Microservices is an approach to software design that decomposes functionality into small autonomous services. This allows large scale functionality to be implemented with manageable units. The following are architecture and design principles associated with microservices.

Small

Microservices are small. They are implemented and operated by a small team.

Replaceable

A microservice is small enough to be replaced without much impact.

Cohesive

Everything in a microservice feels like it belongs together. Anything that doesn't fit is moved to a separate microservice.

Autonomous

Microservices can be deployed independently.

Scalable

Microservices are designed to be scalable and can be scaled independently.

Disposable

Design for fast starts and don't allow a service to fall into an unstable state when hardware suddenly fails.

Dumb Pipes

Ideally, microservices communicate using standard asynchronous network communications. Avoid putting functionality in the middle between services.

Loosely Coupled

Microservices offer an abstracted interface such that consumers don't need to change every time you deploy an update. Design interfaces around enduring business concepts.

Reusable

Interfaces are designed to maximize use of services by diverse consumers. Avoid unnecessary assumptions and constraints.

Business Functionality

Microservices are organized around business functionality as opposed to technology layers.

Cross-functional Teams

Microservices are completely implemented by small cross-functional teams that can change everything from the UI to the data model. This removes the organizational complexity of getting numerous teams involved in every change.

Decentralized Architecture

Allow teams leverage in choosing their own platforms, tools and data models. Encourage teams to share what works.

Ownership

Microservices are products that are owned and operated by the build team.

Emergent Design
Microservices are evolved over time with no big upfront plan.

Resilience

Expect failure from external resources such as other microservices and behave well when failure occurs.

Automation

Automate things such as testing, deployment and service recovery.