Building an Async API with ASP.NET Core

UNDERSTANDING THE POWER OF ASYNC

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Prerequisites and Tooling

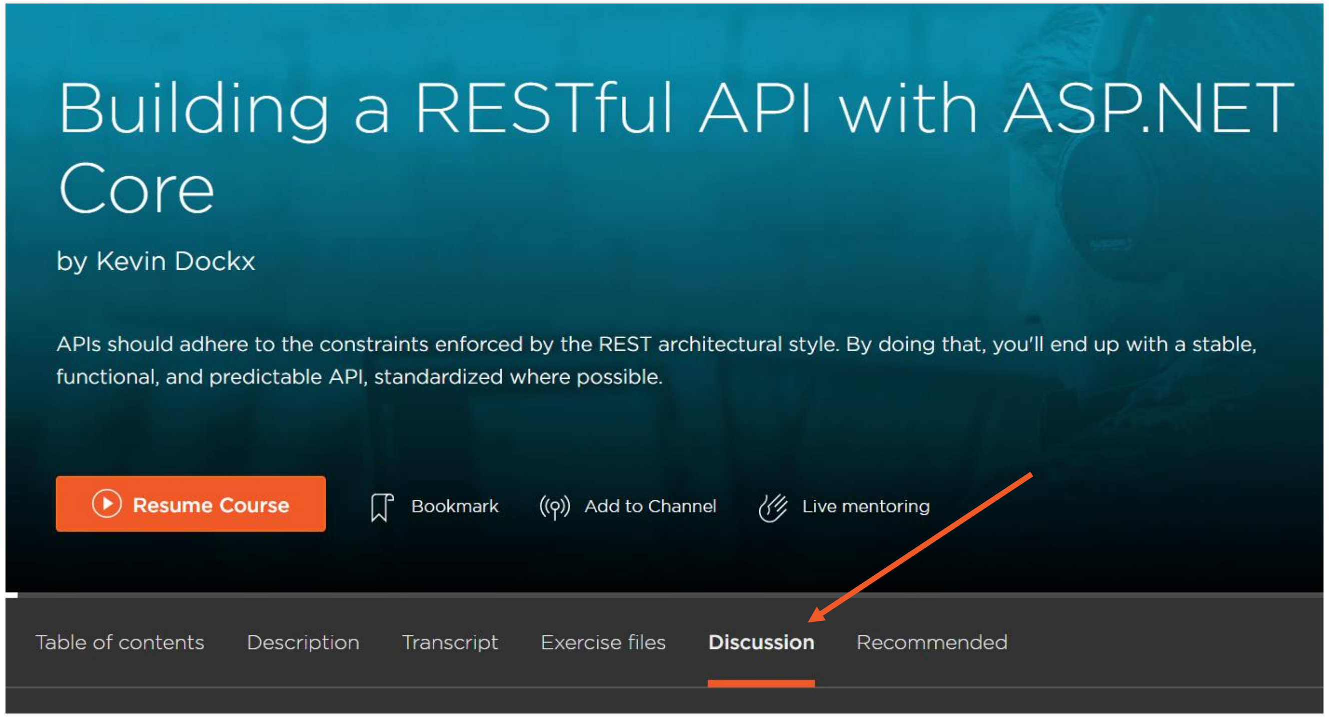
Sync and Async Request Handling

I/O Versus Computational Bound Work

Multithreading, Concurrency, and Parallelism

Discussion tab on the course page

Twitter: @KevinDockx


Building a RESTful API with ASP.NET Core

by Kevin Dockx

APIs should adhere to the constraints enforced by the REST architectural style. By doing that, you'll end up with a stable, functional, and predictable API, standardized where possible.

(Course shown is one of my other courses, not this one)

Course Prerequisites

Good knowledge of C#

Knowledge of building an API with ASP.NET Core

Course Prerequisites

ASP.NET Core Fundamentals (Scott Allen)

- https://bit.ly/2gg9WSH

Building Your First API with ASP.NET Core (yours truly)

