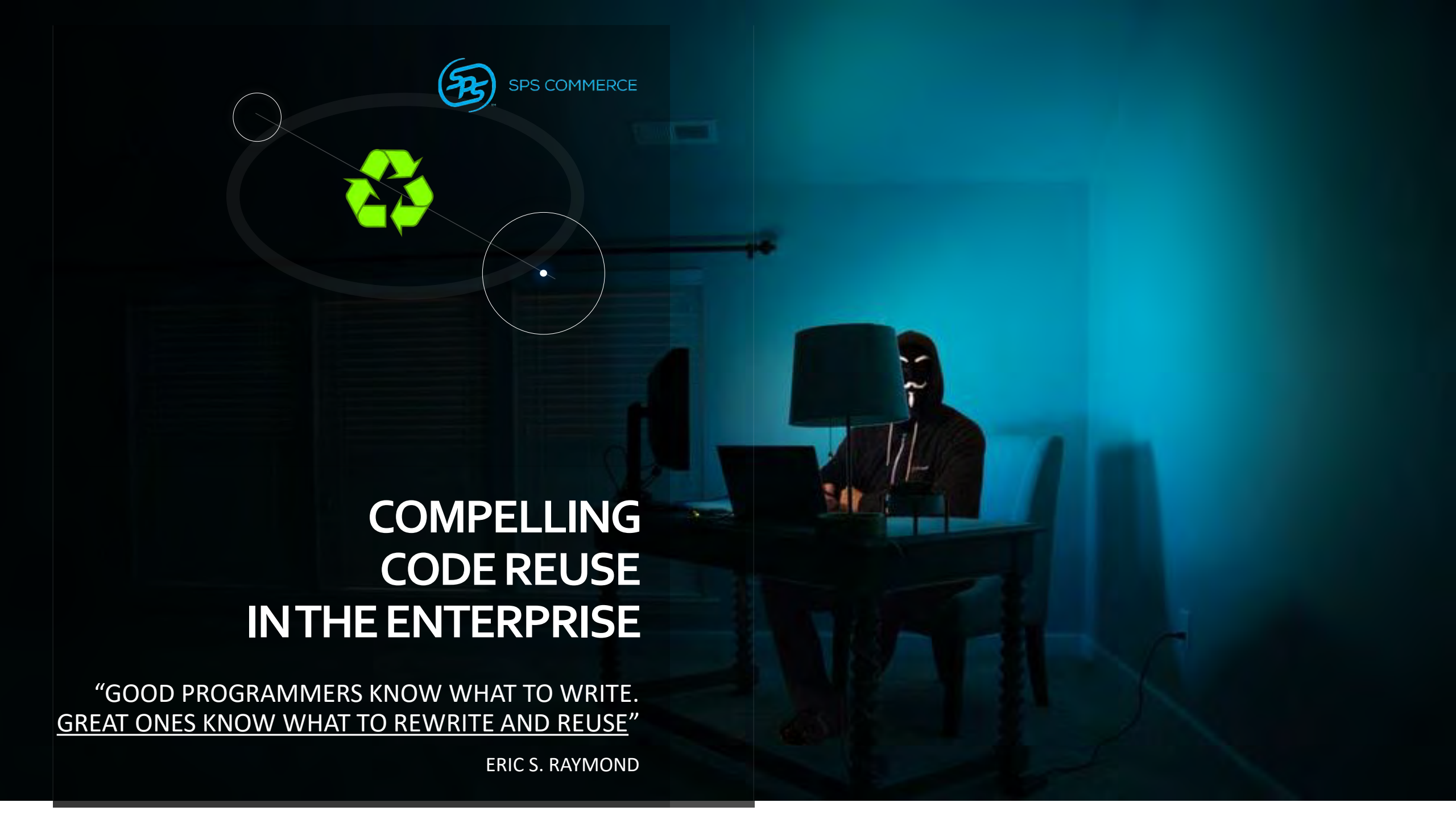
SPS COMMERCE



COMPELLING CODE REUSE IN THE ENTERPRISE

"GOOD PROGRAMMERS KNOW WHAT TO WRITE. GREAT ONES KNOW WHAT TO REWRITE AND REUSE"

ERIC S. RAYMOND

TRAVIS GOSSELIN

PRINCIPAL SOFTWARE ENGINEER

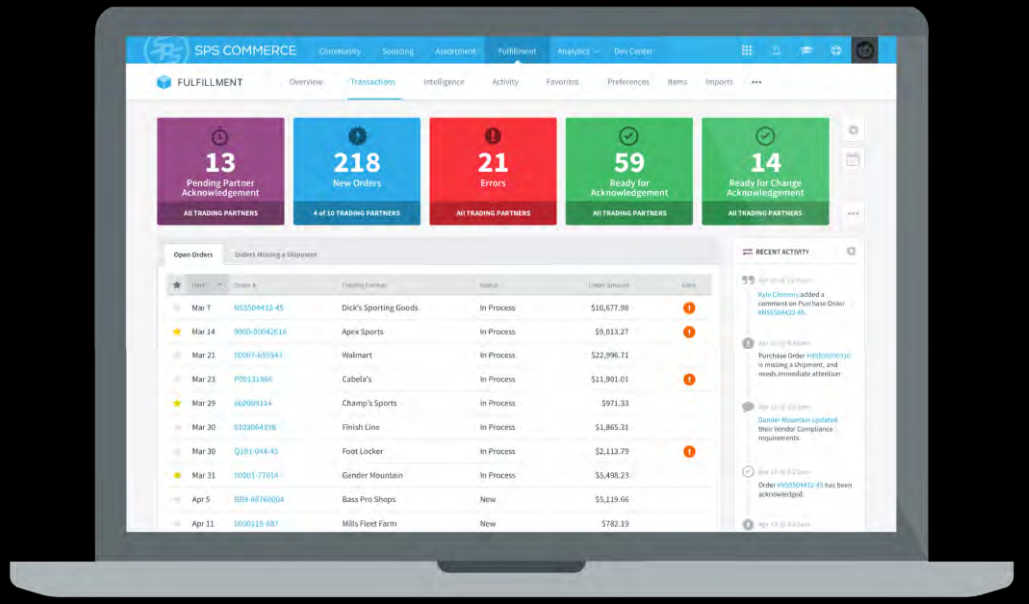
DEVELOPER EXPERIENCE

@TRAVISJGOSSELIN

www.travisgosselin.com

SPS COMMERCE

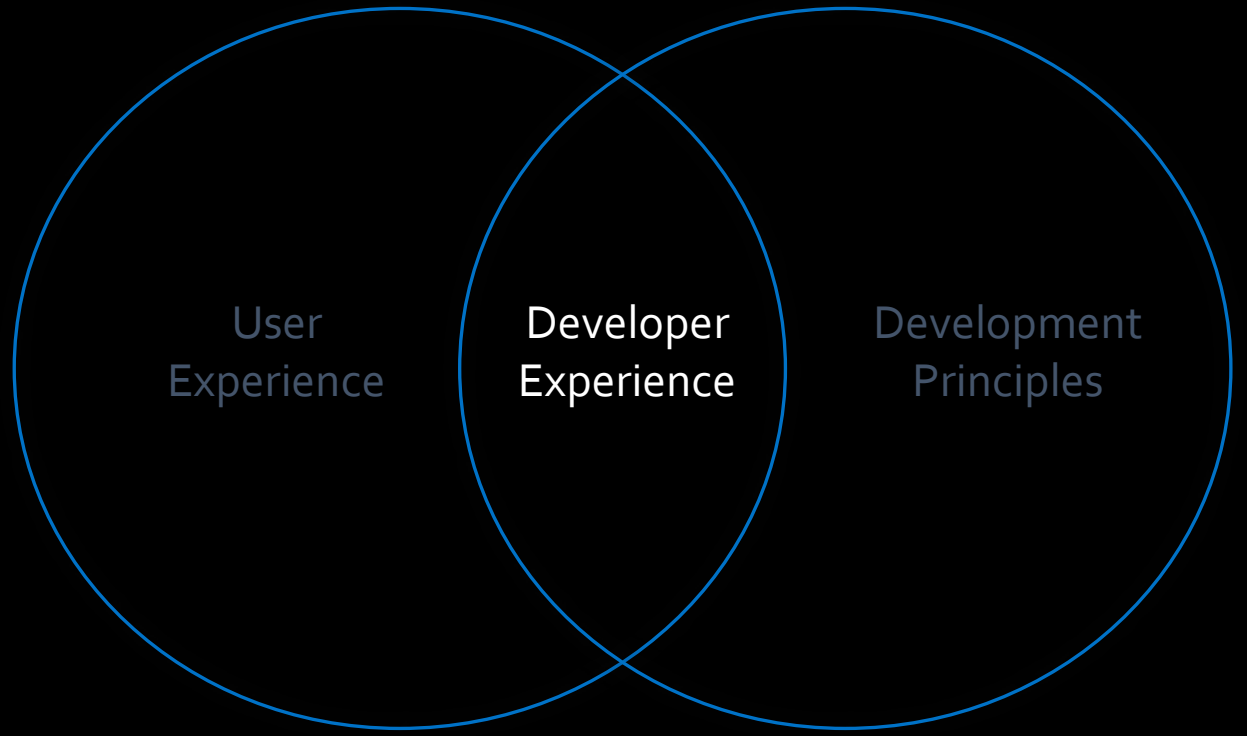
INFINITE RETAIL POWER™



> Developer Experience is the activity of studying, improving and optimizing how developers get their work done.
>
> theappslab.com (2017)

