Microservices Architecture

Connect Architecture

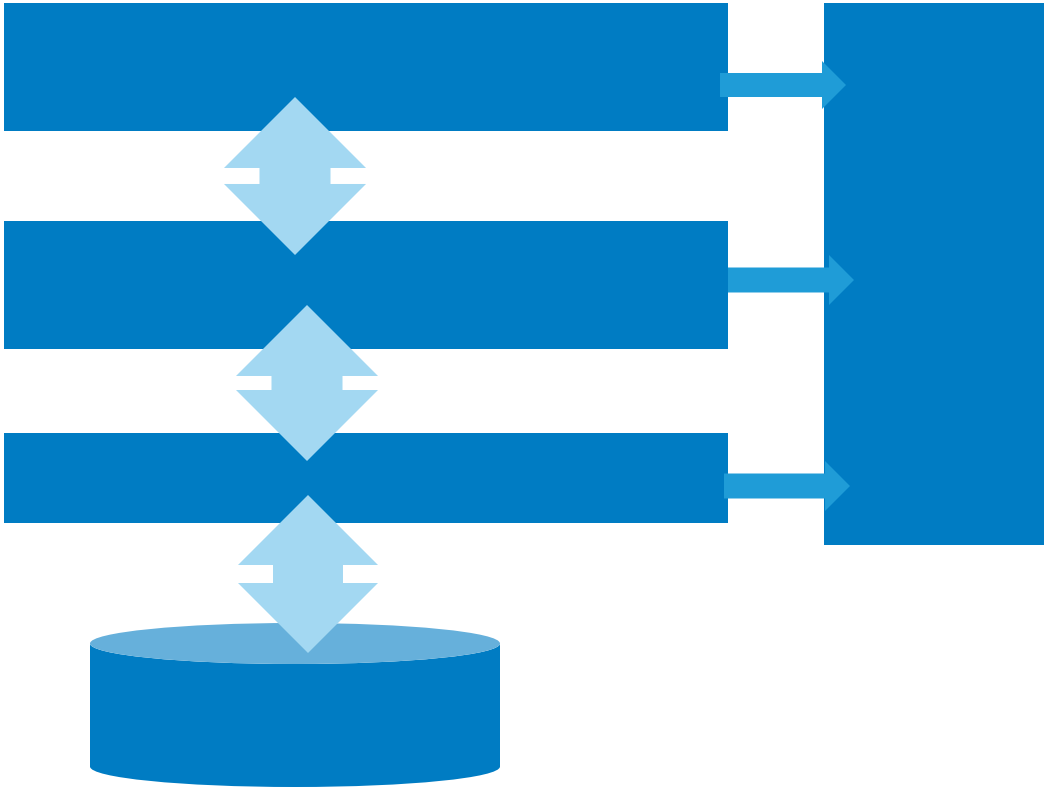
Infosys®

Structure

- Monolithic Architecture - Limitations
- What is Microservices Architecture?
- Microservices Architectural View
- Key Characteristics
- Service Modeling considerations
- How to scale?
- Elements needed for the holistic success
- Industry Case studies
- Usage Scenarios
- Benefits and Limitations
- Docker as a Microservices pattern enabler

