4dotnet

DAPR and .NET Aspire

A royal wedding

Florian van Dillen
Full-stack developer
Futuretech 2024
17/04/2024

Florian van Dillen

Full-stack developer

Building event-driven systems with .NET, Azure, Containers, DAPR and more! Interested in architectural concepts such as DDD, design patterns and scaling.

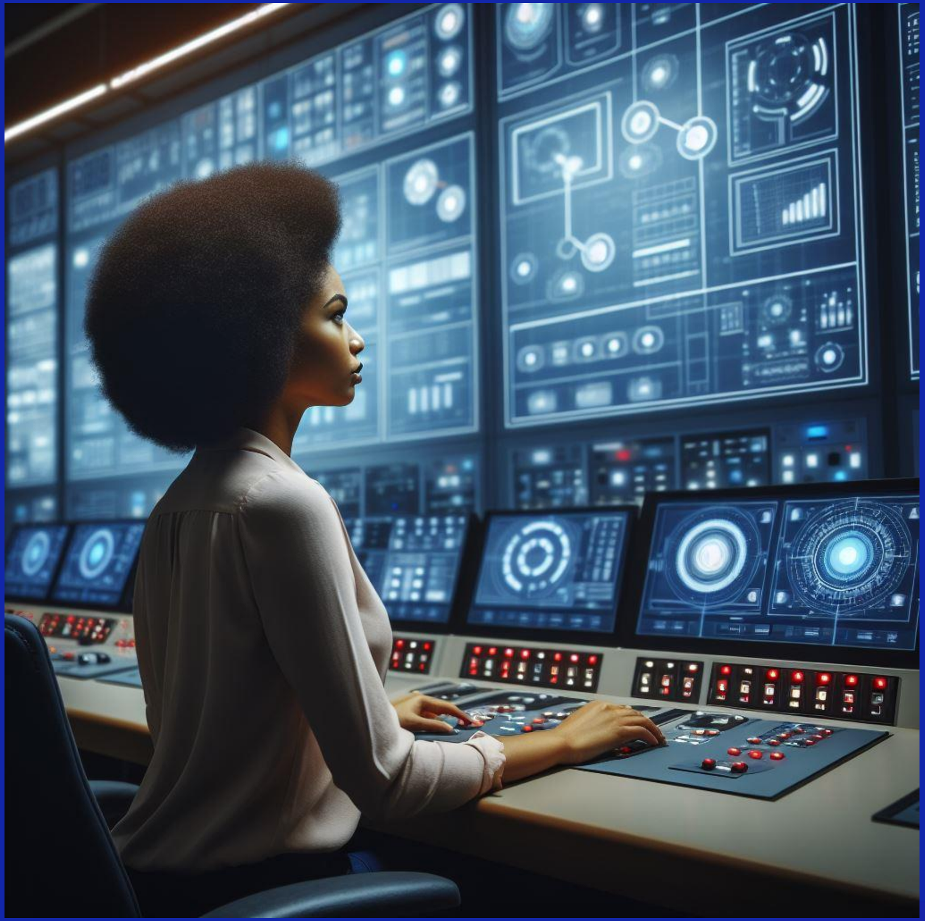
www.linkedin.com/in/florianvandillen
https://twitter.com/fvandillen
https://github.com/fvandillen

Building distributed apps can be hard

Building distributed apps can be hard

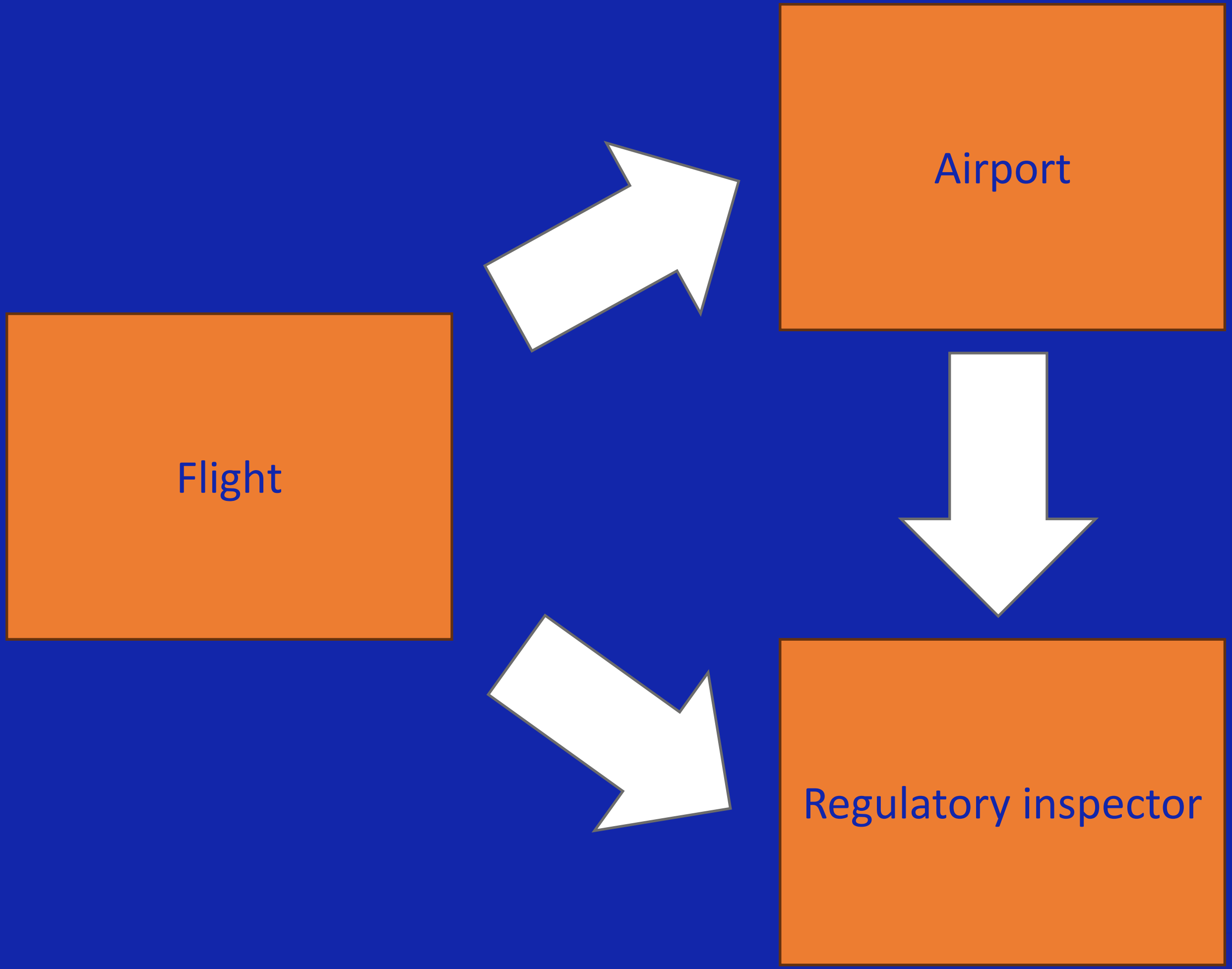
Building distributed apps can be hard

Building distributed apps can be hard



The local dev environment is important too!

