Autenticazione, autorizzazione e firme digitali

Diffie-Hellman e attacchi man-in-the-middle

Lo scambio di chiavi Diffie-Hellman serve per stabilire una chiave condivisa su un canale non sicuro. La sicurezza si basa sul fatto che una parte dei dati è pubblica e una parte resta privata. Tuttavia, Diffie-Hellman è vulnerabile agli attacchi man-in-the-middle, in cui un attaccante intercetta e modifica la comunicazione senza che le parti se ne accorgano.

Firewall e criteri di default

Il criterio di default-deny di un firewall prevede che tutto sia vietato a priori, tranne ciò che è esplicitamente consentito. Questo aumenta la sicurezza limitando solo al traffico strettamente necessario.

Crittografia simmetrica e asimmetrica

La crittografia simmetrica utilizza la stessa chiave, mantenuta segreta, per cifrare e decifrare i messaggi (es. DES, AES). La crittografia asimmetrica usa una coppia di chiavi (pubblica e privata) per cifrare e decifrare o per firmare digitalmente (es. RSA, ElGamal). Spesso si usa l'asimmetrica per scambiare una chiave segreta che poi viene usata per comunicare in modo simmetrico, unendo velocità ed elevata sicurezza.

Funzioni hash

Una funzione hash crittografica genera una stringa di lunghezza fissa indipendentemente dalla dimensione del messaggio di input. Serve a garantire l'integrità dei dati. Non è reversibile: non si può ottenere l'input dal valore hash.

Cifrari e operazioni AES

Nel cifrario di Vigenere si usa una chiave per cifrare un messaggio sommando le lettere del testo con quelle della chiave. AES usa l'operazione SubBytes, che è una sostituzione byte per byte basata su una tabella predefinita.

VPN e sicurezza aziendale

Una VPN (Virtual Private Network) garantisce privacy, anonimato e sicurezza dei dati aziendali, permettendo di proteggere le comunicazioni su reti non sicure.

Virus e worm

Un virus necessita di un programma ospite per attivarsi, mentre un worm è in grado di replicarsi e diffondersi autonomamente senza bisogno di legarsi ad altri file eseguibili.

Firma digitale e firma cieca

Una firma digitale garantisce l'autenticità e l'integrità di un messaggio. Se Alice firma un messaggio con la sua chiave privata, chiunque può verificarne l'autenticità usando la chiave pubblica. La firma cieca è una firma generata senza che il firmatario conosca il contenuto del messaggio, utile ad esempio per sistemi di voto elettronico.

Antivirus e falsi positivi

Un falso positivo (file innocuo rilevato come minaccia) può potenzialmente danneggiare il sistema operativo se porta alla cancellazione di file di sistema.

SQL Injection

Contro attacchi SQL Injection è buona pratica ridurre i privilegi degli account che interagiscono con il database e validare accuratamente l'input.

Phishing

Il phishing è una tecnica che mira a ingannare l’utente fingendosi un’entità affidabile per ottenere informazioni personali, come credenziali o dati bancari.

Password manager

Un password manager memorizza e gestisce password complesse e offre funzioni di completamento automatico dei form sui siti sicuri. È buona norma usare password diverse per ogni account.

Certificati digitali e SSL

SSL è un esempio di certificato digitale usato per stabilire connessioni sicure (HTTPS) autenticando le parti e cifrando i dati trasmessi.

RSA e cifratura

In RSA la cifratura avviene con la formula C = M^e mod n, dove (e,n) è la chiave pubblica e M è il messaggio in chiaro.

Crittografia end-to-end

Garantisce che solo il mittente e il destinatario possano leggere i dati trasmessi, senza che terze parti (compresi i server intermedi) possano accedervi.

Virtual Private Database (VPD)

Un VPD mostra a ciascun utente solo i dati per cui ha l’autorizzazione di accesso, migliorando la riservatezza all’interno di un database condiviso.

Scrum e Manifesto Agile

Iterativo e Incrementale

Il lavoro viene organizzato in cicli ripetuti (Sprint) di durata fissa (da 1 a 4 settimane). Ad ogni Sprint il prodotto cresce sia in qualità che in funzionalità. La consegna frequente permette di raccogliere feedback continuo dal cliente e adattare rapidamente lo sviluppo.

Ruoli in Scrum

- Scrum Master: facilita gli eventi, garantisce che Scrum venga applicato correttamente e rimuove ostacoli. Non comanda il team, ma lo supporta come un allenatore.

- Product Owner: definisce e aggiorna il Product Backlog, stabilisce priorità e massimizza il valore del lavoro svolto dal team.

- Developers: realizzano le funzionalità e assicurano che l'incremento sia pronto secondo la Definition of Done.

Eventi principali

- Sprint: periodo di lavoro con obiettivi chiari e durata fissa scelta all'inizio del progetto.

- Daily Scrum: breve riunione (massimo 15 minuti), sempre alla stessa ora e luogo, per allineare il lavoro quotidiano.

- Sprint Review: momento di confronto con gli Stakeholder per mostrare le funzionalità sviluppate e raccogliere feedback.

- Product Backlog Refinement: attività collaborativa e continua per chiarire e dettagliare le voci del backlog.

Product Backlog

Contiene tutto ciò che serve per sviluppare il prodotto. Gli elementi in alto sono dettagliati e pronti per essere lavorati subito; quelli in basso possono rimanere più generici e saranno chiariti man mano.

Definition of Done
È l’accordo condiviso su cosa significa “fatto”. Se l’organizzazione non ne ha una, ogni Team Scrum ne definisce una propria. Può variare da team a team, ma va rispettata per garantire qualità e coerenza.

Burndown Chart
Grafico opzionale usato dal Team Scrum per monitorare l’avanzamento del lavoro all’interno della Sprint.

Cynefin Framework
Lo sviluppo software rientra spesso in un dominio complesso, dove è necessario sperimentare, imparare e adattarsi continuamente, piuttosto che pianificare tutto in anticipo.

Differenze con il Project Management tradizionale
Nel modello tradizionale il Project Manager è responsabile di costi, tempi, ambito e qualità, con un ruolo di comando e controllo.
In Scrum queste responsabilità sono distribuite:

- Product Owner: valore e priorità.
- Scrum Master: processo corretto.
- Developers: qualità tecnica e implementazione.

