COMPSCI 497S

Scalable Web Systems

01 Introduction

Welcome

Welcome to 497S Scalable Web Systems!

Statement of Inclusivity

The staff for this course support the UMass commitment to diversity, and welcome individuals regardless of age, background, citizenship, disability, sex, education, ethnicity, family status, gender, gender identity, geographical origin, language, military experience, political views, race, religion, sexual orientation, socioeconomic status, and work experience. In this course, each voice in the classroom has something of value to contribute. Please take care to respect the different experiences, beliefs and values expressed by students and staff involved in this course.

Description

The web has become a large and complex area for application development. Access to an abundance of open-source languages, libraries, and frameworks has led to the quick and easy construction of a variety of applications with several moving parts working in coordination to present to the user the illusion of a single program. In reality, web applications are extremely difficult to get right. They involve a large collection of coordinated services, multiple databases, complicated user interfaces, security and performance issues, and ever-changing 3rd party services, spread across physical and virtual machines. These complications are further stressed by the large number of concurrent users that access these applications every second. This course will investigate several well-known web-based applications and the technology and software architecture used to scale these applications. We will also study a specific topic related to scalability in software design in the context of web application architecture.

Prerequisites

- COMPSCI 326 Web Programming
- COMPSCI 320 Software Engineering
- Or equivalent background

Background & Experience

You must have a strong background in the fundamentals of web programming. You should be familiar with at least the following concepts/tools:

- JavaScript
- Document Object Model (DOM)
- Node.js
- NPM
- Express
- Hypertext Transfer Protocol (HTTP)
- Client/Server Architecture
- Single Page Applications (SPA)
- Git and GitHub
- Markdown

Background & Experience

In general, this course expects you to have a solid grasp of JavaScript programming, both in the browser and the server, and single-server full stack web development.

We also expect that you are very familiar with Git and GitHub as we will use both extensively.

We will begin the course with a brief review of these topics if you are a little rusty. However, if you are not familiar with these topics, you will likely struggle to keep up with the course material.

Additional notes about background…

It is recommended to have (COMPSCI 220 Programming Methodology and COMPSCI 230 Computer Systems Principles) or 377 Operating Systems.

However, **this is not required**, but it will make your experience richer.

(e.g., basic security concepts, network protocols, processes, concurrency, client/server architecture, and general operating systems concepts).

Major Applicability

- This course counts as a 400-level Computer Science Major elective.
- This course counts as a 400-level Informatics elective.

Textbook

There is no required textbook for this course. However, you may find the following texts useful for deeper reading. I use these books as base material for this course and often code examples and diagrams are taken from them.

- **Distributed Systems with Node.js: Building Enterprise-Ready Backend Services**, Thomas Hunter II, O'Reilly Media, 2018 (I use this book extensively for the course material)
- **Node.js Design Patterns**, Mario Casciaro, Packt Publishing, 2014
- **Docker in Action**, 2nd Edition, Jeff Nickoloff, Manning Publications, 2019
- **Programming TypeScript**, Boris Cherny, O'Reilly Media, 2018
- **Building Microservices - Designing Fine-Grained Systems**, 2nd Edition, Sam Newman, O'Reilly Media, 2022

Textbook

There is no required textbook for this course. However, you may find the following texts useful for deeper reading. I use these books as base material for this course and often code examples and diagrams are taken from them.

- **Distributed Systems with Node.js: Building Enterprise-Ready Backend Services**, Thomas Hunter II, O'Reilly Media, 2018 (I use this book extensively for the course material)
- **Node.js Design Patterns**, Mario Casciaro, Packt Publishing, 2014
- **Docker in Action**, 2nd Edition, Jeff Nickoloff, Manning Publications, 2019
- **Programming TypeScript**, Boris Cherny, O'Reilly Media, 2018
- **Building Microservices - Designing Fine-Grained Systems**, 2nd Edition, Sam Newman, O'Reilly Media, 2022

I also reference and use many other resources that other educators have created, but you do not need to purchase these books or other resources.

Required Materials

- A laptop with a modern web browser (e.g., Chrome, Firefox, Safari, Edge). Although we can't officially require you to have a laptop, it will allow you to participate more effectively in class.
- A GitHub account. You will be using GitHub for all of your assignments.

Software Platforms

We will be using a few software platforms as part of the management of this course.

- Moodle
- Piazza
- GitHub Classroom
- Zoom
- Echo360
- Other…

Moodle

Moodle is used to communicate course materials. You will be able to find the course syllabus, lecture material, recorded lectures, and other links on Moodle.

We also use Moodle for assignment scheduling, submission (links to GitHub repositories), and grades.

Piazza

Piazza is used to communicate with the instructor and other students.

You can ask questions, post answers, and discuss course material.

You can also use Piazza to communicate with the instructor privately.

GitHub Classroom

GitHub Classroom is also used to distribute and collect assignments.

You will be using GitHub for all your assignments.

You will need to create a GitHub account if you do not already have one.

Zoom and Echo360

- All class meetings will be *synchronously* streamed on Zoom.
- All class meetings will be recorded by Zoom and uploaded to Echo360 at the end of class.
- You will be able to access the class on Zoom if you are unable to make it to class in-person or you are assigned to a Tuesday/Thursday remote.
- If you are unable to make it to class for a personal reason, you can view the recorded class meeting *asynchronously*.
- **DO NOT ABUSE THIS!**

Other…

As part of this course, you will be completing a group semester project.

It will be necessary to communicate with others in your group.

You will decide the best communication mechanism for your group (e.g., Slack, Discord, FaceTime, Facebook).

Software Tools

We will use a variety of software tools in this course. The most important that you will need to install on your computer are:

- Visual Studio Code (VSCode)
- Node.js / npm
- Git
- Docker
- Other languages and tools

Learning Objectives 1

- Define concepts of modern distributed web application architecture.
- Explain issues that relate to scalability of modern web systems and possible solutions.
- Explain the asynchronous nature of JavaScript and how it relates to web application scalability.
- Explain the event-loop model of JavaScript in the browser and Node.js and how it relates to web application scalability.

Learning Objectives 2

- Explain, design, and implement services that communicate with each other using various communication protocols.
- Define and explain micro-services and their importance in scalability.
- Design and implement a microservice based web application.
- Describe the role of Docker containers in modern web application architecture.

Learning Objectives 3

- Identify libraries and tools that help solve scalability problems.
- Apply languages, libraries, and tools that are important for scalability.
- Use TypeScript as a scalable language for web applications.
- Use React as a modern scalable user interface library.

Learning Objectives 4

- Use a modern text editor (VSCode) and terminal for interacting with course material and assignments.
- Analyze an existing web system and identify problems related to and requirements for scalability.
- Give a presentation that effectively communicates the results of a research project.

At the end of this course…

1. You will be able to design and implement a scalable web application using the various technologies and tools that we will cover in this course.
2. You will also be able to analyze an existing web system and identify problems related to scalability and requirements for scalability.
3. You will be able to research the technology and methodologies required to implement such systems.

Self Motivation

The structure of this course is different from standard courses.

It relies on the self motivation of each student. The more you put into this course the more you will get out of it.

Self Motivation

Although several topics will be presented, it is expected that students in this course will contribute a **substantial amount of research** into various topics and technologies in order to be successful.

Self Motivation

This **is not an introductory course**, rather it attempts to scaffold and prepare you to investigate the material to be successful in industry and/or graduate school.

Self Motivation

You will be evaluated on your contributions and dedication to the course material, your own research, and the team you work with on your project.

Self Motivation / Participation

If you do not participate in this course and/or its material, you will likely receive a lower grade than you might expect. I take this seriously, so you should as well.

Teams

This course requires the completion of a team project.

You will be required to participate on a team of approximately 3 individuals.

Teams will be self-organized at the start of the course.

We will spend the first 2 weeks getting to know others in the class through group assignments.

You are welcome to work with anyone in the class. If you know other people who are taking the course, feel free to work with them. If you do not know anyone in the class, then work with others nearby.

Teams

We know that forming groups/teams can be an awkward process. However, this is part of the real-world and a goal in this course is to emulate a real-world setting.

