**Advanced System Analysis and Design   
Hot Topic Report**



**Impact of Cloud Services like Docker Containers and Microservices on Monolithic Applications like Netflix**

**By:**

**Lakshman Ganta**

**U54409861**

[**gantal@usf.edu**](mailto:gantal@usf.edu)

Contents

[**Introduction:** 3](#_Toc116302985)

[**Monolithic Application:** 4](#_Toc116302986)

[**Netflix:** 5](#_Toc116302987)

[**Amazon Web Services (AWS):** 7](#_Toc116302988)

[**Docker Containers:** 8](#_Toc116302989)

[**Before Docker Containers:** 9](#_Toc116302990)

[**Origin of Docker Containers:** 9](#_Toc116302991)

[**Advantages of Docker Containers:** 10](#_Toc116302992)

[**Problems with docker containers:** 11](#_Toc116302993)

[**Microservices:** 12](#_Toc116302994)

[**Advantages of Microservices:** 13](#_Toc116302995)

[**Disadvantages of Microservice:** 15](#_Toc116302996)

[**Advantages of combining Microservices with Docker Containers** 16](#_Toc116302997)

[**Comparison of monolithic architecture with microservices and docker containers architecture:** 16](#_Toc116302998)

[ **Difference When it comes to architecture**: 16](#_Toc116302999)

[ **Scalability comparison:** 16](#_Toc116303000)

[ **Time to market:** 17](#_Toc116303001)

[ **Reliability:** 17](#_Toc116303002)

[**The usage of microservices architecture and docker containers at Netflix:** 17](#_Toc116303003)

[**Decoding the system architecture of Netflix:** 17](#_Toc116303004)

[**Back-End Structure of Netflix:** 18](#_Toc116303005)

[**Recommendations for the Manager:** 20](#_Toc116303006)

[**Conclusion:** 20](#_Toc116303007)

[**References:** 21](#_Toc116303008)

# **Introduction:**

In this “Hot Topic Report”, we will look briefly how the implementation of docker containers and microservices can be a game changer for one of the largest video streaming platforms that is Netflix. A monolithic architecture is a traditional model of a software program that is built as a unified unit as a self-contained and independent from other applications and we use some other service to break the services in the monolithic application architecture into individual services and deployed separately from one another on separate hosts. Before the introduction of microservices, Netflix in its starting phase couldn’t keep up the infrastructure for rapid growing video streaming services and after the introduction of microservices, it was one of the first companies which moved to cloud and the problem is solved as they hosted their application in different thousands of microservices. They used Docker Containers which integrate them seamlessly into the existing developer tools and operational infrastructure. Here, we will look into Netflix, it’s history before using Docker Containers and Microservices and the change it got after they started to use them.

# **Monolithic Application:**

This is how we used to build applications in old days, and they were called monolithic applications and they are still called the same way. Let’s take an example of an application named amazon, which consist of various components namely Storefront user Interface, which is used by users for connection, Mobile Interface for allowing people to connect using phones, Order Service, Account Management, Payment Service, Shipping Service, and Database. All of the above-mentioned components will run on the same host and there would be a complex tangle of connections between all of these different components of the application. The data access layer, business logic, and user interface are combined on a single platform.

Chart, funnel chart

Description automatically generated

# **Netflix:**

Netflix is an American Subscription based streaming platform which was founded on 29th Aug 1997 by Reed Hastings and Marc Randolph as a DVD service Company. As of now, Netflix have 221.6 million world-wide subscribers. Today Netflix have a market cap of nearly $300 billion.

Netflix is a subscription-based Streaming Service which allows us to stream any movie or TV series whichever available in the application without any commercials in between at some price over a span of time. We can have multiple plans for getting subscribed to Netflix like monthly or yearly. We will get a bit of discount if we get subscribed yearly instead of monthly. We have different plans in their like basic, Standard and Premium, there will be cost difference in them as there will be difference in number of features available in there. If we look into basic plan, we will have limited access like good video quality and resolution of 480px, when it comes to standard, the video quality changes from good to better and the resolution as changes to 1080px, and the costly subscription will be premium with best video quality and 4k+HDR resolution.

Netflix started using Docker Containers and Microservices from 2009 and from then on there are many changes and positive outcomes for the company. When we look into the revenue, and it’s change.

Chart

Description automatically generated

From the above graph, we can see that there is a drastic increase in revenue from 2009 after they started using docker containers and microservices. When it comes to the rate of increase in subscribers, the below data helps us understand the impact of docker containers and microservices over the increase of subscribers.

