

Building Big Architecture

Ramit Surana

@ramitsurana

ramitsurana@gmail.com



Agenda

- Why is this important ?
- Defining SOA
- What are Monoliths ?
- What are Microservices ?
- About Docker
- Principles of Microservices
- Refactoring
- 12 Factor App
- Conway's Law
- Coupling
- Case Studies



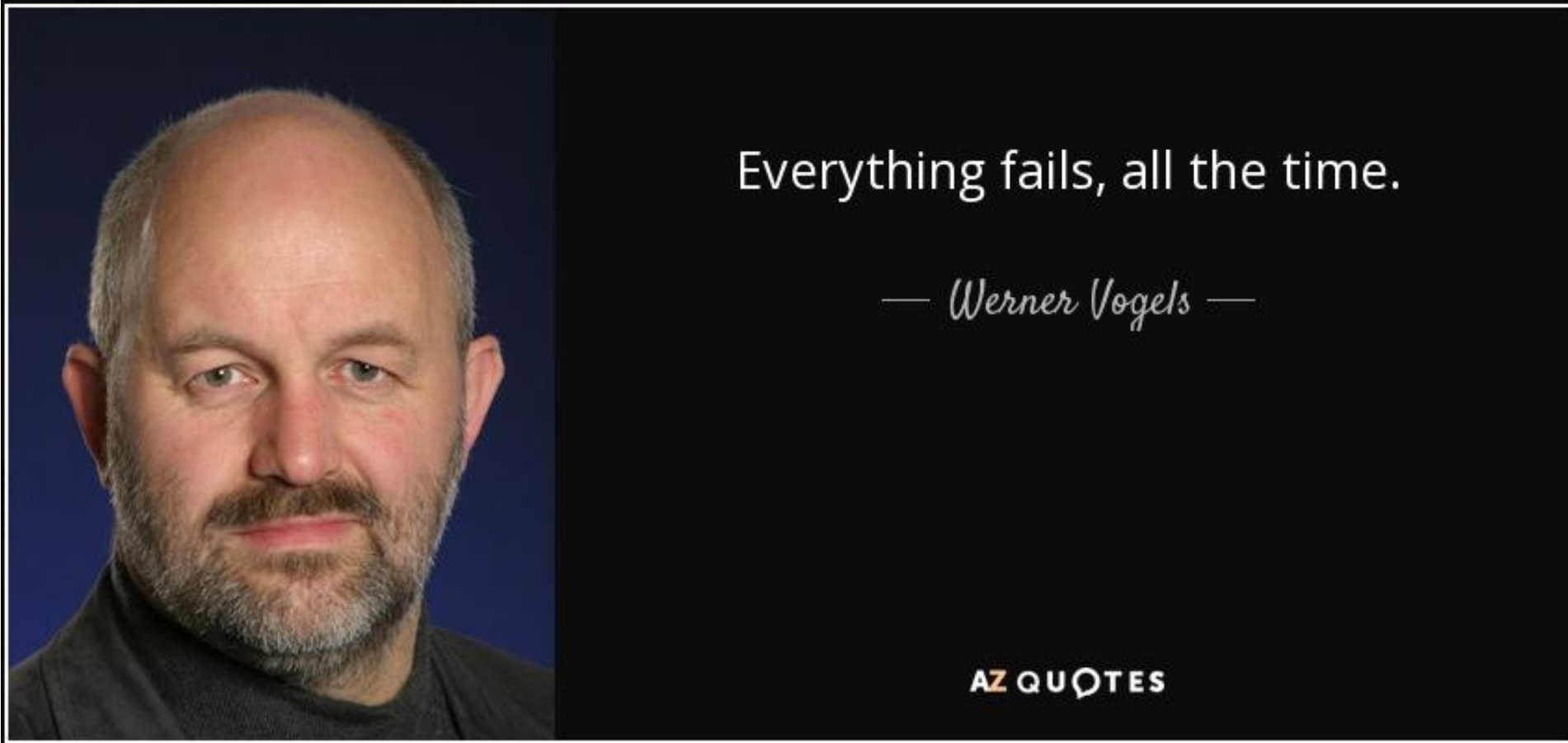
About Me

- Open Source Guy
- Contributor to Docker, CoreOS and Kubernetes community.
- Open Source community speaker.
- Contact me:









ramitsurana@gmail.com



Important Sayings



Why is this important ?

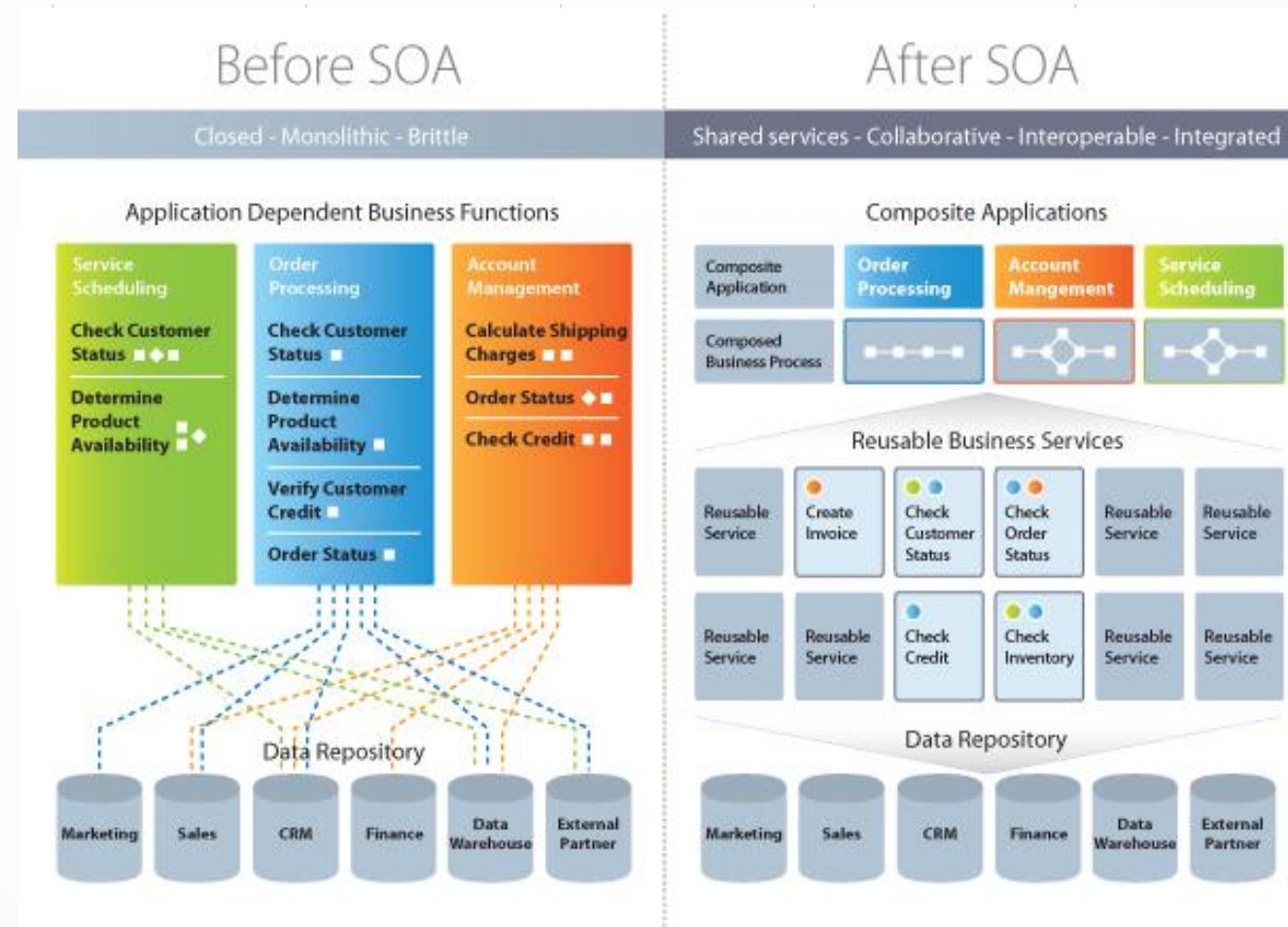
	Old style of IT		New style of IT
Information	 Business intelligence Structured data	➔	 Real-time analytics Multi-media info
Teams & plans	 Single source Gradual & procedural	➔	 Collaborative ecosystems Quick & continuous
Apps & architectures	 Feature bloat Closed & siloed	➔	 Targeted, precise Open & interconnected
Infrastructure	 Centrally planned & procured Dedicated	➔	 Instant & elastic Shared

SOA(Service Oriented Architecture)

