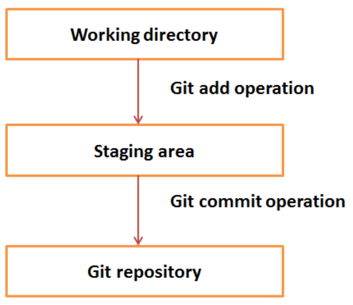
- Local Repository: Every VCS tool provides a private workplace as a working copy.
- Developers make changes in their private workplace and after commit, these changes become a part of the repository. Git takes it one step further by providing them a private copy of the whole repository.

-

  Working Directory and Staging Area or Index: The working directory is the place where files are checked out. In other CVCS, developers generally make modifications and commit their changes directly to the repository. But Git uses a different strategy.
- Whenever you do commit an operation, Git looks for the files present in the staging area. Only those files present in the staging area are considered for commit and not all the modified files.

Workflow of Git

1. You modify a file from the working directory.
2. You add these files to the staging area.
3. You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.



Basic Operations

- **git add**
- Puts current working files into the stage (aka index or cache)
- **git checkout**

- Replaces the current working files with files from a branch.
- **git checkout -b**
- Creates a new local branch from the current branch's tip.
- **git clone**
- Clone an existing repository into a new directory.
- **git commit**
- Commits staged changes to a local branch
- **git commit -a**
- Commits all modified files to a local branch (shorthand for "git add" followed by "git commit" for each modified file)
- **git fetch**
- Downloads changes from a remote repository into the local clone
- **git merge**
- Merges files from a given branch into the current branch.
- **git pull**
- Fetches remote changes ***on the current branch*** into the local clone, and merges them into the current working files.
- **git push**
- Uploads changes from ***all*** local branches to the respective remote repositories.

UNIT 4 DevOps Tools

4.1 Chef for configuration management -Overview of Chef, Workstation Setup, Organization Setup, Test Node Setup, Node

Objects and Search, Environments, Roles, Attributes, Data bags

Chef

- Chef is a configuration management tool for dealing with machine setup on physical servers, virtual machines and in the cloud. Many companies use Chef software to control and manage their infrastructure.
- Chef is a open source tool developed by Opscode and its written in Ruby and Erlang.
- We use Ruby to automate the process.
- If we look at DevOps life cycle – we could see that Chef is falls under operations and deployments.

Large companies changing their infrastructure that requires to be configured and maintained time to time.

That's when we use chef to automate the complete process

Why to useChef?

➢ Continuous deployment

- Software is deployed continuously enabling a company to keep the infrastructure up to date as per current market requirements.

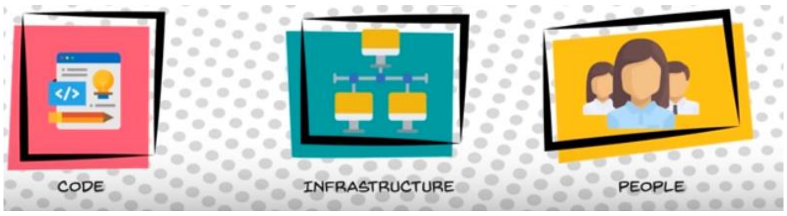
➢ Adapts to cloud

Chef easily adapts to most of the cloud services exists in the current market.

What is chef infrastructure?

What is chef infrastructure?

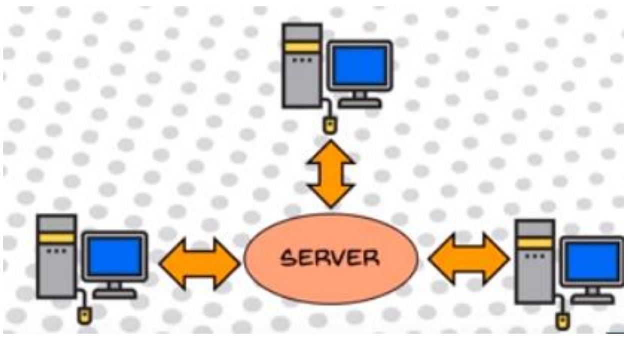
- **Chef Infra** is a powerful automation platform that transforms **infrastructure** into code. Whether you're operating in the cloud, on-premises, or in a hybrid environment, **Chef Infra** automates how **infrastructure** is configured, deployed, and managed acrossConfiguration management is a collection of engineering practices that provides systematic way to manage the entities for efficient deployment.
- Entities are:
- Code
- Infrastructure
- People
- your network, no matter its size.



- **Code**: Basically the code what the system administrator write for configuring different system’s.
- Codes needs to updated whenever the infrastructure needs new configuration or updates in operating system.
- **Infrastructure**: collect of system’s and the server.
- As the required of companies change , we have upgrade our infrastructure.
- **People**: Teams to take care of your infrastructure.

Types of Configuration management

Chef follows under pull configuration



Configuration management

Chef Components

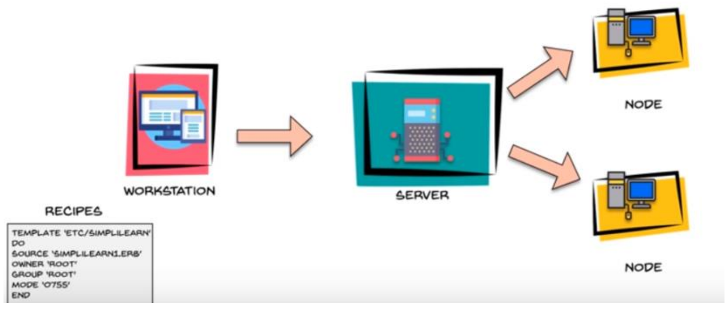
- Chef has major components such as Workstation, Cookbook, Node, Chef-Client, and Chef-Server.
- Chef Server
    - Chef server contains all configuration data and it stores cookbooks, recipes, and metadata that describe each node in the Chef-Client.
    - Configuration details are given to node through Chef-Client.
    - Any changes made must pass through the Chef server to be deployed. Prior to pushing the changes, it verifies that the nodes and workstation are paired with the server through the use of authorization keys, and then allow for communication between workstations and nodes.
    - Chef server can be hosted on local or centralized server.
- Workstation
    - The workstation is used to interact with Chef-server and also to interact with Chef-nodes.
    - Workstation is a place where all the interaction takes place where Cookbooks are created, tested and deployed, and in workstation, codes are tested.

