PROJECT HELIX

Penjelasan Bab 00 sampai Bab 03

Git sebagai metafora version control untuk realitas

Bab 00 - Prolog	commit, log, signature, branch, checkout, remote
Bab 01 - INIT	init, snapshot, branch, merge, conflict
Bab 02 - STATUS	working tree, status, untracked, staging area
Bab 03 - ADD	add, add --patch, restore --staged, staged identity

Disusun dari materi penjelasan Bab 00, 01, 02, dan 03

Pengantar

Dokumen ini menyusun penjelasan empat bab awal Project Helix dalam satu alur yang utuh. Fokusnya bukan hanya pada misteri Adrian dan Maya, tetapi juga pada cara setiap bab memperkenalkan satu lapisan konsep Git lalu menerapkannya pada realitas. Karena itu, pembaca dapat mengikuti cerita sekaligus memahami istilah teknis seperti commit, branch, init, status, staging area, dan add tanpa harus memiliki latar belakang pemrograman.

Peta konsep Bab 00 sampai Bab 03

- **Bab 00 - Prolog:** commit, log, signature, branch, checkout, remote
- **Bab 01 - INIT:** init, snapshot, branch, merge, conflict
- **Bab 02 - STATUS:** working tree, status, untracked, staging area
- **Bab 03 - ADD:** add, add --patch, restore --staged, staged identity

BAB 00

Prolog: Commit dari Orang Mati

Inti bab: Prolog ini pada dasarnya adalah pintu masuk ke seluruh cerita. Di sini pembaca pertama kali diperkenalkan pada gagasan bahwa Git bukan cuma alat untuk mencatat perubahan kode, tetapi mungkin juga mencatat dan mengubah realitas.

Cerita dimulai pukul 02.13 ketika laptop Adrian tiba-tiba menyala sendiri di laboratorium QuantumForge. Adrian tertidur setelah bekerja malam, lalu menemukan sebuah commit baru:

```
[main 9f4c12a] fix: prevent Adrian's death

Author: Maya Pradipta
```

Masalahnya, Maya sudah meninggal tiga tahun sebelumnya.

Maksud adegan awalnya

Secara sederhana, Adrian melihat pesan:

> *“Perubahan untuk mencegah kematian Adrian.”*

Dan yang membuat perubahan itu tercatat sebagai:

> *Maya, orang yang sudah mati.*

Awalnya Adrian mengira ini cuma prank atau manipulasi tampilan terminal.

Tetapi setelah dia menjalankan:

```
git log -1
```

commit itu benar-benar ada di dalam sejarah repository, bukan sekadar tulisan palsu di layar.

Jadi misterinya langsung menjadi:

> *Siapa yang bisa membuat commit memakai identitas orang mati?*

Apa itu commit?

Prolog sengaja menjelaskan konsep Git dari nol.

Bayangkan proyek seperti album foto.

Setiap kali Anda merasa keadaan proyek penting untuk disimpan, Anda mengambil satu “foto” kondisi proyek.

Foto itulah kira-kira commit.

Commit menyimpan:

- perubahan,
- siapa yang membuatnya,
- kapan,
- pesan perubahan,
- dan hubungan dengan commit sebelumnya.

Jadi:

```
commit A
   ↓
commit B
   ↓
commit C
```

adalah sejarah proyek.

`git log` dipakai untuk melihat sejarah tersebut.

Karena itu commit Maya berarti bukan sekadar pesan chat. Ia tercatat sebagai bagian resmi dari sejarah.

Yang membuatnya semakin aneh: tanda tangan Maya valid

Adrian kemudian menjalankan:

```
git verify-commit 9f4c12a
```

Hasilnya:

```
Good signature from "Maya Pradipta"
```

Artinya commit tersebut ditandatangani memakai kunci kriptografi Maya.

Analogi gampangnya:

Kalau nama pengirim email bisa dipalsukan, tanda tangan digital lebih mirip stempel resmi yang hanya bisa dibuat dengan kunci rahasia tertentu.

Dan kunci rahasia Maya seharusnya berada di perangkat fisik dalam brankas yang tidak terhubung jaringan.

Jadi sekarang ada tiga kemungkinan:

```
1. seseorang berhasil mencuri kunci Maya
2. sistem keamanan QuantumForge bohong
3. Maya atau sesuatu yang memiliki akses Maya masih aktif
```

Cerita belum memberi jawaban.

Adrian juga punya ingatan yang tidak cocok

Prolog kemudian memberi petunjuk bahwa masalah Adrian bukan cuma commit.

Ia punya cincin dengan dua hash dan tanggal 12 Oktober 2037.

Ia tidak ingat membeli cincin itu.

Ia punya foto bersama Maya di Philadelphia.

Tetapi menurut ingatan resminya, mereka tidak pernah pergi ke sana.

Ia juga memiliki perasaan dan detail-detail hubungan yang jauh lebih kuat daripada sekadar hubungan profesional.

Artinya ada dua versi informasi di kepala Adrian:

```
Sejarah resmi:
Adrian dan Maya = rekan kerja
Ingatan / benda pribadi:
Adrian dan Maya = hubungan jauh lebih dekat
```

Ini adalah petunjuk awal tentang tema Mandela Effect personal.

Dalam dunia novel, mungkin ada ingatan dari sejarah lain yang masih “bocor” meskipun sejarah utama sudah berubah.

Lalu Adrian melihat isi commit

Adrian menjalankan:

```
git show 9f4c12a
```

Dan menemukan file:

```
src/simulation/adrian.json
```

Isinya hanya mengubah:

```
- "death": "2041-08-28T03:47:21+07:00"
+ "death": null
```

Bahasa gampangnya:

Sebelumnya sistem mempunyai data:

> *Adrian akan mati pukul 03:47:21.*

Commit Maya mengubahnya menjadi:

> *waktu kematian Adrian: kosong.*

Ini bukan sekadar:

> *“Maya memperingatkan Adrian akan mati.”*

Ia justru menghapus catatan kematian itu.

Dan saat Adrian melihat jam, masih pukul 02.17.

Berarti tersisa sekitar satu setengah jam.

Kenapa file itu begitu menakutkan?

Karena Project Helix bukan proyek simulasi biasa.

Helix sejak awal mencoba menjawab pertanyaan:

> *kalau keadaan dunia dapat disimpan sebagai data, apakah mengubah data tersebut bisa mengubah dunia?*

Dalam software biasa:

```
ubah file JSON
↓
hanya data program berubah
```

Dalam Helix, kemungkinan yang ditakuti adalah:

```
ubah file JSON manusia
↓
manusianya ikut berubah
```

Jadi:

```
"death": null
```

mungkin benar-benar berarti:

> *kematian Adrian dihapus dari realitas.*

Itulah premis besar novel.

Prolog juga mulai menjelaskan Mandela Effect

Maya sebelumnya meneliti orang-orang yang mengingat sejarah secara berbeda.

Misalnya:

- kematian tokoh pada tanggal berbeda,
- nama/ejaan berbeda,
- peta berbeda.

Ilmuwan biasa menganggapnya kesalahan memori.