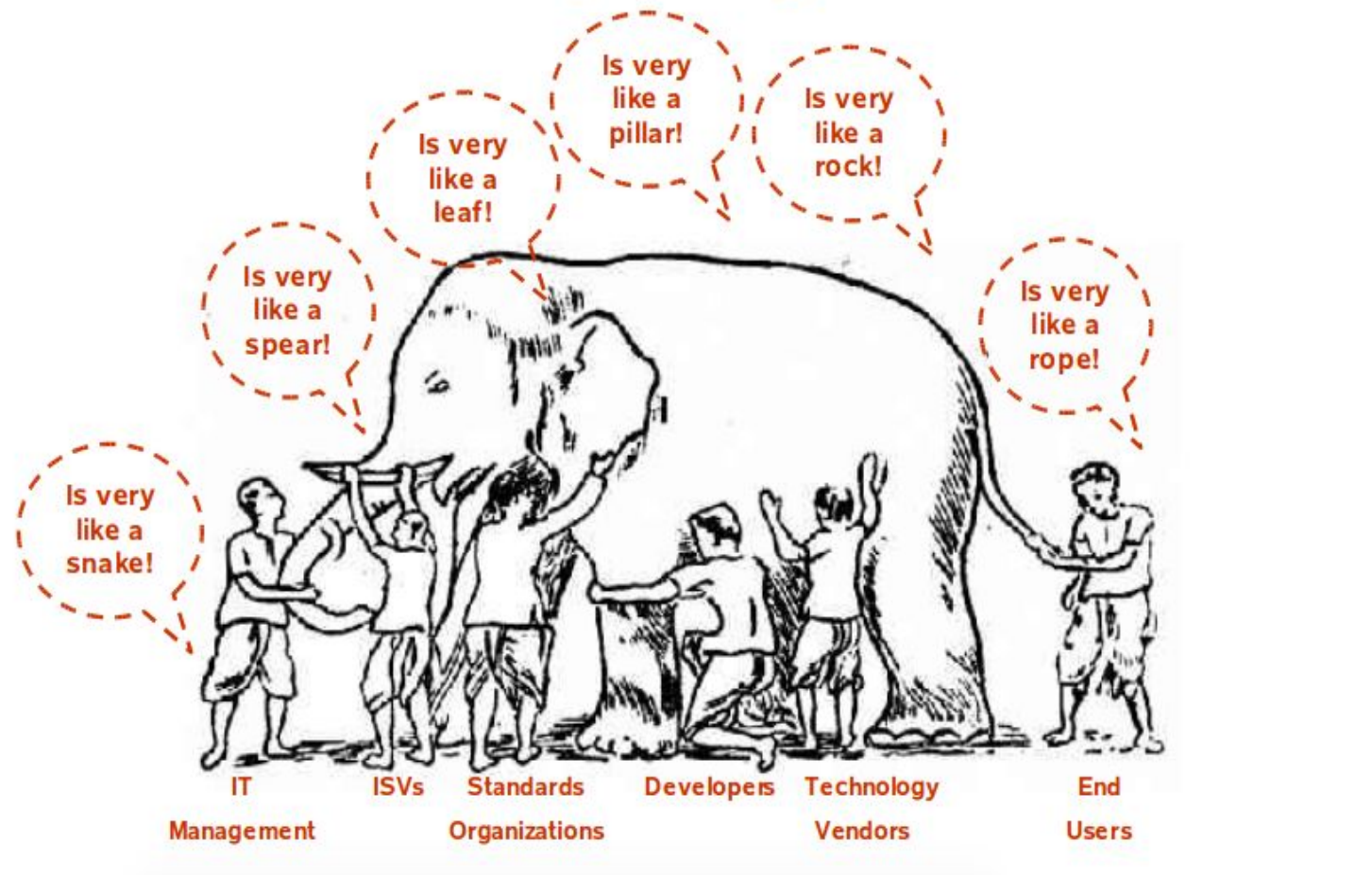
Architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network.



