



MIDWEST ARCHITECTURE COMMUNITY COLLABORATION 2020

NOVEMBER 5, 2020

MACC 2020 Microservices Architecture

Practical guide to Microservices and their
use in the real world

MACC MISSION

- The Midwest Architecture Community Collaboration's (MACC) purpose is to bring all domains of architecture together to share information and techniques of interest to all of us. It is our shared belief that through collaboration, we can better understand and promote the significance of architecture to business success.

WORKED FINE IN DEV



OPS PROBLEM NOW

SOA DESIGN AND DEVELOPMENT

- Service Oriented Architecture (SOA)
 - .NET Remoting
 - Data Synchronization
 - WCF and SOAP protocol

 - An then there was Roy Fielding...
 - invented an architectural style called:
 “Representational State Transfer”

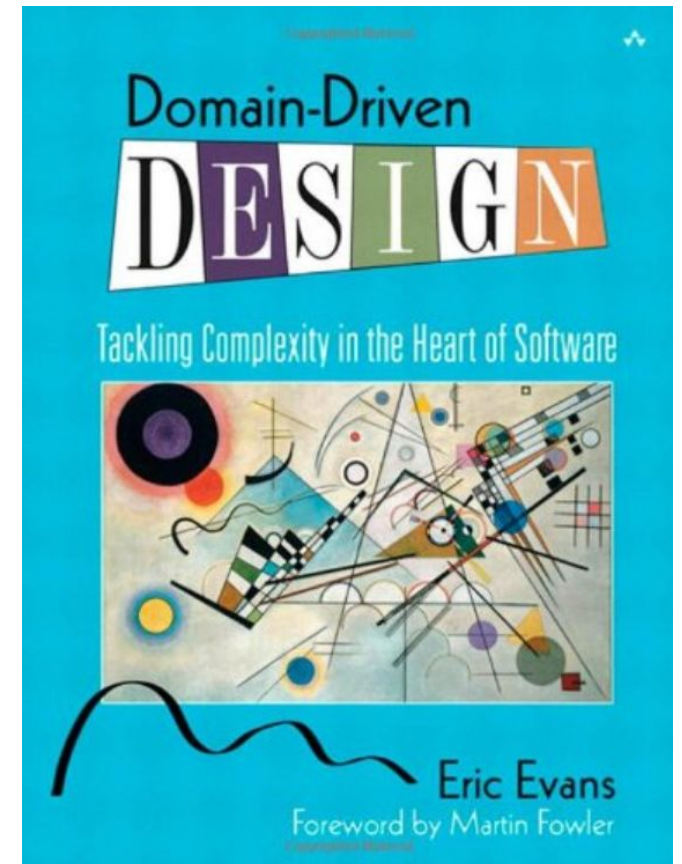
 - Or what we all refer to now as.. the “RESTFul” API

SOA AND DDD

DDD???

- Domain Driven Design
- Middle Tier Development First Approach
- Strong understanding of your business domain

- ... Eric Evans Book on DDD



MICROSERVICES & DEVOPS

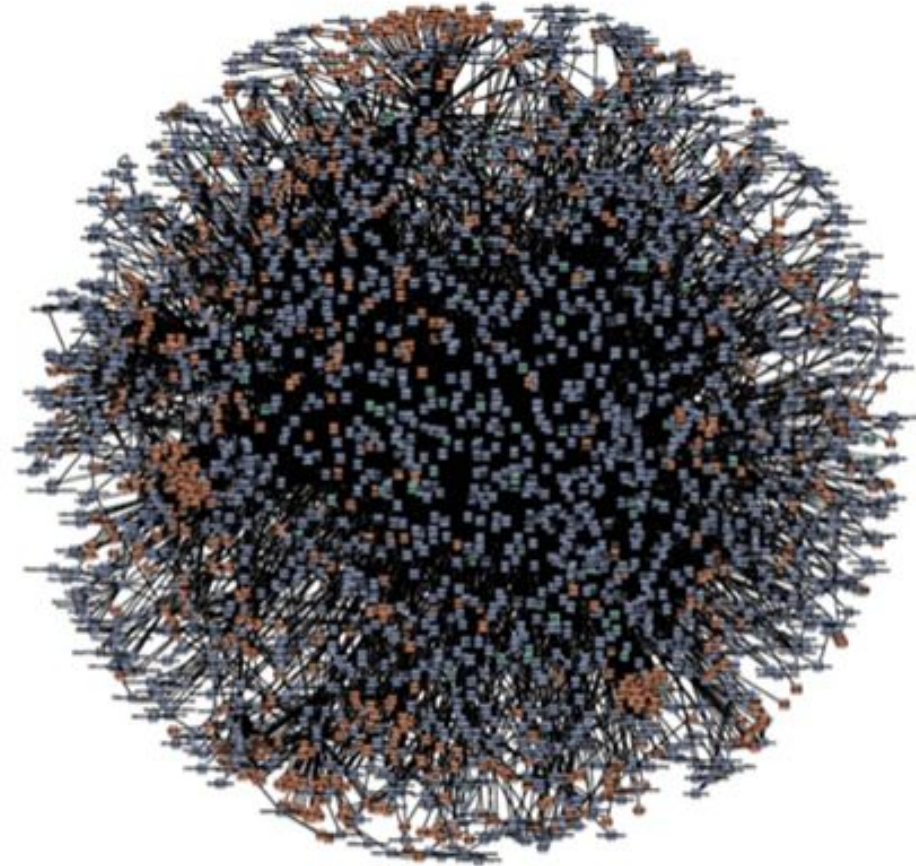
Microservices Architecture and Design

- Small units of work
- Isolation of components, code
- Automated deployments (CI/CD)

