UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di Laurea Triennale in
Ingegneria Informatica

Tesi di Laurea

Studio ed analisi delle soluzioni Cloud per l'Internet of Things

Relatore

Ing. Fabrizio Marozzo

Candidato

Stefano Perna

Matricola: 201130

Anno Accademico 2020-2021

Alla mia famiglia.

"Earn with your mind, not your time."
— Naval Ravikant

Indice

1 Cloud computing **7**
- 1.1 Definizione . . . 7
- 1.2 Origini . . . 8
- 1.3 Caratteristiche essenziali . . . 10
- 1.4 Modelli di servizio . . . 10
  - 1.4.1 SaaS - Software as a Service . . . 11
  - 1.4.2 PaaS - Platform as a Service . . . 12
  - 1.4.3 IaaS - Infrastructure as a Service . . . 12
- 1.5 Modelli di distribuzione . . . 14
  - 1.5.1 Public cloud . . . 15
  - 1.5.2 Private cloud . . . 15
  - 1.5.3 Hybrid cloud . . . 16
  - 1.5.4 Community cloud . . . 16
- 1.6 Architettura e tecnologie Cloud . . . 16
  - 1.6.1 Data center . . . 16
  - 1.6.2 Virtualizzazione . . . 18
- 1.7 Fattori economici . . . 19

2 Internet of Things **21**
- 2.1 Definizioni: un paradigma, diverse visioni . . . 21
- 2.2 Cenni storici . . . 22
- 2.3 IoT: una tecnologia dirompente . . . 23
  - 2.3.1 Vantaggi . . . 23
  - 2.3.2 Tecnologie abilitanti . . . 24
  - 2.3.3 Scenari applicativi . . . 25
- 2.4 Modelli di comunicazione . . . 27
  - 2.4.1 Modello Device-to-Device . . . 28
  - 2.4.2 Modello Device-to-Cloud . . . 28

2.4.3 Modello Device-to-Gateway . . . 28
2.5 Protocolli di comunicazione . . . 29
2.5.1 MQTT . . . 29
2.5.2 REST . . . 33
2.5.3 CoAP . . . 33
2.5.4 AMQP . . . 34
2.5.5 Considerazioni sui protocolli di comunicazione a livello applicativo 35

3 Soluzioni Cloud per l'Internet of Things 37
3.1 Integrazione tra Cloud e IoT . . . 37
3.1.1 Vantaggi del paradigma Cloud-IoT . . . 37
3.2 Architetture IoT . . . 39
3.3 Limiti di una soluzione Cloud-centrica . . . 40
3.3.1 Edge computing . . . 41
3.4 Piattaforme Cloud per IoT . . . 42
3.4.1 Google Cloud IoT . . . 43
Raccolta dei dati . . . 43
Acquisizione ed elaborazione dei dati . . . 43
Google Cloud IoT Core . . . 44
Archiviazione, analisi dei dati e Machine Learning . . . 46
Sicurezza . . . 47
3.4.2 AWS IoT . . . 48
FreeRTOS . . . 49
AWS IoT Greengrass . . . 49
AWS IoT Core . . . 50
AWS IoT Device Management . . . 51
Elaborazione e archiviazione dei dati . . . 52
AWS IoT Analytics . . . 52
AWS IoT SiteWise . . . 53
Sicurezza . . . 53
3.4.3 Azure IoT . . . 54
Azure IoT Central . . . 55
Azure IoT Hub . . . 56
Things . . . 57
Insights . . . 58
Actions . . . 59
3.4.4 Considerazioni finali . . . 59

Introduzione

L'Internet of Things (IoT) rappresenta una delle tecnologie più dirompenti dell'ultimo decennio ed è alla base di innumerevoli scenari di elaborazione distribuita. L'idea alla base del modello IoT è quella di interconnettere il mondo reale con il mondo digitale, integrando gli oggetti di tutti i giorni con capacità computazionali e di telecomunicazione, al fine di automatizzare processi, semplificare la vita degli utenti, raccogliere dati e molto altro. L'IoT è quindi costituito da miliardi di dispositivi eterogenei intelligenti, chiamati anche nodi o *thing*, che una volta interconnessi tramite una rete dinamica e globale comunicano tra di loro, si interfacciano con il mondo reale e raccolgono un'enorme quantità di dati. L'eterogeneità dei dispositivi a disposizione e la possibilità di interconnetterli attraverso una rete pubblica come Internet, consente di applicare il paradigma IoT in tantissimi contesti applicativi, ad esempio nel settore industriale, sanitario ed energetico. Tuttavia, tali nodi hanno tipicamente *capacità elaborativa e spazio di archiviazione limitata*, con conseguenti problemi di affidabilità, prestazioni e sicurezza. Per contro, il Cloud computing (CC) descrive un nuovo modello di elaborazione che consente l'accesso ad un insieme di risorse IT (server, strutture di archiviazione e di telecomunicazione ecc.) *virtualmente illimitate*, condivise e configurabili. Queste risorse possono essere rapidamente fornite come *servizio su richiesta* con il minimo sforzo di gestione, secondo il modello *pay-per-use*, in modo tale che l'utente paghi solo per quello che utilizza. Inoltre, l'accesso ai servizi in Cloud può avvenire da qualsiasi luogo e con qualsiasi dispositivo tramite Internet. Per tanto, l'*integrazione* tra l'Internet of Things ed il paradigma Cloud sembra essere una delle soluzioni più promettenti per abilitare tutte le potenzialità dell'IoT su larga scala e al fine di sviluppare applicazioni IoT scalabili, affidabili, flessibili ed efficienti. Su questa linea, negli ultimi anni i *Cloud provider* hanno sviluppato diverse piattaforme Cloud per IoT.

L'obiettivo di questa tesi è quello di approfondire i motivi che portano ad una integrazione tra Cloud computing ed Internet of Things attraverso lo studio di entrambi i paradigmi e l'analisi delle soluzioni Cloud per IoT che attualmente offre il mercato. In particolare, viene condotta un'approfondita analisi ed una comparativa delle principali

piattaforme Cloud per IoT che attualmente detengono la leadership del mercato, ovvero *AWS IoT* di Amazon, *Google Cloud IoT* e *Azure IoT* di Microsoft.

Nel primo capitolo viene presentato il paradigma Cloud. Si descrive: come il modello è definito; quali sono le caratteristiche essenziali; i modelli di servizio e di distribuzione; le tecnologie utilizzate ed i motivi che lo hanno portato ad emergere. Il secondo capitolo offre una panoramica sull'Internet of Things, si discutono le diverse definizioni che sono state date nel corso degli anni, le tecnologie che ne hanno permesso la diffusione, i vantaggi che può portare ed alcuni scenari applicativi. Infine, vengono illustrati i modelli ed i protocolli di comunicazione utilizzati in ambito IoT. Nel terzo capitolo l'attenzione è focalizzata sull'integrazione tra Cloud computing ed Internet of Things, viene discusso il perchè di questa integrazione, i vantaggi che può portare, i limiti e le sfide future che ne derivano. Si presenta un modello architetturale di riferimento per una tipica applicazione IoT, quindi si analizzano le tre piattaforme Cloud prese in esame. Per ogni piattaforma si presenta l'architettura, i numerosi servizi offerti ed il modo in cui possono essere combinati per implementare una soluzione IoT.

CAPITOLO 1

Cloud computing

Il *Cloud computing* descrive un nuovo modello di elaborazione distribuito in cui le risorse, dinamicamente scalabili e spesso virtualizzate, sono fornite come servizio su richiesta tramite Internet [1]. I dettagli implementativi e strutturali sono nascosti all'utente finale, il quale non deve preoccuparsi della gestione del sistema e paga solo per quello che utilizza.

Questo capitolo contiene una panoramica sul paradigma *Cloud computing*. Il paragrafo 1.1 presenta una definizione di *Cloud computing*. Di seguito vengono illustrati i punti salienti della sua storia, saranno descritte le caratteristiche essenziali, i modelli di servizio e di distribuzione, per poi passare ad analizzare l'architettura e le tecnologie utilizzate per implementare soluzioni Cloud. Infine, viene fatta una considerazione sui fattori economici che hanno portato il modello Cloud ad emergere.

1.1 Definizione

Che cosa significa *Cloud Computing*? Nel corso degli anni sono state suggerite diverse definizioni. Tra queste, la definizione data dal NIST (*National Institute of Standards and Technology)*)[2] è quella che ha ottenuto il maggior consenso sia da parte del mondo industriale che da parte della comunità scientifica:

> *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*
> — *The NIST definition of cloud computing[2]*

Essa racchiude le parole chiave che maggiormente caratterizzano il paradigma *Cloud*. Un modello nel quale le risorse non sono più considerate come un prodotto da possedere, bensì come un *servizio* sempre disponibile, fornito su *richiesta*, erogato tramite la rete Internet e con il minimo sforzo di gestione.

1.2 Origini

In questo paragrafo vengono illustrati alcuni punti salienti della storia del Cloud computing.

Idea iniziale

L'idea di trasferire l'elaborazione su *cloud* nasce da una vecchia intuizione: vedere l'elaborazione o più in generale le risorse di sistema come un servizio pubblico *pay-per-use* al pari di acqua, elettricità, gas, ecc. Questo concetto sta alla base del modello conosciuto come *Utility computing* proposto già nel 1961 da John McCarthy:

> *"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."*
> — *John McCarthy, speaking at the MIT Centennial in 1961[3]*

Questa idea di elaborazione come servizio era molto popolare alla fine degli anni '60, tuttavia divenne subito chiaro che l'hardware, il software e le tecnologie di telecomunicazione dell'epoca non erano ancora pronte per quella sfida.

Evoluzione tecnologica

Il cloud è definito dalla combinazione di tre concetti fondamentali: il primo consiste nel fornire un servizio su richiesta; il secondo consiste nel permettere a più persone di condividere la stessa risorsa IT; il terzo consiste nell'erogare i servizi tramite la rete [4]. L'evoluzione tecnologica nel campo della virtualizzazione, della rete e delle risorse hardware hanno consentito di fare passi da gigante e di concretizzare quelle che inizialmente erano solo idee. Alla fine degli anni '60 e nei primi '70 IBM sviluppò *VM*, un sistema operativo virtualizzato che consente a più utenti di condividere la stessa risorsa hardware. Nel 1969 venne lanciato dal dipartimento di Difesa americano *ARPANET* (Advanced Research Projects Agency Network), la prima rete geografica basata sul protocollo TCP/IP che fu precorritrice della rete Internet che utilizziamo tutt'oggi. Negli anni '90 le tecnologie fondamentali per il cloud raggiunsero un certo livello di maturità. Nel 1991 la nascita del *World Wide Web* consentì a più di un *milione* di dispositivi di essere connessi ad internet.

Primi riferimenti

Alcuni osservatori del settore ritengono che il primo utilizzo del termine *cloud* sia avvenuto nel 1993, quando General Magic lo usò mentre descriveva Telescript, il suo linguaggio di programmazione distribuito. Tuttavia, l'origine dell'espressione *cloud computing* è oggetto di accesi dibattiti. Nel 1996, un documento interno della *Compaq* chiamato "Internet Solutions Division Strategy for Cloud Computing" descriveva il concetto di cloud computing.

L'inizio

La prima generazione di Cloud venne alla luce nei primi anni 2000.

- Nel 2002, Amazon ha creato l'azienda partecipata Amazon Web Services (AWS), con l'obiettivo di "consentire agli sviluppatori di creare da soli applicazioni innovative ed imprenditoriali".
- Nel marzo del 2006 Amazon ha introdotto Simple Storage Service (S3), seguito da Elastic Compute Cloud (EC2) nell'agosto dello stesso anno. Questi prodotti hanno aperto la strada all'utilizzo della virtualizzazione per fornire IaaS su richiesta e ad un prezzo più conveniente.
- Nell'aprile 2008, Google ha rilasciato la versione beta di Google App Engine, un PaaS (uno dei primi nel suo genere) che forniva un 'infrastruttura completamente gestita e una piattaforma di distribuzione. L'obiettivo era eliminare alcune attività amministrative tipiche del modello IaaS.
- Nel febbraio 2010, Microsoft ha rilasciato Microsoft Azure.
- Nel luglio 2010, Rackspace Hosting e la NASA hanno lanciato congiuntamente un'iniziativa di software cloud open source nota come OpenStack.
- Nel maggio 2012, Google Compute Engine è stato rilasciato in anteprima, prima di essere rilasciato pubblicamente nel dicembre 2013.

1.3 Caratteristiche essenziali

Riprendendo la definizione di *Cloud computing* data dal *NIST*, questo modello è composto da 5 caratteristiche essenziali, tre modelli di servizio (paragrafo 1.4) e quattro modelli di distribuzione (paragrafo 1.5)).

On-demand self-service

Gli utenti possono usufruire delle risorse, come tempo computazionale e spazio di archiviazione, erogate sotto forma di servizio su richiesta senza l'intervento o l'interazione con il fornitore.

Broad network access

Le risorse sono erogate tramite internet e sono accessibili da un gruppo vasto ed eterogeneo di dispositivi grazie all'utilizzo di interfacce standardizzate (*Web service*).

Resource pooling

Le risorse *Cloud*, fisiche e virtuali, sono dinamicamente assegnate e riassegnate in base alla domanda degli utenti. Esse sono gestite in modo da poter essere condivise tra più utenti mediante un modello *multi-tenant*, tale modello permette a più utenti (*tenant*) di condividere una singola istanza di un software. Ogni utente ha la sua vista sull'applicazione che utilizza, amministra e personalizza rimanendo all'oscuro della presenza degli altri *tenant* [5].

Rapid elasticity

La quantità di risorse rilasciate si adatta dinamicamente in base al carico di lavoro richiesto. Agli occhi del consumatore la capacità disponibile appare illimitata in ogni momento.

Measured service

I sistemi *Cloud* sono controllati e monitorati dai fornitori (*provider*). Essi gestiscono l'accesso e l'ottimizzazione delle risorse, garantendone la disponibilità ed erogandole sotto forma di servizio. Gli utenti pagano soltanto per quello che utilizzano, secondo un modello *pay-per-use*.

1.4 Modelli di servizio

Un modello di servizio cloud rappresenta una specifica combinazione preconfezionata di risorse IT fornite da un *cloud provider* [1] e comprende una divisione di responsabilità unica tra l'utilizzatore finale e il fornitore di servizi, come è possibile vedere in [Fig. 1.2]. Il paradigma cloud fornisce tre modelli di servizio, i quali differiscono in base al tipo

[1] Un provider di servizi cloud consiste in una società di terze parti che fornisce servizi di archiviazione, applicazioni, infrastruttura o piattaforma basati sul cloud.

di risorsa remota offerta. Essi sono organizzati in una struttura piramidale a tre livelli. Tale piramide [Fig. 1.1] è detta di Sheehan, dal nome del suo ideatore.

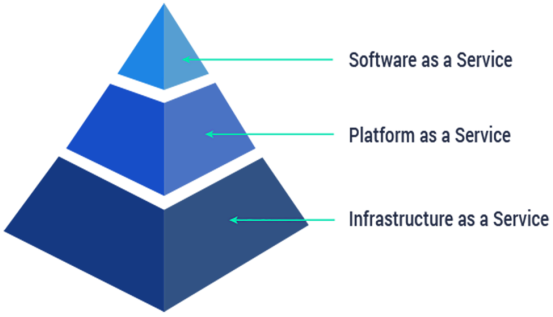

Figura 1.1: *— Piramide di M.Sheehan (Fonte immagine: https://www.extrasys.it/it/redblog/differenza-iaas-paas-saas)*

1.4.1 SaaS - Software as a Service

> *"The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure."*
> *— The NIST definition of cloud computing[2]*

Software as a Service (SaaS) è il modello di servizio cloud che fornisce l'accesso a un prodotto software completo, eseguito e gestito dal *cloud provider*, sotto forma di *servizio su richiesta* accessibile da una vasta gamma di dispositivi eterogenei attraverso una interfaccia standardizzata tipicamente disponibile tramite Web.

L'utente finale non deve preoccuparsi delle risorse hardware richieste per l'esecuzione, dell'installazione e della configurazione del software ma si può concentrare esclusivamente sull'utilizzo dello stesso. Nella maggioranza dei casi, una singola istanza del software viene condivisa tra più utenti secondo il modello *multi-tenant*, pur garantendo la separazione logica tra i dati di ogni utente.

Le applicazioni sono fornite secondo la logica *pay-per-use*, il consumatore dovrà pagare in base a quanto e come utilizza il servizio (in termini di ore, carico computazionale ecc.).

I vantaggi del modello SaaS sono:

- **Costi iniziali più bassi.**
  SaaS utilizza un modello *pay-per-use* e non prevede costi di licenza iniziali. Inoltre il provider SaaS gestisce l'infrastruttura che ospita il software, il che riduce le spese per la manutenzione sia di hardware che di software.

- **Maggiore facilità d'uso.**
  Il consumatore non deve né installare nè configurare il software fornito tramite SaaS, può usufruire del servizio in maniera rapida ed estremamente semplice.

- **Accessibilità.**
  I servizi SaaS sono accessibili tramite Web, l'utente può accedervi con diversi dispositivi, ovunque si trovi (a patto che abbia a disposizione una connessione a Internet).

Alcuni esempi sono Google Apps, Salesforce.com, Microsoft Office 365, Dropbox.

1.4.2 PaaS - Platform as a Service

> *"The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider."*
> *— The NIST definition of cloud computing[2]*

Platform as a Service (PaaS) è il modello di servizio cloud che fornisce una piattaforma di sviluppo software corredata da un insieme di servizi che supportano lo sviluppatore attraverso l'intero processo di produzione software. Permette di eseguire le fasi di progettazione, sviluppo, testing, monitoraggio, distribuzione e hosting esclusivamente su cloud, ottenendo così un ambiente di sviluppo completo e facilmente accessibile tramite la rete. L'utilizzo di un servizio cloud PaaS consente di sviluppare e distribuire applicativi in modo semplice e veloce evitando l'*overhead* dovuto alla configurazione e gestione dell'ambiente di sviluppo.

I vantaggi del modello PaaS sono:[6]

- **Riduzione del tempo per la scrittura del codice.**
  Gli strumenti di sviluppo PaaS aiutano a ridurre il tempo necessario per la scrittura del codice di nuove app, grazie a componenti pre-codificati integrati nella piattaforma, come flusso di lavoro, servizi di directory, strumenti per la sicurezza, ricerca ecc.

- **Uso di strumenti sofisticati ad un prezzo contenuto.**
  Un modello di pagamento in base al consumo consente a utenti singoli o organizzazioni di usare strumenti di analisi, di business intelligence e software di sviluppo sofisticati, che non potrebbero permettersi o che non converrebbe acquistare.

- **Supporto di team di sviluppo distribuiti a livello geografico.**
  Poiché l'accesso all'ambiente di sviluppo avviene tramite Internet, i team di sviluppo possono collaborare ai progetti anche quando i loro membri si trovano in posizioni remote.

Alcuni esempi sono AWS Elastic Beanstalk, Heroku, Google App Engine, Apache Stratos, OpenShift.

1.4.3 IaaS - Infrastructure as a Service

> *"The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer*

> *is able to deploy and run arbitrary software, which can include operating systems and applications."*
> *— The NIST definition of cloud computing[2]*

Infrastructure as a Service (IaaS) è il tipo di modello di servizio cloud più flessibile in quanto dà la possibilità a sviluppatori e amministratori di sistema di poter *noleggiare* componenti hardware (CPU, RAM, storage ecc.) *virtualizzati* con la massima libertà di scelta, il tutto offerto come servizio su cloud. In pratica, un servizio IaaS corrisponde ad una macchina virtuale che l'utente può configurare secondo le proprie esigenze.

Le risorse IaaS prese in affitto si possono scalare in qualsiasi momento, il *cloud consumer* paga solo per le risorse che utilizza e può adattare la spesa in base alle sue necessità. Questa elevata flessibilità deriva dal fatto che le risorse non sono legate ad un hardware specifico, bensì sono risorse *virtualizzate*.

IaaS segue il *principio di shared responsability* (principio di responsabilità condivisa). Sia il consumatore che il provider hanno delle responsabilità ben precise. In particolare, il provider IaaS deve occuparsi della manutenzione, del continuo ammodernamento e della sicurezza dell'infrastruttura IT; deve mettere a disposizione le risorse di calcolo (CPU, RAM), di archiviazione e di rete; deve fornire un ambiente virtualizzato attraverso il quale i clienti possono accedere alle risorse IaaS e software per l'amministrazione dell'infrastruttura IT. Il consumatore, invece, ha il compito di scegliere i componenti virtualizzati che vuole utilizzare; deve occuparsi della installazione, configurazione e aggiornamento dei software di gestione dell'infrastruttura; In [Fig. 1.2] viene illustrata la suddivisione di responsabilità, in base al modello di servizio scelto. Alcuni esempi sono Amazon Elastic Compute Cloud (EC2), Openstack e VMware vCloud Hybrid Service.

