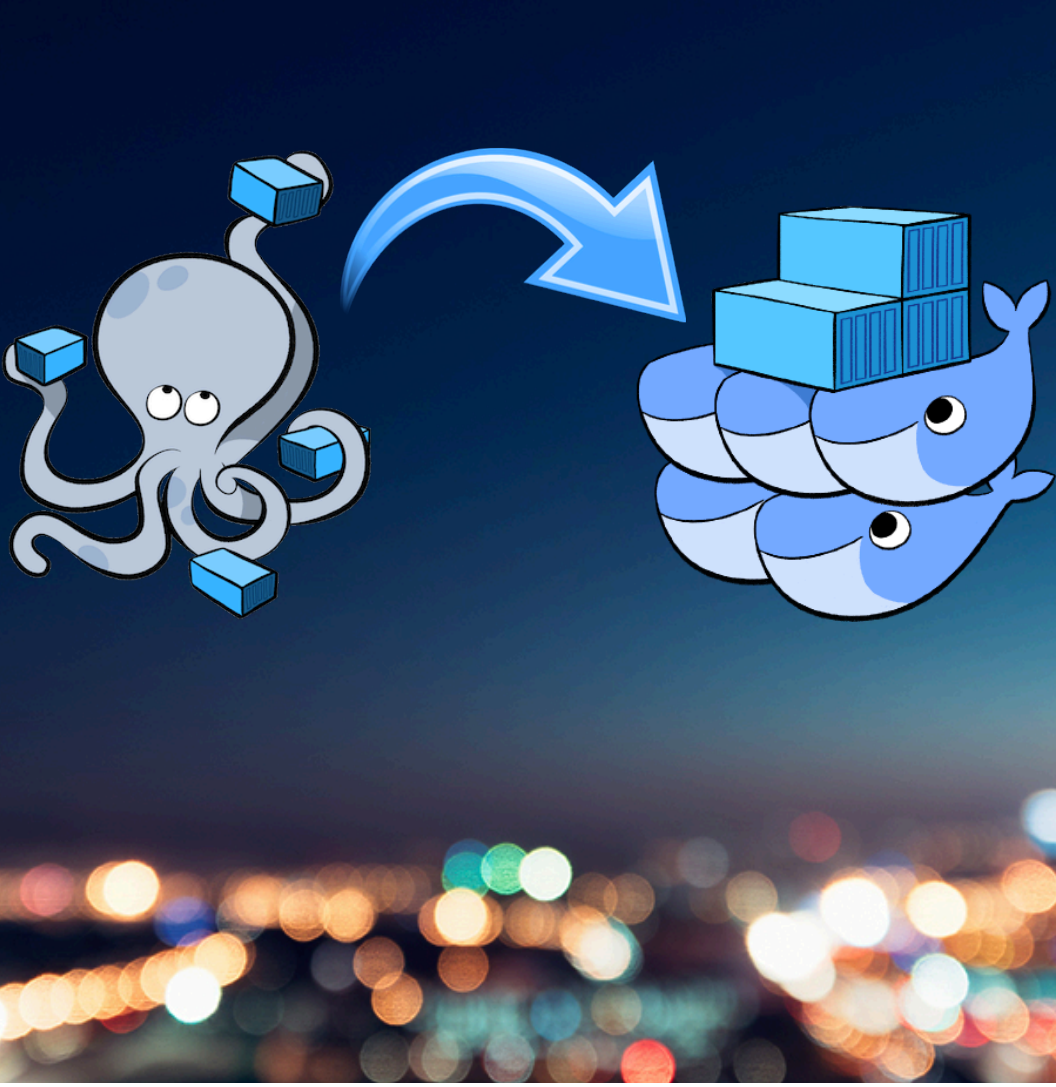


Deploying MICROSERVICES over DOCKER SWARM

Sunil

Sunil

CONTAINERIZING THE MICROSERVICE ARCHITECTURE

DOCKER & SWARM ORCHESTRATION

TABLE OF CONTENT

Page Left Intentionally

MICROSERVICE ARCHITECTURE

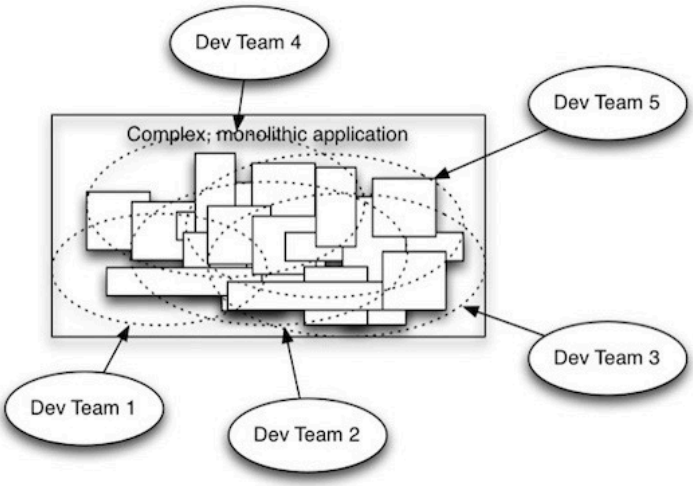
What is Microservices

Microservices are a software development technique —a variant of the service-oriented **architecture** (SOA) structural style— that arranges an application as a collection of loosely coupled services. In a **microservices architecture**, services are fine-grained and the protocols are lightweight.

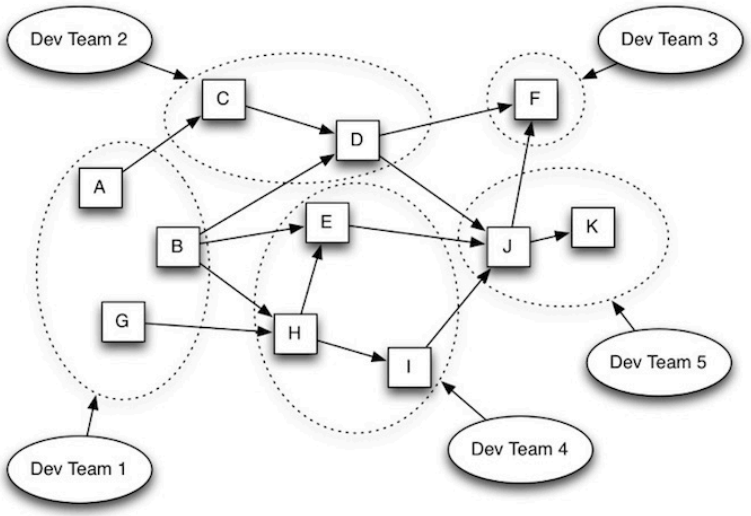
