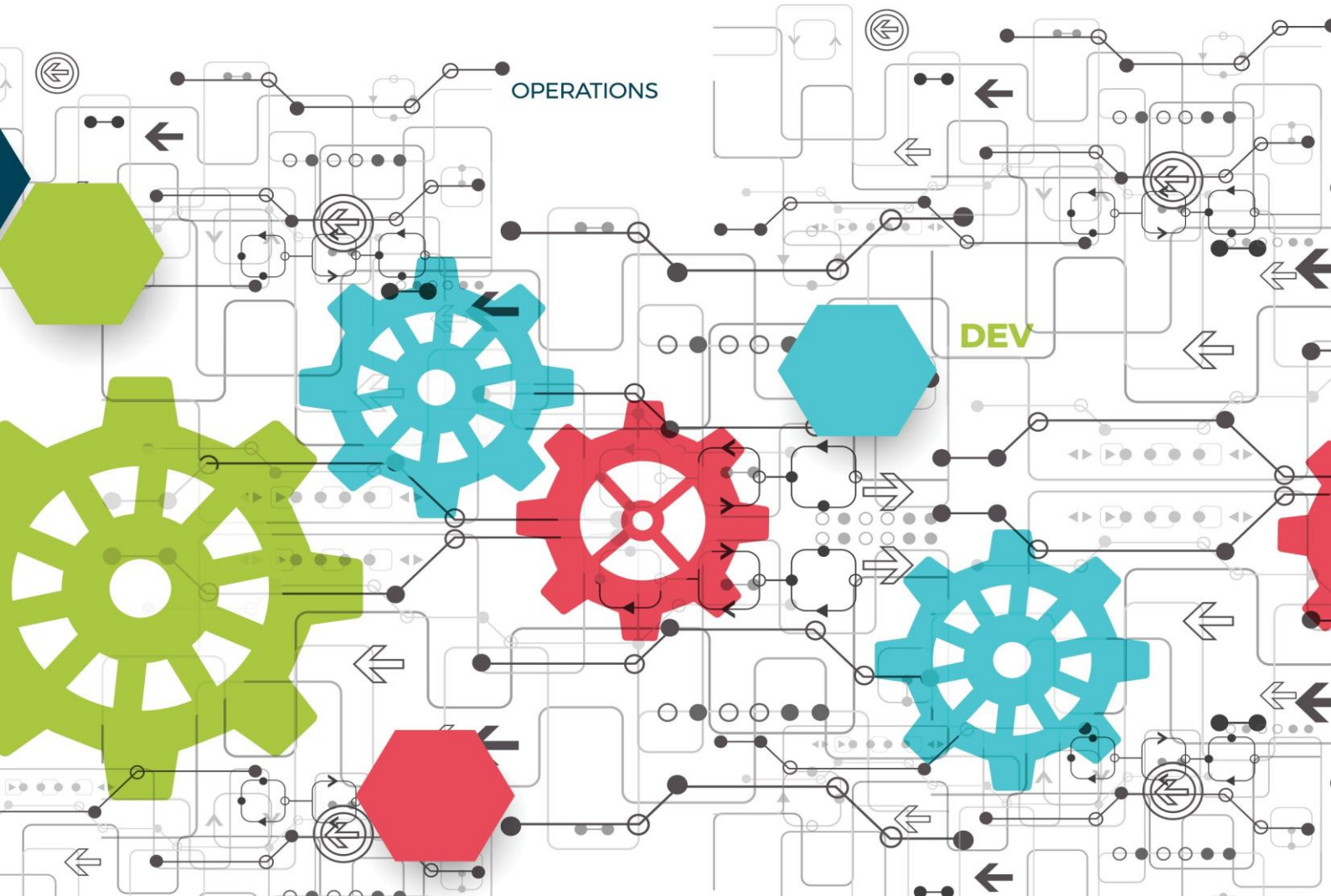
Xebia

B.Tech Computer Science and Engineering in DevOps

DevOps Overview

MODULE 5

Introduction to DevOps Tools & Technologies



Content

MODULE 7

DevOps Tools

Facilitator Notes:

Welcome the participants and give them an overview of the module. Tell them that they will learn about 'DevOps Periodic Table' and brief Introduction to DevOps Tools & Technology

Module Objectives

At the end of this module, you will be able to learn:

- Introduction to Dev Ops Periodic Table
- Implementation of Periodic Table

7.1 DevOps Periodic Table

Every enterprise uses DevOps for a successful software development lifecycle. DevOps requires vast number of tools for implementation in its complete lifecycle. There are different DevOps tools. These tools get used in one or more phases in the SDLC lifecycle. The above slide illustrates the alignment of these DevOps tools in the complete SDLC lifecycle.

Depending on the organization, the implementation of one or more DevOps tools gets done. How to select the DevOps tools depends on organization management. There are no guidelines to select these tools. But one should consider every aspect while selecting DevOps tools.

If more and more tools are added to the product, it may make the infrastructure complex leading to decreased performance.

This diagram represents how DevOps works in developing a product. With DevOps we are basically targeting to reach out to each stage in SDLC cycle. DevOps directly or indirectly impacts all stages of development lifecycle. Using DevOps tools we can automate each of these stage to get more efficient and quality product.

Created by DevOps practitioners for DevOps practitioners, over 18,000 votes were cast across more than 400 products in 17 categories to produce the 2020 Periodic Table of DevOps Tools.

The Periodic Table of DevOps Tools is a dynamic, embed-able and aesthetically pleasing device that allows users to visualize the most popular DevOps tools, define them, and categorize each tool by functionality and pricing model.

The Periodic Table is depicted as on Next Page

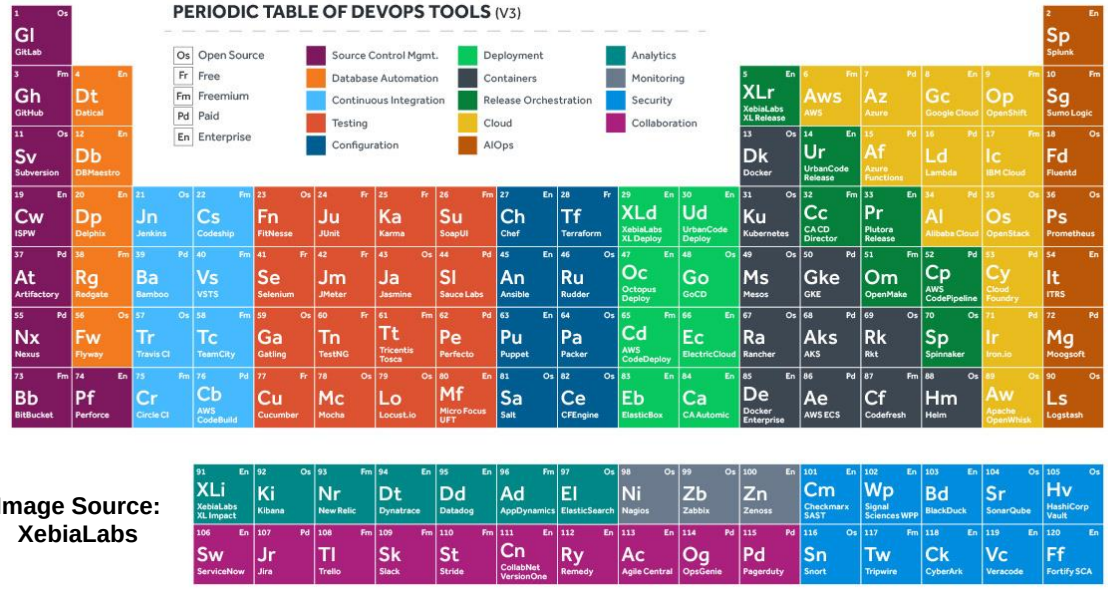
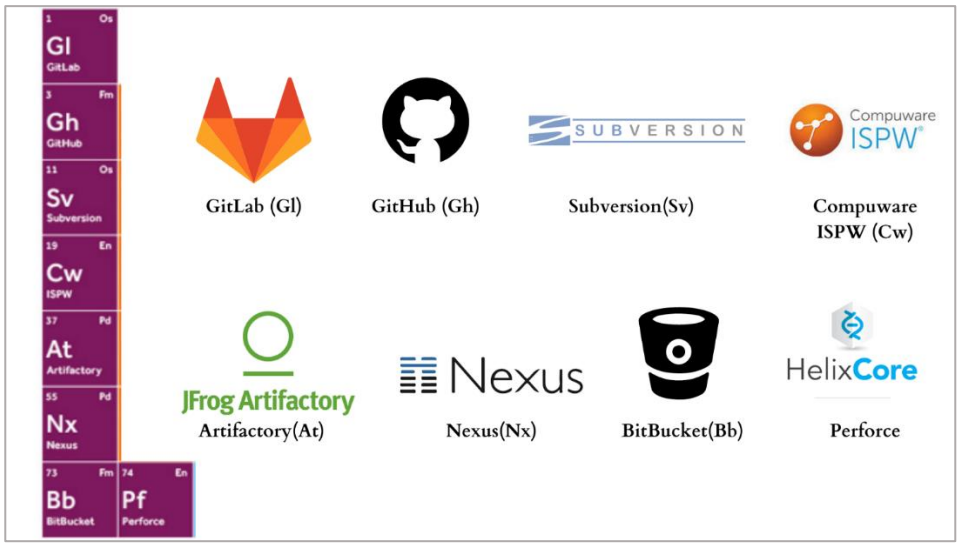


Image Source: XebiaLabs

7.1.1 Periodic Table: Source Code Management

One of the initial steps when starting development is to build the code. The source code management tools provide versions to indicate which user has made the changes at what time.

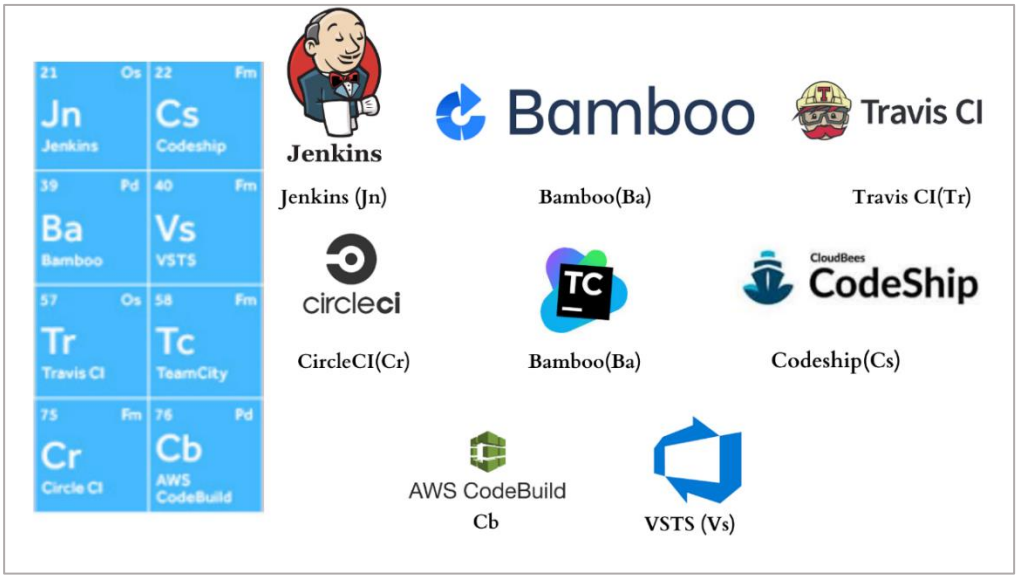


7.1.2 Periodic Table: Database Automation

Database automation is the usage of self-updating and unattended processes for various administrative tasks in the database. Hence automation, requires reducing errors in deployments, improve the speed, and increase reliability. Given below are the tools used for this purpose.

7.1.3 Periodic Table: Continuous Integration

We all know continuous integration is the heart of the DevOps Lifecycle, as all the members of a team integrate their work quite frequently.

