stackoverflow

The Evolution of Containerization

Sponsored by

ORACLE

It's hard to miss the growing popularity of containerization and container-related technologies since Docker burst onto the scene in 2013. The technology was not new -- containers have been built into Unix for about a decade. While the primitives for containers were in Linux, containers weren't really available outside of Facebook or Google prior to Docker.

These days, other technologies are also to thank for the rise of containerization. Industry standards are set by the Open Container Initiative, which means fragmentation between container formats has been averted, and companies are free to produce and use container software without the limitations of closed platforms. This has resulted in a thriving container technology market.

- Kubernetes is Cloud Native Foundation's open-source answer to Docker's commercial Swarm. It dominates the market as a free, open source container management system.
- Google Kubernetes Engine is an orchestration service that runs on Container OS, an operating system designed explicitly for containers.
- Amazon EC2 Container Service is an orchestration system that supports Docker containers and AWS.
- Apache Mesos is an open source orchestration project developed by the University of Berkeley, California. Mesosphere is a datacenter-scale operating system built on top of Apache Mesos.
- Red Hat's OpenShift works with Docker containers and Kubernetes orchestration, and claims to be "the industry's most secure and comprehensive enterprise-grade container platform."
- Rancher is a management platform for Docker containers that runs off a Kubernetes distribution, with additional options of Docker Swarm and Apache Mesos.

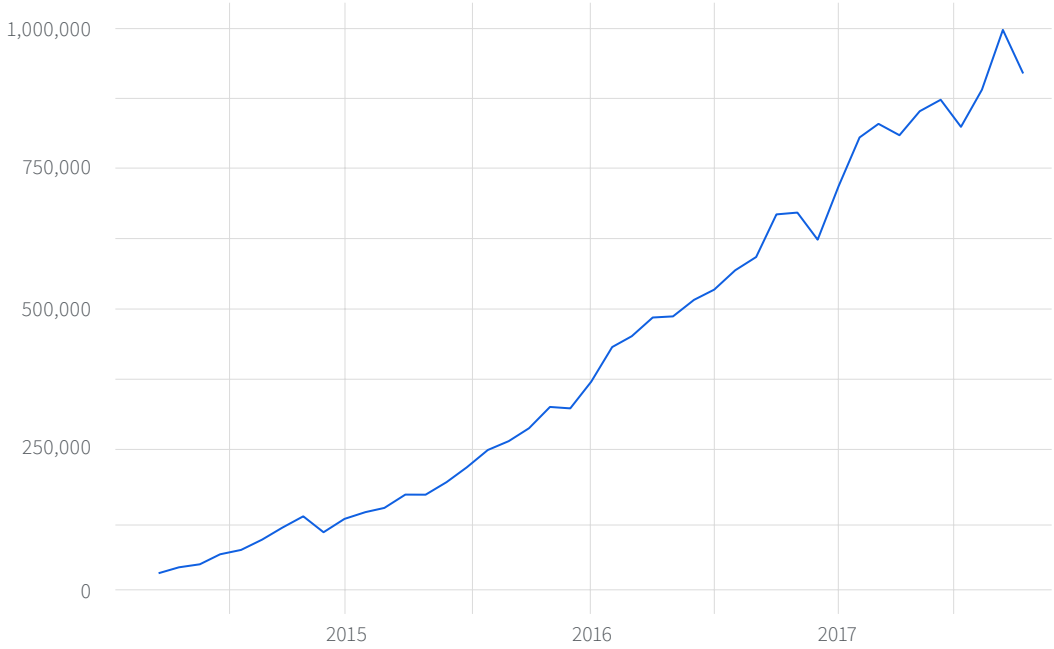
This report will look at the growth in popularity of container technology using proprietary data from Stack Overflow, as well as how the usage of containers has evolved and will continue to evolve.

Activity on Stack Overflow

For our analysis, we selected some of the most popular tags for both container engines and cluster management and orchestration. While there are other tags we consider relevant to this subject matter, they were not sizeable enough to include. Due to the extreme size of Docker compared to the other tags, we have separated this out in our graphs. Our data begins in 2014, the point at which Stack Overflow question data became robust enough for analysis.

