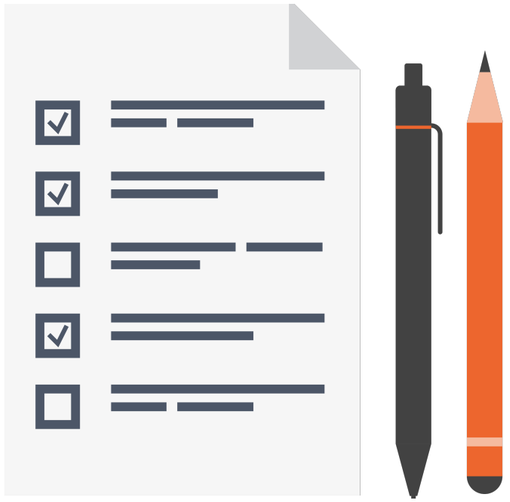
Moving Forwards with Microservices

Rag Dhiman

@RagDhiman

Module Overview



Brownfield Microservices

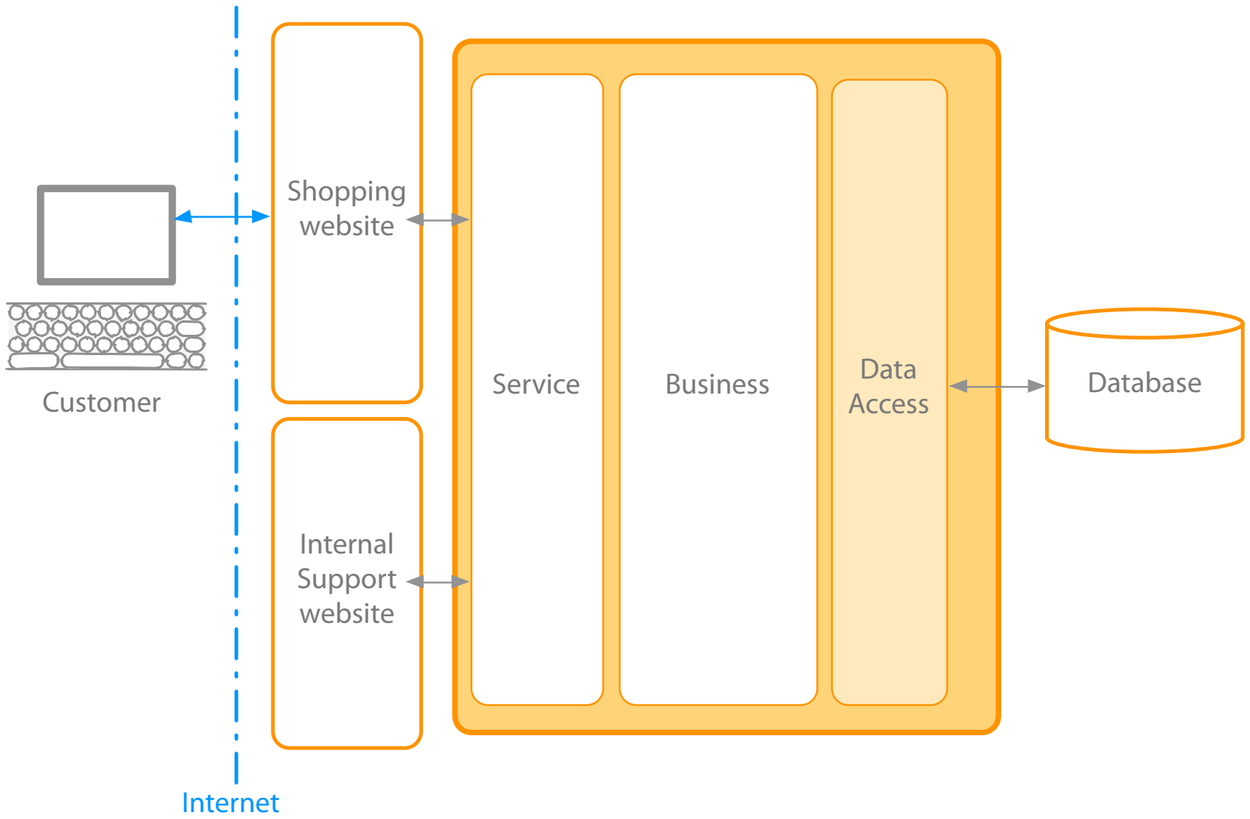
Greenfield Microservices

Microservices Provisos

Brownfield Microservices

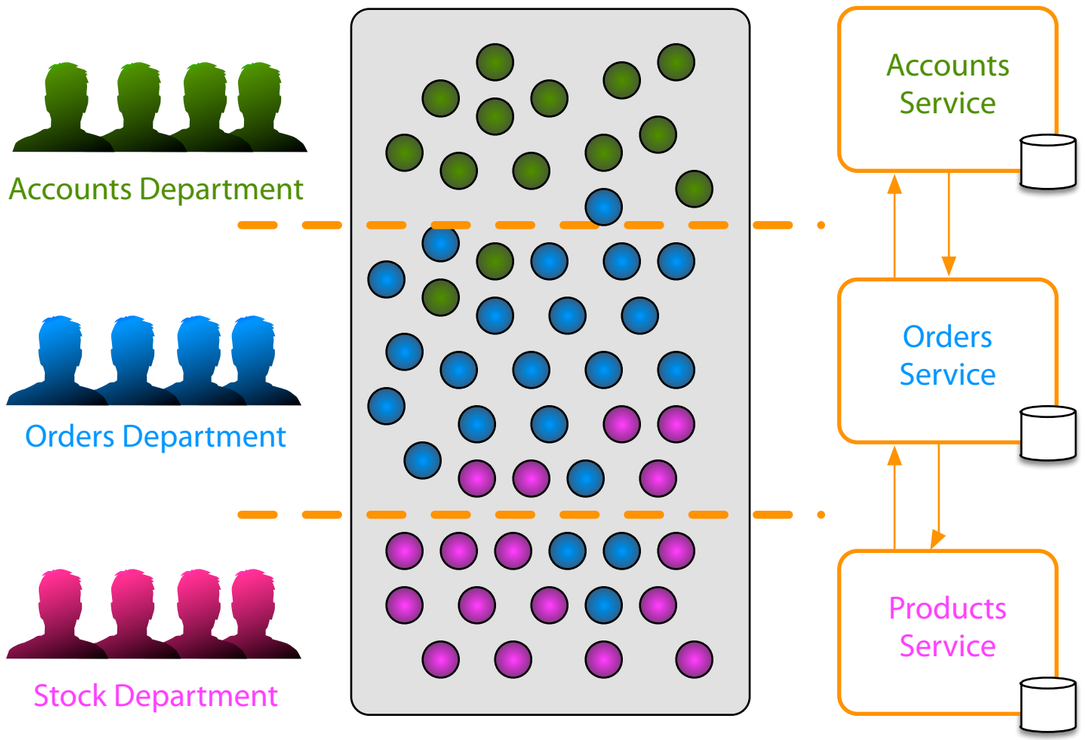
Approach | Migration | Database Migration | Transactions | Reporting

