**DDBMS Backup and Recovery**

ssh -i labsuser.pem ec2-user@34.201.173.67

ssh -i labsuser.pem ec2-user@44.201.73.180

CREATE KEYSPACE arya

WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};

CREATE TABLE arya.dosen (

id uuid PRIMARY KEY,

nama\_dosen text

);

CREATE TABLE arya.mahasiswa (

nim int PRIMARY KEY,

nama text

);

cd data/data/

SET PASSWORD

sudo su

cd /etc/ssh/

nano sshd\_configs

passwor autentication = yes

systemctl restart sshd.service

passwd root

masuk ke cd data/data/arya/

cd /home/

mkdir -p backupnya

sudo mkdir -p arya/dosen

scp \* root@172.31.95.247:/home/backupnya/arya/dosen

pada node2 ketika mau restore

masuk ke cd cassandra,bin

./sstableloader -d 172.31.95.247 /home/backupnya/arya/mahasiswa

Pada node2

<CREATE KEYSPACE>

CREATE KEYSPACE arya

WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};

<CREATE TABLE>

CREATE TABLE arya.mahasiswa (

nim int PRIMARY KEY,

nama text

);

<EDIT CASSANDRA.YAML>

seeds : IP PRIVATE NODE2

listen\_address : IP PRIVATE NODE2

rpc\_address : 0.0.0.0

broadcast\_rpc\_address : IP PRIVATE NODE2

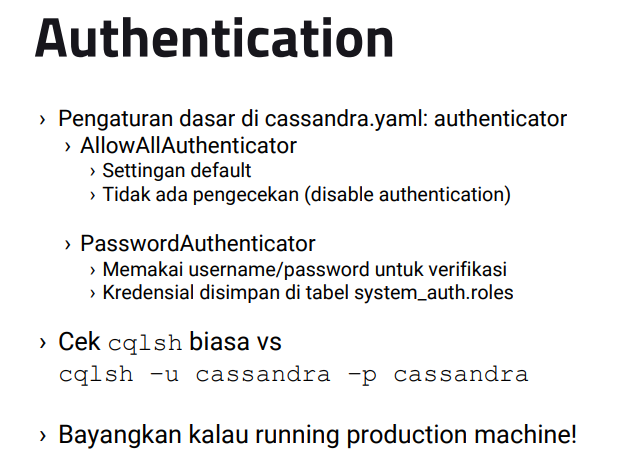
endpoint\_snitch : SimpleSnitch

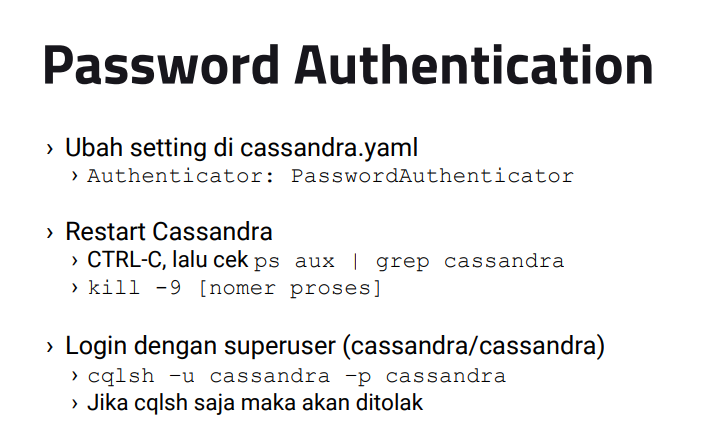
sudo su

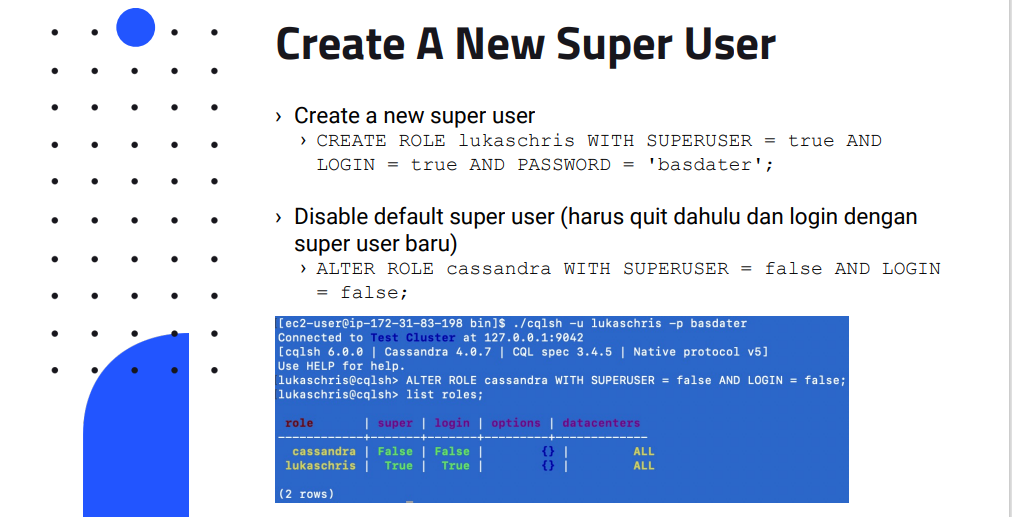
cd cassandra,bin

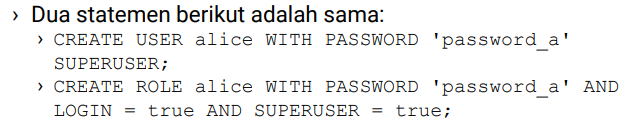
./sstableloader -d IPPRIVATENODE2 /home/backupnya/arya/mahasiswa