We will help you become acquainted with a group, and we hope it leads to a fruitful experience.

Course Structure (first couple of weeks)

- We will all *initially* meet twice a week for 75 minutes.
- The first 20-60 minutes will be a lecture with live coding and demonstrations.
- The remaining time will be for discussion and/or research.
- There will be lecture meeting times where it will be purely teamwork.
- The lecture will cover the material for the week and the discussion/research will be for questions and discussion the material or for team study and research. The lecture will be recorded and posted on the course website.

Course Structure (the rest of the semester)

- After teams are organized…
- Teams will be split into Tuesday and Thursday groups
- The Tuesday group will join in-person on Tuesday; the Thursday group will join on Zoom
- The Thursday group will join in-person on Thursday; the Tuesday group will join on Zoom

Why do it this way?

Flexible Learning and Remote Development

Three reasons

1. **Flexible learning** is a way to provide alternate forms of learning environments that are different from traditional forms of learning. One way of doing this is to utilize technology to allow for remote access to a course.
2. The world (especially software development) is now at a place where remote teams are a norm. As part of this course, we try to practice real-world technology, team interactions, and trends. **Remote development** is commonplace.
3. This allows the instructor and course staff to interact with teams directly during a class meeting as there will be fewer teams present in the physical room.

Assignments and Grading

- (20%) Homework
- (20%) Developer Notebook
- (30%) Participation
- (20%) Project Milestones
- (10%) Project Presentation

(20%) Homework

Individual or group activities that range from setting up various software components and/or platforms (e.g., git, GitHub, coding environment, docker, AWS/Azure) to programming assignments. These are often started in class.

(20%) Developer Notebook

You will be required to maintain a developer notebook.

This will be a Github repository used to record your own research work that you accomplish through out the semester.

Each report will consist of a markdown file and possibly code exemplars that you used to explore the material.

We will evaluate your notebook periodically and it will be used as part of the final grade assignment and adjustment.

(30%) Participation

This course requires your participation.

This is especially important when working on a team with other people.

Participation can take many forms; however, we need a mechanism to evaluate that participation.

You will be required to complete several surveys and evaluation forms during the semester that records your independent work as well as the work you complete with your team.

In addition to your own evaluation, you will be required to evaluate your team members. This will provide the course staff with a good understanding of your participation activities as well as those of your team members.

(20%) Project Milestones

Team-based activities that involve the construction of a scalable web system that your team will design and implement.

This will range from a team GitHub repository setup to the construction of various micro-services and front-end UI components.

(10%) Project Presentation

A live or recorded presentation of your team’s project and demonstration of a working prototype.

Moodle Gradebook

All grades will be accessible through the Moodle Gradebook and updated at the completion of grading each assessment.

Assignment Submission and Review

The assignments in this course are often open-ended.

This makes it difficult to grade in a reasonable amount of time as we **do not use** an auto-grader.

This means that we will be looking at your submissions from a high-level point of view.

We want you to explore the material, not simply follow step-by-step instructions.

Standards-Based Grades

This course is graded on a standards-based grading system. That is, grades in individual assignments and group work are based on specific criteria that you or your group meet.

All assignments are reviewed using the criteria that follows…

Standards-Based Grades

A Grade: The student's performance is *exceeding* the standards of the course and/or assignment. The student has demonstrated a deep understanding of the material and has gone above and beyond the requirements of the assignment. The student has demonstrated a high level of creativity and innovation in their work.

Standards-Based Grades

B Grade: The student's performance is *meeting* the standards of the course and/or assignment. You must demonstrate a good understanding of the course material and have completed the assignments showing good quality and is fully satisfactory.

Standards-Based Grades

C Grade: The student's performance is *approaching* the standards of the course and/or assignment. You must demonstrate a basic understanding of the course material and have completed the assignments showing inconsistent satisfactory quality. It shows an improving level of quality.

Standards-Based Grades

D Grade: The student's performance is *rudimentary* and just *beginning* to meet the standards of the course and/or assignment. Improvement is necessary to pass the course.

Standards-Based Grades

F Grade: The student's performance demonstrates minimal understanding of the course material. Quality is poor and/or missing assignments/participation and the student has not met the standards of the course and/or assignment.

What it means to exceed

To be clear, you do not achieve an A grade by simply completing an assignment.

You achieve an A if you **complete an assignment fully** and go **above and beyond** what is requested of you.

This is what **exceeding** means.

To **exceed** will require you to do some research into a topic that goes beyond what is presented.

This research ability is part of what the course tries to achieve.

Justifying your grade

As part of your submissions, you will be required to grade your own work and justify the grade that you think your work achieves.

We may or may not agree with you and provide arguments as to why you should achieve a higher or lower grade.

Communication

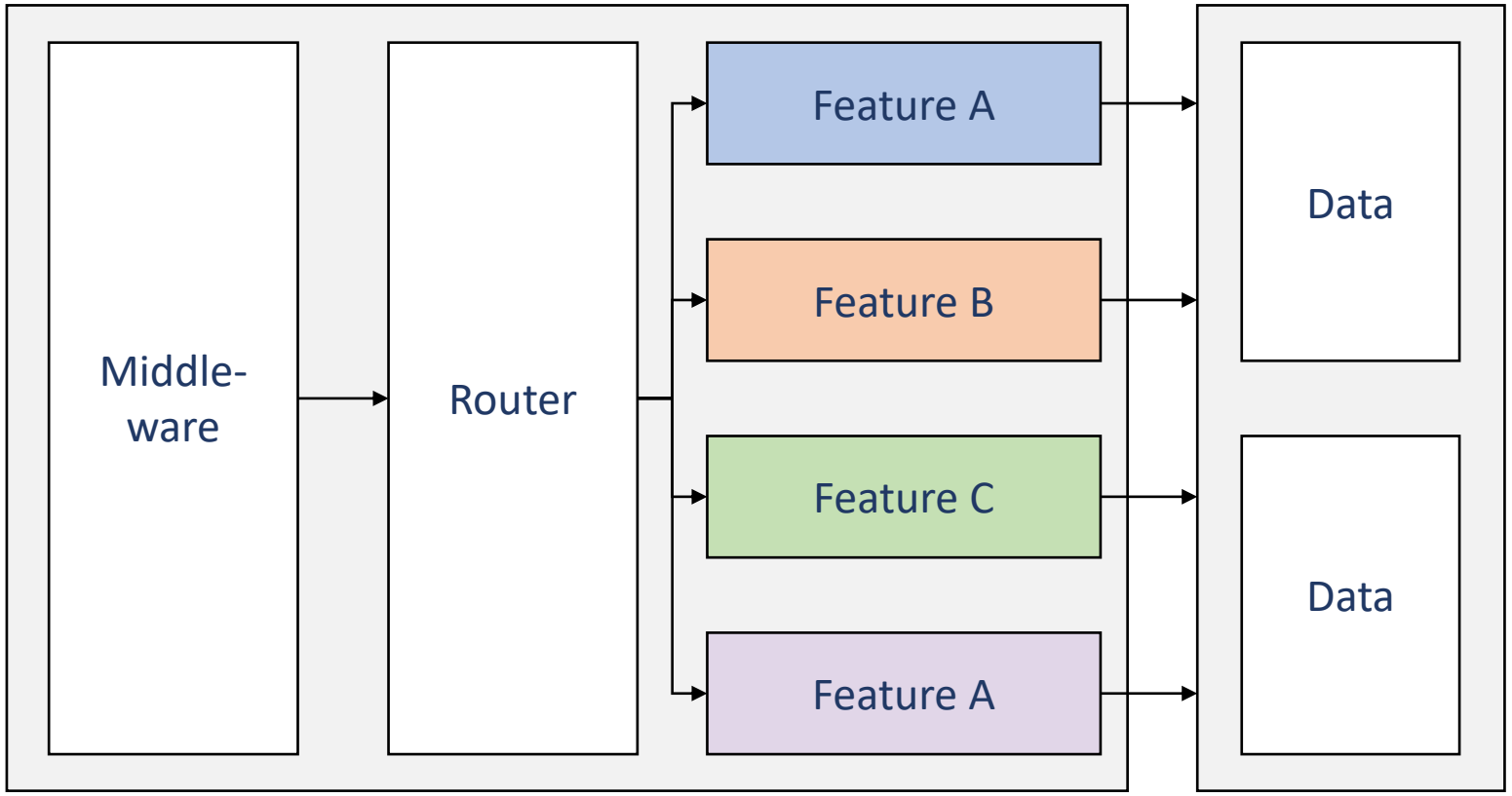
- We will be using the **Piazza** discussion forum system for communication.
- The discussion forum should be your first choice for asking questions as others most certainly have the same question.
- You should check the discussion forum before asking your question to see if the same question has already been posted.
- We will tend not to answer questions directed towards us that have already been answered in the discussion forum.
- Think before you post.

