Building Multi-Tenant ASP.NET Core Applications

Alper Ebicoglu

Co-Founder of **Volosoft**

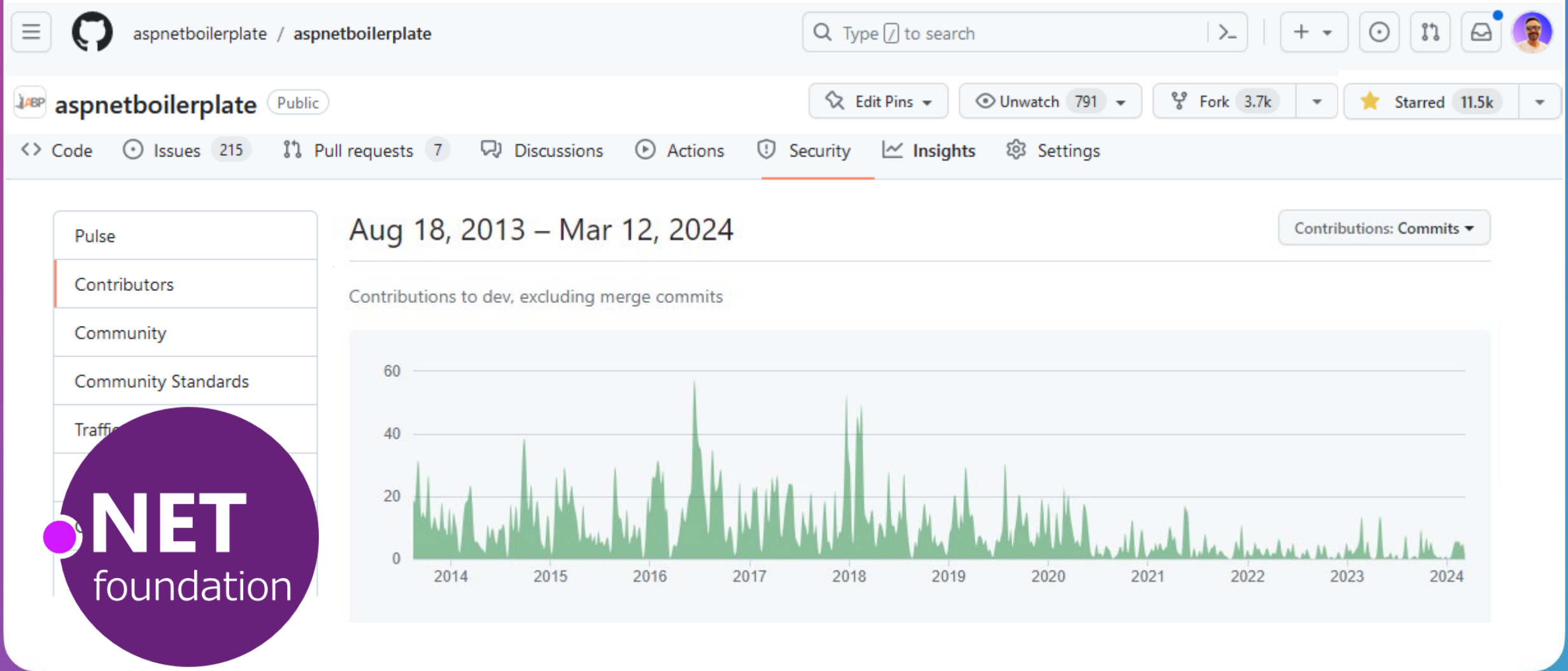
alper.ebicoglu@volosoft.com

twitter.com/**alperebicoglu**

abp.io

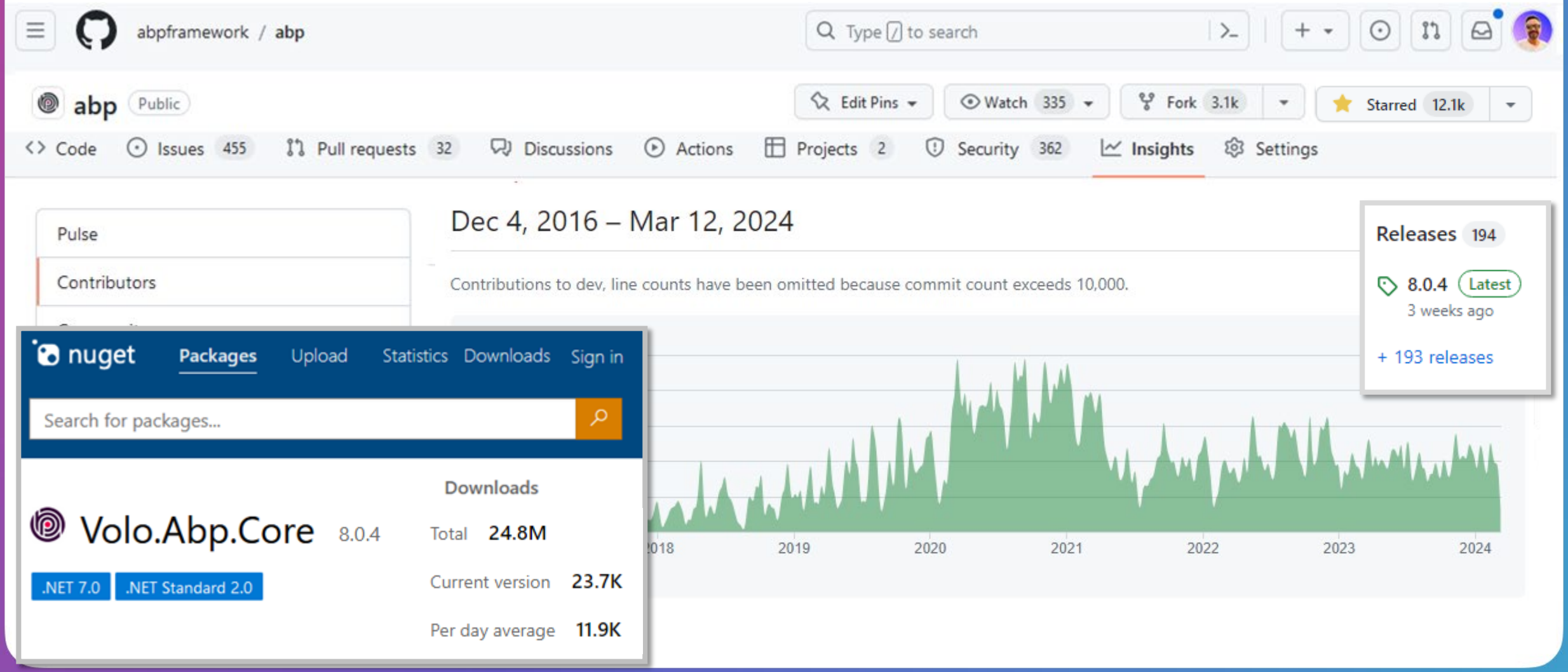
ASP.NET Boilerplate

Open-source Framework on ASP.NET Core



.NET foundation

abp.io | Open-source Framework on ASP.NET Core



What is **ABP Framework**?