Chart, histogram

Description automatically generated

# **Amazon Web Services (AWS):**

Amazon Web Services is the world’s most comprehensive and widely adopted cloud platform which is offering over 200 fully featured services like data centers worldwide. It has millions of customers who are in top 500 companies worldwide and it also has customers who are becoming as a super-giant who once started as a startup. Most of the companies prefer AWS for going global in minutes with less cost for using cloud services when compared to other competitors. It also provides agile methodology and used for innovations most easily and efficiently. That’s the reason most of the companies use Amazon Web Services. It has many cloud services like EC2, Simple Storage Service, AWS Lambda and others which are being popularly known and used for storing data and performing tasks related to daily work of an organization. AWS also has few services that are being used by Netflix for its business which brought it the fame and name which it has currently. The services are namely Docker Containers and Microservices.

Amazon started using Amazon Web Services in early 2000’s and started the first-generation services like simple storage services and elastic cloud computing from 2006. They help in storing data in different data centers. It started growing from 2010 worldwide and is currently the market leader when it comes to services provided by cloud at a reasonable price.

# **Docker Containers:**

Chart

Description automatically generated with medium confidenceDocker is a standard unit of software development system that packages up code and all its dependencies so that the application runs quickly and reliable as if there is some modification or update to be done for one part, the other parts won’t get disturbed from one computing environment to another one. A Docker container image is a lightweight, occupies less storage, standalone, executable package of software that includes everything needed for an application to run.

Docker images become containers at runtime. In the case of docker containers, the images become containers when they are run on docker engine. Available for both Linux and Windows. Below mentioned are few of the docker containers that run on docker engine:

1. **Standard:** Containers are created on a standard of industry. So that they can be carried anywhere and can be opened from any system similar to virtual machine.
2. **Lightweight:** The docker containers share the machines’ Operating System and therefore doesn’t require an Operating System per application which increases the efficiency, decreases the latency, and reduces the server and licensing costs.
3. **Secure:** Application which are built using the containers and docker engines are safer as they provide the strongest default isolation capabilities in the industry.

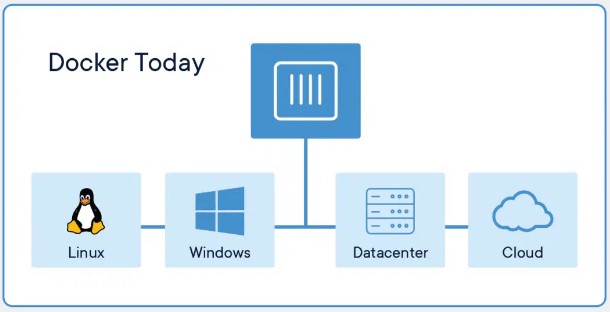
## **Before Docker Containers:**

For many years, the software applications are either deployed on bare metal i.e., installed on an operating system that has full control over the underlying hardware or on virtual machine i.e., installed on an operating system that shares underlying hardware with guest operating systems. Using bare metal type is very hard it doesn’t support portability and it is difficult to update. These two needs made information technology to move into or change to new business needs. Then the virtualization or hypervisors came into picture. This have the exact behavior that of the system that is the operating system, storage, I/O. The VMs thus can be cloned, migrated, spun up or down depending on the needs of the clients for meeting the business needs and to conserve resources.

Although VMs are great in portability and all, they have their set of problems. As they are large like normally in gigabytes, every one of them containing operating system, only many of the virtualized applications can be consolidated onto a single system. Finally, initializing an VM itself takes a lot of time depending on the amount of data and size of applications present in that VM.

## **Origin of Docker Containers:**

Docker Containers technology was first launched in 2013 as an open source docker engine. The uniqueness of docker containers depend on it’s focuses on the requirements of developers and system operators to separate the application dependencies from the infrastructure. The computing concepts got leveraged by the docker engine around containers and especially in the Linux World. The Success in Linux world drove a partnership with Microsoft that did bring the docker container concept and its functionalities into windows. The technology available with docker containers and its open-source project has been leveraged by most of the data centers vendors and cloud providers. The docker container is mainly used in Infrastructure as a Service.



Docker containers and Virtual Machines have similar resource isolation and allocation benefits, The Main difference between docker containers and virtual machines is that containers virtualize the operating system instead of hardware whereas containers are more portable and efficient.

## **Advantages of Docker Containers:**

Although Containers work similar to Virtual Machines, they are a bit more granular and specific in nature. They isolate single application and its dependencies. All the containerized apps share a single, common operating system either Linux or Windows.

