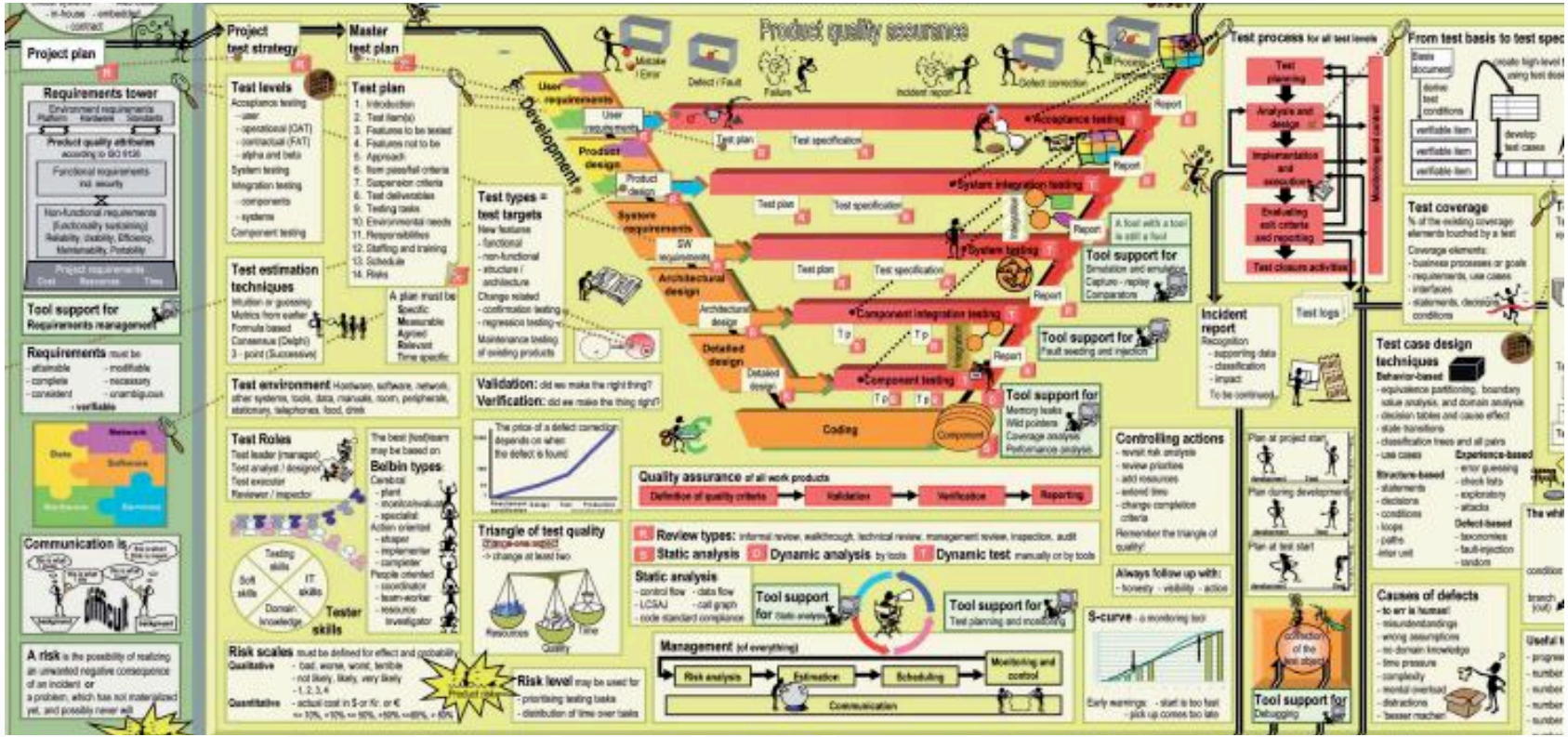
We Have A Need For Speed

Agile Meant Speed

….but we couldn’t deploy fast enough

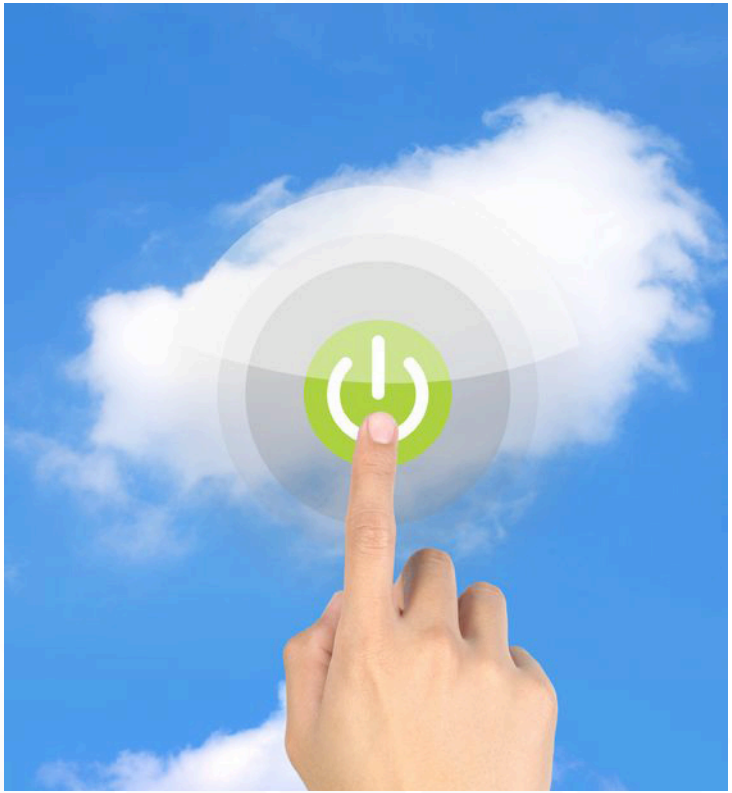
Now We Have DevOps

….but getting infrastructure takes too long



The Cloud Made Infrastructure Easy

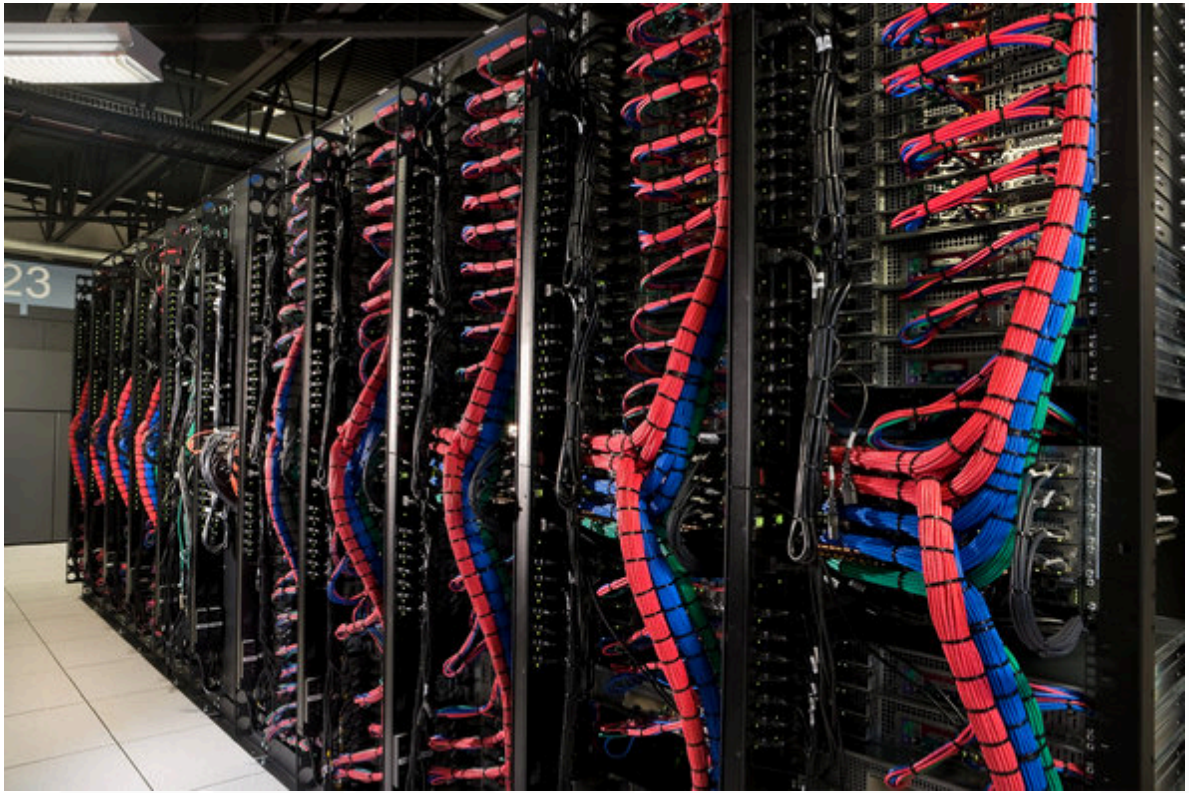
