Taming Service Dependencies with Aspire

Alex Crome

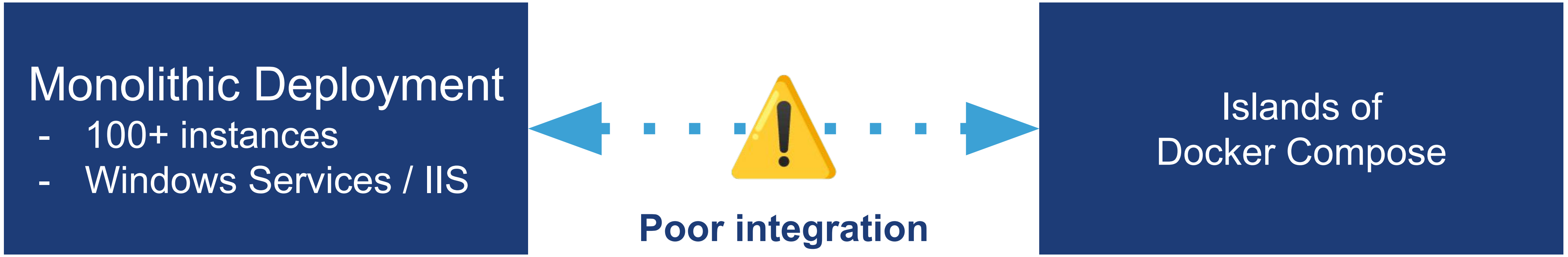
TMX TRAYPORT

Legal Disclaimer

Our Problem

TMX TRAYPORT

The Problem



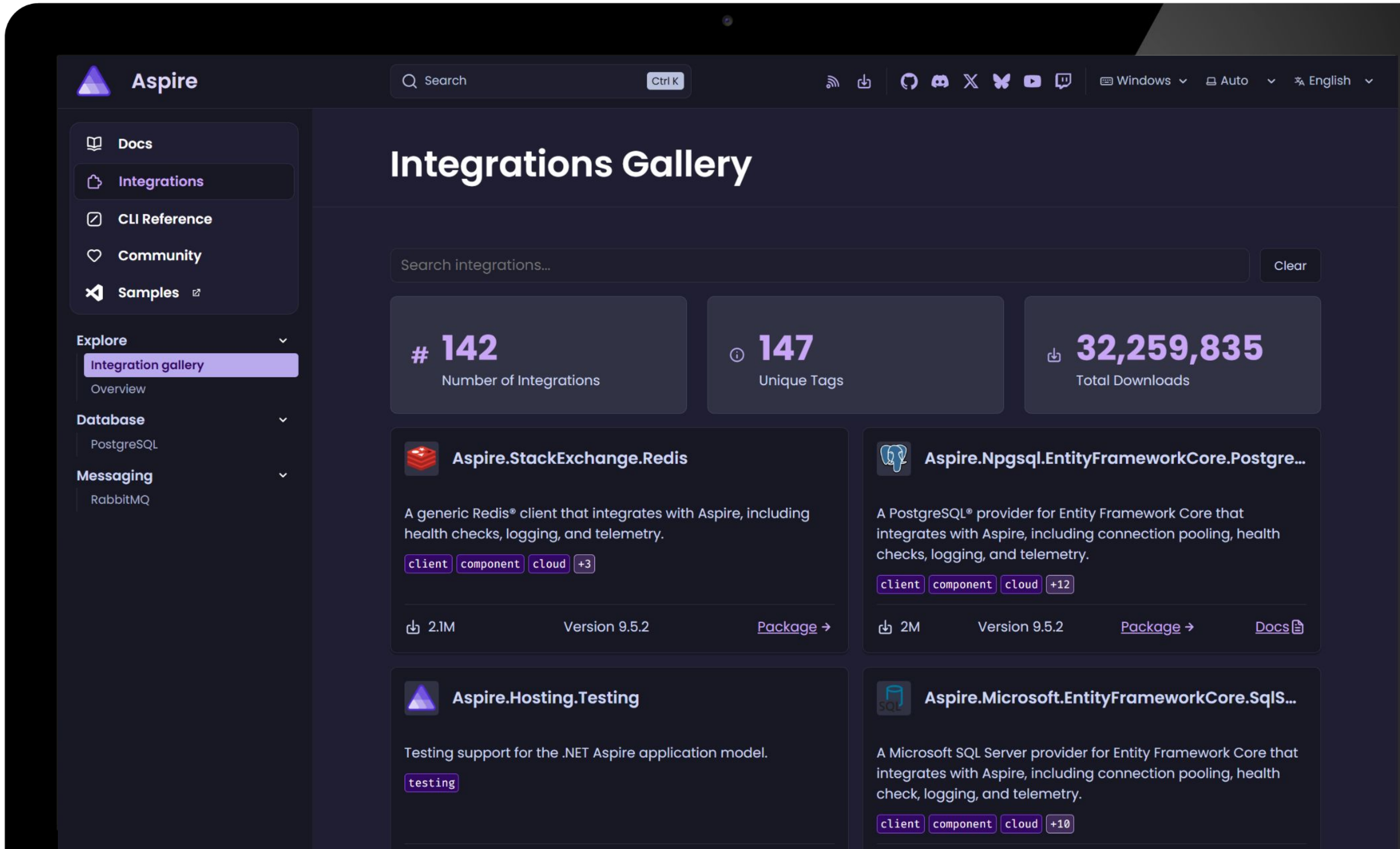
Poor integration between old and new worlds

