Microsoft

tech·days

Kistamässan Stockholm

25th-26th October 2017

+AI

Design your Architecture for Continuous Delivery

Jakob Ehn

@jakobehn

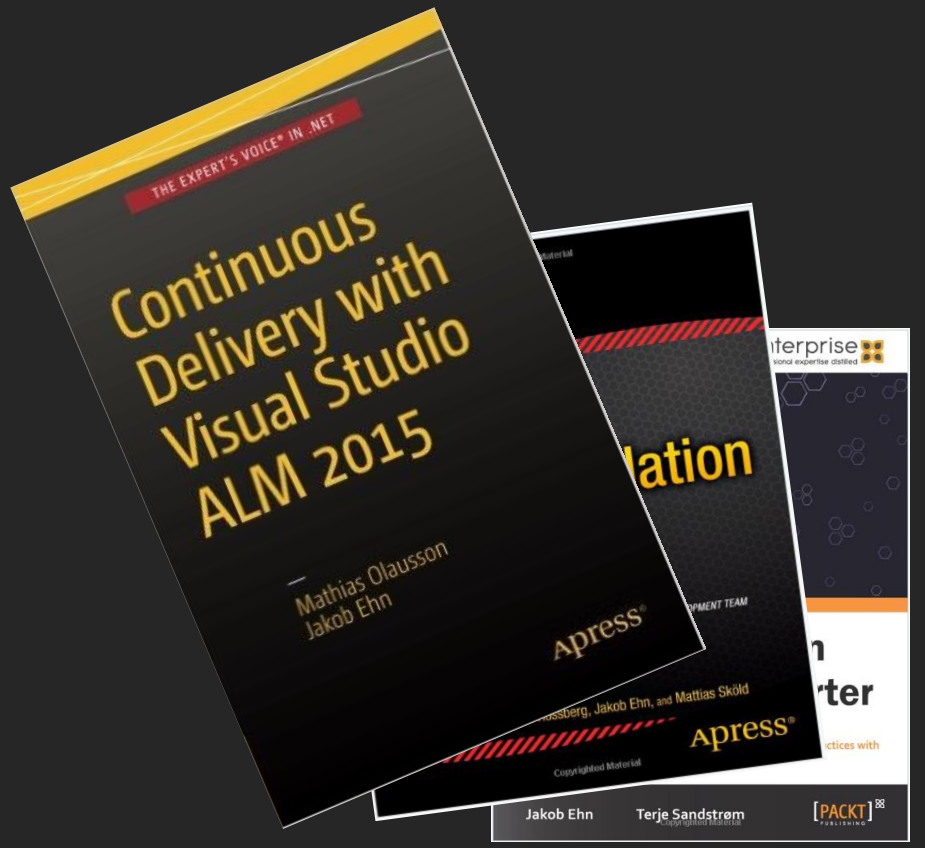
jakob.ehn@activesolution.se

https://blog.ehn.nu

active SOLUTION

About //Me

- Developer @ Active Solution
- Visual Studio MVP
- http://blog.ehn.nu
- @jakobehn



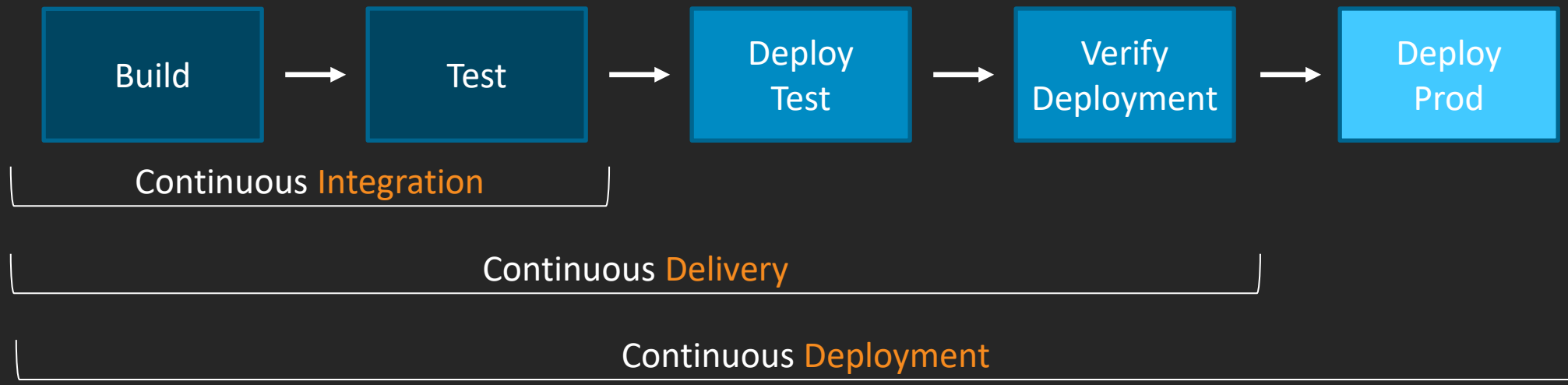
Active Solution

Microsoft Most Valuable Professional

Visual Studio ALM Rangers

Continuous Delivery

"The ability to get changes into production *safely* and *quickly* in a *sustainable* way"

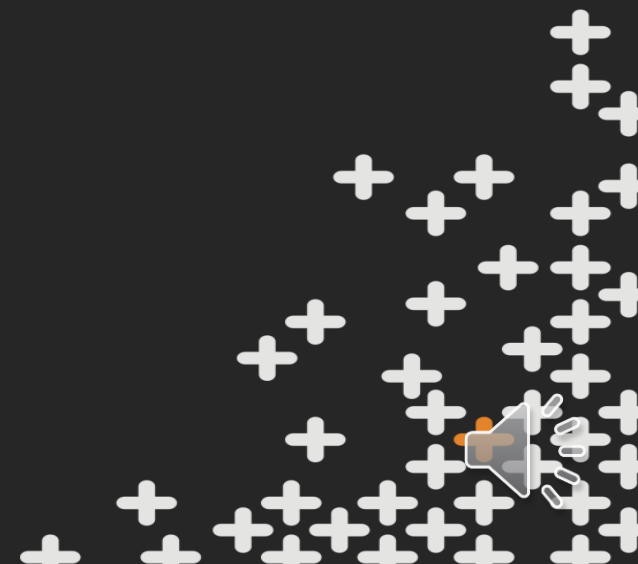


Principles for Continuous Delivery

- Deployment process must be repeatable and reliable
- Automate everything
- It something is painful, do it more often
- Keep everything in source control
- Done means ”released”
- Everybody is responsible for the release process
- Improve continuously

What’s stopping us from doing CD?

- Tightly coupled architecture (”Monolith”)
- Legacy tech
- Large single codebase
- Long lived branches
- Large batch sizes
- Big bang releases