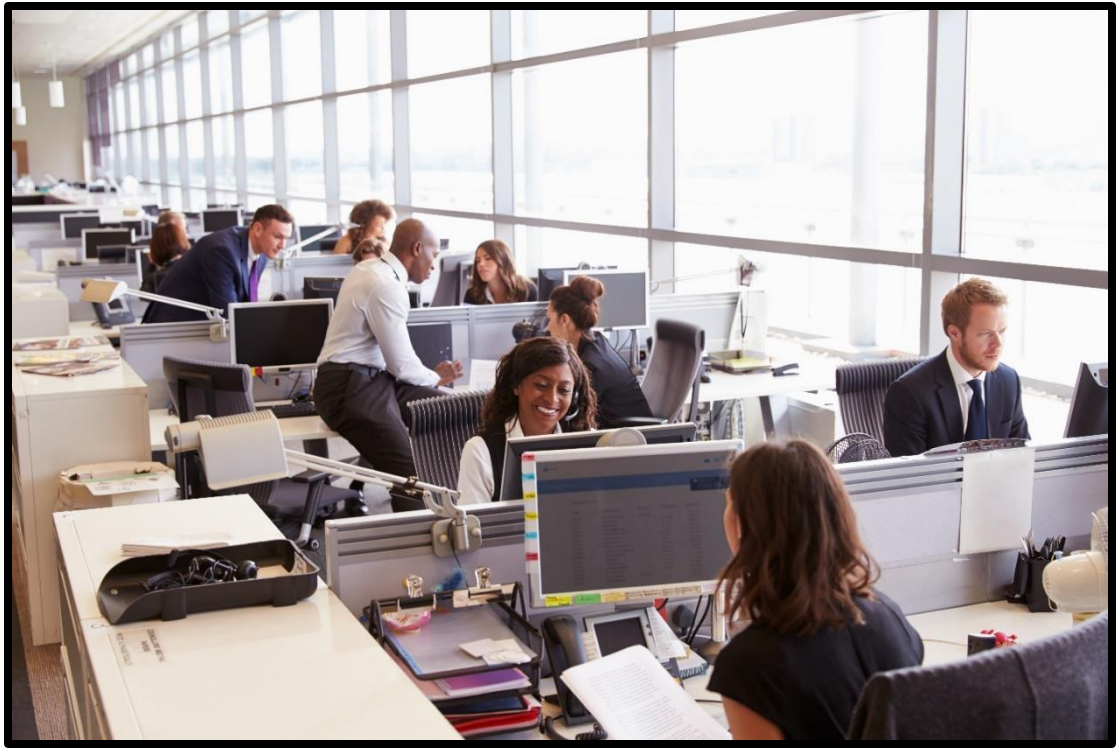
Monolithic Architecture

Monolithic Architecture

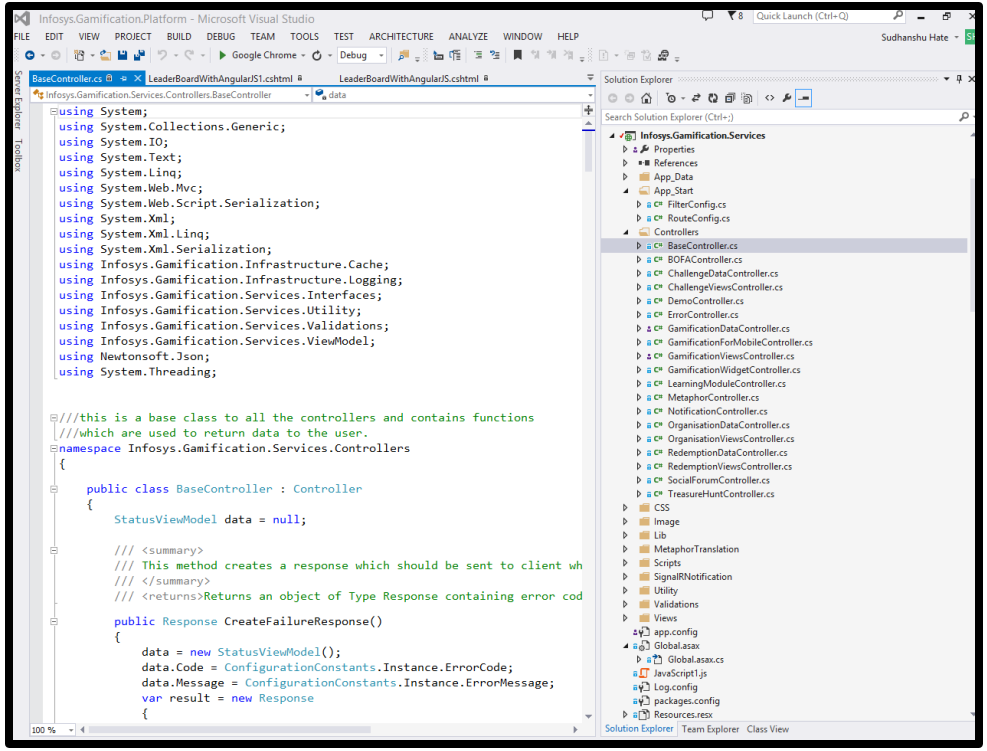


Defined tech. stack - married to choices

Lack of Agility – onboarding teams

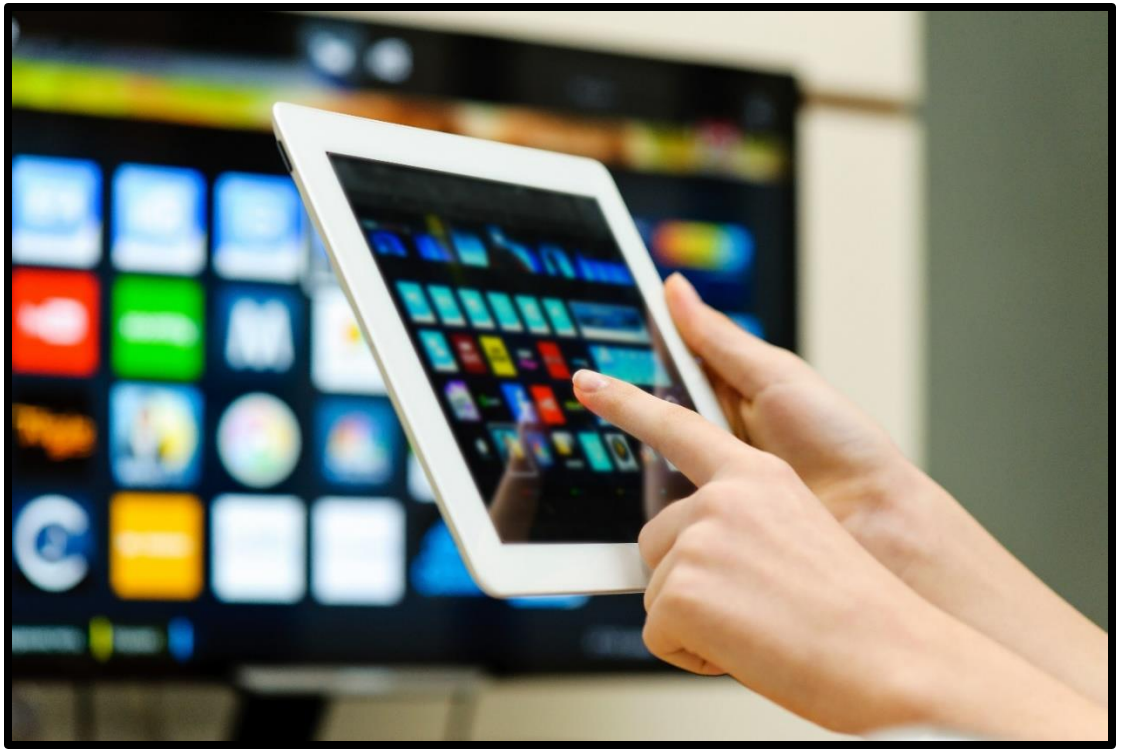


Large code base – loaded IDE

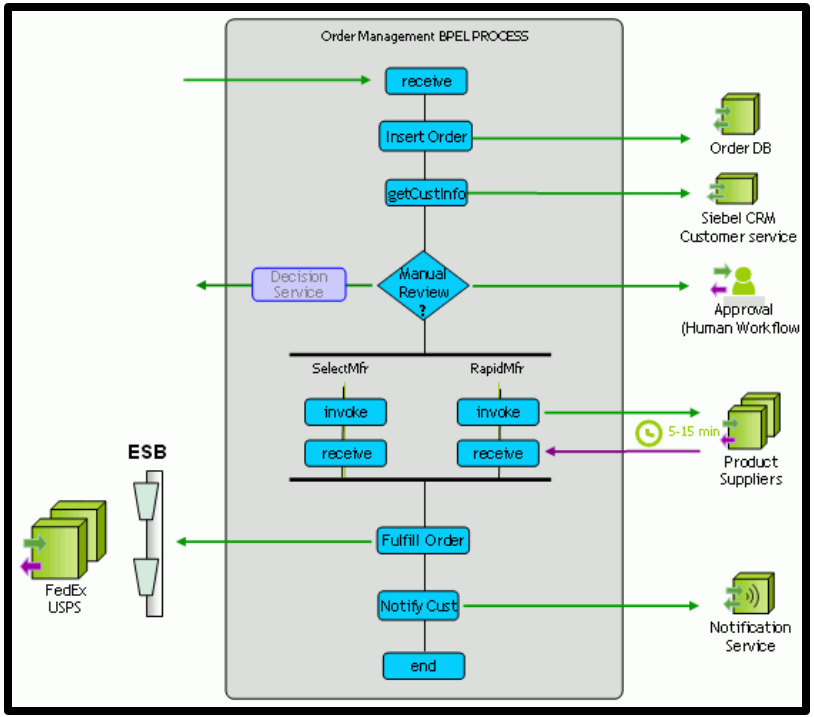


Long cycle Testing and Deployment

Application stability



Complex Orchestration (SOA, ESB, etc.)



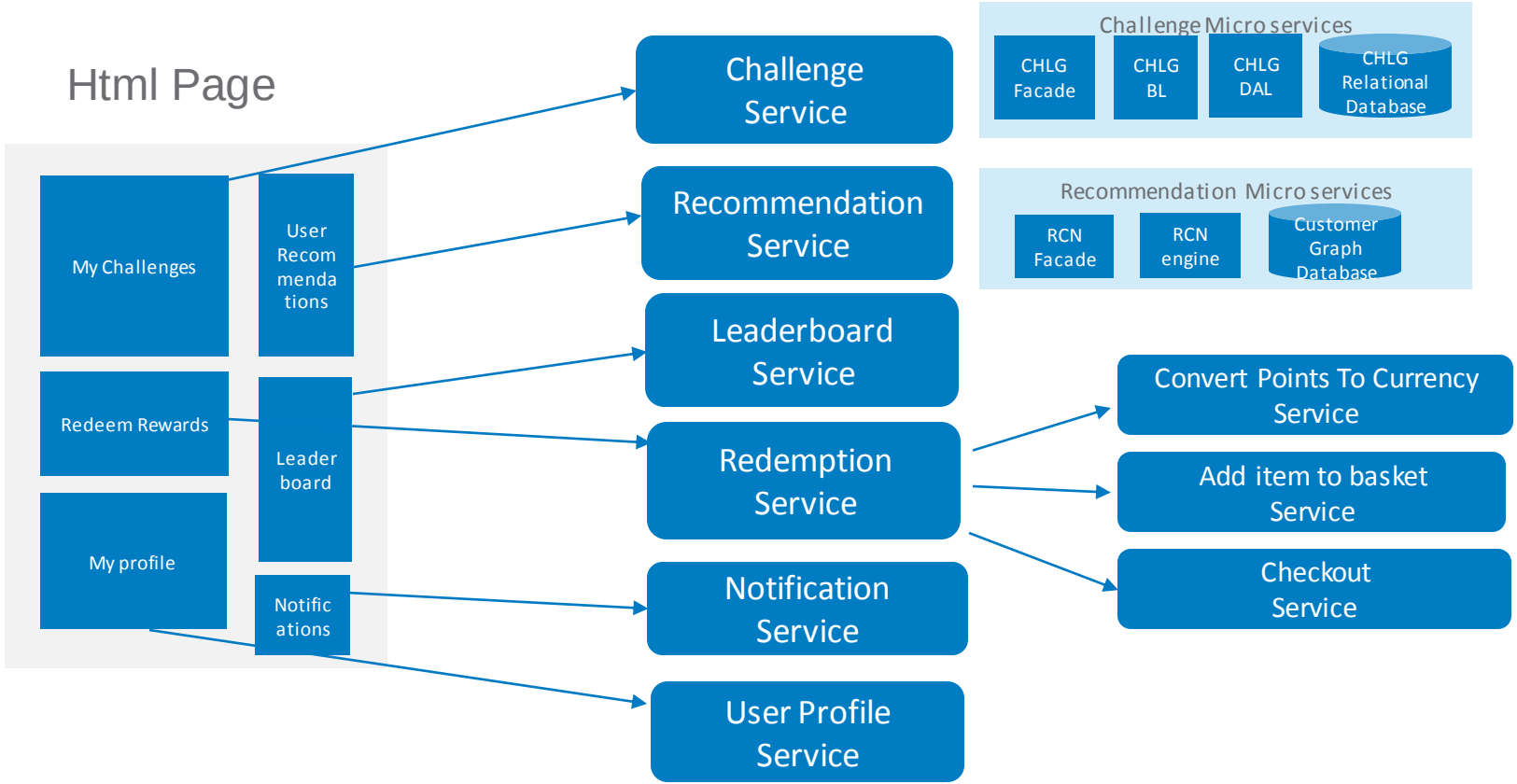
What is Microservices?

