1.What are microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

• Highly maintainable and testable

• Loosely coupled

• Independently deployable

• Organized around business capabilities

• Owned by a small team

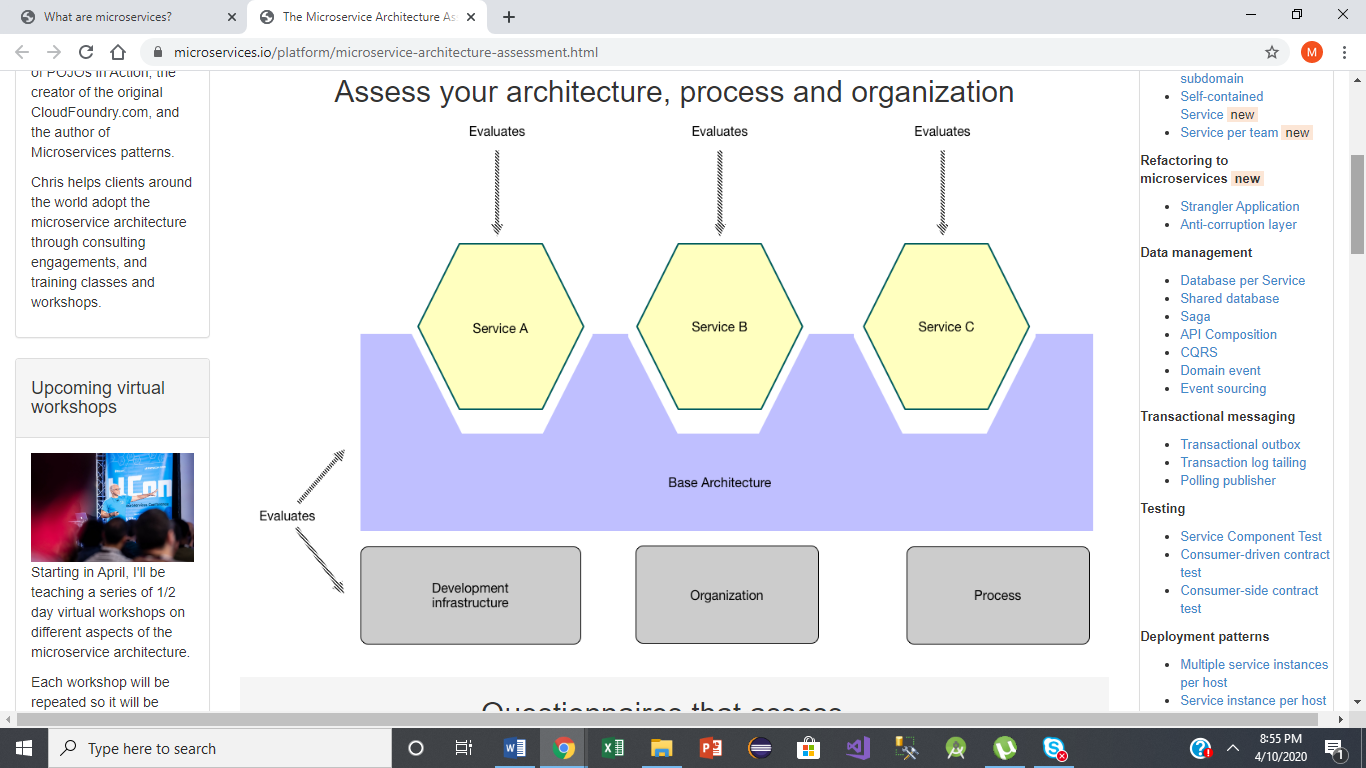
microservice apps consist of multiple independent components that are glued together with APIs.

