
Pivotal
Cloud Native Design &
Development
.NET

Defining Cloud-Native *(AKA: 'Agility, Agility, Agility')*

Cloud Native is not about **where** you run apps, but **how**

- **'Twelve-Factor' Architecture**
- **Microservices Designs**
- **"Antifragility"**
- **Containers**
- **The Application "Dial Tone"**
- **Cultural Shift from Silo IT to DevOps**

Pivotal Cloud Foundry

Cloud Native App Development

Twelve-Factor Apps

I. Codebase
One codebase tracked in SCM, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Configuration
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, Release, Run
Strictly separate build and run stages

VI. Processes
Execute app as stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep dev, staging, prod as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin / mgmt tasks as one-off processes

Architectural and development practices – http://12factor.net

Pivotal

Pivotal Cloud Foundry

Microservices

Microservice: Definition

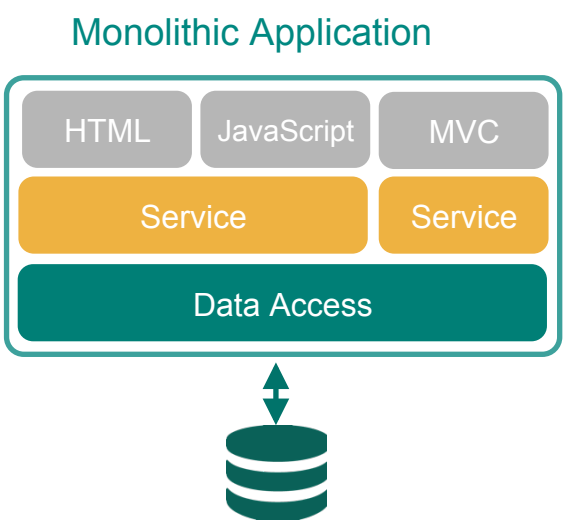
If every service has to be updated in concert,
it's not loosely coupled!

"Loosely coupled service oriented architecture with bounded contexts"

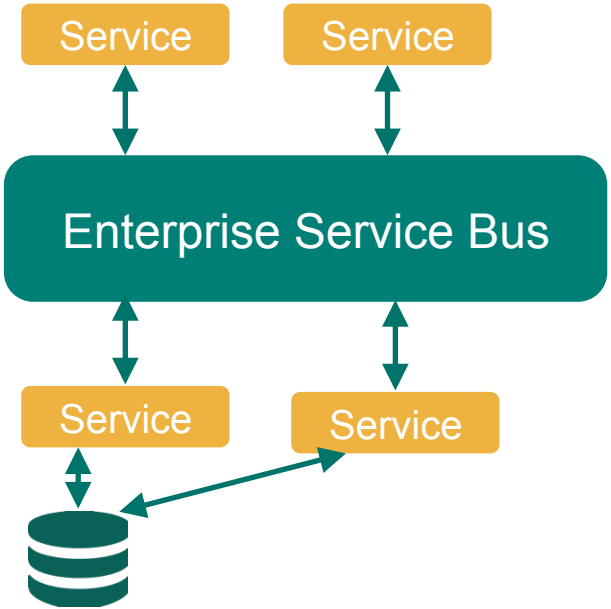
If you have to know about surrounding services you don't have a bounded context.

