

# Developing ASP.NET Core Microservices with Dapr

Marc Müller  
Principal Consultant



marc.mueller@4tecture.ch  
@muellermarc  
www.4tecture.ch

4tecture®  
empower your software solutions



About me:

Marc Müller  
Principal Consultant  
@muellermarc



4tecture<sup>®</sup>  
empower your software solutions

Our Products:

Multi-Tenant OpenID  
Connect Identity Provider



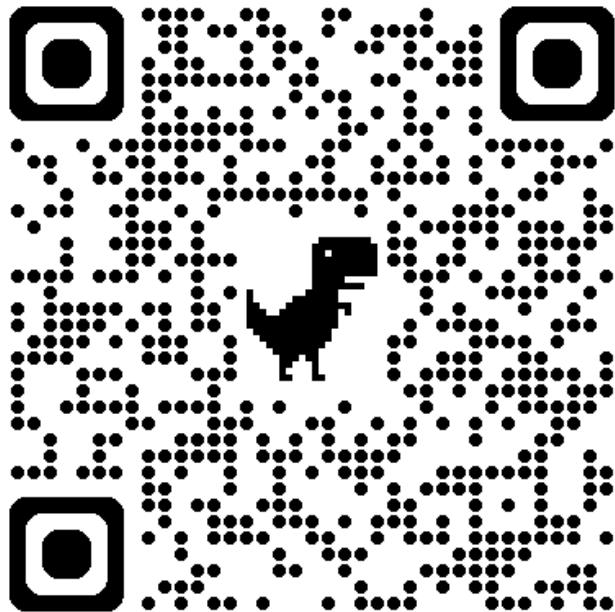
[www.proauth.net](http://www.proauth.net)

Enterprise Application  
Framework for .NET



[www.reafx.net](http://www.reafx.net)

# Slide Download



<https://www.4tecture.ch/events/dotnetday2025>

# Agenda

- Intro
- Dapr Basics
- Hands-On!
- Building Blocks
- Conclusion





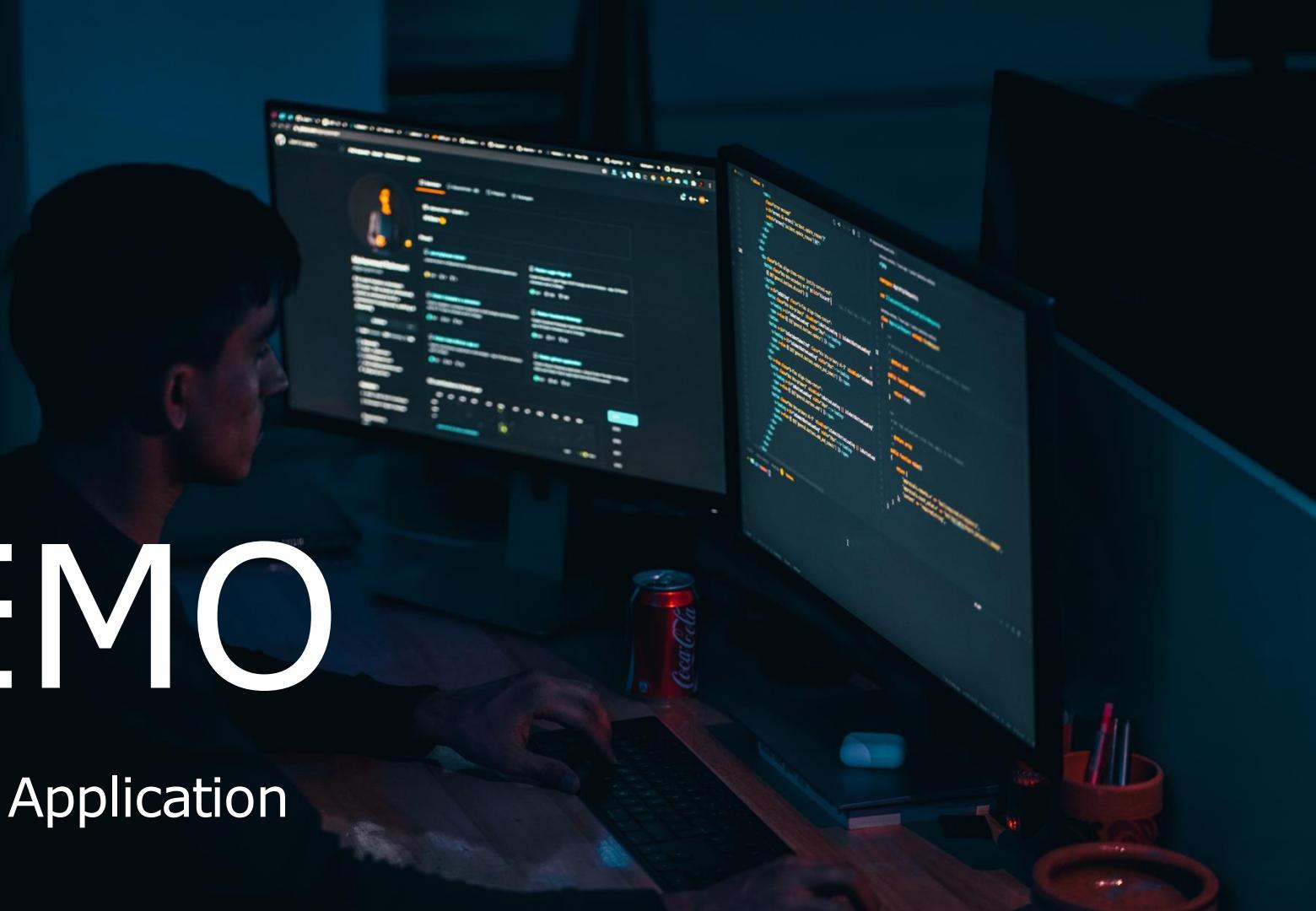
Dapr

# Intro

4tecture®  
empower your software solutions

# DEMO

Fast Food Application



The image displays three separate browser windows side-by-side, each showing a different aspect of a restaurant's ordering system.

- Left Window:** A self-service POS interface titled "selfservicepos". It features a large green button labeled "Start Ordering".
- Middle Window:** A "kitchenmonitor" interface titled "Kitchen Monitor". It displays the text "Kitchen Monitor" above a blank white area.
- Right Window:** A "customerorderstatus" interface titled "orderstatus.localtest.me". It has two main sections: "Orders in Preparation" and "Finished Orders", both currently empty.



ARCHITECTING AND IMPLEMENTING  
DISTRIBUTED APPLICATIONS  
IS NOT STRESSFUL AT ALL

John, 25 years old

# Challenges with Distributed Apps



## Complexity

Many services and components  
Scaling, Orchestration



## Communication

Service-to-service communication  
HA / Failover  
Resilience, Security, Tracing



## Decoupling

Events  
Competing Consumer Pattern  
Retry / Poisonous message detection



## State Handling

Single state across multiple instances  
Stateful services  
Virtual Actors, Workflows



## Different Target Environments

Local dev environment vs. cloud environment  
Different persistency services



Portable, event-driven runtime



Build connected distributed applications faster



APIs for solving distributed application challenges



Cloud and Edge

The image displays two side-by-side screenshots of the Cloud Native Computing Foundation (CNCF) website. Both screenshots feature a blue header bar with the CNCF logo, navigation links (About, Projects, Training, Community, Blog & News), and a call-to-action button "REGISTER TODAY".