…but our apps were not built for the cloud



New Challenges In Today's World

- As developers, we are facing new challenges
- Speed
- Flexibility
- Multiple platforms
- Unprecedented scalability
- Infrastructure that is out of our control

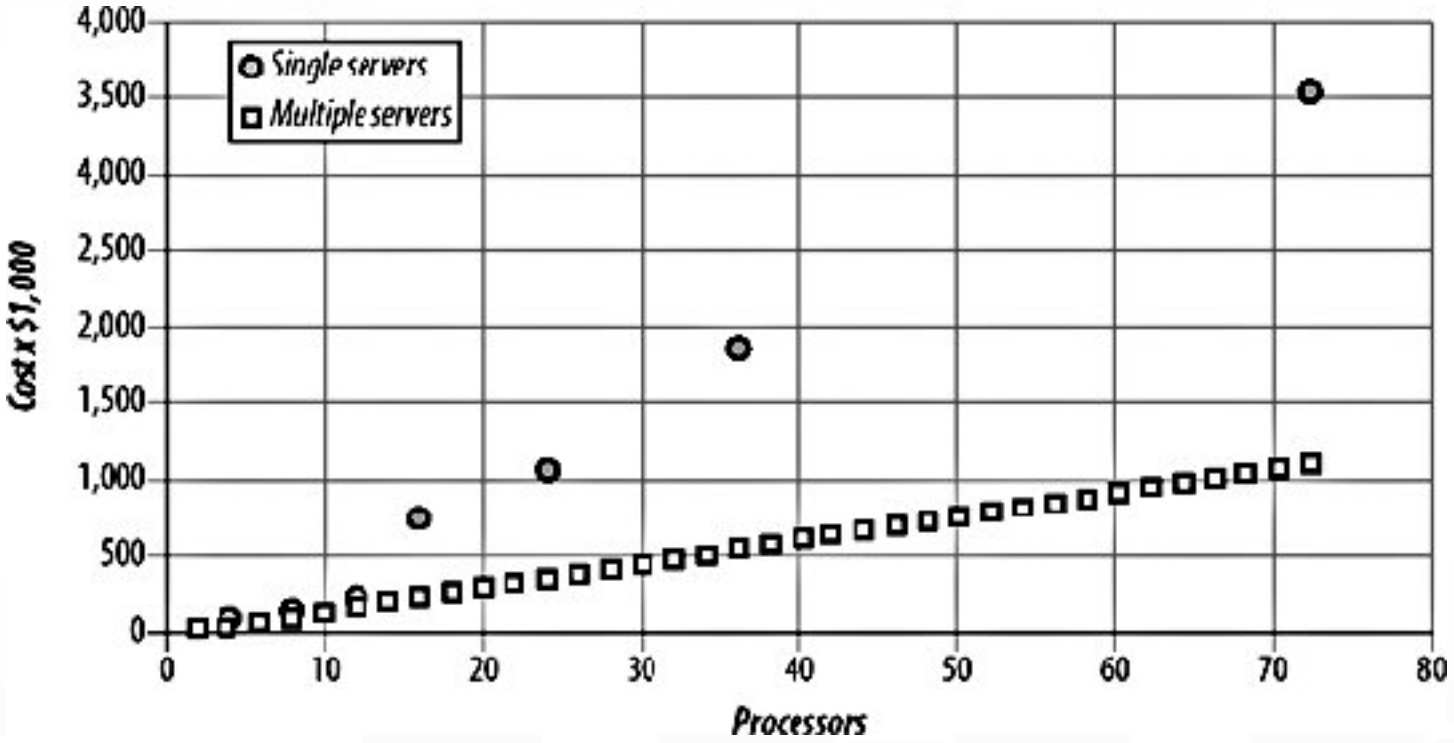
Infrastructure That Is Out Of Our Control



