Developing an Asynchronous ASP.NET Core Web API

Understanding the Power of Asynchronous Code

Kevin Dockx
Architect

@KevinDockx | www.kevindockx.com

Version Check

This version was created by using:

- ASP.NET Core 8.0
- .NET 8.0
- Visual Studio 2022

Version Check

This course is 100% applicable to:

- ASP.NET Core 8.x
- .NET 8.x

Version Check

New course versions are regularly released:

- https://app.pluralsight.com/profile/author/kevin-dockx

Coming Up

Positioning this course

Course prerequisites and tooling

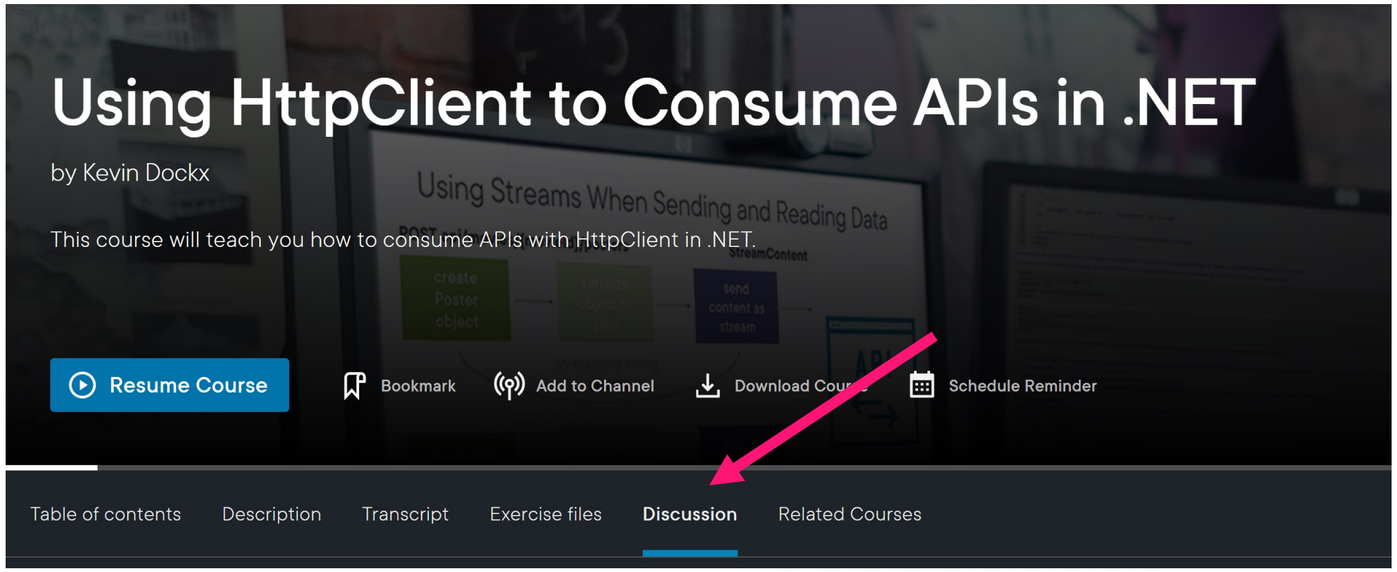
Synchronous and asynchronous request handling

I/O-bound vs. computational-bound work

Threads, multithreading, concurrency, and parallelism

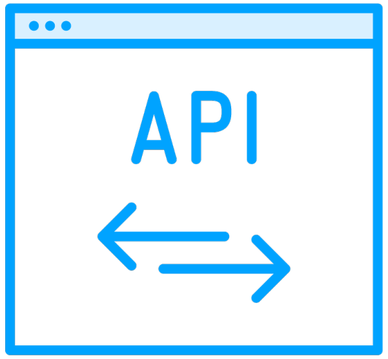
Discussion tab on the course page

X: @KevinDockx



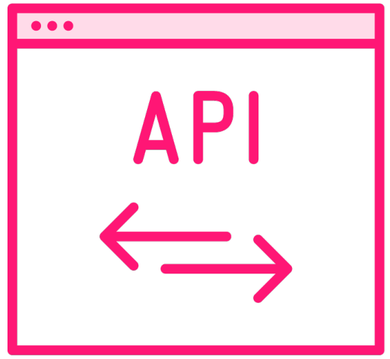
(course shown is one of my other courses, not this one)