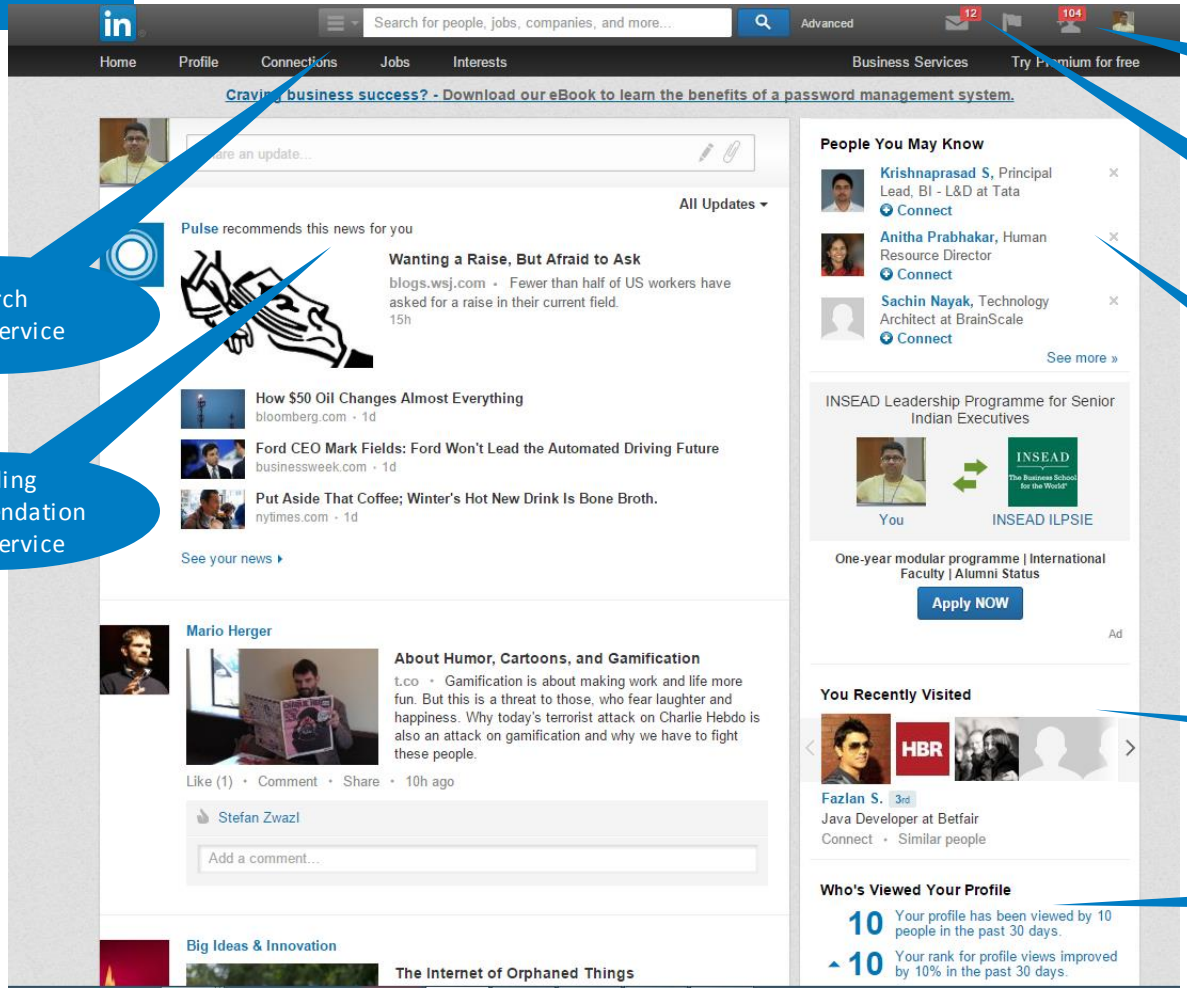
Definition

Microservices architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API

Reference - https://www.linkedin.com/pulse/20140826225233-22873789-microservices-architecture-collaborating-applications-via-services

Microservices Architectural View





Search Microservice

Reading Recommendation Microservice

Connection Request Microservice

Notifications Microservice

People You May Know Microservice

You visited Microservice

Who viewed Your Profile Microservice

Microservices Architecture-Characteristics

To architect large, complex and long lived applications as a set of cohesive services

Services can be short lived

Services are functionality oriented (challenges, orders, catalog, Leaderboard, Notifications, etc.)

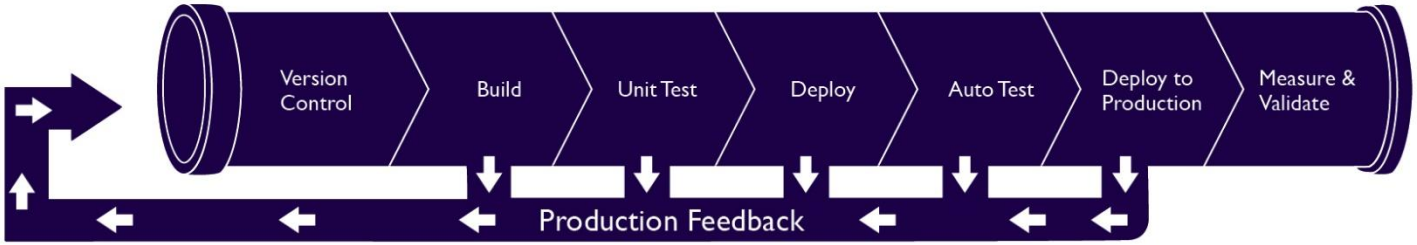
Services are developed and deployed independent of one another - container

Resources (memory, cpu) throttling based on services

Services usually communicate using HTTP/REST

Each Service can have its persistence store or shared between set of services

Facilitates continuous deployment/delivery of business functionality



Each services team has varied roles (no centralized teams)

Service teams interacts with end customer (you build, you run)

Service teams are usually of size 6 to 10 (2 pizza teams)

Modeling, Scaling Microservices

How to model Services?

- Partition services using noun, verb or use case

E.g.

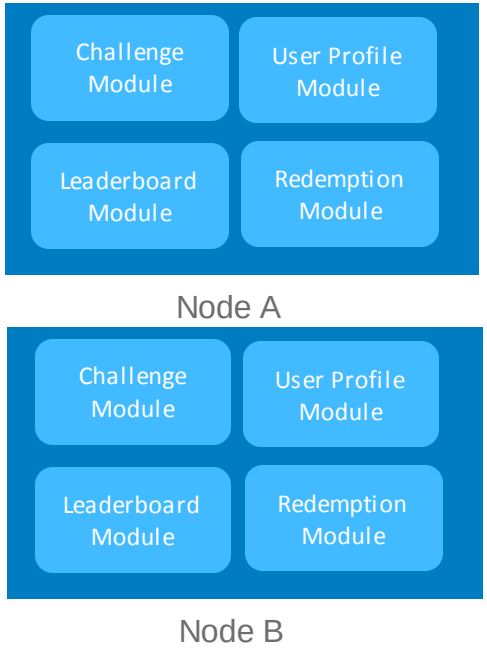
- Use case, or Verb -> login
- Noun, Or resource -> Inventory Service, Catalog Service

- Group things that change at the same time in the same module

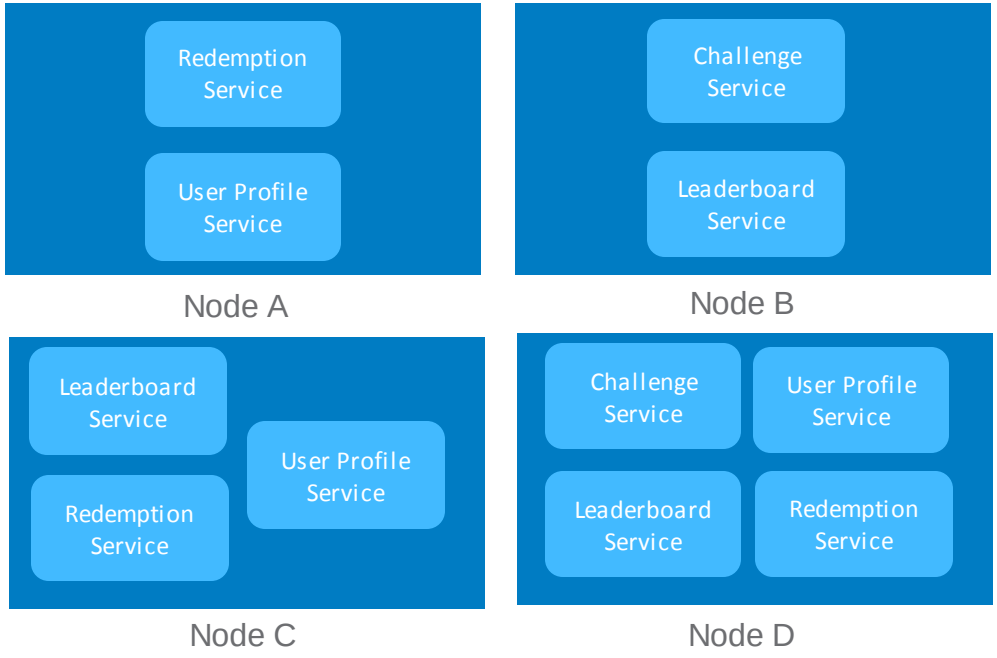
How to Scale - Deployment View

Each element of functionality into different service, scales by distributing services across servers, replicating as needed