Microservices is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.



Monolithic Architecture



Microservices Architecture

Key Benefits of Microservices

The benefits that organizations realize from deploying microservices include:

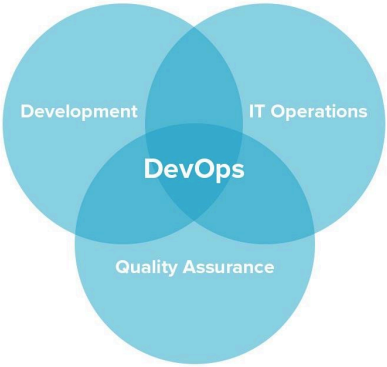
1. Improved scalability
2. Better fault isolation
3. Optimized scaling decisions
4. Localized complexity
5. Increased business agility
6. Increased developer productivity
7. Simplified debugging and maintenance
8. Future-proofed applications
9. Smaller and more agile development teams

Drawbacks of Microservices

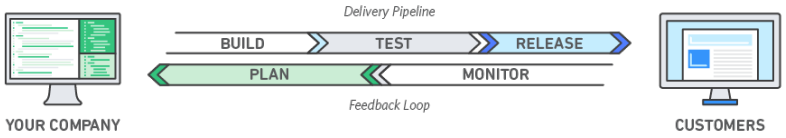
The Microservice architectures do carry their own costs.