Tetapi Maya melihat pola tersebut seperti:

```
git diff
```

yakni daftar perbedaan antara dua versi sejarah.

Jadi ide Maya kira-kira:

```
history A

vs

history B

↓ diff

orang tertentu mengingat

sisa dari history B

meskipun sekarang hidup di A
```

Itulah konsep awal Mandela Effect dalam novel.

Philadelphia dan Bermuda diperkenalkan sebagai bekas "perubahan sejarah"

Maya menghubungkan pola aneh tersebut dengan dua legenda:

- Philadelphia Experiment 1943,
- Bermuda Triangle.

Dalam cerita, Maya tidak bilang legenda-legenda itu pasti benar.

Ia menyebutnya bekas luka.

Artinya:

> *mungkin pernah terjadi perpindahan atau penggabungan sejarah di sana, lalu kejadian aslinya ditutupi oleh penjelasan yang lebih biasa*.

Ini menyiapkan ORPHEUS dan relay Bermuda yang menjadi penting nanti.

Kemudian Adrian mendapat peringatan

Nomor tidak dikenal mengirim pesan:

```
JANGAN CHECKOUT BRANCH MAYA.
```

Lalu:

```
KALAU KAMU SUDAH MELIHAT COMMIT-NYA,

MEREKA JUGA SUDAH TAHU KAMU MASIH HIDUP.
```

Ini memperkenalkan konsep berikutnya:

Apa itu branch dan checkout?

Branch adalah jalur sejarah alternatif.

Misalnya:

```
          ┌── branch maya

A → B → C

          └── main
```

Keduanya bermula dari sejarah yang sama, tetapi kemudian mengambil jalan berbeda.

`checkout` berarti berpindah ke salah satu jalur.

Dalam Git biasa:

```
git checkout maya
```

akan mengganti file proyek agar sesuai dengan branch `maya`.

Dalam Helix, yang dikhawatirkan:

> *bukan hanya file yang berubah*.

Realitas tempat Adrian berdiri bisa ikut berubah.

Jadi pesan:

> *“Jangan checkout branch Maya”*

sebenarnya berarti:

> *jangan pindah ke sejarah tempat Maya mungkin masih hidup*.

Karena belum diketahui apa konsekuensinya.

Lalu branch `maya` benar-benar muncul

Laptop yang bahkan sudah ditutup tiba-tiba membuka sendiri.

Kemudian muncul:

```
From origin
* [new branch] maya -> origin/maya
```

Ini berarti repository Adrian baru menerima informasi bahwa di remote ada branch bernama:

```
maya
```

atau secara lokal dicatat sebagai:

```
origin/maya
```

Masalahnya:

> *server Helix sudah dimatikan tiga tahun lalu*.

Adrian sendiri yang mencabut koneksi terakhirnya.

Jadi remote itu seharusnya tidak mungkin aktif.

Dan alamat `origin` bukan alamat biasa

Adrian menjalankan:

```
git remote -v
```

Hasilnya:

```
origin qnt://00000000000000000000000000000001
```

Ini bukan internet.

Bukan jaringan kantor.

Bahkan Wi-Fi dimatikan, tetapi koneksi tetap berjalan.

Artinya `origin` tersebut mungkin bukan server biasa.

Ini petunjuk pertama bahwa remote itu bisa berada:

> *di tempat lain,*
>
> *branch lain,*
>
> *realitas lain,*

atau sesuatu yang jauh lebih dalam.

Penting juga: pada titik prolog ini, `origin` belum boleh dianggap sebagai realitas asli. Itu hanya nama remote.

Maya kemudian berbicara

Terminal menampilkan:

```
maya: Adrian, jangan percaya waktu kematian yang sudah kuhapus.
```

Lalu:

```
Itu bukan ramalan.
```

Dan:

```
Itu catatan kejadian sebelumnya.
```

Ini salah satu kalimat paling penting dalam prolog.

Artinya waktu kematian Adrian:

```
03:47:21
```

bukan prediksi masa depan.

Menurut Maya, kejadian itu sudah pernah terjadi sebelumnya.

Dengan kata lain, mungkin ada sejarah lain:

```
versi sejarah sebelumnya
↓
Adrian mati 03:47:21
sejarah sekarang
↓
Maya mengubah commit
↓
death = null
```

Ini langsung membuka kemungkinan bahwa dunia sudah pernah:

di-reset,

di-branch,

di-rewrite,

atau dijalankan lebih dari sekali.

Kemudian ORPHEUS muncul

Terminal membuka:

```
ORPHEUS/1943/DE-173
checkout status: INCOMPLETE
last known remote: BERMUDA-03
```

DE-173 adalah kode yang berhubungan dengan Philadelphia Experiment.

Jadi prolog mulai menghubungkan:

```
Git
+
Philadelphia Experiment
+
Bermuda
+
checkout
+
realitas alternatif
```

Belum dijelaskan lengkap, tetapi pembaca diberi petunjuk bahwa kapal Philadelphia mungkin mengalami semacam checkout realitas yang gagal selesai.

Lalu Maya menyebut cincin

Pesan terakhir:

```
Kalau kamu masih menyimpan cincin kita,
berarti tidak semua tentang kita
berhasil mereka hapus.
```

Ini sangat penting secara emosional.

Artinya Maya tahu cincin yang seharusnya tidak diketahui siapa pun.

Dan kata:

> *“cincin kita”*

menegaskan bahwa hubungan Adrian-Maya memang lebih dalam dari ingatan resmi Adrian.

Kemungkinan besar:

> *sebagian sejarah hubungan mereka pernah dihapus dari Adrian.*

Tetapi tidak sempurna.

Buktinya masih tersisa dalam:

- cincin,
- foto,
- rasa kehilangan,
- ingatan yang tidak cocok.

Dan prolog ditutup dengan pintu terbuka

Sejak awal pintu laboratorium bertuliskan:

```
LOCKED
```

Setelah semua pesan Maya:

```
OPEN
```

Secara literal, pintu laboratorium terbuka.

Tetapi secara simbolis, ini juga berarti:

> *pintu menuju seluruh misteri Helix sekarang terbuka.*

Adrian awalnya hidup dalam dunia yang sederhana:

```
orang hidup
↓
orang mati
↓
selesai
```

Setelah prolog:

```
orang bisa hidup di branch lain
history bisa berbeda
memori bisa berasal dari history lain
kematian bisa berupa commit
realitas bisa punya remote
dan orang mati bisa membuat commit
```

Kalau prolog ini diringkas sangat sederhana:

```
Adrian tertidur di lab
        ↓
laptop menyala sendiri
        ↓
ada commit baru
        ↓
Author: MAYA
        ↓
padahal Maya sudah mati 3 tahun
        ↓
signature Maya VALID
        ↓
commit mengubah:
death Adrian 03:47
menjadi NULL
        ↓
Adrian sadar Helix
mungkin benar-benar bisa
mengubah realitas
        ↓
pesan misterius:
JANGAN CHECKOUT MAYA
        ↓
branch origin/maya muncul
        ↓
server Helix seharusnya mati
        ↓
origin ternyata alamat qnt://...
        ↓
Maya berbicara lewat terminal
        ↓
kematian Adrian bukan prediksi
        ↓
itu kejadian yang
```

Jadi dalam satu kalimat:

Prolog ini menceritakan Adrian yang menerima commit misterius dari Maya, perempuan yang sudah meninggal tiga tahun lalu. Commit itu menghapus waktu kematian Adrian yang seharusnya terjadi satu setengah jam lagi. Setelah signature Maya terbukti valid dan sebuah branch `maya` muncul dari remote misterius yang seharusnya sudah mati, Adrian mulai sadar bahwa Project Helix mungkin benar-benar dapat memperlakukan sejarah dan realitas seperti repository Git, dan bahwa kematian Maya, kematian Adrian, bahkan ingatannya sendiri mungkin bukan sesederhana yang selama ini ia percaya.

