# Selium

Be productive in minutes.
Not days.

# G'day!

me

Steve

## What is messaging?

- Comms between two or more systems
- Generally async and stateful
- Various patterns - pub/sub, fanout, RPC

## Why?

CONFLUENT

Apache ACTIVEMQ

RabbitMQ

ØMQ

kafka

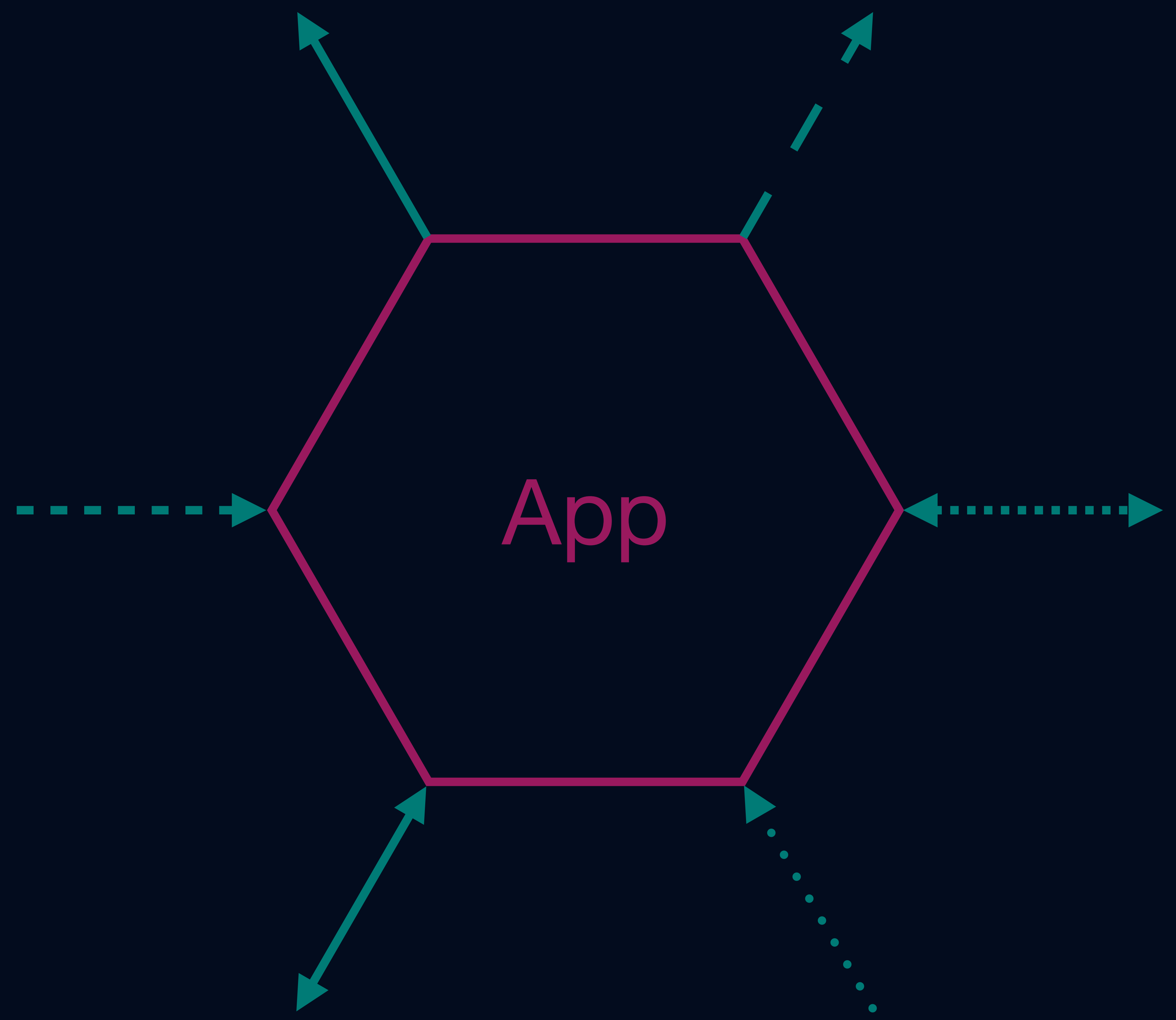
amazon KINESIS

IBM MQ