Our demo app



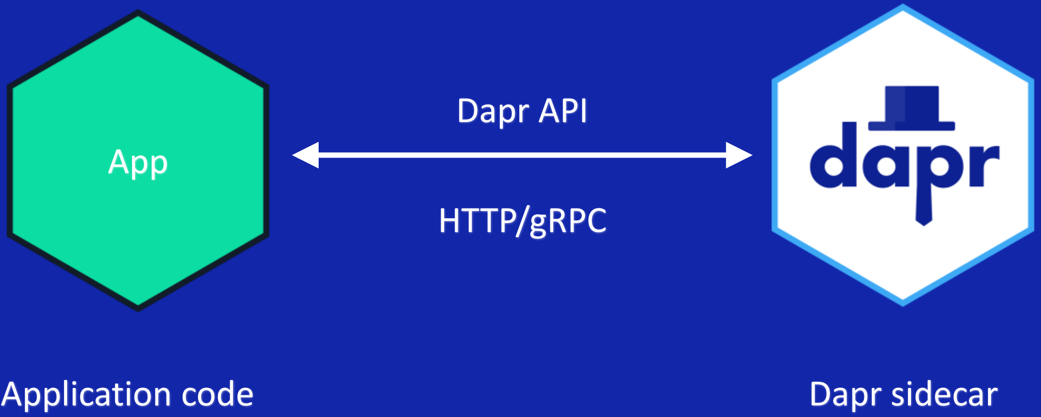
Introducing Dapr

- APIs for building **secure** and **reliable** microservices
- Serves as the **glue** between your services

Dapr goals

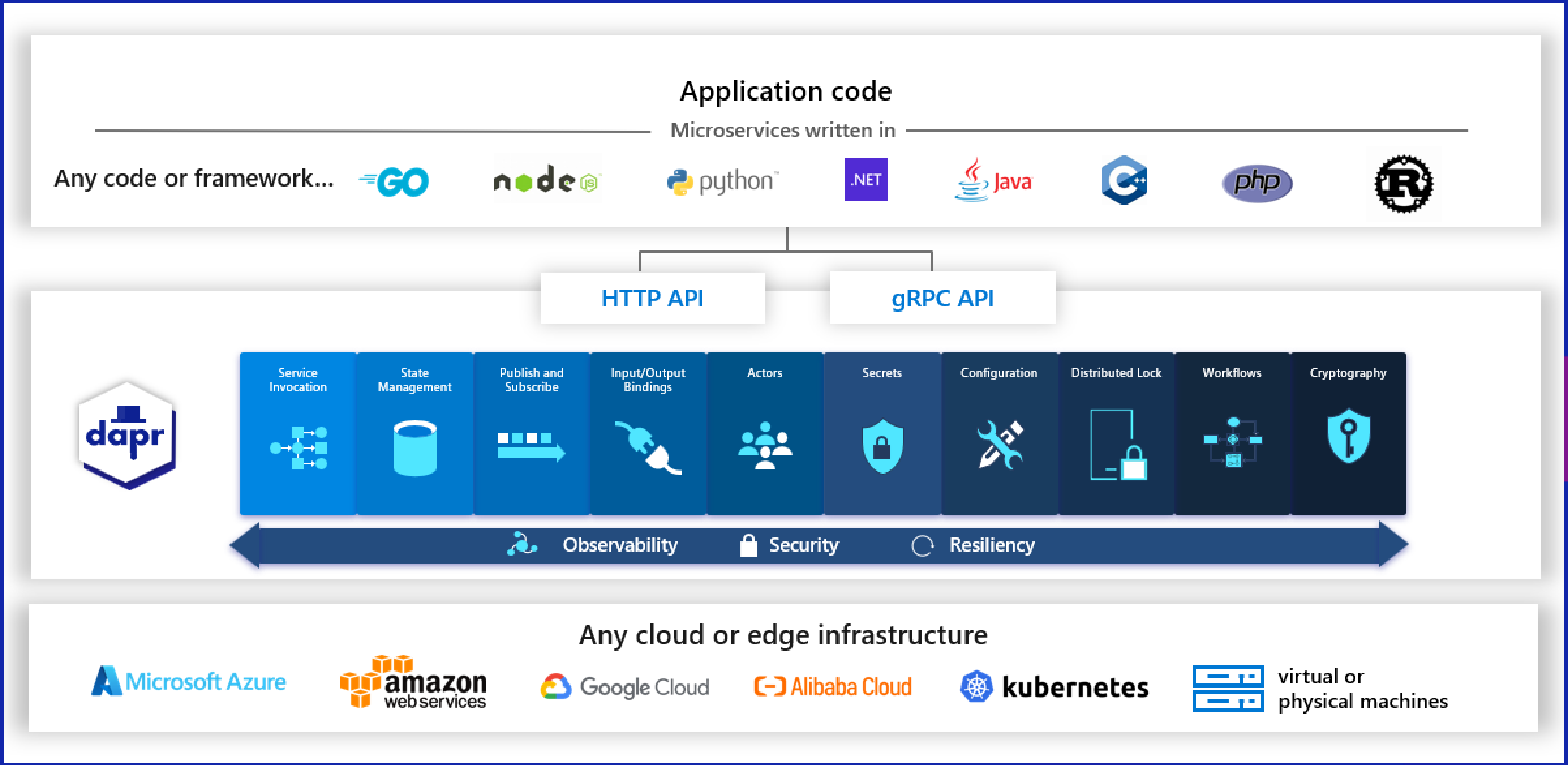
- Provide an integrated set of APIs
- Includes best practices & standards
- Extensible and pluggable
- Any language or framework
- Platform agnostic
- Community driven, vendor neutral