- Chef Cookbook - Cookbooks are fundamental working units of Chef, which consists of all the details related to working units, having the capability to modify configuration and the state of any system configured as a node on Chef infrastructure. Cookbooks can perform multiple tasks. Cookbooks contain values about the desired state of node.
- Key Components of a Cookbook
  - Recipes
  - Metadata
  - Attributes
  - Resources
  - Templates
  - Libraries
  - Anything else that helps to create a system
- **Cookbooks**
- Cookbooks are created using Ruby language.
- A cookbook contains single or multiple recipes which specify resources to be used and in which order it is to be used.
- The cookbook contains all the details regarding the work and it changes the configuration of the Chef-Node.
- Cookbooks are created only on workstations.
- **Nodes**
- Nodes are the systems that require the configuration.

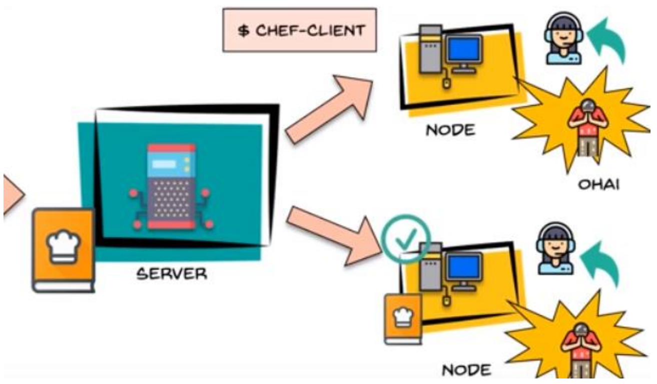
- Nodes are managed by Chef and each node is configured by installing Chef-Client on it. Chef-Nodes are a machine such as physical, virtual cloud etc.
- CookbookKnife: Is a command line tool that uploads the cookbook to the server.
- Ohai:Ohai is used for determining the system state at beginning of Chef run in Chef-Client. It Collects. All the system configuration data.
- Knife: Is a command line tool that uploads the cookbook to the server.
- Ohai:Ohai is used for determining the system state at beginning of Chef run in Chef-Client. It Collects. All the system configuration data.

Chef Client

Chef architecture



```
TEMPLATE 'ETC/SIMPLILEARN'
DO
SOURCE 'SIMPLILEARN1.ERB'
OWNER 'ROOT'
GROUP 'ROOT'
MODE '0755'
END
```

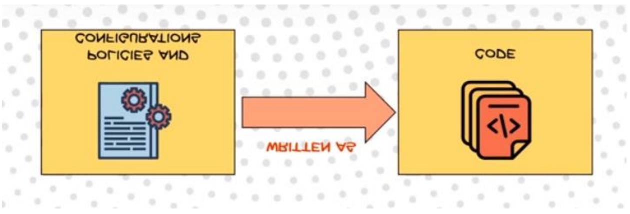


Flavours of chef

Roles of Chef in DevOps

- Chef is for automating and managing the infrastructure.
- Chef IT automation can be done using various Chef DevOps products like Chef-server, Chef-client.
- Chef helps solve the problem by treating **infrastructure as code**. Rather than manually changing anything, the machine setup is described in a Chef recipe.

Infrastructure as code



- **Infrastructure as code** (IaC) is the process of managing and provisioning computer data centers through machine-readable **definition** files, rather than physical hardware configuration or interactive configuration tools.

4.2 AWS OpsWorks, managed configuration management service, Chef Automate, Puppet Enterprise, Stacks. Templated Infrastructure Provisioning- AWS CloudFormation, model it all, automate and deploy, it's just code

- **AWS OpsWorks** (Amazon Web Services **OpsWorks**) is a cloud computing service from Amazon Web Services (**AWS**) that manages infrastructure deployment for cloud administrators.
- The service automates deployment, configurations and operational tasks for distributed applications

- AWS OpsWorks is a configuration management service that helps you configure and operate applications in a cloud enterprise by using Puppet or Chef.

Chef Automate

- Chef Automate is the highly scalable foundation of the Chef automation platform. You can use Chef Automate to create and manage dynamic infrastructure that runs on the AWS Cloud, or manage the servers in your on-premises data center.

- **Cost and Licenses -** You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using the Quick Start.

Puppet Enterprise - AWS OpsWorks

- AWS OpsWorks for Puppet Enterprise is a fully managed configuration management service that hosts Puppet Enterprise, a set of automation tools from Puppet for infrastructure and application management.

- OpsWorks also maintains your Puppet master server by automatically patching, updating, and backing up your server. OpsWorks eliminates the need to operate your own configuration management systems or worry about maintaining its infrastructure.

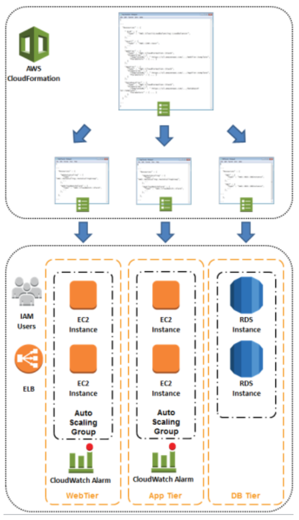
- OpsWorks gives you access to all of the Puppet Enterprise features, which you manage through the Puppet console. It also works seamlessly with your existing Puppet code.

Benefits

- Fully managed puppet master
- Programmable infrastructure
- Scaling made easy & SECURE
- Simple to manage hybrid environments

AWS - Stacks

- A stack is a collection of AWS resources that you can manage as a single unit. In other words, you can create, update, or delete a collection of resources by creating, updating, or deleting stacks.
- All the resources in a stack are defined by the stack's AWS CloudFormation template.



AWS CloudFormation

- AWS CloudFormation provides a common language for you to describe and provision all the infrastructure resources in your cloud environment.
- CloudFormation allows you to use a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all regions and accounts.
- This file serves as the single source of truth for your cloud environment.

Benefits

- Model it all
- Automate and deploy
- It's just code

Model it all

- AWS CloudFormation allows you to model your entire infrastructure in a text file. This template becomes the single source of truth for your infrastructure.
- This helps you to standardize infrastructure components used across your organization, enabling configuration compliance and faster troubleshooting.