1. Microservice architectures can be complex.
2. A microservices approach requires careful planning
3. Proper sizing of microservices is critical, and hard to calculate
4. You may have little control over third-party microservices
5. Downstream dependencies are difficult to track.

DEVOPS



DevOps is a set of software development practices that combines software development (Dev) and information technology operations (Ops) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.



How DevOps Works

Under a DevOps model, development and operations teams are no longer "siloed." Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test

to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team's velocity.

DevOps principles

Principles of DevOps are automation, continuous delivery, and fast reaction to feedback. More detailed explanation of DevOps pillars in the **CAMS** acronym:

Culture represented by human communication, technical processes, and tools

Automation of processes

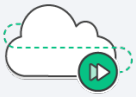
Measurement of KeyPrincipal Indicators

Sharing feedback, best practices, and knowledge

Adherence to these principles is achieved through a number of DevOps practices that include continuous delivery, frequent deployments, QA automation, validating ideas as early as possible, and in-team collaboration.

Benefits of DevOps

Speed and Scalability



Rapid Delivery

Reliability

Security

How microservices enable DevOps

The microservices approach also goes nicely with DevOps, which accelerates the process of getting software from development into production by breaking down barriers between software development and IT operations groups. In a DevOps environment, developers and operations engineers work together seamlessly in an iterative manner to produce higher-quality software faster.

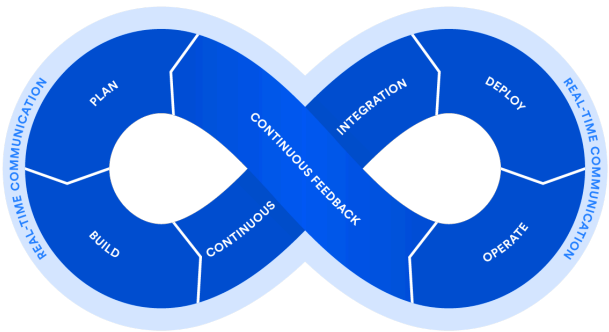
Microservices, when used Effectively

Used effectively, microservice architectures allow us to scale our application as the number of developers working on our application increases. The key is to build applications without creating a complex, unwieldy beast at the macro level. That means keep tracking each time a new service is added to your system or a new connection between microservices is made.

It also means examining the complexity increase and making sure it is warranted and well understood. Regularly examining the entire application system is critical to keep an interconnected set of microservices working effectively and reliably.

Although not a panacea, the benefits of microservices are clearly worth it for increasing numbers of modern software organizations. By changing how software development teams are structured, organizations can create teams centered on specific business services and give them both the responsibility and the authority to act as they see best. This approach lets teams quickly move with the business as it evolves in response to market demand without disrupting central business activities.

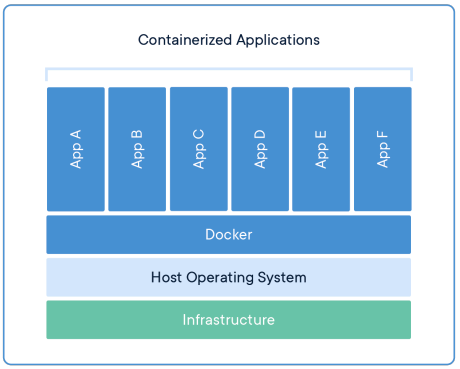
DevOps model and practices

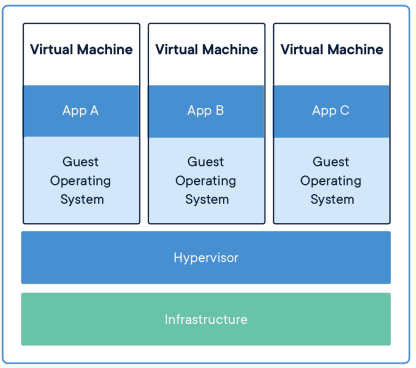


CONTAINERIZATION

Containerization involves bundling an application together with all of its related configuration files, libraries and dependencies required for it to run in an efficient and bug-free way across different computing environments.