- Adrian Cockcroft

Microservices are NOT



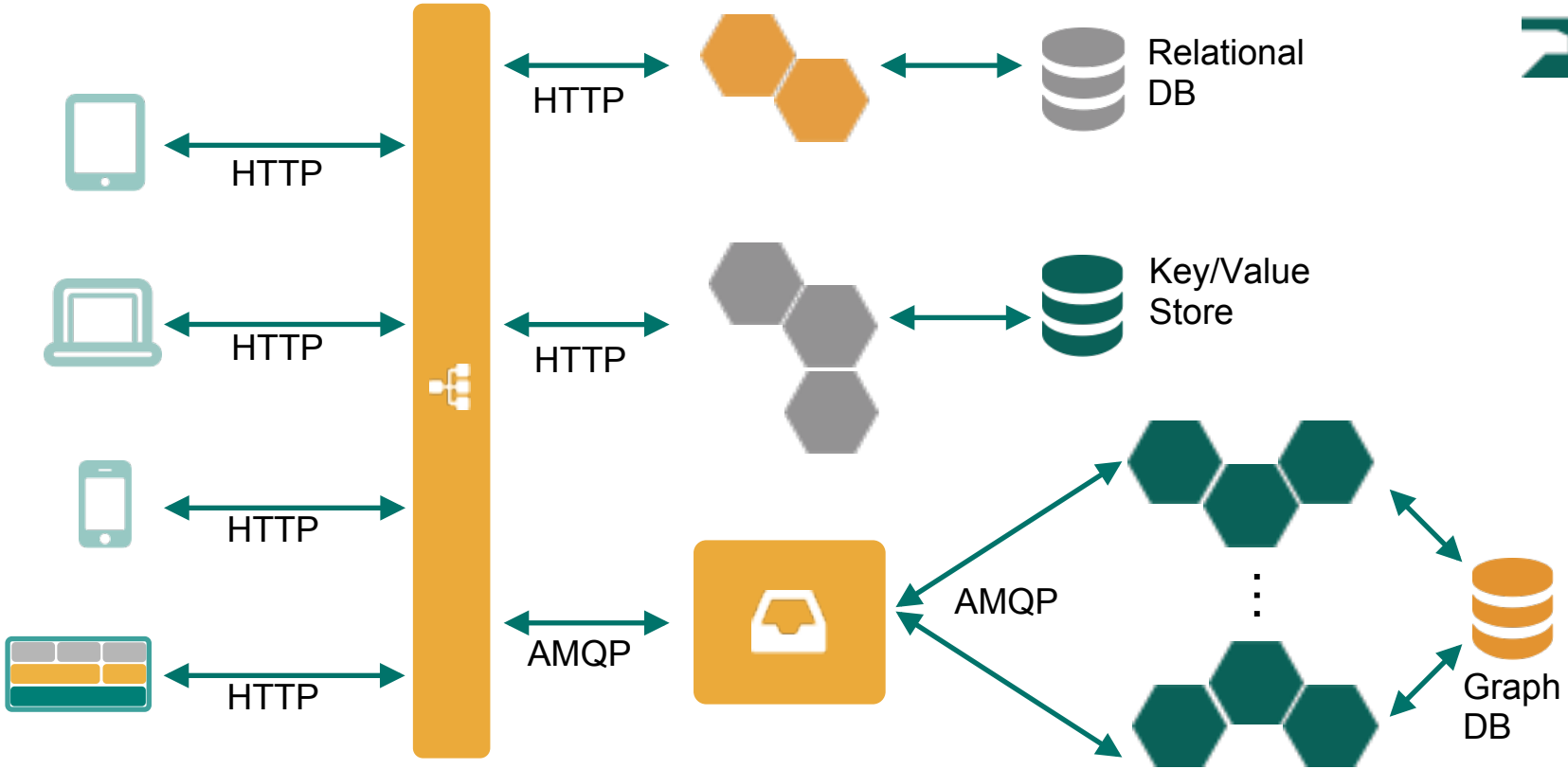
OR



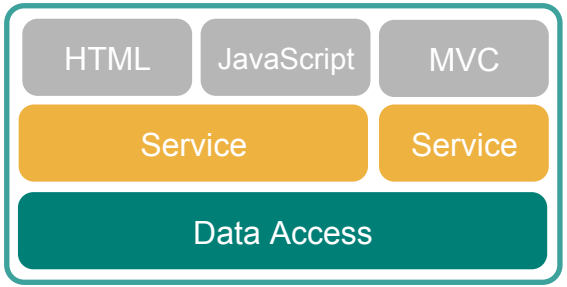
Tightly Coupled

Centralized

Microservice Architecture



Monoliths: challenges



- Traditional monolithic design patterns are not appropriate for the cloud.
- Monoliths couple change cycles together.
- Monoliths services can’t be scaled independently.
- Difficult coordination: too many developers in one code base.
- Developers struggle to understand a large codebase.
- Long term commitment to the tech stack.

Microservice: benefits

- Change cycles are decoupled: Enabling frequent deploys
- Allow for efficient and independent scaling
- Developers learn a smaller codebase faster
- Better coordination and scaling of development: Fewer developers in each code base
- Eliminate long-term commitment to technical stack

Pivotal
Cloud
Foundry

.NET Development in the Cloud

Writing 12 Factor .NET Apps

- Use an external session/cache provider
  - RDBMS, Redis or GemFire
- Avoid writing log files and use stdout/stderr
- Use environment variables for environment config
- Use OAuth instead of Integrated Windows Auth
- Avoid tight integration with on-server dependencies

Integrated Windows Authentication (IWA)

- Compares user credentials against windows cell users
  - Don’t create snowflakes
  - CF/Bosh should manage cells, not AD Group policies
- Routers and windows challenge/response auth do not get along
- Solution:
  - Forms authentication
  - ADFS
  - UAA Federated with AD/LDAP
  - OAuth2

Windows Services & Self-Hosted WCF

- Self-Hosting bootstraps web server
  - WCF terms: *server endpoint*
  - WCF creates server, *and* binds to port
  - **Fails**: AddressAccessDeniedException

*Self hosted Windows Communication Foundation applications are an anti-pattern: "self-hosted WCF services and windows services both count as .NET console apps with a start command, and those cannot bind to a port because of CAS restrictions. WCF services **must** be hosted inside an ASP.NET app and configured inside the web.config".*

More ASP.NET Antipatterns

- In-process session state
  - Solution: out-of-process (i.e. SQL Server, Redis)
- Storing or Retrieving Files on Local Disk
  - Even temp files. Cannot assume files will be there when app restarts!
  - Solution: S3-compatible blob store
- Environment-specific configuration in `web.config`.
  - Solution: externalized configuration via VCAP

ASP.NET Core vs ASP.NET 4.x

Deploying to Cloud Foundry

- **ASP.NET Core**
  - Develop on Win, Mac, or Linux
  - Run on Linux: with .NET Core stack
    - Use .NET Core buildpack
    - Deploy to `cflinuxfs2` stack.
  - Run on Windows: with either .NET Framework or Core stack
    - Use binary buildpack
    - Deploy to `windows2012R2` stack (Windows Cell on Diego)

- **ASP.NET 4.x**
  - Use binary buildpack
  - Deploy to `windows2012R2` stack (Windows Cell on Diego)
  - Develop on Windows


Pivotal
Thank You!
Pivotal