Accommodations

- Accommodations are collaborative efforts between students, faculty, and Disability Services (DS).
- Students with accommodations approved through DS are responsible for contacting the faculty member in charge of the course prior to or during the first week of the term to discuss accommodations.
- Reasonable arrangements will be made in accordance with your accommodations provided by DS in the context of this course.

Monolithic Server

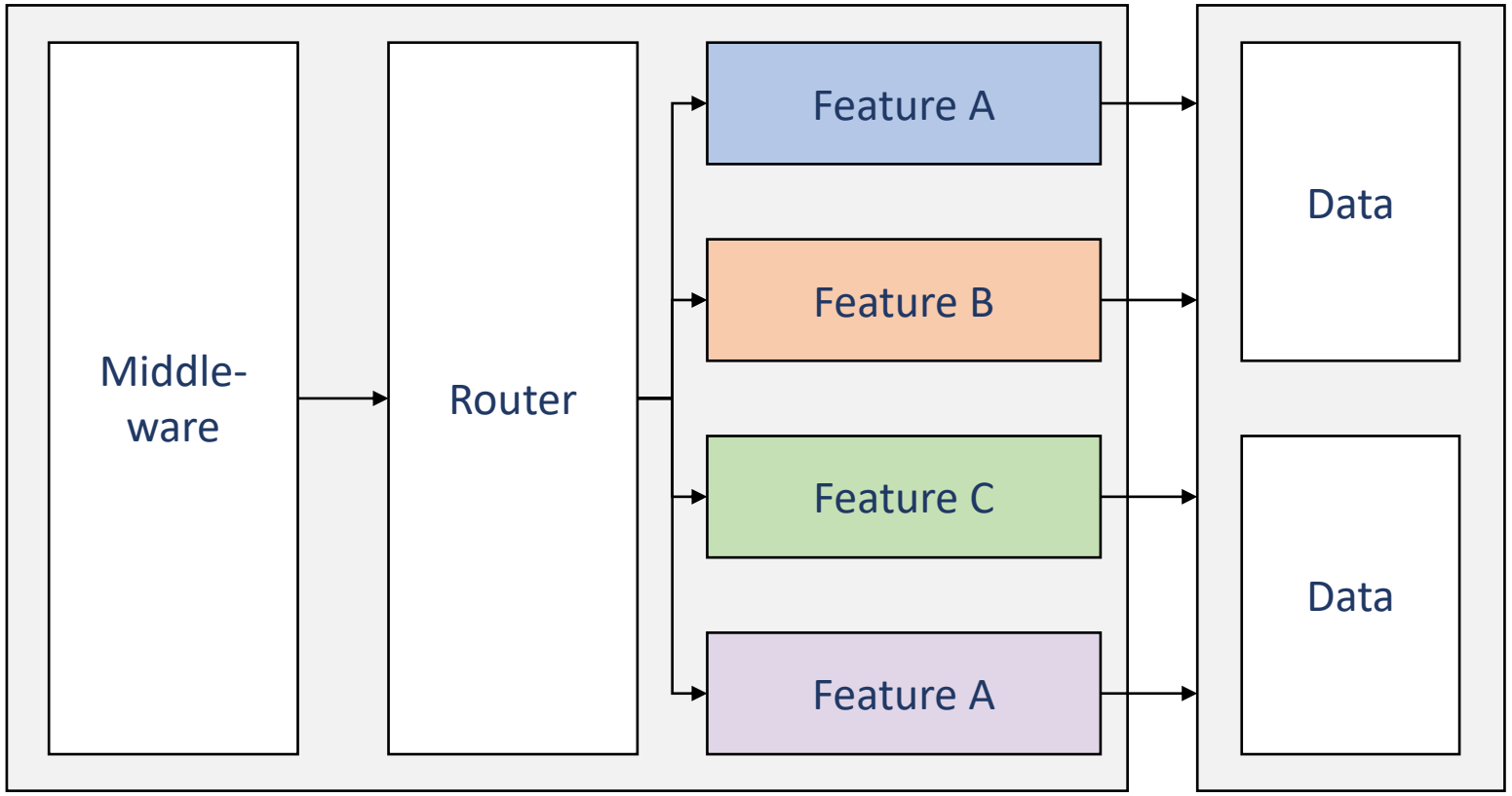
This is probably how you are building your systems right now.



Monolithic Server

A request arrives on some port given some route.

