Optimizing for Happiness and Speed

Darko Fabijan

Belgrade, Serbia 2023

PERCONA UNIVERSITY

FerretDB

PERCONA

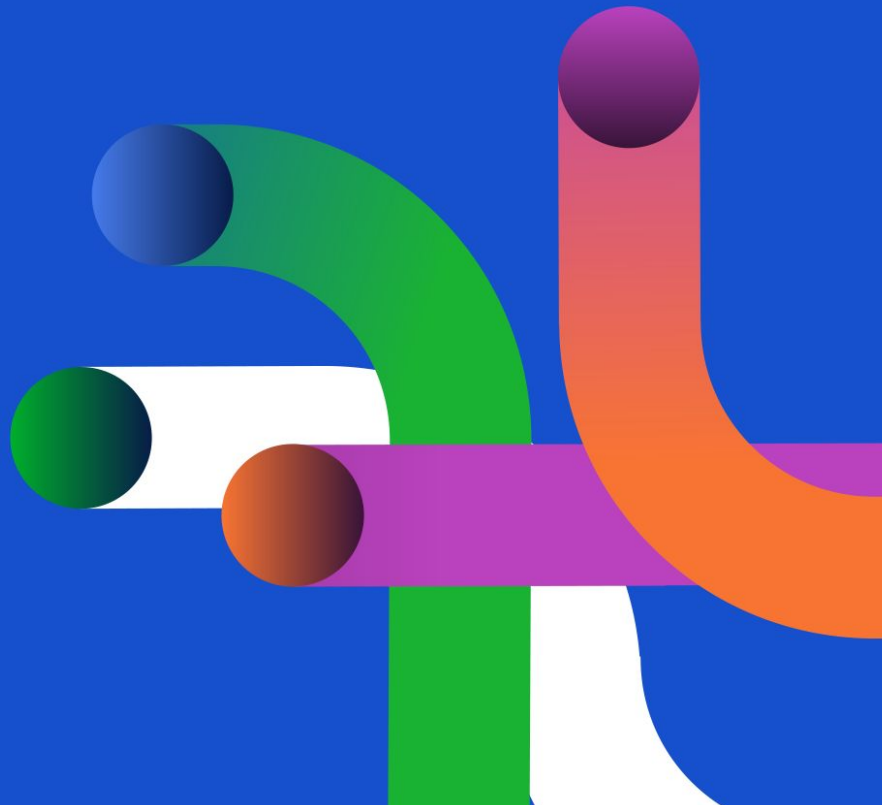
Singidunum University

semaphore

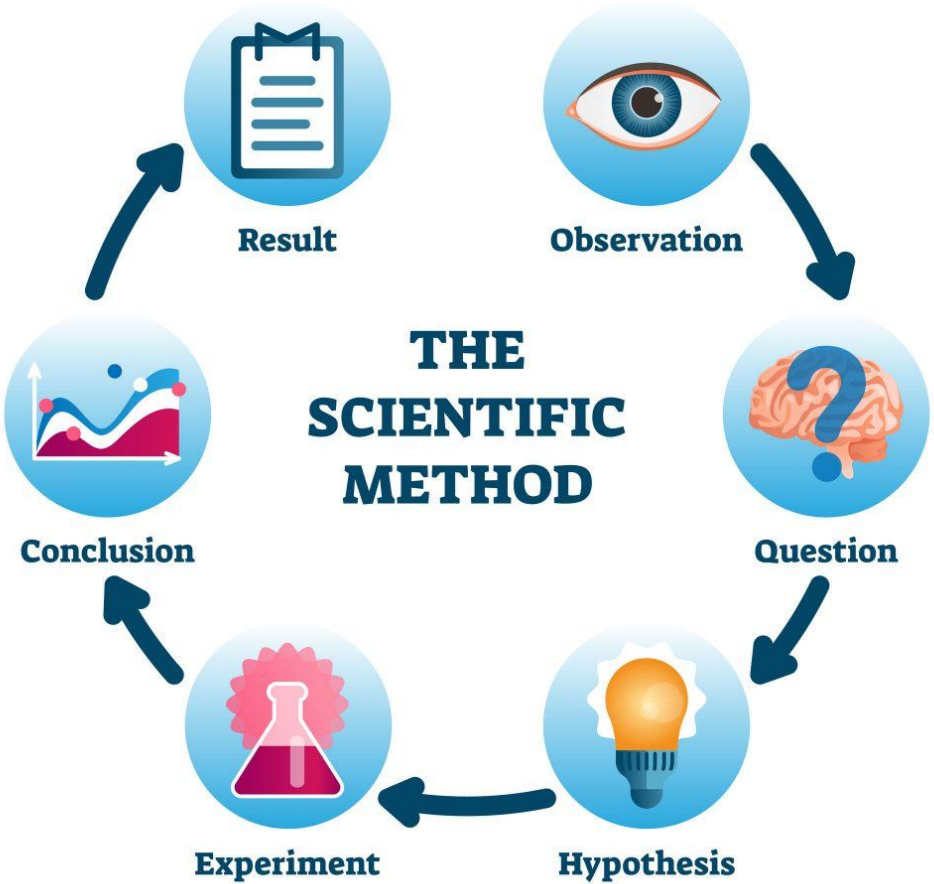
Who am I?