Dari sisi Git, prolog mengenalkan fondasi:

```
COMMIT

= snapshot perubahan

GIT LOG

= melihat sejarah commit

HASH

= sidik jari commit

VERIFY-COMMIT

= memeriksa keaslian tanda tangan

GIT SHOW

= melihat isi perubahan commit

DIFF

- = keadaan lama dihapus

+ = keadaan baru ditambahkan

BRANCH

= jalur sejarah alternatif

CHECKOUT

= berpindah ke jalur tersebut

REMOTE

= repository di tempat lain

ORIGIN

= nama konvensional remote,

bukan otomatis "dunia asli"
```

Sedangkan pesan utama prolog adalah: jangan terlalu cepat percaya bahwa satu-satunya sejarah yang kita ingat adalah satu-satunya sejarah yang pernah terjadi. Dalam dunia Helix, sesuatu bisa

dihapus dari catatan resmi tetapi masih meninggalkan jejak dalam benda, memori, hubungan, dan commit yang seharusnya mustahil ada.

BAB 01

INIT: Ketika Realitas Menjadi Repository

Inti bab: Bab ini pada dasarnya menjelaskan awal mula Project Helix dan mengungkap bahwa teknologi yang mereka buat ternyata bukan sekadar komputer canggih. Helix seperti Git, tetapi Git untuk realitas.

Di awal cerita, Adrian berada pada tahun 2041. Ia baru menerima pesan misterius dari Maya yang mengatakan bahwa waktu kematian Maya yang pernah dihapus bukanlah ramalan, tetapi **catatan dari kejadian yang sudah pernah terjadi sebelumnya**. Maya juga menyebut cincin mereka, padahal menurut sejarah yang Adrian ingat, hubungan itu seharusnya tidak pernah terjadi. Dari sini Adrian mulai curiga bahwa sejarah yang ia kenal mungkin bukan satu-satunya sejarah.

Kemudian cerita mundur sebelas tahun, ke tahun 2030, saat Adrian pertama kali bergabung dengan Project Helix. Adrian adalah engineer yang ahli membuat sistem komputer terdistribusi. Ia direkrut oleh QuantumForge dan dibawa ke laboratorium rahasia enam lantai di bawah tanah. Di sana ia bertemu Profesor Maya Pradipta. Anehnya, meskipun mereka seolah baru pertama kali bertemu, Adrian merasa Maya sangat familiar baginya. Bahkan narasi memberi petunjuk bahwa mungkin ini bukan benar-benar pertemuan pertama mereka.

Maya kemudian memperlihatkan sebuah mesin raksasa kepada Adrian. Ketika Adrian bertanya mesin itu apa, Maya menjawab sederhana: **"Ini Git."** Maksudnya bukan Git biasa untuk menyimpan source code, tetapi sebuah sistem yang memperlakukan **keadaan dunia seperti source code yang memiliki version history**.

Cara termudah membayangkannya begini.

Git pada programming bisa menyimpan keadaan sebuah project. Misalnya hari ini aplikasi memiliki versi A. Besok programmer mengubahnya menjadi versi B. Git menyimpan hubungan antara versi A dan B sehingga programmer bisa melihat sejarah perubahan atau kembali ke versi sebelumnya.

Helix melakukan konsep yang sama, tetapi yang disimpan bukan file PHP, Java, atau source code.

Yang disimpan adalah **realitas**.

Contohnya, Maya menunjukkan keadaan Jakarta pada beberapa waktu berbeda sebagai commit. Mesin bahkan dapat menampilkan miniatur Jakarta yang sangat detail, termasuk kendaraan, hujan, gedung, dan manusia. Maya mengatakan itu bukan rekaman kamera atau simulasi. Itu adalah **snapshot keadaan Jakarta pada suatu saat tertentu**.

Untuk menjelaskan konsep tersebut, Maya melakukan eksperimen dalam sebuah kotak berisi lampu, tanaman, air, jam, dan seekor tikus.

Perintah:

```
reality init chamber-01
```

artinya kira-kira:

"Mulai catat sejarah realitas di ruangan ini."

Sama seperti `git init` yang membuat sebuah folder menjadi repository Git. Setelah itu mereka membuat commit pertama yang disebut **Genesis Commit**, yaitu snapshot awal keadaan seluruh isi ruangan.

Mereka kemudian mengubah keadaan ruangan dan membuat commit baru. Misalnya lampu dinyalakan atau mangkuk air dipindahkan.

Sehingga sejarahnya seperti:

```
G0 → A1 → A2
```

Artinya keadaan A1 berasal dari G0, kemudian A2 berasal dari A1.

Jadi Helix dapat menyimpan **sejarah perubahan sebuah ruang fisik** seperti Git menyimpan sejarah perubahan software.

Masalah mulai menjadi jauh lebih aneh ketika mereka menggunakan konsep **branch**.

Dalam Git, branch memungkinkan programmer membuat dua jalur pengembangan berbeda dari satu titik yang sama.

Misalnya:

```
        experiment
       /
A0 → A1
       \
        main
```

Dalam Helix ternyata hal tersebut juga bisa dilakukan terhadap realitas.

Dari keadaan awal tikus yang sama, mereka membuat dua branch. Pada satu branch lampu dimatikan dan tikus bergerak ke kanan. Pada branch lain lampu menyala dan tikus bergerak ke kiri.

Yang membuat Adrian bingung adalah hanya ada **satu tikus fisik**.

Jadi kalau dua sejarah benar-benar berjalan secara bersamaan, pertanyaannya menjadi:

Di mana realitas yang satunya?

Bahkan Maya sendiri menjawab bahwa mereka belum tahu.

Lalu Kevin melakukan kesalahan besar.

Ia mencoba menjalankan:

```
reality merge experiment-a
```

Dalam Git biasa, `merge` berarti menggabungkan dua branch.

Kalau perubahan kedua branch tidak bertabrakan, Git bisa menggabungkannya otomatis. Tetapi kalau dua programmer mengubah bagian yang sama dengan hasil berbeda, Git akan menghasilkan **merge conflict**.

Pada Helix, yang mengalami conflict bukan file.

Yang conflict adalah **tubuh makhluk hidup**.

Branch pertama berkata:

> *tikus berada di kanan*.