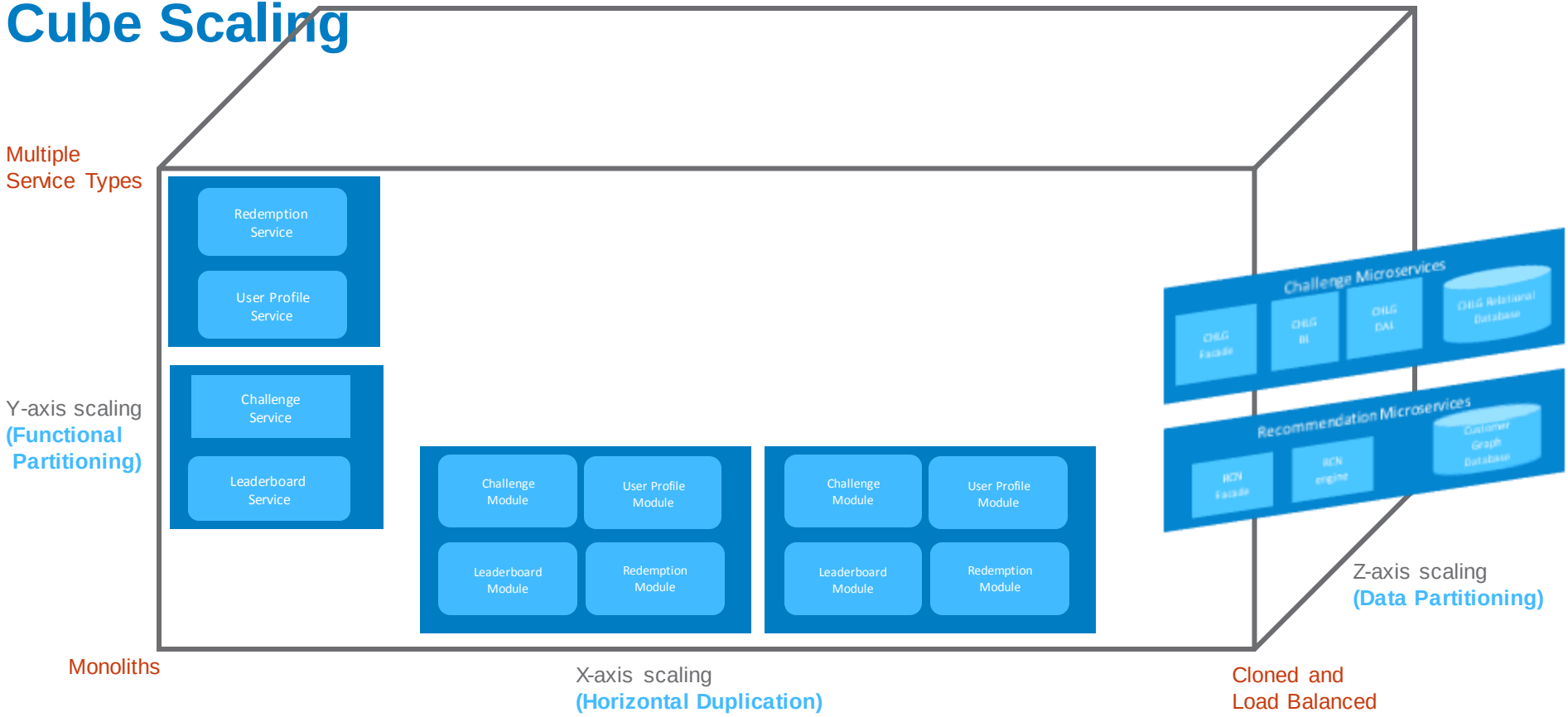
Monolithic (n-tiered) Architecture



Microservices Architecture



Cube Scaling

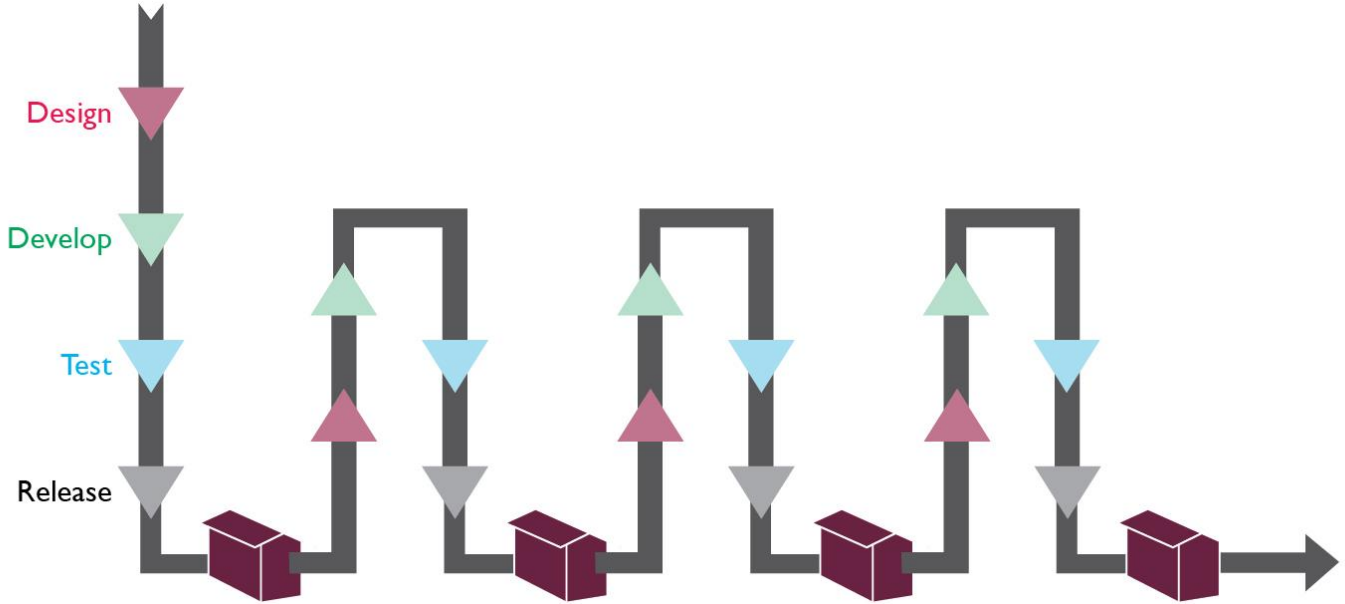


Microservices Augmenting Elements

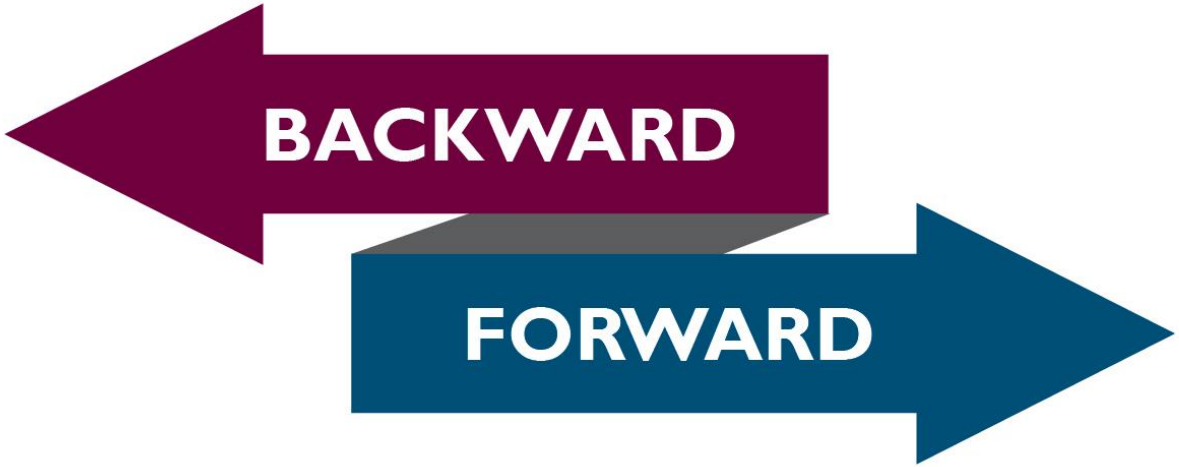
Decentralized ownership, code base, release processes