DEVELOPER EXPERIENCE

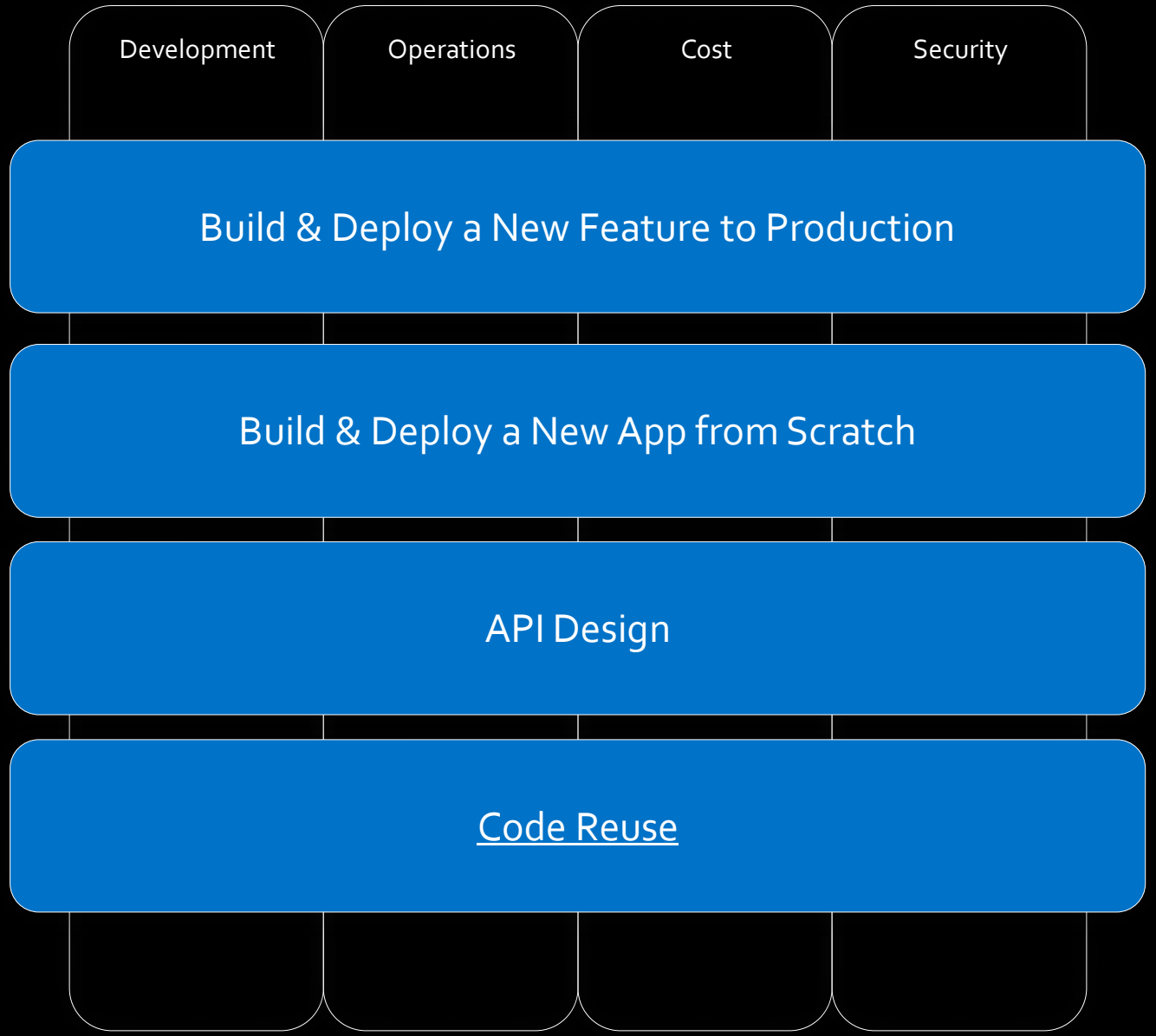
WHAT IS THAT…EXACTLY?



> Developers work in rainforests, not planned gardens.
>
> a16z.com

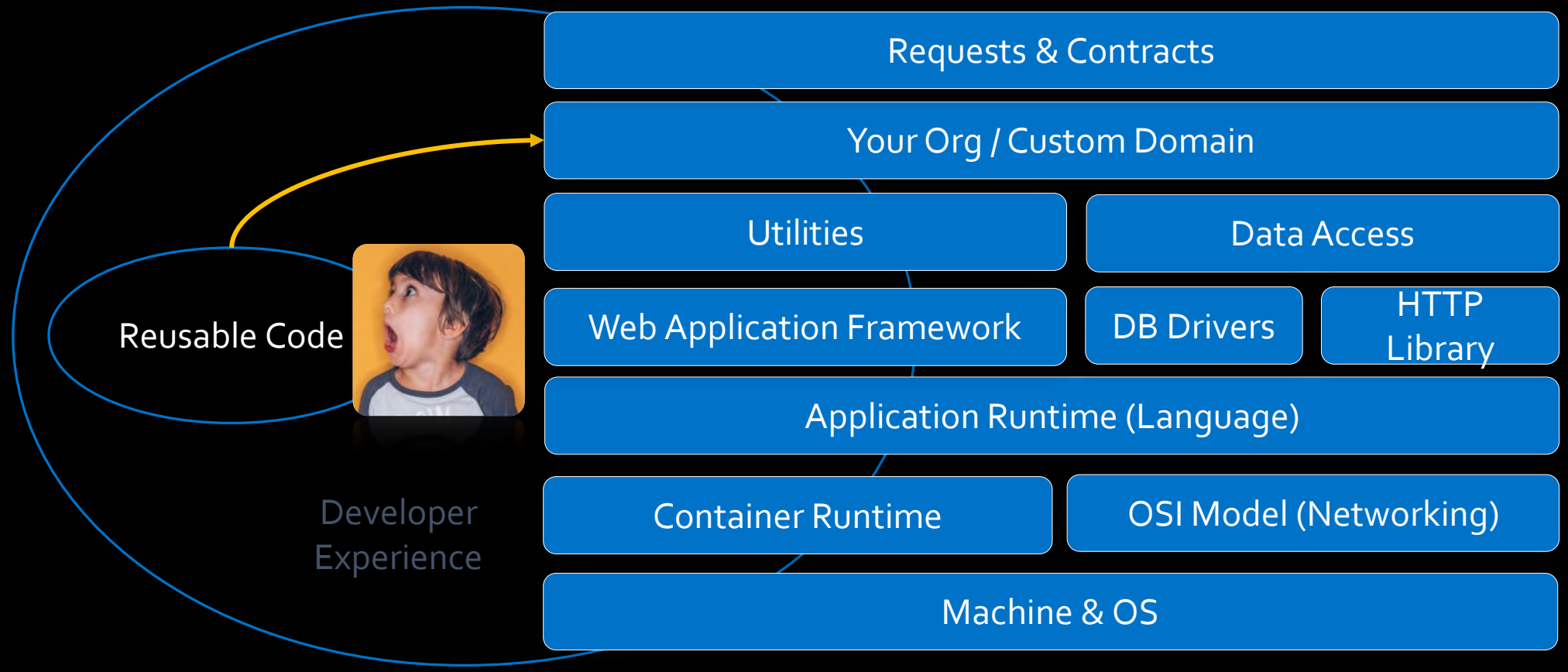
DEVELOPER EXPERIENCE: CAPABILITIES

IDENTIFIED HORIZONTAL FAST TRACKS TO BE CURATED FOR MAXIMUM PRODUCTIVITY.



DEVELOPER EXPERIENCE & CODE REUSE

> **Code Reuse …** is the act of recycling or repurposing code parts to improve existing or to create new software. Write it once, use it multiple times."
>
> filestack.com