1. **Docker enables more efficient use of system resources:** Instances of containerized apps use very less memory when we compare them to virtual machines. We can start them up and stop them down very quickly and easily. The applications can be installed and packed very densely on the system they are hosted right now. These all sums up to decrease the business costs and overall costs. The cost saving will also depend on apps used on the docker; this helps to save costs on software licensing as they need only fewer operating systems instances to run on the same workload.
2. **Docker enables faster software delivery cycles:** The developing team, the business team, and the enterprise software team all of them need to respond quickly to changing conditions. That is the business should meet scaling to meet the demand and easy updating to add new features as the business continues or evolves.

Docker containers make it easy for adding new versions of the software, with new business features and moving them quickly into production. They should be able to roll back the changes if they need to. This makes it easier to implement strategies like blue/green deployments.

1. **Docker enables application portability:** When we run an entire enterprise application matter behind the firewalls, we need to keep everything close to each other and they should be secure. They should be in the desired cloud but not in any other cloud like public cloud. The application should be available only for the desired people but not for all the public and there should be high elasticity in the application. The docker containers encapsulate everything needed for the application to run, this allows applications to get shifted from one environment to another with ease whether it might be developer’s laptop or clients. Any host which is preinstalled with docker runtime can run a docker container.
2. **Docker Shines for microservices architecture:** The benefits of docker containers like lightweight, scalability, portability, and self-contained makes docker containers easier to build software with future thinking capacity. So that they can solve tomorrow's problems with yesterday’s development methods.   
   one of the software patterns containers make easier is microservices, by decomposing traditional application like monolithic application into different microservices, we can modify each of them separately without affecting the other one at all.

## **Problems with docker containers:**

The main issue with docker containers is that it itself is not a *“Silver Bullet”,* It cannot solve any problem on its own. It is just an advice that applies to any software technology.

1. **Docker won’t fix our security issues:** Software in a container is more secure than a software in bare metals. Containers will add extra layer of protection to the application when we compare it with bare metal application.
2. **Dockers won’t turn applications magically into microservices:** If we move an application into container, the application won’t turn into microservices instead it only reduces the resources consumption and make it easier for deployment, but it won’t change the design of the application rom monolithic to microservice application or vice-versa. Instead, the application will be a bit more secure and useful.
3. **Docker isn’t a substitute for virtual machines:** Many think that both Virtual Machines and Docker Containers are similar, but the thing is in a big regulatory company, we cannot swap containers with virtual machines or vice-versa as they because of regulatory compliance because virtual machines provide more isolation than containers.

# **Microservices:**

Microservice architecture, simply known as microservices is an architecture that relies on a series of independently deployable services. These services have their own business logic and databases with a specific goal. The enterprise team then can update, test, deploy and scale within each service. The main advantage of microservices is that we can decouple monolithic application into microservices with respect to compliance of business, domain-specific concerns into separate, independent code bases. Microservices are not used to reduce the complexity, but they make any complex structure to get visualize easily and convert it into a more manageable by separating tasks into small groups or processes that can function independently of each other and contribute to the overall Microservices.

Diagram

Description automatically generated

## **Advantages of Microservices:**

Organizations do benefit from both monolithic applications as well as microservices depending on so many factors. The advantage of monolithic applications is fast development through different environments, this is because that we will have single source for the entire code. Easy to search functionalities and all. When it comes to microservices, they are used to solve number of software development with respect to growing software and companies. As the microservices have all the services separately, they can be developed, updated, scaled, and deployed without affecting the other services. The updates related to software can be done frequently, which in return improves the reliability, uptime, and performance. One of the excellent things about microservices is that it makes easier for teams to update code, and accelerate release cycles, with continuous integration and continuous delivery which makes teams to experiment and roll back changes if there is some issue. Below mentioned are few advantages of microservices over monolithic applications:

1. **Agility:** Microservices promote agility by working in small teams with frequent deployments.
2. **Flexible Scaling:** If the microservice reaches its load capacity, it will rapidly trigger new instance by deploying to the accompanying cluster for releasing the pressure. Now a days, microservices can support much larger instance size.
3. **Continuous Deployment:** Thanks to microservices for frequent and faster deployments which we can do more than two times a day.
4. **Highly maintainable and testable: Teams can experiment new feature and rollback changes if there is something that went wrong. This makes faster code execution and for staying top when we compare with the competitors.**
5. **Independently deployable**: As microservices are independent individual units, they allow for faster and easy independent deployments for every feature, and we can do it multiple times.
6. **Technology flexibility**: One of the major advantages of Microservice architectures is that it gives teams the independence to choose the tools and services they desire for developing and maintaining them.
7. **High reliability**: With the help of microservices, we can deploy, update, and remove changes for a particular service without the threat of bringing down the entire application.
8. **Happier teams**: Since most of teams using microservices for their software development and deployment, they are more autonomous and can build and deploy themselves without waiting weeks for a pull request to be approved.