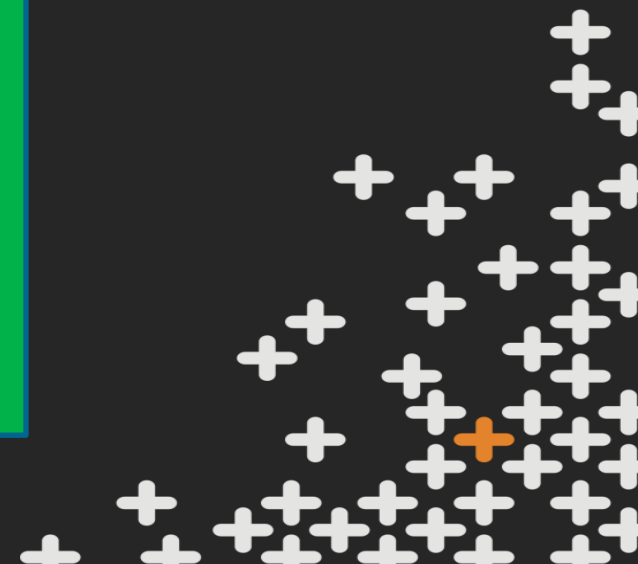
Continuous Delivery Enablers

Breaking up the Monolith

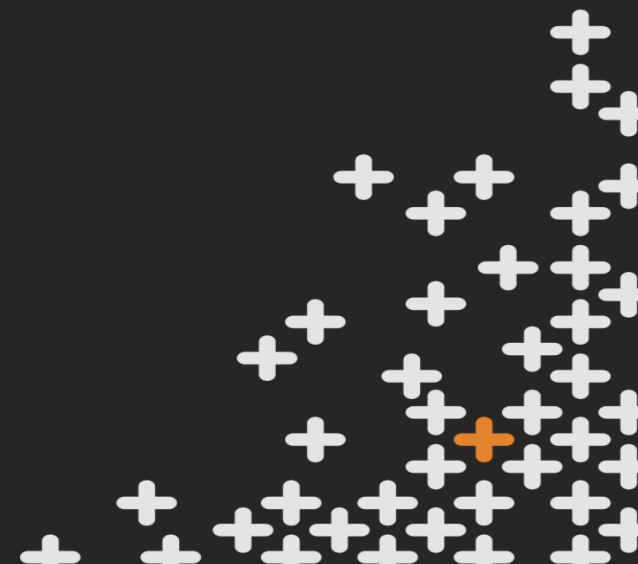
Designing for Resilience

Keeping your application releasable

Low risk Deployments

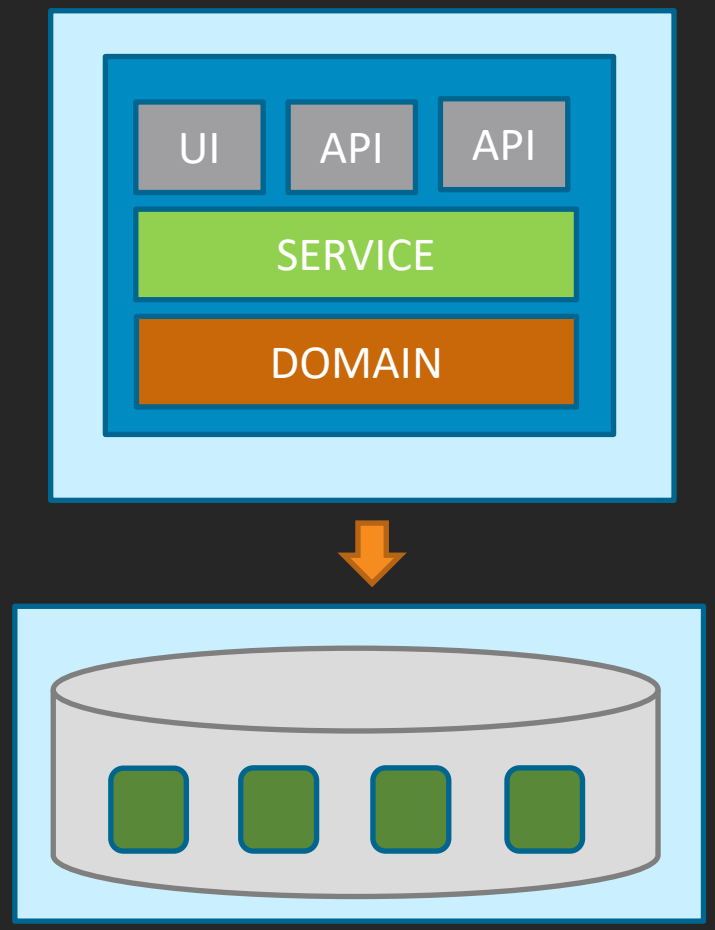


BREAKING UP THE MONOLITH

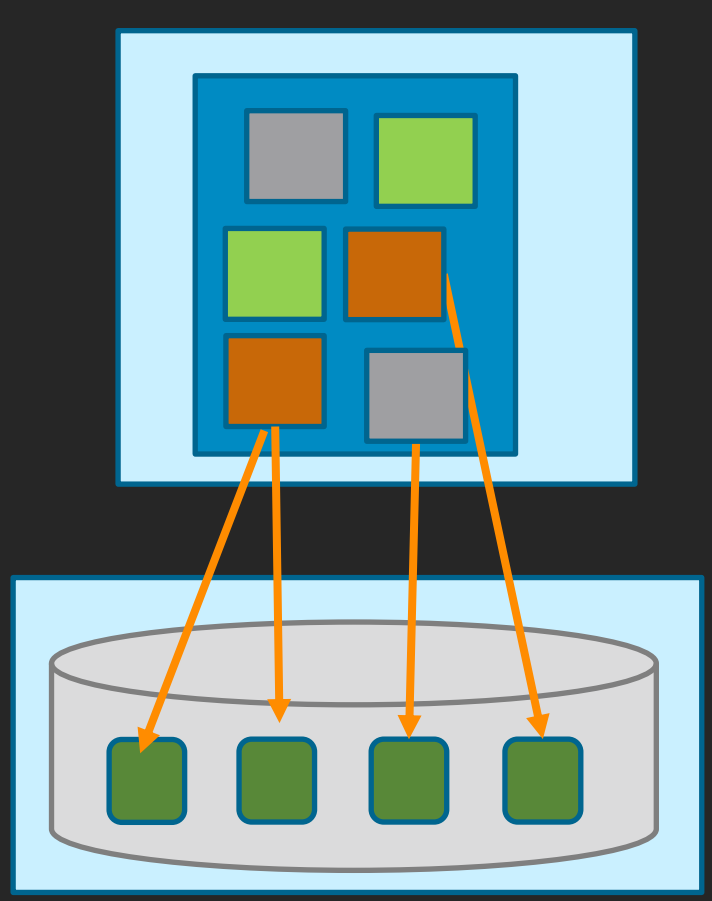


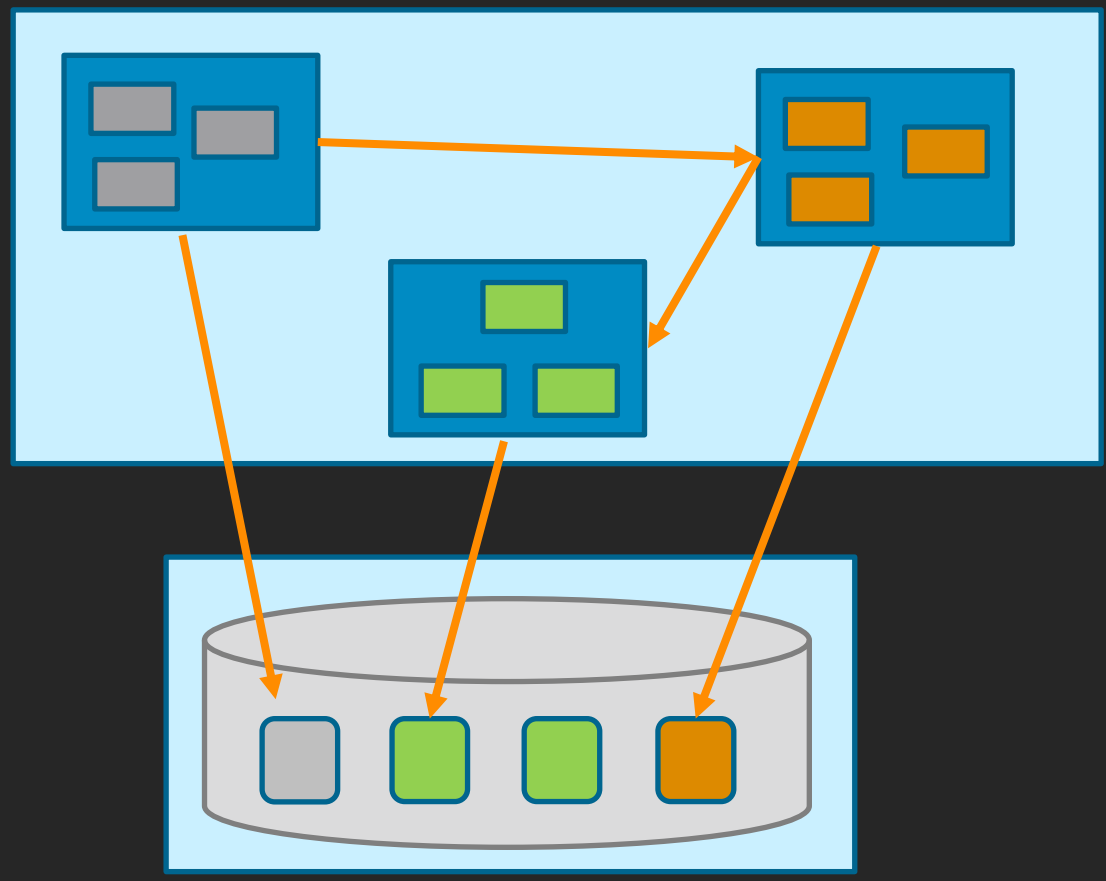
The Monolith

- Tightly coupled
- Usually implemented in the same technology
- Often integrates with a common database (SQL)
- Everything is deployed together

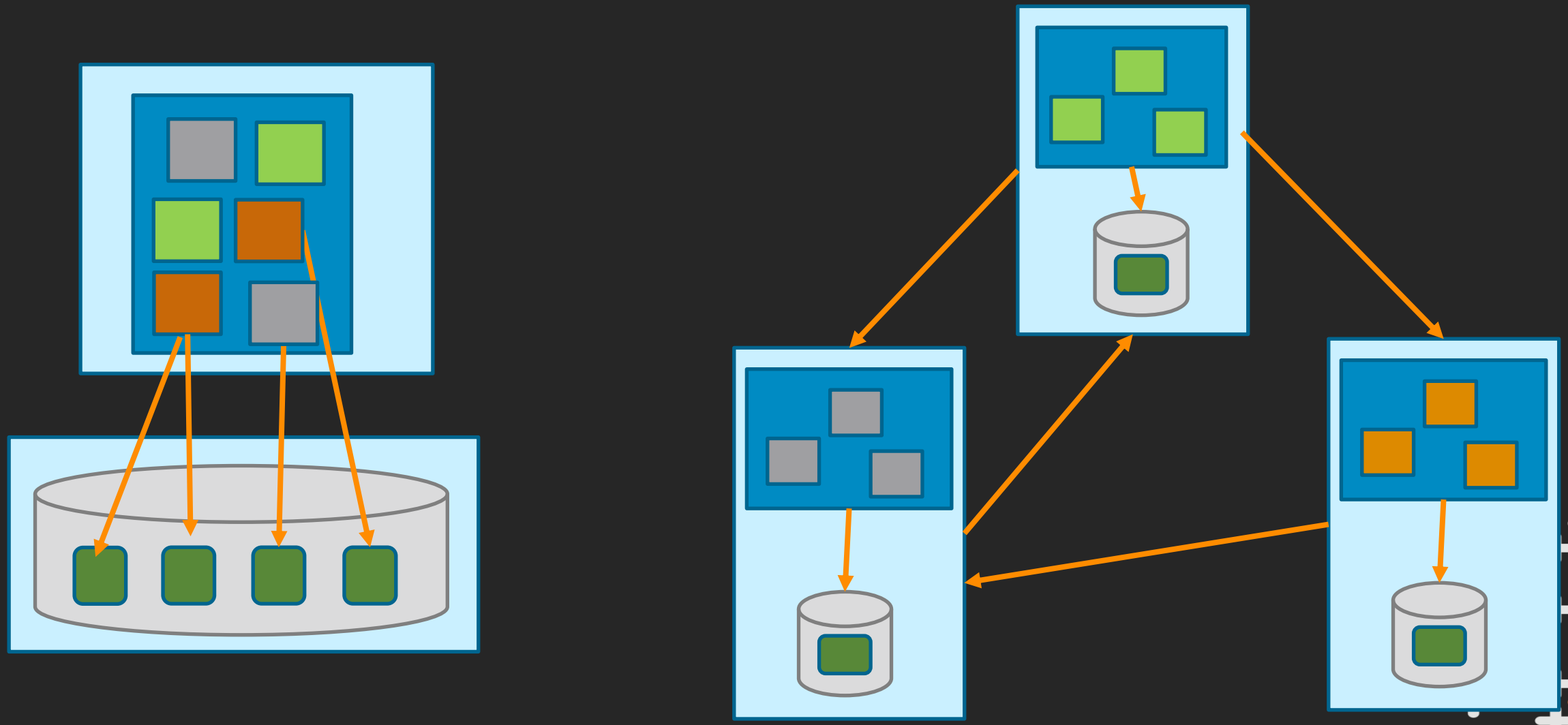


Component Based Architecture



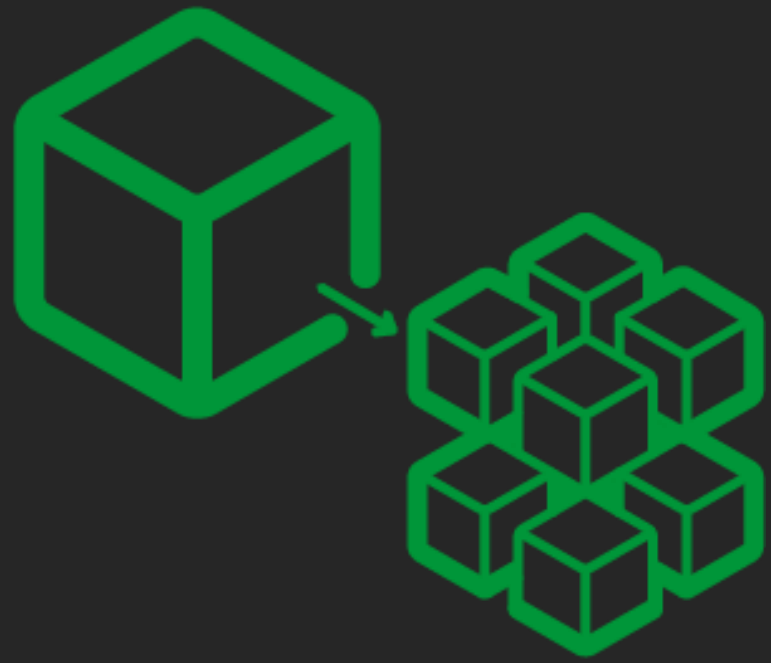


Microservices Architecture

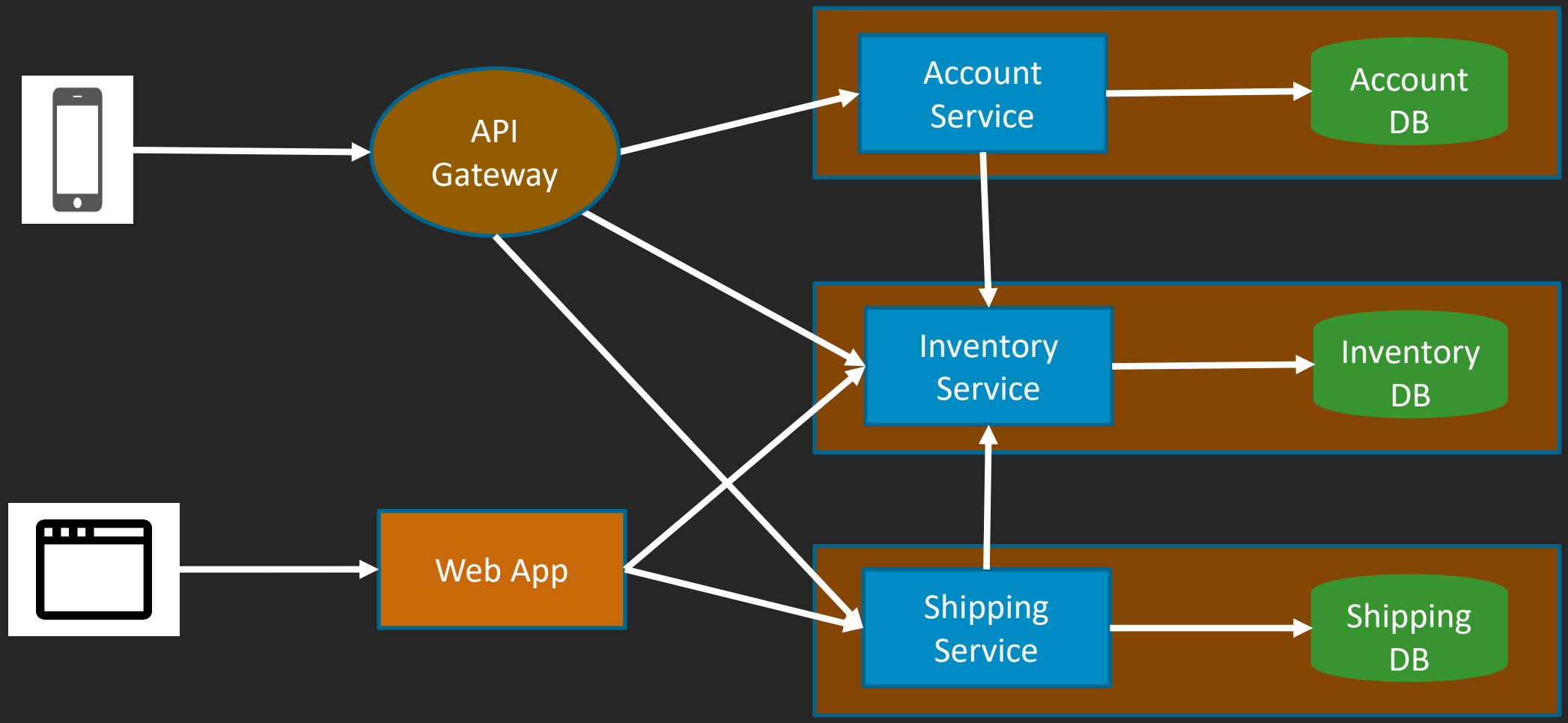


Principles of Microservices

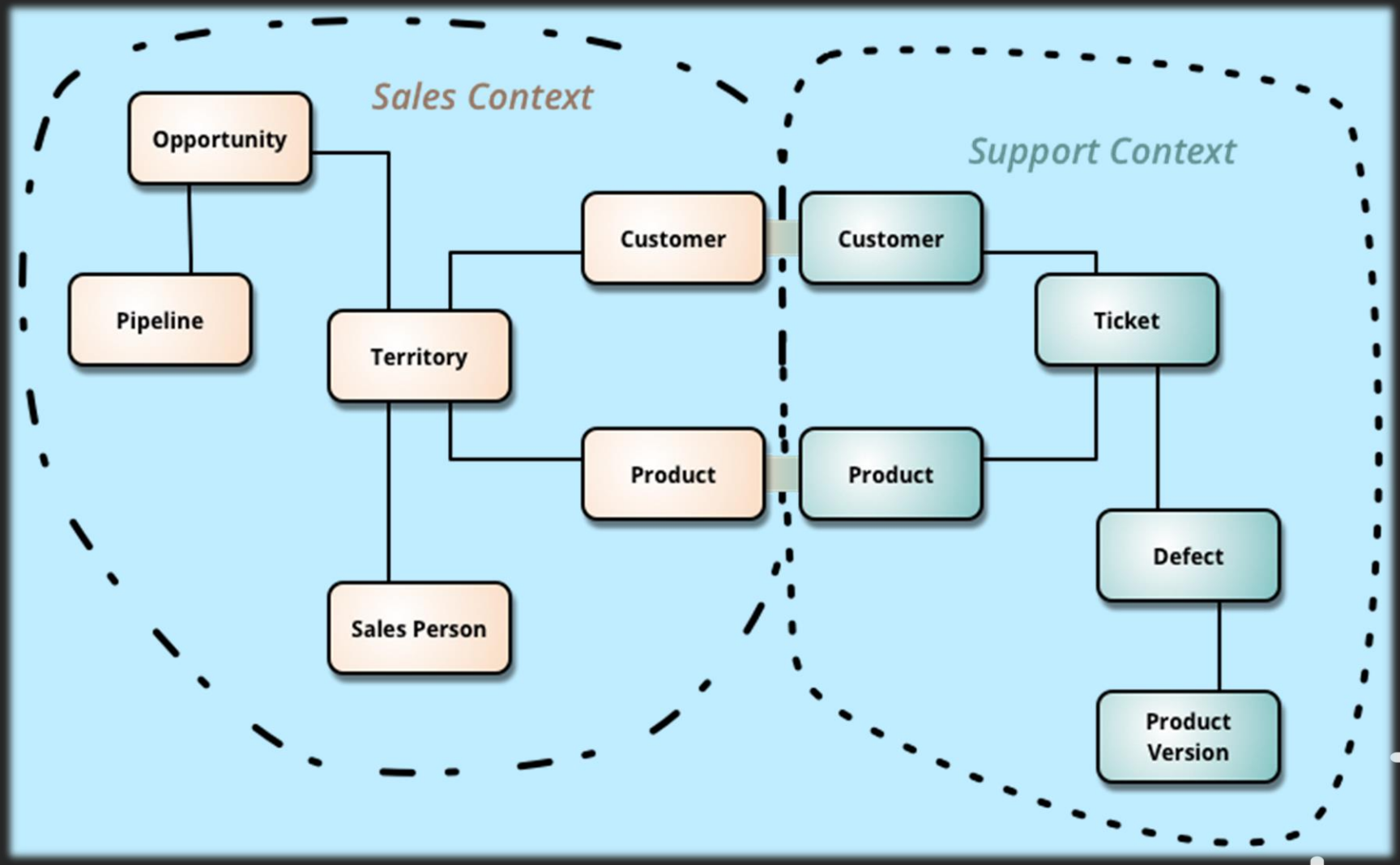
- Modelled around a business domain
- Separate codebase
- Decentralized
- Automated
- Deployed independently
- Implemented by cross-functional teams