Your Application

Focus on your business code

- do what you do best

ABP Web Framework

An opinionated architecture
to build line-of-business web apps

- Multi-tenancy
- Audit logging
- Exception handling
- Background jobs
- Modularity
- Event bus
- Unit of work
- etc...

ASP.NET Core Web Framework

Generic web framework

- Routing
- Dependency injection
- Session management
- Request / response
- Security
- etc...

Agenda

- **Introduction** to SaaS & Multi-Tenancy
- **Pros and Cons** of Multi-Tenancy
- Database & **Deployment Scenarios**
- **Identifying** and Changing the **Active Tenant**
- **Data Isolation**
- Conditionally Turning **Multi-Tenancy On / Off**
- Handling **Database Migrations**
- Implementation of the **Feature System**

abp.io

What is Multi-Tenancy?

- A common approach to build SaaS solutions
- Resources are shared between tenants
- Application data is isolated between tenants

Parties

- **Tenants:** Our clients, using the service
- **Host:** Service provider

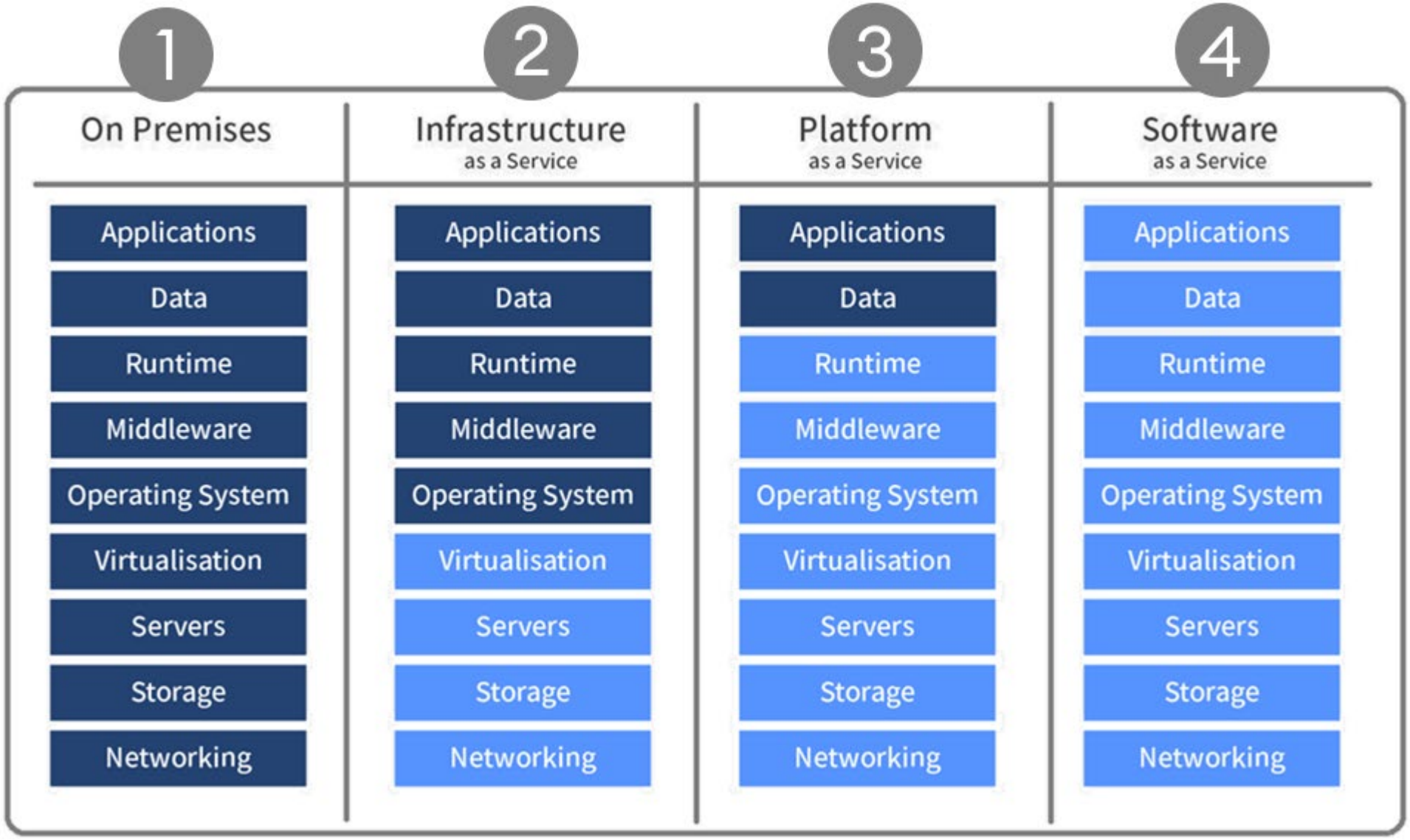
An ideal multi-tenant application should be

✓ Unaware of multi-tenancy as much as possible!

