

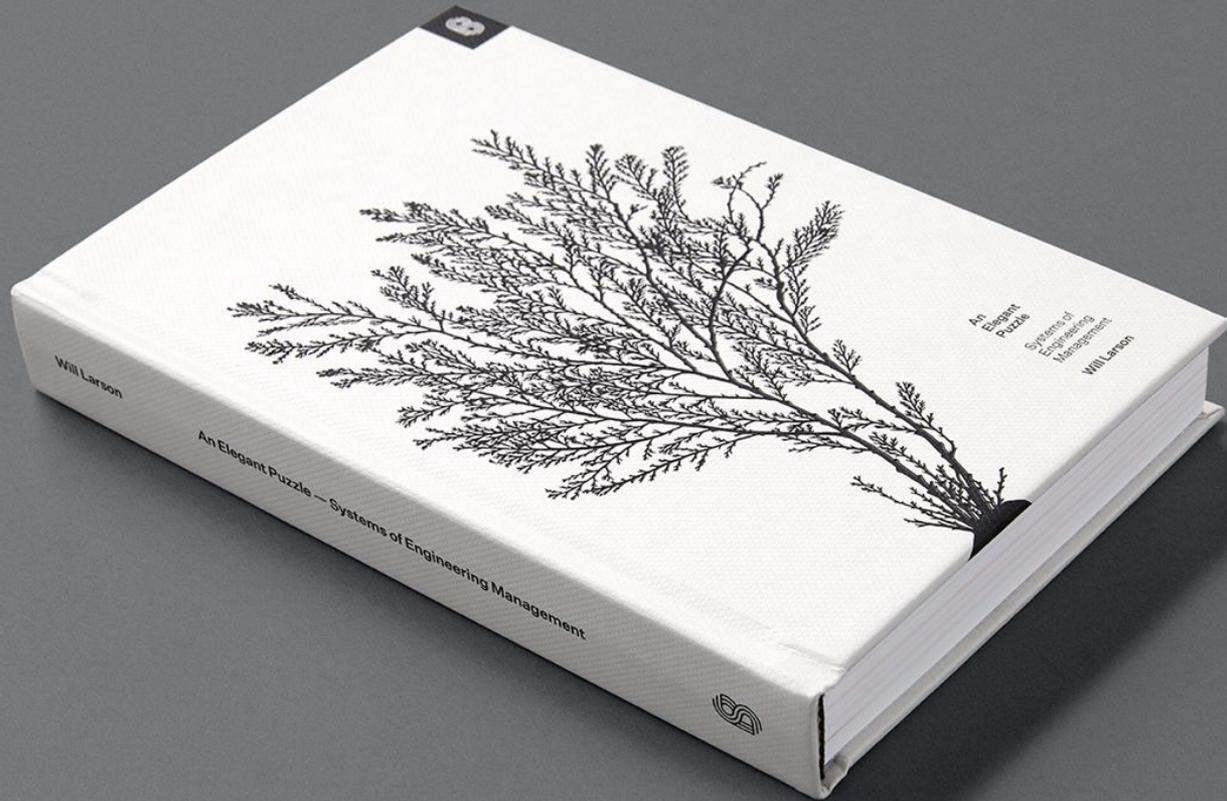
How to invest in technical infrastructure

Will Larson
@lethain

2019

stripe

Uber



An
Elegant
Puzzle
Systems of
Engineering
Management
Will Larson

Will Larson

An Elegant Puzzle — Systems of Engineering Management



Prioritizing infrastructure investment...

...in a high autonomy environment...

...within a rapidly scaling business.

How can infrastructure teams...

...be surprisingly impactful...

...without burning out?

What is technical infrastructure?

Technical infrastructure:

Someone's biggest problem they dislike.

Technical infrastructure:

Tools used by 3+ teams for business critical workloads.

Examples of technical infrastructure

Developer tools

Data infrastructure

Core libraries and frameworks

Model training and evaluation

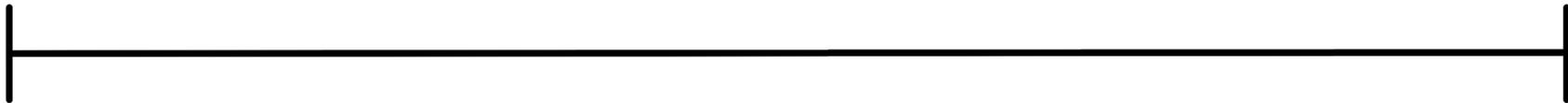
Introduction

- 1. Fundamentals**
2. Escaping the firefight
3. Learning to innovate
4. Navigating breadth
5. Unifying approach