Microservice Architecture



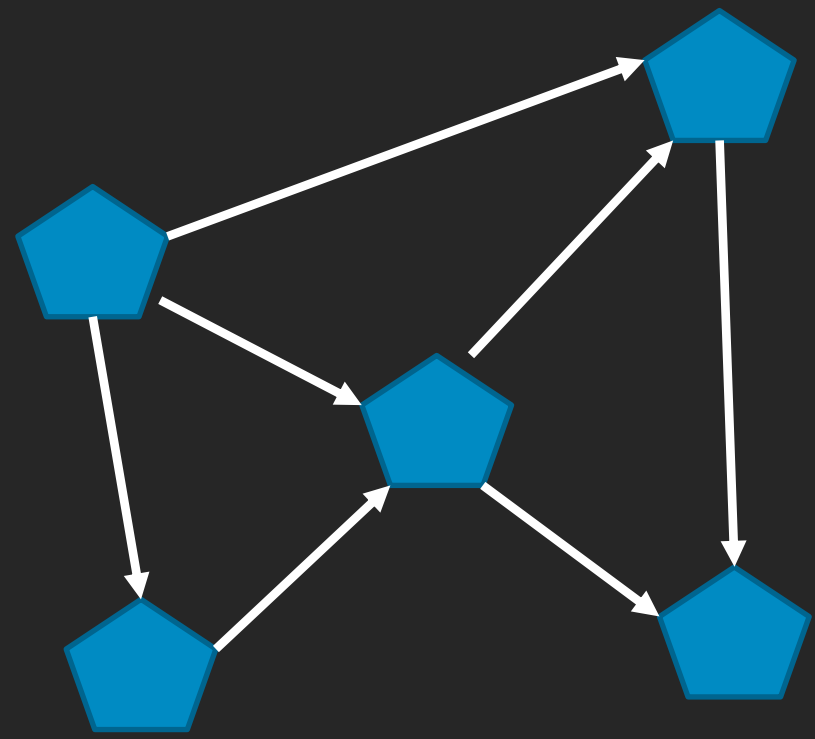
Bounded contexts



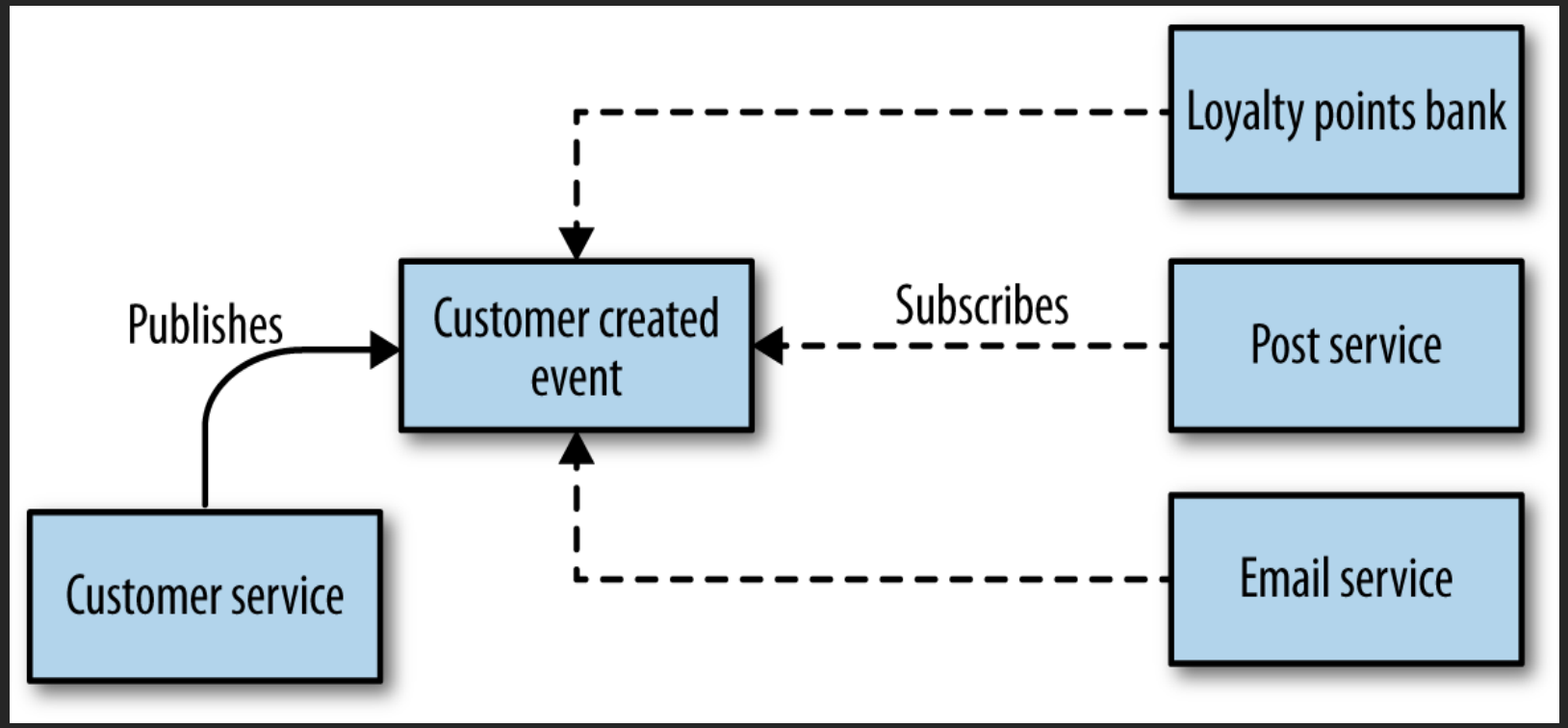
Microservices Integration

Prefer

- REST over RPC/SOAP
- Smart endpoints and dumb pipes
- Orchestration over Choreography