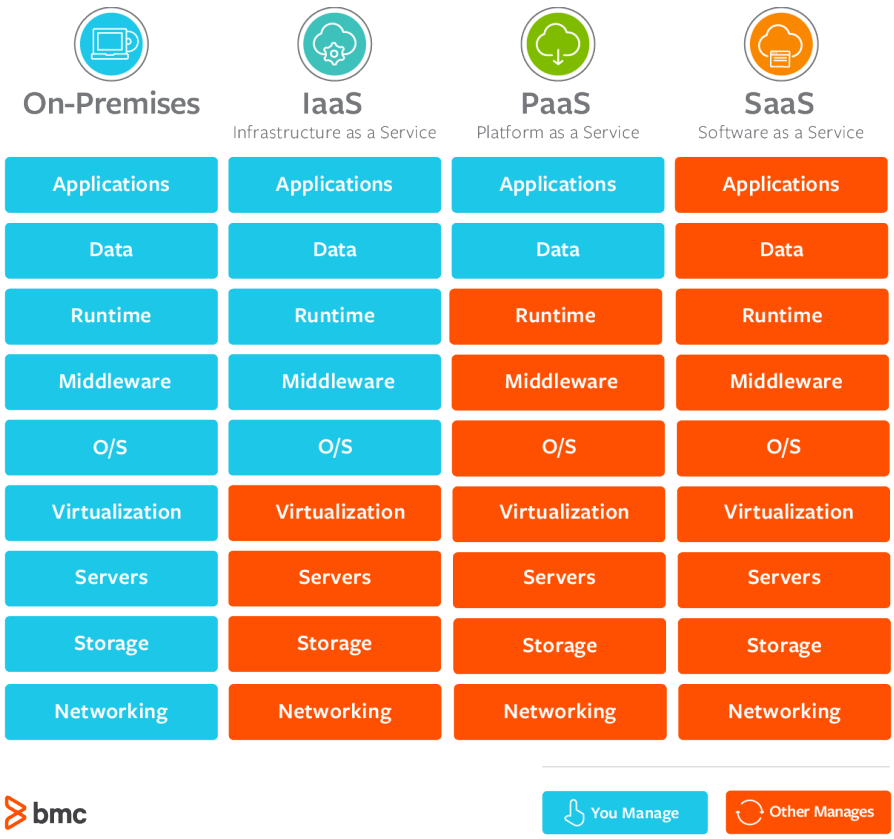

Figura 1.2: *— Suddivisione delle responsabilità di gestione per i modelli di servizi di cloud computing (Fonte immagine: bmc.com)*

1.5 Modelli di distribuzione

Un modello di distribuzione definisce dove risiede l'infrastruttura e chi ne possiede il controllo. Ogni modello di distribuzione soddisfa differenti esigenze, per un'azienda è quindi fondamentale fare una analisi approfondita di quelle che sono le proprie necessità prima di sceglierne uno. Inoltre differenti modelli di distribuzione presentano differenti costi, pertanto in molti casi la scelta dipende o è comunque influenzata dal budget a disposizione. In ogni caso è necessario conoscere le caratteristiche, gli obiettivi, e i problemi del dominio di interesse in modo da prendere una decisione consapevole.

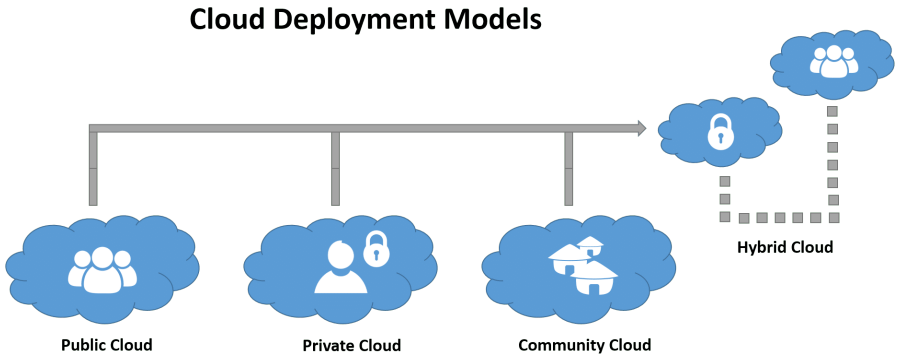

Figura 1.3: *— Modelli di distribuzione cloud (Fonte immagine: uniprint.net)*

Seguendo la definizione data dal *NIST* [2], i modelli di distribuzione sono quattro [fig: 1.3] e sono classificati come segue.

1.5.1 Public cloud

Il cloud pubblico, detto anche *external cloud*, è reso disponibile per utenti singoli o aziende. Esso mette a disposizione le risorse come servizi su richiesta per il pubblico utilizzo, seguendo il modello *pay-as-you-go*. Offre inoltre una rapida implementazione e una scalabilità apparentemente infinita dovuta alle immense infrastrutture IT che colossi come *Google, Microsoft e Amazon* mettono a disposizione. L'infrastruttura IT è di competenza di un ente terzo (*cloud provider*), il quale si prende carico di tutti i rischi e le problematiche derivanti dalla manutenzione e gestione dell'infrastruttura. L'utilizzo di un cloud pubblico comporta una riduzione dei costi iniziali per l'acquisto, la gestione e la manutenzione dell'infrastruttura hardware e software in loco, ed inoltre una maggiore facilità di utilizzo delle risorse.

Tuttavia, desta preoccupazione in molti il fatto che la gestione dei dati è delegata al provider del servizio cloud. Questo si traduce in una minore garanzia di sicurezza e privacy dei dati, aspetto che per molte organizzazioni è di cruciale importanza.

1.5.2 Private cloud

Il cloud privato, detto anche *internal cloud*, consiste nella fornitura di servizi IT accessibili esclusivamente da una singola organizzazione, o da un gruppo di organizzazioni, tramite Internet o una rete interna privata. L'infrastruttura in questo caso è gestita direttamente da tali organizzazioni o da enti di terze parti con accordi di erogazione *esclusivi* con queste. A differenza del cloud pubblico, si ha un maggiore controllo e una maggiore personalizzazione dei servizi. Il che si traduce anche in un livello di sicurezza più elevato. Ciononostante, questa maggiore libertà ha un costo, in quanto sarà l'organizzazione stessa a doversi curare della gestione, manutenzione e configurazione del cloud privato.

1.5.3 Hybrid cloud

Il cloud ibrido consiste nella fornitura di servizi IT attraverso una combinazioni di cloud pubblici e cloud privati, i quali rimangono entità *distinte*. La comunicazione tra entità differenti è ottenuta mediante l'utilizzo di servizi che garantiscono la portabilità di applicazioni e dati. Il cloud ibrido consente di soddisfare le fluttuazioni della domanda di elaborazione dei dati grazie all'utilizzo delle risorse del cloud pubblico per attività non sensibili, ovvero che non richiedono vincoli stringenti per quanto riguarda la privacy e la sicurezza dei dati, mantenendo al contempo le applicazioni critiche per l'organizzazione al sicuro nel cloud privato. Un modello ibrido, utilizzato in modo efficace, offre il meglio di entrambi i modelli (pubblico e privato), garantendo tutti i vantaggi del cloud computing: flessibilità, scalabilità e riduzione dei costi.

1.5.4 Community cloud

Il *community cloud* è una infrastruttura che è reciprocamente condivisa tra un gruppo di organizzazioni che appartengono ad una particolare *comunità* e che condividono obiettivi, requisiti di sicurezza, policy e considerazioni sulla conformità. Esso può essere gestito da una o più organizzazioni facenti parte della comunità, da un ente terzo o da una combinazione di queste.

1.6 Architettura e tecnologie Cloud

I cloud moderni sono sostenuti da una serie di strumenti tecnologici che abilitano quelle che sono le caratteristiche principali associate al *cloud computing*. Seguendo la trattazione di *Thomas Erl* nel libro *Cloud Computing: Concepts, Technology & Architecture* [7] in questo paragrafo saranno descritte alcuni di questi importanti componenti dell'architettura cloud.

1.6.1 Data center

Raggruppare le risorse IT in stretta prossimità tra loro, anziché disperderle geograficamente, consente la condivisione dell'alimentazione, una maggiore efficienza nell'utilizzo delle risorse condivise e una maggiore accessibilità per il personale IT. Questi sono i vantaggi che hanno reso popolare il concetto di *data center*. I data center moderni consistono in una infrastruttura IT specializzata utilizzata per ospitare risorse IT centralizzate, come server, database, dispositivi di rete e di telecomunicazione e sistemi software.[7]

Di seguito sono descritti alcune tecnologie e componenti che generalmente compongono i data center.

Virtualizzazione

Come mostrato in [fig: 1.4], i data center sono composti da risorse IT fisiche e da risorse virtualizzate. Il livello fisico si riferisce all'infrastruttura che ospita sistemi e apparec-

chiature di elaborazione e di rete, insieme a sistemi hardware ed i loro sistemi operativi. Il livello composto da risorse virtualizzate è basato su piattaforme di virtualizzazione che astraggono l'elaborazione su risorse fisiche e il networking come componenti virtualizzati più facili da allocare, rilasciare, monitorare e gestire.

L'argomento sarà approfondito nel paragrafo 1.6.2.

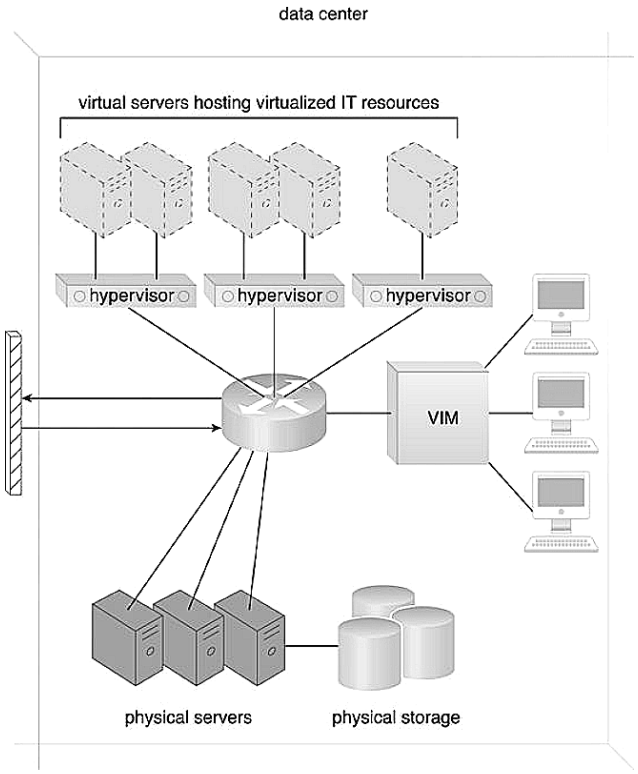

Figura 1.4: *— I componenti di un data center che lavorano insieme per fornire risorse IT virtualizzate (Fonte immagine: Erl T. - Cloud Computing: Concepts, Technology & Architecture)*

Modularità e standardizzazione

I data center sono basati su hardware standardizzato e progettati con architetture modulari che aggregano più elementi costitutivi per supportare scalabilità, espansione, e sostituzione veloce dell'hardware. Modularità e standardizzazione sono requisiti chiave per ridurre gli investimenti e i costi di gestione in quanto consentono di attuare economie di scala per i processi di approvvigionamento, distribuzione, funzionamento e manutenzione delle risorse. Inoltre l'uso di strategie di virtualizzazione comuni favoriscono il *consolidamento*. Consolidare una infrastruttura IT significa ridurre il numero di macchine fisiche in utilizzo, lasciando invariate le funzionalità offerte grazie alla vir-

tualizzazione delle risorse IT. Le risorse IT consolidate possono servire diversi sistemi ed essere condivise tra diversi consumatori di cloud.

Automatizzazione

I data center dispongono di piattaforme specializzate nell'automatizzare attività come il *provisioning* [2], monitoraggio e *patching*.

Elevata disponibilità

Dato che un qualsiasi malfunzionamento del data center ha un impatto significativo sulla *business continuity* [3] per le organizzazioni che utilizzano i loro servizi, i data center sono progettati per funzionare con livelli di ridondanza sempre più elevata per garantire un alto livello di affidabilità. Tipicamente, i data center utilizzano sistemi di alimentazione, cablaggio e sottosistemi per il controllo ambientale per prevenire i guasti, oltre a collegamenti e hardware in cluster per il bilanciamento del carico.

1.6.2 Virtualizzazione

La virtualizzazione è una degli aspetti fondamentali nel cloud computing. È proprio grazie a questa tecnologia che l'utente finale ha l'impressione di utilizzare un ambiente in modo esclusivo, con risorse apparentemente inesauribili.

La virtualizzazione utilizza il software per creare un livello di astrazione sull'hardware, questo consente di suddividere le risorse hardware di una singola macchina fisica in più componenti virtuali, comunemente chiamate macchine virtuali (VM). La [fig: 1.5] mostra un esempio di questa architettura.

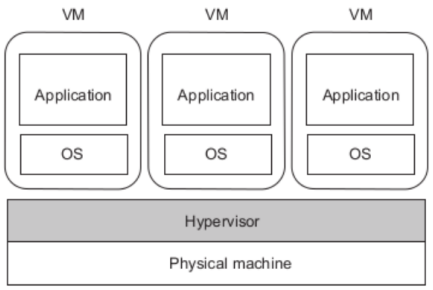

Figura 1.5: *— Più virtual machine su un'unica macchina fisica*

Ogni VM può eseguire un suo sistema operativo e comportarsi in modo indipendente dalle altre VM. [9] Una VM non può interagire direttamente con il livello fisico, bensì

[2] Con provisioning si intende il processo di configurazione di un'infrastruttura IT, ma lo stesso termine è utilizzato per definire anche le procedure necessarie per gestire l'accesso ai dati e alle risorse e per renderle disponibili a utenti e sistemi. [8]

[3] Per business continuity (continuità operativa) si intende la capacità di un'organizzazione di continuare a erogare prodotti o servizi a livelli predefiniti accettabili a seguito di un incidente.

ha bisogno di un livello software aggiuntivo chiamato *hypervisor* che svolge il ruolo di intermediario. L'hypervisor inganna il sistema operativo facendogli credere che abbia un accesso diretto alle risorse hardware.

Indipendenza dall'hardware

In un ambiente non virtualizzato, la configurazione di sistemi operativi e software applicativo per modelli di hardware specifici si traduce in una serie di dipendenze tra hardware e software. Se le risorse hardware vengono modificate, è necessario riconfigurare il sistema.

La virtualizzazione consente di convertire delle specifiche risorse hardware in risorse virtualizzate e standardizzate, il chè garantisce l'indipendenza del software dall'hardware.

Consolidamento dei server

La tecnologia di virtualizzazione consente a server virtuali di condividere un unico server fisico. Al fine di massimizzare l'utilizzazione di quest'ultimo si esegue il processo di *consolidamento*, sfruttando l'indipendenza delle risorse virtuali dall'hardware esse si possono far migrare da un host all'altro in modo semplice e flessibile.

1.7 Fattori economici

Uno dei fattori che ha determinato l'esplosione dei sistemi cloud negli ultimi anni è sicuramente quello economico. L'approccio cloud porta una riduzione dei costi attraverso economia di scala, distribuzione dei costi su un ampio bacino di utenti, centralizzazione delle infrastrutture in aree con costi inferiori, ottimizzazione delle risorse e aumento dei tassi di utilizzo delle stesse [10]. Inoltre il cloud rappresenta una attrattiva economica anche per i clienti in quanto consente di convertire le spese in conto capitale (*CapEx*) in spese operative (*OpEx*). In pratica permette agli utilizzatori del cloud (aziende, organizzazioni, *end-user*) di non dover essere costretti ad investire un grosso capitale per acquistare risorse IT ma piuttosto noleggiare tali risorse. Questo permette di ridurre i costi iniziali e consente di reindirizzare il capitale verso gli investimenti riguardanti il core business. [11]
Un altro aspetto rilevante del Cloud Computing riguarda i benefici economici dati dalla sua flessibilità e dal cosiddetto *trasferimento del rischio*, in particolare i rischi di *over-provisioning* e *under-provisioning*.

Per quanto riguarda la flessibilità: utilizzando un sistema Cloud si ha la capacità di aggiungere o rimuovere risorse con una granularità elevata ed in tempi molto rapidi (minuti), consentendo di adattare al meglio le risorse al carico di lavoro effettivo.

Per quanto riguarda il trasferimento del rischio, bisogna considerare che per molti servizi il picco del carico di lavoro supera di un fattore che va da 2 a 10 volte quella che è la media. Ciò vuol dire che investire in risorse cercando di prevedere il picco si traduce in un notevole spreco, come si può vedere in [fig: 1.6]. D'altro canto, se si

sottostima il picco si rischia di non riuscire a fornire adeguatamente il servizio richiesto, il chè comporta impatti negativi sui guadagni. Come si può vedere in [fig:1.7].

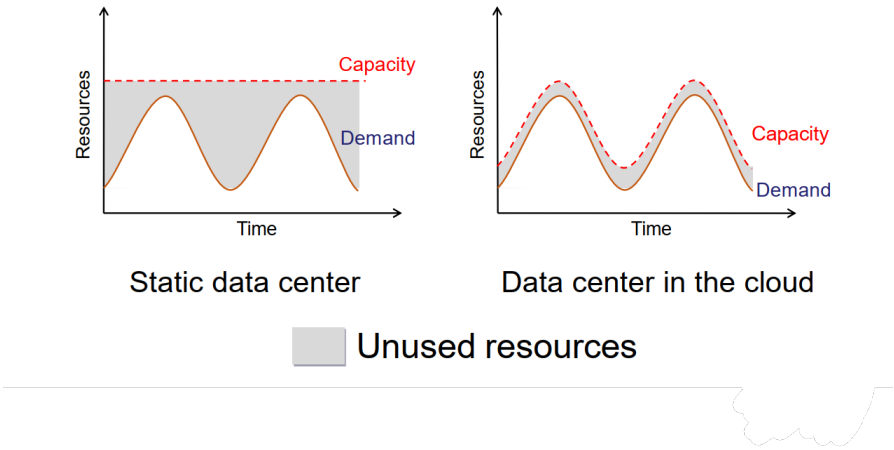

Figura 1.6: *— Previsione del picco Vs Modello pay-as-you-go (Fonte immagine: Above the clouds: A Berkeley View of Cloud Computing by UC Berkeley RAD Lab).*

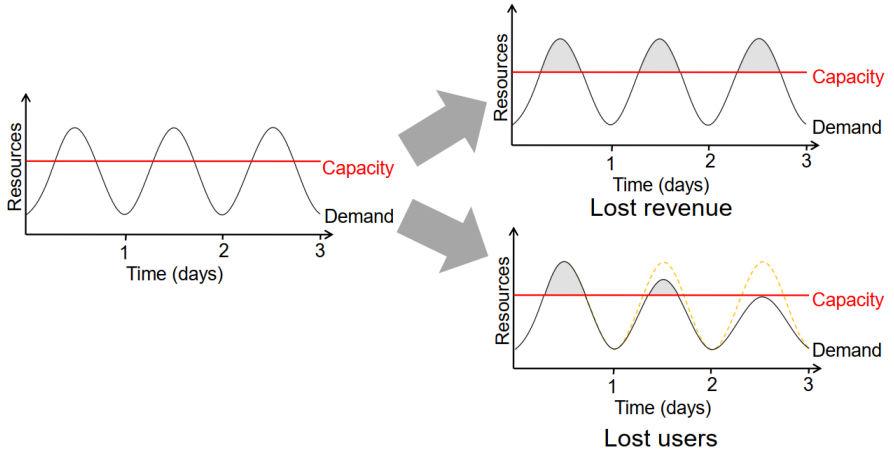

Figura 1.7: *— Underprovisioning (Fonte immagine: Above the clouds: A Berkeley View of Cloud Computing by UC Berkeley RAD Lab).*

CAPITOLO 2

Internet of Things

L'Internet of Things (IoT) rappresenta una delle tecnologie più dirompenti dell'ultimo decennio ed è alla base di innumerevoli scenari di elaborazione pervasiva. Miliardi di nodi (*things*) intelligenti, auto-configuranti e interconnessi tramite una rete dinamica e globale comunicano tra di loro, si interfacciano con il mondo reale e raccolgono un'enorme quantità di dati. Tali nodi possono appartenere ad un'ampia gamma di dispositivi tecnologici: dai comuni laptop e smartphone fino a piccoli dispositivi, come sensori, microcontrollori con spazio di archiviazione e capacità di elaborazione limitati.

In questo capitolo verrà presentato il paradigma IoT, verranno discusse le differenti definizioni che sono state date nel corso degli anni, le tecnologie che ne hanno permesso la diffusione, i vantaggi che può portare ed alcuni scenari applicativi. Infine, saranno analizzati i modelli ed i protocolli di comunicazione che descrivono il modo in cui è possibile interconnettere i diversi dispositivi tra loro.

2.1 Definizioni: un paradigma, diverse visioni

Come spesso accade in ambito informatico, non c'è una definizione universalmente riconosciuta per il termine Internet of Things, il quale racchiude un vasto repertorio di tecnologie, applicazioni e casi d'uso. Di seguito vengono riportate alcune definizioni:

Internet Architecture Board (IAB) - RFC 7452 *"Architectural Considerations in Smart Object Networking"* [12].
Il termine Internet of Things" denota una tendenza in cui un grande numero di dispositivi embedded impiegano i servizi offerti dai protocolli internet. Molti di questi dispositivi, spesso sono chiamati smart objects", non sono direttamente gestiti da essere umani, ma esistono come componenti in edifici o veicoli o sparsi nell'ambiente.

