Key Things to Consider — Runscope Blog

This is the first post in our Featured Guest Series! Janet Wagner shares eight tips on how your company can ensure success when moving to a microservices architecture.

If you're interested in being a part of our next series, fill out this short form and we'll get in touch with you for our next run.

Microservices are all the rage these days, and you can find blog posts on most technology news sites and blogs about the topic. Major technology companies like Amazon and Netflix have been evangelizing microservices for quite some time now. The number of businesses building microservices architectures to power web and mobile applications is growing at a rapid pace and with good reason. There are significant benefits for companies running applications on microservices architectures such as high availability, high scalability of both the architecture itself and the engineering teams developing it, and ability to innovate and add new features faster.

If you are one of the many companies that have decided to venture into the world of microservices, there are a lot of things to consider. Do you start with a monolith or microservices architecture? How do you define the scope of each microservice?

Here are eight key aspects that you should consider to ensure you're prepared for building, deploying, and maintaining a microservices architecture.

1. Monolith or Microservices First

There is some debate as to whether it’s best to start with a monolithic architecture first then move to microservices later or just start with a microservices architecture first. Martin Fowler, a software developer and microservices expert, makes a compelling case that it is best to start with a monolith and move to microservices later if warranted.

Stefan Tilkov, co-founder and principal consultant at innoQ, is convinced that starting out with a monolith is usually the wrong thing to do. He argues that breaking up a monolith into separate individual services later on is tough because its components will have eventually become tightly coupled to each other.

Carefully consider the type of architecture that will work best for your applications. The best approach could be monolith first, microservices first, or another type of architecture altogether. If you're considering refactoring a monolith into microservices, there are many aspects to consider beyond the eight highlighted here. NGINX has a nice series of blog posts covering microservices which includes a post about refactoring a monolith.

2. Infrastructure and Operations Investment

Major tech companies like Amazon and Netflix can afford to make significant investments in the infrastructure and operations needed to build, deploy, and maintain microservices. If your company is unable or unwilling to significantly invest in infrastructure and operations, then it is not a good idea to invest in microservices.

3. Peer Code Reviews

One of the challenging aspects of building a microservices architecture is that it often requires many autonomous engineering teams, with each team working on building and deploying one or more services to production. The chances of coding errors occurring are far lower for a company with only a few autonomous engineering teams and microservices than a company with hundreds of microservices and dozens of engineers. Companies should consider conducting peer code reviews regularly which can help reduce the number of coding errors ensuring the success of their microservices architecture.

4. Open Source Software or Build from Scratch

There are quite a few open source software projects that provide tools for building microservices architectures. Netflix provides a number of open source software projects that companies can use to build their own microservices architectures. Netflix isn't the only company doing that; other companies include Capgemini, Mashape, Pivotal, and Red Hat.

Companies can use these tools so that they don’t have to build microservices architectures from scratch; however, it may be necessary to build some components of a microservices architecture from scratch to meet specific application requirements.

5. Service Size

Some companies set a limit on how large each service should be e.g. no more than 100 lines of code. The size of the service doesn’t matter; however, what does matter is that each service does one task, and does that task well. If a service ends up doing more than one task, then the company should consider breaking the service into separate microservices that perform one task each. Each service should also typically only have one datastore. Runscope co-founder and CEO John Sheehan discusses service size, datastores, and other aspects of microservices on his APIDays talk about "Scale-oriented Architecture with APIs".

6. Continuous Delivery

Continuous delivery is essential when it comes to ensuring microservices success. Continuous delivery is an approach to software development where teams release small sets of changes frequently. It requires a DevOps culture, and that much of the delivery process be automated. Releasing small sets of changes frequently reduces the chances that errors will occur and makes it easier to fix errors when they do.

In an article about microservices tradeoffs on his website, Fowler explains:

> “Being able to swiftly deploy small independent units is a great boon for development, but it puts additional strain on operations as half-a-dozen applications now turn into hundreds of little microservices. Many organizations will find the difficulty of handling such a swarm of rapidly changing tools to be prohibitive.”

Fowler goes on to say “This reinforces the important role of continuous delivery. While continuous delivery is a valuable skill for monoliths, one that's almost always worth the effort to get, it becomes essential for a serious microservices setup.”

7. Service Discovery

Service discovery is an important component when it comes to a microservices architecture. Microservices instances are constantly coming and going making it difficult, if not impossible, for engineers to know where microservices are at any given time. Engineers must be able to find and manage the microservices running your applications. A service discovery system allows services to be detected automatically and help them find each other. With a service discovery system in place, engineers don't have to know where every service is located.

8. Monitoring and Visualization

Applications powered by a microservices architecture are heterogeneous and distributed by nature, making them difficult to monitor, visualize, and analyze. Traditional application monitoring and performance management (APM) solutions are not designed to handle complex distributed applications.

However, today there are APM solutions designed specifically for monitoring, visualizing, and managing complex distributed applications. Engineers working on the development and deployment of microservices need modern APM solutions that will help them identify the root cause of service errors and failures, determine and reduce service dependency bottlenecks, and figure out why an application isn't working the way it's supposed to.

Choosing an effective tool to monitor, visualize, and analyze your microservices architecture is crucial. Continuous, real-time monitoring and analysis of your microservices is an essential component of microservices success.

Choose Your Application Architecture Carefully

These are just some of the things companies should consider before making the leap to microservices. If your company decides to make that leap, these key considerations could make the difference between ensuring microservices success or abject failure. Choose your application architecture carefully as it’s not easy to move from one type of architecture to another.