**ACID Transaction** 30

**API – Application Programming Interface** 1

**CAP Theorem** 30

**CMake** 21

**Database Architecture** 32

**Docker** 14

**Forward Proxy & Reverse Proxy** 26

**Gateway** 2

**GIT** 22

**Gunicorn/G-unicorn Library** 11

**HTTP基本** 4

**IP – Internet Protocol** 2

**IP-Internet Protocol**

IPv4 2

IPv6 2

Kubernetes/Container Orchestration 16

**Latency** 3

**Microservices** 35

Python Flask Libraryの基本 9

**Python Requests** 6

**Redis** 36

**SSL – Secure Sockets Layer TLS – Transport Layer Security** 25

**Subnet Mask** 2

**TCP** 3

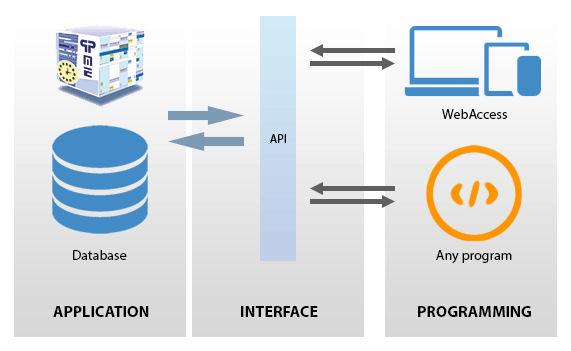
**UDP – User Datagram Protocol** 4

**VE – Vitual Environment** 28

**Web Scraping** 28

**API – Application Programming Interface**

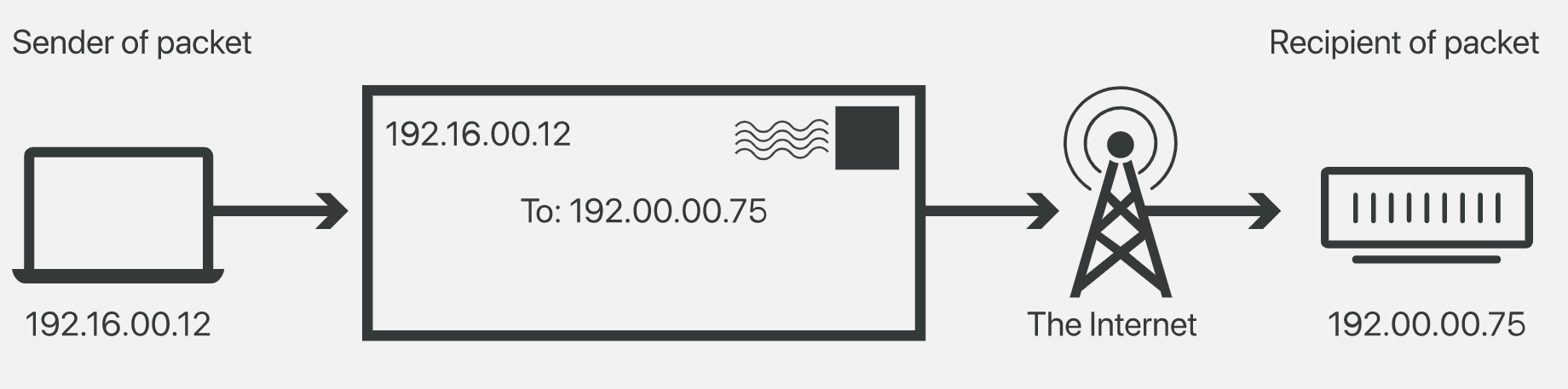
An API is a set of definitions and protocols for building and integrating application software. APIs let your product or service communicate with other products and services without having to know how they’re implemented. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc.



**IP – Internet Protocol**

The Internet Protocol (IP) is the method or protocol by which data is sent from one computer to another on the Internet. Every device/domain that connects to the Internet is assigned an IP address, and as packets are directed to the IP address attached to them, data arrives where it is needed.

Once the packets arrive at their destination, they are handled differently depending on which transport protocol is used in combination with IP. The most common transport protocols are TCP and UDP.



**IPv4** - The 4th version of IP. Because of the limit of the number of permutation/combination of IP address, the supply of available IPv4 addresses has become depleted. It used 32 bit address schema, so it can store up to 2^32 (4.29 billion) addresses.

192.168.123.54 can be represented as 11000000.10101000.01111011.00110110

Each of the numbers can range from 0-255.

**IPv6** - addresses have many more characters and thus more permutations; however, IPv6 is not yet completely adopted, and most domains and devices still have IPv4 addresses. It uses 128 bit address scheme, they are alphanumeric and they are separated by colons, they also contain hexadecimals.

**Public and Private IPs:**

Public IPs are IP addresses that can be accessed over the internet. All accessible websites have a public IP address mapped to a DNS. It is globally unique. Public IPs can incur cost.

Private IPs can be accessed within a LAN and cannot communicate with the outside world without a Public Internet Gateway like VPN. It is unique within the local area network. It is free of charge.

**DNS:**

The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to [IP addresses](https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/) so browsers can load Internet resources.

Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).

**Subdomain:**

Subdomains are the part of a domain that comes before the main domain name and domain extension. They can help you organize your website. For example, **docs**.themeisle.com. In this URL, docs is the subdomain.

This URL contains two parts:

* A protocol (https:) is a set of guidelines that a browser follows to send a request to the server.
* The domain, themeisle.com, or URL to the main website. A domain consists of two parts:
  + the TLD (top level domain) which is the .com part (or [another domain extension](https://themeisle.com/blog/domain-extensions-guide/)), and
  + the SLD (second level domain), themeisle, the name that you buy from a [domain registrar](https://themeisle.com/blog/best-domain-registrars/).

A subdomain contains a second name before the SLD. For instance, if the Themeisle blog was hosted on https://blog.themeisle.com, the blog would be the subdomain.

A subdomain is commonly used to logically separate a website into sections.

**Subnet:**

A subnet or subnetwork is a smaller network inside a larger network. it makes routing more efficient. Through subnetting, network traffic can travel shorter distance without through unnecessary routers to reach its destination.

Each IP address has 2 parts. The first part indicates which network the address belongs to to. The second part specifies the device (host) within the network.

Networks are categorized into 5 categories, A to E.

Class A: can connect to millions of devices. Everything before the 1st period indicates the network, everything after it specifies the device within that network.

In, 10.200.0.20, 10 specifies the network, 200.0.20 specifies the device.

Class B: Everything before the 2nd period indicates the network and rest indicates the device.

Class C: Everything before the 3rd period indicates the network and rest indicates the device.

Since Class A could be connected to millions of devices, it is more efficient to divide the network into multiple subnets and route traffic through them.

Routers use subnet mask to sort the data into subnets.

**Subnet Mask** - The subnet mask is used to subdivide a network into smaller, more manageable chunks. Routers use subnet masks to route data packets to the right place. Subnet masks are not indicated within data packets traversing the Internet — those packets only indicate the destination IP address, which a router will match with a subnet.

Let's assume that the source IP address is 192.168.40.15 and our destination IP address is 192.168.40.39. In order to determine if the devices are on the same subnet, we need the subnet mask. The mask designates where the network boundaries reside. Here, our subnet mask is 255.255.255.0.

The four octets of the IP address align with the four octets of the subnet mask. The 255 octets in the subnet mask tell us that the corresponding numbers in the IP address are static and never change. Therefore, we know that the first three octets -- 192.168.40 of our IP address -- designate the network portion of our destination IP. And since the fourth octet is 0, that means this is the host octet and that individual devices can be assigned any number from 1-254.

The host octet can be divided into subnet and host for eg. 5 bit subnet and 3 bit host.

2 address in the subnet IP range dedicated to the subnet address (first IP address) and the broadcast (last IP address). Rest of the addresses are usable.

| Network Mask Class C | Usable host per subnet |
| --- | --- |
| 255.255.255.0 | 2^8 - 0 - 2 = 254 |
| 255.255.255.128 | 2^8 - 128 - 2 = 126 |
| 255.255.255.192 | 2^8 - 192 -2 = 62 |
| 255.255.255.254 or 255.255.255.255 | 2^8 - 254 - 2 = 0 |

Subnet address can be obtained by binary addition of subnet mask and the IP address.

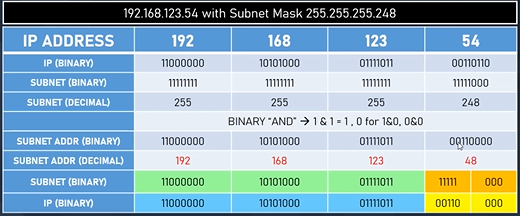
IP address = 192.168.123.30 = 11000000 10101000 01111011 00011110

Subnet Mask = 255.255.255.240 = 11111111 11111111 11111111 11110000

Subnet Address = 192.168.123.16 = 11000000 10101000 01111011 00010000

Broadcast IP = 192.168.123.31 = 11000000 10101000 01111011 00011111

Adding 1 to Broadcast IP will change the subnet.



**Gateway** - The default gateway is used as the destination of all traffic that is not on the same subnet. The gateway is a layer 3 device such as a router or multi-layer switch that is used to route traffic on a hop-by-hop basis.

Source Network: 192.168.**99**.15 255.255.255.0

Destination Network: 192.168.**40**.39 255.255.255.0

The default gateway always resides in the same subnet as the end device IP. The gateway can really be any unique address within the subnet itself, but most network administrators designate the first number of the subnet as the gateway. Therefore, 192.168.99.1 would be the default gateway of the source device given the fact that we have a 255.255.255.0 subnet mask.

**Latency :** Latency is the delay between a user's action and a web application's response to that action, often referred to in networking terms as the total round trip time it takes for a data packet to travel.

**Port forwarding:**

Port forwarding, or tunneling, is the behind-the-scenes process of intercepting data traffic headed for a computer’s IP/port combination and redirecting it to a different IP and/or port. A program that’s running on the destination computer (host) usually causes the redirection, but sometimes it can also be an intermediate hardware component, such as a router, proxy server or firewall.

Normally, a network router will examine the header of an IP packet and send it to a linked and appropriate interface, which in turn sends the data to the destination information that’s in the header.

But in port forwarding, the intercepting application (or device) reads the packet header, notes the destination, and then rewrites the header information and sends it to another computer—one that’s different from the one intended. That secondary host destination may be a different IP address using the same port, a different port on the same IP address, or a completely different combination of the two.

Port forwarding is an excellent way to preserve public IP addresses. It can protect servers and clients from unwanted access, “hide” the services and servers available on a network and limit access to and from a network. Port forwarding is transparent to the end-user and adds an extra layer of security to networks.

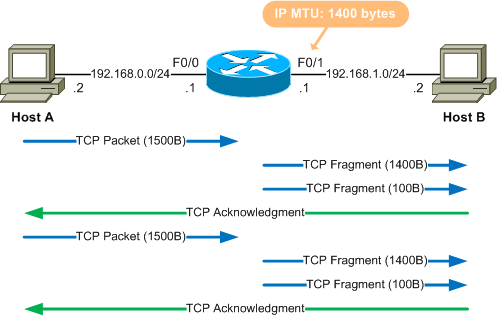
port forwarding is used to keep unwanted traffic off networks. It allows network administrators to use one IP address for all external communications on the Internet while dedicating multiple servers with different IPs and ports to the task internally. Port forwarding is useful for home network users who may wish to run a Web server or gaming server on one network.

The network administrator can set up a single public IP address on the router to translate requests to the proper server on the internal network.

**Path MTU discovery:**

When a host needs to transmit data out an interface, it references the interface's *Maximum Transmission Unit (MTU)* to determine how much data it can put into each packet. Ethernet interfaces, for example, have a default MTU of 1500 bytes, not including the Ethernet header or trailer. This means a host needing to send a TCP data stream would typically use the first 20 of these 1500 bytes for the IP header, the next 20 for the TCP header, and as much of the remaining 1460 bytes as necessary for the data payload. Encapsulating data in maximum-size packets like this allows for the least possible consumption of bandwidth by protocol overhead.

MTU is not constant among hosts, and Path MTU Discovery (PMTUD) is a standardized technique in computer networking for determining the maximum transmission unit (MTU) size on the network path between two Internet Protocol (IP) hosts, usually with the goal of avoiding IP fragmentation.

****

**TCP – Transmission Control Protocol**

TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain [a network conversation](https://www.searchmobilecomputing.techtarget.com/opinion/Deep-dive-into-internet-capacity-during-a-pandemic) through which application programs can exchange data. TCP works with the Internet Protocol ([IP](https://searchunifiedcommunications.techtarget.com/definition/Internet-Protocol)), which defines how computers send [packets](https://searchnetworking.techtarget.com/definition/packet) of data to each other. Together, TCP and IP are the basic rules defining the Internet.

