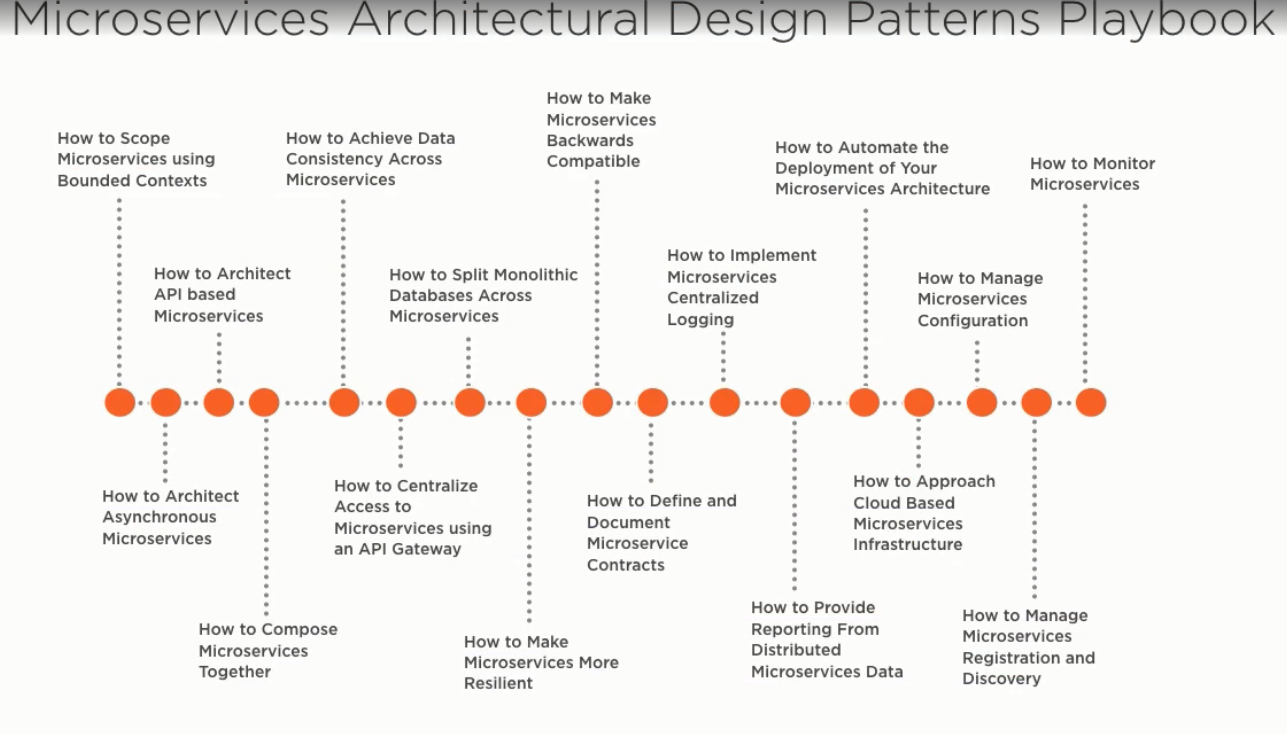
Course Overview:

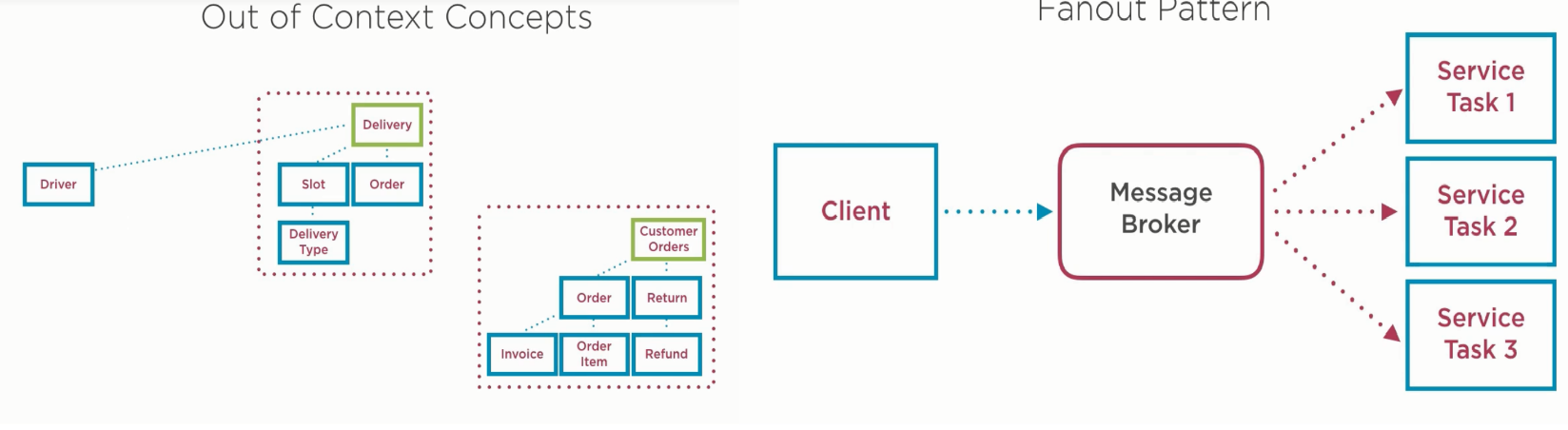


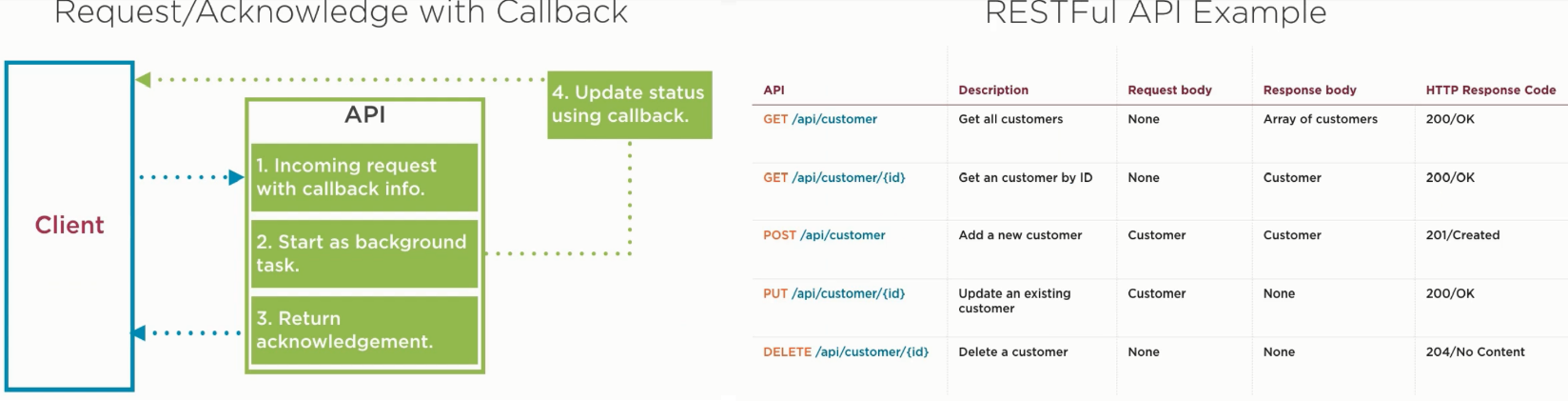
In this course we will look at design patterns and practices that help implement a successful microservice architecture, so over the last few years other software engineers responsible for software architecture have had the same design questions, worries and interest to get microservices right as you have, so we will implement the microservices in the most effective way based on a lot of good practices and design patterns over the last years.