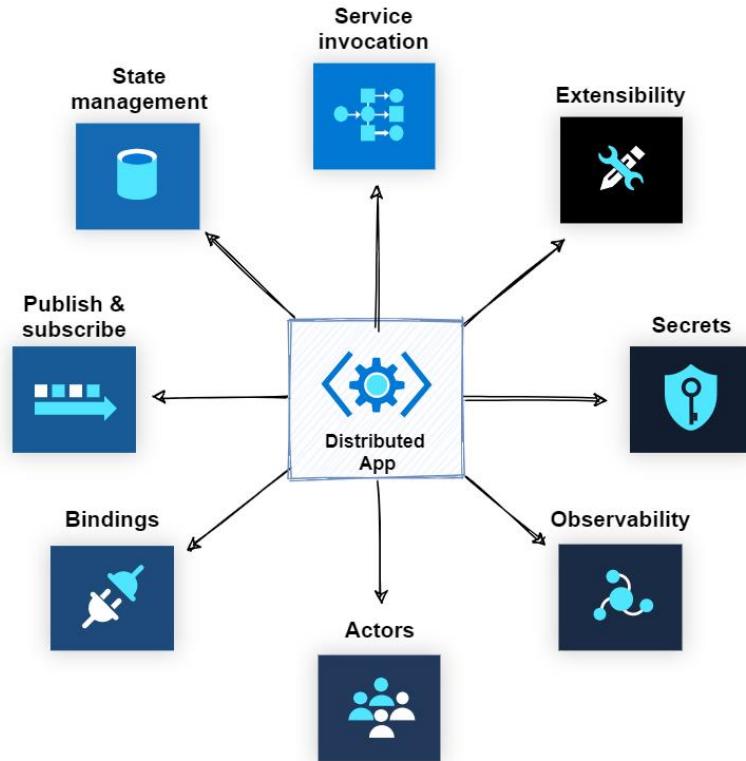
**Left Screenshot (Announcements Page):**

- Section Header:** ANNOUNCEMENTS
- Title:** Cloud Native Computing Foundation Announces Dapr Graduation
- Text:** Dapr provides a set of integrated APIs for building reliable and secure distributed applications, increasing developer productivity by 20-40%.
- Text:** SALT LAKE CITY, Utah – KubeCon + CloudNativeCon North America – November 12, 2024 – The Cloud Native Computing Foundation®, which builds sustainable ecosystems for cloud native software, today announced the graduation of Dapr.
- Text:** Dapr, or Distributed Application Runtime, is a portable runtime that makes it easy for any developer to build resilient distributed applications that run across cloud and edge. It provides integrated APIs for communication, state, and workflow for building production-ready applications. It leverages industry best practices for security, resiliency, and observability, increasing developer productivity by between 20 and 40 percent.
- Text:** "Dapr has a single mission: to meet the emerging needs of developers and solve the most complex problems in distributed computing," said Yaron Scheider, Dapr maintainer and Steering Committee member and CTO, co-founder of Diagrid. "The project has done very well in helping application developers navigate the complexities of cloud native architectures, and the engagement with the CNCF community proved to be an amazing catalyst for the project's growth and maturity."
- Text:** The project was first released in 2019 at Microsoft and was accepted into the CNCF Incubator in November 2021. Since then, Dapr has grown to over 3,700 individual contributors from more than 400 organizations. It is used by tens of thousands of organizations, including Grafana, FICO, HDFC Bank, SharperImage, Zeiss and more. Today, it is maintained by 21 individuals affiliated with eight organizations who have released regular versions every quarter with many new developer APIs, including workflows, secrets, cryptography, configuration management, and LLMs. Together, the Dapr SDKs have over 70 million downloads, with 50 million image pulls.
- Text:** "Dapr's API approach coupled with its ability to rapidly swap underlying infrastructure, whether that is storage, message brokers or secret stores, has enabled any developer to tackle the complexities of building microservice architectures and deliver business value," said Mark Fussell, Dapr maintainer and Steering Committee member, CEO of Diagrid.

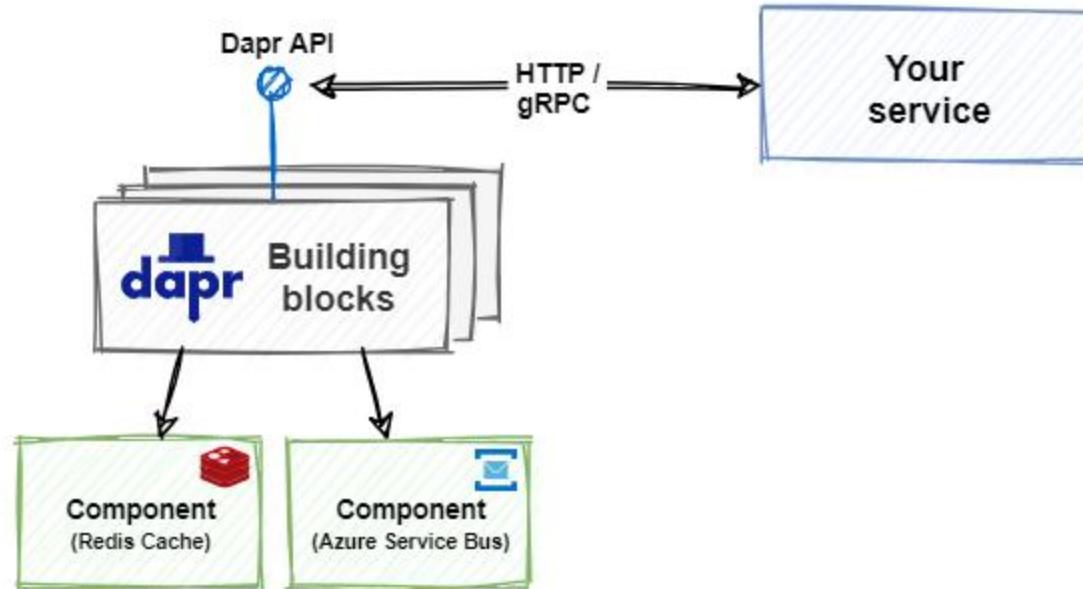
**Right Screenshot (Dapr Project Page):**

- Section Header:** PROJECTS
- Title:** Dapr
- Text:** Dapr is a portable, event-driven, runtime for building distributed applications across cloud and edge.
- Text:** Dapr was accepted to CNCF on November 9, 2021 at the Incubating maturity level and then moved to the Graduated maturity level on October 30, 2024.
- Text:** VISIT PROJECT WEBSITE
- Icons:** GitHub, Docker, Kubernetes, Helm, and others.