AUTOMATE AND DEPLOY

- AWS CloudFormation provisions your resources in a safe, repeatable manner, allowing you to build and rebuild your infrastructure and applications, without having to perform manual actions or write custom scripts.
- CloudFormation takes care of determining the right operations to perform when managing your stack, and rolls back changes automatically if errors are detected.

IT'S JUST CODE

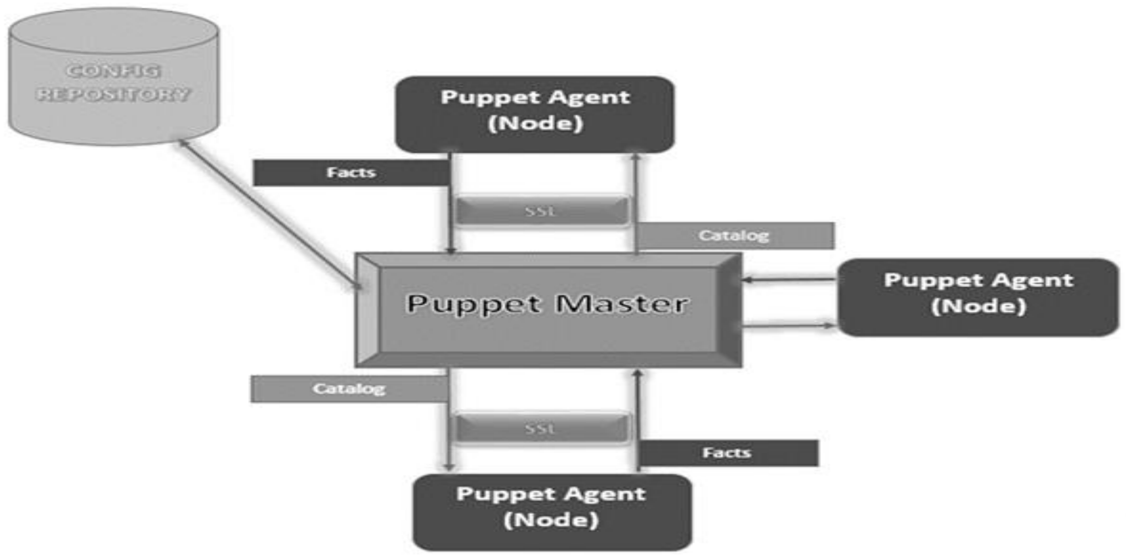
- Codifying your infrastructure allows you to treat your infrastructure as just code. You can author it with any code editor, check it into a version control system, and review the files with team members before deploying into production.

4.3 Puppet for configuration management-What is Puppet? Puppet Architecture, Master and Agents, Puppet Language Basics, Templates, Puppet Forge

- Puppet is a Configuration Management tool that is used for deploying, configuring and managing servers.
- Puppet performs following operations:
    - Defining distinct configurations for each and every host, and continuously checking and confirming whether the required configuration is in place and is not altered (if altered Puppet will revert back to the required configuration) on the host
    - Dynamic scaling-up and scaling-down of machines.
    - Providing control over all your configured machines, so a centralized (master-server or repo-based) change gets propagated to all, automatically.

- Puppet uses a Master Slave architecture in which the Master and Slave communicate through a secure encrypted channel with the help of SSL.
- Puppet works on the client server architecture, wherein we call the server as the Puppet master and the client as the Puppet node. This setup is achieved by installing Puppet on both the client and well as on all the server machines.

Puppet Architecture



- **Puppet Master**: Puppet Master is the key mechanism which handles all the configuration related stuff. It applies the configuration to nodes using the Puppet agent.
- **Puppet Agents** are the actual working machines which are managed by the Puppet master. They have the Puppet agent daemon service running inside them.
- **Config Repository** : This is the repo where all nodes and server-related configurations are saved and pulled when required.
    - **Facts** are the details related to the node or the master machine, which are basically used for analyzing the current status of any node. On the basis of facts, changes are done on any target machine. There are pre-defined and custom facts in Puppet.

- Catalog : All the manifest files or configuration which are written in Puppet are first converted to a compiled format called catalog and later those catalogs are applied on the target machine.

Puppet - Manifest Files

- In Puppet, all the programs which are written using Ruby programming language and saved with an extension of .pp are called manifests. In general terms, all Puppet programs which are built with an intension of creating or managing any target host machine is called a manifest.

Manifest File Workflow

- Puppet manifest consists of the following components –
- **Files** (these are plain files where Puppet has nothing to do with them, just to pick them up and place them in the target location)
- **Resources -** Resources are the fundamental building blocks used to model system state in Puppet. They describe the desired end state of unique elements managed by Puppet on the system. Everything that Puppet manages is expressed as a resource.
- **Templates** (these can be used to construct configuration files on the node).
- **Nodes** (all the definition related to a client node is defined here)
- **Classes - Puppet classes** are defined as a collection of resources, which are grouped together in order to get a target node or machine in a desired state. These **classes** are defined inside **Puppet** manifest files which is located inside **Puppet** modules.

Puppet Templates

- Templates are documents that combine code, data, and literal text to produce a final rendered output. The goal of a template is to manage a complicated piece of text with simple inputs.
- In Puppet, you'll usually use templates to manage the content of configuration files (via the content attribute of the file resource type).

- **Templating languages**
- Templates are written in a templating language, which is specialized for generating text from data.
- Puppet supports two templating languages:
  - ➢ Embedded Puppet (EPP) uses Puppet expressions in special tags. It's easy for any Puppet user to read, but only works with newer Puppet versions. (≥ 4.0, or late 3.x versions with future parser enabled.)
  - ➢ Embedded Ruby (ERB) uses Ruby code in tags. You need to know a small bit of Ruby to read it, but it works with all Puppet versions.

Puppet Forge

- The Puppet Forge is a repository of modules written both by Puppet and by the Puppet user community. These modules solve a wide variety of problems, and using them can save you time and effort.
- The puppet module tool provides a command line interface for managing modules from the Forge.

4.4 Introduction-Understanding continuous integration, Introduction

about Jenkins, Build Cycle, Jenkins Architecture, Jobs, Build Deployments, Securing Jenkins

What is Jenkins?

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose.

Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

Advantages of Jenkins include:

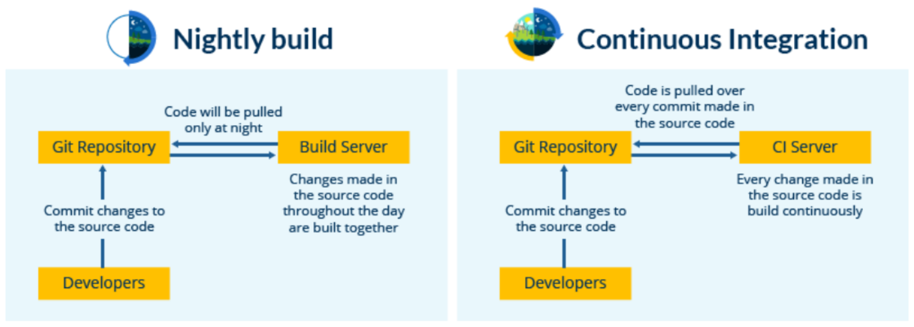
It is an open source tool with great community support.

It is easy to install.

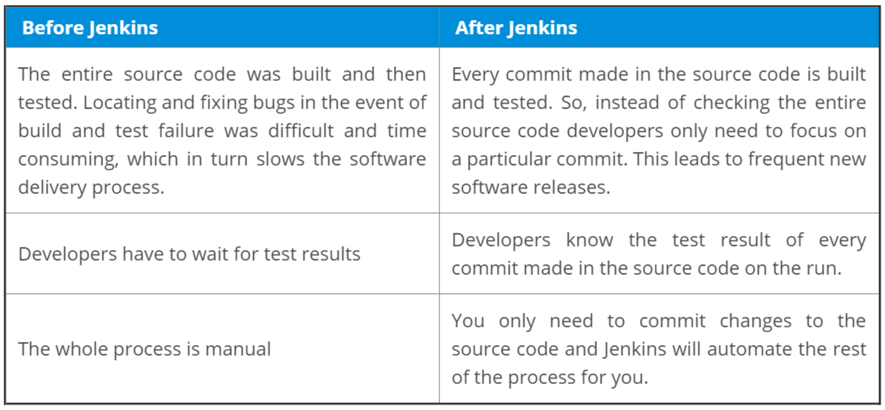
It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share with the community.

It is free of cost.

It is built with Java and hence, it is portable to all the major platforms.


Nightly build
Code will be pulled only at night
Git Repository
Build Server
Commit changes to the source code
Changes made in the source code throughout the day are built together
Developers
Continuous Integration
Code is pulled over every commit made in the source code
Git Repository
CI Server
Commit changes to the source code
Every change made in the source code is build continuously
Developers

[Type here]

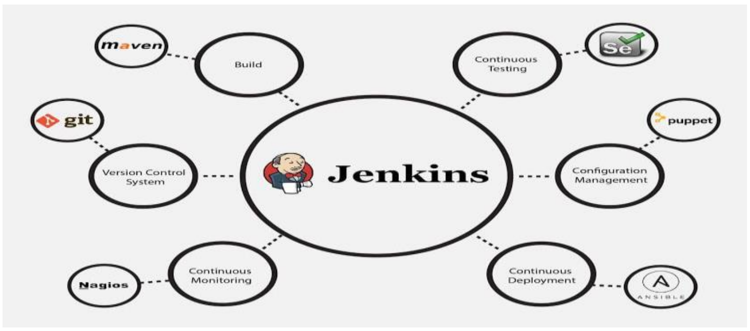


Before Jenkins	After Jenkins
The entire source code was built and then tested. Locating and fixing bugs in the event of build and test failure was difficult and time consuming, which in turn slows the software delivery process.	Every commit made in the source code is built and tested. So, instead of checking the entire source code developers only need to focus on a particular commit. This leads to frequent new software releases.
Developers have to wait for test results	Developers know the test result of every commit made in the source code on the run.
The whole process is manual	You only need to commit changes to the source code and Jenkins will automate the rest of the process for you.

What is a build in software development?

The process of building software is usually managed by a build tool. Builds are created when a certain point in development has been reached or the code has been deemed ready for implementation, either for testing or outright release. A build is also known as a software build or code build.

Build Cycle

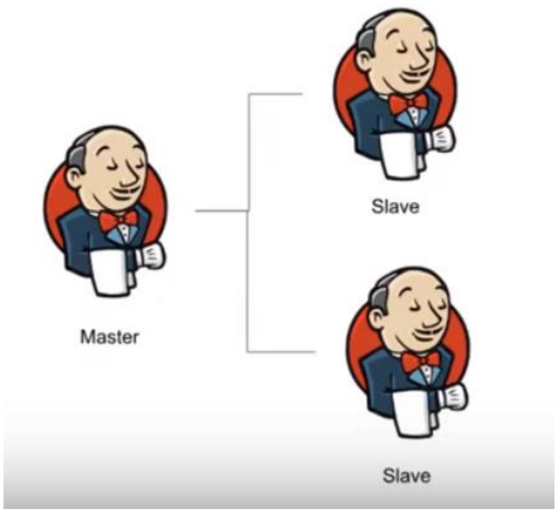


Jenkins Architecture

Single Jenkins server was not enough to meet certain requirements like:

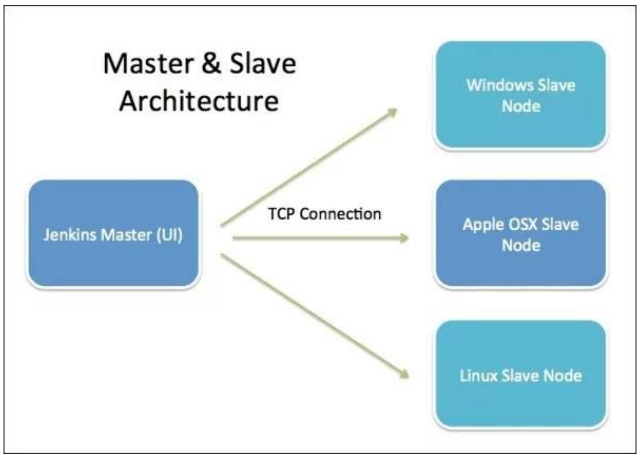
Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.

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 was introduced.

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 Master

Scheduling build jobs.

Dispatching builds to the slaves for the actual execution.

Monitor the slaves (possibly taking them online and offline as required).

Recording and presenting the build results.

A Master instance of Jenkins can also execute build jobs directly.