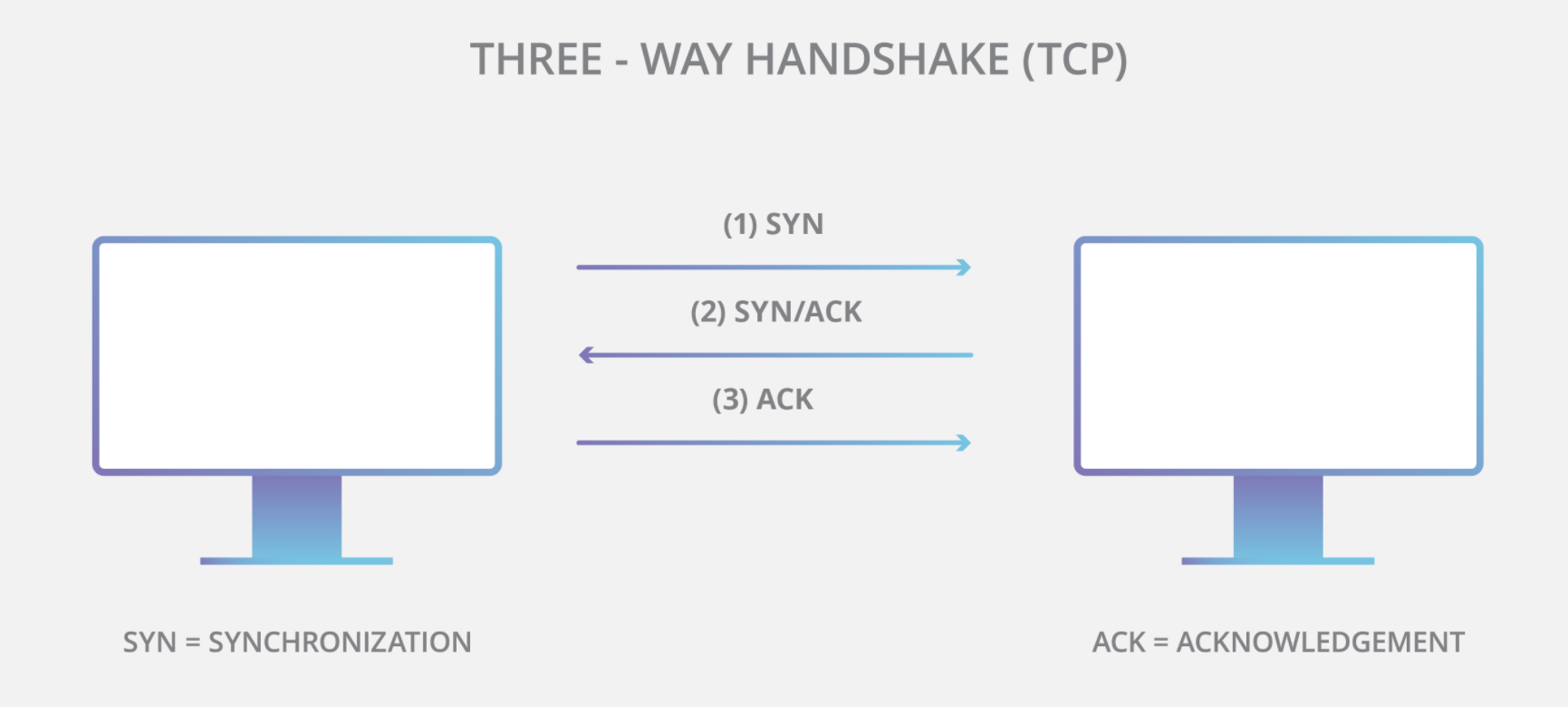
It is a layer above IP and below transport layer (HTTP, FTP etc.) and provides 3 way security:

Request, ACK, and transfer. It is computationally expensive.

IP is a connectionless protocol, which means that each unit of data is individually addressed and routed from the source device to the target device, and the target does not send an acknowledgement back to the source. That’s where protocols such as the Transmission Control Protocol (TCP) come in. TCP is used in conjunction with IP in order to maintain a connection between the sender and the target and to ensure packet order.

TCP is designed for reliability, not speed. Because TCP has to make sure all packets arrive in order, loading data via TCP/IP can take longer if some packets are missing. This can introduce latency.





**UDP – User Datagram Protocol**

The User Datagram Protocol, or UDP, is another widely used transport protocol. It's faster than TCP, but it is also less reliable. UDP does not make sure all packets are delivered and in order, and it doesn't establish a connection before beginning or receiving transmissions.

UDP/IP is usually utilized for streaming audio or video, as these are use cases where the risk of dropped packets (meaning, missing data) is outweighed by the need to keep the transmission real-time. For instance, when users are watching a video online, not every pixel has to be present for every frame of the video. Users would rather have the video play at normal speed than sit and wait for every bit of data to be delivered.

**SSH:**

It allows secure data access between two computers, it also supports tunnelling. It uses public key cryptography for authentication.

Installation:

$ sudo apt install openssh-client

$ sudo apt install openssh-server

To connect, we need the IP address of the remote machine.

$ sudo ip addr show

Connect:

$ ssh user\_name@ip\_address

$ # enter password

$ exit

**File transfer protocols - SSL, FTP, SCP and SFTP:**

They are network protocols for Remote File Transfer.

SFTP:

IETF extension of SSH 2.0 and also TLS. SSH is at layer 7 of OSI model.

It is platform independent, and assumes SFTP clients are authenticated using SSH. It supports remote file access, file transfer and file management. It can also resume broken transfers and guarantee delivery.

We can use FileZilla to use sftp with gui.

SCP:

it only supports file transfers. It uses SSH (Port 22) and is mostly supported by Linux. It is faster than SFTP.

SCP protocol is outdated and inflexible and SFTP is recommended.

*To copy a file from computer A to B, when logged into A:*

$ scp A\_file\_path B\_username@B\_ip:folder\_path

*To copy a file from computer B to A, when logged into A.*

$ scp B\_username@B\_ip:file\_path A\_folder\_path

FTP:

It is the standard protocol to transfer files over a network, such as the internet. People can upload their files on a FTP server and other people connected to that server would be able to access them. It is insecure and there is no encryption. It uses port 21. It uses TCP for transport, and guarantees file delivery.

TFTP (Trivial File Transfer Protocol):

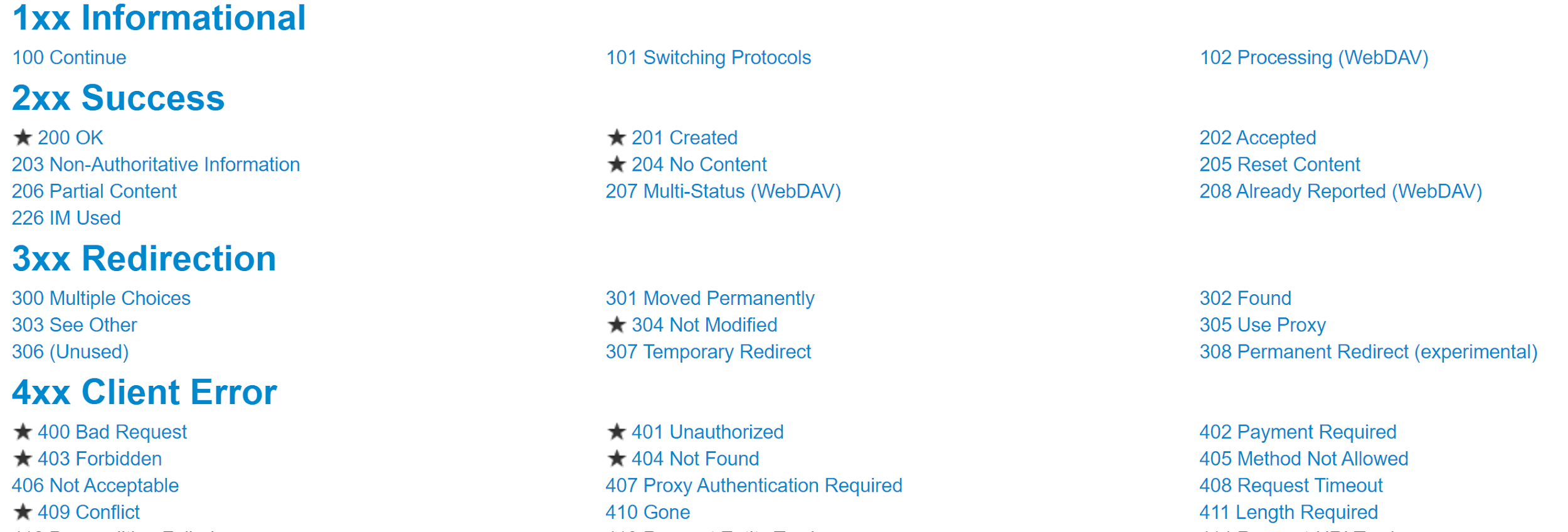
It is a connectionless protocol and uses UDP.

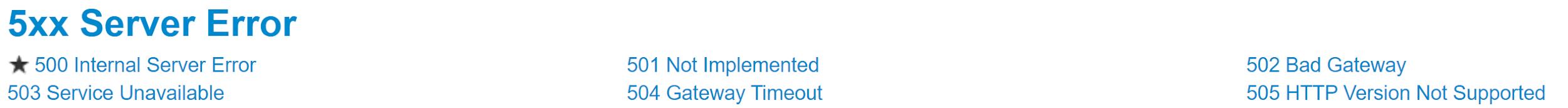
**OpenSSL:**

It is a robust, commercial grade cryptographic library and toolkit for TLS/SSL.

**HTTP基本：**

HTTP status code：responseステータスコードは、特定の HTTP リクエストが正常に完了したどうかを示します。レスポンスは 5 つのクラスに分類されています。





1) 情報レスポンス (100–199)

100(Continue）サーバがリクエストの受信をし拒否していない

102(Processing) 処理を継続中

2) 成功レスポンス (200–299)

200(OK) 正しく表示されている

3) リダイレクト (300–399)

302(Moved Temporarily) 一時的にページを転送

304(Not Modified) リソースが未更新

4) クライアントエラー (400–499)

400 (Bad Request) リクエストが不正

401 (Unauthorized) 認証が必要

403(Forbidden) アクセス禁止

404(Not Found) ページが見つからない

410 (Gone) リクエストが消滅

5) サーバエラー (500–599)

500(Internal Server Error) サーバ内のエラー

503(Service Unavailable) サーバが一時的に利用不可

参照：

<https://developer.mozilla.org/ja/docs/Web/HTTP/Status>

**HTTP Method：**

クライアントが行いたい処理をサーバに伝えるメソッドです。GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE, CONNECTと八つに分類されています。

主にGET, POST, PUT, DELETEを使います。

CURD名 意味 メソッド

Create 作成 POST/PUT

Read 　　　読込み GET

Update 更新 PUT

Delete 削除 DELETE

参照：

<https://qiita.com/fukulingo/items/a9e8d18467fe3e04068e>

<https://www.ipride.co.jp/blog/2201>

**Python Requests:**



インストール方法：pip install requests

HTTP reqestを聞いてresponseオブジェクトをもらえる。responseはWebサイトの情報を受け取る仕組み

response = request.get(url)

print(response.status\_code) # 成功した場合の結果：　200

Get関数：

１）HTTP通信で、サーバから情報を取得してくる時に使用する

２）他人に見られたくない情報は、GETでは送らない（後述する）

３）送信できるデータ量に制限がある

４）テキストデータのみ送信できる（バイナリデータは送信できない）

Python response objectのheaders関数でレスポンスヘッダを取得できる。

print(response.headers)

# print(response.headers['Content-Type']) # 'text/html'

print(response.headers['Date']) # 'Thu, 12 Jul 2018 11:58:54 GMT'

print(response.headers['Last-Modified']) # 'Fri, 09 Aug 2013 23:54:35 GMT'

print(print(response.headers['Server']) # 'ECS (oxr/8313)'

HTML情報はtextパラメータやjson関数でもらえます

print(response.text)　　　＃print(response.json())

# <!doctype html>

# <html>

# <head>...</head>

# <body>...</body>

# <html>

URLパラメータを指定

URLの末尾に?をつけてそのあとにkey=valueの形式で値を指定することでパラメータを指定することができる。

# <https://www.google.co.jp/search?q=%E6%97%A5%E6%9C%AC%E4%BB%A3%E8%A1%A8&tbm=nws>

url = '<https://www.google.co.jp/search>'

params = {'q': 'abc', 'tbm': 'abc'}

headers = {...}

r = requests.get(url, headers=headers,params=params)

print(r.url)

post関数：

HTTP通信で、サーバへ情報を登録する時に使用する（データベースへの格納など）

データ量が多い場合（GETでのデータ送信量制限を超えてしまう場合）

バイナリデータを送信したい場合

他の人に見られたくない情報を送る場合Postを使える（パスワードなど）

data = {Key-Value pair}

response = request.post(url,data=data)

参照：

<https://qiita.com/sqrtxx/items/49beaa3795925e7de666>

<https://realpython.com/python-requests/>　　（英語）

-----------------------------------------------

Python Flask Libraryの基本：



Djangoと似たようなLibraryで、Web開発でJavascriptの代わりに使うことが出来ます。

インストール方法：pip install Flask

Flask経由でwebpageをserverに繋がってデータ処理、経路選択が出来ます。

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/')

def display():

name = "Hello World"

return name

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

local環境で既存portは5000に設定されているので、実行すると結果は、<http://localhost:5000/>でから見れます。

HTMLのコードを表示したい場合は、

@app.route('/')

def display():

return '<!doctype html><html>.....</html>'

または、HTMLファイルを別途に作成して添付することも出来ます。

@app.route('/')

def display():

return render\_template('hello.html', title='title')

参照：

<https://qiita.com/zaburo/items/5091041a5afb2a7dffc8>

----------------------------------------

**Gunicorn/G-unicorn Library:**



インストール方法：pip install gunicorn

用語：

Web Server Gateway Interface (WSGI; ウィスキー) は、WebサーバとPython Web アプリケーションに接続するための、標準化されたインタフェースです。

GunicornはUNIXに使うPython WSGI HTTP Serverです。

Webサイトを公開するときにはいろいろなサーバーが動いています、その中でNginx（エンジンエックス）とApache HTTP Serverは最も使われています。

NGINXは、

１）Apacheと同じオープンソースのWebサーバーソフトである

２）高い処理性能や並行処理能力を持つ

３）メモリ消費が少ない

４）ロードバランサーなどとしても使える

Nginx was quite different than the traditional way in which web servers implement thread-based models for serving the request. In the traditional thread-based models, for each client there is one thread which is completely separate and is dedicated to serve that thread. This might cause I/O blocking problems when process is waiting to get completed to release the resources (memory, CPU) in hold. Also, creating separate processes consumes more resources.

