1. What is DevOps?

DevOps is a set of practices that combines software development and IT operations to shorten the development lifecycle while delivering features, fixes, and updates continuously.

1. What is CI/CD?

CI/CD (Continuous Integration/Continuous Deployment) is a method to frequently deliver apps by introducing automation into the stages of app development.

1. Name some popular DevOps tools.

Jenkins, Docker, Kubernetes, Ansible, Terraform, Git, Prometheus, and Nagios.

1. What is containerization?

Containerization is a lightweight alternative to virtualization that involves encapsulating an application with its operating environment.

1. What is infrastructure as code (IaC)?

IaC is managing and provisioning infrastructure through code instead of manual processes.

1. What is Git branching strategy?

A Git branching strategy is a convention or set of rules that describes when branches are created, naming guidelines, and how they should be used.

1. What is Docker?

Docker is a platform that uses containerization technology to create, deploy, and run applications in isolated environments called containers.

1. What is Kubernetes?

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.

1. What is blue-green deployment?

Blue-green deployment is a technique that reduces downtime by running two identical production environments, with only one live at any time.

1. What is meant by configuration management?

Configuration management is the process of maintaining systems, such as servers, in a desired, consistent state.

1. What is a microservices architecture?

Microservices architecture is an approach to building applications as a collection of small, loosely coupled services that can be deployed independently.

1. What is the purpose of load balancing?

Load balancing distributes network traffic across multiple servers to ensure no single server becomes overwhelmed, improving reliability and availability.

1. What is a Docker image?

A Docker image is a lightweight, standalone, executable package that includes everything needed to run a piece of software.

1. What is a webhook?

A webhook is an HTTP callback that occurs when something happens; it's a way for apps to provide real-time information to other applications.

1. What is service discovery?

Service discovery is the process of automatically detecting devices and services on a network, enabling applications to locate and communicate with each other.

1. What is Git?

Git is a distributed version control system that tracks changes in source code during software development.

1. What is the difference between Git and GitHub?

Git is a version control system for tracking code changes, while GitHub is a cloud-based hosting service for managing Git repositories.

1. What is a Git repository?

A Git repository is a storage location for your project that contains all of your files and the history of changes made to those files.

1. What is a branch in Git?

A branch is a parallel version of a repository that allows you to work on different features without affecting the main codebase.

1. What is a pull request in GitHub?

A pull request is a method of submitting contributions to a project by requesting that someone review and merge your changes into their branch.

1. What is the purpose of the Git command "git clone"?

"git clone" creates a copy of an existing Git repository, typically from a remote server to your local machine.

1. What is the difference between "git pull" and "git fetch"?

"git pull" downloads changes and merges them into your current branch, while "git fetch" only downloads changes without merging them.

1. What is a merge conflict in Git?

A merge conflict occurs when Git cannot automatically resolve differences in code between two commits during a merge operation.

1. What is GitHub Actions?

GitHub Actions is a CI/CD platform that allows you to automate your build, test, and deployment workflows directly from your GitHub repository.

1. What is Git stash?

Git stash temporarily shelves (or stashes) changes you've made to your working directory so you can work on something else and come back to them later.

1. What is Ansible?

Ansible is an open-source automation tool used for configuration management, application deployment, task automation, and IT orchestration.

1. What is an Ansible Playbook?

An Ansible Playbook is a YAML file containing a series of tasks to be executed on remote hosts in a defined order.

1. What is Ansible inventory?

An Ansible inventory is a file that contains information about the hosts and groups that Ansible can manage.

1. What is an Ansible module?

An Ansible module is a reusable, standalone script that Ansible runs on your behalf to accomplish a specific task.

1. What is Ansible Galaxy?

Ansible Galaxy is a repository for Ansible roles that allows users to share and reuse Ansible configurations.

1. What is Maven?

Maven is a build automation and project management tool primarily used for Java projects to manage dependencies, build, and documentation.

1. What is a POM file in Maven?

POM (Project Object Model) is an XML file that contains project configuration information used by Maven to build the project.