Darko Fabijan
Co-founder & CTO @ Semaphore CI

Novi Sad, Serbia
@darkofabijan



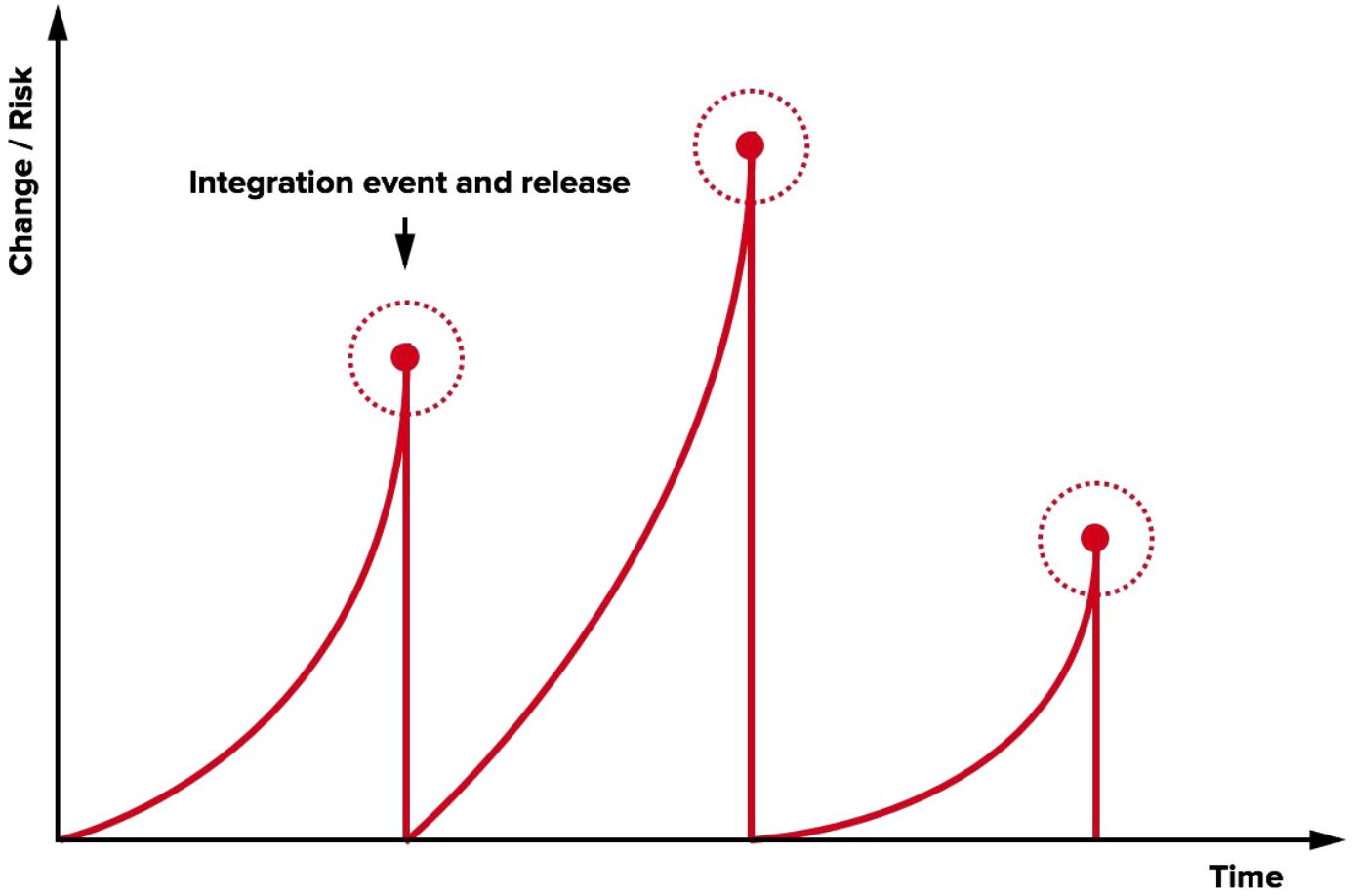
1. What do you need to be happy?
2. Long term happiness with your app?
3. How to get and stay there?