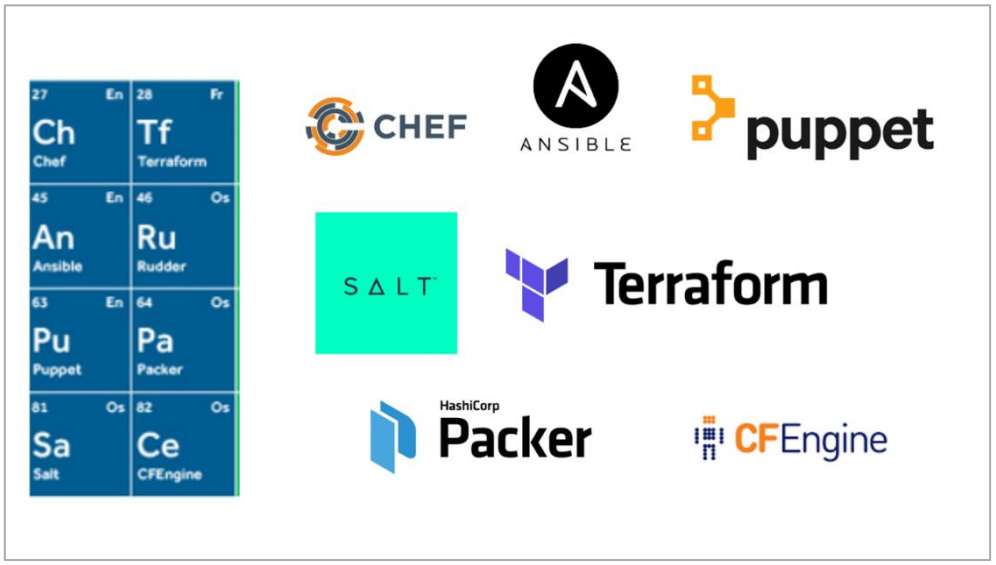


7.1.4 Periodic Table: Testing

Testing stage checks your application/ software for bugs and resolve the same. For the various types of testing there are many tools. Listed above are the few.

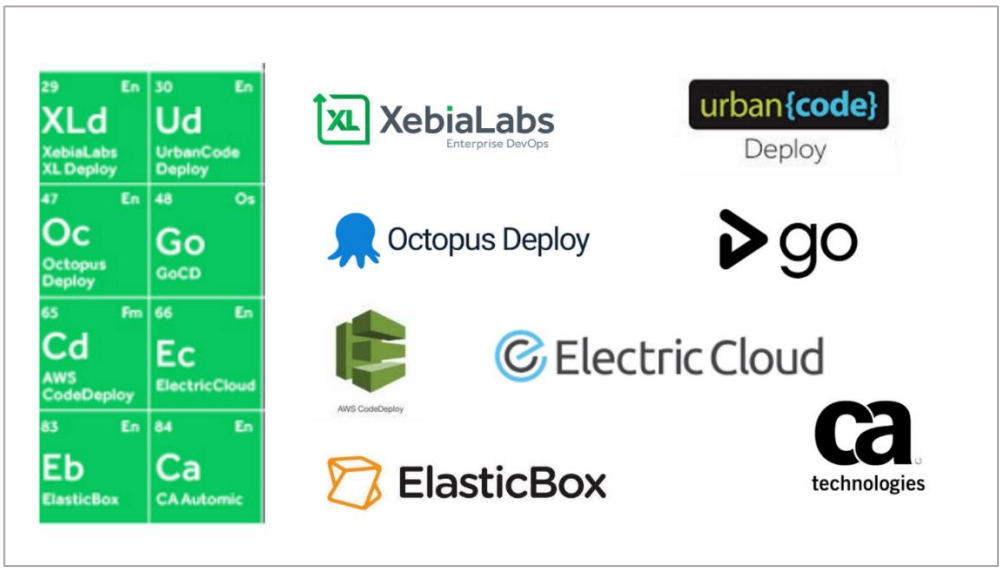
7.1.5 Periodic Table: Configuration Management

Configuration Management is a process through which you can handle the changes in a systematic manner. Top tools to manage configurations are listed above.



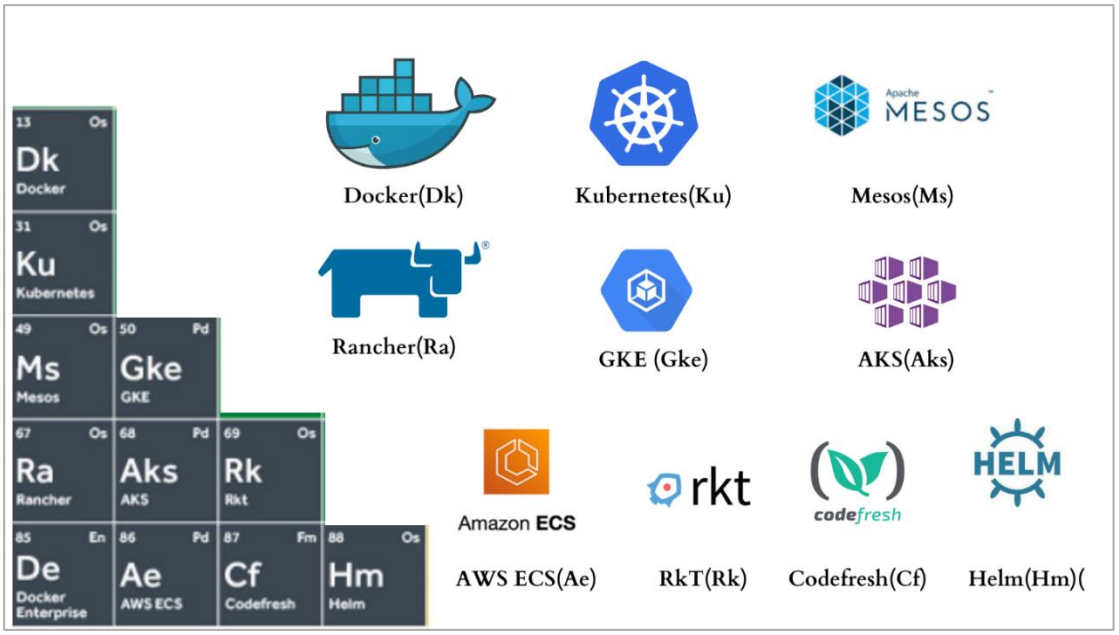
7.1.6 Periodic Table: Deployment

The testing and verified application is deployed into the production environment using various tools based on the enterprise or the application structure.



7.1.7 Periodic Table: Containers

Containerization has enabled the users to build the application with the help of microservices wherein all the required packages and libraries for service are packaged into a single container. Listed above are many tools used for the same.



7.1.8 Periodic Table: Orchestration

The above tools help in release orchestration

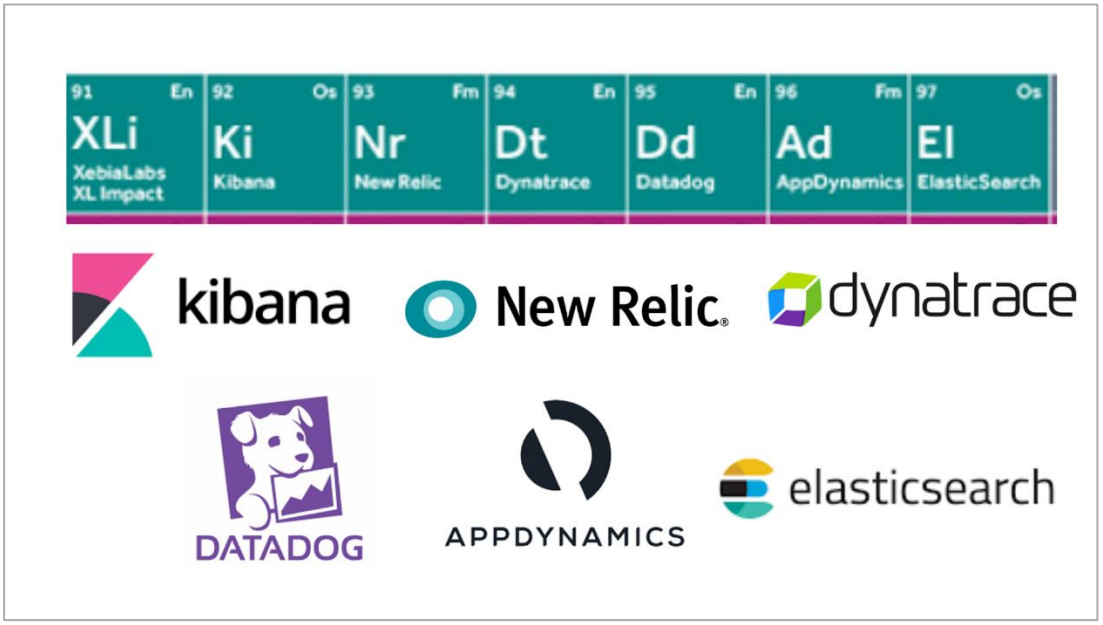
7.1.9 Periodic Table: Cloud

Cloud is the means of storing or accessing your data over the internet rather than your own hard drive. Listed above are popular cloud providers

7.1.10 Periodic Table: Artificial Intelligence Operations

Few tools used for Artificial Intelligence Operations are listed above.

7.1.11 Periodic Table: Analytics



There are many tools used to analyze the data, but few tools are very popular in the Devops are listed above.

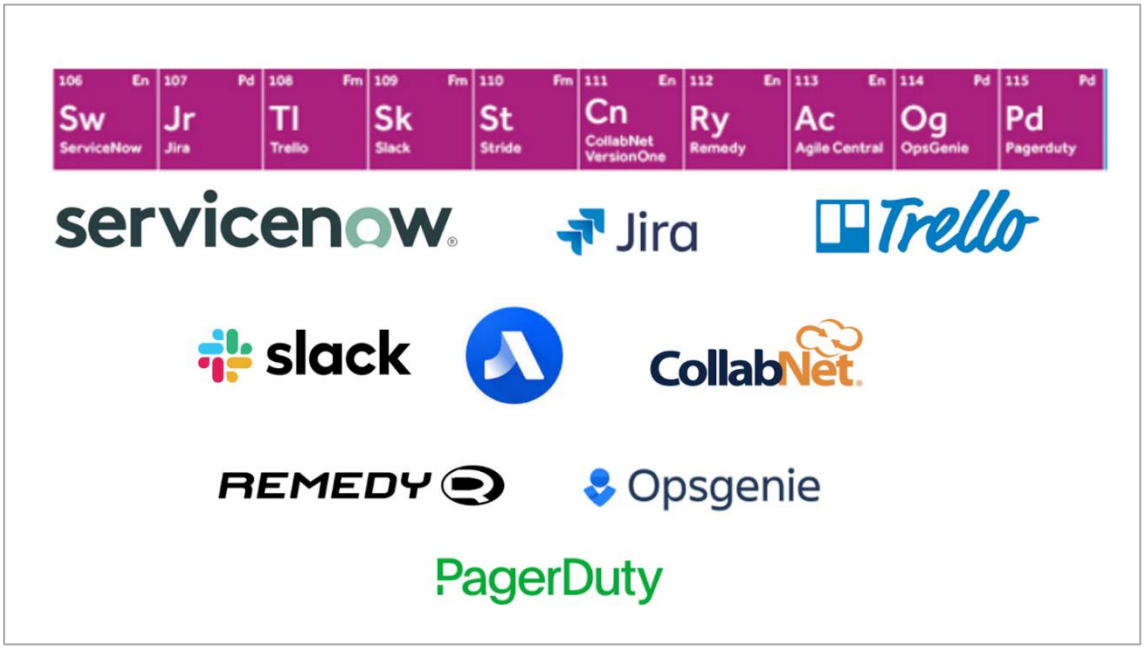
7.1.12 Periodic Table: Monitoring

The above-mentioned tools can be used for continuous monitoring.

7.1.13 Periodic Table: Security

The top tools you can use to secure your application are provided above.

7.1.14 Periodic Table: Collaboration



The top tools you can used for collaboration are given above.