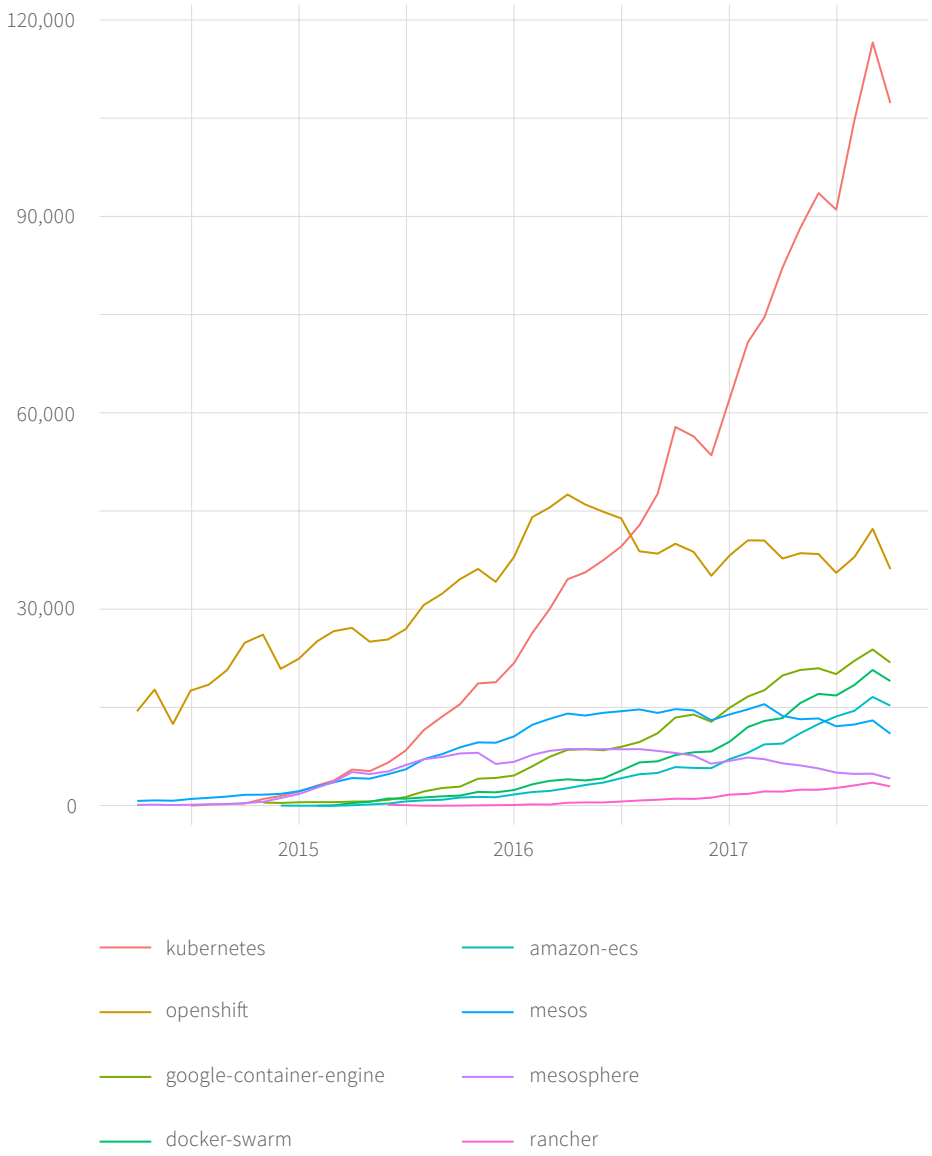
Globally

Growth of Docker globally over time
Population of developers who visited questions tagged with ‘docker’ since 2014



Aside from a few seasonality dips, growth to the Docker tag has been consistent since 2014, and looks like it will continue to grow. The growth of other tags, though at a slightly lower velocity, is also notable.

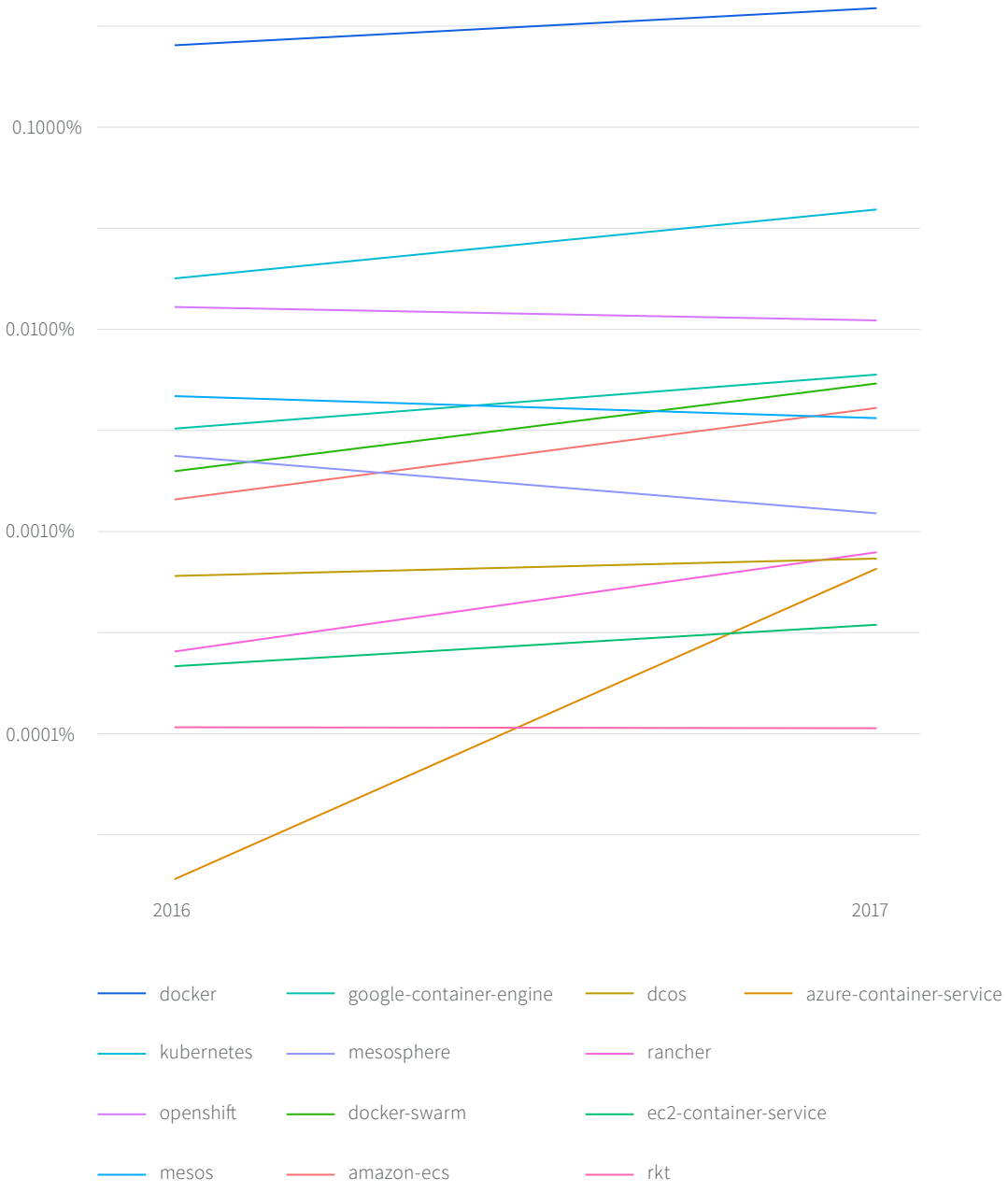
Growth of technologies globally over time
Population of developers who visited questions tagged with this



