Introduction to Cloud Computing

quentin.bramas@lip6.fr - PROGRES 2016-2017

History

How to host a website / application ?

- Buy a computer and connect it to the internet

  If it works well, build a data center

- Pay someone that own a datacenter, and rent a server

  If it works well, just rent more servers

- Pay someone that own datacenter and just rent a virtual server.

  If it works well, just ask that the virtual server to have be more powerful

- Pay someone that can execute your application.

  If it works well, just pay accordingly.

- Make an application that delegates all the heavy computations to other application and pay someone that can execute your application.

What is cloud computing?

Cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you as a service by another company and accessed over the Internet

Exactly where the hardware and software is located and how it all works doesn't matter

What is cloud computing?

Cloud Computing is a way to abstract the execution environment of an application.

Pros:

- It’s managed
- It's « on-demand »
- Only pay for what you use.
- Lower upfront costs and reduced infrastructure costs.
- Easy to grow your applications.

Cons

- Potential privacy and security risks
- Greater dependency on service providers.

Examples

Google Cloud Platform

GitHub

amazon web services

Microsoft Azure

Google™ Apps

OVH.com

Dropbox

heroku

iCloud

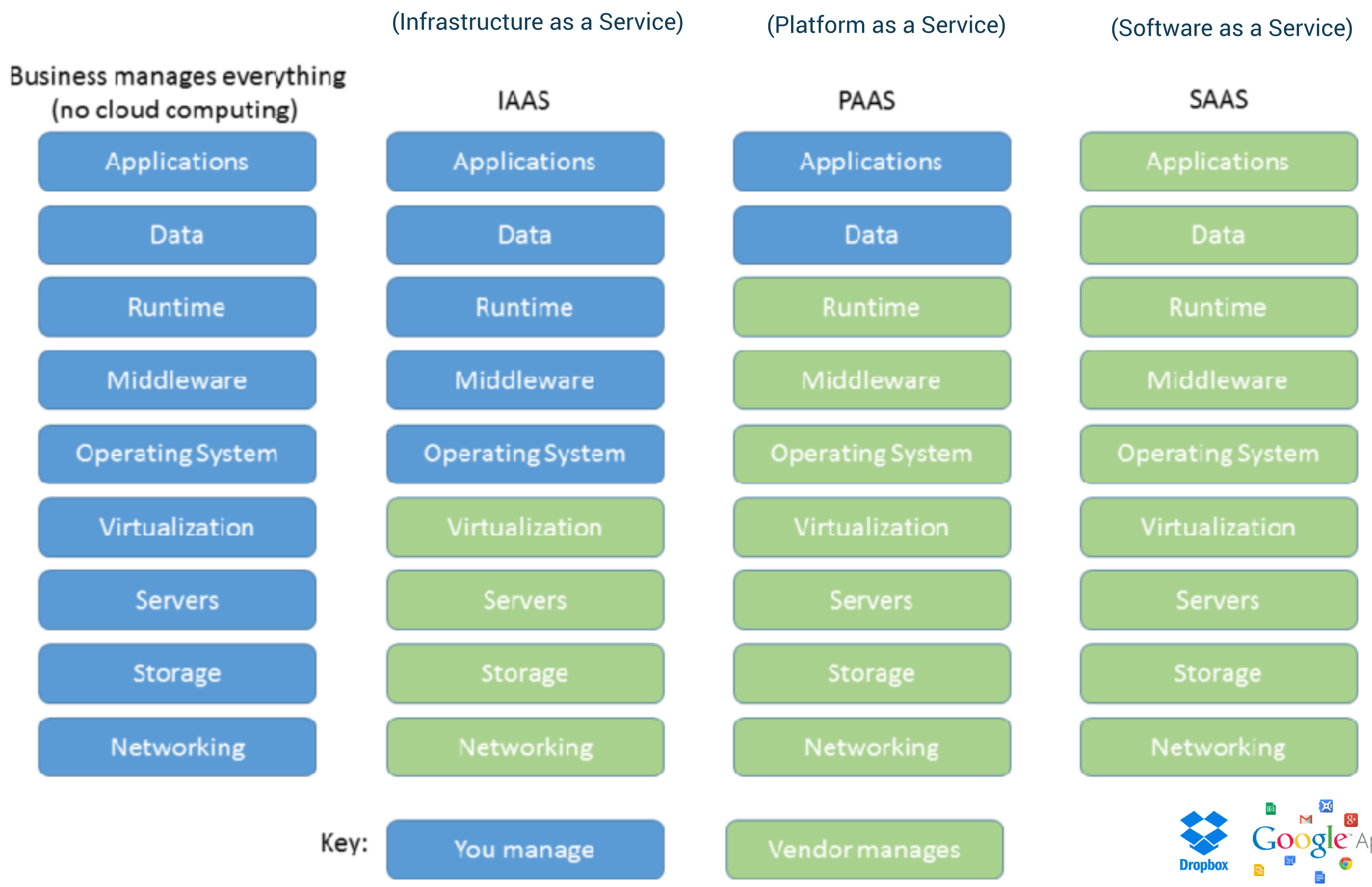
Taxonomy

SaaS (Software as a Service)

PaaS (Platform as a Service)

IaaS (Infrastructure as a Service)

Taxonomy



Business manages everything (no cloud computing)	IAAS (Infrastructure as a Service)	PAAS (Platform as a Service)	SAAS (Software as a Service)
🟦 Applications	🟦 Applications	🟦 Applications	🟩 Applications
🟦 Data	🟦 Data	🟦 Data	🟩 Data
🟦 Runtime	🟦 Runtime	🟩 Runtime	🟩 Runtime
🟦 Middleware	🟦 Middleware	🟩 Middleware	🟩 Middleware
🟦 Operating System	🟦 Operating System	🟩 Operating System	🟩 Operating System
🟦 Virtualization	🟩 Virtualization	🟩 Virtualization	🟩 Virtualization
🟦 Servers	🟩 Servers	🟩 Servers	🟩 Servers
🟦 Storage	🟩 Storage	🟩 Storage	🟩 Storage
🟦 Networking	🟩 Networking	🟩 Networking	🟩 Networking