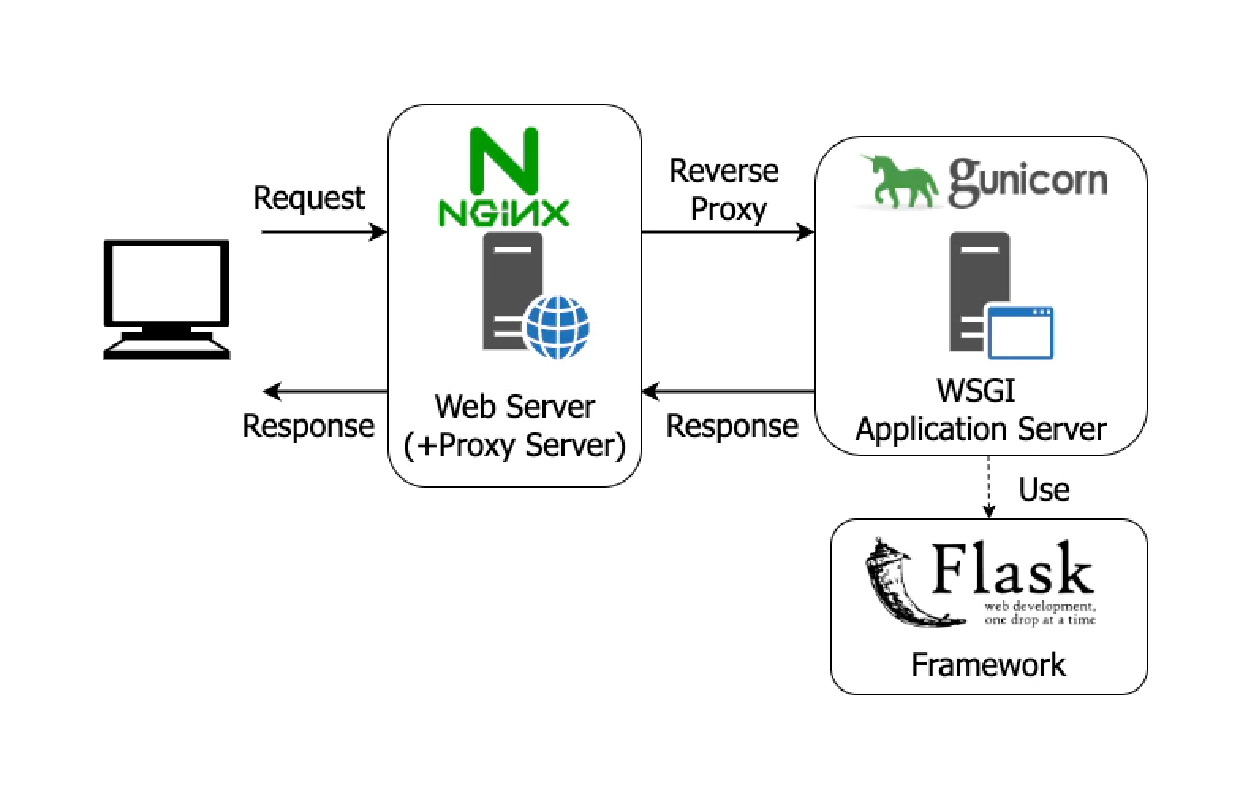
In Nginx, the solution to solve the above problem is to use Event-driven, asynchronous, non-blocking and single threaded architecture as shown in the diagram below.

ダイアグラム

自動的に生成された説明

It means that the threads can be executed concurrently with out blocking each other. It enhances the sharing of resources without being dedicated and blocked.

Applicationの仕組み：



$cd nginx-1.19.4

$start nginx # to start nginx

$tasklist /fi "imagename eq nginx.exe" # to see the process

$nginx -s quit # to stop and remove the process

$nginx -s signal

Where signal may be one of the following:

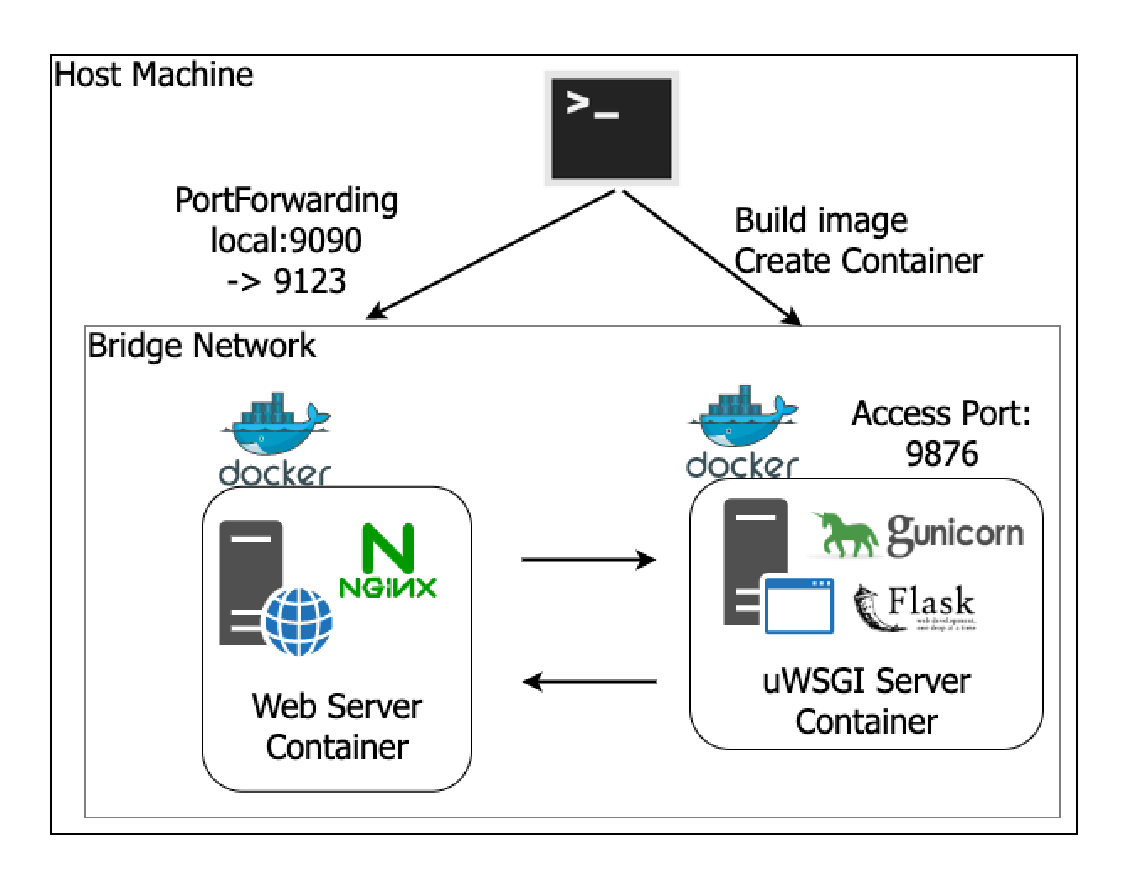
stop — fast shutdown

quit — graceful shutdown

reload — reloading the configuration file

reopen — reopening the log files

Docker及びContainer化



Gunicornを使うため、Web server設定ファイルを作成します。

＃　server\_settings.py

import os

bind = '127.0.0.1:' + str(os.getenv('PORT', 9876)) #portの設定

proc\_name = 'Infrastructure-Practice-Flask'

workers = 1　　　　　　　　　　　　　　　　　　　　　　#worker数の設定

またNginxの.confファイルを作成します

下記のコマンドで実行出来ます。

# Webサーバの起動

*＄nginx -c nginx.conf*

# アプリケーションサーバの起動

*＄gunicorn flask\_app:app -c server\_settings.py*

nginx.confファイルで指定されたportからFlaskのアプリケーションを見れます。

また続く

参照：

<https://qiita.com/mintak21/items/eeba4654a0db21abcb1c>

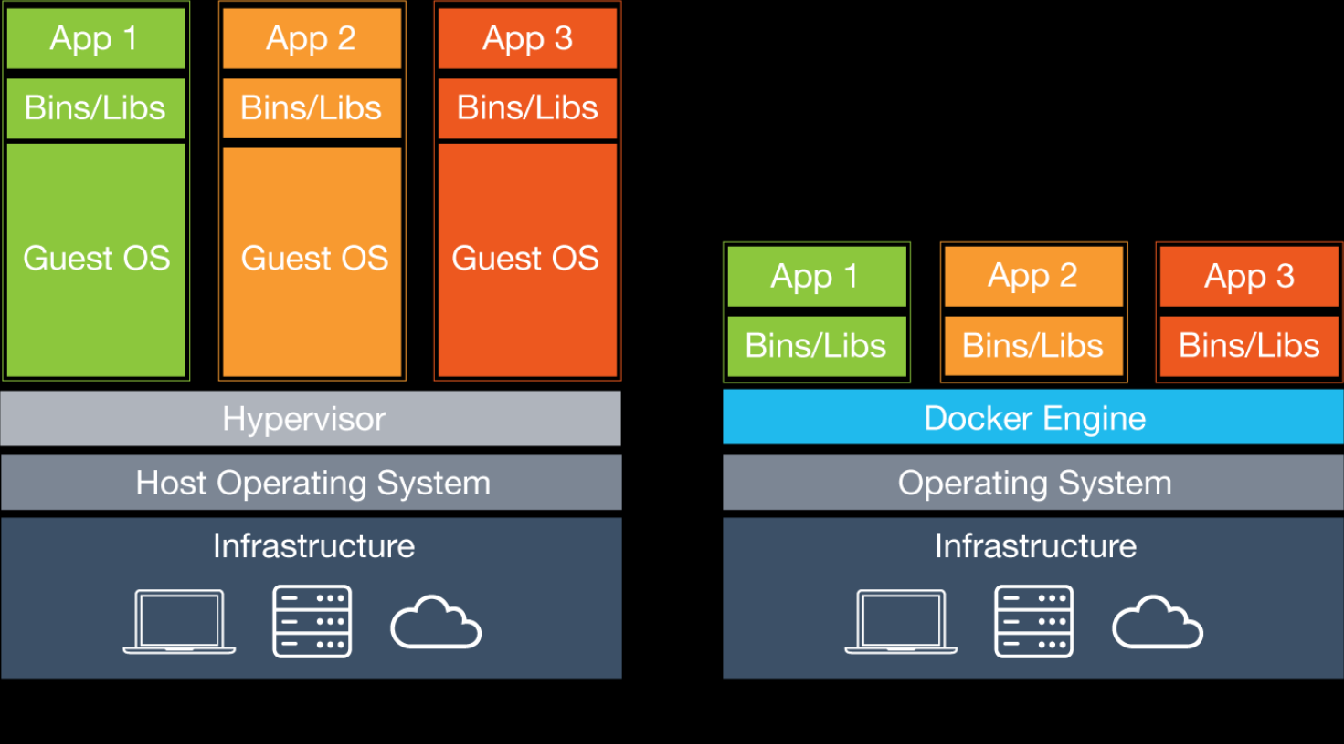
-------------------------------

**Docker：**

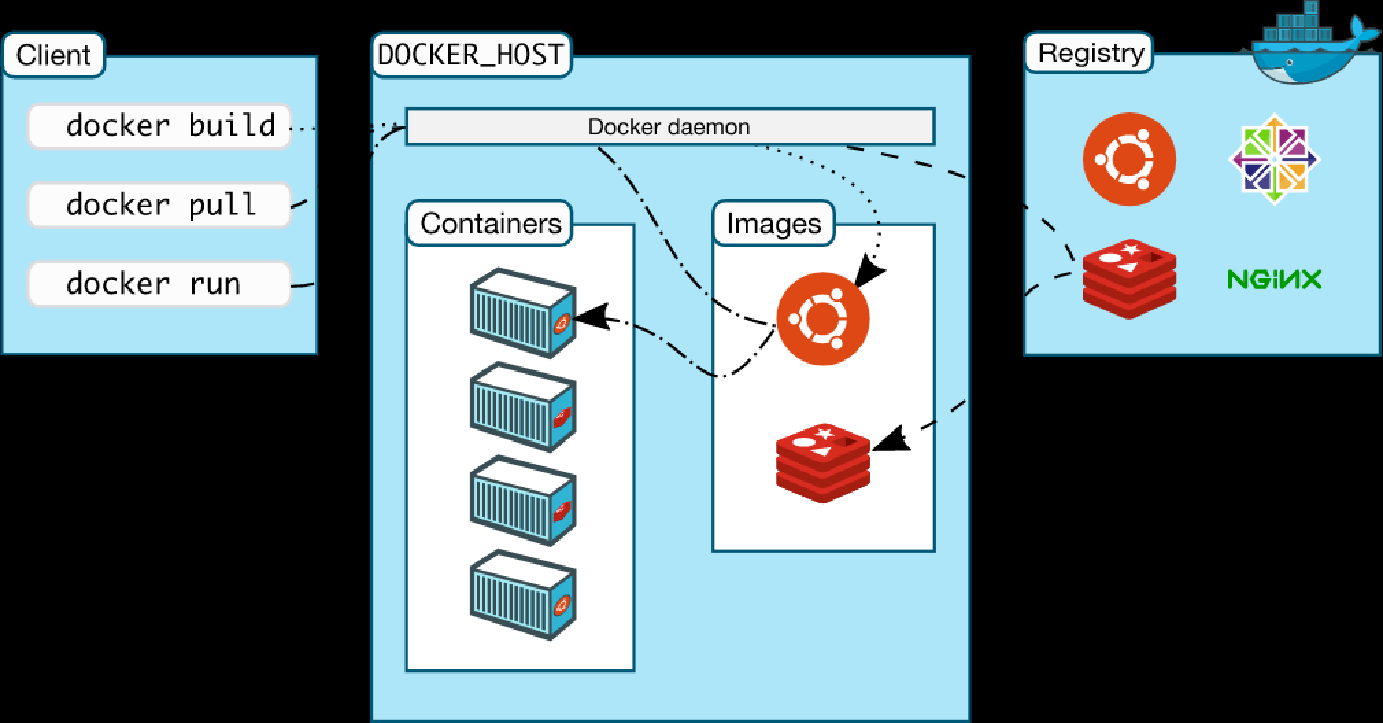
DockerソフトウェアでApplicationをContainer化出来ます。

Applicationを単独に稼働するため従来はVirtual Machine（仮想化）を使われていましたが、Dockerの大きな特徴は、どんなアプリケーションでも、場所を問わずに構築・移動・実行できるプラットフォームを提供することです。場所を問わずとは、物理環境でも、仮想化基盤でも、クラウド環境でも、LinuxでもWindowsでも対応可能という意味です。

仮想マシンのときのように使う、仮想化ソフトフェア + ゲストOSが不要になります。



Docker HubはDocker Image Repositoryです、ここにあるImageをそのままPullするかまたはそのImageを基に新しいContainerを作ることが出来ます。Docker Imageを作るためにDockerfileが必要です。



事例：

FROM centos:7 # ①　osまたはDocker Hubにある他のdocker container

RUN yum install -y java # ②　インストール

ADD files/apache-tomcat-9.0.6.tar.gz /opt/ # ③

CMD [ "/opt/apache-tomcat-9.0.6/bin/catalina.sh", "run" ] # ④

DockerfileはFromから始まり、必要なパッケージをインストールし、ファイル保存先を設定し、実行します。CMDは最後のコマンドで、一つのCMDしかありません。

Docker Build:

*$docker build -t image\_name ./dockerfile\_path*

Create Container:

*$docker run -it -p 5000:5000 –name container\_name -v $pwd/:/test image\_name* (Interactive mode)

Or

*$docker run -d -p 5000:5000 –name container\_name -v $pwd/:/test image\_name*  (Detached mode)

Execute on bash:

*$docker exec -it container\_name bash*

また続く

参照：

<https://knowledge.sakura.ad.jp/15253/>

-----------------

Kubernetes/Container Orchestration:

一つのアプリケーションで大量のアクセスが必要な場合、ネットワークのルーティングや複数コンテナの連携、複数台のサーバーを対象にコンテナを横断的に管理することが必要です。