Comandi base Ubuntu

mkdir [nome-cartella] - crea una nuova directory
rm [file] - elimina un file
ls - elenca file e cartelle nella directory corrente
cp [origine] [destinazione] - copia un file o una directory
mv [origine] [destinazione] - sposta o rinomina un file o directory
pwd - mostra il percorso della directory corrente
cd [percorso] - cambia directory

chmod [permessi] [file] - modifica i permessi di un file o directory
chmod +x script.sh - rende eseguibile uno script Bash
chown [utente] [file] - cambia il proprietario di un file

Permessi:
r = lettura
w = scrittura
x = esecuzione

apt-get install [pacchetto] - installa un pacchetto
apt install [pacchetto] - installa un pacchetto
dpkg -i [pacchetto].deb - installa un pacchetto .deb
apt-get update && apt-get upgrade - aggiorna pacchetti e sistema

uptime - mostra da quanto tempo il sistema è attivo
top - mostra processi e uso CPU/memoria in tempo reale
df - mostra l'utilizzo del disco
cat /var/log/syslog - visualizza i log di sistema
ps -e - mostra tutti i processi in esecuzione
ip addr show - mostra indirizzi IP
ifconfig - mostra indirizzi IP
hostname -I - mostra indirizzi IP

service [servizio] status - verifica lo stato di un servizio
service [servizio] stop - ferma un servizio
systemctl list-units --type=service - elenca i servizi attivi
service --status-all - elenca lo stato di tutti i servizi
crontab -l - mostra i cron job dell'utente corrente

passwd - cambia la password dell'utente loggato

Metabase

Fondamenti di Business Intelligence (BI)

La Business Intelligence è il processo di analisi e visualizzazione dei dati per supportare decisioni aziendali informate. Fondamentale è la qualità dei dati e la loro integrazione, che permette di consolidare informazioni provenienti da diverse fonti per ottenere insight affidabili. Componenti chiave includono l'ETL (Extract, Transform, Load) per la gestione dati, la visualizzazione e l'analisi OLAP (Online Analytical Processing).

Strumenti e Funzionalità di Metabase

Metabase consente di creare e salvare query chiamate "Questions", e di raggrupparle in dashboard per monitorare informazioni importanti. La condivisione pubblica permette di far accedere i dati anche a utenti esterni. Funzioni come le "Dashboard Subscription" inviano report automatici via email. L'organizzazione tramite "Collections" aiuta a gestire domande, dashboard e modelli. La modalità SQL permette query personalizzate, mentre la sicurezza si basa su controllo accessi e crittografia SSL.

Applicazioni di BI nel Business

BI è utile per migliorare le vendite analizzando i comportamenti d'acquisto e gestendo meglio le scorte. Supporta anche team di marketing tramite report sulle campagne. Gli analisti di dati sono fondamentali per interpretare i dati e creare report efficaci, facilitando decisioni aziendali basate su fatti concreti.

Tecnologie Avanzate e Trend in BI

L’intelligenza artificiale automatizza l’analisi dati e permette previsioni più accurate. Il Natural Language Processing (NLP) consente di usare il linguaggio naturale per interrogare gli strumenti BI. La blockchain assicura integrità e sicurezza dei dati in modo decentralizzato. Il cloud computing garantisce scalabilità, flessibilità e accessibilità. L’augmented analytics automatizza la generazione di insight dai dati.

Sfide e Considerazioni nella Implementazione di BI

L’adozione di soluzioni BI può incontrare resistenze al cambiamento e difficoltà di integrazione tra sistemi. Il data warehouse è fondamentale come archivio centralizzato dove organizzare dati strutturati e trasformati, migliorando l’accessibilità e l’efficienza delle analisi.

Visualizzazioni e Supporto Metabase

Metabase supporta vari tipi di visualizzazioni, come grafici a barre e scatter plot, ma non proiezioni olografiche 3D. Offre un’interfaccia intuitiva che democratizza l’accesso ai dati, permettendo anche agli utenti non tecnici di esplorare e comprendere facilmente le informazioni aziendali.

NoSQL

NoSQL - Panoramica generale

- NoSQL significa “Not Only SQL” e include diversi modelli di database alternativi a quelli relazionali.
- I principali tipi sono:
    - **Documenti** (es. MongoDB)
    - **Chiave-valore** (es. Redis)
    - **Colonne** (es. Cassandra)
    - **Grafi** (es. Neo4j)
- Sono progettati per scalabilità orizzontale, alta disponibilità e flessibilità nella modellazione dei dati.

MongoDB - Introduzione

- È un database NoSQL orientato ai documenti.
- I dati sono memorizzati in formato BSON, una rappresentazione binaria di JSON.
- È schema-less: ogni documento in una collezione può avere struttura diversa.
- Usato per applicazioni moderne, microservizi e Big Data.

Struttura dei Dati in MongoDB

- I dati sono organizzati in database, che contengono collezioni, le quali contengono documenti (oggetti JSON).
- Un documento ha campi con valori (stringhe, numeri, array, oggetti annidati, ecc.).
- I documenti hanno un campo _id come identificatore univoco.

Operazioni CRUD in MongoDB

- Create: db.collection.insertOne({ ... })
- Read: db.collection.find({ campo: valore })
- Update: db.collection.updateOne({ filtro }, { $set: { campo: nuovoValore } })
- Delete: db.collection.deleteOne({ campo: valore })

Query e Operatori

- Query semplici: db.collection.find({ campo: valore })
- Operatori di confronto: $gt, $lt, $eq, $in, $ne, $gte, $lte
- Query su array: db.collection.find({ arrayCampo: "valore" })
- Accesso a campi annidati: db.collection.find({ "campo1.campo2": valore })

Aggregazioni e Join

- MongoDB usa aggregate() per pipeline avanzate.

L'operatore $lookup permette di fare un join tra collezioni:

```
 js
CopiaModifica
db.movies.aggregate([
  { $lookup: {
      from: "comments",
      localField: "_id",
      foreignField: "movie_id",
      as: "comments"
  }},
  { $sort: { title: 1 } },
  { $limit: 5 }
])
```

- È simile a un LEFT OUTER JOIN in SQL.

