Microservices

- Microservices are an architectural and organizational **approach to software development**
- **where software is composed of small independent services that communicate over well-defined APIs.**
- **These services are owned by small, self-contained teams.**

- Microservices architectures make applications **easier to scale and faster to develop**, enabling innovation and accelerating time-to-market for new features.

-Amazon AWS

https://aws.amazon.com/microservices/

Microservices

- Also known as the microservice architecture
- An architectural style that structures an application as a **collection of services that are:**
  - **Independently deployable**
  - **Loosely coupled**
  - **Organized around business capabilities**
  - **Owned by a small team**

“Do one thing and do it well”: the motto

- Projects that are considered to be “microservices” follow Ken Thompson’s **Unix philosophy**:
- **“Do one thing and do it well”**.
- This can be summarized as: focus on one task, and perfect it.
- This statement is not just relevant to programming, but also describes the functionality of individual microservices.

Microservices

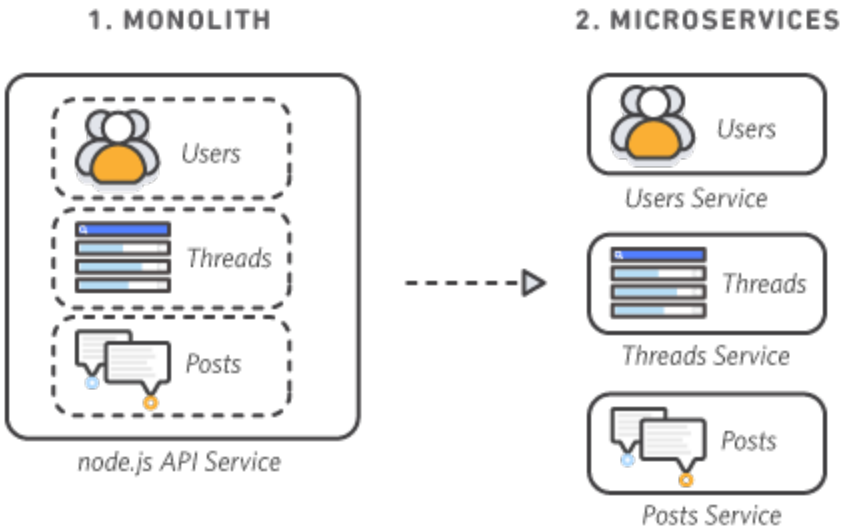
- Microservice architecture is basically a **further development of service-oriented architecture** (SOA): small services also play a role in this architectural pattern

Microservices

- The microservice architecture enables an organization to deliver large, complex applications
- rapidly,
- frequently,
- reliably and
- sustainably - a necessity for competing and winning in today's world.

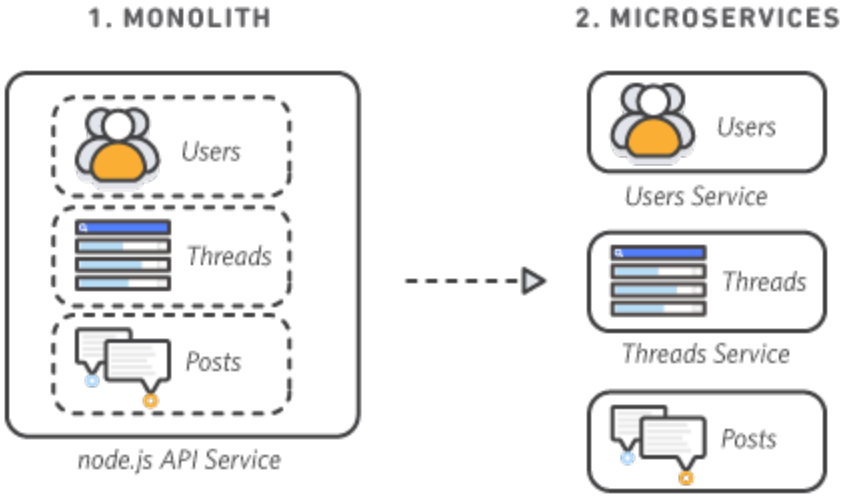
Monolithic vs. Microservices Architecture

- With monolithic architectures, **all processes are tightly coupled and run as a single service.**
- This means that **if one process of the application experiences a spike in demand, the entire architecture must be scaled.**



