Operation: Micro or Macro Architecture?

In this lesson, we'll discuss some factors that influence the operation of applications.

We'll cover the following

Some decisions in the area of micro and macro architecture mostly influence the operation of the applications. Let’s take a look at a few.

Configuration

We must define **the interface with which a microservice obtains its configuration parameters**. For example, a microservice can get these settings via an environment variable or read them from a configuration file. These parameters include both:

- **Technical** parameters such as thread pool sizes
- Parameters for the **domain logic**

The decision of how to store and generate the configuration data is independent of these parameters. The data can be stored in a database, for example. Either configuration files or environment variables can be generated from the data in the database.

Note that the information on which computer and under which port a microservice can be reached, **does not** belong to the configuration, but to the **service discovery**.

Configuring passwords or certificates is also a challenge that can be solved with other tools. To do this, Vault (https://www.vaultproject.io/) is a good choice because this information must be stored in a particularly secure way and must be visible to as few employees as possible in order to prevent unauthorized access to production data.

Monitoring

Monitoring is about the **technology that tracks metrics**. Metrics provide information about the state of a system. Examples include the number of requests processed per second or business metrics, such as revenue.

The question of *which* technology is used to track the metrics is independent of which metrics are captured. Every microservice has different metrics because every microservice has different challenges. For example, if a microservice is under a very high load, then performance metrics are useful.

Log Analysis

Log analysis defines a **tool for managing logs**.

Although logs were originally stored in log files, they are now stored on **specialized servers**. This has a few advantages. For example, it makes it **easier to analyze and search the logs**, even with large amounts of data and many microservices.

In addition, new instances of a microservice can be started when the load increases and can be deleted again after the load decreases. In this case, the logs of this microservice instance should still be available, even if the microservice was deleted long ago due to a decreasing load. If the logs are stored only on a local device, the logs would be gone after the microservice has been deleted.

Deployment Technology

Deployment technology determines **how the microservices are rolled out**. For example, this can be done with Docker images (see chapter 6 (https://www.educative.io/collection/page/10370001/6518081205567488/6331908113825792)), Kubernetes Pods, a PaaS, or installation scripts.

These decisions define how a microservice behaves from an operational point of view. Typically, these decisions are either all part of the macro architecture or the micro architecture.

Macro architecture operation with separate operations teams

Whether decisions in the area of operation belong to micro or macro architecture **depends on the organization**.

For example, a team can develop microservices but bear no responsibility for their operation. The operations team is responsible for the operation of all microservices. In this scenario, decisions for operation must be made at the level of macro architecture.

It is generally unacceptable for the operations team to learn a different approach for the operation of each microservice, especially because the number of microservices is much larger than the number of deployment monoliths for the same project.

Another reason for a macro architecture decision on the operation of microservices is that individual solutions in this area bring few advantages. Although a programming language or framework can be more or less suitable for a particular problem, the same applies to a much lesser extent to technologies in the field of operation.

Standardize only technologies!

When these decisions are made at the level of macro architecture, **they standardize only the technologies**.

Which configuration parameters, monitoring metrics, log messages, and deployment artifacts of a microservice are a decision at the level of the individual microservice.

The independent deployment must also be retained as a core feature of the microservices. This means that it must be possible to independently change the configuration parameters for each microservice in order to adjust the configuration when a new deployment takes place.

Testing the operation macro architecture

Adherence to the macro architecture rules can be checked with tests. The microservices are deployed in an environment. The tests check whether the rules for uniform deployment are adhered to. The test then verifies whether the microservice delivers metrics and log information in the defined way. Something similar is also possible for configuration.

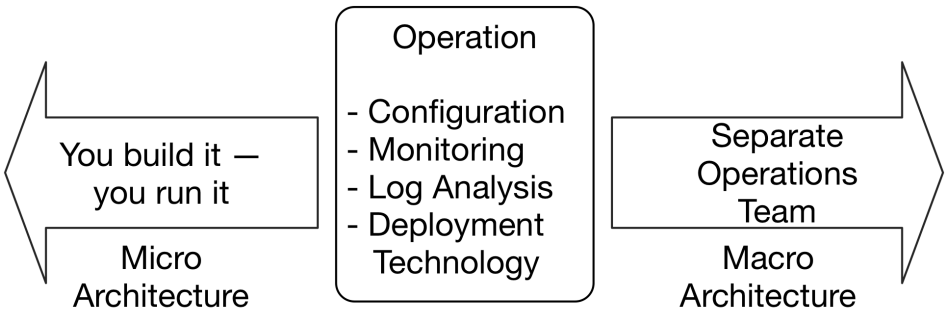
The test environment for these tests should be very minimalistic and should not contain any other microservices or a database. In this manner, the microservice is tested in an environment in which it cannot possibly work. When such a situation occurs in production, it is particularly important that the microservice provides logs and metrics to analyze potential problems. Similarly, the test also checks the resilience of the microservice.

“You build it, you run it”: operation as micro architecture

There is a form of organization in which operational aspects have to be part of the micro architecture. If the same team is to develop and operate the microservice, they must also be able to choose the technology. This approach can be described as “you build it, you run it”. The teams are each responsible for a microservice, for its operation *and* development. You can only expect this level of responsibility from the team if you allow them to choose their own technologies.

Operation as a whole is micro or macro architecture

Decisions for operation can be taken either at the level of micro or macro architecture. Making operation decisions part of the macro architecture is useful if there is a separate operations team, while a “you build it, you run it” organization must make these decisions at the level of micro architecture. The drawing below illustrates this point.



Depending on the organization, operation is part of the micro or macro architecture

QUIZ

1

Which of the following best describes why storing logs on a specialized log server is adventageous?

A) Since servers are always less prone to crashes than local machines, the data is safer and less prone to be lost due to a crash.

B) The logs from all microservices would be available regardless of their current status and searching and analyzing logs is simpler.

C) Servers are always faster than local machines, hence storing logs will be made efficient on a server.

Submit Answer

Question 1 of 2
0 attempted

Reset Quiz

In the next lesson, we'll discuss the reasons for making as many decisions as possible at the micro architecture level rather than the macro architecture level.

← Back

Next

Architecture Decisions

Give a Preference to Micro Architecture!

Mark as Completed

Report an Issue