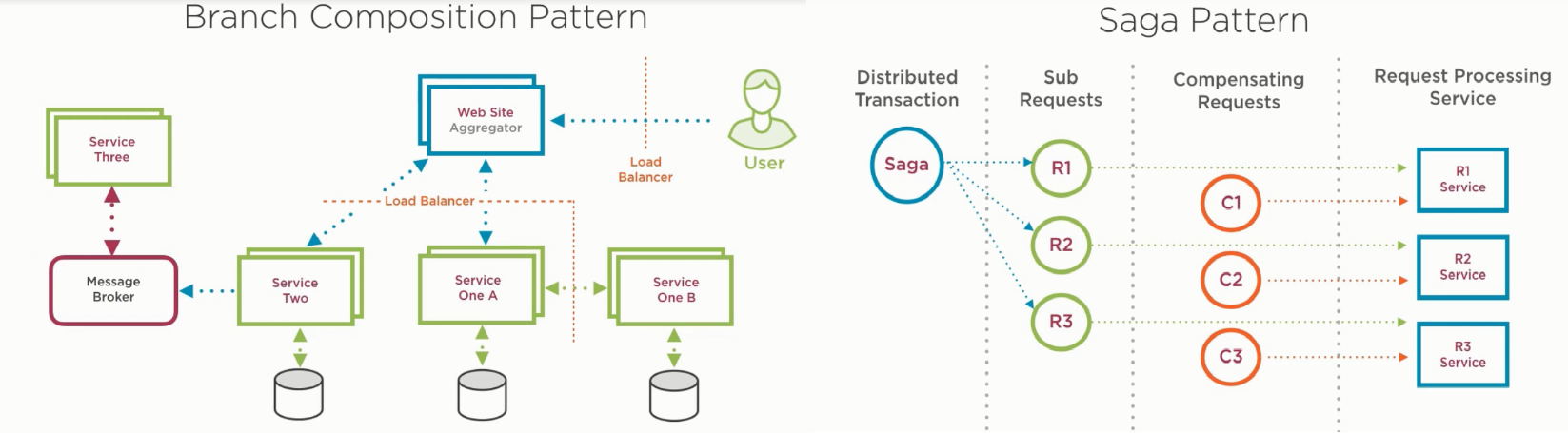
The next stage after understanding this design principles is to implement them using an effective tool set.

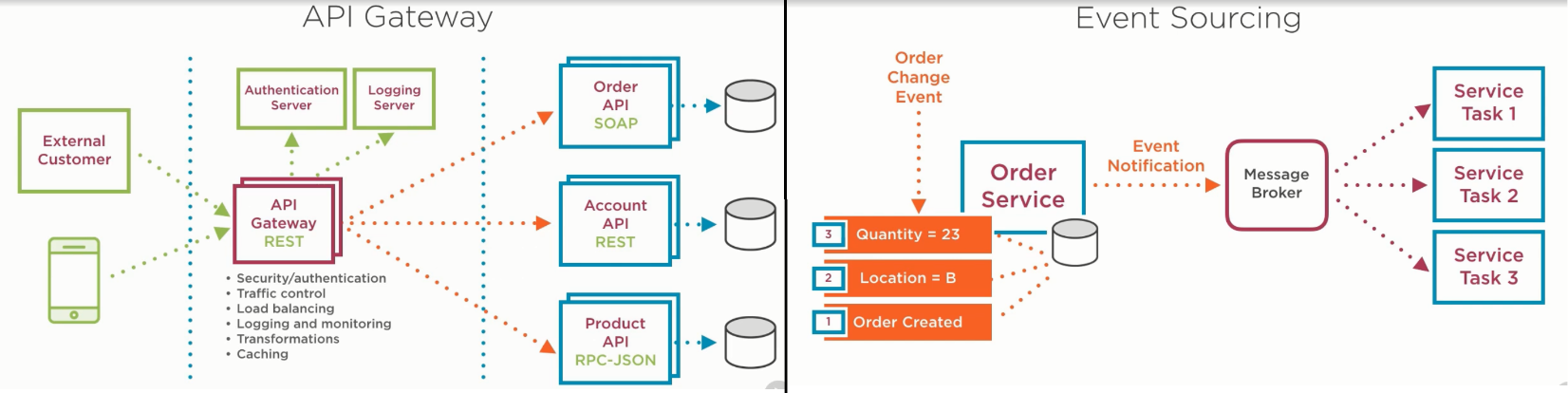
The effective tool set needs to consist of practical design patterns, and we will list them all in this course to started implementing a highly effective microservices system.