✓ Deployable to on-premise as well

abp.io

As-a-Service Business Models



You manage

3rd Party Manages

abp.io

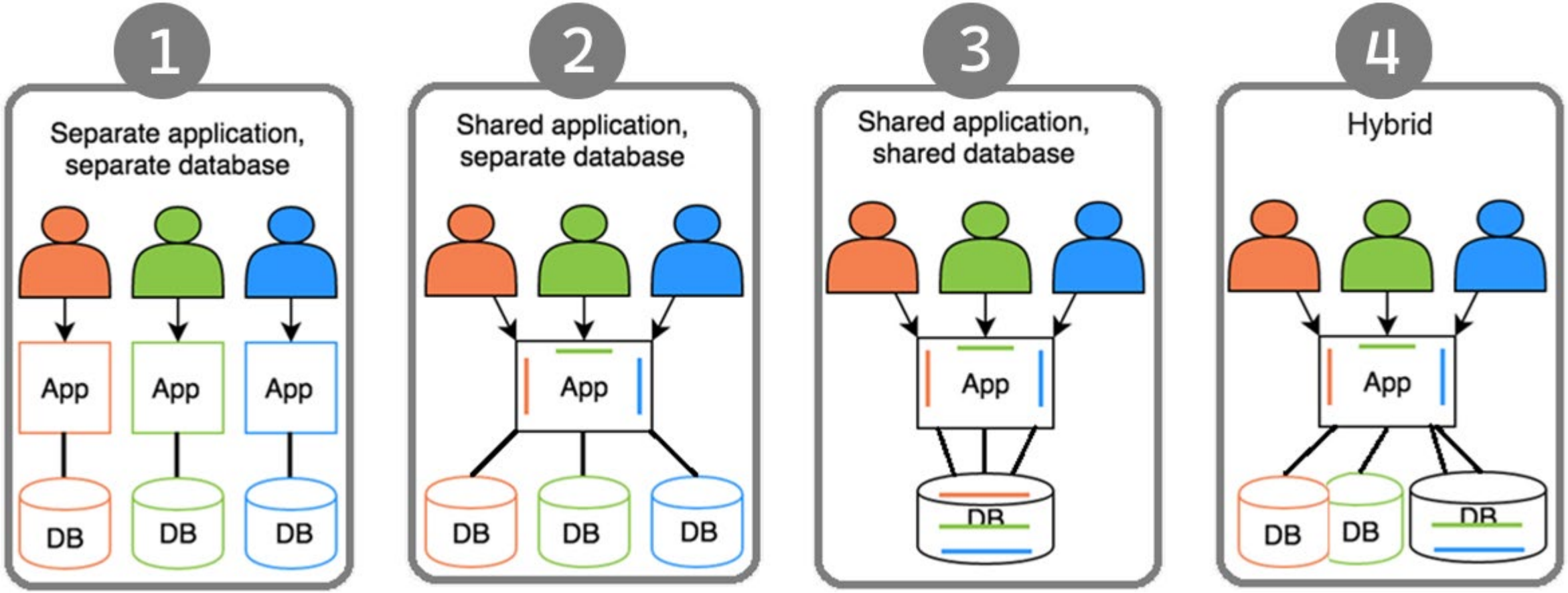
Advantages of Multi-Tenancy

1. Cost efficiency – max utilization
2. Consistent user experience
3. Ease of maintenance
4. Scalability
5. Rapid deployment for new users

Challenges of Multi-Tenancy

1. **Data isolation**
2. **Configuration** & **customization** per tenant
3. Performance balance: **Noisy neighbors**!
4. **Security**
5. **Backup** and recovery

Deployment & Database Architectures



abp.io

Maintaining Application States

Application code & services should be stateless!

Where should we save the state? 🤔

- ✓ **HTTP Request** (cookie, header, query string, payload)
- ✓ **Authentication ticket**
- ✓ **Database**
- ✓ **Distributed cache** (Redis, Memcached, ...)

abp.io

Identifying the Active Tenant

Identifying the Active Tenant

How to determine the current tenant? 🤔

1. CurrentUserTenantResolveContributor
2. QueryStringTenantResolveContributor
3. RouteTenantResolveContributor
4. HeaderTenantResolveContributor
5. CookieTenantResolveContributor
6. DomainTenantResolver

abp.io

Identifying the Active Tenant

1. Current User (claims)

```
var currentUser = context.ServiceProvider.GetRequiredService<ICurrentUser>();
if (currentUser.IsAuthenticated)
{
    context.Handled = true;
    context.TenantIdOrName = currentUser.TenantId?.ToString();
}
```

```
HttpContext.User.Identity.Claims
.FirstOrDefault(c => c.Type == “TenantId”)
```

Identifying the Active Tenant

2. Query String

```
var tenantId = httpContext.Request.Query["tenantId"].ToString();
if (!string.IsNullOrWhiteSpace(tenantId))
{
    context.Handled = true;
    context.TenantIdOrName = tenantId;
}
```

https://fabrikam.com?tenantId=3

Identifying the Active Tenant

3. Route

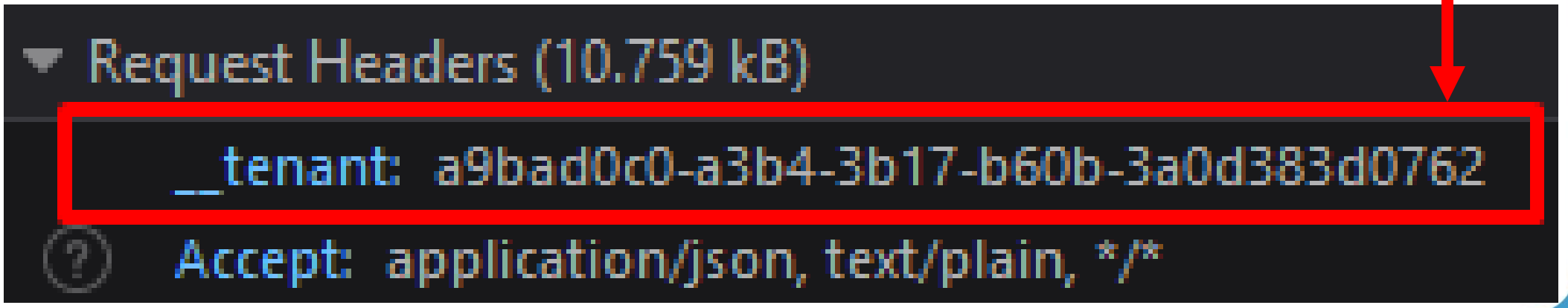
```
var tenantId = httpContext.GetRouteValue("tenantId");
if (tenantId != null)
{
    context.Handled = true;
    context.TenantIdOrName = tenantId.ToString();
}
```