1. What is a Maven artifact?

A Maven artifact is a file, usually a JAR, that is deployed to a Maven repository as a result of a build process.

1. What are Maven repositories?

Maven repositories are directories of packaged JAR files with metadata that Maven uses to download dependencies.

1. What is Jenkins?

Jenkins is an open-source automation server that enables developers to build, test, and deploy software continuously through automated pipelines.

1. What is a Jenkins pipeline?

A Jenkins pipeline is a suite of plugins that supports implementing and integrating continuous delivery pipelines using code (Pipeline as Code).

1. What is the difference between Freestyle and Pipeline jobs in Jenkins?

Freestyle jobs are configured through the UI for simple tasks, while Pipeline jobs use code (Jenkinsfile) to define complex, multi-stage build processes.

1. What are Jenkins agents/nodes?

Jenkins agents (or nodes) are machines that connect to the Jenkins controller and execute tasks when directed by the controller.

1. What is a Jenkinsfile?

A Jenkinsfile is a text file containing the definition of a Jenkins Pipeline written in Groovy DSL that is checked into source control.

1. What are Jenkins plugins?

Jenkins plugins are extensions that provide additional functionality to enhance the capabilities of the Jenkins automation server.

1. What is Blue Ocean in Jenkins?

Blue Ocean is a Jenkins plugin that provides a modern, visual user interface with improved pipeline visualization and personalized dashboards.

1. How does Jenkins achieve Continuous Integration?

Jenkins achieves Continuous Integration by automatically building and testing code changes whenever they are committed to the repository.

1. What is a Jenkins shared library?

A Jenkins shared library is a collection of reusable code that can be shared across multiple Jenkins pipelines to promote code reuse.

1. What is a Jenkins multibranch pipeline?

A Jenkins multibranch pipeline automatically creates a pipeline for each branch discovered in source control that contains a Jenkinsfile.

1. What is Selenium?

Selenium is an open-source automation testing framework used primarily for web application testing across different browsers and platforms.

1. What are the main components of Selenium?

The main components of Selenium are WebDriver, IDE, Grid, and the deprecated RC (Remote Control).

1. What is Selenium WebDriver?

Selenium WebDriver is a browser automation API that allows you to control browser actions programmatically from various programming languages.

1. What is Selenium Grid?

Selenium Grid is a tool that allows running test cases in parallel across different machines, browsers, and operating systems.

1. What are locators in Selenium?

Locators in Selenium are methods used to identify and locate web elements on a page, such as ID, XPath, CSS Selector, Name, Class Name, Link Text, and Tag Name.

1. What is Puppet?

Puppet is an open-source configuration management tool that automates the provisioning, configuration, and management of infrastructure using a declarative language.

1. What is a Puppet manifest?

A Puppet manifest is a file with a .pp extension that contains code written in Puppet's declarative language to define the desired state of resources.

1. What is Puppet Forge?

Puppet Forge is a repository of pre-built Puppet modules created by the community that can be used to extend Puppet's functionality.

1. What is Kubernetes?

Kubernetes is an open-source container orchestration platform that automates deploying, scaling, and managing containerized applications.

1. What is a Pod in Kubernetes?

A Pod is the smallest deployable unit in Kubernetes that can contain one or more containers which share network and storage resources.

1. What is a Kubernetes Service?

A Kubernetes Service is an abstraction that defines a logical set of Pods and a policy to access them, enabling network connectivity to a set of Pods.

1. What is a Kubernetes Deployment?

A Kubernetes Deployment is a resource object that provides declarative updates for Pods and ReplicaSets, allowing for easy scaling and rolling updates.

1. What is Chef?

Chef is an infrastructure automation platform that transforms infrastructure into code, automating how infrastructure is configured, deployed, and managed.

1. What is a Chef cookbook?

A Chef cookbook is a fundamental unit of configuration and policy distribution that contains recipes, attribute values, file distributions, and more.

1. What is a Chef recipe?

A Chef recipe is a file written in Ruby that describes the desired state of a resource and the steps needed to bring that resource to the desired state.