NETFLIX “DEATH STAR” - +700 MICROSERVICES



AMAZON.COM - ???? MICROSERVICES

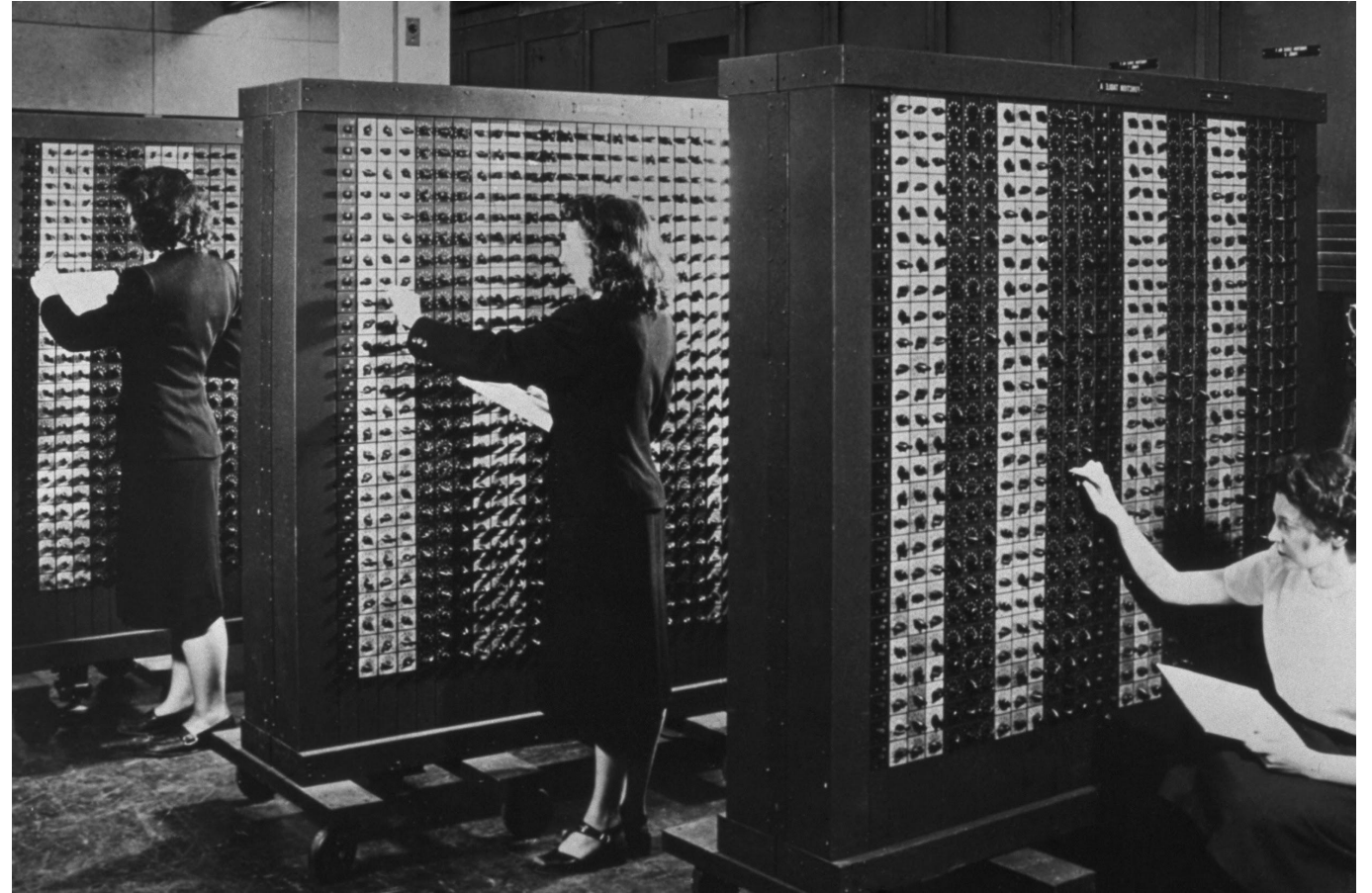


HISTORY

“You need to know the past...

...to understand the future”

~ Dr. Carl Segan



CLIENT / SERVER APPROACH

Connecting clients to backend corporate servers

- Solved out data problem – data is in one place
- Did not solve our distribution/deployment problem

THE “BOOM” OF THE .COM

- Webpages or Web-Resources
 - You have everything under the sun – the wild west
 - ColdFusion development
 - PHP development
 - Backend processing
 - And eventually the “Post Back” development

API DEVELOPMENT

- Webpages or Web-Resources
 - The idea of a stand alone webpage the served up data
 - Browser applications could ingest this data

EVERYONE LOVES “JAVASCRIPT”

- The invention of SPA (Single Page Applications)
 - We needed a better User Experience
 - DevExpress / Infragistics / Telerik Controls
 - Ruby on Rails
 - Microsoft’s “Silverlight”... yes, you can breath now
 - Everyone was struggling to have a better UI

THE MARRIAGE WITH SPA

- The invention of Microservices
 - Miniature little APIs
 - Monolithic Systems are now in place... What do we do?

THE MARRIAGE WITH SPA

- The invention of Microservices

- Miniature little APIs
- We can isolate business logic and ultimately business domains
- Monolithic Systems are now in place... What do we do?