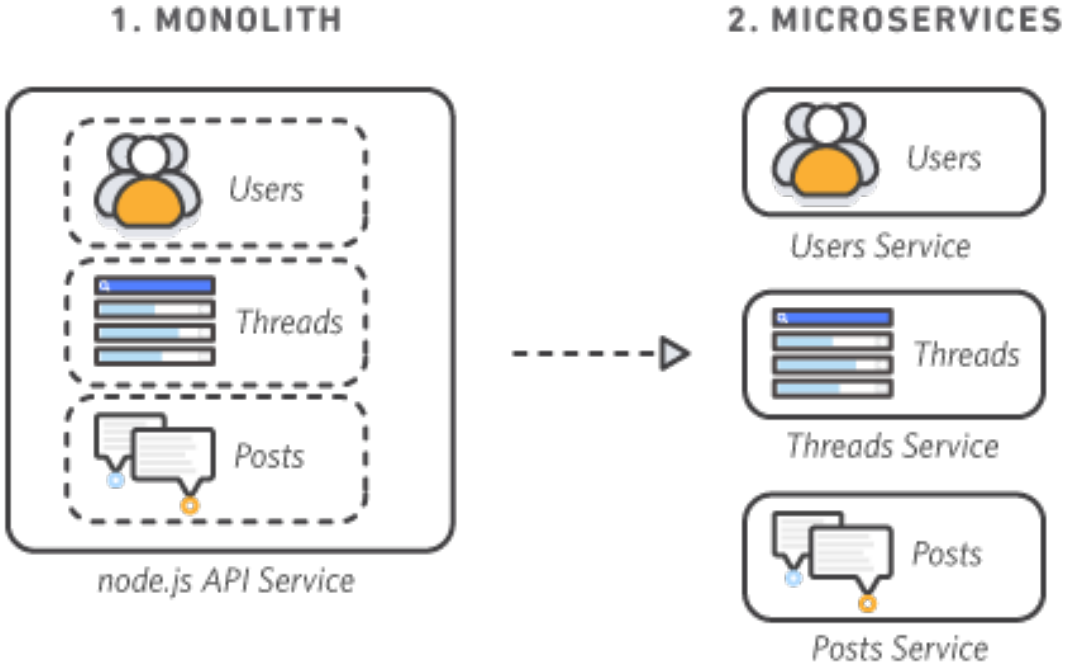
Monolithic vs. Microservices Architecture

- **Adding or improving a monolithic application's features becomes more complex** as the code base grows. This complexity limits experimentation and makes it difficult to implement new ideas.
- Monolithic architectures add risk for application availability because **many dependent and tightly coupled processes increase the impact of a single process failure.**



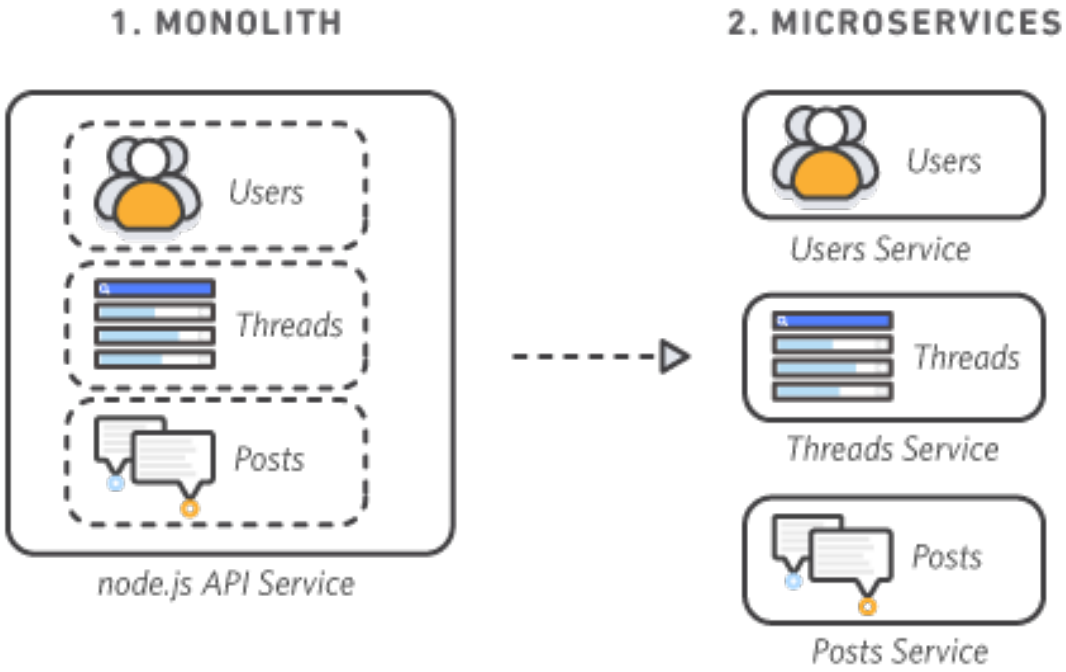
Monolithic vs. Microservices Architecture

- With a microservices architecture, **an application is built as independent components that run each application process as a service.**
- These services **communicate via a well-defined interface using lightweight APIs**.