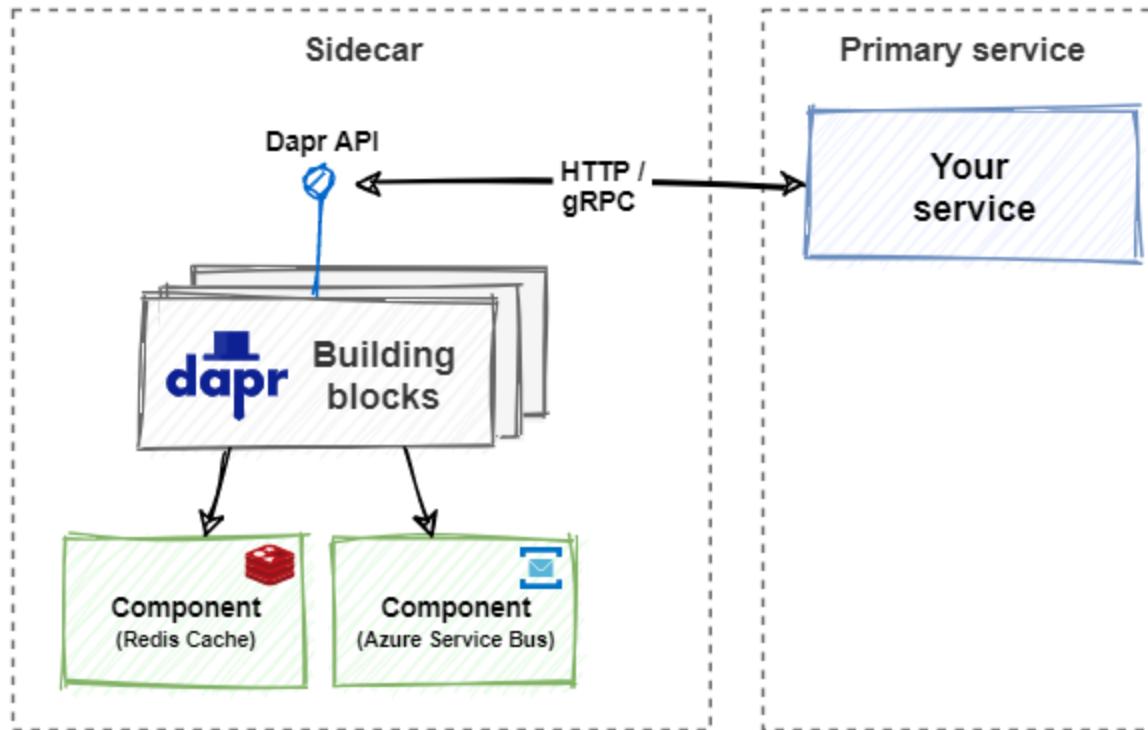
# Dapr Building Blocks



# Dapr Building Blocks Abstraction



# Dapr Sidecar Architecture





# Dapr Basics

4tecture®  
empower your software solutions

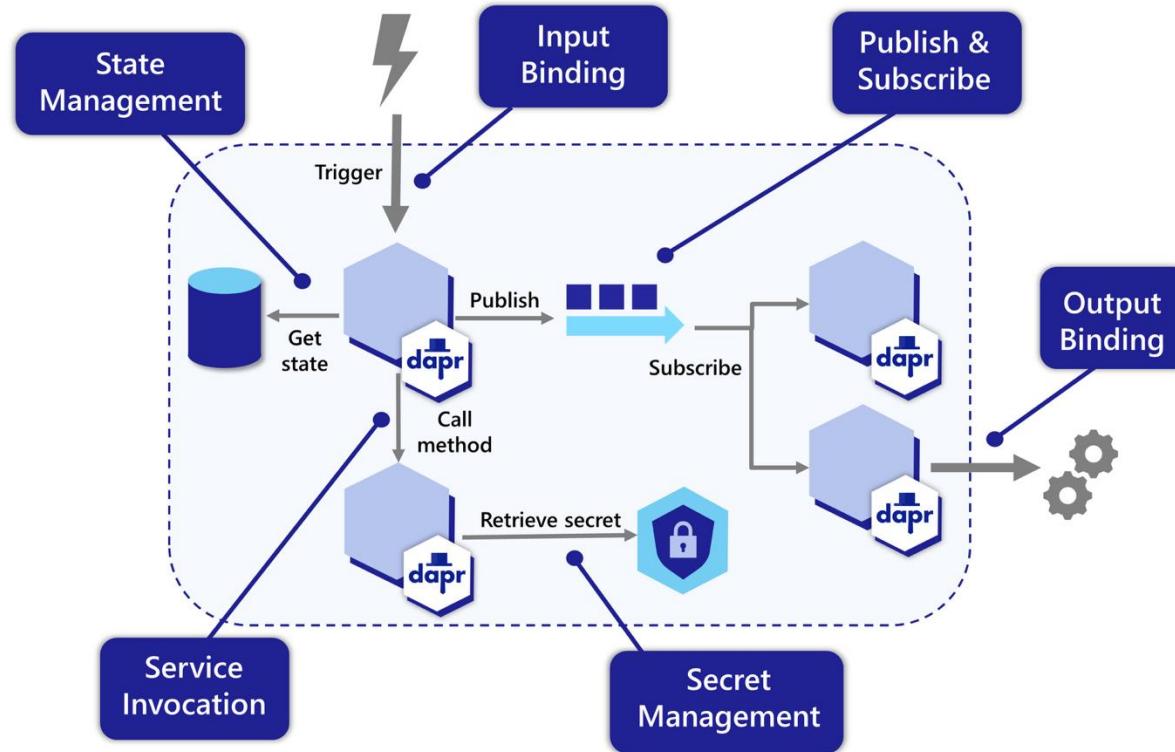
# Dapr

- [dapr.io](https://dapr.io)
- Open source
- Originated at Microsoft
- Cloud Native Computing Foundation (CNCF) – graduated maturity level

# Dapr – High Level Definition

- Any language or framework
- Portable APIs
- Building blocks applying best practices
  - Use the blocks you need
  - No big bang framework
- Platform agnostic
- Extensible and pluggable components

# Dapr Building Blocks Overview



# Single API - Multiple Components

```
var weatherForecast =  
await daprClient.GetStateAsync<WeatherForecast>("mystatestore", "ZH");  
  
daprClient.SaveStateAsync("mystatestore", "ZH", weatherForecast);
```

- AWS DynamoDB
- Aerospike
- Azure Blob Storage
- Azure CosmosDB
- Azure Table Storage
- Cassandra
- Cloud Firestore (Datastore mode)
- CloudState
- Couchbase
- Etcd
- HashiCorp Consul
- Hazelcast
- Memcached
- MongoDB
- PostgreSQL
- Redis
- RethinkDB
- SQL Server
- Zookeeper

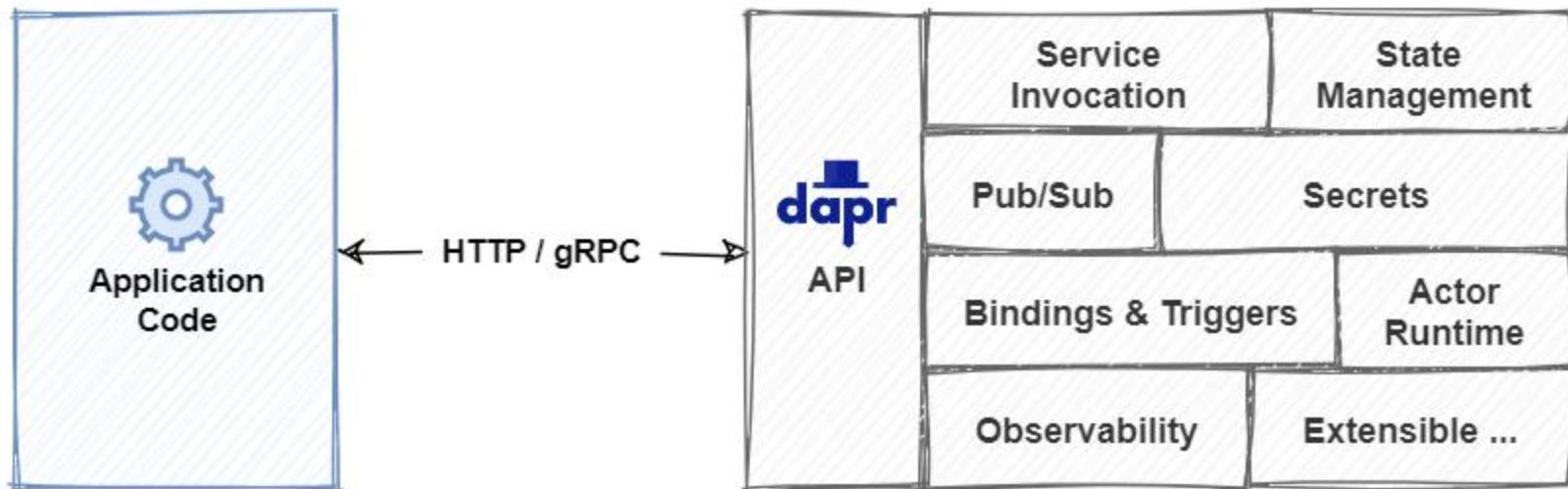


The power of side-cars!