Kubernetes has also shown a rise in popularity since its creation, although this accelerated in the latter part of 2016. OpenShift has been more-or-less stable since the Spring of 2016. Google Container Engine and Docker Swarm have both overtaken Amazon ECS in 2017, while Rancher shows a decline.

The next graph looks at the comparative ranking of these tags according to data from September 2016 and 2017.

Growth of tags popularity from 2016 to 2017
Based on traffic measured in September of both years.
% of Stack Overflow question visits (log scale)

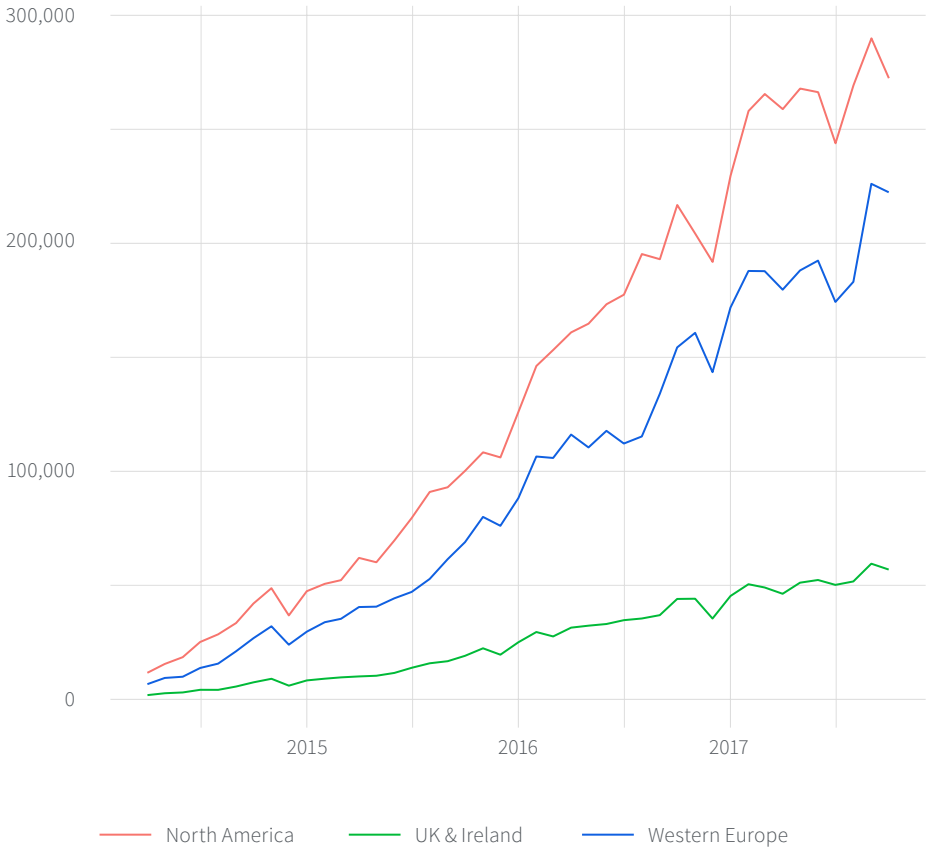


Docker was the most popular of these technologies in both years, and has shown growth from one year to the next. Whereas Kubernetes and OpenShift were on similar footing in September 2016, the gap had widened by 2017. Azure Container Service showed the largest growth (albeit from the smallest base), while Rkt and DC/OS stayed stagnant.

