



# Releasing features at the flick of a switch

*Dimitri Holsteens*

# Thank you!

Capgemini

ae architects  
for business  
& ict

REALDOLMEN  
to get there, together

DEVOTeam  
Consulting • Solutions • Expertise

Microsoft

cegeka  
Van dichtbij  
meemaken

involved  
User-Centered Software

aariXa  
> ict and business optimization

Queaso  
systems

TOBANIA

kenze  
Experts in .NET

ORDINA

Sparkles  
IT Consultancy & Training

codit  
integrating your business

AXXES  
IT CONSULTANCY

visug

# Releasing features at the flick of a switch

Dimitri Holsteens

@holstdi

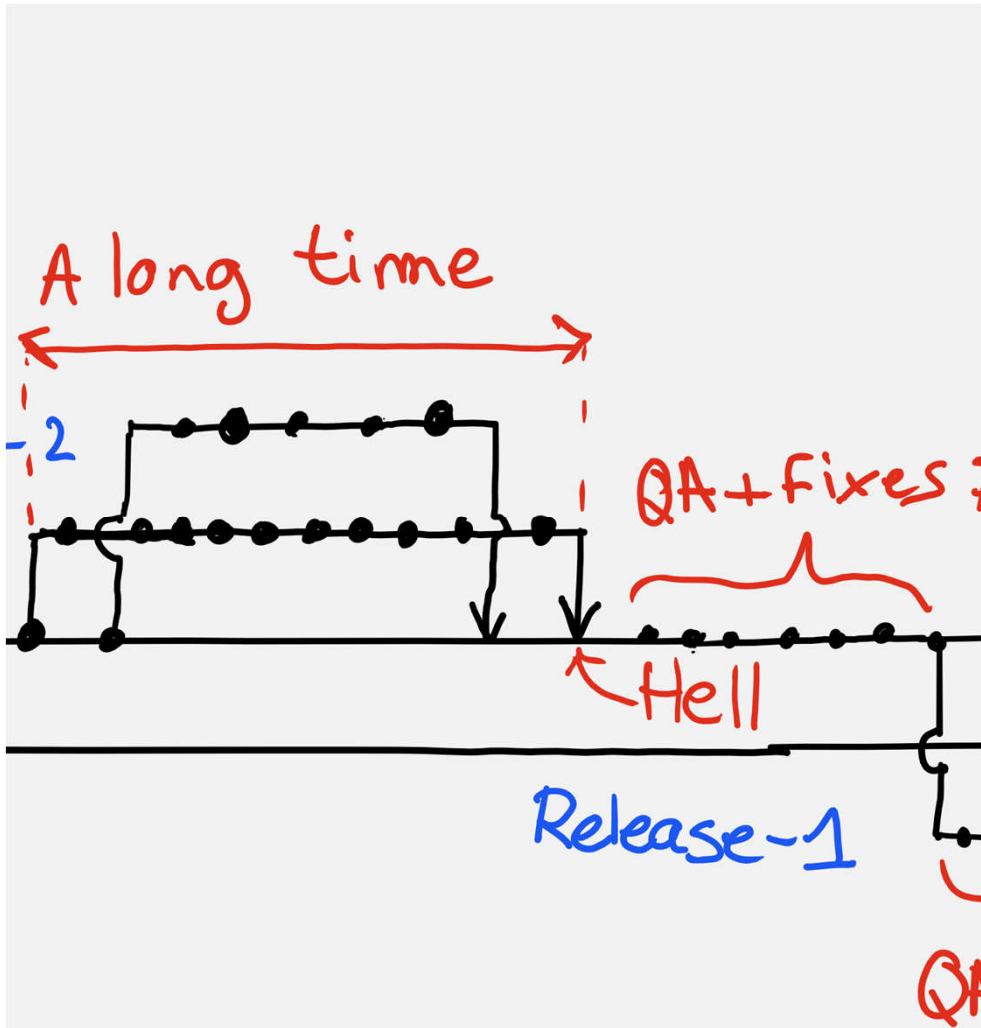
23/10/2018



Once upon a time...









Eliminate Waste

- Trunk based development
  - Feature Toggles





FEATURE  
FLAG  
TOGGLE  
FLIPPER  
BIT

...



Just a conditional statement...  
That's easy peasy



0 references

```
class Program
```

```
{  
    public const bool FeelingNostalgic = true;
```

0 references

```
static void Main(string[] args)
```

```
{  
    if (FeelingNostalgic)  
    {  
        Console.BackgroundColor = ConsoleColor.Black;  
        Console.ForegroundColor = ConsoleColor.DarkGreen;  
    }
```

```
    Console.WriteLine("Feature Toggle 101");
```

```
    Console.ReadLine();
```

```
}
```

```
}
```



All Caught Up !

Here's  
an accident waiting to happen... ?



Pete Hodgson, [martinfowler.com](http://martinfowler.com) :

- Technique allowing teams to modify behavior without modifying code
- Various usage categories
- Introduce complexity
  - Can be mitigated by using smart implementation practices, appropriate tools and by constraining the number of toggles in our system



Business  
OPS  
Experiment  
Canary  
Permission  
Phased Rollout  
AB Test  
Feature Toggle  
Release  
Split traffic



*'Do not activate the new price calculation until it's finished,  
tested and approved'*

*'Do not activate the sync with our marketing system until it's up  
and running'*



# Release Toggle

- Decouple deployment and actual feature release
- Supports trunk based development and continuous delivery



*'We should be able to turn off this profile export when end-user performance degrades due to high I/O impact'*

*'We should be able to turn of the push to the accounting software since because of it's instability'*

# OPS Toggle

- Kill switches for operational reasons
- Support non-functional requirements
- Overlap with concept of 'Circuit Breakers'