1. What is Docker?

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization technology.

1. What is a Docker container?

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

1. What is a Docker image?

A Docker image is a read-only template that contains a set of instructions for creating a Docker container.

1. What is Docker Compose?

Docker Compose is a tool for defining and running multi-container Docker applications using a YAML file to configure application services.

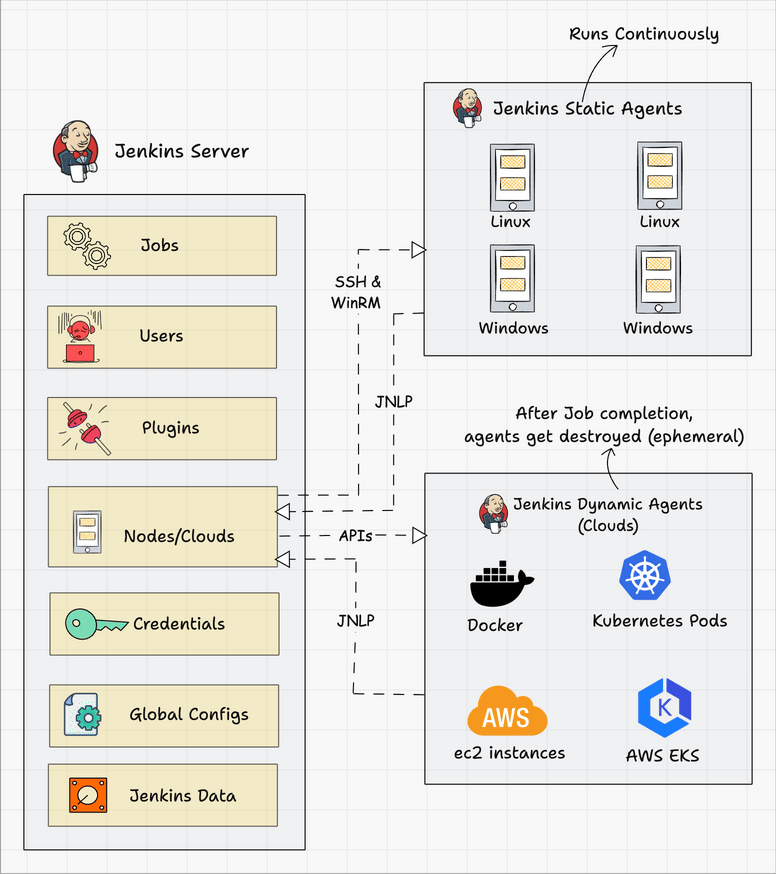
1. What is Docker Swarm?

Docker Swarm is a native clustering and orchestration tool for Docker that turns a group of Docker hosts into a single, virtual Docker host.

1. What is the difference between Docker and a virtual machine?

Docker containers share the host OS kernel and isolate application processes, while virtual machines include a complete OS and hypervisor, making containers more lightweight and efficient.

**Jenkins Architecture**



**Jenkins Server (Formerly Master)**

Jenkins's server or master node holds all key configurations. Jenkins master server is like a control server that orchestrates all the workflow defined in the pipelines. For example, scheduling a job, monitoring the jobs, etc.

**Jenkins Agent**

Jenkins agents are the worker nodes that actually execute all the steps mentioned in a Job. When you create a Jenkins job, you have to assign an agent to it. Every agent has a label as a unique identifier.

**Jenkins Jobs**

A job is a collection of steps that you can use to build your source code, test your code, run a shell script, run an Ansible role in a remote host or execute a terraform play, etc. We normally call it a [Jenkins pipeline](https://devopscube.com/jenkins-pipeline-as-code/).

**Jenkins Plugins**

Plugins are official and community-developed modules that you can install on your Jenkins server. It helps you with more functionalities that are not natively available in Jenkins.

**Jenkins Logs**

Provides logging information on all Jenkins server actions including job logs, plugin logs, webhook logs, etc.