COMPELLING CODE REUSE

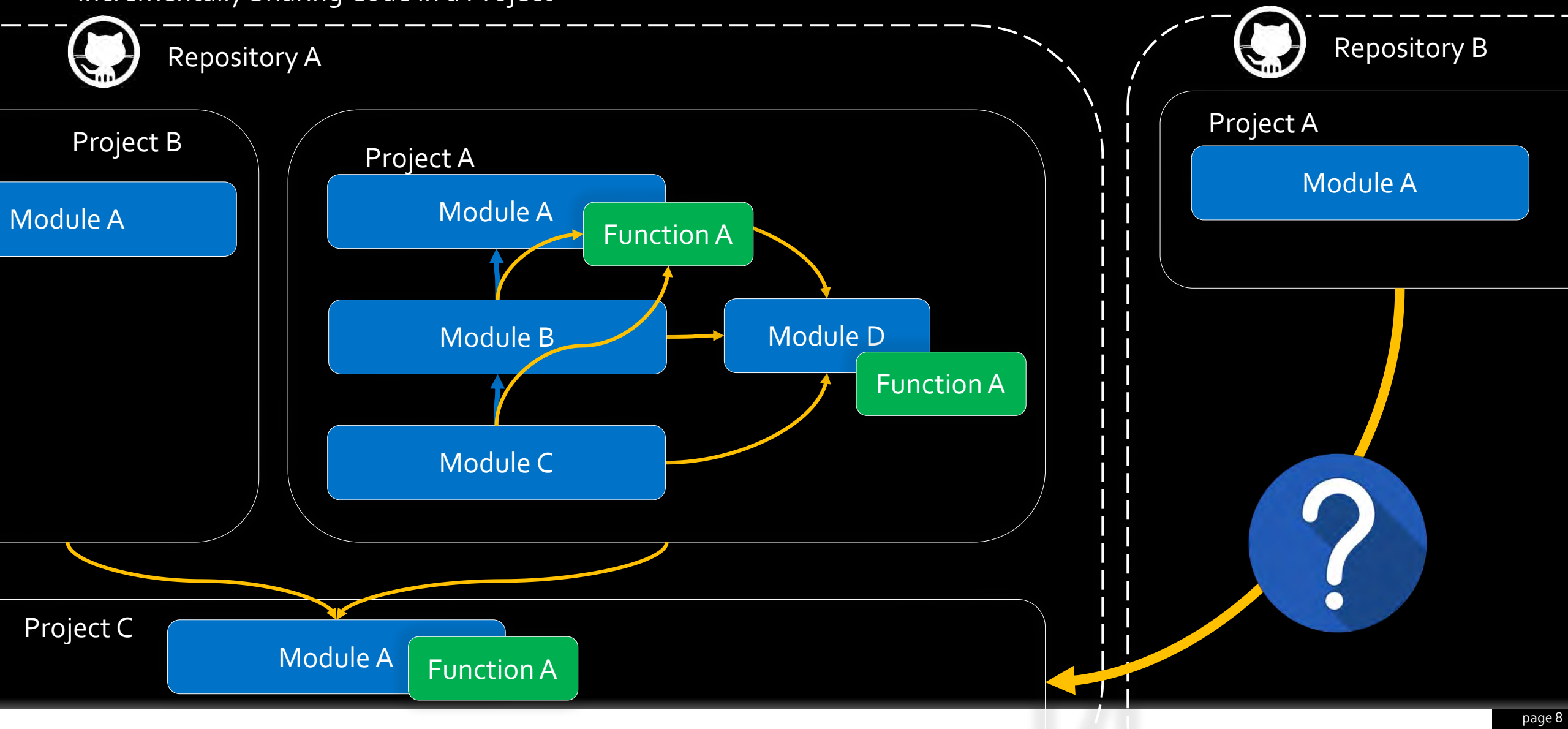
Definition

> *COMPELLING*
>
> 1: as to force or push toward a course of action.
>
> **Dictionary.com**

> *COMPELLING*
>
> 2: having a powerful and irresistible effect; requiring acute admiration, attention or respect.
>
> **Dictionary.com**

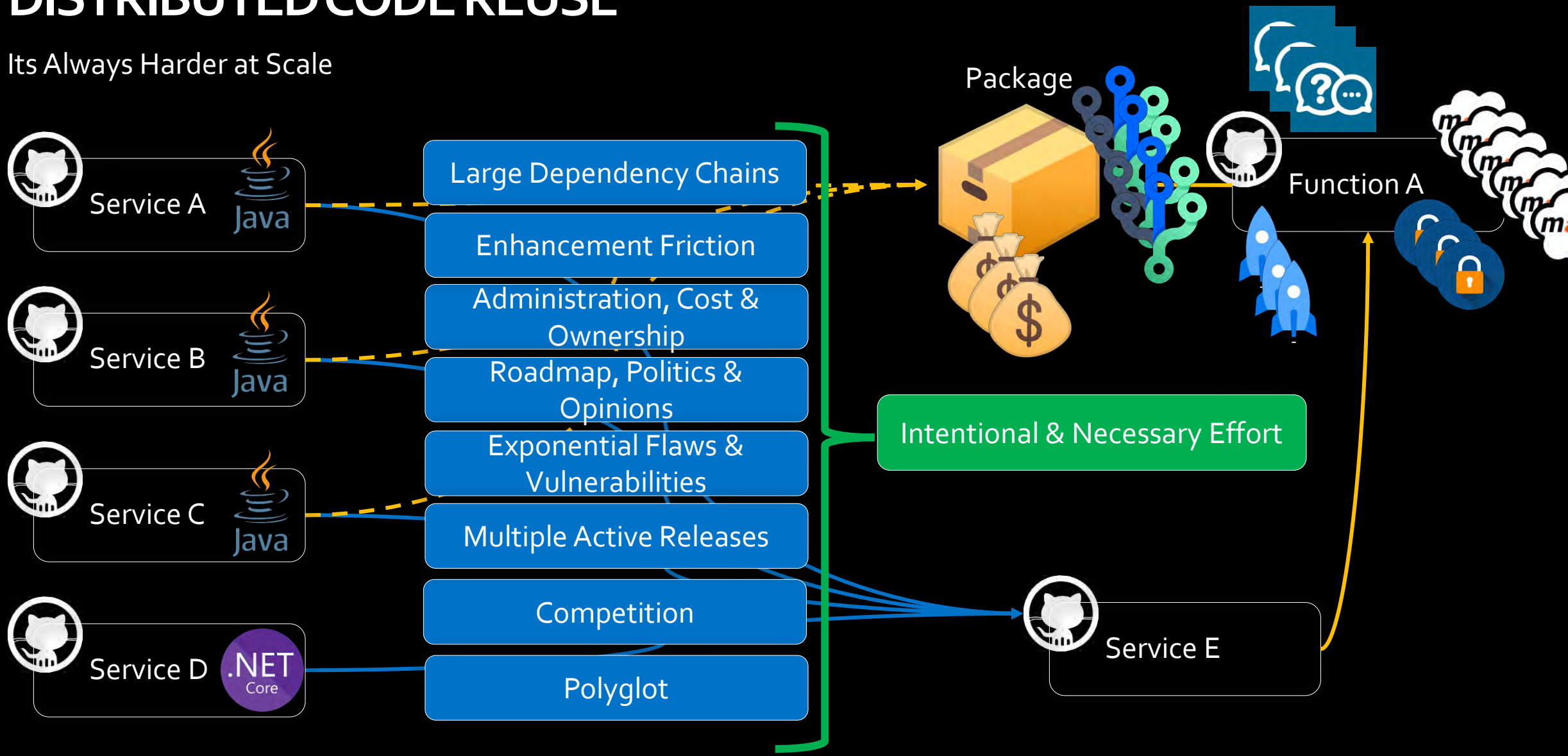
THEORITICAL CODE REUSE

Incrementally Sharing Code in a Project



DISTRIBUTED CODE REUSE

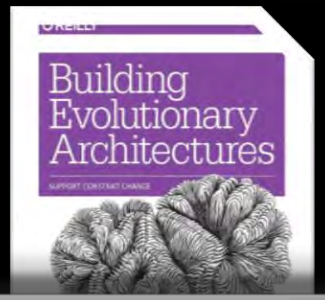
Its Always Harder at Scale



NO CODE SHARING IN MICROSERVICES

Myth & Reality

> Microservices eschew code reuse, adopting the philosophy of prefer duplication to coupling: reuse implies coupling, and microservices architectures are extremely decoupled.
>
> **Building Evolutionary Architecture**