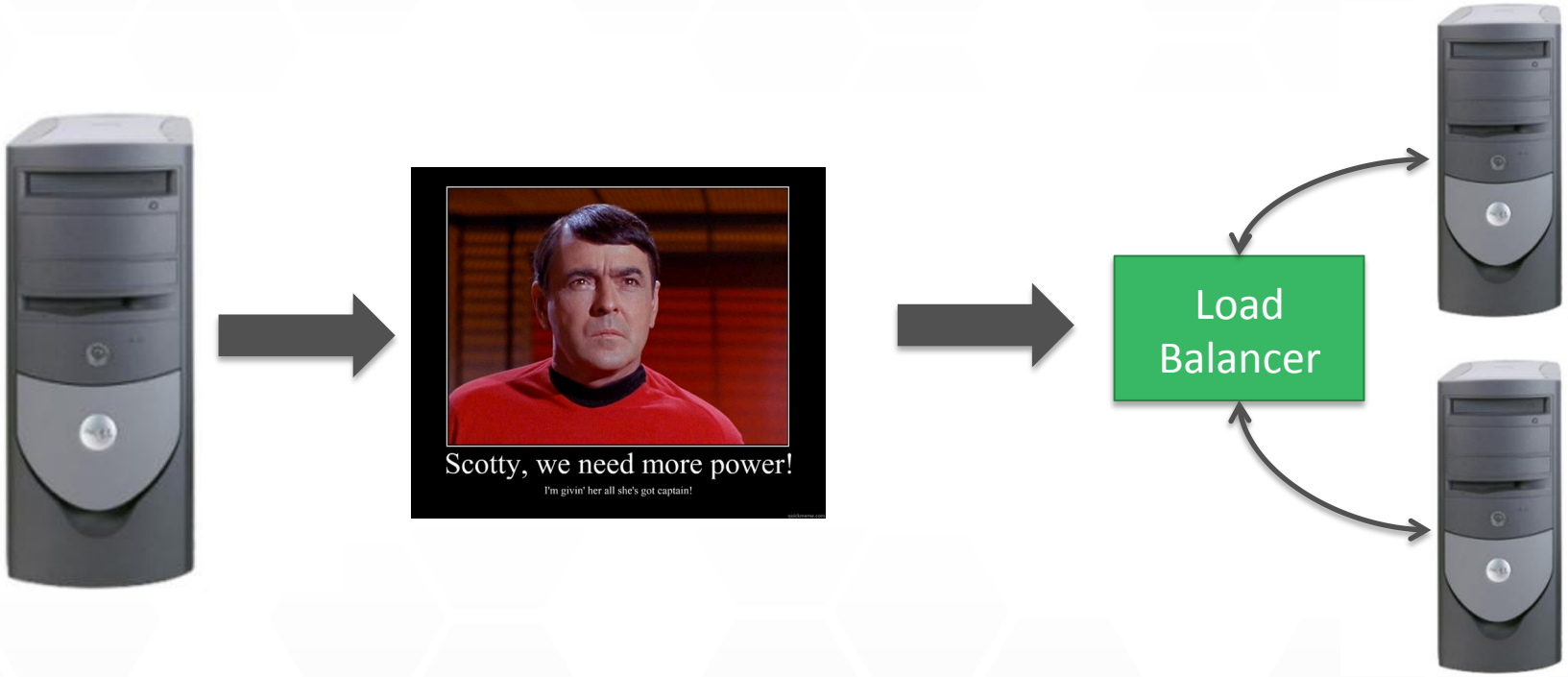
Unprecedented Scale

How Did We Solve This Problem Before?

Scaling Horizontally Is Cheaper



We End Up With Something Like This



Different Platforms

Requires A New Way Of Thinking

- Our old application designs just can't keep up
- Massive monolithic applications break down under these new challenges

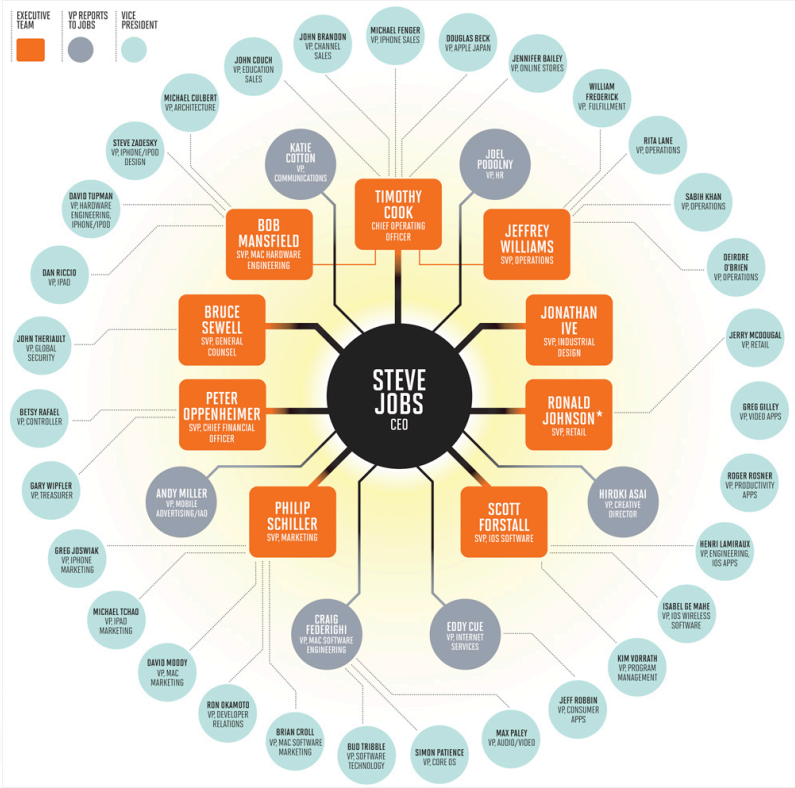
We Need A New Architecture

- Applications that are composed of smaller more focused “micro” apps

What is a Microservice?

"In short, the microservice 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." - Martin Fowler

It's Not All About The Code



Guidelines For Microservices

- Make each one focused – no real rule on size
  - “2 pizza teams”
- Each service should be treated like one app
  - Have its one SCM repo, own pipeline, etc
- Should not require the use of other services
- Service owners are responsible for the entire lifecycle
- Use lightweight protocols
- Use the right tool for the job

Guidelines For Microservices

- Each service should have its own datastore
- Failure will happen, design for it
- Automate everything

Isn't This Just SOA?