- **Slow**
- **Brittle**
- **Lots of copy pasta**

- **Lots of copy pasta**
- **Worked in isolation**

Aspire Integrations Why not Build our Own?

```
builder
  .AddRedis("cache")
  .WithDataVolume();
```



Integrations Gallery

Search integrations...
Clear

142 Number of Integrations

147 Unique Tags

32,259,835 Total Downloads

Aspire.StackExchange.Redis
A generic Redis® client that integrates with Aspire, including health checks, logging, and telemetry.
client component cloud +3
2.1M Version 9.5.2 Package →

Aspire.Npgsql.EntityFrameworkCore.Postgre...
A PostgreSQL® provider for Entity Framework Core that integrates with Aspire, including connection pooling, health checks, logging, and telemetry.
client component cloud +12
2M Version 9.5.2 Package → Docs

Aspire.Hosting.Testing
Testing support for the .NET Aspire application model.
testing

Aspire.Microsoft.EntityFrameworkCore.SqlS...
A Microsoft SQL Server provider for Entity Framework Core that integrates with Aspire, including connection pooling, health check, logging, and telemetry.
client component cloud +10

Building an Integration

TMX TRAYPORT

Building an Integration The Resource Type

```
public class AccountsResource(string name)
 : ContainerResource(name)
{
}
```

TMX TRAYPORT

Initial Integration Add Extension Method

```
public static IResourceBuilder<AccountsResource> AddAccounts(
 this IDistributedApplicationBuilder builder,
 string name)
{
 var resource = new AccountsResource(name);

 return builder
  .AddResource(resource)
  .WithImage("web/accounts","latest")
  .WithImageRegistry("registry.io")
  .WithEnvironment("ASPNETCORE_ENVIRONMENT", "Development");
}
```

Initial Integration Builder Extension Method

```
public static IResourceBuilder<AccountsResource> WithMockUsers(
 this IResourceBuilder<AccountsResource> builder,
 string path)
{
 const string mountPath = "/mnt/MockUsers.json";

 return builder
  .WithBindMount(path, mountPath, isReadOnly: true)
  .WithEnvironment("FakeUsers__Path", mountPath);
}
```

Initial Integration Use it

```
var builder = DistributedApplication
  .CreateBuilder(args)

builder
  .AddAccounts("accounts")
  .WithMockUsers("./MockUsers.json");
```