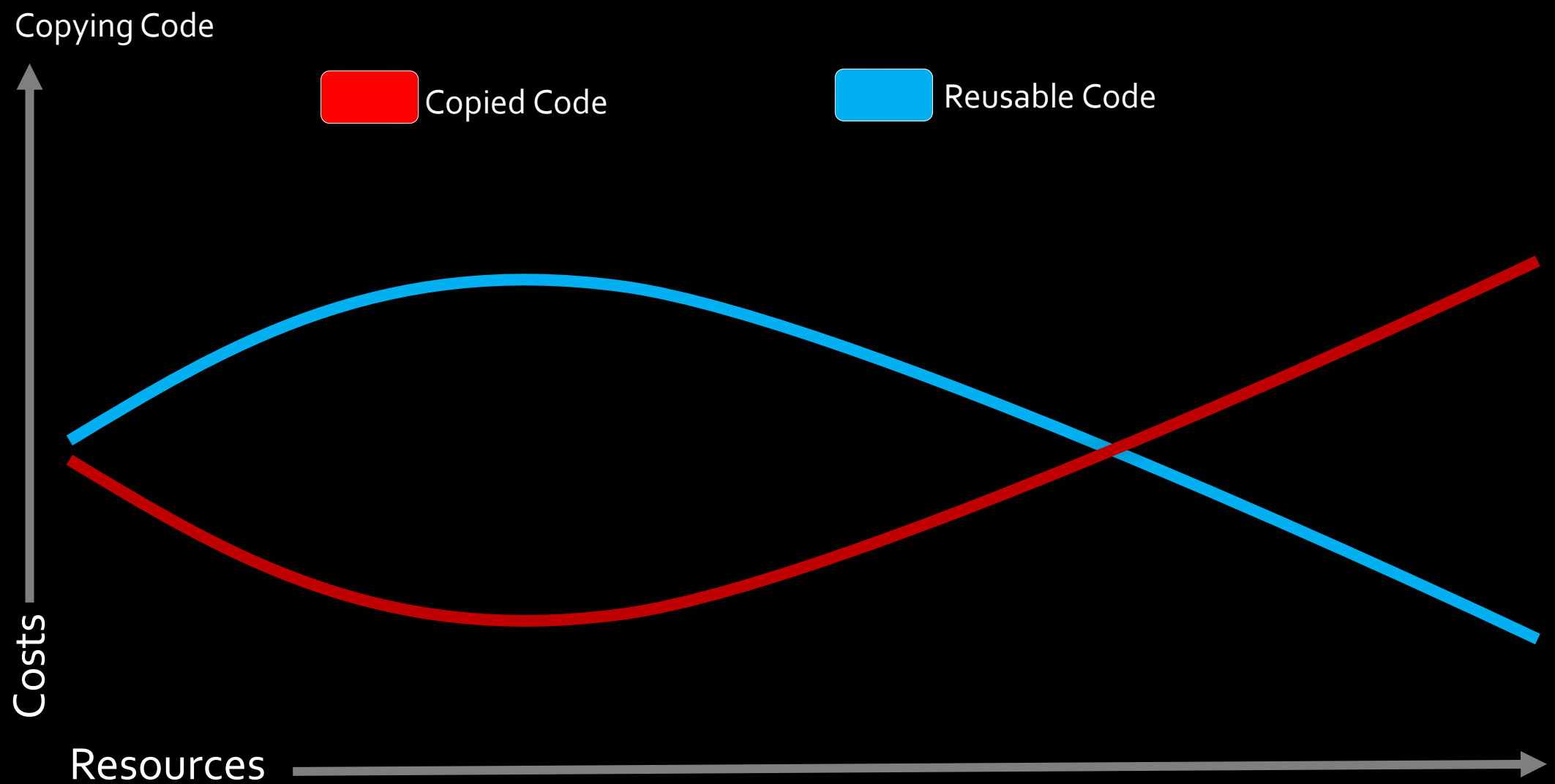


> Code reuse can be an asset but also a **potential liability.** Make sure the coupling points introduced in your code don't conflict with other goals in the architecture.
>
> **Building Evolutionary Architecture**

Duplicate?

Reuse?

DIMINISHING RETURNS

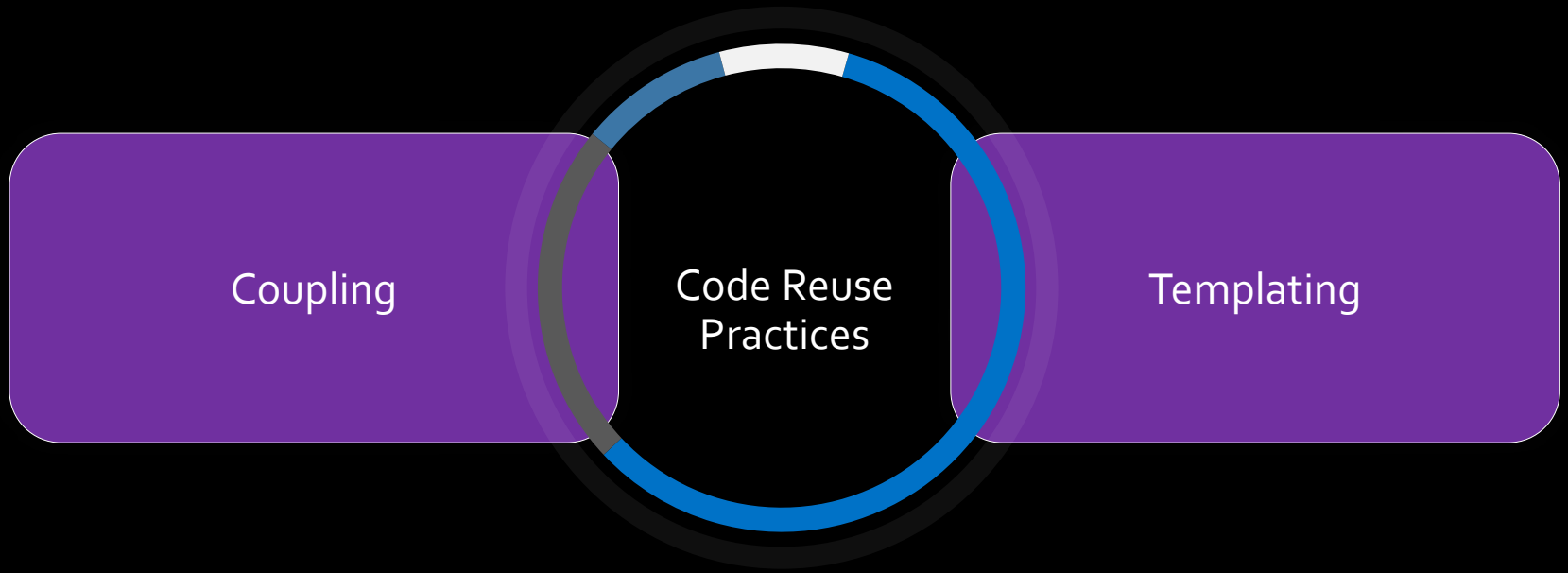


Site Pen – The Law of Diminishing Returns

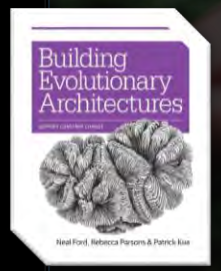
ARCHITECTING CODE REUSE

Libraries

How do we mature our code reuse practices?



> Coupling is the degree of interdependence between software modules:
>
> - A measure of how closely connected two routines or modules are;
> - The strength of the relationships between modules.
>
> **Wikipedia**



COUPLING

"HOW THE PIECES OF THE ARCHITECTURE CONNECT AND RELY ON ONE ANOTHER"

Reuse

Use it more than once from a single code base.

Duplicate

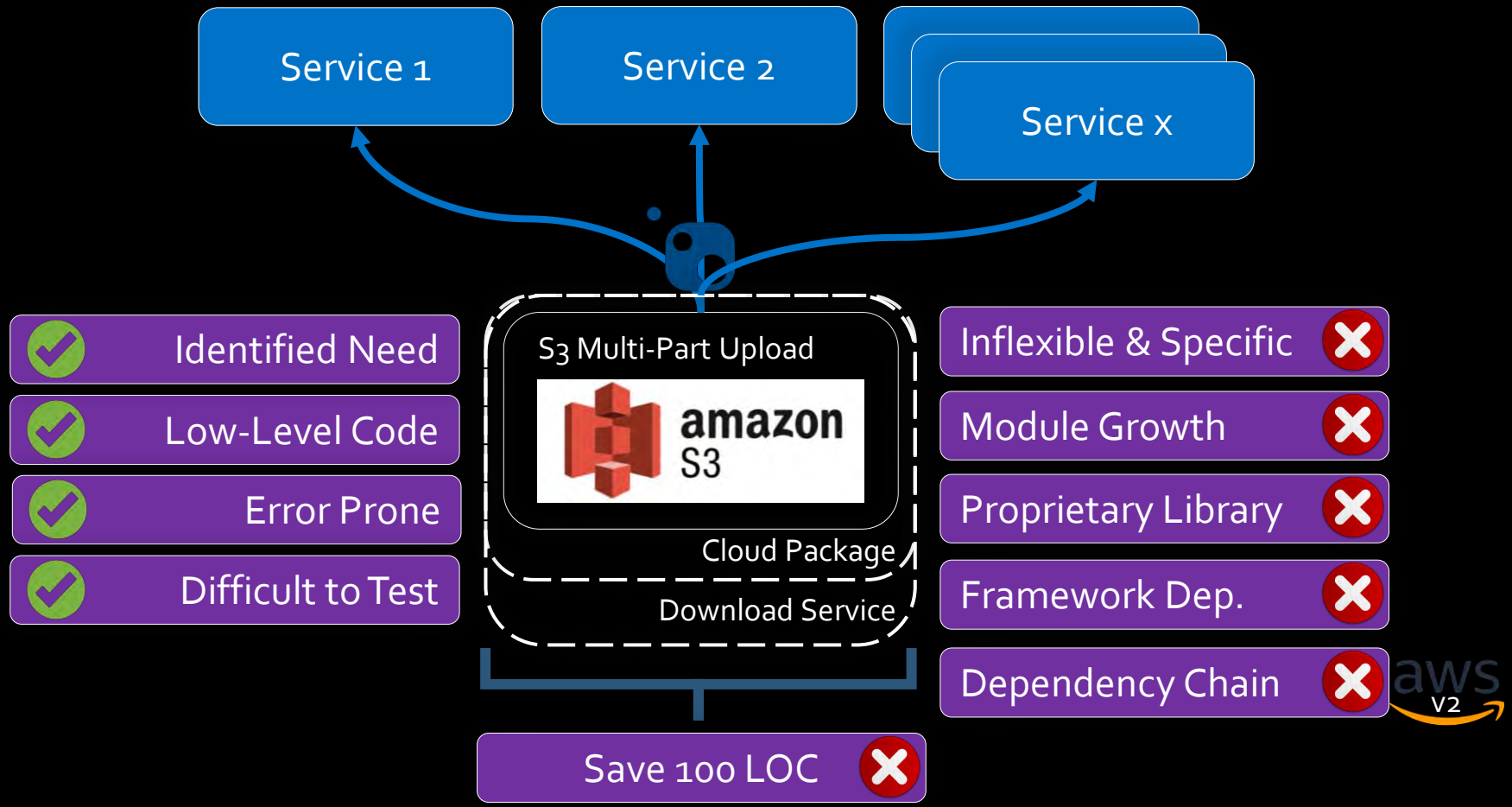
Use it more than once from multiple code basis.