https://fabrikam.com/acme/

Identifying the Active Tenant

4. Header

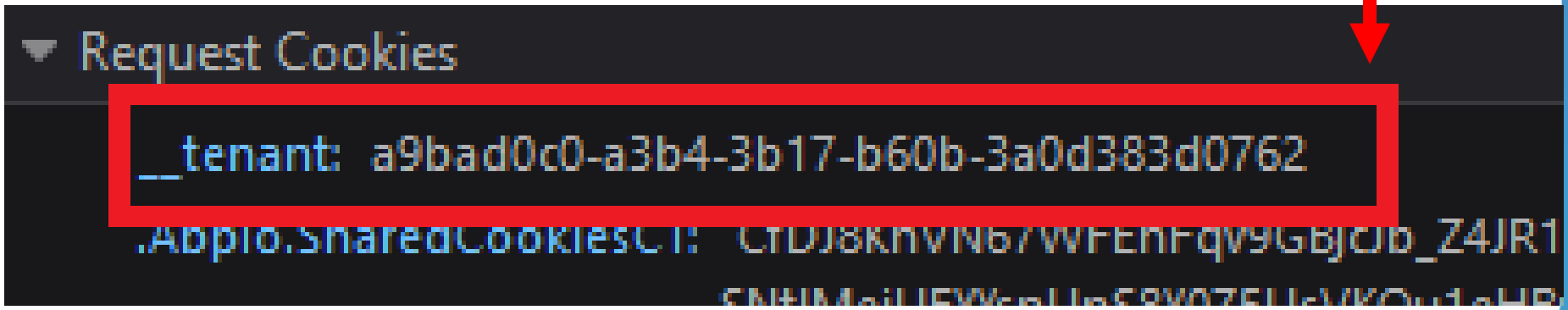
```
var requestHeader = httpContext.Request.Headers["__tenant"];
if (requestHeader.Any())
{
    context.Handled = true;
    context.TenantIdOrName = requestHeader.First();
}
```



Identifying the Active Tenant

5. Cookie

```
var cookieValue = httpContext.Request.Cookies["__tenant"];
if (cookieValue != null)
{
    context.Handled = true;
    context.TenantIdOrName = cookieValue;
}
```



Identifying the Active Tenant

6. Domain

```
var host = httpContext.Request.Host.Value;
var tenantName = Parse(host, "{0}.fabrikam.com");
if (tenantName != null)
{
    context.Handled = true;
    context.TenantIdOrName = tenantName;
}
```

https://acme.fabrikam.com

✓ Identifying the Active Tenant

Data Isolation

abp.io

Data Isolation – Traditional way

```
public class EfCoreBookRepository : EfCoreRepository ,IBookRepository
{
    private readonly CurrentTenant _currentTenant;

    protected List<Book> GetAllBooks()
    {
        return DbContext.Books.Where(x => x.TenantId == _currentTenant.Id).ToList();
    }
```

You normally do this

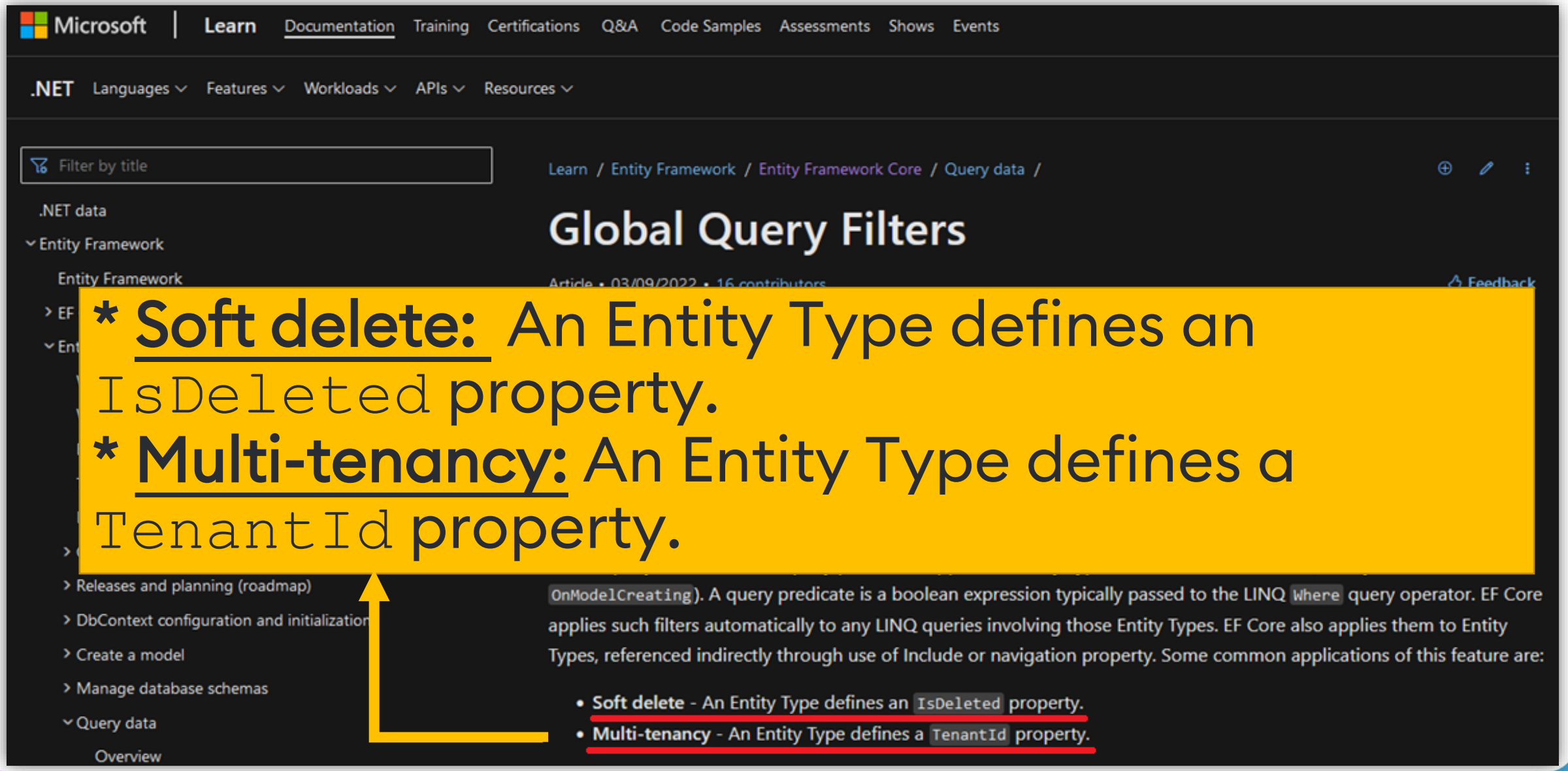
abp.io

Data Isolation

```
public class Book : Entity<Guid>, IMultiTenant
{
    public Guid? TenantId { get; set; }
    public string Name { get; set; }
}
```

abp.io

Data Isolation – EF Core



* **Soft delete:** An Entity Type defines an `IsDeleted` property.
* **Multi-tenancy:** An Entity Type defines a `TenantId` property.