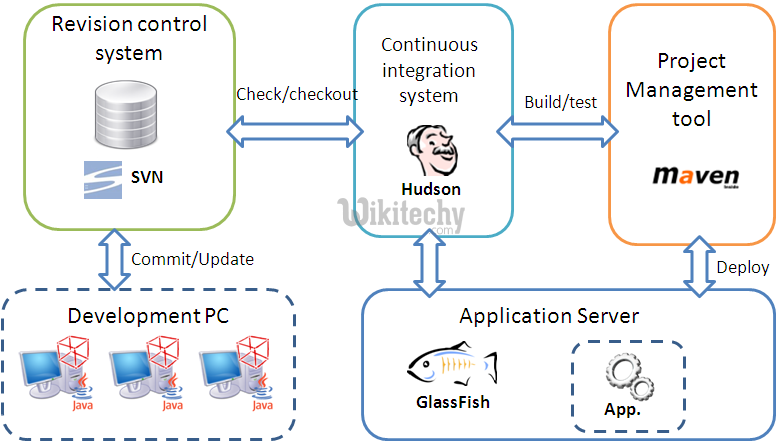
**Maven**

A diagram of a project

AI-generated content may be incorrect.

Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. It has a similar development process as ANT, but it is more advanced than ANT. Maven takes care of processes like releases, distribution, reporting, builds, documentation, and SCMs. Maven connects to the Maven Central repository and loads them in local.

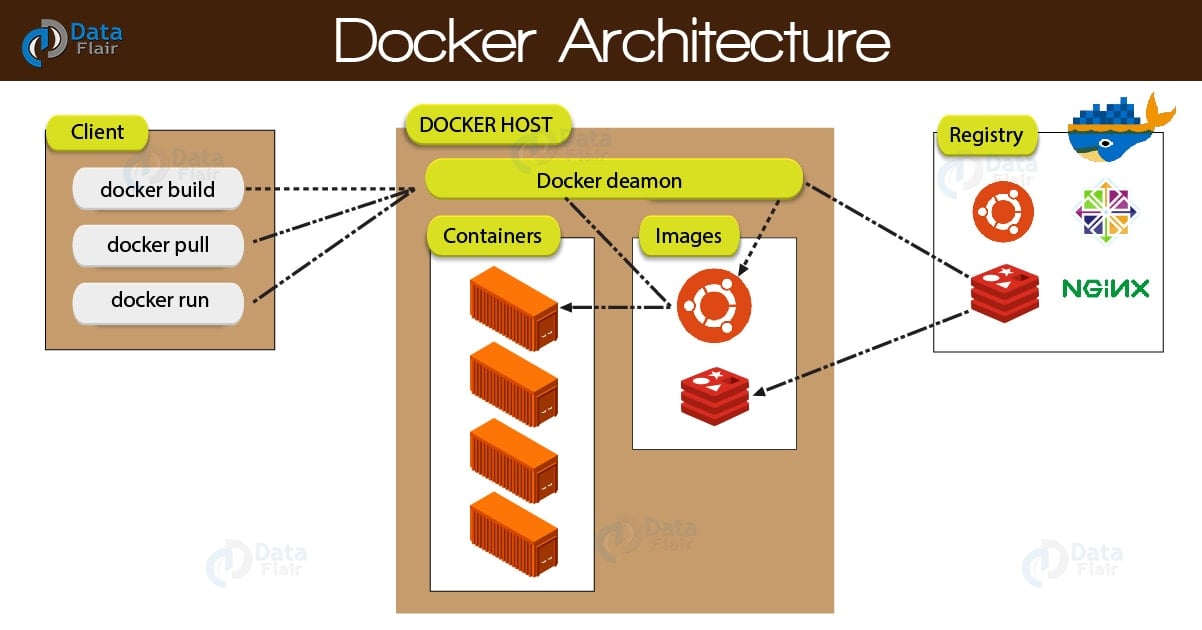
**Deployment Automation in maven**



**Docker Architecture**

Simply put, a tool which is designed to make it easier to create, deploy, and run applications by using containers is what we call *Docker*. It is an open platform for developing, shipping, as well as running applications.