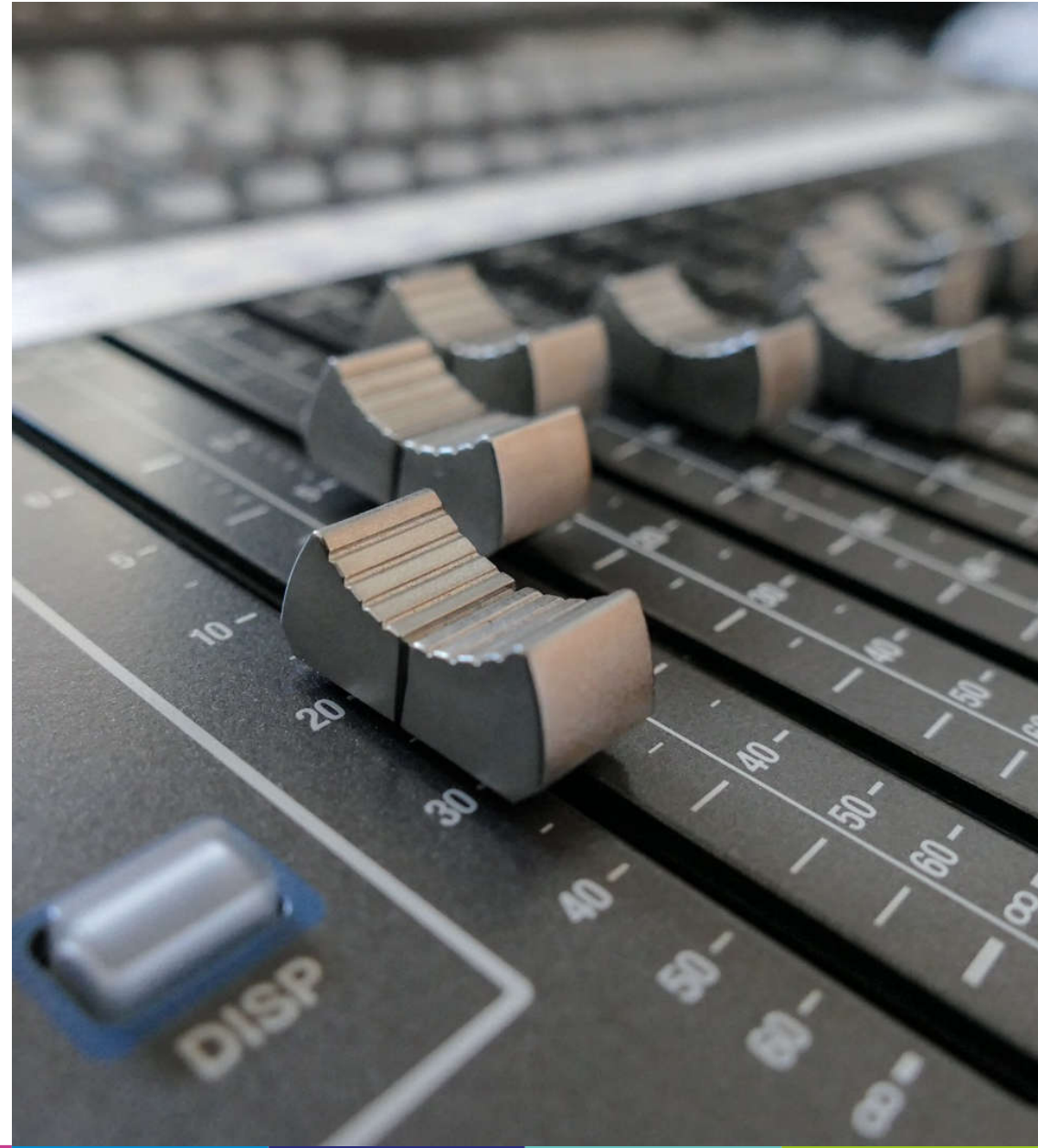


# Splitting Traffic

Allow variation of feature toggle state for different users

## Controlled Rollout

- Release toggle with traffic splitting
- Gradually introduce the feature across the userbase
- User cohorts can be random or predetermined





## A/B Test

- Test user response on 2 variants
- Randomized
- Capture metrics



## Canary Release

- Introduce a feature to a very small subset of the userbase
- Random or predetermined
- Measure efficiency of the new feature before rolling out to general product
- Explicit opt-in (and opt-out)



# Experiment Toggles

- Controlled enabling of experimental features
- Random or targeted toward specific users or classes of users
- Capture metrics to evaluate success



*'We can't enable this feature for customers in Europe since it's not GDPR compliant'*

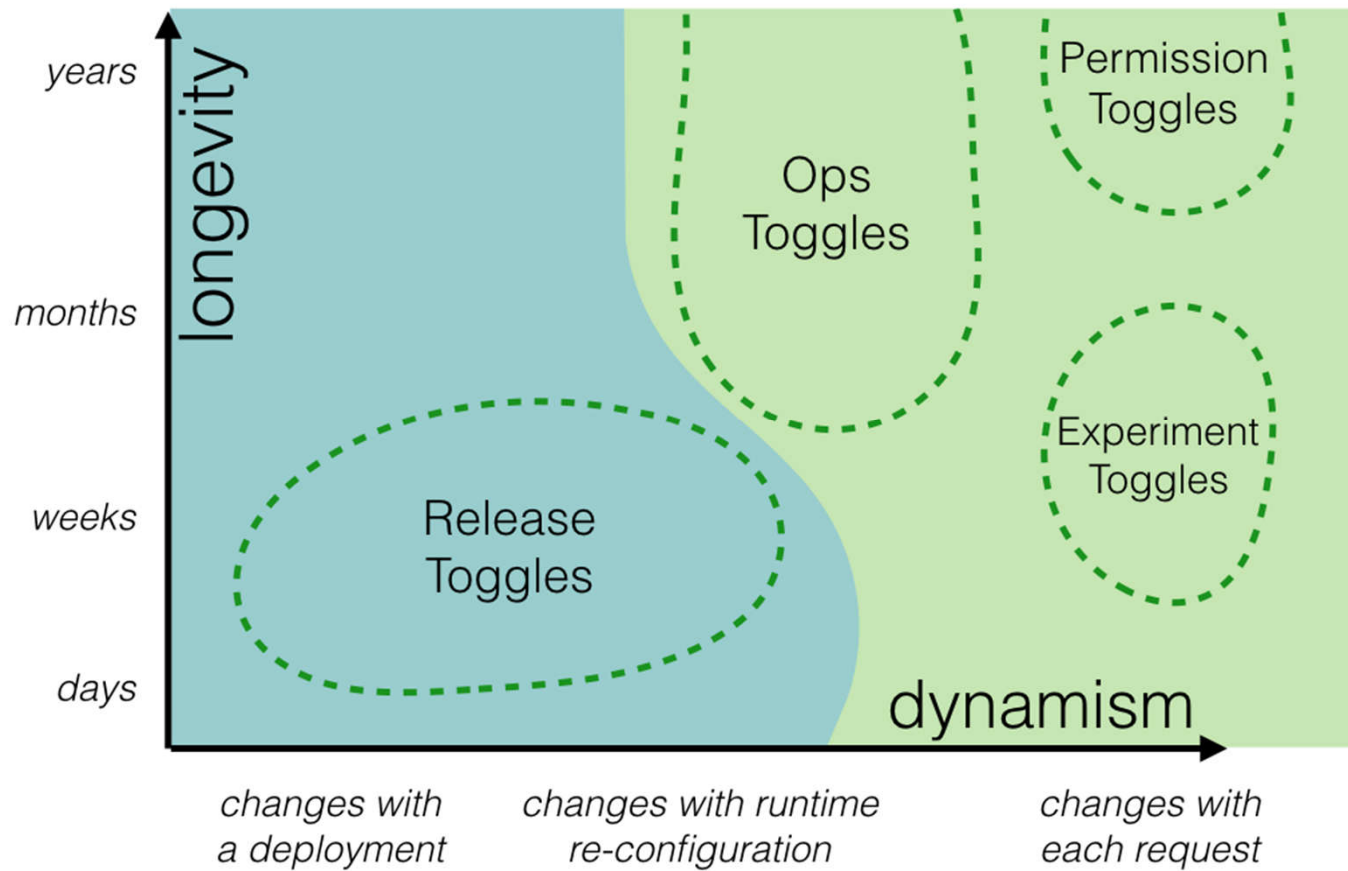
*'Users which have a premium subscription should be able to download these courses for offline viewing'*