Jenkins Slave

A Slave is a Java executable that runs on a remote machine.

It hears requests from the Jenkins Master instance.

Slaves can run on a variety of operating systems.

The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.

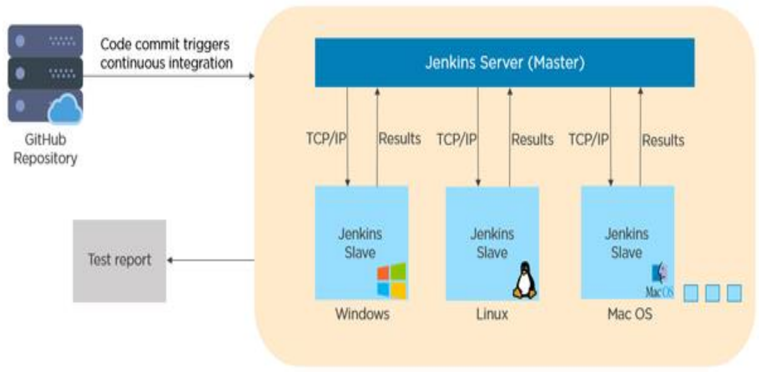
You can configure a project to always run on a particular Slave machine,

or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.

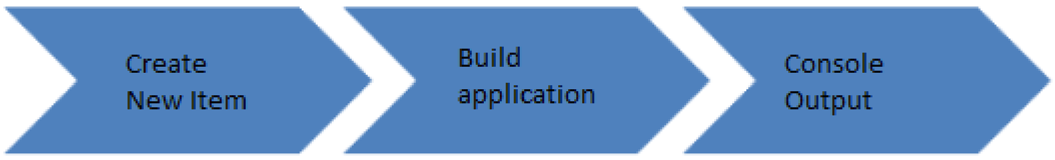
Jenkins checks the Git repository at periodic intervals for any changes made in the source code.

Each builds requires a different testing environment which is not possible for a single Jenkins server. In order to perform testing in different environments Jenkins uses various Slaves as shown in the diagram.

Jenkins Master requests these Slaves to perform testing and to generate test reports.



What is job in Jenkins?



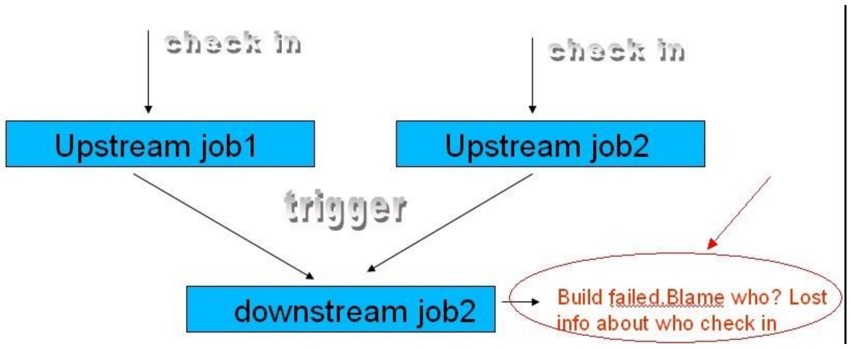
You are requesting Jenkins to perform any task on perform timeframe or immediate basis.

What is a Jenkins freestyle project?

A Jenkins project is a repeatable build job which contains steps and post-build actions. The types of actions you can perform in a build step or post-build action are quite limited. There are many standard plugins available within a Jenkins freestyle project to help you overcome this problem. They allow you to configure build triggers and offers project-based security for your Jenkins project.

Upstream Job vs Downstream Job

The upstream job is the one that is triggered before the actual job is triggered. The downstream job is the one that is triggered after the actual job is triggered.



We can configure the actual job not to be triggered if the upstream job is failed.

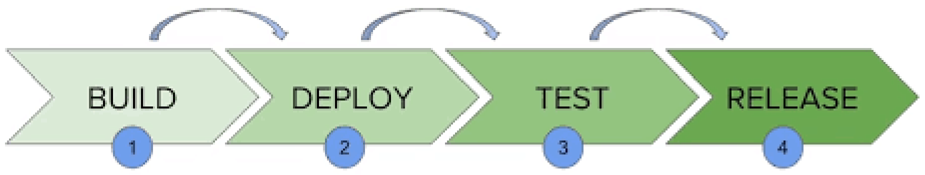
In the same way, we can configure the downstream job not to be triggered if the actual job is failed.

Order of job triggers:

Upstream Job -> Actual Job -> Downstream Job

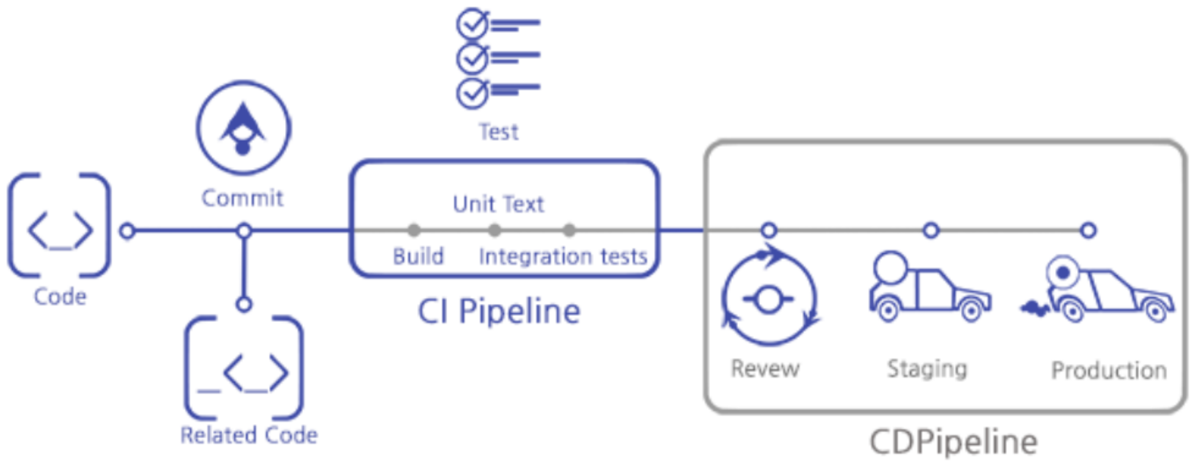
Jenkins Pipeline Concepts

The pipeline is a set of instructions given in the form of code for continuous delivery and consists of instructions needed for the entire build process.