Sidecar architecture

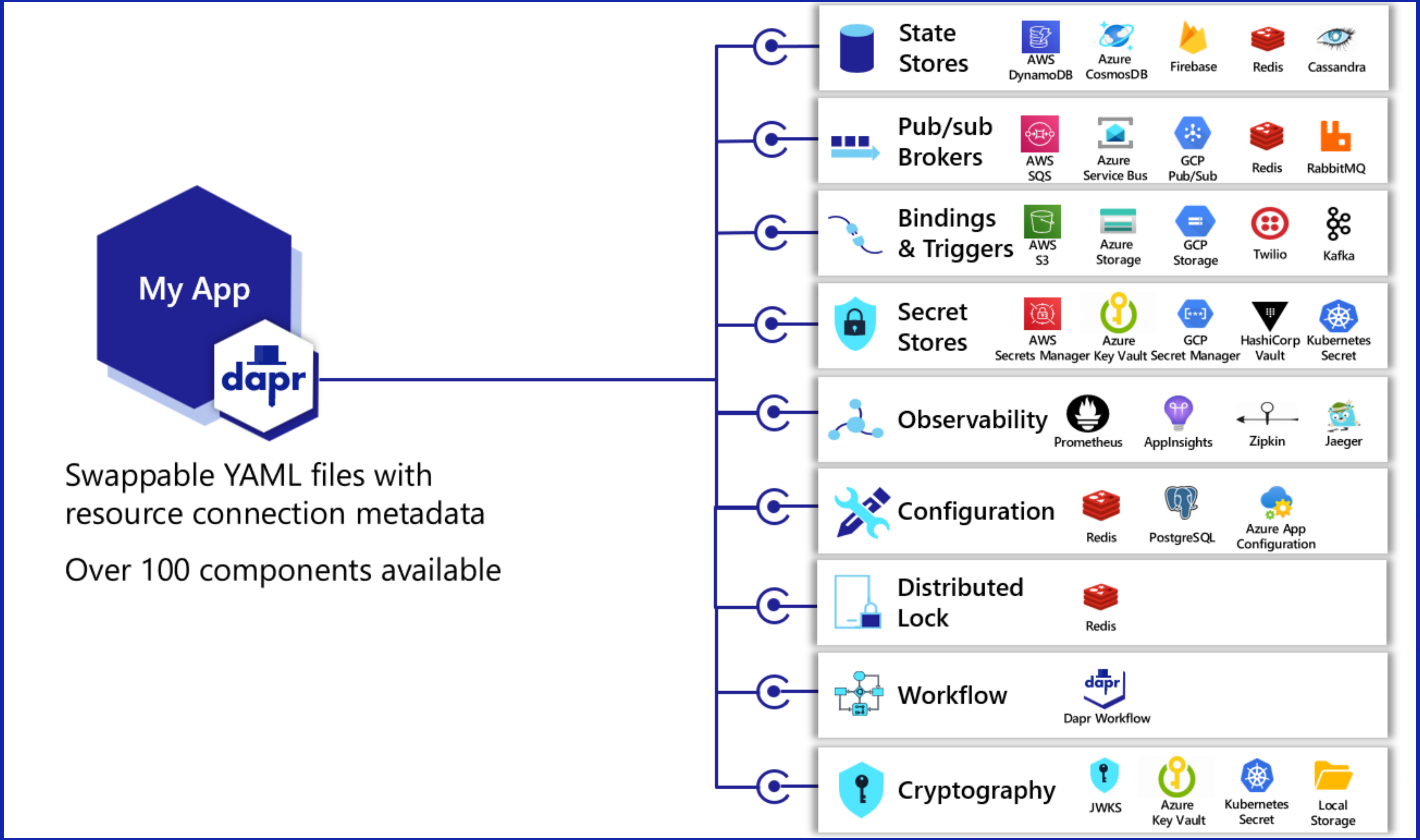


POST	http://localhost:3500/v1.0/**invoke**/cart/method/order
GET	http://localhost:3500/v1.0/**state**/inventory/item50
POST	http://localhost:3500/v1.0/**publish**/mybroker/order-messages
GET	http://localhost:3500/v1.0/**secrets**/vault/dbaccess
POST	http://localhost:3500/v1.0-beta1/**workflows**/dapr/businessprocess/start