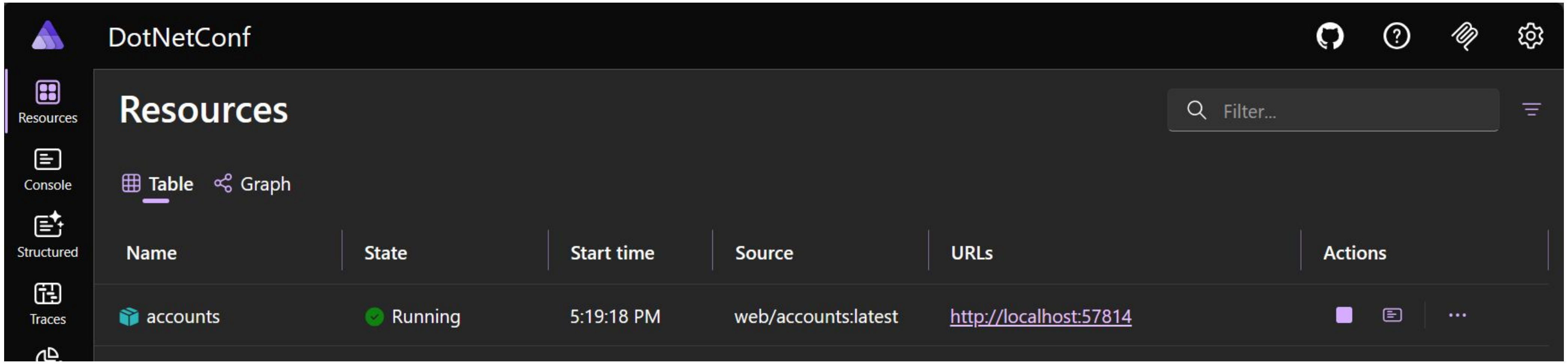
Expanding the Integration Open Telemetry

```
public static IResourceBuilder<AccountsResource> AddAccounts(...)
{
 return builder.AddResource(new AccountsResource(name);)
  //...
  .WithOtlpExporter();
}
```

TMX TRAYPORT

Enhancing the Integration Endpoints

```
public static IResourceBuilder<AccountsResource> AddAccounts(...)
{
 return builder.AddResource(new AccountsResource(name))
  //...
  .WithHttpEndpoint(targetPort: 8080,
          env: "ASPNETCORE_HTTP_PORTS");
}
```

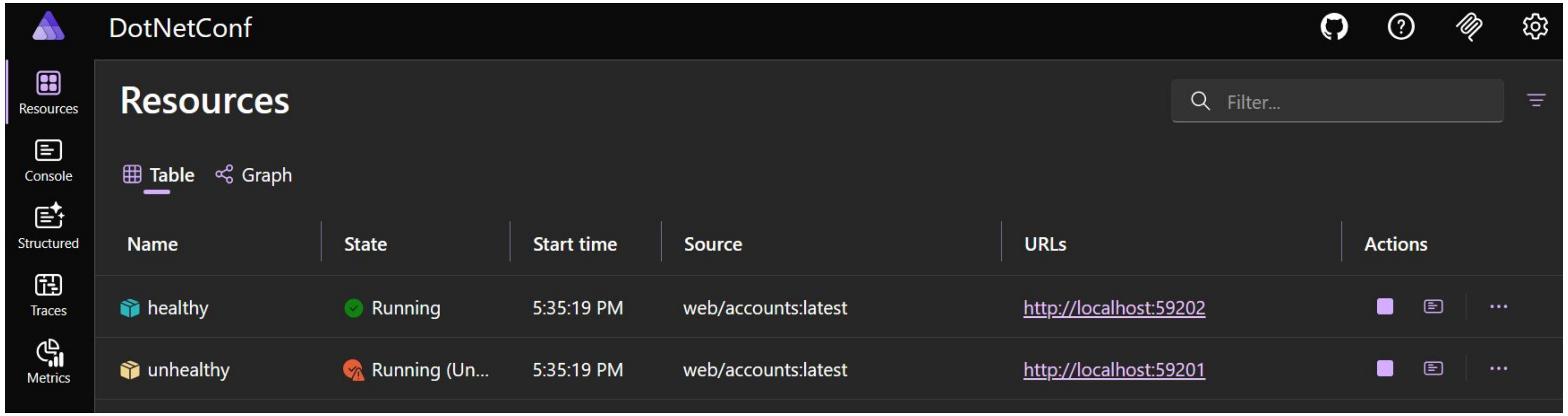


Resources

Name	State	Start time	Source	URLs	Actions
accounts	Running	5:19:18 PM	web/accounts:latest	http://localhost:57814	