With pipeline, you can build, test, and deliver the application. Node. The machine on which Jenkins runs is called a node.

CI pipeline and CD pipeline



Pipeline concepts

Pipeline

This is a user defined block which contains all the processes such as build, test, deploy, etc. It is a collection of all the stages in a Jenkins file.

```
pipeline {
}
```

Node

A node is a machine that executes an entire workflow. It is a key part of the scripted pipeline syntax.

```
node {
}
```

4.5 Docker– Containers- What is a Docker, Dockers vs. Virtualization, Architecture, Docker Hub-Docker images, AWS ECS- Understanding the containers, images, Docker Networking

What is a Docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Dockers vs. Virtualization

Docker is container based technology and containers are just user space of the operating system. At the low level, a container is just a set of processes that are isolated from the rest of the system, running from a distinct image that provides all files necessary to support the processes. It is built for running applications. In Docker, the containers running share the host OS kernel.

A Virtual Machine, on the other hand, is not based on container technology. They are made up of user space plus kernel space of an operating system. Under VMs, server hardware is virtualized. Each VM has Operating system (OS) & apps. It shares hardware resource from the host.

VMs & Docker – each comes with benefits and demerits. Under a VM environment, each workload needs a complete OS. But with a container environment, multiple workloads can run with 1 OS. The bigger the OS footprint, the more environment benefits from containers. With this, it brings further benefits like Reduced IT management resources, reduced size of snapshots, quicker spinning up apps, reduced & simplified security updates, less code to transfer, migrate and upload workloads.

Docker Networking

Docker includes support for networking containers through the use of network drivers. By default, Docker provides two network drivers for you, the bridge and the overlay drivers. You can also write a network driver plugin so that you can create your own drivers but that is an advanced task.

4.6 Ansible

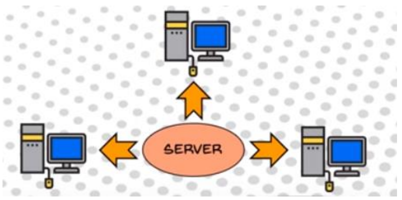
What Ansible is Ansible? Why we use Ansible ?

- **Introduction to Ansible**
- **Creating a playbook with YAML language to define the automation job.**

What is Playbook?

- **YAML , YAML Basics, Variables.**
- **Inventory**
- **Ansible Modules**
- **Ad hoc Commands**
- **Creating a New Role on Ansible**

What is Ansible?



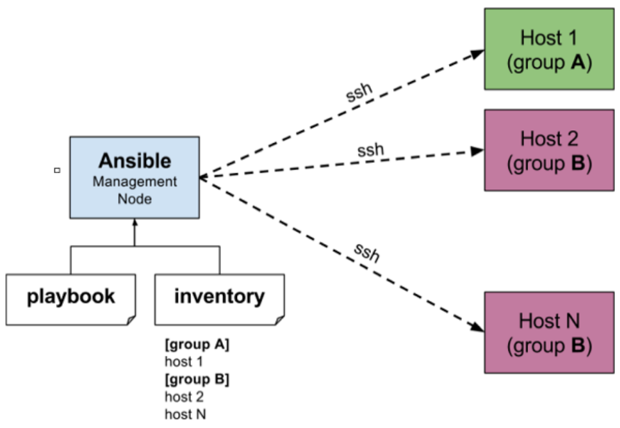
Ansible is an open-source software provisioning, configuration management, and application-deployment .

Ansible is easy to deploy because it does not use any agents or custom security infrastructure.

Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. YAML

Its comes under push based configuration.

How Ansible Works?



Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. Ansible then executes these modules (over SSH by default), and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.

Understanding YAML

“Yet another mark-up language”

YAML uses simple key-value pair to represent the data. The dictionary is represented in key: value pair.

Note – There should be space between : and value.

---

james:

name: james john rollNo: 34 div: B

sex: male

…

Inventory

The Ansible inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The file can be in one of many formats depending on your Ansible environment and plugins.

The inventory file can list individual hosts or user-defined groups of hosts.

Modules

Ansible ships with a number of modules (called the 'module library') that

can be executed directly on remote hosts or through Playbooks. Users can also write their own modules. These modules can control system resources, like services, packages, or files (anything really), or handle executing system commands.

Features of anisible

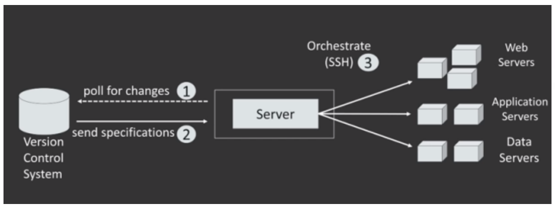
Its agentless

Its build on python

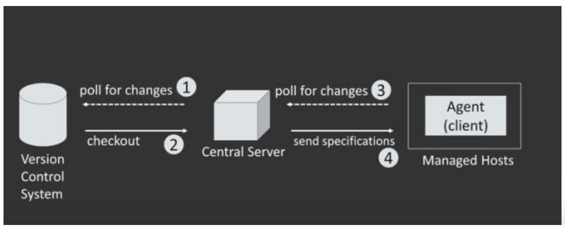
SSH

Push based

Push Based vs Pull Based

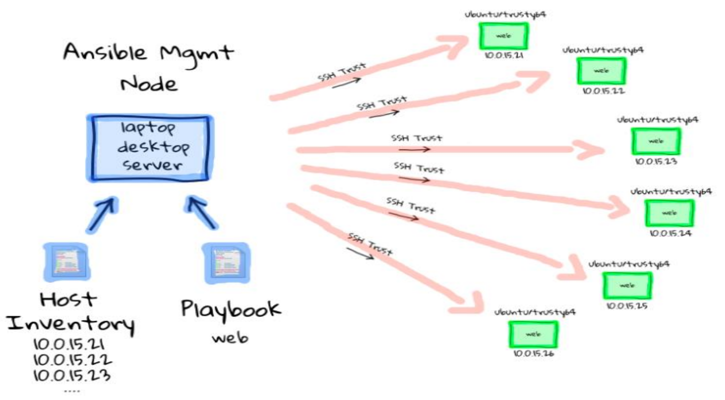


Pull



Ansible architecture

[Type here]



4.7 Maven -What is Maven? Overview about POM

- **Build Life Cycle, Build Profiles , Types of Build Profile.**
- **Managing Plugins**

Jenkins – Maven Setup

Maven is a build automation tool used primarily for Java projects.