- Hot topic, they are certainly very similar and try to solve the same problem
- SOA was always considered very heavy
  - ESBs
  - Driven by vendors
  - SOA allowed for monoliths
  - SOA was more about reuse than team organization and application architecture

Benefits Of Microservices

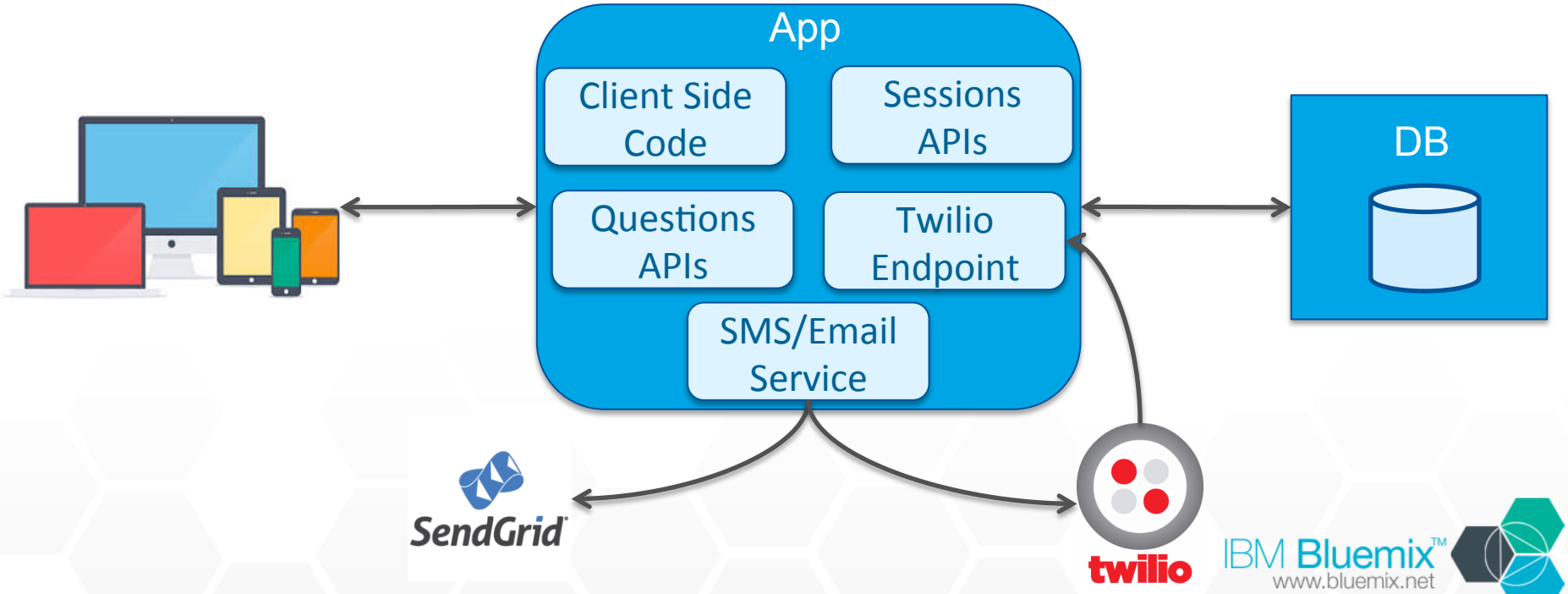
- Code for each service is easier to understand
- Small code base means more speed
- Each service can be deployed independently
- Failures do not bring down the whole app
- Technology flexibility
- Easier to evolve
- Well defined APIs

Cons Of Microservices

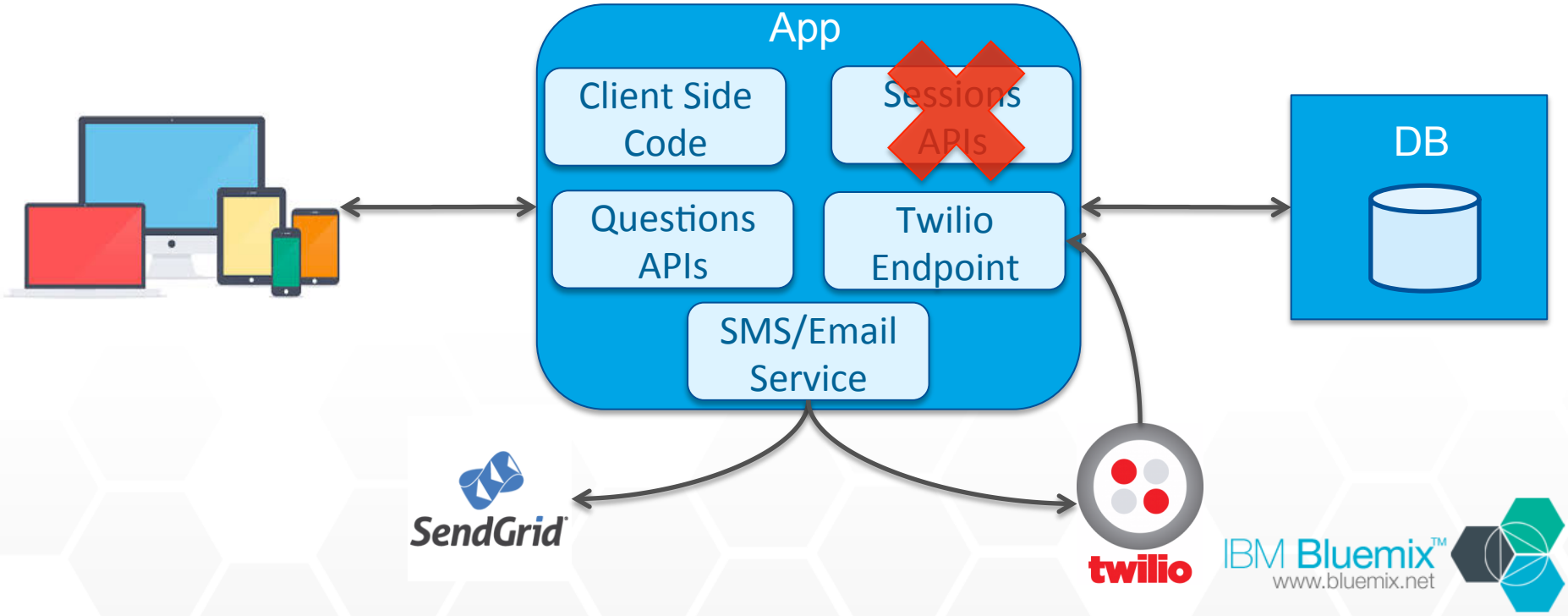
- Complexity
  - You have to manage multiple apps
  - Deploying 1 feature may involve coordination between multiple services
- Performance – multiple service calls
- When to switch to microservices
- Dev tools focus on monolithic apps
- Duplication of code
- End To End Testing