THE SCIENTIFIC METHOD
Observation
Question
Hypothesis
Experiment
Conclusion
Result

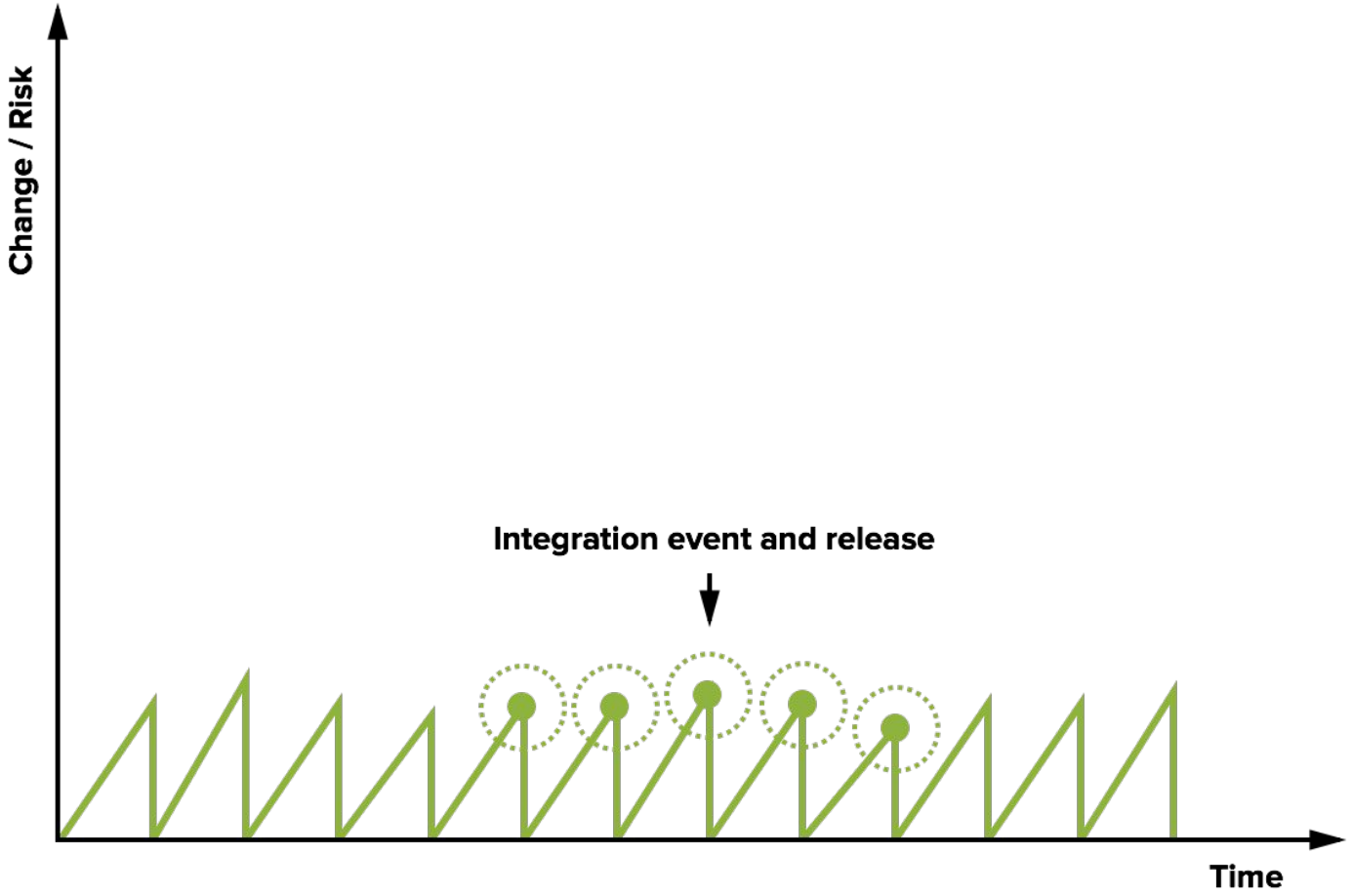
Business & Creating Value

Product creation is hard iterative process

Long term happiness with your app?