Maven addresses two aspects of building software:

It describes how software is built.

it describes its dependencies.

Overview about POM

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project.

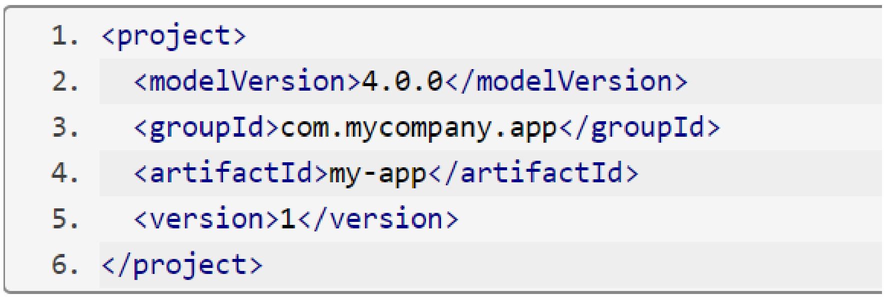
Minimal POM requirement

project root

modelVersion - should be set to 4.0.0 • groupId - the id of the project's group.

artifactId - the id of the artifact (project)

version - the version of the artifact under the specified group

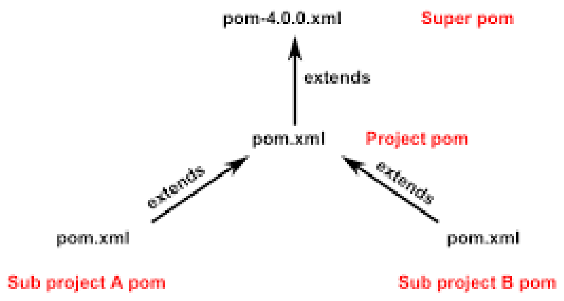


```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
</project>
```

Super POM

The Super POM is Maven's default POM. All POMs inherit from a parent or default (despite explicitly defined or not).

This base POM is known as the Super POM, and contains values inherited by default.



Maven use the effective POM (configuration from super pom + project configuration) to execute relevant goal.

Build Lifecycle

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular 88rtefact (project) is clearly defined. Its necessary to learn a small set of commands to build any Maven project, and the POM will ensure they get the results they desired.

There are three built-in build lifecycles: default, clean and site. The default lifecycle handles your project deployment, the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

Validate – validate the project is correct and all necessary information is available

compile – compile the source code of the project

test – test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

package – take the compiled code and package it in its distributable format, such as a JAR.

Verify – run any checks on results of integration tests to ensure quality criteria are met

install – install the package into the local repository, for use as a dependency in other projects locally

deploy – done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

Build Profiles

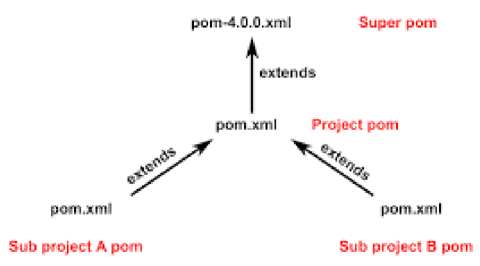
A Build profile is a set of configuration values, which can be used to set or override default values of Maven build.

Using a build profile, you can customize build for different environments such as Production v/s Development environments.

Profiles are specified in pom.xml file using its activeProfiles/profiles elements and are triggered in variety of ways.

Profiles modify the POM at build time, and are used to give parameters different target environments (for example, the path of the database server in the development, testing, and production environments).

Types of Build Profile



Maven Plugins

Maven is actually a plugin execution framework where every task is actually done by plugins. Maven Plugins are generally used to

- create jar file

- create war file
- compile code files
- unit testing of code
- create project documentation
- create project reports

Java project can be compiled with the maven-compiler-plugin's compile-goal by running the following command.

```
mvn compiler:compile
```

4.8 Git &Github

- **Basic Concepts, Life Cycle**
- **What is Branches on Git?**
- **Performing following operations on Git,**
- **Distributed version-control system vs centralized version control system.**
- **Understanding SCM.**

Github

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface.

It also provides access control and several collaboration features.

Creating new repo

Importing new repo

New gist **

New org

New project

Fork

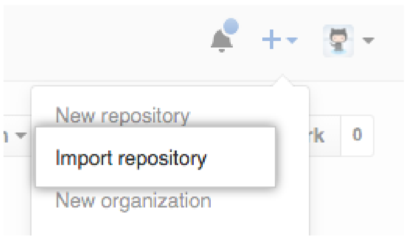
Creating new repo

In revision control systems, a repository is a data structure which stores metadata for a set of files or directory structure.

A central location in which data is stored and managed.

Importing new repo

Just copying the repo from different repo to



New org

Organization accounts allow your team to plan, build, review, and ship software — all while tracking bugs and discussing ideas.

An organization is a collection of user accounts that owns repositories. Organizations have one or more owners, who have administrative privileges for the organization.

New project

Template	Description
Basic Kanban	Track your tasks with To do, In progress, and Done columns
Automated Kanban	Cards automatically move between To do, In progress, and Done columns

[Type here]

Automated kanban with review	Cards automatically move between To do, In progress, and Done columns, with additional triggers for pull request review status
Bug triage	Triage and prioritize bugs with To do, High priority, Low priority, and Closed columns

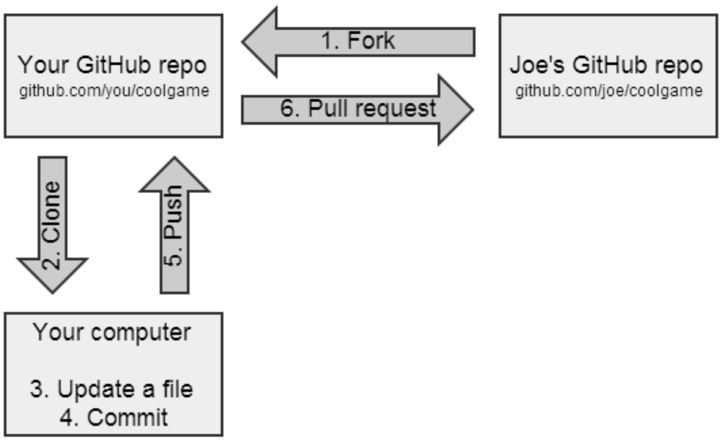
Project boards on GitHub help you organize and prioritize your work. .You can create notes within columns to serve as task reminders, references to issues and pull requests from any repository on GitHub, or to add information related to the project board.