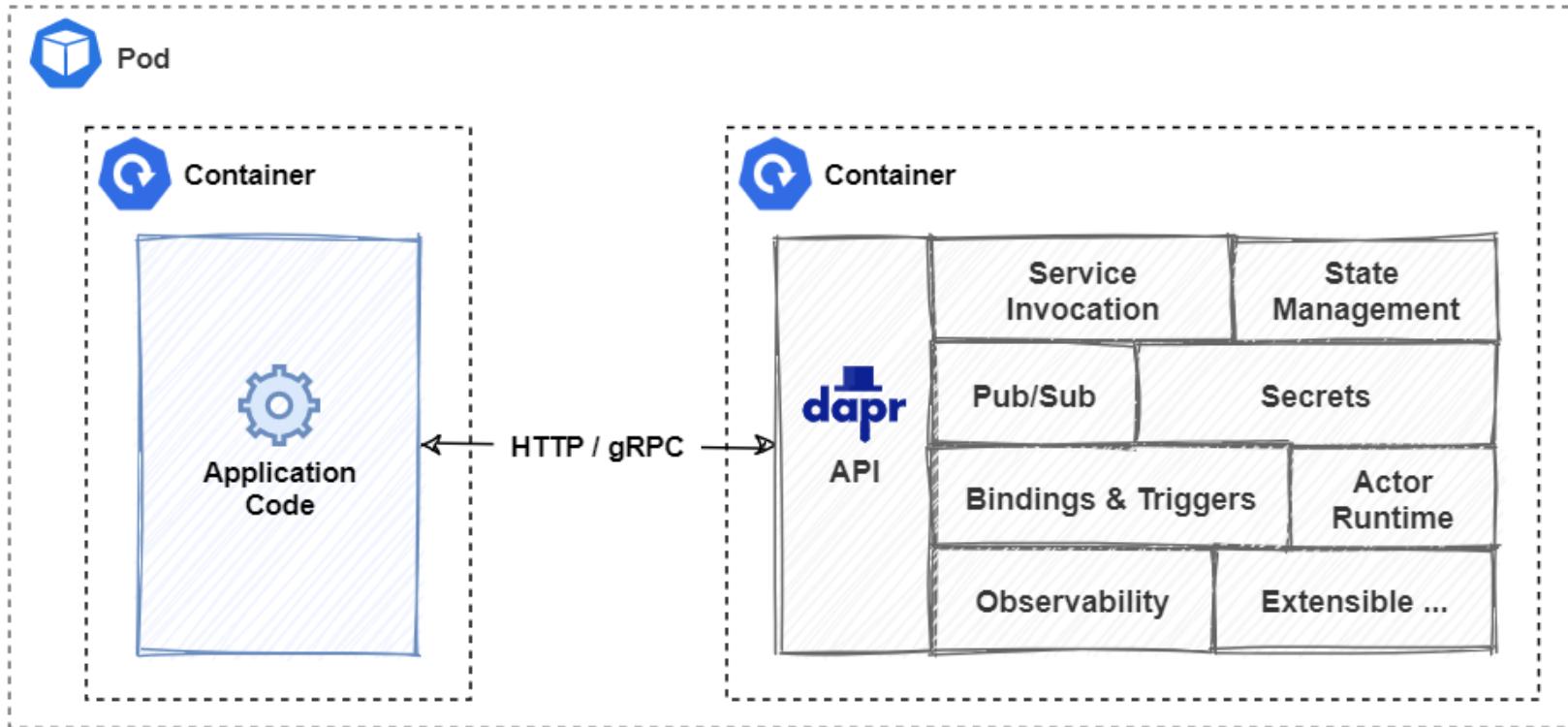
The Dapr sidecar provides built-in  
**security**, **resiliency** and **observability**  
capabilities.

Speeds up application development by providing an integrated set of APIs for communication, state, and workflow.

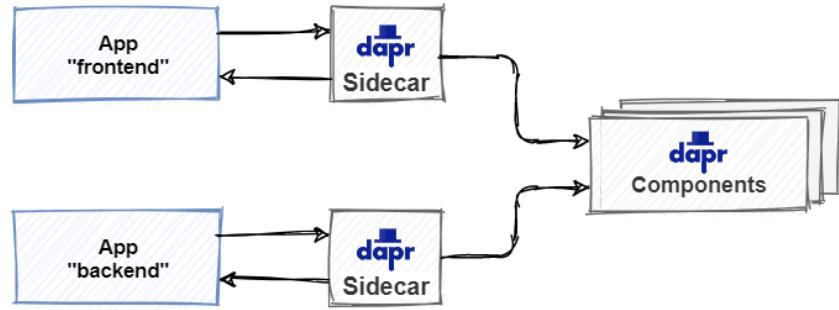
# Self-hosted Sidecar



# Kubernetes-hosted Sidecar



# Sidecar Performance Considerations



- Dapr operation:  
>= 1 out-of-process network call
- Heavily optimized sidecar implementation
- gRPC with multiplexing, bidirectional full-duplex, streaming
- Overhead should be sub-millisecond

# SDK Overview

Language	Status	Client	Server extensions	Actor	Workflow
.NET	Stable	✓	ASP.NET Core	✓	✓
Python	Stable	✓	gRPC FastAPI Flask	✓	✓
Java	Stable	✓	Spring Boot Quarkus	✓	✓
Go	Stable	✓	✓	✓	✓
PHP	Stable	✓	✓	✓	
Javascript	Stable	✓		✓	✓
C++	In development	✓			
Rust	In development	✓		✓	

# Runtimes



## Self-Hosted

- Dedicated process next to your application process
- Executable or Docker Image