Regionally

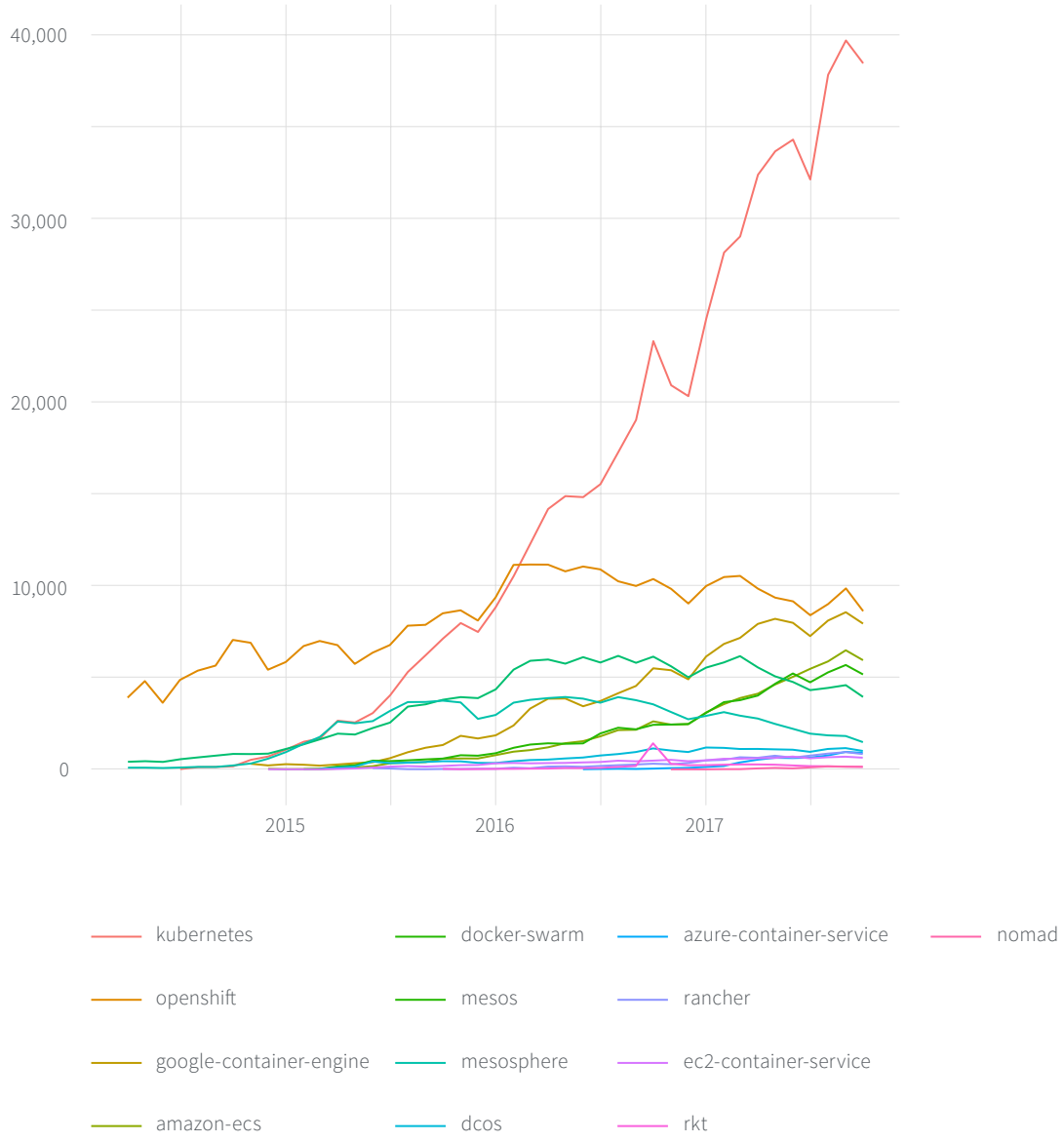
How does this activity vary by region? We identified five regions as North America (United States and Canada), Western Europe (Belgium, Denmark, Finland, France, Germany, Iceland, Italy, Netherlands, Norway, Portugal, Spain, Sweden and Switzerland), the United Kingdom and Ireland, Eastern Europe (Belarus, Bulgaria, Czech Republic, Hungary, Moldova, Poland, Romania, Slovakia, Ukraine and Russia) and APAC (Australia, Bangladesh, Bhutan, Brunei, Cambodia, China, Fiji, Hong Kong, India, Indonesia, Japan, South Korea, Malaysia, Maldives, Myanmar, Malaysia, Mongolia, New Zealand, Pakistan, Philippines, Singapore, Sri Lanka, Taiwan, Thailand and Vietnam). The traffic to the smaller tags in Eastern Europe and APAC were not substantial enough for analysis, so we looked at Docker and Kubernetes.