Comandi utili

- Visualizzare collezioni: show collections
- Selezionare il database: use nomeDatabase
- Contare documenti: db.collection.countDocuments()
- Ordinare e limitare:
  db.collection.find().sort({ campo: 1 }).limit(5)

Import/export e backup

- mongoimport: importa dati in MongoDB da file JSON o CSV
- mongoexport: esporta dati
- mongodump: crea backup binario
- mongorestore: ripristina backup creati con mongodump

Uso di array e oggetti

- I campi possono contenere array e oggetti annidati, rendendo la struttura dati flessibile.
- Esempio: cast: ["Tom Hanks", "Meg Ryan"]
  Ricerca: db.movies.find({ cast: "Tom Hanks" })

Vantaggi di NoSQL

- Estrema scalabilità (soprattutto in ambienti distribuiti)
- Flessibilità nello schema dei dati
- Performance elevate per operazioni su grandi volumi di dati non relazionali
- Supporto a dati semi-strutturati e non strutturati

Quando usare NoSQL

- Grandi volumi di dati
- Applicazioni real-time o con latenza minima
- Necessità di gestire dati flessibili e non normalizzati
- Architetture a microservizi o sistemi distribuiti

Economia

Interesse semplice

Viene usato quando gli interessi vengono calcolati solo sul capitale iniziale, senza reinvestire gli interessi maturati.

Formula dell'interesse:
$I = (C * r * t) / T$

Dove:

- I = interesse maturato
- C = capitale prestato o investito
- r = tasso d'interesse (espresso in percentuale)
- t = durata del prestito/investimento
- T = divisore che dipende dall'unità di tempo:
    - $T = 100$ se t è espresso in anni
    - $T = 1200$ se in mesi
    - $T = 36500$ se in giorni (calendario civile)
    - $T = 36000$ se in giorni (calendario commerciale)

Montante:

$M = C + I$

È la somma totale che il debitore restituirà al creditore: capitale + interesse.

Interesse composto

Viene usato quando gli interessi maturati vengono reinvestiti nel capitale, producendo a loro volta altri interessi: è il meccanismo alla base della crescita esponenziale del risparmio.

Formula dell’interesse composto:

$M = C * (1 + r)^T$

Dove:

- M = montante (capitale finale)
- C = capitale iniziale
- r = tasso di interesse (in forma decimale, es. 6% = 0,06)
- T = numero di periodi (es. anni)

Questa formula non considera versamenti aggiuntivi periodici, ma solo l’accumulo su un investimento fisso.

Che cos’è la blockchain

È un registro digitale condiviso e immutabile, dove le informazioni vengono salvate in ordine cronologico e in blocchi concatenati tra loro. È la base tecnologica di molte criptovalute.

Catena di blocchi

Ogni blocco contiene dati (es. transazioni), l’hash del blocco precedente e un timestamp. Questo collegamento impedisce la modifica retroattiva dei dati.

Registro condiviso e immutabile

Tutti i nodi della rete possiedono una copia identica del registro. Una volta aggiunto un blocco, questo non può più essere modificato.

Reti peer-to-peer

I partecipanti (nodi) comunicano direttamente tra loro, senza un server centrale. La rete è distribuita e più resistente agli attacchi.

Algoritmo gossip

Tecnica usata per diffondere rapidamente le informazioni nella rete: ogni nodo trasmette dati a uno o più nodi casuali, fino a coinvolgere l’intera rete.

Origini della blockchain

La blockchain moderna nasce con Bitcoin nel 2008, ma concetti simili erano già stati introdotti nel 1991 da Haber e Stornetta con i primi sistemi di timestamp digitale.

Il movimento cypherpunk

Gruppo attivo dagli anni ‘80 che promuoveva la crittografia come mezzo per la privacy e la libertà. Diede impulso a idee come Bitcoin.

Bitcoin e nascita della prima criptovaluta

Bitcoin è la prima criptovaluta basata su blockchain, anonima, senza autorità centrale e con un massimo di 21 milioni di unità. Le transazioni sono pubbliche ma semi-anonime.

Funzionamento di Bitcoin

Ogni utente ha indirizzi (pubblici) che usa per inviare e ricevere bitcoin. I bitcoin sono generati dai miner che validano i blocchi.

Funzioni hash e proprietà

Una funzione hash prende in input qualsiasi dato e restituisce una stringa fissa. Serve a garantire integrità, anonimato e collegamento tra blocchi. Le proprietà principali: non invertibilità e resistenza alle collisioni.

SHA256 e SHA2562

SHA256 è la funzione hash usata da Bitcoin. SHA2562 significa applicare SHA256 due volte di seguito. È usata per aumentare la sicurezza.

Merkle tree

Struttura ad albero usata per organizzare le transazioni in un blocco. Permette di verificare in modo rapido e sicuro se una transazione è inclusa.

Proof of Work (PoW)

Meccanismo di consenso che richiede la risoluzione di un problema computazionale per aggiungere un blocco. Rende difficile manipolare la blockchain.

Esempi ed esercizi di PoW

Esempi pratici mostrano come trovare un valore (nonce) che, combinato a un messaggio, genera un hash inferiore a un certo target. Serve per simulare il lavoro del mining.

Mining e ricompense

I miner sono ricompensati in bitcoin per ogni blocco valido aggiunto. Il mining richiede potenza di calcolo e consuma energia. Esistono mining farm professionali.

Crittografia a chiave pubblica

Ogni utente ha una chiave privata (segreta) e una pubblica. La chiave privata serve a firmare, quella pubblica a verificare. Garantisce autenticità e sicurezza.

Firma digitale e verifica

Serve per dimostrare che una transazione è stata autorizzata dal proprietario. Le firme non possono essere falsificate se le chiavi sono segrete.

Curve ellittiche ed ECDSA

Bitcoin usa una curva ellittica chiamata Secp256k1 per generare chiavi. L'ECDSA è l'algoritmo che firma digitalmente le transazioni.

Generazione di indirizzi Bitcoin

L'indirizzo Bitcoin si ottiene dalla chiave pubblica attraverso funzioni hash. Ogni indirizzo rappresenta un "conto" sulla blockchain.