Graphical user interface, chart, line chart

Description automatically generated

The above graph shows the usage of microservices from time to time and we can clearly see how rapidly it is being choose by many industries.

## **Disadvantages of Microservice:**

With pros comes cons, Microservices can add increased complexity that leads to development sprawl, or rapid or unmanageable growth. It can be challenging to define and understand how one service is related to other. There is another complexity of who owns the specific part of a software namely service.

1. **Development sprawl**: When it comes to complexity, Microservices are more complex than monolith architecture, as there are many services at different places created by multiple teams. If development sprawl isn’t properly managed and followed, that will result in slower development speed and poor operational performance.
2. **Exponential infrastructure costs**: Every new microservice will have its own cost allocation for its test suite, deployment, hosting, and costs for storing in cloud which might be private, public, multi, or hybrid and many more.
3. **Added organizational overhead**: Microservices add an extra level of communication among and in between different teams as they need collaboration and basic knowledge of service from which they are going to take care of.
4. **Debugging challenges**: Every microservice has their own set of logs, which makes debugging of the service a bit more complicated. In addition, too that, a single business process which can run across multiple virtual machines or containers, further adds more complexity and it there by complicates the debugging.
5. **Lack of standardization**: By using Microservices, there won’t be a single common platform for a particular format, logging standards, and monitoring.
6. **Lack of clear ownership**: With the increase in use of more and more microservices, there will be increase in number of teams, and so the number of people in each team. Over a course of time, it becomes hard to know the available services a team can leverage and who to contact for support regarding which service.

# **Advantages of combining Microservices with Docker Containers**

As we make the different services in the application independent of each other using microservices, if we add each one of them in different docker container, the application looks efficient with less complexity. It helps the development team the as they process the packing of application into docker containers. Some of the advantages for using microservices with Docker containers are mentioned below.

1. Very good community support from Cloud Service.
2. As we are working with both microservices and Docker containers, we can build a cloud native to applications very quickly and efficiently.
3. It's very easy and lightweight than virtual machines and therefore, we will have benefits of cost when it comes to resource and duration of project.
4. Provides uniform environment for both production and development.
5. It provides continuous integration and deployment.
6. We can integrate with popular tools and services like Amazon Web Services, Microsoft Azure, and Kubernetes.

# **Comparison of monolithic architecture with microservices and docker containers architecture:**

There is a lot of differences between monolithic architecture and microservices architecture when combined with docker containers. Few of the main differences are mentioned below:

* **Difference When it comes to architecture**: The monolithic architecture has a unified code which has a single build while the microservices architecture have the collection of small services.
* **Scalability comparison:** With microservices architecture, we will have precise scaling and better usage of resources while for monolithic architecture, the scalability is hard as we won’t be able to define what might the usage can be.
* **Time to market:** Microservices are easy to build and deploy while the monolithic application is complicated and time taking deployments.
* **Reliability:** microservices architecture is very reliable as if one of the services fails, the application won’t go down as a whole.

# **The usage of microservices architecture and docker containers at Netflix:**

In 2019, Netflix, A video-streaming application, consumed over15% of internet’s bandwidth, all across the world. For scalability, robust, managing resources, and for managing data infrastructure Netflix started using Amazon Web Services as they replaced monolithic architecture with docker containers and microservices architecture and hosted it on public cloud. With this decision, Netflix started using Amazon Web Services, a cloud service platform for hosting and developing application as per our needs. Netflix experienced a zero point of failure and an extremely scalable IT infrastructure for supporting millions of requests in a second or two.

## **Decoding the system architecture of Netflix:**

The Netflix architecture have two main components namely:

* Amazon Web Services which is a cloud service provider for hosting the data and storing it.
* Open connect: This is an in-house content delivery service, which responds to the requests.

When it comes to critical back-end services of Netflix:

* AWS EC2, which are auto scalable computing instances.
* AWS S3, Simple Storage Service for storing the data.
* AWS dynamo DB which are scalable databases.
* Video processing and transcoding tools, custom build by Netflix.
* Task oriented frameworks, business logic microservices.