# Permission Toggles

- Enable or disable parts of the software for specific classes of users
- Not to be confused with authorization rules
- AKA 'Business Toggles'





# Smart implementation practices

```
var d = a(c.form.querySelectorAll('input[type=checkbox][name="' + b.el.name + '"']));  
if (0 == d.index(b.el)) {  
    var e = d.filter(":checked").length;  
    return e >= b.arg || g.minChecked.replace("{count}", b.arg)  
}  
},  
maxSelected: function(a) {  
    return null != a.val ? a.val.length <= a.arg || g.maxSelected.replace("{count}", a.arg) : null  
},  
minSelected: function(a) {  
    return null != a.val ? a.val.length >= a.arg || g.minSelected.replace("{count}", a.arg) : null  
},  
radio: function(b) {  
    var c = a(this.form.querySelectorAll('input[type=radio][name="' + b.el.name + '"']));  
    return 1 == c  
},  
custom: function(a, b) {  
    var c = b.options.custom[a.arg],  
        d = new RegExp(c.pattern);  
    return d.test(a.val) || c.errorMessage  
},  
remote: function(a) {  
    a.remote = a.arg  
}
```

# Simple Rules

- Keep toggles as short lived as possible
  - Avoid leftover flags
  - Make sure long term / permanent flags are easily recognizable
- Do not reuse feature toggles
  - Single responsibility principle : flag should have exact 1 reason to exist



# Simple Rules

- Use proper naming
  - ☒ Pomolo
  - ☒ HubspotSyncEnabled
  - ☒ TI685\_HubspotSyncEnabled
- Avoid conflicting flags
- Make default value explicit

# Managing toggle Configuration

- Code
- Config
- Database
- Central configuration store (zookeeper, consul, ...)
- Hand-rolled 'Feature Service'
- 'Feature as a service' provider