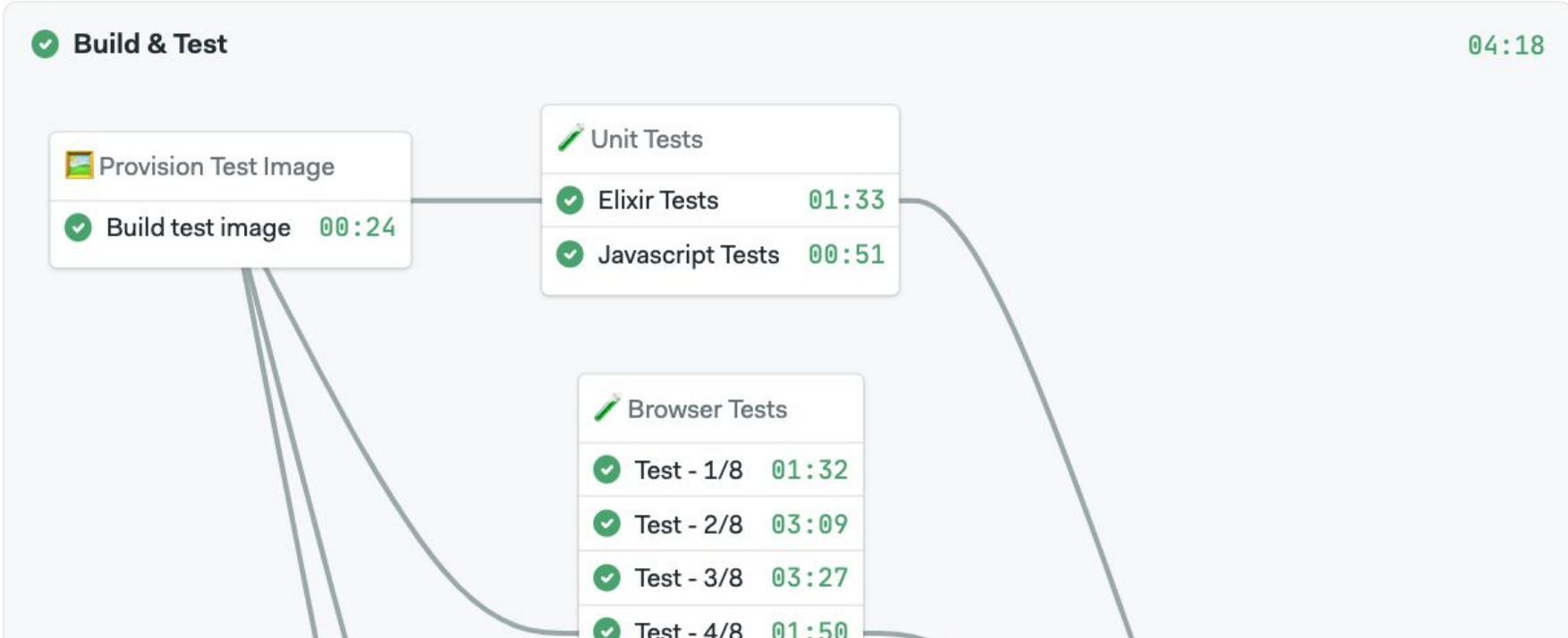
Traditional Integration



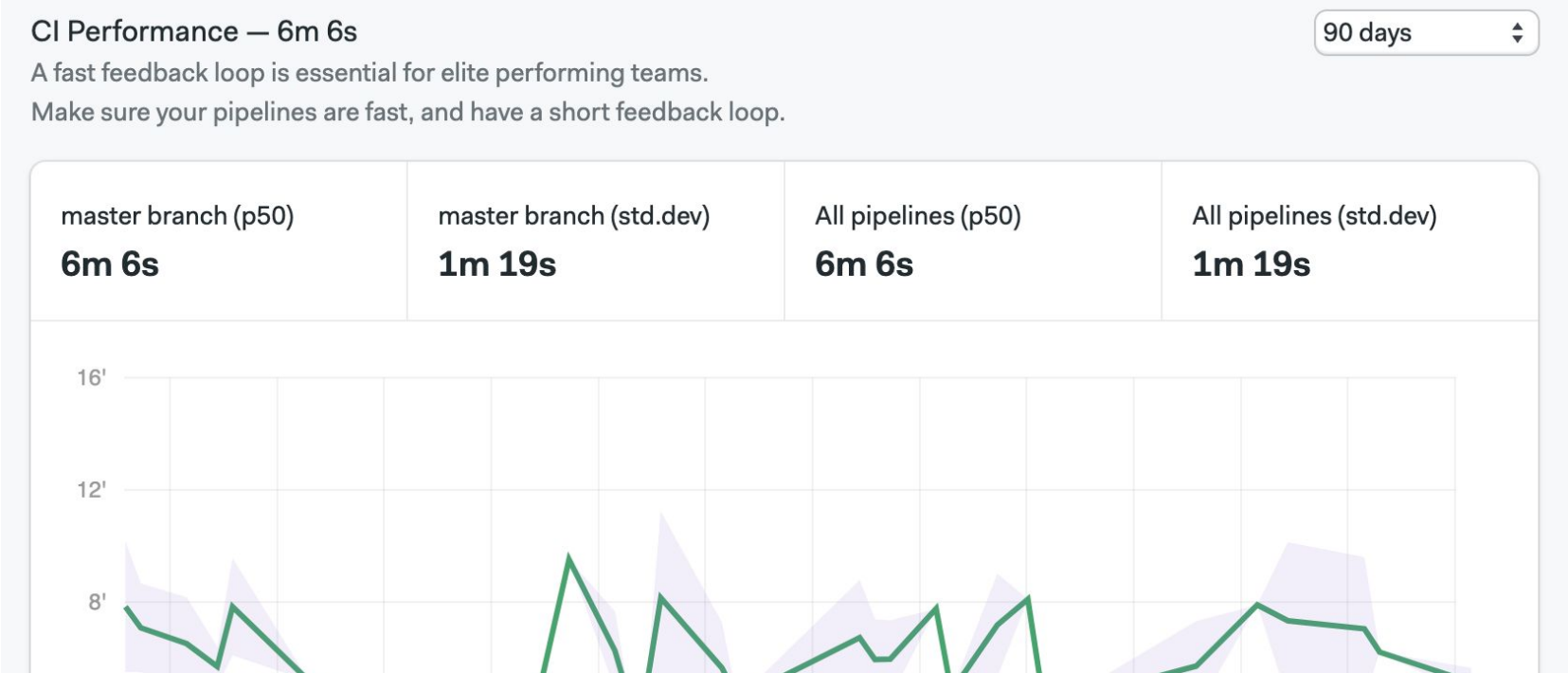
Continuous Integration

Making and keeping your feedback