Closing

Forced

Discretionary



Forced

- Scale MongoDB
- Lower AWS costs
- GDPR

Discretionary

- Sorbet
- Monolith -> μ services
- Deep learning

Short-term



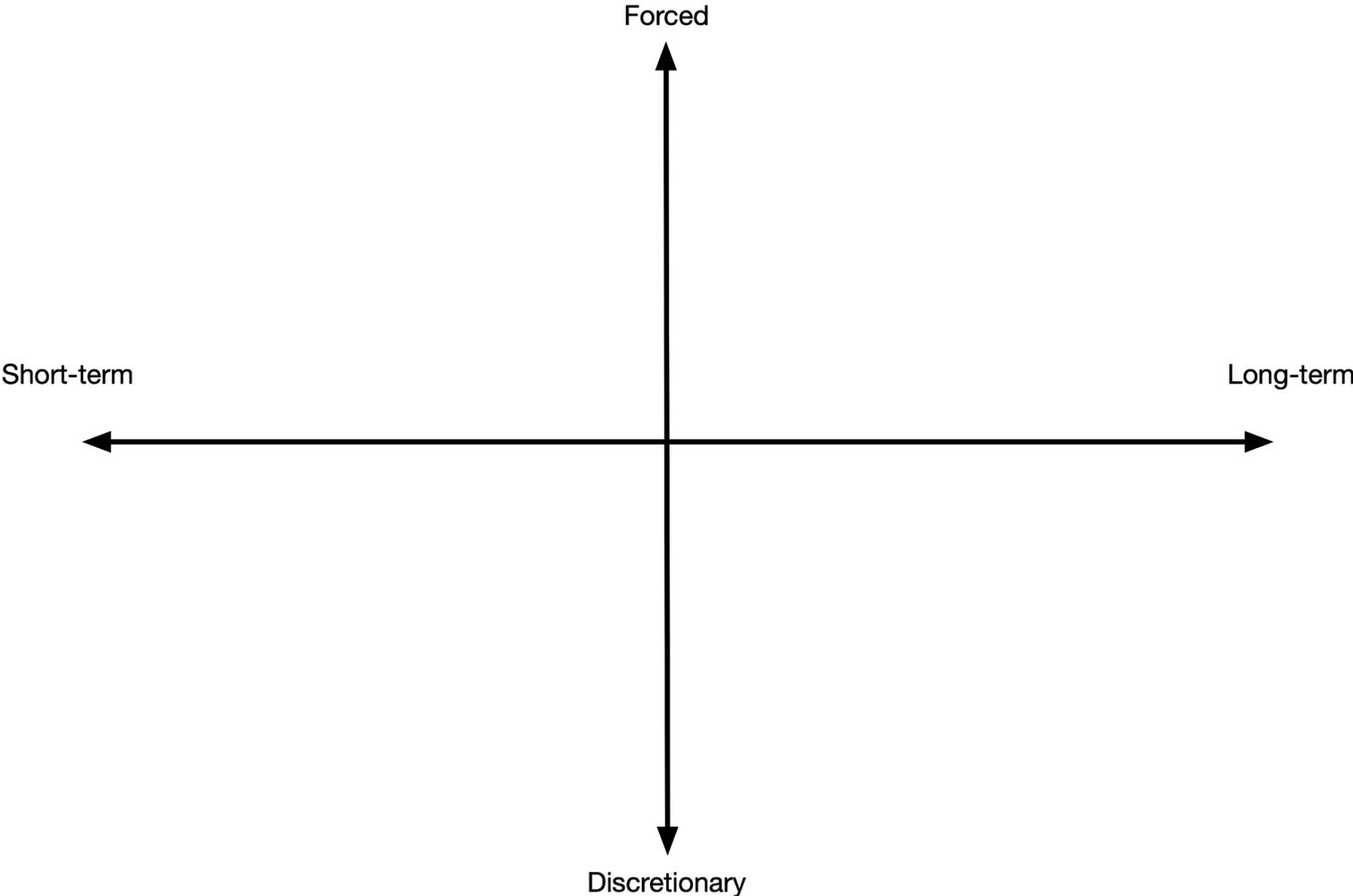
Long-term

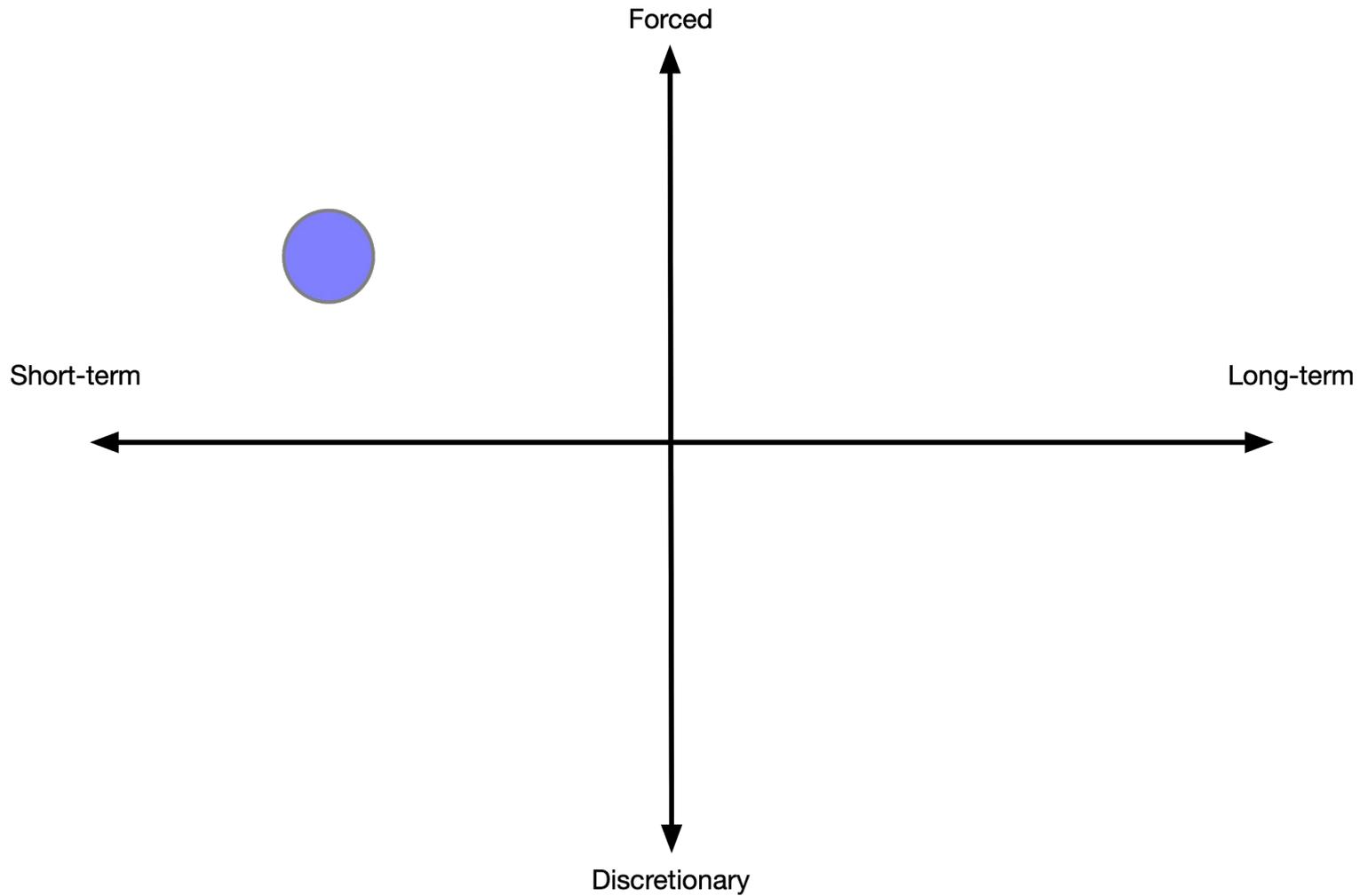
Short-term

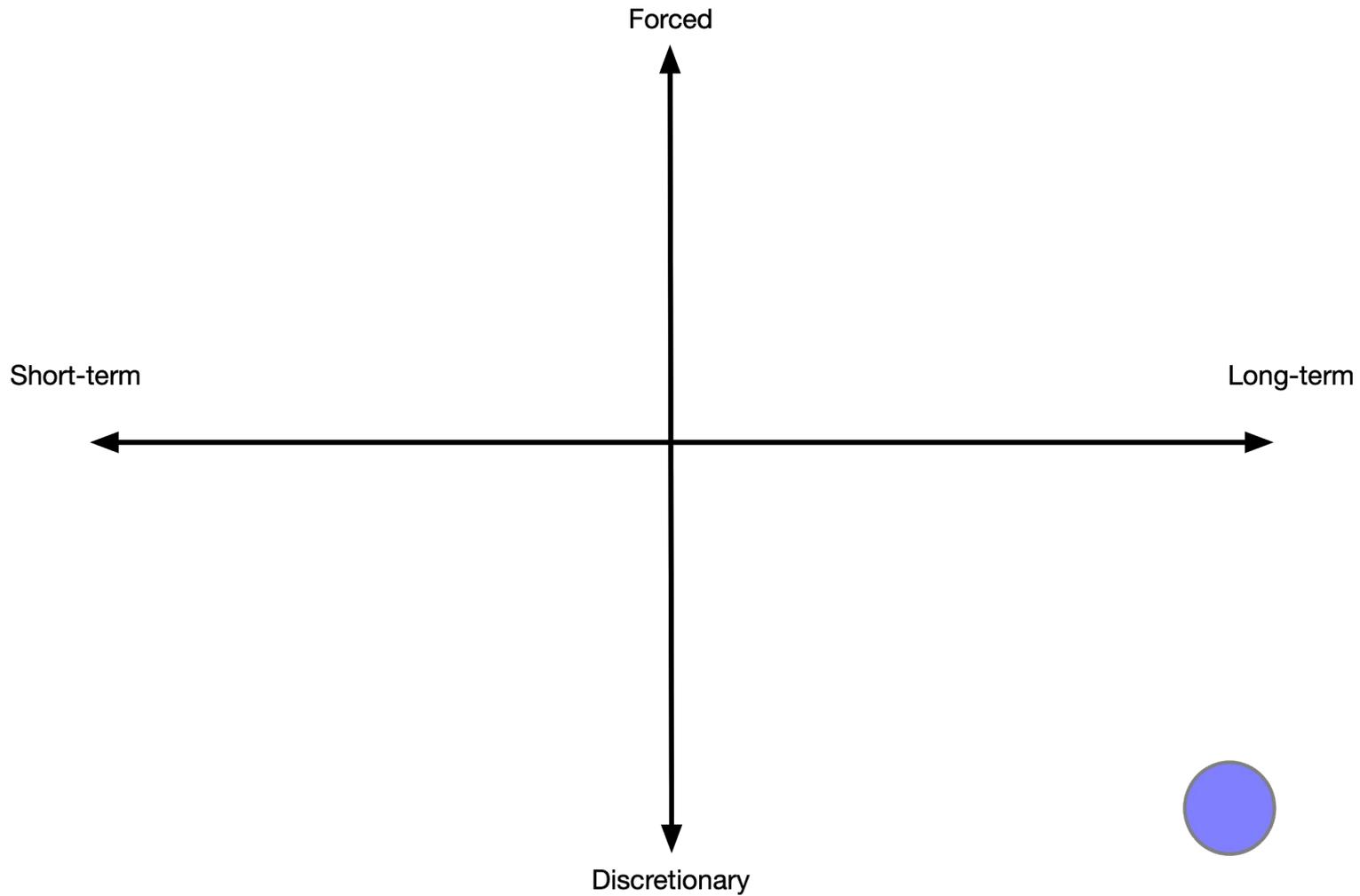
- Critical remediation
- Scale for holidays
- Support launch

Long-term

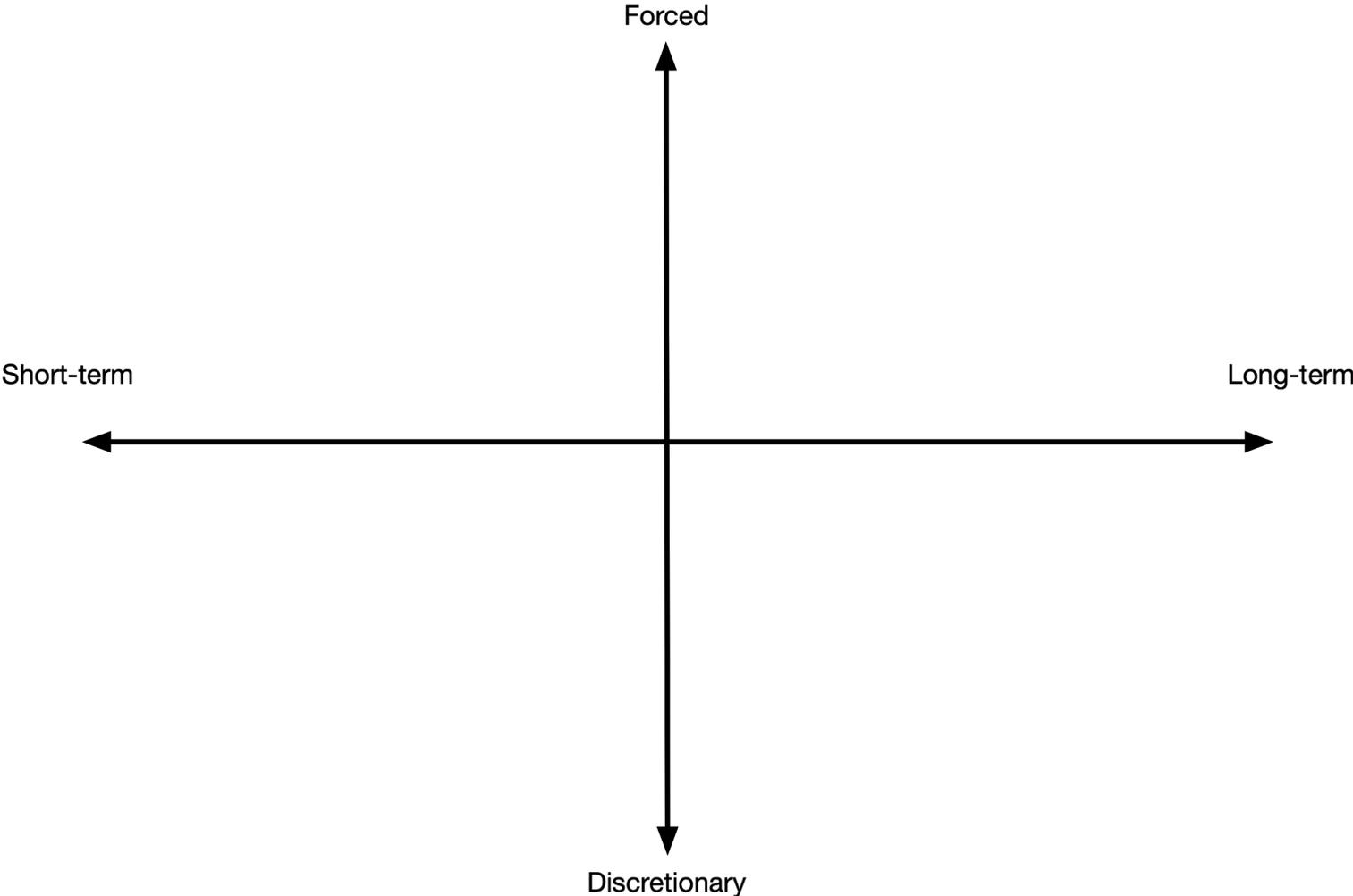
- QoS strategy
- “Bend the cost curve”
- Rewrite monolith