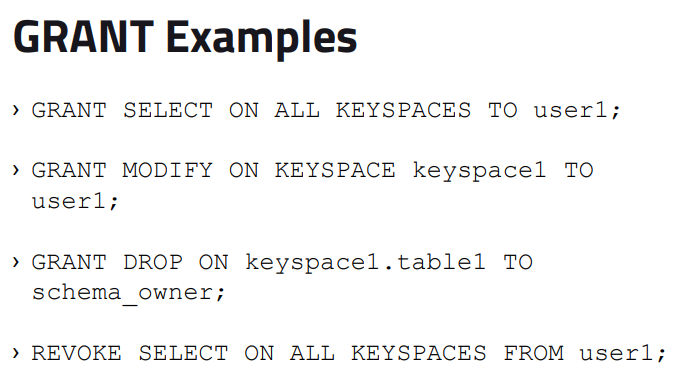
**DDBMS Security**

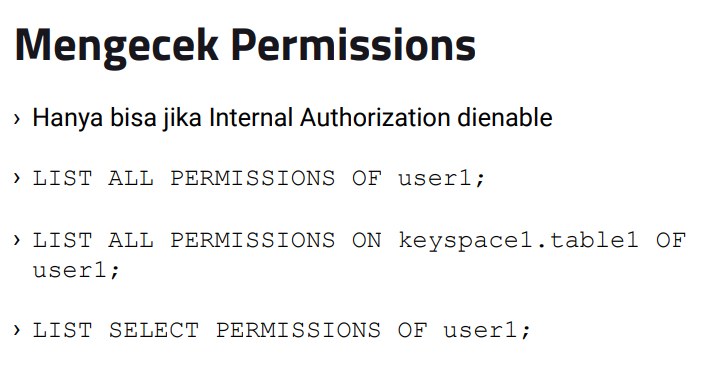












<<Buat Replikasi 2 NODE>>

ssh -i labsuser.pem ec2-user@18.232.181.154

ssh -i labsuser.pem ec2-user@44.192.33.66

INSTALL PADA 2 NODE

curl -OL https://dlcdn.apache.org/cassandra/4.0.11/apache-cassandra-4.0.11-bin.tar.gz

tar xzvf apache-cassandra-4.0.11-bin.tar.gz

mv apache-cassandra-4.0.11 cassandra

cd cassandra

sudo su

mkdir -p /var/lib/cassandra/{data,commitlog,saved\_caches}

mkdir -p /var/log/cassandra

chmod 777 /var/lib/cassandra

chmod 777 /var/log/cassandra

exit

sudo amazon-linux-extras install java-openjdk11

KONFIGURASI CASSANDRA.YAML

Pada node1

cluster name : mycluster

seeds : 172.31.78.104 IP PRIVATE NODE 1

listen\_address : 172.31.78.104 IP PRIVATE NODE 1

rpc\_address : 0.0.0.0

broadcast\_rpc\_address : 172.31.78.104 IP PRIVATE NODE 1

endpoint\_snitch : GossipingPropertyFileSnitch

Pada node2

cluster name : mycluster

seeds : 172.31.78.104 IP PRIVATE NODE 1

listen address : 172.31.67.204 IP PRIVATE NODE 2

rpc\_address : 0.0.0.0

broadcast\_rpc\_address : 172.31.67.204 IP PRIVATE NODE 2

endpoint\_snitch : GossipingPropertyFileSnitch

CREATE KEYSPACE basdater

WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};

CREATE TABLE mahasiswa (

nim int PRIMARY KEY,

nama text,

angkatan int

);

CREATE TABLE dosen (

nip int PRIMARY KEY,

nama text,

jabatan text

);

INSERT INTO mahasiswa (nim, nama, angkatan) VALUES (987001, 'Dr. Albert Einstein', 2023),

(987002, 'Dr. Marie Curie', 2023),

(987003, 'Mr. Carl Sagan', 2023);

INSERT INTO mahasiswa (nim, nama, angkatan) VALUES (123002, 'Jane Smith', 2021);

INSERT INTO mahasiswa (nim, nama, angkatan) VALUES (123003, 'Bob Williams', 2019);

INSERT INTO dosen (nip, nama, jabatan) VALUES (987001, 'Dr. Albert Einstein', 'Professor');

INSERT INTO dosen (nip, nama, jabatan) VALUES (987002, 'Dr. Marie Curie', 'Associate Professor');

INSERT INTO dosen (nip, nama, jabatan) VALUES (987003, 'Mr. Carl Sagan', 'Senior Lecturer');

KONFIGURASI PADA CASSANDRA.YAML

Authenticator: PasswordAuthenticator

Authorizer: CassandraAuthorizer

<<>>

cassandra/bin/cqlsh -u cassandra -p cassandra

BUAT SUPERUSER

CREATE ROLE kelompok WITH SUPERUSER = true AND LOGIN = true AND PASSWORD = 'basdater';