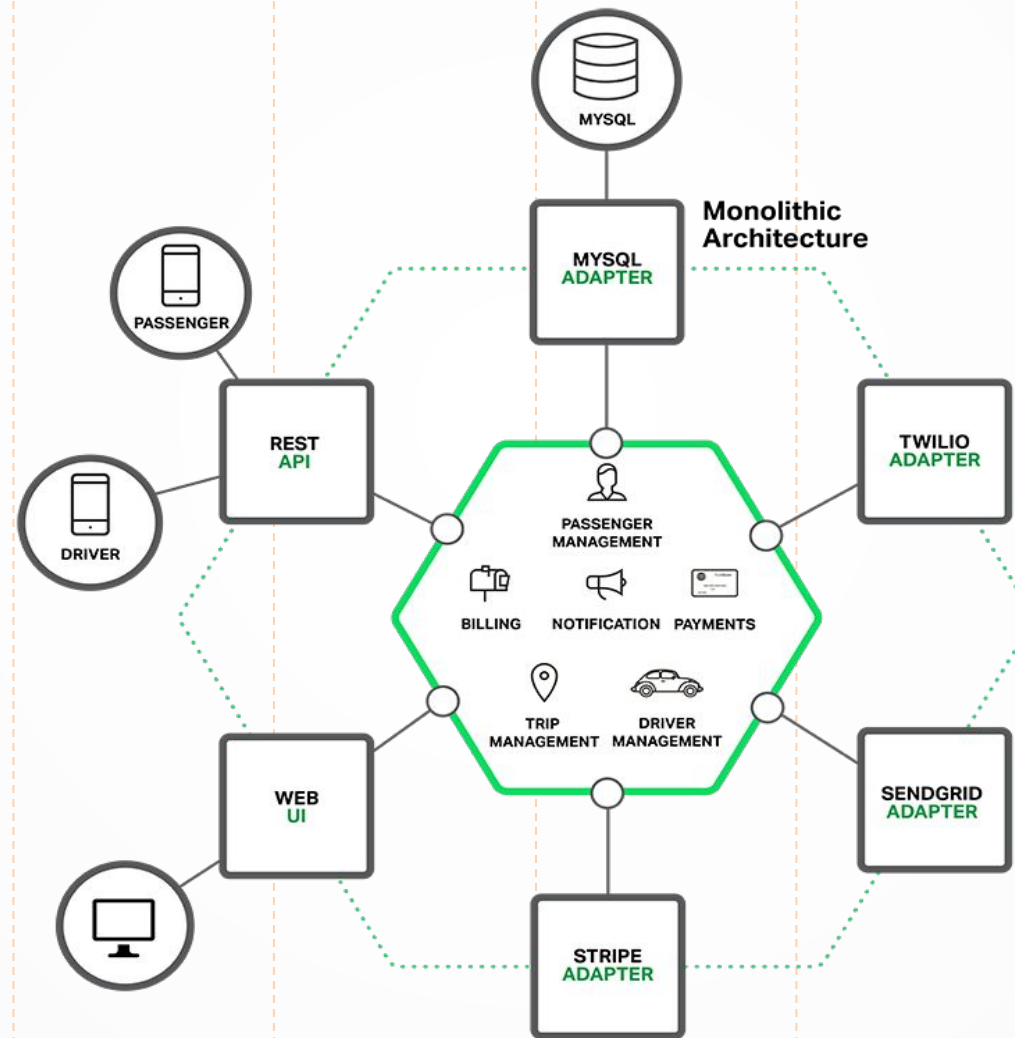
Before and After SOA



SOA Mythology

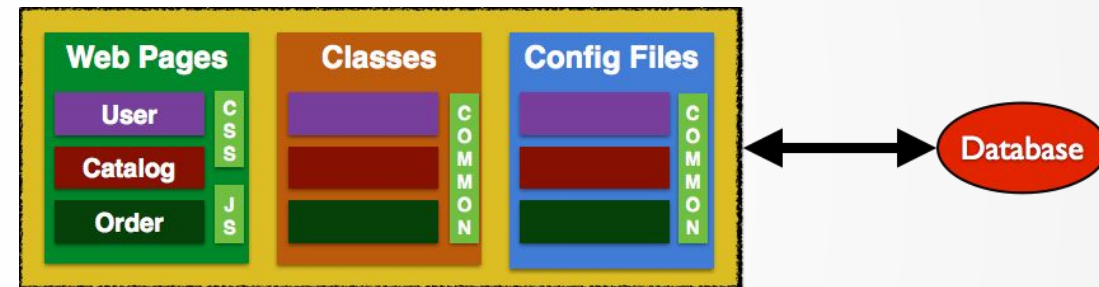


Overview of Monolith Architecture

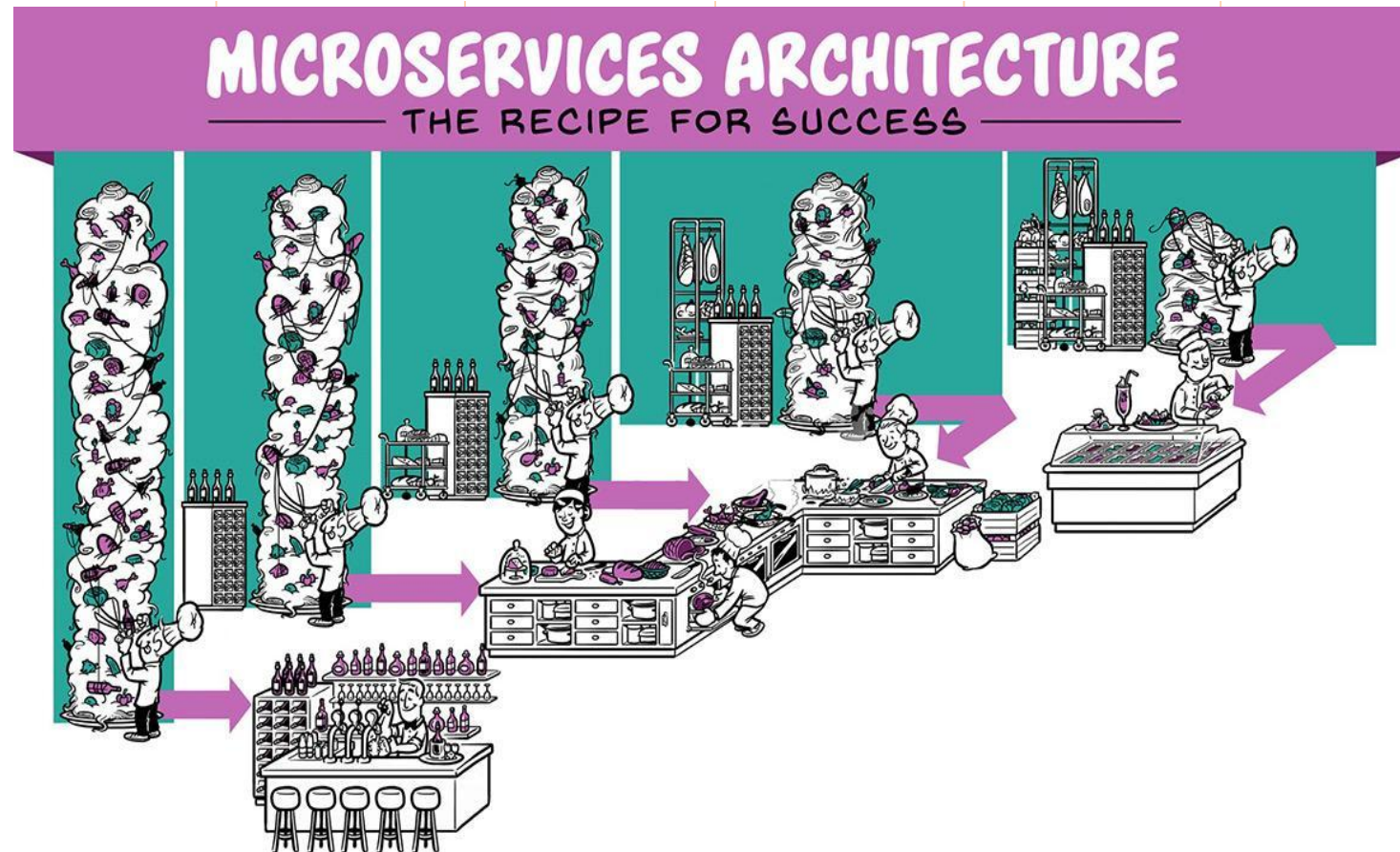


Monolithics

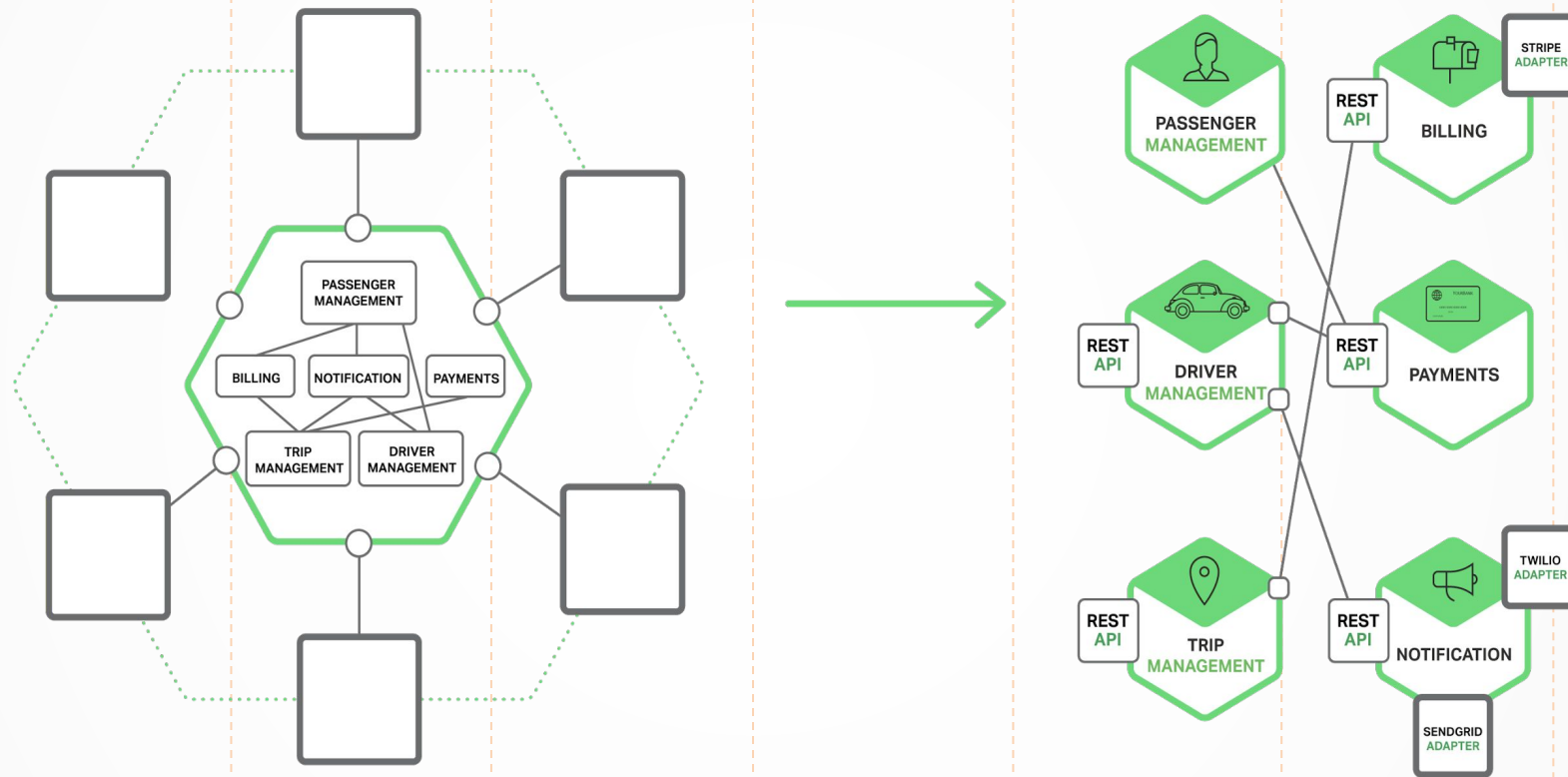
- Single Runtime
- Single Codebase
- Layered architecture
- Initialization of the system may be tricky or laborious.
- Change to the control flow is impossible.
- An application where all of the logic runs in a single app server.