## **Back-End Structure of Netflix:**

The back end consists of services, storage and databases and rest everything behind the actual streaming process is handled by playback applications.

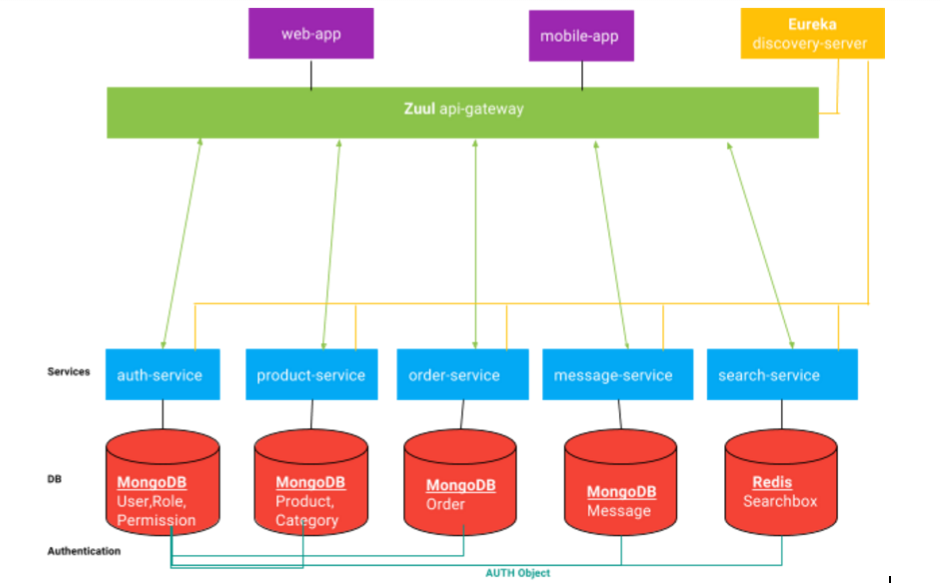
In Netflix, the microservices are used for tasks that include user management, billing, subscription management, video transcoding, personalized recommendations to each and every user and many more.

Below mentioned is the backend architecture of Netflix which is using microservices.

Diagram

Description automatically generated

* AWS Load balancer (ELB) for handling the client’s requests and sends them to backend which then will be running on Amazon Web Services.
* AWS ELB will send this request to API Gateway service, which is running on AWS EC2 instances. Zuul component has been developed by Netflix which handles these backend requests by AWS ELB, and it allows advanced features like dynamic routing, traffic monitoring, and security, the safety of data, which makes sure of minimum or zero point of failure.
* From Zuul, the request is forwarded to the Application API component, which is the business logic that gives power to Netflix. If we take an example of playback, the request is of Playback, and for this, Play API is engaged, under API Gateway Service. For remaining requests like user authentication or subscription checks, different corresponding APIs are deployed under the Application API component.



* Hystrix is an advanced program developed by Netflix, Hystrix isolates every microservice from every other and enables minimum or no failures at all, and enables unwanted resilience that ensures maximum success rate for fulfilling the requests from clients.
* Microservices can also be used to track the users’ activities, their history, personalized recommendations, and other data to Stream Processing Pipeline for real-time recommendations, and suggestions to the user, for enhancing engagement and user experience.
* The then processed data which comes from Stream Processing Pipeline is then provided to big data processing tools such as AWS S3, Hadoop HDFS, Cassandra for the next action.

# **Recommendations for the Manager:**

As there are many positives for organization if they start using microservices, few are the below mentioned recommendations based on which we can suggest out higher officials for moving monolithic applications to microservices.

* Our business gets empower and expand If we empower our customers by providing the best possible methods and modes of service.
* We can quickly build and adopt to developer friendly interface which in turn helps to get the best out of them.
* With the use of database clustering, we can have backup for database individually and if we loose one database related to one service that won’t affect the database of other services.
* With the use of microservices instead of monolithic architecture, each service has different function or purpose that helps in reduction of complexity and cost effective.
* Microservices focuses mainly on incremental improvements that helps us to deliver application or product in parts in less time than delivering the entire application after a very long time.

# **Conclusion:**

From the above discussion as a whole, we saw how services like docker containers and microservices can help the vast application to get better from both the business side and the client view. It helped them grow as a business and expand it.