Branch kedua berkata:

> *tikus berada di kiri*.

Mesin mencoba mempertahankan kedua kondisi tersebut sekaligus.

Hasilnya mengerikan.

Tubuh tikus mulai berubah, seperti dua versi realitas sedang dipaksakan menjadi satu. Setelah sistem dimatikan, tikus tersebut mati dengan **dua kepala**, satu menghadap kanan dan satu menghadap kiri.

Bagian ini penting karena menunjukkan bahwa Helix bukan sekadar alat yang **merekam realitas**.

Helix kemungkinan bisa **mengubah realitas**.

Ini mengubah arti seluruh Project Helix.

Kemudian cerita kembali ke tahun 2041.

Adrian sadar ada sesuatu yang salah dengan ingatannya. Selama sebelas tahun ia selalu mengingat insiden tikus itu terjadi pada hari ke-19 setelah ia bergabung. Tetapi ingatan yang baru muncul mengatakan bahwa kejadian tersebut terjadi **pada hari pertama**.

Artinya kemungkinan ada dua sejarah berbeda.

Di sejarah A, kejadian terjadi hari pertama.

Di sejarah B, kejadian terjadi hari ke-19.

Dan Adrian mungkin memiliki sisa ingatan dari keduanya.

Hal ini juga menjelaskan ide **Mandela Effect** dalam dunia novel.

Dalam kehidupan nyata, Mandela Effect biasanya dianggap sebagai ingatan kolektif yang salah. Tetapi teori Project Helix mengatakan sesuatu yang lebih ekstrem:

Mungkin sebagian orang sebenarnya mengingat **realitas dari branch lama**.

Setelah sejarah diubah, sebagian kecil ingatan manusia tidak terhapus sempurna.

Seperti ada sisa file setelah proses merge atau rebase.

Bab ini juga mulai menghubungkan Helix dengan **Philadelphia Experiment dan Bermuda Triangle**.

Project ORPHEUS menemukan dugaan bahwa kejadian Philadelphia Experiment tahun 1943 mungkin merupakan percobaan primitif terhadap mekanisme serupa Helix. Mereka tidak sengaja membuat kapal menjadi seperti kondisi **Detached HEAD dalam Git**, yaitu berada pada keadaan yang tidak terhubung secara normal dengan branch sejarah utama.

Sedangkan beberapa titik di Bermuda diduga berfungsi seperti **remote repository** realitas.

Artinya teori besar novel mulai terbentuk.

Bayangkan dunia seperti sebuah project software:

```
REALITAS

Repository
│
├─ commit
│  keadaan dunia pada waktu tertentu
│
├─ branch
│  sejarah alternatif
│
├─ checkout
│  berpindah ke keadaan tertentu
│
├─ merge
│  menggabungkan dua sejarah
│
├─ conflict
│  dua realitas tidak cocok
│
├─ remote
│  repository realitas lain
│
└─ memory leak
   manusia mengingat branch lama
```

Tetapi twist terbesar muncul di akhir bab.

Adrian menemukan folder tersembunyi bernama:

```
.helix
```

Sama seperti repository Git memiliki folder tersembunyi `.git`.

Folder `.helix` dapat dianggap sebagai **tempat penyimpanan ingatan realitas**. Masalahnya, folder tersebut seharusnya sudah dihapus tiga tahun sebelumnya.

Namun tiba-tiba muncul lagi pada tahun 2041.

Di dalamnya ada permintaan:

```
Target: Jakarta
Requested by: maya

Awaiting maintainer approval.

Upstream candidate: BERMUDA-03
Recovery protocol: ORPHEUS
```

Artinya seseorang, kemungkinan Maya, meminta sistem untuk melakukan **`init` terhadap Jakarta**.

Dengan bahasa sederhana:

> *"Jadikan Jakarta sebuah repository realitas."*

Dan yang lebih mengerikan, komputer tidak menunggu Adrian menyetujuinya.

Kursor bergerak sendiri.

Tombol `INITIALIZE` ditekan sendiri.

Seluruh lampu Jakarta padam bersamaan.

Lalu muncul satu informasi terakhir:

```
Recovered memory: adrian/maya/philadelphia

Status: differs from current history
```

Artinya sistem menemukan **sebuah ingatan Adrian dan Maya yang berhubungan dengan Philadelphia**, tetapi ingatan itu tidak cocok dengan sejarah dunia yang sedang berlaku sekarang.

Jadi misteri utama setelah bab ini menjadi jauh lebih besar: **Adrian dan Maya mungkin pernah mengalami kehidupan lain bersama dalam branch sejarah yang berbeda. Maya mungkin sudah mengetahui bahwa realitas pernah diubah. Philadelphia Experiment mungkin bukan sekadar kejadian sejarah, melainkan bagian dari repository realitas yang lebih tua. Dan sekarang seseorang sedang melakukan `git init` terhadap seluruh Jakarta.**

Kalau disederhanakan menjadi satu kalimat, **Bab 1 adalah cerita tentang Adrian yang menemukan bahwa dunia ternyata memiliki version control seperti Git, dan seseorang baru saja mulai membuat repository baru untuk realitas Jakarta.**

BAB 02

STATUS: Ketika Adrian Menjadi Untracked

Inti bab: Bab 2 ini intinya menjelaskan apa yang terjadi setelah Jakarta resmi dijadikan “repository realitas” oleh Helix.

Kalau Bab 1 memperkenalkan ide bahwa dunia bisa bekerja seperti Git, maka Bab 2 mulai menunjukkan konsekuensinya.

Setelah seluruh lampu Jakarta padam tiga detik, Helix menyatakan:

```
Initialized empty reality repository in qnt://local/jakarta
```

Artinya, secara sederhana: **Jakarta sekarang dianggap sebagai sebuah project yang bisa dipantau, dibandingkan, dicatat perubahannya, dan mungkin nantinya disimpan sebagai sejarah baru.**

Masalahnya, Adrian tidak pernah menyetujui proses itu. Ia mencoba membatalkan, tetapi sistem mengatakan proses inisialisasi sudah selesai. Bahkan Helix mulai membaca keadaan Jakarta secara langsung, bukan mengunduh data dari tempat lain.

Cara paling mudah membayangkannya seperti ini.

Sebelumnya Jakarta hanyalah kota biasa.

Setelah `reality init`, Helix seperti berkata:

> *"Mulai sekarang saya akan menganggap seluruh Jakarta sebagai folder kerja saya."*

Dan dalam Git, folder kerja aktif disebut **working tree**.

Jadi dalam dunia novel ini:

Jakarta = working tree.

Artinya semua yang ada di dalam Jakarta, seperti bangunan, pintu, kamera, kendaraan, bahkan manusia, sekarang dapat dianggap sebagai bagian dari sesuatu yang bisa berubah.

Adrian kemudian menjalankan perintah:

```
reality status
```

Ini sama seperti perintah `git status`.

Fungsi perintah ini sebenarnya sederhana. Ia tidak mengubah apa pun. Ia hanya bertanya:

“Sekarang keadaan project kita seperti apa?”

Hasilnya sangat aneh.