loop fast and reliable

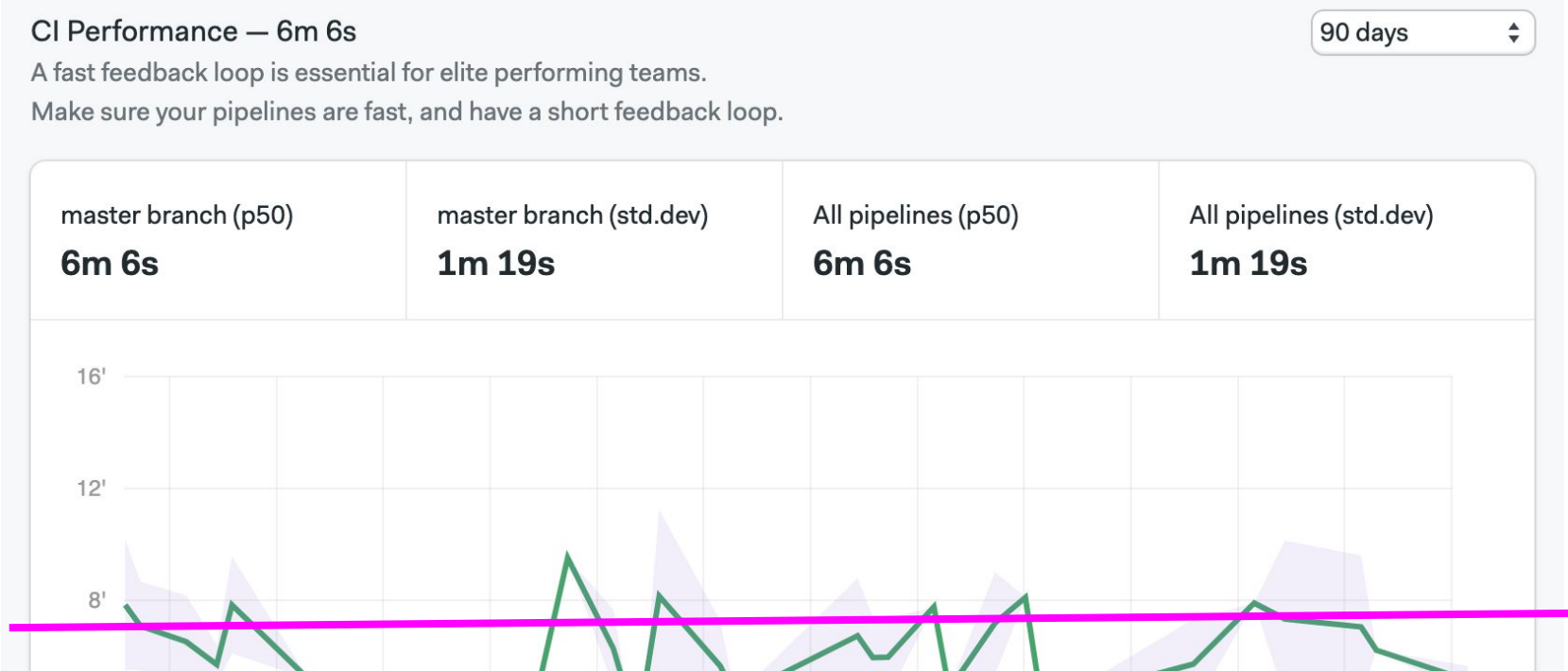
Know your pipeline



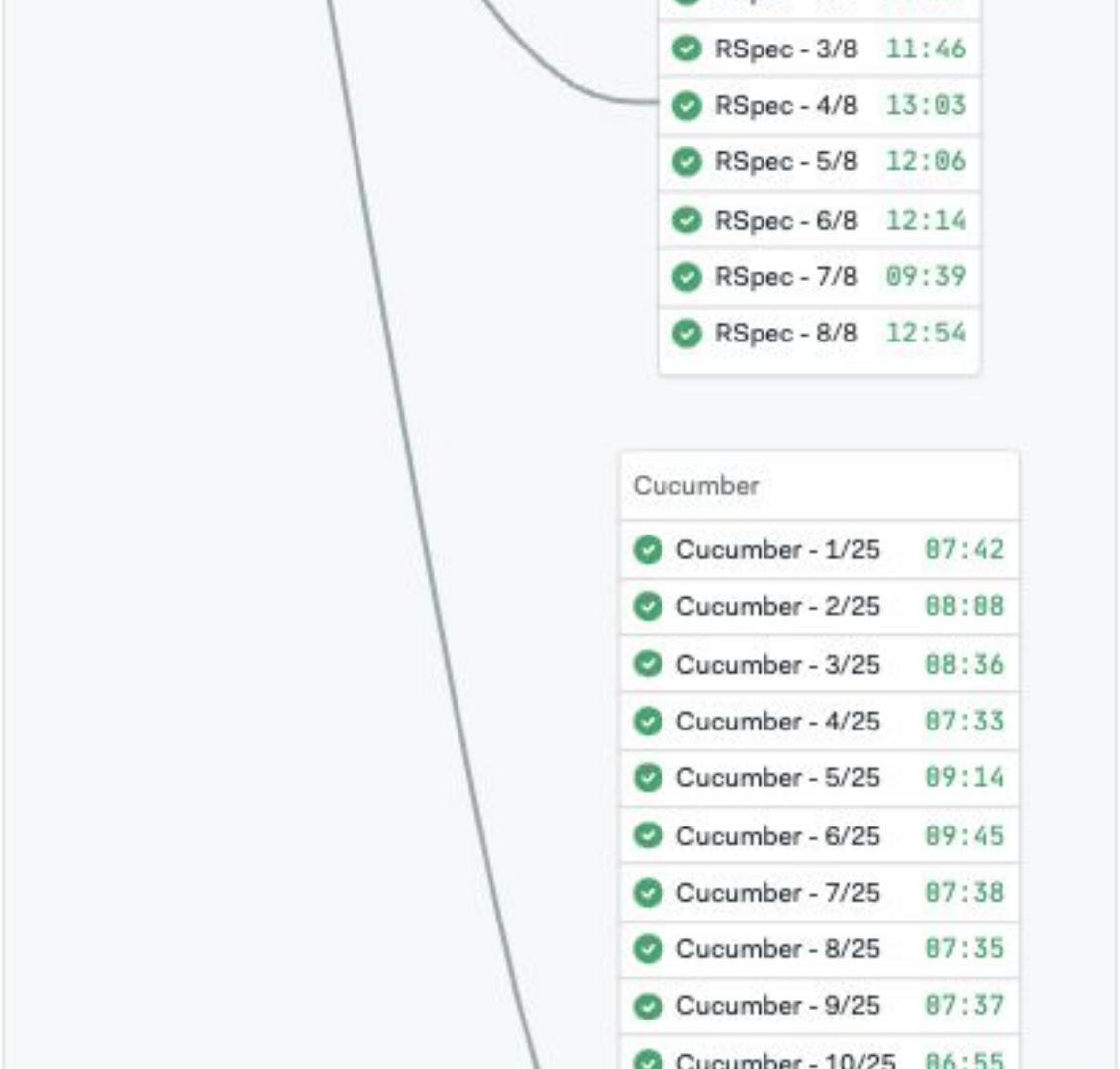
Measure everything