- https://bit.ly/2GCkkFV

Frameworks and Tooling

Visual Studio 2017

http://bit.ly/2dSGoN5

Visual Studio for Mac

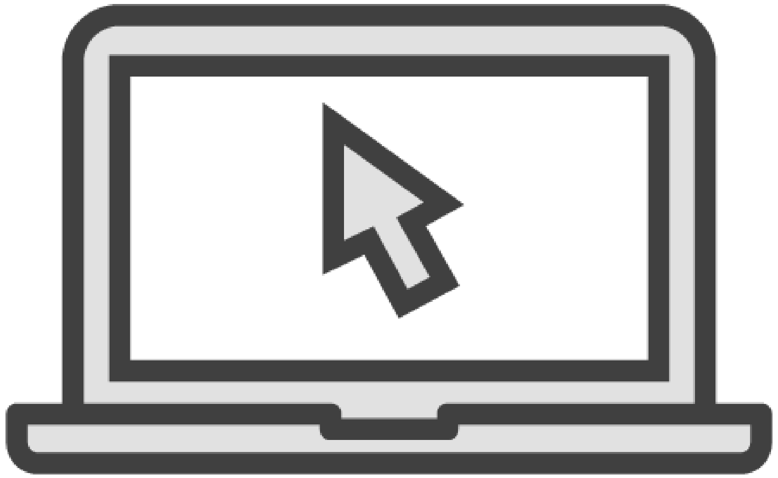
http://bit.ly/2fXmQpH

Visual Studio Code

http://bit.ly/1J6QrU6

JetBrains Rider,
Sublime, …

Introducing the Demo Project



Library API, built with ASP.NET Core 2.1

- Start from scratch

Exercise files

- Exercise files tab on the course page
- GitHub: https://bit.ly/2ONREdi

The Advantage of Asynchronous Code

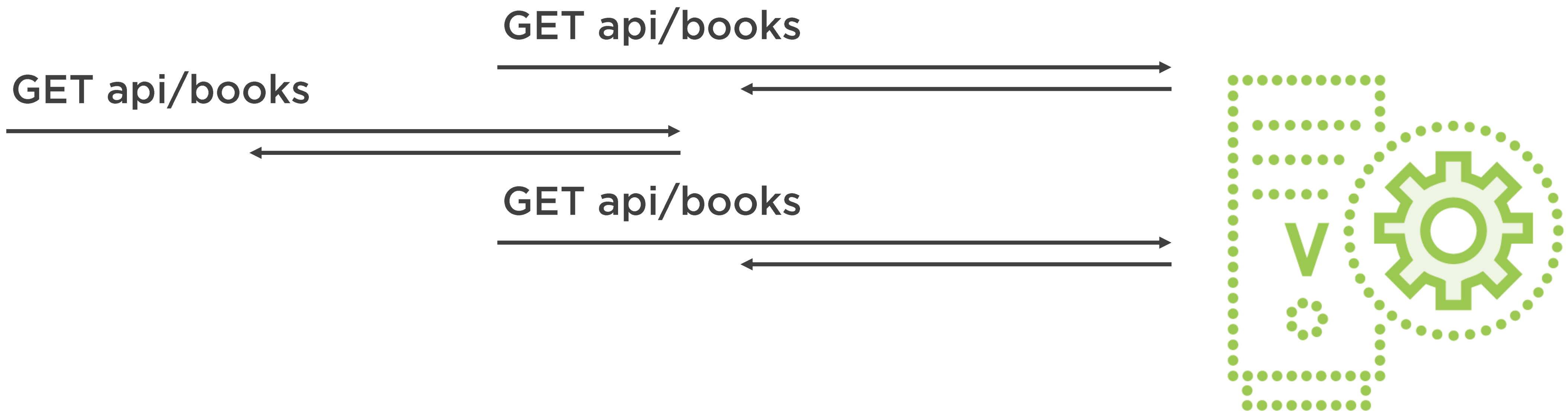
Performance is not the key benefit

The key benefit of writing async server-side code is increased scalability

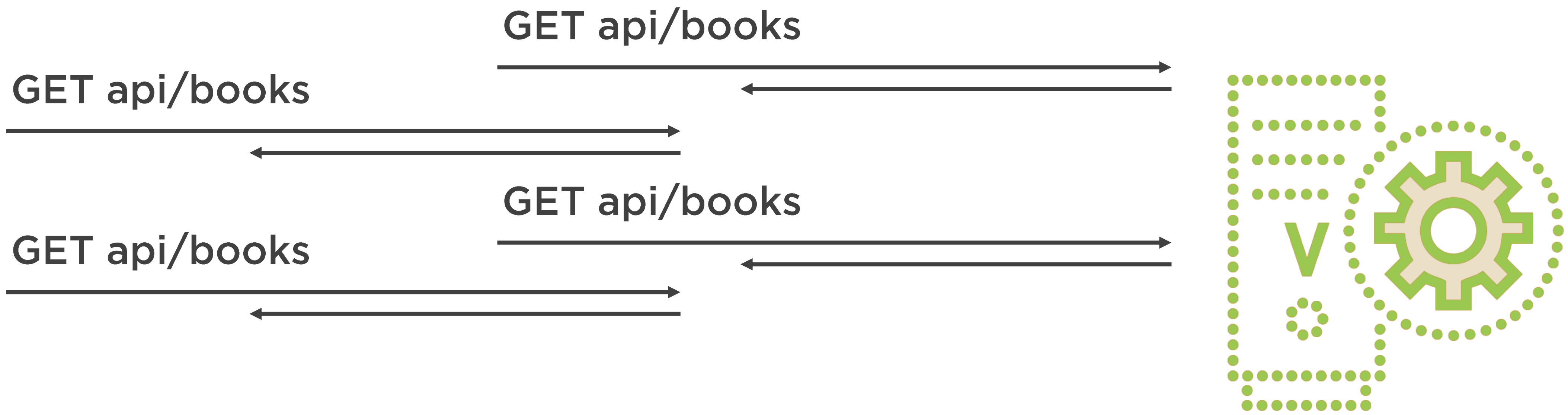