Internet Engineering Task Force (IETF) - *"The Internet of Things at the IETF"* [13].
Il termine Internet of Things fa riferimento ad una rete di oggetti fisici o *things* integrati con elettronica, software, sensori, attuatori e connettività per consentire agli oggetti di scambiare dati con operatori o altri dispositivi connessi.

Internet Telecommunication Union (ITU) - *"Overview of the Internet of Things"* [14].
Internet of Things: una infrastruttura globale per la società dell'informazione, che abilità servizi avanzati per l'interconnessione (fisica e virtuale) di oggetti in base a tecnologie dell'informazione o di comunicazione interoperabile esistenti o in evoluzione.

Un aspetto comune tra la diverse definizioni è il concetto di connettività. Alcune definizioni si focalizzano su una visione *Things-oriented*, parlando di interconnettività tra dispositivi ma senza legare in modo esplicito IoT ad Internet. Altre, invece, sono *Internet-oriented* e vedono Internet come l'elemento principale di IoT [15]. Non per questo le diverse definizioni sono in disaccordo, ma sottolineano diversi aspetti del paradigma IoT.

Si può concludere che, quando si parla di Internet of Things, si fa riferimento ad un insieme di dispositivi fisici eterogenei, interconnessi, più o meno intelligenti, che una volta connessi a Internet sono capaci di collezionare e trasferire dati con un intervento manuale limitato o del tutto assente.

2.2 Cenni storici

La concezione dell'Internet of Things risale al 1982, quando all'Università Carnegie Mellon di Pittsburgh un distributore di Coca Cola modificato appositamente fu il primo apparecchio ad essere connesso ad *ARPANET*, in grado di comunicare informazioni riguardo la disponibilità delle bevande. Il termine *Internet of Things*, letteralmente "Internet delle cose", fu usato per la prima volta da *Kevin Ashton*, un imprenditore britannico, in un articolo alla RFID Journal nel 1999:

> *"We're physical, and so is our environment. Our economy, society and survival aren't based on ideas or information – they're based on things. Yet today's information technology is so dependent on data originated by people that our computers know more about ideas than things. If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so."*
> — *Kevin Ashton al RFID Journal - 1999 [16]*

Ashton, nell'articolo, evidenzia il fatto che le tecnologie sviluppate fino ad allora erano basate esclusivamente sui dati generati dalle persone, quindi tecnologie con una limitata dinamicità ed interazione con il mondo fisico che non sfruttano a pieno le potenzialità date dall'avvento di Internet. Ashton mostra una nuova prospettiva, dove gli oggetti, una volta connessi alla rete, sono in grado di fornirci informazioni sul loro stato e sul mondo fisico che li circonda.

In quel periodo, Ashton, stava lavorando alla tecnologia RFID (*Radio Frequency Identification*), un nuovo modo di tracciare oggetti in modo automatico e catturare dati tramite radiofrequenze e grazie all'utilizzo di due componenti: tag e *reader*. I *reader*, o lettori, sono dispositivi che possiedono uno o più antenne e che emettono onde radio e ricevono un segnale di ritorno da parte dei *tag*. I tag, sono dispositivi elettronici *passivi* o *attivi*, capaci di rispondere a segnali di prossimità inviati dai *reader* [17]. Questa tecnologia fu alla base dello sviluppo dello standard EPCglobal network, utilizzato per condividere tra partner commerciali dati dinamici sui singoli prodotti, come il loro stato e la loro posizione.

La nascita vera e propria dell'IoT, secondo *Cisco Internet Business Solutions Group*, risale al periodo 2008-2009 [18]. Ovvero quando il numero di dispositivi (*things*) connessi a Internet ha superato quello delle persone.

2.3 IoT: una tecnologia dirompente

In questo paragrafo si illustreranno alcuni dei vantaggi che il paradigma IoT può portare, le tecnologie che hanno permesso all'Internet of Things di essere così popolare oggi ed alcuni scenari applicativi.

2.3.1 Vantaggi

- **Possibilità di tracciare e monitorare oggetti.** L'IoT permette di monitorare una miriade di oggetti eterogenei e tenere traccia, in tempo reale, della loro posizione e del loro stato.

- **Pervasività.** Una caratteristica fondamentale dell'IoT è quella di integrare gli oggetti che si utilizzano tutti i giorni con capacità di elaborazione, che permettono di svolgere attività utili e che ci semplificano la vita, minimizzando l'interazione tra utente e calcolatore. Al contrario di un classico computer Desktop, l'elaborazione pervasiva può essere utilizzata in ogni momento, in ogni luogo ed ha come obiettivo quello di sviluppare dispositivi intelligenti che si adattano al contesto in cui vengono calati e che migliorano l'esperienza di vita degli utenti nella vita di tutti i giorni.

- **Più dati, migliori decisioni.** L'Internet of Things consiste in miliardi di nodi interconnessi, che si interfacciano con il mondo reale e interagendo con esso raccolgono un'enorme quantità di dati. L'ingente volume di dati catturato da sensori e dispositivi IoT può essere utilizzato per ottenere analisi approfondite e quindi guidare il processo decisionale e operativo delle aziende.

- **Automazione.** L'Internet of Things consente di portare l'automazione nei processi produttivi ad un altro livello ed è alla base del nuovo sviluppo industriale, chiamato *industria 4.0*. Con il termine *industria 4.0* si fa riferimento all'uso di strumenti di automazione avanzati, che integrano nuove tecnologie come IoT, intelligenza artificiale, *Big Data analytics* durante le fasi del processo produttivo. Le *Smart factory* consentono di aumentare l'efficienza della produzione, permettono una forte personalizzazione dei prodotti in base alle esigenze dei clienti, la tracciabilità dei prodotti lungo tutto il loro percorso dalla materia prima alla consegna, una maggiore efficienza energetica con una conseguente diminuzione dei costi ed un'accelerazione dei tempi decisionali grazie alla costante disponibilità di informazioni [19].

2.3.2 Tecnologie abilitanti

- **Miniaturizzazione.** I progressi ottenuti negli ultimi anni nel campo dell'elettronica hanno permesso di produrre dispositivi sempre più piccoli e con un consumo di energia sempre più basso, grazie a nuove tecnologie di microfabbricazione avanzata. L'avvento di questi nuovi dispositivi consente di estendere il paradigma IoT verso nuovi orizzonti, come l'Internet of Medical Things (IoMT) che consiste nello sviluppo di soluzioni IoT per il supporto clinico, analisi di dati sanitari, telemedicina e molto altro.

- **Ubiquità connessione.** Nel corso degli anni il forte interesse verso l'IoT ha portato allo sviluppo di numerose tecnologie e protocolli di comunicazione, sia via cavo che in modalità wireless, come Wi-Fi, Bluetooth, BLE (Bluetooth Low Energy), LPWAN (Low Power Wide Area Network) e 5G. Questo fermento ha reso possibile l'implementazione di soluzioni IoT nei contesti più disparati.

- **Diffusa adozione del protocollo IP.** La diffusa adozione di IP ha consentito di sviluppare software e strumenti che possono essere utilizzati da una vasta gamma di dispositivi in modo del tutto omogeneo.

- **Ascesa del Cloud computing.** Il Cloud computing permette di processare, filtrare, archiviare e analizzare l'ingente quantità di dati generati dai dispositivi IoT grazie a piattaforme flessibili e modulari, accessibili in remoto.

Per tutti questi motivi l'IoT sta diventando sempre più popolare, tanto da essere definita già nel 2008 come una delle 6 tecnologie più diromponenti dal *US National Intelligence Council* [20]. Come si può vedere in figura [fig: 2.1], negli ultimi anni i dispositivi IoT connessi alla rete sono aumentati in maniera vertiginosa con un tasso di crescita del 26% dal 2015 al 2020, diminuito al 9% solo nell'ultimo anno a causa dell'impatto che la pandemia di COVID-19 ha causato sia sulla domanda che sull'offerta. Spesso, infatti, la produzione è stata interrotta e numerosi sono stati i problemi sulla catena di approvvigionamento e l'accesso alle materie prime per la produzione dei chip. Nonostante

questo, secondo gli analisti, nei prossimi anni si tornerà almeno al tasso di crescita pre-pandemico [21].

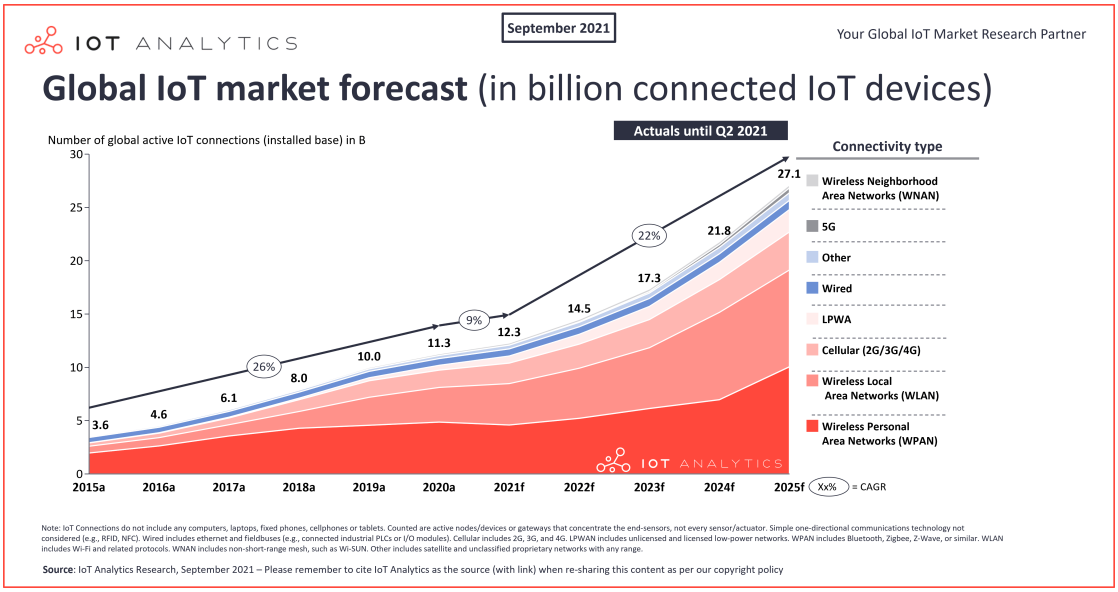

Figura 2.1: *— Previsione del numero di dispositivi IoT connessi (in miliardi) (Fonte immagine: [21])*

2.3.3 Scenari applicativi

La visione data dall'Internet of Things di oggetti di diversa fattura e diverse capacità computazionali, interconnessi e *Internet-connessi* in una rete dinamica e globale ha dato spazio ad una serie di innovazioni nel campo ICT (*Information and Communication Technology*) ed ha aperto a numerosi scenari applicativi in una vasta gamma di settori, come quello dell'*automotive*, sanitario, industriale, dell'agricoltura, dei negozi al dettaglio, dell'elettronica di consumo e molti altri.

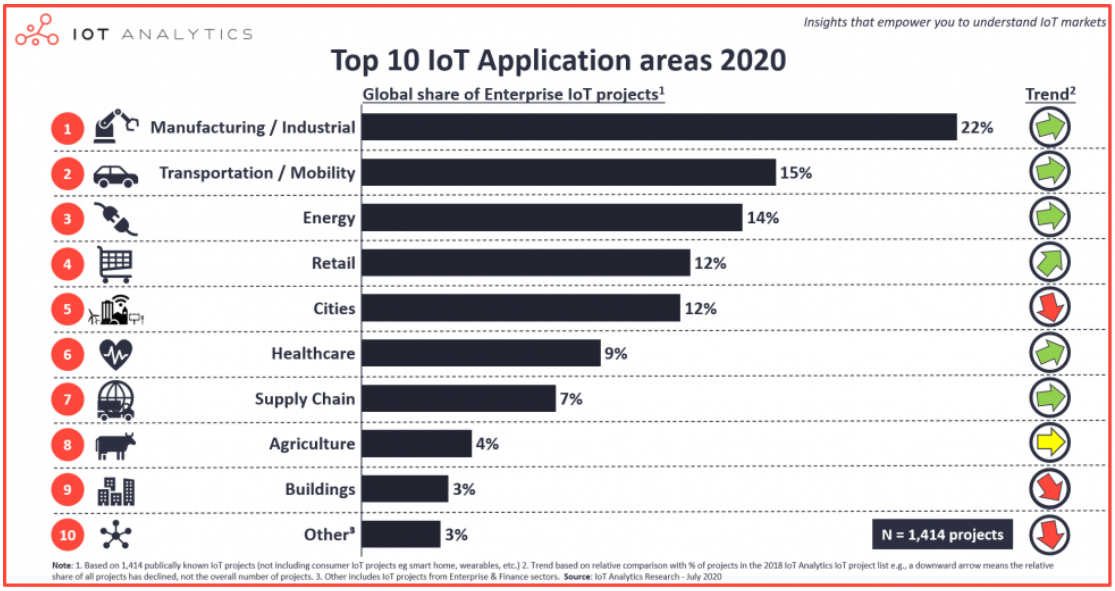

Figura 2.2: *— Top 10 IoT Application areas 2020 (Fonte immagine: [22])*

Nel report *Top 10 IoT Applications in 2020* di *IoT Analytics* [22] vengono illustrati i 10 scenari applicativi più diffusi, in base a 1,414 progetti IoT per *aziende*. (Lo studio esclude quindi i progetti IoT orientati verso il cliente, o *consumer-focused*, come *Smart Home* o dispositivi *Wearable*). Dall'analisi dei progetti, si evidenzia che la maggiorparte (22%) sono in ambito industriale o manifatturiero, con settori come trasporti e mobilità, energia, vendita al dettaglio e sanità in crescita rispetto alle analisi precedenti del 2018. Come si può vedere in [fig: 2.2].

> *"Industrial IoT is transforming the rules of manufacturing, fueling cloud and edge innovation, accelerating the evolution of digital factories, and enhancing operational performance."*
> *— Satya Nadella, CEO di Microsoft, Nov 2019 [22]*

Industrial Internet of Things L'IIoT (*Industrial Internet of Things*) copre una vasta gamma di progetti sia all'interno che all'esterno della fabbrica. Ad esempio, all'interno, sono stati sviluppati e sono in corso di sviluppo molti progetti di automazione, controllo del processo produttivo e sistemi di controllo della qualità basati su IoT. Mentre, all'esterno, i progetti includono il controllo remoto di apparecchiature e macchinari, la gestione e il controllo di intere operazioni industriali da remoto come ad esempio le piattaforme petrolifere. Molti dei casi di studio indicano che l'introduzione di una soluzione IoT comporta una riduzione dei tempi di inattività e un risparmio sui costi. Secondo S. Nadella (CEO di Microsoft), l'Industrial IoT sta trasformando i processi produttivi, accelerando l'evoluzione delle fabbriche verso il digitale e migliorando le prestazioni operative.

Trasporti, mobilità ed automotive Il settore che racchiude trasporti, mobilità ed automotive si trova al secondo posto tra le maggiori aree di applicazioni dell'IoT, nel 2020. Le applicazioni tipiche includono soluzioni per la diagnostica ed il monitoraggio, come il monitoraggio della batteria, della pressione degli pneumatici, del conducente e del veicolo. In fase di sviluppo ci sono invece soluzioni più avanzate come le auto a guida automatica, la comunicazione *vehicle-to-vehicle* (V2V) e *vehicle-to-everything* (V2X) per abilitare la comunicazione tra veicoli e una qualsiasi entità intelligente che può influenzare il veicolo e viceversa in modo da aumentare la sicurezza a bordo e ridurre il numero di incidenti.

Energia Il settore energetico rappresenta l'11% dei progetti analizzati. La maggior parte dei progetti si concentra sulla distribuzione dell'energia, l'ottimizzazione della rete, il monitoraggio e la gestione remota delle risorse, e la manutenzione predittiva. Un esempio applicativo è dato da Enel, multinazionale italiana dell'energia e uno dei principali operatori globali nei settori dell'energia elettrica e gas. Per migliorare l'affidabilità della rete e ridurre i guasti, Enel, utilizza l'intelligenza artificiale per analizzare i dati dei sensori sparsi lungo tutta la sua rete di distribuzione in tempo reale. Enel, per implementare la sua soluzione IoT si è affidata ad AWS (Amazon Web Service), migrando l'80% della sua infrastruttura su Cloud, questo gli ha permesso di ridurre del 60% i costi di storage e del 20% i costi di elaborazione [23].

Retail Sempre più rivenditori riconoscono che grazie allo sviluppo di soluzioni IoT innovative è possibile migliorare l'esperienza dei clienti in negozio, aumentare l'efficienza e diminuire i costi. C'è, infatti, un crescente interesse verso la digitalizzazione dei negozi: la vendita al dettaglio ora rappresenta il 9% dei progetti identificati, rispetto al 5% nell'analisi del 2018. Le soluzioni IoT tipiche includono il monitoraggio e il coinvolgimento del cliente all'interno del negozio, gestione dell'inventario, distributori e casse automatiche intelligenti. Un esempio è Amazon Go. Avviato nel 2016 per i dipendenti, e nel 2018 per il pubblico, Amazon Go è il sistema di vendita al dettaglio *cashierless* più avanzato di sempre, il quale elimina dal processo di acquisto la fase di check-out secondo l'idea di *Just Walk Out Shopping*. Tutto quello che bisogna fare è scannerizzare con uno smartphone un QR code all'entrata, quindi è possibile prendere i prodotti che si vogliono acquistare dagli scaffali e semplicemente uscire. La fase di check-out viene gestita in maniera automatica grazie all'utilizzo di deep learning, *computer vision*, e sensori [24].

2.4 Modelli di comunicazione

Da un punto di vista operativo, è utile vedere come è possibile connettere i dispositivi IoT tra loro e verso altre entità. La seguente discussione presenta questo quadro. In particolare vengono illustrati i diversi modelli di comunicazione e se ne descrivono le caratteristiche chiave, in accordo con il documento *Architectural Considerations in Smart Object Networking* rilasciato dall'*Internet Architecture Board* [12].

2.4.1 Modello Device-to-Device

Il modello di comunicazione *device-to-device* descrive la comunicazione tra due o più dispositivi IoT che si collegano e comunicano direttamente tra loro, piuttosto che tramite un server intermedio. Questi dispositivi comunicano su molti tipi di reti, comprese le reti IP o Internet, utilizzando protocolli come Bluetooth, WiFi, ZigBee, 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks*) che operano a livello fisico o datalink per stabilire comunicazioni dirette da dispositivo a dispositivo.

Questo modello è tipicamente utilizzato da applicazioni IoT dove lo scambio di informazioni è relativamente basso, ovvero dove vengono scambiati pacchetti di piccola dimensione. Ad esempio in sistemi di automazione domestica, composti da lampadine, interruttori, termostati e serrature *smart*. Queste reti permettono ad un insieme di dispositivi, che *condividono* un particolare protocollo di comunicazione, di scambiarsi messaggi. Questo si traduce in una limitata possibilità di scelta da parte dell'utente, che si trova spesso costretto a dover scegliere una particolare famiglia di dispositivi [25].

2.4.2 Modello Device-to-Cloud

In un modello di comunicazione *device-to-cloud*, un dispositivo IoT si collega direttamente ad una piattaforma Cloud che fornisce meccanismi per lo scambio di dati, strumenti per il controllo e la gestione dei dispositivi, servizi per l'archiviazione e l'analisi dei dati. Questo approccio sfrutta protocolli di comunicazione standard come Ethernet e WiFi per stabilire una connessione tra il dispositivo e la rete IP, che permette la connessione al servizio Cloud. Il modello *device-to-cloud* estende le capacità di elaborazione e archiviazione del dispositivo grazie all'integrazione con il Cloud, questo approccio sarà approfondito nel capitolo 3.

2.4.3 Modello Device-to-Gateway

In un modello di comunicazione *device-to-gateway*, un generico dispositivo IoT non viene connesso direttamente ad una piattaforma IoT (ad esempio una piattaforma Cloud), bensì si introduce un nuovo nodo, chiamato *gateway*, che svolge il ruolo di intermediario tra dispositivo IoT e piattaforma. Una soluzione *device-to-gateway* può essere implementata per risolvere problemi di interoperabilità tra dispositivi, gestire una famiglia di dispositivi utilizzando il gateway come *hub* o connettere dispositivi che non hanno la capacità nativa di connettersi direttamente ad un servizio cloud. Ad esempio, questo modello, è utilizzato in articoli di consumo come i *fitness tracker* dove lo smartphone funge da dispositivo gateway per inviare i dati verso il Cloud. Inoltre, un gateway si occupa di garantire una comunicazione sicura tra i dispositivi IoT e la piattaforma utilizzata, può eseguire una prima elaborazione dei dati, limitando il consumo di banda ed evitando sovraccaricare la piattaforma IoT inviando dati già filtrati, o può essere utilizzato per eseguire elaborazioni complesse, come inferenza di Machine learning e analisi in tempo reale direttamente in locale garantendo bassi tempi di latenza.

In conclusione, le funzionalità fornite dai dispositivi gateway possono essere diverse e la scelta di gateway più o meno *intelligenti* dipende dallo scenario applicativo.