Data Isolation – EF Core Manual Way

```
public class MyDbContext : DbContext
{
    private readonly CurrentTenant _currentTenant;
    public DbSet<Book> Books { get; set; }
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
        builder.Entity<Book>(b =>
        {
            b.HasQueryFilter(x => x.TenantId == _currentTenant.Id);
        });
    }
```

HasQueryFilter()
for global filtering

Data Isolation – EF Core

```
public class AbpContext<TDbContext> : DbContext, IAbpEfCoreDbConte
{
    protected virtual void ConfigureGlobalFilters<TEntity>(
        ModelBuilder modelBuilder, IMutableEntityType mutableEntit
        where TEntity : class
    {
        if (typeof(IMultiTenant).IsAssignableFrom(typeof(TEntity)))
        {
            Expression<Func<TEntity, bool>>
                multiTenantFilter = e => EF.Property<Guid>(e, "TenantId") == CurrentTenantId;

            modelBuilder.Entity<TEntity>().HasQueryFilter(multiTenantFilter);
        }
    }
```

1-) Find all entities implement *IMultiTenant*

2-) Create LINQ expression

3-) Add to global filters

Data Isolation – EF Core PROS & CONS


Easy to implement


Supports navigation properties as well


Works only with EF Core

abp.io

Data Isolation – EF Core PROS & CONS

😡 `IgnoreQueryFilters()` disables all filters

```
var allBlogs = dbContext.Blogs
    .Include(x => x.Posts)
    .IgnoreQueryFilters()
    .ToList();
```

abp.io

Data Isolation – EF Core PROS & CONS

😡 Can be defined for the root entity of the inheritance hierarchy

```
class Animal { /* Root entity type */ }

class BigAnimal : Animal { /* Subtype of Animal */ }

class SmallAnimal : Animal { /* Subtype of Animal*/ }
```

Define to Animal

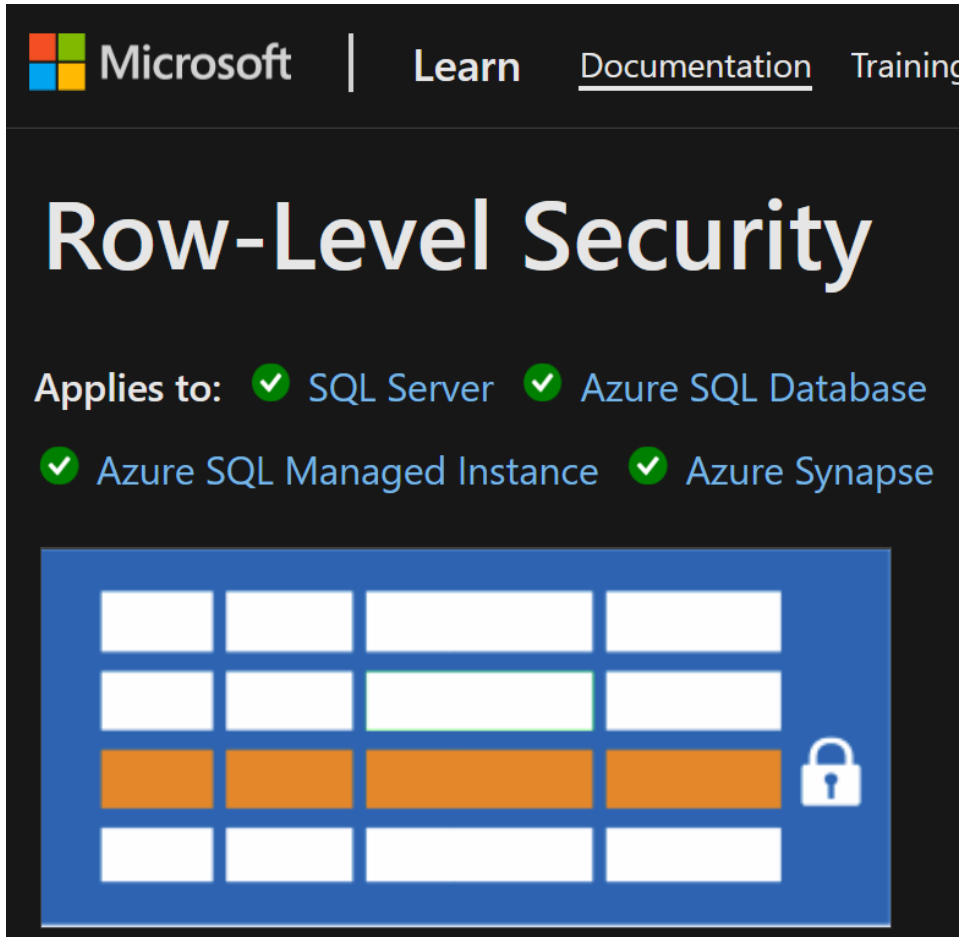
abp.io

Data Isolation – EF Core PROS & CONS

😡 Does not support Stored Procedures or T-SQL

```
var popular = dbContext.Blogs
    .FromSql($"EXECUTE dbo.spGetPopularBlogs")
    .ToList();


var all = dbContext.Blogs
        .FromSqlRaw("SELECT * FROM Blogs")
        .ToList();
```

Data Isolation – EF Core PROS & CONS



Database level solution

👉 Row Level Security

Rows filtered based on user roles, attributes

Restriction logic is done in the DB

https://learn.microsoft.com/en-us/sql/relational-databases/security/row-level-security

abp.io

Data Isolation – MongoDB

```
public virtual async Task<FilterDefinition<TEntity>> CreateEntityFilterAsync(TKey id,
{
    var filters = new List<FilterDefinition<TEntity>>
    {
        Builders<TEntity>.Filter.Eq(e => e.Id, id)
    };

    if (typeof(IMultiTenant).IsAssignableFrom(typeof(TEntity)))
    {
        filters.Add(Builders<TEntity>.Filter.Eq(e =>
            ((IMultiTenant)e).TenantId, CurrentTenant.Id));
    }

    return Builders<TEntity>.Filter.And(filters);
}
```