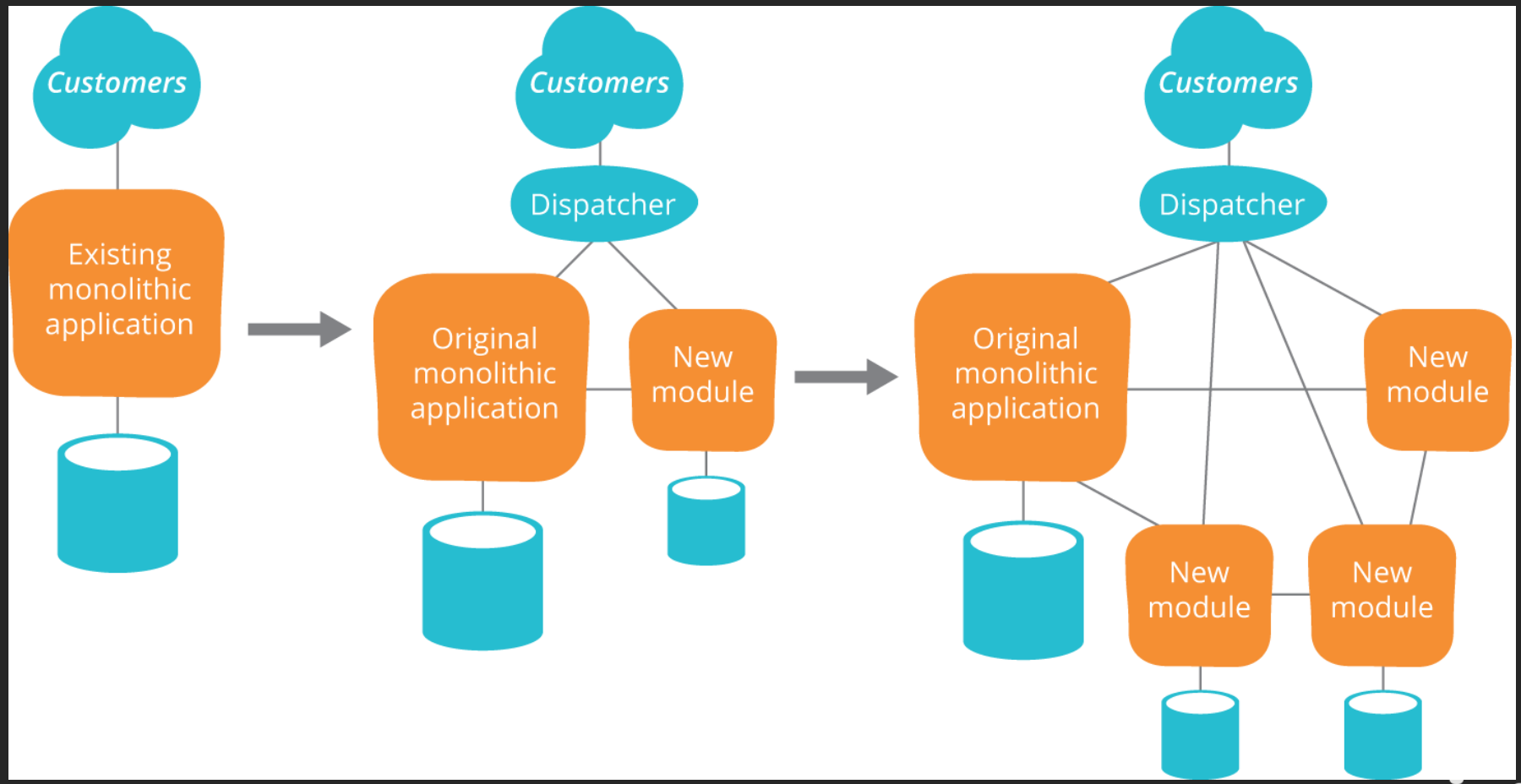
Orchestration vs Choreography



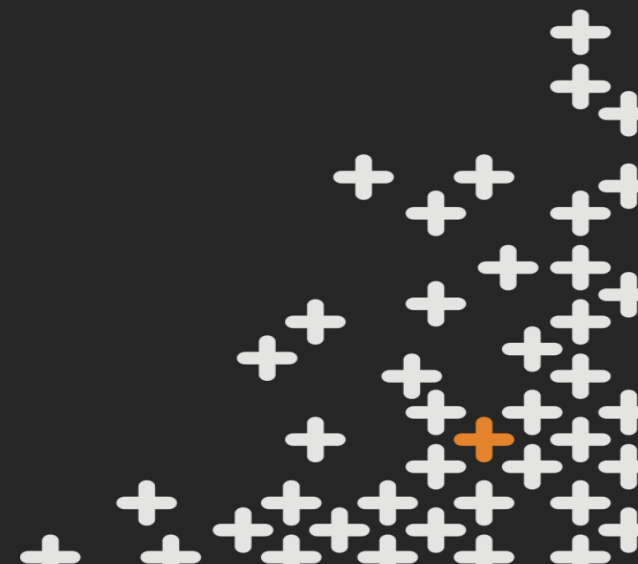
Orchestration Choreography

Migrating to Microservices

- Strangler Application

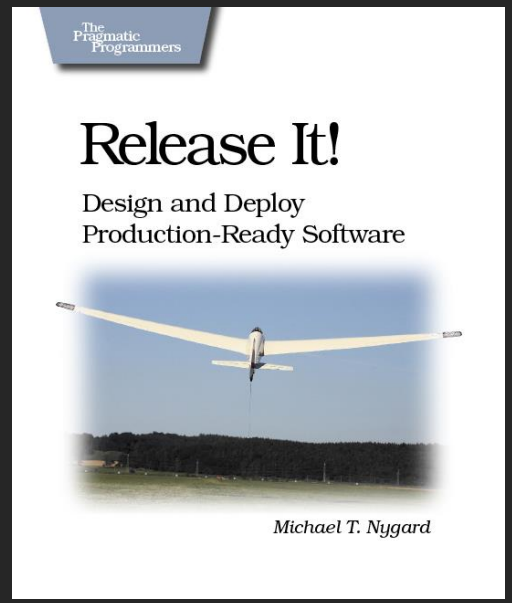


DESIGNING FOR RESILIENCE

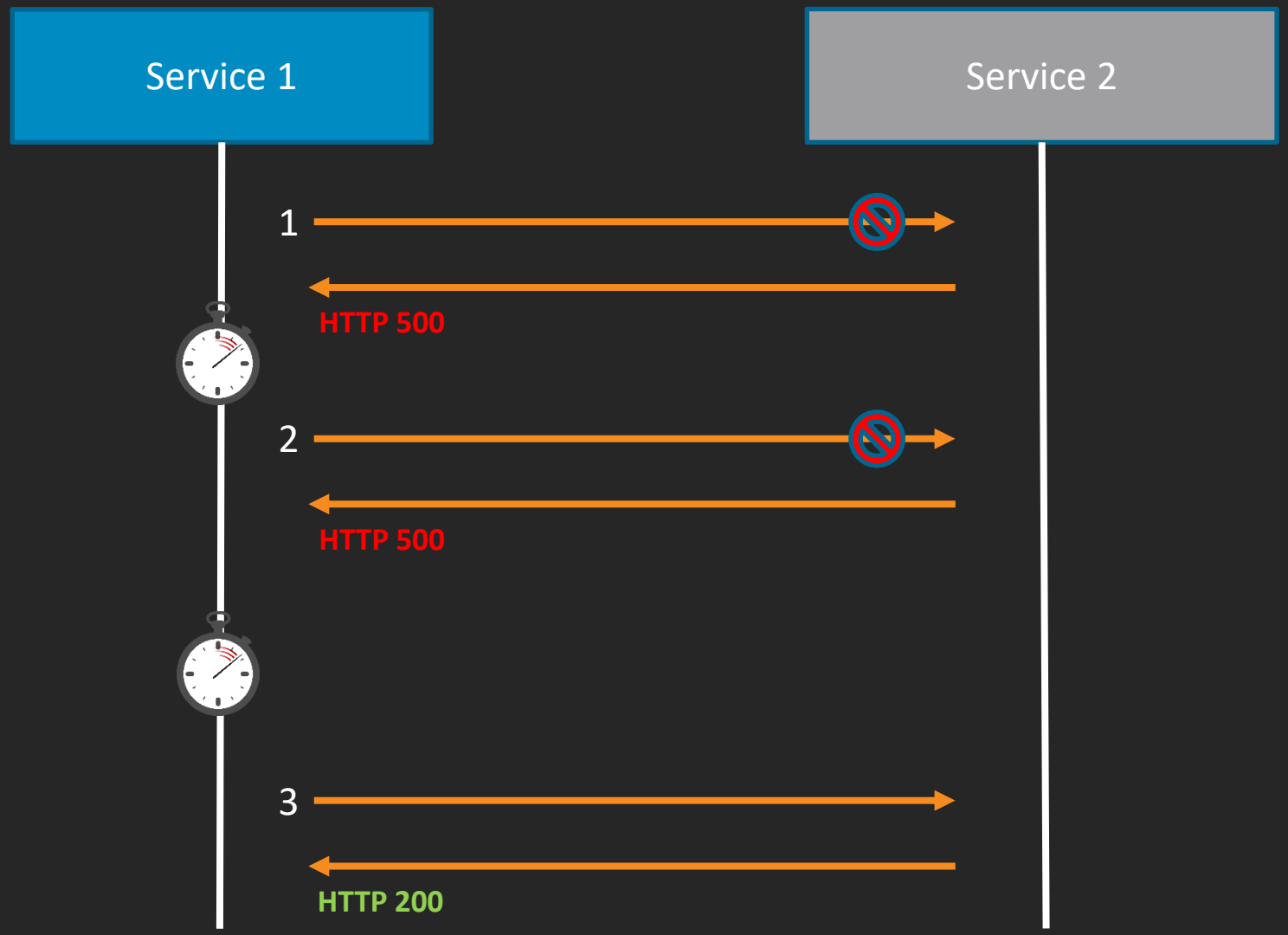


Designing for resilience

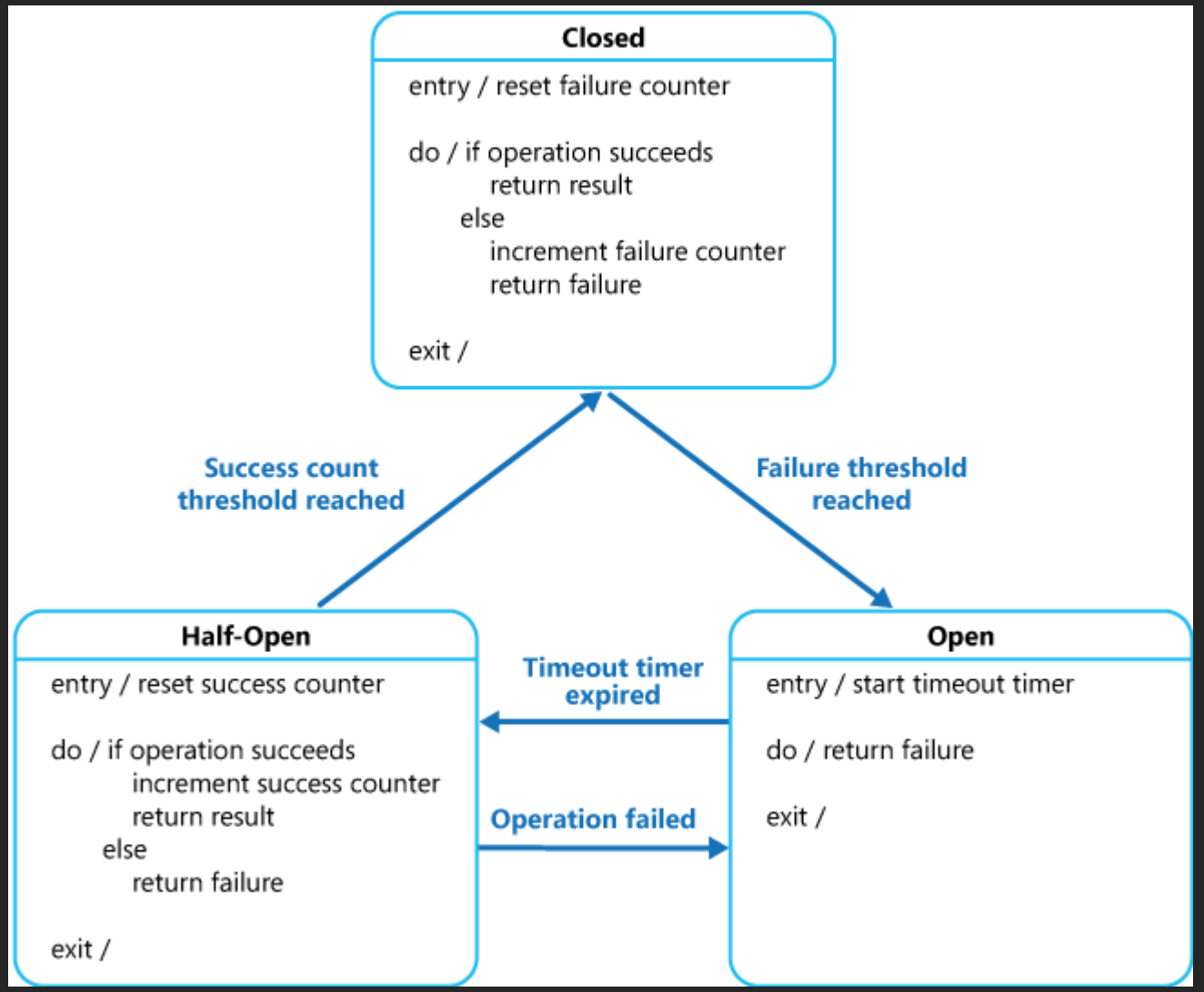
- “Integration points are the number-one killer of systems. Every single one of those feeds presents a stability risk.”
  - Michael T. Nygard, “Release It!”
- Be sceptic, expect failures to happen!
- Use established patterns