Positioning This Course



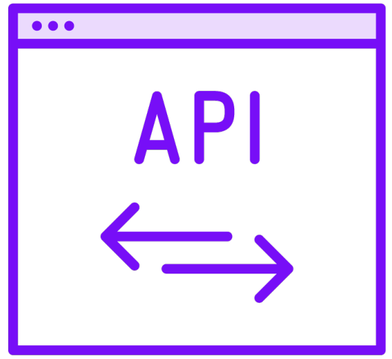
ASP.NET Core Web API Fundamentals

Beginner



ASP.NET Core Web API Deep Dive

Intermediate



This course

Intermediate

Course Prerequisites

Good knowledge of C#

Knowledge of building web APIs with ASP.NET Core

Introducing the Demo Project

Library web API, built with ASP.NET Core

- We'll start from scratch

Exercise files

- Exercise files tab on the course page
- GitHub

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

GET api/books

GET api/books

GET api/books

Horizontal Scaling

GET api/books

GET api/books

GET api/books

GET api/books

Horizontal Scaling

GET api/books

GET api/books

GET api/books

GET api/books

GET api/books

Horizontal Scaling

GET api/books

GET api/books

GET api/books

GET api/books

GET api/books

Horizontal Scaling

GET api/books

GET api/books

GET api/books

GET api/books

GET api/books

Horizontal Scaling

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

- RESTful (stateless) systems are a good start

Other components can still hurt scalability

- E.g.: non-distributed databases or caches

Vertical Scaling

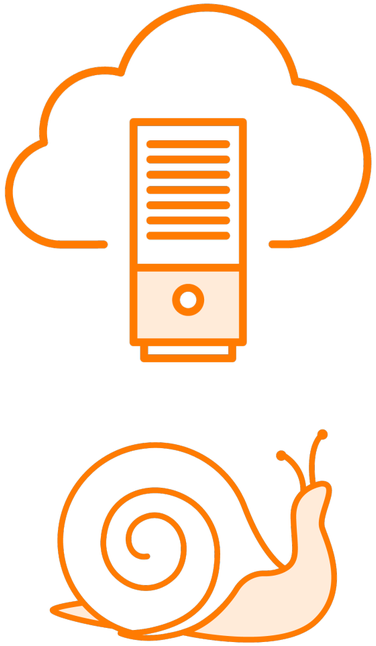
GET api/books

GET api/books

GET api/books

GET api/books

GET api/books



Vertical Scaling

GET api/books

GET api/books

GET api/books

GET api/books

GET api/books

Vertical Scaling

GET api/books

GET api/books

GET api/books

GET api/books

GET api/books

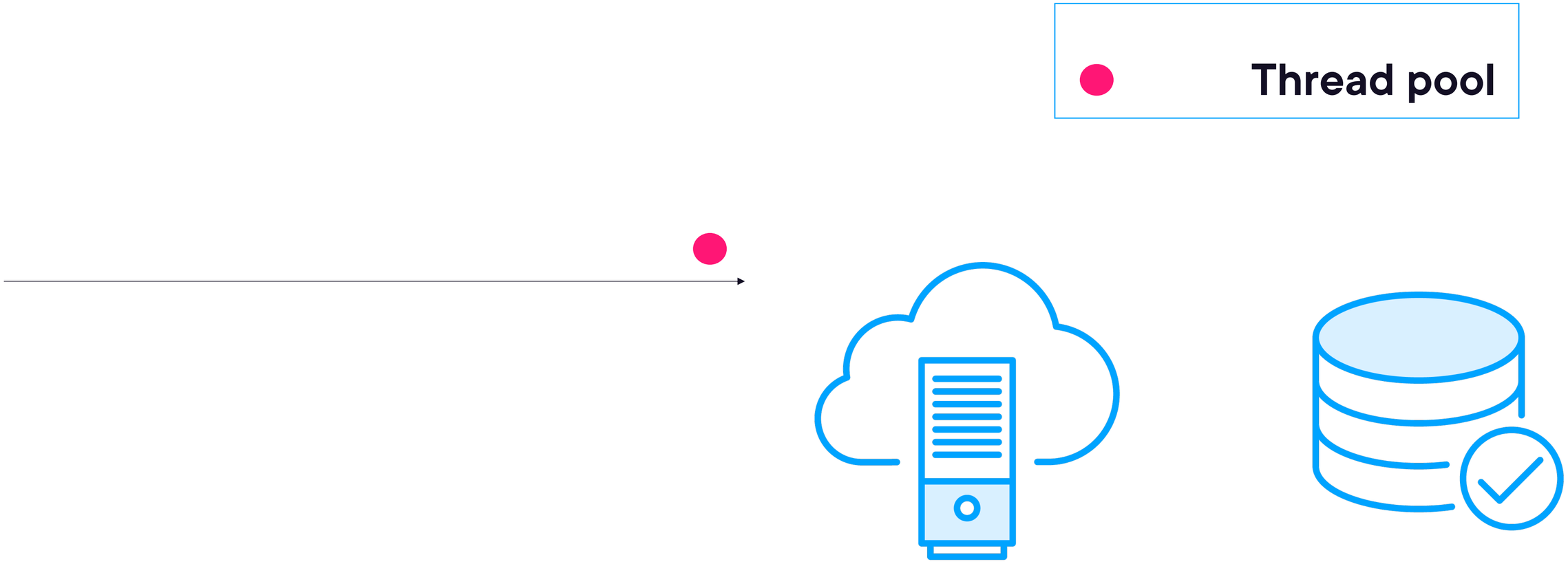
Vertical Scaling

Another way to increase scalability is by writing an API in such a way that resource utilization is improved