As its best feature, it enables us to separate our applications from our infrastructure so delivering software quickly will become easy. Moreover, usage of Docker helps to manage the infrastructure easily and in the same ways, one manages their applications.



At very first, Docker client talks to the Docker daemon, which performs the heavy lifting of the building, running, as well as distributing of our Docker containers. Basically, both the Docker client and daemon can run on the same system or we can connect a Docker client to a remote Docker daemon.

In addition, by using a REST API, the Docker client and daemon communicates, over UNIX sockets or a network interface. Now, let’s learn about all three components of Docker Architecture in detail:

i. Docker Host

Basically, a Docker Host runs the*Docker Daemon*. As its job role, the daemon listens for Docker API requests such as ‘docker run’, ‘docker builds’, anything and also manages Docker objects like images, containers, networks, etc.

Moreover, one daemon can easily communicate with other daemons in order to manage Docker services.

ii. Docker Client

Here, the Client is nothing but the primary way by which many Docker users interact with Docker.

While we use various commands like (docker build, docker run, etc) the Docker client sends these commands to the Docker Daemon, which carries them out. One of the best features of Docker client is, it can easily communicate with more than one daemon.

iii. Docker Registries

A stateless, highly scalable server-side application, which stores and lets us distribute Docker images is what we call the Registry. There is a flexibility that either we can create our own image or we can use public registries such as **Docker Hub** and Docker Cloud.

By default, Docker is configured to look for images on one of its public registries “Docker Hub”. One of the advantages it offers is we can create our own registry as well.

Hence, while we run the docker pull or docker run command, it is possible to pull the required images from our configured registry. Basically, our image is pushed to our configured registry, while we use the docker push command.

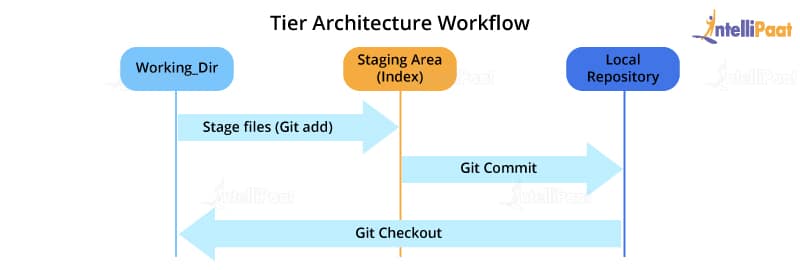
**Ansible Architecture**

A diagram of a cloud computing system

AI-generated content may be incorrect.

* **Users:**Users who create Ansible playbook has a direct connection with ansible automation Engine.
* **Ansible playbook:**Playbooks can finely orchestrate multiple slices of your infrastructure topology, with very detailed control over how many machines to tackle at a time. This is where Ansible starts to get most interesting. It also interacts with the ansible automation engine and configuration Management Database.
* **Public or Private cloud**: They help in interacting with all the modules and API with this but also with the entire cloud which proves that it has security measures as well.
* **Inventory**: Inventory which is a part of the automation engine helps in provisioning and internal provisioning using automation.
* **API**: It helps in creating necessary API for the interaction of end to end modules.
* **Modules**: The modules are directly run using playbooks the modules can control all services, packages, AWS cloud formation, etc. Ansible works by connecting to your nodes and pushing out scripts called “Ansible modules” to them. Most modules accept parameters that describe the desired state of the system. You can write your own modules, though you should first consider whether you should.
* **Plugins**: Plugins augment Ansible’s core functionality. While modules execute on the target system in separate processes (usually that means on a remote system), plugins execute on the control node within the /usr/bin/ansible process. Plugins offer options and extensions for the core features of Ansible - transforming data, logging output, connecting to inventory, and more.
* **Networking**: It helps to automate different networks that make use of all agentless frames and generate useful configurations.
* **Hosts**: Hosts here refers to the machines like Linux or Unix machines which are getting automated using Ansible.
* **CMDB** (Configuration Management Database): It is a kind of repository that consists of an entire network of computers of operational or IT infrastructure.