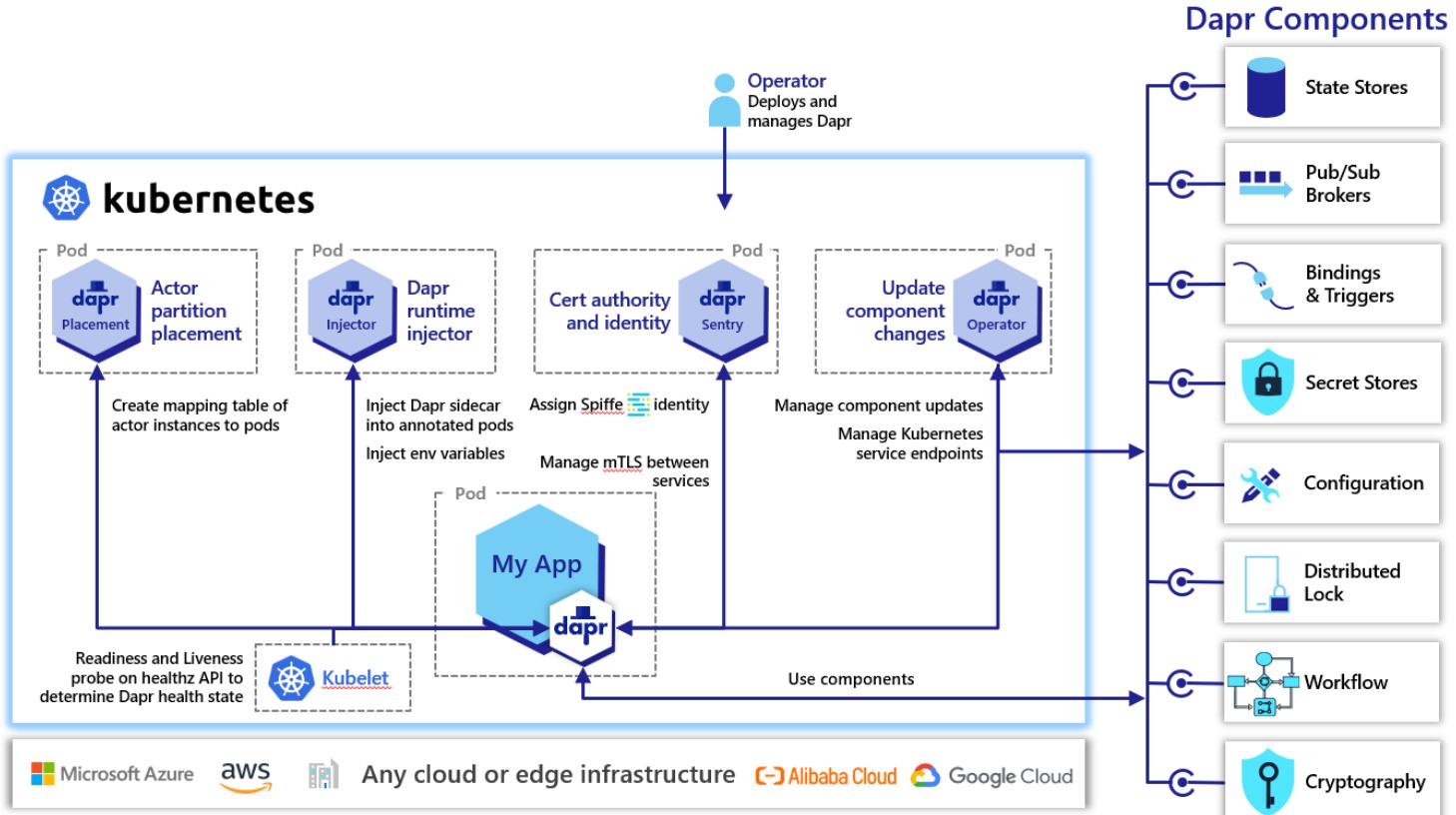
## Kubernetes

- Sidecar container in your pod, uses localhost interface
- Usually injected based on attributes