7.2 GIT/GITHUB- Version Control System

7.2.1 Introduction to Version Control System

Source Code Management (SCM) is used to keep track of changes to source code repositories. SMC keeps the history of execution of changes in the code base and helps to resolve conflicts when merging updates from multiple contributors. SMC is also synonymous with version control.

VCS(Version Control System) enables any user to keep track of the modifications done in software development projects, and allow them to collaborate with each other on those projects. Developers, using VCS, can work together on code and keep their tasks separated by saving them in different branches.

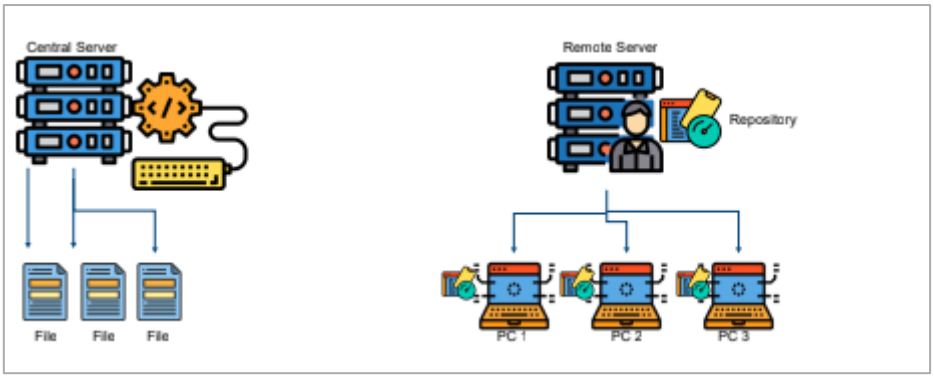
There can be several branches in a version control system, according to the number of collaborators. The branches maintain individuality and keep a specific part of code isolated from the changes made as a particular code remain in specified branch/branches.

Developers can work together to combine the code changes when required. Further, they can view the history of changes, go back to the previous version(s) and use/manage code in the desired fashion.

Key Features of VCS

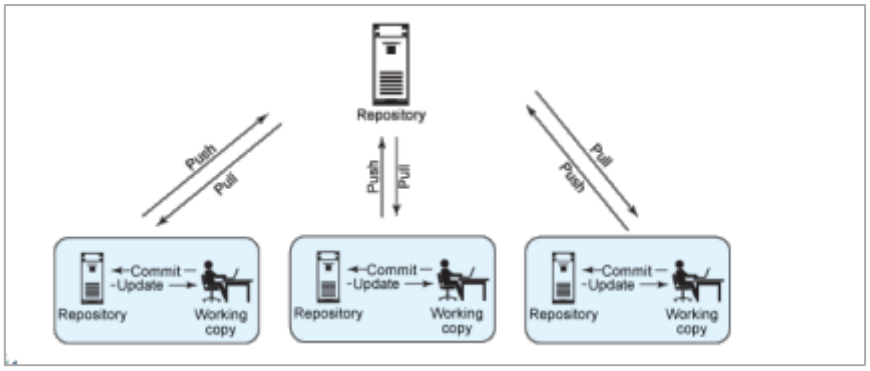
- Manages changes to software source files and other documents
- Acts as an interface using which different users can modify source code according to the requirement
- Without VCS, it's very difficult for the developers to collaborate with each other
- Revert changes at any point of time
- Each modification done to a specific file in VCS creates a new version of that file
- Two types of VCS: Distributed VCS, and Centralized VCS

Centralized Vs Distributed VCS



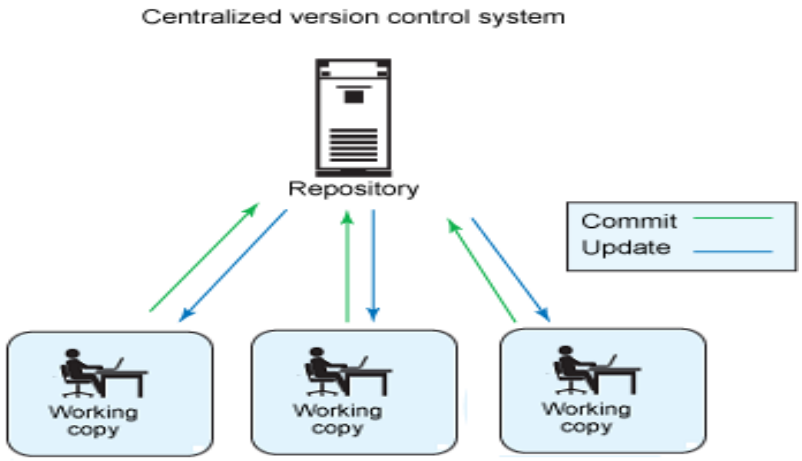
Distributed VCS

Considering a normal workflow, the Distributed Version Control System will create a new setup cloning the local working copy from remote repository into the developer's machine. Every developer machine will have that working copy in which they can perform modifications. Due to distributed nature of VCS, one can perform commits, merge and rebase between branches without any remote repository connectivity. Once the changes are done, changes can be pushed from local repository to remote repository. Similarly, changes can be pulled from remote repository to local repository.



Centralized VCS

Centralized VCS comprises two components: Server component and Client component. Unlike distributed VCS, only working copy is created locally which requires network connectivity to perform commits and merging. Hence, centralized VCS will require a dedicated network connectivity while performing commits and other operations.



7.2.2 GIT

Why GIT?

- Used to track modifications done to source code during software development.
- Also, used to perform various changes even in offline mode.
- Efficient and faster

- Supports branching.
- one of the most secure version control system which supports cryptographic method SHA-1.

Introduction to Git

Git is a **Distributed VCS** for tracking changes in the source code during development of a software. It has been specifically designed to coordinate the work among programmers and developers; however, it can also be used to track changes in any set of files. Benefits of Git include data integrity, speed, and support for non-linear and distributed workflows. Multiple teams of developers can use tools such as Pull requests for collaboration on Git branches and later reviewing code of each other.

Today, Git is the most prominently used version control system in the world and has been considered as a modern standard for software development with its distinguishable elements being:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

Since its creation in 2005, Git has been the most popular tool for source code management and version control. The truly distributed nature of Git has attracted many developers, who themselves use this product for their individual source code management. Git works well with most of the operating systems and IDEs.

Important features of Git

The three important features of Git are performance, security and flexibility.

1. Performance

Git exceeds many of its counterparts in performance. Git has been optimized for performance in terms of committing changes, branching, merging and comparing past versions. Git doesn't look at the name of the files while determining the storage and the version history of the file. Instead, Git looks at the content of the files. Git uses a combination of delta encoding (storing content differences), compression and explicitly stores directory contents and version metadata objects. Many of the version control systems store information as a list of file-based changes (delta-based version control). On the other hand, Git handles data more like a series of snapshots of a file system. During each commit, Git

takes a snapshot of the files in that project and stores a reference to that snapshot. Most importantly, if files are not changed, that file is not saved again.

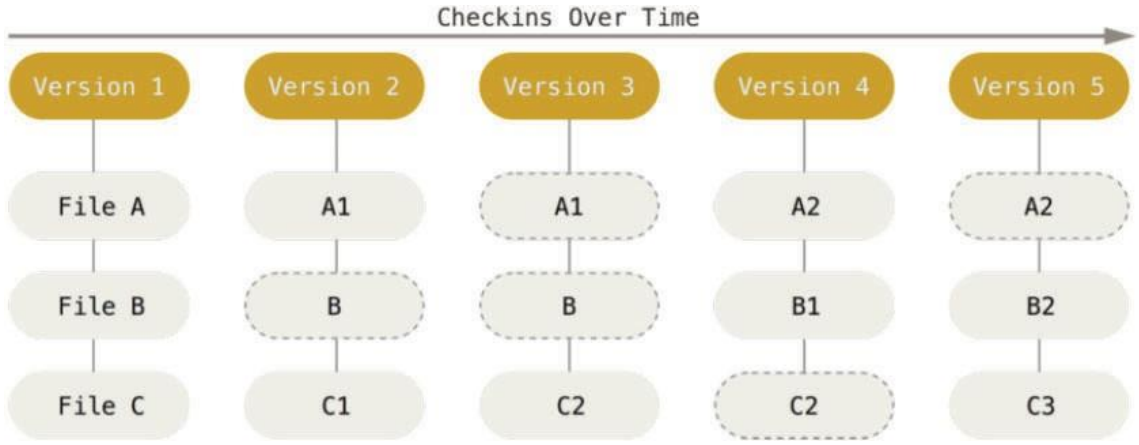


Image source: Pro Git

2. Security

One of the major goals behind the creation of Git is integrity. Git uses a hashing algorithm called SHA1 to secure the file contents, directories, versions, tags and commits. Everything in Git is check-summed before it is saved, and is then referred to by the checksum. SHA-1 hash is a 40-character string composed of hexadecimal characters (0–9 and a–f) and is calculated based on the contents of a file or directory structure in Git. A SHA-1 hash looks something like this: *24b9da6552252987aa493b52f8696cd6d3b00373*. This makes the history completely traceable and protects the code against accidental and malicious damage. Most other version control systems lack this feature and can lead to serious information security vulnerability.

3. Flexibility

Git is flexible in a number of aspects. It supports multiple nonlinear workflows. Git is efficient in handling both small- and large-scale projects.

Basics of Git

Three Stages of a File

Within Git, our files travel through three different stages as follows:

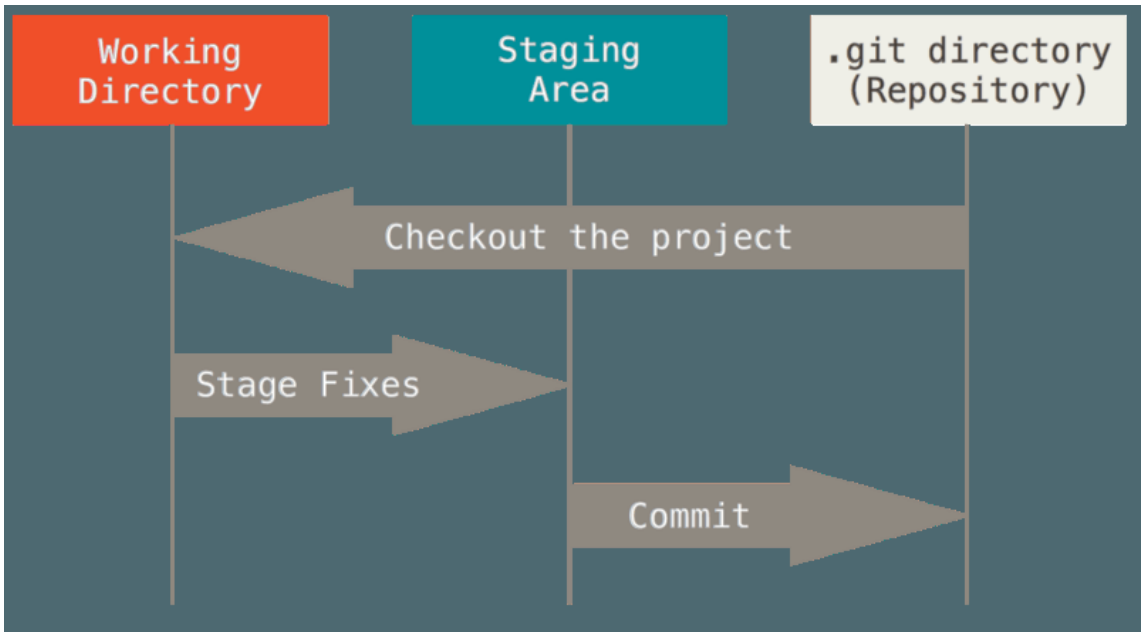
- **Committed-** data is saved in the local database.
- **Modified –** changes are made to the file, but the file is not committed to the database yet.
- **Staged –** a modified file has been marked in its current version to go into the next commit snapshot.

With this, it is important to know three important concepts of a Git project.