このようなDockerと連携して利用できるデプロイ/オーケストレーションツールとして、主にKubernetesとDocker Swarmを使われています。このソフトウェアから、コンテナの状態を管理し、不具合なコンテナがある場合は故障を避けるため新しいコンテナを作ってアプリケーションを持続出来ます。

Kubernetes Features:

Automatic bin packing (based on memory):

Service discovery:

Storage Orchestration:

Self-healing:

Automated Rollout and Rollbacks without downtime:

Secret and config management:

Batch execution:

Hrizontal scaling:

kubernetes components:

apiserver

kubernetesを操作するためのAPIを提供する

controller-manager

コンテナの状態管理やノードの管理と言った各種管理作業を行う

proxy

コンテナへのネットワークルーティングおよび負荷分散を行う

scheduler

各ノードに対しコンテナの割り当てなどを行う

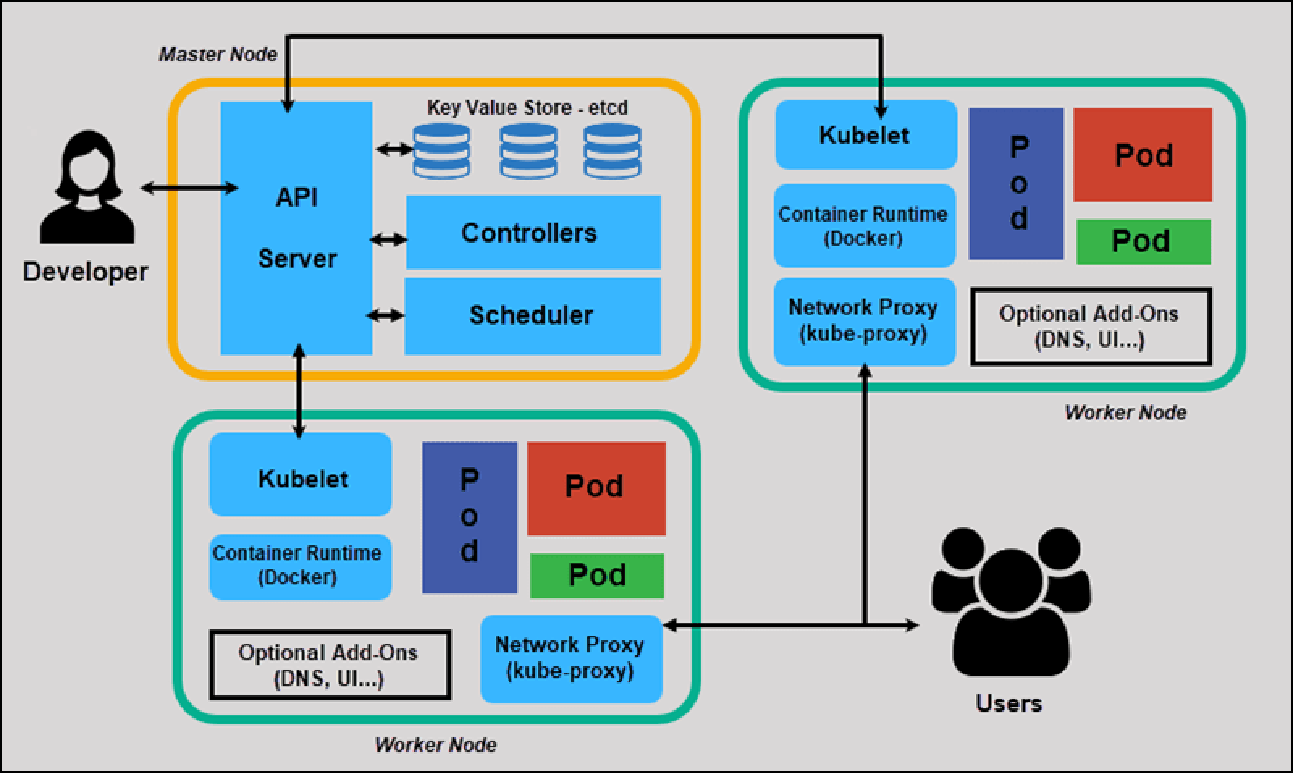
kubelet

各ノード上でノードの状態を通知して、コンテナ作成/削除やボリュームの割り当てなどを行う

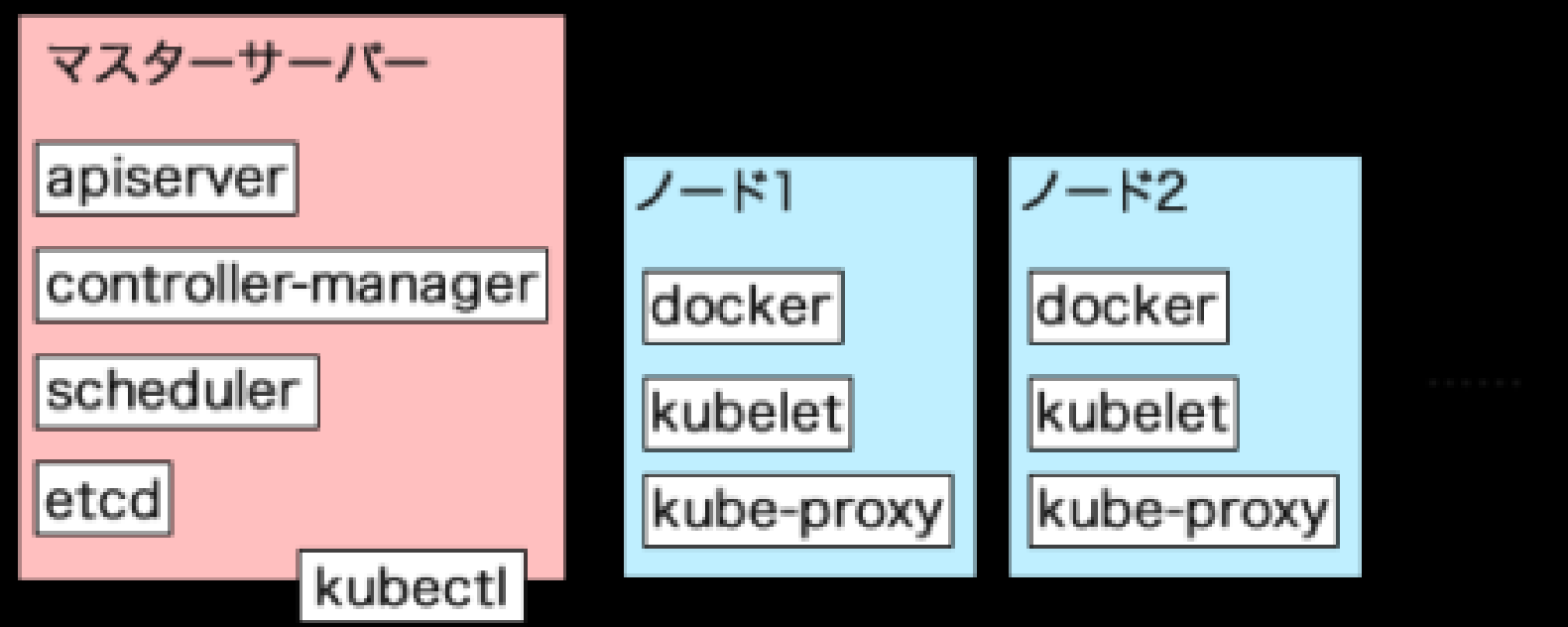
kubectl

API経由でKubenetesを操作するためのクライアントツール

ETCD:



KubernetesでMaster NodeがWorker Nodeを管理します。apiserverやcontroller-manager、scheduler、etcdはクラスタの管理を行うマスターサーバーで実行されるコンポーネントとなる。また、proxyやkubelet、dockerはコンテナを稼動させる各ノード（minionとも呼ばれる）上で実行されている必要がある。



kubectlコマンドでPodを作成するには、作成するPodの情報を記述したYAML形式ファイルを用意します。

＃事例

apiVersion: v1

kind: Pod ←Podに関する設定ファイルであることを指定

metadata: ←メタデータに関する情報を指定

name: httpd ←Podの名前を指定+

labels: ←Podに付与するラベルを指定

app: httpd

spec:

containers:

- name: httpd ←コンテナ名を指定

image: httpd ←コンテナを作成する際に使用するイメージを指定

ports:

- containerPort: 80 ←コンテナに外部からアクセスできるポートを指定

kubectl create -f ファイル名.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: rss-site

labels:

app: web

spec:

replicas: 2

selector:

matchLabels:

app: web

template:

metadata:

labels:

app: web

spec:

containers:

- name: front-end

image: nginx

ports:

- containerPort: 80

- name: rss-reader

image: nickchase/rss-php-nginx:v1

ports:

- containerPort: 88

Secret file is used for storing private information.

apiVersion: v1

kind: Secret

metadata:

name: mysecret

type: Opaque

data:

password: $(echo -n "s33msi4" | base64 -w0)

username: $(echo -n "jane" | base64 -w0)

また続く

参照：

<https://tech-lab.sios.jp/archives/18811>

<https://knowledge.sakura.ad.jp/3681/>

**CMake**

CMake is cross-platform free and open-source software for build automation, testing and packaging using a compiler-independent method. It is a meta build system that uses scripts called CMakeLists.txt to generate build files for a specific environment (for example, makefiles on Unix machines).

cmake\_minimum\_required(VERSION "3.18.2")

project("HelloWorld")

add\_executable("${PROJECT\_NAME}" "Main.cpp")

install(TARGETS "${PROJECT\_NAME}" DESTINATION bin) #${CMAKE\_INSTALL\_PREFIX}/bin

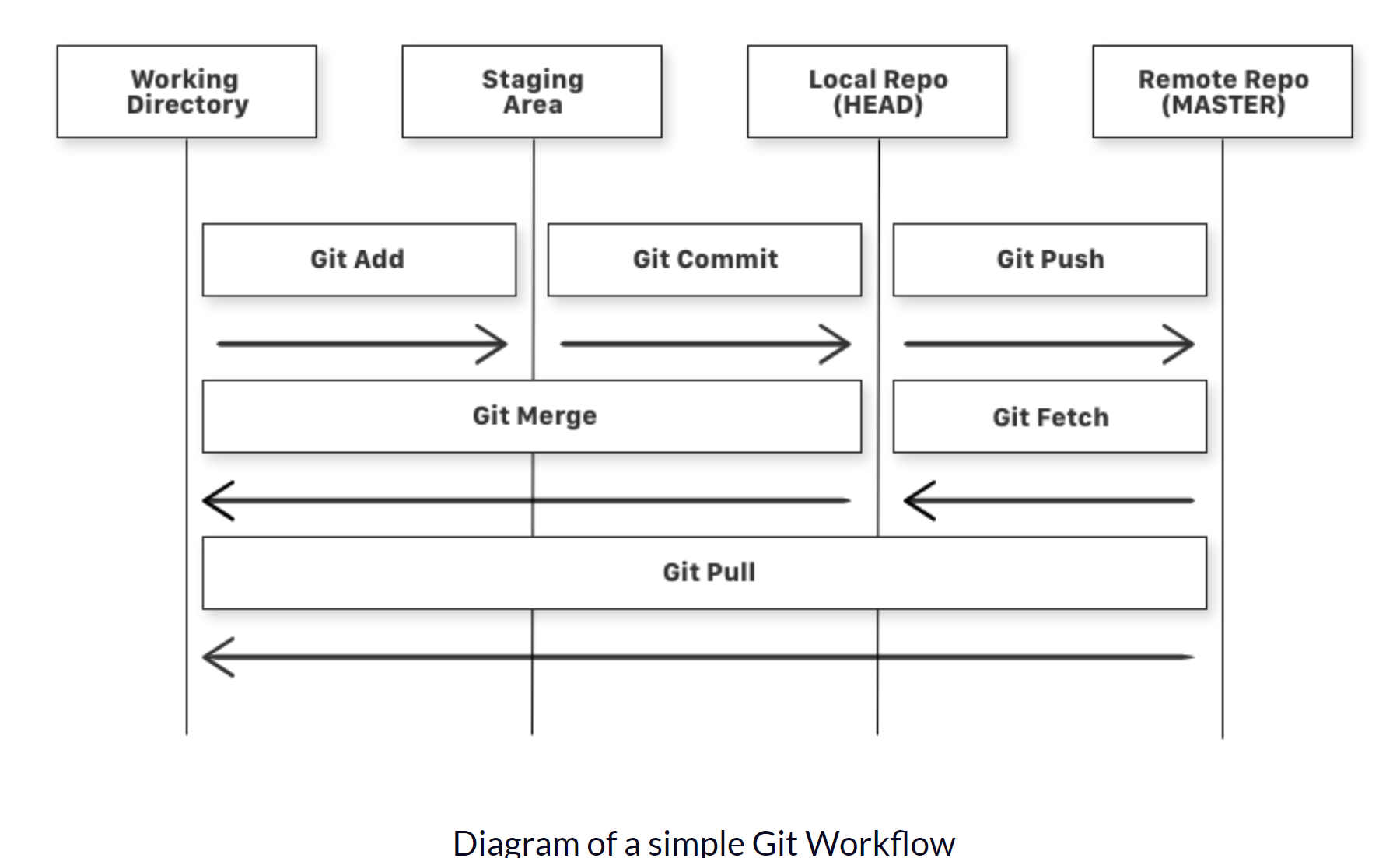
install(FILES "Main.cpp" DESTINATION src) #${CMAKE\_INSTALL\_PREFIX}/src

open CMake gui, choose build folder and VS with admin privilege, configure and generate files.

Open VS, set HelloWorld as the startup project, then select build from “INSTALL” target

Check HelloWorld folder in C://Program files(x86)

**GIT**



Check version:

$ git --version

Configure git:

$ git config --global user.name "YOUR\_USERNAME"

$ git config --global --list

Initialize git folder:

$ git init

Add contents:

$ git add .

Remove add:

$ git reset

$ git reset filename

Check status:

$ git status

Commit

$ git commit -m "message"

Remove commit:

$ git reset HEAD~1# Remove the most recent commit

Add remote origin:

$ git remote add origin remote\_repository\_URL

Change remote origin:

$ git remote set-url origin remote\_repository\_URL

Check remote:

$ git remote -v

Push commit:

$ git push -u origin master

Show changes to file:

$ git diff # To show the files changes not yet staged

Revert back to last committed version of git:

$ git checkout .