The bigger picture



Swappable components

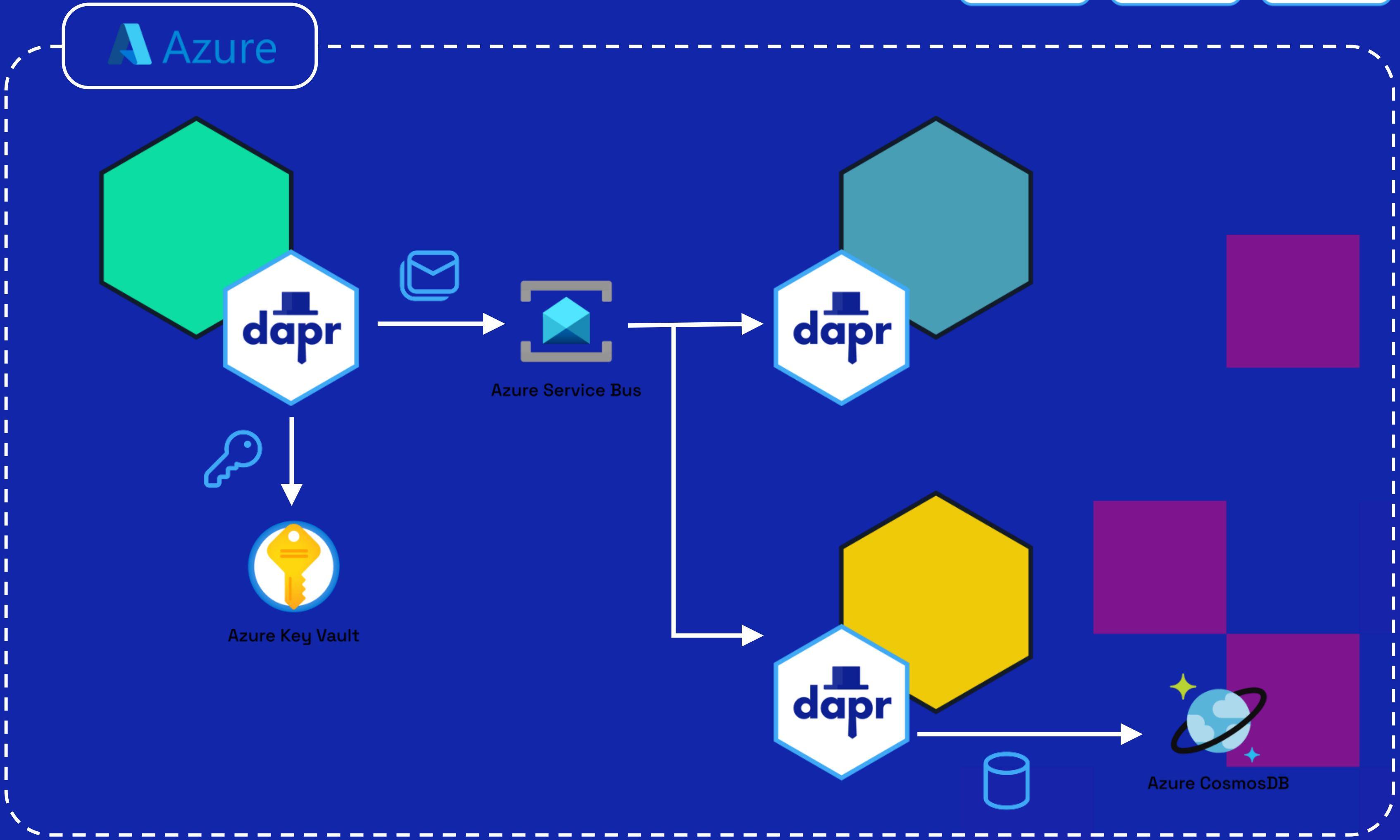


Swappable YAML files with resource connection metadata

Over 100 components available

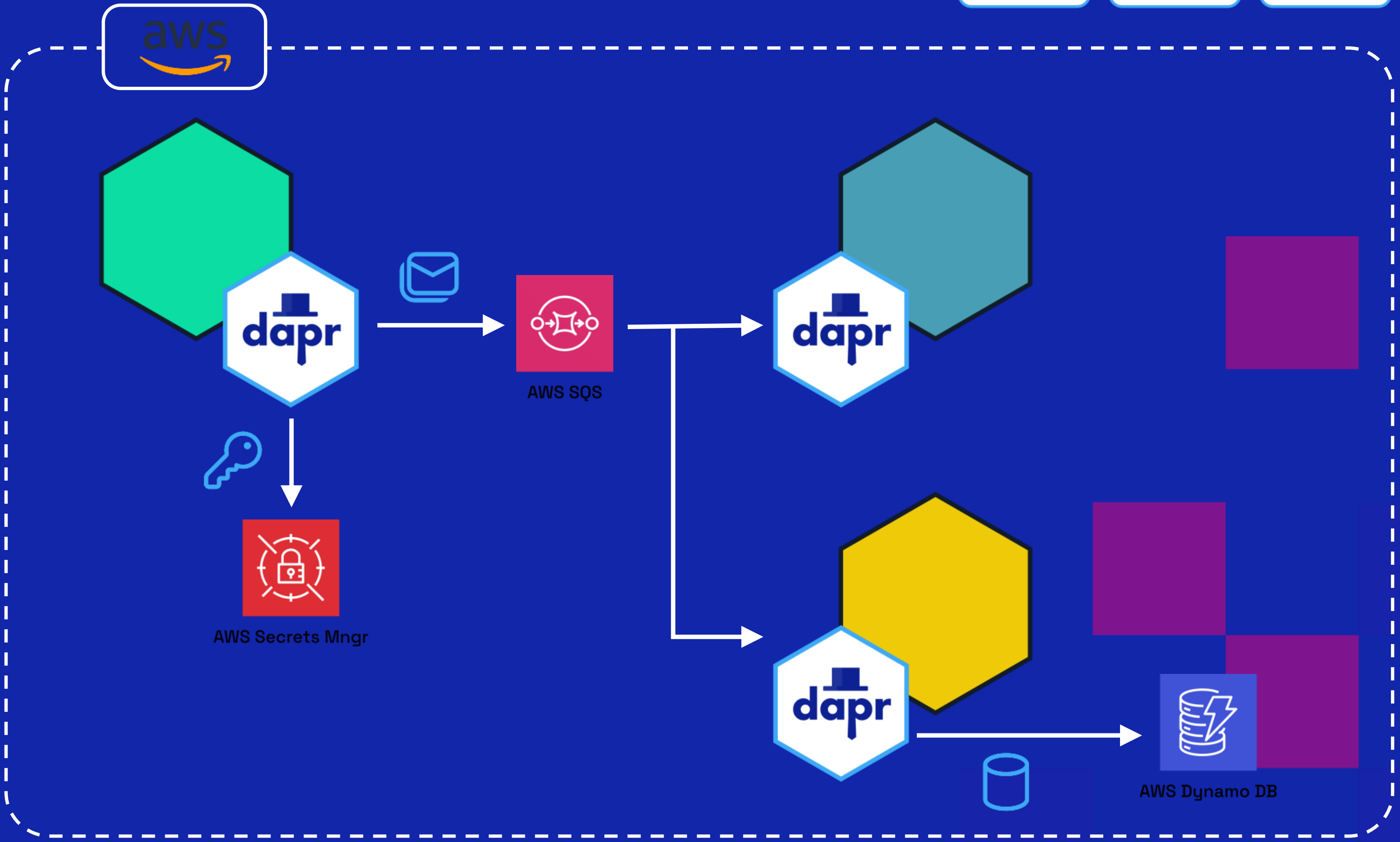
Swappable components (2)

Secrets
Publish / Subscribe
State Management

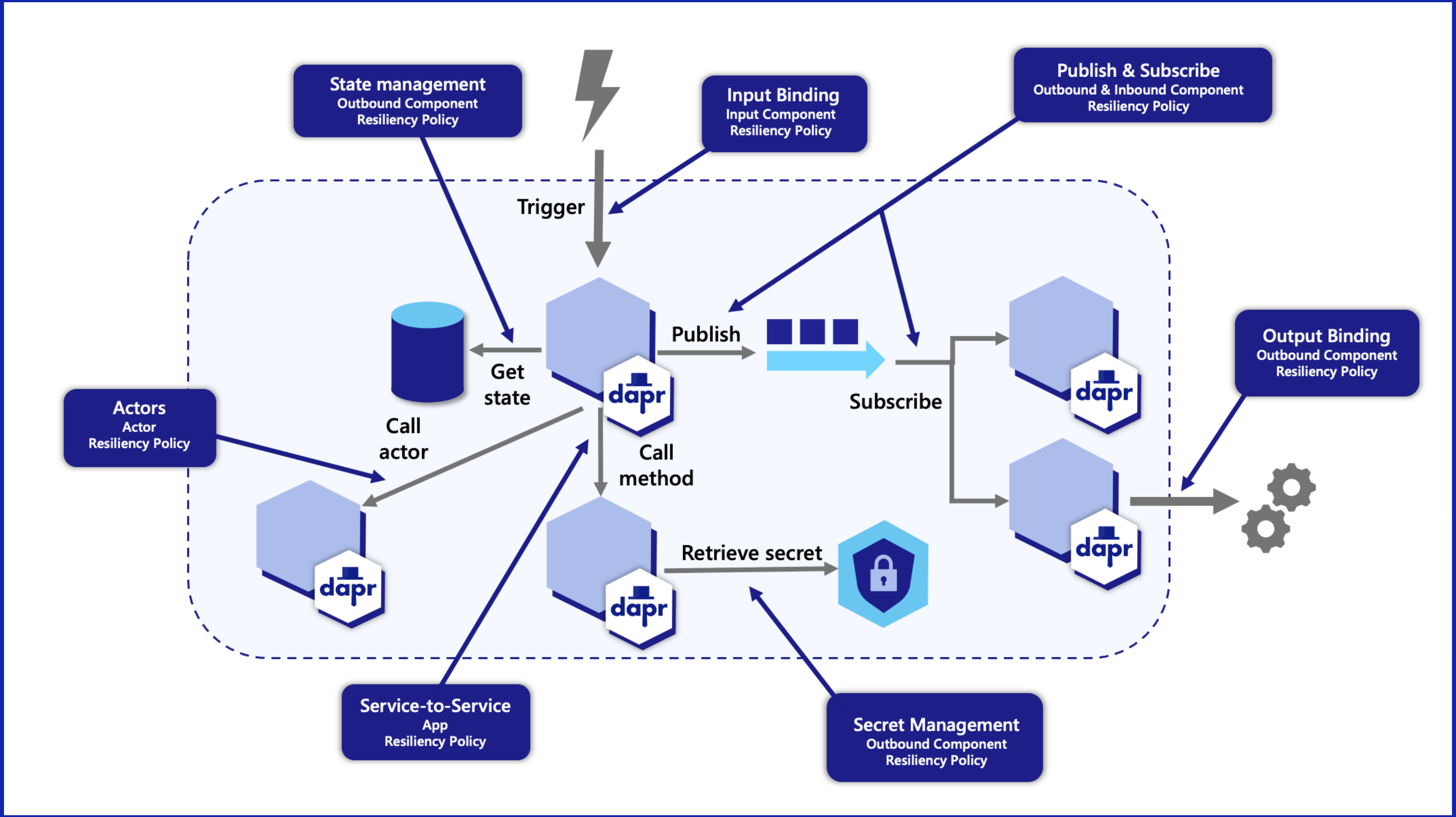


Swappable components (3)