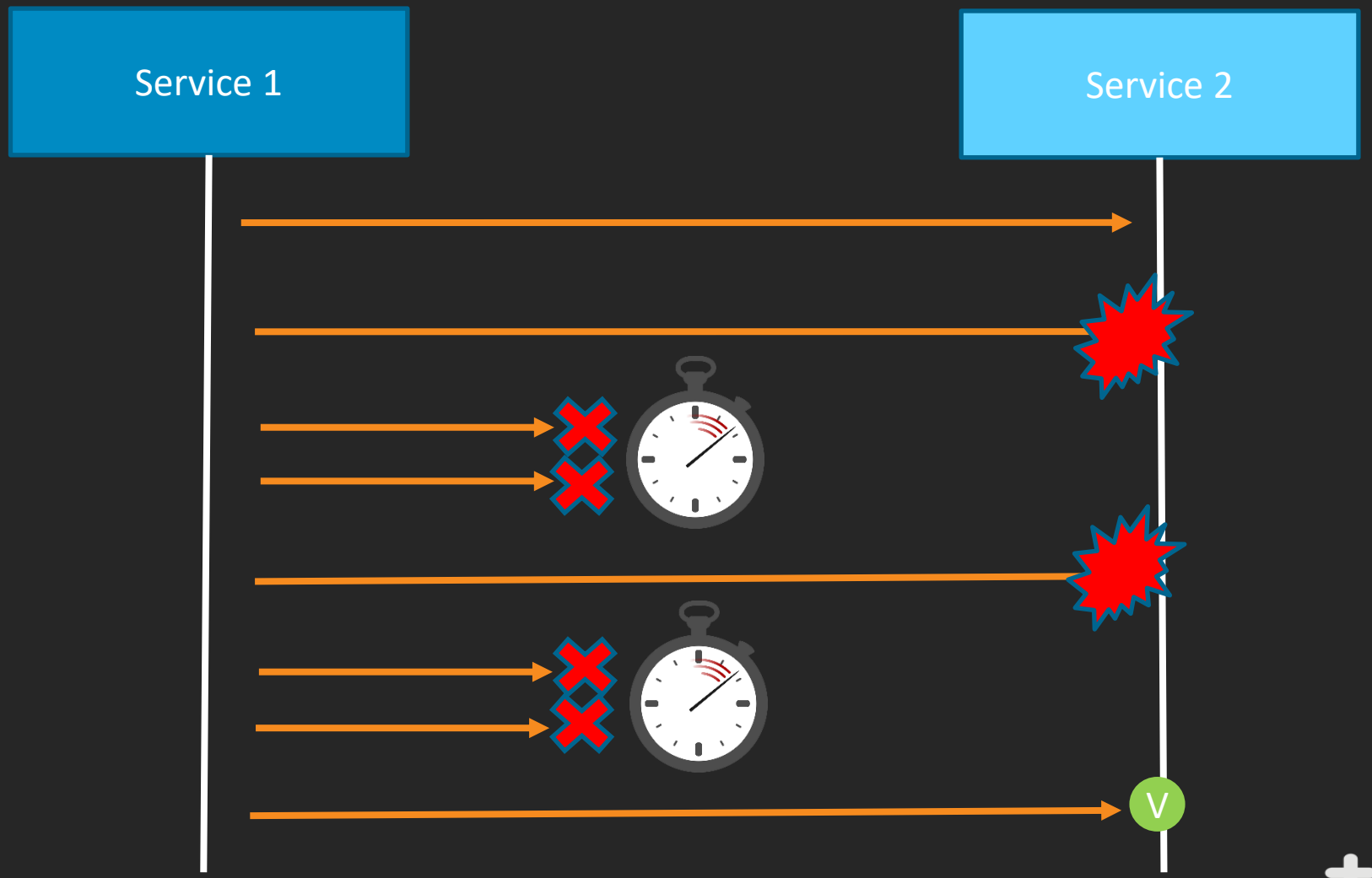
Wait & Retry



Circuit Breaker

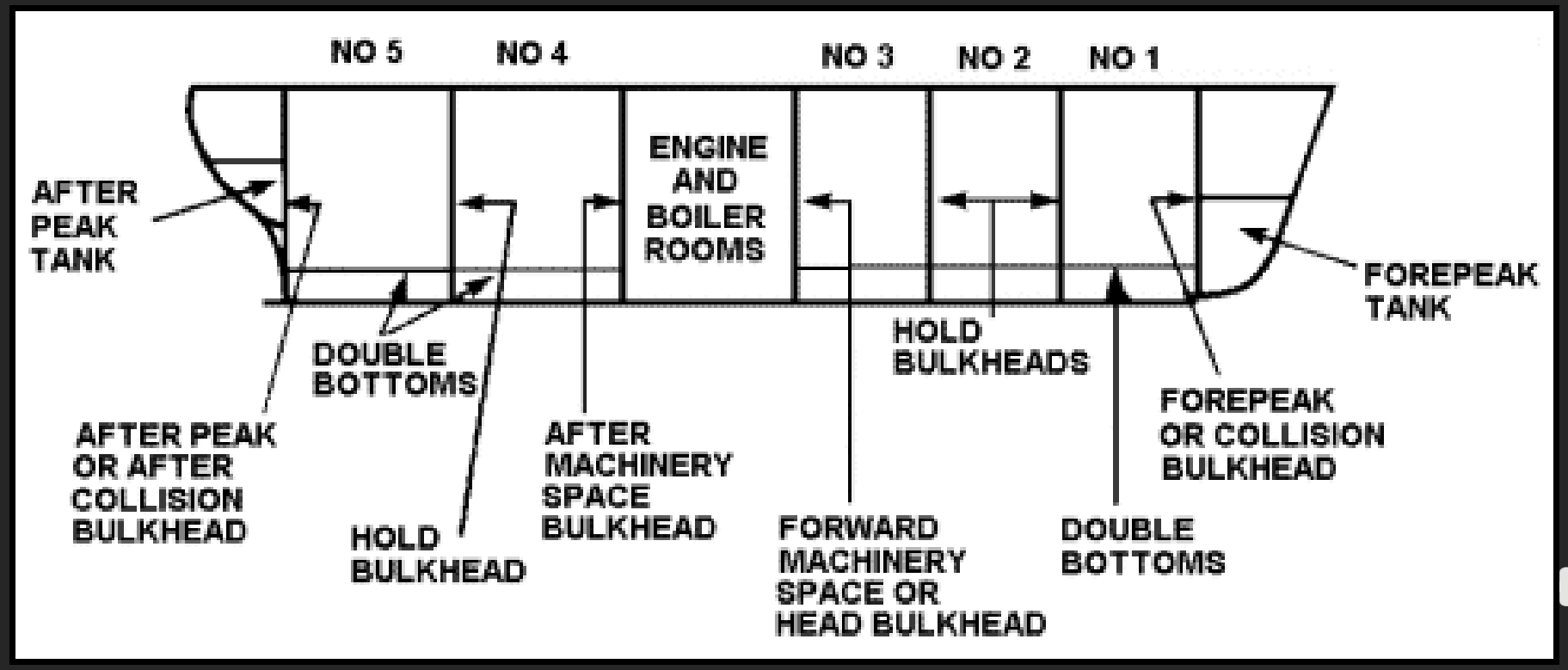


Circuit Breaker

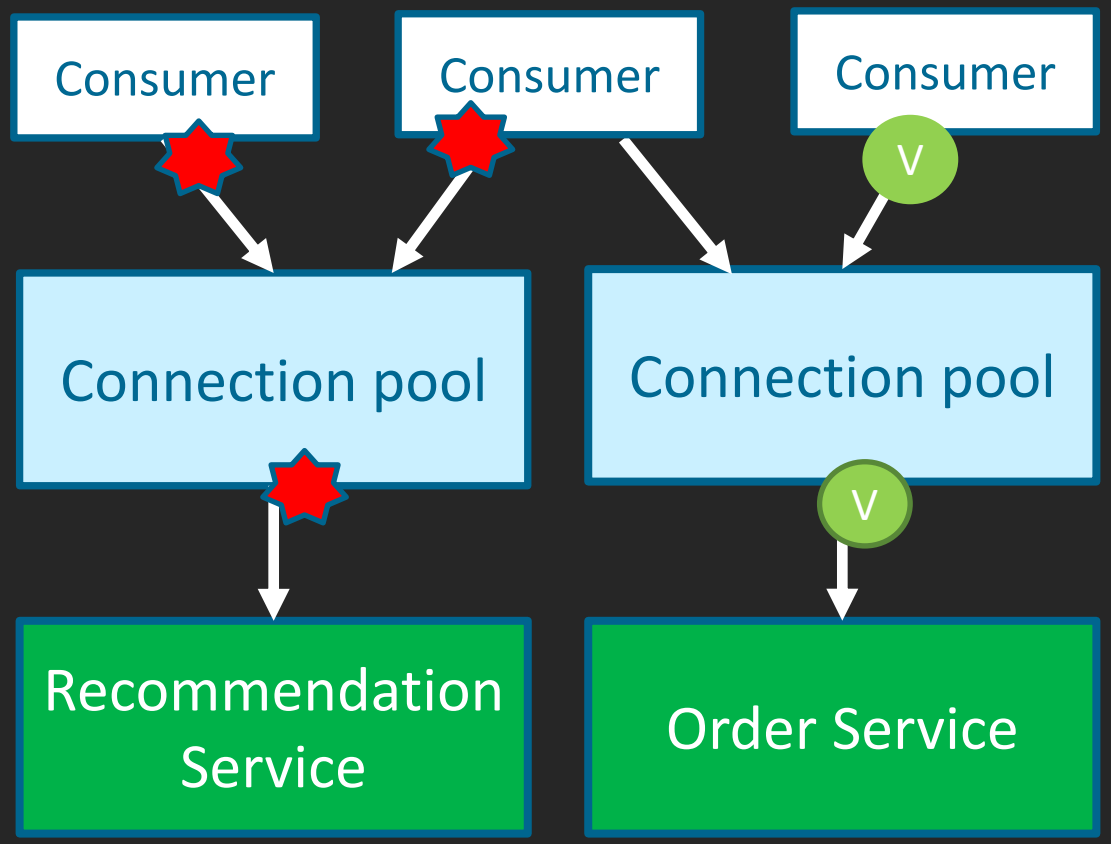


Bulkhead

- Partition resources to isolate errors



Bulkhead Partioning



Polly

- .NET resilience and transient-fault-handling library
- Support policies such as Retry, Circuit Breaker, Timeout, Bulkhead Isolation, and Fallback
- https://github.com/App-vNext/Polly

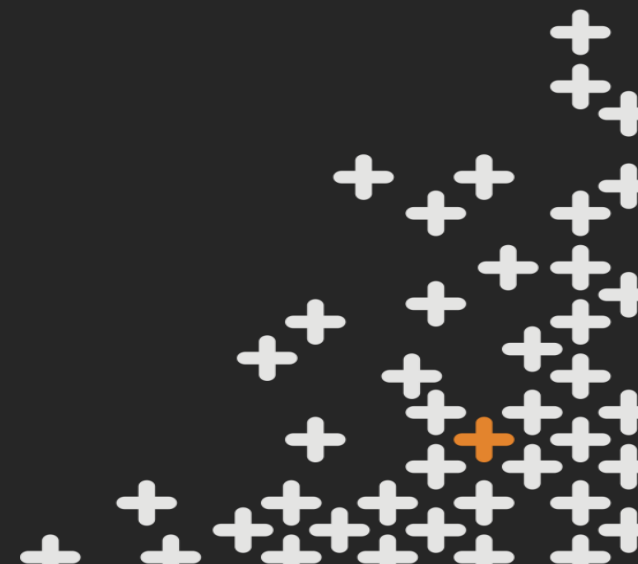
```
Policy.Handle<WebException>()
      .WaitAndRetry(5, count => TimeSpan.FromSeconds(count))
      .Execute(CallSomeService);
```

KEEPING YOUR APPLICATION RELEASABLE

Releasing continuously

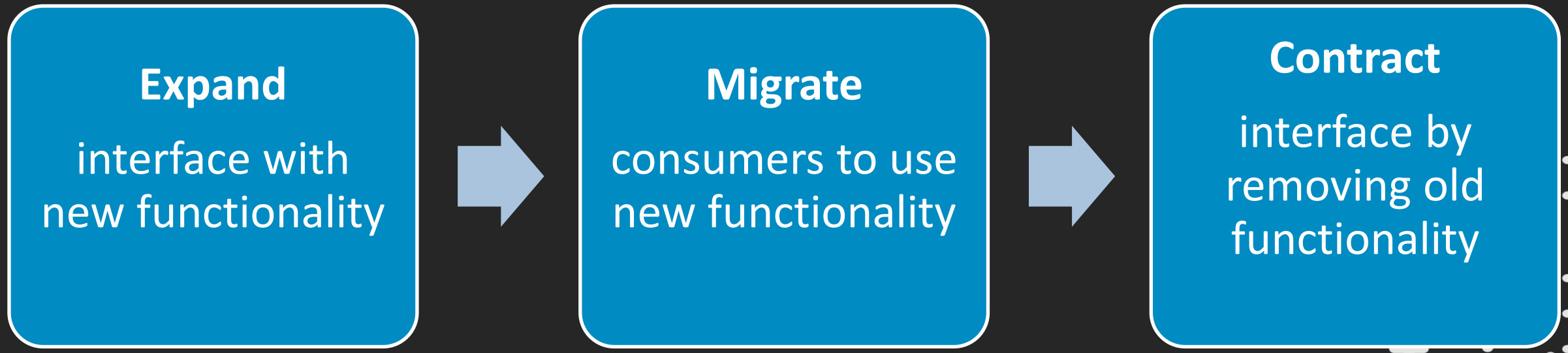
- Everybody works on master branch
- Every commit is potentially shippable

- Changes needs to be done in small batches
- Changes must be compatible
  - Service endpoints
  - Message contracts
  - Database schema changes
- New functionality is hidden until finished

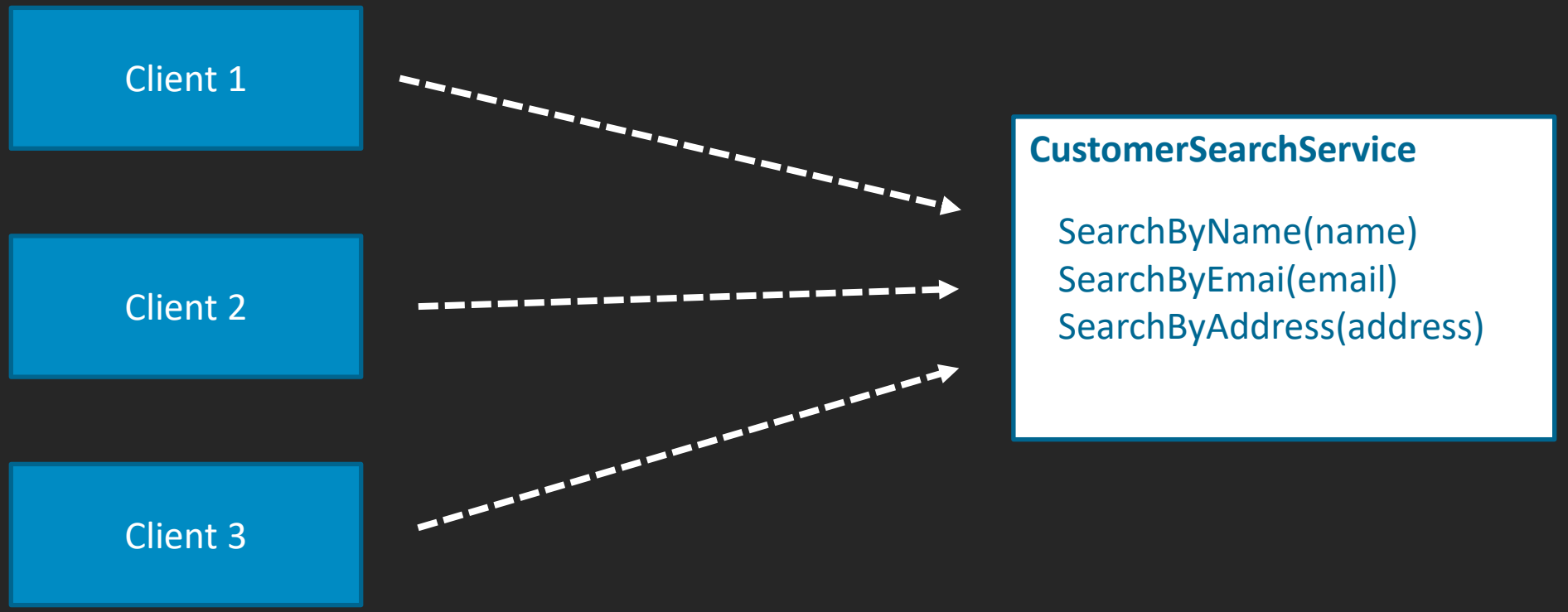


Breaking Changes

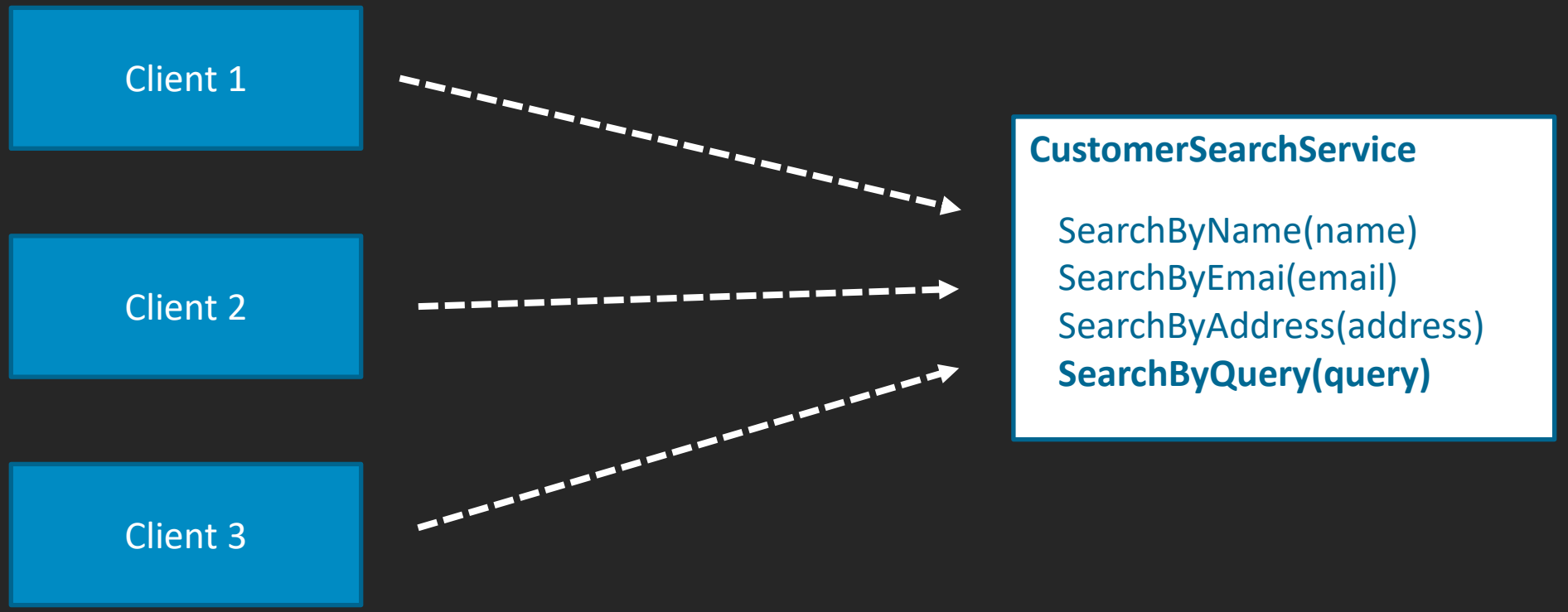
- Use Expand/Contract for rolling out breaking changes incrementally