Reference

Code Knowledgebases & Examples

COUPLING

Code Reuse Example



COUPLING

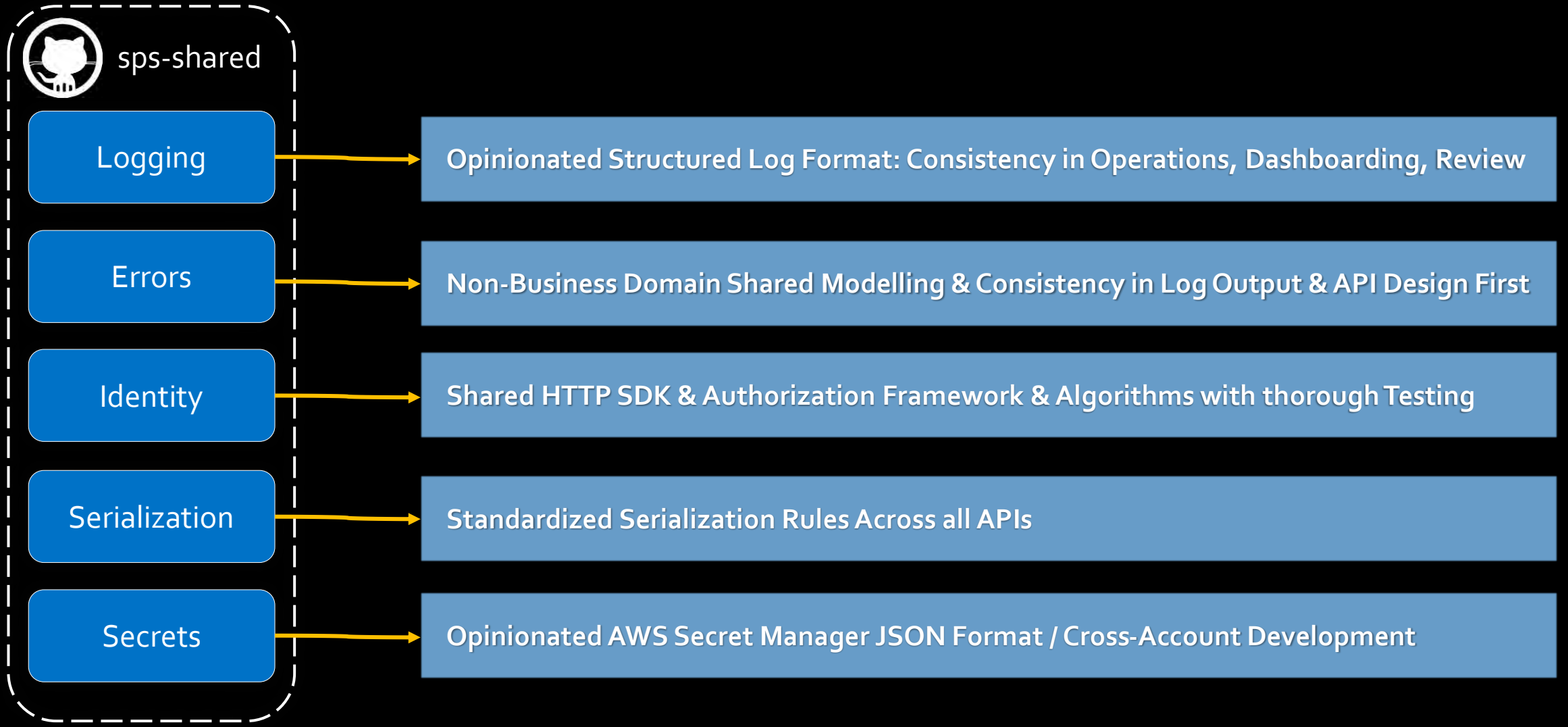
When to Reuse?

Technical & Cross Functional Concerns

- Support Code
- Authentication & Authorization
- Standardized Configuration and Platform Features
- Operational Opinions like Logging & Monitoring
- HTTP Client SDKs / Wrappers
- Error Handling & Validation
- Serialization

COUPLING

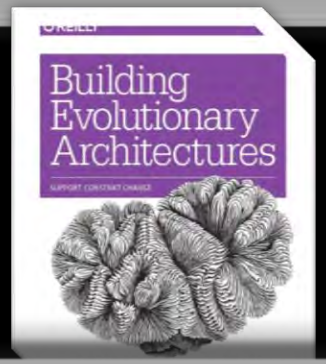
Reuse at SPS Commerce



COUPLING

Appropriate Coupling

> Dimensions of the architecture [that] should be coupled to provide maximum benefit with minimal overhead and cost.
>
> **Building Evolutionary Architectures**



> The more reusable code is, the less usable it is.
>
> **Building Evolutionary Architectures**

Consistent Logging Format

Operator Usage

Reusable Dashboards

Feature Flagging

Flag Keys

Flag Cleanup

User Context

Feature Flag Library

- Keys: ("UseEmailProvider")
- User Context ("Id", "Department")