2.5 Protocolli di comunicazione

L'Internet of Things, come detto precedentemente, è una tecnologia che si sta espandendo in modo estremamente veloce. Ogni anno miliardi di dispositivi sono distribuiti e contribuiscono alla crescita del volume di dati scambiato all'interno delle rete globale. Quindi, conoscere le politiche che descrivono come la comunicazione tra dispositivi deve essere implementata può essere di enorme importanza per fare la giusta scelta in fase di progettazione, in modo da limitare l'overhead di comunicazione. I protocolli di comunicazione per IoT possono essere classificati in due categorie, in base al livello della pila protocollare ISO/OSI in cui agiscono:

- *IoT Data protocols*: protocolli di comunicazione a livello applicazione e presentazione, come MQTT, AMQP, CoAP e HTTP.
- *Network protocols for IoT*: protocolli di comunicazione che operano a livello fisico e datalink, come Bluetooth, WiFi, ZigBee.

In questa tesi l'attenzione sarà focalizzata su alcuni dei protocolli a livello applicazione e ad alcuni standard di uso comune. I protocolli di comunicazione a livello applicazione si possono suddividere a loro volta in due macrocategorie, secondo il paradigma utilizzato: request-response (*pull-based*) e publish-subscribe (*push-based*).

- **Request-response.** I protocolli di tipo request-response seguono il paradigma client-server. Un dispositivo *client* invia una richiesta al *server* e resta in attesa di una risposta. Il *server* provvederà quindi a rispondere fornendo i dati o i servizi richiesti.
- **Publish-subscribe.** Nell'architettura publish-subscribe, o pub-sub, una risorsa centrale chiamata *broker* si occupa di ricevere e distribuire tutti i dati. I client pub-sub possono inviare un messaggio al broker per *pubblicare* dei dati o per richiedere l'*iscrizione* ad un determinato *topic* (argomento) per ricevere dati. Il broker, quindi, non si occupa di memorizzare i dati ma funge da intermediario tra *publisher* e *subscriber*, e tiene traccia delle iscrizioni effettuate. I *publisher* invieranno i dati solo quando questi cambiano, cosicché i *subscriber* possano aggiornarli.

2.5.1 MQTT

Il protocollo *Message Queuing Telemetry Transport* (MQTT) è un protocollo di messaggistica leggero di tipo *publish-subscribe*. È stato creato nel 1999 da Andy Stanford-Clark (IBM) e Arlen Nipper (Arcom, ora Cirrus Link) per collegare degli oleodotti su connessione satellitare con l'obiettivo di minimizzare la larghezza di banda utilizzata ed il consumo della batteria. MQTT è stato progettato per essere *leggero*, semplice da implementare e ottimizzato per gestire lo scambio di dati che non necessitano di essere

memorizzati a lungo, secondo un approccio *stream-like*. Caratteristiche che lo hanno reso il protocollo ideale da utilizzare in comunicazioni *Machine-to-Machine* e in soluzioni per l'IoT dove si hanno spesso a disposizione dispositivi con risorse limitate [26].

MQTT è un protocollo di comunicazione asincrono che lavora in cima alla pila protocollare TCP/IP. Secondo il paradigma pub-sub, il protocollo definisce due entità di rete: un *message broker* e un certo numero di *client*. Un MQTT broker è un server che si occupa di gestire tutti i messaggi che riceve dai client, tiene traccia delle iscrizioni e quindi instrada i messaggi verso i client destinatari appropriati. Un MQTT client è un qualsiasi dispositivo, un microcontrollore o un vero e proprio server, che esegue le librerie MQTT e si connette ad un MQTT broker. Le librerie MQTT sono disponibili per un vasto repertorio di linguaggi di programmazione come C++, C, C#, Java, Go, iOS, Android, JavaScript, Python, .NET ecc. Un altro compito del broker è quello di autenticare e autorizzare i client che richiedono la connessione.

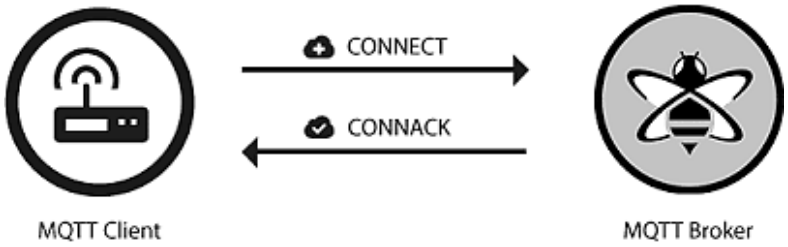

Figura 2.3: *— Connessione MQTT (Fonte immagine: [27])*

Ogni connessione MQTT [fig: 2.3] avviene tra un client ed un broker. I client non si connettono mai direttamente tra di loro. Per inizializzare la connessione, un cliente invia un messaggio di *connect* al broker. Quindi il broker risponde con un messaggio di *connack* ed uno status code. Una volta instaurata la connessione, il broker la mantiene aperta fino a che il client invia un messaggio di disconnessione o la connessione cade [27].

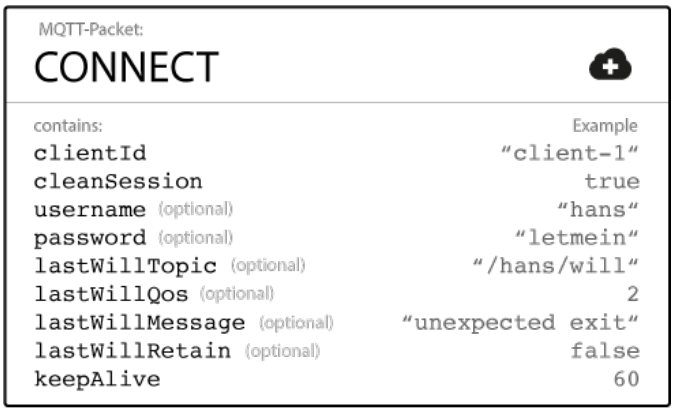

Figura 2.4: *— Messaggio MQTT di connect (Fonte immagine: [27])*

Il messaggio di connect [fig: 2.4] contiene i seguenti campi:

- **clientID**. Il broker utilizza il clientId per identificare ogni client ed il suo stato attuale.
- **cleanSession**. Indica al broker se il client vuole instaurare una sessione persistente o meno. Per una connessione persistente (*cleanSession = false*), il broker memorizza tutte le iscrizioni del client e tutti i messaggi persi per il client che ha sottoscritto un livello di qualità del servizio (QoS) 1 o 2. Se la sessione non è persistente (*cleanSession = true*), il broker non memorizza nulla per il client ed elimina tutte le informazioni delle sessioni persistenti precedenti.
- **username/password**. Utilizzati per l'autenticazione del client. Ovviamente si consiglia di utilizzare un protocollo crittografico come TLS.
- **Will message**. In MQTT si utilizza la funzione *Last Will and Testament (LWT)* per informare altri client quando si ha una disconessione non voluta.
- **keepAlive**. Indica un intervallo di tempo in secondi che il client specifica al broker quando instaura la connessione. Questo intervallo definisce il periodo di tempo più lungo che può intercorrere senza che client e broker si scambino messaggi. Allo scadere del tempo il broker provvede a chiudere la connessione.

Il broker risponderà con un messaggio di *connack*, che contiene i seguenti campi:

- **sessionPresent**. Indica se il broker ha già memorizzato una sessione persistente per questo client.
- **returnCode**. È un valore intero che può assumere un valore da 0 a 5 e che indica lo status code della richiesta di connessione effettuata dal client. Il valore 0 indica che la connessione è stata instaurata correttamente, gli altri valori indicano che la connessione è stata rifiutata e per quale motivo.

Una volta instaurata la connessione un client può decidere di pubblicare dei messaggi o iscriversi ad un *topic*.

Publish

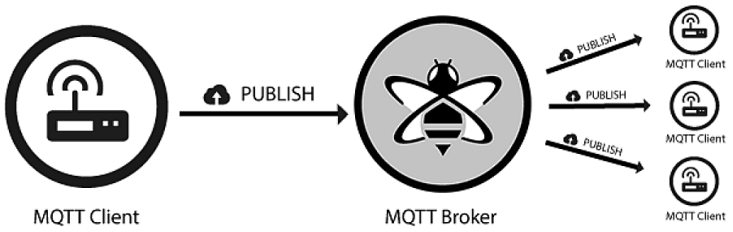

Figura 2.5: *— MQTT Publish message (Fonte immagine: [28])*

Ogni messaggio di *publish* contiene un *topicName*, il quale indica la gerarchia del topic, e che il broker utilizza per instradare il messaggio verso i client interessati. Tipicamente ogni messaggio ha un *payload* che contiene i dati da trasmettere. MQTT non indica il formato in cui questi dati devono essere inviati. Il client può decidere come strutturare il payload, ad esempio in formato XML, Json, txt ecc. Altri campi del messaggio di *publish* sono i seguenti:

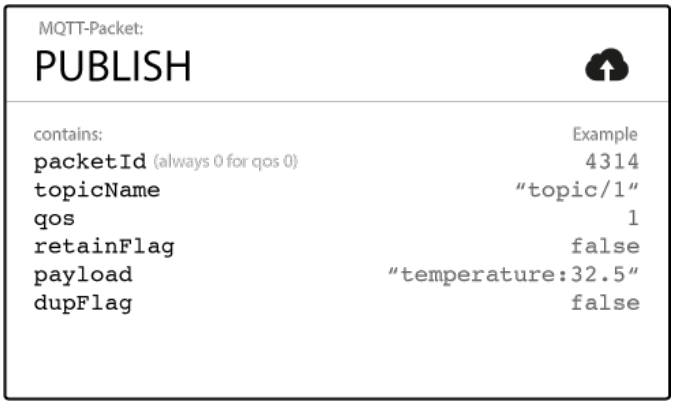

Figura 2.6: *— Messaggio MQTT di connect (Fonte immagine: [28])*

- **QoS**. Indica il livello di affidabilità del servizio, che può essere di tre tipi:
  - 0 - Fire and forget. Questo livello garantisce un servizio di consegna di tipo *best-effort*. Non c'è nessuna garanzia sulla consegna del messaggio.
  - 1 - Delivered at least once. Garantisce che il messaggio venga consegnato almeno una volta al destinatario. È possibile che un messaggio venga inviato e consegnato più volte.
  - 2 - exactly once. QoS 2 è il più alto livello di affidabilità in MQTT. Questo livello garantisce che ogni messaggio è ricevuto solo una volta. È il livello più sicuro ma anche quello più lento.
- **retainFlag**. Indica se il messaggio deve essere salvato dal broker per questo topic, come ultimo valore conosciuto, in questo modo se un client si iscrive al topic riceve subito l'ultimo messaggio disponibile.

Subscribe

Per ricevere i messaggi su *topic* di interesse, i client inviano messaggi di *subscribe* verso l'MQTT broker. Il messaggio di *subscribe* è estremamente semplice, contiene una stringa che identifica il pacchetto (*packetId*) e una lista di iscrizioni. Ogni iscrizione è formata da due stringhe, con una si indica *topic* a cui ci si vuole iscrivere con la seconda si indica il livello di QoS. Per confermare ogni iscrizione, il broker invia un messaggio di *suback*. In modo analogo un client può inviare un messaggio di *unsubscribe* per disiscriversi dal *topic*.

2.5.2 REST

L'uso di web API (Application Program Interface) è ormai onnipresente in ogni tipo di applicazione che si collega ad Internet. Una API consiste in un insieme di routine, protocolli, funzioni e strumenti software utili per lo sviluppo di nuove applicazioni accessibili tramite Internet. Le API sono particolarmente collegate al mondo IoT in quanto permettono di esporre in modo sicuro strumenti software verso tutti dispositivi connessi nella propria infrastruttura IoT e consentono di integrare in modo semplice diversi servizi per sviluppare nuove soluzioni IT. Le web API, sono uno sviluppo dei servizi web classici, dove l'accento è spostato su comunicazioni più semplici basate sull'architettura *Representational Transfer State* (REST), a differenza dei servizi web basati su XML come SOAP e WSDL.

REST non è un vero e proprio protocollo, bensì è uno stile di architettura software creata per guidare la progettazione e lo sviluppo di architetture che lavorano sul *World Wide Web*. REST definisce una serie di vincoli su come una architettura web distribuita dovrebbe essere costruita. Lo stile di architettura REST prevede l'utilizzo di interfacce uniformi e standardizzate, la distribuzione indipendente dei componenti e la creazione di un'architettura a strati per facilitare la memorizzazione nella cache dei componenti e ridurre la latenza.

Le API di tipo REST utilizzano i metodi tipici di HTTP, come POST, GET, PUT, DELETE per fornire un sistema di messaggistica *state-less* orientato alle risorse, standardizzato e altamente scalabile. L'header HTTP è utilizzato per specificare il tipo di formato utilizzato per lo scambio dei dati, ad esempio Json o XML, e dipende dal server HTTP a cui si invia la richiesta. REST è un componente importante per l'IoT e la comunicazione M2M in quanto fornisce una interfaccia uniforme, richiede solo l'utilizzo della libreria HTTP ed è supportato dalla maggior parte delle piattaforme Cloud per IoT moderne. Tuttavia, essendo un modello request-response non ottimizzato per dispositivi IoT a basso consumo, quindi con un continuo *polling* delle risorse, il suo utilizzo è limitato a dispositivi con una buona capacità elaborativa e quindi non è adatto a comunicazioni device-to-device in ambito IoT.

2.5.3 CoAP

Constrained Application Protocol (CoAP) è un protocollo di comunicazione di tipo *request-response*, progettato dall'Internet Engineering Task Force (IETF) per essere utilizzato in soluzioni IoT dove è necessario abilitare la comunicazione tra nodi con scarse capacità di elaborazione, a bassa potenza e su reti che richiedono un consumo di banda ridotto. Il protocollo è progettato per lavorare su applicazioni Machine-to-Machine (M2M), ad esempio nel campo della Smart Energy e dell'automazione [29]. CoAP mira a realizzare un architettura che può essere facilmente integrata con HTTP ed i servizi di tipo REST, in una forma più adatta ai dispositivi con capacità limitate, tipici dell'Internet of Things.

Modello di interazione

Il modello di interazione di CoAP è simile al modello client/server di HTTP. Tuttavia, nell'implementazione CoAP ogni entità può agire sia da client che da server. La richiesta CoAP è equivalente a quella HTTP e viene inviata da un client ad un server per richiedere un'azione (usando un *Method Code*) su una risorsa (identificata da un URI). Il server, quindi, invia una risposta che contiene un *Response Code* e la rappresentazione delle risorse richieste.

A differenza di HTTP, CoAP gestisce la comunicazioni in modo asincrono utilizzando come protocollo di trasporto UDP con il supporto di connessioni *unicast* e *multicast*. UDP, al contrario di TCP, è un protocollo senza connessione e per questo più leggero, senza nessuna garanzia sulla consegna dei messaggi. Il suo utilizzo consente di limitare l'overhead dell'intestazione dei messaggi e la complessità del *parsing* dei messaggi. CoAP introduce dei livelli di affidabilità opzionali per colmare il vuoto lasciato da UDP, ed utilizza i seguenti quattro tipi di messaggi:

- **Confirmable**. Indica un messaggio per cui è richiesta la conferma di ricezione (ack).
- **Non-Confirmable**. Indica un messaggio per cui non richiede una trasmissione affidabile e per cui non è richiesta la conferma di ricezione.
- **Acknowledgment**. Conferma la ricezione di un messaggio *Confirmable*.
- **Reset**. Conferma la ricezione di un messaggio che al momento non può essere processato.

Mentre, per quanto riguarda la sicurezza, CoAP fa uso del protocollo Datagram TLS (DTLS) e fornisce meccanismi per l'autenticazione, per garantire l'integrità dei dati, per la gestione automatica delle chiavi e algoritmi crittografici come RSA e AES.

2.5.4 AMQP

Advanced Message Queuing Protocol (AMQP) è un protocollo orientato ai messaggi, basato sul modello pub-sub. AMQP è open ed è stato sviluppato nel settore finanziario per garantire sicurezza, affidabilità ed interoperabilità. Inoltre è ottimizzato per gestire lo scambio di dati che necessitano di essere registrati in un buffer, a differenza di MQTT.

Di seguito vengono riportate le caratteristiche principali di AMQP:

- Supporta connessioni peer-to-peer e il multiplexing della connessione.
- I dati sono memorizzati utilizzando una mappa, con una coppia *key-value*.
- È integrato con TLS *Transport Layer Security* e SASL *Simple Authentication and Securtiy Layer* per garantire una connessione criptata e sicura.
- I messaggi rimangono in coda per tutto il tempo richiesto, anche se nessuno sta usando la coda.

2.5.5 Considerazioni sui protocolli di comunicazione a livello applicativo

In questo paragrafo sono stati analizzati diversi tra i più diffusi protocolli di comunicazione a livello applicazione utilizzati in ambito IoT. L'uso di uno piuttosto che un altro dipende esclusivamente dal caso d'uso e dai dispositivi che è necessario connettere. Nel caso in cui si ha che fare con dispositivi a bassa potenza e limitata capacità elaborativa, come dei microcontrollori, l'utilizzo di AMQP è sconsigliato in quanto è un protocollo più *pesante* rispetto a MQTT e CoAP. Motivo per cui l'utilizzo di AMQP in ambito IoT rimane limitato. Il protocollo AMQP può essere utile, piuttosto, quando è necessario effettuare *multiplexing* di connessione o gestire la comunicazione *bufferizzando* i dati. Ad esempio, un caso d'uso di AMQP è nei dispositivi gateway che hanno la necessità di gestire più nodi e quindi dove la possibilità di utilizzare più canali logici su una singola connessione AMQP risulta essere comoda. MQTT, invece, è il protocollo più *leggero* ed inoltre, grazie all'utilizzo del paradigma pub-sub, permette una gestione più efficiente della comunicazione evitando di eseguire un continuo polling delle risorse. Questo aspetto, oltre alla possibilità di configurare le connessioni con diversi livelli di QoS e al basso consumo energetico, rende MQTT il protocollo di comunicazione più diffuso per lo scambio di messaggi tra dispositivi IoT. Il protocollo CoAP, ha dalla sua la possibilità di essere integrato con DTLS e quindi fornire maggiore privacy e sicurezza dei dati rispetto ad altri protocolli. Tuttavia, utilizzando UDP come protocollo di trasporto e pur introducendo strumenti aggiuntivi per la conferma di ricezione dei messaggi rimane un protocollo meno affidabile rispetto a MQTT. In particolare, con CoAP non è possibile verificare se il messaggio è stato ricevuto nella sua interezza o se è stato decodificato correttamente. A seguire, la tabella 2.1 riassume le caratteristiche principali dei protocolli a livello applicativo analizzati.

Tabella 2.1: *Protocolli di comunicazione a livello applicativo a confronto*

	MODELLO	CARATTERISTICHE PRINCIPALI
MQTT	publish-subscribe	• *Leggero* • Supporta diversi livelli di QoS per garantire la consegna dei messaggi • Basso consumo energetico
CoAP	request-response	• Modello di interazione simile ad HTTP • Utilizza UDP come protocollo di trasporto • Può essere combinato con DTLS per criptare i dati
AMQP	publish-subscribe	• È un protocollo *open* • Supporta più canali su una singola connessione (*multiplexing di connessione*) • Supporta connessioni *peer-to-peer* • Più *pesante* rispetto a MQTT

CAPITOLO 3

Soluzioni Cloud per l'Internet of Things

Cloud computing e Internet of Things sono due tecnologie estremamente differenti, affermate negli ultimi anni fanno già parte della nostra vita e si prevede che il loro utilizzo aumenterà ulteriormente, rendendo loro componenti fondamentali per l'Internet di domani. Sebbene in letteratura queste due tecnologie sono state spesso trattate separatamente, in questo capitolo l'attenzione è focalizzata sull'integrazione tra Cloud e IoT, un nuovo paradigma a cui si fa riferimento con il nome Cloud-IoT. Viene discusso il perchè di questa integrazione, i vantaggi che può portare e le sfide future che ne derivano. Inoltre, viene presentato un modello di riferimento di una tipica architettura IoT e vengono evidenziati i limiti di una soluzione *Cloud-centrica*. Infine, vengono analizzate le tre principali piattaforme Cloud per IoT (Google Cloud IoT, AWS IoT e Azure IoT), i numerosi servizi che offrono e come è possibile combinarli per implementare una soluzione IoT

3.1 Integrazione tra Cloud e IoT

Se l'IoT è alla base dello sviluppo di tecnologie come Smart Home, Smart City, IoMT (Internet of Medical Things), il Cloud computing è la chiave per abilitarle e far sì che possano diffondersi su larga scala. Il Cloud permette di processare, filtrare, archiviare e analizzare l'ingente quantità di dati generati dai dispositivi grazie a piattaforme flessibili e modulari, accessibili in remoto.

3.1.1 Vantaggi del paradigma Cloud-IoT

Di seguito, vengono riportati i vantaggi che si possono ottenere adottando il paradigma Cloud-IoT [30]:

Risorse di archiviazione.

Il modello IoT, intrinsecamente, coinvolge un gran numero di dispositivi che raccolgono informazioni e che generano un'enorme quantità di dati per lo più non strutturati o semi strutturati aventi le tre caratteristiche tipiche dei *Big Data*: volume, varietà e velocità (ovvero l'elevata frequenza con cui i dati vengono generati). Tutto questo implica che i dati devono essere collezionati, archiviati, processati, analizzati e facilmente accessibili. Il cloud risponde in modo efficace a queste necessità offrendo spazio di archiviazione virtualmente illimitato, accessibile in remoto, a basso prezzo e su richiesta.