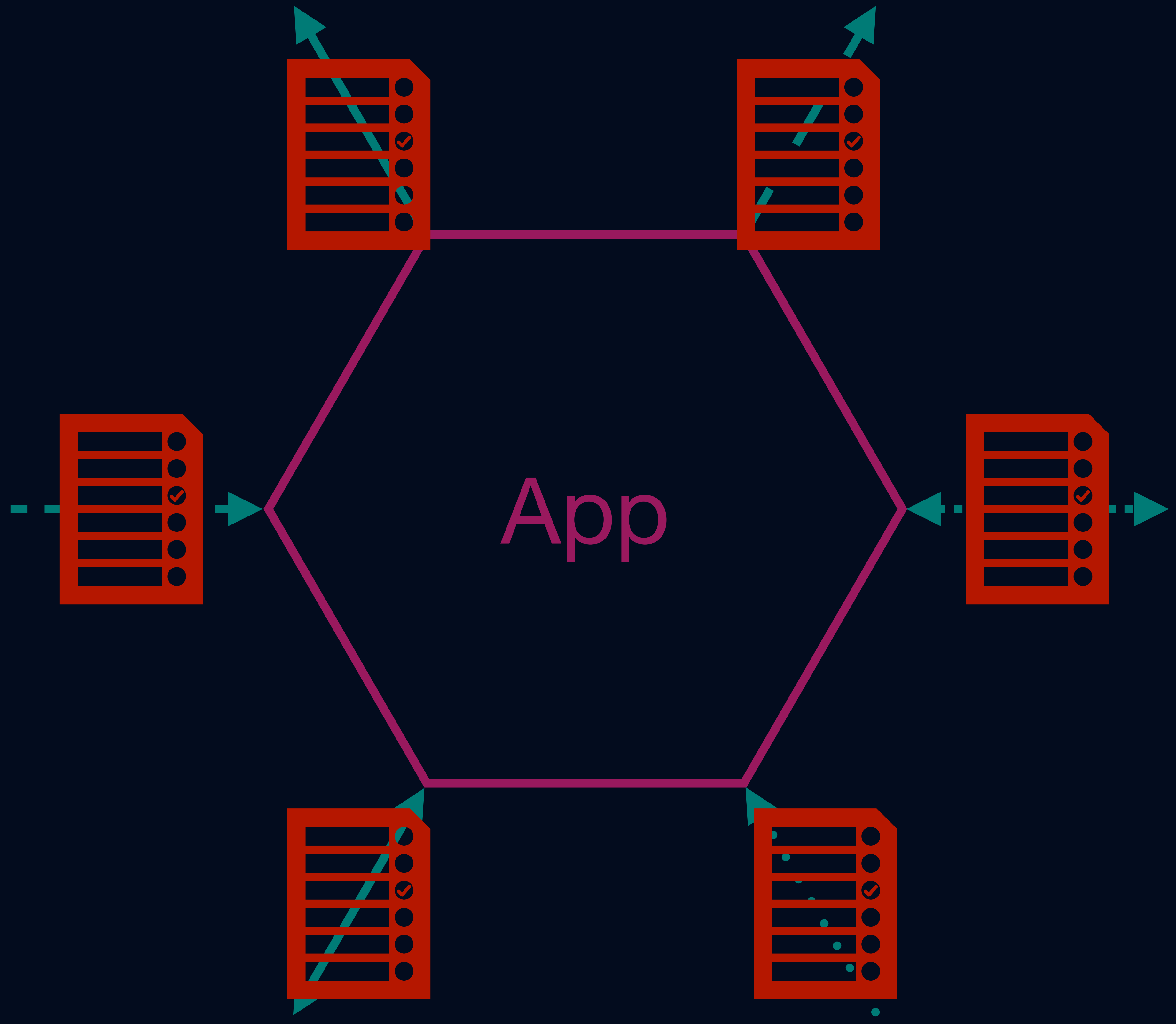
redis

Google Cloud Pub/Sub

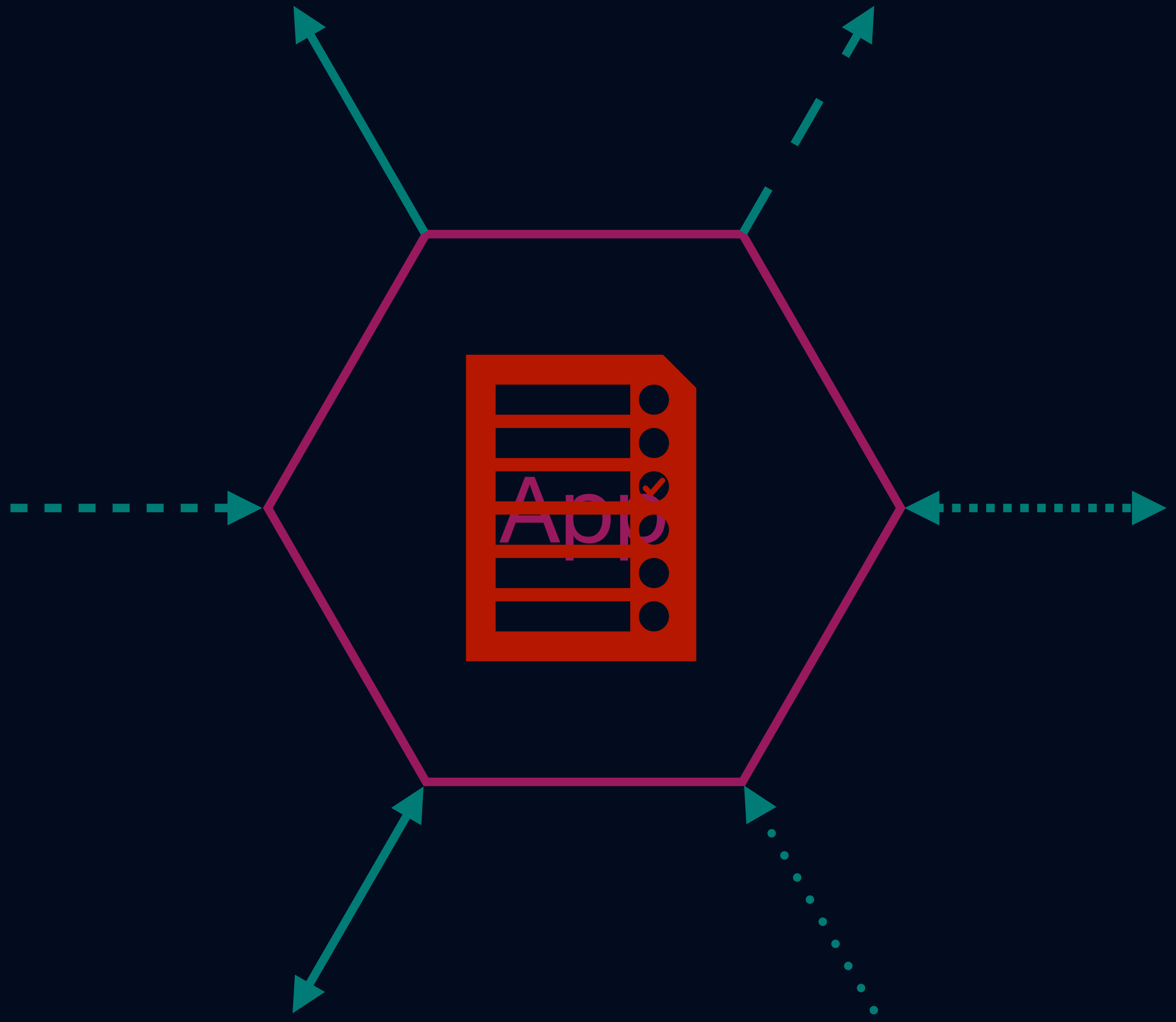
## Apps have different types of communication.