$ git checkout -- <filename>

View commit history:

$ git log

Cloning other repositories:

$ git clone remote\_repository\_URL

git fetch followed by a git merge equals a git pull

When you use git pull, Git tries to automatically do your work for you. It is context sensitive, so Git will merge any pulled commits into the branch you are currently working in. git pull automatically merges the commits without letting you review them first.

When you git fetch, Git gathers any commits from the target branch that do not exist in your current branch and stores them in your local repository. However, it does not merge them with your current branch. This is particularly useful if you need to keep your repository up to date, but are working on something that might break if you update your files. To integrate the commits into your master branch, you use git merge.

\*\*\*\*Git Flow\*\*\*\*

**SSL – Secure Sockets Layer TLS – Transport Layer Security**

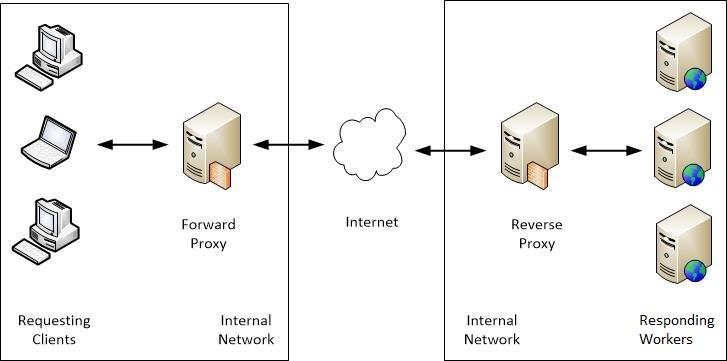
SSLサーバ証明書のお申込み準備の際、SSL/TLSを導入するサーバで「公開鍵」と「秘密鍵」を生成します。



**Forward Proxy:**

A forward proxy, often called a proxy server, is a server that sits in front of a group of client machines. When those computers make requests to sites and services on the Internet, the proxy server intercepts those requests and then communicates with web servers on behalf of those clients

This can be a firewall to provide restrictions, block certain content, and protect identity online.

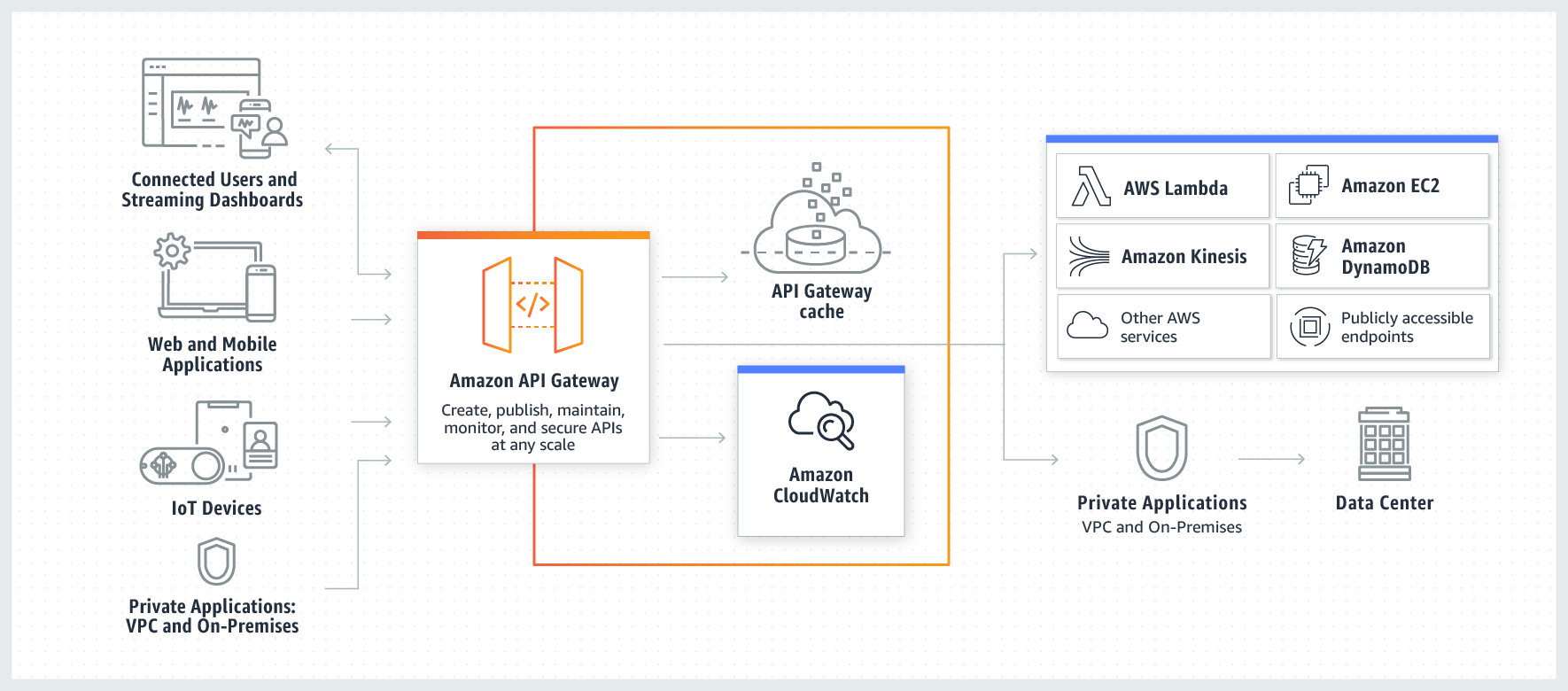


**Reverse Proxy:**

A reverse proxy is a server that sits in front of one or more web servers, intercepting requests from clients. The reverse proxy server will then send requests to and receive responses from the origin server.

This can be used as for load balancing, protection against attacks by hiding the server IP addresses, caching content for faster performance or SSL encryption.

**API Gateway** is an example of reverse proxy to accept API calls, aggregate the services and return back the result.



**VE – Vitual Environment**

Python Applicationはよく標準ライブラリ以外のパッケージやモジュールを利用します。あるプロジェクトでlibraryの特定バージョンを必要とする場合は、他のプロジェクトにも影響を及ぼしてしまうので、Applicationに使うLibraryを独立するため仮想環境を設定します。

Create Virtual Environment in Python:

python3 -m virtualenv env

source env/bin/activate

**仮想環境の作成：**

$python3 -m venv dir-name

macOSまたはLinuxの場合:

＄source env/bin/activate

windowsの場合：

＄.\env\Scripts\activate

実際の環境に取り戻したい場合は

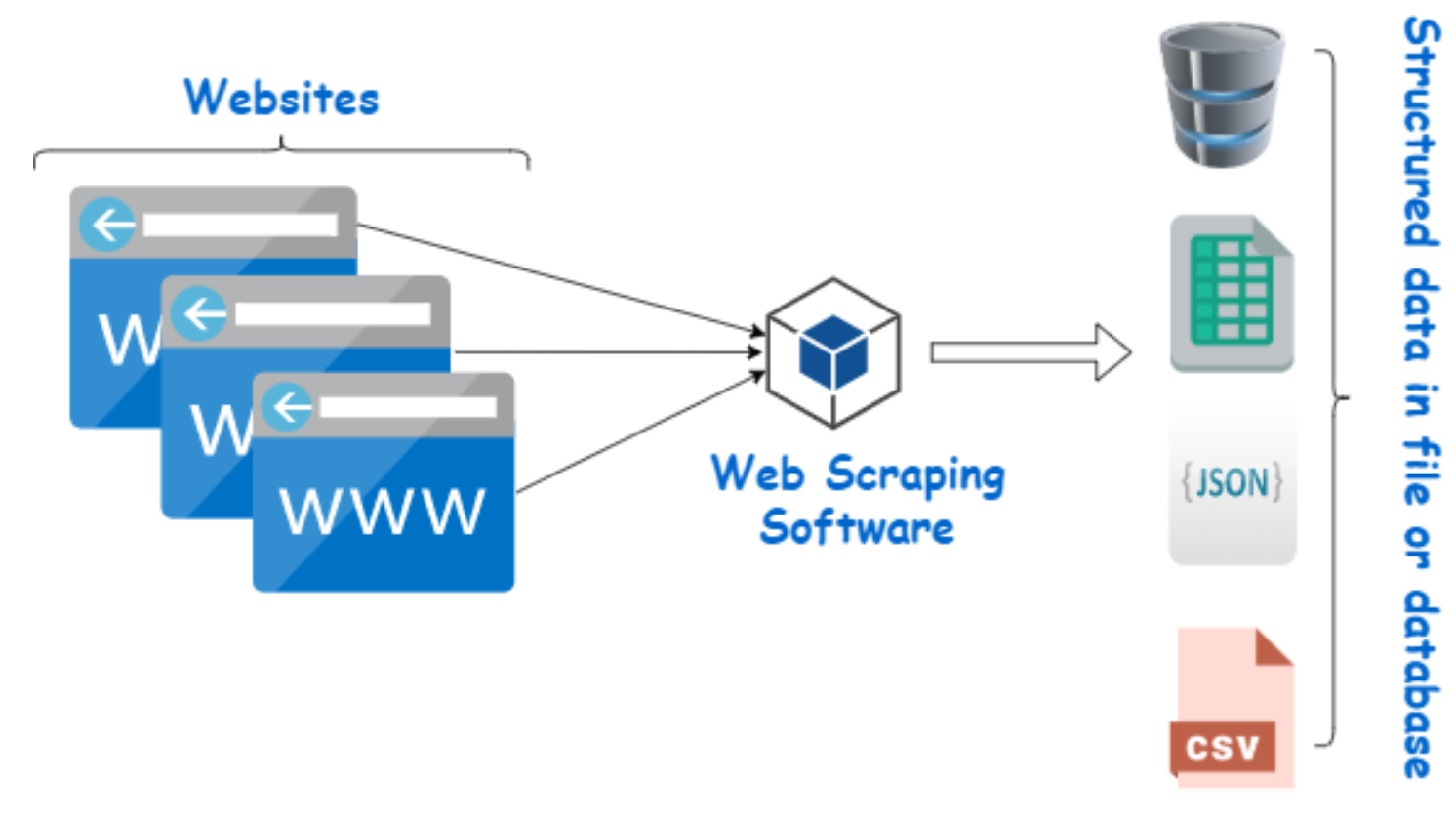
＄deactivate

参照：

<https://docs.python.org/ja/3/tutorial/venv.html>

**Web Scraping**

Web Scraping (also termed Screen Scraping, Web Data Extraction, Web Harvesting etc.) is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet) format.



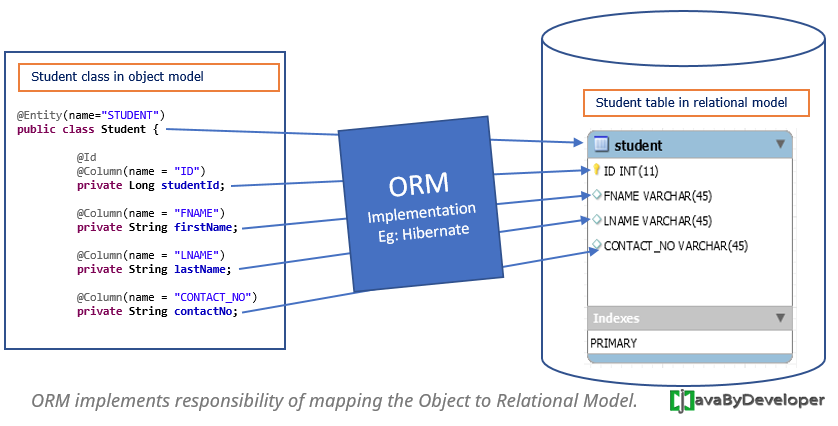
It is done in Python using BeautifulSoup library.

pip install beautifulsoup4

From bs4 import BeautifulSoup

**Flask** – web development library in python, similar to Django.

**ORM – Object Relational Mapping**, used for accessing databases from Objects and classes.



**ACID Transaction:**

A **transaction** symbolizes a unit of work performed within a database. It is often composed of multiple operations.

**ACID (Atomicity, Consistency, Isolation, Durability)** is a set of properties of database transactions intended to guarantee validity even in the event of system crashes, power failures, and other errors.

Atomic – All operations in a transaction are treated as a single unit. It either succeeds or fails completely.

Consistent - Ensures that a transaction can only bring the database from one valid state to another by preventing data corruption.

Isolation - Determines how and when changes made by one transaction become visible to the other.

Durable - Ensures that the results of the transaction are permanently stored in the system. The modifications must persist even in case of power loss or system failures.

**CAP Theorem:**

The CAP theorem says that a distributed system can deliver only two of three desired characteristics: ***consistency****,****availability****,*and***partition tolerance*** (the ‘**C**,’ ‘**A**’ and ‘**P**’ in CAP).

**Consistency**: Consistency means that all clients see the same data at the same time, no matter which node they connect to.

**Availability**: Availability means that that any client making a request for data gets a response, even if one or more nodes are down.

**Partition tolerance**: A partition is a communications break within a distributed system—a lost or temporarily delayed connection between two nodes. Partition tolerance means that the cluster must continue to work despite any number of communication breakdowns between nodes in the system.

**MongoDB** is a CP data store—It has a master node, it resolves network partitions by maintaining consistency, while compromising on availability.

**Apache Cassandra** is an AP database— Cassandra is also a NoSQL Database that has a masterless architecture, and as a result, it has multiple points of failure, rather than a single one. it delivers availability and partition tolerance but can't deliver consistency all the time.

[**NoSQL (non-relational) databases**](https://www.ibm.com/cloud/learn/nosql-databases) are ideal for distributed network applications. NoSQL databases are horizontally scalable and distributed by design—they can rapidly scale across a growing network consisting of multiple interconnected nodes. SQL Databases are vertically scalable.

**Database Architecture:**

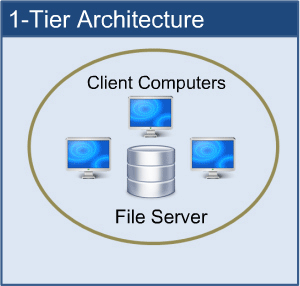
Database architecture uses programming languages to design a particular type of software for businesses or organizations.

The architecture of database has different tiers,

* 1-tier architecture
* 2-tier architecture
* 3-tier architecture
* n-tier architecture

**1-tier architecture:**

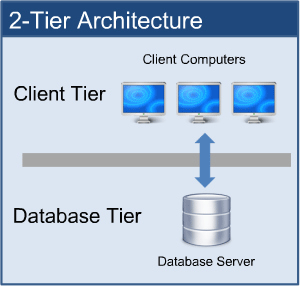
One-tier architecture involves putting all of the required components for a software application or technology on a single server or platform.