*“When we said we were going to move all of Netflix to cloud everyone said we were completely crazy,” explained Cockcroft at a recent GOTO*[*conference*](https://www.youtube.com/watch?v=BeNrVl2_nyI)*. “They didn’t believe we were actually doing that, they thought we were just making stuff up.”*

Today, there are many big companies that moved to cloud for carrying operations, and adopting a modern microservices architecture like Amazon, Google, Microsoft. Each of them has their own cloud environment. And the rest are LinkedIn, IBM, Nike, Spotify, Twitter, etc.,

Netflix was one of the first companies to move to cloud and to implement microservices that help to make a way for other organizations to follow. Netflix has become leader when it comes to cloud services usage and microservices. That is the reason when everyone talks about cloud services, they also refer to Netflix.

# **References:**

1. Chandler Harris, “Microservices vs Monolithic Architecture”; from:

[https://www.atlassian.com/microservices/microservices architecture/microservices-vs-monolith](https://www.atlassian.com/microservices/microservices%20architecture/microservices-vs-monolith)

1. AWS Startup blogs, “Using Containers to build a Microservices Architecture”

Jan 20, 2015; from:

<https://aws.amazon.com/blogs/startups/how-to-use-containers-to-build-microservices-architecture/>

1. Projects on AWS, “Break a Monolithic application into microservices”; from:

<https://aws.amazon.com/getting-started/hands-on/break-monolith-app-microservices-ecs-docker-ec2/>

1. Aayushi Sahu, “Advantages of using Docker for microservices”; from:

<https://www.westagilelabs.com/blog/advantages-of-using-docker-for-microservices/>

1. Shivramvenugopal, “The Story of Netflix and microservices”; from:

<https://www.geeksforgeeks.org/the-story-of-netflix-and-microservices/>

1. Netflix Technology Blog, “The Evolution of containers usage in Netflix”; from:

<https://netflixtechblog.com/the-evolution-of-container-usage-at-netflix-3abfc096781b>

1. <https://help.netflix.com/en/node/412>
2. <https://en.wikipedia.org/wiki/Netflix>
3. <https://aws.amazon.com/what-is-aws/>
4. <https://en.wikipedia.org/wiki/Amazon_Web_Services>
5. <https://www.docker.com/resources/what-container/>
6. Serdar Yegulalp, “Why you should use docker and containers”,

Oct 10, 2018; from:

<https://www.infoworld.com/article/3310941/why-you-should-use-docker-and-containers.html>

1. Blogs from Merits, Microservices Architecture using Docker and Kubernetes ; from:

<https://www.meritdata-tech.com/resources/whitepaper/digital-engineering-solutions/microservices-architecture-docker-kubernetes/#:~:text=Using%20Docker%20Containers%20for%20Microservices,of%20packaging%20applications%20into%20containers>

1. Alfonso Valdes, Microservices vs Monolith: The Ultimate Comparison 2022,

March 10th, 2021; from:

<https://www.clickittech.com/devops/microservices-vs-monolith/>

1. Netflix Technology Blogs, The Evolution of Container usage at Netflix,

April 18, 2017; from:

<https://netflixtechblog.com/the-evolution-of-container-usage-at-netflix-3abfc096781b>

1. Ritesh Patel, Getting Started with Microservices using Netflix OSS & Docker.

August 13, 2014; from:

<https://nirmata.com/2014/08/13/getting-started-with-microservices-using-netflix-oss-docker/>

1. Zuul, <https://github.com/Netflix/zuul/wiki>
2. Ketan Varshneya, understanding the design of microservices architecture at Netflix, December 15, 2021

<https://www.techaheadcorp.com/blog/design-of-microservices-architecture-at-netflix/#:~:text=Due%20to%20their%20decision%20to,service%20requests%2C%20without%20any%20hiccup>.

1. TIBCO, Automation by Everyone,

<https://www.tibco.com/products/cloud-integration/automation?utm_medium=cpc&utm_source=google&utm_content=s&utm_campaign=ggl_S_NAM_TCI_NB_Events&utm_term=microservices%20architecture&_bt=602212014479&_bm=p&_bn=g&gclid=Cj0KCQjwhY-aBhCUARIsALNIC05weyD8rUEfg7XAzqlJ6griiozNWE1kLB_ivhRufWMwH9FQslPylBQaApcVEALw_wcB>

1. Mike Mackrory, 5 patterns for better microservices architecture,

November 17, 2016,

<https://www.sumologic.com/blog/5-patterns-better-microservices-architecture/>

1. Blog from Smartbear, why you can’t talk about microservices without mentioning Netflix.

<https://smartbear.com/blog/why-you-cant-talk-about-microservices-without-ment/>