BUAT USER CASSANDRA NON AKTIF

ALTER ROLE cassandra WITH SUPERUSER = false AND LOGIN = false;

CREATE ROLE arya WITH LOGIN = true AND PASSWORD = 'arya';

CREATE ROLE indy WITH LOGIN = true AND PASSWORD = 'indy';

GRANT SELECT ON basdater.dosen TO arya;

GRANT MODIFY ON basdater.dosen TO arya;

GRANT ALTER ON basdater.dosen TO arya;

GRANT DROP ON basdater.dosen TO arya;

GRANT SELECT ON basdater.mahasiswa TO indy;

LIST ALL PERMISSIONS ON basdater.mahasiswa OF arya;

LIST ALL PERMISSIONS ON basdater.dosen OF arya;

LIST ALL PERMISSIONS ON basdater.dosen OF indy;

LIST ALL PERMISSIONS ON basdater.mahasiswa OF indy;

INSERT INTO dosen (nip, nama, jabatan)

VALUES (71210792, 'Indy Exodio', 'Junior Lecturer');

UPDATE dosen

SET jabatan = 'Professor'

WHERE nip = 987003

DELETE FROM dosen

WHERE nip = 987003

**Cassandra Client Drivers**

ssh -i labsuser.pem ec2-user@44.210.141.9

curl -OL https://dlcdn.apache.org/cassandra/4.0.11/apache-cassandra-4.0.11-bin.tar.gz

tar xzvf apache-cassandra-4.0.11-bin.tar.gz

mv apache-cassandra-4.0.11 cassandra

cd cassandra

sudo su

mkdir -p /var/lib/cassandra/{data,commitlog,saved\_caches}

mkdir -p /var/log/cassandra

chmod 777 /var/lib/cassandra

chmod 777 /var/log/cassandra

exit

sudo dnf install java-11-amazon-corretto-devel

sudo dnf update -y

sudo dnf groupinstall "Development Tools"

sudo dnf install -y openssl-devel bzip2-devel libffi-devel zlib-devel

cd /opt

sudo wget https://www.python.org/ftp/python/3.8.12/Python-3.8.12.tgz

sudo tar zxf Python-3.8.12.tgz

cd Python-3.8.12

sudo ./configure --enable-optimizations

sudo make altinstall

python3.8 --version

sudo rm -rf /opt/Python-3.8.12

sudo alternatives --install /usr/bin/python3 /usr/local/bin/python3.8 1

python3 --version

sudo yum install -y python3 python3-pip

pip install cassandra-driver

buat keyspace

CREATE KEYSPACE basdater

WITH replication = {'class' : 'SimpleStrategy', 'replication\_factor' : 1};

USE basdater;

buat tabel

CREATE TABLE Mahasiswa (

nim int PRIMARY KEY,

nama text

);

nano tes\_koneksi.py

from cassandra.cluster import Cluster

try:

cluster = Cluster(['localhost'])

session = cluster.connect('basdater')

print("Koneksi Berhasil")

except Exception as e:

print(f"Error: {e}")

print("Koneksi Gagal")

finally:

# Tutup sesi dan cluster setelah digunakan

if cluster:

session.shutdown()

cluster.shutdown()

nano insert\_data.py

from cassandra.cluster import Cluster

try:

cluster = Cluster(['localhost'])

session = cluster.connect('basdater')

# Daftar data yang akan dimasukkan

data\_to\_insert = [

(71210767, "Arya Ramandika Marunduh"),

(71210792, "Indy Exodio")

]

# Menambahkan pernyataan INSERT ke dalam batch

for data in data\_to\_insert:

query = "INSERT INTO mahasiswa (nim, nama) VALUES (%s, %s)"

session.execute(query, data)

print("Data mahasiswa berhasil dimasukkan")

except Exception as e:

print(f"Error: {e}")

print("Insert data gagal")

finally:

# Tutup sesi dan cluster setelah digunakan

if cluster:

session.shutdown()

cluster.shutdown()

nano retrieve\_all\_data.py

from cassandra.cluster import Cluster

try:

cluster = Cluster(['localhost'])

session = cluster.connect('basdater')

# Pernyataan SELECT untuk mengambil semua data dari tabel mahasiswa

query = "SELECT \* FROM mahasiswa"

result = session.execute(query)

# Cetak hasil retrieve data

print("Data Mahasiswa:")

for row in result:

print(f"NIM: {row.nim}, Nama: {row.nama}")

print("Retrieve all data berhasil")

except Exception as e:

print(f"Error: {e}")

print("Retrieve all data gagal")

finally:

# Tutup sesi dan cluster setelah digunakan

if cluster:

session.shutdown()

cluster.shutdown()

nano retrieved\_filtered\_data.py

from cassandra.cluster import Cluster

try:

cluster = Cluster(['localhost'])

session = cluster.connect('basdater')

# Filter berdasarkan NIM tertentu

nim\_filter = 71210767

query = f"SELECT \* FROM mahasiswa WHERE nim = {nim\_filter}"

result = session.execute(query)