Problema del double spending

Rischio che una stessa moneta digitale venga spesa due volte. La blockchain lo evita validando ogni transazione e impedendo duplicazioni.

Validazione dei blocchi

Un blocco è valido se rispetta il Proof of Work. La catena con più blocchi (quella più lunga) è considerata la corretta. Se la maggioranza dei nodi è onesta, il sistema è sicuro.

Transazioni nella blockchain

Le transazioni sono pubbliche e verificabili. Una volta confermate e inserite in un blocco, non possono più essere modificate.

SQL

Cos'è sql

E’ un linguaggio per gestire dati in database relazionali. permette di creare tabelle, inserire, leggere, modificare e cancellare dati. è usato con sistemi come mysql, postgresql, oracle e sqlite

Tabelle e struttura del database

I dati sono organizzati in tabelle
ogni tabella ha righe (record) e colonne (campi)
le tabelle possono essere collegate tra loro

Chiave primaria (primary key)

E’ un campo che identifica in modo univoco ogni riga di una tabella
non può contenere valori duplicati né null
esempio: id studente

Chiave esterna (foreign key)

E’ un campo che crea un collegamento con la chiave primaria di un’altra tabella
serve a mantenere l’integrità dei dati tra tabelle
esempio: studente_id nella tabella esami che punta alla tabella studenti

Tipi principali di comandi

Select: per leggere i dati
Insert: per inserire nuovi dati
Update: per modificare dati esistenti
Delete: per cancellare dati
Create: per creare tabelle o database
Alter: per modificare tabelle
Drop: per eliminare tabelle o colonne

Join tra tabelle

Le join servono per unire i dati di più tabelle collegate tra loro
Inner join: mostra solo i dati con corrispondenza in entrambe le tabelle
Left join: mostra tutti i dati della prima tabella, anche senza corrispondenza nella seconda
Right join: mostra tutti i dati della seconda tabella
Full outer join: mostra tutti i dati di entrambe le tabelle, anche senza corrispondenze

Filtri e ordinamenti

Where: per filtrare i risultati
Order by: per ordinare i dati
Limit: per limitare il numero di righe mostrate
Distinct: per eliminare duplicati

Funzioni di aggregazione

Count: conta le righe
Sum: somma dei valori
Avg: media
Min e Max: minimo e massimo
Group by: raggruppa per valore di un campo
Having: filtra dopo il group by

Transazioni

Commit: conferma definitiva delle modifiche
Rollback: annulla modifiche non ancora confermate
sono usate per garantire coerenza nei dati

Permessi e sicurezza

Grant e Revoke: servono a dare o togliere permessi su tabelle e database agli utenti

AWS

Cos'è AWS

AWS è la piattaforma cloud di Amazon che permette di utilizzare risorse informatiche in modo flessibile e scalabile attraverso internet. In pratica, invece di acquistare e mantenere fisicamente server, database o altri dispositivi, si affittano risorse on demand, pagando solo per quello che si usa. AWS offre una vasta gamma di servizi per soddisfare esigenze diverse, dalla gestione di siti web, applicazioni, archiviazione di dati, fino all'intelligenza artificiale.

Perché si usa

AWS è molto apprezzato perché consente alle aziende di scalare facilmente la propria infrastruttura, cioè aumentare o diminuire risorse in base al bisogno senza grandi investimenti iniziali. Inoltre, garantisce un'alta affidabilità grazie alla sua struttura distribuita in tutto il mondo, permette di risparmiare sui costi operativi, e offre strumenti per automatizzare e semplificare la gestione dell'infrastruttura IT.

Modello di pagamento

Con il modello pay-as-you-go, paghi solo per le risorse che effettivamente utilizzi, senza costi fissi anticipati. Questo rende AWS accessibile anche a progetti piccoli o startup. Il Free Tier, invece, offre un insieme di risorse gratuite per i primi 12 mesi, così da poter testare e imparare senza spese.

Regioni e Availability Zones

AWS divide la sua infrastruttura in Regioni, cioè aree geografiche distinte (ad esempio Europa, Stati Uniti, Asia). Ogni Regione contiene diverse Availability Zones, che sono data center isolati tra loro. Questo permette di distribuire le applicazioni su più zone per garantire che, in caso di guasto in una zona, il servizio continui a funzionare in un'altra.

Servizi principali

- EC2: consente di creare macchine virtuali, cioè server su cui far girare applicazioni o siti.

- S3: servizio di archiviazione di oggetti, utile per salvare file, immagini, backup in modo scalabile.

- RDS: servizio gestito per database relazionali come MySQL o PostgreSQL, che si occupa di backup, aggiornamenti e sicurezza.

- Lambda: permette di eseguire codice senza dover gestire server, ideale per funzioni che si attivano in risposta a eventi.

- VPC: crea reti virtuali private all'interno del cloud, per gestire la comunicazione tra risorse in modo sicuro.

- IAM: gestisce utenti, gruppi e permessi, per controllare chi può fare cosa.

- CloudWatch: monitora lo stato delle risorse e raccogliere log per diagnosticare problemi.

- CloudFormation: permette di definire l'intera infrastruttura tramite file di testo, così da replicare facilmente.

Sicurezza e permessi

La sicurezza in AWS è molto importante e viene gestita tramite IAM, che permette di creare ruoli e permessi specifici per ogni utente o servizio, limitando così l'accesso solo alle risorse necessarie. I Security Group sono firewall virtuali che proteggono le istanze EC2 controllando il traffico in entrata e in uscita. Inoltre, per i dati salvati su S3, si possono applicare Bucket Policy e ACL per definire chi può leggere o scrivere i file.

Come si lavora su AWS

Si può usare la Console Web, un'interfaccia grafica semplice e intuitiva, oppure la CLI (Command Line Interface) da terminale per operazioni più rapide o automatizzate. Gli SDK sono librerie in vari linguaggi di programmazione per interagire con AWS direttamente dal codice. CloudFormation e Terraform sono strumenti per definire e gestire l'infrastruttura come codice, migliorando la ripetibilità e il controllo delle modifiche.