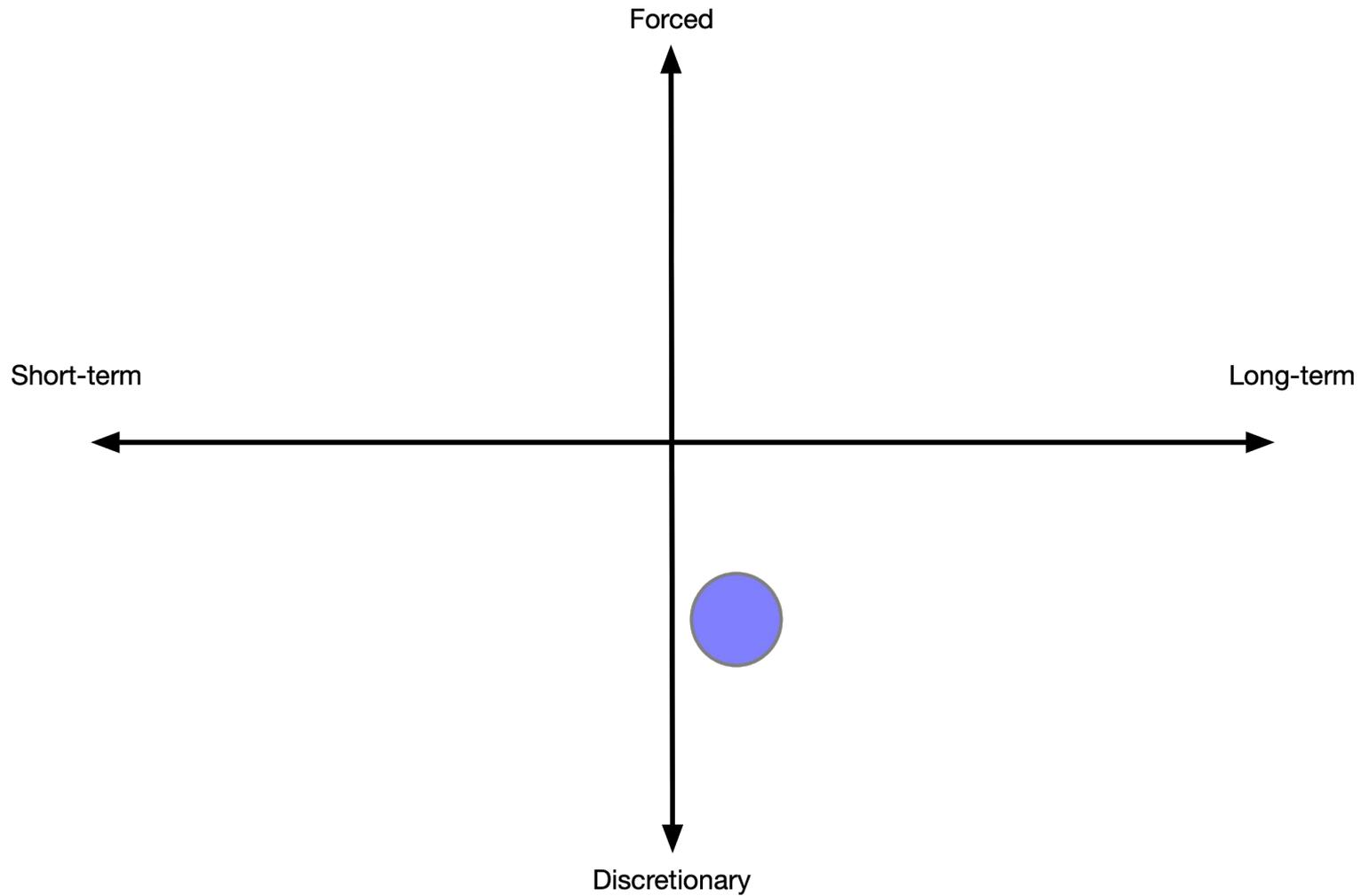




Where is your team now?



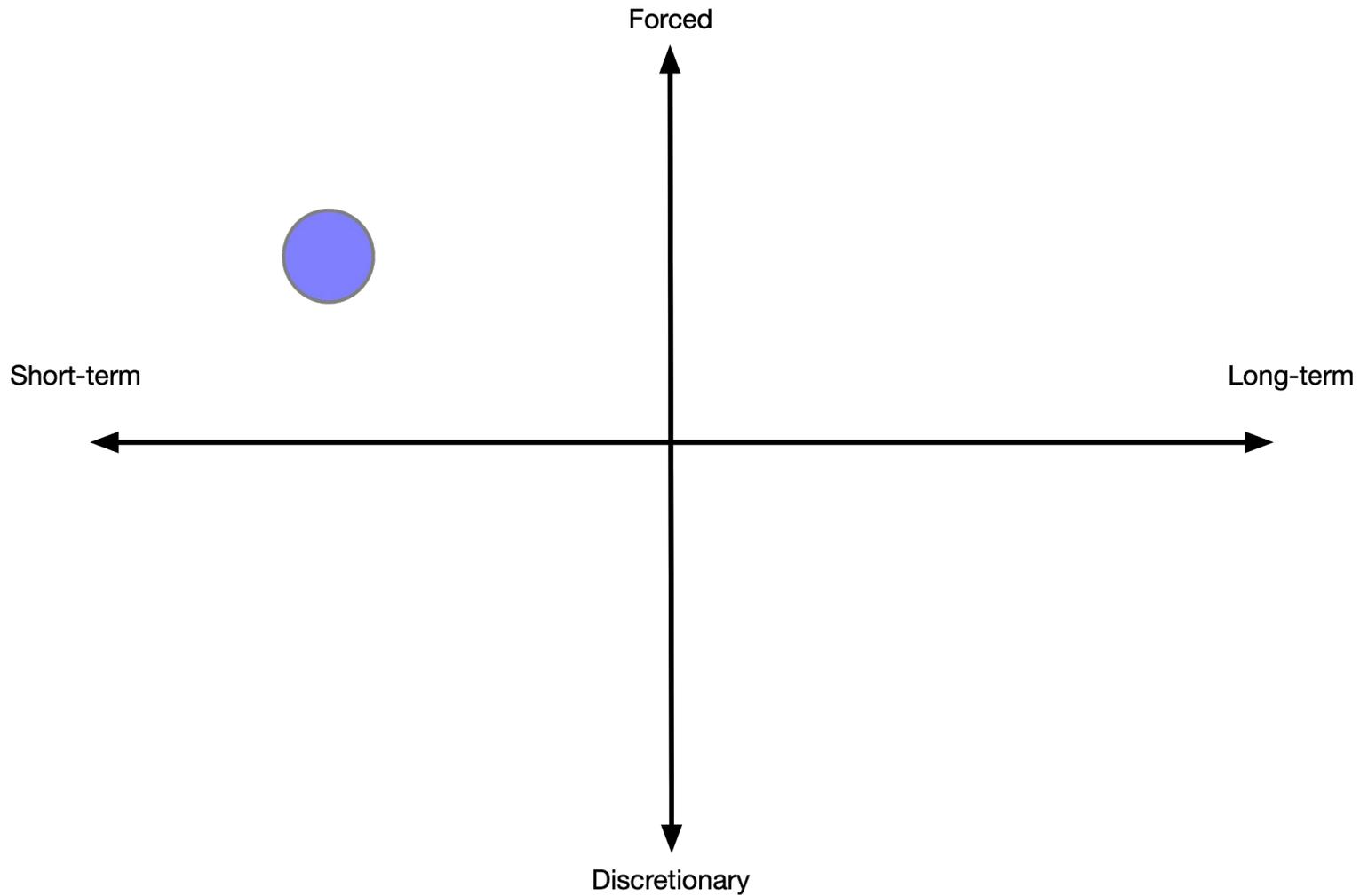
Where do you want to be?