1-tier architecture keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place.

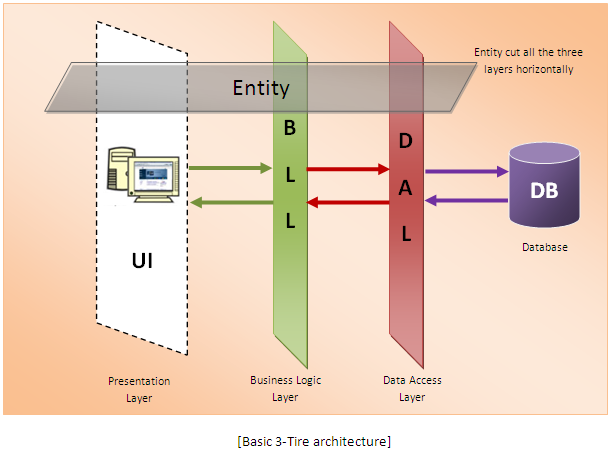
**2-tier architecture:**

The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server.



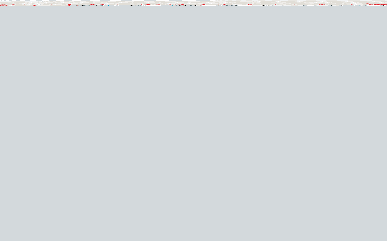
**3-tier architecture:**

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



**N-tier architecture:**

N-tier architecture would involve dividing an application into three different tiers. These would be the logic tier, the presentation tier, and the data tier.

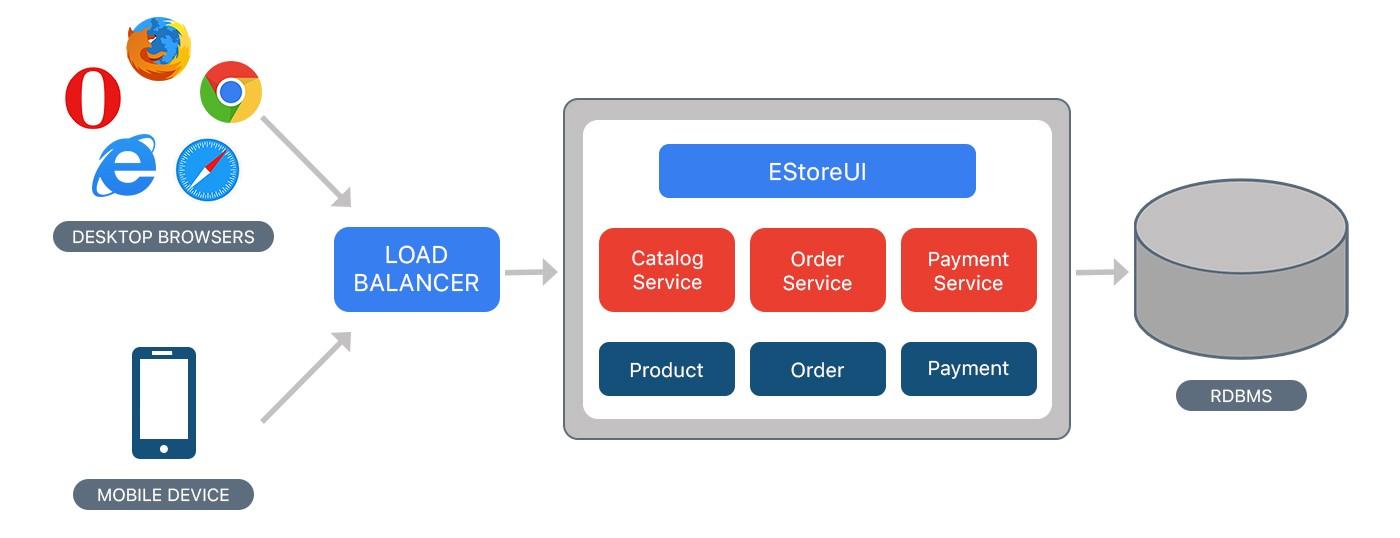


It is the physical separation of the different parts of the application as opposed to the usually conceptual or logical separation of the elements in the model-view-controller (MVC) framework.

*# Service Architecture*

**Monolith Architecture :**

Monolithic application has single code base with multiple modules. Modules are divided as either for business features or technical features. It has single build system which build entire application and/or dependency.



Despite having different components/modules/services, the application is built and deployed as one Application for all platforms (i.e. desktop, mobile and tablet) using RDBMS as a data source.

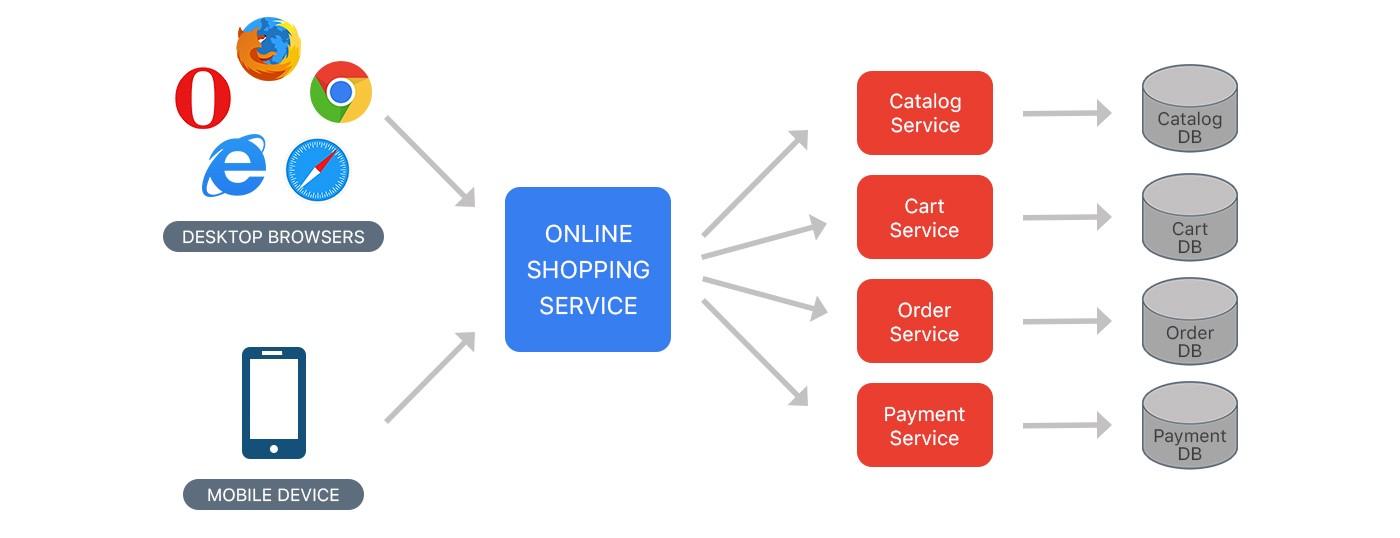
Drawbacks:

* Maintenance — If Application is too large and complex to understand entirely, it is challenging to make changes fast and correctly.
* The size of the application can slow down the start-up time.
* You must redeploy the entire application on each update.
* Monolithic applications can also be challenging to scale when different modules have conflicting resource requirements.

**Microservices:**

Microservices are an approach to application development in which a large application is built as a suite of **modular services** (loosely coupled modules/components). Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other sets of services.

Instead of sharing a single database as in Monolithic application, each microservice has its own database. Having a database per service is essential if you want to benefit from microservices, because it ensures **loose coupling**.



The application starts faster, which makes developers more productive, and speeds up deployments. Microservices Enables the continuous delivery and deployment of large, complex applications.

**SKU (Stock Keeping Unit)** is a product code that you can use to search and identify stock on hand from lists, invoices, or order forms. It is a term that is typically used when talking about inventory management.

**POE – Power over Ethernet**

Power over Ethernet (PoE) is a means of carrying electrical power through data cables. With PoE enabled devices, electrical current passes through the ethernet cable along with the data normally carried by an ethernet cable. So there is no need for the additional power cable.

**Redis:**

It is an in-memory key-value database, it has a cluster so horizontal scaling is possible.

Datatypes: String, Lists, Sets, Sorted sets, Hashes, Bitmaps, Hyperlogs, Geojson

Features: No schemas, very fast (110,000 sets/sec 81,000 gets/sec), caching & disk persistence

$SET username1 “barath”

$ GET username1

# can increment numbers

$ SET total 10

$ INCR total # 11

$ EXISTS username1 # No: 0 Yes: 1

$ GET username2 # No: None/nil

# Delete everything

$ FLUSHALL

# set val to a dict

$ SET server:name localhost

$ SET server:port 5000

$ GET server:name

# expire key

$ EXPIRE username1 60 # seconds

$ TTL username1 # returns the time to expire

$ SETEX username1 30 “barath” # sets expiry time 30 and value

$ PERSIST username1 # will not expire, TTL username => -1

# set multiple key values

$ MSET username1 “Barath” username2 “Wally”

# append value to existing key

$ APPEND username1 “ Kumar” # username1 -> “Barath Kumar”

$ RENAME username1 username0 # username1 -> None, username0 -> “Barath Kumar”

Redis lists are sorted by insertion order, we can add new elements by pushing values to existing list

$ LPUSH mylist 1

$ LPUSH mylist 2

$ RPUSH mylist 2

$ LRANGE mylist # 2, 1, 3

$ LLEN mylist # 3

$ LPOP mylist # 2, 1

$ LINSERT mylist BEFORE 1 3 # 2, 3, 1

# sets

$ SADD myset “a”

$ SADD myset “b”

$ SMEMBERS myset # “a”, “b”

$ SISMEMBER myset “c” # 0 no->0 yes->1

$ SCARD myset # 2 number of elements

$ SMOVE myset myset2 “a” # moves the value to another set

$ SREM myset “a” # removes the value from set

# sorted sets, they are sorted by key

$ ZADD users 2000 “A”

$ ZADD users 2005 “B”

$ ZADD users 1990 “C”

$ ZRANK users “B” # 2

$ ZRANK users “C” # 0

$ ZRANGE users 0 -1 # “C”, “A”, “B”

$ ZINCRBY users 20 “C” # 2010 C

# hashes

$ HSET myhash a 1

$ HSET user:a name “a”

$ HSET user:a rank 1

$ HGET user:a name # “a”

$ HGETALL user # returns keys and values

$ HVALS user

# set multiple values

$ HMSET user:b name “b” age “20”

$ HINCRBY user:b age 10

$ HDEL user:b

$ HLEN user:b

# getting overlapping items in two sets

$ sinter var1 var2

# Database persistance

RDB: Redis Database Backup # a dump of all user data stored in an internal, compressed serialization format at a particular timestamp, a single compact file used for point-in-time recovery

$ SAVE 60 1000 # save every 6 seconds if atleast 1000 keys are changed

$ SAVE # creates a dump file, it gets stores in /usr/local/etc/redis/redis.conf

AOF: Append Only File # a persistence technique in which an RDB file is generated once and all the data is appended to it as it comes

$ appendonly yes # AOF file gets stored in /etc/redis/redis.conf

When Redis needs to dump, it uses a fork to create parent-child process, the child writes the data into a temporary file and replaces it with the old dump file once it finishes.

**AJAX**

AJAX – Asynchronous JavaScript and XML

Used for sending and retrieving data in the background without refreshing the webpage.

Uses JS XMLHttpRequest Object

var xhr = new XMLHttpRequest();

Streaming data with ajax:

<script type=text/javascript>

function fetchdata(){

$.ajax({

url: '/livedata',

type: 'post',

success: function(data){

// Perform operation on the return value

// var livegraph = {{ data | safe }};

if (data && data != "none"){

var data = JSON.parse(data);

Plotly.newPlot('live\_dashboard', data);

}

},

complete:function(data){

//document.getElementById("setval").innerHTML = 30;

setTimeout(fetchdata,5000);

}

});

}

$(document).ready(function(){

setTimeout(fetchdata,5000);

});

</script>

**PHP-FPM:**

PHP-FPM stands for “PHP-FastCGI process manager”. CGI refers to the common gateway interface which is scripted to work as an interface between the web server and dynamic content serving programs. It listens on a port much like the web server itself does, and passes the request between the PHP and web server.

XSS: cross site scripting

Javascript code gets executed in php echo command, to prevent execution html2chars function is used.

Session Object is generated to store login information, and token is used for security and CSRF prevention.

**CSRF Token:**

A CSRF token is a unique, secret, unpredictable value that is generated by the server-side application and transmitted to the client in such a way that it is included in a subsequent HTTP request made by the client. When the later request is made, the server-side application validates that the request includes the expected token and rejects the request if the token is missing or invalid.

CSRF tokens should be treated as secrets and handled in a secure manner throughout their lifecycle. An approach that is normally effective is to transmit the token to the client within a hidden field of an HTML form that is submitted using the POST method. The token will then be included as a request parameter when the form is submitted:

<input type="hidden" name="csrf-token" value="CIwNZNlR4XbisJF39I8yWnWX9wX4WFoz" />

For additional safety, the field containing the CSRF token should be placed as early as possible within the HTML document, ideally before any non-hidden input fields and before any locations where user-controllable data is embedded within the HTML. This mitigates against various techniques in which an attacker can use crafted data to manipulate the HTML document and capture parts of its contents.

httpd(HTTP Daemon): httpdcommonly refers to the Apache HTTP server, it can also refer to other web servers.

**OAuth2:**

Authorization code flow:

Client sends request to authorization server (front channel)

It contains a redirect URI, response code and scope of the request.

The Authorization server requests consent from the resource owner of the given scope. (front channel)

The resource owner accepts the request and sends an authorization code to the redirect URI of the client. (front channel)

The redirect URI sends the authorization code to the Authorization server via back channel and gets the access token, this cannot be intercepted through the browser.

The redirect URI then talks to the resource server through access token. (back channel)

The authorization server can be run by Google, Okta, or Facebook etc.

Front channel is used for communicating with the user. Back channel takes care of the security.

The client has to register itself to the authorization server beforhand to proccess the authorization. The client gets a clientID and client secret, which is used during the token exchange.

Resource owner password credentials flow (back channel only) for older applications

Client credentials (back channel only)

OAuth2 was not designed for authentication, it was primarily meant for authorization (permissions and scopes). Using OAuth for authentication is not recommended because there is no standard way of getting the user's information.

**\*\*\*OpenID Connect** was built as a standard for authentication, which is built on top of OAuth2.0 and has additional features.