## Serverless

- Integrated in Azure
- i.e. Container Apps

# Dapr in Kubernetes



# Dapr from development to hosting

Use any language or runtime



HTTP/gRPC

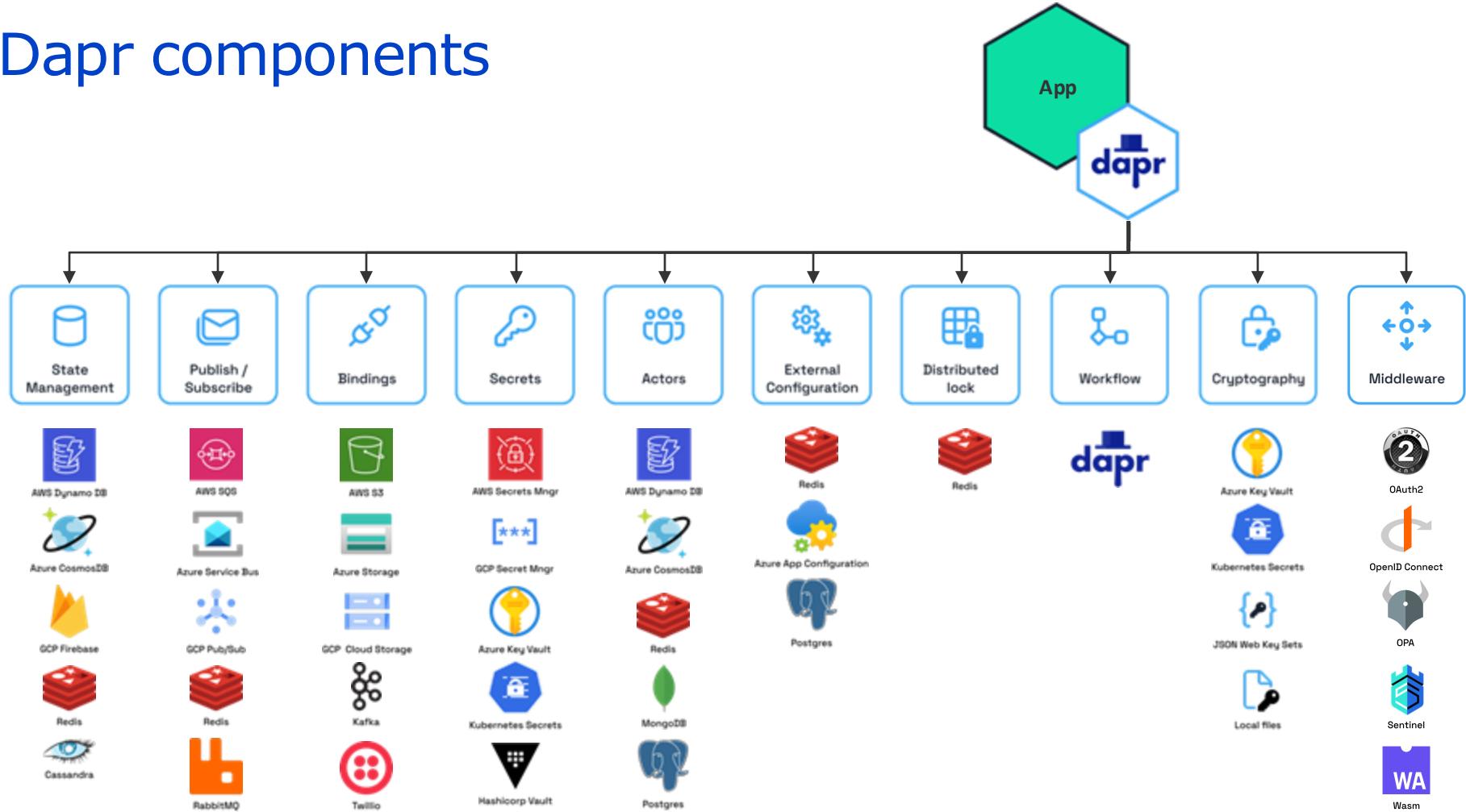


Host on any cloud or edge infrastructure



Virtual or  
physical machines

# Dapr components



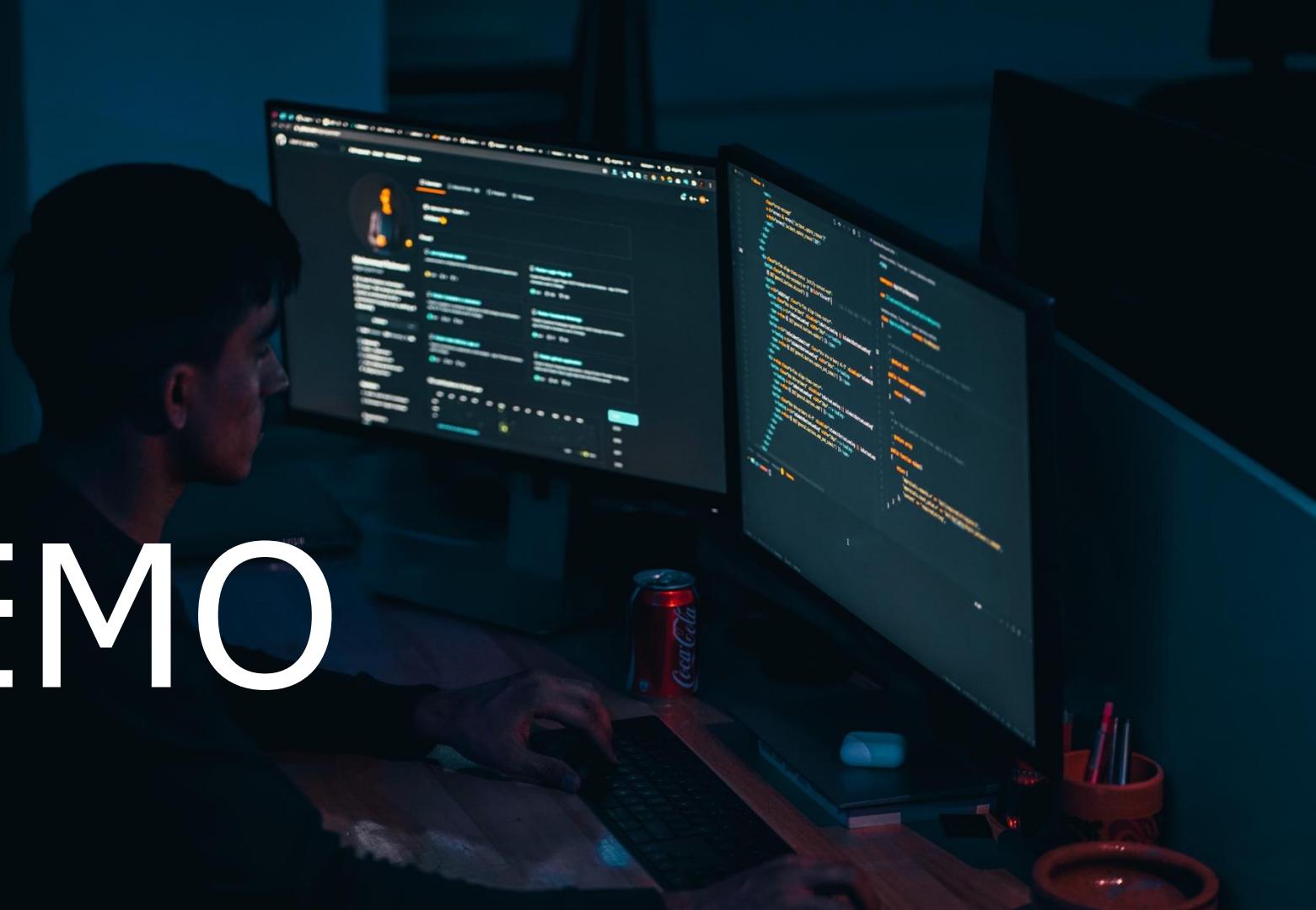


# Hands-on!

4tecture®  
empower your software solutions

# DEMO

Setup



The screenshot shows the Visual Studio Code interface with the following details:

- Solution Explorer:** Displays the project structure for "FastFoodDelivery". The "order" project is selected. Other projects include "Common", "Services", and "SystemTests".
- File Explorer:** Shows files like "appsettings.json", "values.yaml", and "Program.cs".
- Editor:** The "Program.cs" file is open, showing C# code for a WebApplication builder.
- Red Box:** A red rectangular box highlights the following code block, which configures Dapr clients for HTTP and gRPC endpoints:

```
1 {} > using ...
10
11 var builder = WebApplication.CreateBuilder(args);
12
13 var observabilityOptions = builder.Configuration.GetObservabilityOptions();
14 builder.Services.AddObservability<IOrderServiceObservability, OrderServiceObservability>(observabilityOptions, observabilityFactory);
15
16 var daprHttpPort:string = Environment.GetEnvironmentVariable("DAPR_HTTP_PORT") ?? "3600";
17 var daprGrpcPort:string = Environment.GetEnvironmentVariable("DAPR_GRPC_PORT") ?? "60600";
18 builder.Services.AddDaprClient(configure:builder => builder
19     .UseHttpEndpoint($"http://localhost:{daprHttpPort}")
20     .UseGrpcEndpoint($"http://localhost:{daprGrpcPort}")
21     .UseJsonSerializationOptions(new JsonSerializerOptions().ConfigureJsonSerializerOptions()));
22
23 builder.Services.AddSingleton<IOrderEventRouter, OrderEventRouter>();
24 builder.Services.AddSingleton<IOrderProcessingServiceActor, OrderProcessingServiceActor>();
25 builder.Services.AddSingleton<IOrderProcessingServiceState, OrderProcessingServiceState>();
26 builder.Services.AddSingleton<IOrderProcessingServiceWorkflow, OrderProcessingServiceWorkflow>();
27
28 builder.Services.AddDaprWorkflow(configure: options =>
29 {
30     options.RegisterWorkflow<OrderProcessingWorkflow>();
31     options.RegisterActivity<CreateOrderActivity>();
32     options.RegisterActivity<AssignCustomerActivity>();
33     options.RegisterActivity<AssignInvoiceAddressActivity>();
34     options.RegisterActivity<AssignDeliveryAddressActivity>();
35     options.RegisterActivity<AddItemActivity>();
36     options.RegisterActivity<RemoveItemActivity>();
37     options.RegisterActivity<ConfirmOrderActivity>();
38     options.RegisterActivity<ConfirmPaymentActivity>();
39     options.RegisterActivity<StartProcessingActivity>();
```

**Status Bar:** Shows the file path "FastFoodDelivery > Services > order > OrderService > Program.cs", the status "32:56", and other system information.

```
dapr run  
  --app-id orderservice  
  --app-port 8601  
  --dapr-http-port 3600  
  --dapr-grpc-port 60600  
  --config ..../dapr/config/config.yaml  
  --resources-path ..../dapr/components  
dotnet run
```