Introduction

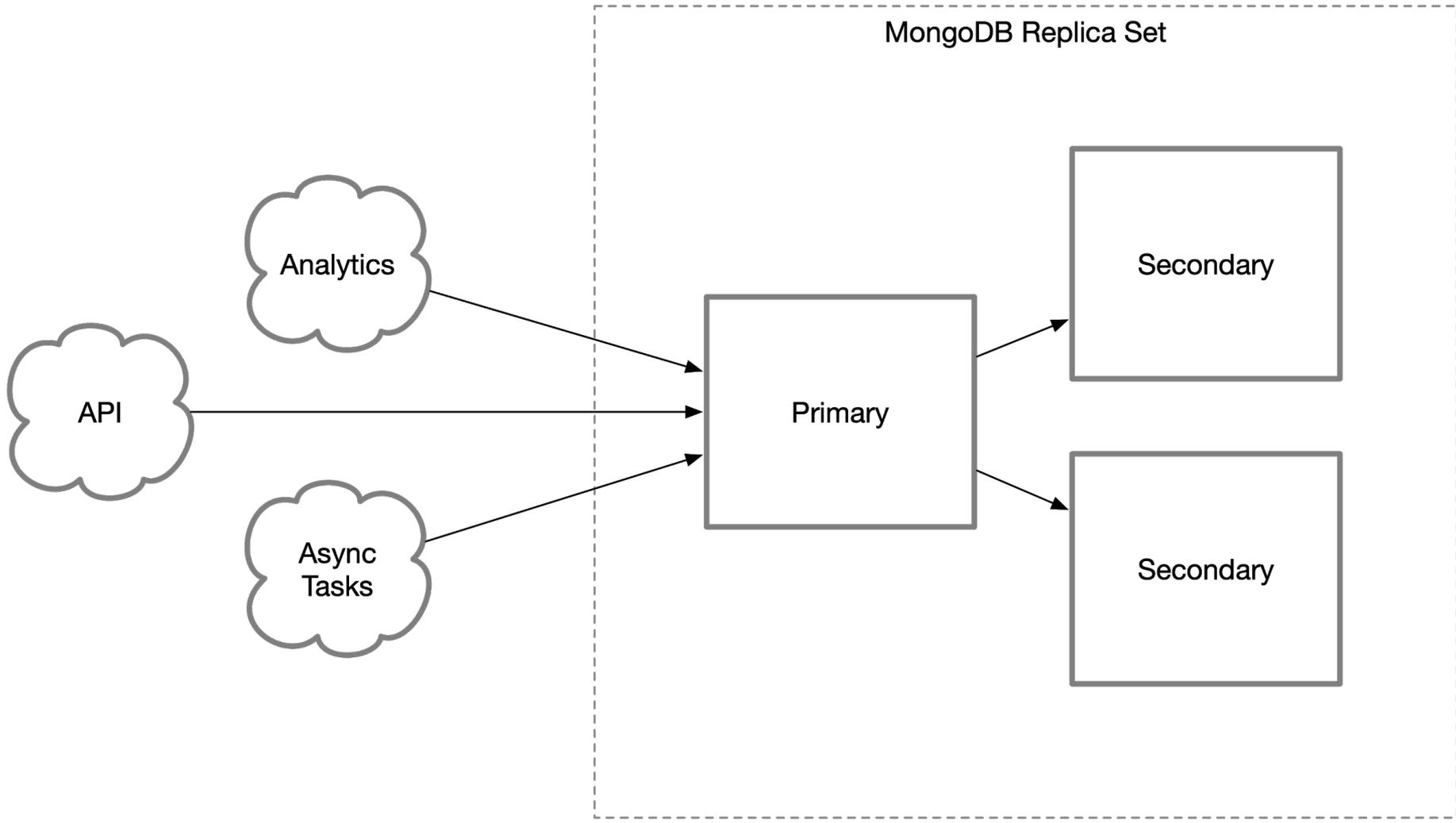
1. Fundamentals
- 2. Escaping the firefight**
3. Learning to innovate
4. Navigating breadth
5. Unifying approach

Closing



Even Stripe...

MongoDB



Shared replsets

Easy to provision :-)

Don't cost much :-)

Shared everything :-\

Joint ownership :-/

Limited isolation :-(
:-)

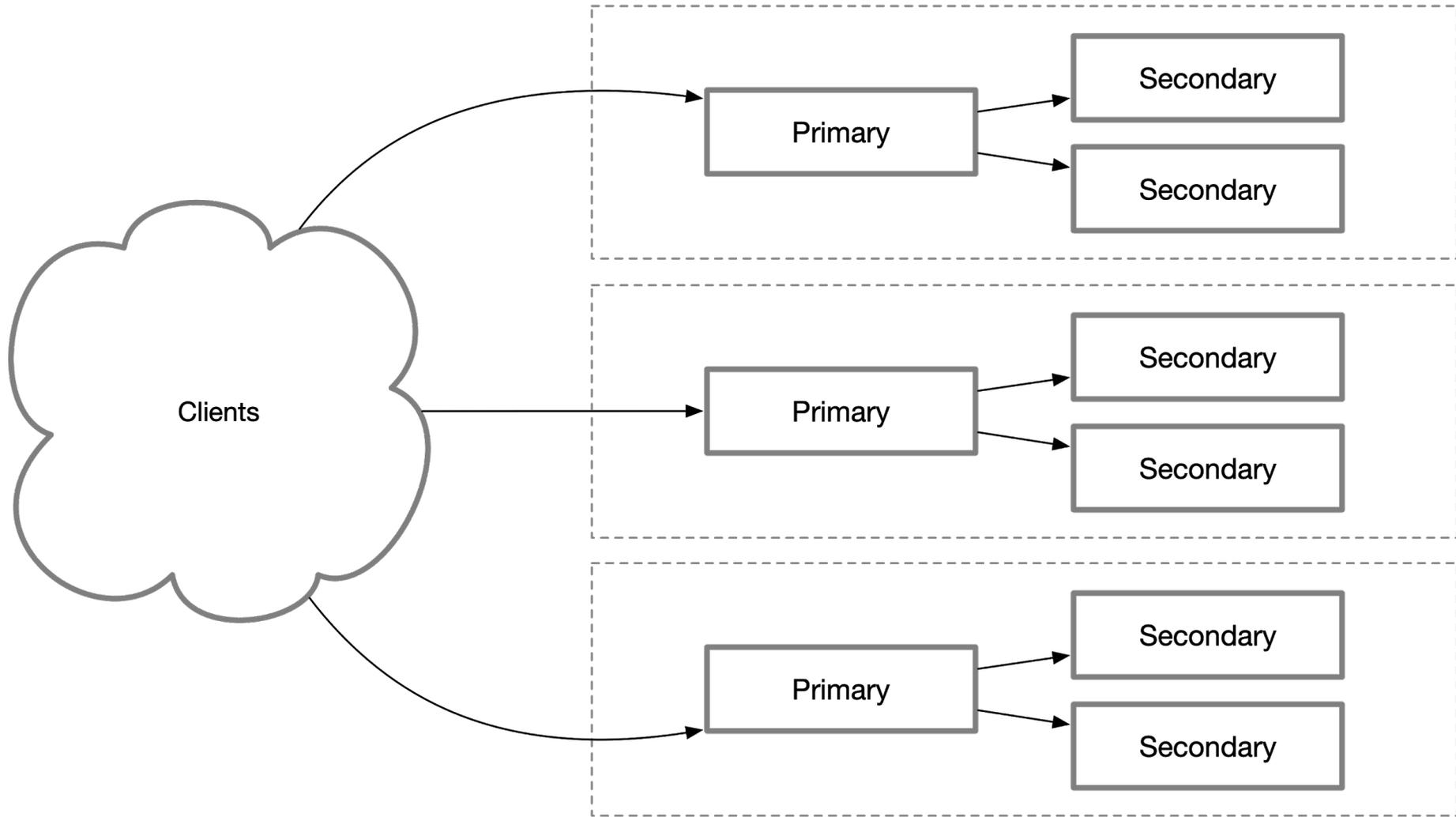
Big blast radius :-(
:-)