# Cetak hasil retrieve data

print(f"Data Mahasiswa dengan NIM {nim\_filter}:")

for row in result:

print(f"NIM: {row.nim}, Nama: {row.nama}")

print("Retrieve filtered data berhasil")

except Exception as e:

print(f"Error: {e}")

print("Retrieve filtered data gagal")

finally:

# Tutup sesi dan cluster setelah digunakan

if cluster:

session.shutdown()

cluster.shutdown()

**ScylaDB**ssh -i "labsuser.pem" ubuntu@ec2-35-175-105-16.compute-1.amazonaws.com

sudo su

sudo apt-get remove -y abrt

sudo gpg --homedir /tmp --no-default-keyring --keyring /etc/apt/keyrings/scylladb.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys d0a112e067426ab2

sudo wget -O /etc/apt/sources.list.d/scylla.list http://downloads.scylladb.com/deb/debian/scylla-5.4.list

sudo apt-get update

sudo apt-get install -y scylla

apt-cache madison scylla

apt-get install scylla{,-server,-jmx,-tools,-tools-core,-kernel-conf,-node-exporter,-conf,-python3}=5.4.0~rc3-0.20231129.b956646ba231-1

sudo apt-get update

sudo apt install openjdk-11-jre-headless

sudo update-java-alternatives --jre-headless -s java-1.11.0-openjdk-amd64

sudo nano /etc/scylla/scylla.yaml

sudo scylla\_setup

sudo systemctl start scylla-server

nodetool status

cqlsh

35.175.105.16

3.80.220.246

apt-get install scylla{,-server,-jmx,-tools,-tools-core,-kernelconf,-node-exporter,-conf,-python3}=

apt-get install scylla{,-server,-jmx,-tools,-tools-core,-kernel-conf,-node-exporter,-conf,-python3}=[4.5.3+dfsg-1]

apt-get install scylla{,-server,-jmx,-tools,-tools-core,-kernel-conf,-node-exporter,-conf,-python3}=5.4.0~rc3-0.20231129.b956646ba231-1

**Blockchain**

Blockchain adalah teknologi penyimpanan data digital yang bekerja dengan cara menghubungkan serangkaian blok data secara kronologis dan kriptografis. Dalam konteks ini, "blok" merujuk pada set data tertentu yang disimpan dalam bentuk sebuah unit.

1. **Struktur Berantai**: Setiap blok dalam blockchain mengandung hash (sebuah string unik) dari blok sebelumnya, sehingga membentuk rantai blok yang tak terputus.
2. **Distribusi**: Blockchain umumnya tersimpan di banyak komputer yang saling terhubung, membentuk jaringan peer-to-peer (P2P). Ini berarti tidak ada satu entitas tunggal yang mengontrol atau memiliki seluruh blockchain.
3. **Kriptografi**: Setiap transaksi atau perubahan pada blockchain dilindungi dengan teknik kriptografi. Ini memastikan keamanan dan integritas data.
4. **Transparansi dan Imutabilitas**: Sekali data dimasukkan ke dalam blockchain, sangat sulit untuk mengubahnya tanpa mendeteksi. Hal ini memberikan tingkat transparansi dan keamanan yang tinggi.
5. **Konfirmasi dan Konsensus**: Transaksi baru ditambahkan ke blockchain setelah mendapatkan persetujuan dari mayoritas peserta jaringan. Ini memastikan konsistensi dan integritas data di seluruh jaringan.
6. **Aplikasi Luas**: Meskipun dikenal pertama kali sebagai infrastruktur di balik mata uang kripto seperti Bitcoin, teknologi blockchain telah menemukan penerapan dalam berbagai industri, termasuk keuangan, logistik, kesehatan, dan banyak lagi.

A blockchain concept with text

Description automatically generated

1. **Mining**:
   * **Definisi**: Mining adalah proses di mana transaksi baru ditambahkan ke blockchain dan blok-blok baru dibuat. Ini juga merupakan cara di mana unit kripto seperti Bitcoin diciptakan.
   * **Bagaimana Kerjanya**: Para penambang (miners) menggunakan kekuatan komputasi mereka untuk memecahkan teka-teki matematika kompleks. Jika mereka berhasil memecahkan teka-teki tersebut, mereka dapat menambahkan blok baru ke blockchain dan mendapatkan hadiah berupa kripto.
2. **Distributed P2P Network**:
   * **Definisi**: Jaringan peer-to-peer (P2P) adalah jaringan yang terdiri dari komputer atau node yang saling terhubung tanpa ada server pusat.
   * **Keuntungan**: Desentralisasi memastikan bahwa tidak ada satu titik kegagalan tunggal, sehingga meningkatkan keamanan dan keandalan jaringan. Selain itu, tidak ada satu entitas yang mengendalikan seluruh jaringan.
3. **Consensus Protocol**:
   * **Definisi**: Protokol konsensus adalah aturan atau algoritma yang digunakan oleh peserta jaringan untuk mencapai kesepakatan tentang status blockchain (seperti transaksi yang valid atau blok yang akan ditambahkan).
   * **Contoh**: Proof-of-Work (PoW), Proof-of-Stake (PoS), dan Byzantine Fault Tolerance (BFT) adalah beberapa protokol konsensus yang populer.
4. **Immutable Ledger**:
   * **Definisi**: Ledger (buku besar) yang tidak dapat diubah-ubah. Ini berarti sekali data ditambahkan ke dalam blockchain, sangat sulit untuk mengubah atau menghapusnya.
   * **Keamanan**: Sifat imutabilitas memberikan keamanan dan kepercayaan pada data dalam blockchain, karena perubahan data akan dengan cepat dideteksi.
5. **Hash Cryptography**:
   * **Definisi**: Kriptografi hash adalah teknik matematika yang mengonversi data input menjadi string angka dan huruf yang unik dengan panjang tetap.
   * **Pentingnya**: Hash digunakan dalam blockchain untuk mengamankan dan mengintegritaskan data. Sebuah blok dalam blockchain mengandung hash dari blok sebelumnya, sehingga menciptakan rantai yang aman.