MICROSERVICES

Structure / Layout / Design



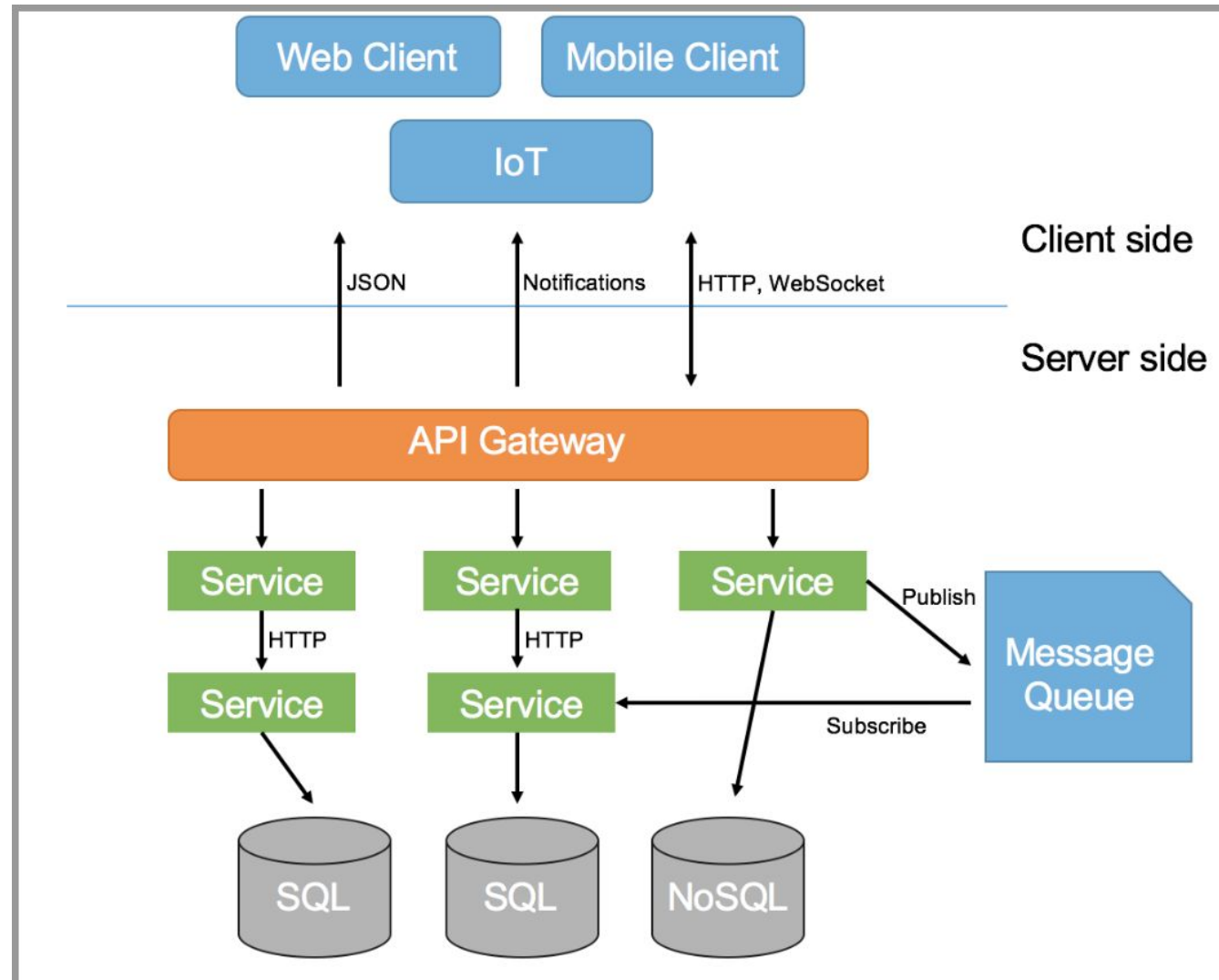
GOALS – ENTIRE SYSTEM

- Overall goals for the entire set of Microservices
 - Databases are isolated/separated
 - Microservices are isolated/separated
 - Different versions can interact with each other (with consumers)
 - Quick/Automated deployments
 - Consumers can plug and play

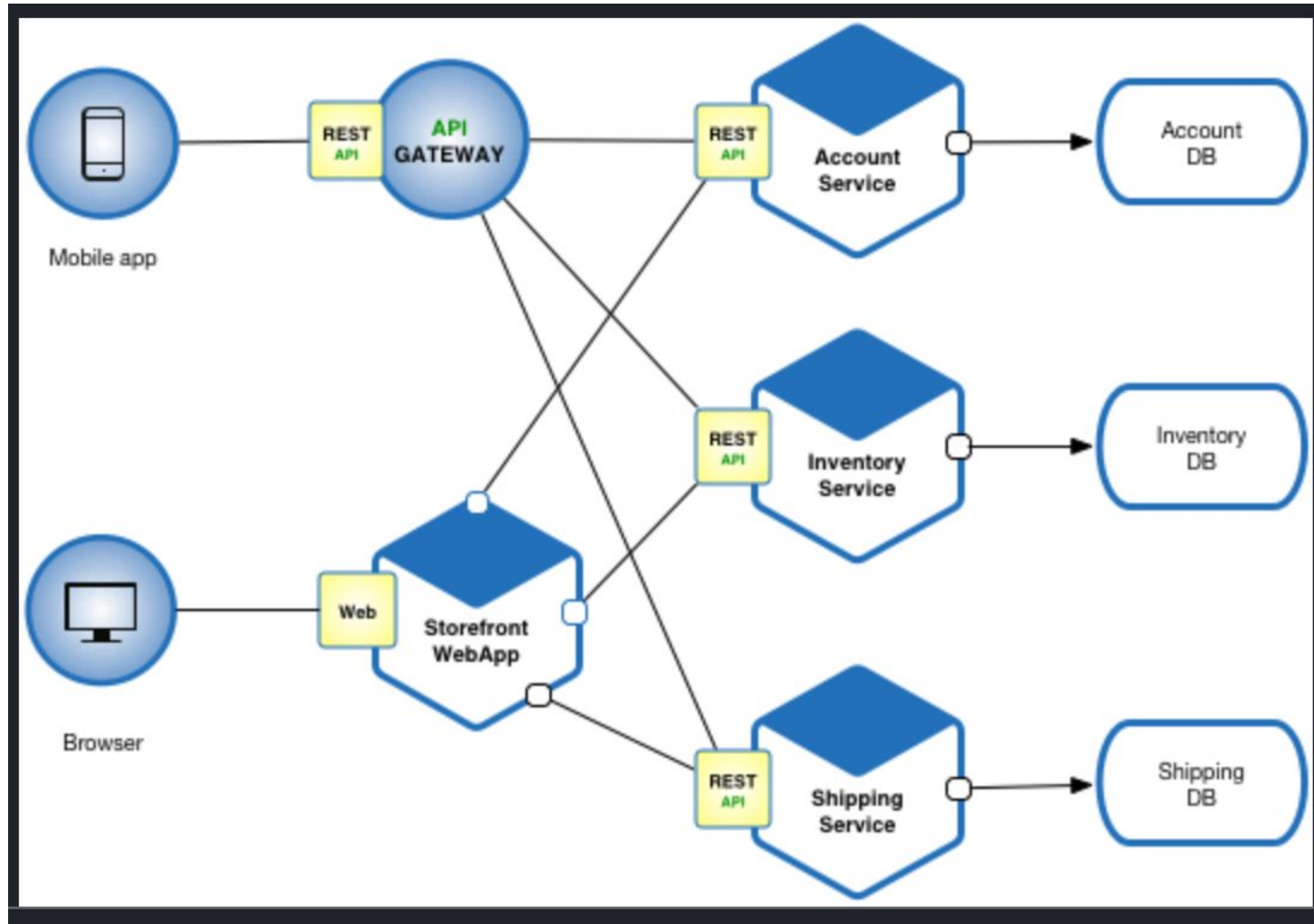
OVERALL STRUCTURE

- Components – These are your basics
 - Microservices
 - “Atomic” and “Composite”
 - Single/Isolated data storage system (each Microservice)
 - API Gateway (front side caching)
 - Data Warehouse (backend data movement)
 - Messaging System