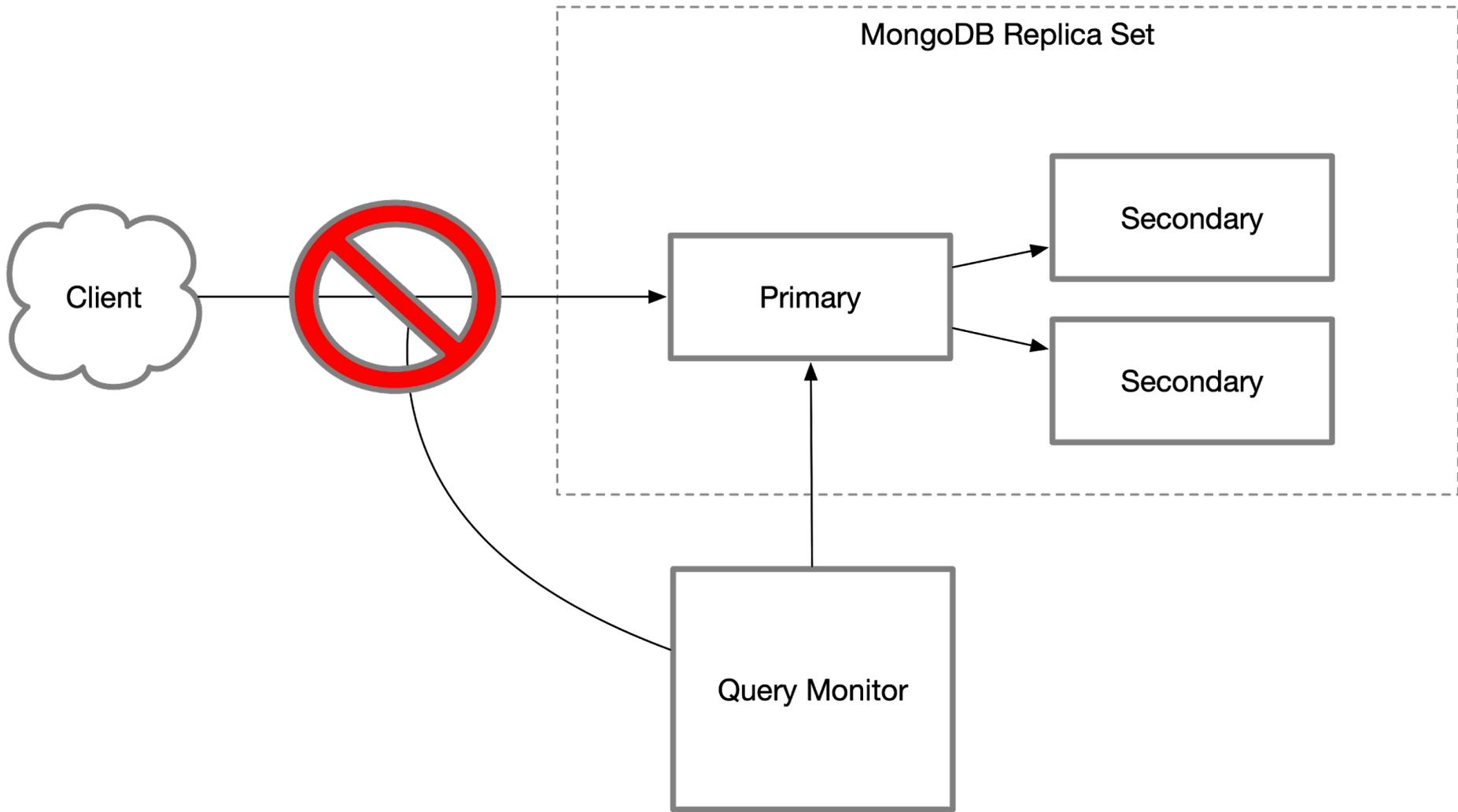
More time on incidents

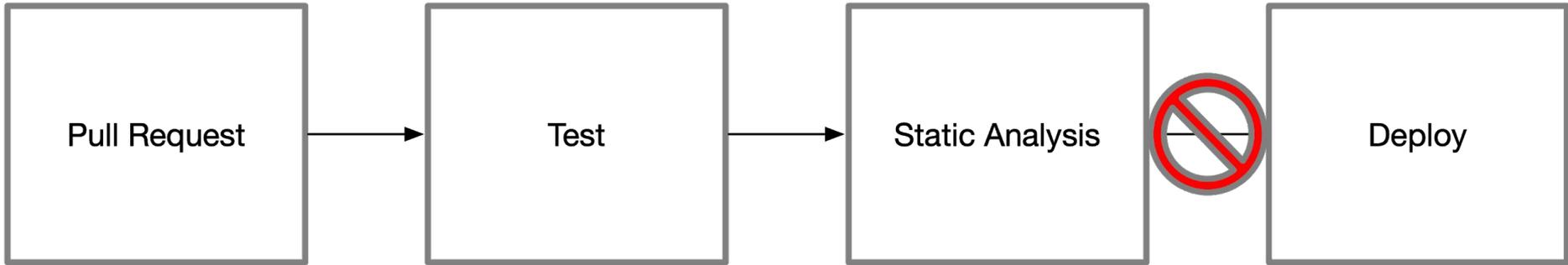
Incident impact increasing

When things aren't getting better,
they are getting worse

How to fix?







Ok, so what's the firefighting playbook?

Finish something

Reduce concurrent work

Automate

Eliminate categories of problems

Are you seeing signs of progress?

No? You've gotta hire

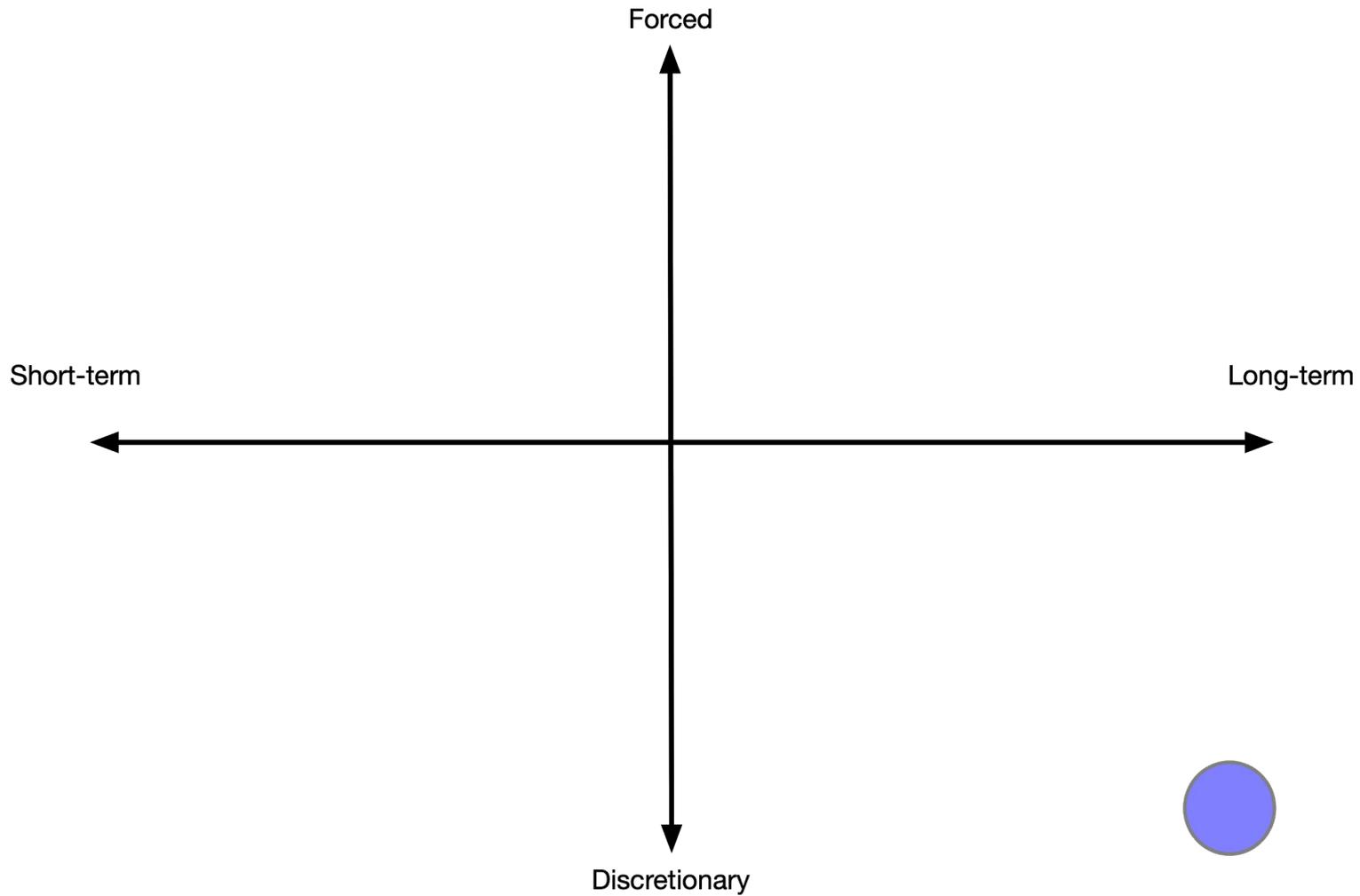
Once there's progress, stay the course!

btw, don't fall in love with firefighting

Introduction

1. Fundamentals
2. Escaping the firefight
- 3. Learning to innovate**
4. Navigating breadth
5. Unifying approach

Closing



Rare opportunity in infrastructure

Rare also means inexperienced

tl;dr

Talk to your users more

tl;dr

~~Talk~~ to your users more

tl;dr

Listen to your users more

Ways innovation goes wrong...

Problem

Making the most intuitive fix

Problem

AKA fixating on your local maxima

Discover

Discover

Benchmark with peer companies

Coffee chats with users

SLOs

Surveys

“Ruby is a terrible language.”

Sorbet:

A Typechecker for Ruby

Dmitry [@darkdimius](#) Petrashko
Nelson [@nelhage](#) Elhage
Paul [@ptarjan](#) Tarjan



Problem

Infinite possibilities, what to pick?