Scalability

The capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth

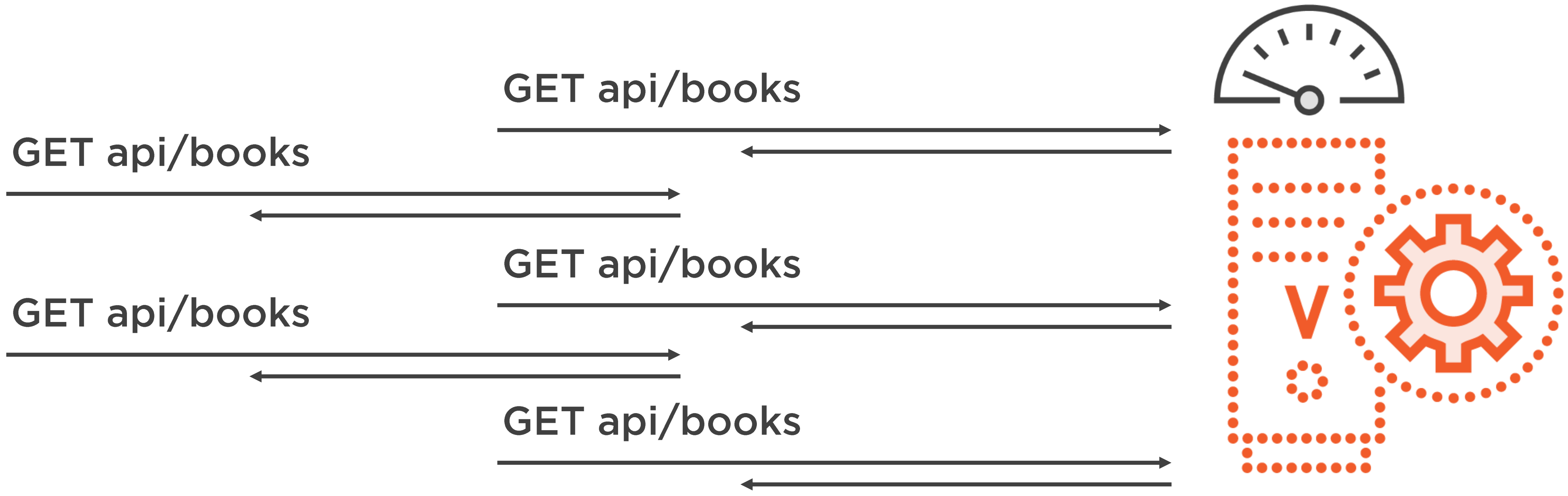
Horizontal Scaling



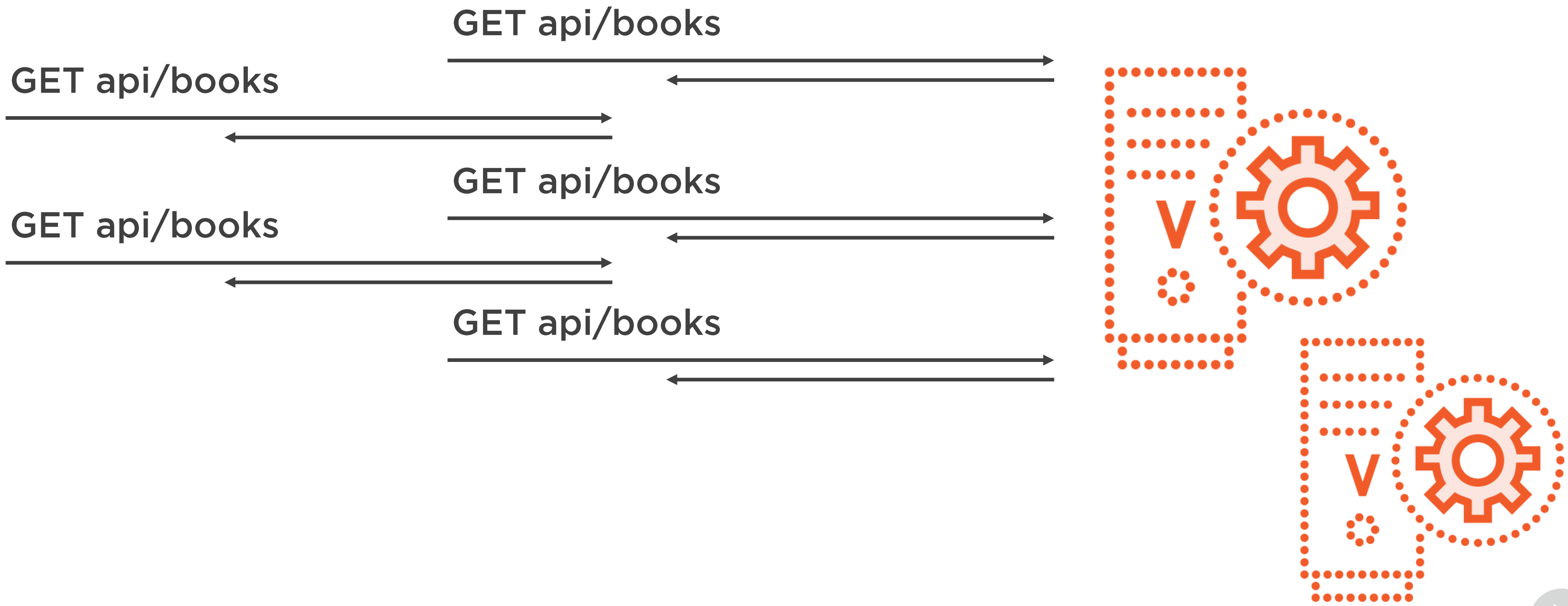
Horizontal Scaling



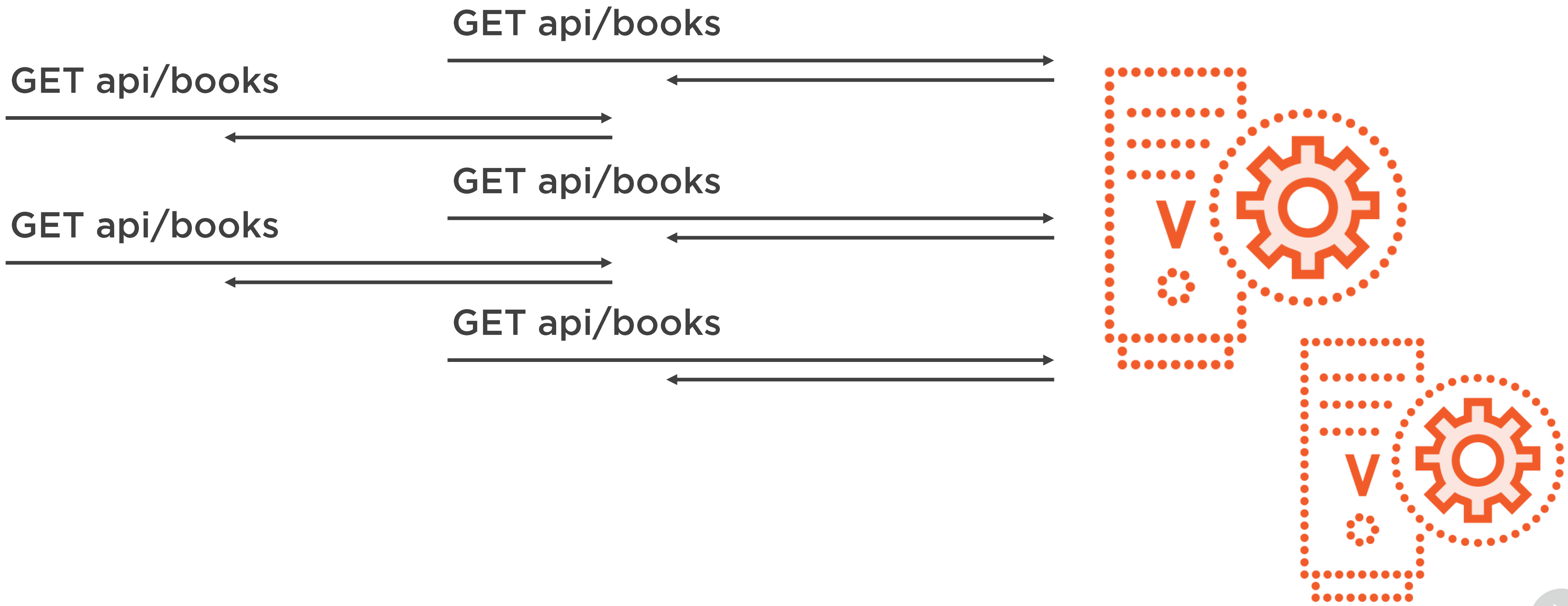
Horizontal Scaling



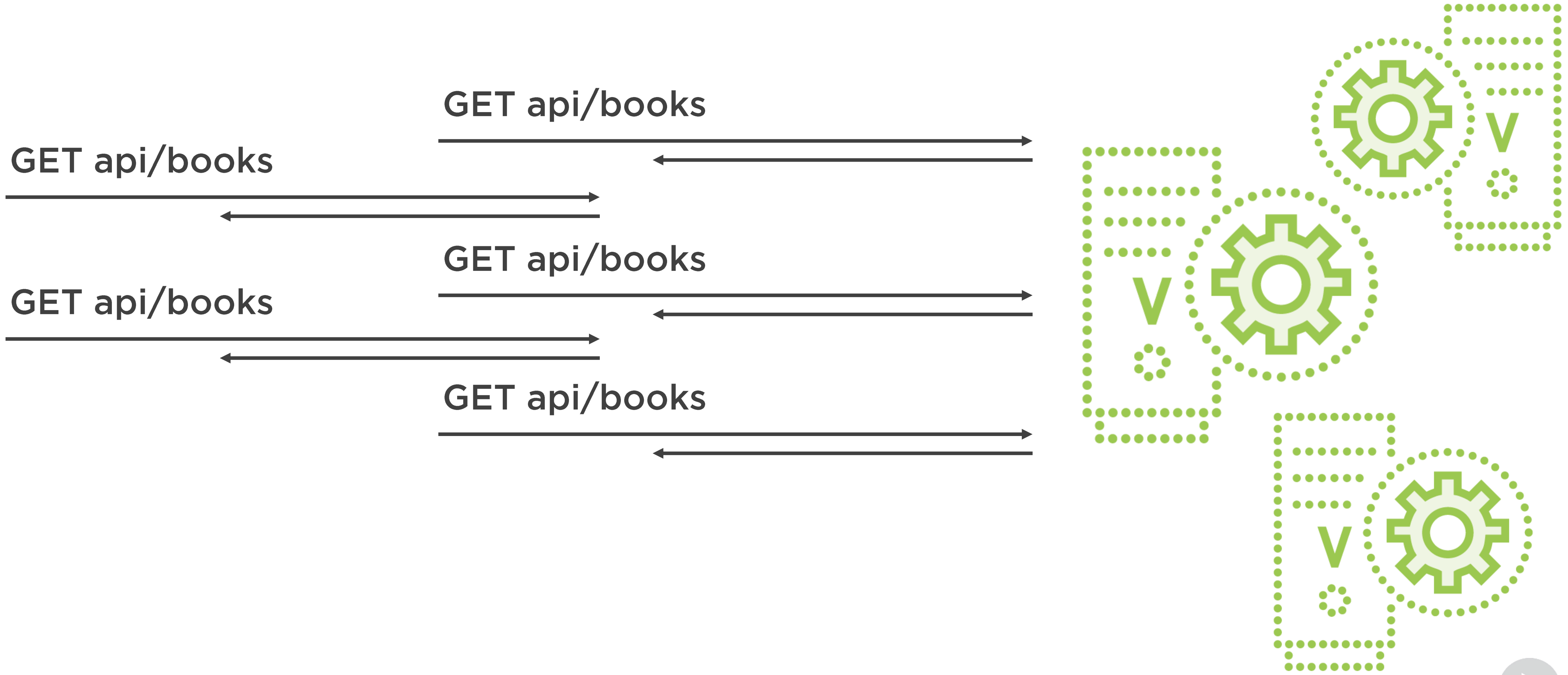
Horizontal Scaling



Horizontal Scaling



Horizontal Scaling



Horizontal Scaling

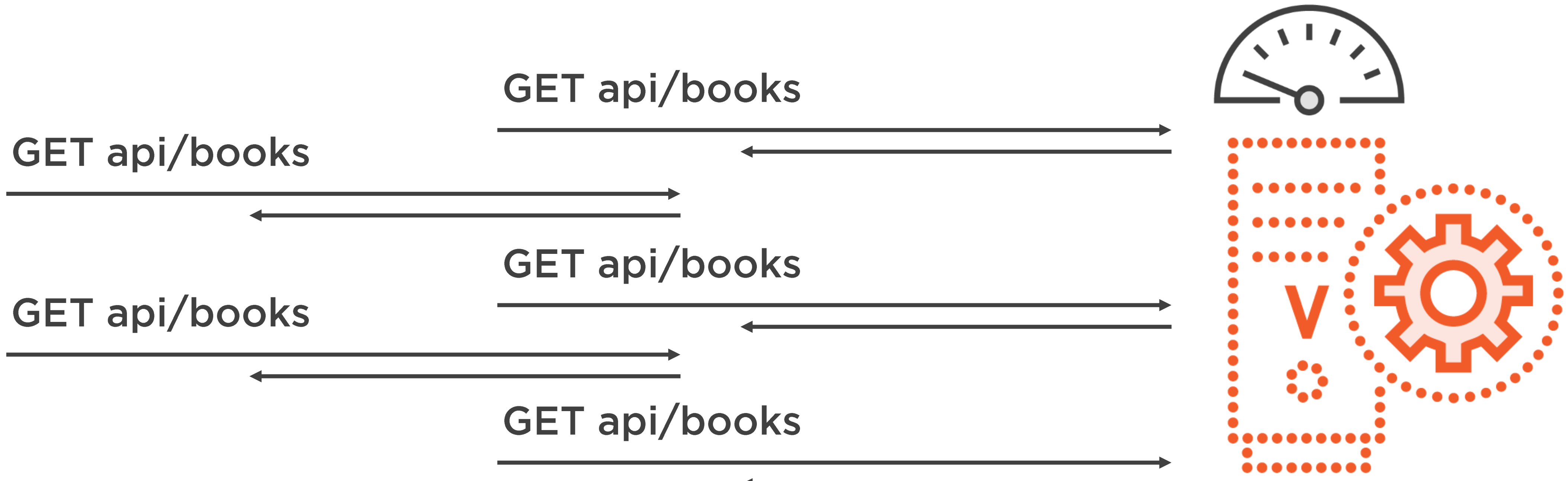
One way of increasing scalability is by writing an API in a way that can accommodate horizontal scaling

- RESTful systems are a good start

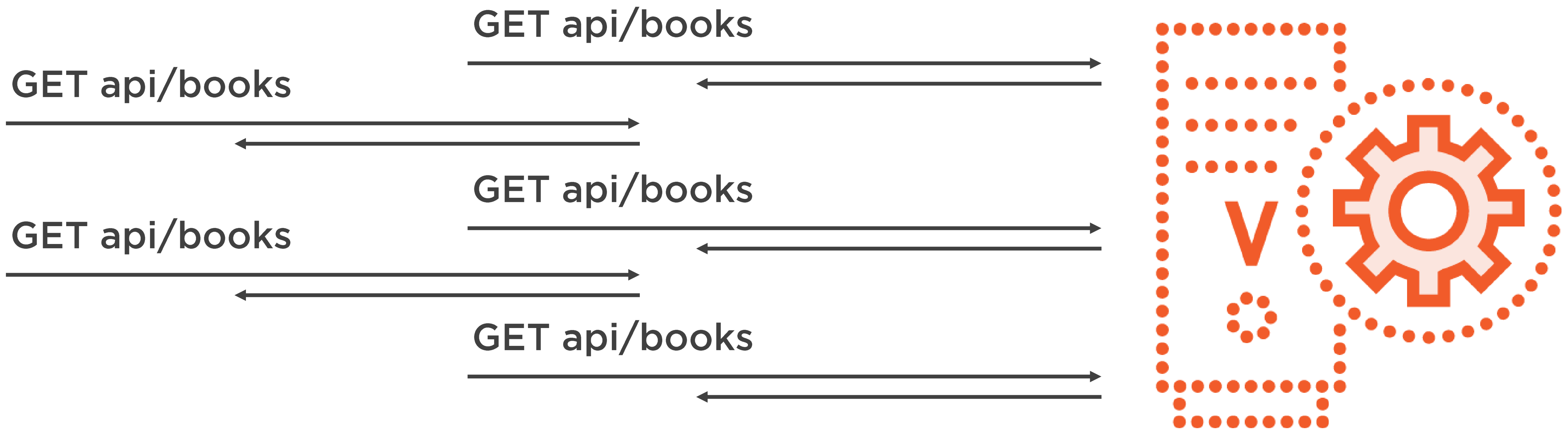
Other components can still hurt scalability