Service centric Release management and ownership

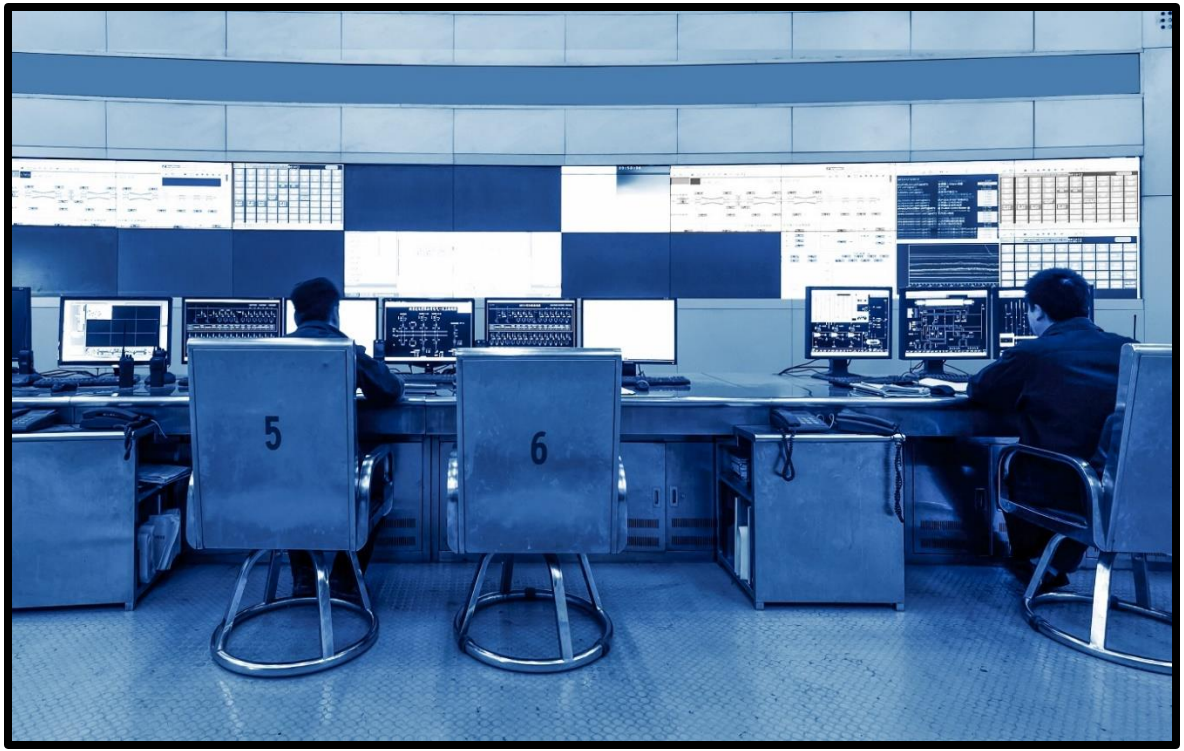
Continuous Delivery



Backward compatibility owned by individual services team

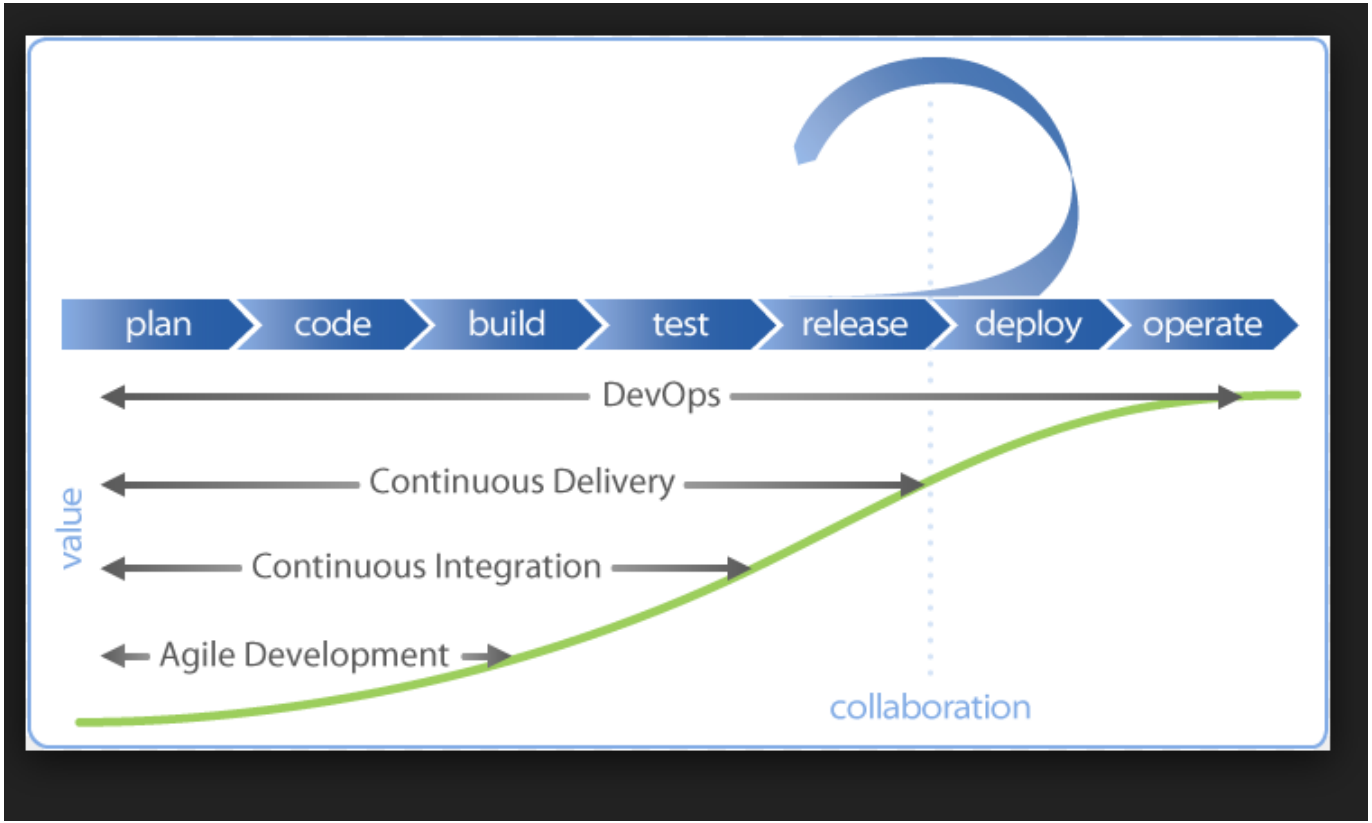


Needs auto provisioning, monitoring elements to be in place



Benefits Of Microservices architecture

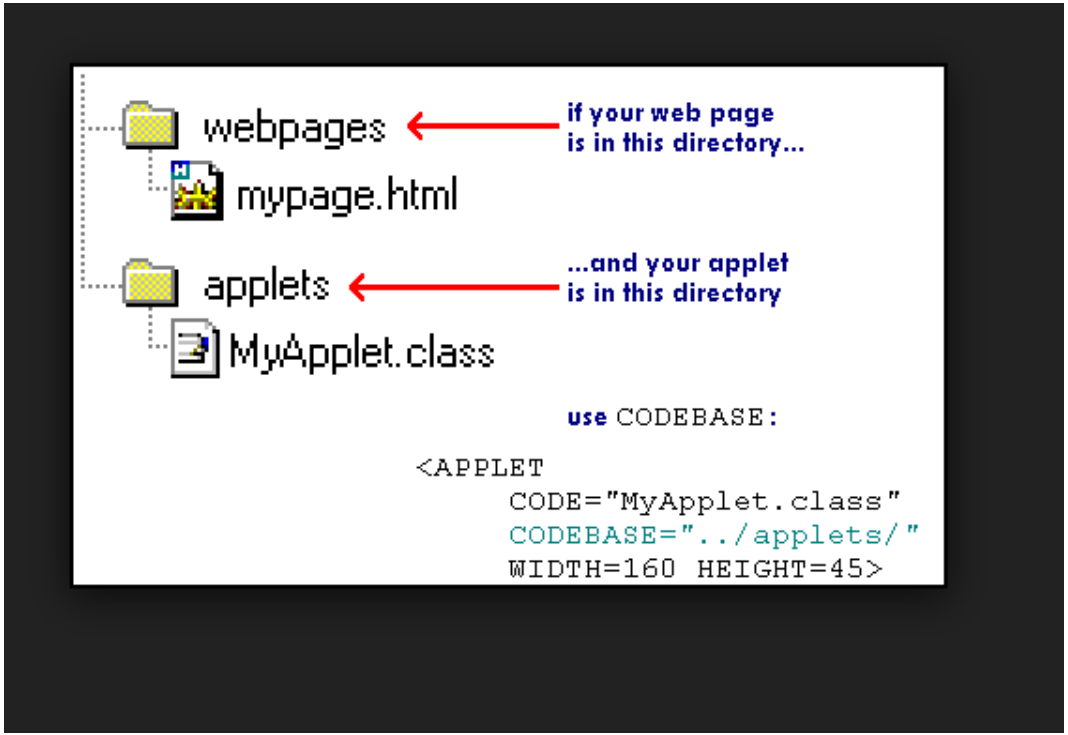
Continuous deployment



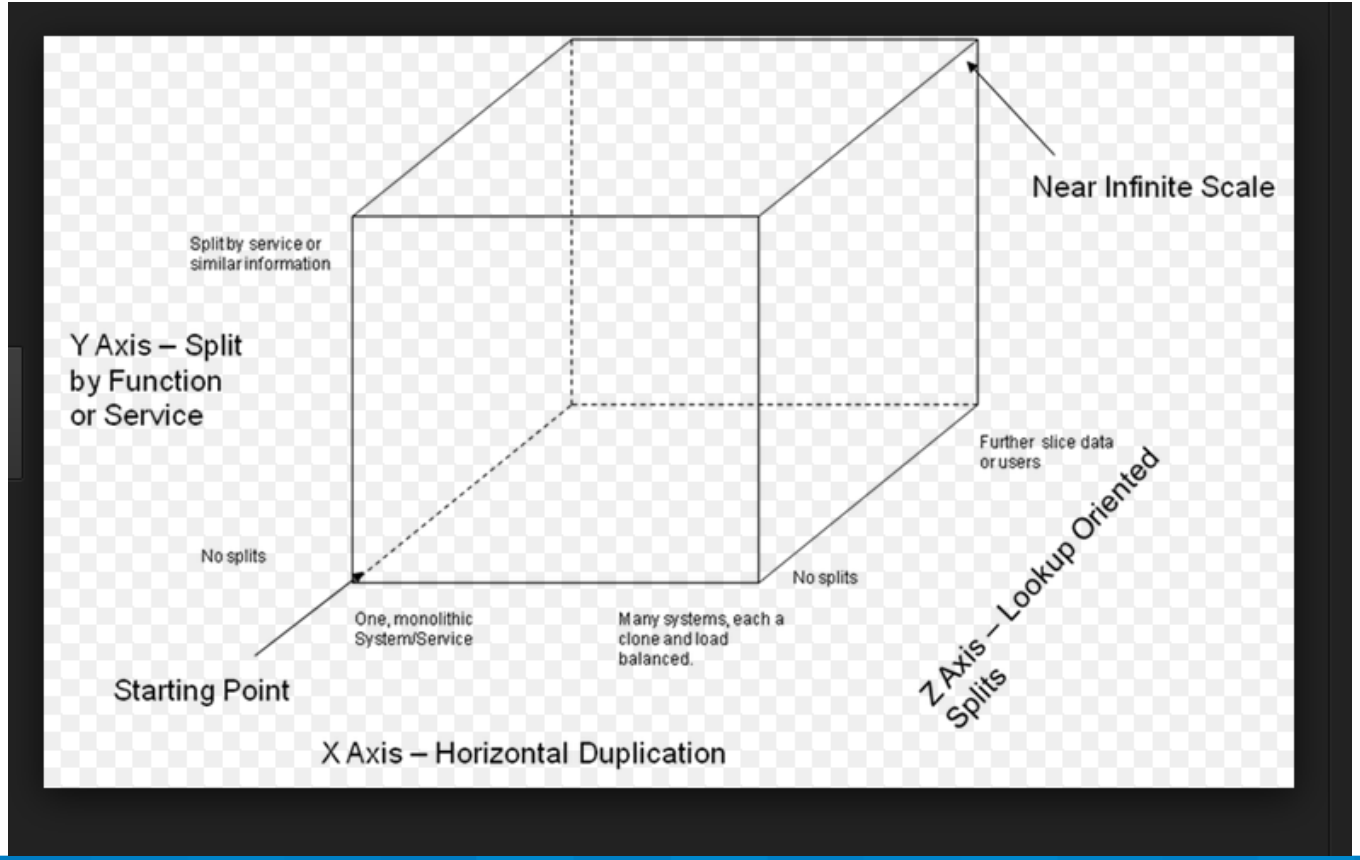
Allows technology experimentation

Functionality based partitioning

Lighter code base



Cube scaling can be easily achieved



Limitations

Limitations of Microservices Architecture

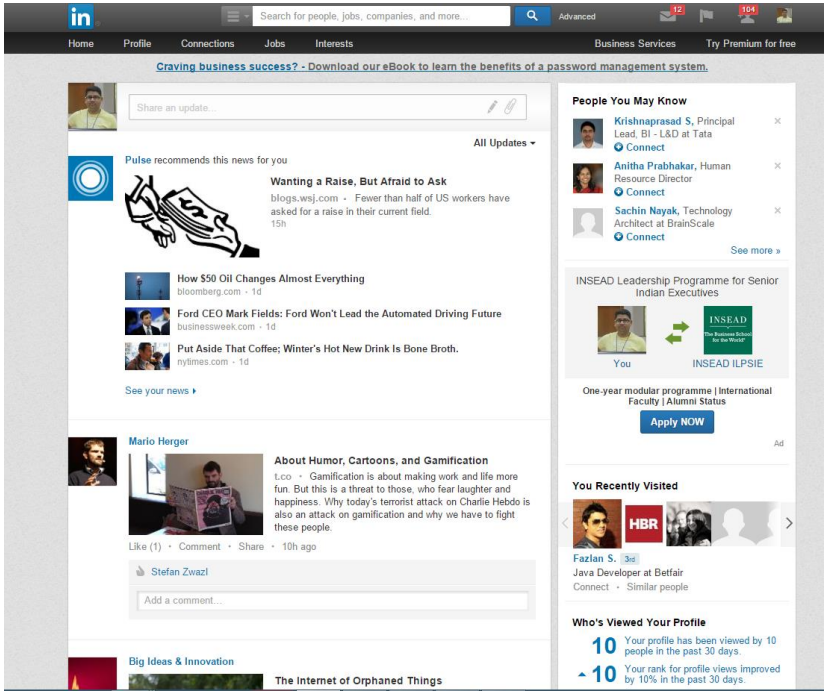
- Increased Memory Footprint (separate process)
- Existing IDEs not suited, supporting technologies, frameworks are evolving
- Services needing Distributed Transactions
- Implement use cases that needs coordination between distributed teams

Limitations of Microservices Architecture

- Deployment and management complexity
- Duplication of effort, skills