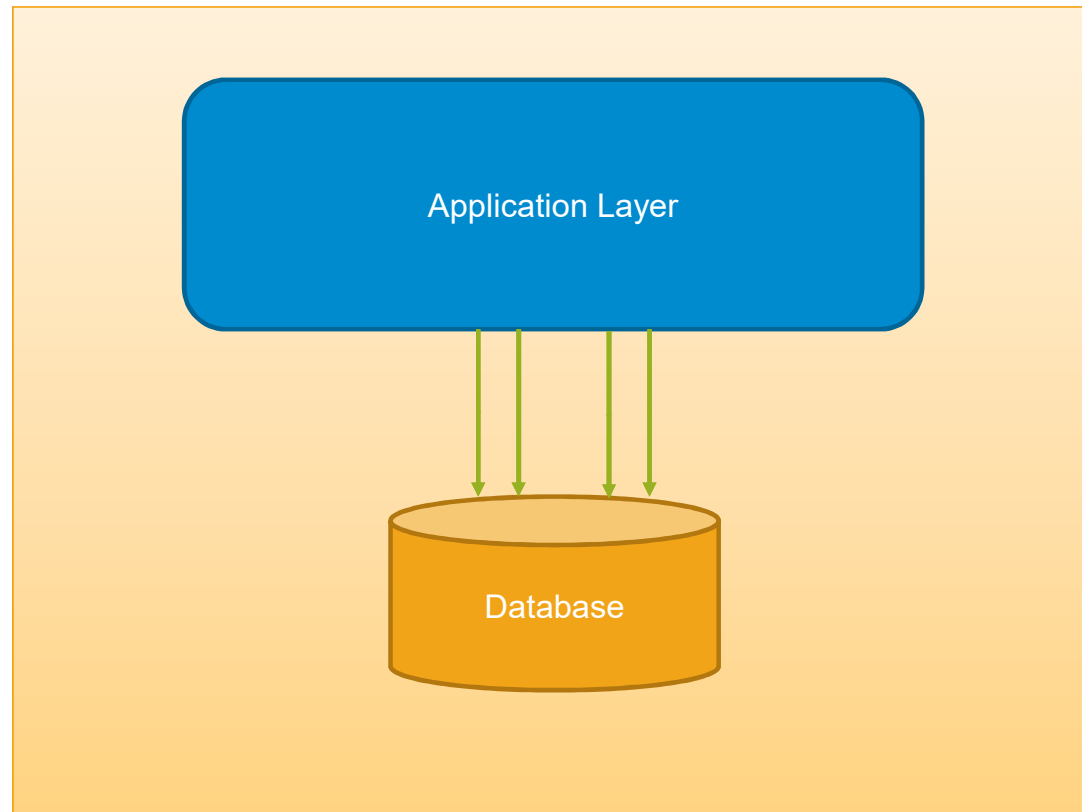


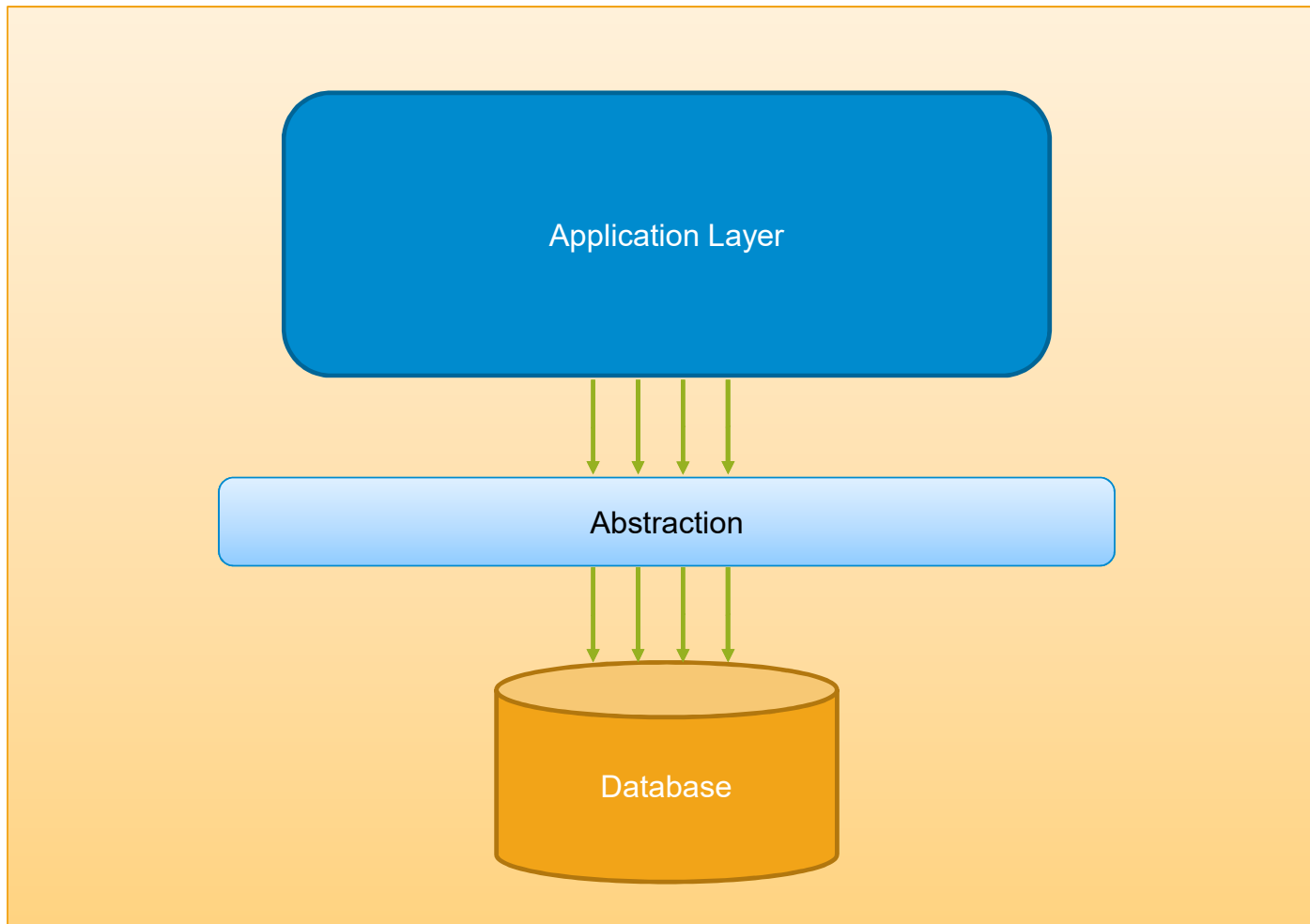


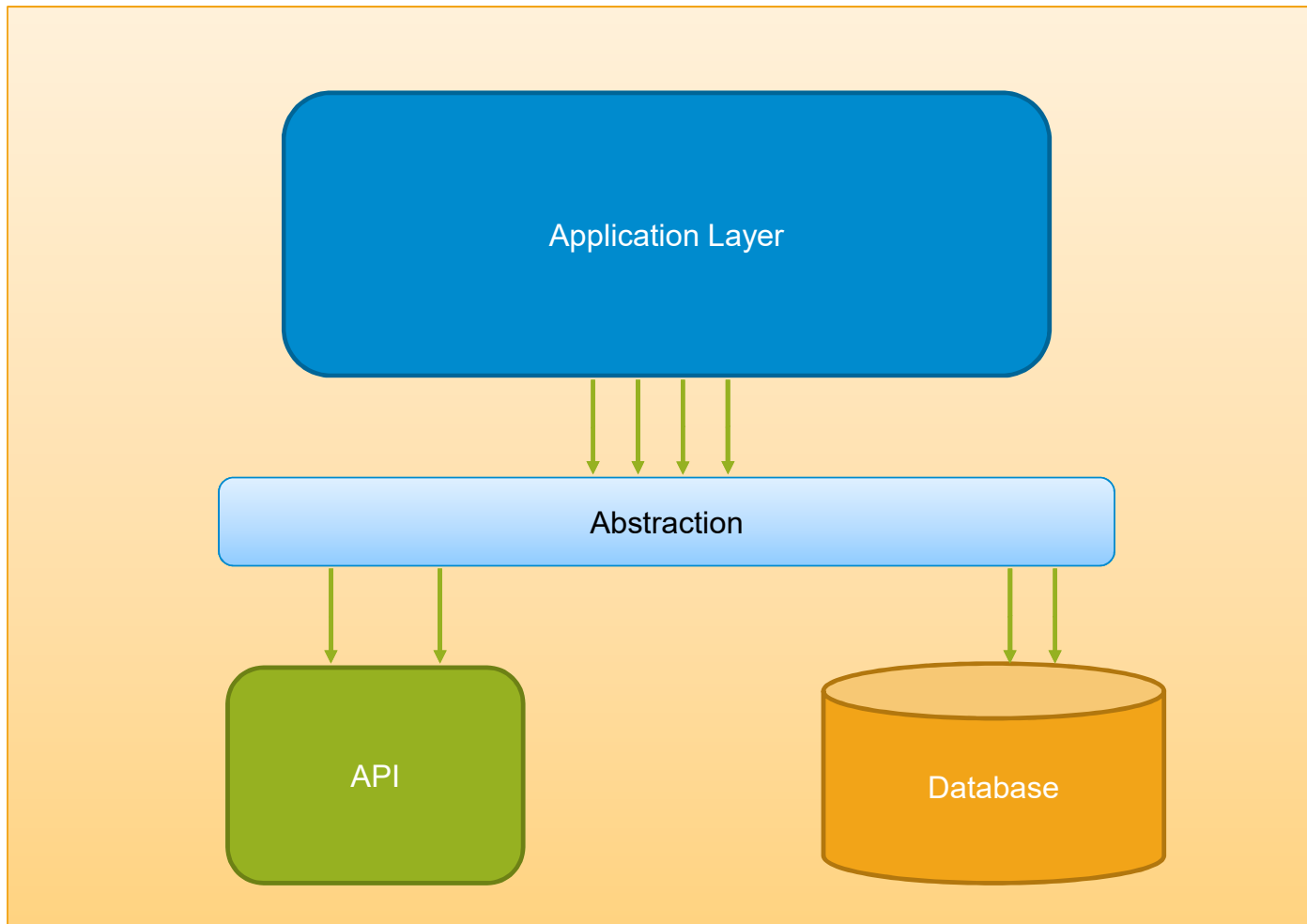
Factor feature toggles into your  
application design