Key: 🟦 You manage 🟩 Vendor manages

OVH.com amazon web services Microsoft Azure heroku Google Cloud Platform

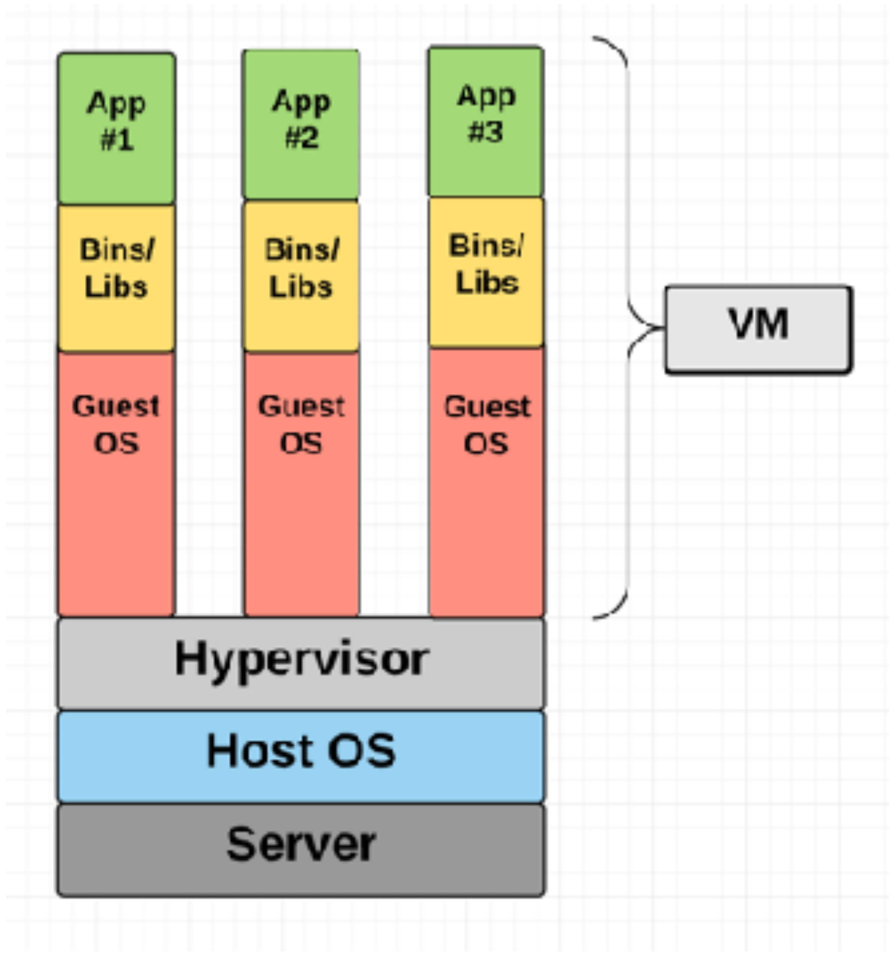
Dropbox Google Apps GitHub iCloud

image : www.joebarrios.com/

Virtualization

Virtualization

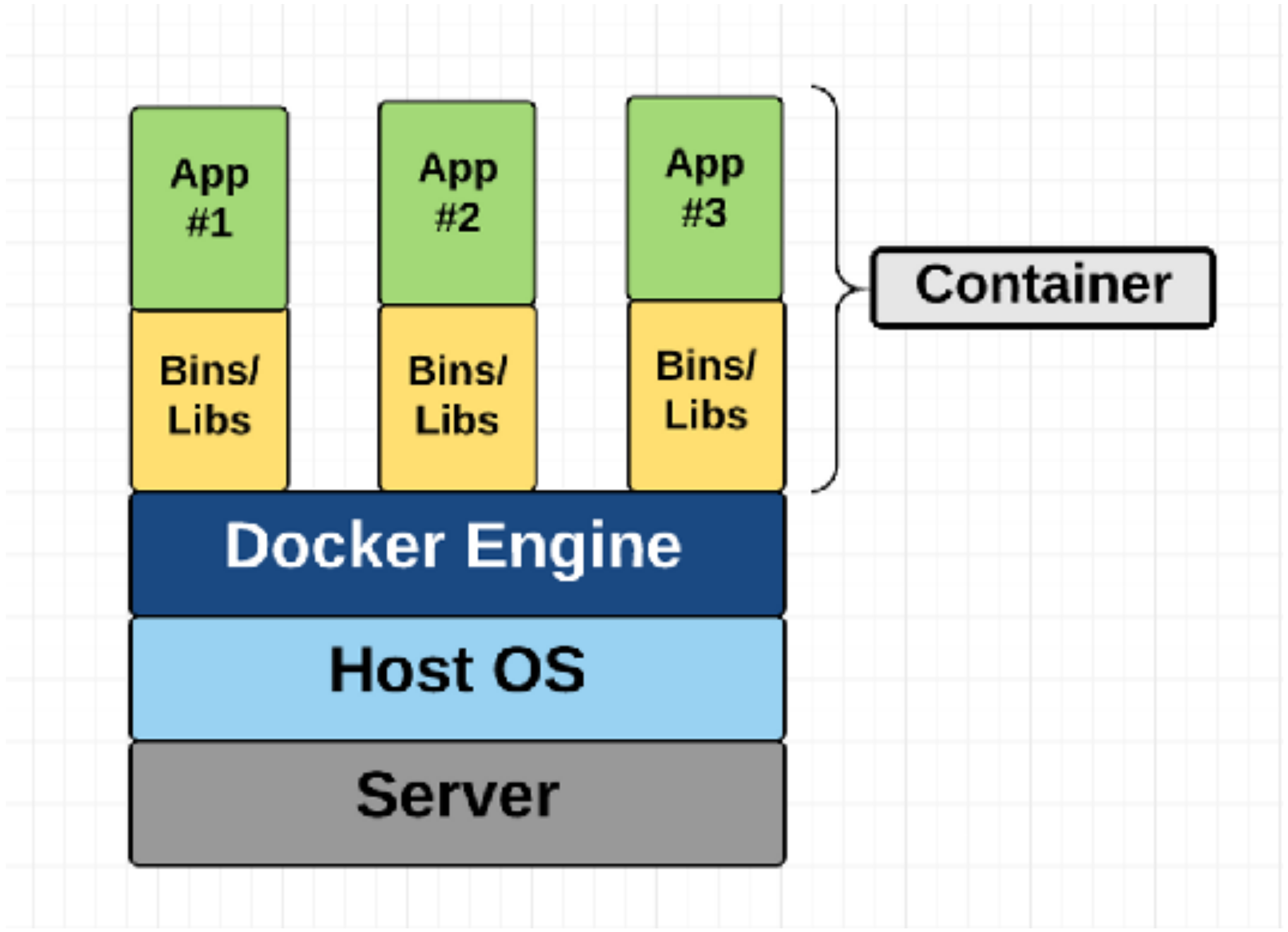
A Virtual Machine (VM) is essentially an emulation of a real computer that executes programs like a real computer. VMs run on top of a physical machine using a "hypervisor". A hypervisor, in turn, runs on either a host machine or on "bare-metal".