DEMO

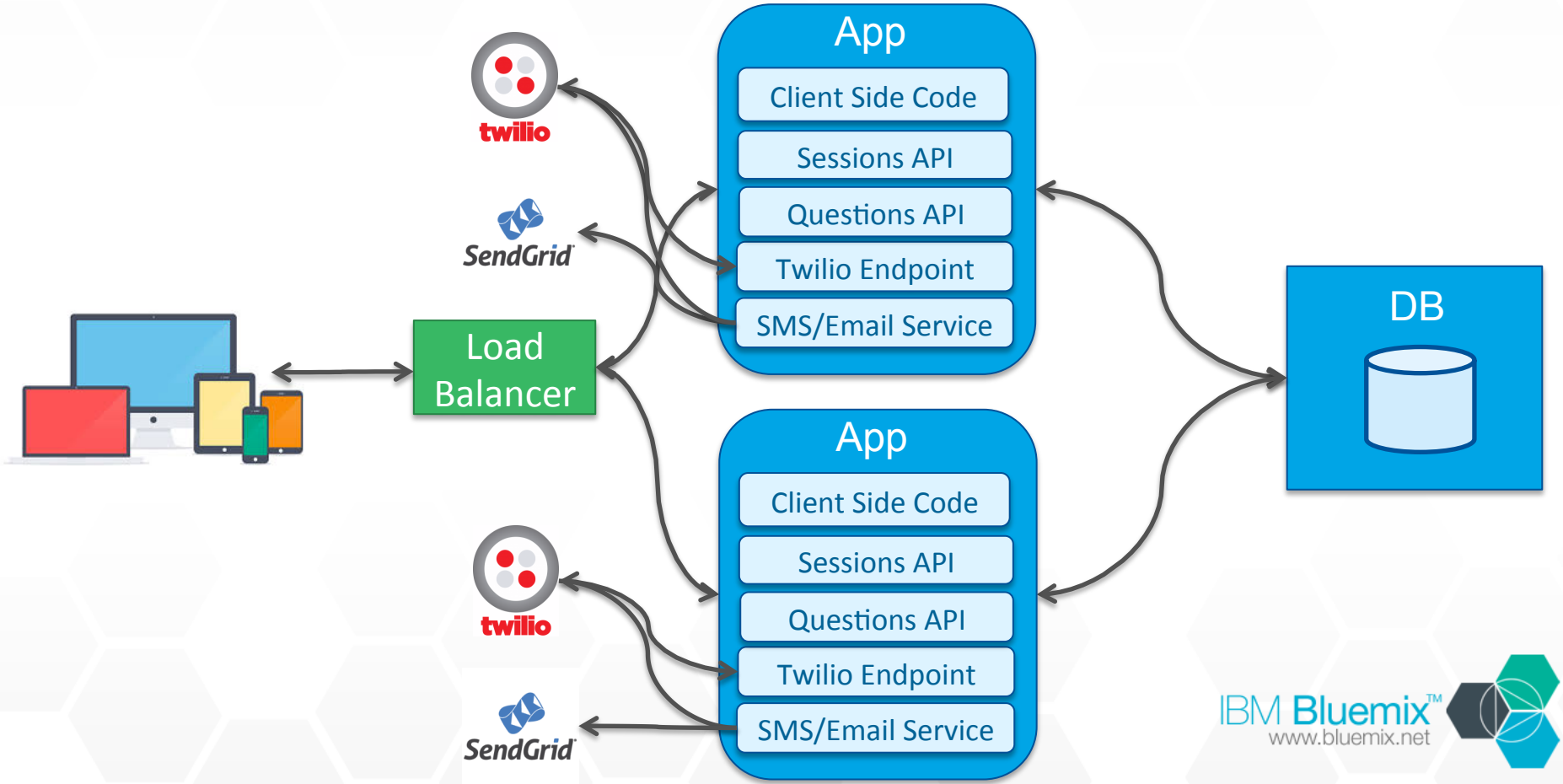
Monolith Architecture



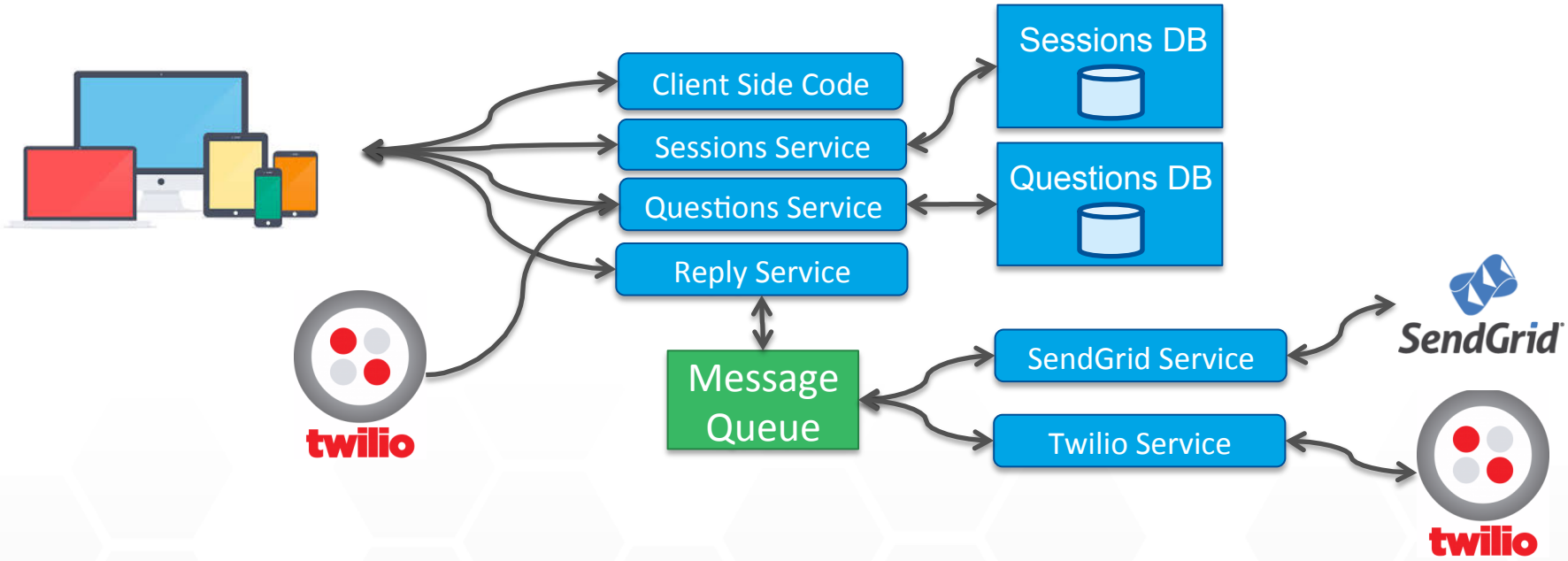
Failures In A Monolith



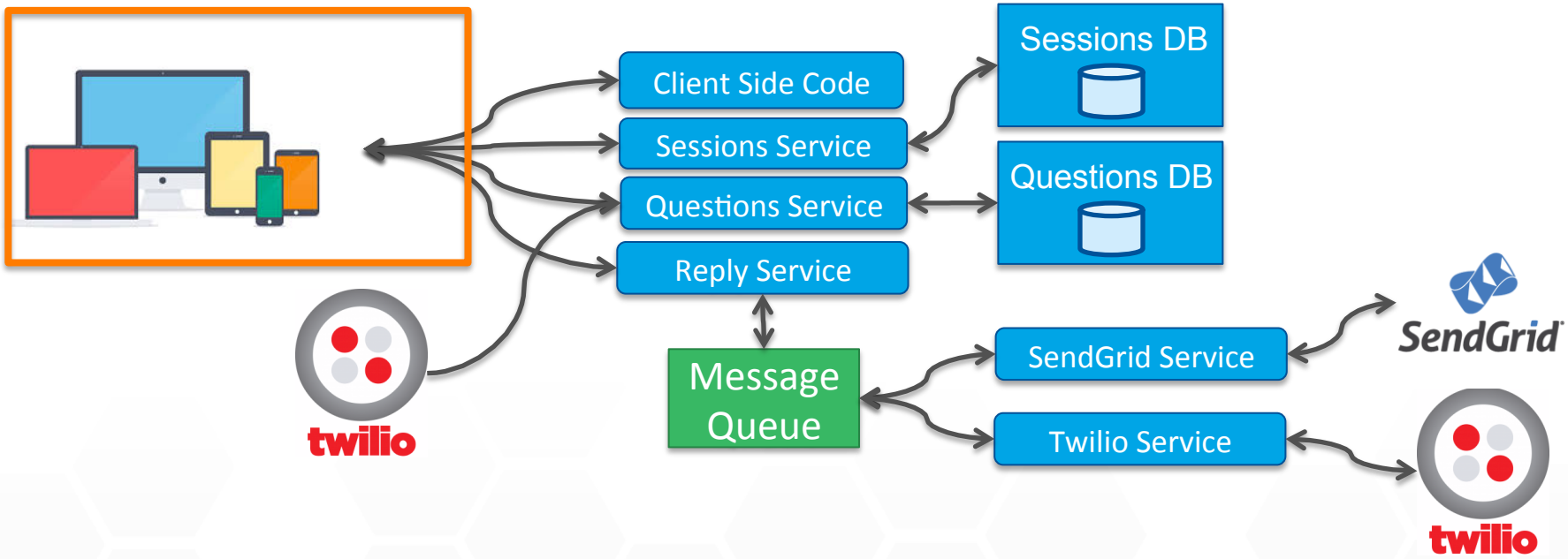
Scaled Monolith



