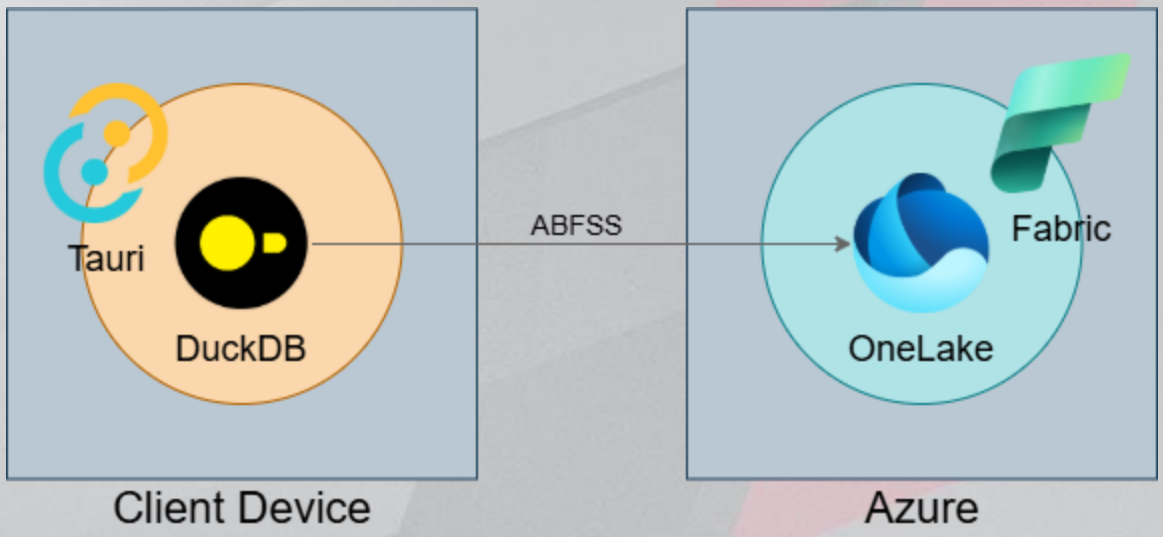
Native App Development Using DuckDB

Toyota Gazoo Racing World Rally Team

Data Engineering Tool

- A hybrid offline/online tool for car engineers
- Tauri + DuckDB + Microsoft Fabric (Delta)