Risorse computazionali.

I dispositivi IoT, tipicamente, hanno una limitata capacità di elaborazione in loco. Per questo motivo, spesso, i dati raccolti vengono trasmessi verso altri nodi più potenti, tuttavia senza un'infrastruttura adeguata è difficile rendere il sistema scalabile e affidabile. Le risorse *illimitate* del Cloud, disponibili su richiesta, consentono che le necessità d'elaborazione di un'infrastruttura IoT siano adeguatamente soddisfatte ed inoltre lasciano spazio ad analisi di complessità senza precedenti.

Risorse di comunicazione.

Uno dei requisiti dell'IoT è far comunicare i dispositivi tra di loro, ed il supporto per tale comunicazione può essere molto costoso. Il cloud offre una soluzione efficace ed economica per connettere, tracciare e gestire qualsiasi cosa, da qualsiasi luogo ed in qualsiasi momento. Inoltre, avendo accesso a reti ad alta velocità, consente il monitoraggio e il controllo di dispositivi remoti, il loro coordinamento e l'accesso in tempo reale ai dati generati.

Nuove possibilità.

L'IoT è caratterizzato da un'altissima eterogeneità di dispositivi, tecnologie e protocolli. Perciò qualità come: scalabilità, interoperabilità, affidabilità, efficienza, disponibilità, e sicurezza possono essere molto difficili da ottenere. L'integrazione con il Cloud consente di risolvere la maggior parte di questi problemi, fornendo anche funzionalità aggiuntive come facilità di accesso, facilità d'uso e costi di implementazione ridotti.

In breve, la combinazione tra Cloud e IoT trae il massimo da entrambi i modelli [31], che, come si evidenzia nella seguente tabella [Fig.3.1], sono del tutto complementari.

IoT	Cloud
pervasive (things placed everywhere)	ubiquitous (resources usable from everywhere)
real world things	virtual resources
limited computational capabilities	virtually unlimited computational capabilities
limited storage or no storage capabilities	virtually unlimited storage capabilities
Internet as a point of convergence	Internet for service delivery
big data source	means to manage big data

Figura 3.1: *— Complementarietà tra IoT e Cloud (Fonte immagine: [30])*

3.2 Architetture IoT

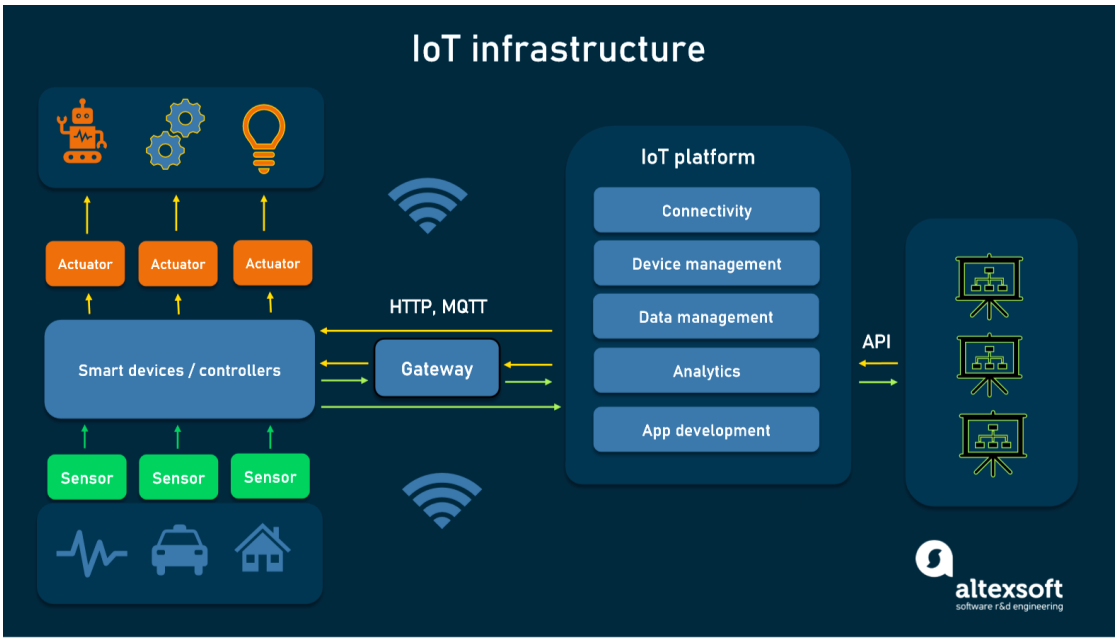

Figura 3.2: *— How an IoT system works. (Fonte immagine: https://www.altexsoft.com/blog/iot-platforms/)*

Prima di passare ad analizzare le piattaforme Cloud-IoT prese in esame è utile vedere come è organizzata una tipica architettura IoT *Cloud-based*, la quale può essere schematizzata in diversi strati, come illustrato in [fig: 3.2].

- perception layer (strato di rilevamento): è costituito da una vasta gamma di componenti hardware come sensori, attuatori ed altri dispositivi più o meno potenti. Essi interagiscono con l'ambiente raccogliendo dati o compiendo azioni, convertendo le informazioni provenienti dal mondo reale in dati digitali.

- transport layer (strato di trasporto): è composto da reti e gateway. Un dispositivo *gateway* svolge il ruolo di intermediario tra i dispositivi IoT connessi e il Cloud, gestisce la comunicazione sicura tra di essi, e in alcuni casi, esegue in tempo reale analisi, inferenza di Machine learning o si limita a *pulire* i dati prima di inoltrarli verso il Cloud, secondo il modello *Edge computing*. La comunicazione è tipicamente gestita adottando lo schema *publish/subscribe* [Fig. 3.3], dove viene utilizzato un *middleware* orientato ai messaggi che consente una comunicazione distribuita, asincrona e con basso accoppiamento tra il produttore e il consumatore dei messaggi.[32].

- processing layer (strato di elaborazione): si occupa di accumulare, processare, riorganizzare i dati *grezzi* che vengono dai livelli precedenti. Questo compito spetta ad una piattaforma IoT, ed è tipicamente diviso in due fasi [33]: la **fase di accumulo dei dati** e la cosiddetta fase di ***data abstraction***, ovvero il processo con cui si riduce un certo volume di dati (non strutturati o semi-strutturati), in una rappresentazione semplificata e organizzata [34].

- application layer (strato applicativo): è composto da soluzioni software pensate per l'*end-user*. Permette di elaborare ulteriormente i dati, analizzarli e fornire strumenti per la *business intelligence* [33]. Oltre che monitorare, controllare i dispositivi e in generale interagire con il mondo fisico da remoto utilizzando la piattaforma IoT, agisce da ponte tra il mondo fisico e i processi di business.

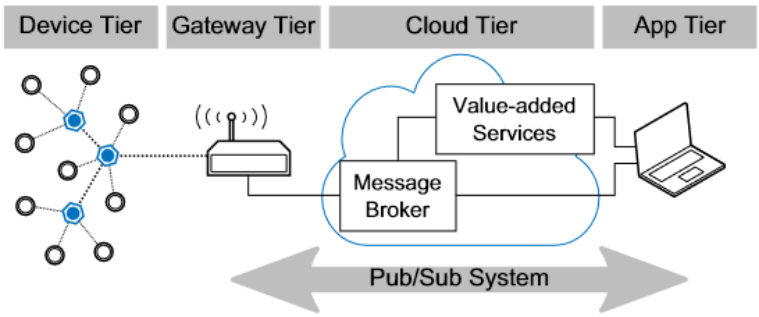

Figura 3.3: *— General architecture of a cloud-centric IoT platform (Fonte immagine: [35])*

3.3 Limiti di una soluzione Cloud-centrica

Sebbene trasferire interamente l'elaborazione su Cloud sembra essere una soluzione efficace ed efficiente grazie alla potenza che mette a disposizione e la conseguente velocità di elaborazione, con l'esponenziale crescita del numero di dispositivi e quindi di dati generati da essi la rete si sta rivelando il vero collo di bottiglia delle infrastrutture Cloud-based. È ormai chiaro, quindi, che una soluzione completamente centralizzata

sui data center Cloud non è sempre applicabile, in particolare in applicazioni IoT che richiedono l'elaborazione di tanti dati in tempo reale, quindi con minima latenza. Basti pensare che un'auto a guida autonoma genera un quantitativo di dati nell'ordine di 1 Gigabyte per secondo [36]. Da qui la necessità di ripensare l'architettura IoT, secondo il modello *Edge computing*.

3.3.1 Edge computing

Con il termine *Edge computing* si fa riferimento a tutte le risorse IT che consentono l'elaborazione lungo il percorso dati tra sorgente e data center Cloud, secondo un modello di infrastruttura IT decentralizzata e distribuita. L'introduzione di dispositivi sempre più intelligenti, organizzati in *micro data center*, ai bordi della rete permette di filtrare, elaborare, analizzare i dati ed eseguire inferenza di Machine learning in locale riducendo i tempi di latenza, riducendo il consumo di banda Internet e incrementando la sicurezza. Inoltre consente di operare anche quando i dispositivi IoT non riescono a connettersi al cloud.

Questo modello si basa sull'idea di spostare parte dell'elaborazione ai margini della rete, laddóve i dati vengono prodotti [37]. Si tratta, quindi, di un concetto per certi versi agli antipodi rispetto al cloud computing, che sfrutta la rete internet per comunicare con un data center remoto, che sebbene disponga di una grande capacità di elaborazione, tipicamente si trova molto distante dal luogo in cui i dati vengono richiesti o prodotti [38]. Quest'ultima riflessione, però, non deve trarre in inganno. L'Edge computing non è da considerarsi come una tecnologia alternativa che sostituisce il modello Cloud basato su data center centralizzati, bensì le due soluzioni vengono tipicamente implementate per operare in modo coordinato e cooperativo. In quest'ottica è possibile pensare a soluzioni dove i dispositivi *edge* si occupano semplicemente di *ripulire* i dati prima di inviarli in Cloud o eseguono inferenza di Machine learning su modelli addestrati precedentemente in Cloud, grazie alla potenza disponibile.

3.4 Piattaforme Cloud per IoT

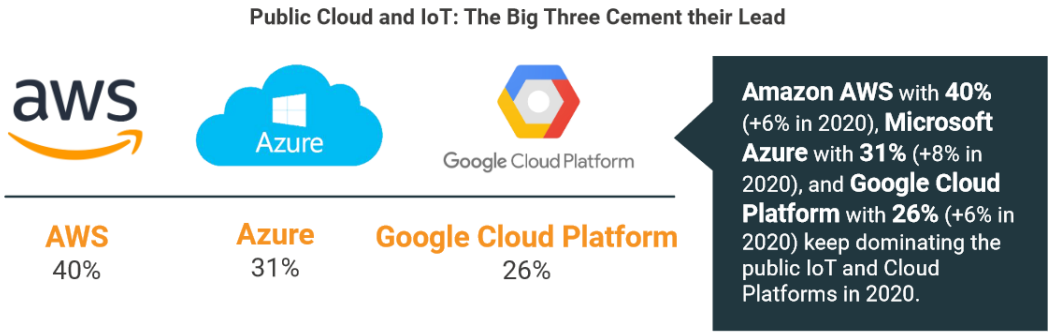

Figura 3.4: — *Public Cloud and IoT: The Big Three Cement their Lead (Fonte immagine: [39])*

Già da qualche anno diversi tra i maggiori *Cloud providers* offrono servizi Cloud pensati unicamente per il mondo IoT. Una piattaforma Cloud per IoT fornisce un punto d'incontro per una moltitudine di dispositivi eterogenei connessi simultaneamente, colleziona i dati ricevuti tramite la rete e rende disponibili servizi per l'elaborazione e l'analisi degli stessi. In questo paragrafo saranno presentate le tre piattaforme Cloud-IoT che attualmente detengono la leadership del mercato globale (AWS IoT, Google Cloud IoT, Microsoft Azure for IoT), in accordo con "The Eclipse Foundation Releases Results from the 2020 IoT Developer Survey" [39][Fig.3.4].

3.4.1 Google Cloud IoT

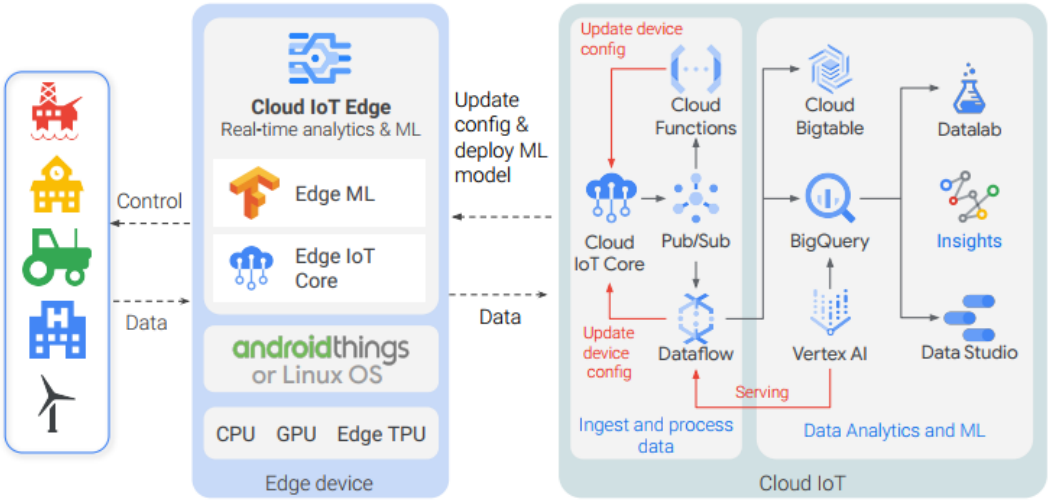

Figura 3.5: *— Architettura di Google Cloud IoT (Fonte immagine: https://cloud.google.com/iot)*

Google Cloud IoT permette di gestire milioni di dispositivi IoT e li connette con una moltitudine di servizi Google Cloud per l'elaborazione e l'analisi dei dati sia in Cloud che ai bordi della rete, il tutto in un ambiente automaticamente scalabile e con il minimo sforzo di gestione. La sua architettura [Fig. 3.5] può essere suddivisa in tre diversi livelli che realizzano le seguenti tre fasi: raccolta dei dati (*data gathering*), acquisizione ed elaborazione dei dati (*data ingest and processing*) ed analisi dei dati.

Raccolta dei dati

La prima fase è quella di raccolta dei dati. In questa fase i dispositivi IoT rilevano i dati e possono inviarli direttamente in Cloud o attraverso un dispositivo gateway. Nell'architettura di Google la raccolta dei dati può includere l'utilizzo di Cloud IoT Edge, che estende la potenza di Google Cloud verso milioni di nodi ai bordi della rete e mette a disposizione una collezione di *edge device* per l'analisi di dati in *real-time*, per l'inferenza tramite tecniche di Machine learning o per ripulire i dati prima di inviarli verso il cloud. L'utilizzo di gateway intelligenti capaci di eseguire una prima elaborazione dei dati è cruciale per evitare di sovraccaricare la rete e fa sì che solo i dati filtrati siano trasmessi verso il Cloud.

Acquisizione ed elaborazione dei dati

Come si può vedere in figura [Fig. 3.5] la fase di elaborazione dei dati è abilitata dai seguenti servizi: Cloud IoT Core, Cloud Functions, Pub/Sub e Dataflow. *IoT Core* si

preoccupa di gestire i dispositivi connessi e si assicura che la comunicazione avvenga in modo sicuro. I dati inviati dai dispositivi vengono gestiti da *Pub/Sub*, un servizio orientato ai messaggi per l'acquisizione dei dati che abilita la comunicazione asincrona tra i dispositivi e gli altri servizi Cloud secondo il modello publish/subscribe [40]. I produttori (*publisher*) inviano i messaggi specificando un *Pub/Sub topic*, in modo tale che i consumatori (*subscriber*) iscritti al topic ricevano il messaggio. I messaggi sono quindi inoltrati verso *Google Cloud Dataflow*, che processa i dati e li invia verso gli altri servizi cloud, come *BigQuery*, *Cloud Storage* o *BigTable* [32]. Dataflow è un servizio *serverless* basato su *pipeline*, ottimizzato per gestire l'elaborazione di un gran volume di dati in un colpo solo (*batch data*) e per l'analisi di un flusso di dati continuo (*streaming data*). Cloud Dataflow utilizza le librerie di *Apache Beam*, un modello unificato e *open-source* per la definizione di pipeline e l'elaborazione di *batch* e *streaming data* [41]. Suddividere l'elaborazione in pipeline consente di lavorare su grandi moli di dati in modo efficiente, in quanto l'elaborazione viene distribuita verso diverse macchine virtuali, cosicché possano processare pezzi (*chunks*) di dati in parallelo [42]. Inoltre, si occupa in maniera completamente automatica di distribuire il carico di lavoro e scalare dinamicamente.

In alternativa, è possibile utilizzare *Cloud Function* per personalizzare il comportamento in base al tipo di evento che si verifica. Cloud Function consente di scrivere funzioni *event-driven*, che vengono innescate a seguito di una pubblicazione di un messaggio verso un determinato *Pub/Sub topic* [43].

Google Cloud IoT Core

Al centro di Google Cloud IoT c'è IoT Core, un servizio completamente gestito che consente di connettere, gestire e acquisire dati in modo semplice e sicuro da milioni di dispositivi sparsi in tutto il mondo. IoT Core, combinato con gli altri servizi offerti dalla piattaforma Google Cloud IoT fornisce una soluzione completa per elaborare, analizzare e visualizzare i dati IoT in tempo reale [40].

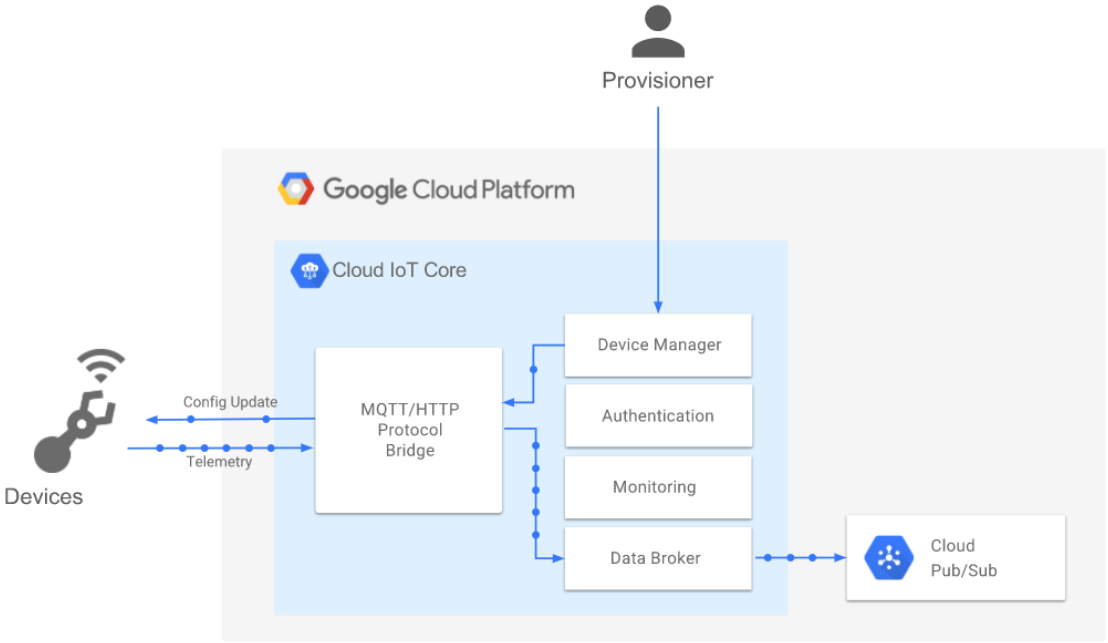

Figura 3.6: *— Google Cloud IoT Core (Fonte immagine: https://cloud.google.com/iot/docs/concepts/overview)*

IoT Core è composto da due moduli principali, *Device manager* e *Protocol bridge*. Il ***Device manager*** consente di gestire, configurare e controllare i dispositivi da remoto in modo sicuro tramite *Google Cloud Console*, *gcloud commands* o utilizzando librerie per C#, Java, NodeJS, GO, PHP, Python e RUBY accessibili tramite API che facilitano lo sviluppo. Inoltre determina l'identità dei dispositivi e fornisce il meccanismo per autenticarli quando si connettono. Con il *Device manager* è possibile creare e configurare dei registri. Ogni registro è configurato con uno o più *topic* di *Cloud Pub/Sub* tramite cui vengono pubblicati i dati telemetrici verso tutti i dispositivi del registro, in un formato *orientato agli eventi* (si parla di *telemetry events*). Un evento cattura un formato dati comune che consente un utilizzo più ampio e condiviso degli strumenti di elaborazione di dati [44]. Ogni registro è creato in una specifica *cloud region* ed appartiene ad un particolare progetto (*cloud project*). Dove per *cloud region* si intende l'attuale posizione geografica delle risorse del cloud pubblico in uso [45]. Un registro è quindi identificato dal suo *nome completo*, del tipo:

$$projects/\{project\text{-}id\}/locations/\{cloud\text{-}region\}/registries/\{registry\text{-}id\}$$

Il ***Protocol bridge*** gestisce la connettività e supporta nativamente la connessione sicura su protocolli standard del settore come MQTT e HTTP, in particolare si occupa di caricare il flusso dati prodotto dai dispositivi connessi verso *Pub/Sub*. Utilizzando MQTT, i dispositivi inviano una richiesta di iscrizione (modello publish/subscribe) ad uno specifico *topic*, ovvero una particolare stringa tramite la quale l'*MQTT broker* filtra

i messaggi per ogni connessione. Mentre utilizzando HTTP i dispositivi non mantengono una connessione alla piattaforma e inviano specifiche richieste [32].