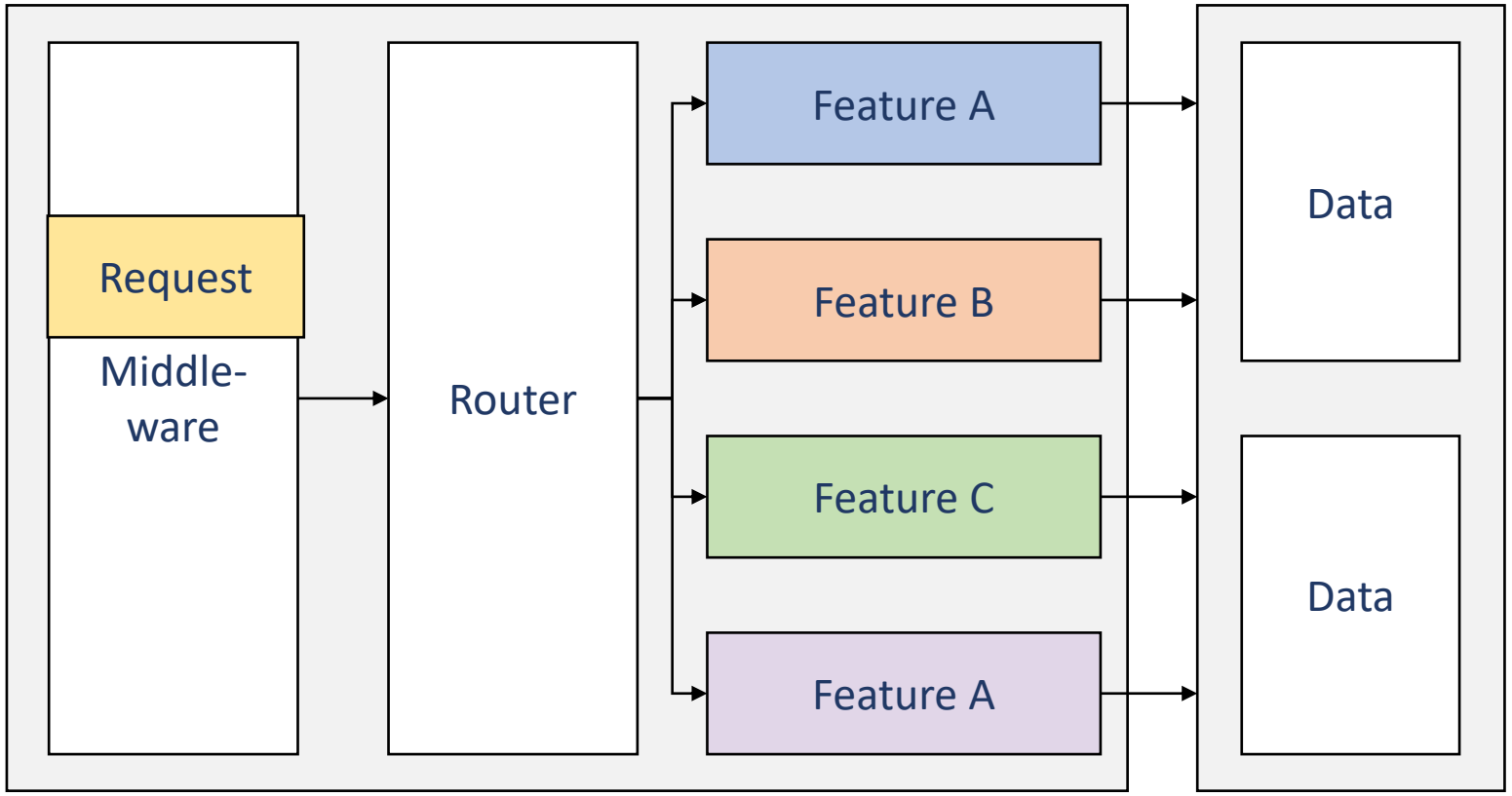
Request



Monolithic Server

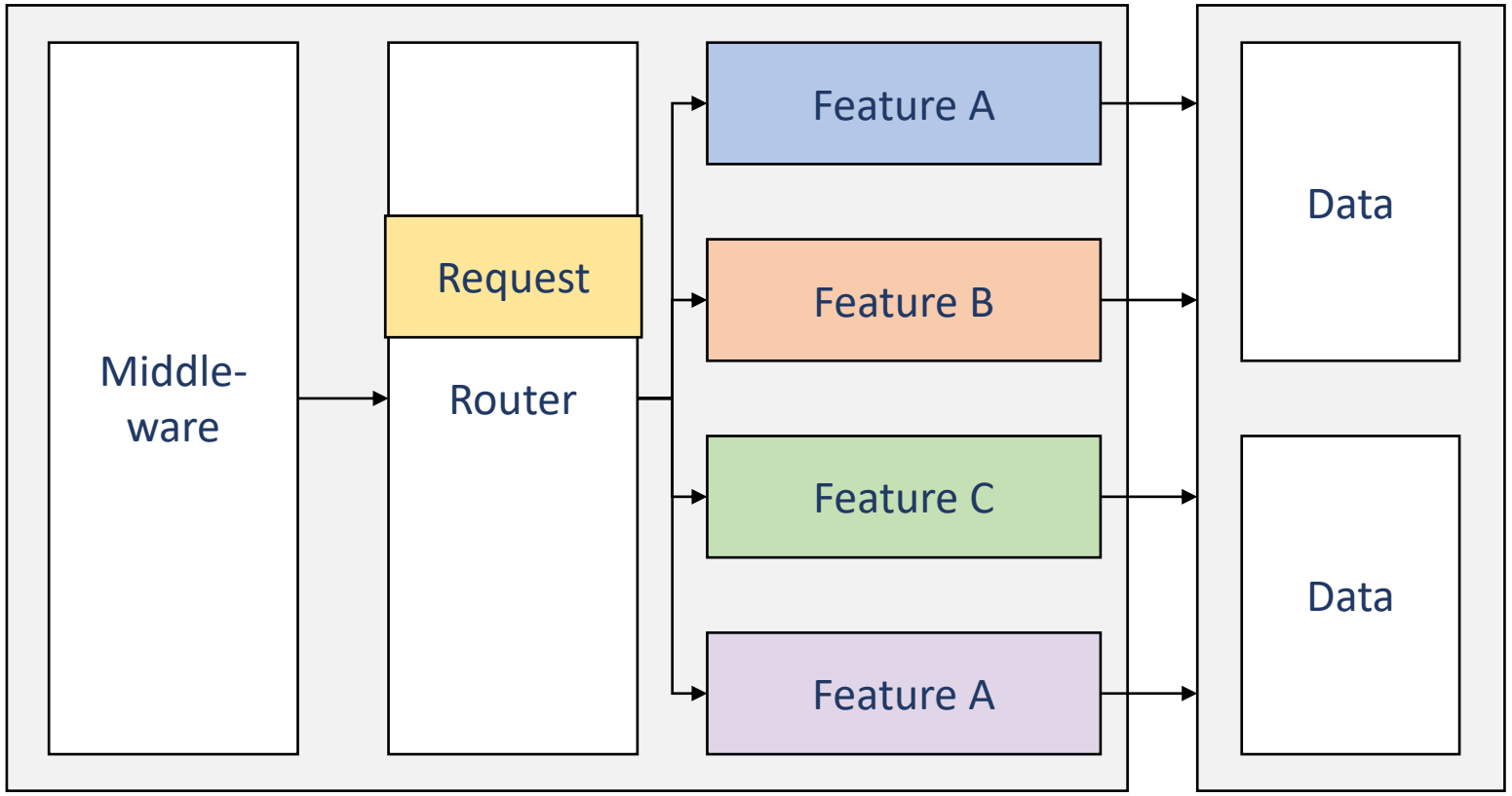
The various middleware processes the request.

Body parsing, sessions, authentication, etc.



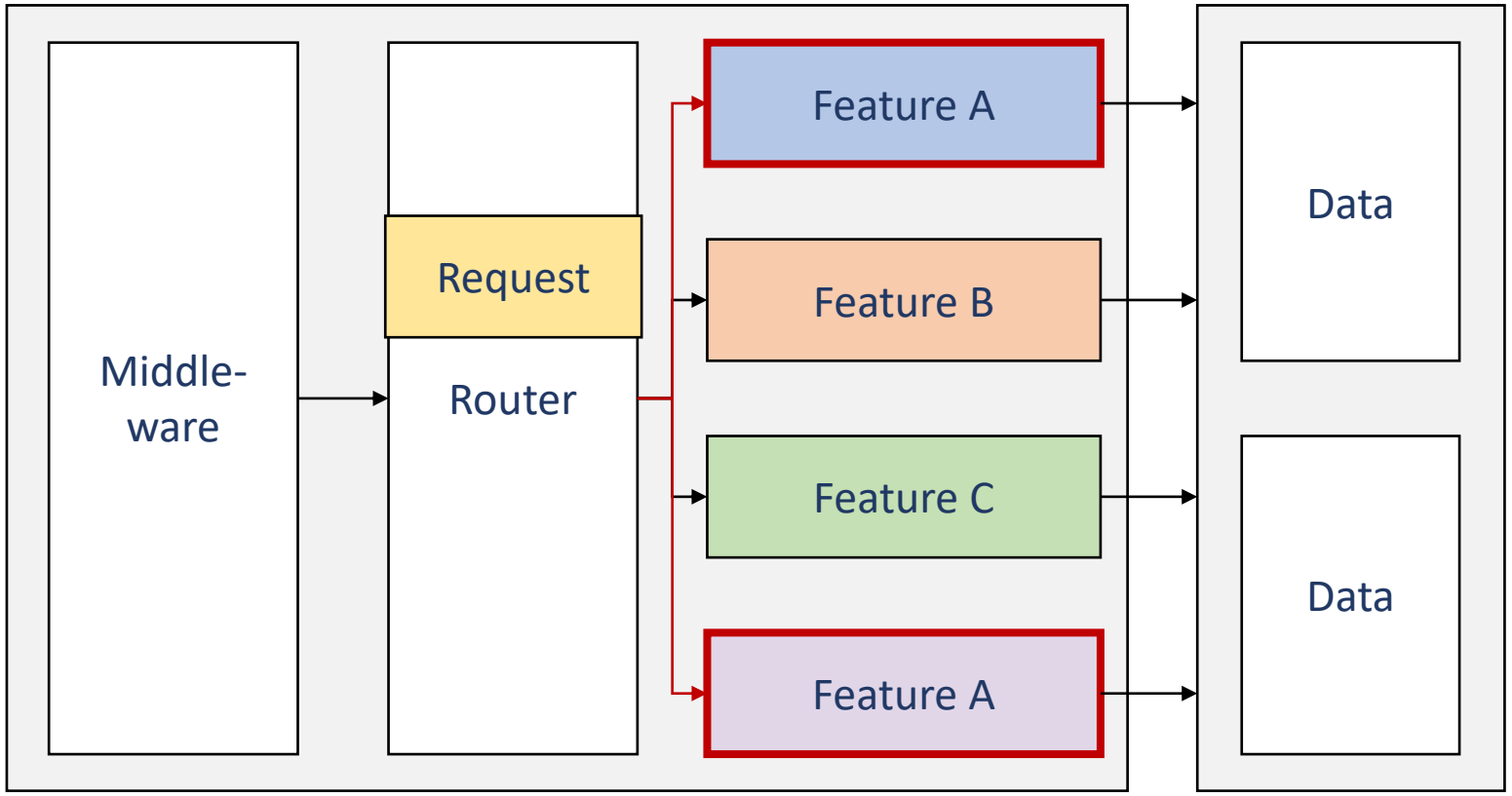
Monolithic Server

The router looks at the route to determine what the request and invokes the appropriate controller.