Brownfield Microservices: Approach



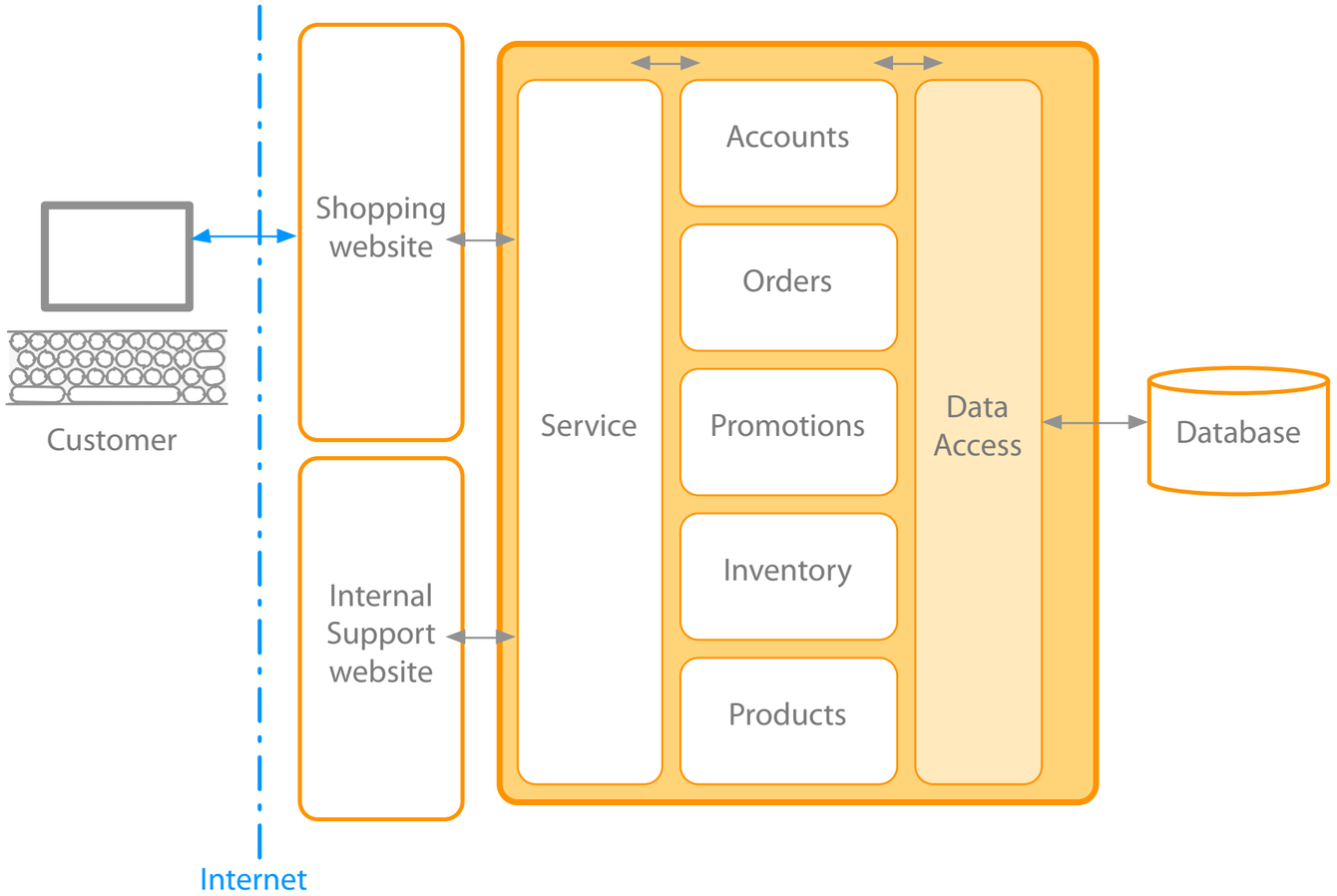
- Existing system
  - Monolithic system
  - Organically grown
  - Seems to large to split
- Lacks microservices design principles
- Identify seams
  - Separation that reflects domains
  - Identify bounded contexts
- Start modularising the bounded contexts
  - Move code incrementally
  - Tidy up a section per release
    - Take your time
    - Existing functionality needs to remain intact
  - Run unit tests and integration tests to validate change
  - Keep reviewing
- Seams are future microservice boundaries