Microservices

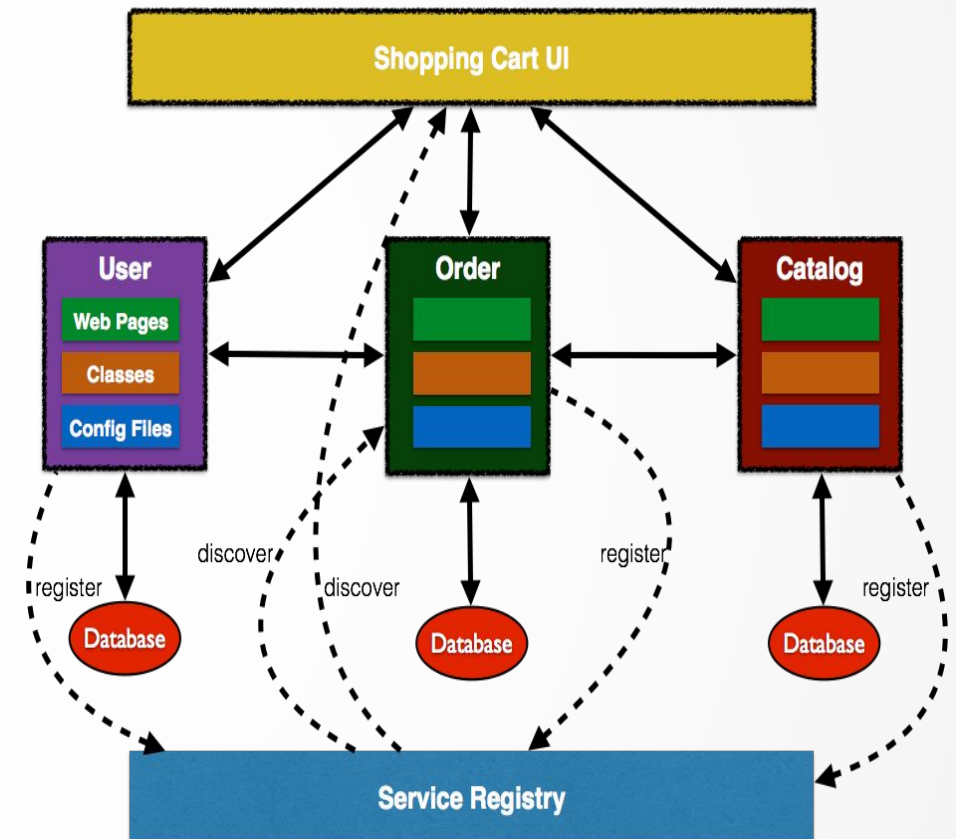


Overview of Microservices Architecture

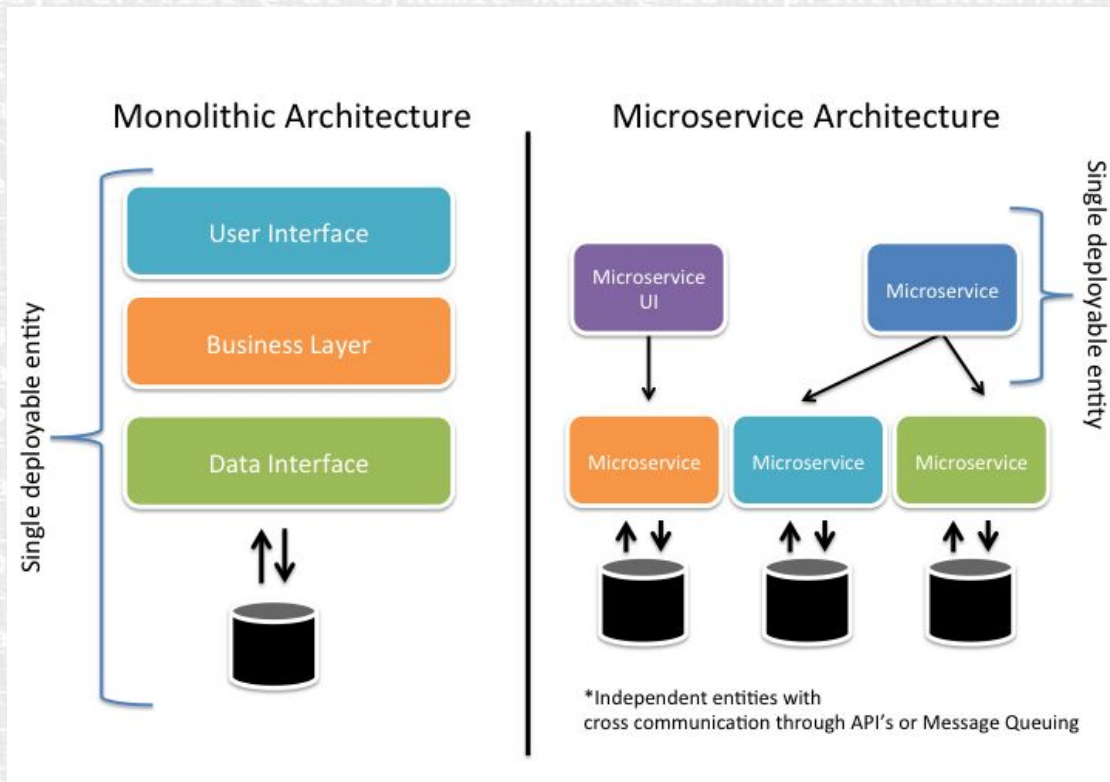


Microservices

- Loosely coupled service oriented architecture with bounded contexts.
- Design for failure
- Decentralized Governance
- Decentralized Data Management.
- Componentization via Services



Microservices vs Monolithic



Category	Monolithic architecture	Microservices architecture
Code	A single code base for the entire application.	Multiple code bases. Each microservice has its own code base.
Understandability	Often confusing and hard to maintain.	Much better readability and much easier to maintain.
Deployment	Complex deployments with maintenance windows and scheduled downtimes.	Simple deployment as each microservice can be deployed individually, with minimal if not zero downtime.
Language	Typically entirely developed in one programming language.	Each microservice can be developed in a different programming language.
Scaling	Requires you to scale the entire application even though bottlenecks are localized.	Enables you to scale bottle-necked services without scaling the entire application.

About Docker

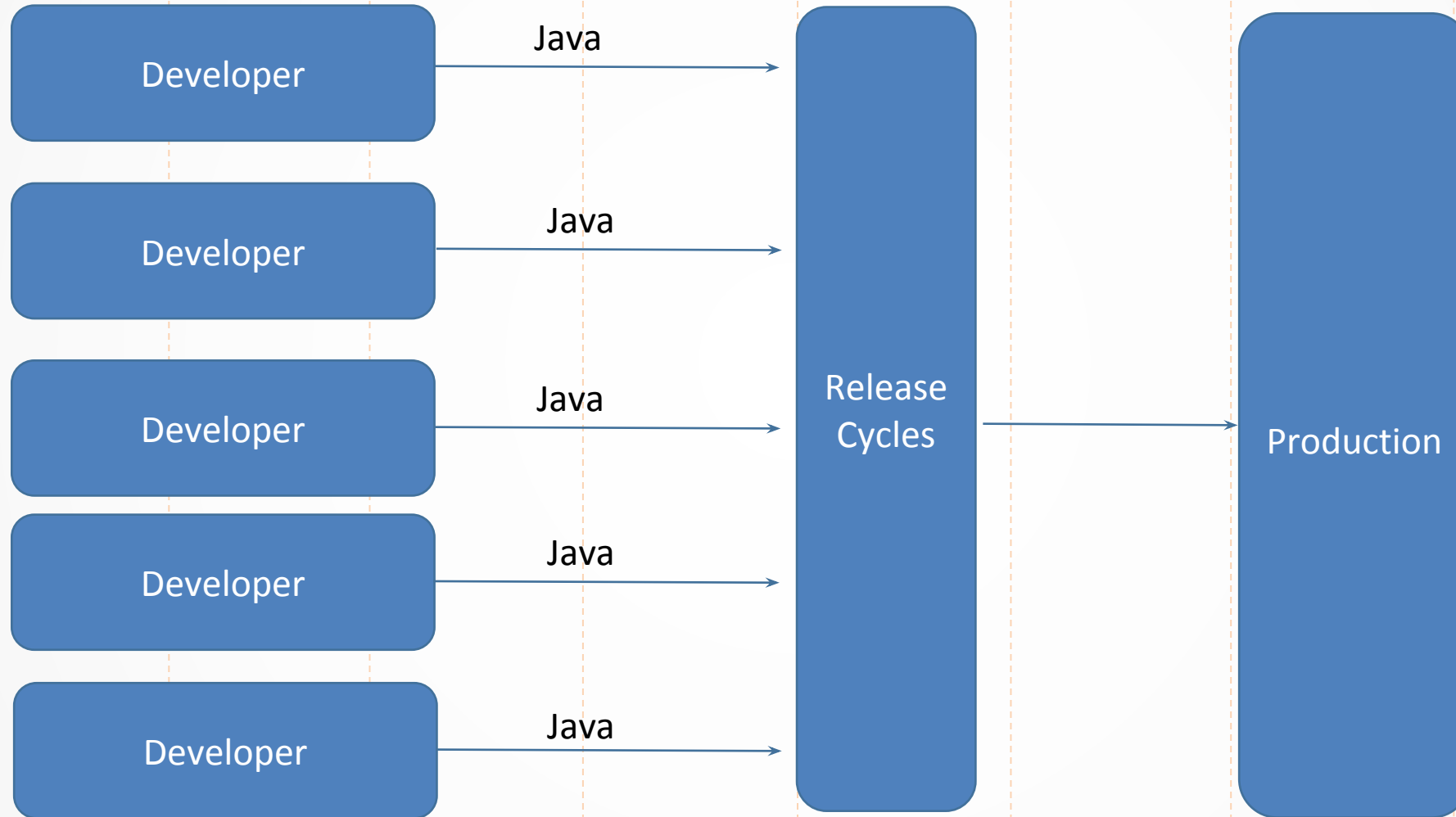
- Open platform for developers and sysadmins to build, ship, and run distributed applications.
- Docker enables apps to be quickly assembled from components.
- It eliminates the friction between development, QA and production environments.