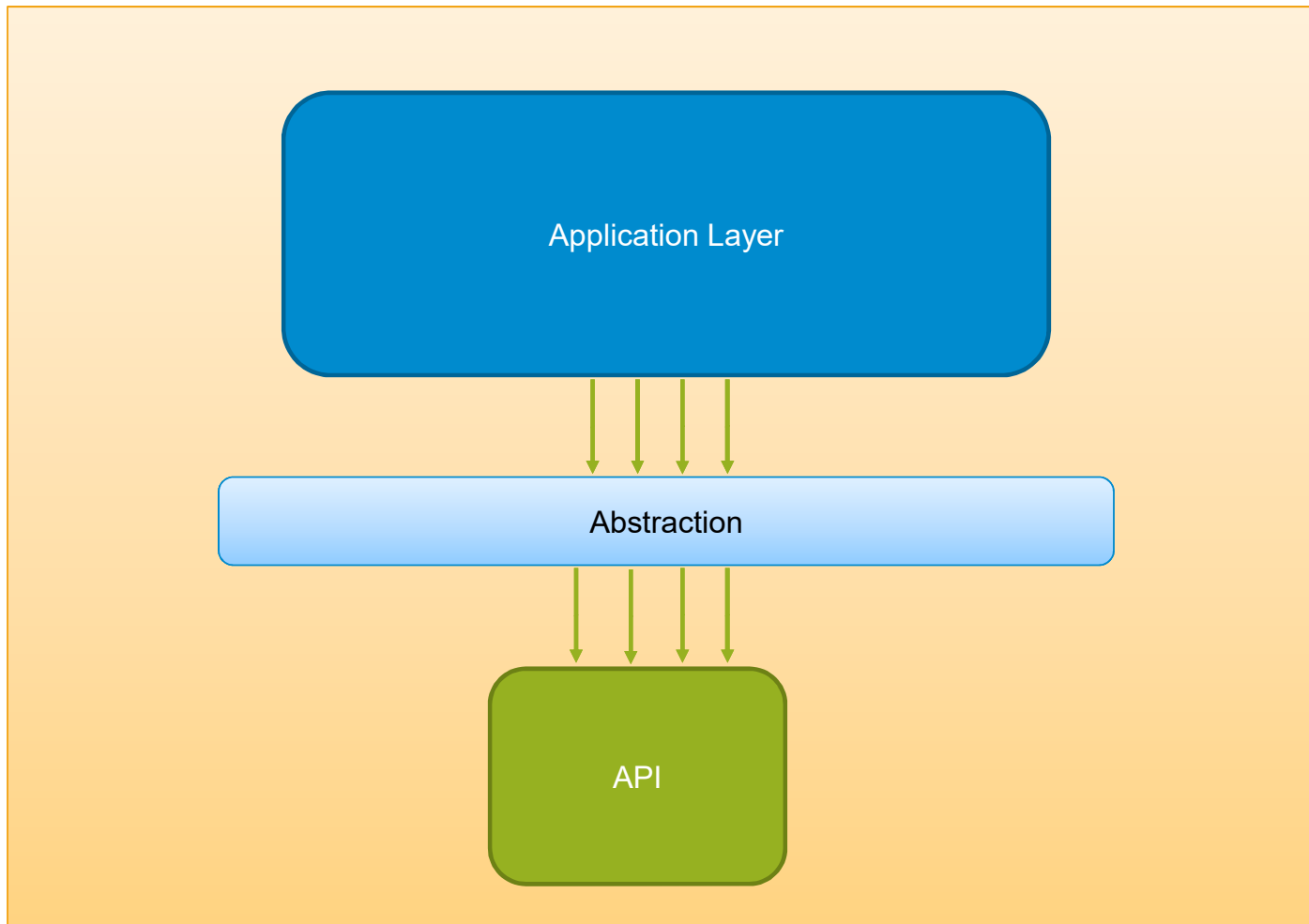


# Branch by abstraction





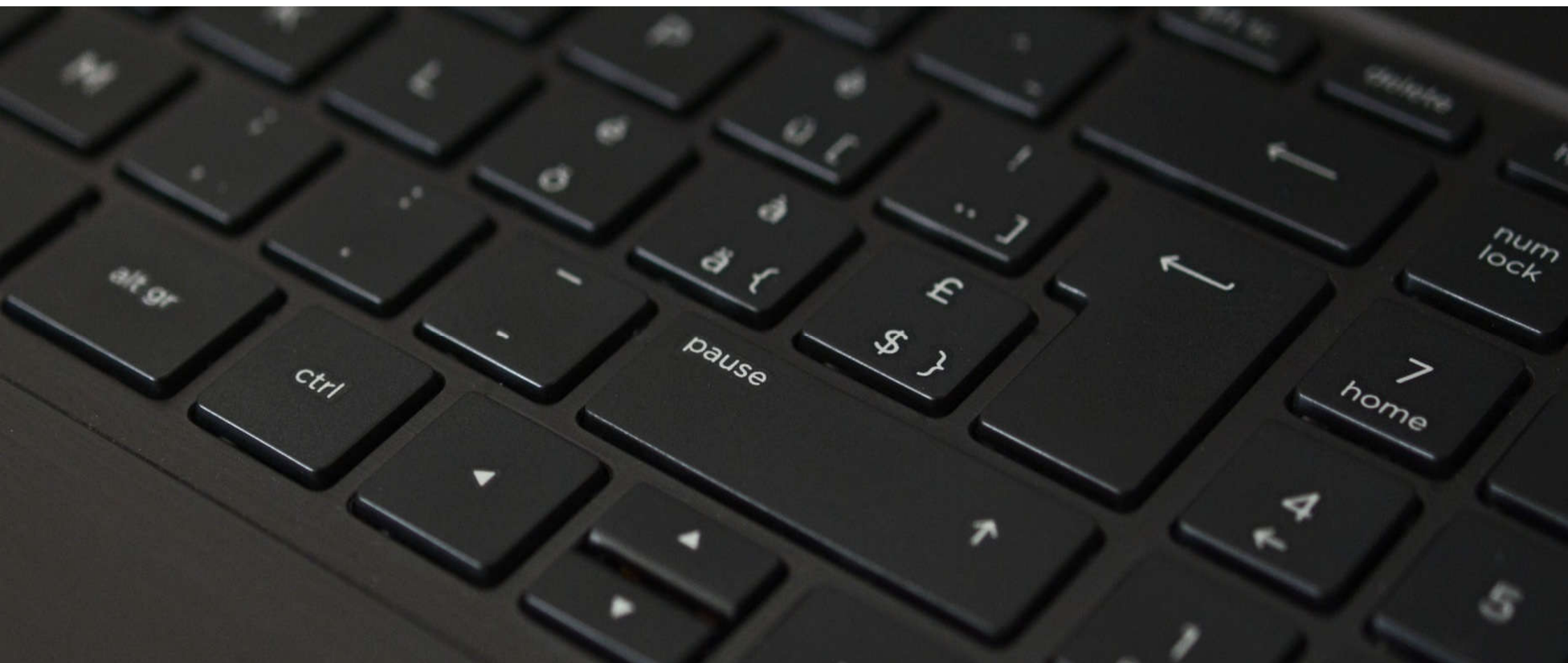




# Feature Toggle Packages

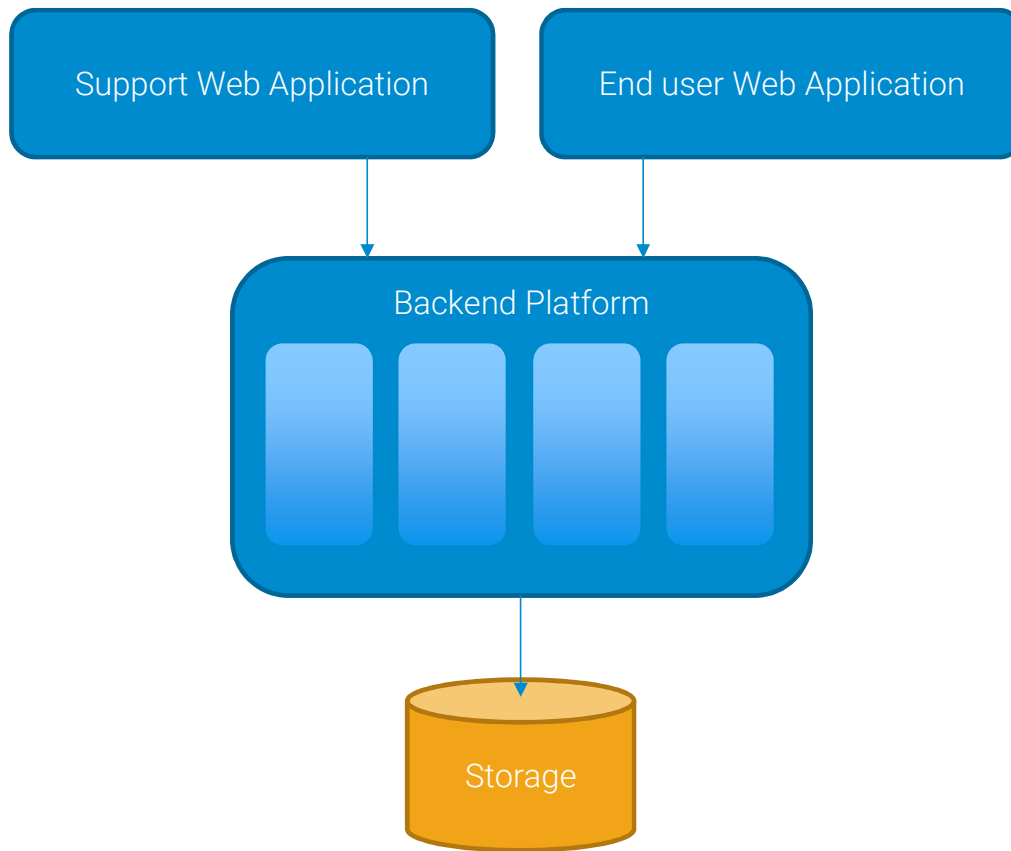
	Configuration	Toggle code	Toggle rules
.NET Feature flags	sqlserver db	Class	Static on/off
NFeature	custom configsection	Enum	Static on/off, timebased
FeatureToggle	code / appsettings / sqlserver	Class	Static on/off, timebased, assemblyversion, random