OpenID Connect has in-addition,

ID token,

UserInfo endpoint for getting more user information.

So, the client gets an ID token and user info from the Authorization server in exchange for the authorization code via back-channel.

ID-token is a JWT(Json Web Token), it can be decoded using jsonwebtoken.io.

OpenID Connect can also be used for Mobile app login (codeflow + PKCE (Proof Key for Code Exchange)),  Simple Sign On(SSO).

PKCE (RFC 7636) is an extension to the Authorization Code flow to prevent several attacks and to be able to securely perform the OAuth exchange from public clients. It was originally designed to protect mobile apps, but its ability to prevent authorization code injection makes it useful for every OAuth client, even web apps that use a client secret.

Authorization server can also use SAML under the hood for older systems.

Once the user is authenticated, they can access the underlying backend service. The backend service doesn't need to know about how the user is authenticated, it just need the identity information in form of JWT, as a header in HTTP request.

Reference: <https://www.youtube.com/watch?v=996OiexHze0>

**\*\*\*XQuery:**

XQuery (XML Query) is a query and functional programming language that queries and transforms collections of structured and unstructured data, usually in the form of XML, text and with vendor-specific extensions for other data formats (JSON, binary, etc.).

IDP – Identity Provider SP – service provider

SAML – Security Assertion Markup Language

LDAP – Lightweight Directory Access Protocol – used for maintaining directory info services over IP. Commonly used for storing passwords, username etc.

ダイアグラム

自動的に生成された説明

**SSO – Single Sign On Agent** It authenticates users against active directory.

**\*\*\*Security Assertion Markup Language (SAML):**

SAML is a standard for logging users into applications based on their sessions in another context.  This single sign-on (SSO) login standard has significant advantages over logging in using a username/password.

Most organizations already know the identity of users because they are logged in to their Active Directory domain or intranet. It makes sense to use this information to log users in to other applications, such as web-based applications, and one of the more elegant ways of doing this is by using SAML.

SAML SSO works by transferring the user’s identity from one place (the identity provider) to another (the service provider). This is done through an exchange of digitally signed XML documents.

Consider the following scenario: A user is logged into a system that acts as an identity provider. The user wants to log in to a remote application, such as a support or accounting application (the service provider).

The following happens:

1. The user accesses the remote application using a link on an intranet and the application loads

2. The application identifies the user's origin by subdomain, IP address etc., and redirects the user back to the identity provider for authentication. This is called authentication request.

3. The user either has an existing active browser session with identity provider, or logs into the identity provider

4. The identity provider creates an authentication response in the form of an XML-document containing the username and email address of the user, signs it using an x.509 certificate and posts the information to the service provider.

5. The service provider verifies the identity provider using the certificate fingerprint already stored in the system.

6. The identity of the user is establishedand the user is granted the app access.

OneLogin, ActiveDirectory are examples of Identity providers.

Reference: <https://developers.onelogin.com/saml>

**\*\*\*JSON Web Tokens (JWT):**

JWT is used for authorization (making sure that the user sending the request is the same user that logged in) not authentication (to verify user’s username and password during login).

Authorization is done using session ID. Instead of using sessions and cookies, we can use JWT. When use use sessions, the session is stored in server memory and the session ID is sent as a Cookie to the client browser. Then everytime user sents a request browser also sends the session ID and it gets authorized in server side.

Using JWT, server creates a JWT for user with secret key. It doesn’t store any key explicitly in the memory, and it sends the JWT back to the browser. When the user sends a request, they send the JWT along with the request. The JWT has all the information about the user, so the server can validate the JWT using its signed secret key. So, this eliminates the need to store the user’s session details in the server side.

If the JWT gets tampered inbetween the process, then it becomes invalid and the server can invalidate it.

JWT has an encoded and decoded information. Encoded version is a random group of strings which is sent to the client. The decoded version has a header, payload and verify signature. Header determines the algorithm to encode message, Payload is the information stored in the token, and the signature is used for verifying if the token is valid or invalid.

In the encoded string, header is always present at the beginning before the first period. It is base64 encoded. The data or payload comes inbetween the two periods, it contains a subject/userID, some information about the user, token issued date, expiry time etc.

Signature takes the header and payload, base64 encodes it along with a secret key, and then uses an hash algorithm like HS256 to encode the message.

secret key is also base64 encoded.

Eg. of encoded JWT:

eyASenfwoSIOrngOS.EIFgeeSknOAfosdfiESNJ.soiHAfESUfgfdHAves

Eg. of decoded JWT:

Header:

{

‘alg’ : ‘HS256’,

‘typ’:’JWT’

}

Payload:

{

‘sub’:123456789, # userid

‘name’:’Jonny Depp’

‘iat’: 169234223 # issued at

}

Signature:

HMSCSHA256{

base64UrlEncode(header) + ‘.’+

base64UrlEncode(payload).

secrey key

} secret base64 encoded

When there are multiple servers in different applications or miroservices which are integrated together, sessions cannot be used, so users would have to login again when switching different pages, so using JWT eliminates the need for servers to store user’s session information.

**\*\*\*Bleeding edge technology:**

Bleeding edge technology is a category of technologies so new that they could have a high risk of being unreliable and lead adopters to incur greater expense in order to make use of them.

charateristics:

Lack of consensus

Lack of testing

Industry resistance to change

**\*\*\*Mantis Bug Tracker** - An open source issue tracker and web-based bug tracking system to build realtime processing applications.

The most common use of MantisBT is to track software defects. However, MantisBT is often configured by users to serve as a more generic issue tracking system and project management tool.

**\*\*\*Time to live(TTL):**

Time to live or hop limit is a mechanism which limits the lifespan or lifetime of data in a computer or network. TTL may be implemented as a counter or timestamp attached to or embedded in the data. Once the prescribed event count or timespan has elapsed, data is discarded or revalidated by a router.

DNS TTL (time to live) is a setting that tells the DNS resolver how long to cache a query before requesting a new one.

**\*\*\*TCP+TLS handshake**:

Takes about 300-400 ms

Wireless connection is lossy

To solve this Point of Presence(PoP) is used.

PoP is an artificial demarcation point or interface point between communication entities.

The routers, switches, servers, and other devices necessary for traffic to cross over networks are all present at PoPs. Internet service providers and edge networks like StackPath typically have multiple points of presence located near large Internet exchange points (IXPs) at which they have peering agreements. The proximity of points of presence and Internet exchange points is one very important factor in how quickly traffic is able to traverse the Internet.

An **Internet exchange point** (IXP) is the physical infrastructure through which Internet service providers (ISPs) and content delivery networks (CDNs) exchange Internet traffic among their networks (autonomous systems) and peer together.

Reference: <https://blog.stackpath.com/point-of-presence>

**\*\*\*CDN**:

A content delivery network, or content distribution network (CDN), is a geographically distributed network of proxy servers and their data centers. The goal is to provide high availability and performance by distributing the service spatially relative to end users.

CDNs are a layer in the internet ecosystem. Content owners such as media companies and e-commerce vendors pay CDN operators to deliver their content to their end users. In turn, a CDN pays Internet service providers (ISPs), carriers, and network operators for hosting its servers in their data centers.

CDN is an umbrella term spanning different types of content delivery services: video streaming, software downloads, web and mobile content acceleration, licensed/managed CDN, transparent caching, and services to measure CDN performance, load balancing, Multi CDN switching and analytics and cloud intelligence.

**\*\*\*Great System Architecture**:

Scales Development Teams

Delivers Quality

Enables High Performance/Low Cost

Supports Future Features Naturally

API resource is modelled using a json type structure. It contains a resource name, fields, and the specific information of the input.

API definitions should not be in microsevice repos, they should be in a dedicated API service git repo with CI.

APIs dhould be created through a specification, which makes it uniform, consistent, readable and less error-prone. Code generation can then be used to create client libraries in multiple languages with less effort.

Each microservice owns its own database. No other service is allowed to connect to that database, they interact through API or service interface.

Define storage requirements in metadata.

Continuous Delivery is a prerequisite to managing microservice architectures.

Deploy is triggered by a git tag, and tags get creted automatically by a change on master. Auto Deploy on new commit on master.

Document microservice in yaml or json in a simple way.

Perform standard health checks.

**Events Approach:**

Internally services can use events to process asyncronously, events are described in well defined schema, it is recommended to use binary formats like gRPC. Binary formats are compatible with codde generation to produce and consume events. Create a journal of all operations on table, record operation(insert, update, delete), on creation of data, queue the journal to be published, all asyncronously. The events are then consumed and processed in micro-batches. They record failures locally.

API Builder can be used to build events. 1 model/event.

**\*\*\*Subroutine:**

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), a subroutine is a sequence of [program instructions](https://en.wikipedia.org/wiki/Instruction_(computer_science)) that performs a specific task, packaged as a unit. This unit can then be used in programs wherever that particular [task](https://en.wikipedia.org/wiki/Task_(computing)) should be performed.

Subroutines may be defined within programs, or separately in [libraries](https://en.wikipedia.org/wiki/Library_(computer_science)) that can be used by many programs. In different programming languages, a subroutine may be called a routine, subprogram, function, [method](https://en.wikipedia.org/wiki/Method_(computing)), or procedure.

**\*\*\*Remote Procedure Calls (RPC):**

In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network).

Remote Procedure Call (RPC) is a [protocol](https://searchnetworking.techtarget.com/definition/protocol) that one program can use to request a service from a program located in another computer on a [network](https://searchnetworking.techtarget.com/definition/network) without having to understand the network's details. RPC is used to call other processes on the remote systems like a local system. A procedure call is also sometimes known as a *function call* or a *subroutine call*.

RPCs are a form of [inter-process communication](https://en.wikipedia.org/wiki/Inter-process_communication) (IPC), in that different processes have different address spaces: if on the same host machine, they have distinct virtual address spaces.

RPC uses the [client-server](https://searchnetworking.techtarget.com/definition/client-server) model. The requesting program is a client, and the service-providing program is the [server](https://whatis.techtarget.com/definition/server). Like a regular or local procedure call, an RPC is a [synchronous](https://whatis.techtarget.com/definition/synchronous) operation requiring the requesting program to be suspended until the results of the remote procedure are returned. However, the use of lightweight processes or [threads](https://whatis.techtarget.com/definition/thread) that share the same address space enables multiple RPCs to be performed concurrently.

**\*\*\*gRPC (gRPC Remote Procedure Calls)** is an open source remote procedure call (RPC) framework initially developed at Google in 2015, that can run in any environment.

gRPC is roughly 7 times faster than REST when receiving data & roughly 10 times faster than REST when sending data for this specific payload.

It uses HTTP/2 for transport, Protocol Buffers as the interface description language, and provides features such as authentication, bidirectional streaming and flow control etc.

**\*\*\*Idempotent:**

From a RESTful service standpoint, for an operation (or service call) to be idempotent, clients can make that same call repeatedly while producing the same result. In other words, making multiple identical requests has the same effect as making a single request.

**\*\*\*Lint or Linter:**

Lint, or a linter, is a static code analysis tool used to flag programming errors, bugs, stylistic errors and suspicious constructs. The term originates from a Unix utility that examined C language source code.

**\*\*\*API Builder** - An open source tool that lets you build and deploy a project that is comprised of API endpoints that can be consumed by any client application. An API Builder project is a Node.js application. It is a framework to create endpoints that render HTML for client applications.

You can design your API directly in Postman using the API Builder.

**\*\*\*Code generation:**

In computing, code generation is the process by which a compiler's code generator converts some intermediate representation of source code into a form (e.g., machine code) that can be readily executed by a machine.

**\*\*\*NVIDIA® TensorRT™** is an SDK for high-performance deep learning inference. TensorRT provides INT8 and FP16 optimizations for production deployments of deep learning inference applications such as video streaming, speech recognition, recommendation, fraud detection, and natural language processing.

**\*\*\*Lua:**

Lua ia a free open-source software that is lightweight and embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description.

Lua has been used in [many industrial applications](https://en.wikipedia.org/wiki/Category:Lua_(programming_language)-scriptable_software) (e.g., [Adobe's Photoshop Lightroom](http://since1968.com/article/190/mark-hamburg-interview-adobe-photoshop-lightroom-part-2-of-2)), with an emphasis on embedded systems (e.g., the [Ginga](http://www.ginga.org.br/) middleware for digital TV in Brazil) and [games](https://en.wikipedia.org/wiki/Category:Lua_(programming_language)-scripted_video_games) (e.g., [World of Warcraft](http://www.wowwiki.com/Lua) and Angry Birds). Lua is currently [the leading scripting language in games](http://www.satori.org/2009/03/the-engine-survey-general-results/).

Lua is fast and portable. It is native to Redis, and executes sequentially.

tar file,

gzip

**High-throughput computing (HTC) and High Performance Computing (HPC):**

It is a computer science term to describe the use of many **computing** resources over long periods of time to accomplish a **computational** task. The key to HTC is to efficiently harness the use of all available resources.

In contrast, High Performance Computing (HPC) environments deliver a tremendous amount of compute power over a short period of time. HPC environments are often measured in terms of Floating point Operations Per Second (FLOPS).

**Clickstream Analysis:**

It is the process of collecting, analyzing and reporting aggregate data about which pages a website visitor visits -- and in what order. The path the visitor takes though a website is called the clickstream.

There are two levels of clickstream analysis, traffic analytics and e-commerce analtyics. Traffic analytics operates at the server level and tracks how many pages are served to the user, how long it takes each page to load, how often the user hits the browser's back or stop button and how much data is transmitted before the user moves on. E-commerce-based analysis uses clickstream data to determine the effectiveness of the site as a channel-to-market. It's concerned with what pages the shopper lingers on, what the shopper puts in or takes out of a shopping cart, what items the shopper purchases, whether or not the shopper belongs to a [loyalty program](https://searchcustomerexperience.techtarget.com/definition/loyalty-card-program) and uses a coupon code and the shopper's preferred method of payment.

They rely on big-data analytics like Hadoop to generate reports.

**CORS (Cross-Origin Resource Sharing):**

**Cross-Origin Resource Sharing** ([CORS](https://developer.mozilla.org/en-US/docs/Glossary/CORS)) is an [HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP)-header based mechanism that allows a server to indicate any other [origin](https://developer.mozilla.org/en-US/docs/Glossary/Origin)s (domain, scheme, or port) than its own from which a browser should permit loading of resources. CORS also relies on a mechanism by which browsers make a “preflight” request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

An example of a cross-origin request: the front-end JavaScript code served from https://domain-a.com uses [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) to make a request for https://domain-b.com/data.json.

