
# Università di Pisa

**Dipartimento di Informatica**
**Corso di Laurea Triennale in Informatica**

Corso a Libera Scelta - 6 CFU

## Cloud Computing

**Professore:**
Prof. Antonio Brogi

**Autore:**
Filippo Ghirardini

---

**Anno Accademico 2023/2024**

# Contents

# Cloud Computing

Realizzato da: Ghirardini Filippo

A.A. 2023-2024

---

# 1 Introduzione

## 1.1 Service-based economy

L'economia basata sui servizi (*Everything as a service*) si fonda sulla tendenza degli ultimi 30 anni di passare dai *beni* ai *servizi*.

**Esempio 1.1.1** (Bicicletta)**.** Una signora compra una bicicletta dal venditore. Questa diventa di sua proprietà e si deve occupare della manutenzione. La bicicletta diventa un **servizio** (*CicloPi*) e la signora paga per usare la bici che però non è più sua (e non deve più preoccuparsi di manutenzione e furto).

Esempi più vicini all'informatica sono il passaggio dai supporti fisici per la musica allo streaming o i dispositivi di memorizzazione passati ai Cloud Drive.

## 1.2 Service contracts

Quando usiamo un servizio non vogliamo sapere come viene implementato. L'unica cosa che ci interessa è cosa è specificato sul **contratto di utilizzo**. Nella maggior parte dei casi l'utente non lo legge.

### 1.2.1 Quality of Service

Ci sono più fornitori che ci danno lo stesso servizio ma con qualità del servizio diverse. Dobbiamo chiederci se il prezzo più basso vale la pena del sacrificio della qualità.

### 1.2.2 Service Level Agreement

Sono i contratti di servizio che includono anche il livello di **affidabilità di servizio**. In questa situazione abbiamo tre figure:

- *Programmatore*
- *Business expert*: colui che sa il livello di affidabilità in base al mercato
- *Legale*: colui che sa come scriverlo

**Esempio 1.2.1** (Google SLA)**.** Google Compute Engine fornisce un **Service Level Objective** (SLO) del 99.95%. In caso di non raggiungimento del SLO si viene rimborsati con del credito in percentuale a quanto si è distanti dal target.

| **Montlhy Uptime Percentage** | **Rimborso** |
|---|---|
| $95.00\% - < 99.95\%$ | 10% |
| $90.00\% - < 95.00\%$ | 25% |
| $< 90.00\%$ | 100% |

Se ad esempio l'1 e il 2 Aprile dalle 8am alle 5pm (orario di lavoro) non era disponibile il servizio, a quanto ammonta il rimborso in credito?
Dobbiamo prima capire cos'è il **Montly Uptime Percentage** secondo l'SLA:

**Definizione 1.2.1** (Monthly Uptime Percentage)**.** *Total number of minutes in a month, minus the number of minutes of* ***Downtime*** *suffered from all* ***Downtime Period*** *in a month, divided by the total number of minutes in a month.*

**Definizione 1.2.2** (Downtime Period)**.** *A period of one of more consecutive minutes of* ***Downtime****.*

**Definizione 1.2.3** (Downtime)**.** *For virtual machines instances: loss of external connectivity or persistent disk accessfor the Single Instance or, with respect to Instances in Multiple Zones, all applicable running instances.*

Inoltre il cliente deve richiedere entro 60 giorni il rimborso fornendo anche dei **log file** che mostrino il **Downtime Period** assieme alla data e all'orario.

*a*

**Definizione 1.2.4** (Legge del pesce rosso)**.** *Quando un fornitore di servizi non garantisce in alcun modo il prodotto.*

**Esempio 1.2.2** (Microsoft Service Agreement)**.** Microsoft non è responsabile per alcuna perdita di dati o interruzione di servizi e non li garantisce in alcun modo (vedi art.6 e art.11).