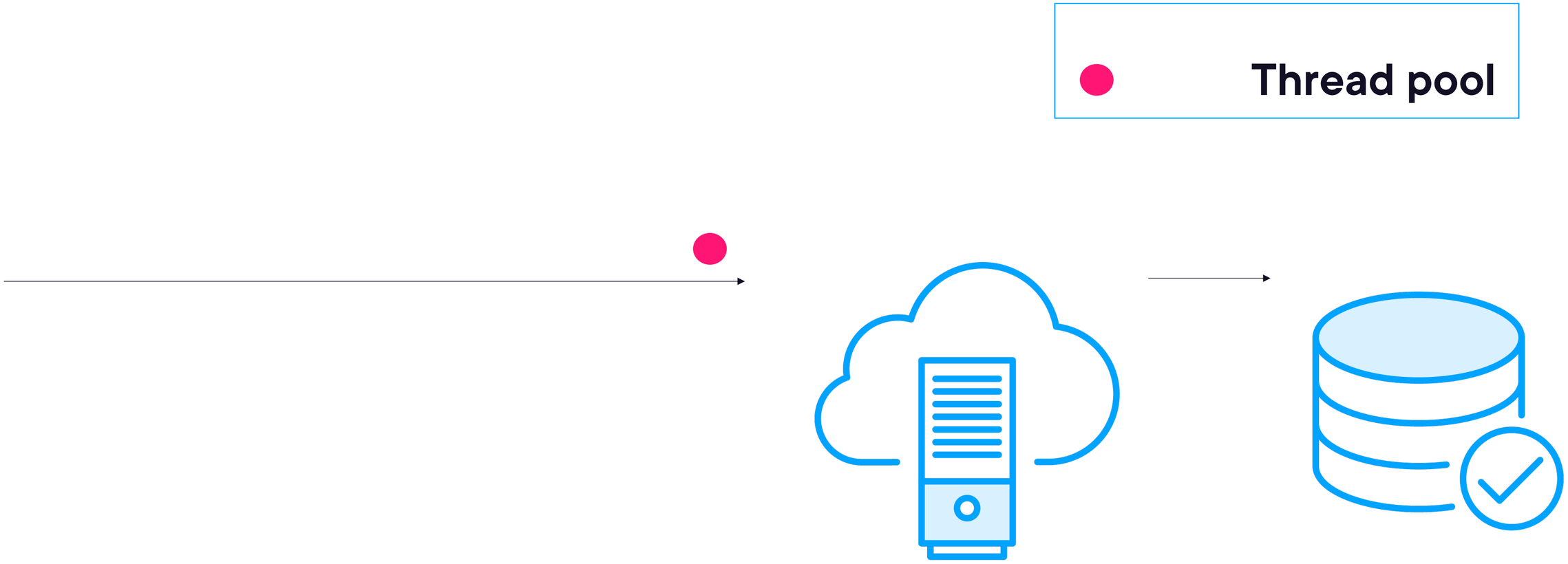
Writing async code helps with improving the vertical scalability at server level