The most popular containerization ecosystems are Docker and Kubernetes.





Containers versus Virtual Machines

Containers are an abstraction of the application layer (meaning that each container simulates a different software application). Though each container runs isolated processes, multiple containers share a common Operating System.

VMs are an abstraction of the hardware layer (meaning that each VM simulates a physical machine that can run software). VM technology can use one physical server to run the equivalent of many servers (each of which is called a VM). So, while multiple VMs run on one physical machine, each VM has its own copy of an Operating System, applications and their related files, libraries and dependencies.

Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.

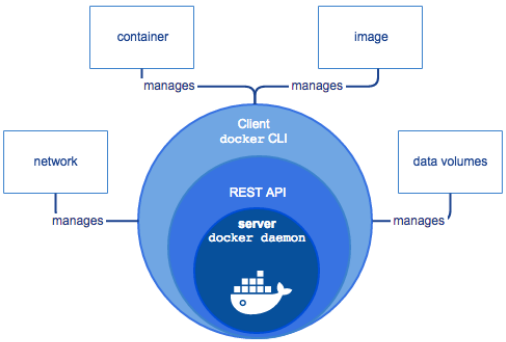
DOCKER

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

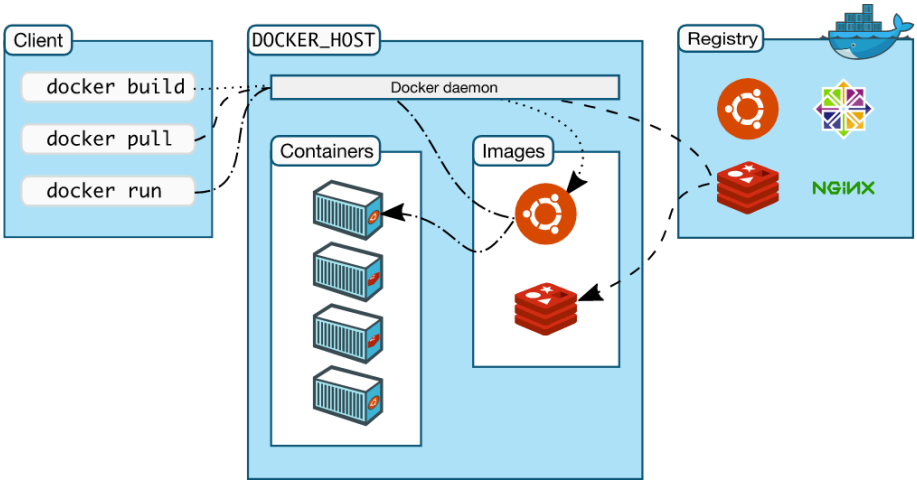
Docker Engine

Docker Engine is a client-server application with these major components:

- A server which is a type of long-running program called a daemon process (the `dockers` command).
- A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
- A command line interface (CLI) client (the `docker` command)

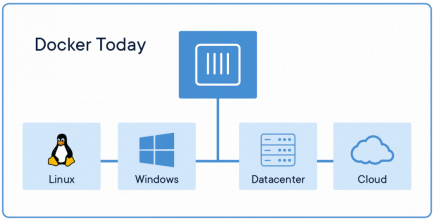


Docker Architecture



Who is Docker for ?

Docker is a tool that is designed to benefit both developers and system administrators, making it a part of many DevOps (developers + operations) toolchains. For developers, it means that they can focus on writing code without worrying about the system that it will ultimately be running on. It also allows them to get a head start by using one of thousands of programs already designed to run in a Docker container as a part of their application. For operations staff, Docker gives flexibility and potentially reduces the number of systems needed because of its small footprint and lower overhead.



Docker Features

Although Docker provides lots of features, we are listing some major features which are given below.