Helix menemukan pintu laboratorium berubah, kamera berubah, lalu menemukan dua hal yang berstatus **untracked**:

```
population/adrian-wiratama
```

dan

```
memory/adrian/maya/philadelphia
```

Nah, kata **untracked** sangat penting di bab ini.

Dalam Git, misalnya Anda punya project dengan 10 file. Kemudian tiba-tiba ada file baru bernama `rahasia.txt`.

Git akan mengatakan:

```
untracked file: rahasia.txt
```

Artinya:

> *"File ini ada, tetapi belum pernah menjadi bagian dari sejarah project."*

Hal anehnya, Helix mengatakan hal yang sama tentang **Adrian**.

Adrian secara fisik berada di Jakarta.

Tetapi repository Jakarta mengatakan:

> *Adrian ada di sini, tetapi Adrian bukan bagian dari sejarah Jakarta yang saya kenal.*

Ini petunjuk besar.

Kemungkinan Adrian yang sekarang bukan sepenuhnya berasal dari branch sejarah yang sedang aktif.

Tubuhnya ada.

DNA-nya cocok.

Wajahnya cocok.

Tetapi secara “sejarah”, sistem tidak mengenalinya.

Analogi sederhananya seperti Anda membuka album keluarga tahun 2041, lalu menemukan seseorang berdiri di ruang tamu.

Orangnya nyata.

Tetapi orang tersebut tidak ada di satu pun foto keluarga sebelumnya.

Helix tidak mengatakan orang itu palsu.

Helix hanya mengatakan:

“Saya tidak punya riwayat tentang orang ini.”

Hal yang sama terjadi pada ingatan Adrian tentang Maya di Philadelphia.

Helix menemukan sebuah kenangan yang menunjukkan Adrian dan Maya pernah pergi ke Philadelphia bersama. Mereka bahkan memakai cincin.

Dalam ingatan itu Maya mengatakan:

> *“Cari aku di branch tempat kamu masih ingat mencintaiku.”*

Masalahnya, dalam sejarah yang Adrian ingat sekarang, hubungan itu tidak pernah terjadi.

Jadi kemungkinan besar ada sejarah lain seperti:

```
Branch A

Adrian + Maya hanya rekan kerja


Branch B

Adrian + Maya saling mencintai
Pergi ke Philadelphia
Memakai cincin
```

Entah bagaimana, sebagian ingatan dari Branch B masuk ke Adrian yang sekarang hidup di Branch A.

Lalu muncul entitas kedua:

```
population/unknown-01
```

Helix mengatakan seseorang tidak dikenal sedang bergerak hanya beberapa puluh meter dari Adrian.

Ternyata orang itu adalah **Nara Santoso**.

Nara masuk melalui koridor servis sambil membawa pistol.

Yang menarik, Nara langsung curiga apakah Adrian benar-benar Adrian.

Ia memeriksa wajah.

Cocok 99,9981%.

Sidik jari.

Cocok 100%.

DNA.

Cocok hampir 100%.

Tetapi Nara masih mengatakan:

> *Itu hanya membuktikan tubuhmu adalah tubuh Adrian. Belum tentu kamu Adrian yang saya kenal.*

Ini salah satu konsep penting cerita.

Dalam dunia biasa kita berpikir:

DNA sama = orang yang sama.

Tetapi dalam dunia Helix, bisa jadi ada beberapa versi Adrian dari beberapa branch sejarah.

Misalnya:

```
Adrian-A
DNA sama
hidup di sejarah A

Adrian-B
DNA sama
hidup di sejarah B

Adrian-C
DNA sama
mengalami sejarah C
```

Tubuh mereka mungkin identik.

Tetapi pengalaman dan sejarah mereka berbeda.

Itulah mengapa Nara tidak cukup puas dengan biometrik.

Kemudian Nara mengungkap hal penting.

Ternyata Maya sudah mempersiapkan semua kejadian ini **tiga tahun sebelumnya**.

Ia membuat sebuah **dead-man switch**.

Dead-man switch sederhananya seperti surat otomatis.

Misalnya seseorang berkata:

> *"Kalau selama beberapa hari saya tidak memberi kabar, kirim pesan ini kepada orang tertentu."*

Maya membuat versi yang jauh lebih canggih.

Setelah Maya meninggal, perangkat itu menunggu suatu commit tertentu muncul. Ketika commit dengan hash tertentu muncul malam ini, pesan Maya otomatis dibuka untuk Nara.

Pesan Maya sangat penting:

> *Jangan biarkan siapa pun memasukkan Adrian ke staging area*.

Lalu Maya mengatakan hal yang lebih mengerikan.

Kalau ingatan Philadelphia muncul, Nara tidak boleh membiarkan Adrian mengejar Maya hanya karena ia kembali mengingat bahwa ia pernah mencintainya.

Alasannya:

"Itulah cara semua percobaan sebelumnya berakhir."

Kalimat ini memberi petunjuk bahwa kejadian yang mereka alami mungkin **sudah pernah terjadi berkali-kali**.

Mungkin pernah ada Adrian dari branch sebelumnya.

Ia menemukan ingatan Maya.

Ia mencoba mencari atau menyelamatkan Maya.

Kemudian sesuatu buruk terjadi.

Lalu sejarah berubah lagi.

Dan siklusnya dimulai kembali.

Setelah itu Bab 2 memperkenalkan konsep Git berikutnya:

Staging area

Bayangkan Git seperti memasak.

Ada tiga tahap.

```
MEJA DAPUR
working tree

↓ pilih bahan

NAMPAK / NAMPAN
staging area

↓ simpan

FOTO HASIL
commit
```

Working tree berarti semua perubahan yang sedang terjadi.

Staging area berarti perubahan yang sudah **dipilih untuk dimasukkan ke commit berikutnya**.

Commit berarti keadaan yang akhirnya resmi disimpan ke sejarah.

Adrian menjelaskannya kepada Nara dengan analogi dapur: benda di meja belum tentu akan ikut masuk ke hidangan, tetapi benda yang sudah ditempatkan di nampan sedang dipersiapkan untuk tahap berikutnya.

Karena itu pesan Maya:

“Jangan biarkan Adrian masuk staging area.”

sebenarnya sangat menakutkan.

Kalau manusia bisa dimasukkan ke staging area, berarti manusia bisa dijadikan bagian dari perubahan yang akan disimpan ke sejarah berikutnya.

Kemudian Adrian dan Nara memeriksa perubahan pada pintu laboratorium.

Helix menunjukkan:

```
- state: locked
+ state: open
+ last_access: maya-pradipta
```

Artinya Maya tercatat membuka pintu.

Padahal Maya sudah meninggal.

Adrian menjelaskan kemungkinan Helix tidak perlu benar-benar memakai kartu akses.

Helix cukup **mengubah keadaan realitas**.

Bukan:

> *“Saya membuka pintunya.”*

Tetapi:

> *“Saya mengubah realitas sehingga pintu sekarang tercatat sudah terbuka.”*

Ini menunjukkan kekuatan Helix yang jauh lebih berbahaya.