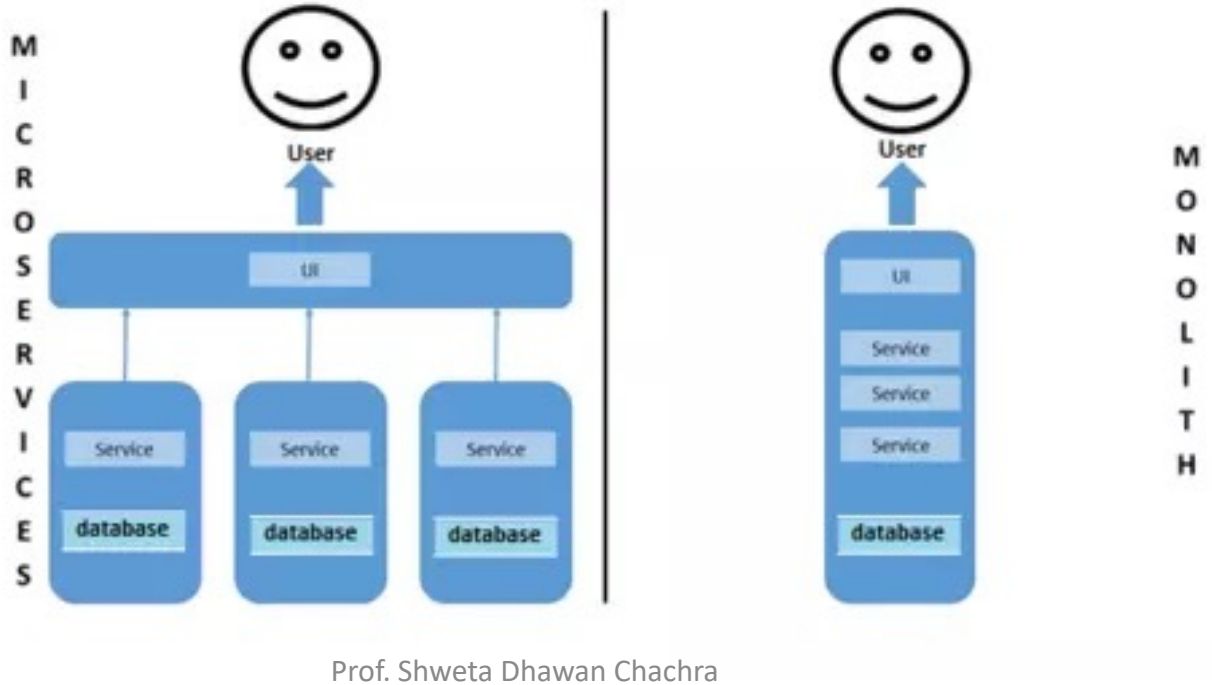
Monolithic vs. Microservices Architecture

- Services are built for business capabilities and each service performs a single function.
- **Because they are independently run, each service can be updated, deployed, and scaled to meet demand** for specific functions of an application.



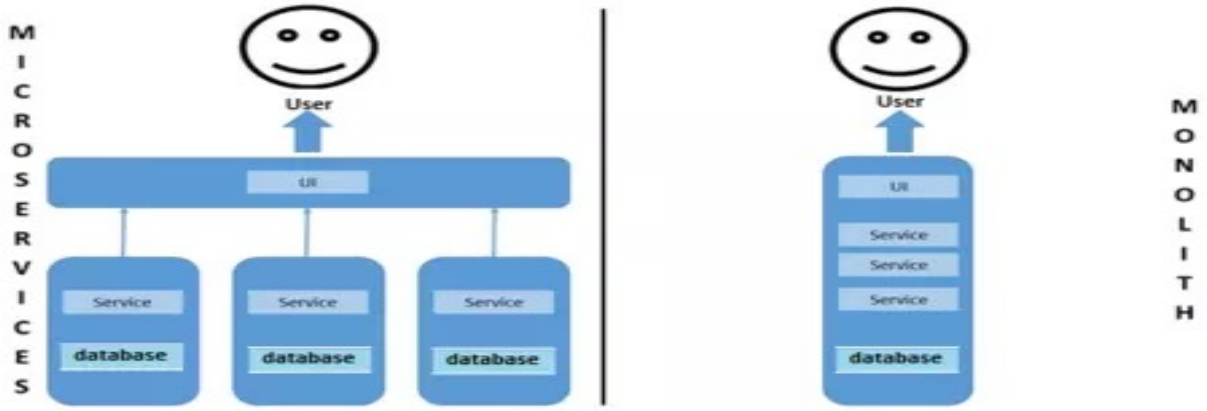
Monolithic vs. Microservices Architecture