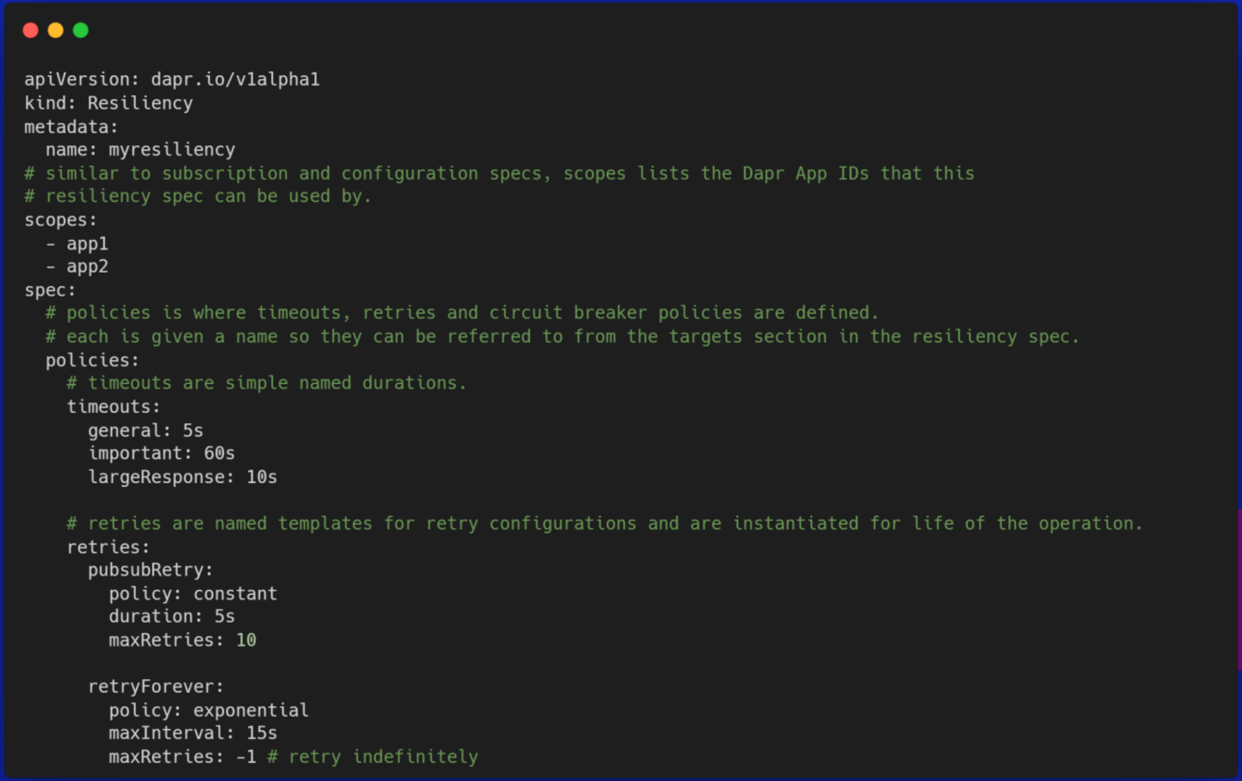
Secrets
Publish / Subscribe
State Management