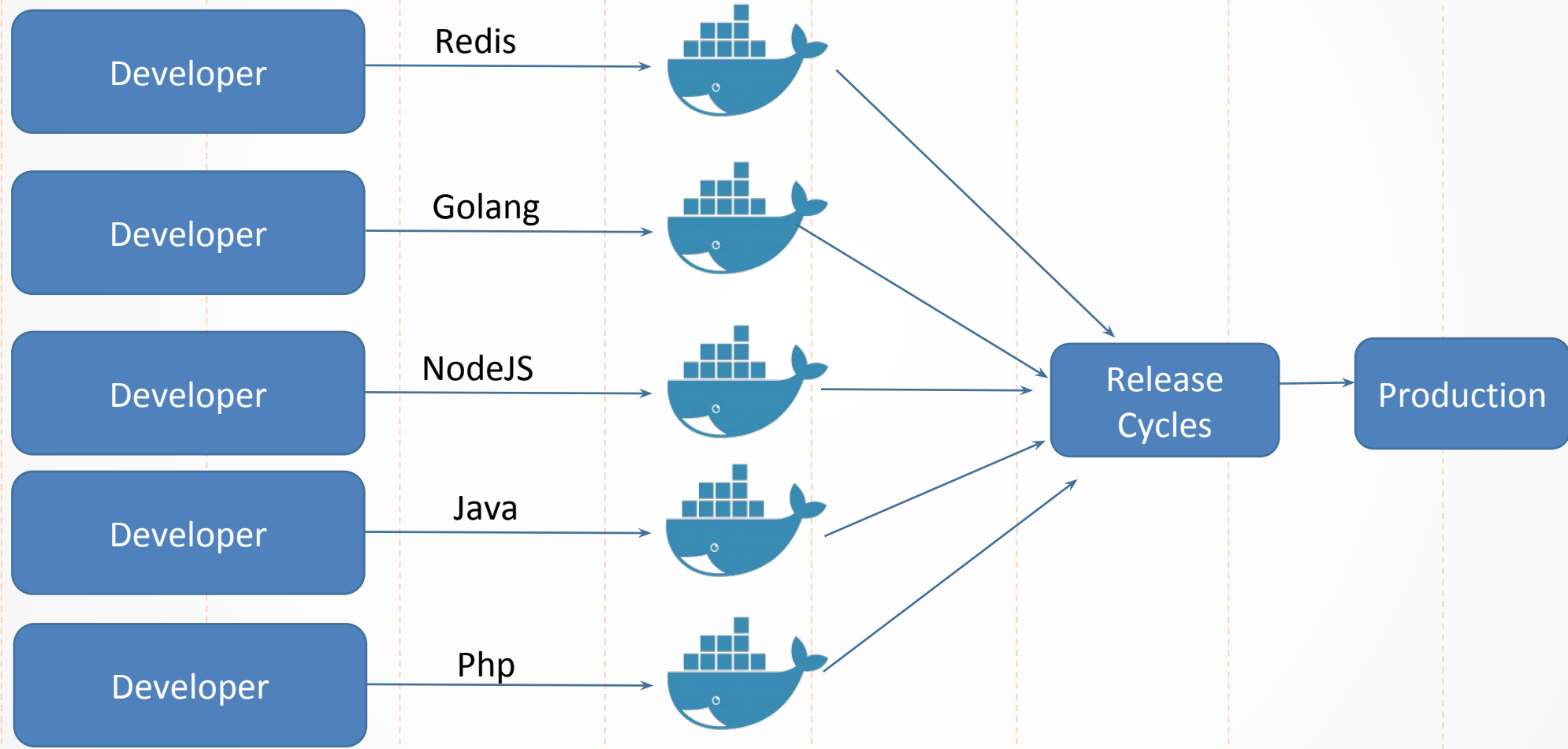
The Problem



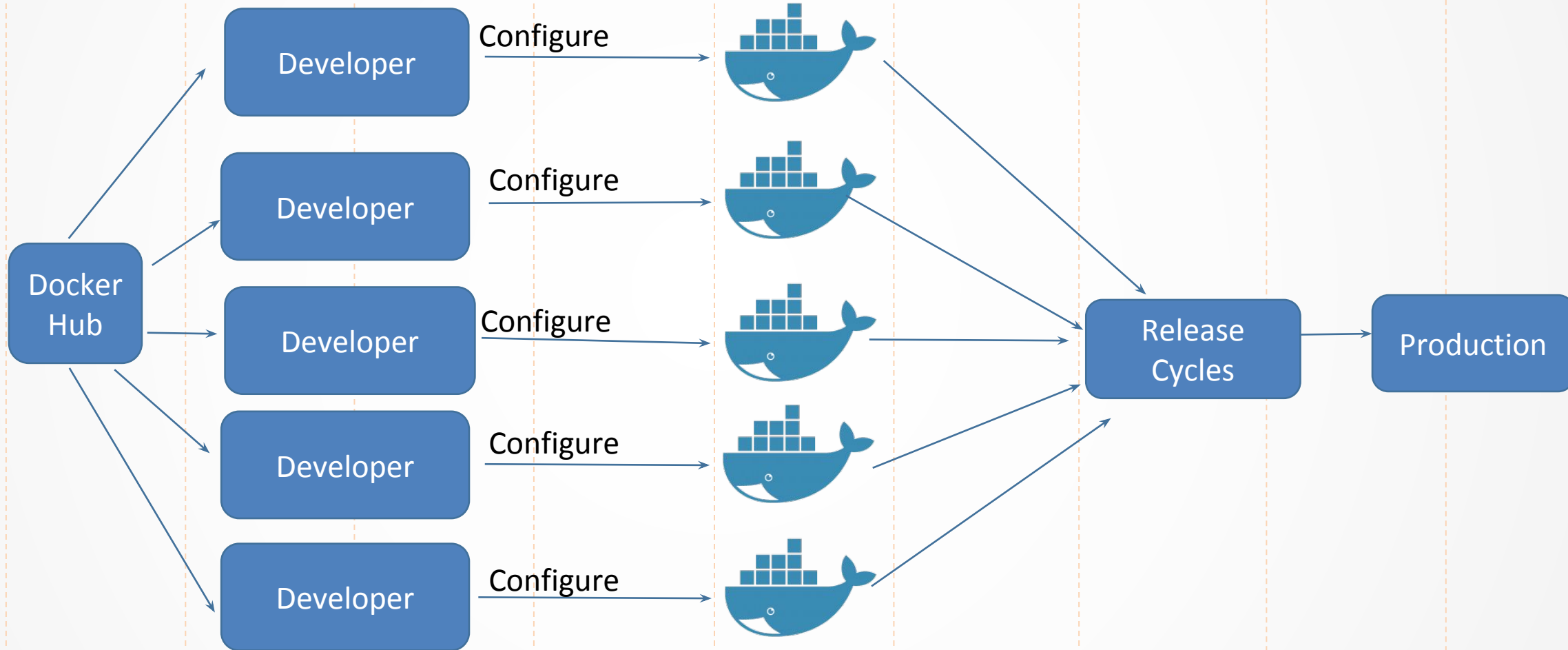
Using Monolithic



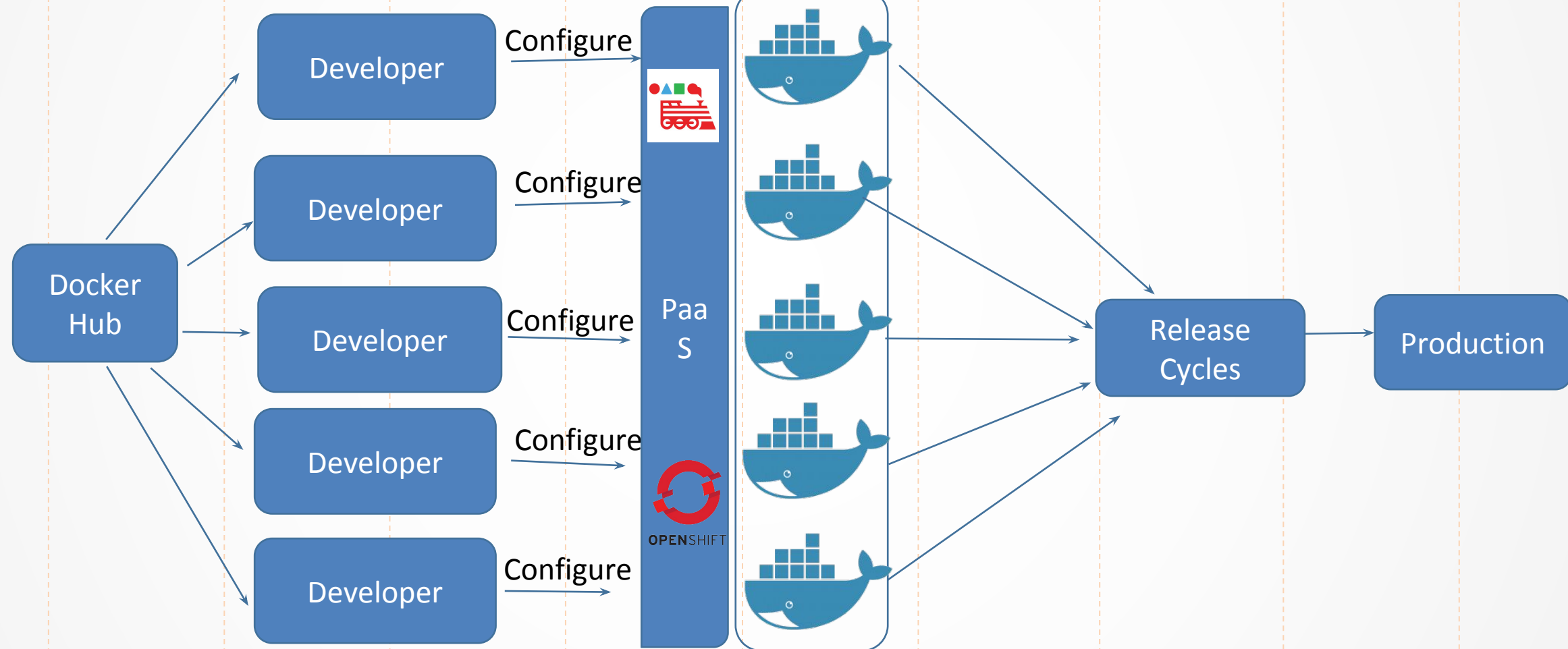
Using Microservices



Using Docker Hub with Microservices



Using PaaS

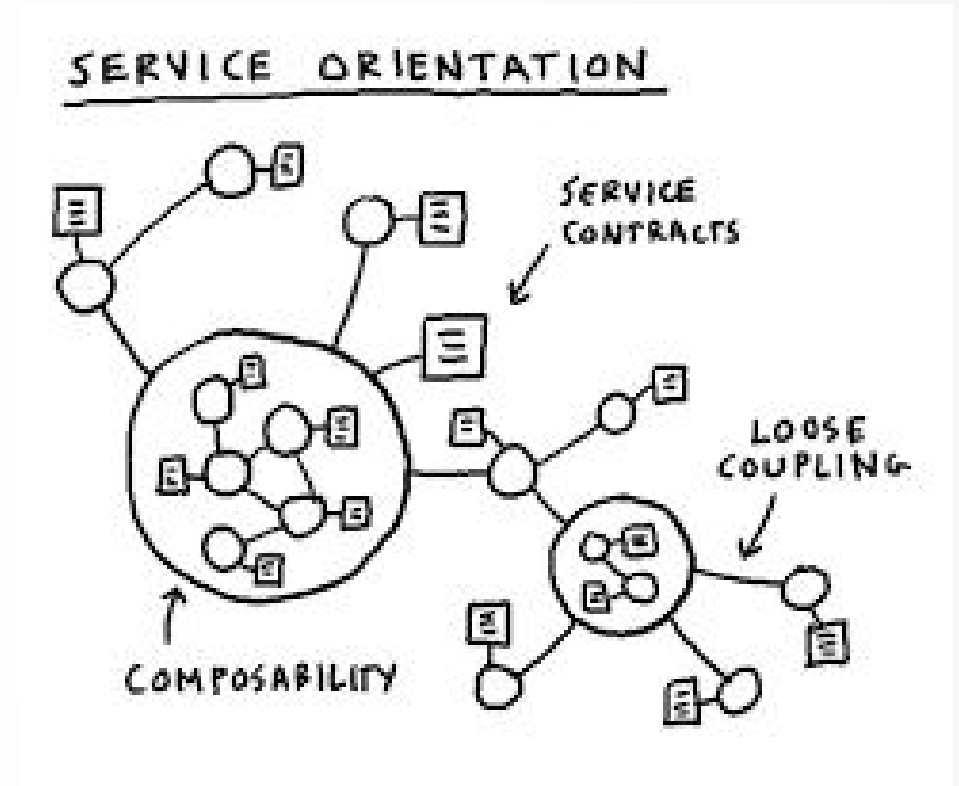


Principles Governing Micro services



Coupling

- Tight coupling leads to huge, monolithic systems that are difficult to maintain or improve upon
- If changing one module in a program requires changing another module, then coupling exists.



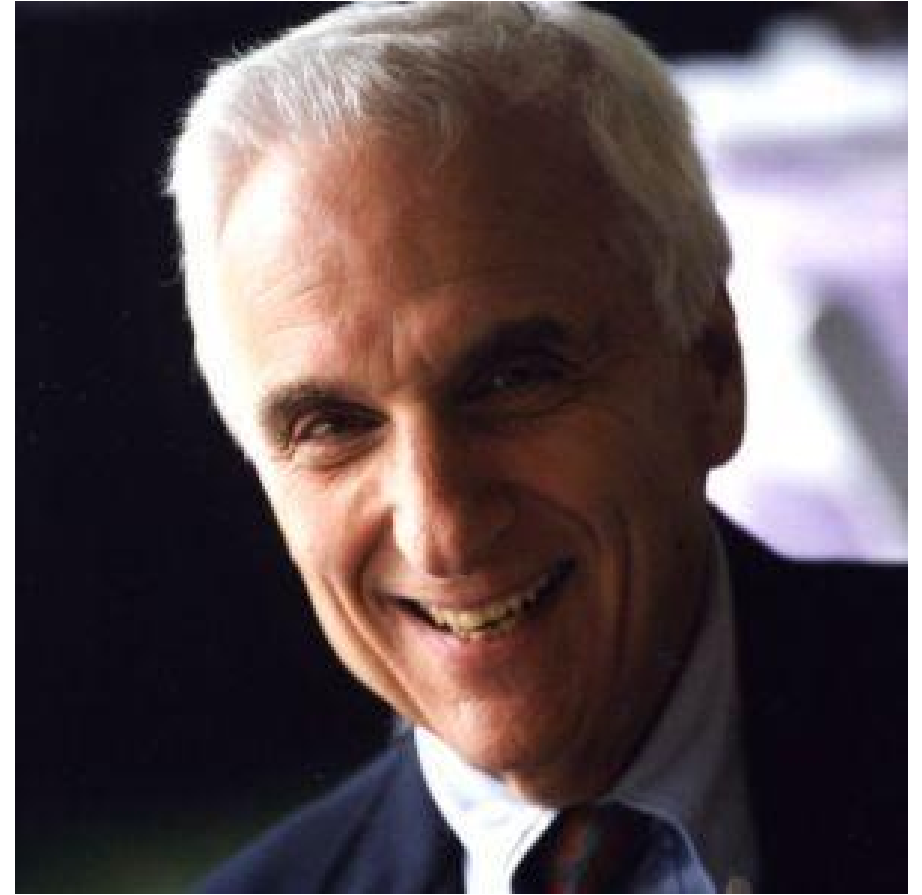
12 Factor app

As its name says, the Twelve-Factor App consists of 12 rules:

- **Codebase:** One codebase tracked in revision control, many deploys
- **Dependencies:** Explicitly declare and isolate dependencies
- **Config:** Store config in the environment
- **Backing services:** Treat backing services as attached resources