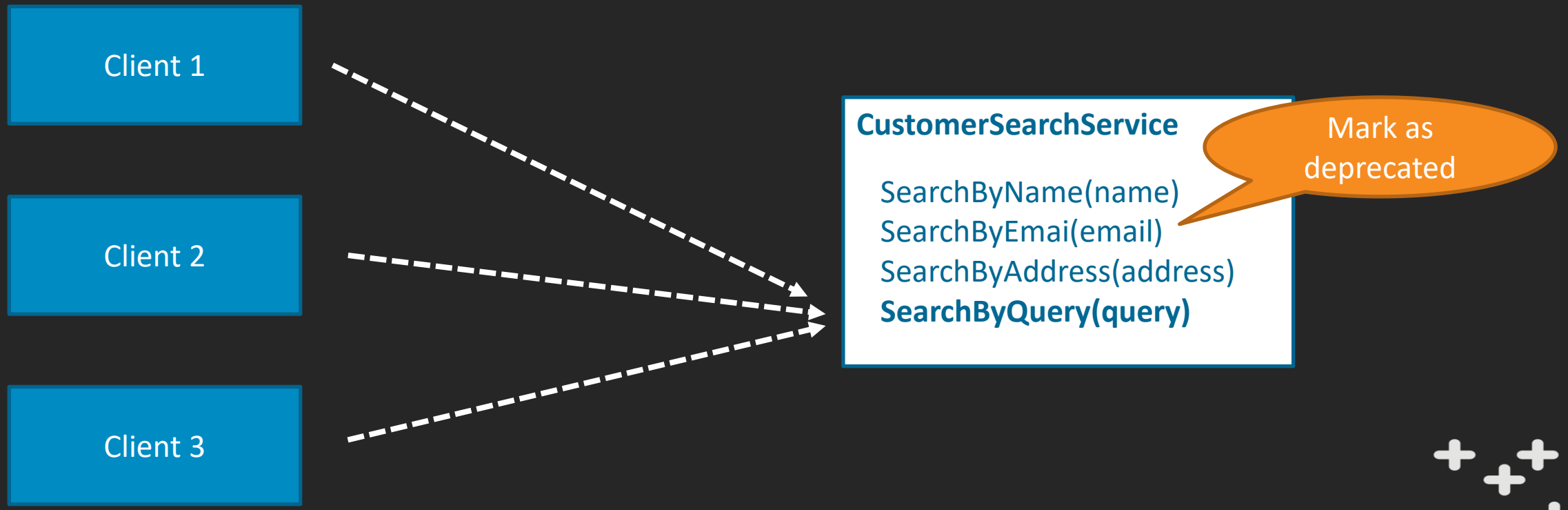
Expand/Contract



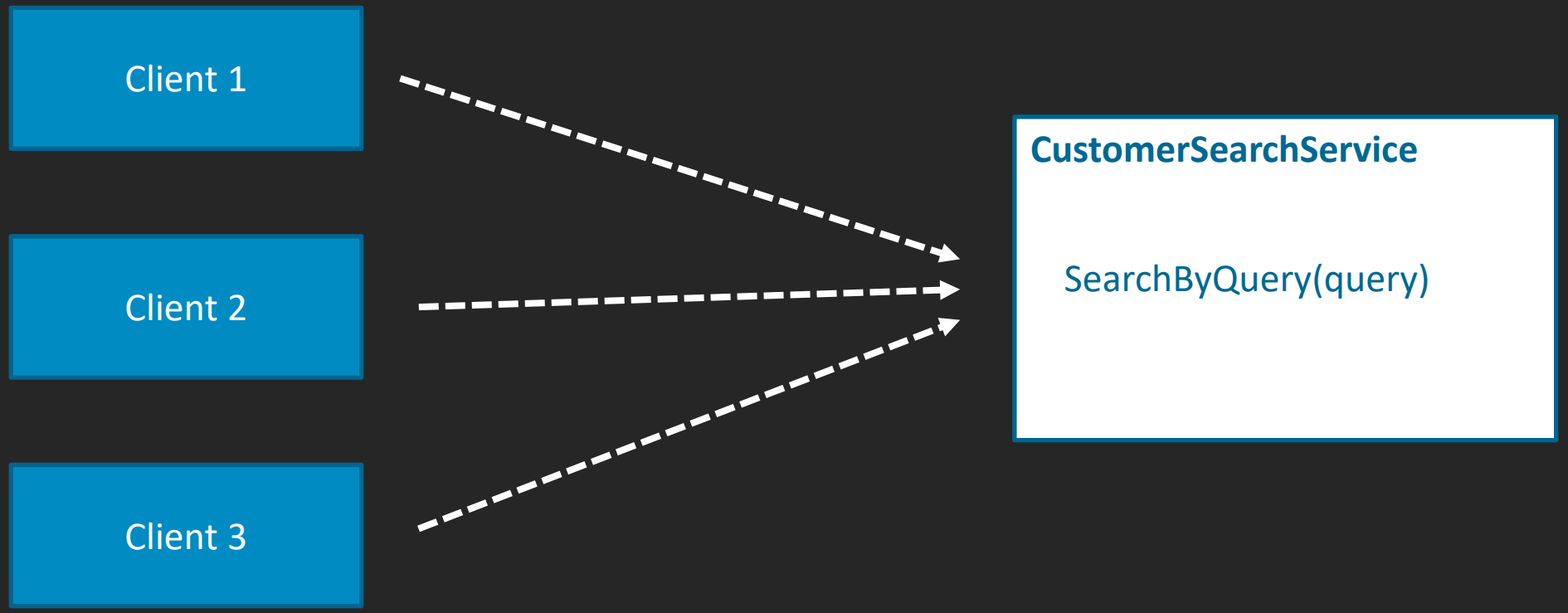
Expand


Client 1
Client 2
Client 3
CustomerSearchService
SearchByName(name)
SearchByEmai(email)
SearchByAddress(address)
SearchByQuery(query)

Migrate

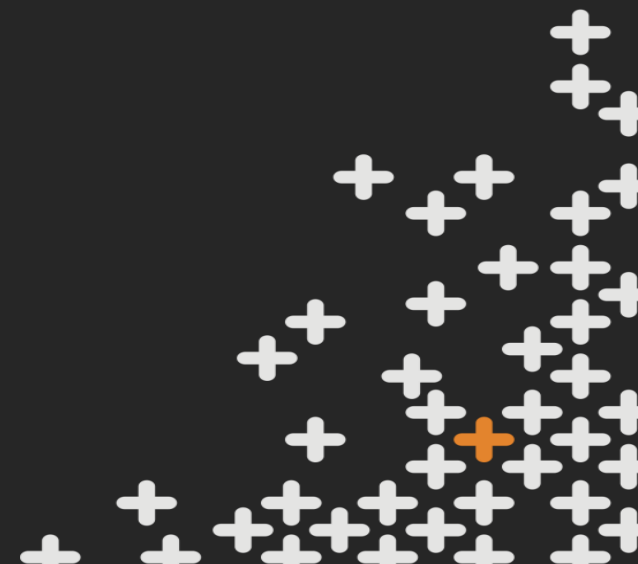

Client 1
Client 2
Client 3
CustomerSearchService
SearchByName(name)
SearchByEmai(email)
SearchByAddress(address)
SearchByQuery(query)
Mark as deprecated

Contract


Client 1
Client 2
Client 3
CustomerSearchService
SearchByQuery(query)

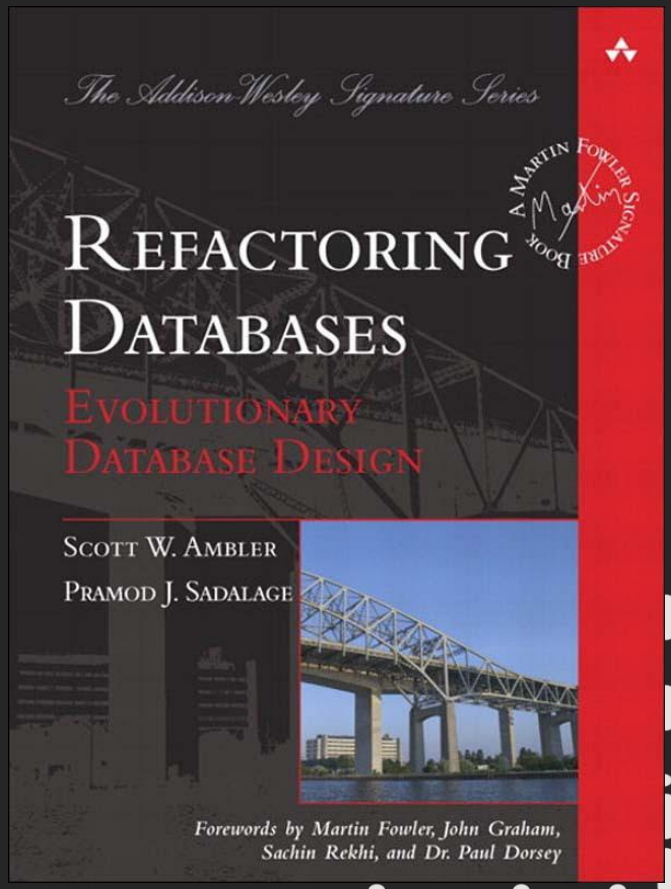
Changing Message/Schemas

- Expand schemas with non-breaking changes
  - Backwards compatible
- Avoid assumptions when reading schemas
  - ”Tolerant reader”
  - Forward compatible



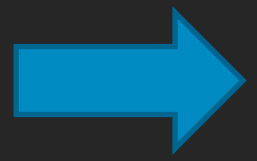
Incremental Database Changes

- Never change existing objects
- Break changes up in small increments
- Expand/Contract



Example – Rename column

Column	Type
Id	Guid
FirstName	Varchar(255)
LastName	Varchar(255)



Column	Type
Id	Guid
FirstName	Varchar(255)
Surname	Varchar(255)

```
public string GetLastname()
{
    return _lastName;
}

public void SetLastName(string lastName)
{
    _lastName = lastName;
}
```

Rename column – Expand

Column	Type
Id	Guid
FirstName	Varchar(255)
LastName	Varchar(255)
Surname	Varchar(255) **(NULL)**

```
public string GetLastname()
{
    return _surname ?? _lastName;
}

public void SetLastname(string surName)
{
    _lastName = surName;
    _surname = surName;
}
```

Rename column – Migrate

- Run migrations to populate new field in all records

Column	Type
Id	Guid
FirstName	Varchar(255)
LastName	Varchar(255)
Surname	Varchar(255) **(NULL)**

Rename column – Contract

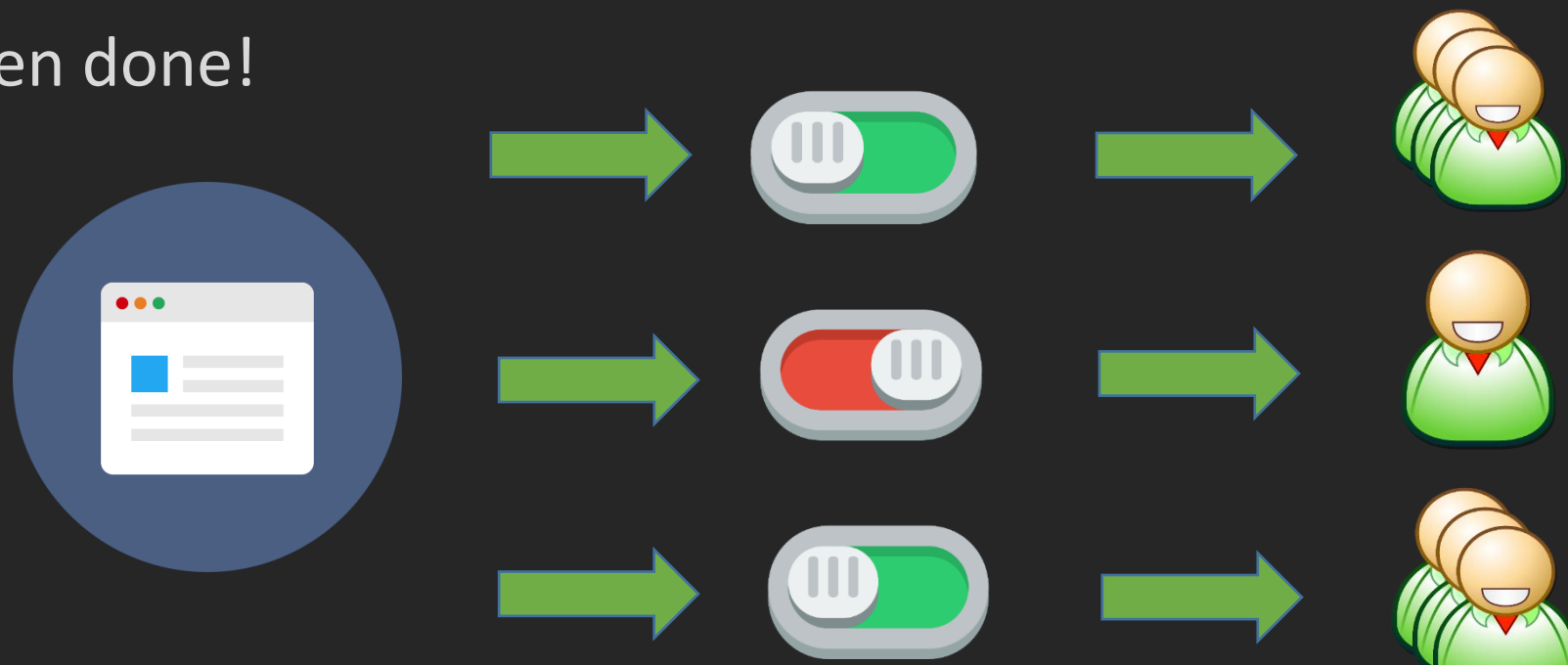
Column	Type
Id	Guid
FirstName	Varchar(255)
Surname	Varchar(255) **(NOT NULL)**

```
public string GetSurname()
{
    return _surname;
}

public void SetSurname(string surName)
{
    _surname = surName;
}
```