For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, XMLHttpRequest and the [Fetch API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) follow the [same-origin policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy). This means that a web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the right CORS headers.

**Portable Operating System Interface (POSIX):**

The **Portable Operating System Interface** (**POSIX**) is a family of standards specified by the IEEE computer society for maintaining compatibility between operating systems.[[1]](https://en.wikipedia.org/wiki/POSIX#cite_note-1) POSIX defines the both system- and user-level APIs, along with command line shells and utility interfaces, for software compatibility (portability) with variants of Unix and other operating systems. POSIX is intended to be used by both application and system developers.

[Cygwin](https://en.wikipedia.org/wiki/Cygwin) provides a largely POSIX-compliant development and run-time environment for Windows.

**Hashing:**

Searching a record from a table usually takes a linear time. If the records are sorted then search takes O(log(n)) time, but insertion and deletion becomes costly.

Using balanced binary search trees (BST), the insertion, deletion and search takes O(log(n)) time. Using Direct Access Table, we can perform those operations in O(1) time but they require a lot of memory.

Hashing provides O(1) time on average for insert, delete and search operations.

Hash fn. converts a big number into smaller number that is used as an index in hash table, so it maps data of an arbitrary size to a data for fixed size.

Good Hash Function: It should be efficiently computable. It should uniformly distribute the keys.

A collision occurs when 2 different numbers are mapped to the same value. Collision can be handled using some techniques,

1. Chaining is a method to store a linked list of keys at each cell of the hash table. Each key points to another key which has the same hash value. It is easy to implement. It consumes a lot of memory and is not very efficient.

Expected time of operations would be O(1+a), where a is the load factor.

a = (no. of keys stored)/(no. of slots in table), a<1

1. Open addressing - All elements are stored within the hash table. It is done using linear probing, quadratic probing and double hashing. Size of the table would be greater than the no. of keys. Hash function specifies order of slots to probe instead of a single slot. It takes O(1/(1-a)) time, where a is load factor.
   1. **Linear Probing** - It uses a iteration sequence i to compute another hash value,

hi(x) = (hash(x) + i) % hashtable\_size, if there is another key that is stored in the same slot then we increment the i value. It the slot is empty then insert the key there.

Each slot has 3 states, Empty, Deleted, Occupied. If a key is deleted then that slot is marked as deleted, so the searching operation does not stop there. It stops when it hits an empty slot.

It has the best caching performance, but suffers from clustering.

* 1. **Quadratic Probing** - It is similar to Linear probing but the increments are done quadratically.

hi(x) = (hash(x) + i^2) % hashtable\_size

It has average caching performance, but has less clustering.

* 1. **Double Hashing** - A second hash fn. is used to calculate the hash value.

hi(x) = (hash(x) + i\*hash2(x)) % hashtable\_size

It has poor caching performance and more computations but no clustering.

**Cryptographic Hash Functions:**

They are deterministic (the output value is same for a given input), fast, irreversible (can’t be decrypted from the hash value), utilize the ‘avalanche effect’ (even if the input slightly different, the output is completely different), collision resistant (odds of having two values with same hash value is probabilistically negligible).

For a given input and hash value It is not easy to find another input which will result in the same hash value. It is unbreakable without using brute force.

Use-cases:

Verifying file and message integrity, if the file is modified then the hash value changes. Git uses it to track changes.

Verifying passwords. Passwords can be stored in the database by encryption with an encryption key, but the encryption key can still be stolen and the password can be decrypted back. We can use the hash functions to store passwords and they can’t be decrypted back.

Hashing can be compromised with brute force attack, and since a given input has a deterministic output, cracking the hash value of one account might lead to hacking multiple accounts. So, salt is added to the password before hashing which results in a completely different hash value.

To reduce brute force attack some techniques like, bcrypt, script and argon2 can be used. They take password, salt and a cost as the input, the cost parameter defines the number of rounds the algorithm goes through and reduces the brute force attacks.

We can combine multiple techniques to hash stronger, such as computing a simple hash, and using bcrypt it with a salt and cost, and further use AES (Advanced Encryption Standard).

**SHA1:**

It was commonly used in 2000s but it is considered no longer secure. It takes a text of 40 digit hexadecimal value.

First, we take the input text and split it into a char array of ASCII values. We convert the ascii values to binary and pad zeros to the left to make it 8 bit (1 byte) long. We join them together and append 1 to it on the right. We pad the message with 0’s to the right until its length is 448 mod 512.

We take the total char count in the char array (length of the char array \* 8) and convert it to binary. Pad it with zeros to the left to make it 64 characters long. Put it at the end of the 448 mod 512 characters. The resulting length is divisible by 512.

Break the message in to chunks of 512, and break each chunk into subarray of length 16 containing 32 bit messages.

Do bitwise operations like XORs on these subarrays and append it to the original chunk to make the chunk size to 80. Loop through all the chunks and do the same operation.

Initialize some variables (about 5) of 32 bit length. Continuously update the variables by doing bitwise operations with the chunks and looping through all the chunks.

Convert the variables into hexadecimal and concatenate them to get the hash value. It has a uniform length regardless of the input size.

**HMAC**  Hashing Message Authentication Code

**Dynamic Website:**

Dynamic website displays different content each time it is viewed. It can change with time of the day, user and the type of interaction. There are two types of dynamic webpages, client-side and server-side scripting.

Client-Side scripting:

The webpage changes that occur due to an action in the webpage, such as a mouse-click or keyboard entry that uses client-side scripting like JS.

Server side scripting:

The webpage changes that occur when a web page is reloaded or loaded from server or use server-side scripting. This can be login page, forums, shopping carts etc.

**Malware and Bug:**

It stands for malicious software, is intentionally designed to attack and be harmful to computer systems. In contrast, a software which unintentionally causes harm to computer systems is known as Software bug.

Malwares can be viruses, trojan horses, randsomeware, spyware, adware etc.

**Data Corruption:**

**Data corruption** refers to errors in [computer data](https://en.wikipedia.org/wiki/Computer_data) that occur during writing, reading, storage, transmission, or processing, which introduce unintended changes to the original data. Computer, transmission, and storage systems use a number of measures to provide end-to-end [data integrity](https://en.wikipedia.org/wiki/Data_integrity), or lack of errors.

In general, when data corruption occurs, a [file](https://en.wikipedia.org/wiki/Computer_file) containing that data will produce unexpected results when accessed by the system or the related application. Results could range from a minor loss of data to a system crash. For example, if a [document file](https://en.wikipedia.org/wiki/Document_file_format) is corrupted, when a person tries to open that file with a document editor they may get an [error message](https://en.wikipedia.org/wiki/Error_message), thus the file might not be opened or might open with some of the data corrupted

**OS Hardening:**

In computing, hardening is usually the process of securing a system by reducing its surface of vulnerability, which is larger when a system performs more functions; in principle a single-function system is more secure than a multipurpose one.

The idea of **OS hardening** is to minimize a computer's exposure to current and future threats by fully configuring the operating system and removing unnecessary applications.

**Boot Device:**

A boot device is any piece of hardware containing the files required for a computer to start. For example, a hard drive, floppy disk drive, CD-ROM drive, DVD drive, and USB jump drive are all considered bootable devices. However, unless the necessary boot files are stored on the drive, diskette, or disc, the computer cannot boot from that device.

The most commonly used boot device or boot drive is the hard drive. When an operating system (e.g., Microsoft Windows) is installed on the hard drive, it copies the boot files and drivers required to load Windows on the computer.

**Software Patch:**A patch is a set of changes to a computer program or its supporting data designed to update, fix, or improve it. This includes fixing security vulnerabilities and other bugs, with such patches usually being called bug fixes or bug fixes. Patches are often written to improve the functionality.

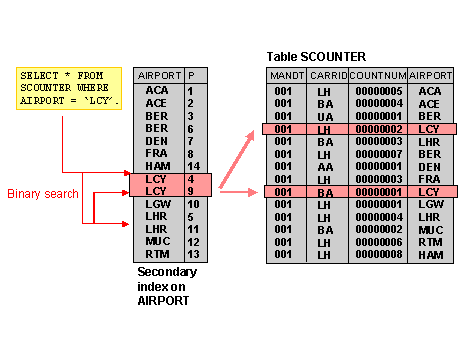
**Cardinality in DB:** It is the number of distinct values in a table column relative to the number of rows in the table. Repeated values in the column don’t count. It is linked to query performance and query optimization.

**Indexing in DB:** Indexes are used to increase the system's ability to retrieve data by not scanning all of the data records in search of the desired record(s). Indexes can change the organization of data so that it is structured similarly to how it will be accessed for retrieval. The indexes are created on column(s) to help the database find data based upon the value(s) contained in those indexed column(s).

The index column is sorted and stored in a separate table which and its value contains a pointer to the original record. Sorted indexes can be retrieved using binary search in log running time. Index can be **Dense index** or **Sparse index.**

There are 3 ways of indexing,

* Clustered Indexing
* Non-Clustered or Secondary Indexing
* Multilevel Indexing



The more indexes you have, the more likely you will retrieve data from the system quickly. However, it's equally true that the more indexes you have, the more storage it requires and the longer it will take to insert new data into the system.

**GraphQL:**

**CI/CD:**

**Deployment strategies:**

Multi-service deployment:

Rolling deployment:

Blue-green deployment:

Canary deployment:

A/B testing:

**ElasticSearch:**

**InfluxDB:**

**Simple Authentication Security Layer (SASL):**

**SPF, DKIM, DMARC:**

**VUE.js**

**VMware:**

**Mime Type:**

**Postgres vs MySQL:**

**GeoJson:**

**\*\*\*Dependency Injection(DI):**

**Terraform:**

**Terraform is an open-source infrastructure as code software (IAC)** for building, changing, and versioning infrastructure safely and efficiently. **Terraform** can manage existing and popular service providers as well as custom in-house solutions.

**Ansible:**

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.

Ansible vs Terraform:

<https://cloudify.co/blog/ansible-vs-terraform/>

**Bastion Host:**

A bastion host is a special-purpose computer on a network specifically designed and configured to withstand attacks. The computer generally hosts a single application, for example a proxy server, and all other services are removed or limited to reduce the threat to the computer.

**Telemetry:**

It is the in situ collection of measurements or other data at remote or inaccessible locations and their automatic transmission to receiving equipment for monitoring. In Software development, it is used to offer insights on which features end users use most, detection of bugs and issues, performance monitoring etc. It can also be used in meteorology, medicine and other fields.

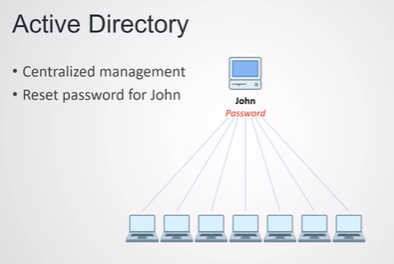
**Active Directory:**

It is a directory service by Microsoft for Windows domain networks. It is included in most Windows server operating systems as a set of processes and services. It enables administrators to manage permissions and access to network resources.

It stores data as objects. An object is a single element such as a user, group, application or device such as a printer. Objects are normally defined as either resources, such as printers or computers, or security principals such as users or groups.

The main service in AD is Domain Service (AD-DS), which stores directory information and handles the interaction of the user with the domain.

LDAP – Lightweight Directory Access Protocol – used for maintaining directory info services over IP. Commonly used for storing passwords, username etc.



**SSD vs HDD:**

* **HDDs**: An enclosure contains a series of platters covered by a ferromagnetic coating. The direction of the magnetization represents the individual bits. Data is read and written by a head (similar to [the way vinyl record albums work](https://www.explainthatstuff.com/record-players.html)) that moves extremely fast from one area of the disk to another. Since all of these pieces are “mechanical,” the hard disk is the slowest component of any computer – and the most fragile.
* **SSD**: These newer types of disks store information on flash memory, which consists of individual memory cells storing bits that are instantly accessible by the controller. They are non-mechanical, require less power, improve battery life and are shock-resistant.

**Churn rate**

Churn rate, in its broadest sense, is a measure of the number of individuals or items moving out of a collective group over a specific period. It is one of two primary factors that determine the steady-state level of customers a business will support.

**Race conditions, Deadlock:**

**IEEE 802:**

Itis a family of IEEE standards for LAN, PAN (Personal Area Networks), MAN (Metropolitan Area Networks). The IEEE 802 standards are restricted to [computer networks](https://en.wikipedia.org/wiki/Computer_network) carrying variable-size packets. The number 802 has no significance.

| **Name** | **Description** | **Status** |
| --- | --- | --- |
| [IEEE 802.1](https://en.wikipedia.org/wiki/IEEE_802.1) | Higher Layer LAN Protocols Working Group | Active |
| [IEEE 802.2](https://en.wikipedia.org/wiki/IEEE_802.2) | [LLC](https://en.wikipedia.org/wiki/Logical_link_control) | Disbanded |
| [IEEE 802.3](https://en.wikipedia.org/wiki/IEEE_802.3) | [Ethernet](https://en.wikipedia.org/wiki/Ethernet) | Active |
| [IEEE 802.4](https://en.wikipedia.org/wiki/IEEE_802.4) | [Token bus](https://en.wikipedia.org/wiki/Token_bus) | Disbanded |
| [IEEE 802.5](https://en.wikipedia.org/wiki/IEEE_802.5) | [Token Ring](https://en.wikipedia.org/wiki/Token_Ring) MAC layer | Disbanded |
| [IEEE 802.6](https://en.wikipedia.org/wiki/IEEE_802.6) | [MANs](https://en.wikipedia.org/wiki/Metropolitan_area_network) ([DQDB](https://en.wikipedia.org/wiki/Distributed-queue_dual-bus)) | Disbanded |
| [IEEE 802.7](https://en.wikipedia.org/wiki/IEEE_802.7) | Broadband LAN using Coaxial Cable | Disbanded |
| [IEEE 802.8](https://en.wikipedia.org/wiki/IEEE_802.8) | Fiber Optic TAG | Disbanded |
| [IEEE 802.9](https://en.wikipedia.org/wiki/IEEE_802.9) | Integrated Services LAN (ISLAN or isoEthernet) | Disbanded |
| [IEEE 802.10](https://en.wikipedia.org/wiki/IEEE_802.10) | Interoperable LAN Security | Disbanded |
| [IEEE 802.11](https://en.wikipedia.org/wiki/IEEE_802.11) | [Wireless LAN](https://en.wikipedia.org/wiki/Wireless_LAN) (WLAN) & Mesh ([Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) certification) | Active |
| IEEE 802.12 | [100BaseVG](https://en.wikipedia.org/wiki/100BaseVG) | Disbanded |