Archiviazione, analisi dei dati e Machine Learning

Un altro aspetto da analizzare è quello dell'archiviazione dei dati in cloud. I dispositivi inviano differenti tipi di dati: *metadata*, ovvero dati che contengono informazioni sul dispositivo e che raramente cambiano; dati che descrivono lo stato dei dispositivi (di solito in formato strutturato); dati telemetrici o dati non strutturati detti *BLOB (Binary Large Object)* (file multimediali, streaming video, ...). La piattaforma Cloud deve essere in grado di gestire adeguatamente l'archiviazione di tutti questi tipi di dati, nel modo più efficace ed efficiente possibile. Quando è necessario gestire dati strutturati la memorizzazione viene gestita direttamente all'interno dei servizi di *IoT Core*. Mentre quando si ha a che fare con dati telemetrici che arrivano con una elevata frequenza e devono essere disponibili con bassi tempi di latenza ed alte *performance* Google offre altri servizi, quali: *Cloud Datastore, Cloud BigQuery e Cloud BigTable* [32].
Google Cloud Datastore è un database cloud-based NoSQL. Supporta le transazioni *ACID*, query *SQL-like* e *REST API*. *Datastore* è ottimizzato per lavorare su un insieme relativamente piccolo di dati [46].
Google Cloud BigTable è un database cloud NoSQL, di tipo *wide-column*, ottimizzato per l'accesso ai dati con *bassa latenza*, per gestire un elevato numero di letture e scritture e per mantenere le prestazioni elevate anche su larga scala.
Google Cloud BigQuery è un soluzione cloud di *data warehouse* pensata per le aziende. È in grado di gestire un grande quantitativo di dati relazionali strutturati ed è ottimizzato per realizzare analisi *SQL-based* e report, su larga scala [47].

Oltre a servizi per l'archiviazione, interrogazioni su database *SQL-like* e *NoSQL* ed analisi dei dati, Google Cloud IoT include tutte le funzionalità di *Vertex AI* per il *training* dei modelli e l'esecuzione di inferenza di Machine learning. Vertex AI racchiude i servizi Google per il *ML* in un'unica API e in un'unica interfaccia utente (UI). Inoltre, è nativamente integrato con altri servizi come *BigQuery*, *Dataproc* e *Spark* e supporta librerie *open-source* come *TensorFlow* e *scikit-learn*. Ad esempio si può utilizzare *BigQuery ML* per creare ed eseguire modelli di ML tramite BigQuery utilizzando delle classiche query SQL o è possibile esportare dei *dataset* da BigQuery ed importarli direttamente in Vertex AI per eseguire i modelli di ML [48]. Come detto precedentemente, l'architettura di Google estende alcune delle funzionalità sopra citate anche ai dispositivi *edge*, grazie all'utilizzo di hardware specializzati come *Edge TPU (Tensor Process Unit)*. *Edge TPU* è una tecnologia progettata per essere complementare a Google Cloud TPU, in modo che sia possibile eseguire il *training* dei modelli in Cloud con la massima potenza ed eseguire algoritmi di inferenza ai bordi della rete garantendo tempi minimi di latenza. In questo i modo i dispositivi locali non sono più solo rilevatori di dati, ma sistemi intelligenti in grado di prendere decisioni in tempo reale.

Sicurezza

Come sappiamo, la sicurezza è un aspetto critico e gioca un ruolo chiave quando si tratta di gestire dispositivi IoT. *Cloud IoT Core* offre le seguenti funzionalità [49]:

- Autenticazione con una coppia chiave pubblica/privata per ogni dispositivo, tramite *JWT* (Json Web Token). Questo consente di limitare l'area di un attacco, in quanto la compromissione di una chiave può causare danni solo su un singolo dispositivo. Inoltre dato che i JWT sono validi solo per un periodo limitato di tempo, una chiave compromessa scadrà.
- Supporto di algoritmi *RSA* o *Elliptive Curve* per la verifica delle firme, con l'utilizzo di chiavi di elevate dimensioni.
- Possibilità di *ruotare* le chiavi. Ovvero è possibile registrare, in contemporanea, chiavi multiple per uno stesso dispositivo (fino a 3 chiavi per dispositivo) così da permettere una rotazione ininterrotta tra di esse.
- Utilizzo del protocollo *TLS 1.2* durante la comunicazione.
- *IAM* (Identity and Access Management) un servizio Google con il quale è possibile controllare l'accesso alle API di Cloud IoT Core, con la possibilità di personalizzare ruoli e permessi.

3.4.2 AWS IoT

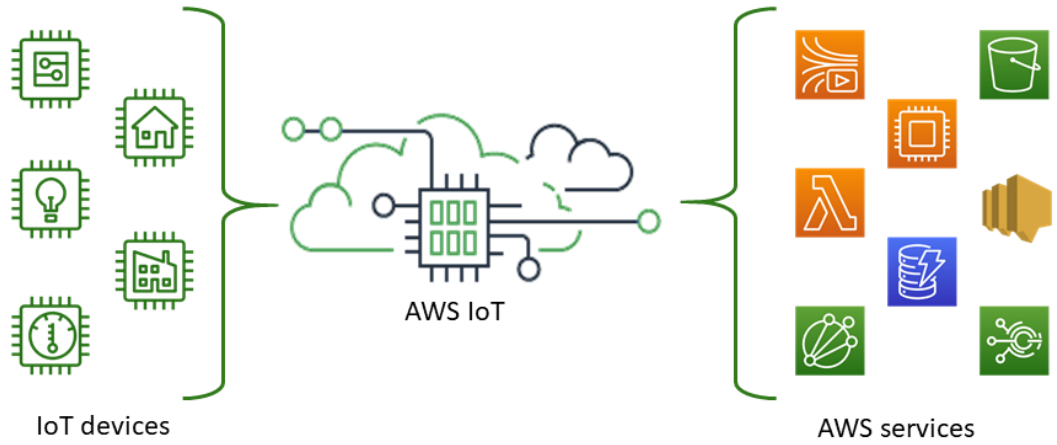

Figura 3.7: *— AWS IoT*
(Fonte immagine: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html)

AWS IoT è una piattaforma che fornisce comunicazioni sicure e bidirezionali tra dispositivi connessi a Internet come sensori, attuatori, microcontrollori *embedded* o qualsiasi altro dispositivo intelligente e i servizi Cloud di *AWS* [fig: 3.7]. Consente di controllare in remoto una miriade di dispositivi, da essi acquisire dati telemetrici e offre servizi per la loro archiviazione e l'analisi. AWS IoT fornisce una serie di funzionalità sia in Cloud che in *edge* per implementare qualsiasi tipo di soluzione IoT, utilizzando un'ampia gamma di dispositivi. Inoltre mette a disposizione un ambiente che unisce intelligenza artificiale (IA) e IoT, in cui è possibile creare i modelli in Cloud per poi distribuirli facilmente verso i dispositivi.

I servizi di *AWS IoT* possono essere suddivisi nelle seguenti 3 macro-categorie.

- **Software per dispositivi**: *FreeRTOS* e *AWS IoT Greengrass.*
- **Servizi di controllo e per la connettività**: *AWS IoT Core*, *AWS IoT Defender* e *AWS IoT Device Management.*
- **Servizi d'analisi**: *AWS IoT Analytics*, *AWS IoT SiteWise* e *AWS IoT Events* e *AWS IoT Things Graph.*

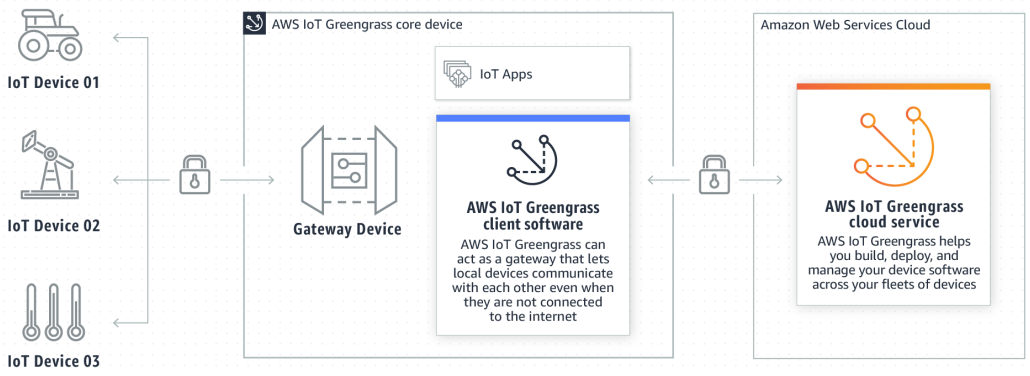

Figura 3.8: *— AWS IoT Greengrass (Fonte immagine: [50])*

AWS IoT mette a disposizione una serie di software pensati esclusivamente per i dispositivi IoT che permettono di raccogliere i dati ed abilitare le funzionalità di *AWS Cloud* ai bordi della rete così da effettuare azioni intelligenti in locale, persino quando si perde la connessione.

FreeRTOS

FreeRTOS è un sistema operativo specializzato per il supporto di applicazioni su sistemi *real-time*, *open source*, per microcontrollori che rende semplice programmare, distribuire, proteggere, collegare e gestire dispositivi *edge* di piccole dimensioni e a basso consumo. Un microcontrollore contiene un semplice processore con risorse limitate, questo tipo di hardware è utilizzato nella maggior parte dei dispositivi IoT di piccole dimensioni che trovano applicazioni in diversi ambienti come: dispositivi di monitoraggio fitness, automazione industriali e automobili. Molti di questi piccoli dispositivi possono trarre vantaggio dalla connessione al cloud o in locale ad altri dispositivi, ma hanno potenza di elaborazione e capacità di memoria limitate e in genere svolgono operazioni semplici e funzionali. Di conseguenza i microcontrollori eseguono spesso sistemi operativi che non dispongono di funzionalità integrate per la gestione di connessioni al cloud o in locale. *FreeRTOS* aiuta a risolvere questo problema fornendo il *kernel* eseguibile su dispositivi a basso consumo, oltre a librerie software che semplificano il collegamento sicuro al cloud o ad altri dispositivi *edge* [51].

AWS IoT Greengrass

AWS IoT Greengrass è pensato per estendere AWS in maniera lineare ai dispositivi edge. I dispositivi che eseguono Linux e supportano le architetture *Arm* o *x86* possono ospitare AWS IoT Greengrass Core, il quale può essere programmato per connettersi ai diversi servizi AWS, filtrare i dati dei dispositivi, gestire il ciclo di vita dei dati sul dispositivo e trasmettere ad AWS soltanto le informazioni necessarie. I dispositivi che eseguono AWS IoT Greengrass Core, detti connettori, agiscono da *hub* [fig: 3.8] e possono comunicare con tutti i dispositivi che eseguono *FreeRTOS* o in cui è installato il *kit Device SDK*

di AWS IoT. Tutti questi dispositivi possono essere configurati per comunicare tra di loro in un *gruppo AWS IoT Greengrass*, cosicché se un dispositivo del gruppo perde la connessione al Cloud, gli altri dispositivi possono continuare a comunicare tra di loro sulla rete locale [50]. *Greengrass* include il supporto di *AWS Lambda*, un servizio di elaborazione *serverless* guidato dagli eventi in grado di ridimensionare automaticamente le risorse necessarie in base al carico di lavoro. Il termine *serverless* può trarre in inganno, in quanto i Cloud server sono comunque utilizzati ma il programmatore non se ne deve preoccupare in quanto opera ad un livello di astrazione più alto e può eseguire il codice senza la necessità effettuare l'approvvigionamento, gestire e configurare i server. Le funzioni *lambda* possono essere invocate in qualsiasi momento direttamente tramite i dispositivi, in locale, per rispondere in modo tempestivo agli eventi, interagire con risorse locali e con l'ambiente circostante pagando solo per il tempo in cui si utilizza il servizio. Inoltre, *AWS IoT Greengrass* rende facile eseguire inferenza di Machine learning in locale utilizzando modelli costruiti ed *allenati* in cloud. Questo permette di ridurre la latenza, azzera i costi dovuti al trasferimento dei dati e contribuisce a ridurre il consumo di banda [52].

AWS IoT Core

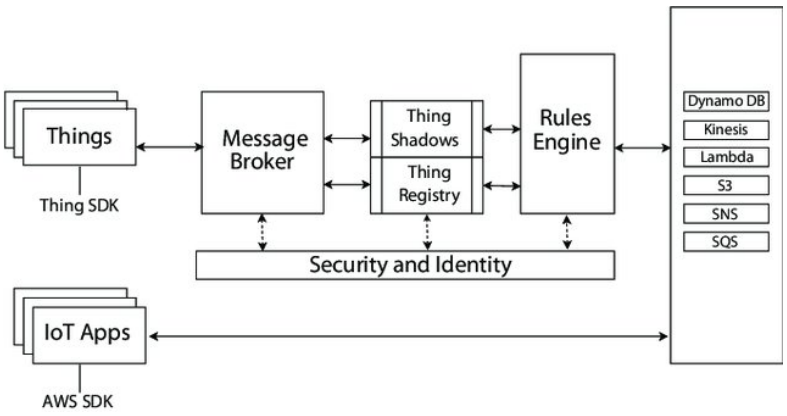

Figura 3.9: *— Architettura di AWS IoT Core (Fonte immagine: [32])*

AWS IoT Core costituisce la spina dorsale della piattaforma AWS per connettere in modo sicuro i dispositivi IoT, controllarli, interagire con essi, acquisire i dati ed instradarli verso altri dispositivi o verso altri servizi AWS. La comunicazione da e verso *AWS IoT Core* è realizzata utilizzando un *message broker* basato sul modello *publish/subscribe*, ovvero un servizio che svolge il ruolo di *mediatore* tra la piattaforma Cloud e i dispositivi. I dispositivi si possono iscrivere ad un determinato *topic* (in modo analogo a Google Cloud Pub/Sub) così da essere notificati quando necessario. La flessibilità data dall'utilizzo dei *topic* supporta diversi modelli di comunicazione garantendo bassi tempi di latenza. È possibile, infatti, definire comunicazioni uno a uno, uno verso milioni o più nodi, sistemi di notifica broadcast e molto altro. Inoltre, consente di gestire in maniera più fine i

controlli di accesso di ogni singola connessione in base al *topic*, così da assicurarsi che i dispositivi e le applicazioni inviino e ricevano solo i dati necessari. Il *message broker* è un servizio gestito, in grado di scalare automaticamente in base al carico di lavoro. La sua implementazione si basa sull'utilizzo del protocollo di comunicazione *MQTT* nella versione v3.1.1, con alcune differenze. Per esempio non è supportato *QoS 2*. *IoT Core* supporta connessioni con dispositivi che utilizzano il protocollo MQTT o *MQTT over WebSocket* e che sono identificate da un *clientID* [53]. In aggiunta, un *client* può pubblicare un messaggio anche facendo una richiesta HTTP 1.0 o 1.1 verso una REST API, ma in questo caso, a differenza di MQTT, non è possibile utilizzare il campo *clientID* [54].

AWS IoT Core fornisce diverse interfacce per interagire con i dispositivi.

- AWS Command Line Interface (AWS CLI): è un'interfaccia a riga di comando che permette di eseguire comandi per AWS IoT su Windows, MacOS e Linux.
- AWS IoT SDK: include una serie di librerie, *open source* e non, per Android, iOS, Java, JavaScript, C++, Python e Embedded C e guide per sviluppatori per la creazione soluzioni IoT innovative utilizzando AWS IoT.
- AWS IoT Core per LoRaWAN (Low Power - Long Range Wide Area Network): consente di creare una rete LoRaWAN privata per connettere alla rete dispositivi e gateway a basso consumo in modo semplice e senza la necessità di gestire, configurare e manutenere il server della rete LoRaWAN.
- AWS IoT API: permette di creare applicazioni IoT a cui è possibile accedervi utilizzando HTTP o HTTPS,

Un'altro servizio interessante di AWS IoT è *Device shadow*, il quale rende disponibile lo stato di un dispositivo ad applicazioni o altri servizi anche se questo non è attualmente connesso. Questo è possibile creando *esplicitamente* uno stato ombra (*shadow*), che può essere aggiornato, rimosso e creato tramite le interfacce descritte precedentemente. Inoltre è possibile creare più di uno stato ombra per dispositivo, così da rendere più flessibile la connessione tra un dispositivo e diverse applicazioni o servizi [55].

AWS IoT Device Management

AWS IoT Device Management è il servizio AWS che offre un'interfaccia per organizzare, monitorare e gestire in modo semplice e intuitivo i dispositivi IoT. Di seguito vengono riportate alcune delle sue funzionalità [56].

- Registrazione in massa dei dispositivi connessi: AWS IoT Device Management aiuta a registrare nuovi dispositivi. Utilizzando la console di gestione IoT o le AWS API è possibile caricare dei template che in seguito verranno popolati con informazioni riguardanti il numero di serie del dispositivo, il produttore, certificati di identità *X.509* o policy di sicurezza. Quindi si può configurare l'intera *flotta* di dispositivi con queste informazioni, in pochi clic.

- Organizzazione dei dispositivi in gruppi: AWS IoT Device Management consente di raggruppare la *flotta* di dispositivi in una struttura gerarchica. Per esempio si possono raggruppare i dispositivi in base alla loro funzione, ai requisiti di sicurezza, o qualsiasi altra categoria. Inoltre è possibile automatizzare il processo di divisione in gruppi tramite *dynamic thing groups*, definendo dei criteri specifici. Tutto questo permette di gestire in maniera più efficiente ed efficace i dispositivi, che possono arrivare ad essere molto numerosi.

- Indicizzazione e ricerca dei dispositivi: Con AWS IoT Device Management è possibile eseguire delle interrogazioni sui gruppi di dispositivi per ottenere statistiche aggregate sulla base di informazioni sullo stato dei dispositivi, sulla connettività ed altre caratteristiche.

- Fleet Hub: AWS IoT Device Management mette a disposizione un'applicazione web automaticamente gestita, *Fleet Hub*, tramite la quale è possibile visualizzare ed interagire con la flotta di dispositivi connessi senza la necessità di scrivere codice.

Elaborazione e archiviazione dei dati

AWS mette a disposizione diversi servizi per l'elaborazione, l'archiviazione e l'analisi dei dati IoT.

AWS Rules Engine permette ai dispositivi IoT di interagire con gli altri servizi *AWS*, in particolare si occupa di valutare i messaggi in entrata pubblicati su *IoT Core*, trasformarli e consegnarli verso altri dispositivi o altri servizi cloud, in base alla logica di business definita tramite delle regole. Una regola può essere applicati ai dati di uno più dispositivi, eseguire una o più azioni in parallelo e instradare i messaggi agli *endpoint AWS* [57].

Per elaborare grandi flussi di dati in tempo reale è possibile utilizzare *Amazon Kinesis Data Stream*. Mentre *Amazon Simple Queue Service* (SQS) è un servizio di accodamento dei messaggi che consente di inviare, archiviare e ricevere messaggi a qualsiasi volume, senza perdere messaggi o richiedere la necessità di altri servizi.

Per quanto riguarda l'archiviazione dei dati, *AWS IoT Core* è direttamente connesso con *Amazon DynamoDB* e *AWS Simple Storage Service* (S3). Amazon DynamoDB è un servizio di database NoSQL completamente gestito, supporta l'accesso e la manipolazione dei dati da più regioni, utilizza politiche di caching e di backup. AWS S3, invece, è un servizio per l'archiviazione di grandi oggetti, tipicamente non strutturati, fino a 5TB. Supporta la replica per l'accesso da più regioni e il backup. AWS S3 è ottimizzato per avere un throughput elevato, gestire un elevato numero di richieste, in particolare su più oggetti contemporaneamente. Dall'altra parta, DynamoDB è ottimizzato per avere un basso tempo di latenza e lavorare su oggetti di piccola dimensione, fino a 400KB [58].

AWS IoT Analytics

AWS IoT Core include le funzionalità di *AWS IoT Analytics*, un servizio completamente gestito di AWS IoT pensato per semplificare l'esecuzione di analisi sofisisticate su un

grande volume di dati IoT. I quali sono tipicamente non strutturati, il che rende difficile ricavare informazioni utili con i classici strumenti di analisi e *business intelligence* per dati strutturati. AWS IoT Analytics filtra, trasforma e arricchisce i dati IoT prima di archiviarli in un *time series datastore*, ovvero un database ottimizzato per la gestione dati che rappresentano metriche ed eventi con *timestamp*. Quindi permette di eseguire interrogazioni SQL o utilizzare modelli preconfezionati per l'inferenza di Machine learning [59].