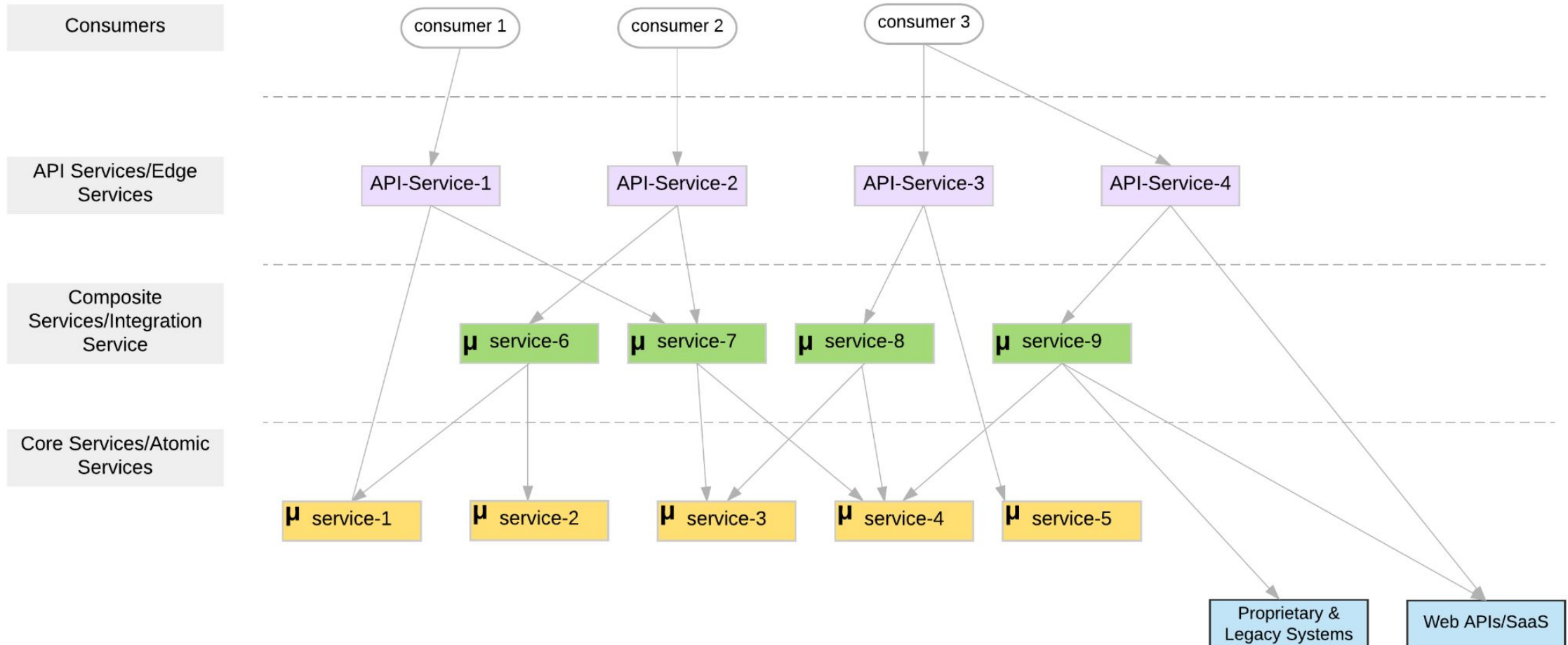
OVERALL STRUCTURE



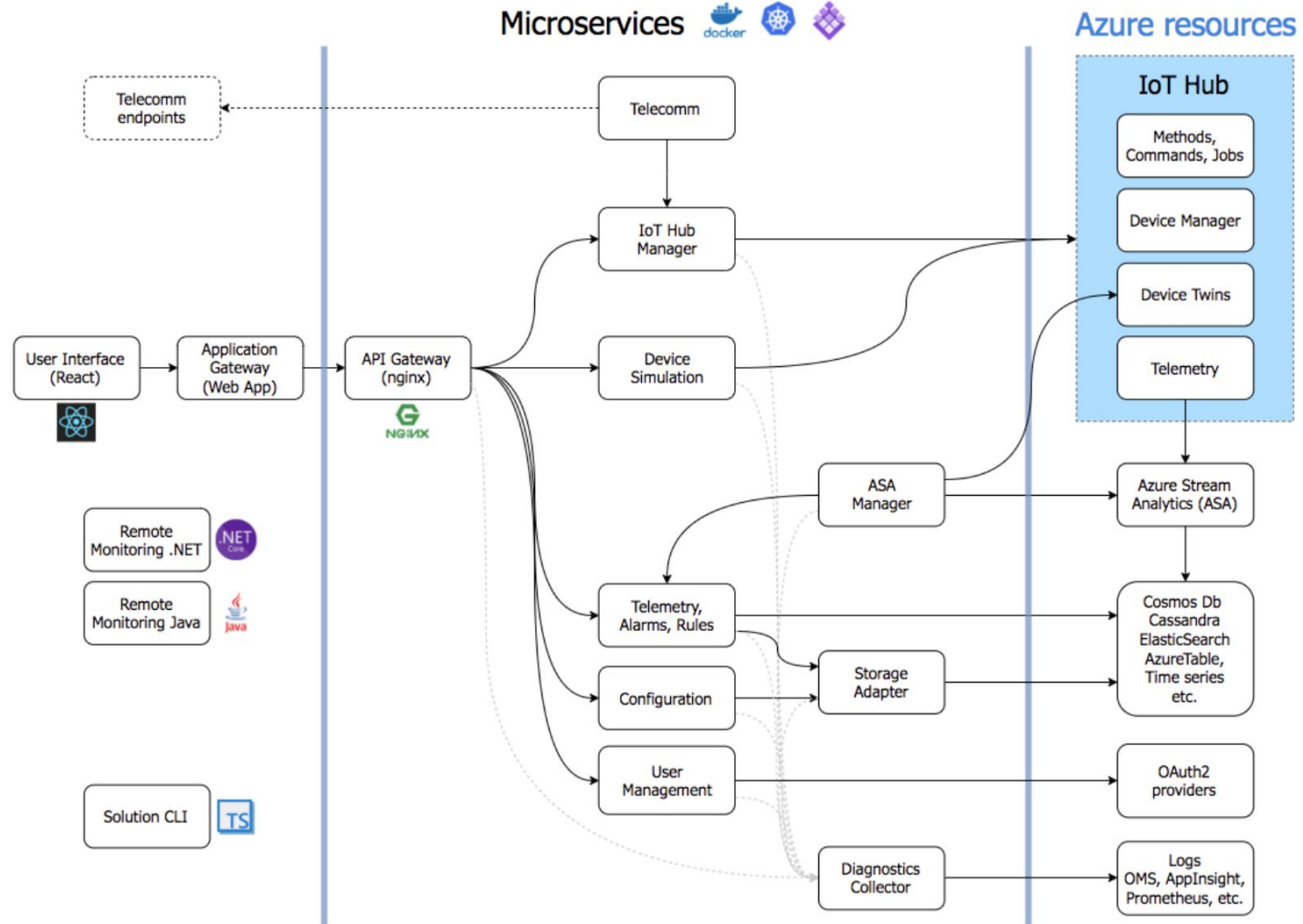
OVERALL STRUCTURE



OVERALL STRUCTURE



OVERALL STRUCTURE



GOALS – SINGLE SERVICE

- Overall goals with a single Microservice
 - Built and checked in – 8 hours flat
 - Built / Deployed / Tested in 16 hours flat (to DEV)
 - Database/Data-storage - created automatically
 - Documentation is automatically created (i.e. Swagger)