Esempi di attività comuni

- Creare un server EC2 e collegarsi via SSH per gestirlo.
- Caricare e scaricare file da S3 tramite console o CLI.
- Configurare un database relazionale con RDS, evitando la gestione manuale.
- Scrivere funzioni Lambda per automatizzare processi senza server dedicati.
- Impostare bucket S3 pubblici o privati a seconda delle necessità.
- Configurare alert in CloudWatch per ricevere notifiche su eventi critici.
- Monitorare l'andamento dei costi con Cost Explorer per evitare sorprese in fattura.

Cos'è Docker

Docker è una tecnologia che permette di creare e gestire container, cioè ambienti isolati in cui far girare applicazioni con tutte le loro dipendenze, in modo identico su qualsiasi computer o server. Questo risolve molti problemi legati al fatto che un'applicazione funzioni su una macchina ma non su un'altra.

Perché usarlo

Docker standardizza l'ambiente di sviluppo e produzione, garantendo che l'applicazione abbia sempre le stesse condizioni di esecuzione. È molto più leggero e veloce rispetto alle macchine virtuali tradizionali, consente di isolare le applicazioni e di distribuire rapidamente.

Concetti base

- Immagine: è il modello statico del container, contiene il sistema operativo minimale e l'applicazione con le sue dipendenze.
- Container: è l'istanza in esecuzione di un'immagine, cioè il processo che gira isolato.
- Dockerfile: file di testo con le istruzioni per creare un'immagine Docker.
- Docker Hub: repository pubblico dove si trovano migliaia di immagini pronte all'uso.
- Volume: spazio di archiviazione persistente per mantenere i dati anche se il container viene fermato o eliminato.

Comandi principali

- docker pull: scarica un'immagine da Docker Hub.
- docker run: avvia un container da un'immagine.
- docker ps: mostra i container attivi.
- docker stop: ferma un container.
- docker build: crea un'immagine partendo da un Dockerfile.
- docker exec: entra dentro un container in esecuzione.

- docker logs: mostra i log di un container.
- docker-compose up: avvia più container definiti in un file YAML.
- docker rm / rmi: rimuove container o immagini non più necessari.

Lavorare con Dockerfile

Il Dockerfile è fondamentale per creare immagini personalizzate. Le istruzioni principali sono:

- FROM: definisce l'immagine di base da cui partire.
- RUN: esegue comandi nel processo di build (es. installare software).
- COPY: copia file locali dentro l'immagine.
- EXPOSE: indica quali porte sono aperte nel container.
- CMD o ENTRYPOINT: specificano il comando che il container eseguirà all'avvio.

Esempio di flusso di lavoro

Si scrive il Dockerfile, poi si costruisce l'immagine con docker build. Successivamente si avvia il container con docker run per testare e sviluppare in locale. Quando tutto funziona, l'immagine può essere pubblicata su Docker Hub o utilizzata per il deploy in cloud.

Usare Docker in team

Con Docker, ogni sviluppatore può lavorare su un ambiente identico, evitando il problema del "funziona solo sul mio PC". È ideale per applicazioni basate su microservizi e per le pratiche DevOps, che richiedono automazione e continuità nello sviluppo.

Integrazione con AWS

AWS offre servizi specifici per gestire container:

- AWS ECS (Elastic Container Service) per orchestrare ed eseguire container in modo scalabile.

- AWS ECR (Elastic Container Registry) per archiviare immagini Docker in modo sicuro e privato.

- AWS Fargate per eseguire container senza doversi preoccupare della gestione dei server sottostanti, semplificando ulteriormente il deploy.

Reti di calcolatori

Che cos'è il protocollo HTTP?

- HTTP (HyperText Transfer Protocol) è un protocollo per il trasferimento di ipertesti su internet.
- Opera al livello applicazione del modello OSI.
- Viene usato principalmente per scambiare pagine web tra client (browser) e server.
- È un protocollo stateless, cioè il server non mantiene informazioni sulle connessioni passate.

Sicurezza: SSL/TLS e HTTPS

- SSL/TLS sono protocolli che forniscono connessioni sicure incapsulando i dati di protocolli di livello superiore come HTTP.
- Grazie a SSL/TLS, la navigazione HTTP diventa HTTPS, garantendo cifratura e integrità dei dati scambiati.
- SSL/TLS non sostituisce i protocolli applicativi, ma li protegge durante la trasmissione.

FTP: File Transfer Protocol

- FTP è usato per il trasferimento di file tra host.
- È considerato un protocollo stateful perché utilizza due canali distinti:
    - Porta TCP 21 per il controllo (comandi)
    - Porta TCP 20 per il trasferimento dati
- Non si occupa di crittografia di default (per quella esiste FTPS o SFTP).

Modelli di Rete: OSI e TCP/IP

- Il modello ISO/OSI è composto da 7 livelli:
    1. Fisico
    2. Collegamento dati
    3. Rete
    4. Trasporto
    5. Sessione
    6. Presentazione
    7. Applicazione
- È stato sviluppato dall'ISO (International Organization for Standardization).

- Il modello TCP/IP originale ha 4 livelli:
    1. Accesso alla rete
    2. Internet
    3. Trasporto
    4. Applicazione
- È stato sviluppato principalmente dalla DARPA (Defense Advanced Research Projects Agency).

Livelli e Funzioni nei modelli di rete

- Livello Applicazione (OSI e TCP/IP): fornisce servizi diretti agli utenti tramite protocolli come HTTP, FTP, SMTP, SSH.
- Livello Trasporto: responsabile della segmentazione dei dati e della comunicazione affidabile (TCP) o non affidabile (UDP).
- Livello Rete: si occupa del routing dei pacchetti e indirizzamento, dove opera il protocollo IP.
- Livello Collegamento dati: garantisce la trasmissione affidabile tra due nodi adiacenti.

Protocolli principali e loro caratteristiche

- TCP (Transmission Control Protocol):
    - Orientato alla connessione
    - Garantisce la consegna dei dati
    - Usa handshake per stabilire la connessione

- UDP (User Datagram Protocol):
    - Non orientato alla connessione
    - Non garantisce la consegna dei dati
    - Più veloce, usato per streaming o giochi
- IP (Internet Protocol): opera al livello Internet, si occupa dell'instradamento dei pacchetti.
- SSH (Secure Shell): protocollo per la gestione remota sicura dei dispositivi, usa la porta TCP 22.
- SMTP: protocollo per l'invio di posta elettronica, lavora al livello applicazione.