Brownfield Microservices: Approach



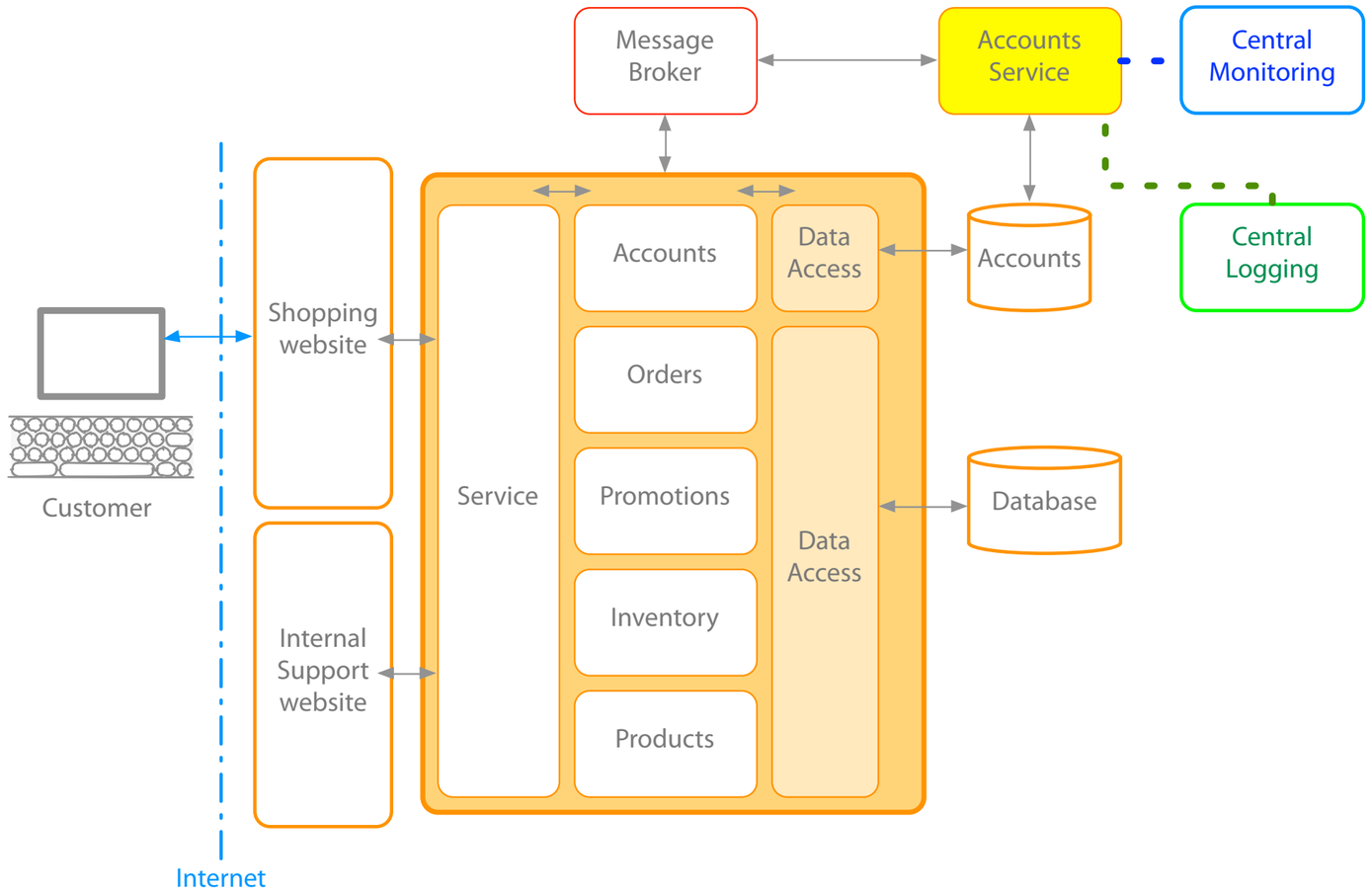
- Existing system
  - Monolithic system
  - Organically grown
  - Seems to large to split
- Lacks microservices design principles
- Identify seams
  - Separation that reflects domains
  - Identify bounded contexts
- Start modularising the bounded contexts
  - Move code incrementally
  - Tidy up a section per release
    - Take your time
    - Existing functionality needs to remain intact
  - Run unit tests and integration tests to validate change
  - Keep reviewing
- Seams are future microservice boundaries

Brownfield Microservices: Migration



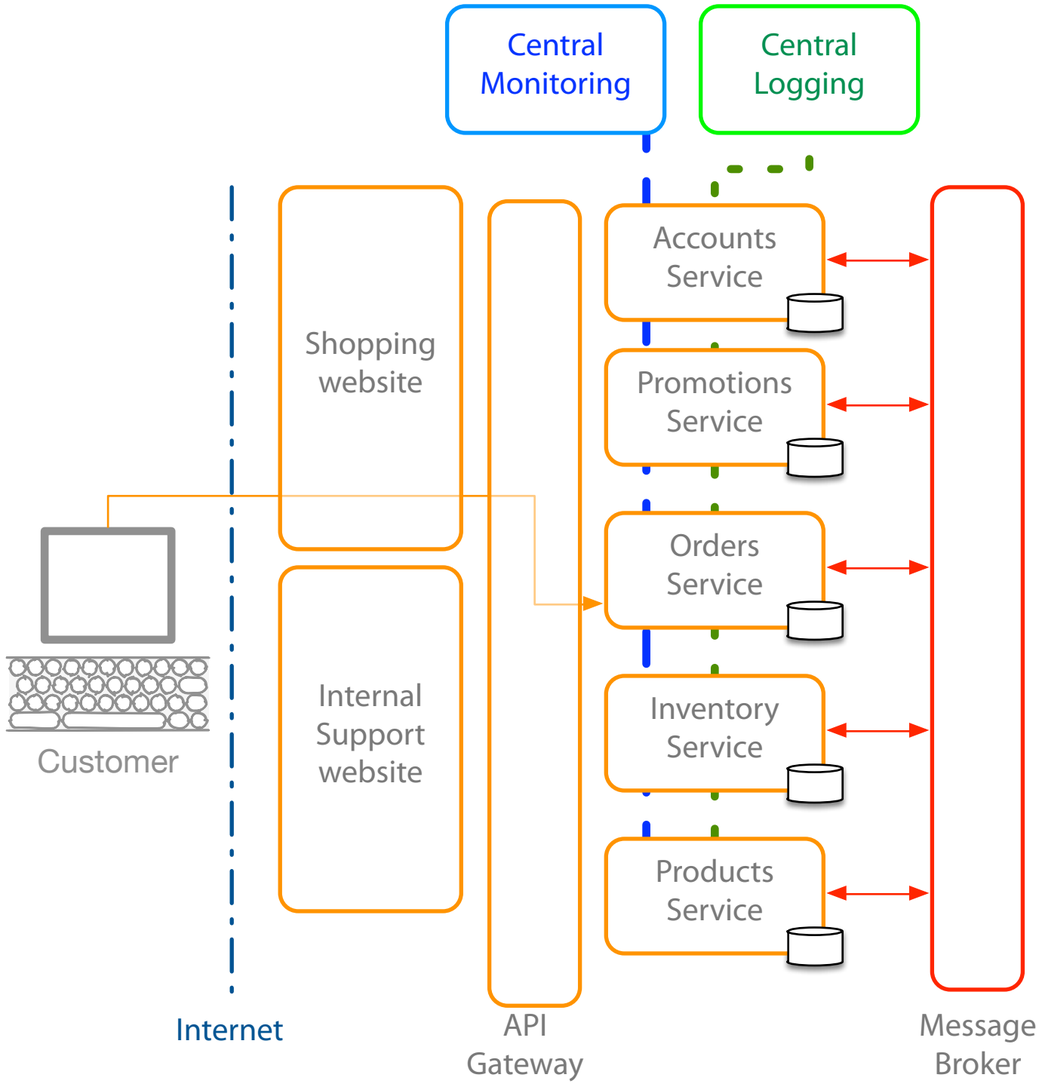
- Code is organised into bounded contexts
  - Code related to a business domain or function is in one place
  - Clear boundaries with clear interfaces between each
- Convert bounded contexts into microservices
  - Start of with one
    - Use to get comfortable
  - Make it switchable
    - Maintain two versions of the code
- How to prioritise what to split?
  - By risk
  - By technology
  - By dependencies
- Incremental approach
- Integrating with the monolithic
  - Monitor both for impact
  - Monitor operations that talk to microservices
  - Review and improve infrastructure
  - Incrementally the monolithic will be converted