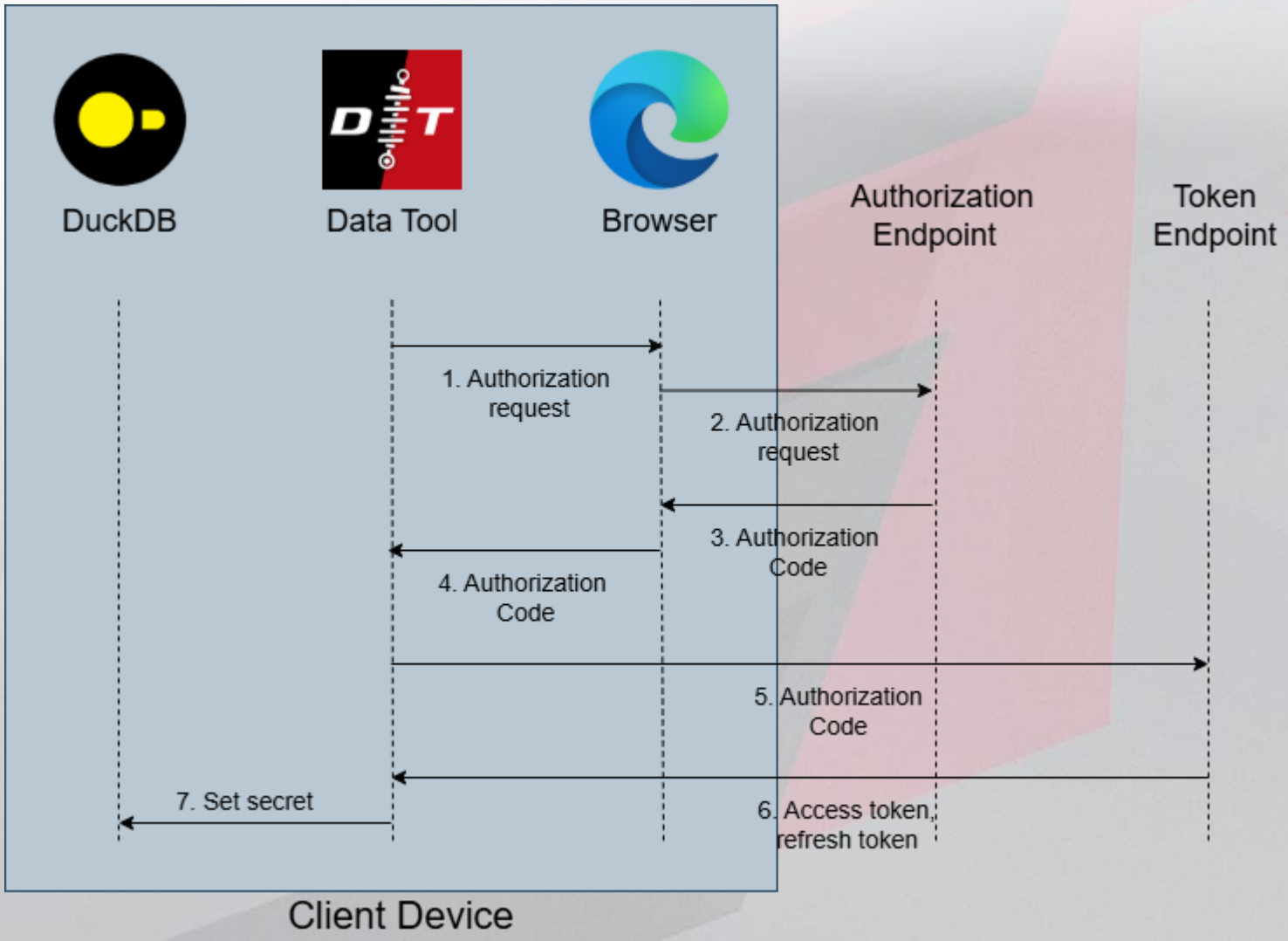


Database Initialization

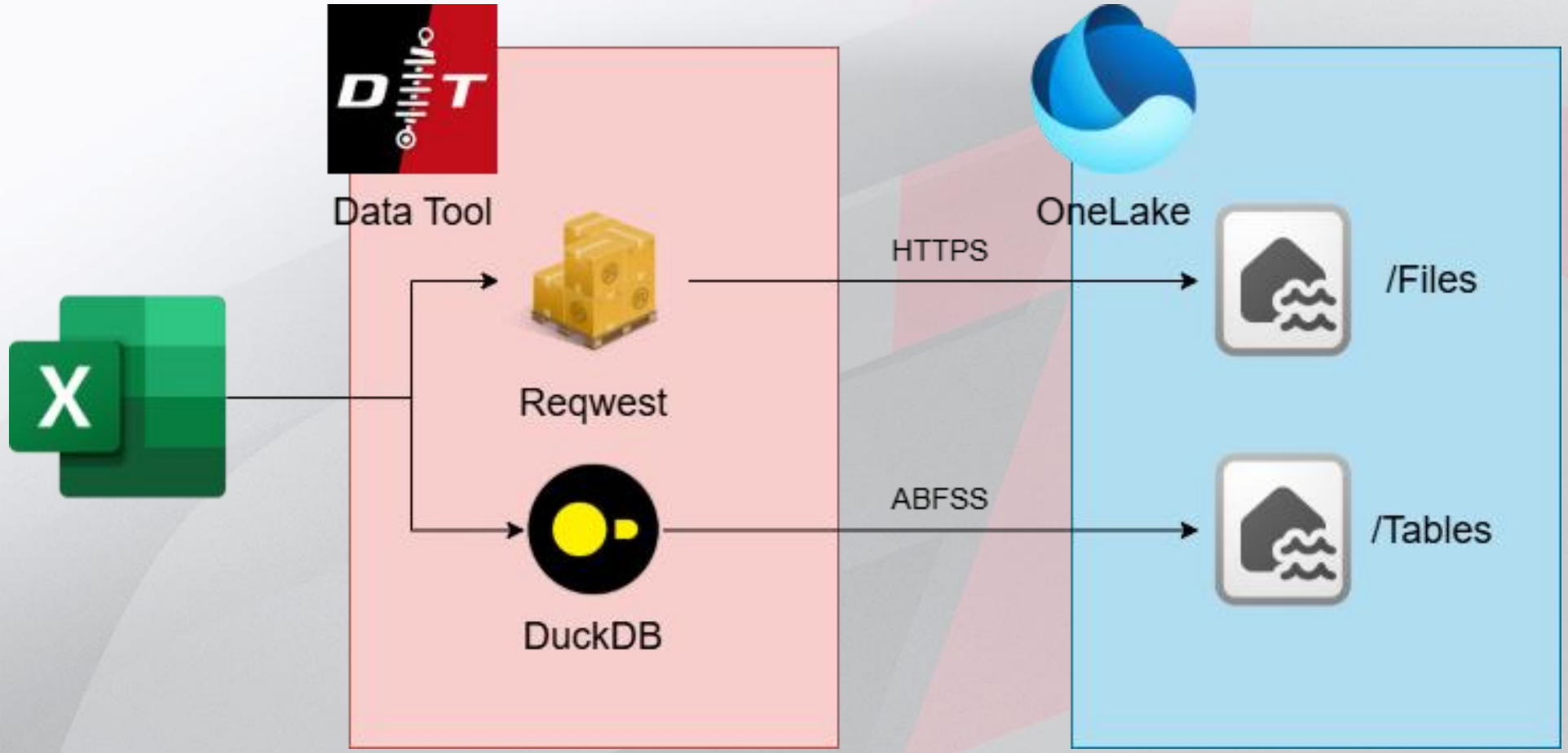
```rust
use std::{sync::mpsc, thread};

type DbJob = Box<dyn FnOnce(&Connection) + Send>;
type DbSender = mpsc::Sender<DbJob>;

fn start_db_thread(conn: Connection) -> DbSender {
    let (tx, rx) = mpsc::channel::<DbJob>();
    thread::spawn(move || {
        for job in rx {
            job(&conn);
        }
    });
    tx
}

fn init_db(conn: &Connection) -> anyhow::Result<()> {
    conn.execute_batch(
        "ATTACH '<LOCAL_DB>' AS local
            (ENCRYPTION_KEY '<KEY>', ENCRYPTION_CIPHER 'GCM');
         USE local;
         CREATE TABLE IF NOT EXISTS measurement (/* ... */);",
    )?;
```

- *Mpsc worker replaces application-level locking and asynchronous lock acquisition*

OAuth 2.0 for Native Apps



Data To OneLake



Insert queries

- Leverage the delta & azure extensions

```rust
fn upload_frames(conn: &Connection, token: String) -> anyhow::Result<()> {
    conn.execute(
        "CREATE OR REPLACE SECRET secret
            (TYPE AZURE, PROVIDER ACCESS_TOKEN, ACCESS_TOKEN ?, ACCOUNT_NAME 'onelake');",
        params![token],
    )?;