Expanding the Integration Health Checks

```
public static IResourceBuilder<AccountsResource> AddAccounts(...)
{
 return builder.AddResource(new AccountsResource(name))
  //...
  .WithHttpHealthCheck(path: "/health");
}
```



Resources

Name	State	Start time	Source	URLs	Actions
healthy	Running	5:35:19 PM	web/accounts:latest	http://localhost:59202	
unhealthy	Running (Un...	5:35:19 PM	web/accounts:latest	http://localhost:59201	

Expanding the Integration Test it

```
[Test]
public async Task GoesHealthy(CancellationToken ct)
{
  var builder = DistributedApplicationTestingBuilder.Create();
  builder.Services.AddLogging(logging => {
   logging.SetMinimumLevel(LogLevel.Debug)
     .AddFilter(builder.Environment.ApplicationName, LogLevel.Debug)
     .AddFilter("Aspire.", LogLevel.Debug);
  });

  builder.AddAccounts("accounts");

  await using var app = builder.Build();
  await app.StartAsync(cancellationToken);
  await app.ResourceNotifications.WaitForResourceHealthyAsync("accounts", ct);
}
```

TMX TRAYPORT

Build Process Publish Integration Package

Build Process

```
dotnet pack Foo.slnx
dotnet nuget push artifacts/pack/**.nupkg
```

Trayport.Aspire.Hosting.Accounts.csproj

```
<Project>
 <PropertyGroup>
  <IsPackable>true</IsPackable>
 </PropertyGroup>
</Project>
```

Directory.Build.props

```
<Project>
 <PropertyGroup>
  <IsPackable>false</IsPackable>
  <UseArtifactsOutput>true</UseArtifactsOutput>
  <ArtifactsPath>$(MSBuildThisFileDirectory)artifacts</ArtifactsPath>
 </PropertyGroup>
</Project>
```

Build Process Publish Container

Build Process

```
dotnet publish Foo.slnx `
 /t:PublishContainer `
 /p:ContainerRegistry=registry.io `
 /p:ContainerImageTags="latest;$(Version)"
```

Directory.Build.props

```
<Project>
 <PropertyGroup>
  <IsPublishable>false</IsPublishable>
 </PropertyGroup>
</Project>
```

Trayport.Accounts.csproj

```
<Project>
 <PropertyGroup>
  <IsPublishable>true</IsPublishable>
  <ContainerRepository>web/accounts</ContainerRepository>
 </PropertyGroup>
</Project>
```

Introducing Dependencies

TMX TRAYPORT

Maging Dependencies Adding Dependencies

Required Dependency - `IDistributedApplicationBuilder.AddXYZ()`

```
public static IResourceBuilder<AccountsResource> AddAccounts(
  this IDistributedApplicationBuilder builder,
  string name,
  IResourceBuilder<PostgresDatabaseResource> db)
```

Optional Dependency - `resource.WithXYZ()`

```
public static IResourceBuilder<AccountsResource> WithDatabase(
  this IResourceBuilder<AccountsResource> builder,
  IResourceBuilder<PostgresDatabaseResource> db)
```

Dependencies Endpoints

```
var endpoint = accounts.GetEndpoint("http");
var host = endpoint.Property(EndpointProperty.Host);
var port = endpoint.Property(EndpointProperty.Port);

return resource
  .WithEnvironment("OIDC__Authority", endpoint)
  .WithArgs("--authority", endpoint)
  .WithEnvironment("OIDC__Host", host)
  .WithEnvironment("OIDC__Authority", $"http://{host}:{port}");


~~.WithEnvironment("OIDC__Port", endpoint.Port)~~
```