Growth of Docker over time
Population of developers who visited questions tagged with ‘docker’



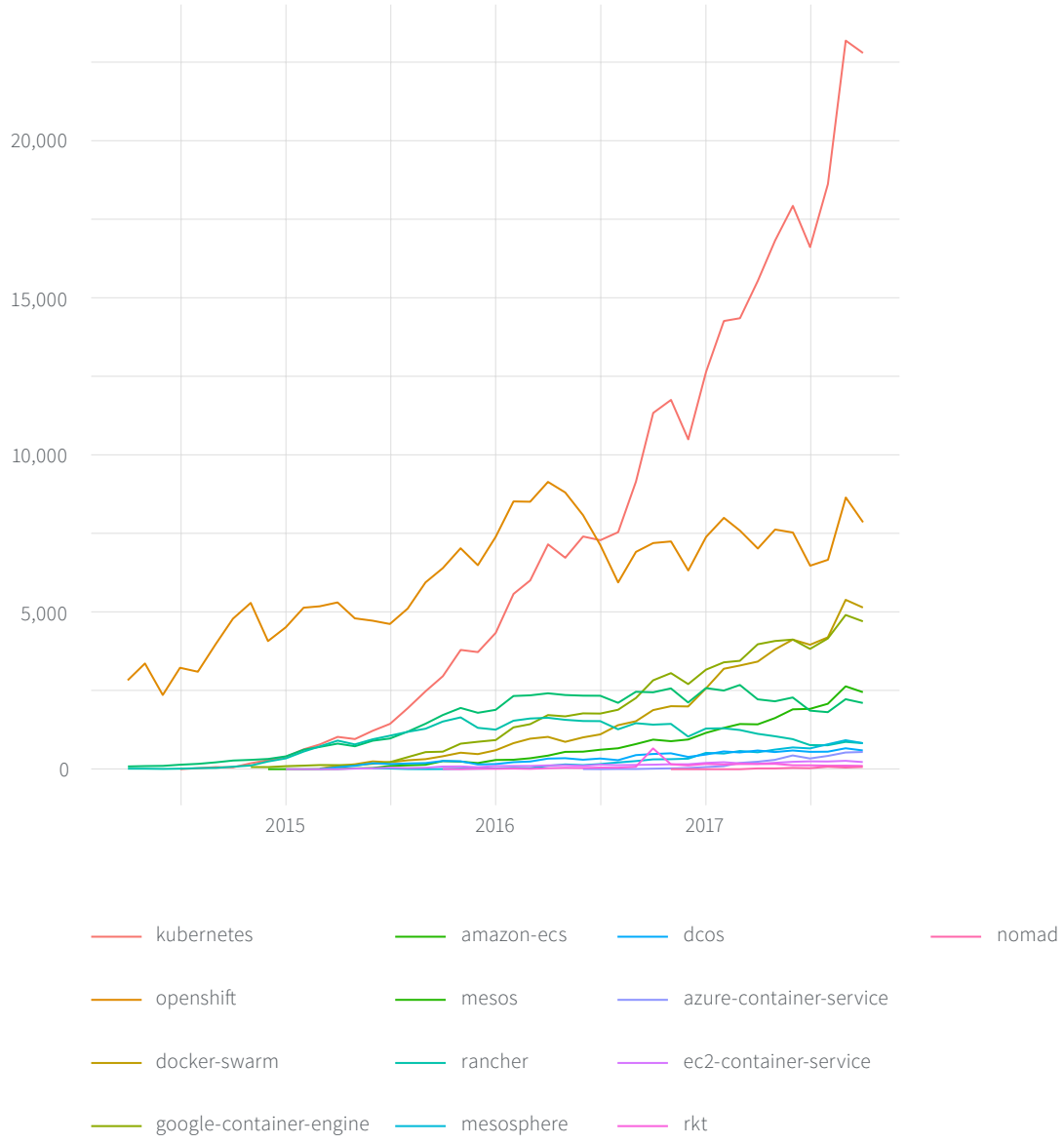
The growth has been the largest and fastest in North America, with Western Europe only slightly behind. The growth has been much slower in the UK and Ireland, where a recent growth spike in the other two regions halfway through 2017 was not reflected.

Growth of technologies in North America over time
Population of developers who visited questions tagged with this