AWS IoT SiteWise

Un altro servizio per l'analisi dei dati IoT, integrato con IoT Core, è *AWS IoT SiteWise*. *AWS IoT SiteWise* è stato progettato per rendere più semplice la gestione, l'archiviazione, l'organizzazione e l'analisi dei dati provenienti da ambienti industriali, su larga scala. È possibile definire dei modelli che rappresentano accuratamente le relazioni tra i diversi dispositivi e le apparecchiature industriali, in modo da interpretare più facilmente i dati nel contesto dell'ambiente industriale. È possibile definire delle metriche personalizzate che vengono calcolate ad intervalli regolari e possono essere configurate per una risorsa o un gruppo di risorse. Inoltre, include automaticamente aggregati statistici comuni come somma, media e conteggio su più periodi di tempo. Tutte queste statistiche sono visualizzabili tramite *SiteWise Monitor*, una web-app completamente gestita per visualizzare e interagire con i dati operazionali senza la necessità di scrivere codice. SiteWise include anche un software, *SiteWise Edge*, che può essere eseguito in locale su tramite *AWS IoT Greengrass* per semplificare la raccolta, l'organizzazione e l'elaborazione dei dati in locale prima di inviarli in cloud [60].

Sicurezza

Per quanto riguarda la sicurezza, ogni dispositivo per comunicare con il *message broker* deve possedere delle credenziali e tutto il traffico deve essere criptato utilizzando il protocollo *TLS*. Il processo di autenticazione si basa sul certificato *X.509*, mentre tramite *AWS Identity and Access Management (IAM)* è possibile definire utenti, ruoli e gruppi in modo semplice. Inoltre, per la gestione dei permessi, AWS IoT mette a disposizione il servizio *AWS IoT Device Defender*. Questo servizio esamina costantemente ed in modo automatico le configurazioni riguardanti la sicurezza dei dispositivi IoT e si assicura che non si discostino da quelle che sono le *best practices*. Nel caso in cui AWS IoT Device Defender noti alcune anomalie invia un allarme cosicché sia possibile prendere provvedimenti. Ad esempio un picco di traffico in uscita può essere sintomo di un attacco di tipo *DoS*. AWS IoT Greengrass e FreeRTOS sono automaticamente integrati con AWS IoT Device Defender [61].

3.4.3 Azure IoT

Azure IoT consiste in una collezione di servizi, piattaforme *Cloud* ed *Edge* di *Microsoft* che consentono di connettere, monitorare e controllare molteplici risorse IoT contemporaneamente. Inoltre include sistemi operativi, servizi per la sicurezza dei dispositivi e per l'analisi dei dati in modo da aiutare le aziende a sviluppare, distribuire e gestire applicazioni IoT.

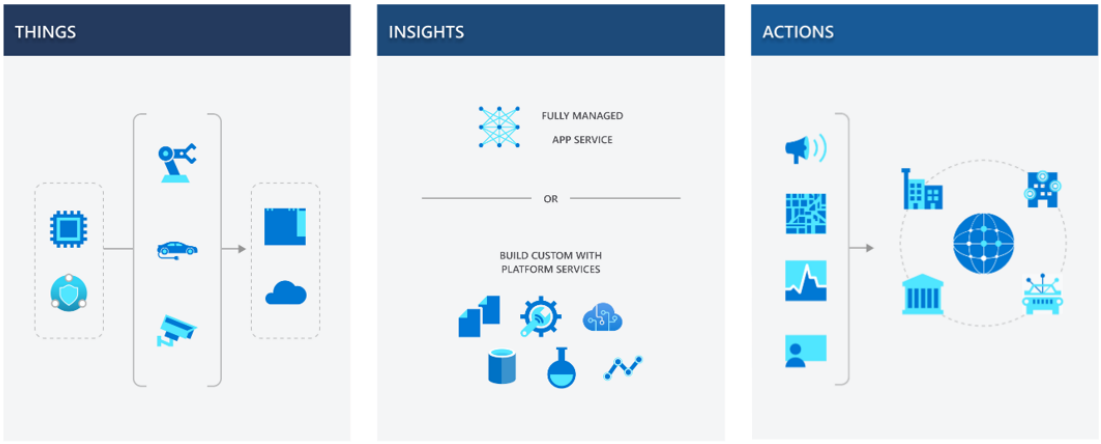

Figura 3.10: *— Azure IoT - Things, insights, actions (Fonte immagine: https://wiki.seeedstudio.com/Connect-Wio-Terminal-to-Microsoft-Azure-IoT-Central)*

> *"Azure IoT solutions involve events that generate insights to inform actions that improve a business or process. IoT solutions use events, insights, and actions to connect devices, or things, to cloud applications and achieve end-to-end scenarios. The terms thing and device both mean a connected physical device in an IoT solution."*
> *— IoT solutions conceptual overview[62]*

Things, insights, actions Le soluzioni sviluppate da *Microsoft* si basano sui seguenti tre componenti chiave [fig: 3.10] e su come i diversi servizi debbano interagire con essi [63].

- Things: gli oggetti fisici, come apparecchiature industriali, sensori, dispositivi che si connettono al cloud in modo permanente o intermittente e generano dati ed *eventi*.
- Insights: le informazioni collezionate dagli oggetti (*things*), che una volta analizzate si tramutano in conoscenza.
- Actions: il modo in cui le persone agiscono in seguito alla conoscenza ottenuta tramite l'analisi delle informazioni estratte per implementare la logica di business.

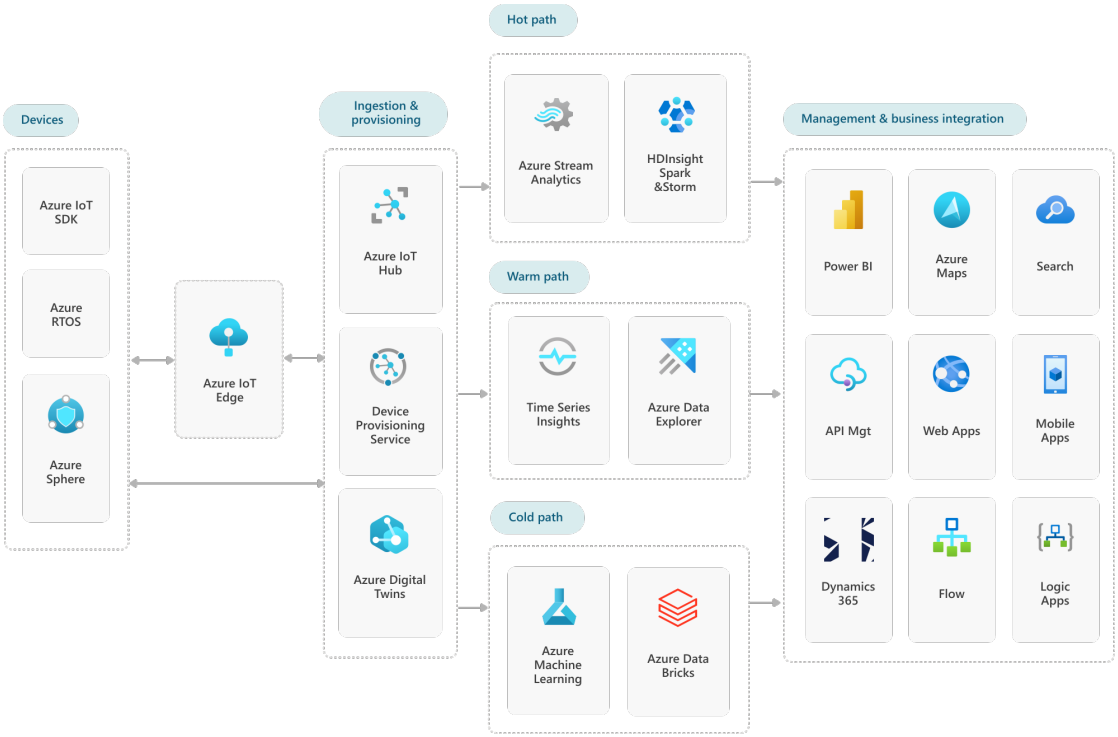

Figura 3.11: *— Azure IoT reference architecture (Fonte immagine: [64])*

Il diagramma [fig: 3.11] illustra i differenti servizi forniti da *Azure* per confezionare una soluzione IoT. Scegliere tra i numerosi componenti offerti non è affatto un compito semplice, per questo *Microsoft* consiglia di iniziare con la piattaforma *Azure IoT Central* [64].

Azure IoT Central

Iot Central è una soluzione IoT di tipo aPaaS (*application platform-as-a-service*), offre un ambiente pronto all'uso per lo sviluppo di soluzioni IoT e ne riduce i costi di sviluppo, gestione e manutenzione. Fornisce una interfaccia utente web interattiva personalizzabile, da cui è possibile monitorare le condizioni dei dispositivi, definire delle regole e gestire milioni di dispositivi e i loro dati durante tutto il ciclo di vita. In alternativa è possibile interagire con l'applicazione anche tramite REST API. IoT Central fornisce, inoltre, una serie di *template* per specifici settori come vendita al dettaglio e sanità, per accelerare il processo di sviluppo [65].

In alternativa, Azure IoT, mette a disposizione *IoT Hub* una soluzione *PaaS* che permette una maggiore flessibilità e possibilità di personalizzazione.

Azure IoT Hub

Azure IoT Hub è un servizio Cloud *PaaS* gestito che opera come *hub* centrale per lo scambio bidirezionale di messaggi tra applicazioni IoT e i dispositivi collegati, garantendo affidabilità e sicurezza. *IoT Hub* permette di gestire milioni di connessioni contemporaneamente e l'ingente volume di eventi e messaggi generati dai dispositivi, scalando automaticamente. Inoltre fornisce meccanismi per l'autenticazione dei dispositivi ed il monitoraggio delle connessioni. Supporta diversi modelli di messaggistica come messaggi telemetrici *device-to-cloud*, messaggi di tipo *request-reply* per controllare i dispositivi in remoto dalla piattaforma cloud e caricamento di file dai dispositivi. In aggiunta, *IoT Hub* è integrato nativamente con tutti i servizi di *Azure cloud*, il chè consente di implementare una soluzione IoT completa [66]. Per esempio, è possibile usare:

- *Azure Event Grid* che permette di rispondere in modo veloce e affidabile ad eventi critici.
- *Azure Machine Learning* per aggiungere modelli di Machine learning e intelligenza artificiale alla propria soluzione.
- *Azure Stream Analytics* per eseguire in tempo reale elaborazioni su un flusso di dati IoT continuo, su larga scala.

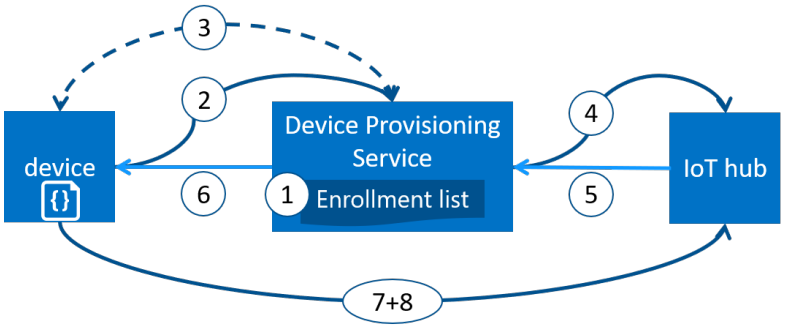

Figura 3.12: *— Azure IoT Hub Device Provisioning Service (Fonte immagine: https://docs.microsoft.com/en-us/azure/iot-dps/about-iot-dps*

Gestione dei dispositivi La registrazione dei dispositivi è gestita tramite IoT HuB *Device Provisioning Service* (DPS), un servizio che permette di collegare i dispositivi ad *IoT Hub* senza richiedere un intervento umano. Come si può vedere in figura [fig: 3.12], il dispositivo contatta un *endpoint* DPS inviando i suoi dati per l'identificazione, il servizio registra il dispositivo in IoT Hub e memorizza lo stato del dispositivo desiderato, in *Azure IoT* si parla di *twin state*. In particolare, grazie alla piattaforma *Azure Digital Twin* è possibile creare delle rappresentazioni digitali dei dispositivi. Un *device twin state* consiste in un documento *JSON* che memorizza informazioni sullo stato

del dispositivo come metadata, configurazioni e condizioni. Queste informazioni possono essere usate per sincronizzare le condizioni e le configurazioni del dispositivo tra il dispositivo stesso e *IoT Hub* o per eseguire interrogazioni e indirizzare operazioni di lunga durata [67]. Inoltre, *Azure IoT* fornisce diversi strumenti e librerie che facilitano l'integrazione dei dispositivi e gateway con *IoT Hub*, disponibili per i seguenti linguaggi di programmazione: .NET, C, Java, Node.js, Python e iOS.

Protocolli di comunicazione Tra i diversi protocolli di comunicazione, *Azure IoT Hub* supporta MQTT/MQTT *over WebSocket*, AMPQ/AMPQ *over WebSocket* e HTTPS. Dato che MQTT e HTTPS supportano un solo *device ID* per ogni connessione TLS, *Azure IoT* consiglia l'utilizzo di AMPQ quando è necessario eseguire *multiplexing* dei messaggi verso diversi dispositivi, uno scenario d'utilizzo sono per esempio i dispositivi gateway che connettono al cloud altri dispositivi IoT. Negli altri casi è sempre consigliato l'utilizzo del protocollo MQTT, limitando l'utilizzo di HTTPS ai soli dispositivi che non supportano gli altri protocolli [68].

Sicurezza Ogni *IoT Hub* utilizza un registro, detto *identity registry*, che è utilizzato per archiviare informazioni riguardanti l'identità dei dispositivi ed i moduli che è possibile connettere all'*Hub*. Tutte le connessioni, come visto in precedenza, *devono* partire dai dispositivi che richiedono di connettersi all'*Hub*. Solo i dispositivi con un identità che corrisponde ad una voce dell'*identity registry* si possono connettere. Le connessioni utilizzano il protocollo di TLS per criptare i dati e certificati X.509 per l'autenticazione. Inoltre *Azure IoT* mette a disposizione *Azure Active Directory*, ovvero una soluzione Cloud-based per la gestione dei permessi di accesso e dell'identità dei dispositivi. Questo servizio permette di gestire le autorizzazioni con una maggiore facilità e con un maggiore livello di dettaglio grazie alla possibilità di definire ruoli, permessi di gruppo, personalizzare i permessi per ogni singolo dispositivo o in base alla loro *categoria* [69].

Things

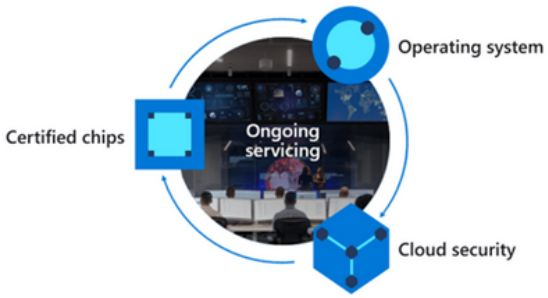

Figura 3.13: *— Azure Sphere (Fonte immagine: https://azure.microsoft.com/it-it/services/azure-sphere*

Azure IoT supporta una vasta gamma di dispositivi e mette a disposizione diversi software per dispositivi.

Azure RTOS *Azure RTOS* è un sistema operativo *real-time*, semplifica l'utilizzo di microcontrollori, sistemi *embedded* ed il modo in cui si connettono al Cloud o ad altri dispositivi della rete locale, aumenta la portabilità delle applicazioni grazie all'introduzione di uno strato di astrazione tra hardware e applicazioni, gestisce in modo del tutto invisibile l'allocazione delle risorse ed è ottimizzato per eseguire diversi *thread* garantendo il minimo tempo di risposta e prestazioni elevate anche con risorse limitate [70].

Azure Sphere *Azure Sphere* è una piattaforma applicativa di alto livello pensata per aggiungere uno strato di sicurezza ai dispositivi IoT. È costituito da un microcontrollore, con chip certificati, per fornire connettività ed hardware affidabile; un sistema operativo che aggiunge strati di protezione e continui aggiornamenti di sicurezza [71]; un *broker* che gestisce le comunicazioni tra dispositivo e Cloud in grado di rilevare eventuali minacce.

Azure IoT Edge È possibile effettuare elaborazioni in locale utilizzando il servizio *Azure IoT Edge*, il quale è composto da tre elementi. *IoT Edge modules*, ovvero unità di esecuzione racchiuse in *container Docker* che consentono di implementare la logica di business ed eseguire servizi Azure, servizi di terze parti o il proprio codice ai bordi della rete. *IoT Edge runtime* che è eseguito su ogni dispositivo edge e gestisce i moduli distribuiti sui diversi dispositivi. *IoT Edge cloud-based interface* che abilita la comunicazione con il cloud, consentendo di gestire e monitorare in remoto i dispositivi edge. Inoltre, con *Azure IoT Edge* è possibile distribuire complesse elaborazioni per rispondere agli eventi, come inferenza di Machine learning, riconoscimento di immagini, senza la necessità di scrivere codice.

Insights

Una volta che i dispositivi sono connessi ad *IoT Hub*, è possibile elaborare i dati che essi rilevano o generano. Come si può vedere in figura [fig: 3.11], *Azure IoT* divide l'elaborazione dei dati in 3 percorsi (*path*): *hot path* o percorso critico, *warm path* o percorso ad accesso caldo e *cold path* o percorso ad accesso sporadico. I tre percorsi differiscono per requisiti sui tempi di latenza e frequenza di accesso ai dati [64].

- **Hot path**: si analizzano i dati quasi in tempo reale rispetto all'arrivo. Tipicamente si tratta di dati telemetrici che richiedono un'elaborazione con una latenza molto bassa. In questo percorso è consigliato l'utilizzo di motori per il processamento di flussi dati, come i servizi *Azure Stream Analytics* o *HDInsight*. Una volta elaborati, è possibile archiviare tali dati tramite *Azure Cosmos DB*, un database NoSQL che garantisce l'archiviazione di dati non strutturati, come dati JSON, garantendo una bassa latenza. In alternativa si può utilizzare *Azure SQL DB* per archiviare dati che per loro natura sono rappresentabili attraverso schemi relazionali.

- **Warm path**: si effettuano elaborazioni sui dati che possono richiedere ritardi più lunghi per ottenere informazioni più dettagliate. In questo caso è consigliato l'utilizzo di servizi come *Azure Time Series Insights* e *Azure Data Explorer*. *Time series Insights* consente di archiviare ed elaborare grandi moli di dati che corrispondono ad una *serie temporale*, ovvero una serie di dati ordinati rispetto al tempo che descrivono la dinamica di un certo fenomeno nel tempo.

- **Cold path**: si eseguono elaborazioni di tipo batch, ovvero su dati già catalogati, ad intervalli più lunghi (ogni ora o ogni giorno). In questo caso si lavora su volumi di dati enormi, strutturati e non, che possono essere archiviati in *Azure Data Lake* che consente di operare su volumi di dati nell'ordine dei *petabyte* garantendo un *throughput* elevato. Una caratteristica di *Data Lake* è la suddivisione gerarchica del *namespace* che consente di accedere o effettuare operazioni come la rimozione in maniera efficiente. Una volta che i dati sono stati memorizzati è possibile sfruttare servizi come *Azure Machine Learning* o *Azure DataBricks*, rispettivamente, per creare e addestrare modelli di Machine learning in cloud o eseguire analisi dettagliate.

Actions

È possibile utilizzare le informazioni dettagliate ottenute a seguito di un'elaborazione sui dati raccolti per gestire e controllare l'ambiente in cui si opera, implementare soluzioni di *business intelligence* o creare applicazioni IoT sulla base delle conoscenze e delle informazioni acquisite. Per questi scopi, *Azure* mette a disposizione diversi servizi, di seguito ne vengono presentati alcuni.

- *Power BI* consente di implementare soluzioni di *business intelligence* unificando i dati tra più sorgenti e sfruttare l'intelligenza artificiale per prendere decisioni guidate da dati.

- *Azure Maps* consente di creare applicazioni Web e per dispositivi mobili per il tracciamento della posizione usando servizi geospaziali (ricerca, mappe, routing, rilevamento e traffico), API e SDK.

- *Azure API Management* fornisce una piattaforma unica per la gestione di tutte le API.

- *Azure Web Apps* consente di distribuire applicazioni Web, scalando automaticamente.

3.4.4 Considerazioni finali

Le tre piattaforme prese in esame permettono di sviluppare soluzioni IoT *end-to-end* con il minimo sforzo di gestione, grazie ad un ambiente completamente integrato con diversi servizi Cloud per la gestione di dispositivi IoT, l'archiviazione e l'analisi dei dati e l'uso di tecniche di Machine learning. Sceglierne una non è affatto un compito semplice, dato

l'elevato numero di servizi offerti e la sovrapposizione di alcune componenti tra un provider e l'altro. Spesso, infatti, i componenti principali sono similari. Per esempio, AWS IoT Core, Azure IoT Hub e Google IoT Core offrono tutti funzionalità Cloud ed Edge, oltre ad un ambiente per la gestione dei dispositivi. Allo stesso modo, ogni piattaforma Cloud-IoT è decorata da strumenti di analisi, sia in *streaming* che *batch*, e strumenti per la visualizzazione. Questo fà sì che utenti, aziende, o più in generale, organizzazioni propendano per scegliere la piattaforma Cloud con cui hanno già familiarità, sebbene vi siano alcune differenze sul loro funzionamento.