Prioritization

Prioritization

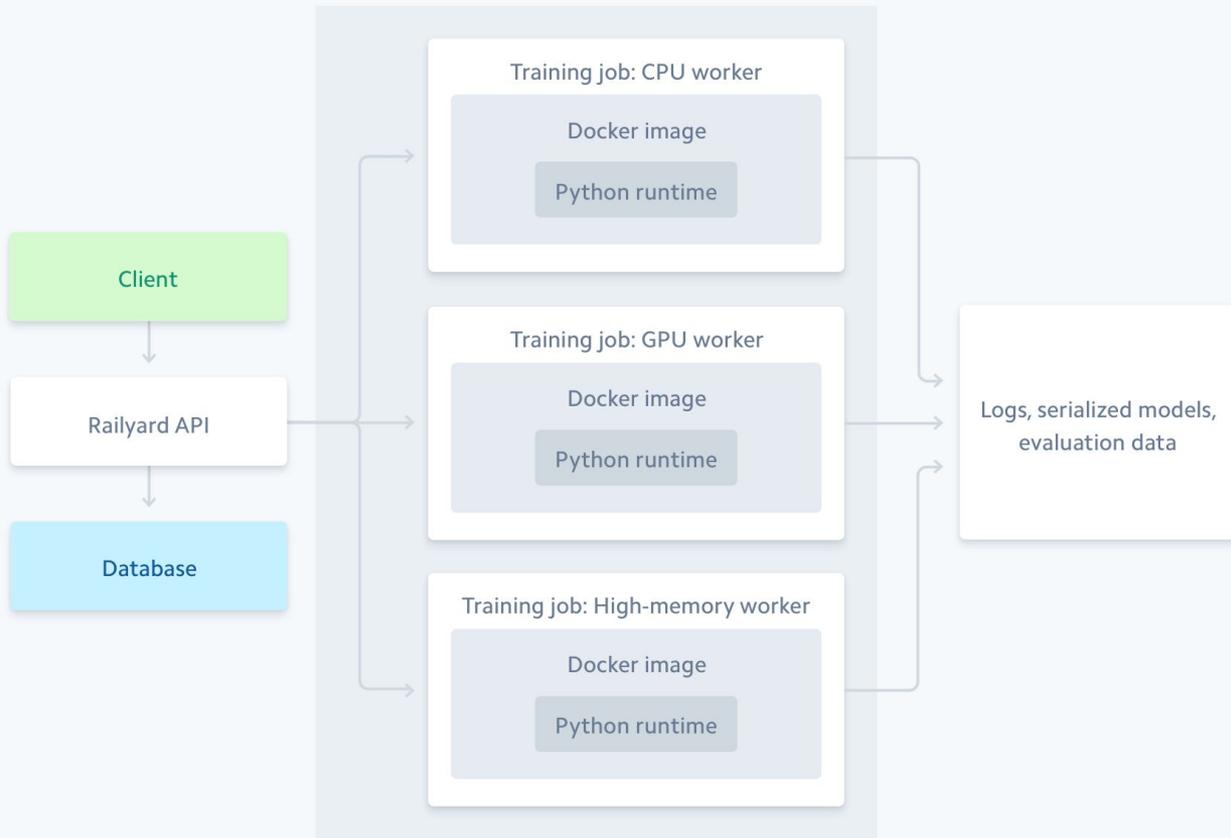
Order by return on investment

Don't try without users in the room

Long-term vision

“The critical business outcome is me learning Elixir.”

KUBERNETES



Problem

Right opportunity with wrong solution

Validation

Validation

Cheaply disprove approach

Try hardest cases early

Embed with owners

“Monster is too unreliable and slow!”

“Let’s just rewrite monster.”

“Let’s just rewrite monster. Again.”

“Let’s ~~just rewrite~~ harden monster.”

“Can we provide a unified interface for task, cronjob and service orchestration?”

