Brown Field – Existing Code

Green field – scratch development

Service – a piece of software which provide services to other parts of the system

SOA – a single service will serve different clients

Ex- Web API can communicate with web app ,mobile app

Microservice architecture – Scalable,flexible,high performance applications

1. Single purpose.
2. Light and quick communication mechanism
3. Communication protocol between services should not be technology specific. like http rest
4. Independently deployable
5. Independent data storage
6. Independent changeable
7. Centralized tooling for service management
8. Less coupling leads to easy changes
9. Distribute transaction which means single transaction is processed by multiple microservices
10. Automated test tools
11. Release and deployment is complex
12. Asynchronous communication
13. Fast issue resolution

Design Principles

1.High Cohesion – SRP of SOLID

2.Autonomous – Loose coupling ,change in one microservice should not affect other microservice

3.Business Domain Centric – Should provide a particular business function

4.Resilience – Should embrace failure by degrading functionality or providing default functionality

5.Observable – Maintaining system health in terms of logs,Centralized monitoring and logging

6.Automation – Automation Testing,Regression,Integration Testing. CI/CD

Docker provides an additional layer of abstraction and automation of [operating-system-level virtualization](https://en.wikipedia.org/wiki/Operating-system-level_virtualization) on [Windows](https://en.wikipedia.org/wiki/Windows) and [Linux](https://en.wikipedia.org/wiki/Linux).[[7]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-SYS-CON_Media-8) Docker uses the resource isolation features of the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) such as [cgroups](https://en.wikipedia.org/wiki/Cgroups" \o "Cgroups) and kernel [namespaces](https://en.wikipedia.org/wiki/Linux_namespaces), and a [union-capable file system](https://en.wikipedia.org/wiki/Union_mount) such as [OverlayFS](https://en.wikipedia.org/wiki/OverlayFS" \o "OverlayFS) and others[[8]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-9)to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining [virtual machines](https://en.wikipedia.org/wiki/Virtual_machine) (VMs).[[9]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-10)

Docker is a tool that can package an application and its dependencies in a virtual container that can run on any Linux server. This helps enable flexibility and portability on where the application can run, whether [on premises](https://en.wikipedia.org/wiki/On-premises_software), [public cloud](https://en.wikipedia.org/wiki/Public_cloud), [private cloud](https://en.wikipedia.org/wiki/Private_cloud), [bare metal](https://en.wikipedia.org/wiki/Bare-metal_server), etc.[[14]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-Linux-15)

**Docker Compose**[[edit](https://en.wikipedia.org/w/index.php?title=Docker_(software)&action=edit&section=6)]

Compose is a tool for defining and running multi-container Docker applications.[[60]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-61) It uses [YAML](https://en.wikipedia.org/wiki/YAML) files to configure the application's services and performs the creation and start-up process of all the containers with a single command.

### Docker Swarm[[edit](https://en.wikipedia.org/w/index.php?title=Docker_(software)&action=edit&section=7)]

Docker Swarm provides native [clustering](https://en.wikipedia.org/wiki/Computer_cluster) functionality for Docker containers, which turns a group of Docker engines into a single, virtual Docker engine.[[61]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-62)

In Docker 1.12 and higher, Swarm mode is integrated with Docker Engine.[[62]](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-63)

[Docker](https://github.com/docker/docker) is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

**docker run busybox**

First, the command is docker, and telling docker to **`run`** the 'busybox' image. If the image 'busybox' is not present, then docker will attempt to fetch an image named 'busybox' from the public [Docker hub](https://hub.docker.com/). Then Docker sets up the layers of this image, all the cgroups and namespaces for this container environment,

**docker pull fedora**

This is a direct way to fetch an image named 'fedora' from the public Docker hub.

**$> docker images**

This shows the images available locally. There are many more docker commands like these.

For a programmatic way to commit new images, there is the "Dockerfile" format. A Dockerfile is the file with the steps to prepare an image

**docker build -t mongodb .**

All the output of the steps are visible and once it finishes successfully a 'mongodb' image is available.

Running service container Using this new 'mongodb' image built with the Dockerfile, fire it up and review it.

**docker run -it -p 127.0.0.1:27017:27017 -v $(pwd)/db:/data/db mongodb**

there is also the [Docker registry](https://github.com/dotcloud/docker-registry), which allows hosting of local Docker images.

 Docker is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called containers) to run on the host operating system i.e. Linux. The key benefit of Docker is that it allows users to **package an application with all of its dependencies into a standardized unit** for software development

Docker Terminologies

* *Images* - The blueprints of our application which form the basis of containers. In the demo above, we used the docker pull command to download the **busybox** image.
* *Containers* - Created from Docker images and run the actual application. We create a container using docker run which we did using the busybox image that we downloaded. A list of running containers can be seen using the docker ps command.
* *Docker Daemon* - The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system to which clients talk to.
* *Docker Client* - The command line tool that allows the user to interact with the daemon. More generally, there can be other forms of clients too - such as [Kitematic](https://kitematic.com/) which provide a GUI to the users.
* *Docker Hub* - A [registry](https://hub.docker.com/explore/) of Docker images. You can think of the registry as a directory of all available Docker images. If required, one can host their own Docker registries and can use them for pulling images.
* **Base images** are images that have no parent image, usually images with an OS like ubuntu, busybox or debian.
* **Child images** are images that build on base images and add additional functionality.

Kubernets architecture

Docker compose

Security

Db connections

Appsettings

ENV EmailServer=127.0.01

ENV ConnectionString= Data Source=127.0.01,1500;Initial Catalog=Test.Database;