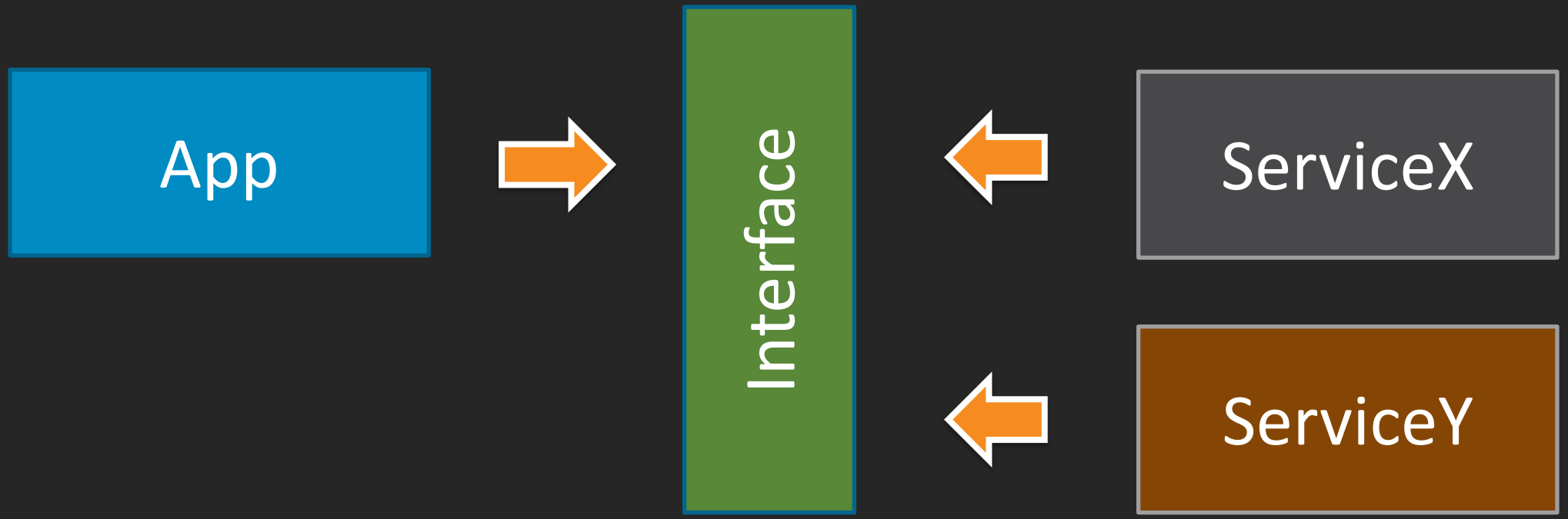
Release new functionality

- Use Feature Toggles
- Alternative to branching
- Continuously deploy unfinished features
- Remove when done!

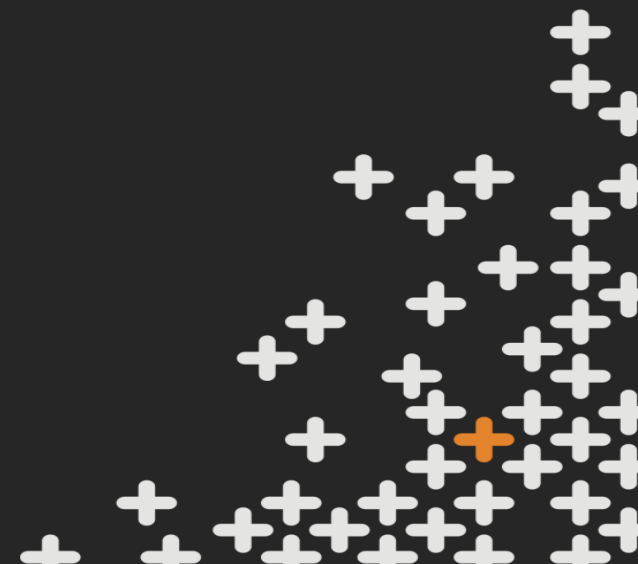


Implementing larger changes

Branch by Abstraction



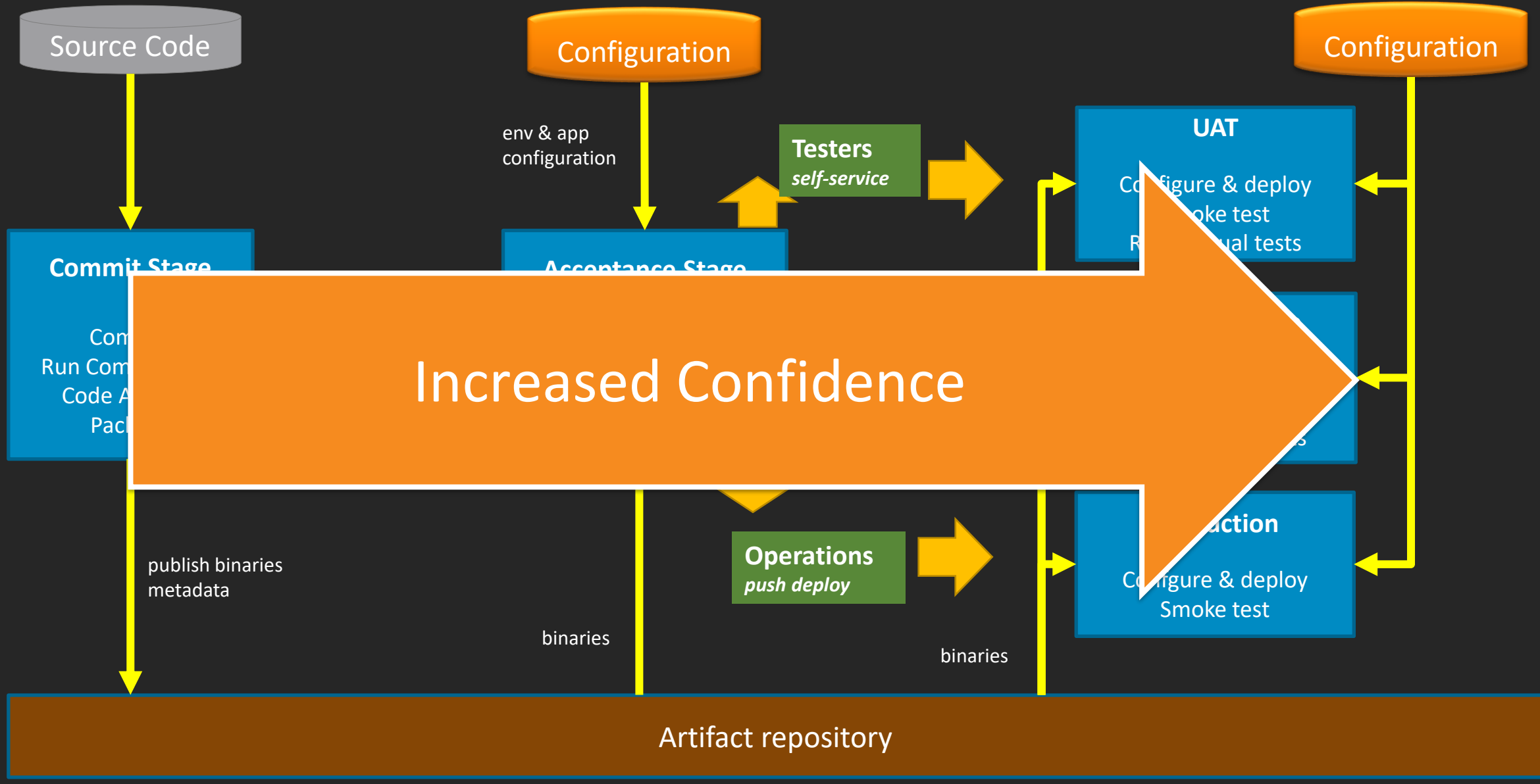
LOW RISK DEPLOYMENTS



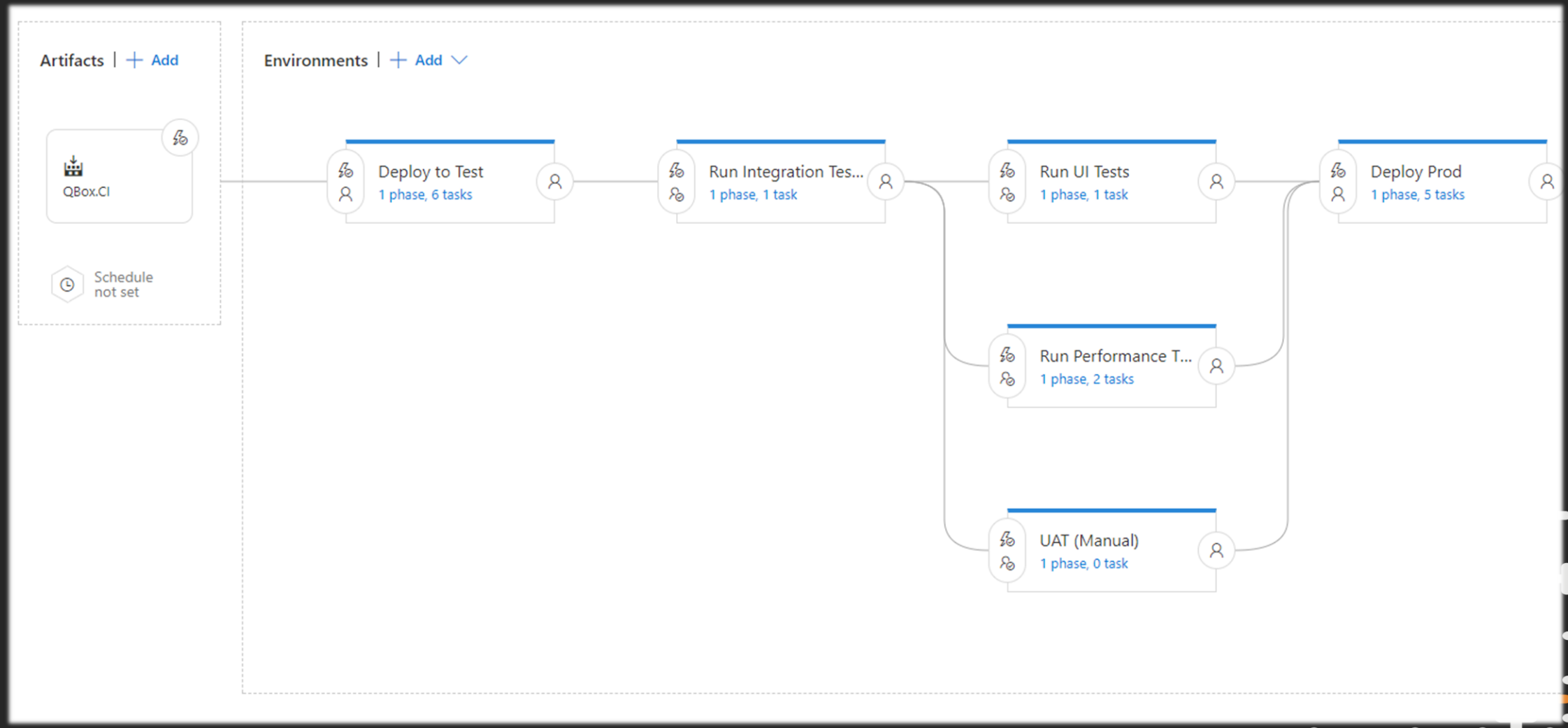
Low Risk Deployments

- Deployment pipeline for predictable releases
- Separate deployment from releases
- Separate app deployment from database migration
- Minimize downtime with Blue/Green deployments
- Use Canary Releases for incremental deployments