- While the **monolith tries to combine everything in one application,**
- Microservices are **only responsible for one task and run independently of each other.**



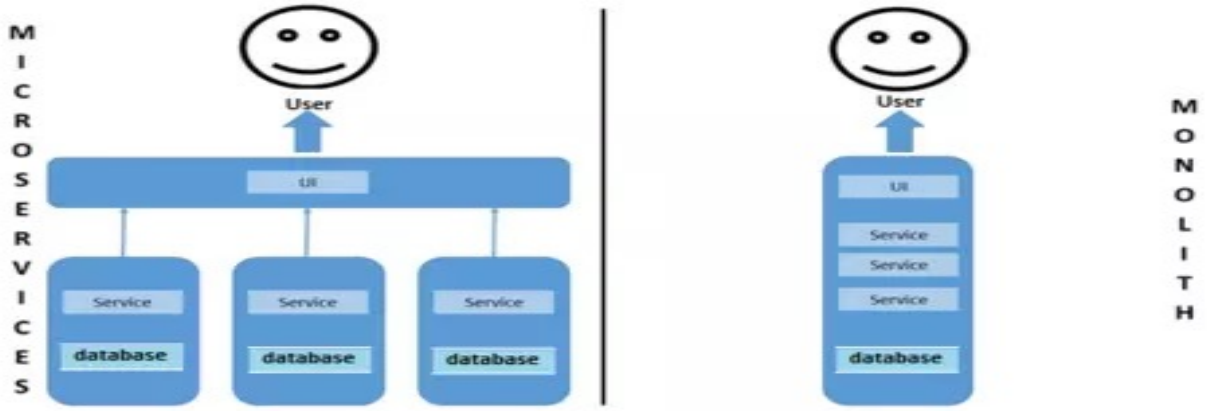
Monolithic vs. Microservices Architecture

- **Traditional program development** works according to monolith principles: all tasks are undertaken in one large project.
- **All individual services access a large database and are output through a user interface – everything is done in one application.**



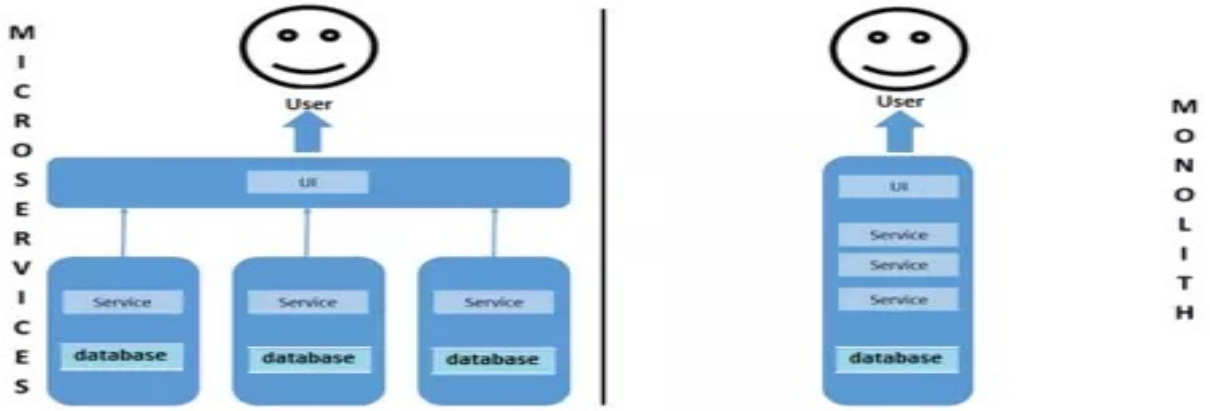
Monolithic vs. Microservices Architecture