FeatureSwitcher	code / custom config setting	Class	Static on/off, context evaluation
nToggle	code / custom config setting	Class	Static on/off
Toggler	appsettings	Class	Static on/off
LaunchDarkly SDK	faas	String	FaaS rules
Split.IO SDK	faas	String	FaaS rules



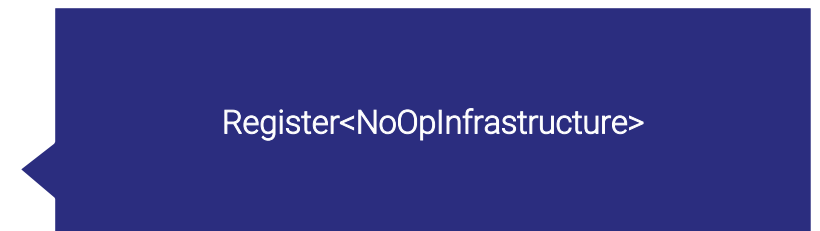
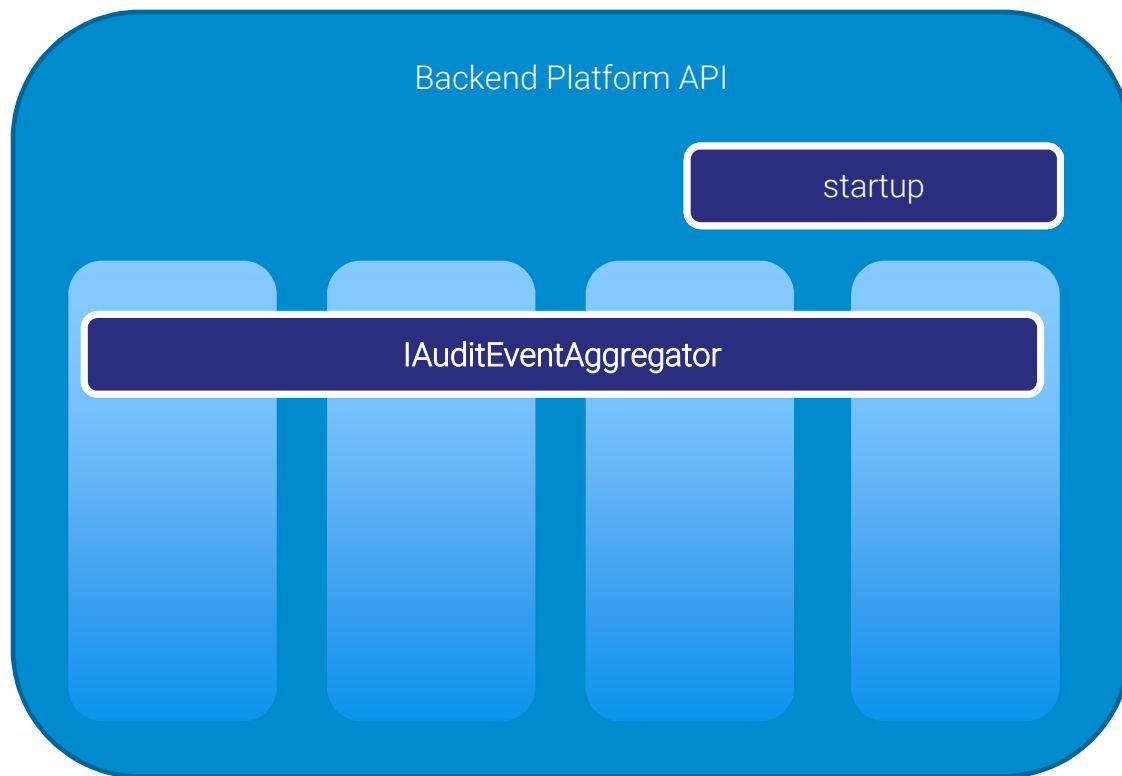