- Non-distributed databases or caches

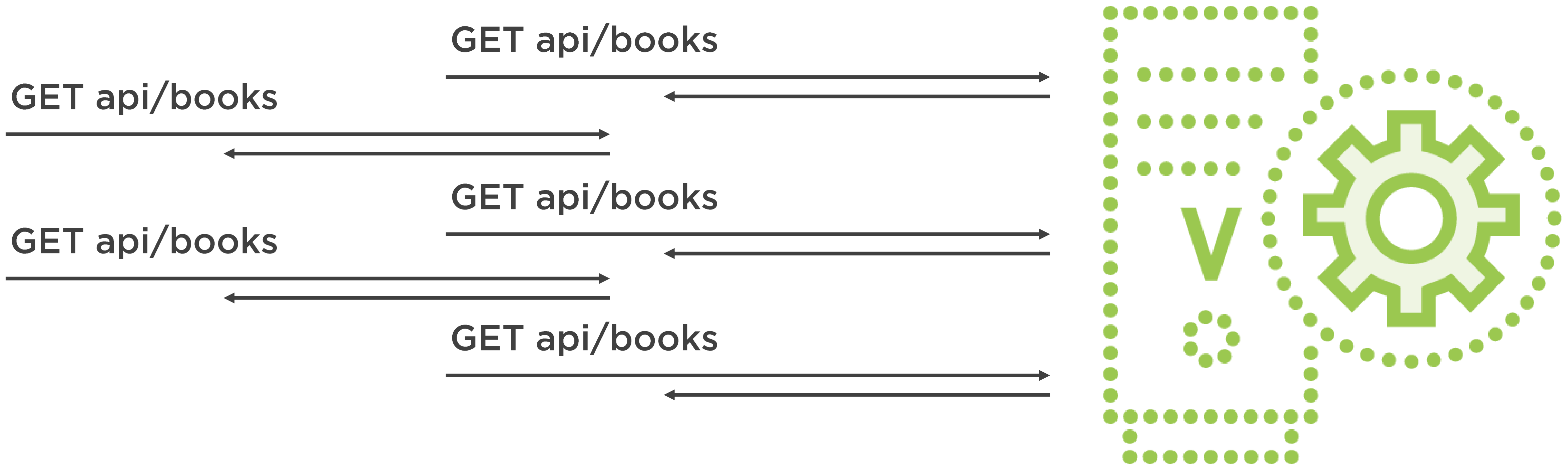
Vertical Scaling



Vertical Scaling



Vertical Scaling

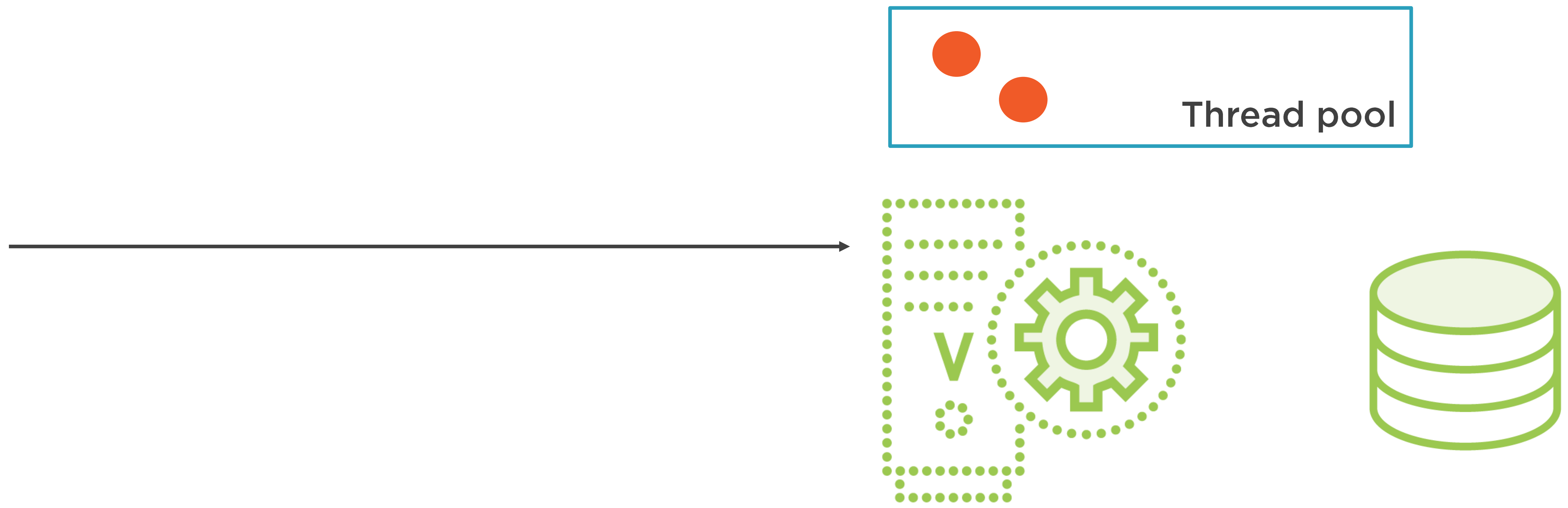


Vertical Scaling

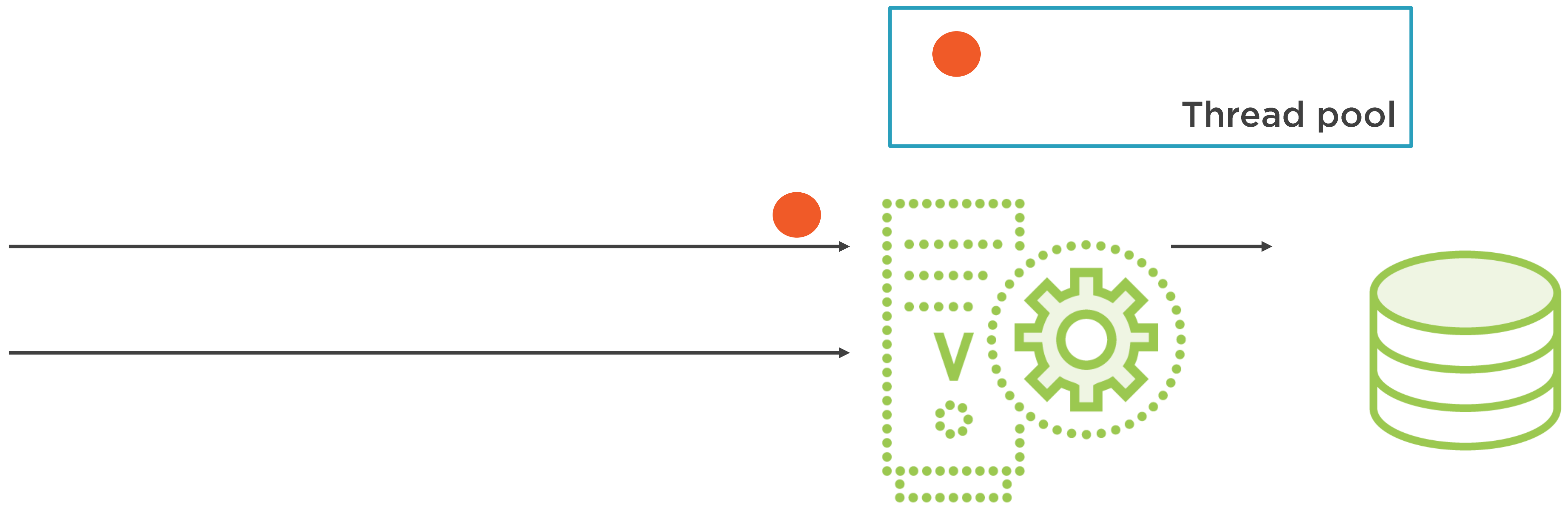
Another way of increasing scalability is by writing an API in such a way that resource utilization is improved