- **The Git directory** – the location where Git stores the metadata and object database for our

project. It is this Git directory that is copied when a repository is cloned from another computer.

- **The working tree** - a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk, which we can use or modify.
- **The staging area** – present in the Git directory. The staging area stores information about what will go into the next commit. It is technically named as “index”, but commonly referred to as “staging area”.



Basic Git Workflow

- File in the working directory is first modified.
- Selected changes that are to be there in the commit (only) are moved to the staging area.
- Staged changes are then committed, and the commit process takes the files in the staging area and stores the snapshot permanently to the Git directory. If something is there in the Git directory, it is considered to be committed.
- A file that has been modified and added to the staging area is considered staged.
- A file that has been changed after being checked out from the Git repository, but not staged is considered modified.

About GitHub

It is a web-based hosting service for version control using Git. It offers plans for public and private repositories. You can add multiple projects by creating multiple public repositories. In this section, you will only demonstrate on public repository and its usage. Navigate to https://github.com/ and click on Sign up for GitHub.

7.3 **Docker -Containerization**

7.3.1 Docker (Containerization)

Docker an open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization on Linux. Docker helps developers to package both application and dependencies in single entity called container.

Docker has multiple benefits and some of them are listed below:

- Built on one machine and run everywhere: With docker we can built up docker images locally and then same docker image can be deployed across multiple machines.
- Easy configuration: Docker supports easy configuration which helps developers to reduce manual configurations steps.
- Faster delivery cycles: Docker containers make its easy to deploy new applications and releases latest version to business.
- Application portability: Docker containers are portable which means we can deploy and run them on any docker host whether its windows or Linux.
- Easy Deployments: Docker containers are easy to deploy across multiple environments and these can be easily deployed using Jenkins pipeline scripts.
- Application testing with new versions: We can use latest version of dependencies to test application source code. We can simply use latest docker image and test application without performing much of installation process.

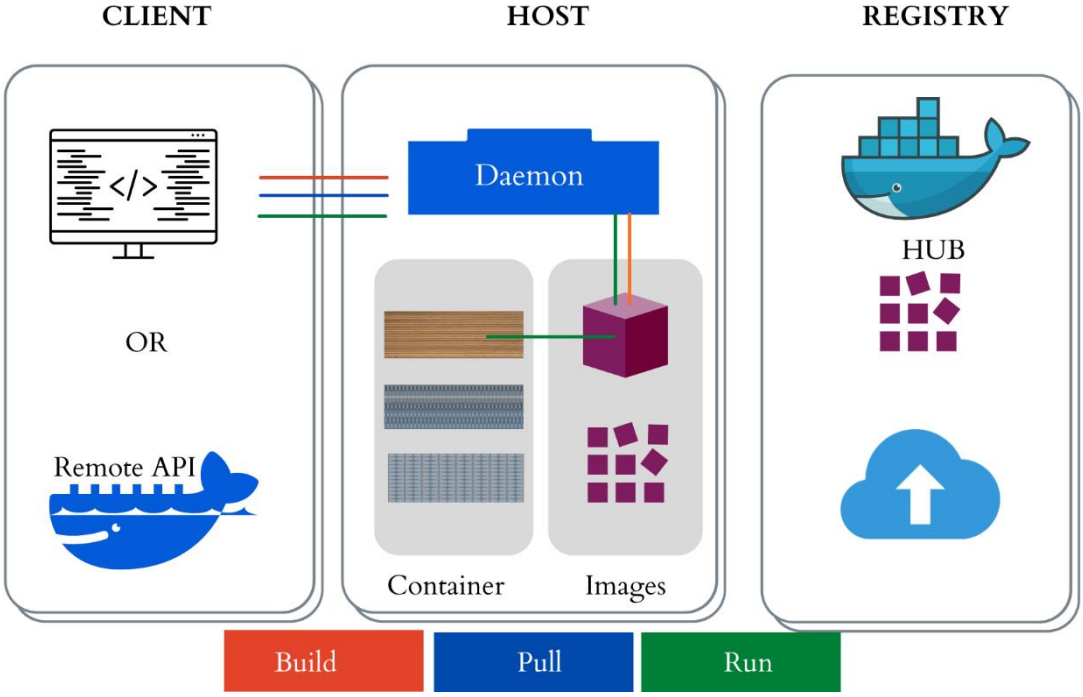
Docker was initially launched for Unix based environment only but later on it got released for other platforms also. Docker installation is available for Ubuntu, centos, Red hat and other Unix based platforms. Installation process ais shown in upcoming slides. Docker is available almost for every platform including Unix and Desktop technologies. Docker can be installed on Linux , Windows and MacOS platform.

When we install Docker on host we have below components installed:

- The Docker daemon: The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.
- The Docker client: The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API.
- Docker registries: A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.

- Docker Images: An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.
- Docker Containers: A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI.

Container can be initiated both in interactive and detached mode. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging. Containers are created once run on the Docker host.



Container is an instance of docker image which gets created once docker run command is executed. A container is a standard unit of software that packages up code and all its dependencies. Same docker image can be used multiple times on different machines. This helps to initialize applications deployed inside these containers. Developers can program these docker images according to there requirement and deploy them whenever needed to build docker image for different environments for deployments.

The main difference between a virtual machine (VM) and a container is that the VM is a hardware-level virtualization, and a container is a OS-level virtualization. VM hypervisor emulates a hardware environment for each VM, where the container runtime emulates an operating system for each container. VMs share the host's physical hardware and containers share both the hardware and the host's OS kernel. Because containers share more resources from the host, their usages of storage, memory, and CPU cycles are all much more efficient than a VM. However, the downside of more sharing is the weaker trust boundary between the containers and the host.

Docker Networking: When containers are deployed, they work totally isolated from each other. Docker network is a mechanism which helps containers to interact with other containers on host.

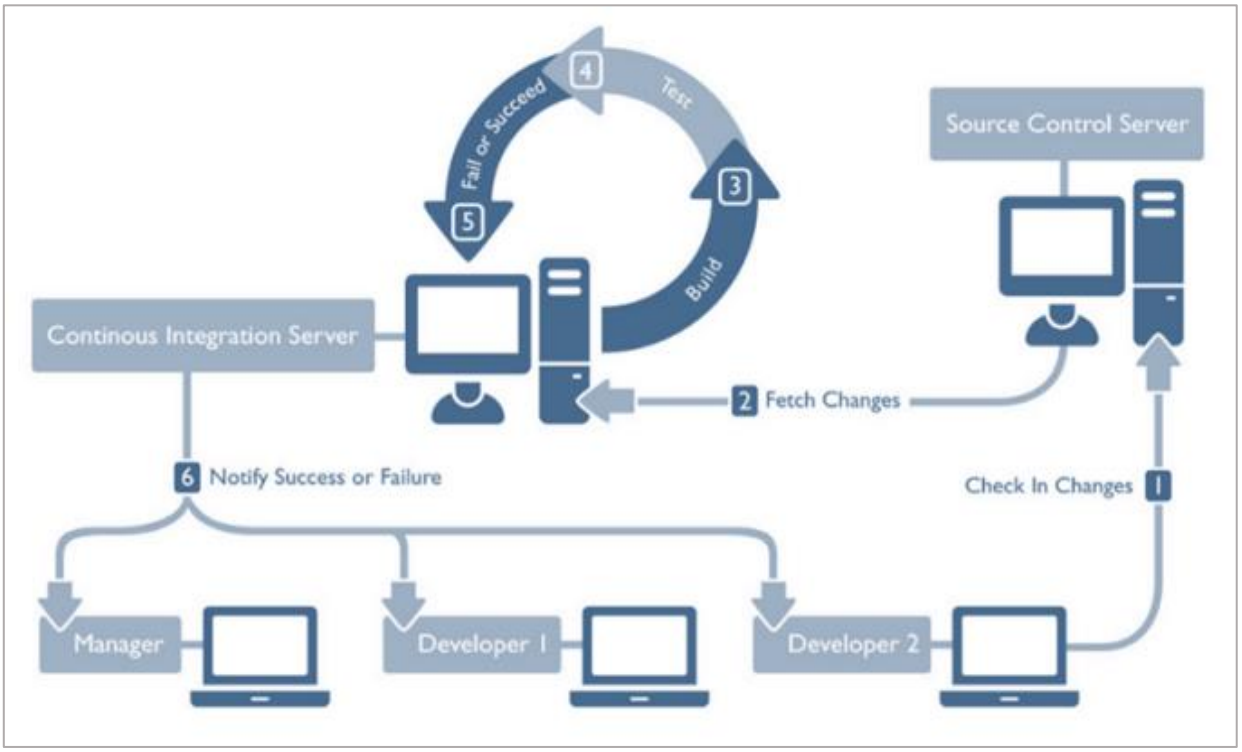
Docker Swarm is an open-source tool that provides features like container orchestration, clustering, and scheduling of containers. Docker swarm allows to deploy containers to multiple nodes machine without

connecting to any of them. It uses Master Slave architecture in which there should be one Swarm manager and all other nodes are considered as worker nodes. Swarm manager sends the request to worker nodes, worker nodes do all the work in scheduling containers.

7.4 **Jenkins (CI/CD)**

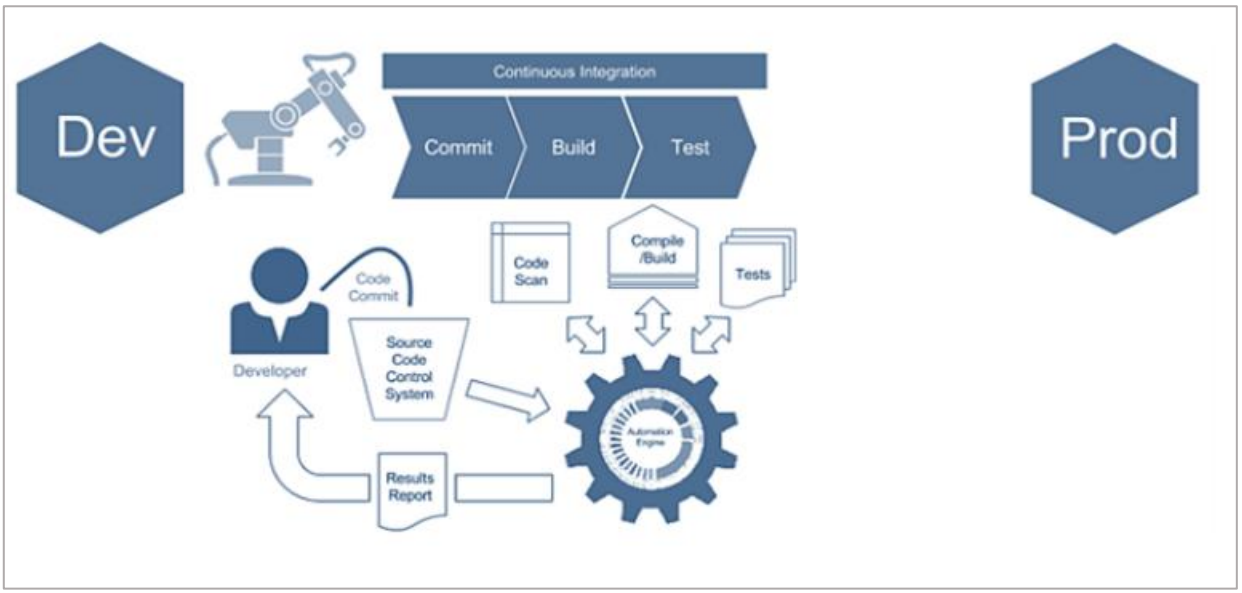
7.4.1 Continuous Integration & Continuous Deployment

Continuous Integration is one of important practices in software development where developers integrate code into shared remote repository. Developers usually perform multiple integrations several time a day. It helps developers to detect errors quickly in initial stage itself. Traditionally, integration is performed in preparation of the deployment of a new release. Continuous Integration (CI) is one such solution: it creates a workflow where every commit made to the codebase is tested and built, and each developer on the project is expected to commit code at least once per day. If the build is successful, it triggers a run of the test suite. This way, conflicts or bugs are discovered quickly over the course of development, making the release process smoother and less painful.