Kubernetes

Kubernetes

Chronos

Railyard

Services

tl;dr

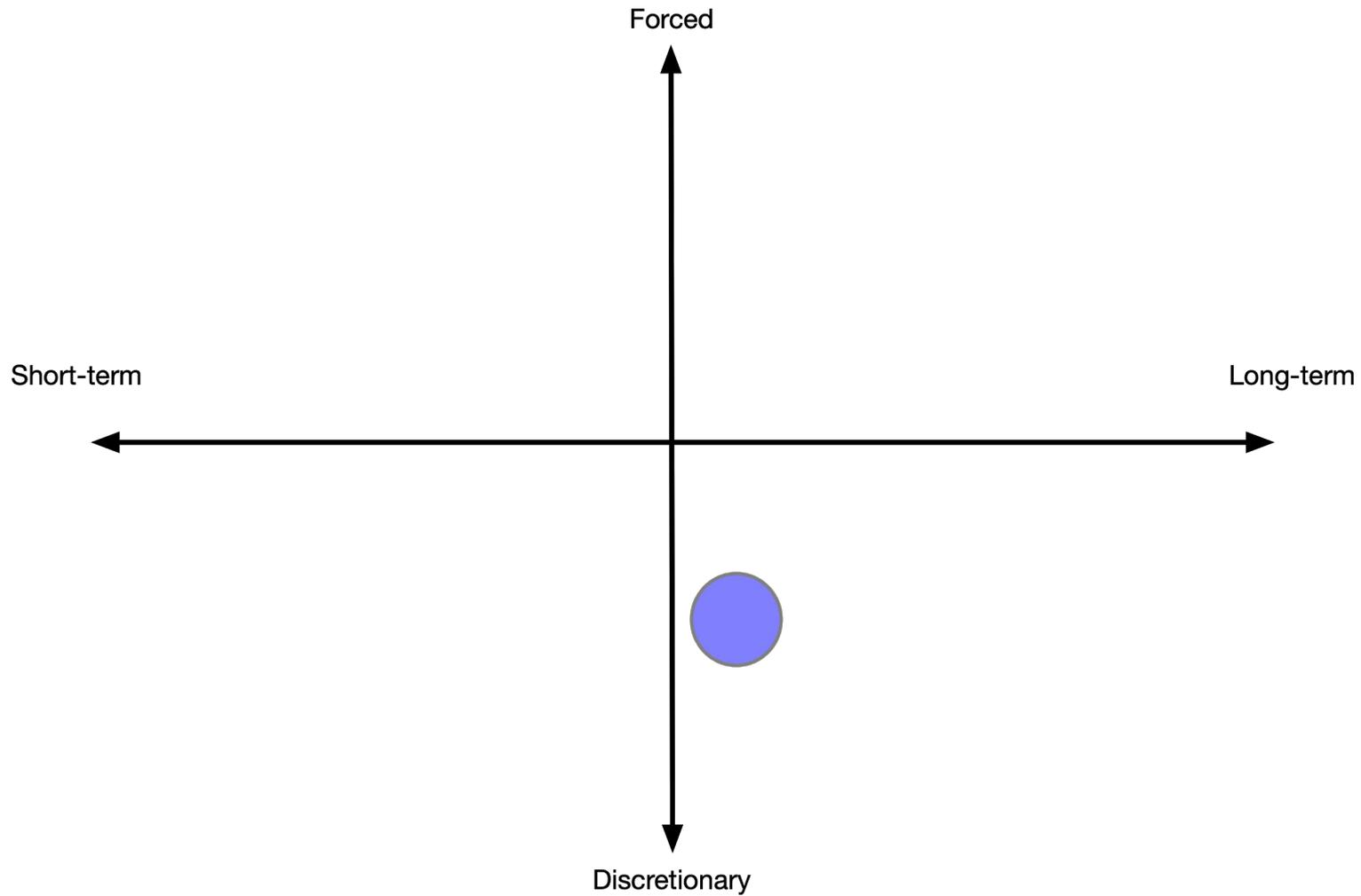
Listen to your users more

Be valuable or go back to firefighting

Introduction

1. Fundamentals
2. Escaping the firefight
3. Learning to innovate
- 4. Navigating breadth**
5. Unifying approach

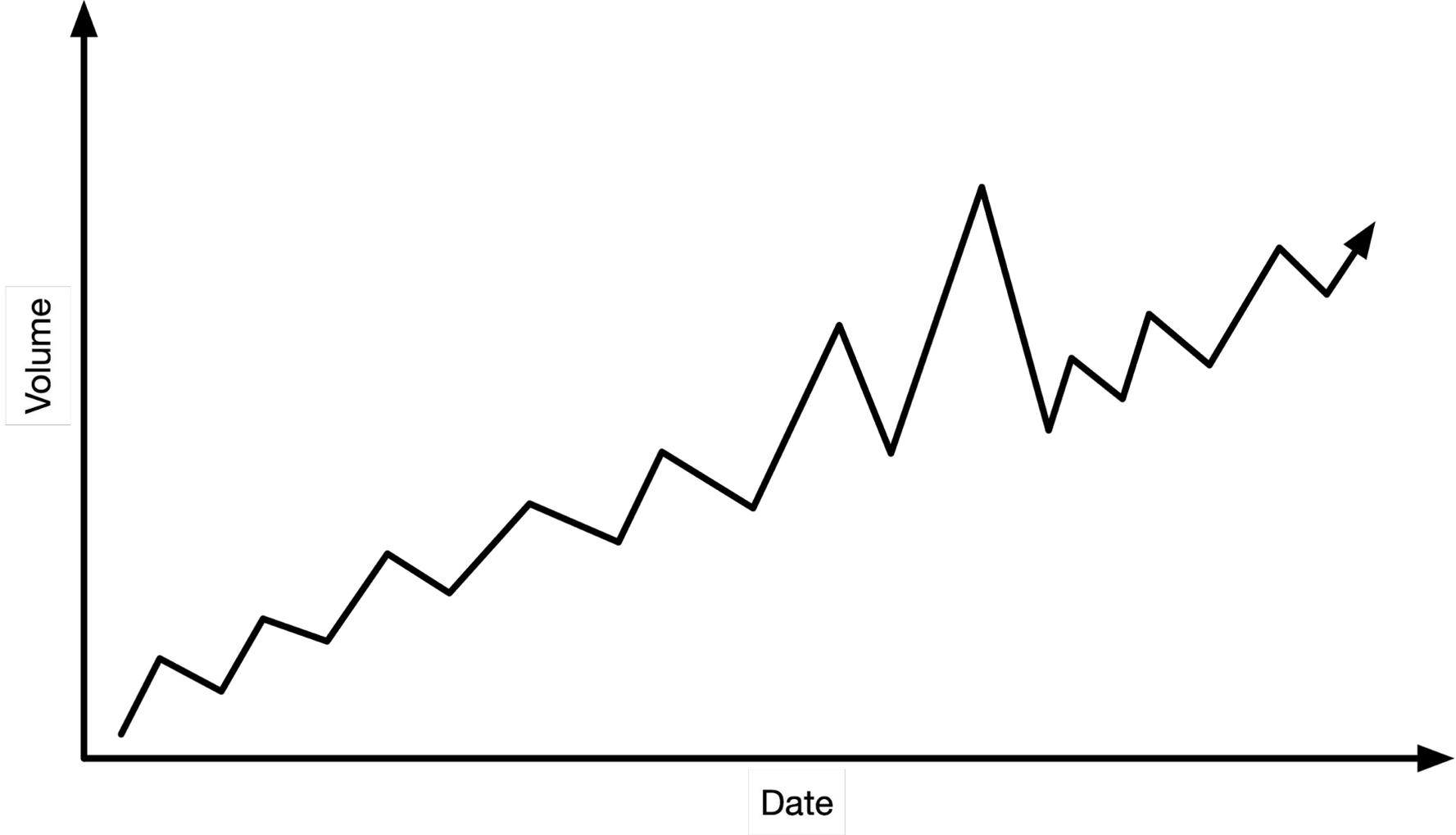
Closing

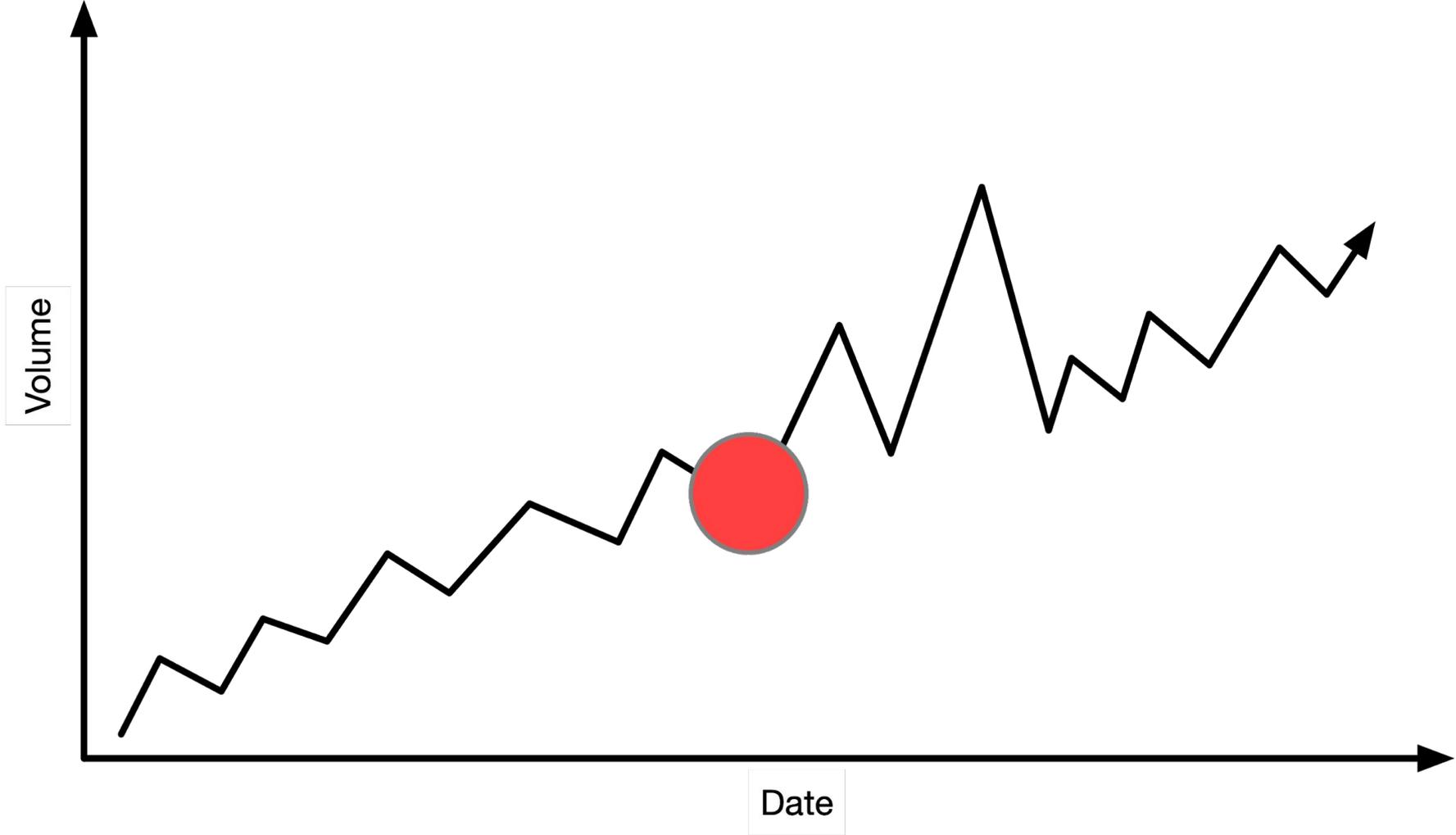


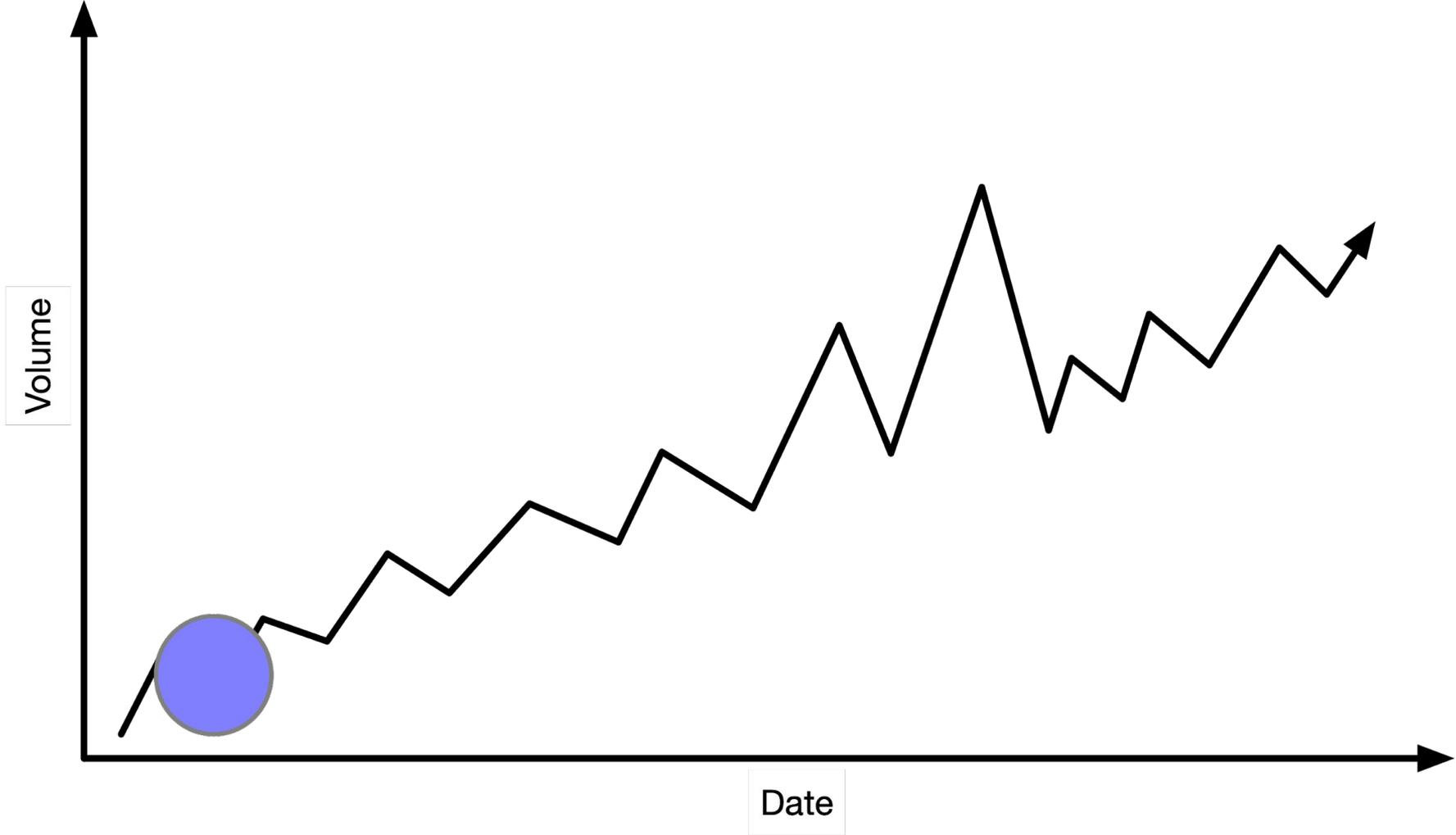
Fool me once, shame on you

Fool me twice, shame on me

Fool me every year on exact same date?