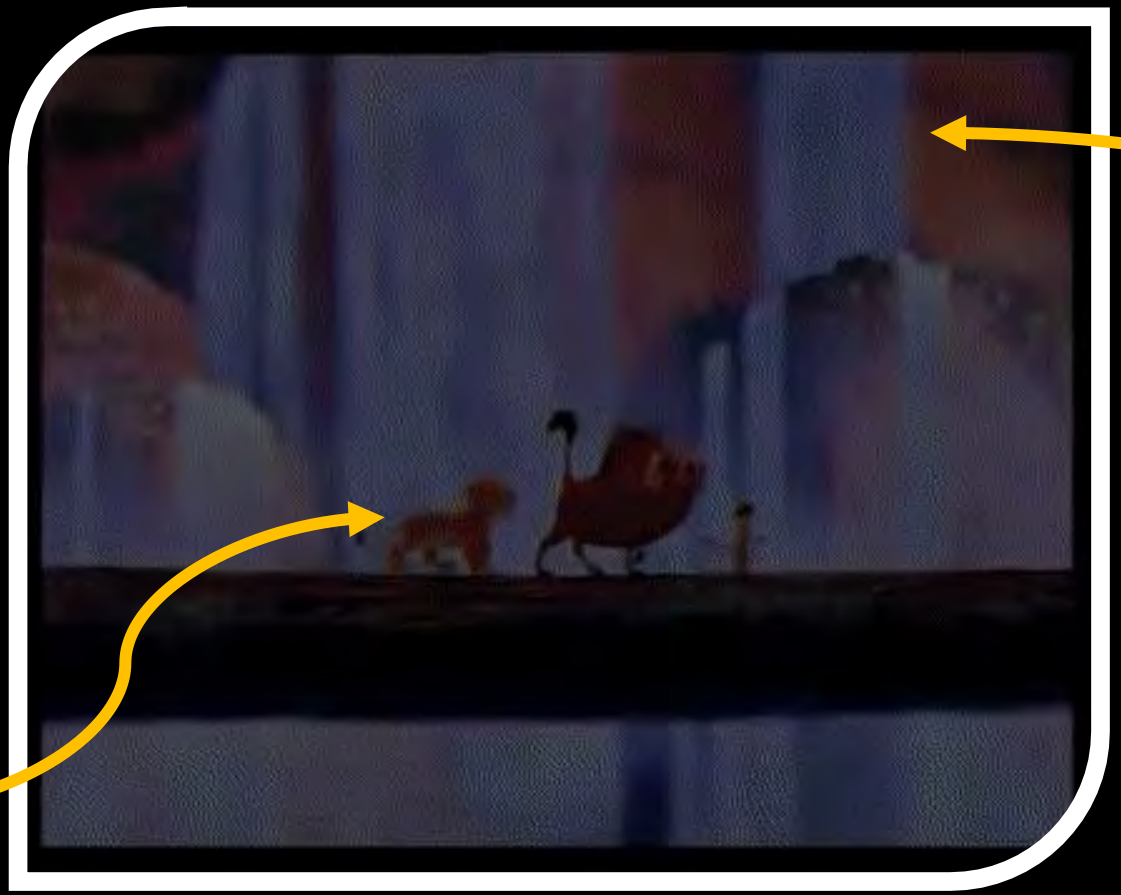
Service 1

Service 2

COUPLING

Traditional Animation Process – Duplicating vs Reuse

Duplicate



Reuse

Behind the Scenes of 'The Lion King' – The Animators Speaks

COUPLING

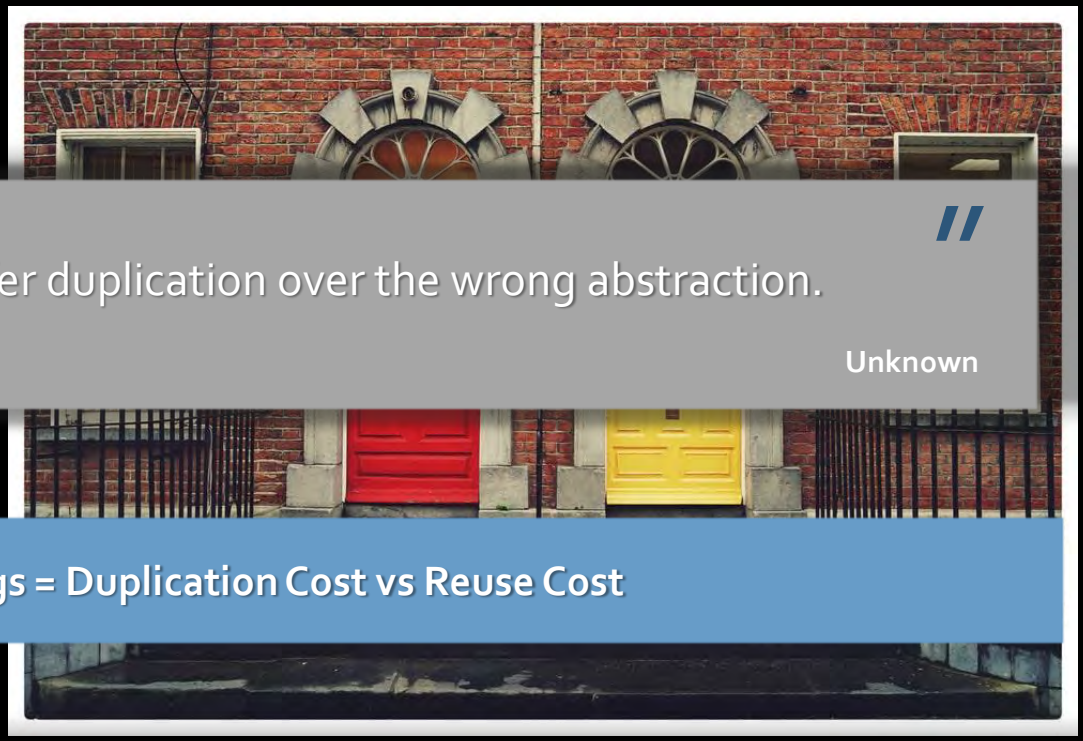
Just Copy It!

Incorrect Abstraction
Duplication

Low Overhead to
Savings Ratio

Low Feasibility

Diversified Opinions



> Prefer duplication over the wrong abstraction.
>
> **Unknown**

Reuse Savings = Duplication Cost vs Reuse Cost

> It can be better to copy a little code than to pull in a big library for one function. Dependency hygiene trumps code reuse.
>
> **Rob Pike**


Snippets
Library: MyOrg.Module.Utilities
COUPLING
SHARING UTILITY LIBRARIES
Content:
• Cooking Kraft Dinner in the Microwave
• Building Custom Furniture

TEMPLATING

ELIMINATING BOILERPLATE FOR ENTIRE PROJECTS

Project Seeds

Service Templates

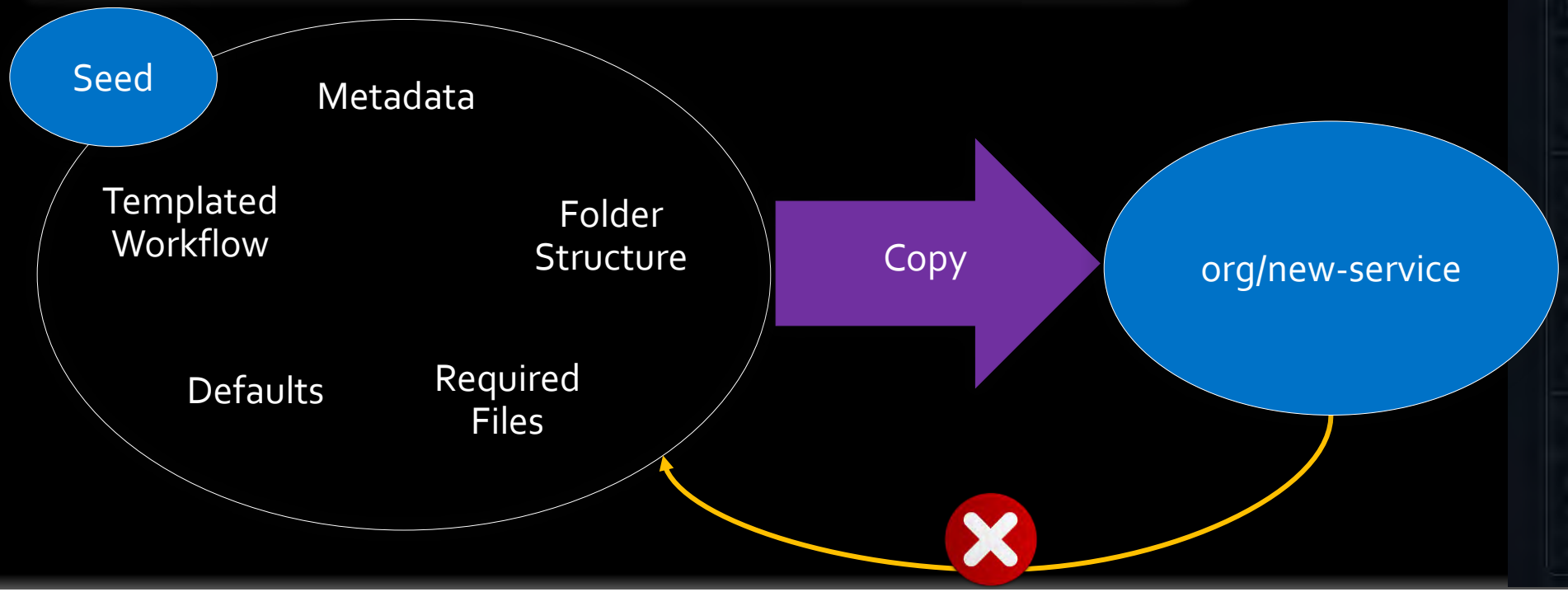
Service Chassis

TEMPLATING

Value ★ | Cost Low

Project Seeds

> "Project Seed: Very high-level reference point for starting a new application that typically provides standardized folder structure along with SDLC workflow via templated files."



Repository Template

2.5.5 | 1 branch | 221 tags

travisgosselin Add in dashboard and ale

- .github
- .vscode
- demo
- deployment
- server-gate
- src
- tests
- .bdp
- .dockerignore
- .editorconfig
- .gitignore
- CODEOWNERS
- CONTRIBUTING.md
- Dockerfile
- README.md
- Spsc.Vsts.Gates.sln

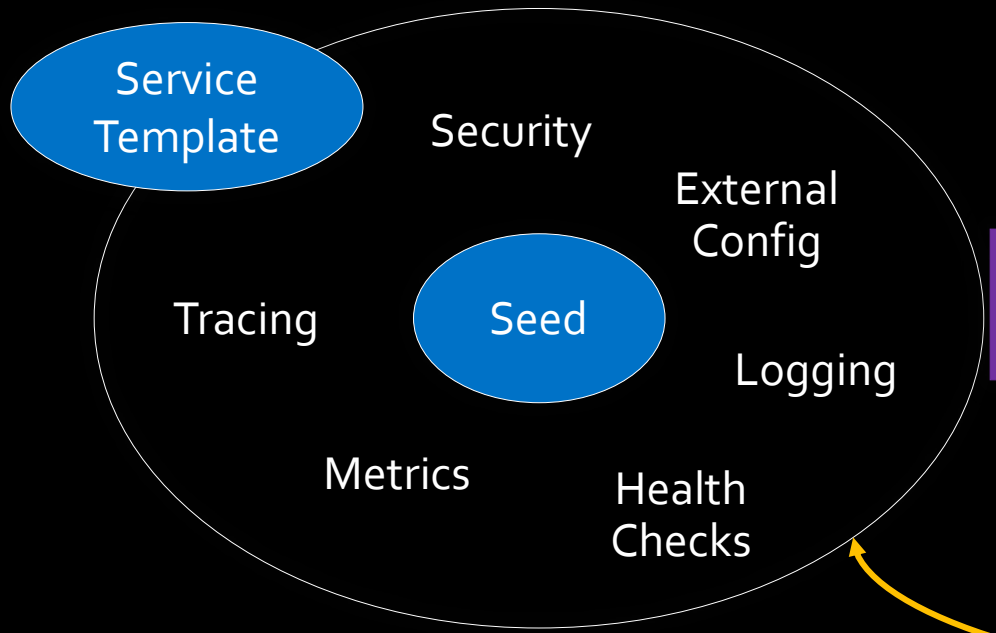
TEMPLATING

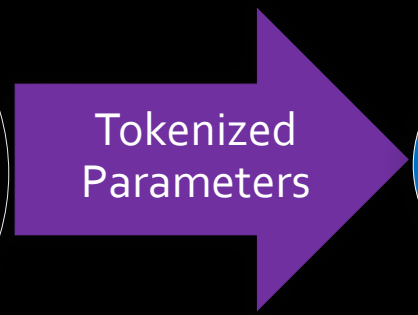
Value ★★

Cost Medium (Increasing Over Time)

Service Template

> "Service Template: Opinionated reference for specific application and language types that reduces boilerplate setup and provides consistency on cross-cutting concerns."