In above image we are representing how CI is implemented and integrated with other components: First developer check-ins the source code in VCS and from there CI will fetch the changes and trigger build in Jenkins. Once build is completed, fail or success status will be sent back to developer so that he will be aware if the quality of source code is good or not.

In software development, **continuous deployment** is very important for releasing source code automatically in live environment. It helps developers to achieve automated deployment process. You can develop faster as there's no need to pause development for releases. Deployments pipelines are triggered automatically for every change.



The above diagram represents the basic CD workflow. In this flow, both CI and CD is implemented with help of build tools like Jenkins, Team city or bamboo. Developer checks in code in version control system from where automated build will be triggered. Automated build compilation of source code will be done. Once final binaries are generated, deployment of source code can be proceeded on whatever infrastructure we want to deploy.

Testing is one of the important phases in complete SDLC cycle. In Agile developers and QA team does not have much of time to trigger test cases manually multiple times. Instead, these test cases are scheduled in pipelines where they will trigger these test cases using automated CI CD pipelines. Complex testing like regression testing can be scheduled every day using Jenkins with automated pipelines. Developers get immediate feedback after performing commit to version control system.

7.4.2 Jenkins

Jenkins is one of CI tools using which developers can design build and deployment automation. Jenkins provides support for all popular source code management (SCM) systems, including Git, Subversion, Mercurial, and CVS, popular build tools like Maven, Ant, Gulp, and Grunt, as well as testing frameworks and report generators. It began as a product called Hudson, developed at Sun Microsystems in 2004-2005, before it was forked from Hudson and renamed Jenkins in 2011, as the result of a dispute between the Hudson community and Oracle. Kohsuke Kawaguchi, the creator of Hudson/Jenkins became the Chief

Technical Officer for Cloud bees in 2014 and Cloud bees now commercially offers Jenkins as a cloud solution.

Jenkins verifies the integration process with an automated build and automated test execution to detect issues with the current source of an application and provide quick feedback. Jenkins plugins provide strong support for technologies like Docker and ECS, which enable the creation and deployment of cloud-based micro service environments, both for testing as well as production deployments.

Jenkins uses master/slave architecture and below two components combined implements Jenkins CI tool.

- **Jenkins Server:** Jenkins server is a web dashboard which is nothing but powered from a war file, default run on 8080 ports. Using Dashboard, Jobs/Projects can be configured, but the build takes place in Nodes/Slave. By default, one Nodes/Slave is configured and running in Jenkins Server. More Nodes/Slave can be added as well.
- **Jenkins Node/Slave/Build Server :** The job of the slaves is to do as they are configured in the Jenkins Server, which involves executing build jobs dispatched by the master. A project can be configured 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 provides one of the important features which can split complex Jenkins job into pieces. It allows to create upstream and downstream jobs on an original job so that some step can be executed before and after original job.

Promotions is a mechanism in Jenkins which helps users to deploy a specific build number which fulfills all checks. Once any build is promoted, a star appears in front of that build to represent promoted build.

Multibranch pipeline is an extension to pipeline job in Jenkins. This helps developers to configure multiple branch configurations for a single project without any manual efforts. Multibranch pipeline automatically scans for new branches and changes once we configure automatic scan in Jenkins configuration.

Shared pipeline is a mechanism in which some common pipeline code or steps configured in shared groovy files. These shared libraries can be called in multiple pipeline projects. It implements reusability of pipeline code there is no need to repeat steps.

7.5 TERRAFORM - System Provisioning

7.5.1 System Provisioning

Provisioning is in essence about allocating the application with a license - the authorization to execute and more so can extend its execution for a defined period. Provisioning (Software) encloses a number of activities, and they are mostly related to an application’s/infrastructure’s license. You can relate to the software(s) you might have used or are still using it with respect to have a certain time limit on a certain license. Provisioning as a practice should be made exclusive and tools for Configuration Management

should not be used for provisioning, otherwise it just makes the situation more complex and more painstaking. Provisioning allows a whole lot of interaction with various teams and businesses and shows the business that this is what your business is powered by and this is how it runs and this is how it will run for certain duration of time with certain effectiveness.

7.5.2 Terraform

Terraform is a product of HashiCorp. It allows infrastructure for variety of providers — Amazon Web Services, Microsoft Azure, Google Cloud, Digital Ocean, and OpenStack. Terraform can manage anything with an API. It uses a simple declarative programming language called Hachiko Configuration Language also known as HCL and to deploy and manage the infrastructures it uses few CLI Commands. It is a tool used for building, changing and versioning infrastructure. Terraform is an Infra provisioning tool which is cloud-agnostic. It gets created by Hashicorp and written in Go. It supports all public and private cloud infrastructure provisioning.

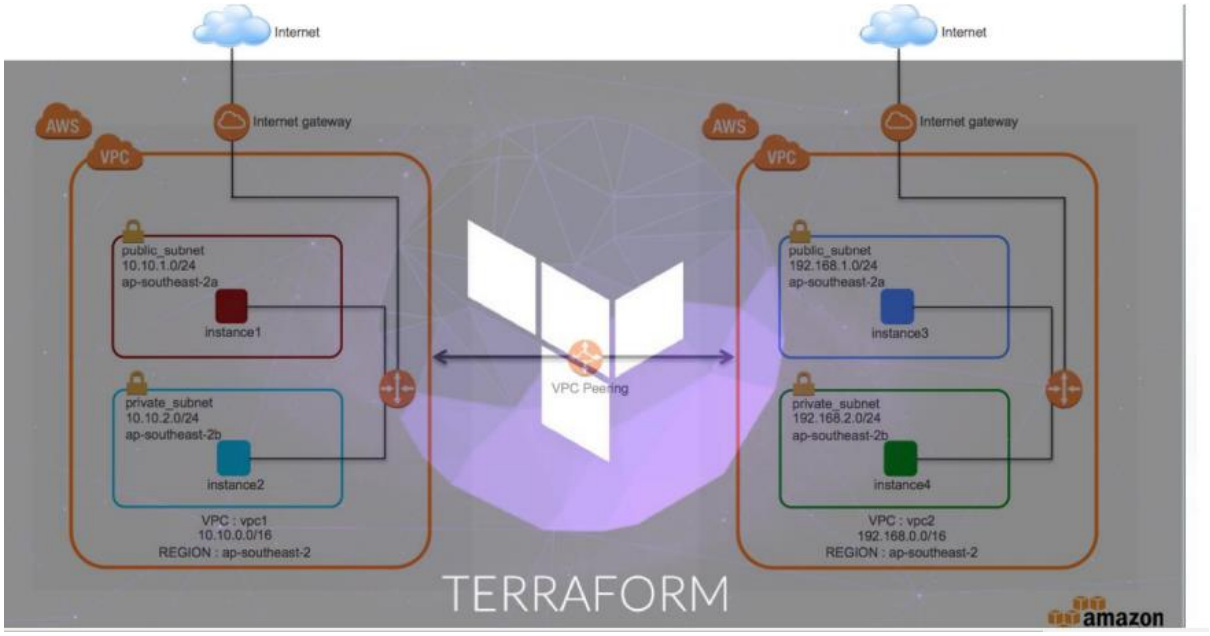
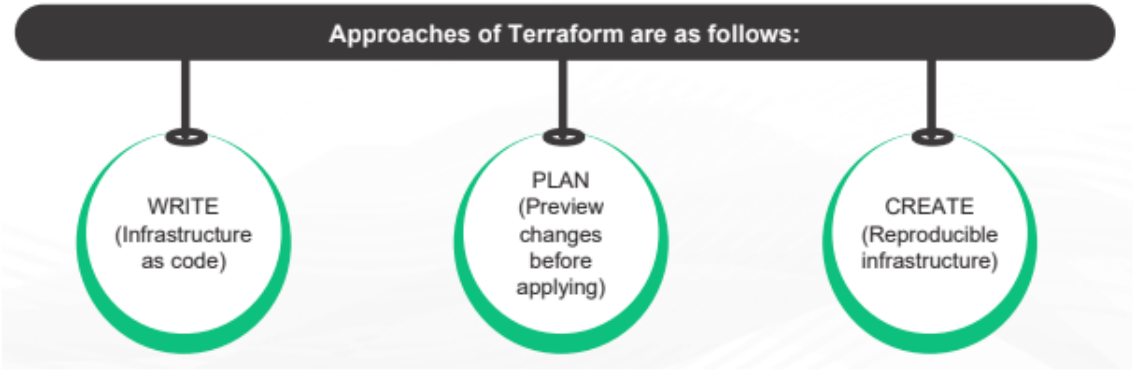


Image Source: http://vcloudynet.blogspot.com/2017/03/how-to-setup-vpc-peering-with-terraform.html

Let's say we are trying to setup a Cloud infrastucture on AWS. We have 2 VPC and 4 Virtual machines in different subnets and we wanted to find necessary security policy between subnets, between virtual machines. We need to do VPC peering. We need to setup the internet gateway and routing so that the VM can go out to the internet and so on. Normally, we will log into Amazon Web Management console, and we will click around the WebUI and create the necessary resources and build the infrastructure. All this is a very manual process. Instead of doing it in a manual way, we can pick one of the IC tools Terraform and it uses HCL, it also supports JSON syntax. Then we write code to define and provision and manage the cloud infrastructure we want. By doing this we automate the whole provisioning and the deployment process very quickly. Terraform is agnostic to underline platform supported by the resource providers. Example: ec2 instances, virtual machines, virtual private cloud VPC, load balancer, security groups, docket containers, etc. The provider is responsible for underlying API interaction with and exposing their

resources and Haschicorp works with a lot of light providers. It includes Version control systems such as GitHub, BitBucket, GitLab. Monetary systems like Digital and Liberado.
For more providers list please head to Terraform Website (https://www.terraform.io/intro/ index.html).



Provisioning resources in more than one cloud and consuming these resources simultaneously, what terraform can do is it can combine multiple providers consistently in one safe single workflow.

7.6 MAVEN-Build & Release Management

7.6.1 Build & Release Management

Build and release management is the process of managing, planning, scheduling, and controlling a software build throughout its life cycle. Build and Release management will control the life cycle of a software product, the process of planning, managing, scheduling, and controlling the build in different stages and environments like development, testing, staging and production stages. Building an application or a software requires build tools like ant, maven, gradle, etc. Build tools compile the source
code files into reusable executable files or packages.
The Release Management as a practice involves:

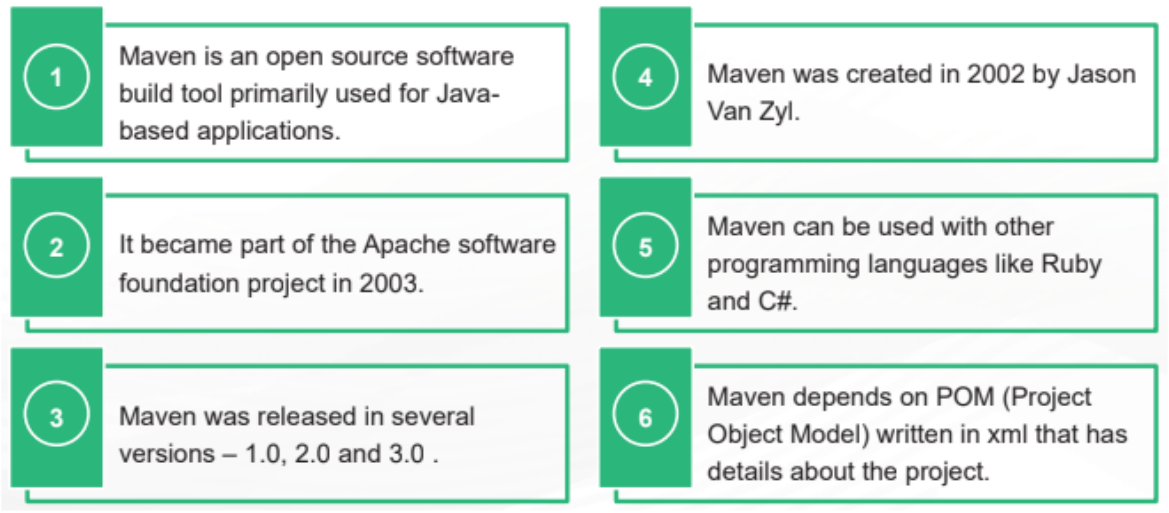
- **Helps make software builds simple, quick, and reliable.** This is achieved by employing the best tools for the job. This means understanding all the various build tools, seeing how they integrate with the systems that already exist in the workplace, and making an informed choice. There's no way you're going to make software builds easier, more reliable and repeatable by implementing a manual solution, so get to grips with the various build tools out there and make them work for you.
- **Helps make software deployments simple, quick, reliable, and repeatable.** Again, this is a bit like the above, but there are fewer tools to choose from. Manually deploying releases is painful and risky, and it also belongs in the dark ages and should be outlawed. There are still plenty of options and combinations of tools to make this task fully automated.
- **Helps take care of configuration management.** When I say configuration management, I'm talking about all those issues with how to make a software release look, feel and behave the same from one environment to the next. For me this falls into Release Management because Release Management, unlike development, QA or Operations, has a direct involvement in every environment along the way to releasing into the wild. It's pointless asking the development team to tackle the issues of configurations between environments when they have very little or no visibility

of the production environment, and besides, their time would be much better spent making that Helps drive software quality. button look cooler because that's what the business has asked for!