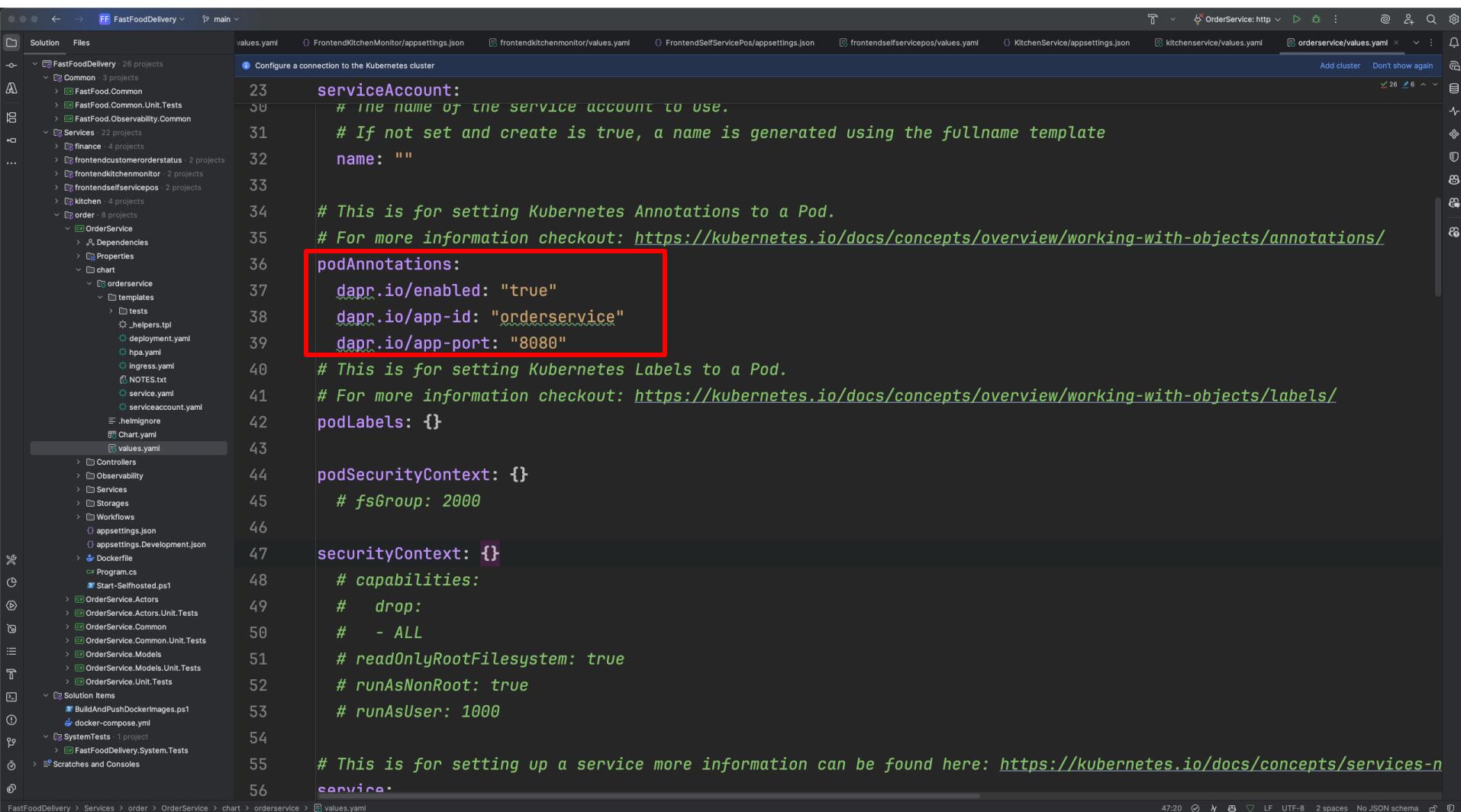
The screenshot shows a Visual Studio Code interface with a PowerShell terminal window open. The terminal contains a command to run a Dapr application with specific configuration and resource paths. A red box highlights the first few lines of the command. The bottom line, 'dotnet run', is also highlighted with a red box. The left sidebar shows a solution structure for a 'FastFoodDelivery' project, including various services like 'Finance', 'Kitchen', and 'Order'. The 'order' service is currently selected. The status bar at the bottom indicates the file is 'main' and shows system information like battery level, network, and encoding.

FastFoodDelivery main

File Explorer

- FastFoodDelivery - 26 projects
  - Common - 3 projects
    - FastFood.Common
    - FastFood.Common.Unit.Tests
    - FastFood.Observability.Common
  - Services - 22 projects
    - finance - 4 projects
    - frontendcustomerorderstatus - 2 projects
    - frontentkitchenmonitor - 2 projects
    - frontendservicecpes - 2 projects
    - kitchen - 4 projects
    - order - 8 projects
      - OrderService
        - Properties
        - chart
        - Controllers
        - Observability
        - Services
        - Storages
        - Workflows
        - appsettings.json
        - appsettings.Development.json
      - Dockerfile
      - Program.cs
    - Start-Selfhosted.ps1
  - OrderService.Actions
  - OrderService.Actors.Unit.Tests
  - OrderService.Common
  - OrderService.Common.Unit.Tests
  - OrderService.Models
  - OrderService.Models.Unit.Tests
  - OrderService.Unit.Tests
  - Solution Items
    - BuildAndPushDockerImages.ps1
    - docker-compose.yml
  - SystemTests 1 project
    - FastFoodDelivery.System.Tests
  - Scratches and Consoles

```
services:  
  orderservice:  
    container_name: orderservice  
    image: orderservice  
    build:  
      context: .  
      dockerfile: services/order/OrderService/Dockerfile  
    ports:  
      - "8601:8080"  
    environment:  
      - DAPR_HTTP_PORT=3500  
      - DAPR_GRPC_PORT=50001  
      - Observability__UseTracingExporter=otlp  
      - Observability__UseMetricsExporter=otlp  
      - Observability__UseLogExporter=otlpAndAnsiConsole  
      - Observability__SamplerType=AlwaysOnSampler  
      - Observability__OtlpExporter__Endpoint=http://otel-collector:4317  
    networks:  
      - fastfoodnet  
  
  orderservice-dapr:  
    container_name: orderservice-dapr  
    image: daprio/daprd:1.15.5  
    command: ["./daprd",  
              "--app-id", "orderservice",  
              "--app-port", "8080",  
              "--placement-host-address", "placement:50003",  
              "--resources-path", "/resources",  
              "--config", "/config/config.yaml",  
              "--sentry-address", "sentry:50001",  
              "--log-level", "info",  
              "--control-plane-trust-domain", "cluster.local",  
              "--enable-mtls"]  
    volumes:  
      - "../infrastructure-dev/dapr/resources/:/resources"  
      - "../infrastructure-dev/dapr/config/:/config"  
    env_file:  
      - ../infrastructure-dev/dapr/certs/mtls.env  
    depends_on: <5 keys>  
    network_mode: "service:orderservice"
```



```
metadata:
  name: pubsub
  namespace: fastfood
spec:
  type: pubsub.rabbitmq
  version: v1
  metadata:
    - name: host
      value: "amqp://rabbitmq:5672"
    - name: durable
      value: "false"
    - name: deletedWhenUnused
      value: "false"
    - name: autoAck
      value: "false"
    - name: reconnectWait
      value: "0"
    - name: concurrency
      value: parallel
  scopes:
    - orderservice
    - kitchenservice
    - financeservice
    - orderserviceactors
    - frontendselfservicepos
    - frontendkitchenmonitor
    - frontendcustomerorderstatus
```

The image shows a screenshot of a code editor with a dark theme. On the left is a file tree for a project named 'FastFoodDelivery'. Two specific sections of the code are highlighted with red boxes: the 'pubsub.yaml' file and a 'dapr' configuration section. The 'pubsub.yaml' file contains configuration for a RabbitMQ-based pubsub service, specifying its name, namespace, type, version, and various metadata and scope settings. The 'dapr' section, located in the 'chart/dapr/templates' directory, defines templates for dapr components like PubSub, Resiliency, Secrets, and StateStore, along with their respective YAML files. The code editor interface includes tabs for other files like 'appsettings.json', 'values.yaml', and 'appsettings.json' for other services.

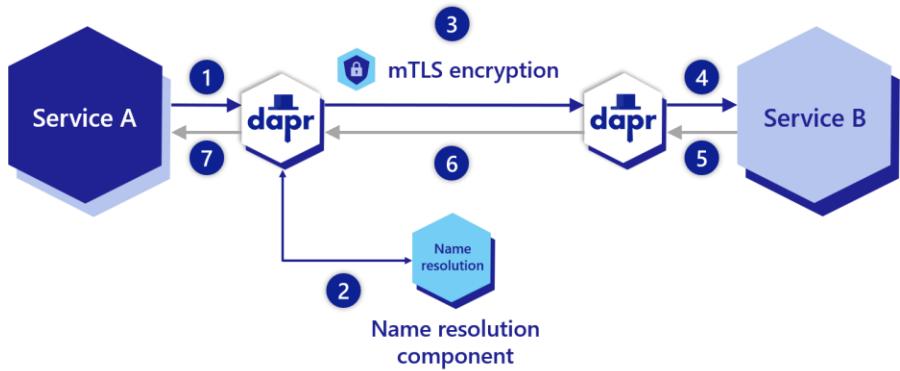
A close-up, low-angle shot of several rowers in a racing shell. Their legs and torsos are visible as they pull on their oars. The water is choppy, creating white spray at the stern. The background is blurred.