- **Build, release, run:** Strictly separate build and run stages
- **Processes:** Execute the app as one or more stateless processes
- **Port binding:** Export services via port binding
- **Concurrency:** Scale out via the process model
- **Disposability:** Maximize robustness with fast startup and graceful shutdown
- **Dev/prod parity:** Keep development, staging, and production as similar as possible
- **Logs:** Treat logs as event streams
- **Admin processes:** Run admin/management tasks as one-off processes

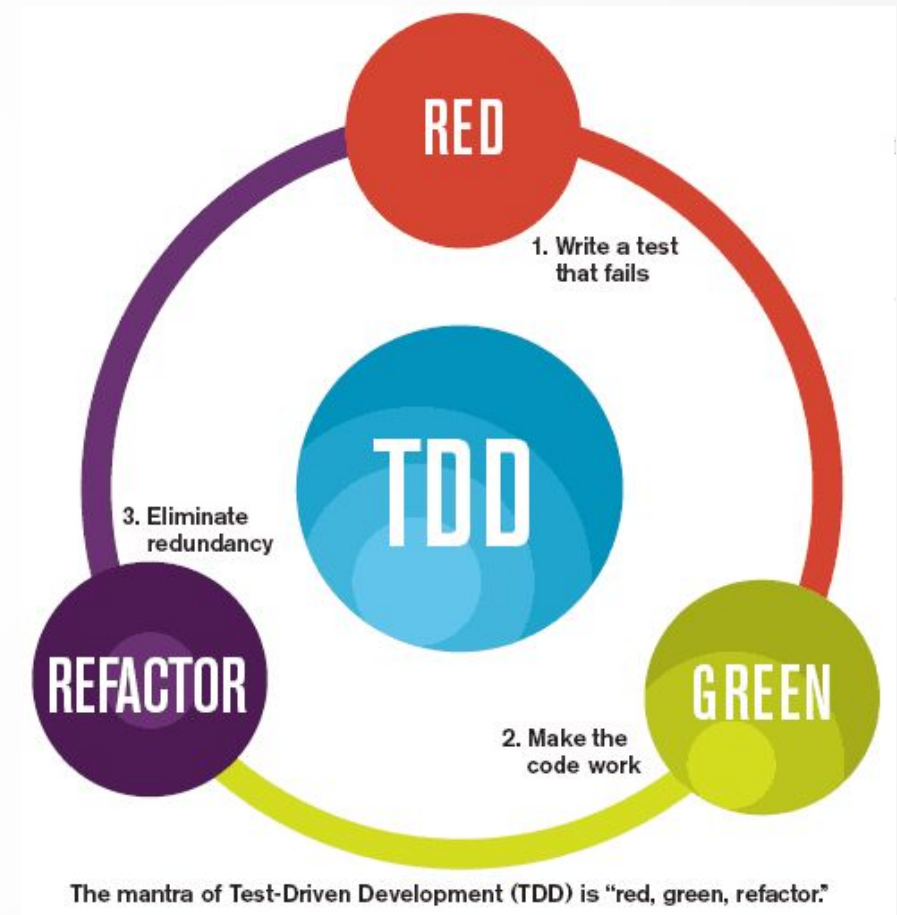
Conway's Law

"Any organization that designs a system ... will inevitably produce a design whose structure is a copy of the organization's communication structure."



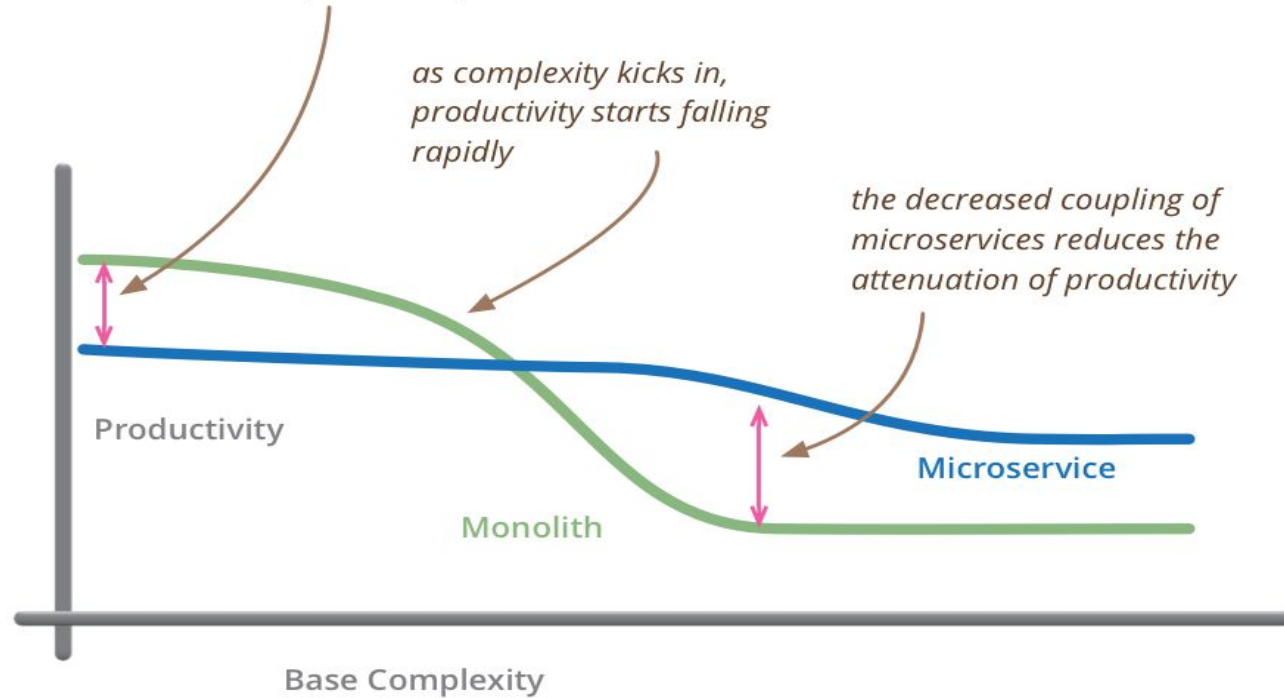
Refactoring

- Process to change the existing code without changing its external behavior.
- Refactoring improves nonfunctional attributes of the software.