Helix mungkin tidak mengendalikan benda dengan cara mekanis.

Ia bisa **mengedit kondisi benda itu sendiri**.

Mereka kemudian menemukan tujuh detik rekaman kamera hilang.

Tepat sebelum bagian hilang, Adrian terlihat tidur.

Setelah tujuh detik itu, posisi kepalanya sedikit berubah.

Dan ada **bayangan seseorang berdiri di dekat sofa**.

Tetapi tidak ada orang yang terlihat.

Setelah diselidiki, mereka menemukan sebuah objek tersembunyi.

Namanya:

```
adrian-wiratama
```

Timestamp-nya sama persis dengan waktu Helix aktif.

Nara kemudian mengatakan:

“Bayangan itu kamu.”

Di sinilah misterinya semakin besar.

Ada kemungkinan pada saat Adrian sedang tidur, **versi Adrian lain pernah berada di ruangan yang sama**.

Sehingga pertanyaannya menjadi:

> *Adrian yang sekarang bangun di sofa sebenarnya Adrian yang mana?*

Bahkan Adrian sendiri tidak tahu.

Kemudian datang ancaman baru.

Seseorang menghubungi laboratorium mengaku sebagai keamanan pusat.

Tetapi Nara mengetahui sambungan itu bukan jalur keamanan resmi.

Suara itu mengatakan mereka akan mengirim **tim pemulihan**.

Kemudian `reality status` mendeteksi empat orang baru:

```
population/recovery-team-01
population/recovery-team-02
population/recovery-team-03
population/recovery-team-04
```

Mereka sedang menuju laboratorium.

Nara percaya tujuan mereka adalah memasukkan Adrian ke staging area.

Tetapi kita belum tahu mengapa.

Lalu muncul twist terbesar bab ini.

Helix tiba-tiba menampilkan:

```
Changes to be committed:

new file: death/adrian-wiratama
```

Artinya **kematian Adrian sudah masuk staging area.**

Dengan kata lain:

Kematian Adrian belum terjadi.

Tetapi seseorang sudah memilih kejadian itu untuk dimasukkan ke sejarah berikutnya.

Ketika Adrian memeriksa detailnya, ia menemukan:

```
name: Adrian Wiratama
time: 03:47:21
location: Repository Chamber
cause: merge failure
witnessed_by: Nara Santoso
```

Jadi sistem mengatakan:

Adrian akan meninggal pukul 03:47 karena kegagalan merge, dan Nara akan menyaksikannya.

Tetapi ini bukan sekadar ramalan.

Karena statusnya sudah berada dalam:

```
Changes to be committed
```

Artinya kejadian tersebut sudah **disiapkan untuk menjadi sejarah resmi**.

Kemudian sistem memberi countdown:

```
Commit scheduled in 01:07:42.
```

Jadi mereka punya sekitar satu jam untuk mencegahnya.

Yang membuat situasinya lebih seram adalah pesan Maya dari bab sebelumnya:

“Jangan percaya waktu kematian yang sudah kuhapus. Itu bukan ramalan. Itu catatan kejadian sebelumnya.”

Sekarang kita mulai mengerti maksudnya.

Kemungkinan Adrian **memang pernah meninggal pukul 03:47 dalam branch sebelumnya**.

Lalu Maya melakukan sesuatu terhadap sejarah sehingga kematian itu dihapus.

Tetapi setelah repository Jakarta diinisialisasi kembali, data kematian tersebut muncul lagi dan masuk staging area.

Jadi Bab 2 sebenarnya bukan cerita tentang Adrian mencoba menghindari masa depan.

Ia mungkin sedang mencoba menghindari **masa lalu yang ingin terjadi kembali.**

Kalau digambarkan sederhana:

```
SEJARAH LAMA

Adrian hidup
   ↓
03:47
   ↓
Adrian mati akibat merge failure
   ↓
Maya mengubah sejarah
   ↓
kematian Adrian hilang


SEJARAH SEKARANG

Adrian hidup lagi
   ↓
Helix aktif
   ↓
data sejarah lama ditemukan
   ↓
death/adrian masuk staging
   ↓
commit dijadwalkan
   ↓
03:47 ???
```

Jadi ancaman utamanya bukan sekadar orang bersenjata yang datang ke laboratorium.

Ancaman sebenarnya adalah **Helix mencoba mengembalikan sebuah kejadian dari sejarah lama menjadi bagian dari sejarah sekarang.**

Dan ada satu pertanyaan yang semakin penting:

Jika kematian Adrian berasal dari sejarah lain, mengapa Adrian yang sekarang ikut dianggap sebagai orang yang harus mati?

Apalagi status Helix sendiri mengatakan Adrian saat ini adalah **untracked**.

Dengan kata lain, ada kemungkinan yang lebih aneh lagi:

kematian yang sudah masuk staging mungkin sebenarnya milik Adrian dari branch lain, bukan Adrian yang sekarang.

Itulah cliffhanger utama Bab 2.

BAB 03

ADD: Memilih Perubahan untuk Menjadi Sejarah

Inti bab: Bab 3 ini menjelaskan konsep Git `add` dan staging area, tetapi diterapkan pada realitas. Kalau Bab 2 menunjukkan bahwa kematian Adrian sudah masuk staging, Bab 3 menunjukkan bahwa memilih sesuatu untuk masuk staging ternyata bisa ikut memengaruhi dunia nyata.

Di awal bab, Adrian dan Nara sedang dikejar empat orang dari tim pemulihan. Pada saat yang sama, ada hitung mundur sekitar satu jam menuju sebuah commit yang sudah dijadwalkan. Masalah terbesarnya adalah file `death/adrian-wiratama` sudah berada di staging area. Artinya, kematian Adrian sudah dipilih untuk masuk ke commit berikutnya.

Cara mudah memahami situasinya seperti ini:

```
REALITAS SEKARANG
      ↓
working tree
semua hal yang sedang terjadi

      ↓ reality add

staging area
hal-hal yang DIPILIH
untuk disimpan

      ↓ commit

SEJARAH RESMI
```

Jadi Adrian saat ini sebenarnya berada dalam keadaan sangat aneh.

Tubuh Adrian sendiri masih **untracked**, artinya repository Jakarta belum menganggapnya bagian dari sejarah yang resmi.

Kenangannya tentang Maya di Philadelphia juga masih untracked.

Tetapi **kematiannya justru sudah masuk staging**.

Sederhananya:

> *Helix belum yakin Adrian termasuk bagian dari dunia ini, tetapi Helix sudah mempersiapkan kematiannya untuk menjadi bagian sejarah.*

Itulah keanehan besar di awal Bab 3.

Kemudian muncul penjelasan penting tentang perintah:

```
reality add .
```

Dalam Git biasa, banyak orang mengira `git add` berarti mengirim file ke server. Sebenarnya bukan.

`add` berarti:

> ***"Pilih perubahan ini untuk dimasukkan ke commit berikutnya."***

Tanda titik pada `add .` berarti **pilih semuanya**.