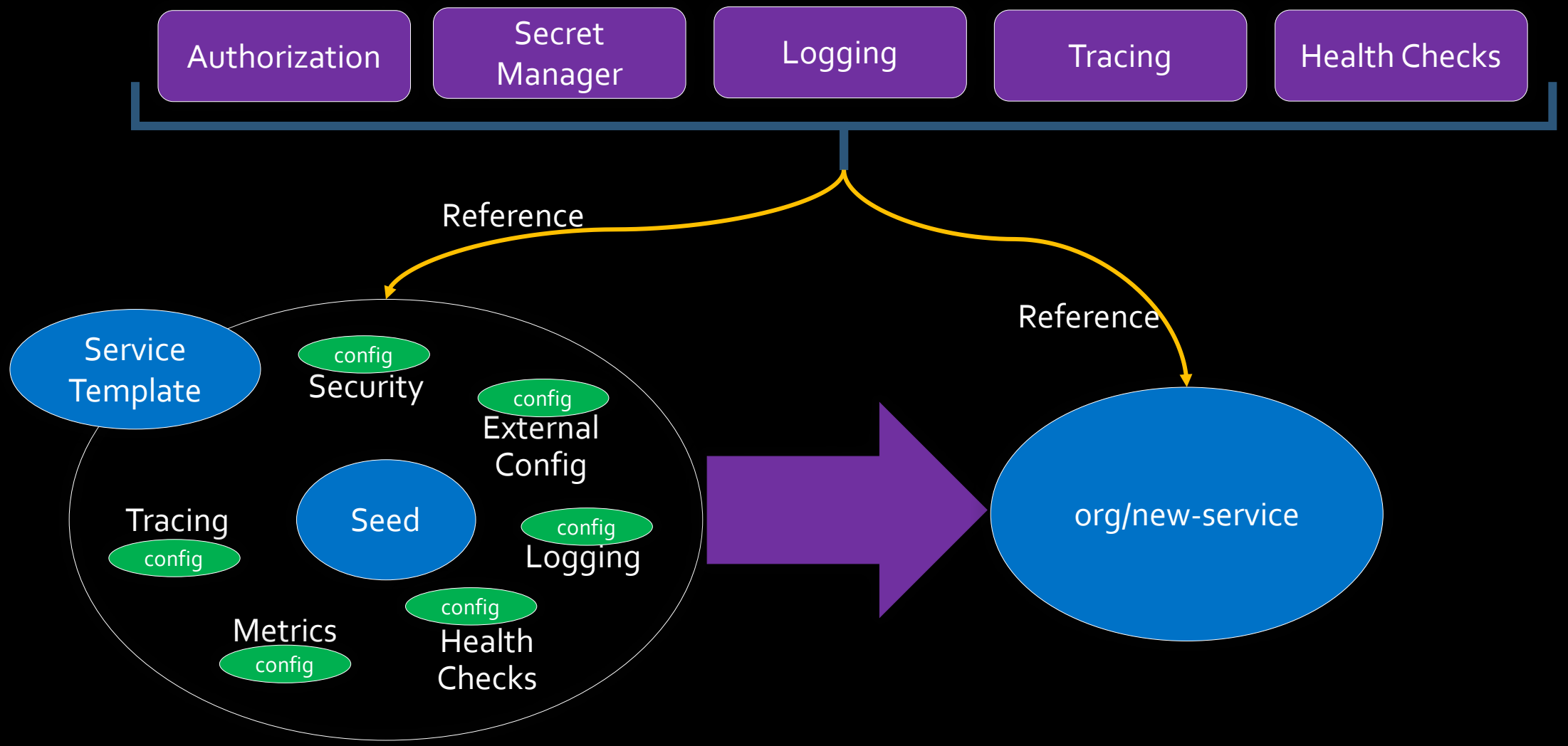


https://microservices.io/patterns/service-template.html

TEMPLATING

Value ★★★ Cost Medium

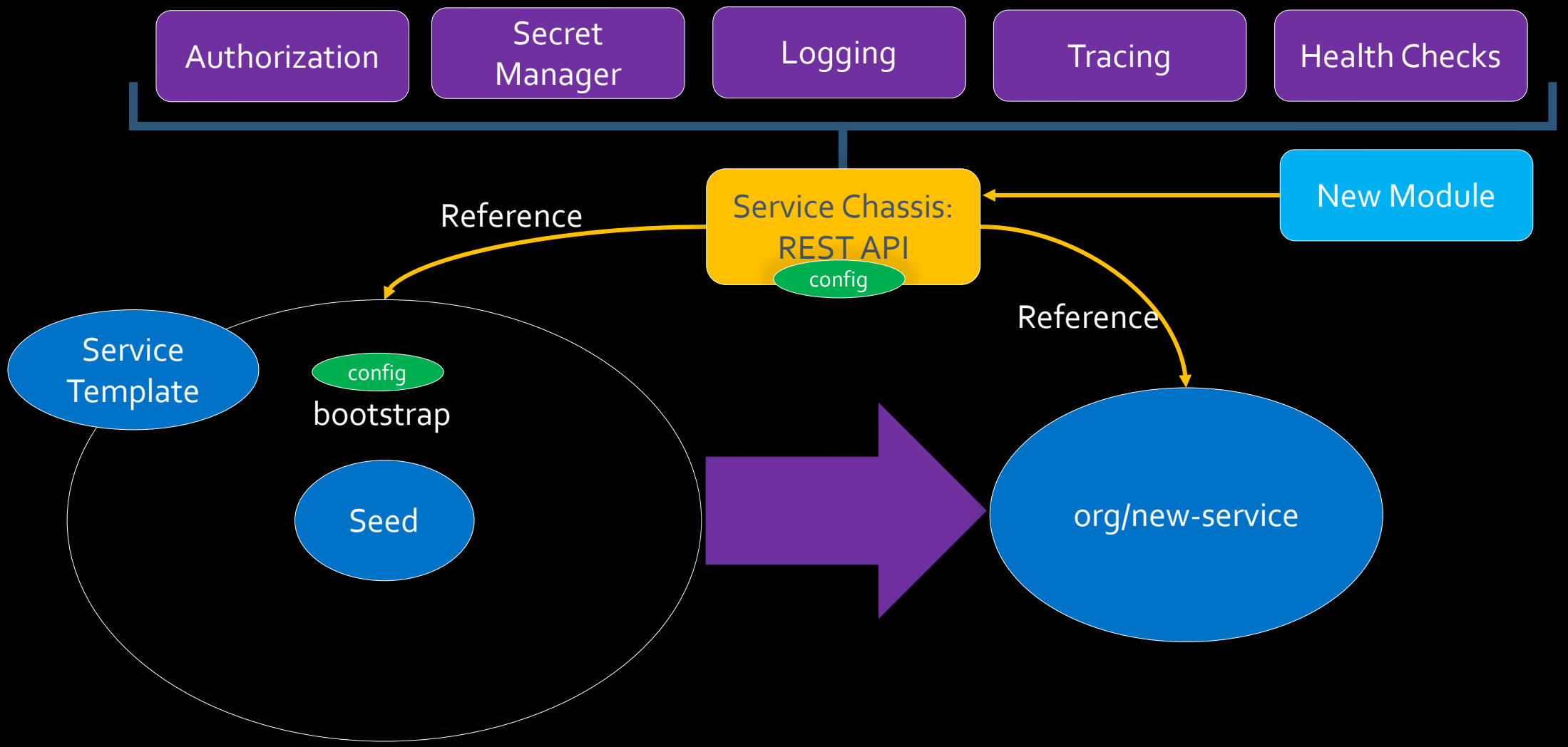
Service Chassis



TEMPLATING

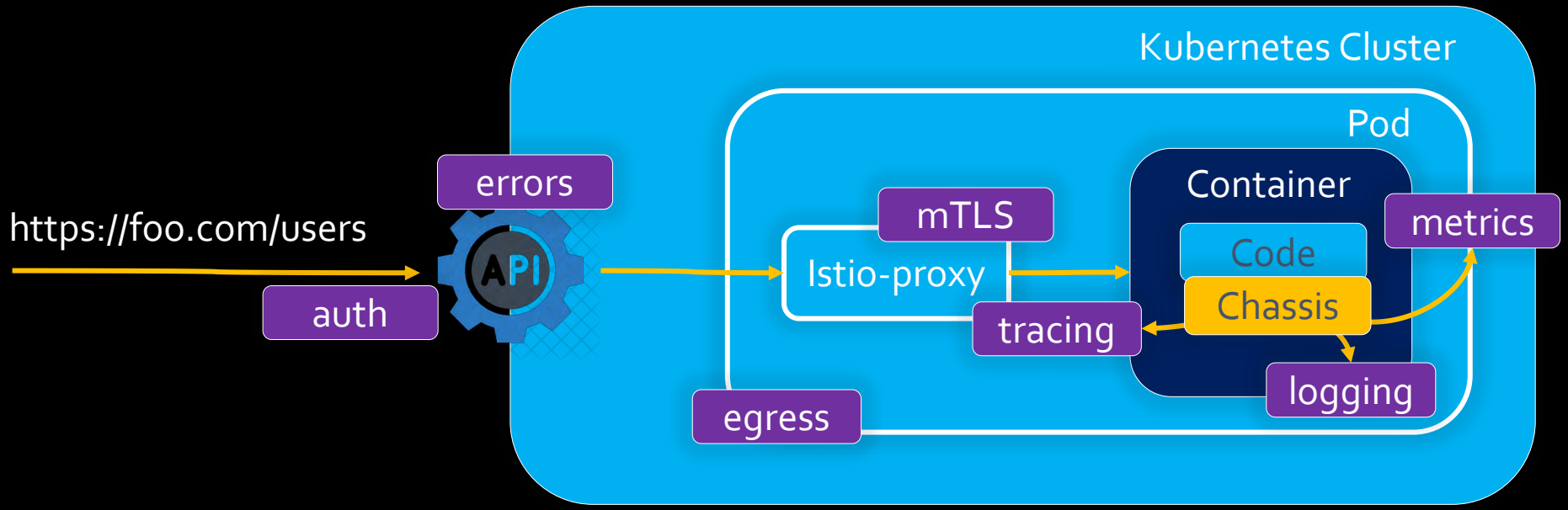
Value ★★★ Cost Low

Service Chassis



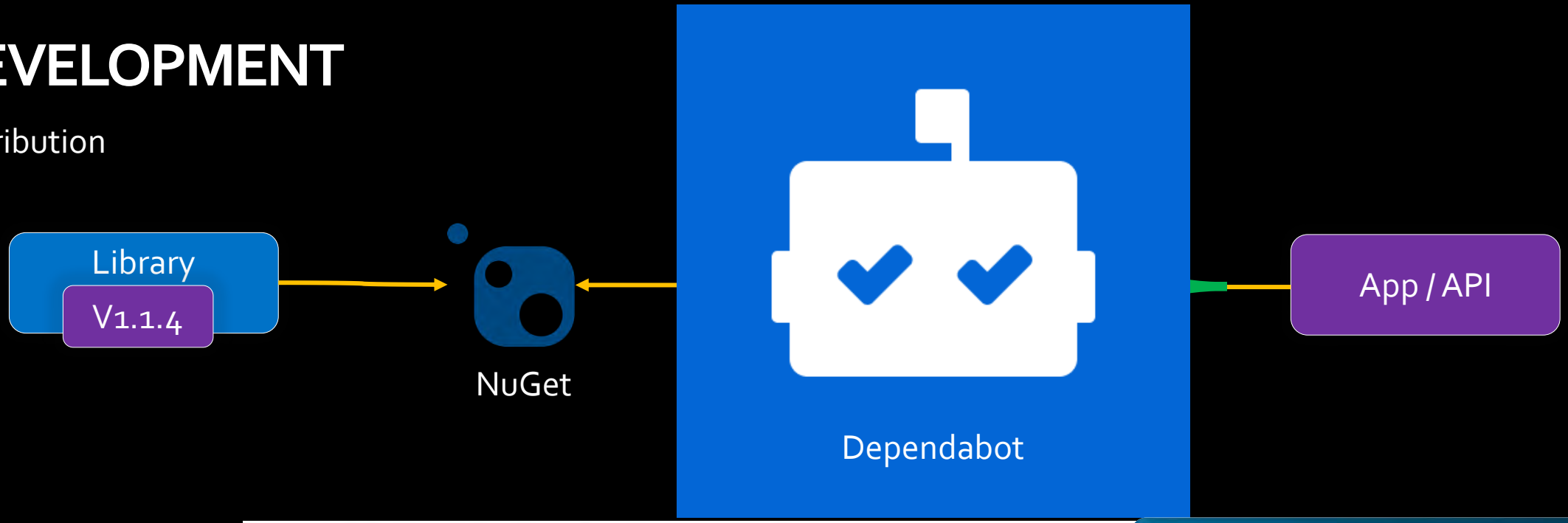
TEMPLATING

The Service Mesh Gap

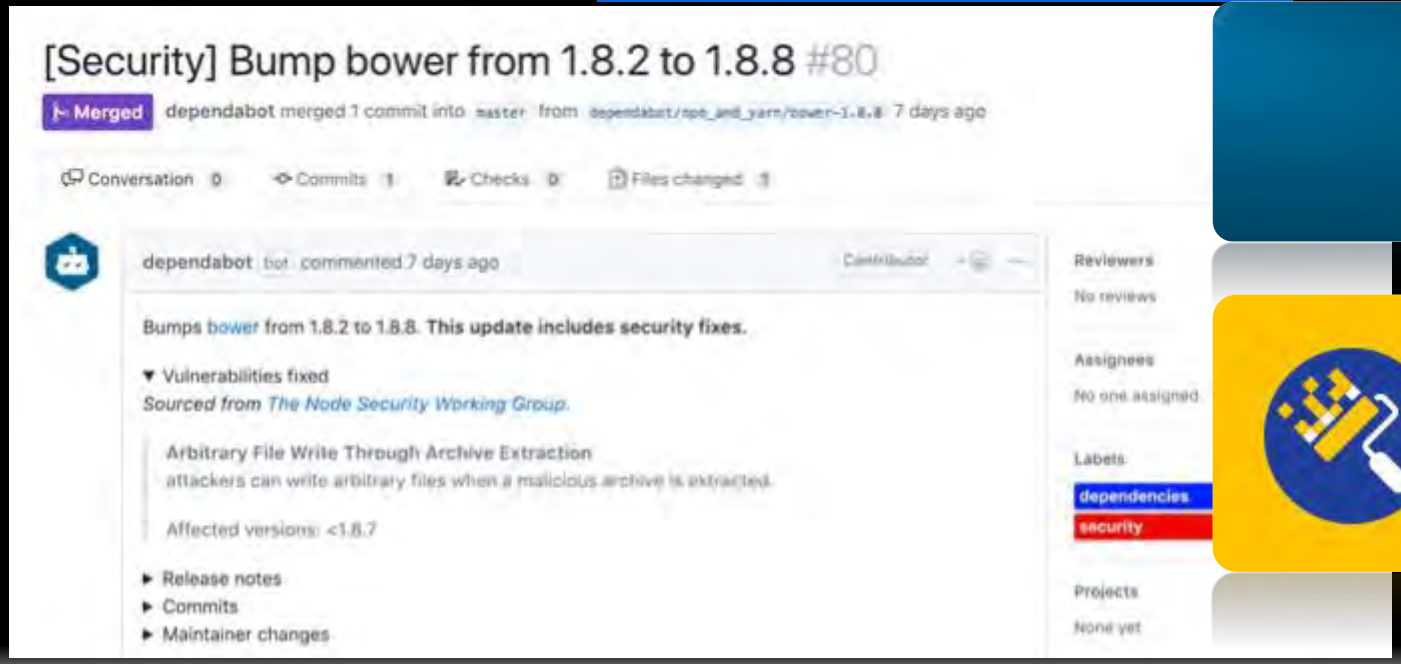


DEVELOPMENT

Distribution



- Private Feeds
- Configurable
- Ecosystems
- Pull Requests



TEMPLATING

Service Chassis at SPS

API Chassis

- Global Exception Handling
- "camelCase Serialization
- Authenticated Endpoints
- CORS Defaults
- Strict Security Headers
- HTTPS Redirect
- Index Redirect
- Reusable Logging Dashboards

- Sentry
- Swagger
- Security Middleware

- Errors
  - ~~Error Format V1~~
  - Error Format V2
- Health Check
  - /up
  - Kubernetes Probes /healthz
- Logging
  - Specialized Context
  - Structured JSON Output
- Secrets
  - ~~AWS SSM Parameter Store~~
  - AWS Secret Manager
- Identity
  - Resilient HTTP Clients
  - Remote / Distributed Auth Handlers
- Tracing
  - Propagators
  - ~~AWS XRay~~
  - OpenTelemetry
- Serialization
  - Newtonsoft.Json
  - System.Text.Json
  - Snake Case

TEMPLATING

Service Chassis at SPS

```
dotnet new webapi
dotnet install Spsc.Service.Chassis
```

```
SpsHost.RunService(args, () => {
    var builder = WebApplication.CreateBuilder(args);