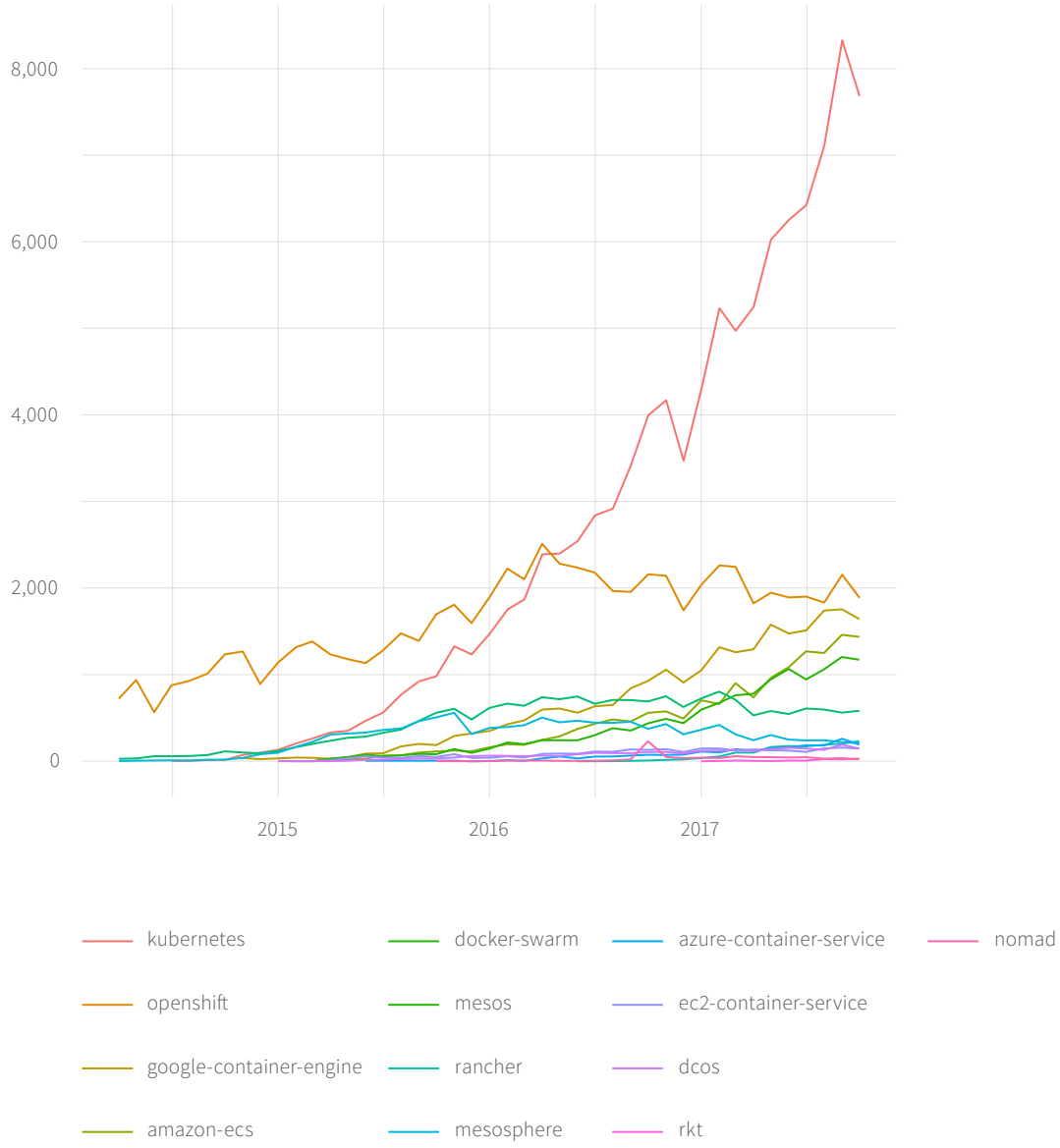
With the population of developers in North America accounting for so much of the global developer population, it's no surprise that the growth trends for these tags so closely resemble global trends. The decline of OpenShift has been much slower than global trends, while in Western Europe the decline was steeper.

Growth of technologies in Western Europe over time
Population of developers who visited questions tagged with this



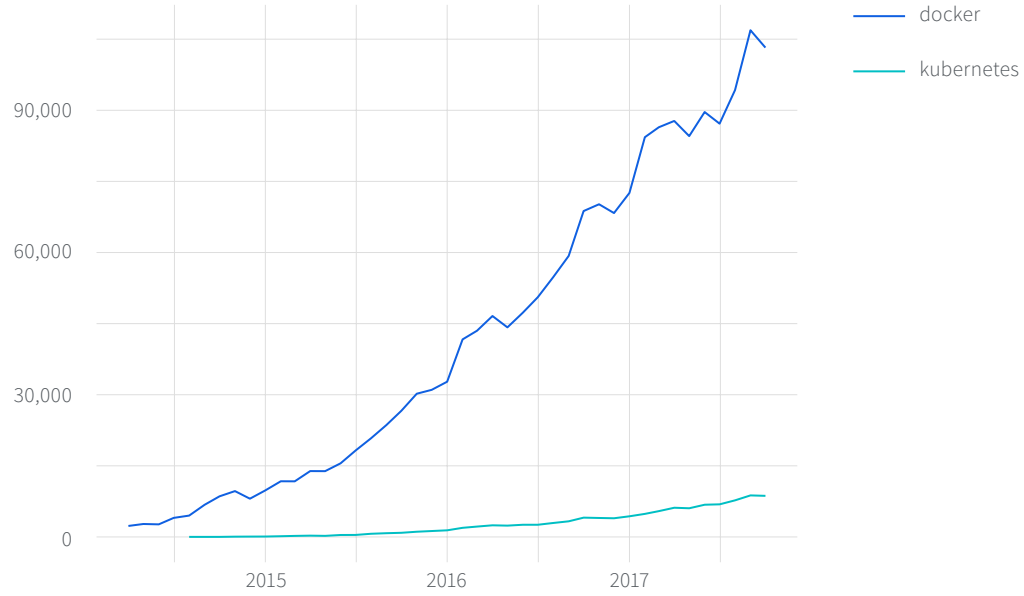
In Western Europe, Docker Swarm and Google Container Engine are on a much more even footing than they are globally, where Docker Swarm is dominant.