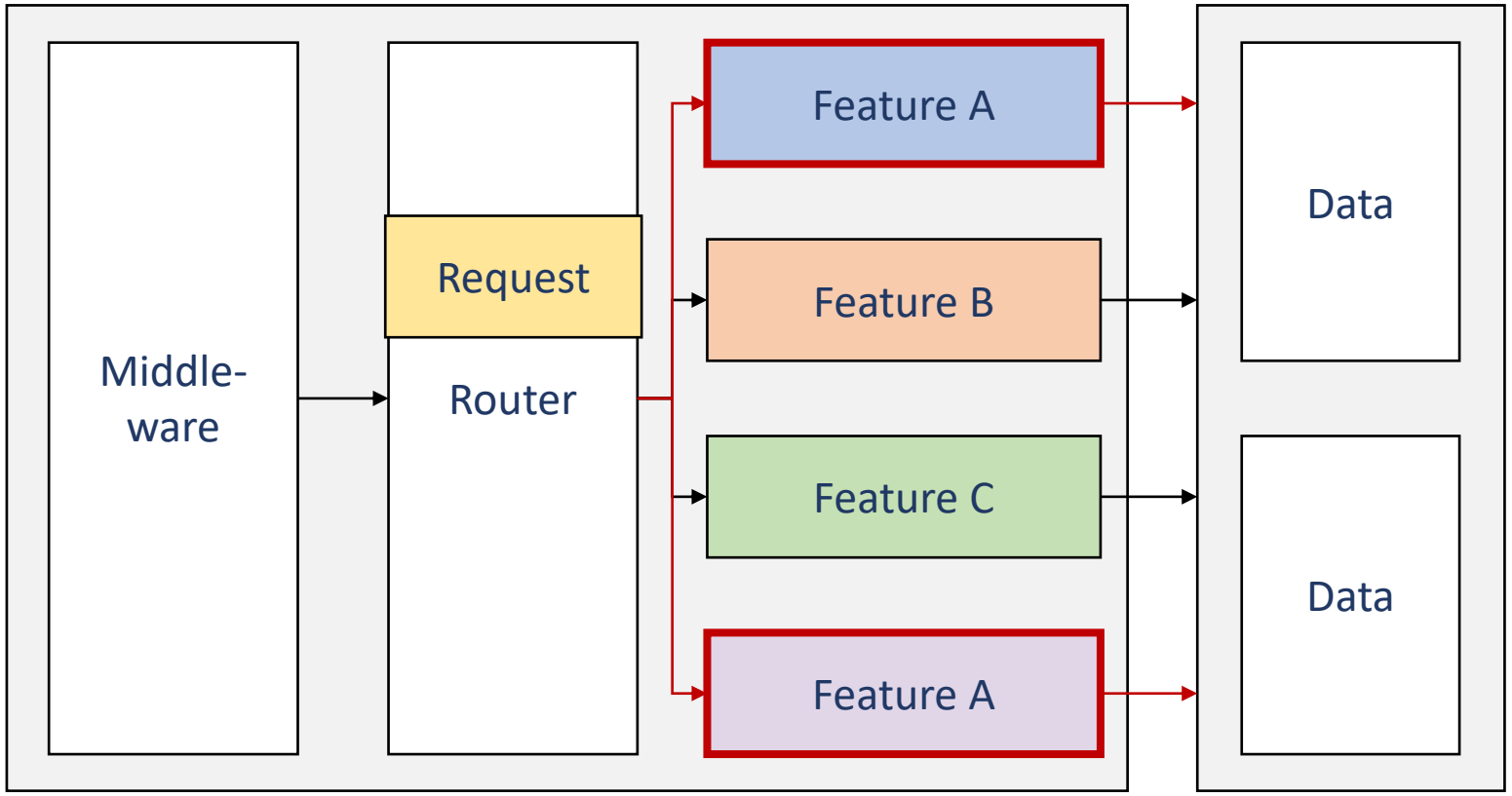
Monolithic Server

This, in turn, causes various features to be activated.



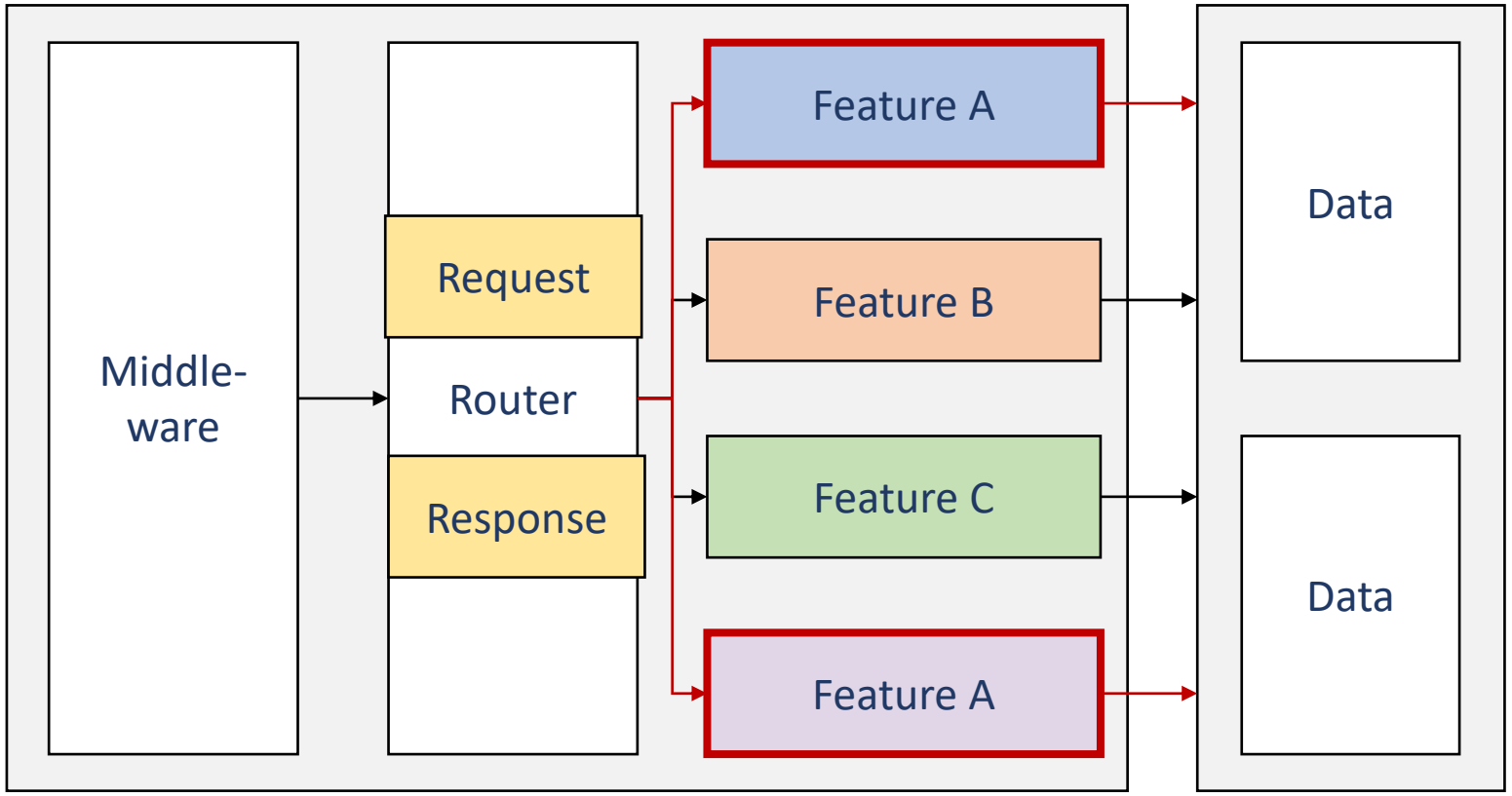
Monolithic Server

Data is fetched/stored through various CRUD operations.