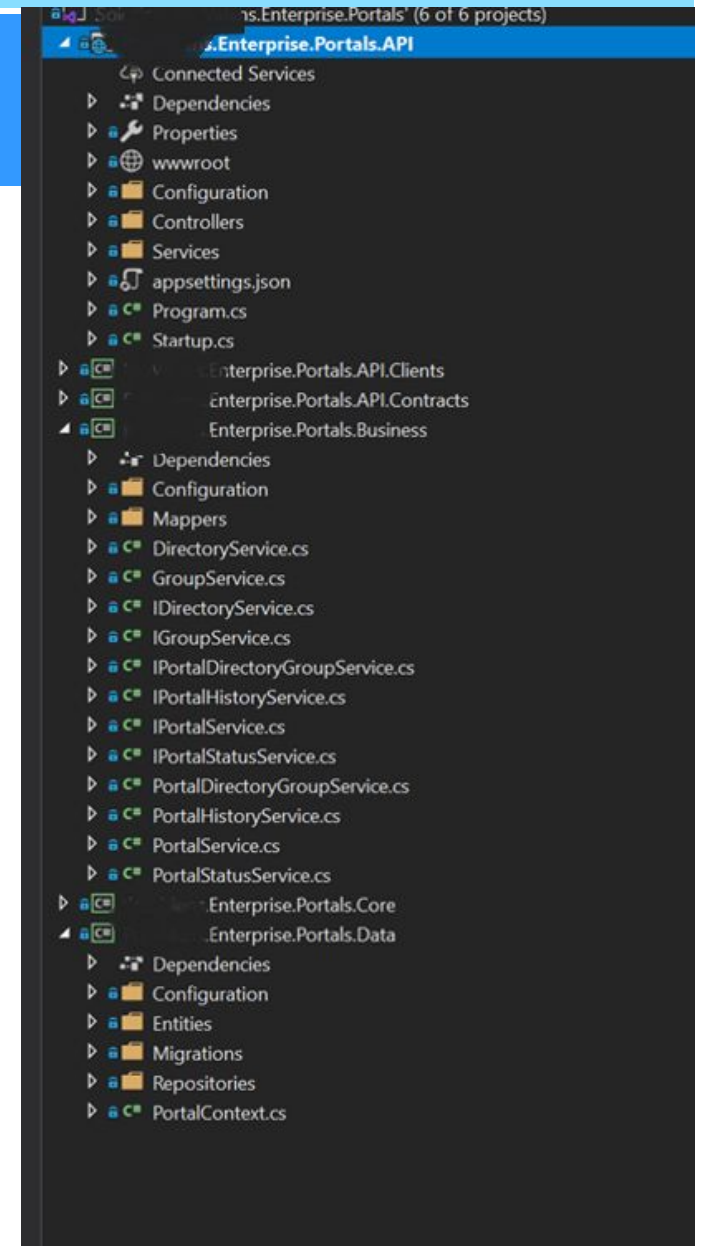
INTERNAL STRUCTURE

- **Microservice Layout for success**
 - 3-Tier Architecture
 - Fully RESTful (do not deviate)
 - URL Versioning (backwards compatible)
 - Data storage ORM tool (very helpful)
 - Unit tests (automated)

INTERNAL STRUCTURE

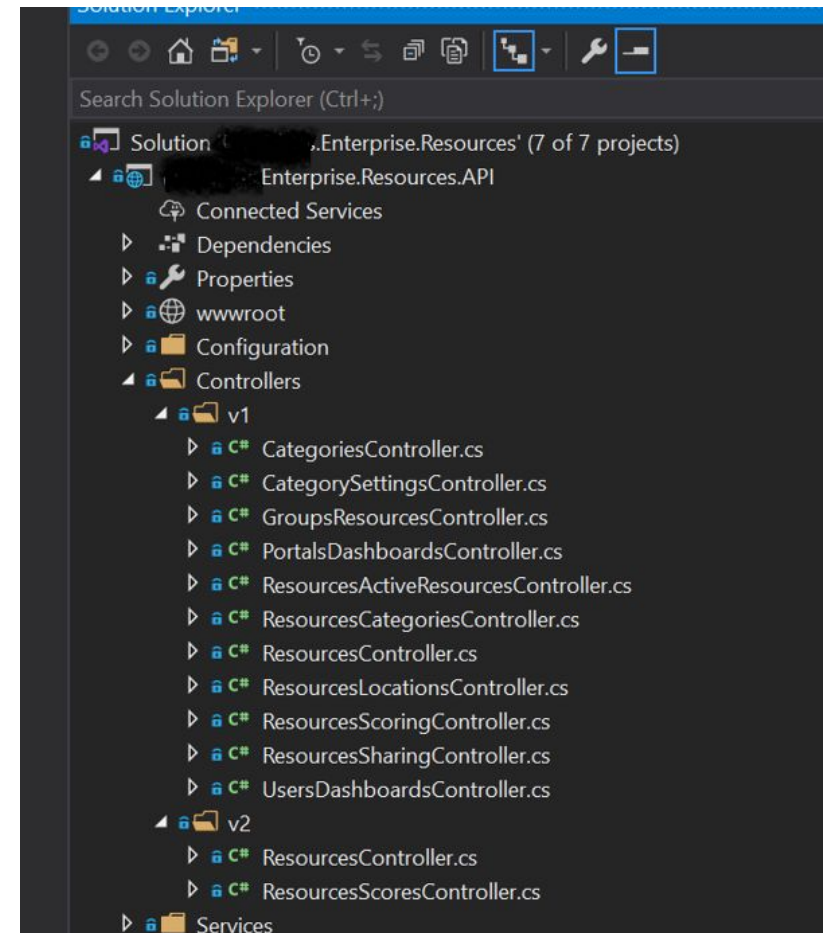
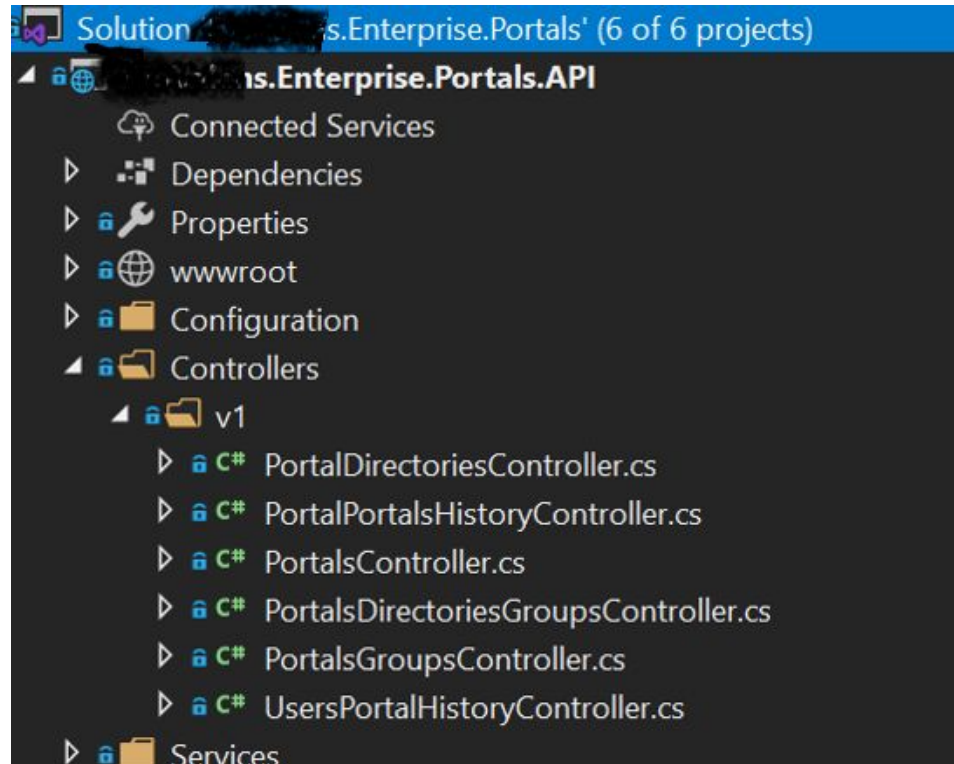
■ 3-Tier Architecture