A screenshot of a computer

Description automatically generated

SHA-256 (Secure Hash Algorithm 256-bit) adalah fungsi hash kriptografis yang menghasilkan keluaran (output) dengan panjang tetap 256 bit (32 byte). Ini adalah salah satu dari beberapa fungsi hash dalam keluarga SHA-2 yang dikembangkan oleh National Security Agency (NSA) Amerika Serikat. Berikut adalah beberapa poin penting mengenai SHA-256:

1. **Penggunaan**: SHA-256 sering digunakan dalam berbagai aplikasi kriptografi dan keamanan informasi, termasuk dalam blockchain (seperti Bitcoin), SSL/TLS, dan banyak protokol keamanan lainnya.
2. **Unik**: Meskipun mungkin ada dua set data yang menghasilkan hash yang sama (disebut kolisi), sangat sulit untuk menemukan dua set data yang menghasilkan hash yang sama dengan SHA-256.
3. **Irreversibilitas**: Fungsi hash bersifat satu arah, yang berarti Anda tidak dapat mengembalikan input asli dari hash yang dihasilkan (kecuali dengan pendekatan brute-force, yang sangat tidak praktis untuk data dengan panjang yang panjang).
4. **Keamanan**: SHA-256 dianggap sebagai fungsi hash yang sangat aman dan memiliki resistensi yang baik terhadap berbagai serangan kriptanalisis.
5. **Dukungan Luas**: Karena keamanan dan efisiensinya, SHA-256 telah menjadi standar industri untuk banyak aplikasi kriptografi.

Dalam konteks blockchain, SHA-256 digunakan untuk menghasilkan hash dari blok, transaksi, atau data lainnya.

A diagram of a house

Description automatically generated

"Immutable ledger" merujuk pada buku besar atau sistem rekaman data yang, setelah data dimasukkan atau transaksi dicatat, tidak dapat diubah atau dihapus. Dengan kata lain, sekali data atau transaksi ditambahkan ke dalam ledger, itu akan tetap ada di sana dalam bentuk yang sama untuk selamanya, tanpa kemampuan untuk diubah tanpa jejak.

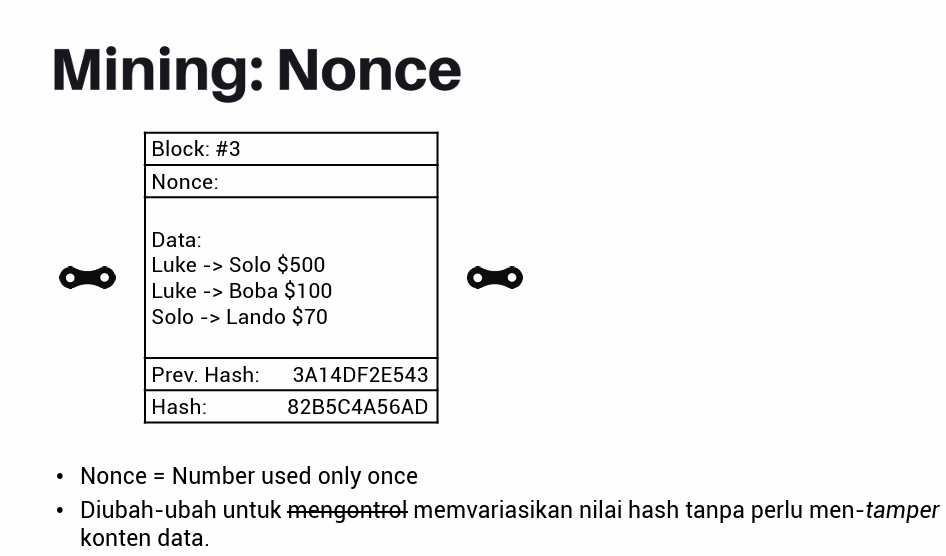
Beberapa poin penting tentang "immutable ledger" meliputi:

1. **Keamanan**: Sifat imutabilitas dari ledger memberikan tingkat keamanan tambahan. Karena data tidak dapat diubah, manipulasi atau pemalsuan data menjadi jauh lebih sulit.
2. **Integritas**: Fakta bahwa ledger tidak dapat diubah memastikan integritas data. Ketika seseorang melihat data di ledger, mereka dapat yakin bahwa data tersebut sama persis seperti saat pertama kali ditambahkan.
3. **Penerapan di Blockchain**: Konsep imutable ledger sangat relevan dalam konteks blockchain. Blockchain adalah contoh utama dari immutable ledger di mana setiap blok data yang ditambahkan ke dalam rantai (chain) tidak dapat diubah. Karena setiap blok mengandung hash dari blok sebelumnya, mencoba mengubah satu blok akan memerlukan perubahan pada semua blok setelahnya, yang memastikan keamanan dan integritas.
4. **Kepercayaan**: Sifat imutabilitas memberikan tingkat kepercayaan kepada pihak-pihak yang terlibat, karena mereka tahu bahwa data yang telah dicatat atau transaksi yang telah dilakukan tidak akan berubah atau hilang.