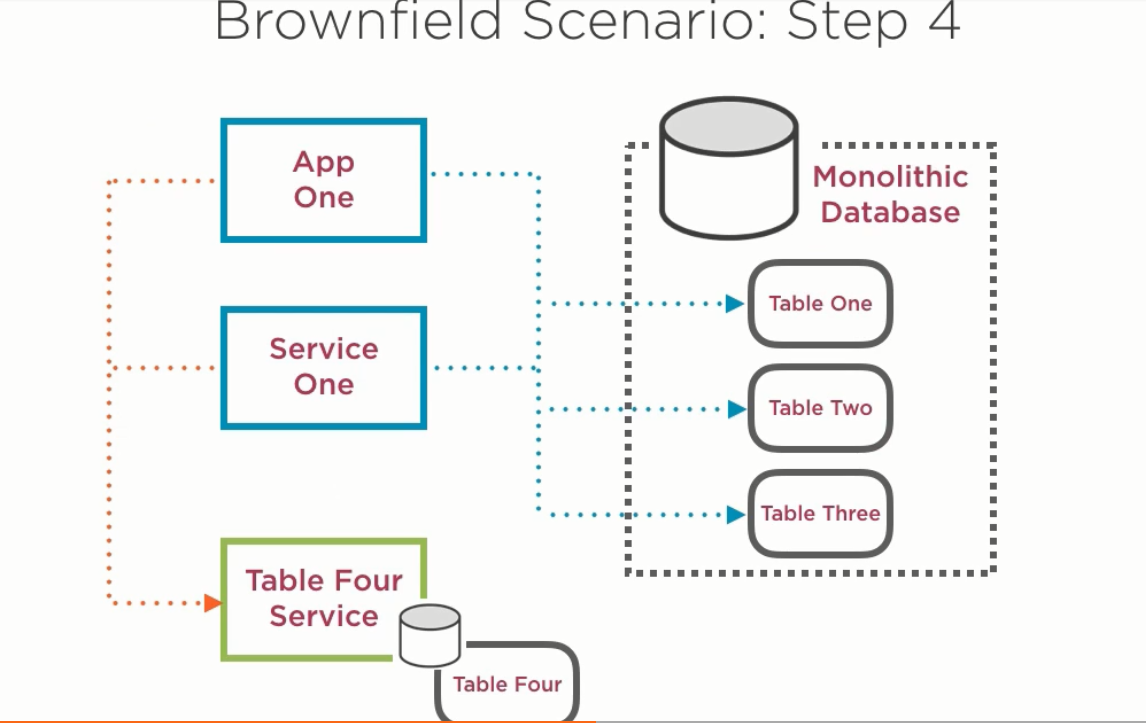
We will try to answer all the needed questions for building a great microservice architecture.