1-Find all IMultiTenant

2-Create filter expression

3-Add to our custom global filters

abp.io

✓ Identifying the Active Tenant

✓ Data Isolation

Set TenantId for New Entities

abp.io

Set TenantId for New Entities

```
public abstract class Entity : IEntity
{
    protected Entity()
    {
        if (this is not IMultiTenant entity)
        {
            return;
        }

        var tenantId = AsyncLocalCurrentTenantAccessor.Instance.Current?.TenantId;

        ObjectHelper.TrySetProperty(entity, x => x.TenantId, () => tenantId);
    }
```

Set TenantId by reflection

abp.io

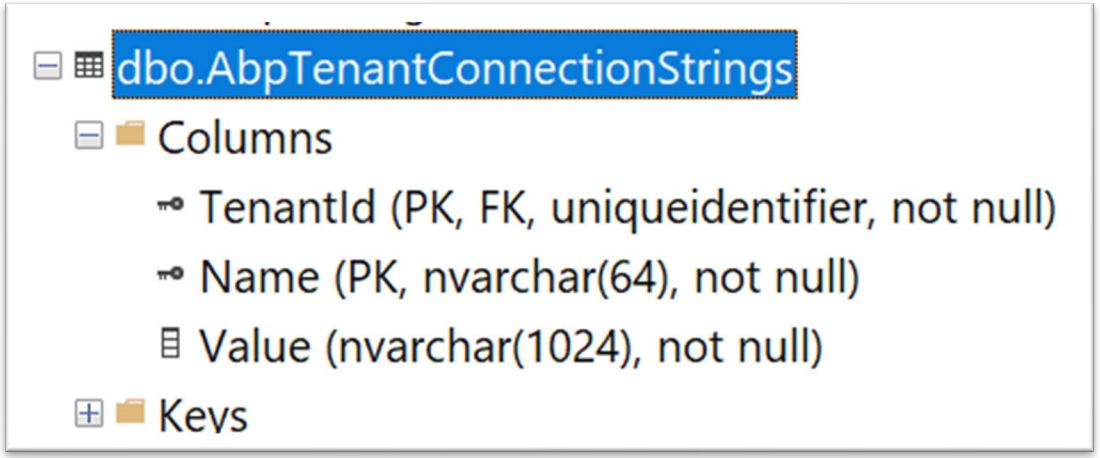
✓ Identifying the Active Tenant
✓ Data Isolation
✓ Set TenantId for New Entities

DB Connection String Selection

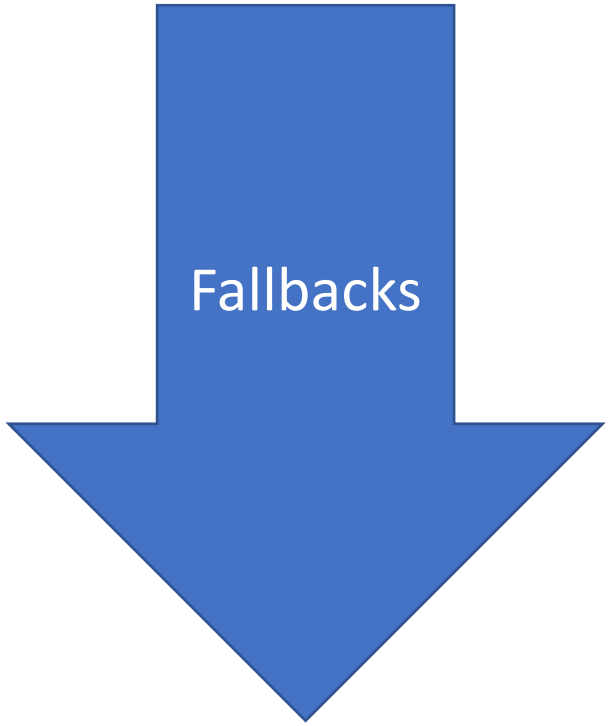
abp.io

Connection String Selection – DB

1. The **current tenant**



2. The **current module / microservice**
3. The **default** connection string



abp.io

Connection String Selection – Code

```
public class MultiTenantConnectionStringResolver : DefaultConnectionStringResolver
{
    public async Task<string> ResolveAsync()
    {
        var tenant = await FindTenant(_currentTenant.Id);
        if (tenant.ConnectionStrings.Any())
        {
            //Send tenant-specific connection string...
            var tenantDefaultConnectionString = tenant.ConnectionStrings.First();
            return await base.ResolveAsync(tenantDefaultConnectionString);
        }

        //No specific connection string! Send the default one
        return await base.ResolveAsync(Options.ConnectionStrings.Default);
    }
```

Dedicated DB

Shared DB

abp.io

✓ Identifying the Active Tenant
✓ Data Isolation
✓ Set TenantId for New Entities
✓ DB Connection String Selection

Changing the Active Tenant

abp.io

Changing the Active Tenant

```
public string GetTenantStatistics(Guid tenantId)
{
    using (_currentTenant.Change(tenantId))
    {
        //queries are filtered for this tenant
    }
}
```

Set active tenant

```
private IDisposable Change(Guid? tenantId, string? name = null)
{
    var originalTenant = _currentTenantAccessor.Current;
    _currentTenantAccessor.Current = new BasicTenantInfo(tenantId, name);

    return new DisposeAction<ValueTuple<ICurrentTenantAccessor, BasicTenantInfo?>>
    (static (state) => {
        var (currentTenantAccessor, originalTenant) = state;
        currentTenantAccessor.Current = originalTenant;
    }, (_currentTenantAccessor, originalTenant));
}
```

Revert back

Setting the Active Tenant in Middleware

```
public class MultiTenancyMiddleware : IMiddleware
{
    public async Task InvokeAsync(HttpContext context, RequestDelegate next)
    {
        using (_currentTenant.Change(_currentTenant.Id))
        {
            await next(context);
        }
    }
}
```

Set the current tenant within the middleware

```
var app = context.GetApplicationBuilder();

app.UseRouting();
app.UseAuthentication();

if (MultiTenancyConsts.IsEnabled)
{
    app.UseMiddleware<MultiTenancyMiddleware>();
}

app.UseAuthorization();
app.UseSwagger();
```

✓ Identifying the Active Tenant
✓ Data Isolation
✓ Set TenantId for New Entities
✓ DB Connection String Selection
✓ Changing the Active Tenant