Dependencies Endpoints Gotcha

```
resource.WithEnvironment("OIDC__Url", $"http://{host}:{port}");
```

```
var url = $"http://{host}:{port}"
resource.WithEnvironment("OIDC__Url", url);
```

```
// Or ReferenceExpressionBuilder (like StringBuilder)
var url = ReferenceExpression.Create($"http://{host}:{port}")
resource.WithEnvironment("OIDC__Url", url);
```

Expanding the Integration Connection Strings

```
var server = db.Resource.Parent;

resource
  .WithEnvironment("DATABASE_HOST", server.Host")
  .WithEnvironment("DATABASE_PORT", server.Port)
  .WithEnvironment("DATABASE_USERNAME", server.UserNameReference)
  .WithEnvironment("DATABASE_PASSWORD", server.PasswordParameter)
  .WithEnvironment("DATABASE_NAME", db.Resource.DatabaseName);

resource
  .WithEnvironment("CS", db.Resource.ConnectionStringExpression)
  .WithEnvironment("JDBC", db.Resource.JdbcConnectionString);
```

Dependencies Waiting

```
resource.WaitFor(db);

resource.WaitForCompletion(migrations);
```

Be careful of long chains

Persistent Containers can mitigate

```
resource.WithLifetime(ContainerLifetime.Persistent)
```

Expanding the Integration Config Files

```
resource.OnBeforeResourceStarted(async (resource, evt, ct) =>
{
  var endpoint = accounts.GetEndpoint("http");
  await endpoint.GetValueAsync(ct);

  var connectionString = await db.GetConnectionStringAsync(ct);

  File.WriteAllText("./config.txt",
   $"""
   endpoint={endpoint.Url}
   connectionString={connectionString}
   """);
});
```

Dependencies Sub Resources

```
public static IResourceBuilder<AccountsResource> WithDatabaseMigrations(
   this IResourceBuilder<AccountsResource> accounts,
   IResourceBuilder<PostgresDatabaseResource> db,
   Action<IResourceBuilder<ExecutableResource>> configure = null)
{
   var migrations = db.ApplicationBuilder
     .AddExecutable($"{db.Resource.Name}-Migrations",
       "dotnet", "./Trayport.Accounts")
     .WithArgs("ef", "database", "update", "--connection", db.Resource)
     .WaitFor(db)
     .WithParentRelationship(db);
   configure?.Invoke(migrations);
   return accounts.WaitForCompletion(migrations);
}
```

Helpers

Helper Conventions

```
public static IDistributedApplicationBuilder WithTrayportDefaults(
 this IDistributedApplicationBuilder builder)
{
 return builder
   .WithRegistryMirrors()
   .WithHostCertificates();
}
```

```
var builder = DistributedApplication
   .CreateBuilder(args)
   .WithTrayportDefaults();
```

TMX TRAYPORT

Helpers Registry Mirrors

```
public static IDistributedApplicationBuilder WithRegistryMirror(
 this IDistributedApplicationBuilder builder)
{
 builder.Eventing.Subscribe<BeforeStartEvent>((evt, _) => {
  foreach (var resource in evt.Model.Resources)
  foreach (var annotation in resource.Annotations.OfType<ContainerImageAnnotation>())
  {
   if (annotation.Registry == "docker.io") {
    annotation.Registry = "mirror.registry.io";
    annotation.Image = $"cache/docker.io/{annotation.Image}";
   }
  }
  return Task.CompletedTask;
 });
 return builder;
}
```

Helpers Certificates

```
public static IDistributedApplicationBuilder WithHostCertificates(
  this IDistributedApplicationBuilder builder)
{
 builder.Eventing.Subscribe<BeforeStartEvent>((evt, ct) => {
  foreach (var resource in evt.Model.GetContainerResources())
  {
   builder.CreateResourceBuilder(resource)
     .WithCertificateTrustScope(CertificateTrustScope.System);
  }
  return Task.CompletedTask;
 });

 return builder;
}
```