Industry Adoption and Usage Scenarios

LinkedIn Implementation



- Uses microservices architecture (evolved from monolithic to Micro services architecture)
- The linked page has multiple parts each is populated asynchronously using respective REST service
- Limits aggregation to front-tier
- Limits cross-talk in the back tier (defined tier boundaries)
- Pre- and post commit automated testing

rest.li framework

Services were fine grained but monolithic build and release processes will not allow the benefit of Micro Services Architecture

Home

gmcmillan100 edited this page on Dec 1, 2014 · 65 revisions

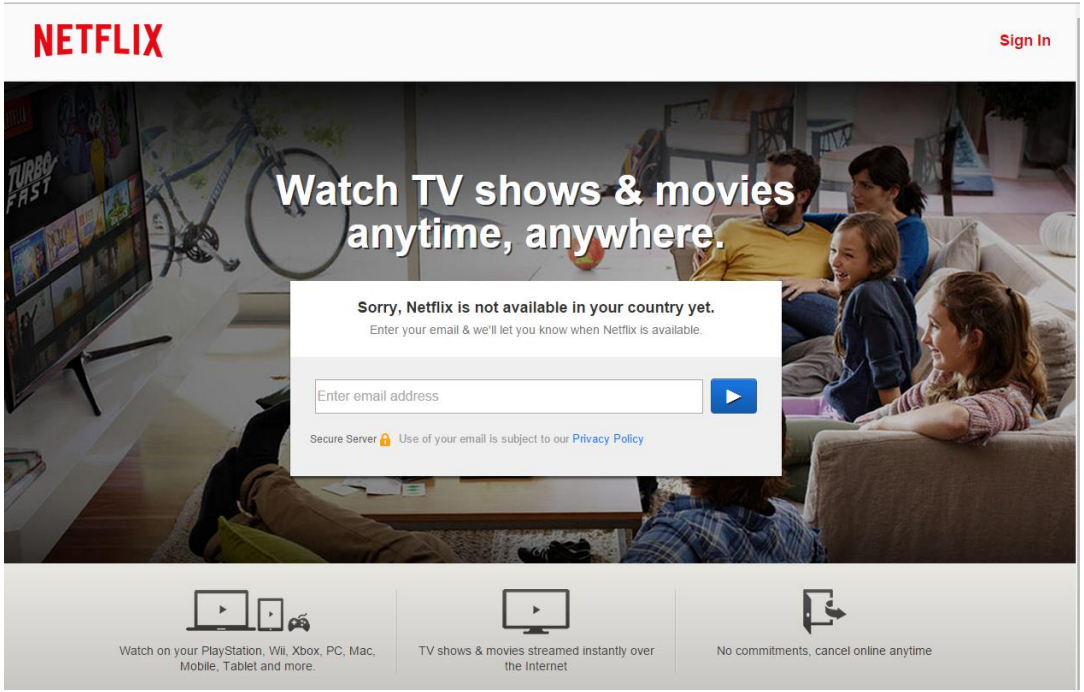
Rest.li is an open source REST framework for building robust, scalable RESTful architectures using type-safe bindings and asynchronous, non-blocking IO. Rest.li fills a niche for applying RESTful principals at scale with an end-to-end developer workflow for buildings REST APIs that promote clean REST practices, uniform interface design, and consistent data modeling.