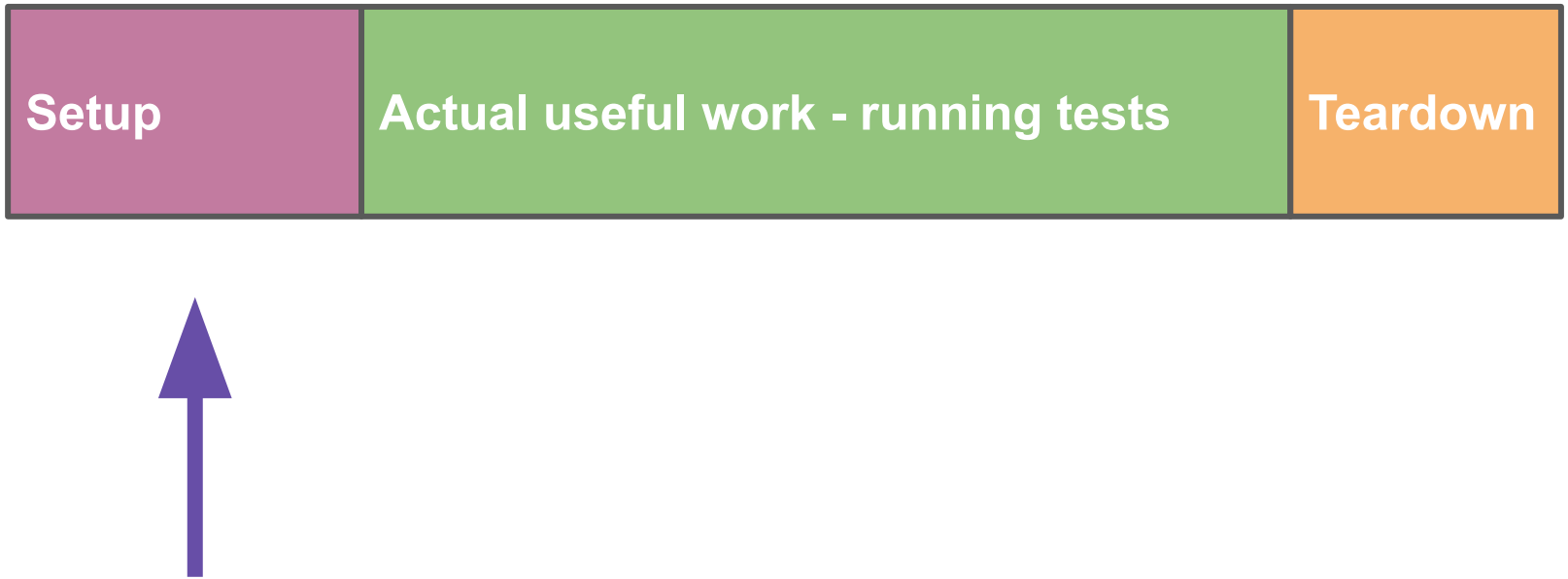
Have a goal


CI Performance — 6m 6s
90 days
A fast feedback loop is essential for elite performing teams.
Make sure your pipelines are fast, and have a short feedback loop.
master branch (p50)
6m 6s
master branch (std.dev)
1m 19s
All pipelines (p50)
6m 6s
All pipelines (std.dev)
1m 19s
16'
12'
8'