Brownfield Microservices: Migration



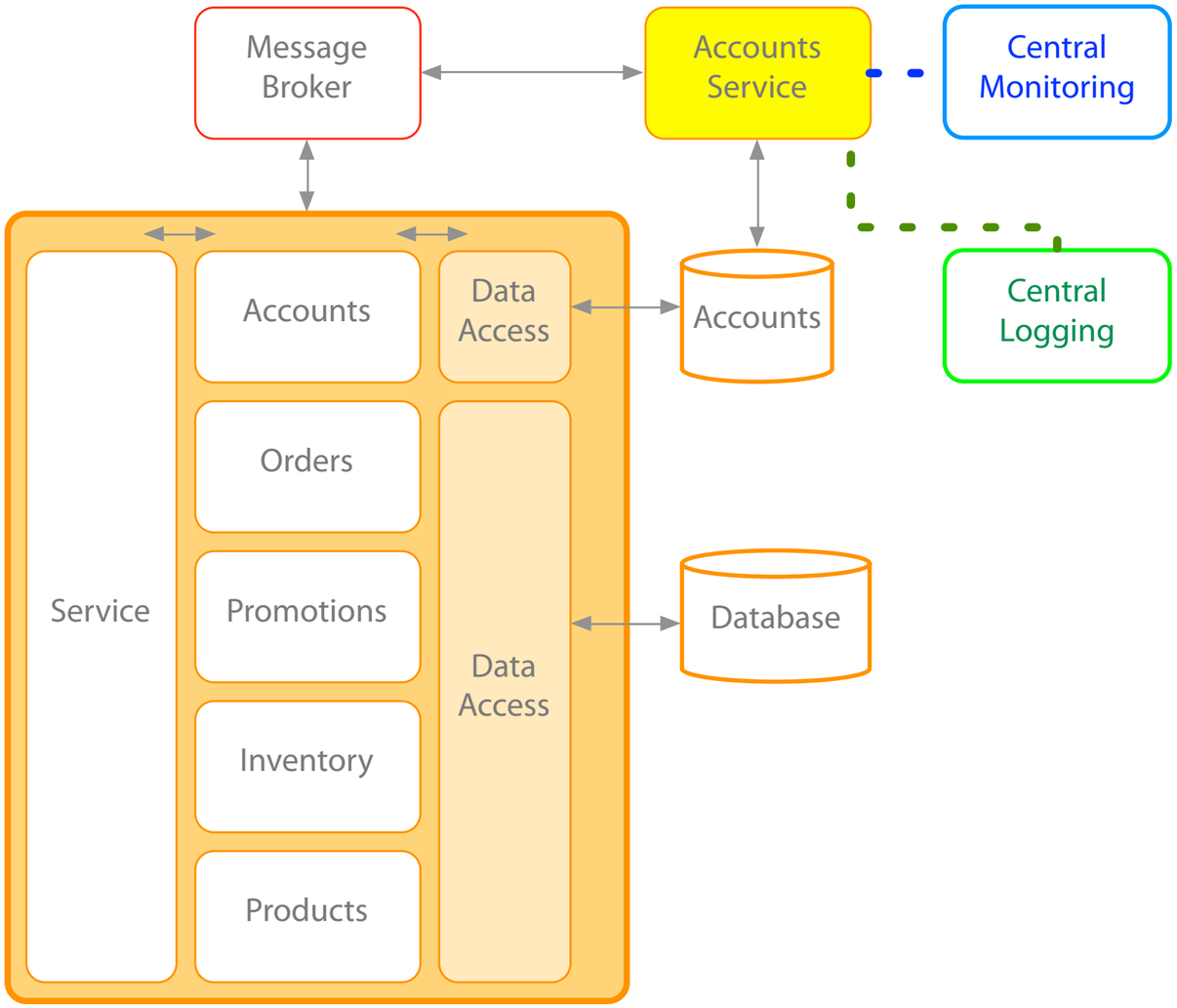
- Code is organised into bounded contexts
  - Code related to a business domain or function is in one place
  - Clear boundaries with clear interfaces between each
- Convert bounded contexts into microservices
  - Start of with one
    - Use to get comfortable
  - Make it switchable
    - Maintain two versions of the code
- How to prioritise what to split?
  - By risk
  - By technology
  - By dependencies
- Incremental approach
- Integrating with the monolithic
  - Monitor both for impact
  - Monitor operations that talk to microservices
  - Review and improve infrastructure
  - Incrementally the monolithic will be converted

Brownfield Microservices: Migration



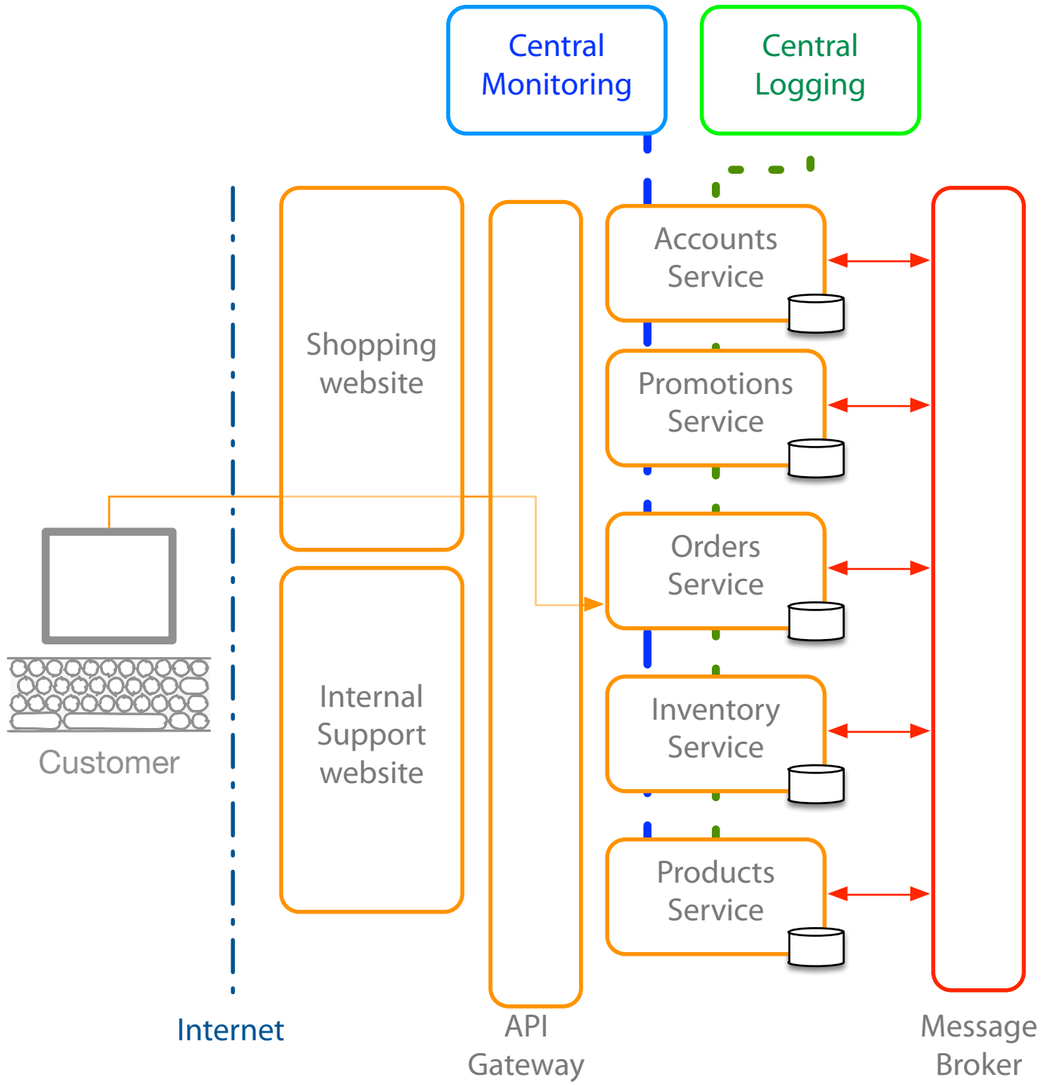
- Code is organised into bounded contexts
  - Code related to a business domain or function is in one place
  - Clear boundaries with clear interfaces between each
- Convert bounded contexts into microservices
  - Start of with one
    - Use to get comfortable
  - Make it switchable
    - Maintain two versions of the code
- How to prioritise what to split?
  - By risk
  - By technology
  - By dependencies
- Incremental approach
- Integrating with the monolithic
  - Monitor both for impact
  - Monitor operations that talk to microservices
  - Review and improve infrastructure
  - Incrementally the monolithic will be converted