**Git**



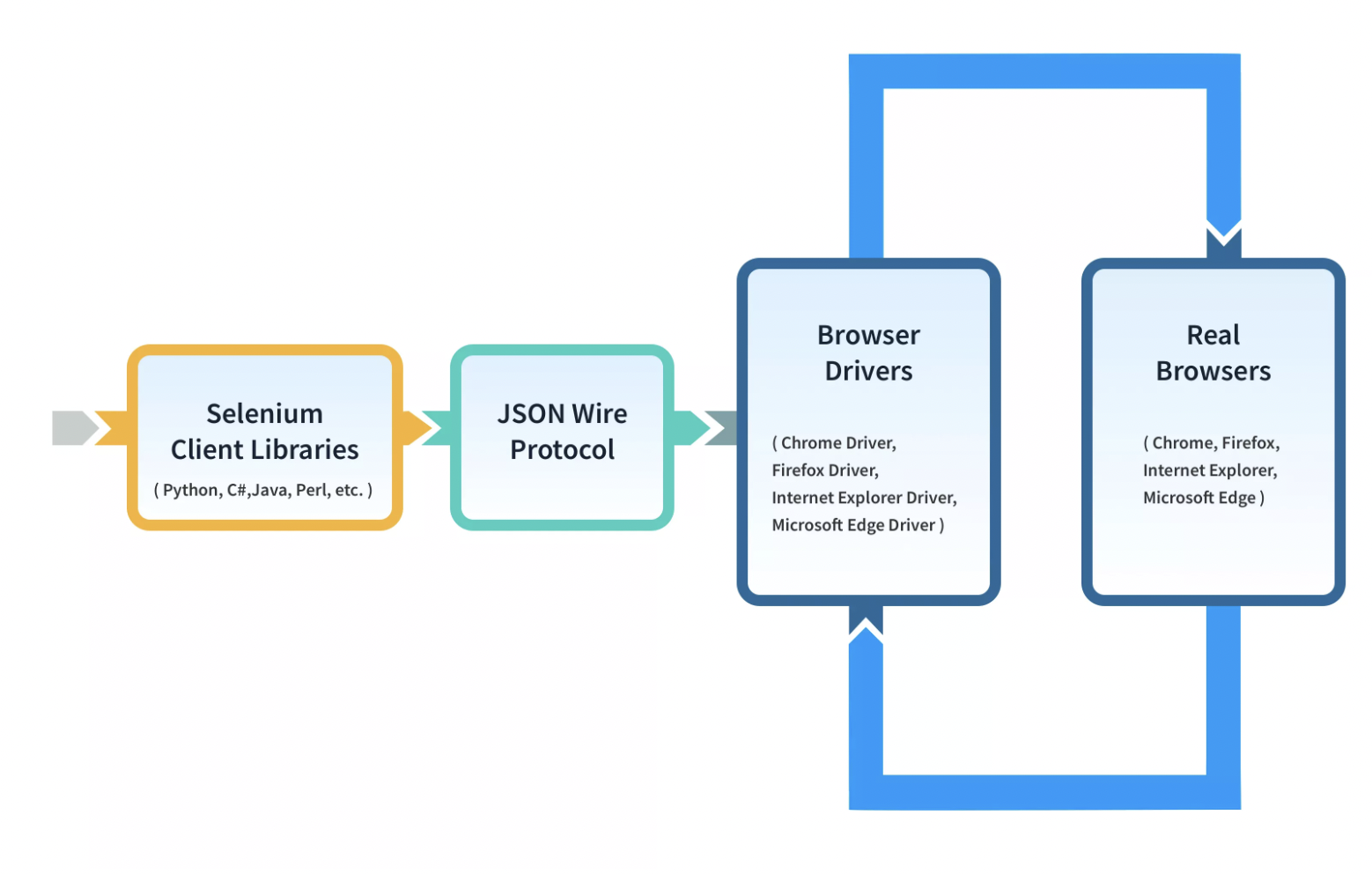
Most of the version control systems will work based on a two-tier architecture. However, what makes Git unique is that Git has an extra layer making it a three-tier architecture. But, why are there three layers of Git? Let’s find out.