Monolithic Server

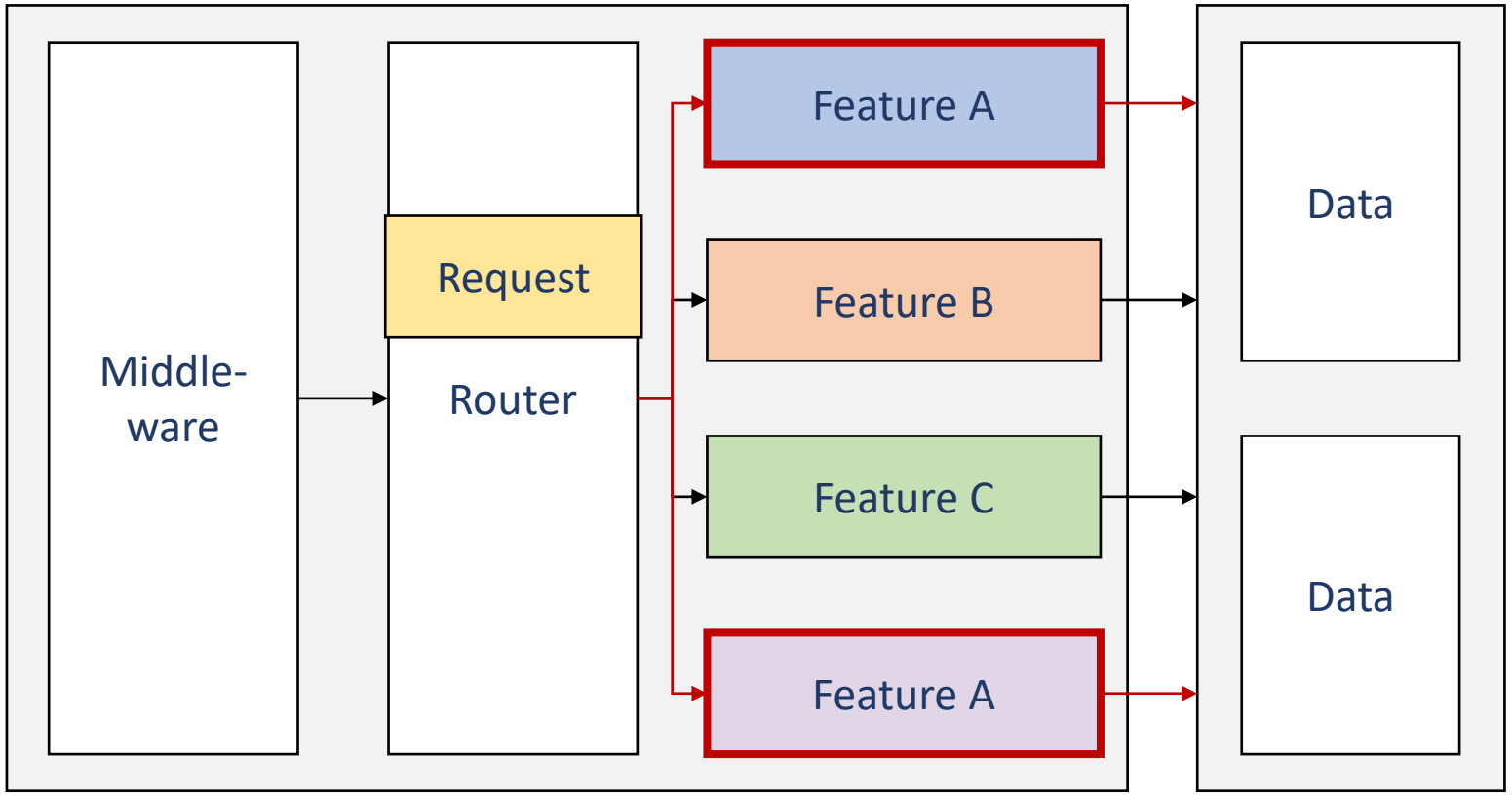
The activated features complete and a response is constructed.



Monolithic Server

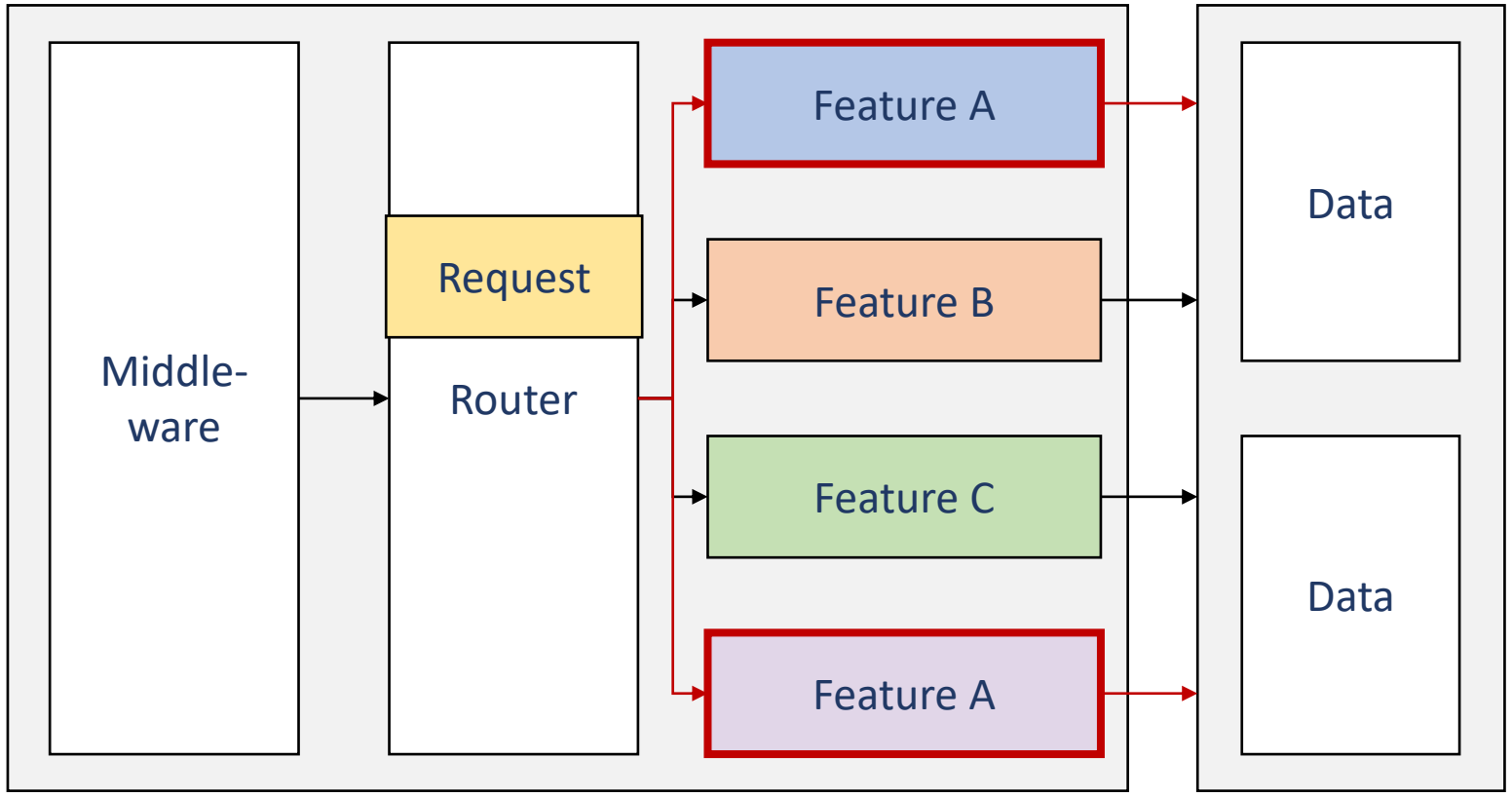
The response is then sent back to the requesting client.