Source | Documentation | Discussion Group

Amazon.com Implementation

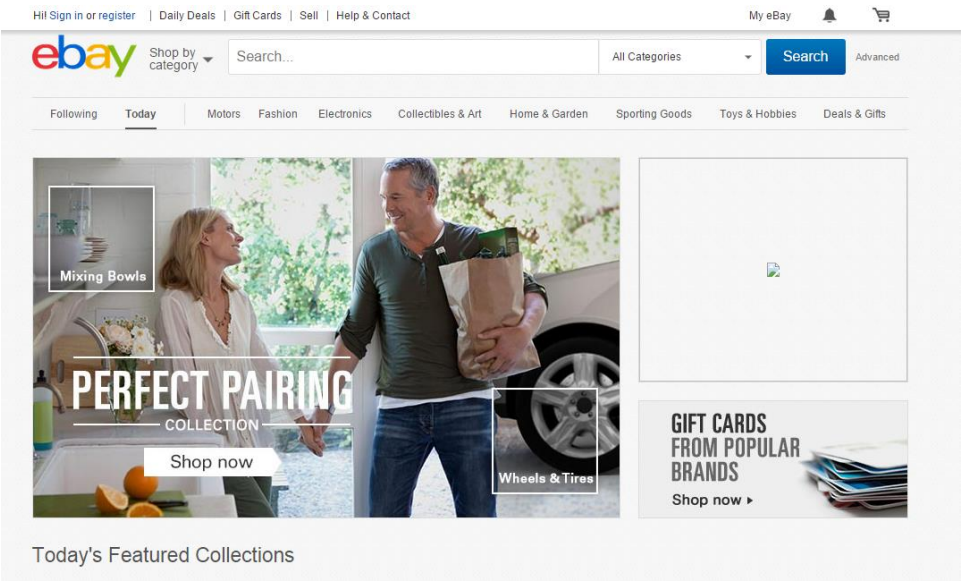
- Originally had a two-tier architecture.
- Migrated to a service-oriented architecture consisting of hundreds of backend services.
- The Amazon.com website application calls 100-150 services to get the data that used to build a web page.
- Several internal and external applications call these services to fulfill respective functionality.

Netflix Implementation



- Is a popular video streaming service
- Responsible for up to 30% of internet traffic,
- had a large scale, service-oriented architecture.
- handles over a billion calls per day to their video streaming API from over 800 different kinds of devices.
- Each API call fans out to an average of six calls to backend services.

ebay.com



- monolithic architecture -> service-oriented architecture.
- The application tier consists of multiple independent applications.
- Each application implements the business logic for a specific function area such as buying or selling.
- Uses cube scaling

Usage Scenarios

- Building large scale Internet facing web sites
  - Retail ecommerce shopping portals
  - Online Banking portals
  - Job Portals
- Building internet scale software products

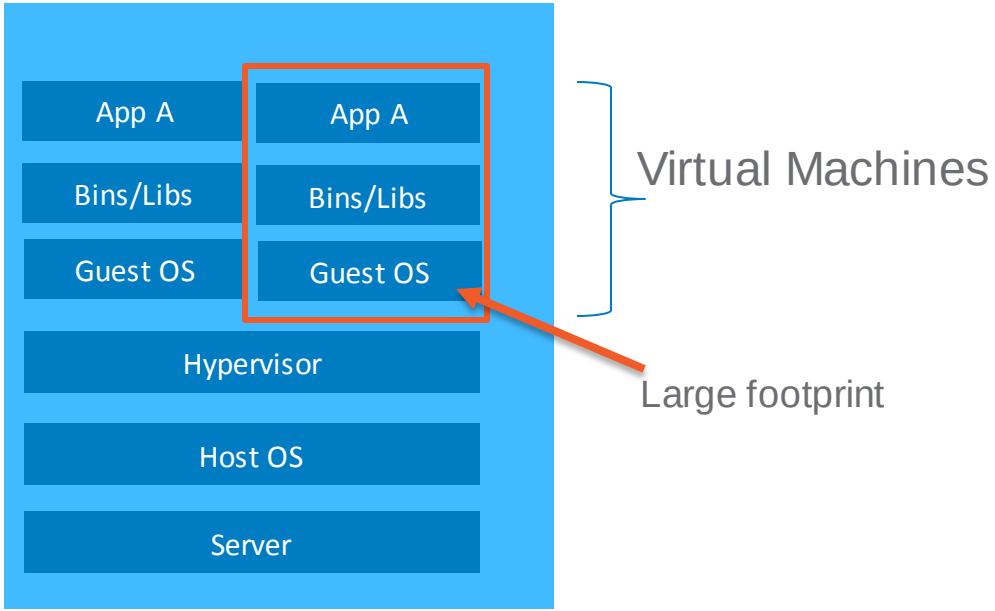
Usage Scenarios

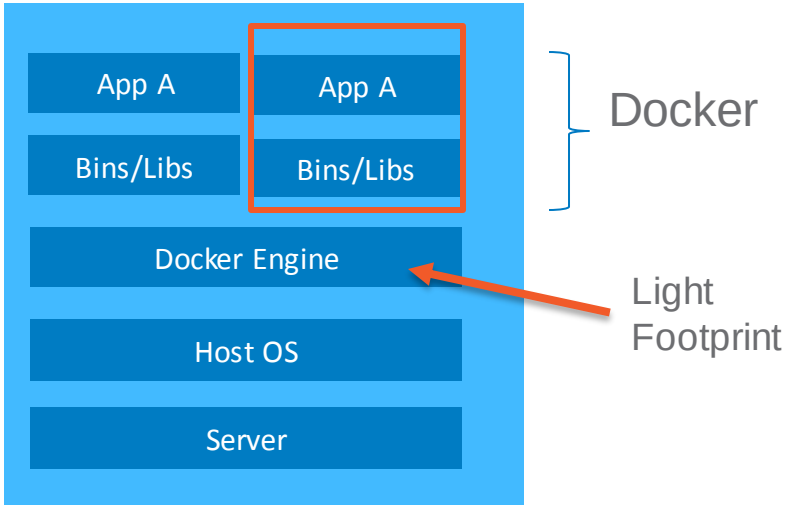
- Building Social Applications
  - Twitter
  - LinkedIn
  - Facebook
- Building B2E intranet portals
- Building multi-channeled Applications
- Highly relevant to products

Dockers

Docker as a Container

- Docker is an open source container virtualization platform (licensed under Apache 2.0 license)
- Used to build, ship and run distributed applications.

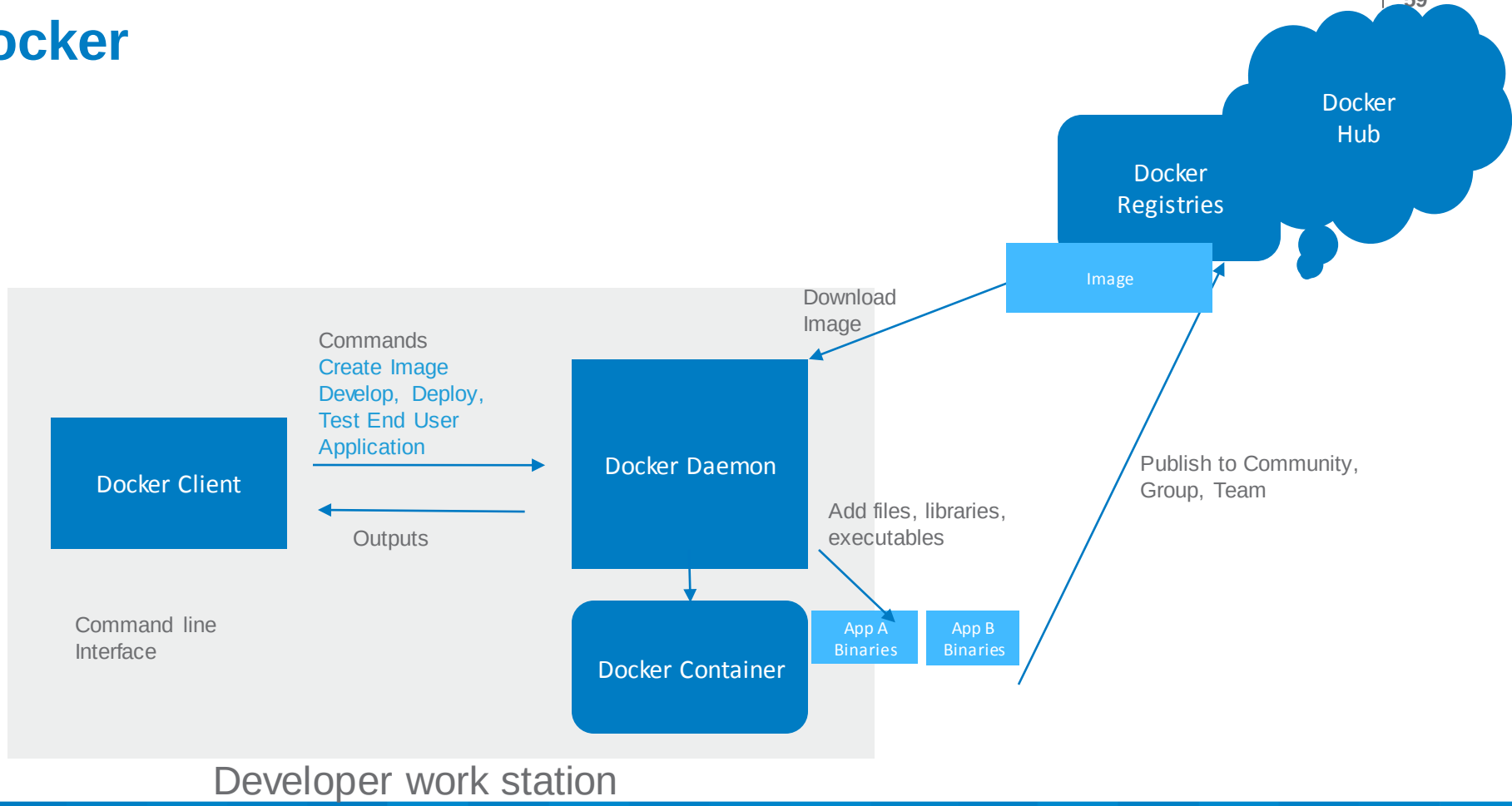




Docker

- Docker engine – portable lightweight runtime and packaging tool
- Docker container holds everything that is needed to run an application - Operating system, database, application specific binaries
- Docker container is the deployable packaging unit and updated using incremental updates (layers)
- Each container has its IP address
- Dockers uses union files system to create layers
- Docker Control Groups allows setting limit on resource usage on Docker Container
- Docker hub is a cloud service to share applications and manage workflows