Porte TCP più comuni

- Porta 80: HTTP
- Porta 443: HTTPS (HTTP protetto da SSL/TLS)
- Porta 21: FTP controllo
- Porta 20: FTP dati
- Porta 22: SSH
- Porta 25: SMTP (posta elettronica)

Concetti importanti

- Stateless vs Stateful:
    - Stateless: il server non mantiene lo stato della connessione (es. HTTP)
    - Stateful: il server mantiene lo stato, più adatto per protocolli come FTP
- I protocolli lato applicazione sono quelli che forniscono servizi specifici agli utenti finali (web, posta, trasferimento file).

Java

Incapsulamento (Encapsulation)

L’incapsulamento è il meccanismo che consente di proteggere lo stato interno di un oggetto dall’accesso diretto esterno. I dati (variabili d'istanza) vengono dichiarati privati, e l'accesso viene consentito solo tramite metodi pubblici (getter e setter).

- Vantaggi principali:
  - Protezione dei dati (data hiding)
  - Separazione tra interfaccia pubblica e implementazione interna
  - Maggiore controllo sulla modifica dei valori

- Esempio base:

```
public class Persona {
    private String nome;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

- Nota importante: anche i metodi possono essere incapsulati (es. metodi private utilizzati solo all’interno della classe).

Ereditarietà (Inheritance)

L’ereditarietà permette la creazione di nuove classi che riutilizzano il comportamento di classi esistenti, favorendo il riuso del codice e la creazione di gerarchie logiche tra oggetti.

- **Parole chiave importanti**:
  - extends → usata per ereditare da una classe
  - super → usata per accedere ai membri della superclasse (anche nel costruttore)

- **Tipi di ereditarietà** in Java:
  - Ereditarietà **singola** (una sola superclasse per ogni sottoclasse)
  - Ereditarietà **multilivello** (una catena di classi ereditate)
  - Java **non supporta** l’ereditarietà multipla tra classi (per evitare ambiguità), ma è possibile implementarla tramite **interfacce**.

- **Esempio**:

```
class Animale {
    void faiVerso() {
        System.out.println("Verso generico");
    }
}

class Cane extends Animale {
    void faiVerso() {
        System.out.println("Bau!");
    }
}
```

Polimorfismo (Polymorphism)

Il polimorfismo permette di utilizzare un oggetto in modi diversi, a seconda del contesto. Java supporta due tipi di polimorfismo: statico e dinamico.

Polimorfismo statico (overloading)

Più metodi con lo stesso nome, ma firme diverse (numero, tipo o ordine dei parametri).

- Esempio:

```
void stampa(String s) { ... }
void stampa(String s, int n) { ... }
```

- Nota importante: l'overloading avviene a tempo di compilazione (compile time polymorphism).

Polimorfismo dinamico (overriding)

Una sottoclasse può ridefinire un metodo della superclasse, mantenendone la stessa firma.

- Esempio:

```
class Animale {
    void verso() {
        System.out.println("Verso animale");
    }
}

class Gatto extends Animale {
    @Override
    void verso() {
        System.out.println("Miao");
    }}
```

🔹 Avviene a runtime (runtime polymorphism)

🔹 Utilissimo per il concetto di dispatch dinamico, in cui il metodo chiamato dipende dal tipo reale dell'oggetto, non dal tipo della variabile di riferimento.

Astrazione (Abstraction)

L'astrazione serve a nascondere i dettagli di implementazione e a mostrare solo gli elementi rilevanti. Java offre due meccanismi per realizzarla:

🔸 Classi astratte

- Si definiscono con la parola chiave abstract
- Possono contenere sia metodi astratti (senza corpo) che metodi concreti
- Non possono essere istanziate

🔹 Esempio:

```
abstract class Forma {
   abstract double calcolaArea();

   void descrizione() {
      System.out.println("Sono una forma geometrica");
   }}
```

🔸 Interfacce

- Dichiarano solo metodi astratti (dal Java 8 anche metodi default e static)
- Una classe può implementare più interfacce, simulando l'ereditarietà multipla
- Usano la parola chiave interface e implements

🔹 Esempio:

```
interface Volante {
    void vola();
}

class Aereo implements Volante {
    public void vola() {
        System.out.println("Sto volando!");
    }
}
```

Caratteristica	**Classe Astratta**	**Interfaccia**
Ereditarietà multipla	❌	✅
Costruttori	✅ (può averli)	❌
Tipo di metodi	Astratti + concreti	Solo astratti (tranne default e static)
Eredita con	extends	implements

API

Concetti base delle RESTful API

- REST (REpresentational State Transfer) è uno stile architetturale per la progettazione di servizi web.
- Le RESTful API usano il protocollo HTTP per comunicare tra client e server.
- Le risorse (es. utenti, prodotti) sono identificate tramite URI (/users, /products/123).
- La comunicazione è stateless: ogni richiesta deve contenere tutte le informazioni necessarie.

Verbi HTTP e operazioni API

Verbo HTTP	Operazione	Descrizione
GET	Lettura	Recupera una risorsa o una lista di risorse
POST	Creazione	Crea una nuova risorsa
PUT	Sostituzione	Aggiorna completamente una risorsa esistente
PATCH	Modifica	Aggiorna parzialmente una risorsa
DELETE	Cancellazione	Elimina una risorsa

Codici di stato HTTP

- 200 OK: richiesta andata a buon fine
- 201 Created: risorsa creata con successo (POST)
- 204 No Content: nessun contenuto da restituire (DELETE, PUT)
- 400 Bad Request: errore lato client (es. parametri mancanti)
- 401 Unauthorized: richiesta non autenticata
- 403 Forbidden: utente autenticato ma non autorizzato
- 404 Not Found: risorsa non trovata
- 500 Internal Server Error: errore generico lato server

Best practices per le API REST

- Usa nomi di risorse al plurale (/users, non /user)
- Le URI devono essere pulite e leggibili
- Usa query string per filtri e paginazione (/products?category=books&page=2)
- Gestione degli errori tramite messaggi chiari in JSON
- Supporto a versionamento (/api/v1/users)

OpenAPI (ex Swagger)

- OpenAPI è uno standard per descrivere le API RESTful in formato leggibile da umani e macchine.
- Definito in YAML o JSON.
- Strumenti principali:
    - Swagger UI: interfaccia grafica per testare le API
    - Swagger Editor: per scrivere e validare la documentazione
    - Swagger Codegen: genera codice client/server

Esempio YAML base:

```yaml
openapi: 3.0.0
info:
  title: API di esempio
  version: 1.0.0
paths:
  /users:
    get:
      summary: Ottiene tutti gli utenti
      responses:
        '200':
          description: OK
```

Autenticazione vs Autorizzazione

- Autenticazione: verifica l’identità dell’utente (chi sei).
- Autorizzazione: verifica i permessi dell’utente (cosa puoi fare).

Esempi:

- Autenticazione: login con email/password → ricevi un token.
- Autorizzazione: controllo dei permessi associati al token.

JWT (JSON Web Token)

- Standard per rappresentare token di accesso in formato compatto e sicuro.
- Composto da tre parti:
    1. Header: tipo di token e algoritmo
    2. Payload: dati utente e claims (es. user_id, role)
    3. Signature: firma per garantire integrità

Formato base:

eyJhbGciOi... (header).eyJzdWIiOi... (payload).SflKxwR... (signature)

Vantaggi:

- Stateless: il server non deve mantenere sessioni
- Facile da trasportare e verificare
- Può scadere (exp) e contenere ruoli o permessi

Protezione delle API

- HTTPS obbligatorio: per proteggere dati in transito
- Rate limiting: previene abusi
- CORS: controlla accessi da origini diverse
- Validazione dei dati: previene injection e input malevoli
- Scopes e ruoli: controllo granulare delle autorizzazioni

Sicurezza avanzata e best practice

- Scadenza breve dei token (JWT) + refresh token
- Blacklist dei token revocati (se necessario)
- Audit log per operazioni sensibili
- Least privilege: accesso minimo necessario per l’utente

Testing delle API

- Postman, Insomnia: strumenti per testare manualmente le API
- Test automatici:
    - Unit test per singole funzioni
    - Integration test per endpoints completi
- Mock server con OpenAPI per simulare risposte

Versionamento e deployment

- Versiona le API (/v1, /v2)
- Mantieni retrocompatibilità il più possibile
- Documenta ogni versione con OpenAPI
- Deployment con CI/CD: test → build → deploy

Linux

Fondamenti di Linux e struttura del sistema

- Distribuzioni comuni: Ubuntu Server, Debian, CentOS, AlmaLinux, Rocky, Arch.
- Il filesystem è gerarchico, con root / alla base.
    - /bin, /sbin, /usr/bin: comandi essenziali
    - /etc: configurazioni
    - /var: log e dati variabili
    - /home: directory utenti
    - /root: home dell’amministratore
    - /tmp: file temporanei
- Shell comune: bash (ma anche zsh, sh, ecc.)

Comandi di base

```
ls -l            # lista dettagliata dei file
cd /percorso     # cambia directory
cp file1 file2   # copia file
mv file1 file2   # sposta/rinomina
rm file          # rimuove file
mkdir dir        # crea directory
touch file       # crea file vuoto
cat file         # visualizza contenuto
nano file        # editor di testo semplice
```

- Autocompletamento: Tab
- Cronologia comandi: history, !n, frecce ↑ ↓

Gestione dei pacchetti

APT (Debian, Ubuntu):

```
sudo apt update          # aggiorna lista pacchetti
sudo apt upgrade         # aggiorna sistema
sudo apt install nginx   # installa pacchetto
sudo apt remove nginx    # rimuove pacchetto
```

DNF/YUM (CentOS, RHEL):

```
sudo dnf install httpd
```

Gestione utenti e permessi
Comandi utenti:

```
sudo adduser mario          # crea utente
sudo passwd mario           # imposta password
sudo usermod -aG sudo mario  # aggiungi a gruppo sudo
```

Permessi file (rwx per owner, group, others):

```
ls -l file
chmod 755 script.sh         # imposta permessi
chown utente:gruppo file    # cambia proprietario
```

Numero	Permessi
7	rwx
6	rw-
5	r-x
4	r–

Gestione processi e risorse