Parallelize



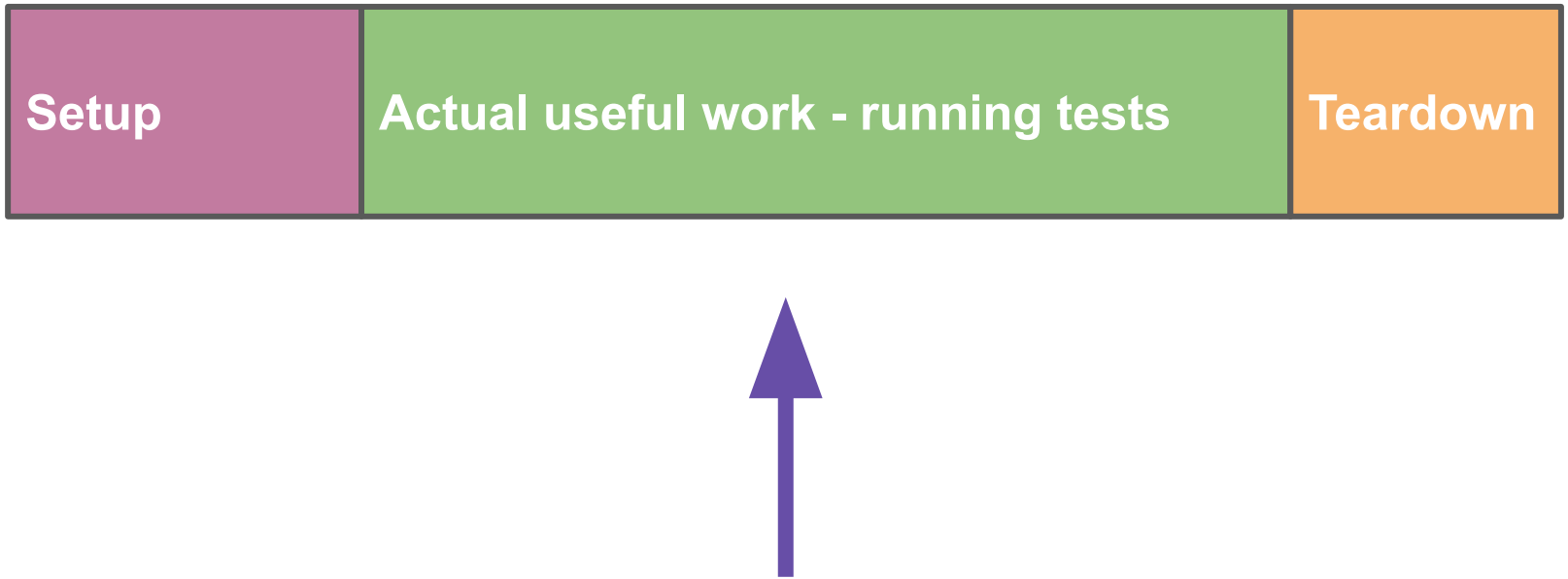
Eliminate waste



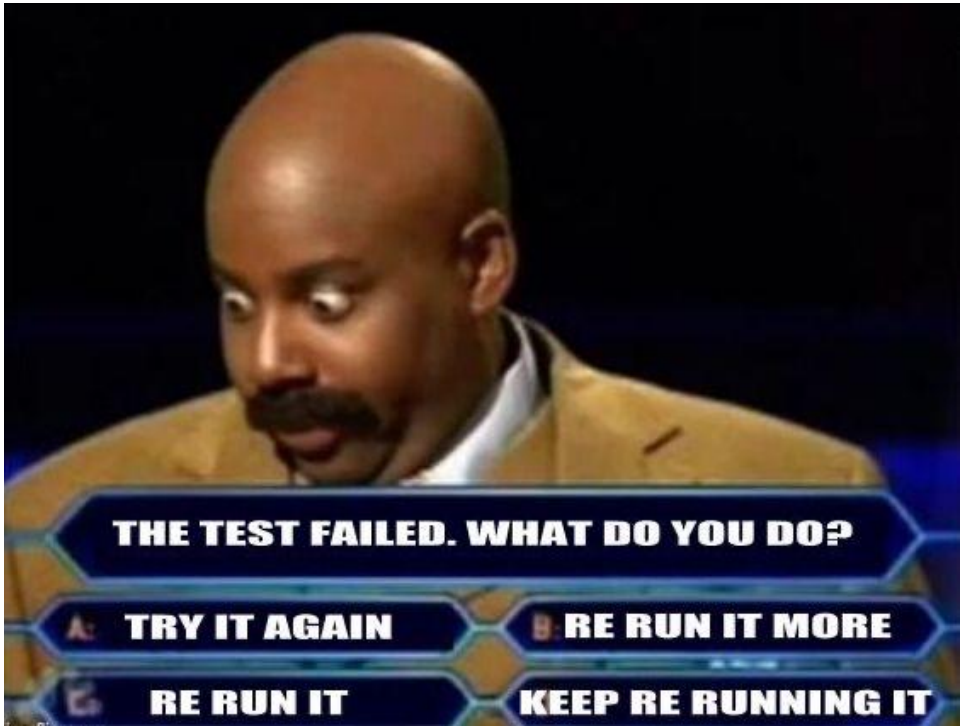
Common setup phase anti-patterns

- Not caching your dependencies
- Using inefficient dependency management system
- Installing unnecessary dependencies
- Migrating databases in inefficient ways
- Slow Docker image builds

Unreliable - Flaky tests



Flaky tests


THE TEST FAILED. WHAT DO YOU DO?
A: TRY IT AGAIN
B RE RUN IT MORE
RE RUN IT
KEEP RE RUNNING IT

Flaky tests

- Finding them & doing forensic work
- Saving debugging information - logs and pictures
- SSH Debugging in real-time
- Documenting & keeping track of them

Flaky tests - FIXING them

The first appearance of a flaky test is the best moment to fix it.

Flaky tests - Determining the cause and fixing the test

- Environmental differences
- Non-deterministic code
- Asynchronous wait
- Concurrency
- Order dependency

Flaky tests - Environmental differences

- Operating system
- Libraries
- Environment variables
- Number of CPUs
- Network speed

Flaky tests - Non-deterministic code

- dates, random values or remote services

```
@Test
public void methodThatUsesNow() {
   String fixedTime = "2022-01-01T12:00:00Z";
   Clock clock = Clock.fixed(Instant.parse(fixedTime), ZoneId.c

   // now holds a known datetime value
   Instant now = Instant.now(clock);

   // the rest of the test...
}
```

Flaky tests - Asynchronous wait

```
click_button “Send”
sleep 5
expect_email_to_be_sent
```


Use polling or callbacks

Flaky tests - Concurrency

Concurrency can be responsible for flakiness due to deadlocks, race conditions, leaky implementations, or implementations with side effects. The problem stems from using shared resources.

```
function testAccountTransfer(fromAccount, toAccount) {
  lockFrom=fromAccount.lock()
  lockTo=toAccount.lock()

  beforeBalanceFrom = getBalance(fromAccount)
  beforeBalanceTo = getBalance(toAccount)
```

Flaky tests - Order dependency

Root of the problem - tests depend on shared mutable data

```
it('Subscribes to newsletter', () => {
…
```

```
it('Unsubscribes from newsletter', () => {
…
```

Questions

The best CI/CD tool for high-performance engineering teams.

semaphore