- **Helps drive software quality.** Thanks to the Continuous Integration process, and the tools that have been built around it, it's now possible for us to build software every single time a piece of code is checked in, run a suite of unit tests, analyze the code for lazy.

7.6.2 MAVEN

Maven is an application that provides functionalities for project management. It has functionalities for project building, reporting, and documentation. Following are the key details of maven build tool:



Key features of Maven include:

- An easy, and standard way to build projects where unnecessary details are hidden
- A standard strategy is followed while building any project
- Automatic dependency management
- Multiple projects can be simultaneously managed
- Necessary plugins and libraries are automatically downloaded from Maven repositories

Maven depends on POM file to build the project.

Most commonly used build commands are as follows:

Maven build commands that are widely used are discussed below:

- Validate: It validates the entire project and downloads required dependencies.
- Compile: Compiles the source code of the project
- Test: It will just test the source code using test framework and to run this commands deploy or packaging is not required.
- Package: Packages the compiled source code into executable application like jar.
- Install: This command will install the package into the local repository.
- Deploy: It copies the final package into a remote repository.

7.7 Ansible- **Configuration Management**

7.7.1 Configuration Management

Configuration management tool mainly covers both software configuration and server configurations. It helps to enforce standardization in software configurations without any errors. It also helps to reduce the time taken for managing configuration manually on each and every single server.

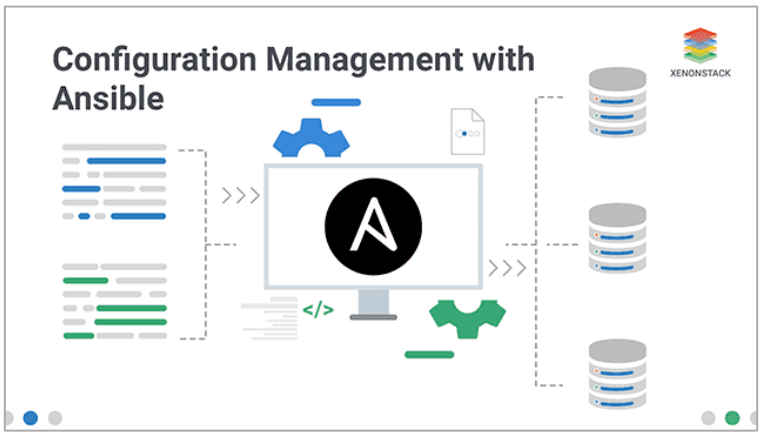
Configuration Management tools provides lot of benefits in organization as below:

- **Quick Provisioning:** Configuration management tool which is used to quickly provision infrastructure with help of configuration management scripts. Admin don't have to perform any manual efforts and they can easily create multiple infrastructure within less duration of time.
- **Version Control Support:** Configuration management scripts can be stored in version control system and can be managed just like any other source code. Admins can work on these scripts by changing these scripts in version control system.
- **Environment Cloning:** Using Configuration management tool we can create a new infrastructure with same configurations similar to an existing infrastructure. This enables admins to create multiple servers having same configurations and binaries installed on them.
- **Less cost and risks:** Working with configuration management tools helps IT team to performs tasks much faster which saves lot of time and cost.
- **Reliability:** Infrastructure provisioned with configuration management tool is quite reliable and we don't face any issue while accessing these infrastructure for long duration of time.
- **Minimum Documentation:** Configuration management tool reduces the requirement for documentation. CM scripts itself are considered as documentation which provides complete infrastructure is designed and deployed.

7.7.2 Ansible

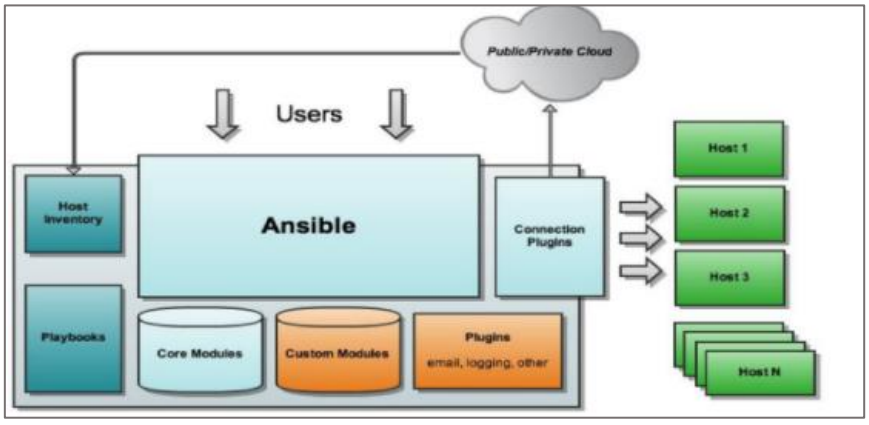
Ansible is an open-source community project that is sponsored by Red Hat. Ansible was first launched in 2012, is acquired by Red Hat in 2015. Ansible can be used as Continuous Deployment tool also to provision deployment for cloud-based applications. It is one of the most powerful and famous tools that is used for the automation. It can automate most of the tasks, it is supported by a very large community. It

enables everyone to share their work or contribute their work to open source through “Ansible Galaxy”.

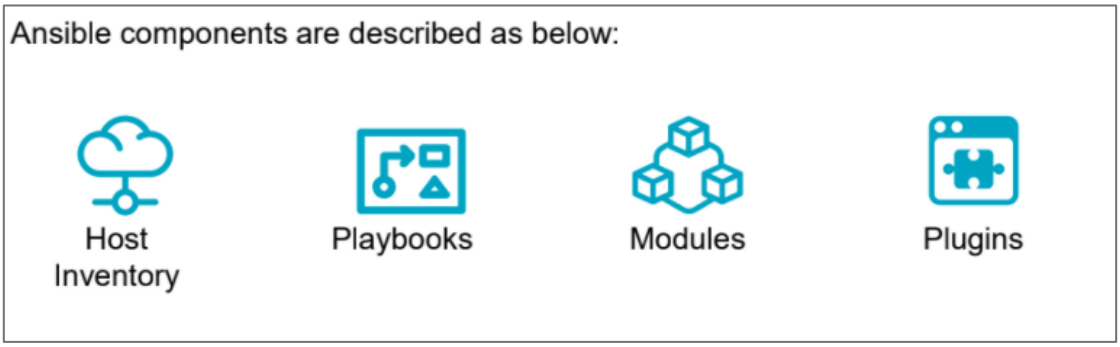


Features Of Ansible:

- Simple to work with, as it is very clear, if anything needs to be widely adopted & supported, then it should be simple.
- Agentless, this tool only needs to be installed on the master node/controller node, & it can connect to any system provided internet connection between those systems is there, & then it can perform the tasks that are desired.
- Uses Universal Declarative language i.e., YAML, therefore, it is very easy to work with Ansible & perform the tasks.
- Promotes collaboration by breaking down the silos & promoting automation.
- Can integrate most of the technologies that are been used in any organization.
- Removes complexity.



Ansible is also called as agentless configuration management too. Ansible comprises of main components like host inventory, playbooks, modules, plugins.



- **Host Inventory:** Host inventory is a very important component of ansible architecture. It is used to store the hosts IP for servers which we want to manage using ansible.
- **Playbooks:** Playbook is used as Infrastructure as Code (IaC) to manage configurations for application and servers. We can install, manage, and upgrade applications using these playbook scripts. We can also store these scripts in Version control system so that we can easily manage versions of these scripts.
- **Modules:** Modules are predefined code available in ansible which can be used to manage resources like files, folders, service, packages etc. we can use these modules in playbooks so that we can manage software configurations.
- **Plugins:** Plugins are pieces of code that augment Ansible core functionality. Ansible uses a plugin architecture to enable a rich, flexible and expandable feature set.

Ansible ships with several handy plugins, and you can easily write your own.
Ansible uses two ways to manage resources on remote server:

- **using ansible commands:** used to manage one or two resources
- **using ansible playbooks:** used to manage multiple resources e.g. installation of complex software's, applications installations, application deployments

Ansible Use-Cases

- Provisioning
- Configuration Management
- Orchestration
- Application Deployment
- Security Automation
- Continuous Delivery

Ansible is also called as agentless configuration management tool. Ansible comprises of main components like host inventory, playbooks, modules, plugins.
What Ansible can Automate?

- Networks
- Infrastructure
- Containers
- Cloud
- Applications
- Security

This tool has helped many companies/organizations/start-ups to perform better & fulfill their needs/goals. Ansible is the most used and supported tools among its competitors.

7.8 **Selenium- Software Testing**

7.8.1 Software Testing

Software testing is one of important phase in software development process. Below are some of reasons why testing is important:

- Testing identifies bugs in early stage which costs less for development team to fix issues.
- It detects vulnerabilities in source code while validation.
- Testing helps to make code effective which satisfies customers requirement.
- Software testing is important to point out defects and bugs from application source code.

Automated software testing is a mechanism of performing software testing automatically with help of test scripts. We don't have to perform any testing manually, instead test scripts will be able to test application.

7.8.2 Selenium

Below are some of benefits of automated testing:

- **Reduce cost and maintenance:** Automated testing helps to automate test cases, which earlier manual tester use to execute manually. This saves a lot of time for testers, so that they can work on other items Increase
- **Ad hoc and Exploratory testing:** Automated testing helps tester to cover ad hoc requirements with helps of automated testing, since they don't have to put lot of manual efforts to complete that testing. Testers can run these automated test cases by modifying small piece of code without thinking about manual efforts.
- **Improve test coverage:** In modern development we are making sure that we should go for 100 percent code coverage. This coverage can only be calculated by adding more and more automated test cases. Without automated test cases we cannot calculate coverage.

Selenium is one of the best tools which is used to write test cases for web- based applications. Selenium helps testers to write test cases for various programming languages like Java, PHP, C#, Python, Groovy. Selenium is used to test web-based application which is written in Java programming languages. We can use eclipse IDE to write selenium test cases and then execute on application source code. These test cases can be executed on modern web browsers like Chrome, Firefox and Internet explorer. It can be used as open-source software for developing test cases. It also provides feature like screen recording and playback to write test cases. We can run selenium test cases on various platforms like Unix, MacOS, Windows.

The features of Selenium are listed Below:

- Selenium is an open-source software, so it is free like other Automation Tools such as QTP or Test Complete, etc. Even the updated versions are free. Users can configure the same once the release is stable and do not have to pay for anything.
- Selenium supports various language bindings like Java, C#, Perl, Python, PHP, Ruby, JavaScript.