Brownfield Microservices: Database Migration



- Avoid shared databases
- Split databases using seams
  - Relate tables to code seams
- Supporting the existing application
  - Data layer that connects to multiple database
- Tables that link across seams
  - API calls that can fetch that data for a relationship
- Refactor database into multiple databases
- Data referential integrity
- Static data tables
- Shared data

Brownfield Microservices: Database Migration



Avoid shared databases

Split databases using seams

- Relate tables to code seams

Supporting the existing application

- Data layer that connects to multiple database

Tables that link across seams

- API calls that can fetch that data for a relationship

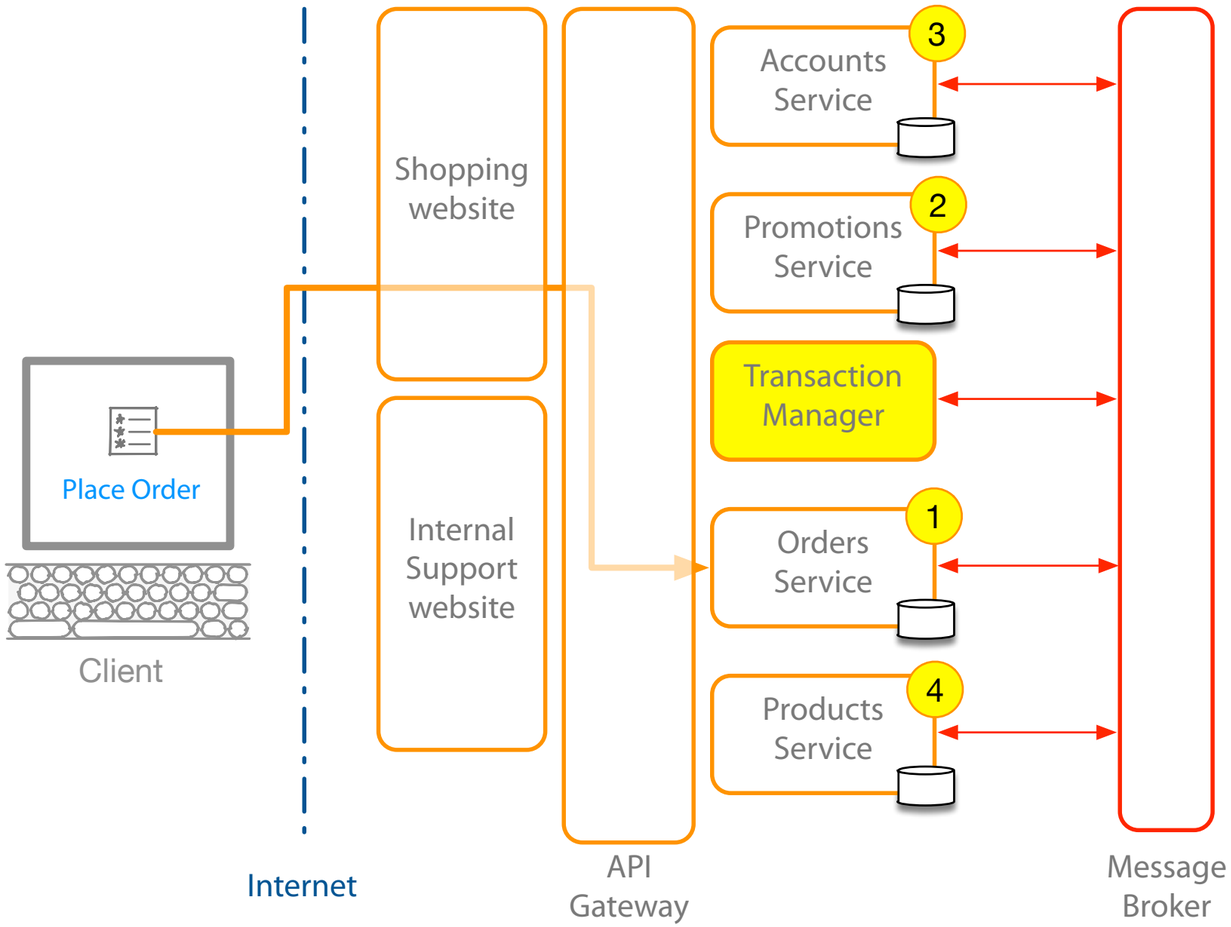
Refactor database into multiple databases