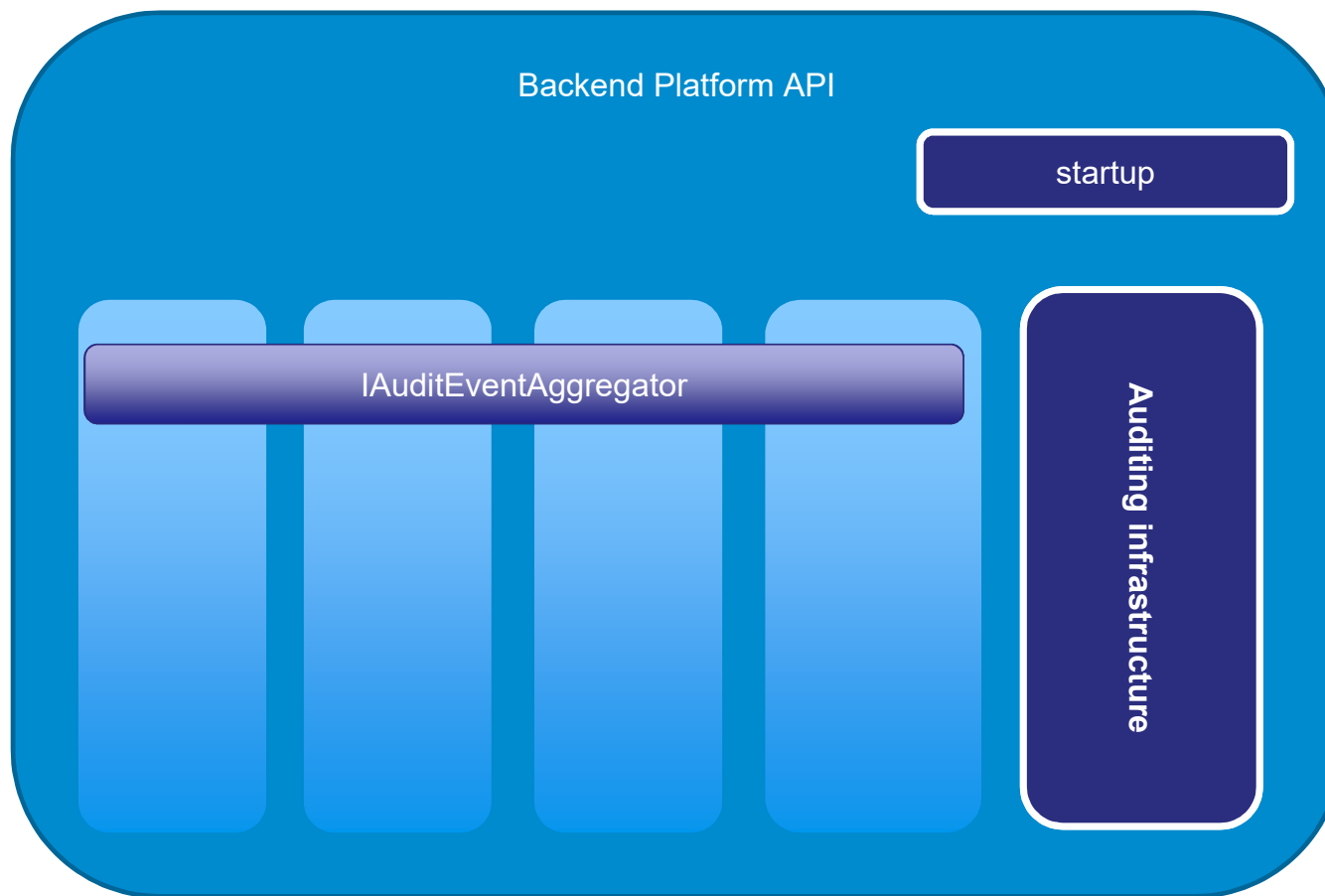
Code Intermezzo

# A more complex use case

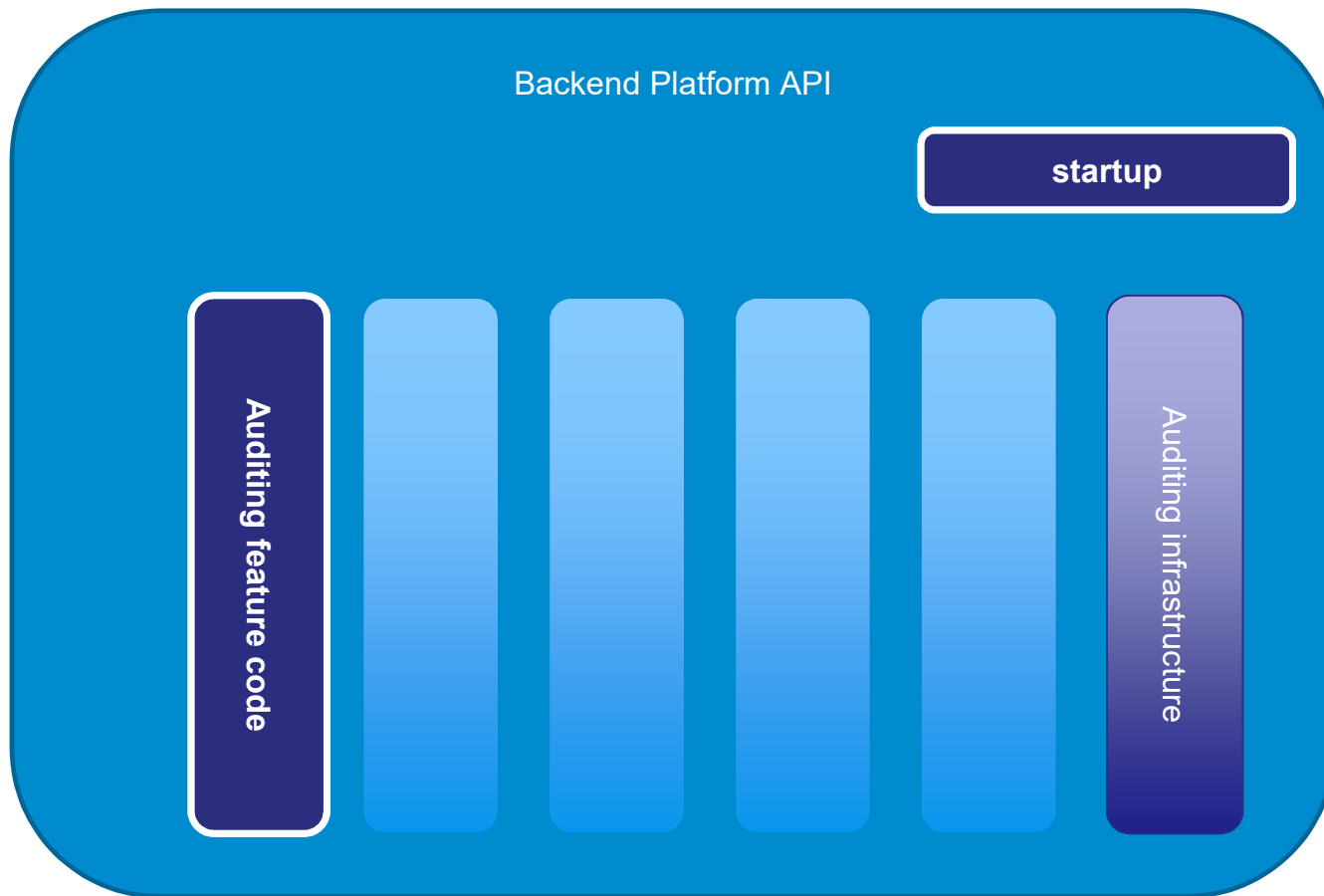


Introduce auditing  
(and use feature toggles)

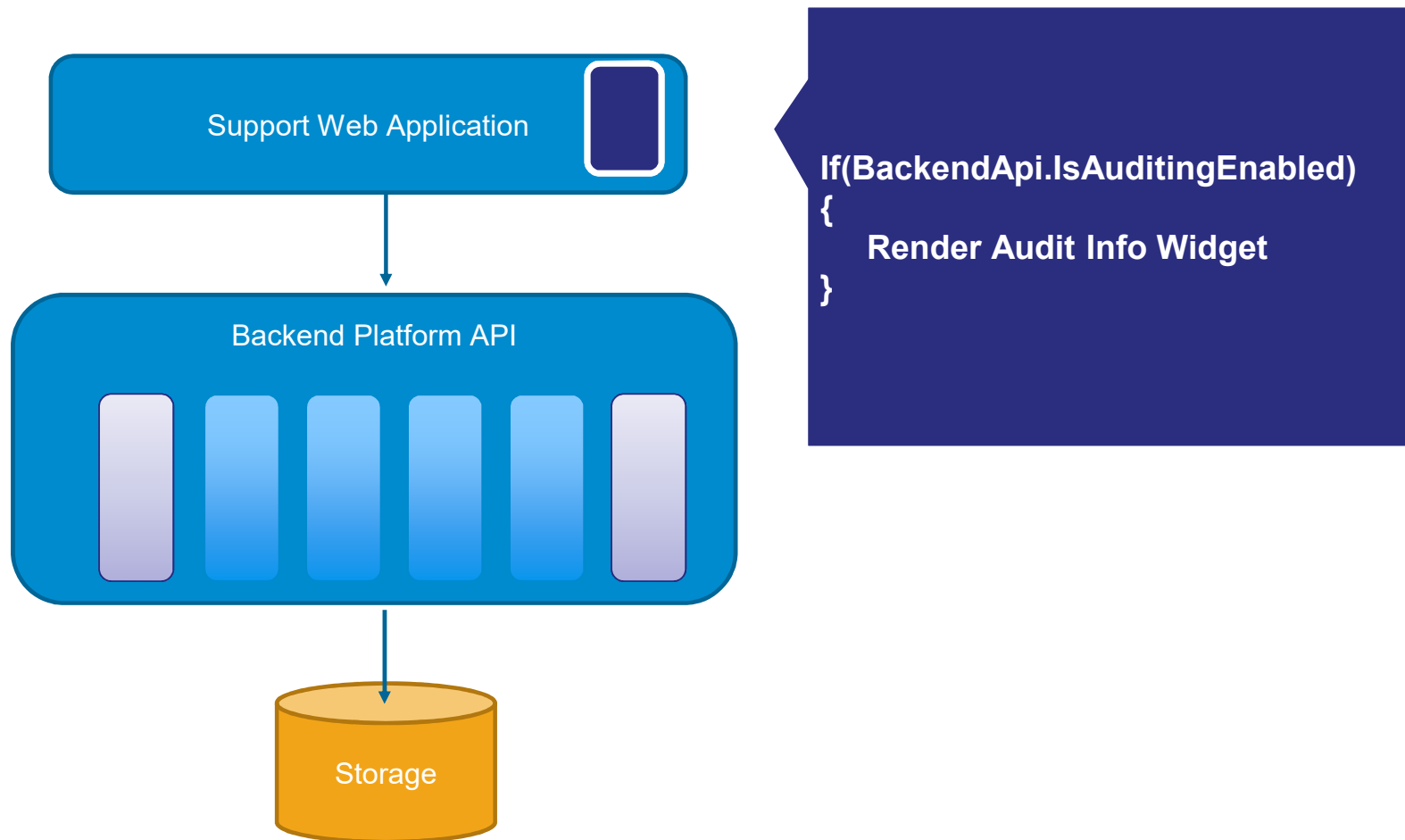




```
Register<NoOpInfrastructure>;  
If(AuditFeatureEnabled())  
{  
  
  Register<AuditingInfrastructure>;  
}
```



```
If(!AuditFeatureEnabled())  
{  
  DisableRoute("api/audit");  
}
```







# Feature Toggle Governance



## Plan for removal

- Self-reporting toggles
  - How often is a toggle consulted ?
  - How often does it return 'enabled' ?
- Make explicit removal task in backlog




## Control access to flags

- Different usage categories require different security settings
- Audit when, why and by whom flag state is changed



Monitor flag state





BRINGing  
*it all*  
TOgeTher

- Feature toggles can be very powerful (but with great power comes great responsibility...)
- Plan before you build
- Identify the use case you want to solve and how to solve it with as little complexity as possible
- Use OOP patterns for smart implementation
- Avoid technical debt **and keep the number of toggles in your system to a minimum**



All Caught Up !

Here's a pony.

# Thank you !



Dimitri Holsteens

@holstdi