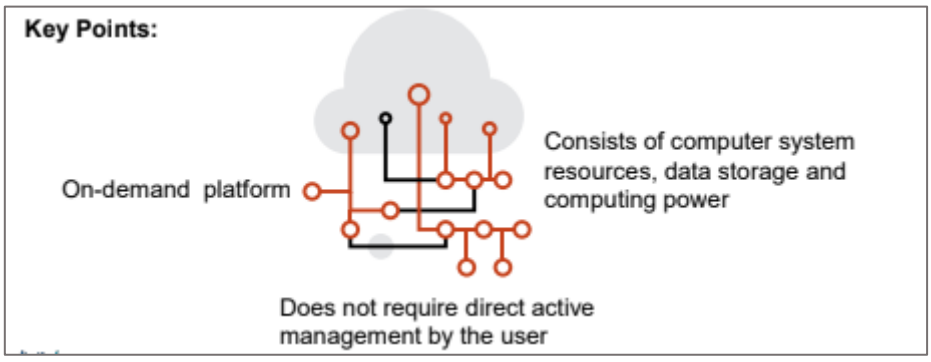
Unlike QTP which supports only Visual Basic, Selenium has lots of support for all these languages. Though it is the most stable with JAVA and 90% of the Selenium Testers use JAVA

- The best part of Selenium is we can create a test suite in one environment and run the same in another. Let's say for example, you create the Scripts in Windows, and you want to run the same in Mac OS then it can be executed with certain changes.
- Selenium supports Browsers like Firefox, Chrome, IE, Edge, Safari, Opera, etc. but it is most stable with Firefox and Chrome.
- Selenium with the powerful feature of Parallel Execution can run multiple test cases in multiple browsers at the same time and help in time saving.
- Initially, setting up Selenium and writing up the Scripts might take time, lot of logic and understanding but that remains the same in case of Manual Testing as well.
- The distinct advantage of Selenium is that once everything is setup, subsequent executions just happen way faster, and results and reports are generated which are quite good looking and also way better in terms of presentation and that is quite important in a professional setup with high profile clients.
- Selenium scripts can easily be pushed and pulled with certain credential validations into various Object Repositories and CI/CD tools too. That helps quite a lot to make everything work in a remote environment. This helps to access the code from anywhere and does not really depend on the physical device.

7.9 **AWS Cloud Computing**

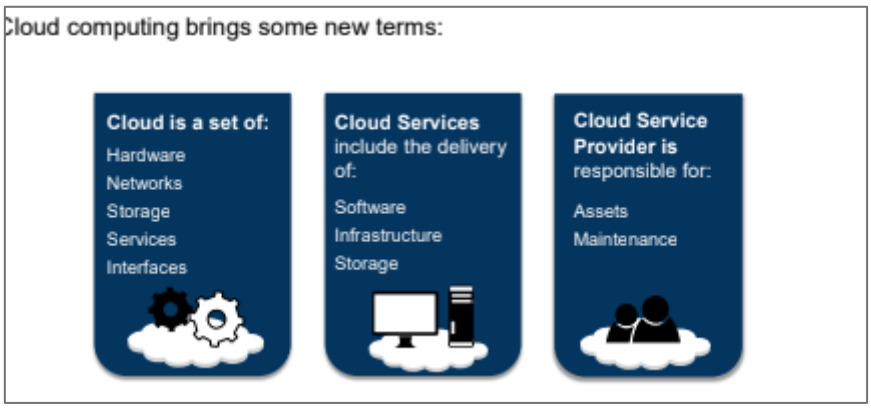
7.9.1 Cloud Computing

Cloud computing is on-demand platform of computer system resources, data storage and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Cloud computing is a method for delivering information technology (IT) services in which resources are retrieved from the Internet through web-based tools and applications, as opposed to a direct connection to a server. We can easily manage these resources over internet and these are cheap resources which we can use to create resources.



Cloud computing is on-demand platform of computer system resources, data storage and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Cloud computing is a method for delivering information

technology (IT) services in which resources are retrieved from the Internet through web-based tools and applications, as opposed to a direct connection to a server. We can easily manage these resources over internet and these are cheap resources which we can use to create resources.



The five essential characteristics of cloud computing are:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Let's understand these characteristics in detail.

- **On-demand self-service.**

  A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- **Broad network access.**

  Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms. For example, mobile phones, tablets, laptops, and workstations.
- **Resource pooling.**

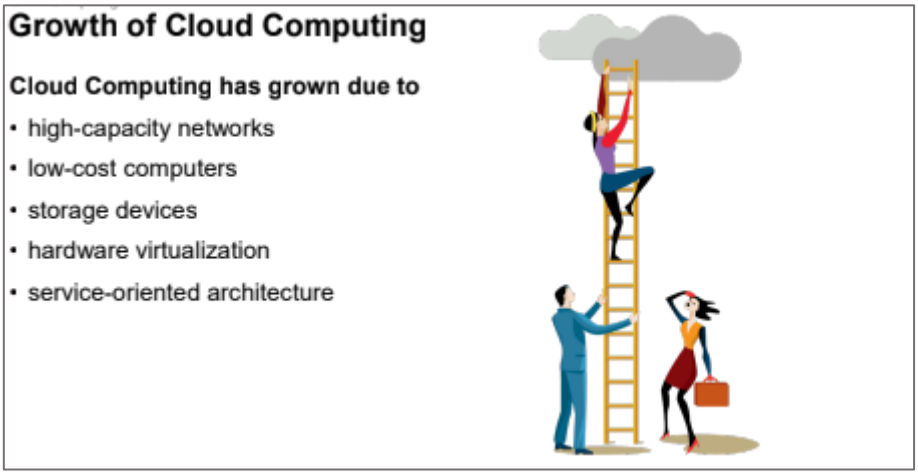
  The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no knowledge about or control over the exact location of the provided resources but may be able to specify location at a higher level of abstraction, such as country, state, or data center. Some examples of resources include storage, processing, memory, and network bandwidth.
- **Rapid elasticity.** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the user, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimize resource use by leveraging a

metering capability at some level of abstraction appropriate to the type of service, such as storage, processing, bandwidth, and active user accounts. Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Rather than keeping files on a proprietary hard drive or local storage device, cloud-based storage makes it possible to save them to a remote database. The availability of high-capacity networks, low-cost computers, and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing has led to growth in cloud computing.

With cloud computing we can get low-cost infrastructure whether its computing, database or network resources. We can easily manage these resources from any remote network. These cloud solutions are highly automated services provided by cloud providers to users so that users can easily provision infrastructure in cloud.



Cloud Service Models

- IaaS
- PaaS
- SaaS

Infrastructure as a service(IaaS) involves a method for delivering everything from operating systems to servers and storage through IP-based connectivity as part of an on- demand service.

Below are some of the features of using IaaS in Cloud Computing:

- Automated administrative tasks

- Dynamic scaling
- Platform virtualization technology
- GUI and API-based access
- Internet connectivity

IaaS is considered as one of the basic layer in cloud computing model. Some of the examples of IaaS service by various cloud providers are as below:

- Amazon EC2
- Windows Azure
- Rackspace
- Google Compute Engine

Amazon EC2: AWS EC2 is one of the services provided by Amazon Cloud which helps us to create servers, IP addresses, load balancers and other resources.

Platform as a service(PaaS) is a service in which cloud provider delivers hardware and software both, which means developers don't have to worry about infrastructure components.

PaaS supports some key features which helps us to easily deploy our applications on various platforms.

- Easy Development
- Flexibility
- Scalability, Load balancing and fail over
- On demand platform
- Container Based PaaS

PaaS platforms are available for public, private and hybrid clouds. Some of the cloud providers who provides PaaS services are listed below:

- AWS Elastic Beanstalk
- Google App Engine
- Azure App Service
- Pivotal Cloud Foundry

Software as a Service (SaaS) is a cloud computing offering that provides users with access to a vendor's cloud-based software. Users do not install applications on their local devices. Instead, the applications reside on a remote cloud network accessed through the web or an API. Through the application, users can store and analyze data and collaborate on projects.

SaaS is an new layer on top of PaaS, in which we user does not have perform any task. Below are some of the applications provided as SaaS:

- Google GSuite (Apps)
- Dropbox
- Salesforce
- Cisco WebEx

- GoToMeeting

7.9.2 Amazon Web Services (AWS)

AWS is a subsidiary of Amazon that provides flexible, reliable, scalable and cost effective cloud computing solutions. AWS provides various cloud services like IaaS, PaaS, SaaS which helps organizations to reduce their efforts in managing infrastructure components.

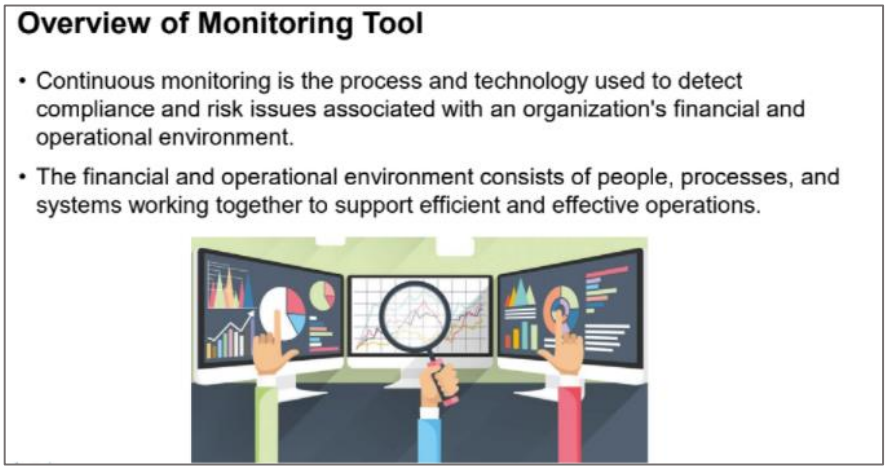
AWS was first launched in 2006 and still is one of the leaders in cloud computing solutions. Using AWS allows to create compute resources, storage resources, network resources, databases, load balancers, domain name and lot more. AWS offers pay-as-per-usage model which helps IT team to minimize infra expenses. AWS offers right now more than 100 cloud services to its user. It offers IaaS and PaaS to all their customers.
Amazon EC2: AWS EC2 is one of the services provided by Amazon Cloud which helps us to create servers, IP addresses, load balancers and other resources.

AWS Elastic Beanstalk: With AWS EBS we don’t have to know how to create servers, with this we just must select which platform we need and then deploy your application. With EBS we can deploy and test multiple applications at the same time.

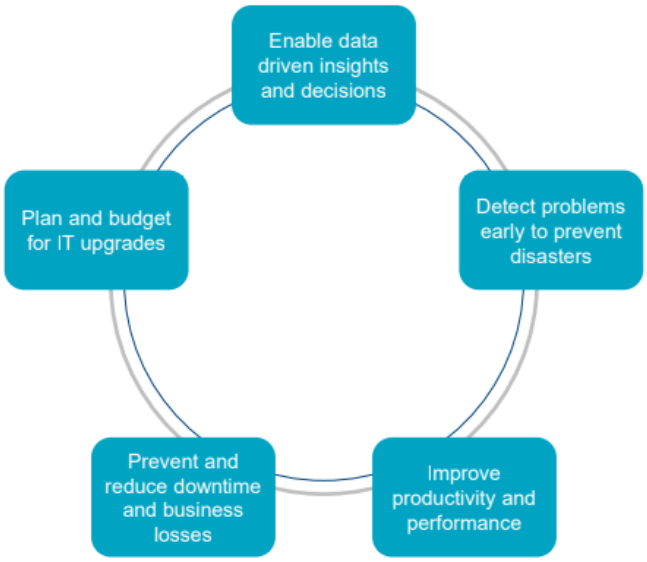
7.10 **Nagios, Prometheus- Monitoring**

7.10.1 Monitoring

Continuous monitoring is the process and technology used to detect compliance and risk issues associated with an organization's financial and operational environment.The financial and operational environment consists of people, processes, and systems working together to support efficient and effective operations.
Through continuous monitoring of the applications and processes, we will be able to detect failures in advance and fix them. With cyber threats ever evolving and growing at an exponential rate, and increased reliance on technology to deliver core services in government, a robust cyber defense is needed by agencies. Acts as an auditing tool where it can navigate through old monitoring data to analyze and improve performance of system.