AWS IoT possiede la più alta quota di mercato (40%) [fig: 3.4], nata nel 2015 è la piattaforma più matura. Tra le piattaforme selezionate è quella che offre la più vasta gamma di servizi e molteplici strumenti per la connettività sia device-to-device che device-to-cloud. AWS IoT Device Management è il servizio più completo per la gestione di massa dei dispositivi e la loro organizzazione in gruppi ed insieme al servizio AWS IoT SiteWise rendono AWS IoT la piattaforma più adatta per applicazioni IoT in ambito industriale. Inoltre, mette a disposizione più di 218 *edge location*, ovvero *data center* ai bordi della rete per diminuire i tempi di latenza e implementare politiche di *caching*. Tuttavia, AWS è la piattaforma mediamente più costosa rispetto alle altre due *competitor*. Azure IoT presenta una interfaccia più facile da utilizzare, supporta il protocollo AMQP oltre ai protocolli standard HTTP e MQTT e fornisce librerie per .NET a differenza di Google Cloud IoT e AWS IoT. Azure è probabilmente la scelta più adatta in ambito aziendale, soprattutto nel caso in cui si utilizzino già i prodotti *business* di Microsoft e i suoi servizi Cloud. *Azure IoT*, infatti, è stato sviluppato pensando proprio a questa interconnessione tra software e applicazioni e mette a disposizione strumenti avanzati per la visualizzazione dei dati e per implementare soluzioni di *business intelligence*. Inoltre, Azure, offre un elevato supporto per strategie di Hybrid Cloud a differenza di AWS, la cui attenzione è focalizzata sul cloud pubblico. Il progetto Google Cloud IoT è stato lanciato solo nel 2018, la piattaforma di Google è la più recente e quindi la meno rodata. Rispetto a quelle di Microsoft e Amazon fornisce un numero minore di servizi e non supporta la comunicazione via WebSocket. Tuttavia, Google si fa forte dell'esperienza acquisita negli anni su AI e Machine learning e fornisce servizi più completi per l'inferenza di Machine learning, completamente integrati con Google Cloud IoT. Di seguito, la tabella 3.1 illustra i punti di forza ed i punti deboli delle piattaforme analizzate.

In conclusione, AWS IoT e Azure IoT sembrano avere una marcia in più rispetto alla proposta di Google. Dalle analisi effettuate, AWS IoT risulta essere la piattaforma più affidabile e completa grazie alla vasta gamma di servizi offerti, molti dei quali dedicati unicamente al mondo IoT. Ciononostante, Azure IoT rappresenta una valida alternativa, soprattutto nel caso in cui si progetta un'infrastruttura in cui l'interoperabilità tra *data center* privati e pubblici è un requisito.

Tabella 3.1: *Piattaforme a confronto: punti di forza e punti deboli*

	PUNTI DI FORZA	PUNTI DEBOLI
AWS IoT	• Piattaforma matura • Capacità di elaborazione elevata • Gamma di servizi più vasta • Offre servizi dedicati al mondo *industrial IoT* • Gestione dei dispositivi più semplice • Più di 218 edge *locations*	• Può generare confusione a causa dei numerosi servizi offerti • Hybrid Cloud poco supportato • Su larga scala è mediamente più costosa
Azure IoT	• Semplice integrazione con gli altri servizi *Microsoft* • Elevato supporto del Cloud ibrido • Servizi di business intelligence avanzati • Supporta il protocollo AMQP • Offre librerie per .NET	• Offre meno servizi rispetto a AWS
Google Cloud IoT	• Semplice integrazione con i prodotti *Google* • Servizi avanzati per l'apprendimento automatico	• Piattaforma nuova e meno rodata • Pochi servizi dedicati all'IoT rispetto alla concorrenza • Non supporta MQTT over WebSocket

Conclusioni

L'Internet of Things è una tecnologia in forte espansione, ogni anno vengono distribuiti miliardi di dispositivi che interagendo con l'ambiente circostante rilevano e generano un volume di dati enorme. Diventa, quindi, fondamentale progettare un'infrastruttura adeguata che fornisca strumenti per collezionare ed elaborare tali dati, controllare i dispositivi IoT e gestire lo scambio di messaggi sia tra i dispositivi che tra applicazioni e dispositivi, garantendo comunicazioni sicure.

L'obiettivo che ci si è proposti all'inizio di questo lavoro di tesi è stato quello di approfondire i motivi che portano ad una integrazione tra Cloud computing ed Internet of Things, i vantaggi ed i limiti che una soluzione di questo tipo può portare attraverso lo studio di entrambi i paradigmi ed in particolare analizzare le tre principali piattaforme Cloud pensate per L'Internet of Things disponibili sul mercato, ovvero AWS IoT, Azure IoT e Google Cloud IoT.

In questo elaborato si sono discussi i diversi modelli di servizio e di distribuzione definiti dal Cloud computing. Per ogni modello sono stati illustrati i principali vantaggi e i casi d'uso. Si è visto che il Cloud permette un'ampia personalizzazione, sia in termini di servizio offerto che di divisione di responsabilità tra *cloud provider* e utente finale. Inoltre sono stati evidenziati i vantaggi economici che si possono ottenere usufruendo dei servizi in Cloud piuttosto che utilizzando risorse *on-premises*. Il cloud permette di ridurre i costi d'investimento iniziali, di manutenzione e di gestione, e consente di reindirizzare il capitale verso gli investimenti riguardanti il *core business* delle aziende. Inoltre, grazie alla sua flessibilità consente di evitare i rischi di *over-provisioning* e *under-provisioning*. Successivamente, si è illustrato il paradigma IoT e si sono approfonditi i protocolli di comunicazione e gli standard web maggiormente utilizzati in ambito IoT. Più nello specifico sono stati analizzati i protocolli a livello applicativo, classificandoli in protocolli di tipo *request-response* e *publish-subscribe*. Dall'analisi è emerso che i protocolli *publish-subscribe* permettono una gestione più efficiente della comunicazione, evitando di eseguire un continuo polling delle risorse. In particolare, il protocollo MQTT è il più adatto quando si ha che fare con dispositivi a bassa potenza che generano dati

telemetrici con una elevata frequenza e dispongono di capacità elaborativa limitata.

Nel terzo capitolo si sono evidenziati i vantaggi che si possono ottenere dall'integrazione tra Cloud e IoT ma anche i limiti di una soluzione totalmente centralizzata sui data center Cloud, in particolare in applicazioni IoT che richiedono l'elaborazione di tanti dati con il minimo tempo di latenza. Da questa evidenza si è visto che è stato necessario modificare la classica architettura IoT *Cloud-centrica* introducendo un nuovo modello di elaborazione, l'Edge computing. Quest'ultimo, mediante l'utilizzo di dispositivi intelligenti distribuiti sui bordi della rete consente di effettuare elaborazioni lungo il percorso dati tra dispositivi IoT e data center Cloud. Dall'analisi delle piattaforme Cloud prese in esame si è visto che l'architettura proposta dai *Cloud provider*, è un compromesso tra i due modelli Cloud ed Edge computing. In particolare, le due soluzioni vengono implementate per operare in modo coordinato e operativo. Per ogni piattaforma è stata descritta l'architettura generale, suddividendola in più strati: strato di rilevamento, di trasporto, di elaborazione e applicativo. Quindi si è entrati nel dettaglio di ogni livello, analizzando i numerosi servizi offerti ed il modo in cui possono essere combinati per implementare una soluzione IoT. Dall'analisi si è constatato che le piattaforme Cloud consentono di: gestire e monitorare i dispositivi IoT attraverso servizi come *Device manager* di Google Cloud IoT, *Device management* di AWS IoT o *DPS* di Azure IoT; gestire comunicazioni bidirezionali attraverso servizi orientati ai messaggi per l'acquisizione dei dati e servizi *event-driven* per implementare la logica di business attraverso funzioni innescate da determinati eventi. Inoltre, garantiscono comunicazioni sicure mediante l'utilizzo di protocolli crittografici e fornendo servizi di autenticazione e autorizzazione che facilitano la gestione di ruoli e gruppi, anche grazie ad interfacce grafiche intuitive e semplici da utilizzare. Ogni piattaforma è integrata con diversi servizi Cloud che consentono di filtrare, processare, archiviare e analizzare l'ingente volume di dati generati dai dispositivi IoT, sfruttando a pieno la potenza delle risorse Cloud.

Scegliere una piattaforma piuttosto che un'altra non è semplice, difatti in molti dei loro componenti sembrano essere piuttosto simili. Inoltre, le aziende potrebbero essere confuse dai numerosi servizi messi a disposizione e quindi optare per la piattaforma con cui hanno già familiarità. Dall'analisi effettuata la piattaforma di Amazon, AWS IoT, sembra essere quella più solida, più matura e che offre un ventaglio più ampio di servizi, ad ogni modo, per prendere una decisione consapevole è necessario conoscere il dominio di interesse e valutare le caratteristiche dello scenario applicativo.

In breve, dallo studio condotto è emerso che il Cloud computing è la *chiave* per abilitare l'Internet of Things su larga scala grazie a piattaforme flessibili e modulari, accessibili ovunque da una vasta gamma di dispositivi. Si è visto che l'offerta dei maggiori *cloud provider* è ormai solida e pronta per rispondere alle criticità e alle necessità che i nuovi scenari applicativi dell'Internet of Things prospettano per il futuro. Inoltre, si è evidenziato come l'introduzione di dispositivi intelligenti ai bordi della rete può favorire l'elaborazione di dati in tempo reale e aiutare a ridurre l'eccessivo consumo di banda Internet. Si prevede quindi che gli sviluppi futuri vadano in questa direzione, ovvero un'infrastruttura IoT su tre livelli che lega dispositivi IoT, dispositivi *edge* o *micro data center* ai bordi della rete e data center Cloud. In tal senso, si può pensare a soluzioni

dove i dispositivi *edge* fanno da filtro, eseguendo una prima elaborazione dell'ingente quantità di dati generato dai dispositivi IoT o eseguono in locale funzioni che richiedono una elaborazione con un minimo tempo di latenza come, ad esempio, analisi in tempo reale di video e immagini eseguite in locale attraverso inferenza di Machine learning su modelli addestrati precedentemente in Cloud.

Ringraziamenti

A conclusione di questo elaborato, desidero dedicare questo spazio a tutte le persone che hanno contribuito, con il loro instacabile supporto, alla sua stesura e che mi sono stati vicini durante l'intero percorso formativo.

Ringrazio, innanzitutto, il relatore di questa tesi Marozzo Fabrizio, che in questi mesi di lavoro ha saputo guidarmi, con preziosi consigli, nelle ricerche e nella stesura dell'elaborato dimostrando sempre grande disponibilità.

Ringrazio tutti quei professori del corso di studi che durante questi tre anni hanno saputo stimolarmi. Ciascuno di loro, in modo speciale, personale, unico, ha contribuito al mio percorso di crescita, prima personale e poi professionale.

Un grazie ai miei genitori, che sono il mio punto di riferimento. Grazie per avermi sempre sostenuto e per avermi insegnato, con le parole e con l'esempio, ciò che è giusto e ciò che non lo è.

Un grazie ai miei fratelli, Diego e Mattia, per avermi guidato in ogni scelta importante della mia vita.

Ringrazio tutti i membri del Master-Slave: Stefano, Fabrizio, Alessandro, Giulio, Ciccio, Marco, Enrico. Amici, colleghi e compagni di viaggio di questo magnifico percorso universitario. Persone intelligenti e motivate dalle quali, grazie a un confronto costante e proficuo, ho potuto imparare molto ed ho capito sia i miei difetti, che i miei pregi. Il raggiungimento di questa mio piccolo obiettivo è anche merito vostro.

Infine, desidero ringraziare tutte le altre persone che in questi anni mi sono state vicino.

Bibliografia

[1] Domenico Talia, Paolo Trunfio e Fabrizio Marozzo. "Chapter 2 - Introduction to Cloud Computing". In: *Data Analysis in the Cloud*. A cura di Domenico Talia, Paolo Trunfio e Fabrizio Marozzo. Computer Science Reviews and Trends. Boston: Elsevier, 2016, pp. 27–43. ISBN: 978-0-12-802881-0. DOI: https://doi.org/10.1016/B978-0-12-802881-0.00002-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128028810000020.

[2] Peter Mell, Tim Grance et al. "The NIST definition of cloud computing". In: (2011).

[3] S. Garfinkel et al. *Architects of the Information Society: 35 Years of the Laboratory for Computer Science at MIT*. Architects of the Information Society: Thiry-five Years of the Laboratory for Computer Science at MIT. MIT Press, 1999. ISBN: 9780262071963. URL: https://books.google.it/books?id=Fc7dkLGLKrcC.

[4] *BCS History of the cloud*. https://www.bcs.org/content-hub/history-of-the-cloud/. Created: 19 Mar 2019.

[5] Thomas Erl, Ricardo Puttini e Zaigham Mahmood. *Cloud Computing: Concepts, Technology & Architecture*. 1st. USA: Prentice Hall Press, 2013. ISBN: 0133387526.

[6] *Microsoft Azure What is PaaS?* https://azure.microsoft.com/it-it/overview/what-is-paas/.

[7] Thomas Erl, Ricardo Puttini e Zaigham Mahmood. *Cloud Computing: Concepts, Technology & Architecture*. 1st. USA: Prentice Hall Press, 2013. ISBN: 0133387526.

[8] *Red Hat - Automazione cos'é il provisioning?* https://www.redhat.com/it/topics/automation/what-is-provisioning.

[9] *IBM What is virtualization?* https://www.ibm.com/cloud/learn/virtualization-a-complete-guide. Created: 19 Jun 2019.

[10] Ergin Bayrak, John Conley e Simon Wilkie. "The Economics of Cloud Computing". In: *Korean Economic Review* 27 (dic. 2011), pp. 203–230.

[11] Armbrust et al. “Above the Clouds: A Berkeley View of Cloud Computing”. In: (gen. 2009).

[12] *Architectural Considerations in Smart Object Networking*. https://www.rfc-editor.org/rfc/rfc7452.txt.

[13] *The Internet of Things at the IETF*. https://www.ietf.org/topics/iot/.

[14] *Overview of the Internet of Things*. https://www.itu.int/rec/T-REC-Y.2060-201206-I.

[15] Luigi Atzori, Antonio Iera e Giacomo Morabito. “The Internet of Things: A survey”. In: *Computer Networks* 54.15 (2010), pp. 2787–2805. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2010.05.010. URL: https://www.sciencedirect.com/science/article/pii/S1389128610001568.

[16] *That ‘Internet of Things’ Thing*. https://www.rfidjournal.com/that-internet-of-things-thing.

[17] *Radio Frequency Identification (RFID)*. https://www.fda.gov/radiation-emitting-products/electromagnetic-compatibility-emc/radio-frequency-identification-rfid.

[18] *IoT history*. https://www.postscapes.com/iot-history/.

[19] *Industria 4.0 e Smart factory: il ruolo dell’Internet of Things*. https://www.mesaconsulting.eu/it/44-blog-it/innovazione/253-industria-4-0-e-smart-factory-il-ruolo-dell-internet-of-things.

[20] *Disruptive Civil Technologies - Six Technologies With Potential Impacts on US Interest Out to 2025*. https://irp.fas.org/nic/disruptive.pdf.

[21] Satyajit Sinha. *State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion*. https://iot-analytics.com/number-connected-iot-devices/. 2021.

[22] Knud Lasse Lueth. *Top 10 IoT applicaitons in 2020*. https://iot-analytics.com/top-10-iot-applications-in-2020/. 2020.

[23] *IoT e Utility 4.0: Enel sceglie il Cloud di AWS*. https://www.internet4things.it/industry-4-0/iot-e-utility-4-0-enel-sceglie-il-cloud-di-aws/. 2017.

[24] Blake Ives, Kathy Cossick e Dennis Adams. “Amazon Go: Disrupting retail?” In: *Journal of Information Technology Teaching Cases* 9 (mar. 2019), p. 204388691881909. DOI: 10.1177/2043886918819092.

[25] Iqra Zain, Abu Rehan e Javed Ashraf. “Internet of Things”. In: 2020.

[26] *MQTT Version 3.1.1 Specification*. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[27] *MQTT Client and Broker and MQTT Server and Connection Establishment Explained - MQTT Essentials: Part 3*. https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment.

[28] *MQTT Publish, Subscribe & Unsubscribe - MQTT Essentials: Part 4*. https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/.

[29] Zach Shelby, Klaus Hartke e Carsten Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. Giu. 2014. DOI: 10.17487/RFC7252. URL: https://rfc-editor.org/rfc/rfc7252.txt.

[30] Alessio Botta et al. "On the integration of cloud computing and internet of things". In: *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014* (dic. 2014), pp. 23–30. DOI: 10.1109/FiCloud.2014.14.

[31] *What is the role of Cloud computing in IoT*. https://contenteratechspace.com/blog/what-is-the-role-of-cloud-computing-in-iot.

[32] Paola Pierleoni et al. "Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison". In: *IEEE Access* 8 (2020), pp. 5455–5470. DOI: 10.1109/ACCESS.2019.2961511.

[33] *4 layers of the Internet of Things*. https://www.jigsawacademy.com/4-layers-of-the-internet-of-things/.

[34] *What is data abstraction*. https://whatis.techtarget.com/definition/data-abstraction.

[35] Daniel Happ et al. "Meeting IoT platform requirements with open pub/sub solutions". In: *Annals of Telecommunications* 72 (2017), pp. 41–52.

[36] *4 layers of the Internet of Things*. https://datafloq.com/read/self-driving-cars-create-2-petabytes-data-annually/172/.

[37] Weisong Shi et al. "Edge computing: Vision and challenges". In: *IEEE internet of things journal* 3.5 (2016), pp. 637–646.

[38] *Edge Computing: l'anima informatica della Internet of Things*. https://tech4future.info/edge-computing-iot/.

[39] Eclipse foundation. "The Eclipse Foundation Releases Results from the 2020 IoT Developer Survey". In: ().

[40] *IoT Core*. https://cloud.google.com/iot-core.

[41] *Beam programming model*. https://cloud.google.com/dataflow/docs/concepts/beam-programming-model.

[42] *Introduction to Google Cloud Dataflow*. https://cloudacademy.com/course/introduction-to-google-cloud-dataflow/introduction-42/.

[43] *Writing event-driven functions*. https://cloud.google.com/functions/docs/writing#event-driven_functions.

[44] *Telemetry - Data collection - Events*. https://firefox-source-docs.mozilla.org/toolkit/components/telemetry/collection/events.html.

[45] *What are cloud regions*. https://www.lucidchart.com/blog/what-are-cloud-regions.

[46] *Google Cloud SQL vs Cloud Datastore vs BigTable vs BigQuery vs Spanner*. https://weidongzhou.wordpress.com/2017/06/10/google-cloud-sql-vs-cloud-datastore-vs-bigtable-vs-bigquery-vs-spanner/.

[47] *BigTable vs BigQuery: What's the difference?* https://cloud.google.com/blog/topics/developers-practitioners/bigtable-vs-bigquery-whats-difference.

[48] *Vertex AI*. https://cloud.google.com/vertex-ai.

[49] *Device security*. https://cloud.google.com/iot/docs/concepts/device-security.

[50] *AWS IoT Greengrass*. https://aws.amazon.com/it/greengrass/.

[51] *FreeRTOS*. https://aws.amazon.com/it/freertos/.

[52] *AWS IoT Greengrass - Features*. https://aws.amazon.com/greengrass/features/.

[53] *AWS IoT - MQTT*. https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html.

[54] *AWS IoT - HTTP*. https://docs.aws.amazon.com/iot/latest/developerguide/http.html.

[55] *AWS IoT - Device shadow*. https://docs.aws.amazon.com/iot/latest/developerguide/iot-device-shadows.html.

[56] *AWS IoT Device management*. https://aws.amazon.com/it/iot-device-management/features/.

[57] *AWS IoT - Rules Engine*. https://aws.amazon.com/about-aws/whats-new/2018/07/aws-iot-rules-engine-now-supports-step-functions-action/.

[58] *Amazon S3 vs Amazon DynamoDB*. https://hevodata.com/learn/amazon-s3-vs-dynamodb.

[59] *AWS IoT Analytics*. https://docs.aws.amazon.com/iotanalytics/latest/userguide/welcome.html.

[60] *AWS IoT SiteWise*. https://www.amazonaws.cn/en/iot-sitewise/features/.

[61] *AWS IoT - Device Defender*. https://aws.amazon.com/iot-device-defender/.

[62] *Azure IoT - Introduction to solutions*. https://docs.microsoft.com/en-us/azure/architecture/example-scenario/iot/introduction-to-solutions.

[63] *Azure IoT - Overview*. https://azure.microsoft.com/en-us/overview/iot/#overview.

[64] *Azure IoT reference architecture*. https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot.

[65] *Azure IoT Central*. https://docs.microsoft.com/en-us/azure/iot-central/.

[66] *Azure IoT Hub*. https://docs.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub.

[67] *Azure IoT Device Twin*. https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-device-twins.

[68] *Azure IoT Hub - Choose a device communication protocol*. https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-protocols.

[69] *Control access to IoT Hub*. https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security.

[70] *Azure RTOS*. https://azure.microsoft.com/en-us/services/rtos.

[71] *Azure Sphere*. https://docs.microsoft.com/it-it/azure-sphere/product-overview/what-is-azure-sphere.