- **The microservices approach** is based on modules:
- **Each microservice is responsible for completing a simple task. Each work process is vastly different, as are often the results.**



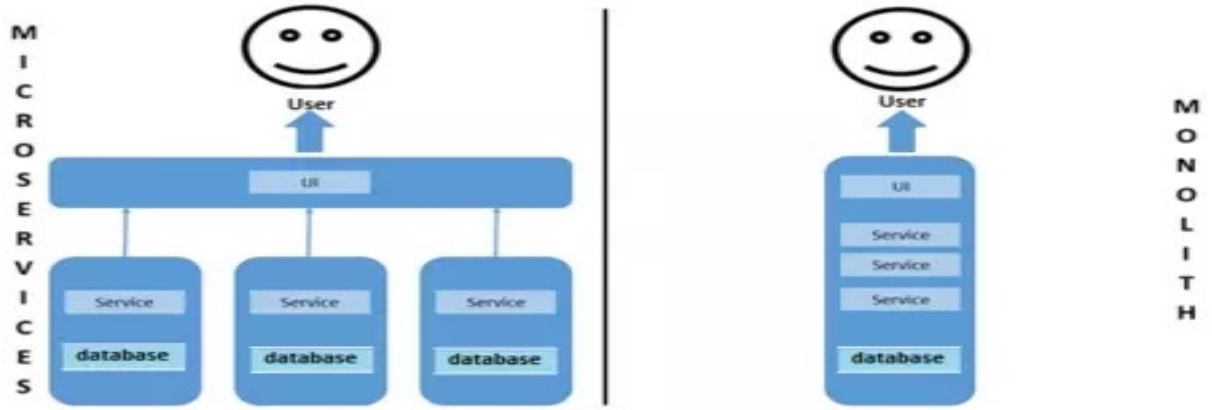
Monolithic vs. Microservices Architecture

- When it comes to **microservice architecture, each team is responsible for their own microservice,**
- **whereas the structure is very different with monoliths**.



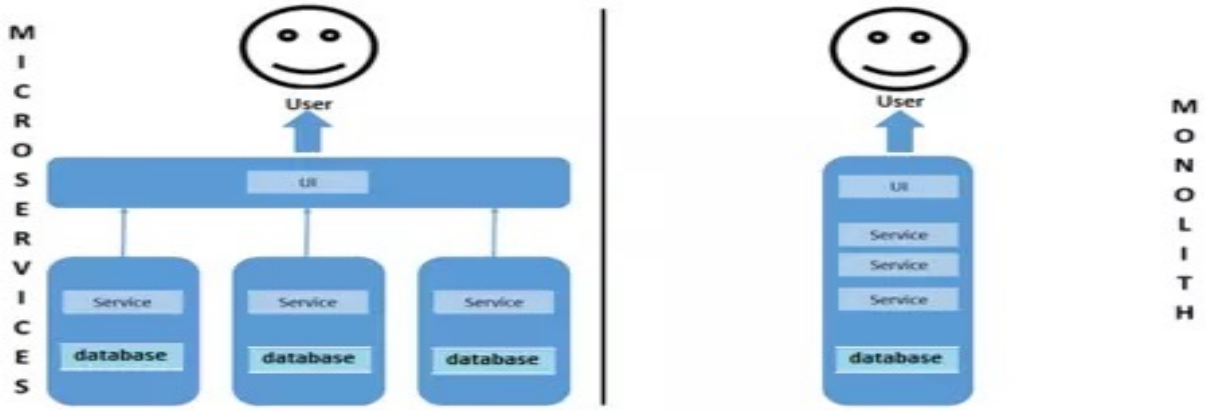
Monolithic vs. Microservices Architecture

- Teams are organized according to the technology they deal with:
- **one team deals with databases,**
- **another one programs the individual services, and**
- **a third one deals with the design of a user interface.**
- **Other working groups are responsible for publishing updates, maintenance, and analysis.**



Monolithic vs. Microservices Architecture

- However, **all teams in a monolith are interdependent.**
- **Dependencies in microservice architecture should be avoided as much as possible.**



Characteristics of Microservices

Autonomous

- **Each component service** in a microservices architecture can be **developed, deployed, operated, and scaled without affecting** the functioning of other services.
- **Services do not need to share any of their code or implementation with other services.**
- Any **communication** between individual components happens **via well-defined APIs.**

Characteristics of Microservices

Specialized

- Each service is designed for a set of capabilities and **focuses on solving a specific problem.**
- **If developers contribute more code to a service over time and the service becomes complex, it can be broken into smaller services.**

Benefits of Microservices

Agility

- Microservices foster an organization of small, independent teams that take ownership of their services. Teams act within a small and well understood context, and are empowered to work more independently and more quickly. **This shortens development cycle times. You benefit significantly from the aggregate throughput of the organization.**

Flexible Scaling

- Microservices allow each service to be **independently scaled to meet demand for the application feature it supports**. This enables teams to **right-size infrastructure needs,** accurately measure the cost of a feature, and maintain availability if a service experiences a spike in demand.