Handling Synchronous Requests

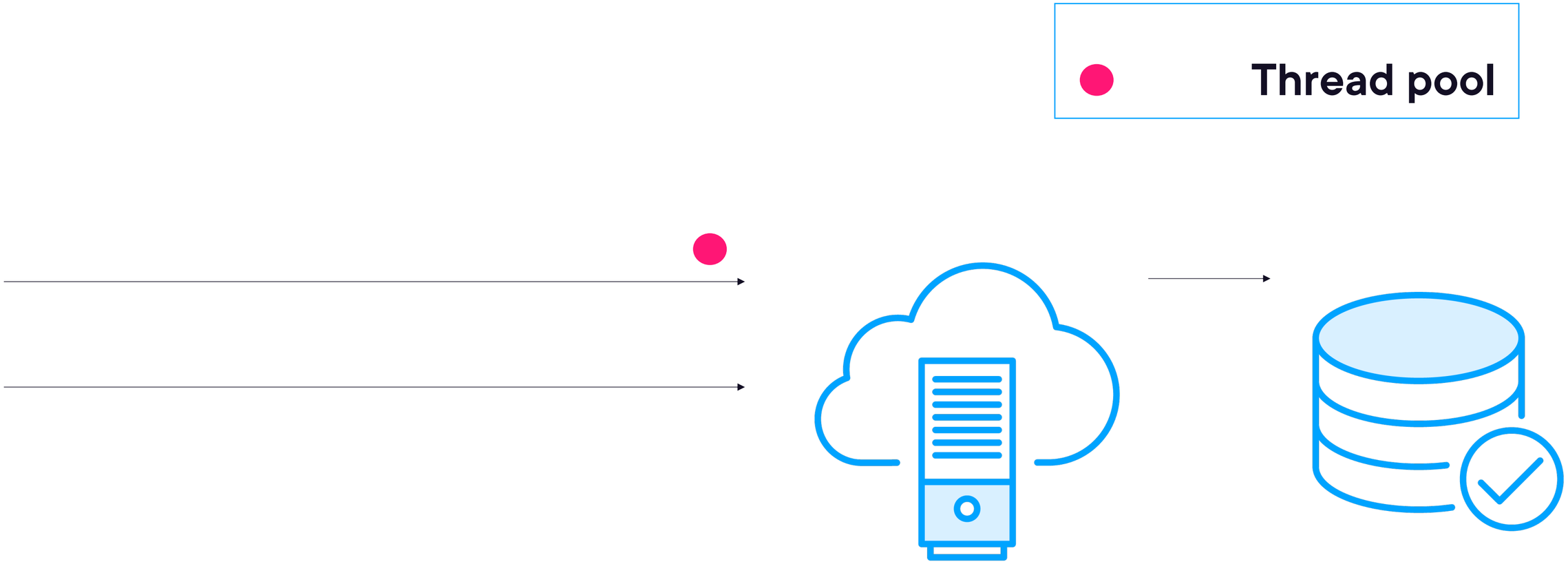
Handling Synchronous Requests



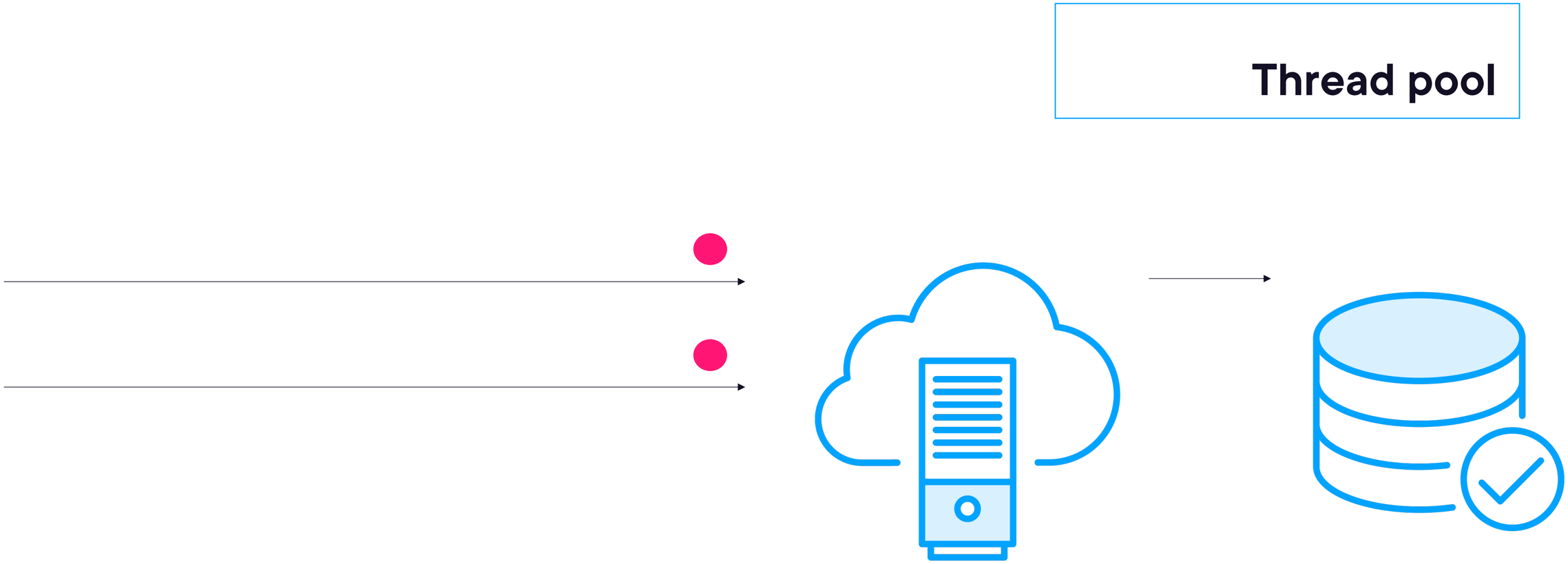
Handling Synchronous Requests



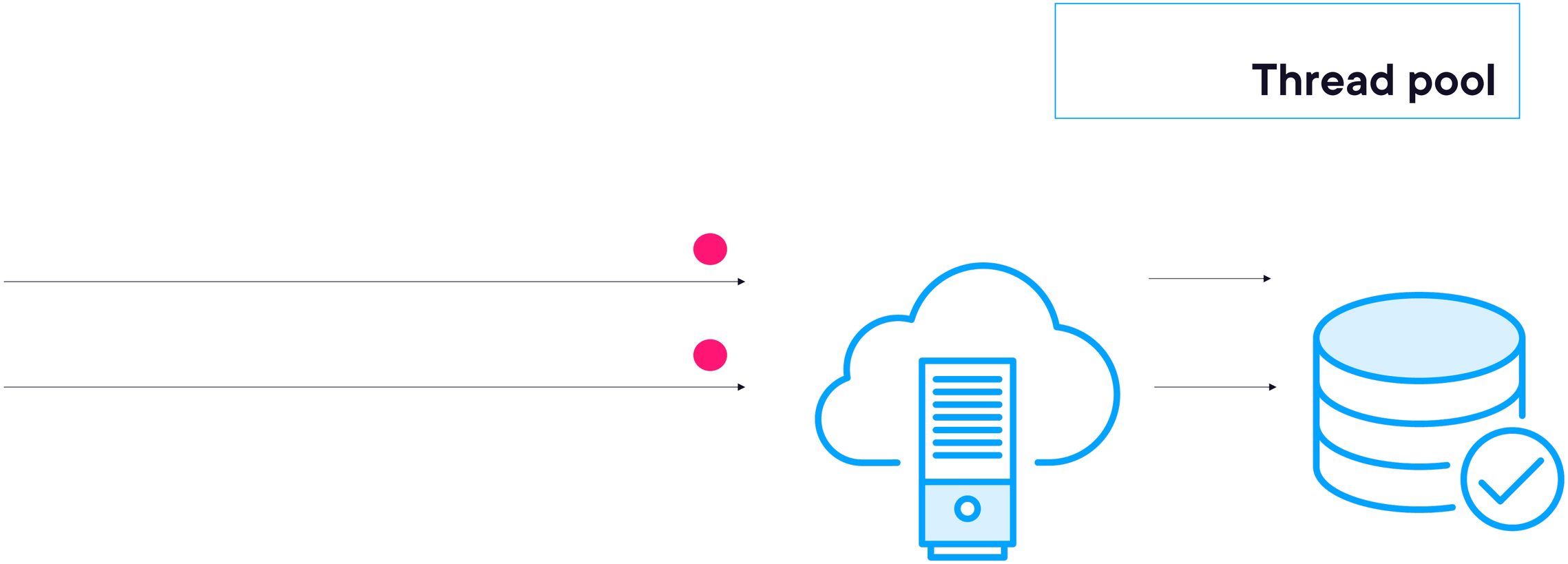
Handling Synchronous Requests



Handling Synchronous Requests



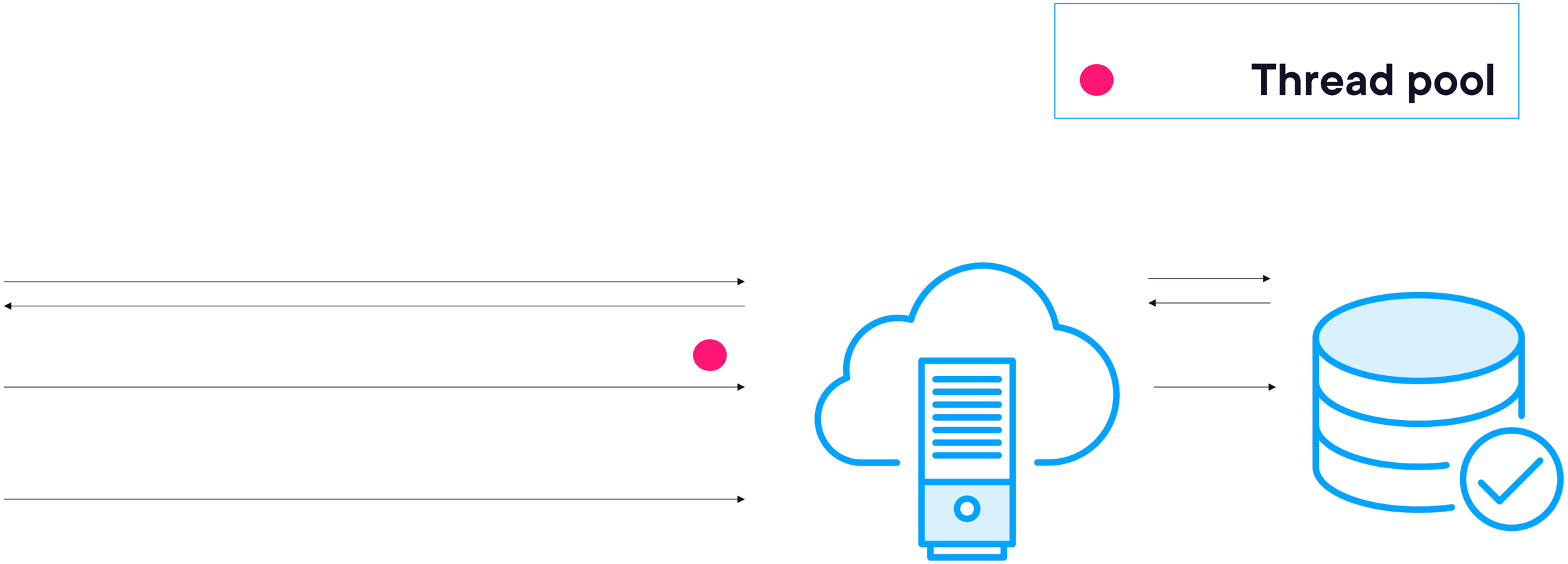
Handling Synchronous Requests



Handling Synchronous Requests

Handling Synchronous Requests