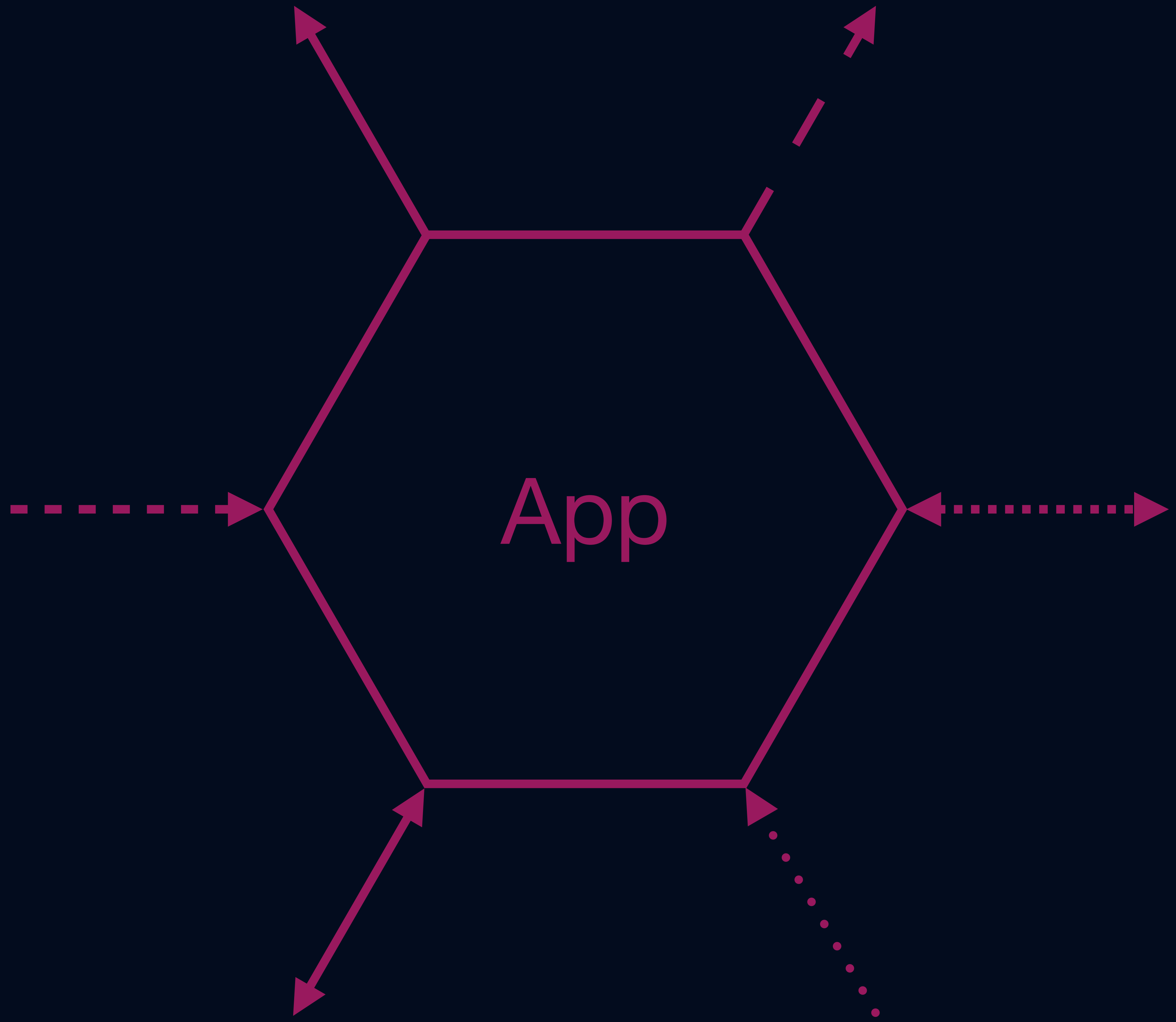
## Apps have different types of communication.



## Apps have different types of communication.



## Make communication just work.



# Subscribe to stock price

```
// Connect to Selium
let selium = ClientBuilder
        .new("/my-trading-firm/secret-project/dev") // Declare your namespace
        .use_localhost() // Connect to 127.0.0.1
        .await;

selium
        .subscribe("stocks")
        // Make sure every message gets delivered
        .guarantee(Delivery::AtLeastOnce)
        // Backup this stream in Selium for 10 days
        .retain_for(Duration::from_days(10))
        // Restart the stream from where we left off
        .replay_from(Cursor::LastOffset)
        // Filter out other stocks
        .filter(|offset, stock_update| stock_update.code == "AAPL")
        // Do something with the data...
        .for_each(|offset, stock_update| {
            println!("I like the stock!");
            future::ready(())
        })
        .await;
```

## Publish awful jokes

```rust
// Connect to Selium
let selium = ClientBuilder
        .new("/my-trading-firm/secret-project/prod") // Declare your namespace
        .use_cloud() // Connect to selium.io
        .add_client_cert(client_cert) // Authenticate me
        .await;

// Publish random XKCD quotes
let sink = selium
    .publish("xkcd")
    // Duplicate this very important stream
    .redundancy(1) // @todo
    .await;
sink.send("This joke isn't very funny".into());
```

## Ask very secret questions

```rust
// Connect to Selium
let selium = ClientBuilder
        .new("/my-trading-firm/public-project/uat") // Declare your namespace
        .use_cloud() // Connect to selium.io
        .add_client_cert(client_cert) // Authenticate me
        .add_server_cert(server_cert) // Authenticate Selium's server
        .await;

// Setup an RPC
selium
    .rpc("secrets")
    .try_for_each(|request| future::Ok("sshhh!"))
    .await?;

// Ask a question
let reply = selium.request("secrets").send("db_password".into()).await?;
dbg!(reply); // Ok("sshhh!")
```

## Some fun specs

- Zero copy path
- Protocol ambivalent
- Reliable UDP
- Soon to be open source (contributions most welcome!)
- *More cool stuff…*

WA

Sign up for updates
and early access
at selium.com

## Selium. Be productive in minutes.

## Not days.

hello@selium.com

www.selium.com