Growth of technologies in the UK & Ireland over time
Population of developers who visited questions tagged with this

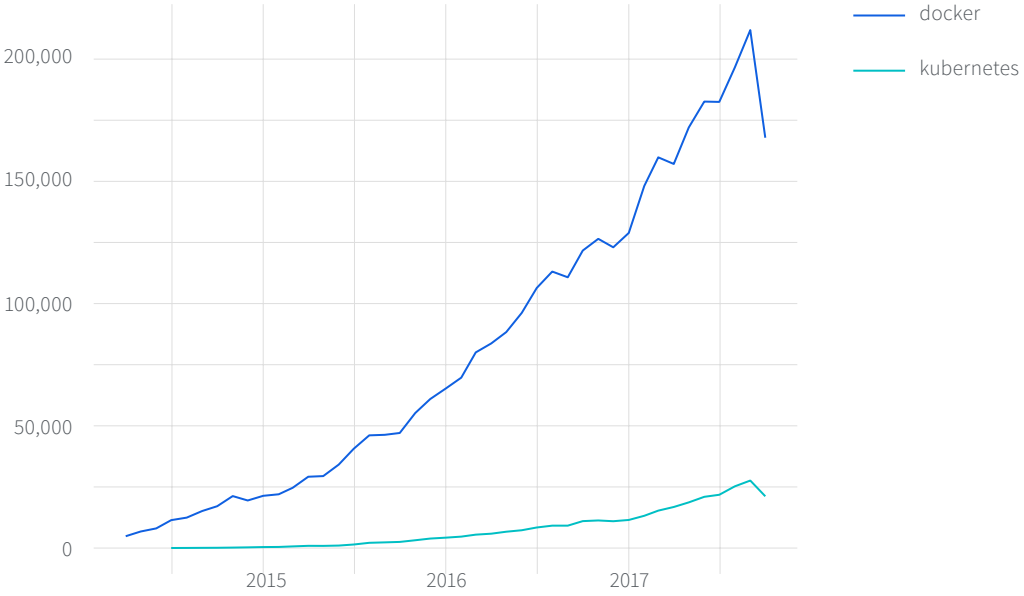


In the United Kingdom and Ireland, the audience sizes of OpenShift and Google Container Engine are a lot closer than elsewhere. Mesos shows a higher growth velocity than in the other regions.

Growth of technologies in Eastern Europe over time
Population of developers who visited questions tagged with this



Growth of technologies in APAC over time
Population of developers who visited questions tagged with this

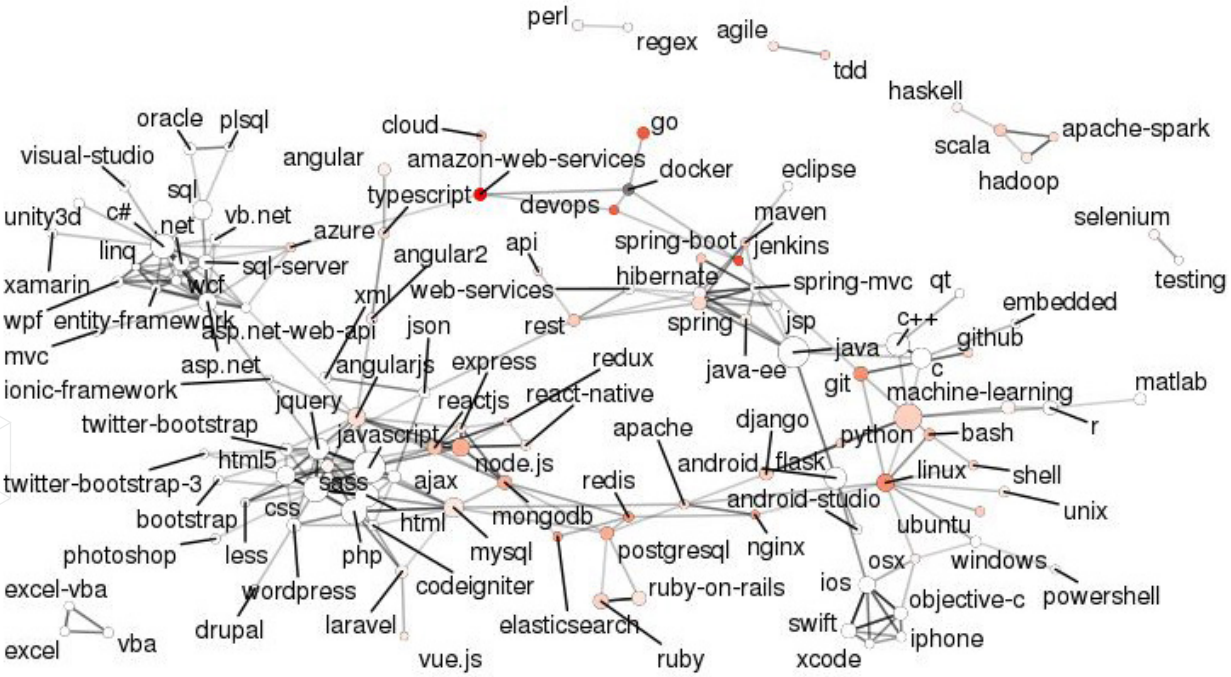


Growth in both Eastern Europe and APAC has followed a very similar trajectory, albeit with the audience size being much larger in APAC. While Docker has demonstrated a clear dominance in both markets, Kubernetes is also growing.