Data referential integrity

Static data tables

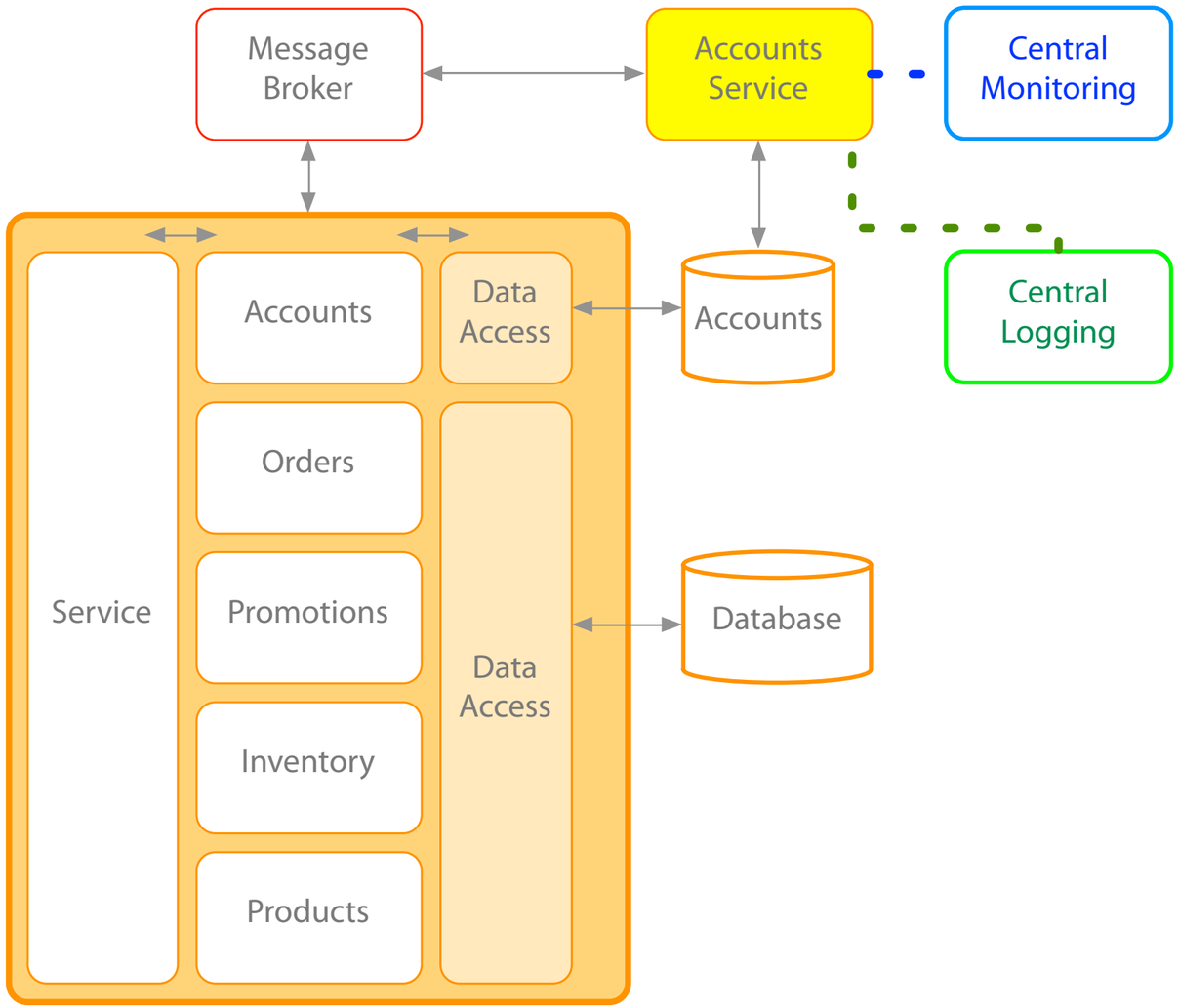
Shared data

Brownfield Microservices: Transactions



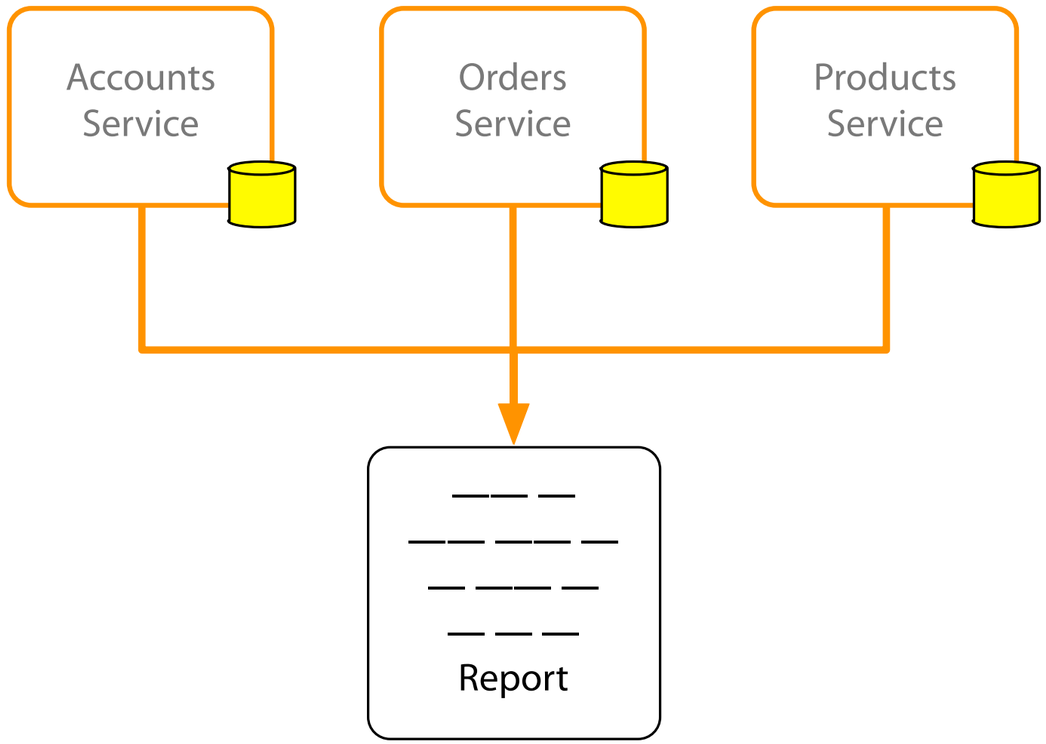
- Transactions ensure data integrity
- Transactions are simple in monolithic applications
- Transactions spanning microservices are complex
  - Complex to observe
  - Complex to problem solve
  - Complex to rollback
- Options for failed transactions
  - Try again later
  - Abort entire transaction
  - Use a transaction manager
    - Two phase commit
  - Disadvantage of transaction manager
    - Reliance on transaction manager
    - Delay in processing
    - Potential bottleneck
    - Complex to implement
- Distributed transaction compatibility
  - Completed message for the monolith

Brownfield Microservices: Transactions



- Transactions ensure data integrity
- Transactions are simple in monolithic applications
- Transactions spanning microservices are complex
  - Complex to observe
  - Complex to problem solve
  - Complex to rollback
- Options for failed transactions
  - Try again later
  - Abort entire transaction
  - Use a transaction manager
    - Two phase commit
  - Disadvantage of transaction manager
    - Reliance on transaction manager
    - Delay in processing
    - Potential bottleneck
    - Complex to implement
- Distributed transaction compatibility
  - Completed message for the monolith

Brownfield Microservices: Reporting



Microservices complicate reporting

- Data split across microservices
- No central database
- Joining data across databases
- Slower reporting
- Complicate report development