Helpers Container Image

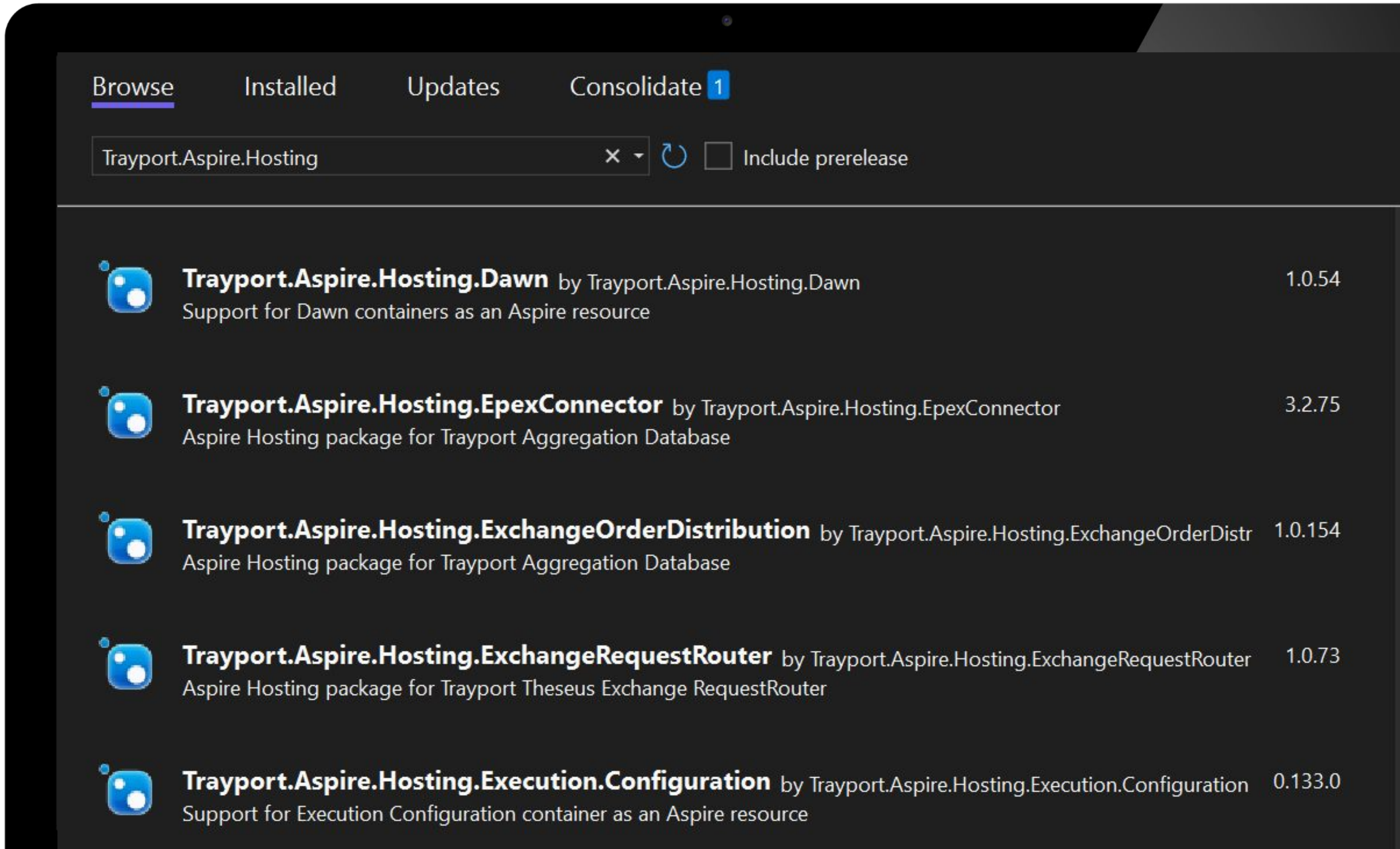
```
public static IResourceBuilder<T> WithTrayportImage<T>(
  this IResourceBuilder<T> resourceBuilder,
  string image,
  string tag = "latest")
 where T : ContainerResource
{
 var registry = imageTag.Contains("-") ? "NonProd.io" : "Prod.io";

 return resourceBuilder.WithImage(image)
        .WithImageTag(tag)
        .WithImageRegistry(registry);
}
```

Did it work?