- Presentation Layer (api endpoints)
- Business Layer (all logic here)
- Data Layer (almost fully generated)



INTERNAL STRUCTURE

■ Presentation Layer – URL Versioning



INTERNAL STRUCTURE

■ Presentation Layer – URL Versioning

```
/// <seealso cref="Microsoft.AspNetCore.Mvc.Controller" />
[Produces(contentType: "application/json")]
[ApiVersion("2.0", Deprecated = false)]
[Route(template: "v{version:apiVersion}")] //Route Prefix
public class ResourcesController : Controller
{
    /// <summary>
    /// RESTful endpoint for retrieving a list of resources
    /// </summary>
    /// <returns></returns>
    [HttpGet, Route(template: "resources"), MapToApiVersion("2.0")]
    public IActionResult Get()
    {
        //just mocked out for now
        var list = new List<Resource>();
        list.Add(item: new Resource());
        return Ok(list);
    }

    /// <summary>
    /// Gets the specified resource identifier.
    /// </summary>
    /// <param name="resourceId">The resource identifier.</param>
    /// <returns></returns>
    [HttpGet, Route(template: "resources/{resourceId}"), MapToApiVersion("2.0")]
    public IActionResult Get(Guid resourceId)
    {
        //just mocked out for now
        return Ok(new Resource());
    }
}
```

INTERNAL STRUCTURE

- Business Layer
 - All logic resides here

```
public List<Resource> Get(Guid? dashboardId, Guid? portalId, Guid? classificationId, string keywords = null, bool activeOnly = true, string tags = null, float? latitude = null)
{
    _logger.LogTrace(message: "Executing GET");
    var resources :IQueryable<Resource> = _repository.Get()
        .Where(x:Resource => x.IsActive && !x.IsDeleted);

    //These are all in separate methods so that the even the null filters don't get applied if not needed
    resources = ApplyClassificationFilter(resources, classificationId);
    resources = ApplyTagFilters(resources, tags);
    resources = ApplyTitleFilter(resources, title);
    resources = ApplyKeywordFilters(resources, keywords);
    resources = ApplyCategoryFilters(resources, categoryIds);
    resources = ApplyLocationFilter(resources, latitude, longitude, distance, location);

    if (portalId != null && !admin)
    {
        resources = ApplyPortalIdFilter(resources, portalId);
    }

    // Order By
    if (groupId != null)
    {
        resources = ApplyGroupIdFilter(resources, groupId);
        resources = resources.OrderBy(x:Resource => x.GroupResources.First(y:GroupResource => y.IsActive && !y.IsDeleted && y.GroupId == groupId).DisplayOrder);
    }
    else
    {
        resources = ApplyIsActiveInDashboardFilter(resources, dashboardId, activeOnly);
        resources = resources.OrderByDescending(x:Resource => x.ModifiedByDate);
    }

    // Pagination
    if (paging?.RequestCount == true)
    {
        paging.TotalCount = resources.Count();
    }

    int page = paging?.Page ?? 0;
    int pageSize = paging?.PageSize ?? 1000;
    resources = resources.Skip(pageSize * page).Take(pageSize);

    var models :List<Resource> = resources.Select(x:Resource => x.ToResourceModel(dashboardId, _blobService.Configuration)).ToList();
    _logger.LogInformation(message: "Successfully executed GET");
    return models;
}
```

INTERNAL STRUCTURE

■ Business Layer – DO NOT EXPOSE DATA ENTITIES!

```
public static Data.Entities.Dashboard ToEntity(this Dashboard model)
{
    var entity = new Data.Entities.Dashboard();
    entity.DashboardId = model.DashboardId ?? Guid.Empty;
    return model.UpdateEntity(entity);
}

public static Data.Entities.Dashboard UpdateEntity(this Dashboard model, Data.Entities.Dashboard entity)
{
    entity.Name = model.Name;
    entity.PortalId = model.PortalId;
    entity.UserId = model.UserId;
    entity.IsDefault = model.IsDefault;
    return entity;
}
```

```
public static Dashboard ToModel(this Data.Entities.Dashboard entity)
{
    var model = new Dashboard();
    return entity.UpdateModel(model);
}

public static Dashboard UpdateModel(this Data.Entities.Dashboard entity, Dashboard model)
{
    model.DashboardId = entity.DashboardId;
    model.Name = entity.Name;
    model.PortalId = entity.PortalId;
    model.UserId = entity.UserId;
    model.IsDefault = entity.IsDefault;
    return model;
}
```

