Reading the Gartner paper on “Innovation Insight for Microservices”, the part of the article that stood out to me the most was the repeated emphasis of a few themes as they relate to microservice architecture (MSA) and its effect on applications: agility, flexibility, scalability, and disruption. It is these themes that appear to direct analysts Anne Thomas and Aashish Gupta to make clear from the get-go that microservices are not the end-all-be-all of software architecture, that (like we discussed in class) the utilization of microservices or the transition of an existing monolith code base to support microservices is one option if certain prerequisites are met.

As stated in the paper, “The overarching principle governing MSA is independence,” and with the decreased dependencies that Microservice Architecture offers appears to come an increased reactivity of software to the needs and requirements of an organization, product, or project. This reactivity allows for agility, flexibility, and scalability. In terms of providing agility with continuous deployment, I believe the article portrays the partnership between CD software delivery and microservices as two-fold, where microservices can be used to implement a CD practice in order to help negate the risks of combining CD and monolithic code (new features can be released in independent pieces rather than within a single code base where it would be more difficult to uncover the cause of a failure), and CD then allows batch releases to be avoided (i.e. faster turnaround time from features being developed to being deployed and saving time/money in testing). With this relationship, it makes sense why many leading-edge companies are moving towards microservices in order to support continuous deployment. Furthermore, for scalability, the reactivity inherent in microservices’ independence means that if parts of an application need to be scaled to prevent issues like bottlenecks, just those points can be reduced or grown effectively, again appearing to save an organization time and money (both in the potential consumer/monetary side effects of having a system that is not scaled appropriately for demand or in having to overhaul a monolithic code base). Also, the fact that microservices are loosely coupled independent units means the flexibility for different development teams to use different technologies/languages in various parts of an application’s architecture, so microservices allow the teams to “react” to the needs of the product or project.

However, the nature of MSA and its goal to allow teams to build, package, deploy, scale, and test pieces of an application independently means that their adoption is disruptive to parts of an organization. As the Gartner paper points out, the benefits of microservices come at the cost of necessary change to technology, organizational culture, infrastructure, data management (MSA impacts reports, analytics practices, and data governance), and mindset (switching to product mindset, which could affect budgeting, funding, and responsibilities, as well as the format of development teams and corporate communication). Changes like these require leaders and teams who are

willing to work with these changes to plan and facilitate responses to their effects, as well as organization members being open to changing management strategies as teams need to have the autonomy and authority to have control over the development of their components of the application. In addition, microservice architecture requires teams to be well-versed in DevOps and agile practices.

Besides these main themes, other aspects of the Gartner report I found interesting were the differentiation between the levels of granularity when comparing a microservice, miniservice, and macroservice, as well as the illustration of each of these implementations through the check-out service on an e-commerce application; the Gartner report explains that a macroservice would be deployed within a monolithic application and have an API for the check-out functionality, a miniservice would have the check-out role be in an unit that can be deployed independently, and the microservice would divide the checkout functionality into separate units that each complete a separate job and are independently deployable. Also, especially as companies are deciding whether their architecture, culture, and technology are a good fit for the adoption of microservices, I think the paper's acknowledgement that the term "microservices" is often confused with other terms like "reusable services," "any service with a REST API", or "any software component" is important. The most informed decision cannot be made on whether or not the organization meets the Gartner paper's prerequisites for MSA adoption unless all involved have an accurate baseline of knowledge.

Unlike microservices themselves which exist as independent units with MSA working to minimize dependencies between components, the decision to adopt microservices is anything but independent from outside consideration. As the Gartner paper makes clear, shifting to MSA is an investment that should not be carried out just because it is the "next big thing." It is a balancing act between the potential benefits and costs, and the exploration into MSA should be completed slowly in steps, working through macroservices to miniservices to microservices, only using microservices where necessary (i.e. need agility or scalability that could not be provided otherwise) and using a combination of microservice, monolithic, and miniservice architectures if needed.