link: https://goo.gl/CX8ekS

Container

Unlike a VM which provides hardware virtualization, a container provides operating-system-level virtualization by abstracting the “user space”. The one big difference between containers and VMs is that containers share the host system’s kernel with other containers.



Docker container

Docker is an open-source project based on Linux containers

Ease of use: “build once, run anywhere.”

Speed: Docker containers are very lightweight and fast. Since containers are just sandboxed environments running on the kernel, they take up fewer resources. You can create and run a Docker container in seconds, compared to VMs which might take longer because they have to boot up a full virtual operating system every time.

Modularity and Scalability

Docker Container

How to create a container?

Dockerfile

A Dockerfile is where you write the instructions to build a Docker image. These instructions can be:

`FROM ubuntu:14.04`: the base container from which we build our container
`RUN apt-get y install some-package`: to install a software package
`EXPOSE 8000`: to expose a port
`ENV ANT_HOME /usr/local/apache-ant`: to pass an environment variable
`CMD ["python","myscript.py"]` : the command that is executed when the container is run.

Docker Container

How to create a container?

Dockerfile, example:

```
FROM docker/whalesay:latest
RUN apt-get -y update && apt-get install -y fortunes
CMD /usr/games/fortune -a | cowsay
```

Docker Container

Docker Client: an application to manage your container

build the container using the DockerFile in the current directory:

```
docker build -t myContainer .
```

name (tag) of my container

run the container:

```
docker run myContainer
```

Docker Containers

Links between containers

Start a mongodb container:

```
docker run --name some-mongo -d mongo
```

Create a DockerFile:

```
FROM ubuntu:16.04
RUN echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2
multiverse" | tee /etc/apt/sources.list.d/mongodb-org-3.2.list
RUN apt-get update
RUN apt-get install -y --allow-unauthenticated mongodb-org-shell

CMD ["mongo", "some-mongo:27017"]
```

Start our container with a link to the mongodb container:

```
docker run  --link some-mongo testapp
```

Output:

```
MongoDB shell version: 3.2.12
connecting to: some-mongo:27017/test
bye
```

Docker Containers

```
FROM ubuntu:16.04
RUN echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2
multiverse" | tee /etc/apt/sources.list.d/mongodb-org-3.2.list
RUN apt-get update
RUN apt-get install -y --allow-unauthenticated mongodb-org-shell python3 python3-
pip
RUN python3 -m pip install pymongo bottle

COPY startup.py .


EXPOSE 8082

CMD ["python3", "startup.py"]
```

```
docker run -d --link some-mongo -p 8082:80 testapp
```

Docker Containers

```
import pymongo
client = pymongo.MongoClient("some-mongo", 27017)
db = client.test

from bson.json_util import dumps
import bottle

@bottle.get('/messages')
def hello():
    return dumps(db.my_collection.find())

@bottle.get('/add_messages/<msg>')
def hello(msg):
    id = db.my_collection.insert_one({"msg": msg}).inserted_id
    return str(id)

if __name__ == '__main__':
    bottle.run(host='0.0.0.0', port=80, debug=True)
```

Docker Containers

docker-compose is a tool to create multi-container applications

`docker-compose.yml`

```
version: '2'
services:
  app:
    build: .
    ports:
    - "8082:80"
    links:
    - some-mongo
  some-mongo:
    image: mongo
```