```
ps aux                  # visualizza processi
top / htop              # monitor in tempo reale
kill PID                # termina processo
df -h                   # spazio disco
du -sh *                # spazio occupato per directory
free -m                 # uso RAM
uptime                  # tempo acceso
```

Directory principali del filesystem Linux (/)

Cartella	Descrizione
/	Root del filesystem: punto di partenza gerarchico. Tutto parte da qui.
/bin	Binary: comandi essenziali eseguibili accessibili da tutti gli utenti (es. ls, cp, mv).
/sbin	System Binary: comandi per l'amministrazione del sistema (es. shutdown, reboot). Usati da root.
/usr	User System Resources: contiene la maggior parte dei programmi e librerie per gli utenti.
/usr/bin	Programmi e utility per utenti normali.
/usr/sbin	Programmi e tool di amministrazione usati da root.
/usr/lib	Librerie condivise per i programmi in /usr/bin e /usr/sbin.
/etc	Et Cetera: file di configurazione del sistema e dei servizi (es. passwd, hostname, nginx.conf).
/var	Variable: file variabili come log, spool di stampa, cache, email.
/var/log	File di log di sistema e applicazioni (es. syslog, auth.log).
/home	Directory personale degli utenti normali (es. /home/mario).
/root	Home directory dell’utente root.
/tmp	File temporanei. Vengono spesso svuotati al riavvio.
/boot	File di avvio del sistema, inclusi kernel e initrd (vmlinuz, grub/).

/lib	Librerie essenziali per i binari in /bin e /sbin.
/media	Mount automatico per dispositivi esterni (es. chiavette USB).
/mnt	Punto di montaggio temporaneo, usato per mount manuali (es. partizioni, dischi esterni).
/opt	Software opzionale installato manualmente o da pacchetti esterni.
/srv	Dati dei servizi forniti dal sistema (es. /srv/www per web server).
/dev	File di dispositivo: rappresentano dispositivi fisici (es. /dev/sda, /dev/null).
/proc	File virtuali con info su processi e kernel (es. /proc/cpuinfo, /proc/meminfo).
/sys	Interfaccia al kernel e ai device, simile a /proc.
/run	Info runtime (PID, socket, mount temporanei), creati all’avvio.

Note utili

- /usr ≠ /home: /usr contiene software per tutti gli utenti, /home i dati personali.
- /lib, /usr/lib, /lib64: librerie di sistema, a volte separate per architettura (32 vs 64 bit).
- /proc e /sys: file virtuali generati dal kernel, non "veri" file salvati su disco.
- /var cresce nel tempo: importante monitorarla per non saturare il disco.

Blockchain

Il protocollo Bitcoin

Bitcoin è la prima criptovaluta decentralizzata, basata su una rete peer-to-peer e un registro pubblico chiamato blockchain. Ogni blocco contiene un insieme di transazioni, collegato al blocco precedente tramite un hash crittografico, creando una catena immutabile.

Transazioni Bitcoin

Le transazioni sono composte da input e output:

- Input: riferimenti a output di transazioni precedenti che si vogliono spendere, includono firme digitali per autorizzare la spesa.

- Output: definiscono a chi vengono inviati i bitcoin e in quale quantità.
  Bitcoin utilizza il modello UTXO (Unspent Transaction Output), dove ogni output non speso può essere usato come input in una nuova transazione.

Metadati nelle transazioni

Bitcoin supporta l'uso di campi come OP_RETURN per inserire metadati nelle transazioni, utili per applicazioni come la notarizzazione o tokenizzazione.

Bitcoin Script

È il linguaggio di scripting stack-based, non Turing-completo, usato per definire le condizioni di spesa degli output.
Esempio di script standard Pay-to-PubKey-Hash (P2PKH):

OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

Proof of Burn

Un meccanismo alternativo in alcune blockchain dove si “bruciano” token inviandoli a indirizzi irrecuperabili per dimostrare impegno o ottenere vantaggi in nuove catene.

Altri aspetti

- Il limite massimo di bitcoin è 21 milioni.
- La rete utilizza Proof of Work per il consenso.
- Sono possibili fork, soft fork (aggiornamenti compatibili) e hard fork (cambiamenti incompatibili).

Introduzione a Ethereum

Ethereum è una piattaforma blockchain programmabile, nata per superare i limiti di Bitcoin, specializzata nell’esecuzione di smart contract.

Modello account-based

Ethereum usa due tipi di account:

- Externally Owned Account (EOA): controllato da chiavi private.
- Contract Account: codice eseguibile (smart contract).

Blocco, epoche e slot

- Ethereum 2.0 introduce il concetto di slot (unità temporale di ~12 secondi) e epoch (32 slot, circa 6 minuti).
- I validatori sono assegnati a slot specifici per proporre blocchi nel sistema Proof of Stake.

Gas e transazioni

- Ogni operazione in Ethereum ha un costo in gas, misurato in unità computazionali.
- Il gas è pagato in ether (ETH), la criptovaluta nativa di Ethereum.
- Il gas price è il costo per unità di gas, determinato dal mercato.
- Le transazioni contengono nonce, destinatario, valore, gas, gas price, dati (per chiamate smart contract) e firma.

Burning di ETH

Con l'EIP-1559, parte del gas pagato viene bruciata, riducendo l'offerta totale di ETH e introducendo un meccanismo deflazionistico.

Patricia Merkle Trie

Ethereum utilizza questa struttura dati per mantenere efficientemente e in modo verificabile lo stato della blockchain:

- Tiene traccia degli account, dello storage e delle transazioni.
- Consente aggiornamenti rapidi e proof crittografici.

Smart Contract e Solidity

Gli smart contract sono programmi che si eseguono sulla blockchain in modo automatico, sicuro e decentralizzato.

Solidity

È il linguaggio di programmazione più usato per scrivere smart contract su Ethereum.

- Sintassi simile a JavaScript o C++.
- Supporta tipi statici, funzioni, ereditarietà, eventi, modificatori.
- Funzioni speciali: payable (accetta ETH), view (non modifica stato), fallback (chiamata quando nessun'altra funzione è corrispondente).

Sicurezza

- Evitare vulnerabilità come il reentrancy (funzioni che possono essere chiamate ripetutamente prima di completare un'operazione).
- Pattern come checks-effects-interactions per garantire coerenza.
- Access control con onlyOwner o ruoli personalizzati.

Standard token

- ERC-20: token fungibili.
- ERC-721: token non fungibili (NFT).
- ERC-1155: token multi-standard che possono rappresentare sia fungibili sia non fungibili.

Non-fungible Token (NFT)

Gli NFT rappresentano asset unici e non intercambiabili, molto usati per arte digitale, collezionabili e giochi.

Caratteristiche

- Ogni NFT ha un identificatore unico tokenId.
- I metadati (nome, immagine, descrizione) sono spesso salvati off-chain su IPFS o simili.
- Basati su standard come ERC-721 o ERC-1155.

Funzioni principali

- ownerOf(tokenId): restituisce il proprietario.
- transferFrom(from, to, tokenId): trasferisce l'NFT.
- safeTransferFrom: trasferimento sicuro che verifica se il ricevente può gestire NFT.

Applicazioni

- Arte digitale, musica, biglietti, diritti digitali, oggetti di gioco, metaverso.

Hyperledger

Hyperledger è un progetto open source della Linux Foundation che fornisce framework blockchain per soluzioni aziendali permissioned (con identità note).

Hyperledger Fabric

- Architettura modulare con canali per privacy e segmentazione della rete.
- Chaincode = smart contract scritti in Go, JavaScript o Java.
- Supporta diversi meccanismi di consenso (Raft, Kafka).
- Gestione delle identità tramite Membership Service Provider (MSP).

Altri progetti Hyperledger

- Sawtooth: modularità, Proof of Elapsed Time (PoET) con Intel SGX.
- Besu: client Ethereum compatibile enterprise.
- Indy: identità decentralizzata (DID).
- Aries: framework per comunicazioni sicure tra identità digitali.
- Ursa: libreria di crittografia condivisa.

Applicazioni tipiche

Supply chain, gestione documentale, identità digitale, finanza, sanità.