How Developers Use Containers

Associated tags and what this can tell us about use cases

Docker's position in the software development ecosystem
Tags correlated with Docker appear in red



Since Docker has by far the largest audience of all the tags studied, we used this as a barometer for how container-related tags correlate with other technologies. The DevOps and Jenkins tags show a strong correlation, which is unsurprising considering their association with continuous integration, something containers are often used to aid. Linux and Git also showed correlation, as well as Bash and AWS. This could potentially indicate that Docker hasn't the impact in Windows Server shops as much as elsewhere yet.

One developer, multiple projects

If a developer is working on multiple applications at one time, some of these may require conflicting dependencies. Containers allow developers to switch between development environments with relative ease, and allows the libraries and dependencies necessary for each application close to hand without having to install each of them onto the host OS. This prevents the developer from 'polluting' the host OS with dependencies that they do not want or need beyond one of many projects they are working on, and becomes particularly useful if one application requires Python 2.7 and the other 3.6. Part of the benefit here is that a developer can (more or less) also develop against the same software (i.e. dependencies) that would be in production, as well as emulate the network architecture of the production system.

Continuous integration/delivery

Since containers allow code to be isolated into a self-contained environment, they allow larger development teams working on a single application to segment the project down to smaller groups. Code modification is simpler and submission is streamlined. Once code is committed and provided it passes unit testing and test coverage, the build of the image can be triggered using a continuous integration server such as Jenkins, and added to the registry. Further automation and scripting can deploy a container from this image to staging for further testing. This process allows developers to build code, test it in any environment type and catch bugs early in a streamlined manner.

While containers being used correctly can help ease some of the complications of CI/CD, they are not magic bullets that can completely remove the need to consider configuration or construction details. In larger enterprises, multiple containers need to be interconnected across multiple host systems.

Software portability

The flexibility of the platform independence of containers is a major benefit. Since a container contains not only the application, but the configuration files and dependencies needed for it to run, it's relatively easy to get your software running reliably in any environment. For instance, you could run an application that runs on Linux on your company's Microsoft Windows servers. Using a container for this will take fewer resources than using a virtual machine (VM) - containers are leaner since they just run a process, over a VM that has to emulate hardware.

Application isolation

Since containers are isolated from each other, despite running on the same server with the same resources, they are an excellent tool to use when updating or applying patches. If one application crashes, others will not be affected and will continue running. This is particularly helpful should an application be breached by malware, or if you need to install applications from an “untrusted source”, since it will only gain access to the resources you allow it to. The container will have a fallback read-only layer to fall back on at any time.

Server consolidation

As we mentioned before, containers require fewer resources than VMs do, due in part to the fact that they are able to work off the host kernel, but also because the pipeline of communication is shortened due to containers talking to the kernel.

While VMs can take up to several minutes to boot up, containerized applications can be started almost instantly, as well as disappearing when they are no longer needed, freeing up resources on their hosts. Additionally, if the time to set up was several minutes, it would be per developer with VMs, and for all developers with containers.

Multi-tenancy

Containers can use tags or environment variables as parameters to deploy an application in multiple versions or configurations, allowing developers to work around environmental constraints. New versions of dependencies can be easily tested without impacting other developers’ workspace.

However, there are some use-cases in which containers help will be limited.

While you can use containers to run programs that are designed to run on a Linux kernel, a native Windows or OSX application cannot be shipped to other systems.

If you have a privileged user with full administrative capabilities that needs access to some applications, perhaps security-related, a container will be of no use. By its nature, a container is an isolated, limited environment, and would not allow said user the access required.

It’s also very important to identify what applications are long-lived vs ephemeral. Containers are perfect build environments since they can be recreated every time a build runs, providing a clean slate and no old artifacts. If you have an application that is stateful and needs to be long-lived, such as a domain controller, a VM or bare metal server will be a better option.

Conclusion

As can be seen by the trends, containerization and correlated orchestration tools are increasing in popularity. It's a new tool to solve many problems in both the development and operational spaces. It allows developers to easily move between projects and allows them to test modifications to environments without impacting others. It creates immutable images that can be deployed quickly - perfect for build and integration testing environments. Its platform-agnostic characteristics allow for consistent deployment across dissimilar architecture -- making working in non-homogenous environments and performing hardware upgrades trivial. Its isolation properties and communication techniques benefit the security of the environment.


stackoverflow