The three layers are:

* **Working directory**: The working directory is created when we initialize the git repository allowing us to edit the source code.
* **Staging area**: Once we have made the appropriate edits in the files we will run the git add command specifying the files that we need to stage. This is essentially a preview for the upcoming step. If further changes are made in the first layer, i.e. the Working Directory then the resulting snapshots will be different. We can however sync these changes using the git add command.
* **Local repository**: Once we are done with all the changes we can finalize the changes for the files that have been staged using the git commit command.

**Selenium Web Driver**

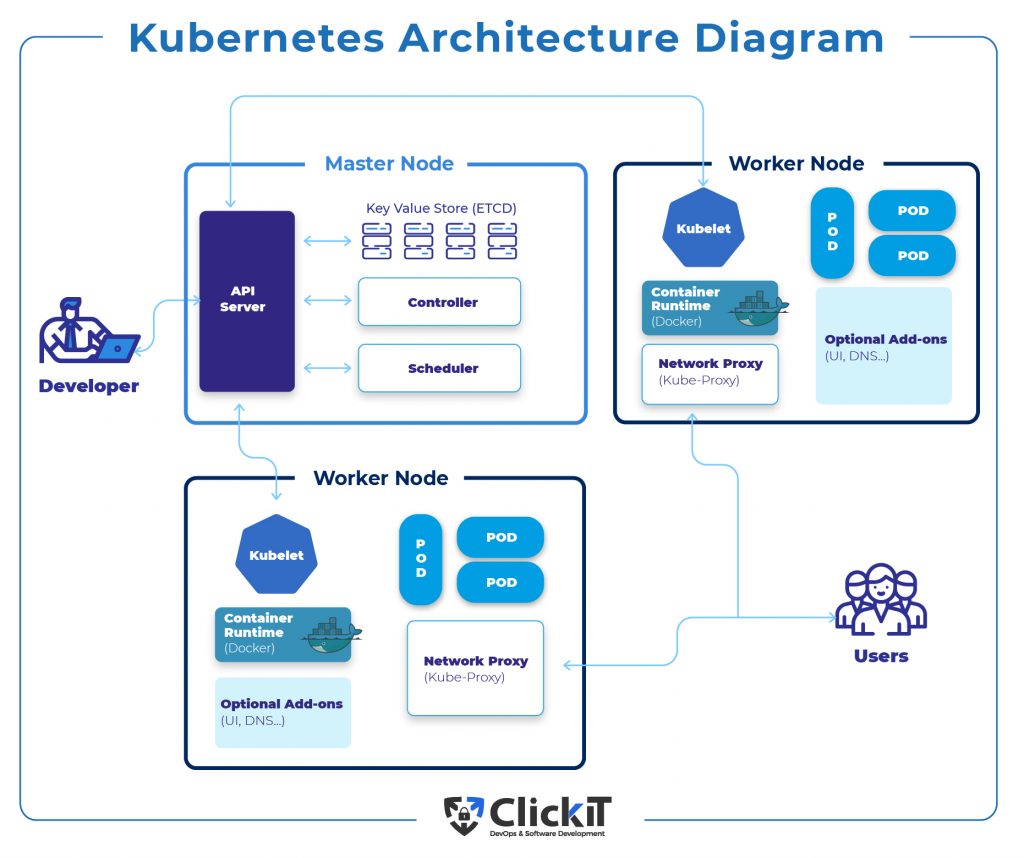
Selenium WebDriver is a popular open-source library and a key component of the Selenium automation framework used to automate testing for web applications. It is a collection of APIs which leverages a programming interface for developers and testers to write scripts in various programming such as Java, JavaScript, C#, Python, etc. to automate web browser’s action and retrieve information from web pages.