- Easy and Faster Configuration
- Increase productivity
- Application Isolation
- Swarm
- Routing Mesh
- Services
- Security Management

Swarm

It is a clustering and scheduling tool for Docker containers. Swarm uses the Docker API as its front end, which helps us to use various tools to control it. It also helps us to control a cluster of Docker hosts as a single virtual host. It's a self-organising group of engines that is used to enable pluggable backends.

Routing Mesh

It routes the incoming requests for published ports on available nodes to an active container. This feature enables the connection even if there is no task is running on the node. It enable port access across cluster... With Stateless Load Balancing...

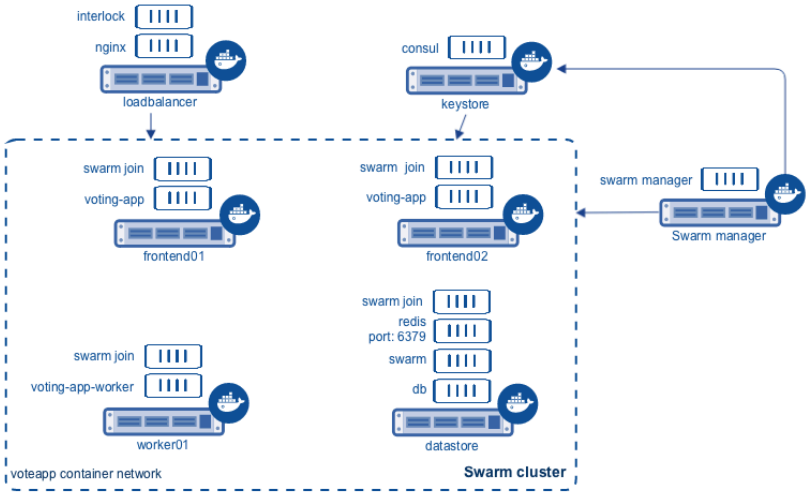
Services

Services is a list of tasks that lets us specify the state of the container inside a cluster. Each task represents one instance of a container that should be running and Swarm schedules them across nodes.

Docker Alternatives

Although Docker provides lots of features, we are listing

1. Virtual Machine
2. Vagrant
3. Rancher
4. Wow
5. LXC Linux Container
6. Apache Mesos



"Services Deployment using Docker Swarm Deployment"

ORCHESTRATION

Orchestration is the automated configuration, management, and coordination of computer systems, applications, and services. Orchestration helps IT to more easily manage complex tasks and workflows

Automation and orchestration are different, but related concepts. Automation helps make your business more efficient by reducing or replacing human interaction with IT systems and instead using software to perform tasks in order to reduce cost, complexity, and errors.

IT orchestration also helps you to streamline and optimize frequently occurring processes and workflows, which can support a DevOps approach and help your team deploy applications more quickly.

Container Orchestration

Container orchestration automates the deployment, management, scaling, and networking of containers. Enterprises that need to deploy and manage hundreds or thousands of Linux® containers and hosts can benefit from container orchestration.

Containers give your microservice-based apps an ideal application deployment unit and self-contained execution environment. They make it possible to run multiple parts of an app independently in microservices, on the same hardware, with much greater control over individual pieces and life cycles.

Managing the lifecycle of containers with orchestration also supports DevOps teams who integrate it into CI/CD workflows. Along with application programming interfaces (APIs) and DevOps teams, containerized microservices are the foundation for cloud-native applications.

What is container orchestration used for?

Use container orchestration to automate and manage tasks such as:

- Provisioning and deployment
- Configuration and scheduling
- Resource allocation
- Container availability
- Scaling or removing containers based on balancing workloads across your infrastructure
- Load balancing and traffic routing
- Monitoring container health
- Configuring applications based on the container in which they will run
- Keeping interactions between containers secure

Container Orchestration Tools

Docker Swarm — Docker Swarm provides native clustering functionality for Docker containers, which lets you turn a group of Docker engines into a single, virtual Docker engine.