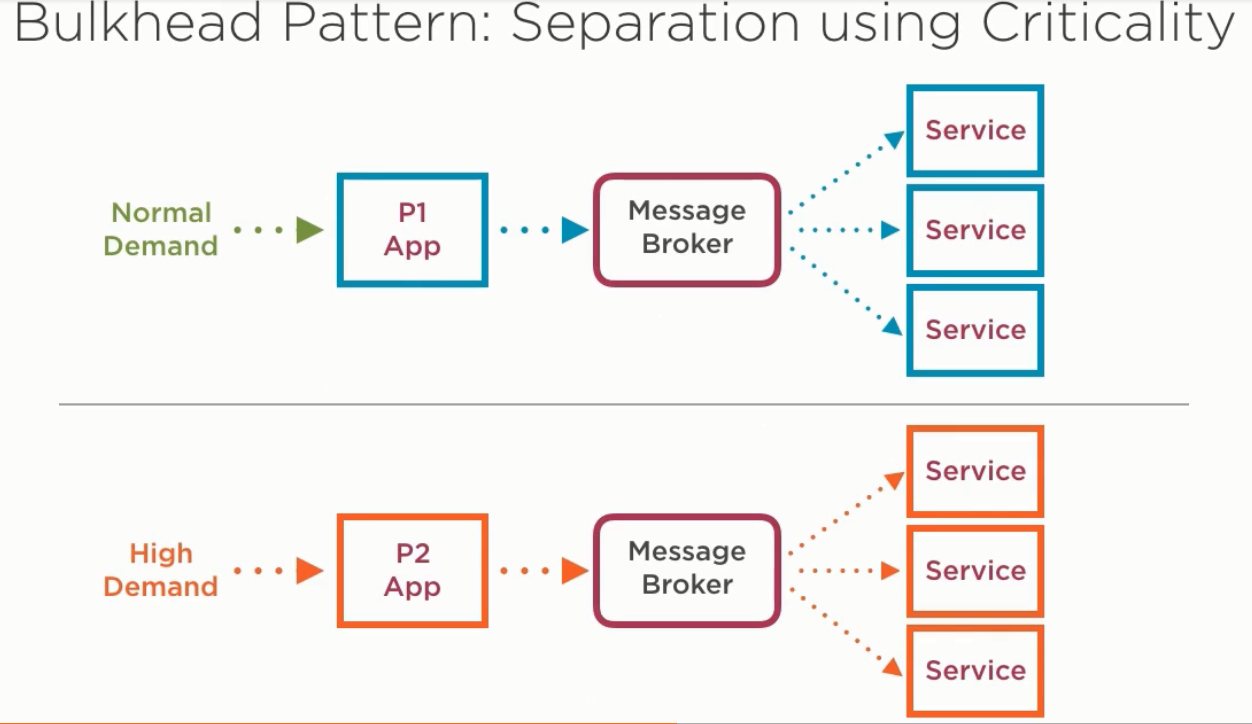


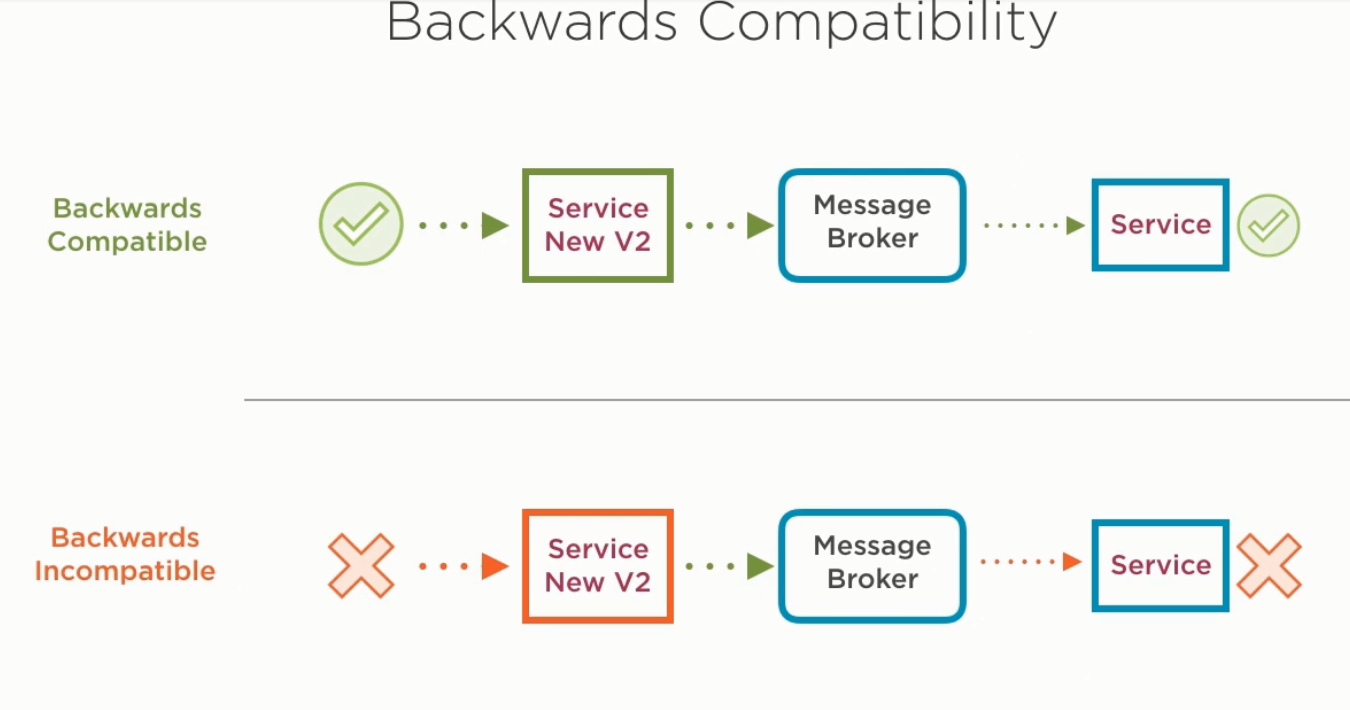


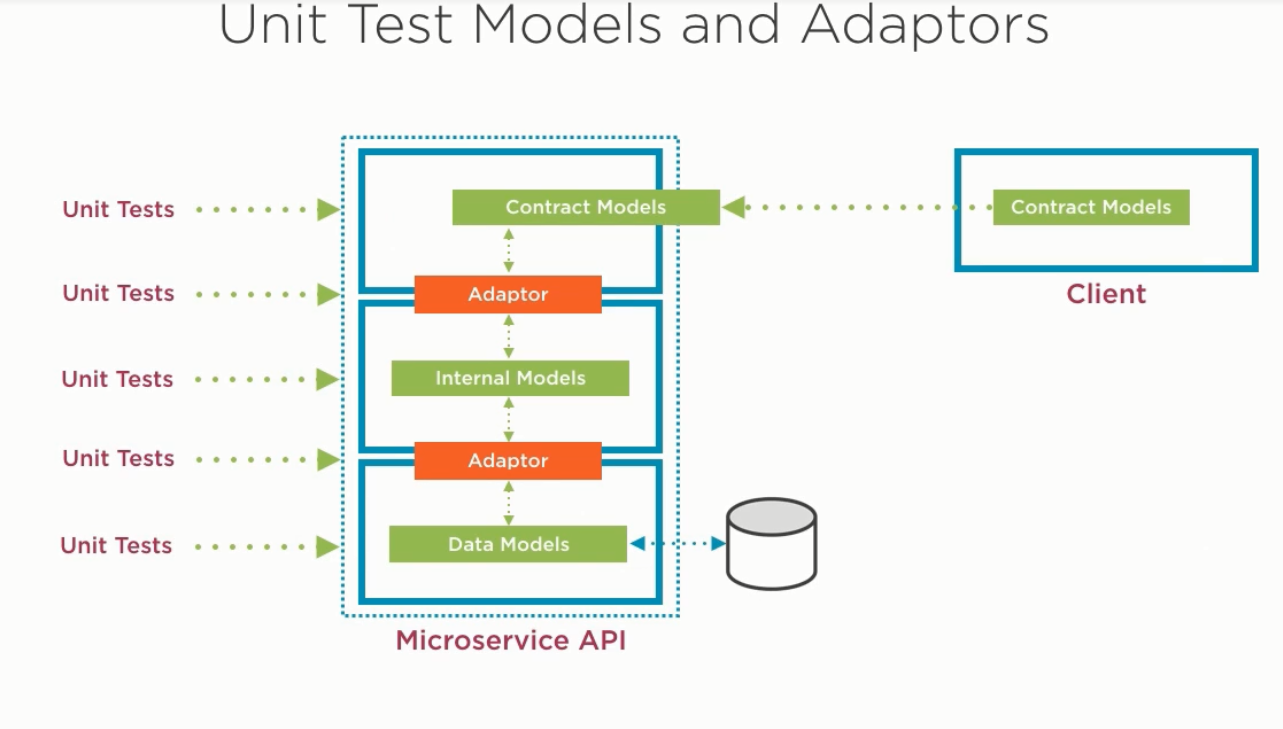


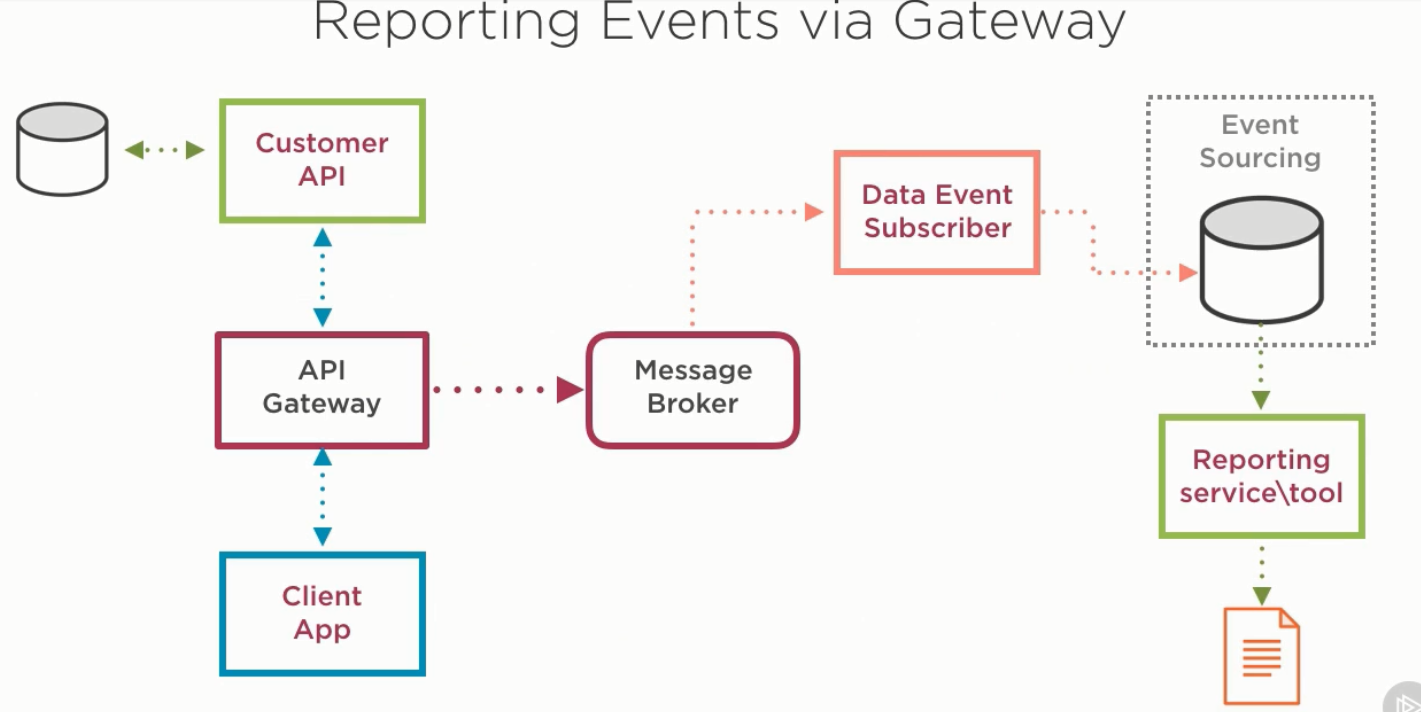


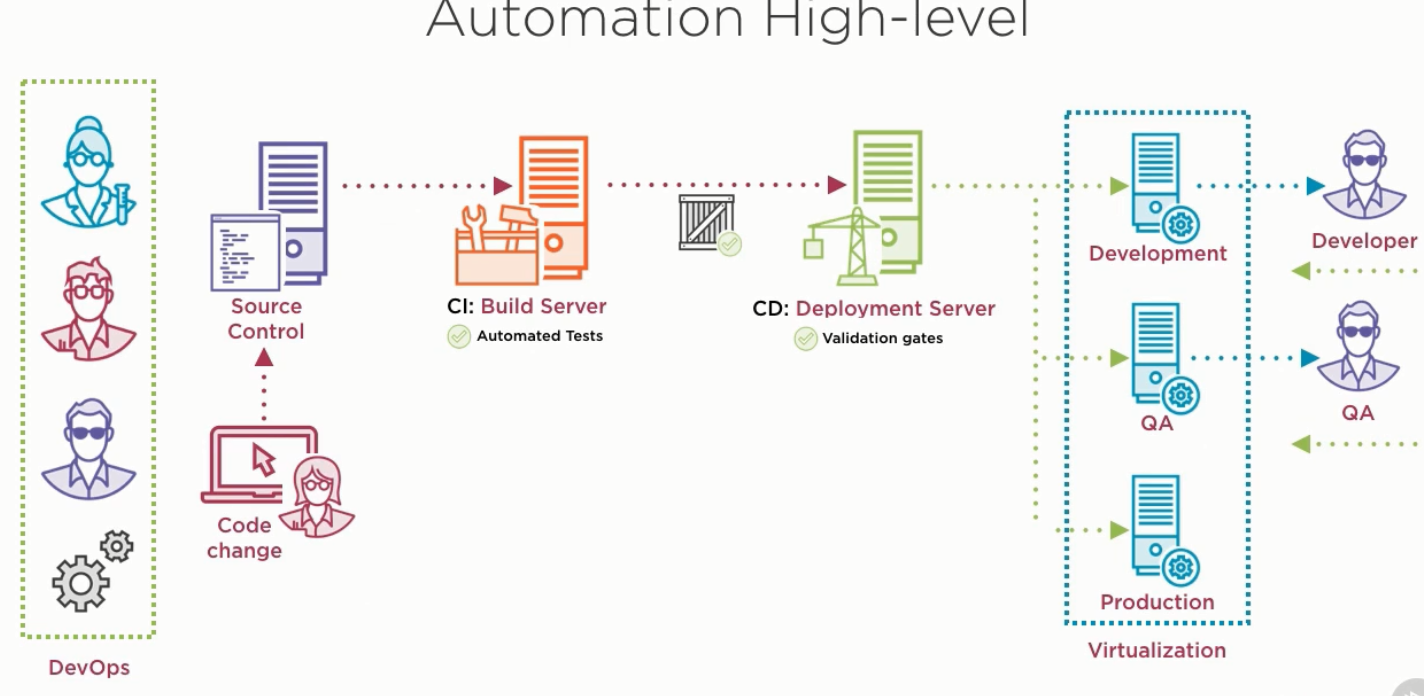






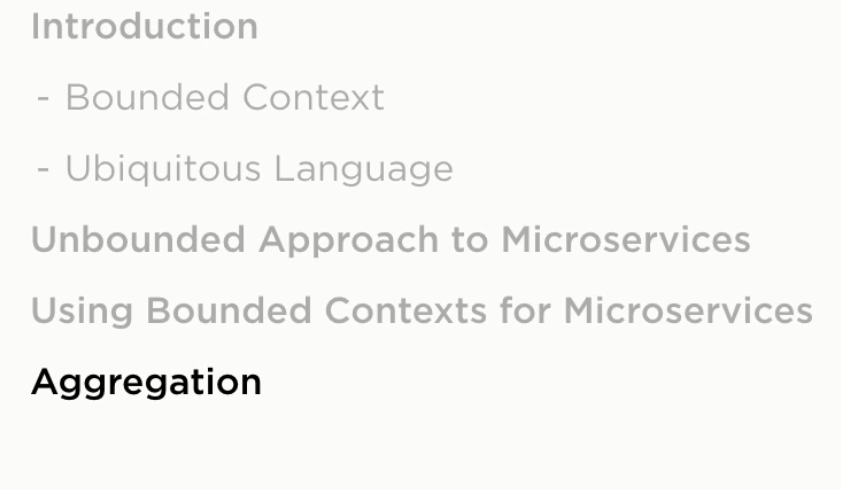






How to scope microservices using bounded contexts:

1. Introduction:



At this section we will see how the bounded context technique can be used as strategic design pattern to size and scope our microservices.

So, we will start learning how to use bounded context to scope and size microservices, and how to use the ubiquitous language to define bounded context and we will look at the disadvantages of not using bounded contexts approach to define microservices. Then we will see a real-time example of how to use bounded context to scope microservices. Then we will see the benefits of aggregation which is the need in sometimes to combine services for specific reasons.

1. Need for scoping microservices:

* We will look on how it’s important to size software components correctly.
* It’s important to scope microservices correctly because we need to remember that the microservice design principles are related to competitive software design which means we should produce software that is easy to change and deploy with minimum change risk and this is made up by producing software that is made of small loosely coupled, high cohesive components.
* How we know exactly what is the size and scope of the microservice component? We are using techniques like Bounded context which is browed from DDD