“Convert unplanned scalability work
into planned scalability work.”

Schedule manual load tests

Schedule automated load tests

Run continuous load tests

Solved out of a job

Great technology fix,
but what's the organizational fix?

Infrastructure properties

Stripe's infrastructure properties

Security

Reliability

Usability

Efficiency

Latency

Lightly ordered but not stack ranked

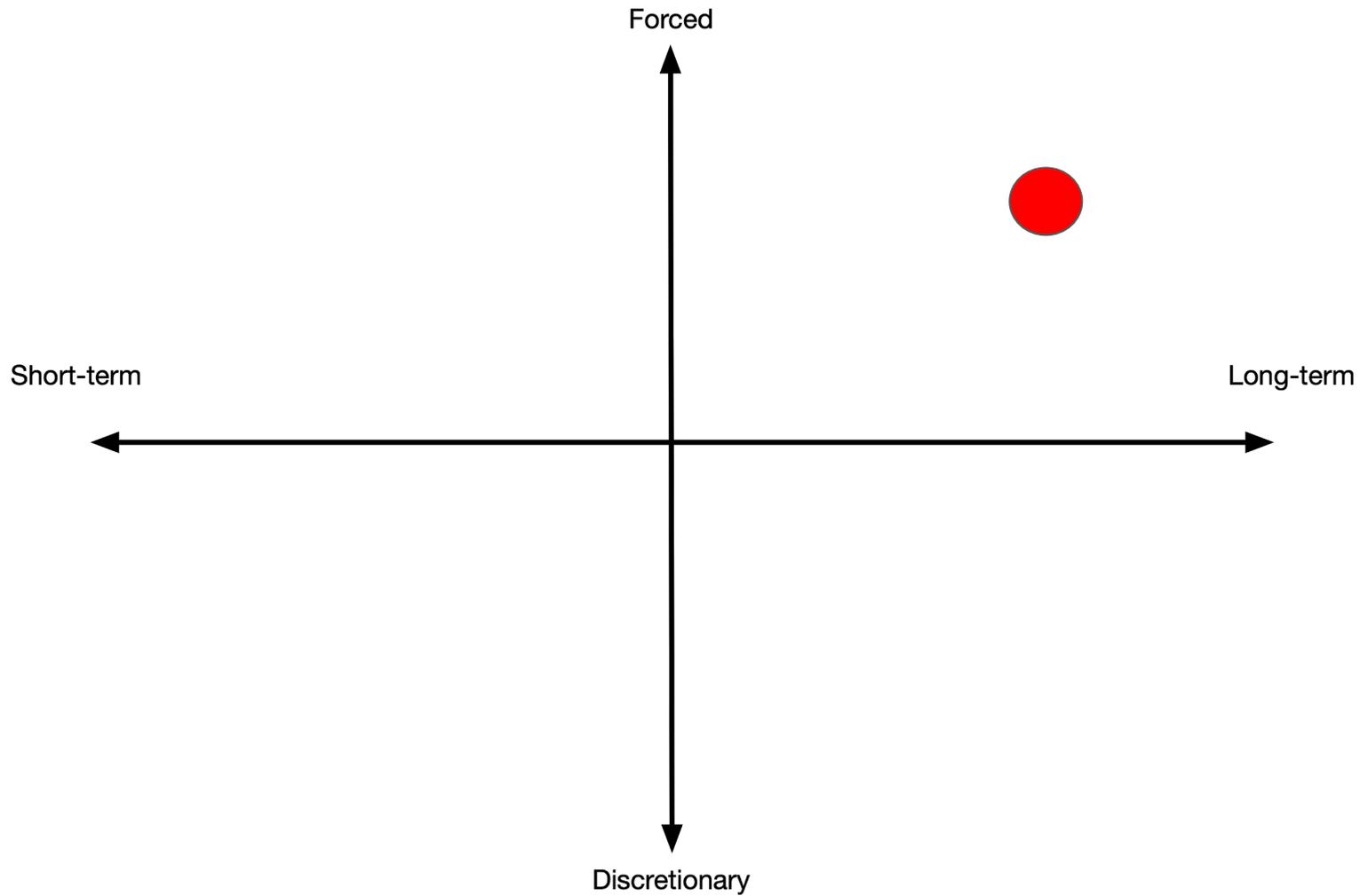
More a portfolio: invest in each

Baselines!

Invest to maintain your baselines

Maintain across timeframes

Long-term forced work!



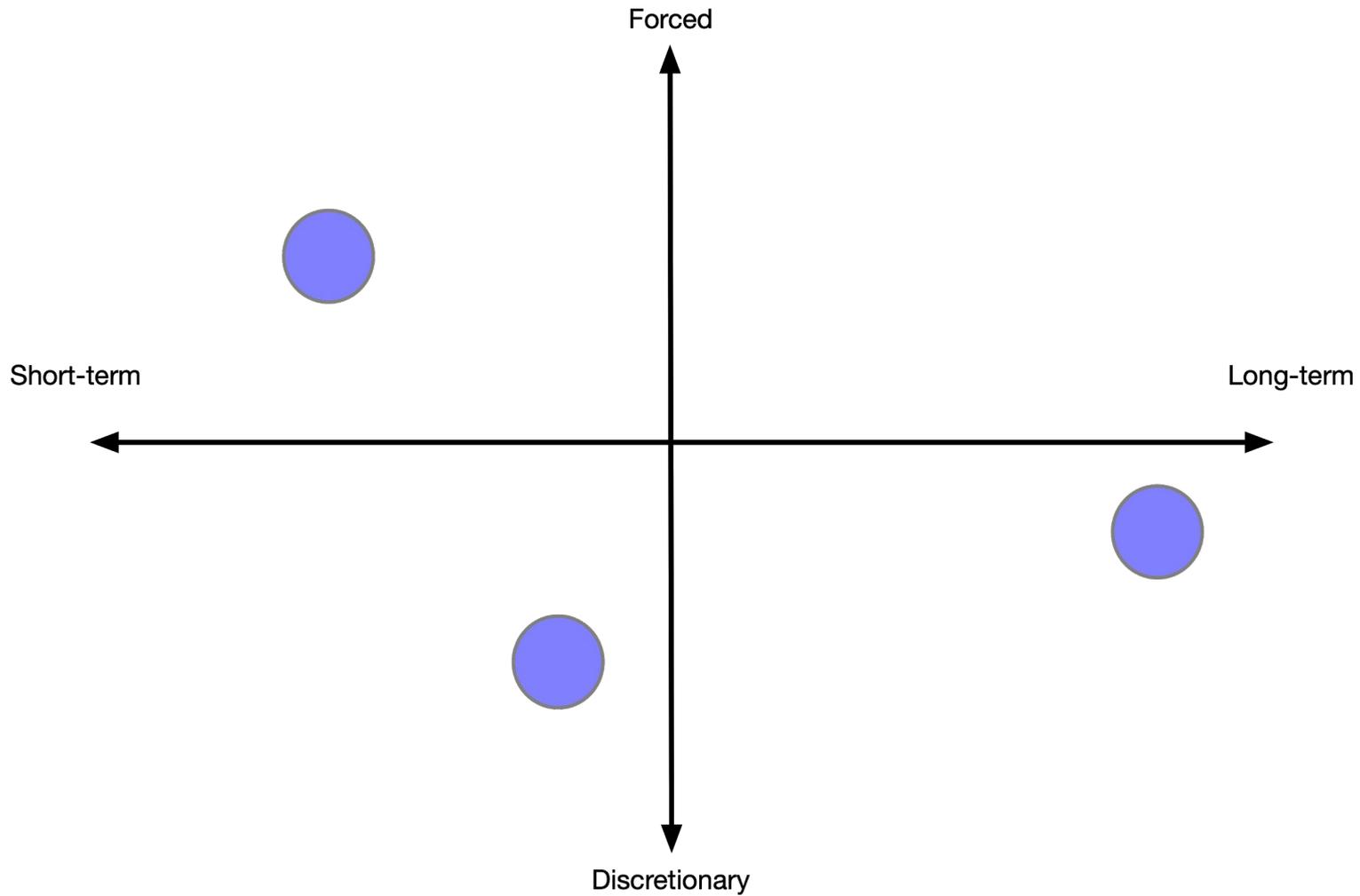
Do it now or firefight it later

Introduction

1. Fundamentals
2. Escaping the firefight
3. Learning to innovate
4. Navigating breadth
- 5. Unifying approach**

Closing

Wait... there's more than one team?



What we actually do today

Investment strategy

40% user asks

30% platform quality

30% “Key Initiatives”

40/30/30?

Solve from your constraints

Introduction

1. Fundamentals
2. Escaping the firefight
3. Learning to innovate
4. Navigating breadth
5. Unifying approach

Closing

Technical infrastructure:

Tools used by 3+ teams for business critical workloads.

Firefighting:

Limit work in progress.

Finish things.

If that's not enough, hire.

Innovation:

Listen to your users.

Listen to your users.

Listen to your users.

Navigating breadth:

Identify principles.

Set baselines.

Plan across timeframes.

Bring it together:

Investment strategy.

Users, baselines and timeframes.

Q&A

@lethain / lethain.com