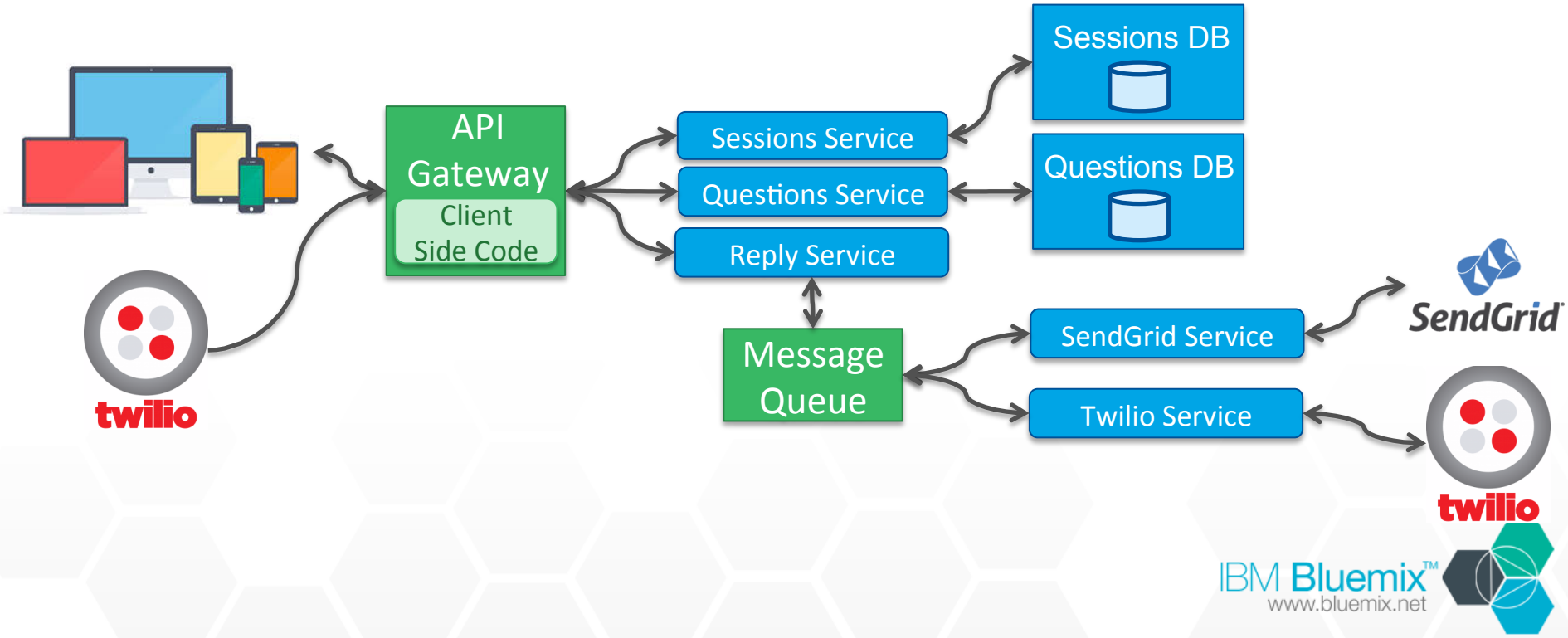
Microservices Architecture



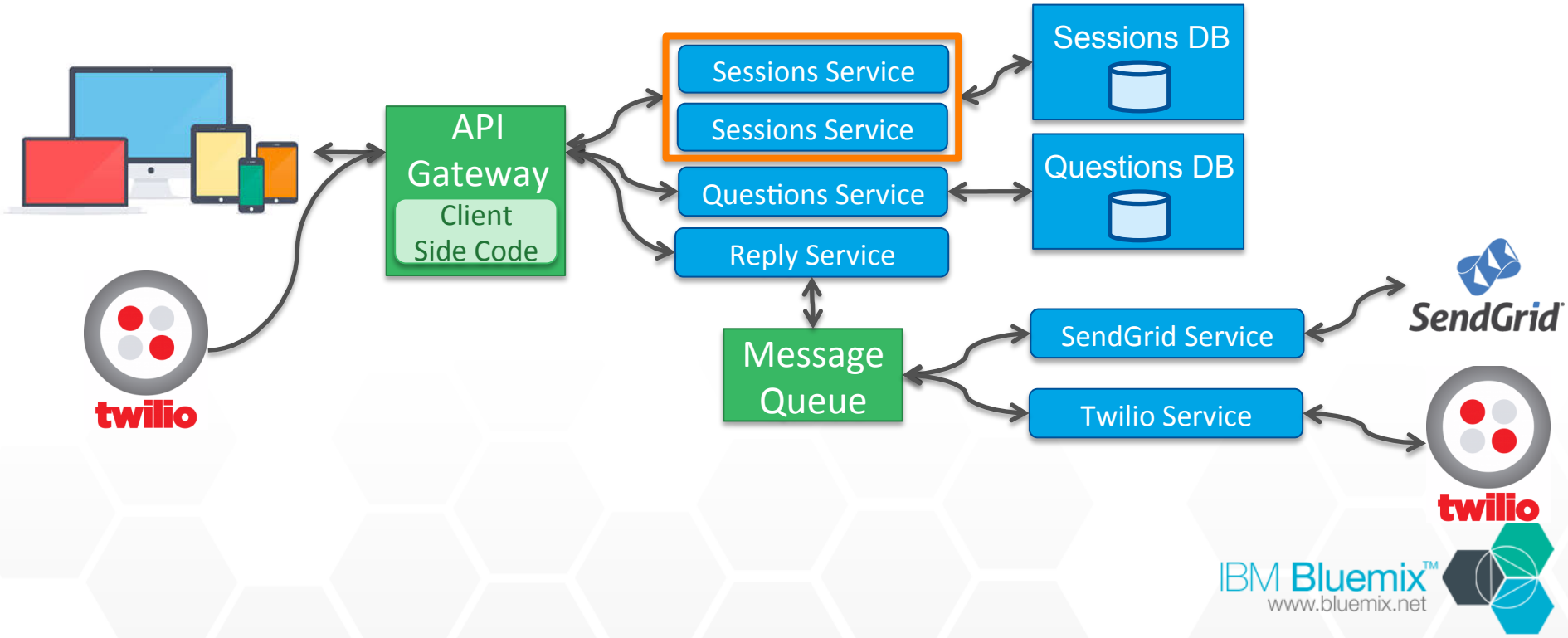
Client Fragility



API Gateways



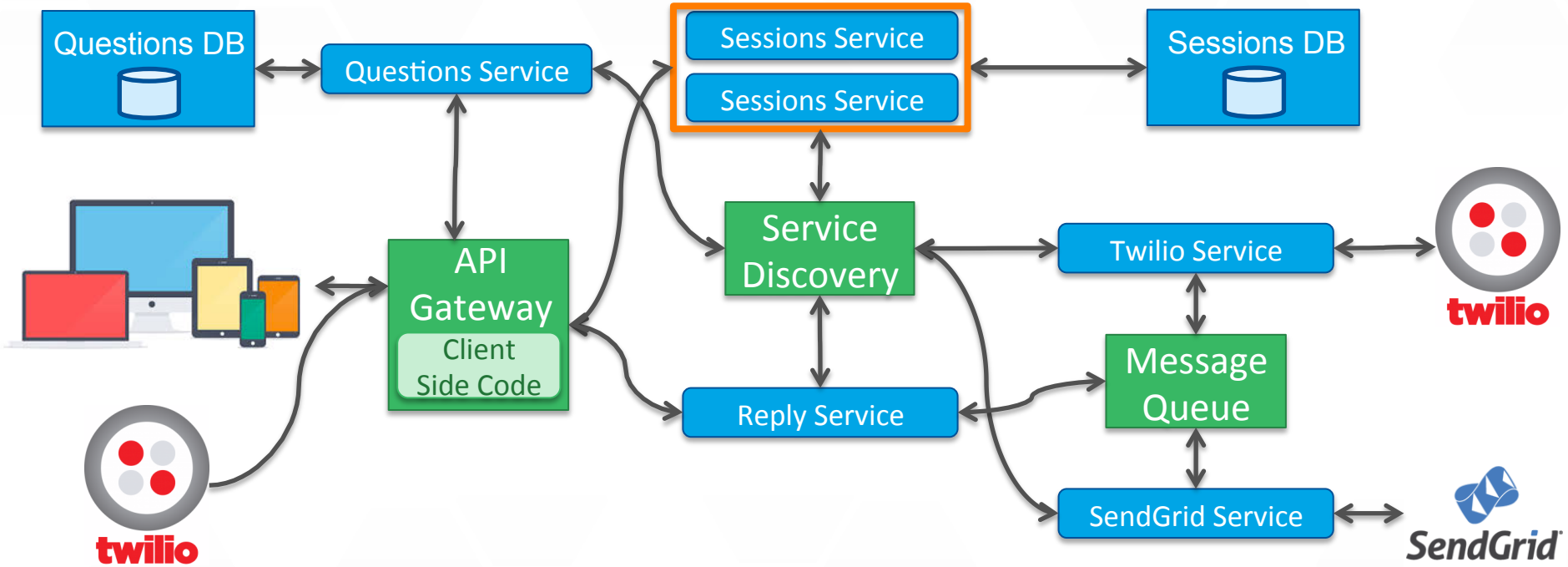
Scaling A Service



APIs

- Services communicate with each other and clients via a well established API
- REST + JSON is a very common choice
- Message queues can be used for asynchronous support
- Use an API gateway!