Resiliency

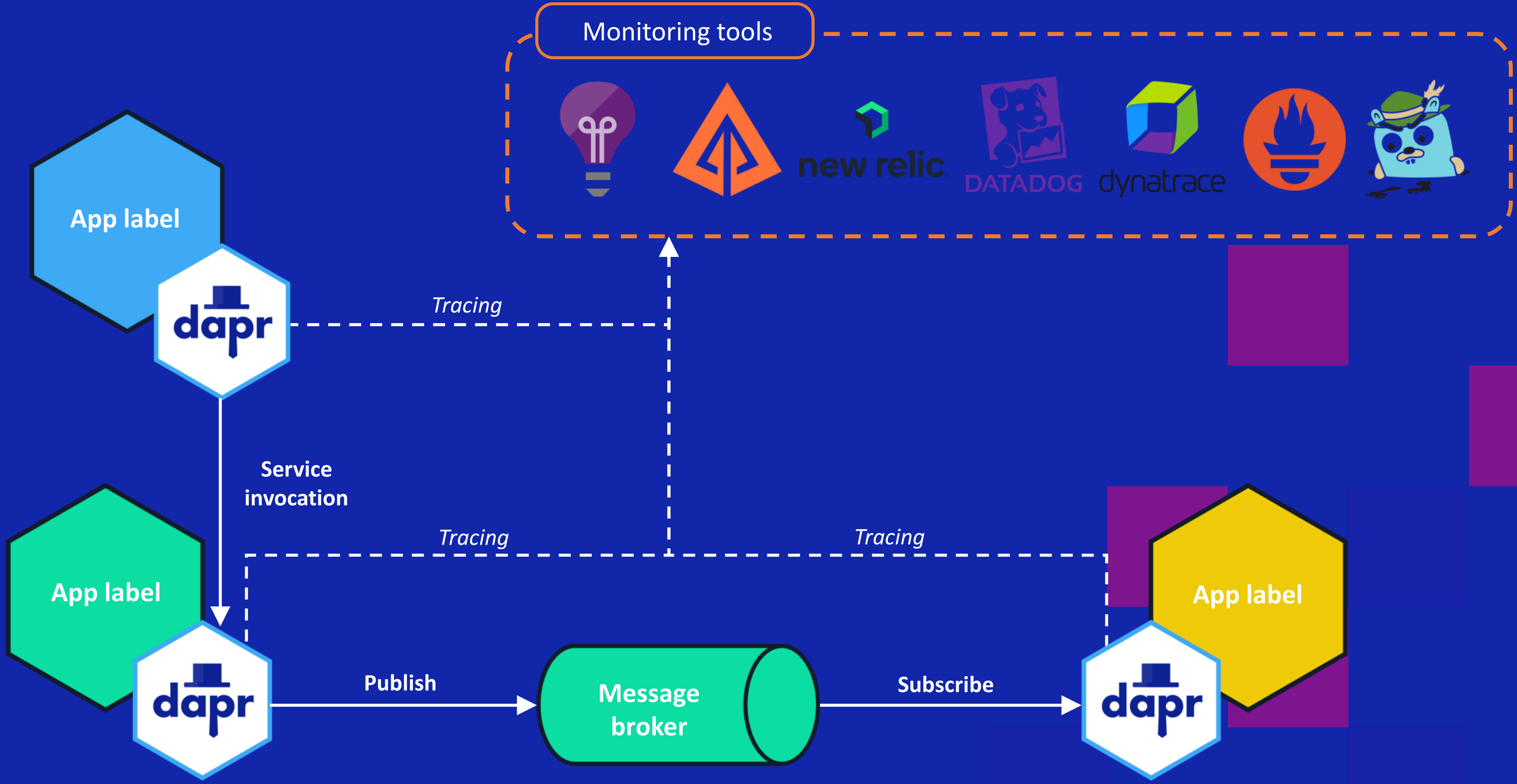

State management
Outbound Component
Resiliency Policy
Input Binding
Input Component
Resiliency Policy
Publish & Subscribe
Outbound & Inbound Component
Resiliency Policy
Trigger
Get state
Publish
Subscribe
Actors
Actor
Resiliency Policy
Call actor
Call method
Output Binding
Outbound Component
Resiliency Policy
Retrieve secret
dapr
Service-to-Service
App
Resiliency Policy
Secret Management
Outbound Component
Resiliency Policy

Resiliency (2)



```
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: myresiliency
# similar to subscription and configuration specs, scopes lists the Dapr App IDs that this
# resiliency spec can be used by.
scopes:
  - app1
  - app2
spec:
  # policies is where timeouts, retries and circuit breaker policies are defined.
  # each is given a name so they can be referred to from the targets section in the resiliency spec.
  policies:
    # timeouts are simple named durations.
    timeouts:
      general: 5s
      important: 60s
      largeResponse: 10s

    # retries are named templates for retry configurations and are instantiated for life of the operation.
    retries:
      pubsubRetry:
        policy: constant
        duration: 5s
        maxRetries: 10

      retryForever:
        policy: exponential
        maxInterval: 15s
        maxRetries: -1 # retry indefinitely
```

Observability



Hosting modes

Self-hosted

Run **`dapr init`** to install Docker images.

Run any app with a Dapr side car using dapr run.

Virtual/Physical machines

Self-deploy Dapr control plane and Hashicorp Consul per machine.

Use the Dapr Installer Bundle for airgapped environments.

Run any app with a Dapr side car using **`dapr run`**.

Kubernetes

Run **`dapr init -k`** to install Dapr (or use Helm). Integrated Dapr control plane. Deploys placement, operator, sentry and injector pods.

Automatically injects a Dapr sidecar into all annotated pods.

Hosting modes (2)

Serverless

The Dapr side car is hosted by a provider.

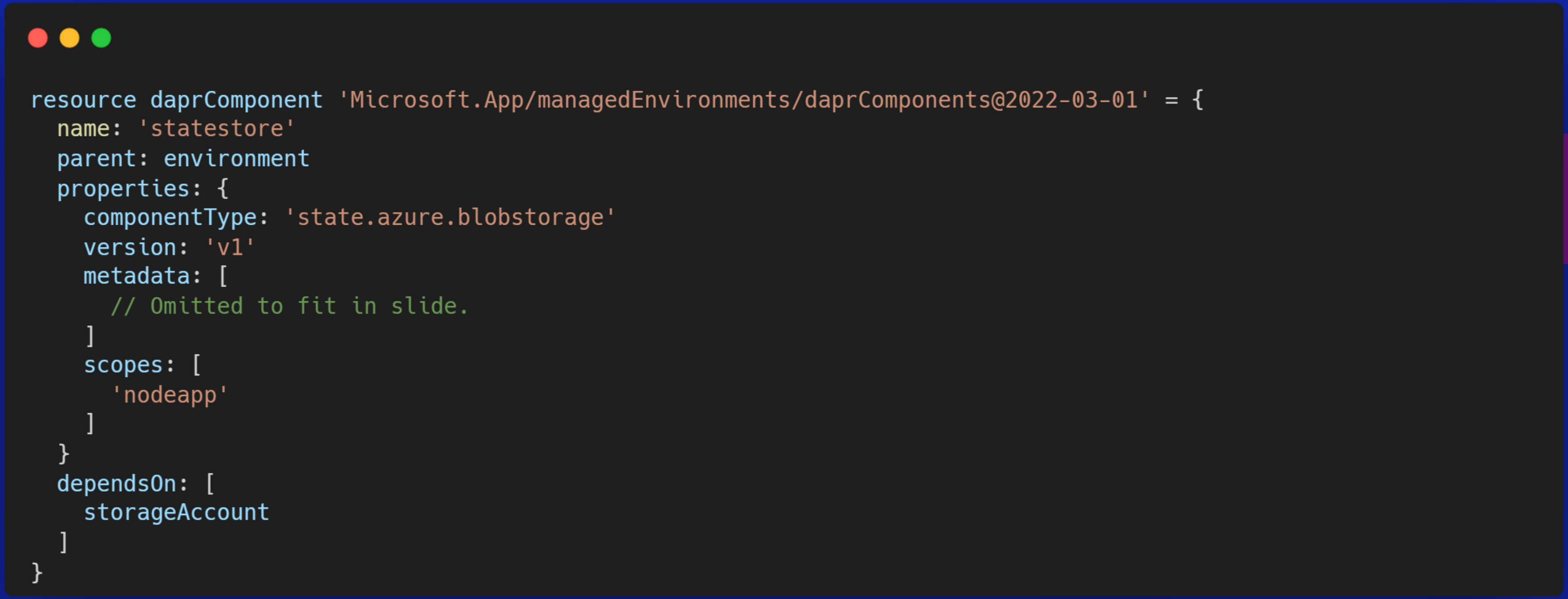
You only manage your applications.

Dapr installation (dev machine)

- Prerequisite: Docker
- Installing the Dapr CLI
- Running *dapr init*

Dapr in Azure container apps

- No management required of control plane!
- Just deploy Dapr components via Azure Portal or Bicep



```
resource daprComponent 'Microsoft.App/managedEnvironments/daprComponents@2022-03-01' = {
  name: 'statestore'
  parent: environment
  properties: {
    componentType: 'state.azure.blobstorage'
    version: 'v1'
    metadata: [
      // Omitted to fit in slide.
    ]
    scopes: [
      'nodeapp'
    ]
  }
  dependsOn: [
    storageAccount
  ]
}
```

Demo: Run a Dapr app

Demo: Dapr sidekick

Demo: Publish event from CLI

- You can send events from the command line!