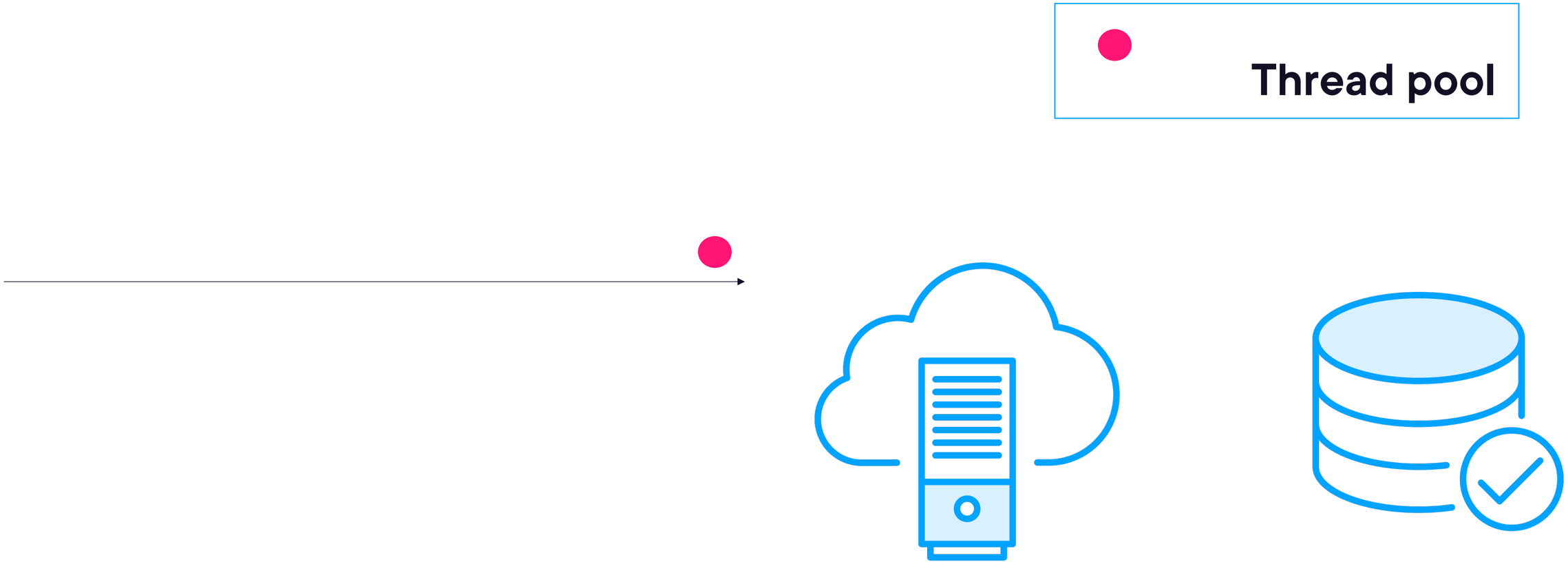
Handling Synchronous Requests



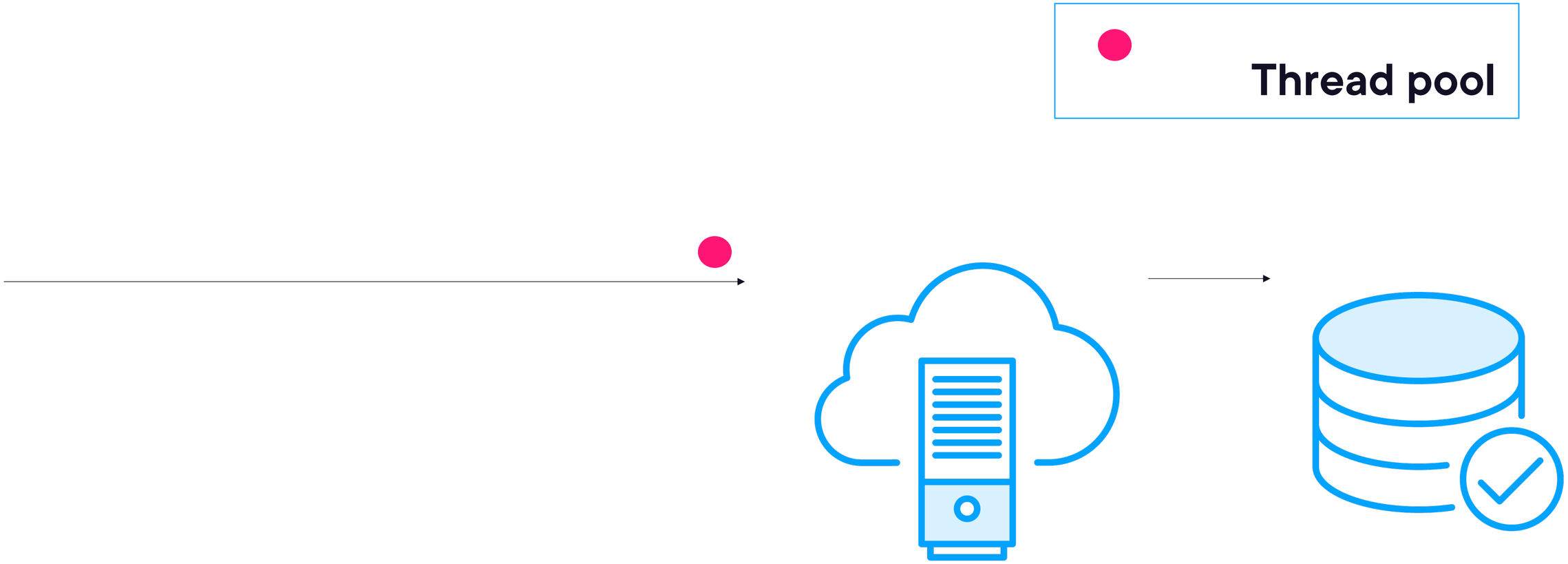
Handling Synchronous Requests

Handling Asynchronous Requests

Handling Asynchronous Requests



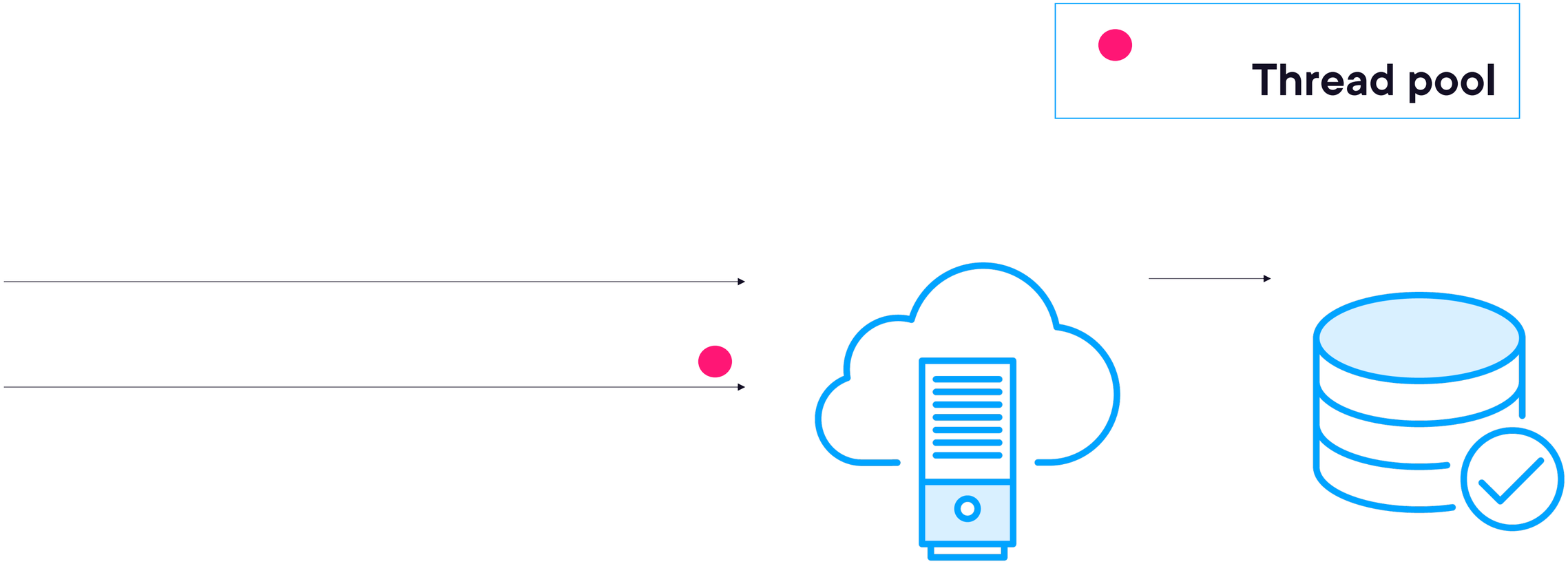
Handling Asynchronous Requests



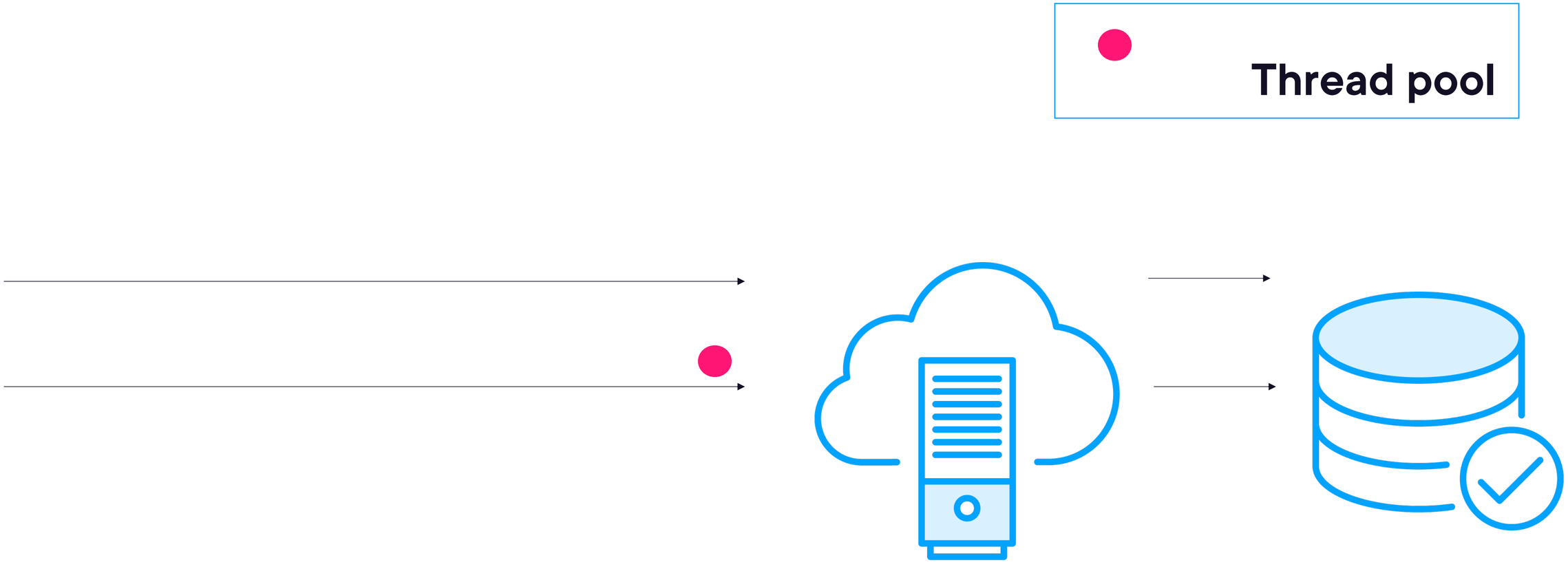
Handling Asynchronous Requests

Handling Asynchronous Requests

Handling Asynchronous Requests