Writing async code helps with improving the vertical scalability at server level

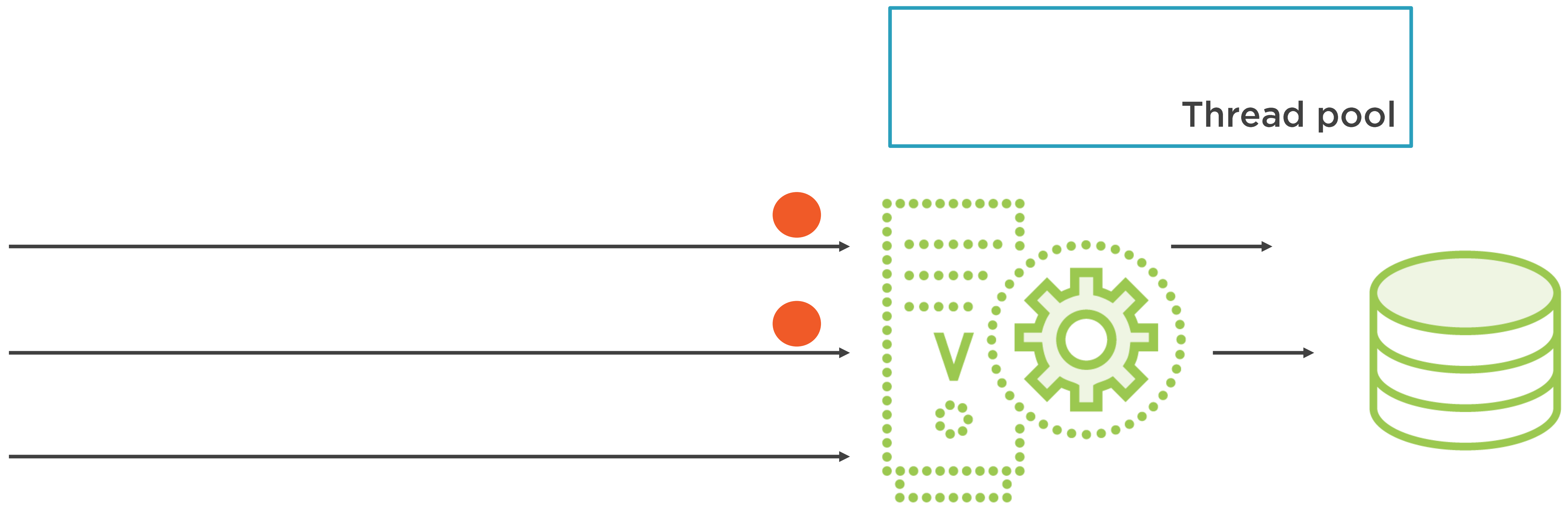
Handling Synchronous Requests



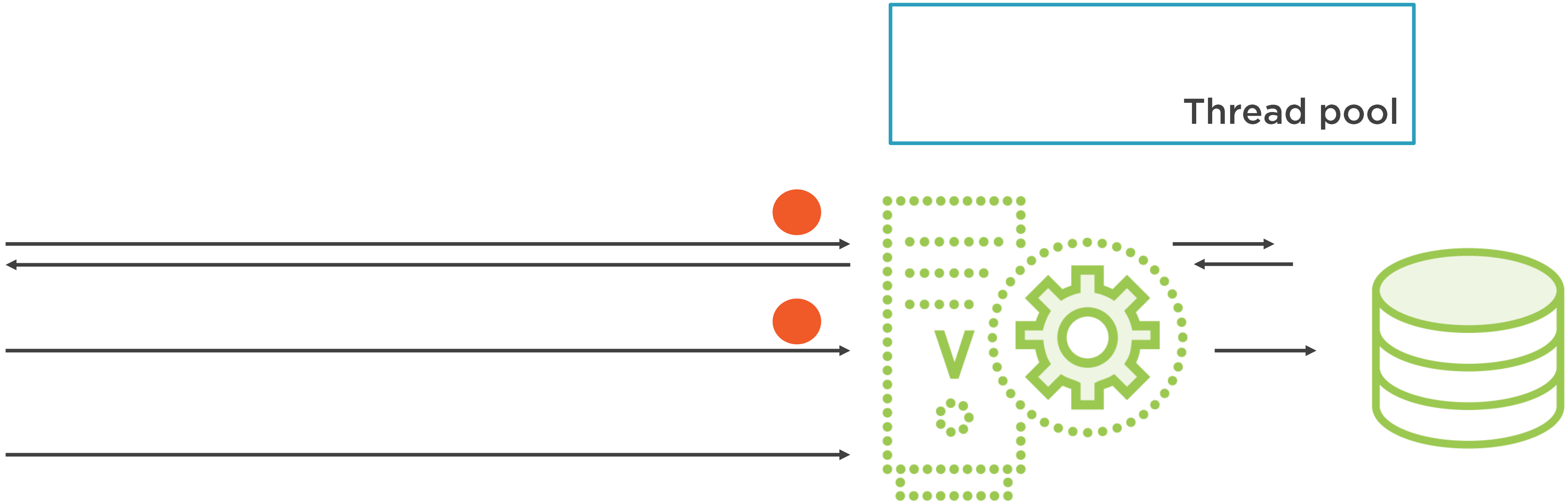
Handling Synchronous Requests



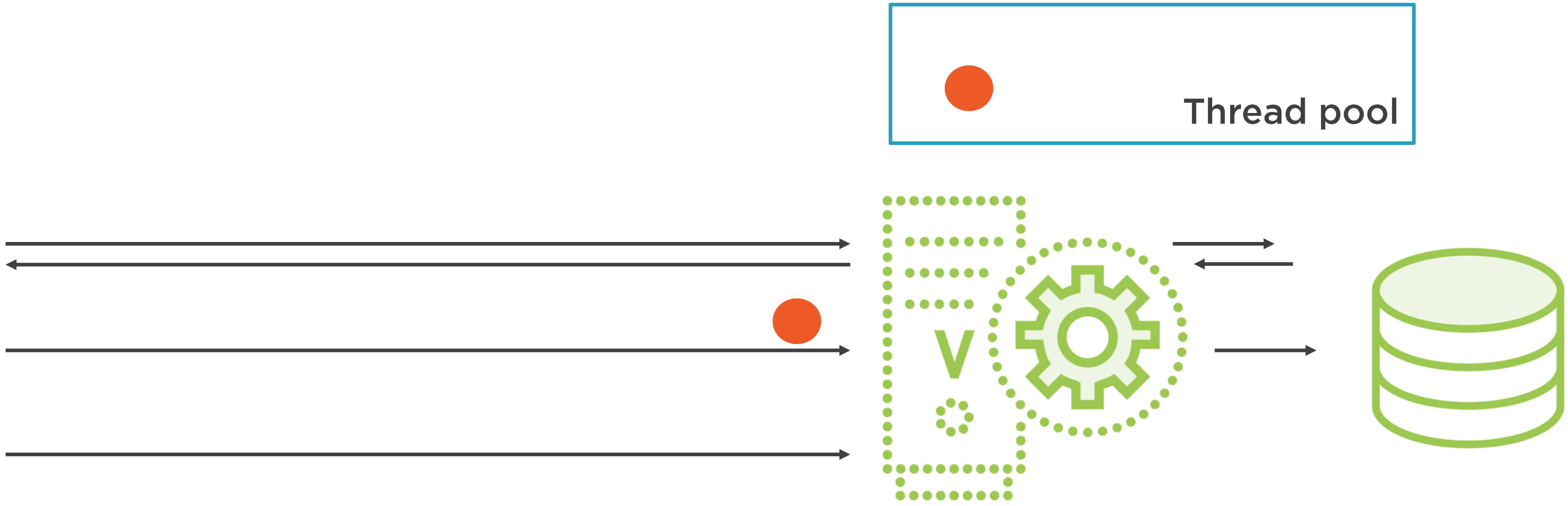
Handling Synchronous Requests



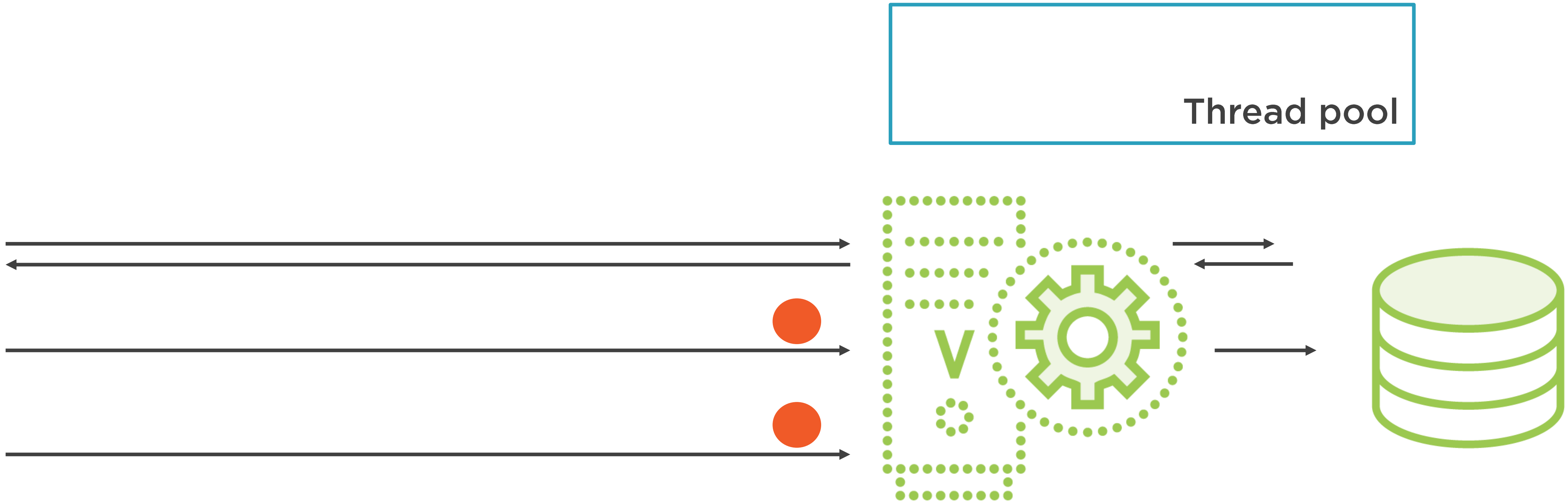
Handling Synchronous Requests



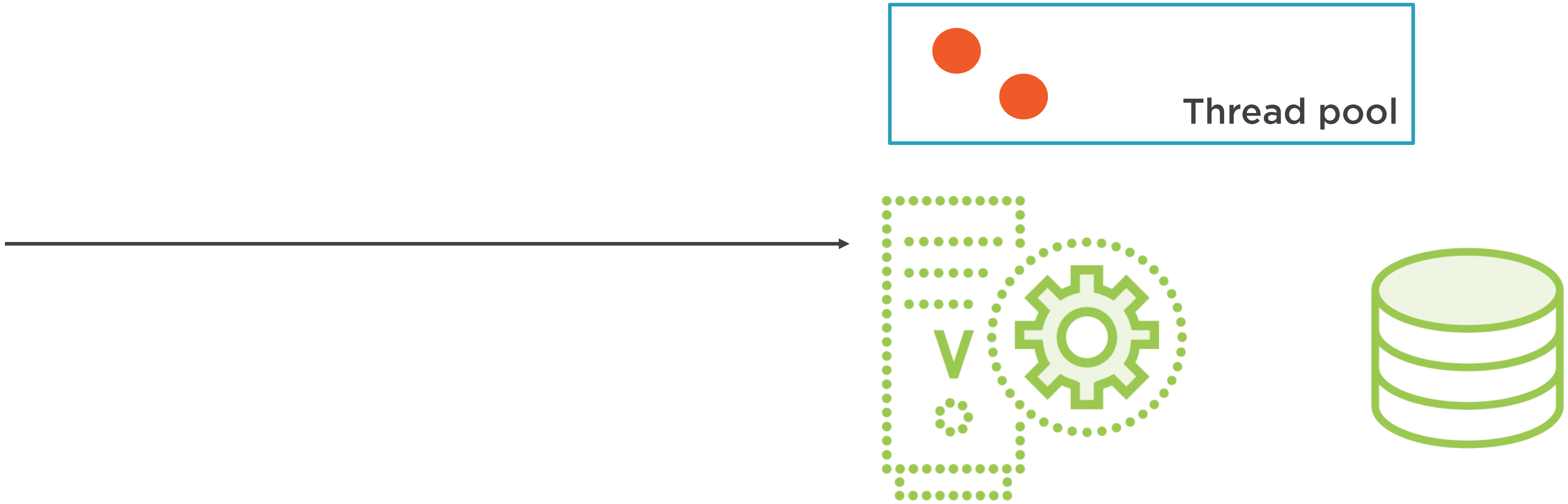
Handling Synchronous Requests



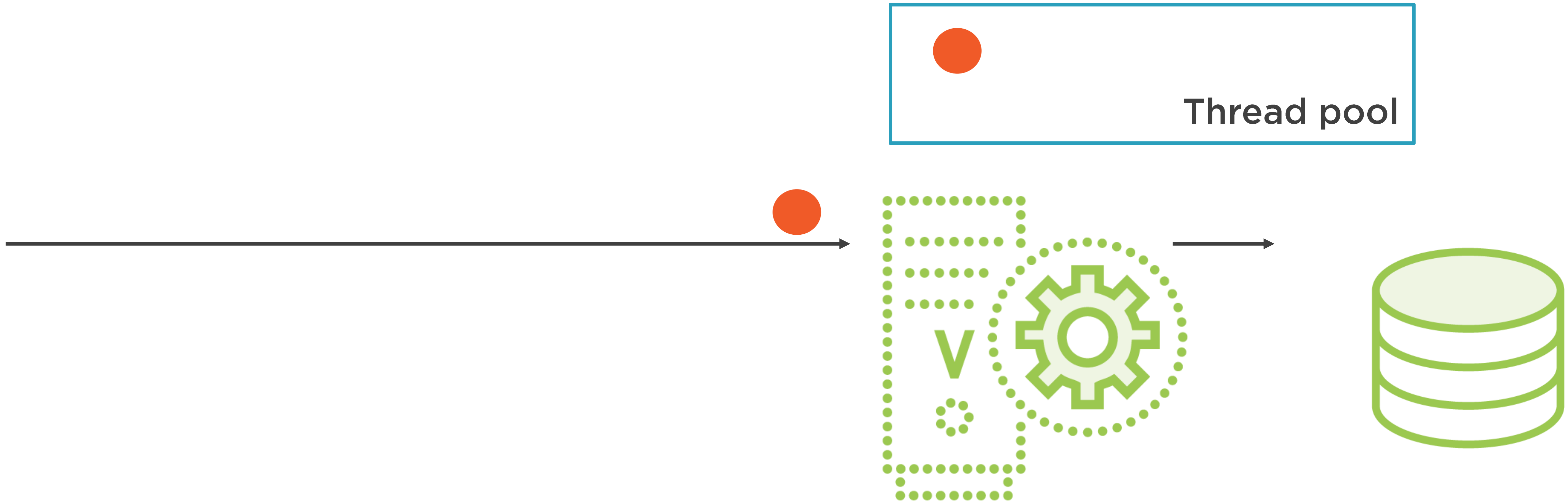
Handling Synchronous Requests



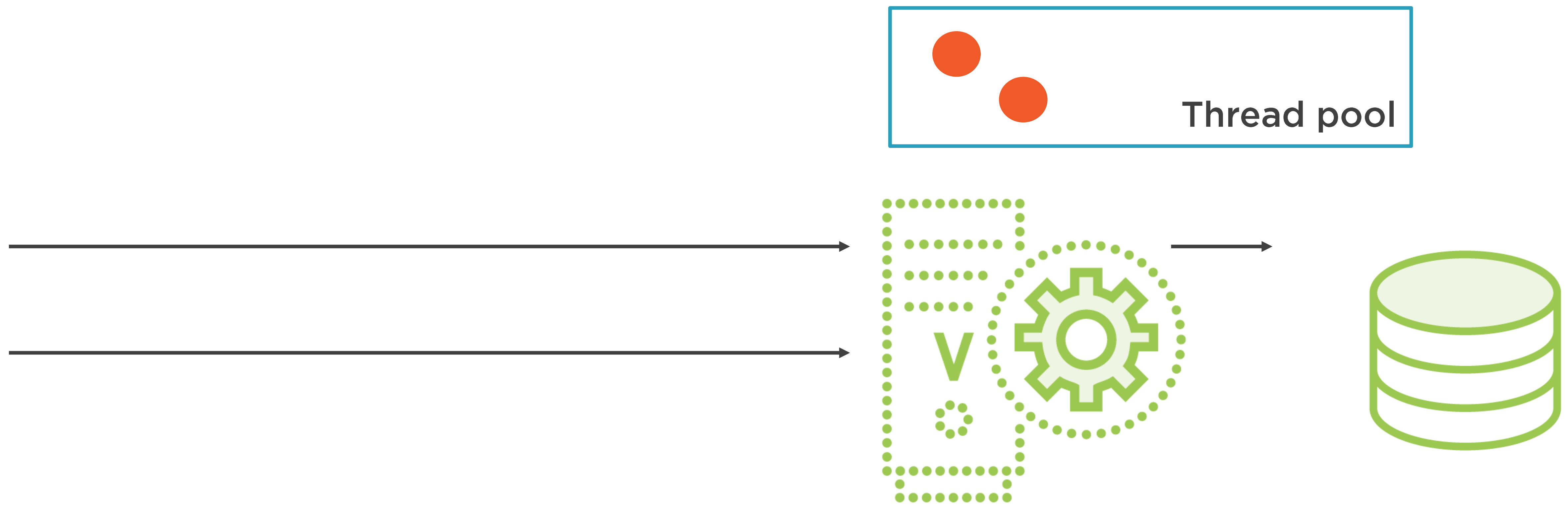
Handling Asynchronous Requests