Response



Monolithic Server

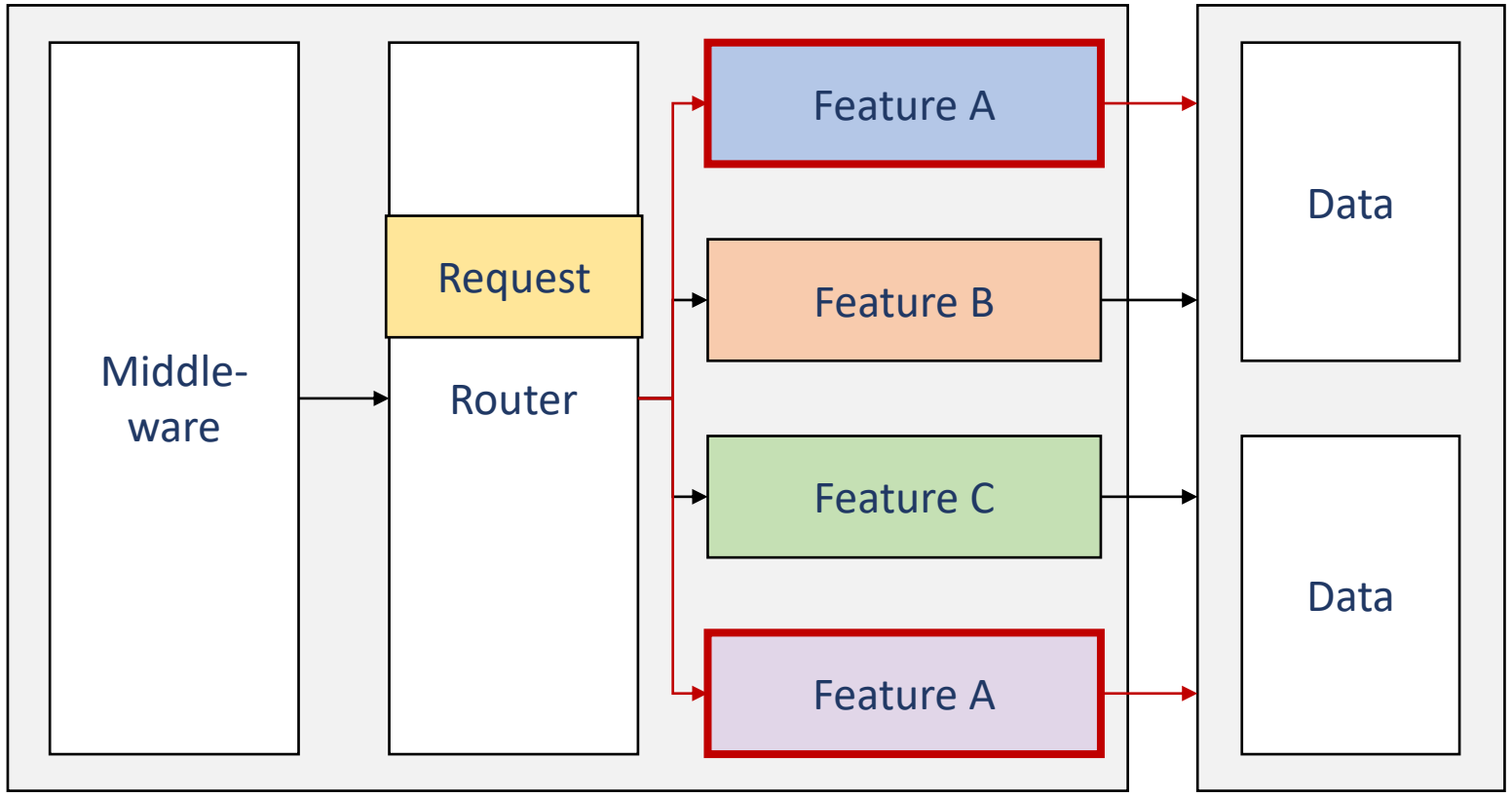
This is not an unreasonable approach to building a web system.



Response

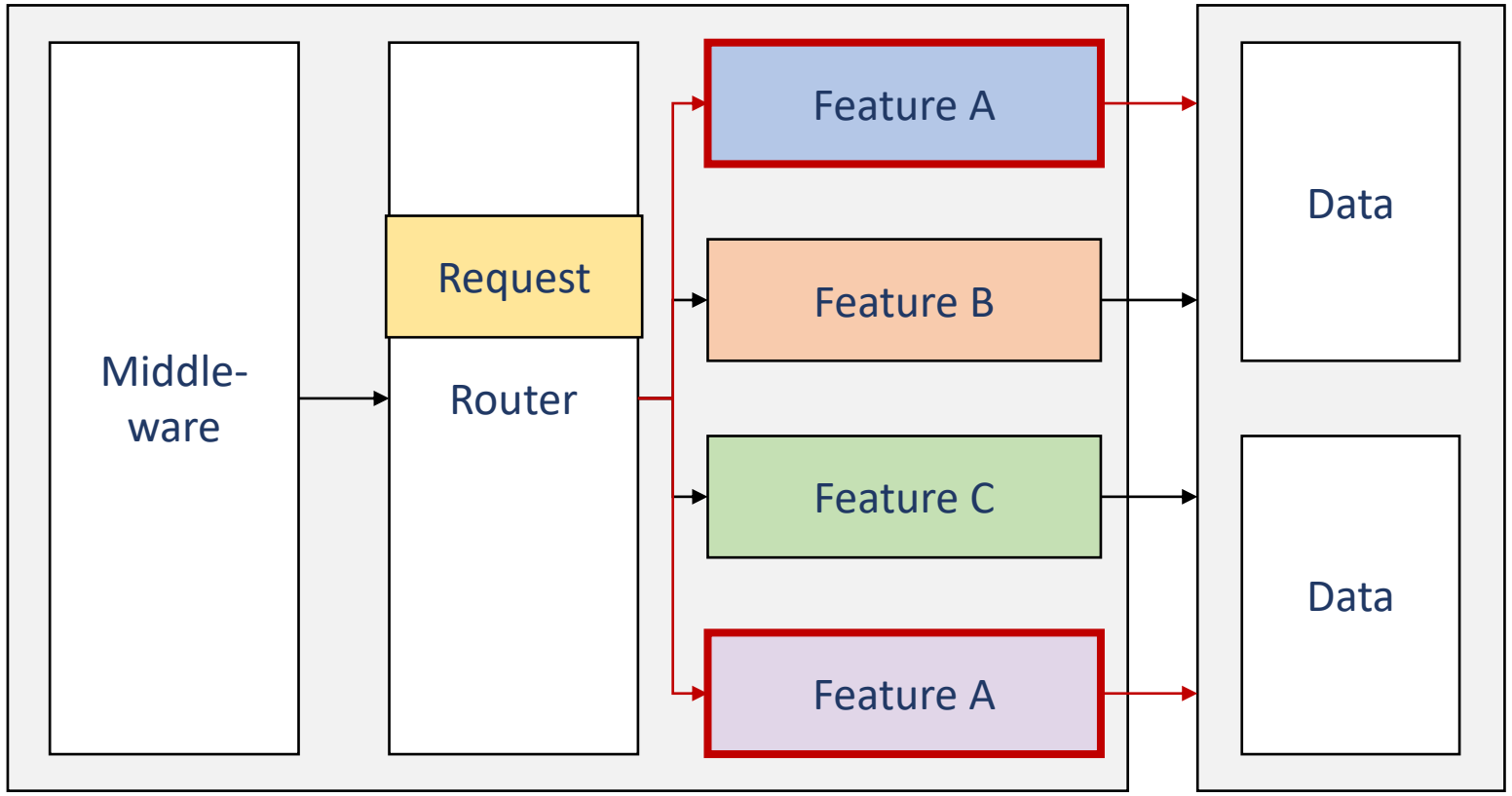
Monolithic Server

Do other architectures exist for doing this?