Kalau menggunakan analogi meja kerja, bayangkan ada banyak barang di meja:

```
MEJA

surat penting
obat
sampah
kunci rumah
dokumen
```

Kemudian Anda menjalankan:

```
add .
```

Artinya seperti menyapu **semua barang itu** ke dalam sebuah kotak bertuliskan:

> *"Siap disimpan."*

Itulah sebabnya Adrian takut menjalankan `reality add .`.

Karena kalau dilakukan, bukan hanya rencana pelarian yang mungkin dipilih. Tubuhnya sendiri dan kenangan misteriusnya tentang Maya juga bisa ikut masuk staging.

Adrian kemudian mencoba mengeluarkan catatan kematiannya dari staging menggunakan:

```
reality restore --staged death/adrian-wiratama
```

Dalam Git biasa, artinya kira-kira:

> *"Keluarkan file ini dari daftar yang akan di-commit."*

Bukan menghapus file, hanya membatalkan pemilihannya.

Tetapi Helix menolak.

```
error:
path 'death/adrian-wiratama'
is locked by another process
```

Artinya ada **pihak atau proses lain yang sedang mengunci kematian Adrian**.

Adrian tidak punya kendali penuh atas staging area itu.

Pada saat bersamaan, tim pemulihan berhasil hampir masuk ke laboratorium. Nara dan Adrian akhirnya kabur melalui koridor servis.

Dalam pelarian, Helix mulai menawarkan dua jalur keluar.

Ketika Adrian dan Nara memilih sebuah jalur, Helix mencatat pilihan tersebut sebagai:

```
modified: plans/escape-route
```

Ini penting.

Helix ternyata bukan cuma membaca benda seperti pintu atau manusia.

Keputusan dan rencana manusia juga dapat menjadi bagian dari working tree.

Adrian kemudian mendapat ide.

Ia ingin memilih bagian tertentu dari rencana pelarian untuk dimasukkan ke staging.

Bukan semuanya.

Ia menggunakan:

```
reality add --patch plans/escape-route
```

Ini merupakan konsep Git yang nyata juga. `git add --patch` memungkinkan kita memilih hanya sebagian perubahan.

Misalnya Anda mengubah satu file dalam dua bagian:

```
Bagian A
memperbaiki typo

Bagian B
menghapus satu fitur besar
```

Anda bisa memilih hanya Bagian A untuk masuk staging.

Adrian melakukan hal yang sama terhadap realitas.

Ia memilih satu per satu:

```
Adrian dan Nara masuk koridor servis

pilih jalur cooling-sector-b

tujuan freight elevator

tutup pintu sekat setelah mereka lewat
```

Pada awalnya kelihatannya berhasil.

Pintu terbuka.

Adrian dan Nara berhasil lewat.

Kemudian pintu baja menutup di belakang mereka sehingga tim pengejar terhalang.

Masalahnya, salah satu anggota tim masih berada di bawah pintu.

Lengannya terjepit.

Dan pintu **terus menekan**.

Adrian panik karena ia tidak bermaksud melukai orang itu.

Tetapi perubahan yang tadi ia pilih hanya mengatakan:

```
close bulkhead after passage
```

Tidak ada instruksi:

> *Jangan tutup kalau ada manusia di bawah pintu.*

Tidak ada instruksi:

> *Gunakan sensor keselamatan.*

Tidak ada instruksi:

> *Tutup perlahan.*

Helix hanya menjalankan hasil yang dipilih Adrian.

Ini salah satu pesan penting bab tersebut:

> ***Sistem tidak membaca niat manusia. Sistem membaca instruksi.***

Adrian bermaksud:

> *"Tutup pintunya supaya kita selamat."*

Tetapi Helix hanya melihat:

> *"Tutup pintu."*

Ini mirip masalah dalam pemrograman maupun AI.

Manusia sering berpikir:

> *"Komputer pasti tahu maksud saya."*

Tidak.

Komputer hanya mengikuti kondisi yang diberikan.

Jadi walaupun Adrian tidak bermaksud mencederai petugas tersebut, hasil dunia nyata tetap terjadi.

Akhirnya mereka berhasil membantu melepaskan tangannya, tetapi orang tersebut sudah terluka. Dan Helix mencatat:

```
Changes not staged for commit:

modified:
population/recovery-team-01
```

Artinya cedera orang itu sekarang menjadi perubahan dalam working tree.

Di sini Adrian memahami sesuatu yang sangat penting:

staging bukan dunia khayalan.

Perubahan harus terjadi terlebih dahulu di working tree sebelum bisa dipilih ke staging.

Misalnya Nara membalut tangan Adrian.

Keadaan dunia berubah:

```
sebelum:
tangan terluka

sesudah:
tangan terluka + perban
```

Kemudian `add` hanya mengatakan:

> *“Saya memilih keadaan dengan perban ini untuk commit berikutnya.”*

Jadi `add` bukan menciptakan perubahan.

`add` memilih **snapshot dari perubahan yang sudah terjadi**.

Nah, dari sini Adrian mendapatkan kesimpulan yang sangat menyeramkan.

Kalau begitu bagaimana dengan file:

```
death/adrian-wiratama
```

Kalau kematian Adrian sudah berada di staging, berarti **kematian tersebut pasti pernah berada di working tree terlebih dahulu**.

Dengan kata lain:

Adrian memang pernah mati.

Mungkin bukan di sejarah sekarang, tetapi di **branch lain**.

Inilah maksud pesan Maya sebelumnya:

> *Itu bukan ramalan. Itu catatan kejadian sebelumnya.*

Adrian kemudian menyimpulkan kemungkinan seseorang mengambil kematiannya dari branch lain dan memasukkannya ke staging branch yang sekarang.

Tujuannya mungkin supaya kematian yang sama **di-commit lagi**.

Bayangkan seperti ini:

```
BRANCH LAMA

Adrian hidup
   ↓
03:47
   ↓
Adrian meninggal
   ↓
kematian tersimpan


          kemudian


BRANCH SEKARANG

Adrian hidup
   ↓
seseorang mengambil
data kematian lama
   ↓
masukkan ke staging
   ↓
commit
   ↓
Adrian mati lagi?
```

Jadi mungkin ada seseorang yang ingin memastikan:

> *Tidak peduli sejarah berubah sebanyak apa pun, Adrian tetap harus berakhir mati pada pukul 03:47*.

Kemudian mereka sampai di sebuah pintu yang hanya bisa dibuka oleh **tracked entity**.

Masalahnya Adrian masih untracked.

Artinya secara fisik Adrian ada di sana, tetapi Helix masih menganggapnya seperti file asing.

Nara berkata bahwa Adrian bisa memasukkan dirinya sendiri menggunakan `add`.

Tetapi Maya sudah memperingatkan:

> *Jangan masukkan Adrian ke staging*.

Adrian akhirnya mencari jalan tengah.

Helix ternyata memungkinkan dirinya memilih hanya bagian tertentu:

```
body
identity
memory
history
```

Adrian tidak memilih semuanya.

Ia hanya memilih:

```
identity
```

Artinya kira-kira:

> *"Masukkan identitas saya saja ke staging. Jangan tubuh, memori, atau seluruh sejarah saya."*

Begitu ia melakukannya, Adrian mengalami efek fisik yang aneh.