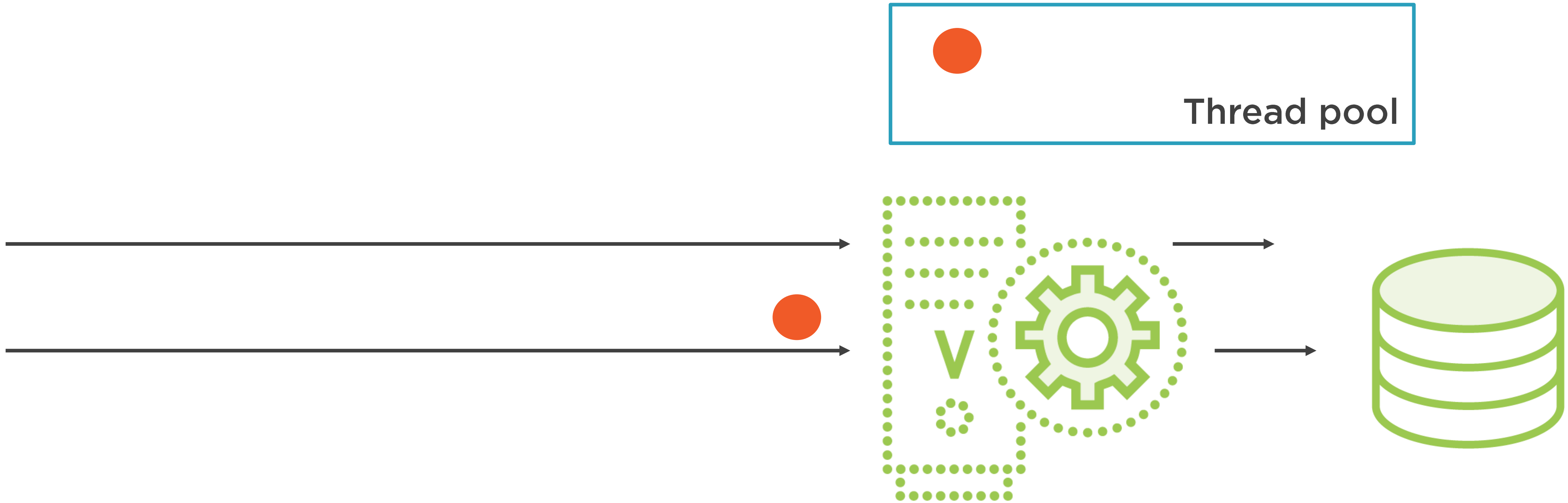
Handling Asynchronous Requests



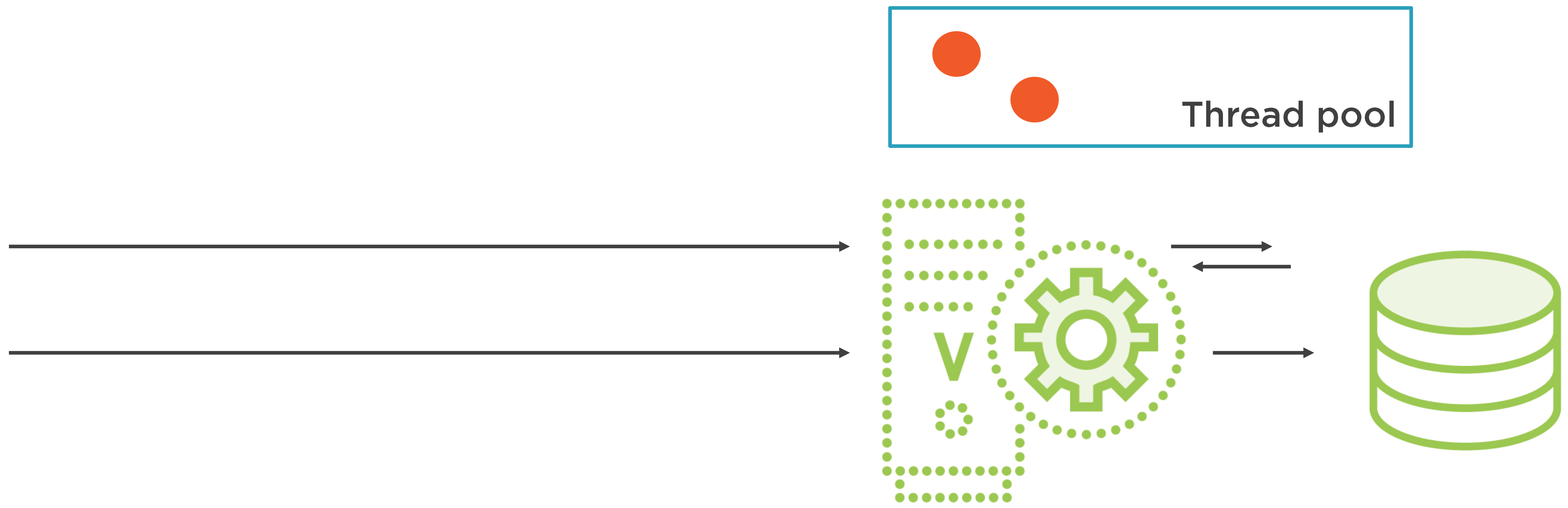
Handling Asynchronous Requests



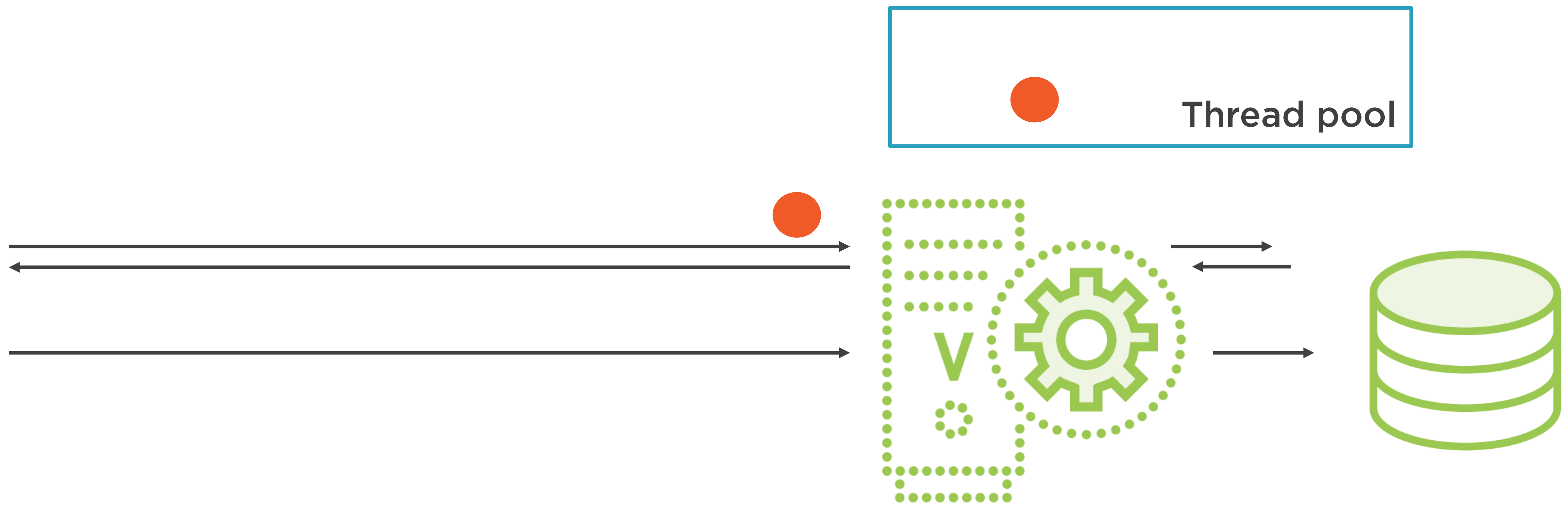
Handling Asynchronous Requests



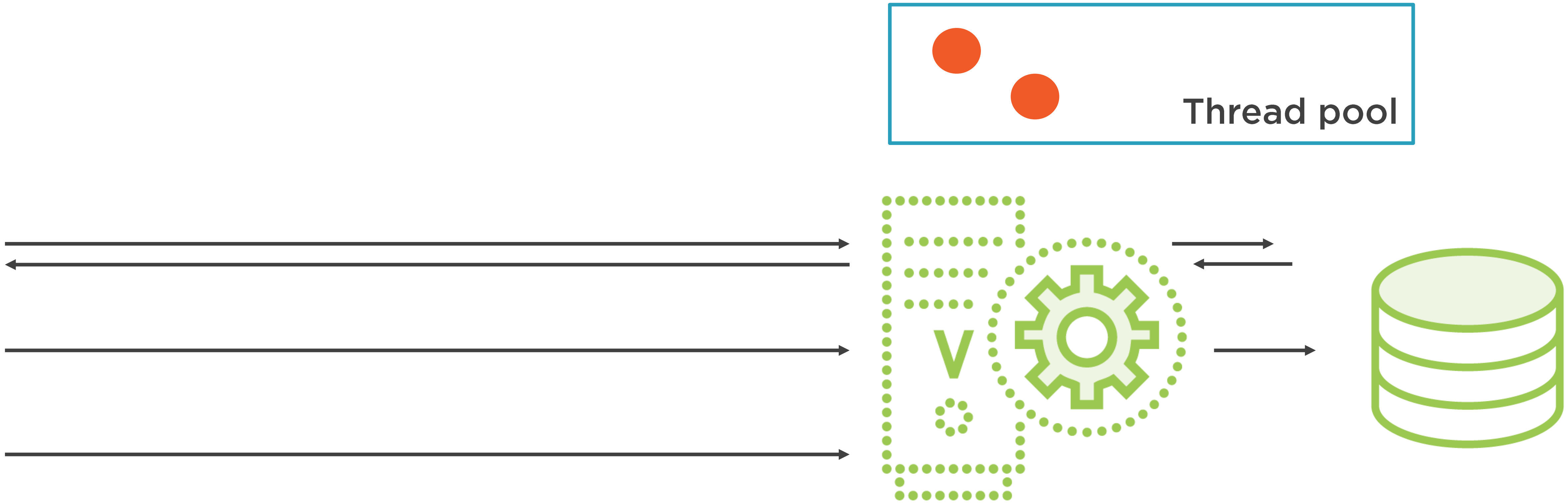
Handling Asynchronous Requests



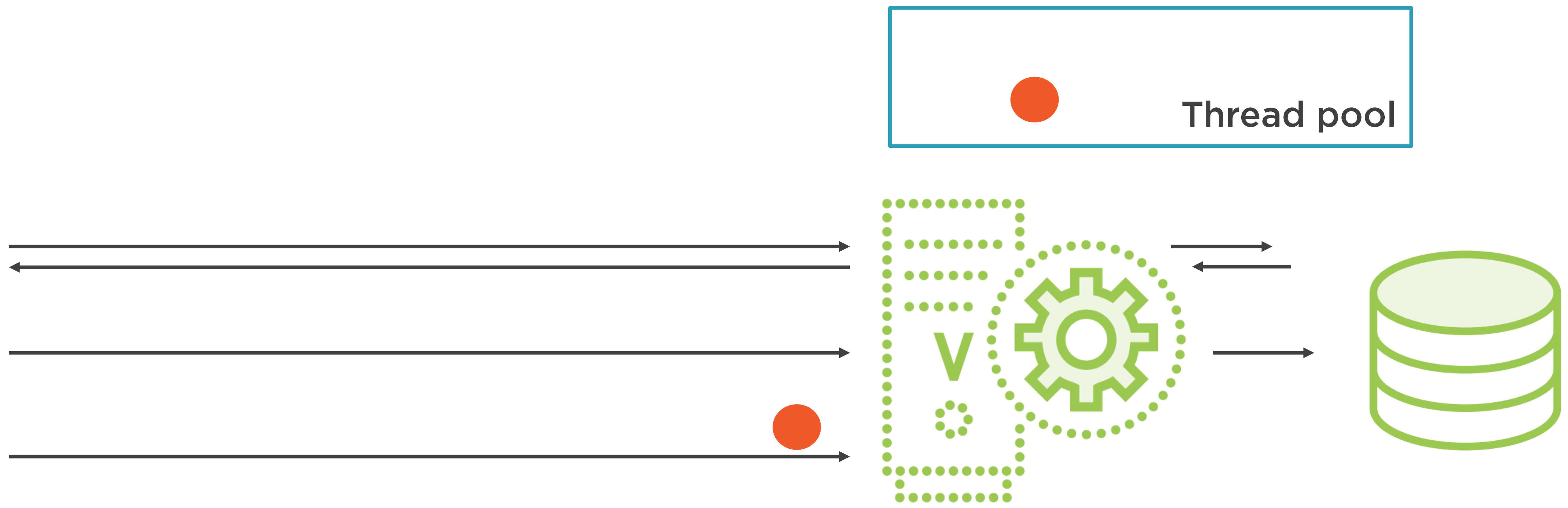
Handling Asynchronous Requests



Handling Asynchronous Requests



Handling Asynchronous Requests



I/O Bound vs. Computational-bound Work

I/O bound work

"Will my code be waiting for a task to be complete before continuing?"

File system, database, network calls

Server-side and client-side