Temporarily Disable Multi-Tenancy

abp.io

Disabling Multi-Tenancy Filter (Usage)

```
private readonly IDataFilter _filter;
public int GetTotalBookCount()
{
    using (_filter.Disable<IMultiTenant>())
    {
        return _bookRepository.GetCount();
    }
}
```

Returns book count without tenantId filter

Disabling Multi-Tenancy Filter (Implementation)

```
public class DataFilter : IDataFilter, ISingletonDependency
{
    private readonly ConcurrentDictionary<Type, object> _filters;

    public IDisposable Disable<TFilter>() where TFilter : class
    {
        GetFilter<TFilter>().Disable();
        return new DisposeAction(() => Enable());
    }

    public IDisposable Enable<TFilter>() where TFilter : class
    {
        GetFilter<TFilter>().Enable();
        return new DisposeAction(() => Disable());
    }
```

✓ Identifying the Active Tenant
✓ Data Isolation
✓ Set TenantId for New Entities
✓ DB Connection String Selection
✓ Changing the Active Tenant
✓ Temporarily Disable Multi-Tenancy

Database Migration

abp.io

Database Migration

Approach-1: Make DB migration with a custom tool

😊 Easy to implement. All tenants are in the same version
😡 May get too long time for big number of tenants and data.
😡 All tenants wait for all upgrade progress

Approach-2: Run migration on first DB access

😊 Upgrading is distributed to time. A tenant does not wait for another
😡 First user may wait too much and see timeout exception.
😡 Hard to implement (concurrency problems)!

abp.io

Database Migration – Ideal Way

Approach-3: Make two types application servers.

Upgraded tenants use the new application, other tenants use the old application

- 😊 Minimum wait time for a tenant
- 😊 Upgrading can be scheduled for tenants
- 😊 Run A/B tests and see bugs before anyone else
- 😡 Requires multiple app servers
- 😡 Hard to maintain and monitor

abp.io

✓ Identifying the Active Tenant
✓ Data Isolation
✓ Set TenantId for New Entities
✓ DB Connection String Selection
✓ Changing the Active Tenant
✓ Temporarily Disable Multi-Tenancy
✓ Database Migration