Possible solutions

- Service calls for report data
- Data dumps
- Consolidation environment

Brownfield Microservices: Reporting

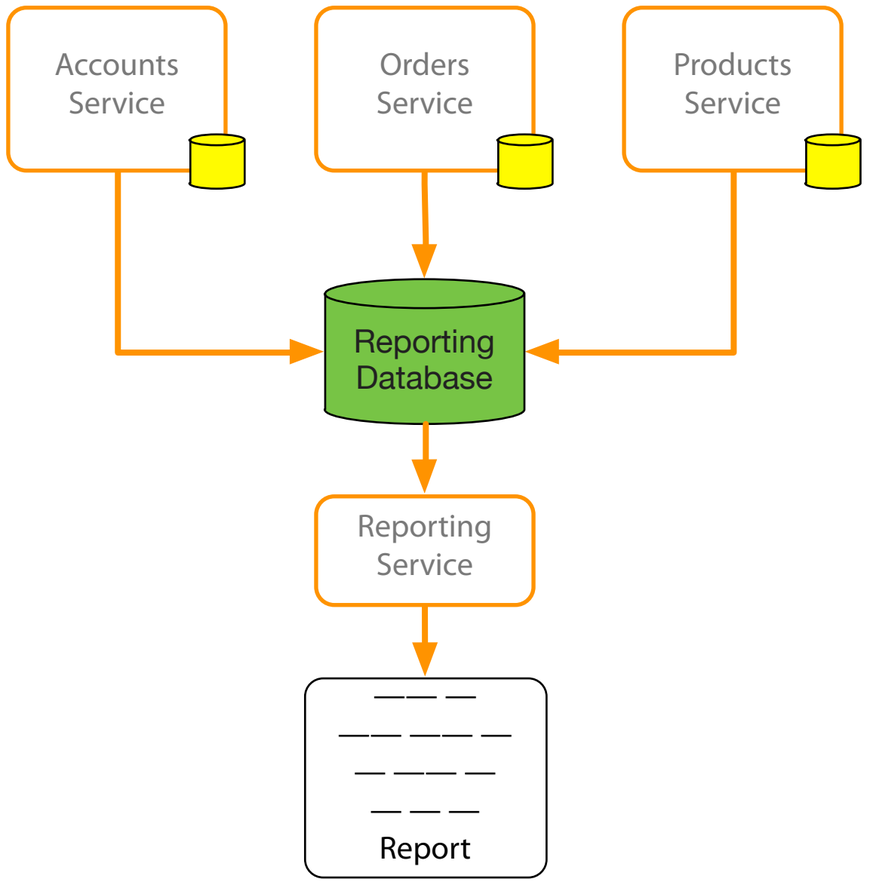
Microservices complicate reporting

- Data split across microservices
- No central database
- Joining data across databases
- Slower reporting
- Complicate report development

Possible solutions

- Service calls for report data
- Data dumps
- Consolidation environment

Brownfield Microservices: Reporting



Microservices complicate reporting

- Data split across microservices
- No central database
- Joining data across databases
- Slower reporting
- Complicate report development