Handling Asynchronous Requests



Handling Asynchronous Requests

Handling Asynchronous Requests

Handling Asynchronous Requests

Handling Asynchronous Requests

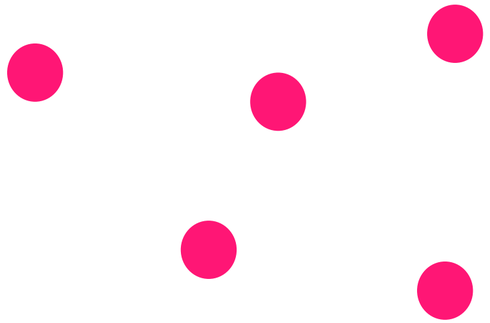
Handling Asynchronous Requests

Handling Asynchronous Requests

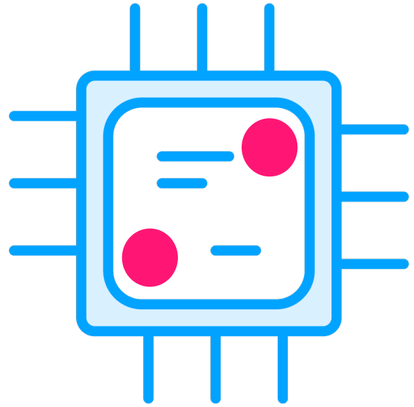
I/O-bound Vs. Computational-bound Work

I/O-bound work	VS	Computational-bound work