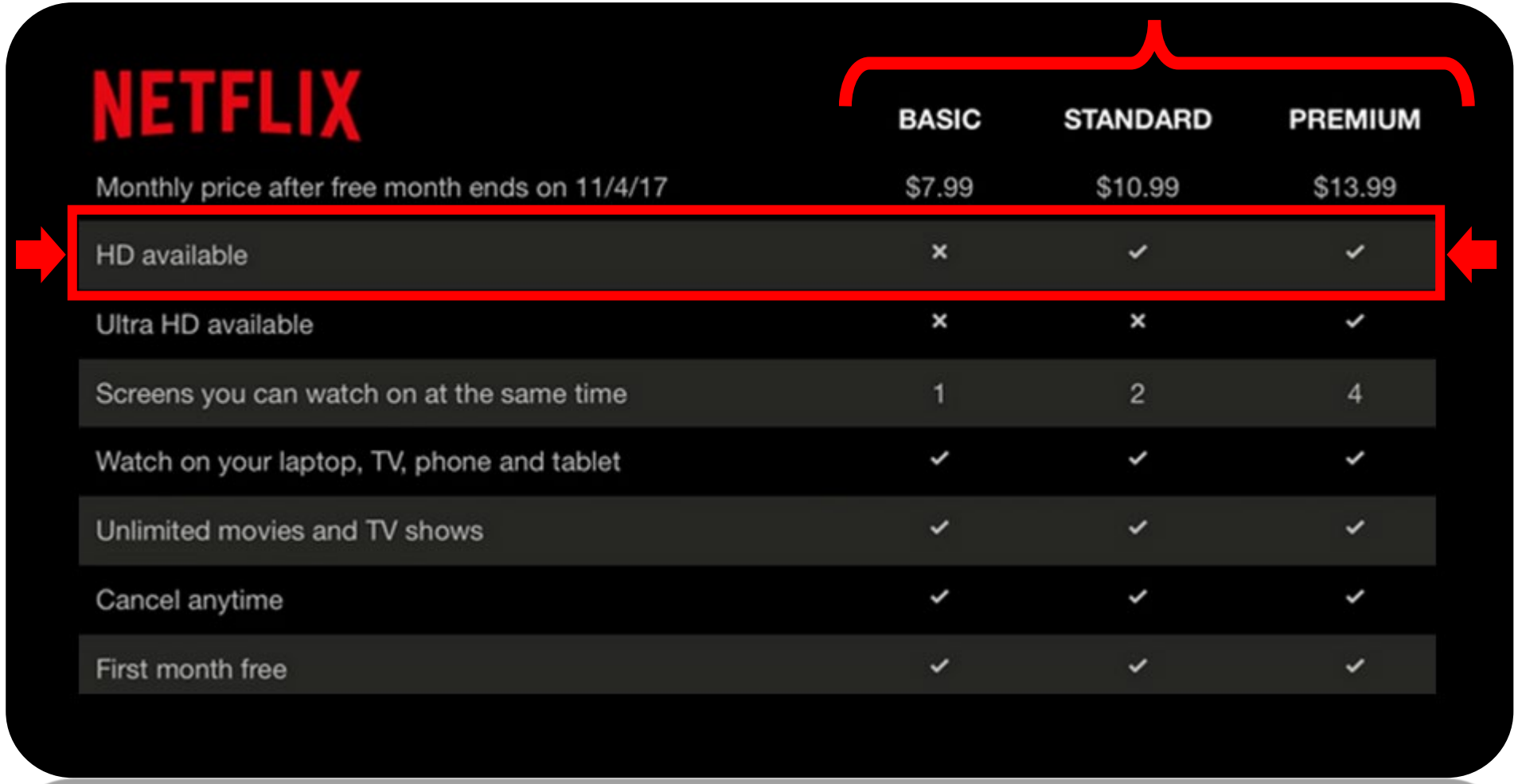
Feature System

abp.io

The Feature System

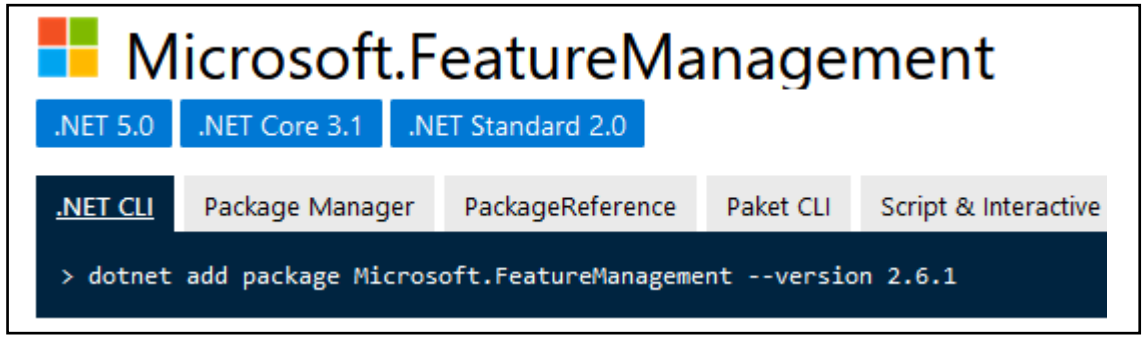
Editions

Features



NETFLIX	BASIC	STANDARD	PREMIUM
Monthly price after free month ends on 11/4/17	$7.99	$10.99	$13.99
HD available	✗	✓	✓
Ultra HD available	✗	✗	✓
Screens you can watch on at the same time	1	2	4
Watch on your laptop, TV, phone and tablet	✓	✓	✓
Unlimited movies and TV shows	✓	✓	✓
Cancel anytime	✓	✓	✓
First month free	✓	✓	✓

The Feature System – Microsoft's Solution



Defined **only for Boolean** values
Usually for **A/B testing**
No multi-tenancy support

appsettings.json

```
{
  "FeatureManagement": {
    "NewBanner": "false"
  }
}
```

```
[FeatureGate("NewBanner")]
public async Task<bool> RenderBanner()
{
    if (await _featureManager.IsEnabledAsync("NewBanner"))
    {
        // Show the new banner
    }
}
```

abp.io

The Feature System – Define features

```
public class MyFeatureProvider : FeatureDefinitionProvider
{
    public override void Define(IFeatureDefinitionContext ctx)
    {
        ctx.AddGroup("VideoFeatures")
          .AddFeature("IsHdAvailable");
    }
}
```

Features are stored in a readonly list

abp.io

The Feature System – Check the features

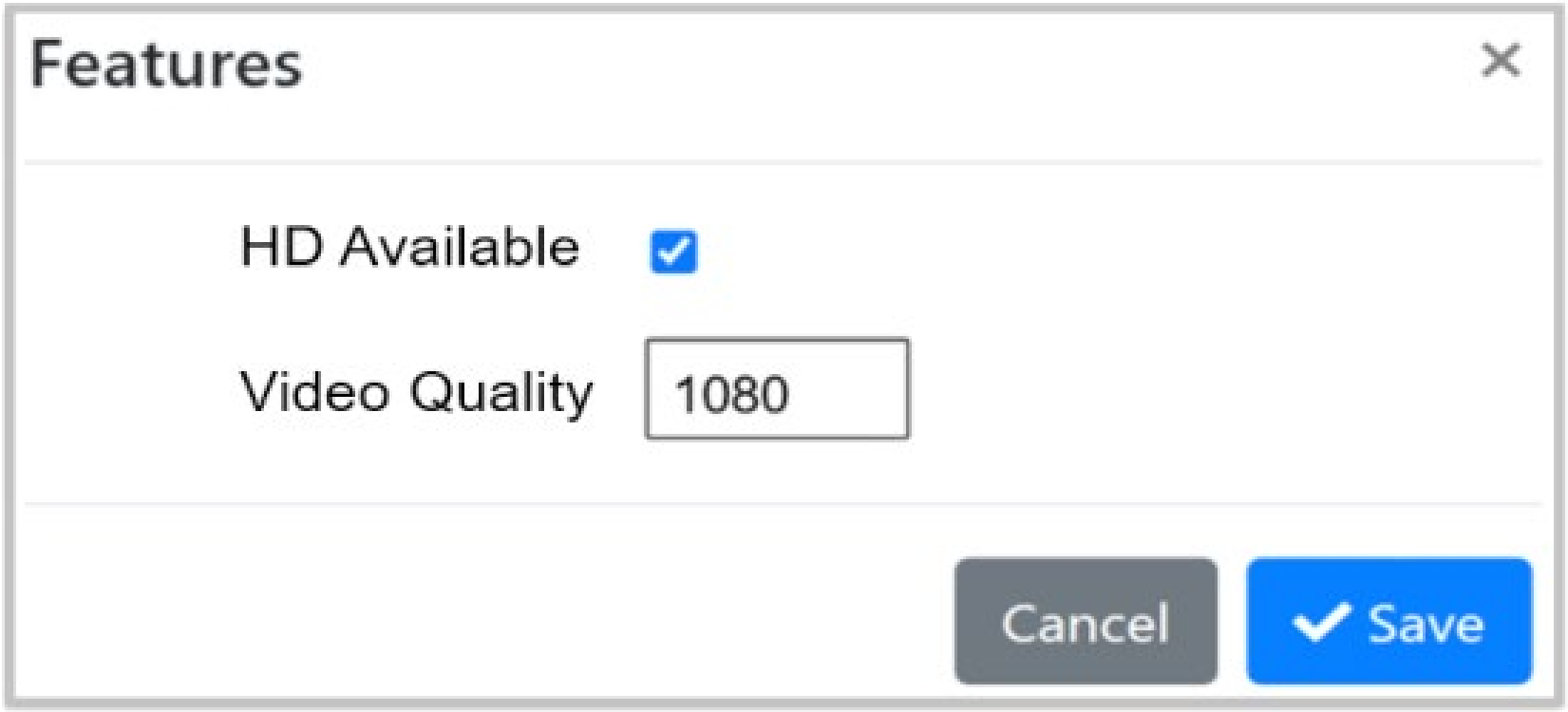
```
private readonly IFeatureChecker _checker;

[RequiresFeature("IsHdAvailable")]
public async Task<Stream> StreamHdVideoAsync()
{
    if (await _checker.IsEnabledAsync("IsHdAvailable"))
    {
        //OK, stream it...
    }
}
```

Declarative check

Conditional check

The Feature System – UI

Use a **Management UI** to manage features for tenants



abp.io

Thank you for joining ☺

https://twitter.com/**alperebicoglu**

https://github.com/**ebicoglu**

https://medium.com/**@alperonline**

Download this presentation:
https://github.com/ebicoglu/presentations

open-source
web application
framework
https://abp.io