Benefits of Microservices

Easy Deployment

- Microservices enable **continuous integration and continuous delivery, making it easy to try out new ideas and to roll back if something doesn't work.**
- **The low cost of failure enables experimentation, makes it easier to update code**, and accelerates time-to-market for new features.

Technological Freedom

- Microservices architectures don't follow a "one size fits all" approach.
- Teams have the **freedom to choose the best tool to solve their specific problems**. As a consequence, **teams building microservices can choose the best tool for each job.**

Benefits of Microservices

Reusable Code

- Dividing software into small, well-defined modules enables teams to use functions for multiple purposes. **A service written for a certain function can be used as a building block for another feature**. This allows an application to bootstrap off itself, as developers can create new capabilities without writing code from scratch.

Resilience

- Service independence increases an application's resistance to failure. **In a monolithic architecture, if a single component fails, it can cause the entire application to fail.** With microservices, applications handle **total service failure by degrading functionality and not crashing the entire application.**

Working with microservices: 3 examples

- **Microservice architecture has now found its way into large company systems.**
- The companies have subsequently been able to fix certain problems or optimize their processes.
- **Some of them were already using modular systems when there was no term for them.**

Working with microservices: 3 examples

- Large companies with established monolithic systems changing to a microservice model.

Examples like

- **Netflix,**
- **Spotify,**
- **eBay**

- Other major IT companies like Google and Amazon also work like this.

Netflix

- Like many other companies, **Netflix used to be based on a monolithic system (back when Netflix was not an online streaming service, but only sent DVDs through the mail).**
- In 2008, there was **an error in a database that caused the entire service to fail for four days.**

Netflix

- The decision was then taken **to break up the old system and split them into microservices.**
- The result was that the company was able to
  - **make live changes much faster, and**
  - **repairs were carried out much more quickly.**

Netflix

- Since the Netflix system is enormously extensive, **a separate program was developed to organize the individual microservices among themselves:**
- **Conductor**
- Conductor **grants Netflix central control to pause, restart or scale their microservices.**

https://www.ionos.com/digitalguide/websites/web-development/microservice-architecture/

Spotify

- **The audio streaming service Spotify also relies on microservices.**
- Spotify's main daily development challenge is keeping ahead of the strong competition.
- The audio streaming services market has some of the largest IT companies in the world as its **main players – such as Amazon, Apple, and Google.**

Spotify

- **Due to the increasing number of users, Spotify developers are constantly having to meet higher demands** and comply with certain business rules (like licensing rights).
- Microservices are a good solution for Spotify, allowing them **to react quickly to new developments their competitors might make,** and **publish their own developments faster – forcing the competitors to react in turn.**

Spotify

- For example,
- **Spotify feature that recommends suggestions to users when they type in a search term is a self-contained microservice** that has its own dedicated team working on it.
- Additionally, Spotify benefits from the robust nature of microservice architecture: **if a single microservice fails, it does not mean that the entire product becomes unusable**.

Spotify

- There are more than **800 microservices** active within Spotify, and **they use Java for a large part of those microservices.**
- However, this has nothing to do with the fact that microservices cannot be written in different programming languages: instead, it has to do with work processes.
- **Developers constantly move from one team to another, and it is easier when everyone uses the same language.**

eBay

- Like many other large systems,
- the eBay sales platform **began as a monolith:**
- **eBay had 3.4 million lines of code in just one file**.
- The company then **decided to break up the monolith and develop Java microservices instead.**
- Individual eBay services **also use REST to communicate with one another.**

Inference

- The fact that eBay and other companies have successfully **gone from a monolithic to microservice architecture is a clear sign of the benefits of a more modern approach.**
- While the monolith structure is perfectly **sufficient in the early days of a website, with a small number of active users and a manageable range of products, it can become growth-inhibiting when demands start to increase.**