Selenium WebDriver Architecture is made up of four major components:

1. **Selenium Client library:** Selenium provides support to multiple libraries such as Ruby, Python, Java, etc as language bindings
2. **JSON wire protocol over HTTP:** JSON is an acronym for JavaScript Object Notation. It is an open standard that provides a transport mechanism for transferring data between client and server on the web.
3. **Browser Drivers:** Selenium browser drivers are native to each browser, interacting with the browser by establishing a secure connection. Selenium supports different browser drivers such as ChromeDriver, GeckoDriver, Microsoft Edge WebDriver, SafariDriver, and InternetExplorerDriver.
4. **Browsers:** Selenium provides support for multiple browsers like Chrome, Firefox, Safari, Internet Explorer etc.

**Kubernetes Architecture**



**Master Node Role in Kubernetes Architecture Diagram**

**Master Node is the starting point for all administrative tasks**, and its responsibility is managing the Kubernetes architecture diagram.

It’s possible to have more than one master node within the cluster, and what’s required for checking the fault tolerance as more master nodes will place the system in the mode known as “High Availability.”

However, one master node is the primary node that performs all the tasks.

**Kubernetes Master Node Components**

**Kubernetes API server**

* API server inside the master node is where all the administrative tasks will be performed.
* REST commands then go to the API server to process and validate the requests.
* The cluster’s resulting state will be stored based on the distributed key value upon request.

**Scheduler**

* This component schedules the tasks to specified slave nodes. Besides, each slave node will store information on resource usage.
* The scheduler will schedule all the work in the form of Services and Pods.
* Before the task is scheduled, the scheduler will consider the service requirements quality, affinity, anti-affinity, data locality, etc.

**Control manager**

* The control manager is a controller, a daemon that adjusts the Kubernetes cluster.
* Kubernetes cluster serves the purpose of managing various non-terminating control loops.
* This component also performs other functions, such as collecting node, event, and cascading-deletion garbage. Moreover, it has lifecycle functions like creating namespaces.
* In essence, a controller looks over the desired state of a managed object, but it also uses an API server to overlook and manage its current state. If the desired state of an object is not met, the control loop will ensure the current and desired state level by taking specific steps to achieve this goal.

**ETCD**

* This component distributes a key-value store that ultimately uses a cluster state.
* You can configure ETCD externally or make it a part of the Kubernetes Master.
* “Go” programming language is the one people use to write ETCD. In Kubernetes, you can store configuration details like Secrets, ConfigMaps, subnets, etc., and store the cluster state.

**Pod**

Pod is a single application controlled by one or several containers. A pod contains a unique network ID, application containers, and storage resources to determine how it’ll run containers.

**Service**

Pods can easily suffer a change. Therefore, Kubernetes can’t assure that a physical pod will remain alive (if the replication controller ends and begins with new pods).

Instead, the service will display a logical set of pods and act as a gateway. This means that you won’t have to keep track of the pods that make up the service, as pods can send requests to it.

**NameSpace**

is a virtual cluster that works in environments with multiple users across numerous projects. It’s worth mentioning that one physical cluster can run several virtual clusters simultaneously.

Resources within a namespace have to be unique, and they won’t be granted access to another namespace. Moreover, it’s possible to allocate a resource quota to a namespace so you can avoid overconsumption of overall resources found in the physical cluster.

**Volume**

In Kubernetes, the volume will apply to a whole pod. Therefore, it’ll mount on all containers located in the specified pod. Even if the container restarts, Kubernetes can guarantee that all the data will be saved. However, if the pod is killed, the volume will also disappear. A pod can have numerous volumes of different types.

**Deployment**

Deployment depicts the pod’s desired state or replica set, usually in a yaml file. The controller will slowly update the environment until the current and expected state match, as specified in the deployment file. This environment update includes deleting or creating replicas.

The yaml file defines two replicas for each pod. However, when only one is running, the yaml file definition will also create another one. Therefore, it’s essential to know that they shouldn’t be directly manipulated when deployment manages replicas. Use new implementations instead.