“Will my code be waiting for a task to be complete before continuing?”		*“Will my code be performing an expensive computation?”*
File system, database operations, network calls		Expensive business algorithm
Server-side and client-side		Client-side
		Don’t use async on the server for computational-bound work

Threads, Multithreading, Concurrency, Parallelism

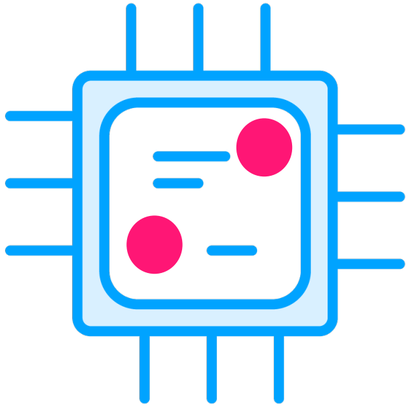


A thread is...
a basic unit of CPU utilization

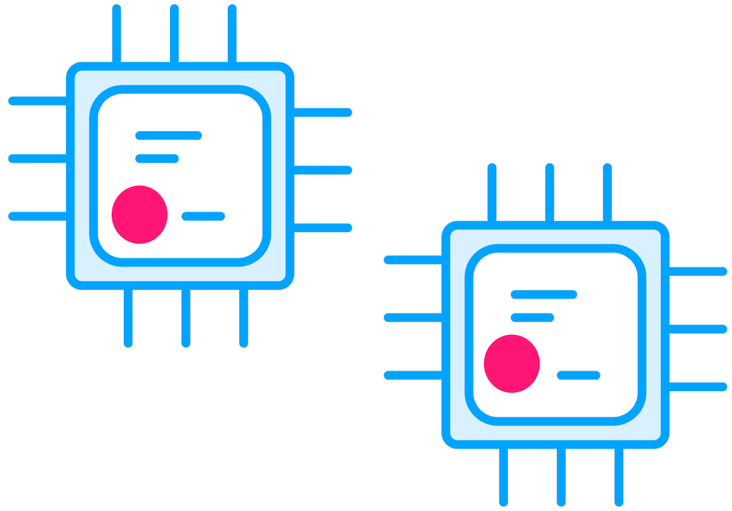


Multithreading means that...
one single CPU or single core in a multi-core CPU can execute multiple threads concurrently

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

- E.g.: file system, network, database requests

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

- E.g.: long-running calculations
- Might have adverse effects
- Can be used on the client

Starting at the Bottom with Your Data Access Layer