Comparison of performance

for less-complex systems, the extra baggage required to manage microservices reduces productivity



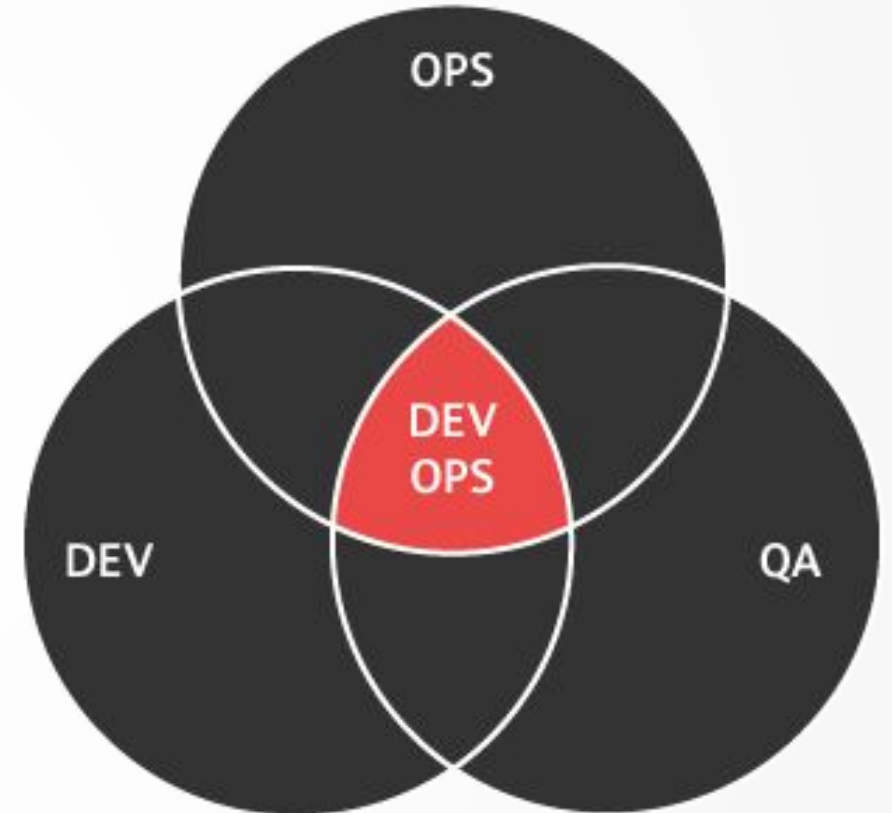
but remember the skill of the team will outweigh any monolith/microservice choice

But What about DevOps ?



DevOps

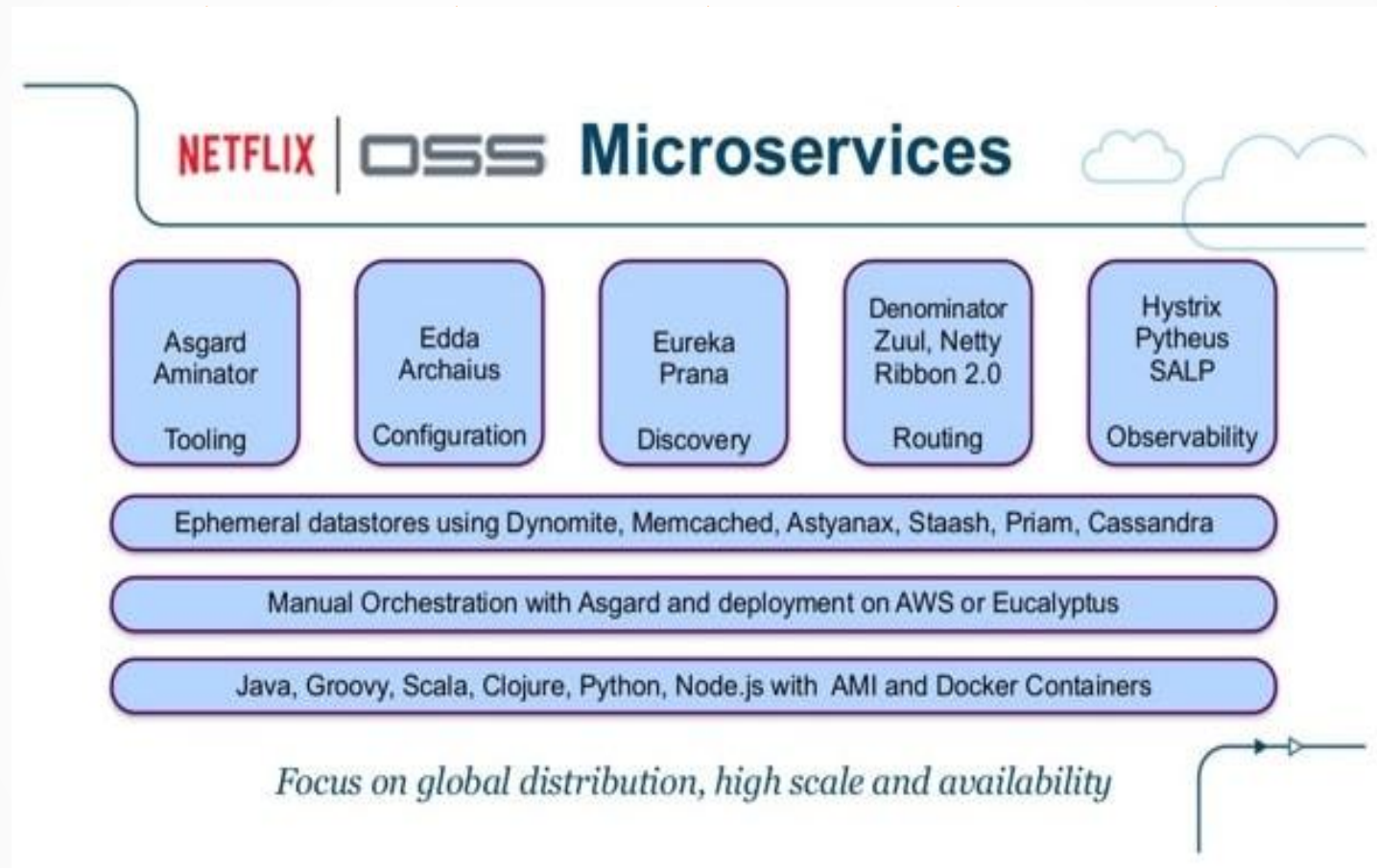
- DevOps is a company culture where the Developers movement or practice emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals.
- It helps in automation and making delivery fast.