Service Discovery



Service Discovery

- Eureka – Neflix OSS Component
- Zookeeper – Apache Project
- Consul

Circuit Breakers

DEMO

Containers/Docker

- Is a great deployment and isolation tool for your microservices
- Allows your microservices to run on various clouds
- Need to use some kind of container management tool to orchestrate, monitor, and scale your microservices

Platform-as-a-Service

- Deploying a large scale distributed set of apps requires a lot of effort (even if you are using Docker)
- A platform-as-a-service can make it somewhat easier
- Built in monitoring, recovery, and scaling can really go a long way
- Deployments are also much easier because you just provide the code
- Take a look at any Cloud Foundry based PaaS

Deploying Microservices With Cloud Foundry

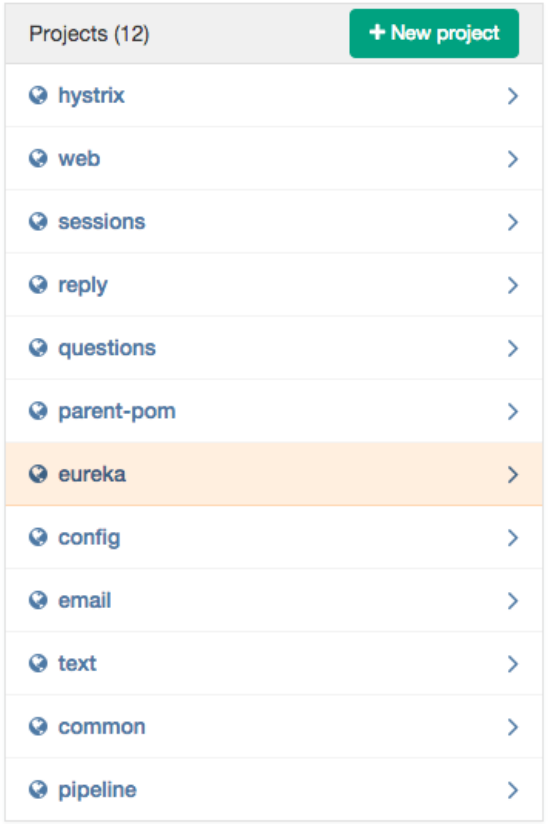
- Easy as *cf push appname –p <jar file>*
  - Or use Cloud Foundry Maven or Gradle plugin
- You can use blue/green deployments to make sure you have no downtime during deploys
- Each microservice can be used as a Cloud Foundry service to facilitate service discovery
- Built-in load balancer can help with dynamic scaling
- Everything can be automated so it all happens on commit

Netflix

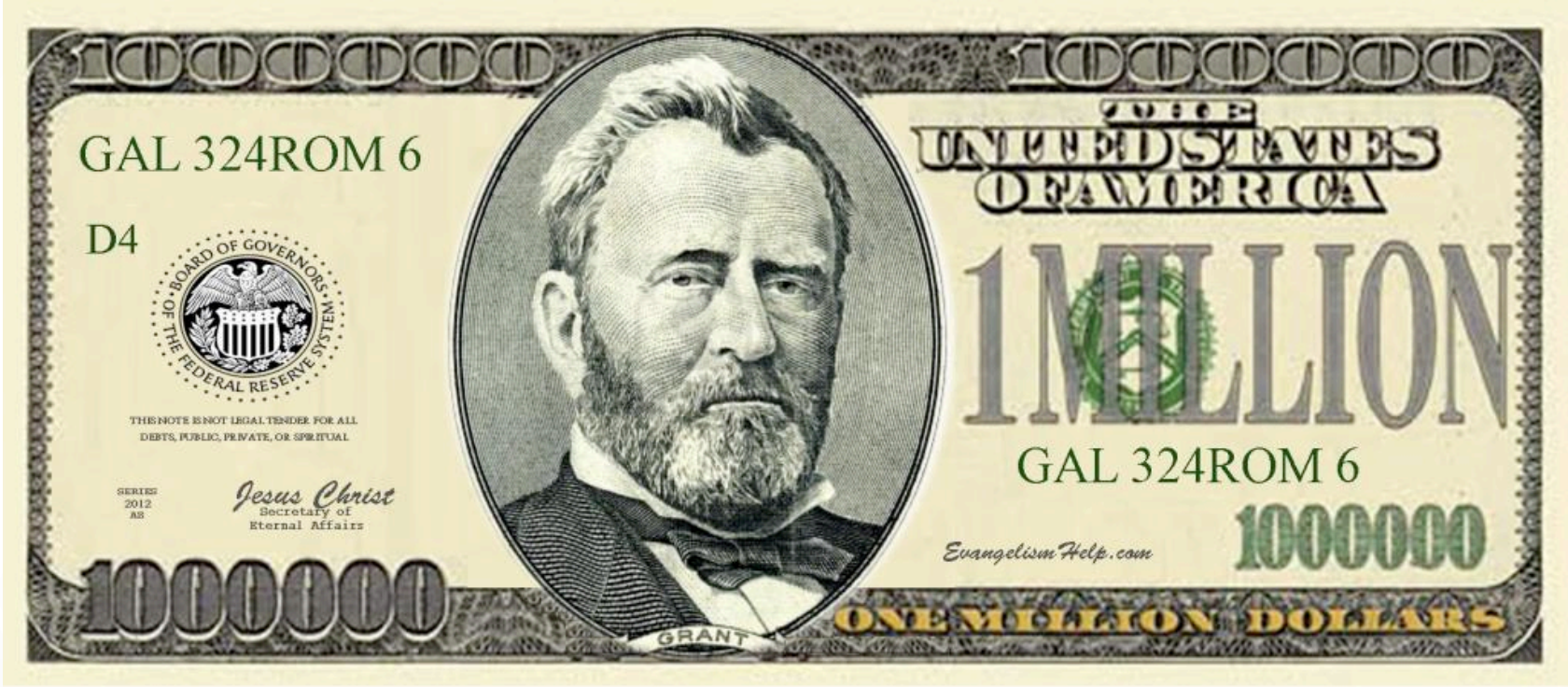
- The poster child for microservices
- They have a great set of open source libraries in the Neflix OSS project
- Service Discovery, Circuit Breakers, Service To Service Communication, and much much more…
- Check out Spring Cloud if you are a Spring user, very easy to integrate Netflix OSS into your apps

Microservices In SCM

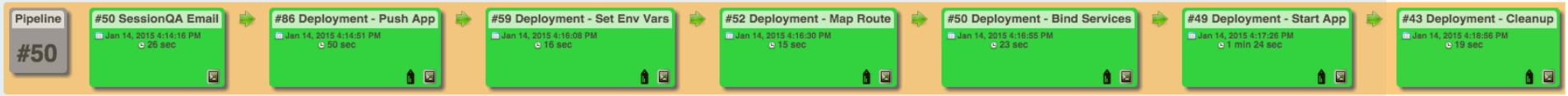
- Don't think you can still put all your code in a single repo
- Each service should have its own repo
- Each one can then be branched and evolve without effecting others



Transitioning To Microservices



Deployment Pipelines



- Commit code
- Do build and unit tests
- Deploy App
- Test App
- Use load balancer to direct users to new version of app
- Kill old version of app

Thank you.

IBM Bluemix™

www.bluemix.net