    conn.execute_batch("CREATE OR REPLACE TEMP TABLE measurement_staged (/* ... */);")?;
    {
        // appender
    }

    conn.execute_batch(
        "ATTACH IF NOT EXISTS '<ABFSS>/Tables/measurement' AS remote (TYPE delta);
         INSERT INTO remote SELECT * FROM measurement_staged;
         DETACH remote;",
    )?;
    Ok(())
}
```

Select queries

- *Delta_scan queries*

```rust
fn sync_from_remote(conn: &Connection, token: String) -> anyhow::Result<()> {
    conn.execute(
        "CREATE OR REPLACE SECRET secret
            (TYPE AZURE, PROVIDER ACCESS_TOKEN, ACCESS_TOKEN ?, ACCOUNT_NAME 'onelake');",
        params![token],
    )?;

    // delta_scan reads the lakehouse Delta table like any other table
    let mut stmt = conn.prepare("SELECT * FROM delta_scan(?);")?;
    let mut rows = stmt.query(params!["<ABFSS>/Tables/measurement"])?;
    while let Some(_row) = rows.next()? {
        // collect into typed frames ...
    }

    // Rebuild the local tables

    Ok(())
}
```

Takeaways

- Considerations
  - A couple of hard-to-trace issues – help is a GitHub ticket away
- Successes
  - Embedded OLAP
  - Encryption
  - Fabric support with user-derived access

Thank you !!! ☺