Aspire

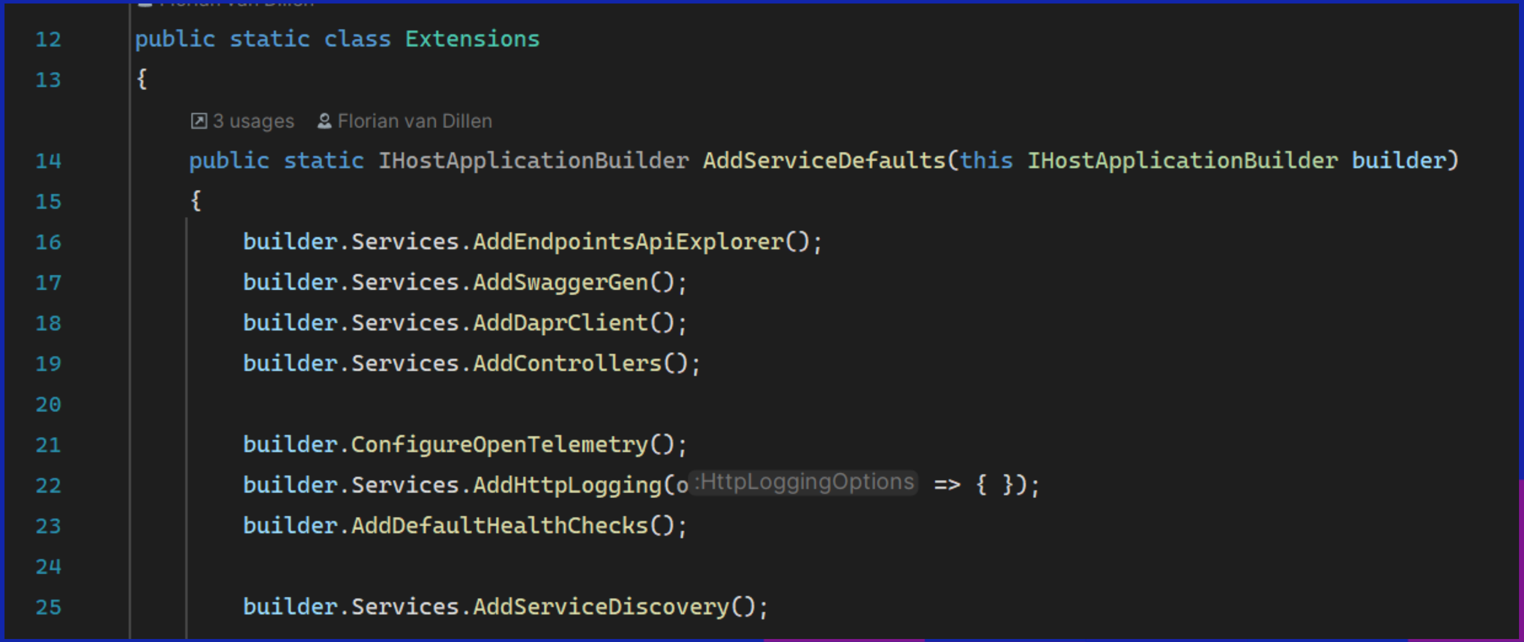
- Orchestrator for local development environment
- Run multiple services easily
- Spin up containers
- See telemetry

Aspire dashboard

- Developer control plane (DCP)
- Metrics, traces, structured logs
- Configuration
- Containers and services

Service defaults

- ***Opinionated*** startup
- ASPNET Core settings
- OpenTelemetry
- Health checks

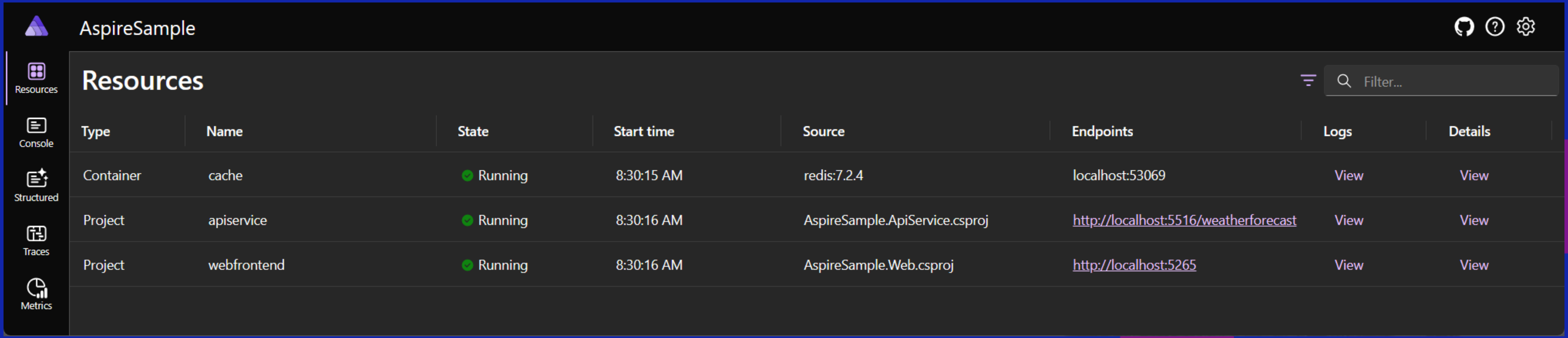


```
public static class Extensions
{
    3 usages   Florian van Dillen
    public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
    {
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen();
        builder.Services.AddDaprClient();
        builder.Services.AddControllers();

        builder.ConfigureOpenTelemetry();
        builder.Services.AddHttpLogging(o :HttpLoggingOptions  => { });
        builder.AddDefaultHealthChecks();

        builder.Services.AddServiceDiscovery();
```

AppHost

- Orchestrator
- We launch our Dapr sidecar from here!