Today Internal Integrations

33

Internal Integrations



Trayport.Aspire.Hosting

Trayport.Aspire.Hosting.Dawn by Trayport.Aspire.Hosting.Dawn 1.0.54
Support for Dawn containers as an Aspire resource

Trayport.Aspire.Hosting.EpexConnector by Trayport.Aspire.Hosting.EpexConnector 3.2.75
Aspire Hosting package for Trayport Aggregation Database

Trayport.Aspire.Hosting.ExchangeOrderDistribution by Trayport.Aspire.Hosting.ExchangeOrderDistr 1.0.154
Aspire Hosting package for Trayport Aggregation Database

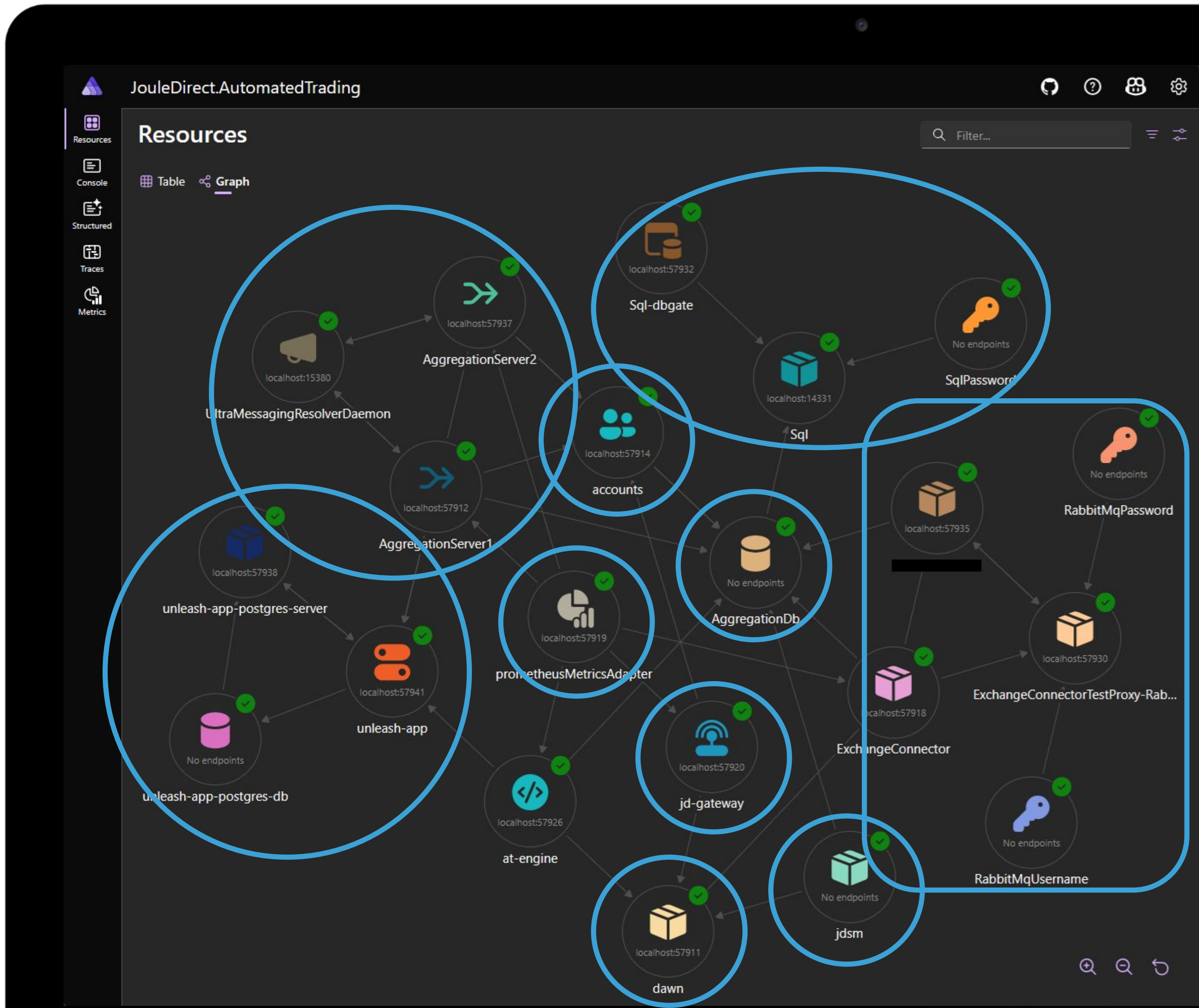
Trayport.Aspire.Hosting.ExchangeRequestRouter by Trayport.Aspire.Hosting.ExchangeRequestRouter 1.0.73
Aspire Hosting package for Trayport Theseus Exchange RequestRouter