INTERNAL STRUCTURE

■ Data Layer – Automate it!

```
ResourceScore.cs
└─ Migrations
  └─ 20180312204800_InitialMigration.cs
  └─ 20180313014402_DashboardTable.cs
  └─ 20180507003128_ActiveResources_DashboardId.cs
  └─ 20180507004230_Dashboards_RenamedOwnerId.cs
  └─ 20180507005132_Dashboards_UserId.cs
  └─ 20180507025004_Dashboards_IsDefault.cs
  └─ 20180613161805_AddressIdAdded.cs
  └─ 20180618145255_LocationAdded.cs
  └─ 20180618230009_ResourceIdAdded.cs
  └─ 20180621193859_ChangedLongDescriptionTo8000Cha
  └─ 20180621235715_NewLatLongTypes.cs
  └─ 20180626183653_LocationCollection.cs
  └─ 20180627145003_ColorTextHexAdded.cs
  └─ 20180627174724_NullableLocation.cs
  └─ 20180709145254_DisplayType.cs
  └─ 20180803225202_ResourceCategories.cs
  └─ 20180805210347_ChangeLongDescriptionToMAX.cs
  └─ 20180817193750_GroupResource.cs
  └─ 20180820144106_GroupResourceEntityChange.cs
  └─ 20180928193704_NewDatesInResourceTable.cs
  └─ 20181017154210_AddedColumnLongDescExtendedToResourceTable.cs
  └─ ResourcesContextModelSnapshot.cs
└─ Repositories
```

```
public IQueryable<ActiveResource> Get()
{
    var resources :IQueryable<ActiveResource> = _context.ActiveResources
        .Where(x :ActiveResource=>x.IsActive);
    return resources;
}

public ActiveResource Get(Guid resourceId, Guid dashboardId)
{
    return _context.ActiveResources.FirstOrDefault(x :ActiveResource => x
}

public ActiveResource Create(ActiveResource entity)
{
    _context.Add(entity);
    _context.SaveChanges();
    return entity;
}

public ActiveResource Update(ActiveResource entity)
{
    _context.Attach(entity);
    _context.SaveChanges();
    return entity;
}
```

DEVOPS (DEVELOPMENT & OPERATIONS)

Why is this so important?



Jenkins



TeamCity



Travis CI



Octopus Deploy



GitLab



Bamboo



Azure DevOps

DEVOPS

- Development/Deployment Operations
 - Need to get your code promoted INSTANTLY
 - CI/CD – Continuous Integration / Continuous Deployment
 - Automated testing and code coverage

DEVOPS

■ Automated builds...

Jobs in run

Jobs

Job	Duration
Phase 1	1m 42s
Initialize Job	2s
Checkout	5s
Restore	57s
Build	28s
Test	1s
Publish	4s
Publish Artifact	1s
Post-job: Checkout	<1s
Finalize Job	<1s

Build
















```
1 Starting: Build
2 =====
3 Task       : .NET Core
4 Description : Build, test, package, or publish a dotnet application, or run a custom dotnet command. For package commands, supports NuGet.org and
5 Version    : 2.151.0
6 Author     : Microsoft Corporation
7 Help       : [More Information](https://go.microsoft.com/fwlink/?linkid=832194)
8 =====
9 C:\windows\system32\chcp.com 65001
10 Active code page: 65001
11 "C:\Program Files\dotnet\dotnet.exe" build d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj --confi
12 Microsoft (R) Build Engine version 15.9.20+g88f5fadfbc for .NET Core
13 Copyright (C) Microsoft Corporation. All rights reserved.
14
15   Restoring packages for d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj...
16   Restoring packages for d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj...
17   Restore completed in 187.67 ms for d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj.
18   Restore completed in 658.81 ms for d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj.
19   d:\a\1\s\F...Enterprise.Portals.API.Contracts -> d:\a\1\s\F...Enterprise.Portals.API.Contracts\bin\release\netcoreapp2.1\F...Enterprise.P
20   d:\a\1\s\F...Enterprise.Portals.API.Contracts -> d:\a\1\s\F...Enterprise.Portals.API.Contracts\bin\release\netcoreapp2.1\F...Enterprise.Por
21
22 Build succeeded.
23     0 Warning(s)
24     0 Error(s)
25
26 Time Elapsed 00:00:10.96
27 "C:\Program Files\dotnet\dotnet.exe" build d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj --c
28 Microsoft (R) Build Engine version 15.9.20+g88f5fadfbc for .NET Core
29 Copyright (C) Microsoft Corporation. All rights reserved.
30
31   Restore completed in 42.66 ms for d:\a\1\s\F...Enterprise.Portals.API.Contracts\...Enterprise.Portals.API.Contracts.csproj.
32   d:\a\1\s\F...Enterprise.Portals.API.Contracts -> d:\a\1\s\F...Enterprise.Portals.API.Contracts\bin\release\netcoreapp2.1\F...Enterprise.P
33
```

DEVOPS

- “Releasing” code
 - Pushing code out

Releases Deployments Analytics

Releases

	Release-094 for build s.API.Portal-2019.05.19.002  s.API.Portal-2019.05.19.002  master
	Release-093 for build s.API.Portal-2019.05.19.001  API.Portal-2019.05.19.001  master
	Release-092 for build s.API.Portal-2019.04.10.002  s.API.Portal-2019.04.10.002  master
	Release-088 for build s.API.Portal-2018.11.16.003  s.API.Portal-2018.11.16.003  master
	Release-082 for build s.Enterprise.Portal.API-2018.09.17.001  s.Enterprise.Portal.API-2018.09.17.001  master

Created

Stages

5/18/2019, 10:00:33 PM

Developm...

QA

UAT

Production

5/18/2019, 8:39:55 PM

Developm...

QA

UAT

Production

4/10/2019, 10:04:51 AM

Developm...

QA

UAT

Production

11/16/2018, 11:51:51 AM

Developm...

QA

UAT

Production

9/19/2018, 1:13:00 PM

Developm...