Dalam banyak aplikasi, khususnya di bidang keuangan, keamanan, dan kesehatan, kebutuhan untuk memiliki buku besar yang imutabel sangat penting untuk memastikan keandalan, keamanan, dan integritas data

Distributed Peer-to-Peer (P2P) Network adalah jaringan komputasi yang terdiri dari sejumlah besar node atau komputer yang terhubung secara langsung satu sama lain tanpa ada server pusat yang mengendalikan komunikasi. Setiap node dalam jaringan ini berfungsi sebagai klien dan server secara bersamaan, memungkinkan komunikasi langsung antara mereka. Berikut adalah beberapa konsep dan karakteristik utama dari Distributed P2P Network:

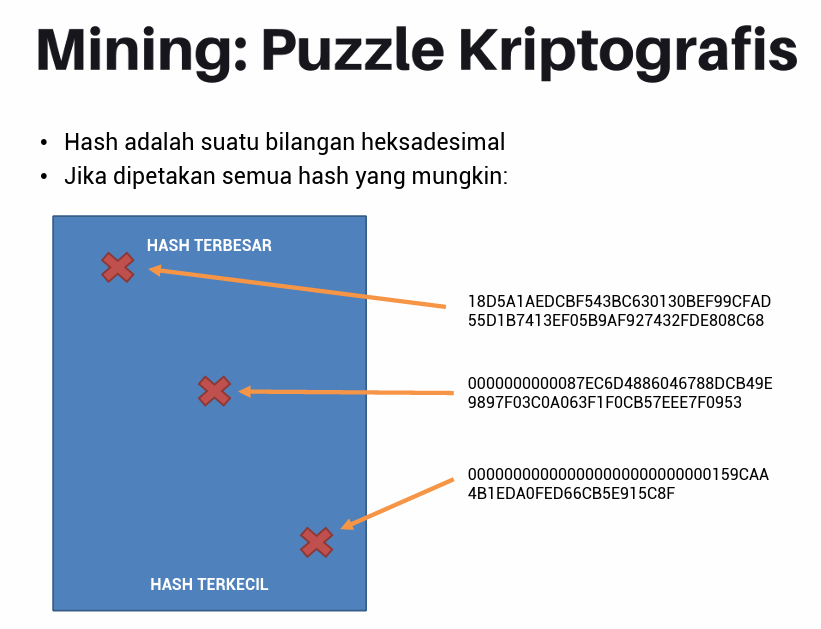
1. **Desentralisasi**: Tidak ada satu entitas pusat atau server yang mengontrol atau mengatur seluruh jaringan. Sebagai gantinya, kontrol dan keputusan didistribusikan di antara semua node di jaringan.
2. **Komunikasi Langsung**: Node dalam jaringan P2P dapat berkomunikasi secara langsung satu sama lain tanpa perlu melalui server pusat. Ini memungkinkan transfer data yang cepat dan efisien.
3. **Skalabilitas**: Karena sifatnya yang terdistribusi, jaringan P2P seringkali dapat dengan mudah diperluas dengan menambahkan lebih banyak node tanpa mempengaruhi kinerja keseluruhan jaringan.
4. **Keamanan**: Beberapa implementasi P2P menggunakan kriptografi dan algoritma keamanan lainnya untuk melindungi data dan privasi pengguna.
5. **Penggunaan Umum**: Meskipun seringkali dihubungkan dengan pertukaran file (seperti protokol BitTorrent), jaringan P2P juga digunakan dalam berbagai aplikasi lain, termasuk komunikasi (seperti VoIP), transaksi keuangan (seperti kriptokurensi), dan penyimpanan data terdistribusi.
6. **Heterogenitas**: Node-node dalam jaringan P2P dapat bervariasi dalam hal perangkat keras, perangkat lunak, dan lokasi geografis. Ini menciptakan lingkungan yang heterogen dan resilient.
7. **Kesulitan dalam Manajemen**: Karena sifatnya yang terdesentralisasi, mengatur dan mengelola jaringan P2P bisa menjadi tantangan, terutama dalam hal pemeliharaan, keamanan, dan penegakan aturan.



Dalam konteks blockchain dan kriptografi, "nonce" adalah singkatan dari "number only used once" atau "nomor yang hanya digunakan sekali". Konsep nonce sangat penting dalam proses mining pada beberapa blockchain, terutama blockchain seperti Bitcoin yang menggunakan algoritma Proof-of-Work (PoW) untuk mencapai konsensus dan memvalidasi transaksi.