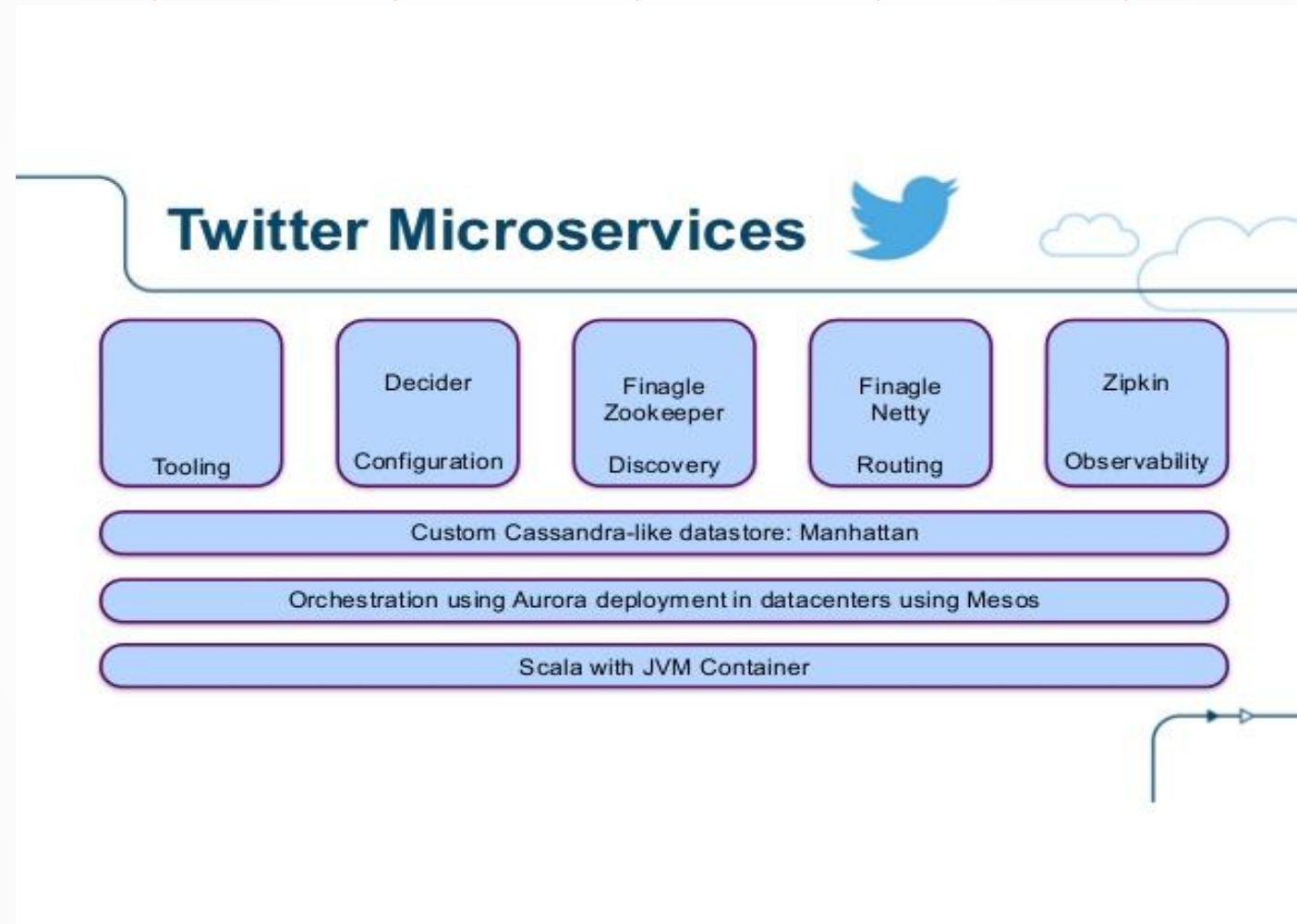
Case Studies



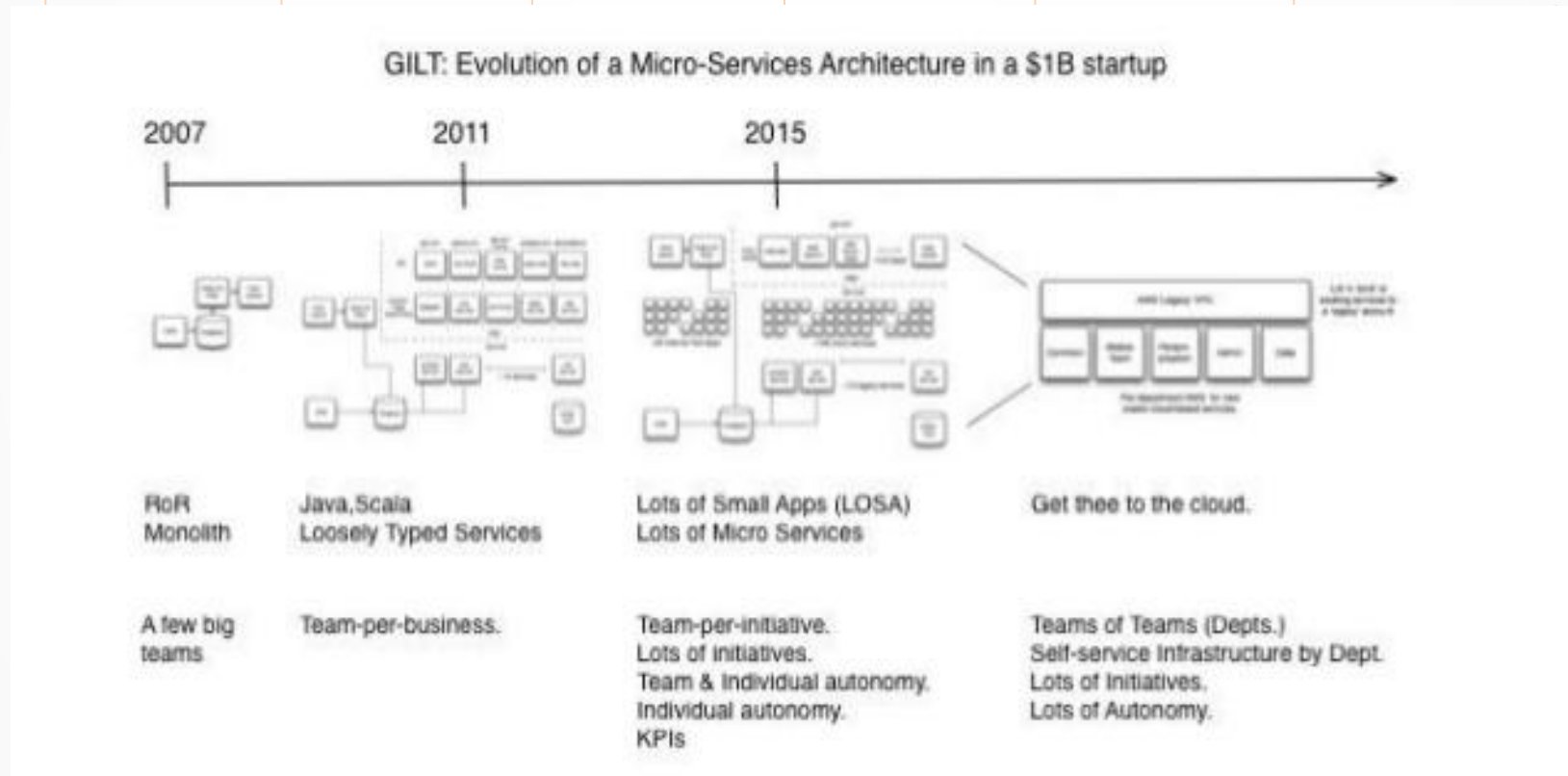
How Netflix does it?



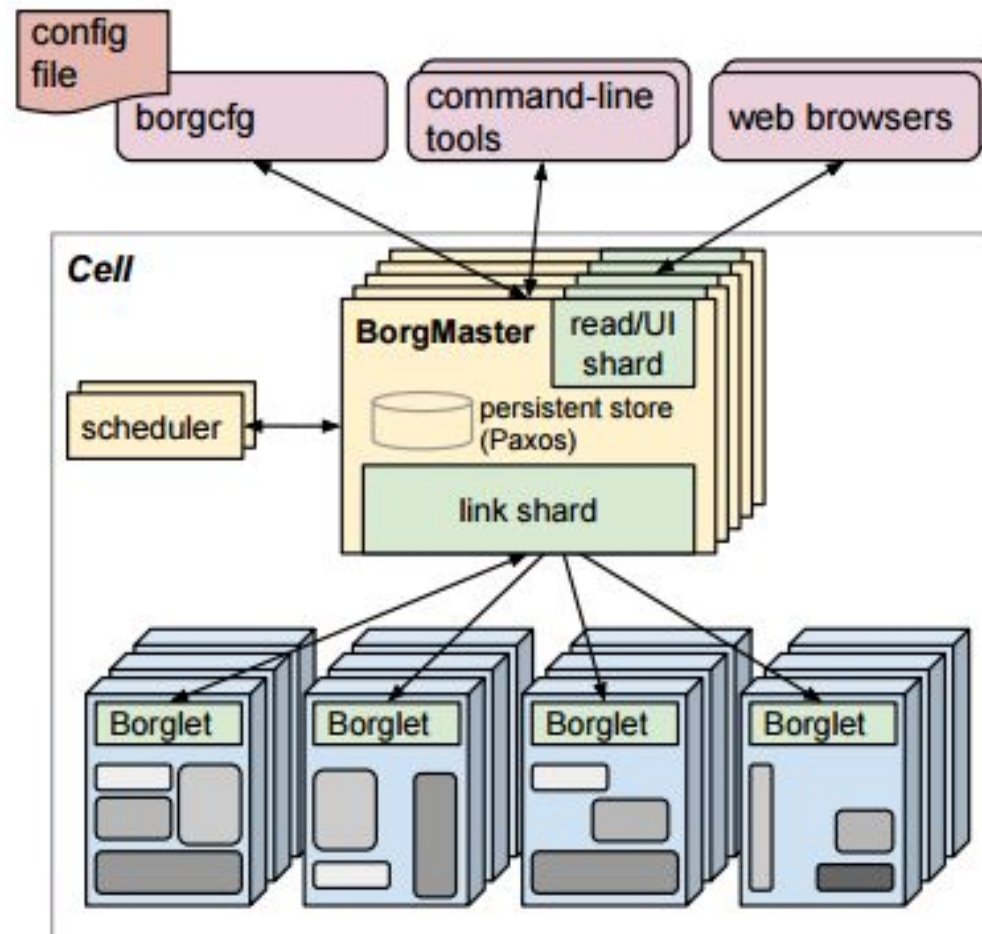
How Twitter do it?



How Gilt does it?



How does Google does it?



Any Questions?



Thank You

May you have an awesome day
ahead !

XP CONFERENCE INDIA 2016