Ia hampir lupa namanya selama beberapa detik.

Nara bahkan mengetesnya:

Siapa namamu?

Tanggal lahir?

Kapan pertama kali kita bertemu?

Adrian masih bisa menjawab.

Kemudian Helix mengatakan:

```
Identity added to staging area.

Entity remains untracked until commit.
```

Jadi identitas Adrian sudah dipilih, tetapi **belum resmi menjadi bagian sejarah** karena belum ada commit.

Analogi sederhananya seperti mendaftar menjadi anggota sebuah organisasi.

Anda sudah menyerahkan formulir.

Formulir ada di meja petugas.

Tetapi keanggotaan belum disahkan.

Jadi secara resmi Anda belum menjadi anggota.

Kemudian muncul twist paling besar di akhir Bab 3.

Adrian menjalankan `reality status` untuk memastikan apa saja yang berada di staging.

Seharusnya ada tiga:

```
plans/escape-route

population/adrian-wiratama/identity

death/adrian-wiratama
```

Tetapi ternyata ada **empat**.

Muncul file baru:

```
memory/adrian-wiratama/maya-incident
```

Adrian tidak pernah memasukkan memori tersebut.

Artinya ada pihak lain yang diam-diam bisa menambahkan sesuatu ke staging.

Ini sangat berbahaya.

Kalau staging diibaratkan keranjang belanja, berarti:

> *Adrian sedang memilih barang sendiri, tetapi ada seseorang yang diam-diam memasukkan barang lain ke keranjangnya.*

Kemudian Adrian membuka memori tersebut.

Ia melihat sebuah kejadian pada **17 Mei 2038**.

Maya terbaring di lantai Repository Chamber.

Kepalanya berdarah.

Adrian melihat kejadian tersebut dari sudut pandangnya sendiri.

Dan yang paling mengerikan:

Tangannya memegang batang logam berlumuran darah.

Maya masih hidup.

Kemudian Maya menatap Adrian dan mengatakan:

“Commit ini sekarang.”

Adrian langsung mengatakan:

> *“Aku tidak pernah melihat ini.”*

Artinya Adrian tidak memiliki ingatan tentang kejadian tersebut.

Tetapi Helix kemudian mengatakan:

```
Staged memory integrity: VERIFIED

Source: adrian-wiratama
```

Artinya menurut Helix:

memori itu asli dan berasal dari Adrian sendiri.

Tetapi Adrian sama sekali tidak mengingatnya.

Ini membuka beberapa kemungkinan besar.

Mungkin Adrian pernah membunuh atau melukai Maya di branch lain.

Mungkin Maya memang sengaja meminta Adrian melakukan sesuatu kepadanya.

Mungkin adegan tersebut bukan pembunuhan, tetapi bagian dari sebuah eksperimen.

Atau mungkin ada **Adrian versi lain** yang melakukan kejadian tersebut.

Yang jelas, Helix percaya kejadian itu benar-benar merupakan bagian dari ingatan Adrian.

Jadi kalau disederhanakan, Bab 3 sebenarnya punya tiga penemuan besar.

Pertama, **`add` tidak menciptakan kejadian. Ia memilih kejadian yang sudah terjadi untuk dimasukkan ke sejarah.**

Kedua, karena kematian Adrian sudah berada di staging, berarti **Adrian kemungkinan pernah benar-benar mati di branch lain**.

Ketiga, ada seseorang atau sesuatu yang dapat **memasukkan perubahan ke staging tanpa persetujuan Adrian**, termasuk sebuah memori yang menunjukkan Adrian berdiri di depan Maya yang terluka parah.

Dan kalimat Maya:

> ***"Commit ini sekarang."***

menimbulkan pertanyaan baru yang jauh lebih besar.

Mungkin Maya bukan sedang menjadi korban Adrian.

Mungkin Maya sendiri yang meminta kejadian tersebut disimpan ke sejarah.

Dengan kata lain, misteri Bab 3 bukan lagi hanya **"siapa yang ingin membunuh Adrian?"**

Sekarang pertanyaannya menjadi:

Apa sebenarnya yang dilakukan Adrian dan Maya pada tahun 2038, mengapa ingatan Adrian tentang kejadian itu hilang, dan mengapa Maya ingin kejadian tersebut di-commit?

Benang Merah Bab 00 sampai Bab 03

Empat bab awal ini secara bertahap mengubah Git dari alat version control software menjadi bahasa untuk memahami realitas. Bab 00 memulai misteri melalui commit dari Maya yang sudah meninggal dan memperkenalkan kemungkinan bahwa sejarah memiliki branch lain. Bab 01 memperlihatkan fondasi Helix melalui reality init, snapshot, branch, merge, dan conflict. Bab 02 menunjukkan bahwa setelah Jakarta menjadi repository, manusia dan ingatan dapat berstatus tracked atau untracked. Bab 03 lalu membawa konsep itu lebih jauh: perubahan realitas dapat dipilih ke staging area, bahkan identitas, rencana, memori, dan kematian.

Dari sisi cerita, pertanyaan utamanya semakin tajam. Adrian bukan hanya mencoba memahami apakah Maya masih hidup di branch lain. Ia juga harus menentukan apakah dirinya sendiri berasal dari sejarah yang sedang aktif, mengapa kematiannya sudah disiapkan untuk commit, siapa yang dapat memanipulasi staging area, dan mengapa sebuah memori yang tidak ia ingat menunjukkan Maya meminta Adrian untuk meng-commit sebuah kejadian pada tahun 2038.

Dari sisi konsep Git, urutannya juga sengaja dibuat seperti proses belajar: melihat sejarah melalui commit dan log, membuat repository melalui init, memeriksa keadaan melalui status, lalu memilih perubahan melalui add. Dengan struktur itu, mekanisme Git bukan sekadar tempelan teknis, tetapi menjadi aturan dasar dunia Project Helix.

Urutan konsep Git yang sudah diperkenalkan

- **`COMMIT:`** snapshot atau keadaan yang resmi disimpan dalam sejarah
- **`LOG:`** melihat urutan sejarah commit
- **`BRANCH:`** jalur sejarah alternatif
- **`CHECKOUT:`** berpindah ke jalur atau keadaan tertentu
- **`REMOTE / ORIGIN:`** repository yang berada di tempat lain, bukan otomatis realitas asli
- **`INIT:`** memulai repository dan mulai mencatat sebuah ruang sebagai realitas yang dapat diberi version history
- **`MERGE / CONFLICT:`** menggabungkan dua jalur sejarah dan menghadapi kondisi yang tidak kompatibel
- **`STATUS:`** membaca keadaan working tree dan melihat apa yang berubah, staged, atau untracked
- **`STAGING AREA:`** tempat perubahan dipilih sebelum menjadi bagian dari commit
- **`ADD:`** memilih perubahan untuk commit berikutnya
- **`ADD --PATCH:`** memilih hanya sebagian perubahan
- **`RESTORE --STAGED:`** mengeluarkan perubahan dari staging tanpa menghapus perubahan dari working tree