Overview of Monitoring Tool



Monitoring Strategies

To have effective monitoring system strategies should be selected accordingly:

- Monitor all components
- Analyze both in house and third-party platforms
- Complete application monitoring
- Performance validation
- Re-evaluate Monitoring strategy
- Configure proper alerts

To have effective Monitoring system we can design strategy as per below:

- **Monitor all components:** We should setup all application components in such a way that all of them will be monitored by monitoring tool. It will help us to identify complete environment metrics.
- **Analyze both in house and third-party platforms:** We should not only analyze in house applications but also third-party application. Usually these third-party applications we skip but we may have some vulnerabilities in these third-party applications also.

- **Complete application monitoring:** We should not only monitor a portion or part of application; complete application monitoring is must for stable monitory strategy.
- Performance validation: In order to avoid alerts and issues in production environment, we should perform better performance testing so that we should not get much of alerts in monitoring system.
- **Re-evaluate Monitoring strategy:** As your company grows, and your application changes, your monitoring strategy should be re-evaluated. This will keep your monitoring system up to date so that you will not get issues.
- **Configure proper alerts:** Monitoring system should be properly configured with alert mechanism so that for every issue, user should be notified with alert system.

There are primarily three types of monitoring tools mentioned as below:

- **Real time monitoring:** This type of monitoring tool is used monitor live stats for CPU, memory, disk space etc.
- **Infrastructure monitoring:** This type of tool is used to monitor infrastructure components like web servers. Db servers, application servers etc.
- **Application monitoring:** This tool is used to monitor applications since sometimes even if server is accessible, application can be down and non-accessible.

A monitoring tool can provide you with a real-time view of your service status and infrastructure health. Most of the time these systems are used to identify the signs of poor performance on an enterprise-grade network.

7.10.2 Nagios

Nagios is a powerful monitoring system that enables organizations to identify and resolve IT infrastructure problems before they affect critical business processes. Designed with scalability and flexibility in mind, Nagios gives you the peace of mind that comes from knowing your organization's business processes won't be affected by unknown outages.

It is a very powerful monitoring tool used specially for system monitoring. Its main goal is to inform about any problem on the system to the administrator/user as soon as it occurs.

It monitors by dividing into 2 different categories:

1. Hosts/Machines: These are the physical machines like printers, servers, network devices like routers etc.
2. Services: These are the actual functionalities that run on top of the host/machine that needs monitoring like web server, mail server.

Multiple services can be grouped also for easy management & working.Monitoring checks are done by using plugins, Nagios doesn't perform anything by its own. Therefore, by having the plugins available for multiple different things, Nagios becomes very powerful & widely supported tool to be used in the market

for system monitoring.

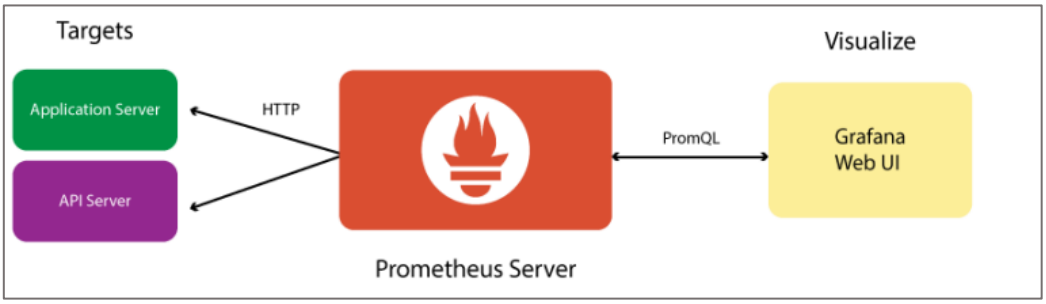
Plugins performs all the monitoring checks & reports are prepared with 4 different categories of output states:

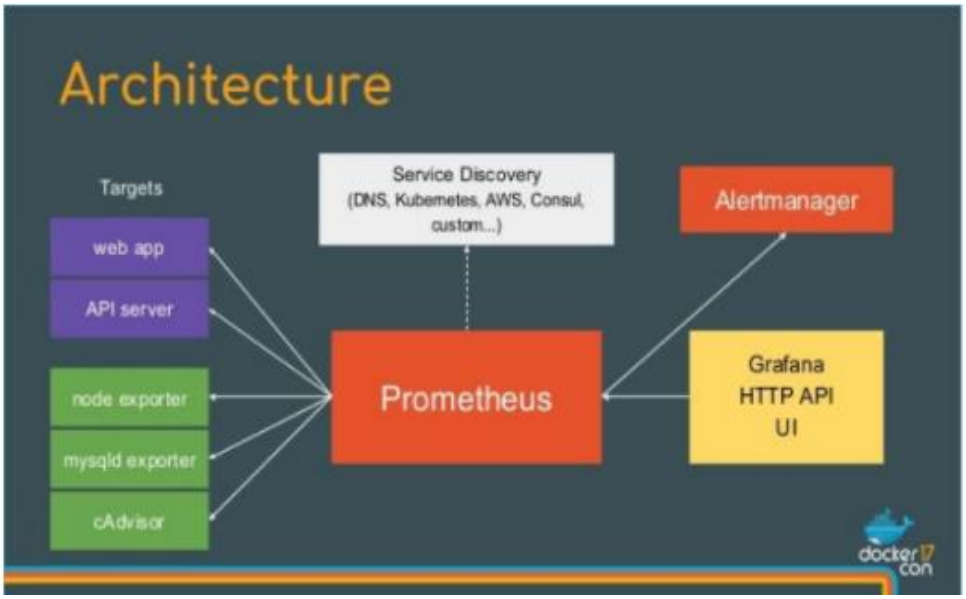
1. OK
2. Warning
3. Critical
4. Unknown

For each service, limits can be set that define the threshold using which its categorization is done. This approach of states helps the admins to only focus on the category instead of the severity level. In addition to that, Nagios also provides grouped reports as well which tells that how many services fall into which different output state that helps a lot in assigning the priority to the problems, as a result most critical problems gets resolved first.

7.10.3 Prometheus

Prometheus is open-source monitoring and alerting tool designed by Sound Cloud. It was first launched in year 2012 and from there it was managed completely by single organization. Grafana is bundled up with Prometheus for visualization monitoring and analyze data. Prometheus server collects metrics from different targets using HTTP connection and then displays it back on Prometheus server.





The Complete architecture of Prometheus monitoring tool.

We have various components for Prometheus.

- **Service Discovery:** It helps Prometheus to configure how different servers' sources will be checked by Prometheus monitoring tool.
- **Alert manager:** This is used to manage alerts in Prometheus so that in case of issues users will be notified.
- **Node exporter:** This component is used collect all metrics which will be pulled by Prometheus monitoring tool.
- **Grafana:** Grafana acts as HTTP GUI which helps to graph metrics stored by Prometheus.

Prometheus depends on pull model, which means Prometheus is responsible for getting metrics from services to be monitored. Tools like Graphite, who depends on clients to push their metrics from services to Graphite centralized server.
There are primarily two ways in which metrics are moved into a monitoring system: either the metrics get pushed (usually via UDP) into the system or they get pulled (usually via HTTP).
The push method is used in systems such as Graphite whereas the pull method is used by monitoring systems like Prometheus. When working with monitoring tools we need to monitor lot of servers and applications.
Service discovery (SD) enables you to provide that information to Prometheus from whichever database you store it in. Prometheus supports many service discovery integrations, such as Consul, Amazon's EC2, and Kubernetes. In case, your source is not supported, you can use the file-based service discovery mechanism.

7.11 **SONALQUBE**

SonarQube is a software testing tool which is used to improve the quality of the code and helps fix errors very early in the development. It is a static analysis code. It goes through developers' code and identifies errors.
Following are the key details about SonarQube:

- Helps to write cleaner code
- Helps in continuous code inspection
- Improves code reliability
- Increases application security

It manages source code quality and consistency. It is an open-source platform for continuous inspection of code quality. It provides reports for various code quality issues for example code duplication. Also gives

an analysis of Test Result. Also gives information about code coverage and information related to code complexity — whether it is simplified way or complex. It Describes design or architecture in simple or complex way. It Provides reports on Historical execution

Code Quality Checks Performed by SonarQube

Some of the code quality checks performed by SonarQube are as follows:

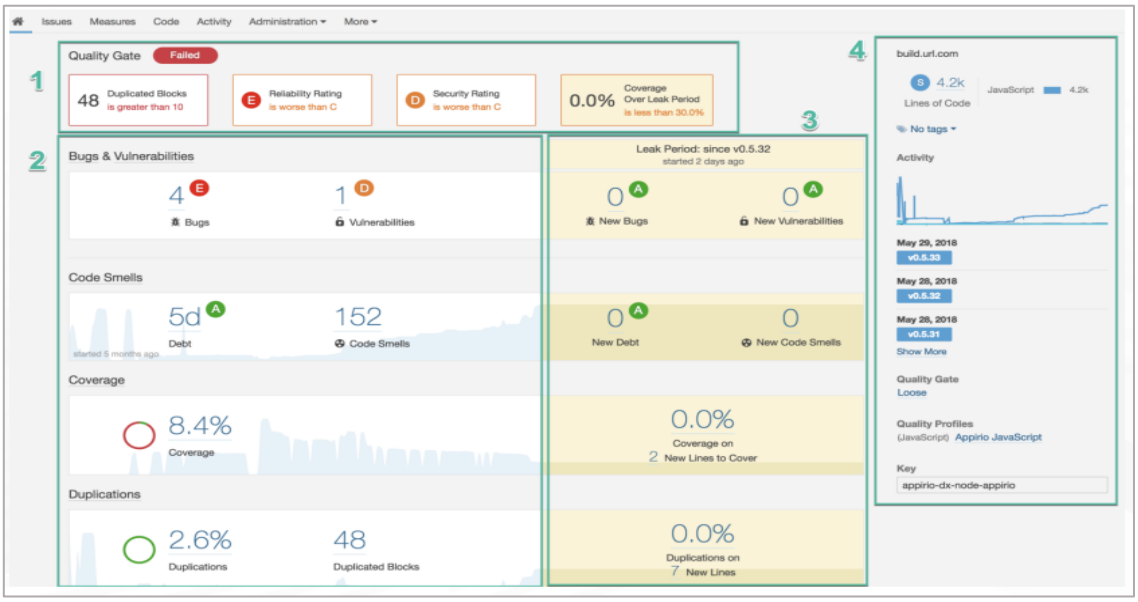
- Potential bugs
- Code defects to design inefficiencies
- Code duplication
- Lack of test coverage
- Excess complexity

Features of SonarQube

Some of the most important features of SonarQube are as follows:

- Can work with 25 different languages — Java, Dot Net, Cobol, Php, C++ to name a few.
- Can identify tricky issues. Some of which are as follows:
  - Detect bugs
  - Code smells
  - Security vulnerability
  - Activate rules needed
  - Execution path

Enhanced workflow: Ensures better CI/CD, automated code analysis, get access through webhooks and API, integrate GitHub, and analyze breaches and decorate pull requests. Built in methodology: Discover memory leaks, good visualizer, enforce a quality gate, digs into issues, and plugins for IDEs.



In a nutshell, we learnt:

1. Periodic Table
2. Git/Git Hub- Version Control System
3. Docker- Containerization
4. Jenkins- CICD
5. Terraform- Provisioning
6. Maven- Build & Release Management
7. Ansible- Configuration Management
8. Selenium- Test Automation
9. AWS- Cloud Computing
10. Nagios, Prometheus- Monitoring
11. SonarQube