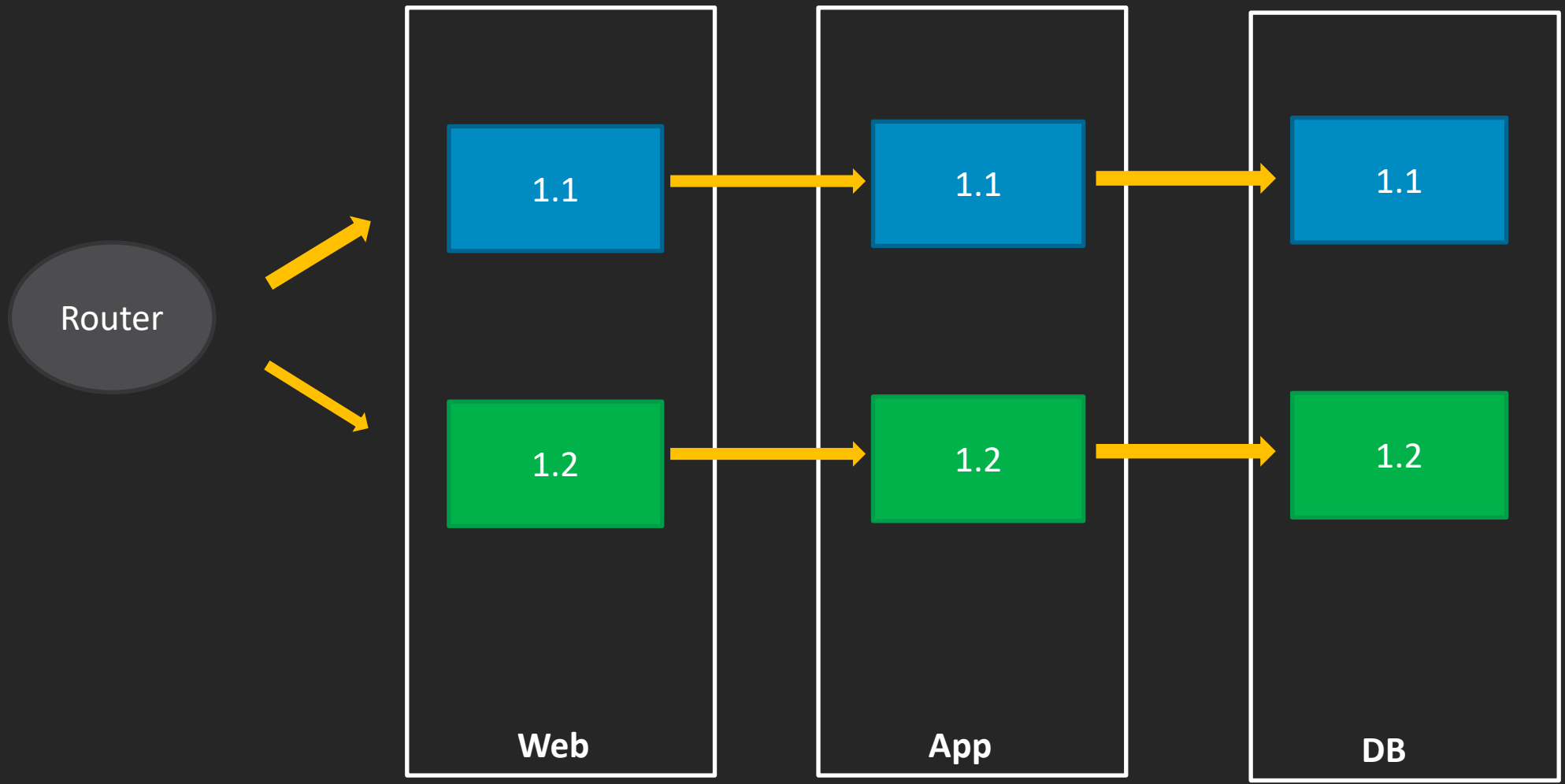
Deployment Pipeline



Deployment Pipeline in VSTS

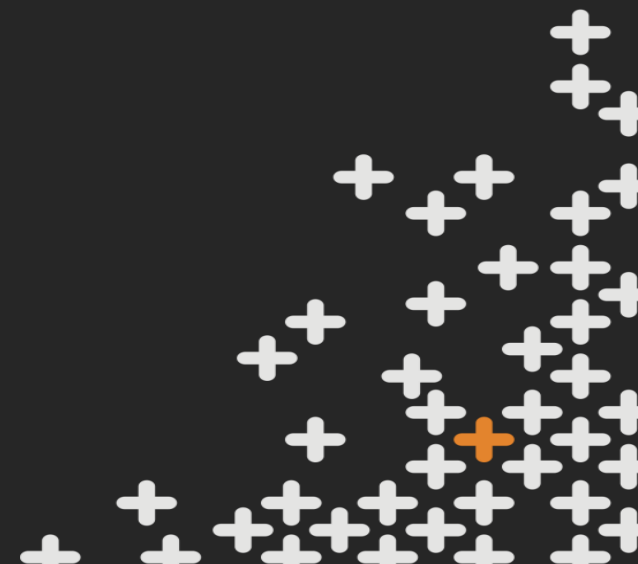


Blue Green Deployments



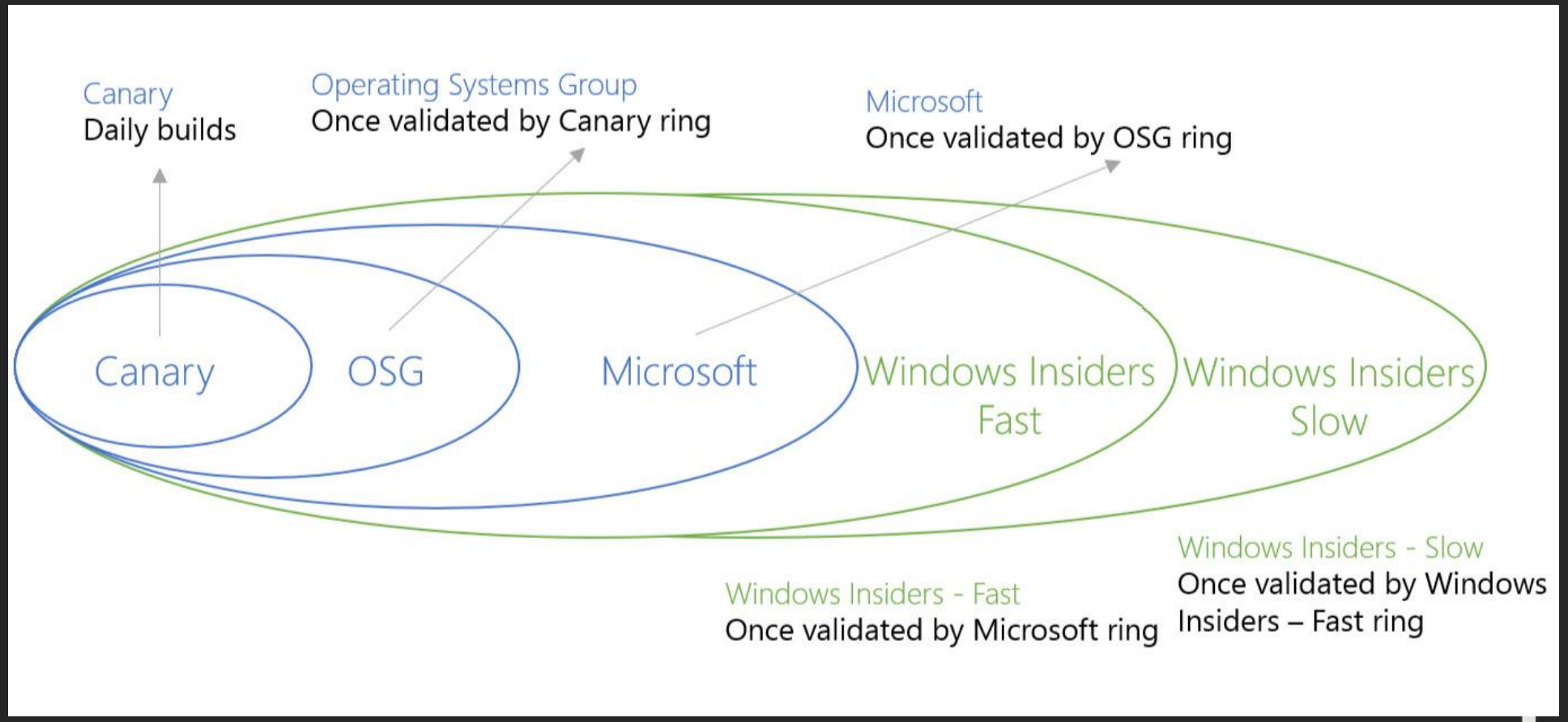
Implementing Blue Green deployments

- Implementation options
  - Azure Deployment Slots
  - Docker containers using clusters (Kubernetes, Swarm, DC/OS..)
  - Service Fabric
  - IIS (With ARR and Web farms)
- Warm up and test new version before switching



Canary Releases

- Roll out changes incrementally to groups of users



active SOLUTION

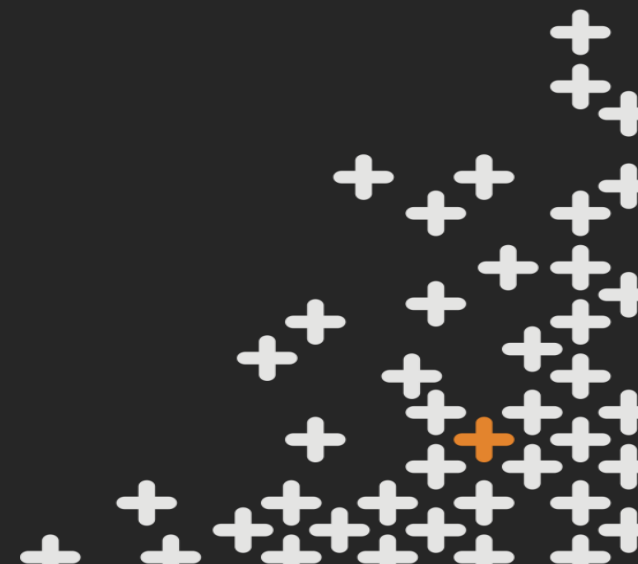
Tack!

Jakob Ehn

@jakobehn

jakob.ehn@activesolution.se

https://blog.ehn.nu

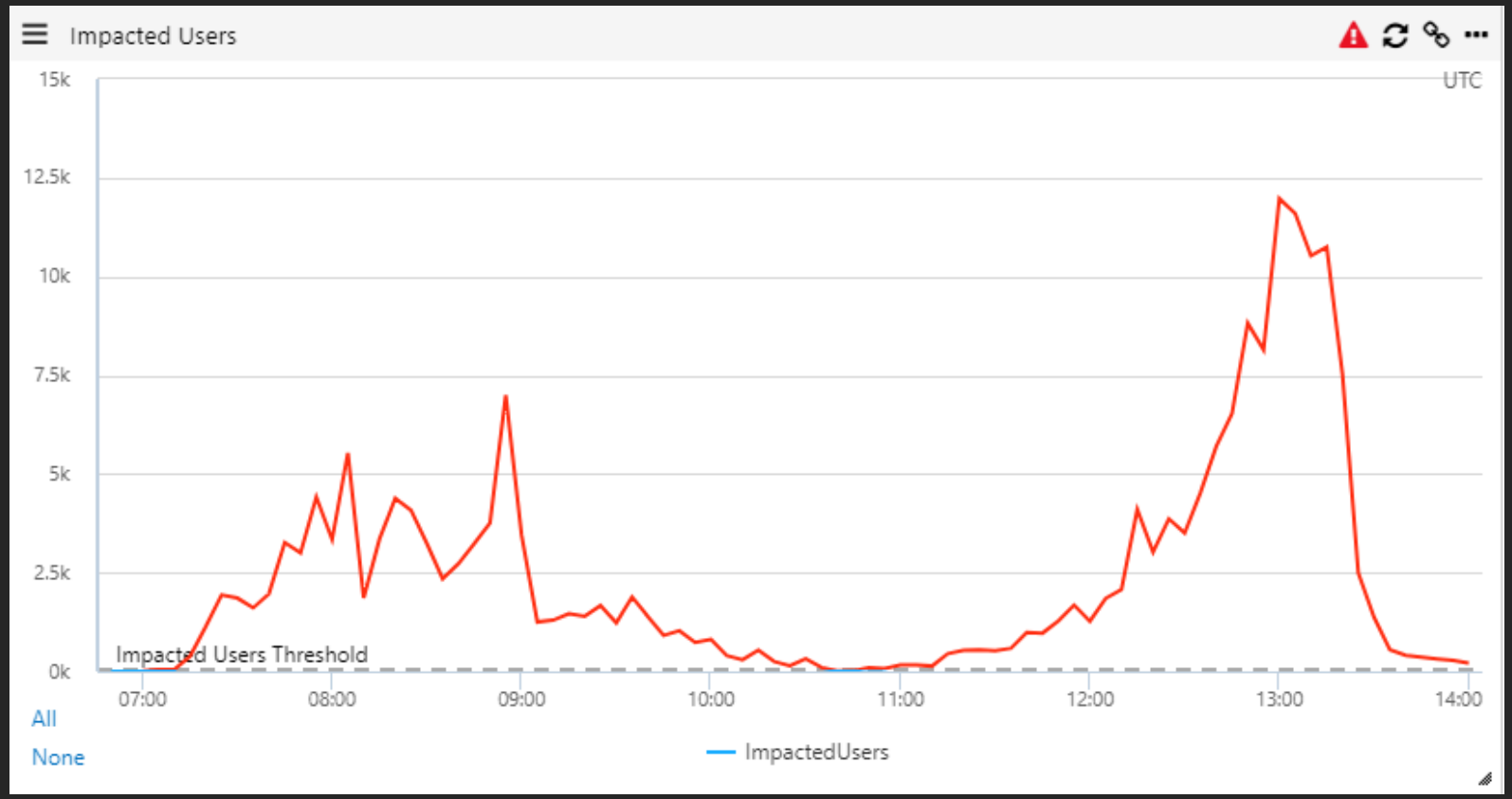


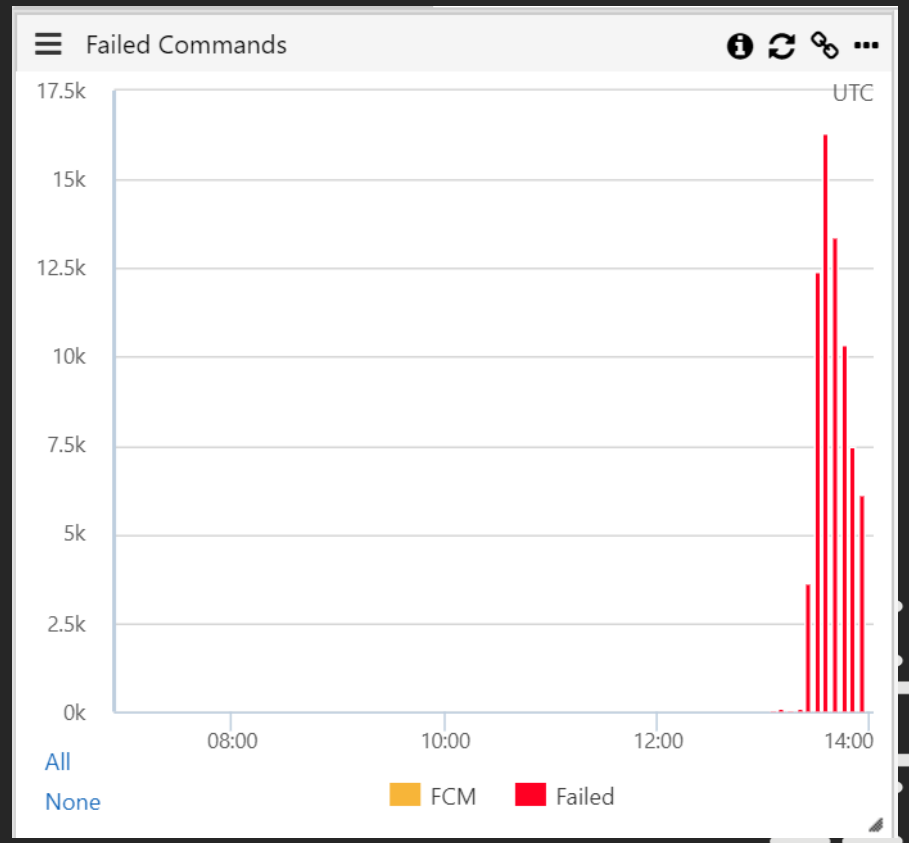
Visit us – Compete with Azure Cognitive Services and Tobii Eyetracking!

Thank you!

Please do not forget to evaluate the session before you leave by using our Lollipolls!

VSTS Postmortem 10/10





https://blogs.msdn.microsoft.com/vsoservice/?p=15276