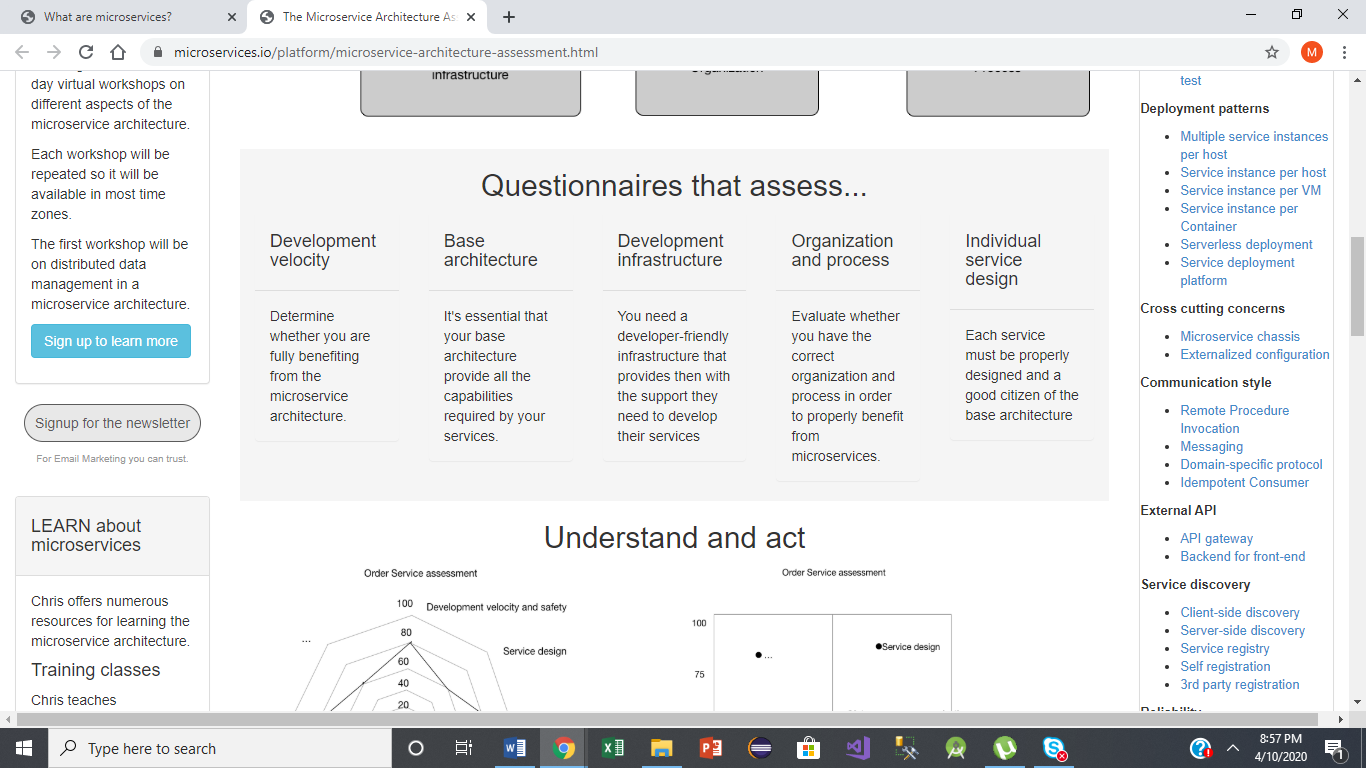
2.The pattern language is your guide

The microservice architecture pattern language is a collection of patterns for applying the microservice architecture. It has two goals:

3.Assess your architecture

If you have built an application with the microservice architecture then you must asses it in a proper way. The platform assesses what you have built and identifies what needs to be improved. It reduce architectural and organizational risk and maximizes the benefits of the microservice architecture





4.Applying the microservice architecture pattern language

The microservice architecture pattern language consists of numerous groups of patterns. The value of a pattern language exceeds the sum of it’s individual patterns because it defines these relationships between the patterns:

• Predecessor – a predecessor pattern is a pattern that motivates the need for this pattern. For example, the Microservice Architecture pattern is the predecessor to the rest of the patterns in the pattern language except the monolithic architecture pattern.

• Successor – a pattern that solves an issue that is introduced by this pattern. For example, if you apply the Microservice Architecture pattern you must then apply numerous successor patterns including service discovery patterns and the Circuit Breaker pattern.

• Alternative – a pattern that provides an alternative solution to this pattern. For example, the Monolithic Architecture pattern and the Microservice Architecture pattern are alternative ways of architecting an application. You pick one or the other. These relationships provide valuable guidance when using a pattern language. Applying a pattern creates issues that you must then address by applying successor patterns. The selection of patterns continuously recursively until you reach patterns with no successor. If two or more patterns are alternatives then you must typically pick just one. In many ways, this is similar to traversing a graph.