title: Microservices notebook: Distributed Systems layout: note date: 2020-08-24 tags: …

Microservices

- **microservices**: small, autonomous services that work together

Small, focused on doing one thing well

- cohesion: related code grouped together
- *Single responsibility principle*: gather together those things that change for the same reason, and separate those things that change for different reasons.
- small enough, and no smaller: probably you have hit this point when a piece of code no longer feels too big
- small enough to be managed by a small team
- decreasing size maximises both benefits and drawbacks of microservices: increased interdependence, increased complexity

Autonomous

- all communication between services are via network calls to enforce separation and prevent tight coupling
- services need to be able to change independently of each other
- services should be able to be deployed by themselves without requiring consumers to change

Benefits

- **technology heterogeneity**: you can pick the right tool for each job, instead of standardised, lowest common denominator, one-size-fits-all approach. You can quickly absorb new technologies.
- **resilience**: in a monolithic service, if the service fails, everything stops working. Microservices allow you to build systems that handle total failure of services and degrade functionality
- **scaling**: monolithic services need to everything to be scaled together. Microservices can be scaled individually as needed
- **ease of deployment**: you can change a single service and deploy it independently of the rest of the system. Faster deployment and rollback.

- **organisational alignment**: you can align the architecture to the organisation, and minimise the total number of people working on any one codebase to hit the sweet spot for team size
- **composability**: imporved reuse of functionality
- **optimising for replaceability**: microservices are small enough that they can be reimplemented better or deleted if no longer in use