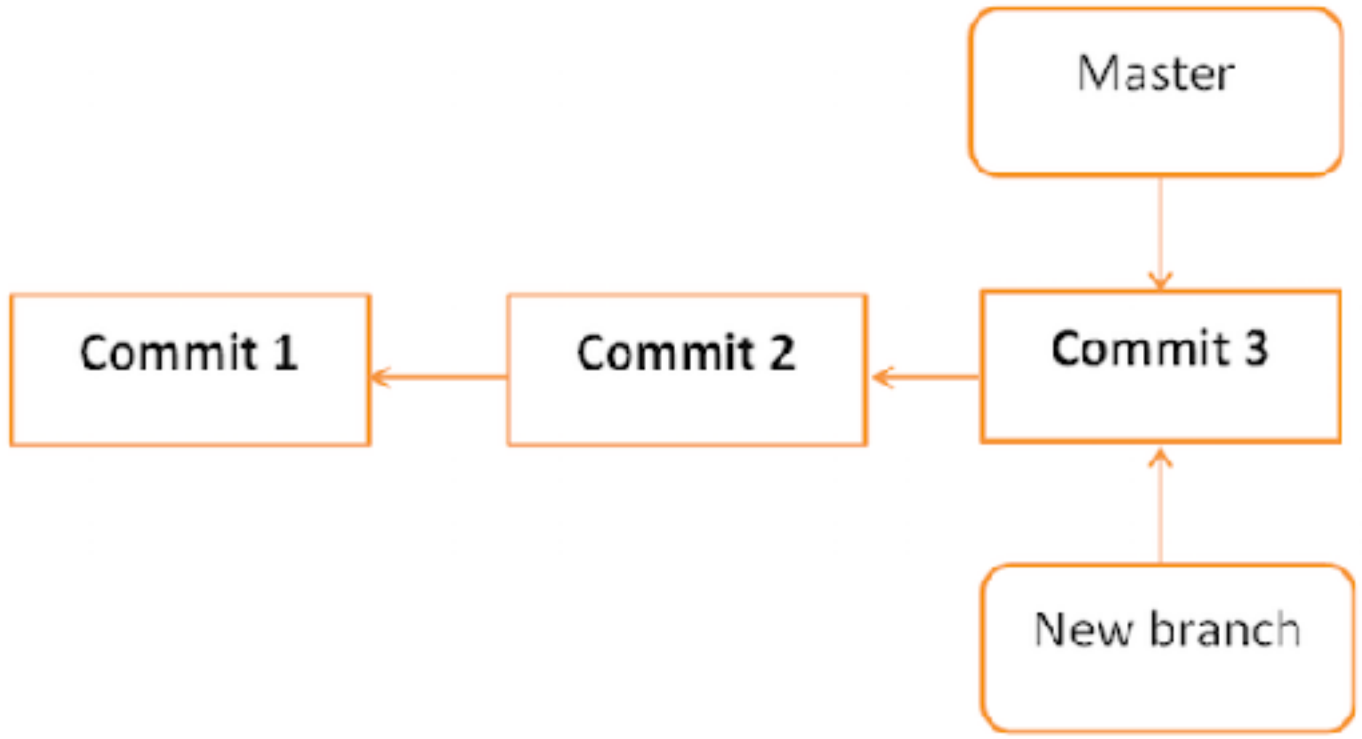
Git

Git

Git is a Version Control System:
A system that keeps track of changes made to files