Kubernetes — Kubernetes is an orchestration system for Docker containers. It handles scheduling and manages workloads based on user-defined parameters.

Amazon ECS — The Amazon EC2 Container Service (ECS) supports Docker containers and lets you run applications on a managed cluster of Amazon EC2 instances.

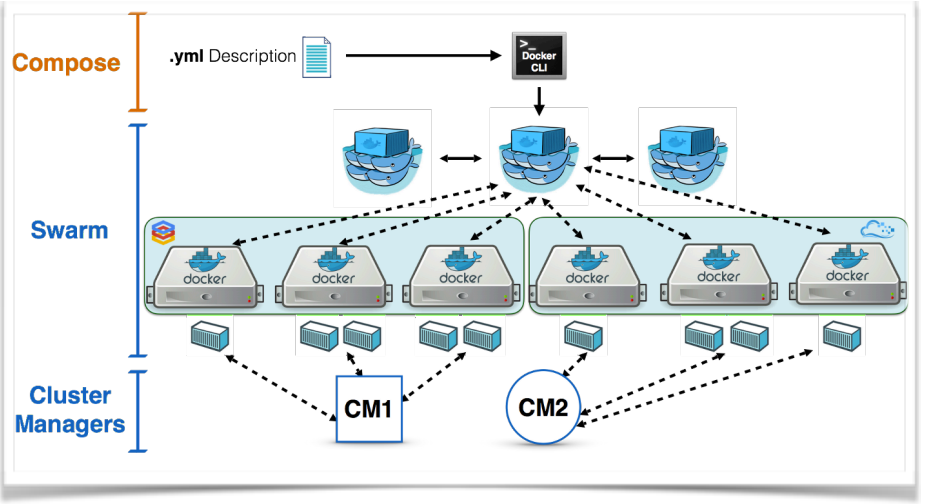
Azure Container Service (ACS) — ACS lets you create a cluster of virtual machines that act as container hosts along with master machines that are used to manage your application containers.

DOCKER SWARM MODE

Introduction

A Docker Swarm is a group of either physical or virtual machines that are running the Docker application and that have been configured to join together in a cluster. Once a group of machines have been clustered together, you can still run the Docker commands that you're used to, but they will now be carried out by the machines in your cluster. The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.

Docker swarm is a container orchestration tool, meaning that it allows the user to manage multiple containers deployed across multiple host machines.



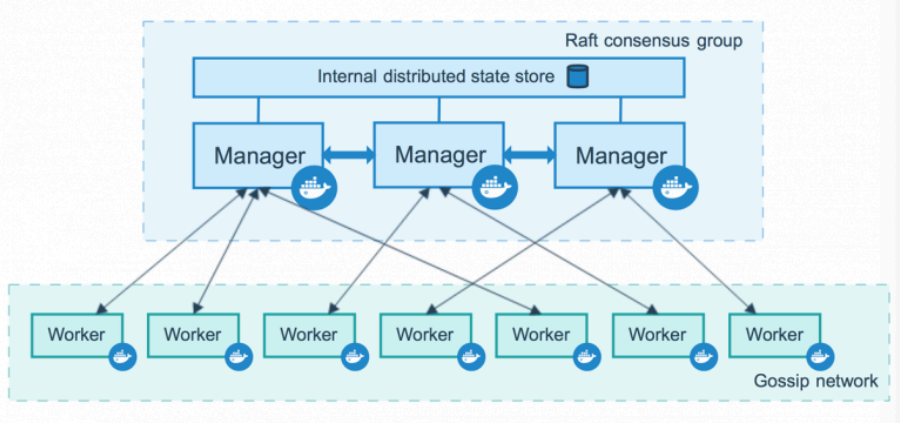
MICROSERVICES WITH DOCKER

Challenges of Building a Microservice Architecture

- Service tracking
- Rapid resource scaling
- Inefficient minimal resourcing
- Increased deployment complexity