Berikut adalah beberapa poin penting terkait nonce:

1. **Fungsi Nonce**: Dalam proses mining, nonce digunakan sebagai variabel yang dimasukkan ke dalam blok baru yang sedang dikerjakan. Penambang (miner) mencoba berbagai nilai nonce untuk mencari kombinasi yang, ketika dikombinasikan dengan data transaksi lainnya, menghasilkan hash dari blok yang memenuhi kriteria tertentu.
2. **Pencarian Solusi**: Tujuan utama dari penambangan adalah untuk menemukan "golden nonce" atau nilai nonce yang, ketika diinputkan ke dalam blok, menghasilkan hash yang kurang dari target tertentu (yaitu, hash yang memiliki sejumlah nol di awalnya dalam sistem hexadesimal).
3. **Proses Trial and Error**: Mencari nilai nonce yang benar untuk memenuhi syarat tertentu adalah proses trial and error. Penambang akan mencoba berbagai nilai nonce sampai menemukan yang sesuai.
4. **Keamanan dan Konsensus**: Algoritma PoW, yang memanfaatkan nonce, memastikan bahwa penambang harus berinvestasi dalam sumber daya (seperti waktu komputasi dan listrik) untuk mencari nonce yang valid. Ketika nonce yang benar ditemukan, blok tersebut dapat ditambahkan ke blockchain, dan transaksi di dalamnya dianggap sah oleh jaringan.
5. **Pentingnya Keunikan**: Karena nonce hanya digunakan sekali dan harus unik untuk setiap blok, ini membantu mencegah kemungkinan duplikasi atau manipulasi yang tidak sah.



A screenshot of a puzzle

Description automatically generated

Puzzle kriptografi dalam konteks penambangan (mining) blockchain, khususnya dengan algoritma Proof-of-Work (PoW), mengacu pada teka-teki matematika yang harus dipecahkan oleh penambang untuk menghasilkan blok baru yang valid. Berikut adalah beberapa konsep terkait:

1. **Hash adalah Bilangan Heksadesimal**:
   * Dalam kriptografi dan komputasi, hash adalah representasi angka heksadesimal dari data. Sebuah hash adalah string angka dan huruf yang dihasilkan oleh fungsi hash dari data input. Hash memiliki panjang tetap meskipun panjang data inputnya bisa bervariasi.
2. **Target Hash**:
   * Dalam proses penambangan dengan algoritma PoW, jaringan menetapkan sebuah target hash yang harus dicapai oleh penambang untuk memvalidasi dan menambahkan blok baru ke blockchain. Target ini biasanya dinyatakan dalam bentuk nilai numerik tertentu, dan hash dari blok harus lebih rendah dari target tersebut untuk dianggap valid.
3. **Mencari Hash yang Sesuai**:
   * Penambang akan mencoba berbagai nilai nonce (Number Only Used Once) dan kombinasi lain dari data blok untuk menciptakan hash yang memenuhi kriteria target yang ditetapkan. Ini adalah proses trial and error yang melibatkan komputasi berat.
4. **Kriteria Tertentu (Misalnya, Nilai Y)**:
   * Nilai Y adalah batas numerik tertentu yang ditetapkan oleh jaringan. Hash dari blok yang dihasilkan oleh penambang harus berada di bawah nilai Y untuk memenuhi kriteria dan validasi. Dengan kata lain, hash tersebut harus memiliki sejumlah nol di awalnya dalam representasi binernya (dalam sistem heksadesimal, ini berarti hash tersebut harus lebih kecil dari nilai Y).
5. **Keamanan dan Konsensus**:
   * Dengan menetapkan kriteria target yang harus dipenuhi, algoritma PoW memastikan bahwa penambang harus berinvestasi dalam sumber daya (seperti waktu komputasi dan listrik) untuk mencari nonce yang valid. Ini membantu mencegah serangan dan manipulasi, serta memastikan bahwa transaksi yang masuk ke dalam blockchain telah diverifikasi dan disetujui oleh jaringan.

• Perhatikan sifat Avalance Effect: akan mencegah pencurangan dengan menebak nilai nonce agar dapat menghasilkan hash sesuai target (GOLDEN NONCE). • Tidak serta merta menurunkan nilai nonce = hash mengecil. • Itulah sebabnya butuh waktu dan resource besar untuk mendapatkan (me-mining) nilai sesuai target. • Tanpa ada Avalanche Effect maka cryptographic puzzle tidak akan berjalan. • Tidak (belum) dimungkinkan ada pemrediksian nilai hash selain dengan tebak tebakan. • Jika sudah ada algoritma untuk reverse engineering, maka seluruh sistem akan runtuh.



Diciptakan oleh Leslie Lamport, Robert Shostak, dan Marshall Pease (1982). Merupakan algoritma yang menentukan keputusan hasil akhir berdasarkan consensus of peers (70%).

Byzantine Fault Tolerance (BFT) adalah konsep dalam ilmu komputer dan kriptografi yang merujuk pada kemampuan suatu sistem untuk tetap berfungsi dengan benar meskipun beberapa komponennya gagal atau berperilaku tidak sesuai. Konsep ini mendapat namanya dari "Masalah Jenderal Byzantine," yang menggambarkan situasi di mana entitas komputer mungkin mengalami kegagalan (seperti crash, delay, atau perilaku tidak terduga lainnya) dan komunikasi yang tidak dapat diandalkan.