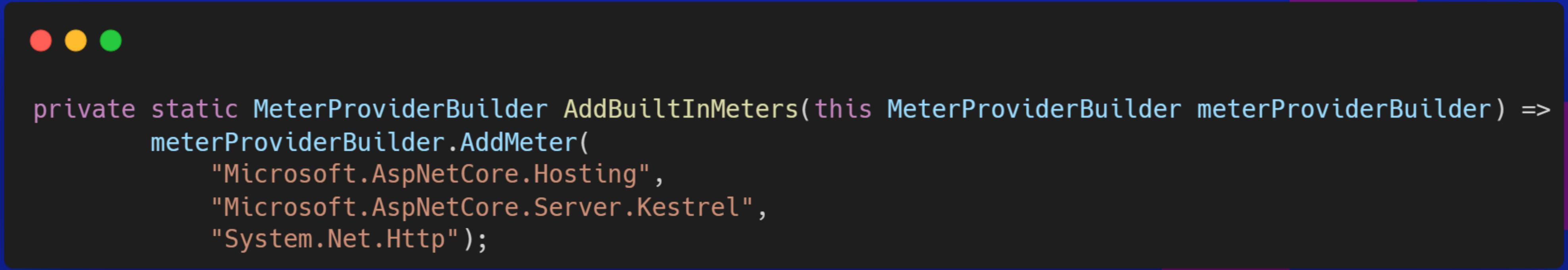
Demo: Aspire dashboard



Resources

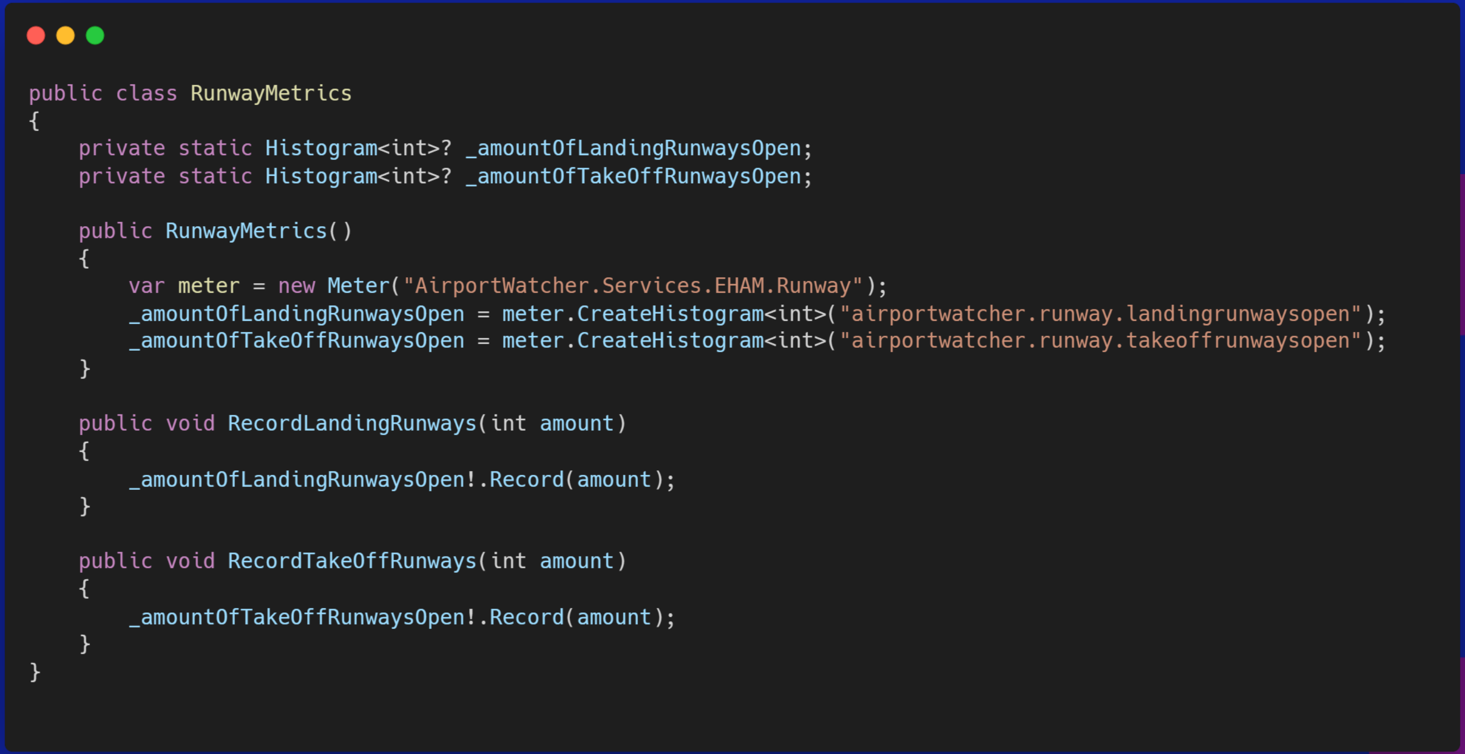
Type	Name	State	Start time	Source	Endpoints	Logs	Details
Container	cache	● Running	8:30:15 AM	redis:7.2.4	localhost:53069	View	View
Project	apiservice	● Running	8:30:16 AM	AspireSample.ApiService.csproj	http://localhost:5516/weatherforecast	View	View
Project	webfrontend	● Running	8:30:16 AM	AspireSample.Web.csproj	http://localhost:5265	View	View

Collecting metrics



```csharp
private static MeterProviderBuilder AddBuiltInMeters(this MeterProviderBuilder meterProviderBuilder) =>
        meterProviderBuilder.AddMeter(
            "Microsoft.AspNetCore.Hosting",
            "Microsoft.AspNetCore.Server.Kestrel",
            "System.Net.Http");
```

Demo: Creating custom metrics



```
public class RunwayMetrics
{
    private static Histogram<int>? _amountOfLandingRunwaysOpen;
    private static Histogram<int>? _amountOfTakeOffRunwaysOpen;

    public RunwayMetrics()
    {
        var meter = new Meter("AirportWatcher.Services.EHAM.Runway");
        _amountOfLandingRunwaysOpen = meter.CreateHistogram<int>("airportwatcher.runway.landingrunwaysopen");
        _amountOfTakeOffRunwaysOpen = meter.CreateHistogram<int>("airportwatcher.runway.takeoffrunwaysopen");
    }

    public void RecordLandingRunways(int amount)
    {
        _amountOfLandingRunwaysOpen!.Record(amount);
    }

    public void RecordTakeOffRunways(int amount)
    {
        _amountOfTakeOffRunwaysOpen!.Record(amount);
    }
}
```

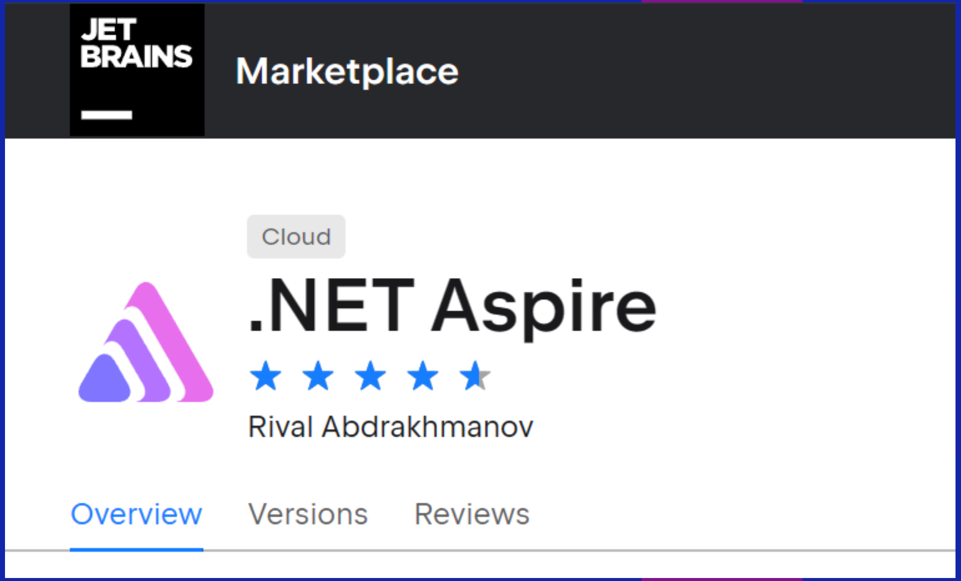
Demo: Adding SQL

Pitfalls to avoid

- Ensure your workload version is the same as the version in your project (*dotnet workload list*).
- Ensure Docker is running.

Pro tip

- Rider users can install an extension to easily run their Aspire projects with the debugger automatically attaching.
- Shows metrics.
- Shows trace diagram.
- Shows dashboard.



Interested in the code?

- https://github.com/fvandillen/futuretech-dapr-aspire