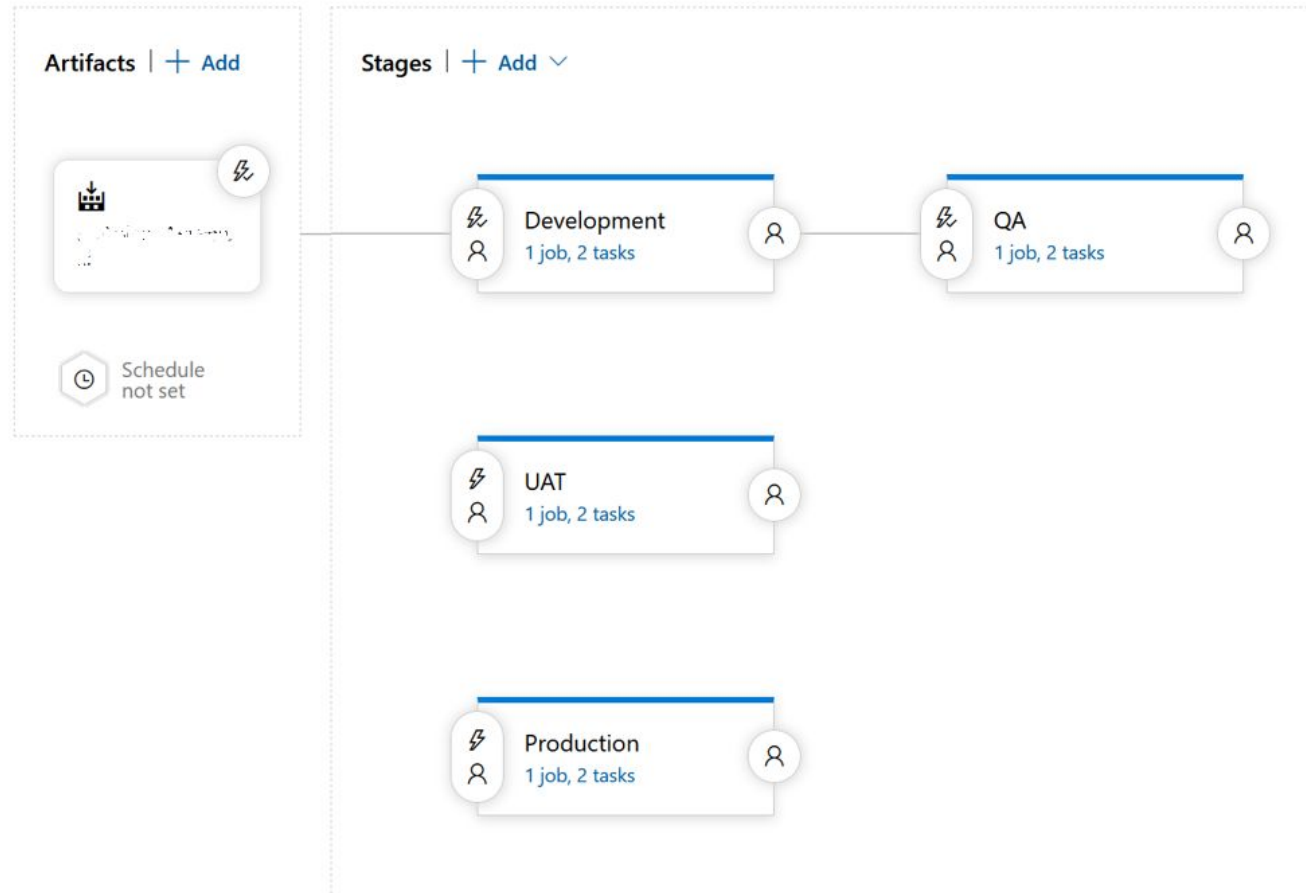
QA

UAT

Production

DEVOPS

- “Releasing” code
 - Pushing code out




























DEVOPS

- “Releasing” code
 - Pushing code out

The screenshot displays the 'Run on agent' task details in an Azure DevOps pipeline. The task is part of a 'Development' deployment process and has succeeded. The task list includes: 'Install Transform Package' (npm), 'Run Transformation' (gulp), 'Delete Transformation Package' (Delete files), 'Deploy: us-nc-dev-prv-portalui1' (Azure App Service deploy), and 'Swap Slots: STAGING to PROD' (Azure App Service manage). The 'Run on agent' task is expanded to show a list of sub-tasks, all of which have succeeded: 'Initialize Job', 'Download artifact - ProValens.UI.DFA - ProValens.UI.DFA', 'Install Transform Package', 'Run Transformation', 'Delete Transformation Package', 'Deploy: us-nc-dev-prv-landingsiteui1', 'Swap Slots: STAGING to PROD', and 'Finalize Job'. The pool is 'Hosted VS2017' and the agent is 'Hosted Agent'.

DEVOPS

■ Creating components in the cloud automatically

us-nc-prod-portalui2 Web App Running 	us-nc-prod-registrati... Web App Running 	us-nc-prod-userui Web App Running 	us-nc-prod-adminui2 Web App Running 	us-nc-prod-landings... Web App Running 
us-nc-prod-portalapi API app Running 	us-nc-prod-organiza... API app Running 	us-nc-prod-resource... API app Running 	us-nc-prod-securitie... API app Running 	us-nc-prod-userprofi... API app Running 
us-nc-prod-searchapi API app Running 	us-nc-prod-imageapi API app Running 	us-nc-prod-notificati... API app Running 	us-nc-prod-categori... API app Running 	us-nc-prod-location... API app Running 
		us-nc-prod-provalens SQL server Available 		
us-nc-prod-portals SQL database Online 	us-nc-prod-resources SQL database Online 	us-nc-prod-organiza... SQL database Online 	us-nc-prod-security SQL database Online 	us-nc-prod-userprofi... SQL database Online 
	us-nc-prod-images SQL database Online 	us-nc-prod-notificati... SQL database Online 	us-nc-prod-categori... SQL database Online 	us-nc-prod-locations SQL database Online 

BENEFITS - VS - CONSTRAINTS



BENEFITS

- Retiring the monolithic systems
 - We can now have small isolated components
 - Physical separation
 - You won't break my stuff – I won't break yours
 - Piece components together that you need

CONSTRAINTS

- Bringing everything together
 - How do you perform complex queries
 - How do you report out large data graphs
 - Separation is problematic – when everything is related

WHAT NOT TO DO



WHAT WE DID WRONG

- Our first Microservice architecture
 - Atomics only – No coupling
 - Skip the API Gateway – over kill
 - Force consumers to merry up data

WHAT WE DID WRONG

- Let's try it again – Round two
 - Too many microservices.. Too granular
 - Mass Data import was (almost) impossible
 - No messages or data movement
 - Failed to explain the benefits of the architecture

NOW WHERE TO?

- The world is changing again!
- Serverless Technology
 - AWS Lambda Functions
 - Azure Functions
 - Serverless Templates (SAM)
- Data Analytics
 - IoT – data collection component
 - Data Lake – not your average data warehouse

OTHER “OUTLIERS”

- Spin-Offs of Microservices
 - GraphQL
 - The gang at Facebook
 - Graph Databases and presentation
 - How do we use this in a Microservice?

THANK YOU

- Please reach out – I'm happy to help

WHO'S AWESOME?



YOU'RE AWESOME.