    builder.AddSpsServiceChassis(config => {
        config.ServiceName = "My Service";
        config.Environment = "integration";

        config.EnableSpsJsonFormat = SpsJsonFormatOption.SnakeCase;
        config.EnableSentry = false;
        config.HealthCheckOptions = options => {
            options.TimeoutSeconds = 30;
        };
    });
    builder.Services.AddControllers();

    var app = builder.Build();
    app.UseSpsServiceChassis();
    app.MapControllers()
       .RequireAuthorization();
    app.Run();
});
```

COMPELLING CODE REUSE

Culture

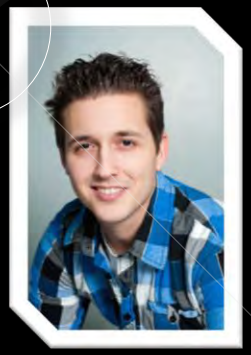
Where can you make incremental gains and achieve value?

> Code reuse is widely accepted as a "good thing", but it's important to manage your expectations of what it might deliver.
>
> You're more likely to be successful if you accept the limitations of your environment and restrict the scope of code reuse to something achievable that can add genuine value and gain widespread acceptance.
>
> Instead of developing grand designs for an internal code framework, it's often best to start small, develop iteratively and progressively build on small successes.
>
> ben-morris.com

> "Code reuse is the Holy Grail of Software Engineering."
>
> Douglas Crockford

COMPELLING CODE REUSE IN THE ENTERPRISE

- Appropriate Coupling
- Service Templates & Chassis



TRAVIS GOSSELIN

@travisjgosselin

www.travisgosselin.com