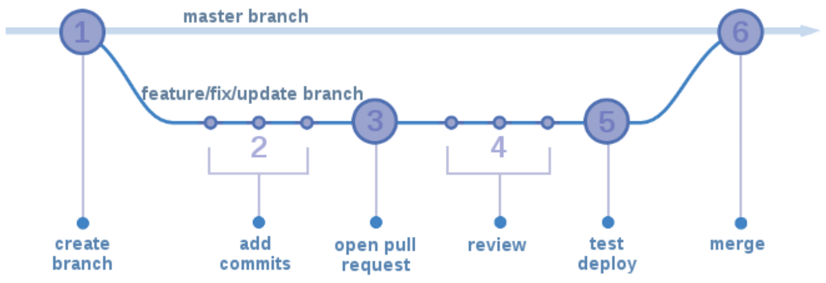
Template	Description
Basic kanban	Track your tasks with To do, In progress, and Done columns
Automated kanban	Cards automatically move between To do, In progress, and Done columns
Automated kanban with review	Cards automatically move between To do, In progress, and Done columns, with additional triggers for pull request review status
Bug triage	Triage and prioritize bugs with To do, High priority, Low priority, and Closed columns

Fork

Forking Workflow is commonly used in public open-source projects. Forking is a git clone operation executed on a server copy of a projects repo.

Work is done to complete the new feature and git commit is executed to save the changes. You then push the new feature branch to your remote forked repo.





GitHub news!!

Starting January 7, 2019 - New GitHub: Announcing unlimited free private repos and unified Enterprise offering.

GitHub Free now includes unlimited private repositories. For the first time, developers can use GitHub for their private projects with up to three collaborators per repository for free.

A collaborator is someone who has been granted write access to the main repository of a project.

A contributor is a person who isn't explicitly a member of your organization or personal account, but wants to contribute to your project via pull requests and creating new issues.

4.9 Nagios

What is Nagios? Why Continuous Monitoring on DevOps?Install Nagios Core, Nagios Plugins And NRPE (Nagios Remote Plugin Executor)

- Nagios is an open source monitoring system for computer systems. It was designed to run on the Linux operating system and can monitor devices running Linux, Windows and Unix operating systems (OSes).
- Nagios software runs periodic checks on critical parameters of application, network and server resources.

Is Nagios free or not?

- Yes! Nagios XI is available free of charge for monitoring small environments.
- Nagios XI installations with a free license are limited to monitoring seven (7) hosts (nodes).
- There is no limitation on the number of services that can be monitored with a free license.

Nagios' benefits

- Scales well for simple, static setups.
- Easy to understand plugin architecture.
- Forks strive to maintain compatibility.
- Simple configuration language.
- Great ecosystem of plugins to extend functionality.

Why Continuous Monitoring on DevOps

- In the Manage phase of the DevOps life cycle, continuous monitoring of applications in production environments is typically implemented with application performance management (APM) solutions that intelligently monitor, analyze and manage cloud, on-premise and hybrid applications and IT infrastructure.
- These APM solutions enable you to monitor your users' experience and improve the stability of your application infrastructure.
- It helps identify the root cause of issues quickly to proactively prevent outages and keep users satisfied.
- With a DevOps approach, we are also seeing more customers broaden the scope of continuous monitoring into the staging, testing and even development environments.
- This is possible because development and test teams that are following a DevOps approach are striving to use production-like environments for testing as much as possible.

Nagios Core

- Nagios Core is the monitoring and alerting engine that serves as the primary application around which hundreds of Nagios projects are built.
- Nagios Core serves as the basic event scheduler, event processor, and alert manager for elements that are monitored.

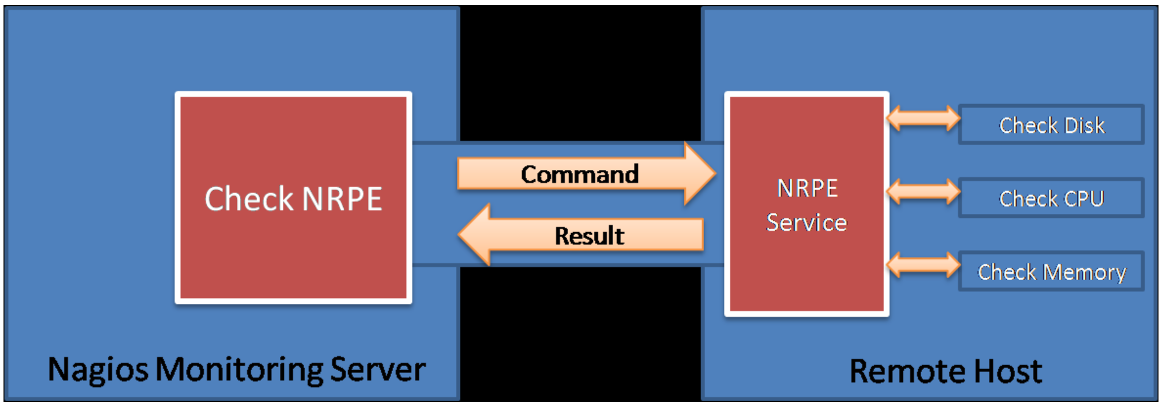
Nagios Plugins

- **Plugins** are compiled executables or scripts (Perl scripts, shell scripts, etc.) that can be run from a command line to **check** the status or a host or service.
- **Nagios** uses the results from **plugins** to determine the current status of hosts and services on your network.

Nagios NRPE

- **Nagios** Remote Plugin Executor (**NRPE**) is a **Nagios** agent that allows remote system monitoring using scripts that are hosted on the remote systems.

- It allows for monitoring of resources such as disk usage, system load or the number of users currently logged in.


Check NRPE
Command
Result
NRPE Service
Check Disk
Check CPU
Check Memory
Nagios Monitoring Server
Remote Host