Computational-bound work

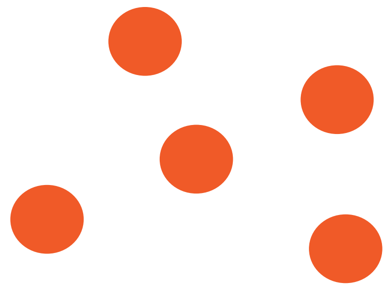
"Will my code be performing an expensive computation?"

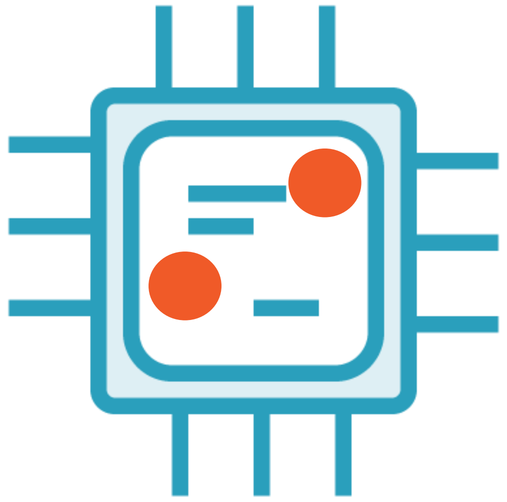
Expensive business algorithm

Client-side

Don't use async on the server for computational-bound work

Threads, Multithreading, Concurrency, Parallelism





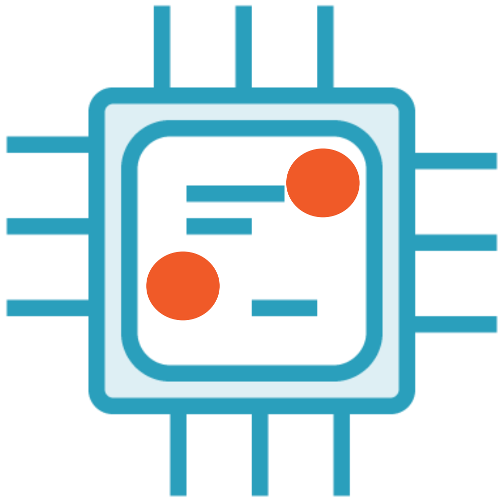
A thread is...

a basic unit of CPU utilization

Multithreading means that...

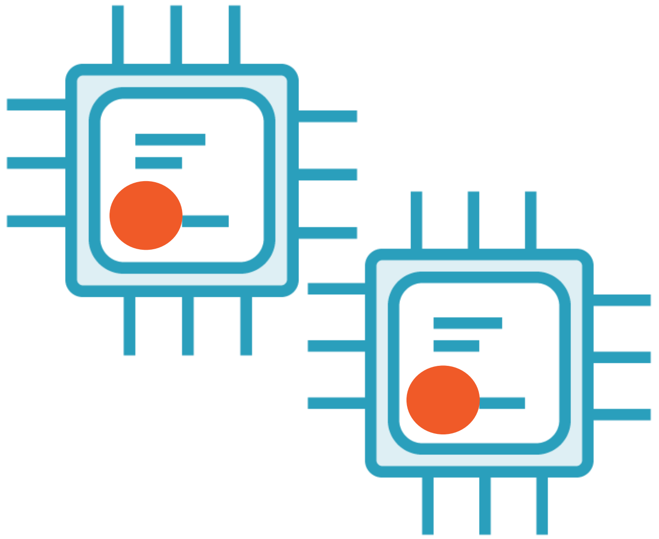
one single CPU or single core in a multi-core CPU can execute multiple threads concurrently

Threads, Multithreading, Concurrency, Parallelism



Concurrency is…

a condition that exists when at least two threads are making progress



Parallelism means that…

at least two threads are executing simultaneously

Summary

Use async on the server to increase scalability

- The thread that's handling an async request is freed up to handle other requests
- It doesn't wait idly for an I/O operation to finish

Summary

Use async on the server for I/O bound work

- Eg: file system, network, database requests

Don’t use async on the server for computational bound work

- Eg: long-running calculations
- Might have adverse effects
- Can be used on the client