Docker to the Rescue for Microservices

- Task isolation
- Support multiple coding languages
- Database separation



FUTURE SCOPE WITH **DOCKER DEPLOYMENT**

An analysis of current trends

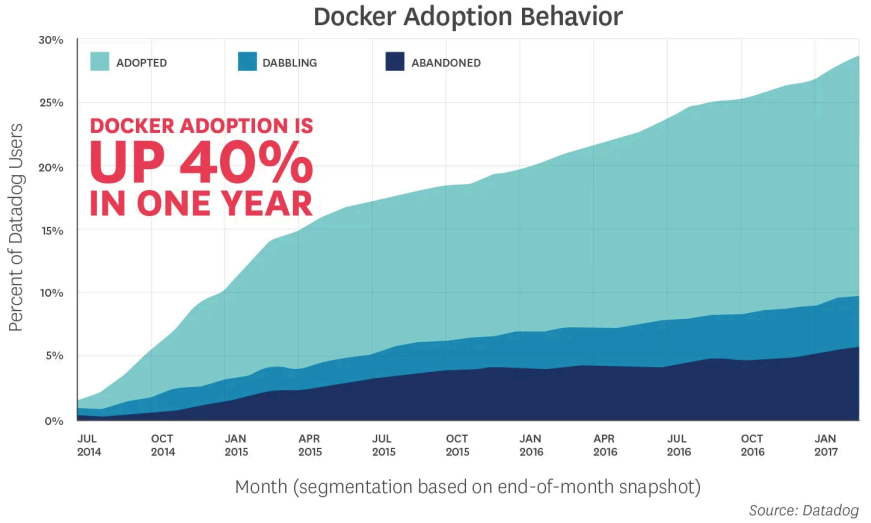
The DockerCon 2016 keynote speech by Docker's CEO Ben Golub provided many insights into the current level adoption of this technology:

- Dockerised applications have grown about 3100 per cent over the past two years to about 460,000 in number.

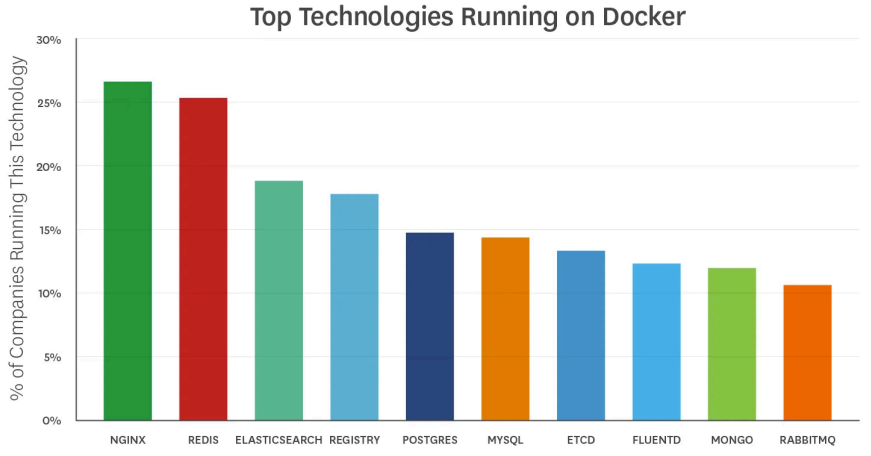
- Over 4 billion containers have been shared among the developer community. Analysts are touting 8 billion as the number for 2017.

- There are nearly 125,000 Docker Meet up members worldwide, which is nearly half of the population of Iceland.

Docker has spurred the development of a number of projects including CoreOS and Red Hat's Project Atomic, which are designed to be minimalist environments for running containers.



Source: Datadog



Source: Datadog

ABOUT ME

Sunil Yadav

Software Engineer at SDIL (Wesee N/W)

Contact No.	+91 9670809602
LinkedIn Profile -	https://www.linkedin.com/in/sunil016/
GitHub Link -	https://github.com/Ysunil016
HackerRank Profile	https://www.hackerrank.com/atworksunil
FaceBook Profile	https://www.hackerrank.com/atworksunil