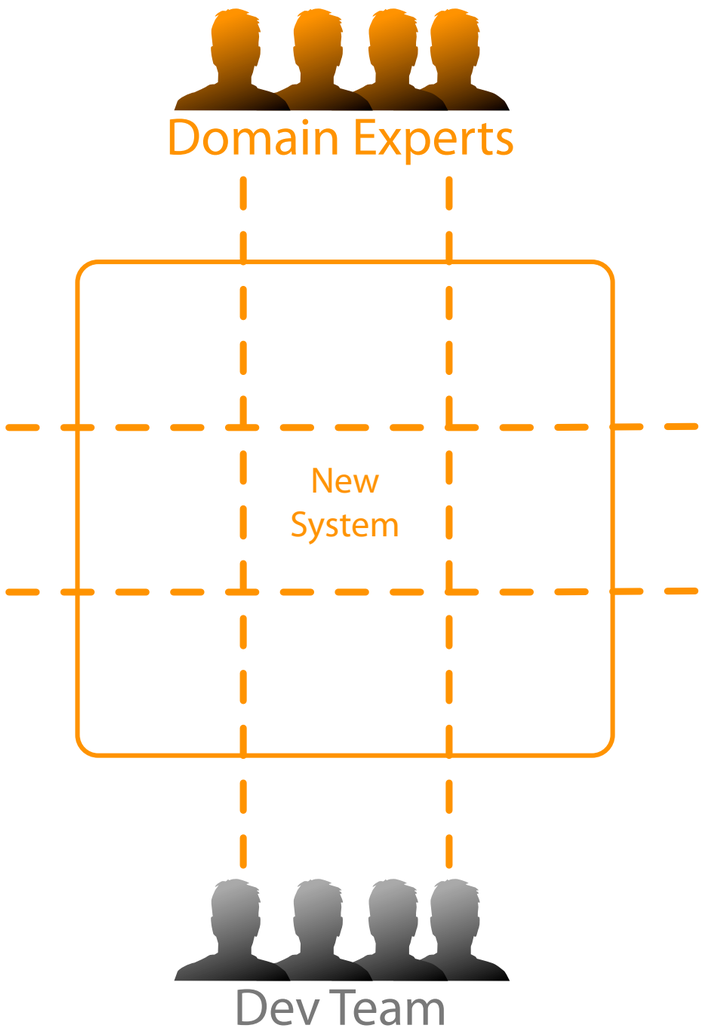
Possible solutions

- Service calls for report data
- Data dumps
- Consolidation environment

Greenfield Microservices

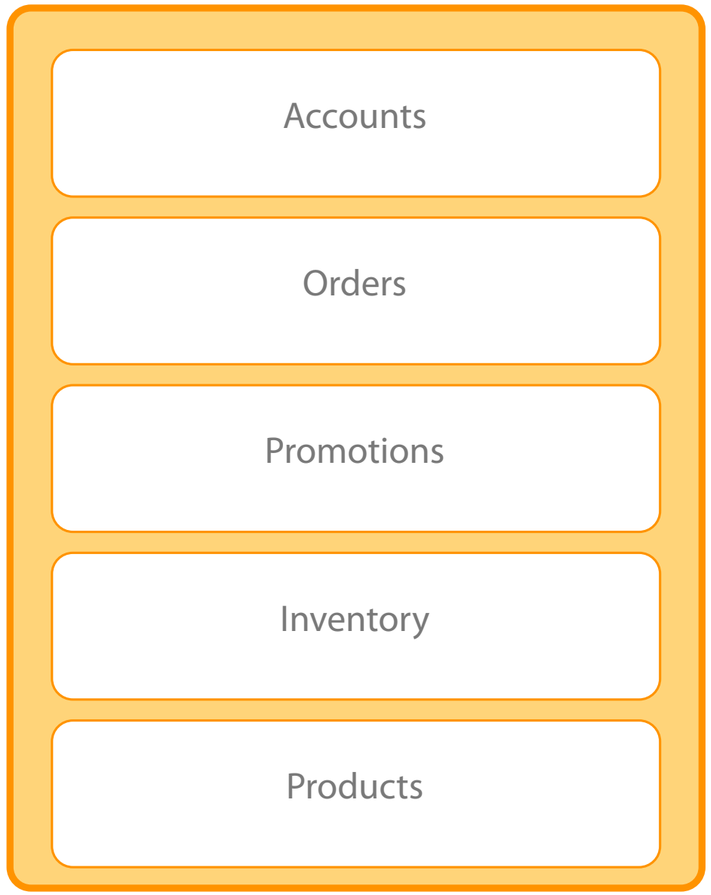
Introduction | Approach

Greenfield Microservices: Introduction



- New project
- Evolving requirements
- Business domain
  - Not fully understood
  - Getting domain experts involved
  - System boundaries will evolve
- Teams experience
  - First microservice
  - Experienced with microservices
- Existing system integration
  - Monolithic system
  - Established microservices architecture
- Push for change
  - Changes to apply microservice principles

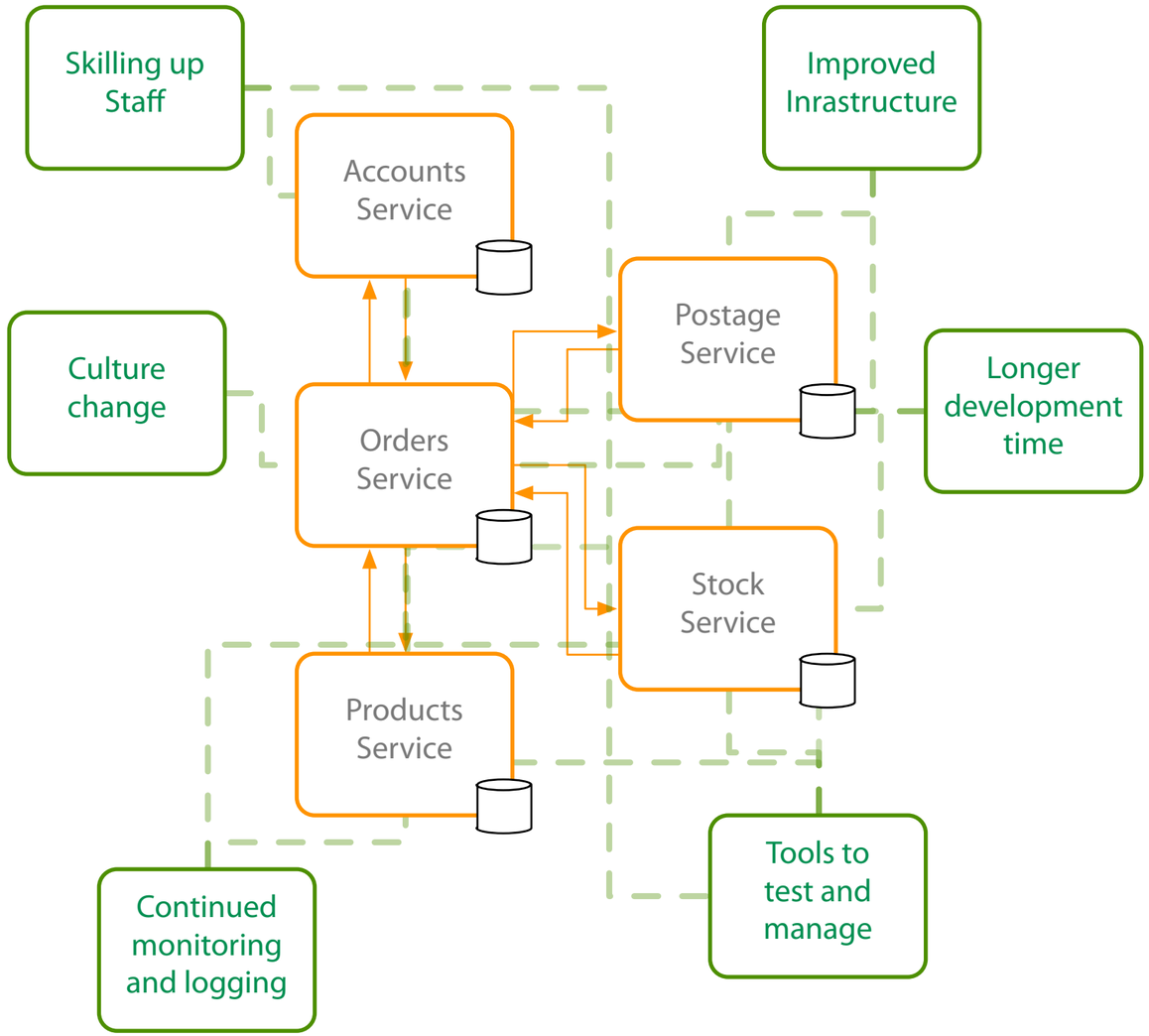
Greenfield Microservices: Approach



- Start of with monolithic design
  - High level
  - Evolving seams
  - Develop areas into modules
  - Boundaries start to become clearer
  - Refine and refactor design
  - Split further when required
- Modules become services
- Shareable code libraries promote to service
- Review microservice principles at each stage
- Prioritise by
  - Minimal viable product
  - Customer needs and demand

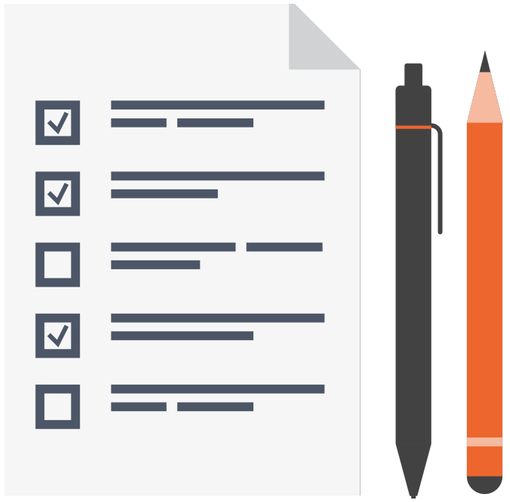
Microservices Provisos

Microservices Provisos



- Accepting initial expense
  - Longer development times
  - Cost and training for tools and new skills
- Skilling up for distributed systems
  - Handling distributed transactions
  - Handling reporting
- Additional testing resource
  - Latency and performance testing
  - Testing for resilience
- Improving infrastructure
  - Security
  - Performance
  - Reliance
- Overhead to mange microservices
- Cloud technologies
- Culture change

Module Summary



Brownfield Microservices

Greenfield Microservices

Microservices Provisos