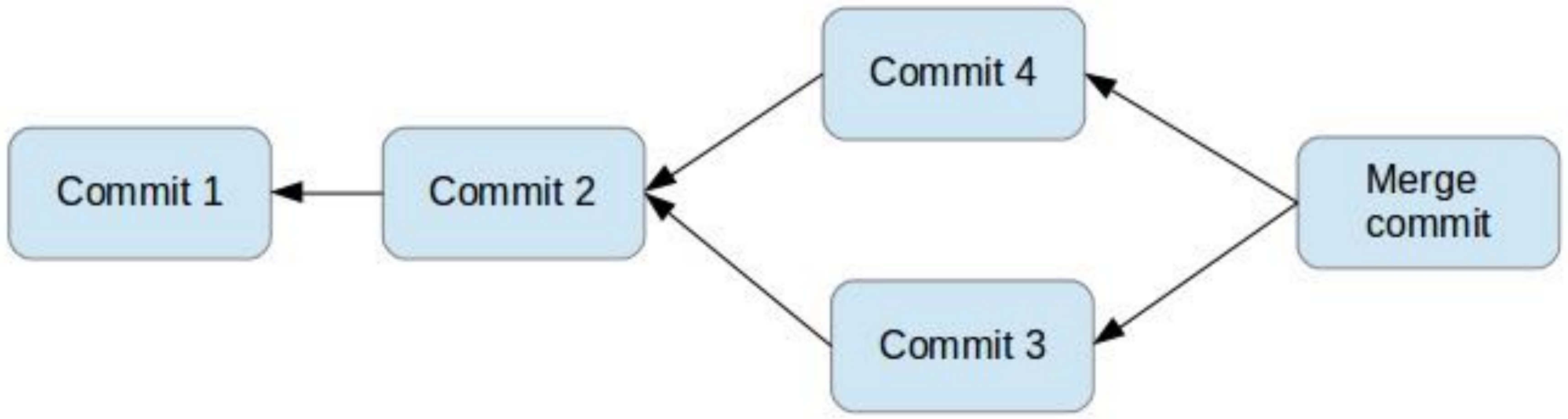
A Git repository contains the history of a collection of files starting from a certain directory

Conceptually a commit represents a version of all files tracked in the repository at the time the commit was created. Commits know their parent(s) and this way capture the version history of the repository.



Git

When you want to try something, you can create another branch, and then merge it when you want



Git

add a file to the version control:

```
git add myFile
```

create a commit with the current state:

```
git commit -m “I’ve added an awesome feature”
```

You can synchronize your repository with a remote one

add a remote repository:

```
git remote add origin URL_OF_THE_REPO
```

push your changes:

```
git push origin master
```

checkout the last version from the remote repository:

```
git pull origin master
```

Heroku deployement

Heroku is a PaaS that allows us to deploy an app that can be scalled atomatically.

Heroku offers a nice CLI to deploy an app quickly

```
heroku login
heroku create
git push heroku master
```

create a git remote repository

deploy the app

Heroku deployement

The git repo of a NodeJS application must not contain the node_modules directory

A web application deployed on Heroku must listen to the port given in the environment variable called PORT.

In nodeJS, using Express:

```
var express = require('express');
var app = express();
app.set('port', (process.env.PORT || 5000));

app.use(express.static(__dirname + '/public'));

app.listen(app.get('port'), function() {
  console.log('Node app is running on port', app.get('port'));
});
```

Micro Services

Using cloud services such as mLab

mLab is a SaaS, it offers a mongo database that is managed, duplicated and backed-up, ready for production.

connect to mongodb from nodejs

```
var MongoClient = require('mongodb').MongoClient
  , assert = require('assert');

// Connection URL
var url = 'mongodb://USER:PASSWORD@HOST:PORT/DB';

// Use connect method to connect to the server
MongoClient.connect(url, function(err, db) {
  assert.equal(err, null);

  console.log("Connected successfully to server");

  var collection = db.collection('documents');
  collection.insertMany([
    {a : 1}, {a : 2}, {a : 3}
  ], function(err, result) {
    assert.equal(err, null);
    console.log("Inserted 3 documents into the collection", result);
    db.close();
  });

});
```

Using cloud services such as cloudsight.ai

```
app.get('/label', function(request, httpResponse) {
    axios.request({
      url:'http://api.cloudsight.ai/image_requests',
      method: 'POST',
      headers: {Authorization: 'CloudSight APIKEY'},
      params:{
        'image_request[remote_image_url]': request.query['url'],
        'image_request[locale]': 'en-US'
      }
    })
    .then(function (response) {
      console.log('start polling');
      setTimeout(pollResponse(response.data.token, httpResponse), requestInterval);
    })
    .catch(function (error) {
      console.log(error.response.data);
    });
  });
```