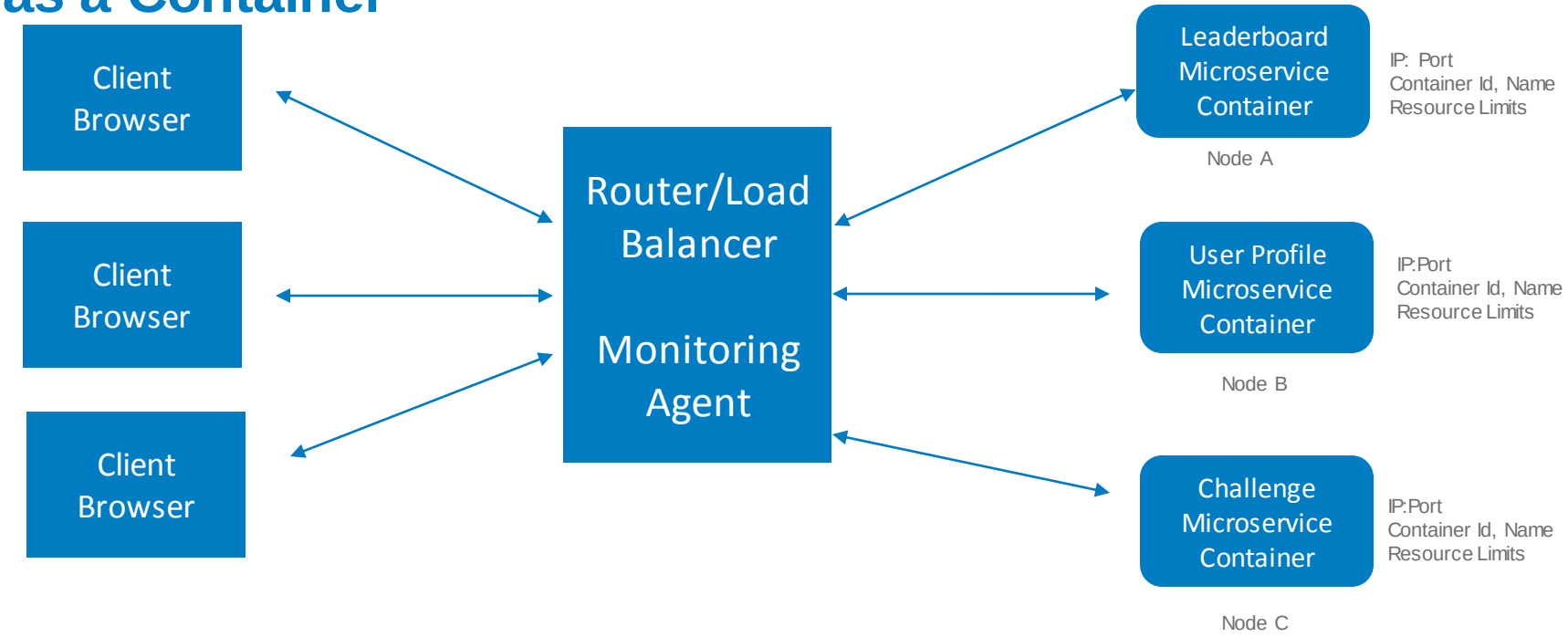
Docker


Docker Hub
Docker Registries
Image
Download Image
Commands
Create Image
Develop, Deploy, Test End User Application
Docker Client
Docker Daemon
Outputs
Add files, libraries, executables
Publish to Community, Group, Team
Command line Interface
Docker Container
App A Binaries
App B Binaries
Developer work station

Container technologies for Desktop, Cloud

- Boot2Docker – Linux VM to run on Microsoft Windows Server and Azure Cloud
- Amazon EC2 container services (ECS) – helps run and manage Docker on Amazon cloud. It provides container management services such as
  - Cluster management
  - Scheduling
  - High Performance
  - Extensible and Portable
  - AWS integration
  - Resource efficiency
  - Security

Logical Micro services deployment Architecture using Docker as a Container



Learnings in the context of Infosys

- Docker runs only on 64 bit platform
- You can download the Docker from sparsh->downloads (choose 64bit option)
- DO NOT try to configure Docker on Infosys Public Cloud VM – needs underlying machine CPU virtualization to be set
- To download Docker images from Docker hub or Git Hub, appropriate Infosys proxy needs to be set in the boot2docker for HTTP_PROXY setting
  - Docker pull Microsoft/aspnet – will download the ASP.NET vNext image

```
you@tutorial:~$
you@tutorial:~$ docker
Usage: Docker [OPTIONS] COMMAND [arg...]
-H="127.0.0.1:4243": Host:port to bind/connect to

A self-sufficient runtime for linux containers.

Commands:

attach    Attach to a running container
build     Build a container from a Dockerfile
commit    Create a new image from a container's changes
diff      Inspect changes on a container's filesystem
export    Stream the contents of a container as a tar archive
history   Show the history of an image
images    List images
import    Create a new filesystem image from the contents of a tarball
info      Display system-wide information
insert    Insert a file in an image
inspect   Return low-level information on a container
kill      Kill a running container
login     Register or Login to the Docker registry server
logs      Fetch the logs of a container
port      Lookup the public-facing port which is NAT-ed to PRIVATE_PORT
ps        List containers
pull      Pull an image or a repository from the Docker registry server
push      Push an image or a repository to the Docker registry server
restart   Restart a running container
rm        Remove a container
rmi       Remove an image
run       Run a command in a new container
search    Search for an image in the Docker index
start     Start a stopped container
stop      Stop a running container
tag       Tag an image into a repository
version   Show the Docker version information
wait      Block until a container stops, then print its exit code
you@tutorial:~$ docker version
Docker Emulator version 0.1.3

Emulating:
Client version: 0.5.3
Server version: 0.5.3
Go version: go1.1
you@tutorial:~$ docker search "tutorial"
Found 0 results matching your query (""tutorial"")
NAME                  DESCRIPTION
you@tutorial:~$
```

References

LinkedIn Implementation

- http://www.slideshare.net/parikhk/restli-and-deco
- https://qconsf.com/presentation/monolith-microservices-rest-evolution-linkedins-service-architecture

Twitter

- http://highscalability.com/blog/2014/7/28/the-great-microservices-vs-monolithic-apps-twitter-melee.html

Amazon

http://highscalability.com/amazon-architecture

Microservices

- http://microservices.io/patterns/microservices.html

References

Martin Fowler article

- http://martinfowler.com/articles/microservices.html
- http://www.thoughtworks.com/talks/software-development-21st-century-xconf-europe-2014

Docker

- https://www.docker.com/tryit/

Microsoft Boot2Docker

- https://github.com/jayway/Aspnet.Docker.Lab
- http://blogs.msdn.com/b/webdev/archive/2015/01/14/running-asp-net-5-applications-in-linux-containers-with-docker.aspx

Connect Architecture

Thank You

Acknowledgements : Sudhanshu Hate

Infosys® | Building Tomorrow's Enterprise

Facebook

- Facebook is also Microservices architecture
- Uses various databases to deliver various use cases
- MySQL for structured data such as walls, user information, posts, etc.
- Memcache – for speeding dynamic driven websites for caching data in RAM
- Haystack and Varnish – for storage of pictures
- Scribe – for logging
- Migrated Cassandra to Hbase for Inbox search

Microsoft promise on Microservices

- Docker will be supported on Microsoft Windows and Windows Azure

Microservices	Technology support
Process isolation	ASP.NET worker process, application pool
Independent services development	Visual Studio project MVC REST service WCF Web API
Service deployment	IIS AppFabric
Service Monitoring	AppFabric

Connect Architecture

Thank You

Acknowledgements : Sudhanshu Hate

Infosys® | Building Tomorrow's Enterprise