Trayport.Aspire.Hosting.Execution.Configuration by Trayport.Aspire.Hosting.Execution.Configuration 0.133.0
Support for Execution Configuration container as an Aspire resource

TMX TRAYPORT

Today Internal Integrations

```
builder.AddDawn();
builder.AddUnleash();

var aggDb = builder
   .AddSqlServer()
   .AddAggregationDatabase();

var accounts = builder
   .AddAccounts()
   .WithAggregationDb(aggDb);

builder
   .AddAggregationServer(aggDb)
   .WithAccounts(accounts);

// ...
```



TMX TRAYPORT

Challenges Keeping Containers Up To Date

```
.WithImageTag("latest")
```

✅ Simple
❌ Different developers will end up with different dependency versions
❌ Dependencies can get out of date (potentially years!)
⚠️ Can be updated by running `docker pull`.

```
.WithImageTag("latest")
.WithImagePullPolicy(ImagePullPolicy.Always)
```

✅ Always up to date
❌ Fail to start if network unavailable

```
var version = Assembly.GetExecutingAssembly().GetName().Version;
.WithImageTag(version.ToString())
```

✅ Stable dependency version
❌ Images updates require nuget package updates

https://github.com/dotnet/aspire/issues/10719

Challenges Non Containerisable Executables

- **Containers handle both acquisition and execution**
- **To run an executable, how do you acquire it?**
  - **Initially: Cheat and assume path from old deployment start**
  - **Exploring packing as dotnet tool**
    - **`dnx / dotnet tool exec`**
    - **Can we smuggle non dotnet tools through native AOT tool packages?**

TMX TRAYPORT https://andrewlock.net/exploring-dotnet-10-preview-features-7-packaging-self-contained-and-native-aot-dotnet-tools-for-nuget/

Challenges Project Substitution

- Containers simple for others to use
- Projects better for debugging

No common base type between Projects and Containers!

1. Remove existing Container Resource
2. Add Project resource with same name
3. Copy annotations from container to project resource
4. Fix up differences for container vs project
   - Endpoints
   - Bind Mounts
5. Forward event handlers

```
builder
   .AddAccounts()
   .AsProject<AccountsResource, Projects.Trayport_Accounts>()
```

TMX TRAYPORT

The Future

Future

More Integrations

- More Hosting Integrations
- Explore Client Integrations

Reduce Dependencies

Ephemeral Environments

- PRs
- A la carte
- Exploratory Testing

Testing

- Exploratory Testing
- Chaos Testing

Thank You

TMX TRAYPORT

Feedback

info@trayport.com www.trayport.com

United Kingdom (Head Office)	Trayport Limited, 3rd Floor, 2 Gresham Street, London, EC2V 7AD, United Kingdom	+44 20 7960 5500
Austria	Trayport Austria GmbH, Euro Plaza 2E/1 OG, Technologiestraβe 10, 1120 Wien, Austria	+43 1 609 2290
Germany	Trayport Germany GmbH, Linzer Straße 11, 28359 Bremen, Germany	+49 421 20109-0
Singapore (Asia Pacific)	Trayport Pte Ltd, One Raffles Place, Office Tower 1, #31-02, Singapore, 048616	+65 6411 4700