**Esempio 1.2.3.** Supponiamo di avere un servizio che si appoggia a due servizi cloud, ognuno disponibile in maniera indipendente il 90% del tempo. Il servizio è probabile che non sarà disponibile per 4 ore e mezza al giorno:

$$(1 - (0.90 * 0.90)) * 24 = 4.56$$

## 1.3 Cloud

Un primo fattore da tenere in considerazione è la **service demand**, che cambia nel tempo.
Prima del cloud c'era il problema di dimensionare il servizio in base a delle stime, che possono essere di due tipi:

- **Overprovisioning**: utilizzare un infrastruttura che possa sopportare anche i carichi massimi previsti. In questo caso abbiamo un problema di **spreco di risorse**.
- **Underprovisioning**: quando l'infrastruttura non è sufficiente per tutti i momenti di carico. Questo causerebbe la perdita di clienti che dopo qualche tentativo fallito di utilizzo abbandonerebbero il servizio.

Il **cloud** ci fornisce risorse apparentemente infinite disponibili su richiesta.

**Definizione 1.3.1** (Cloud)**.** *Secondo il NIST il Cloud Computing è un **modello** per consentire l'accesso ubiquo, conveniente e a richiesta di risorse computazionali.*
*Le idee chiave sono le seguenti:*

- *Messa in comune di strutture autogestite, utilizzate come servizi*
- *Risorse virtuali **scalabili** disponibili attraverso internet a diversi clienti*
- ***Separazione** dei servizi dalla tecnologia alla base*

*Dal punto di vista economico sfrutta il principio **pay-per-use**, che permette all'utente di convertire i costi da Capital Expenses a Operation Expenses. Questo garantisce elasticità e trasferimento dei rischi al fornitore dei servizi.*

### 1.3.1 Service models

Analizziamo ora i modelli di servizio, prendendo come esempio la pizza:

- **IaaS**: fornisce server virtualizzati, archiviazione e rete e li gestisce. Il cliente è responsabile per tutti gli altri aspetti, quali OS e applicazioni. Nel nostro caso sarebbe comprare una pizza da cuocere.
- **PaaS**: fornisce l'intera piattaforma come servizio (VM, OS, servizi, SDKs) e gestisce l'infrastruttura, l'OS e l'abilitazione del software. Il cliente è responsabile dell'installazione e la gestione delle applicazioni. Nel nostro caso sarebbe una pizza d'asporto.
- **SaaS**: fornisce software a richiesta, disponibile tramite API. Viene gestita l'infrastruttura, l'OS e l'applicazione, quindi il cliente non è responsabile di nulla. Nel nostro caso sarebbe la pizzeria.

### 1.3.2 Deployments models

La scelta del deployment model può essere **pubblico**, **privato** o **ibrido**. Il modello privato garantisce un maggiore controllo sui dati mentre quello pubblico una maggiore scalabilità. La versione ibrida divide i dati tra pubblico e privato in base alle esigenze di sicurezza e scalabilità.

### 1.3.3 Domande

Ci si deve fare alcune domande sulla confidenzialità dei nostri dati:

- Dove saranno fisicamente conservati i nostri dati?
- La privacy e l'integrità dei nostri dati sono garantite? Come?
- Come possiamo sapere se c'è stato un problema?

sulla disponibilità dei servizi:

- Cosa succede se il fornitore del servizio non lo fornisce?
- SPoF (Single Point of Failure): quando la nostra architettura sfrutta un solo servizio che potrebbe fallire

Le risposte le troviamo tutte nel SLA.

### 1.3.4 Vendor Lock-In

**Definizione 1.3.2** (Vendor Lock-In)**.** *Il Vendor Lock-In è quando si rende un cliente dipendente da un venditore per prodotti o servizi, rendendogli impossibile il cambio ad un altro gestore senza affrontare costi esorbitanti.*

Cosa succede se si vuole cambiare provider? Il software continuerà a funzionare? Si potrà trasferire facilmente i propri dati? Serve sempre un **exit plan**.