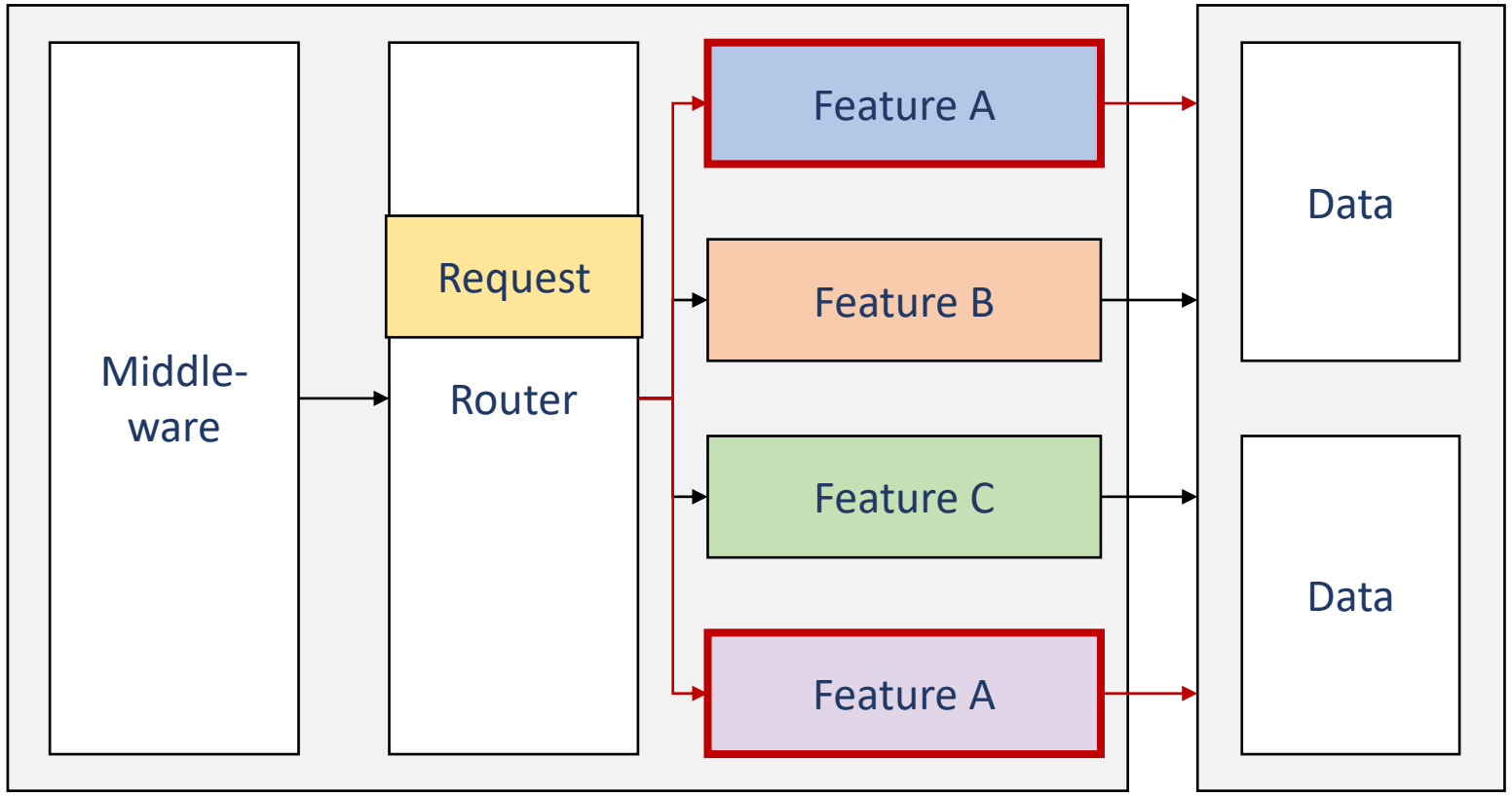
Monolithic Server

Are there ways to make this more efficient?



Monolithic Server

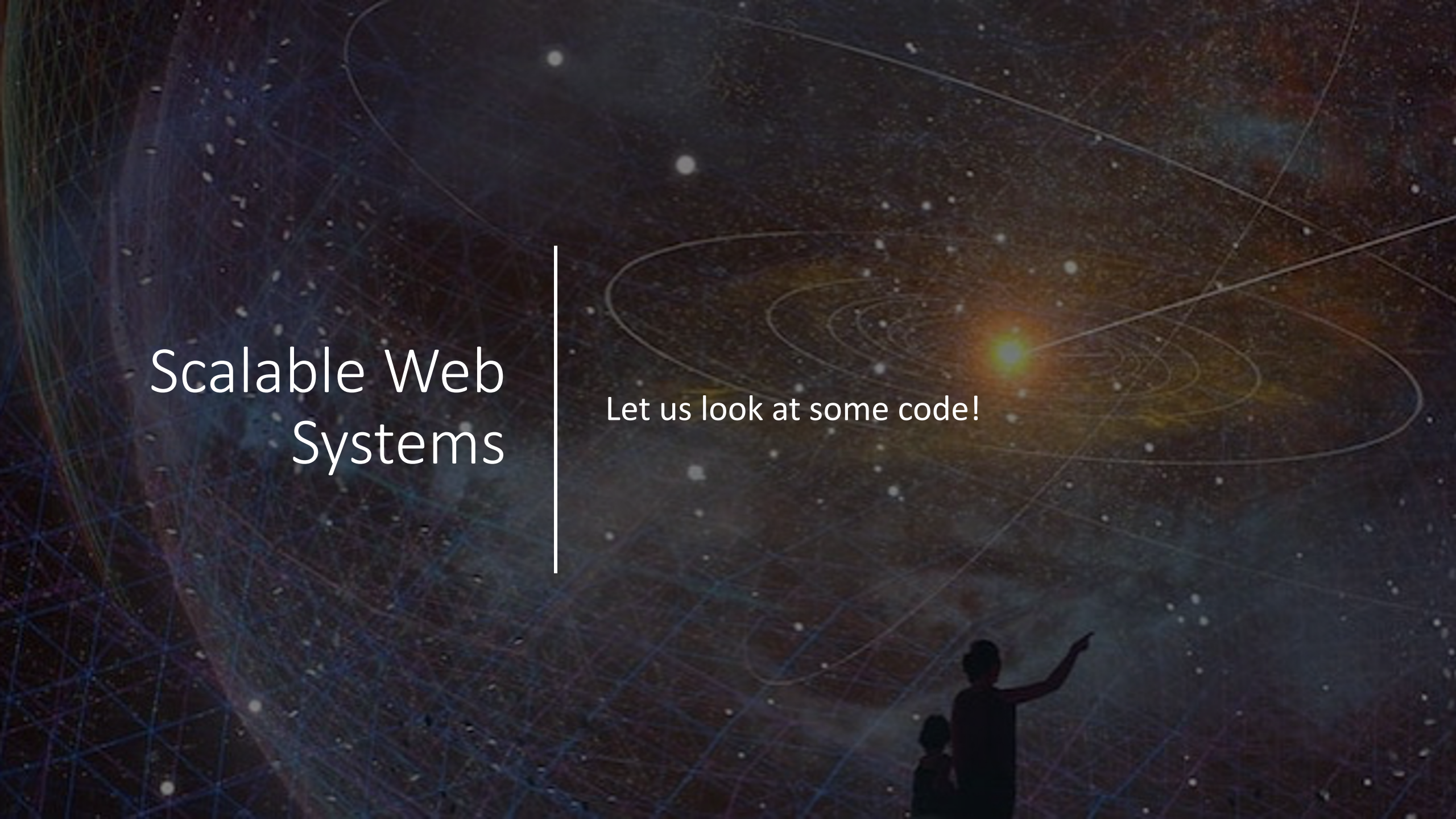
What if we scale this to a million requests per second?



Scalable Web Systems

In this course we are going to explore various dimensions of this including:

- Architecture / Micro-Services
- Containerization / Docker
- Orchestration / Docker Swarm
- Scalable UI / React
- And other various tidbits…



Scalable Web Systems

Let us look at some code!