Berikut beberapa poin kunci terkait Byzantine Fault Tolerance:

1. **Definisi**: Sistem yang memiliki sifat Byzantine Fault Tolerance dianggap dapat berfungsi dengan benar bahkan ketika sejumlah komponen dalam sistem tersebut (seperti node atau entitas) mengalami kegagalan atau berperilaku tidak sesuai.
2. **Pentingnya Konsensus**: Dalam konteks blockchain dan jaringan terdistribusi lainnya, BFT sangat penting untuk mencapai konsensus. Meskipun beberapa entitas dalam jaringan mungkin berusaha menipu atau mengganggu, mekanisme BFT memastikan bahwa mayoritas entitas yang jujur ​​akan mencapai konsensus yang benar.
3. **Protokol Konsensus**: Ada beberapa protokol konsensus yang dirancang untuk mencapai toleransi kesalahan Byzantine, seperti Practical Byzantine Fault Tolerance (PBFT), Federated Byzantine Agreement (FBA), dan lainnya. Setiap protokol memiliki pendekatan dan karakteristik khususnya sendiri.
4. **Keamanan**: Dengan memiliki mekanisme BFT, sistem dapat menawarkan tingkat keamanan dan ketahanan yang lebih tinggi terhadap serangan, kesalahan, atau gangguan dari pihak luar.
5. **Aplikasi**: Selain blockchain, konsep Byzantine Fault Tolerance juga digunakan dalam sistem distribusi data, jaringan komputer, dan berbagai aplikasi lainnya di mana keandalan dan konsensus adalah kritis.

Consensus Protocol Consensus Protocol adalah protokol yang memungkinkan komputer/peers tahu kapan untuk menambah blok dan kapan untuk menolak menambah blok, dengan dasar konsep Byzantine Fault Tolerance. Kita sudah mempelajari di slide Distributed P2P Network bahwa tidak mungkin bagi hacker untuk menyerang satu peer untuk memanipulasi blockchain.

Bagaimana jika serangannya adalah penambahan unauthorized blok baru?

Bagaimana pula jika salah satu peer menambah blok baru hampir bersamaan dengan peer lain yang juga menambah blok baru (competing chains)?  
Dua skenario yang Anda sebutkan adalah situasi yang dapat mempengaruhi integritas dan keamanan dari jaringan blockchain. Mari kita jelajahi masing-masing skenario tersebut:

1. **Penambahan Blok Tidak Sah (Unauthorized)**:

Jika seorang penambang atau entitas jahat mencoba menambahkan blok yang tidak sah ke dalam blockchain, beberapa hal mungkin terjadi:

* + **Validasi**: Sebelum blok baru ditambahkan ke blockchain, biasanya akan ada proses validasi oleh node-node lain di jaringan. Jika blok tersebut tidak memenuhi kriteria validasi yang ditetapkan (misalnya, tidak memenuhi syarat konsensus seperti PoW), maka blok tersebut akan ditolak oleh jaringan.
  + **Reputasi dan Hukuman**: Dalam beberapa sistem, node atau penambang yang mencoba menambahkan blok tidak sah mungkin mendapatkan hukuman atau sanksi. Ini bisa berupa denda, penurunan reputasi, atau mekanisme lain yang mendorong perilaku yang jujur dalam jaringan.

1. **Competing Chains (Rantai Bersaing)**:

Ketika dua atau lebih penambang menemukan solusi untuk blok berikutnya hampir bersamaan, hal ini dapat menyebabkan terbentuknya dua rantai yang bersaing. Ini dapat terjadi karena delay dalam penyampaian informasi antara node atau variasi dalam kriteria validasi.

* + **Temporary Fork**: Ini adalah situasi di mana jaringan sementara memiliki dua rantai yang bersaing. Namun, protokol konsensus akan bekerja untuk memastikan bahwa hanya satu rantai yang tetap ada di jaringan.
  + **Resolve dengan Konsensus**: Protokol konsensus (seperti PoW) memiliki mekanisme bawaan untuk menyelesaikan fork. Biasanya, rantai dengan proof-of-work terbesar akan dipilih sebagai rantai yang valid, sementara rantai lainnya akan ditolak.
  + **Risiko 51% Attack**: Dalam skenario ekstrem, jika satu entitas mengendalikan lebih dari 50% kekuatan hash jaringan (hash power), mereka dapat dengan mudah memanipulasi rantai dengan menambang lebih cepat daripada jaringan lainnya. Namun, ini akan merusak kepercayaan dan integritas jaringan, dan ada insentif ekonomi yang kuat bagi entitas untuk bertindak dengan jujur.

Proof-of-Work (PoW) • Dengan bantuan konsep puzzle kriptografis, peer yang mau menambah node perlu investasi hardware (mining rigs) untuk mencapai golden nonce. • Hasil hash dengan golden nonce ini adalah proof of worknya untuk memenangkan cryptographic challenge. • Ada upah atas mining yang menghasilkan blok yang berisi hash dengan golden nonce, di sisi lain ada hukuman jika blok yang dihasilkan berupa malicious block. • Tiap node sebelum dipropagasikan ke jaringan harus memenuhi ceklist ketat untuk memfilter malicious block

Komputer yang memiliki 51% hashing power akan memenangkan competing chains. Harapannya malicious miner/trader tidak pernah mencapai konsensus. Blok yang lepas dan tidak dipakai disebut orphaned block. Miner yang kalah konsensus juga tidak akan mendapat mining fee