Dapr Building Blocks

# Service Invocation and Configuration

4tecture®  
empower your software solutions

# Service Invocation

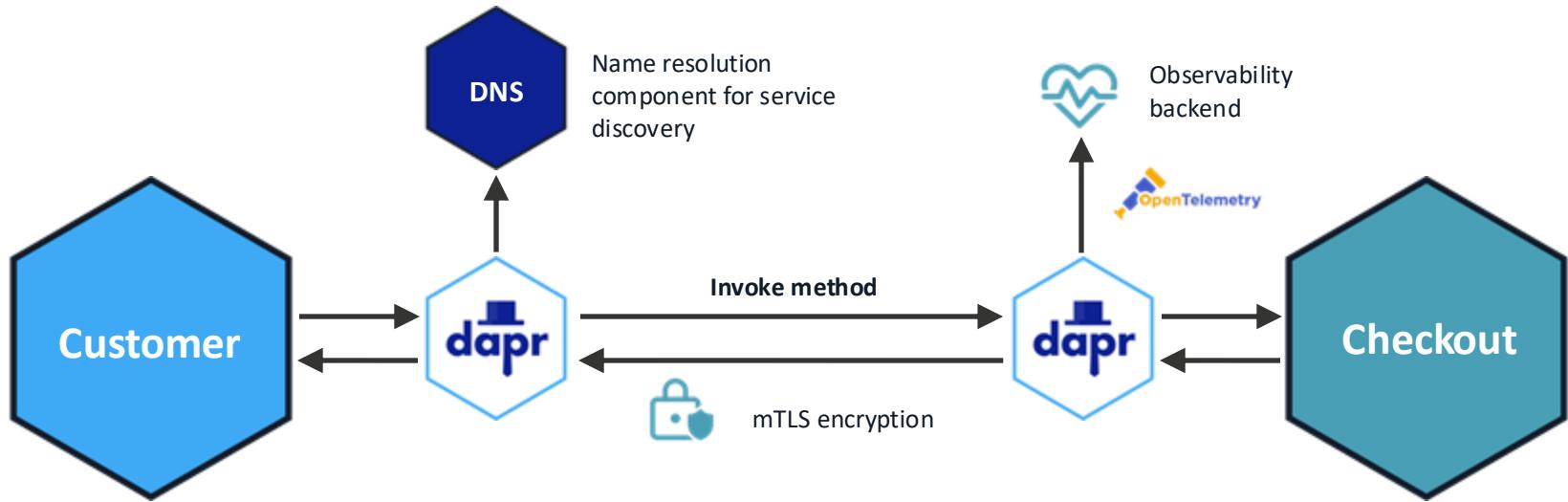


- HTTP and gRPC
- mTLS (with Dapr Sentry)
- Resiliency including retries
- Tracing and metrics with observability
- Access control (policies)
- Namespace scoping
- Load balancing (round robin with mDNS)
- Pluggable service discovery

# Service Invocation



Service  
Invocation



**POST**  
[http://localhost:3500/v1.0/invite/checkout/method/order](http://localhost:3500/v1.0/invoke/checkout/method/order)

**POST**  
<http://localhost:5100/order>

# DEMO

Service Invocation



```
public partial class OrderProcessingServiceState : IOrderProcessingServiceState
{
    public async Task ConfirmOrder(Guid orderid)
    {
        throw new InvalidOperationException("Order is not in the correct state to confirm");
    }

    public async Task ConfirmPayment(Guid orderid)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var order = await _daprClient.GetStateAsync<Order>(FastFoodConstants.StateStoreName, key: GetStateId(orderid));
        if (order.State == OrderState.Confirmed)
        {
            order.State = OrderState.Paid;
            order.PaidAt = DateTimeOffset.UtcNow;

            _observability.OrdersPaidCounter.Add(delta: 1);
            _observability.OrderItemsCount.Record(order.Items?.Count ?? 0);
            _observability.OrderTotalAmount.Record(order.Items?.Select(o => o.Price).Sum());
            _observability.OrderSalesDuration.Record((order.PaidAt - order.CreatedAt).TotalSeconds);

            await _daprClient.SaveStateAsync(FastFoodConstants.StateStoreName, key: GetStateId(orderid), value: order);
            await _daprClient.PublishEventAsync(FastFoodConstants.PubSubName, topicName: FastFoodConstants.EventNames.OrderPaid, data: order.ToDto());
        }
        else
        {
            throw new InvalidOperationException("Order is not in the correct state to confirm payment");
        }
    }

    public async Task StartProcessing(Guid orderid)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var order = await _daprClient.GetStateAsync<Order>(FastFoodConstants.StateStoreName, key: GetStateId(orderid));
        if (order.State == OrderState.Paid)
        {
    }

    public static class Services
    {
        public const string FinanceService = "financeservice";
        public const string OrderService = "orderservice";
        public const string KitchenService = "kitchenservice";
    }
}

await _daprClient.InvokeMethodAsync<Order>(HttpMethod.Post, appId: FastFoodConstants.Services.FinanceService, methodName: "api/OrderFinance/newOrder", data: order.ToFinanceDto());
```

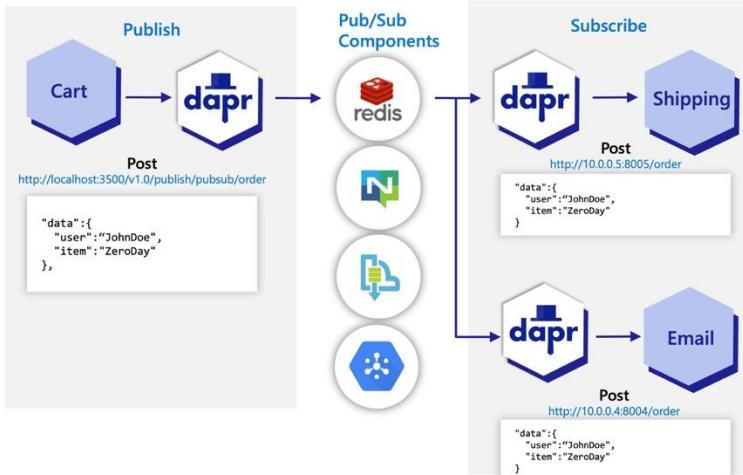
A close-up, slightly blurred photograph of several rowers in a boat. They are wearing blue and red athletic gear. Their oars are extended into the water, creating a spray of white foam. The perspective is from the side of the boat, looking towards the stern.

Dapr Building Blocks

# Publish & Subscribe

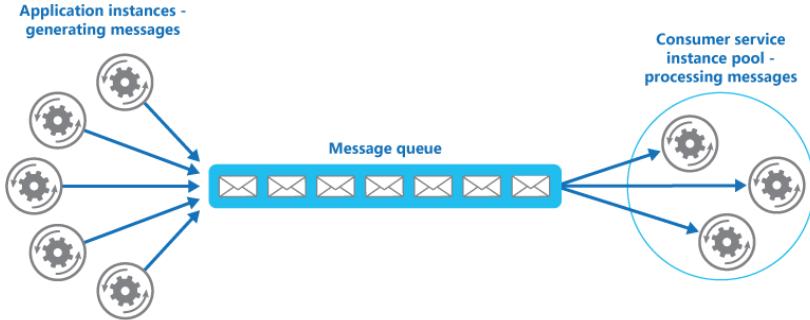
4tecture®  
empower your software solutions

# Publish & Subscribe



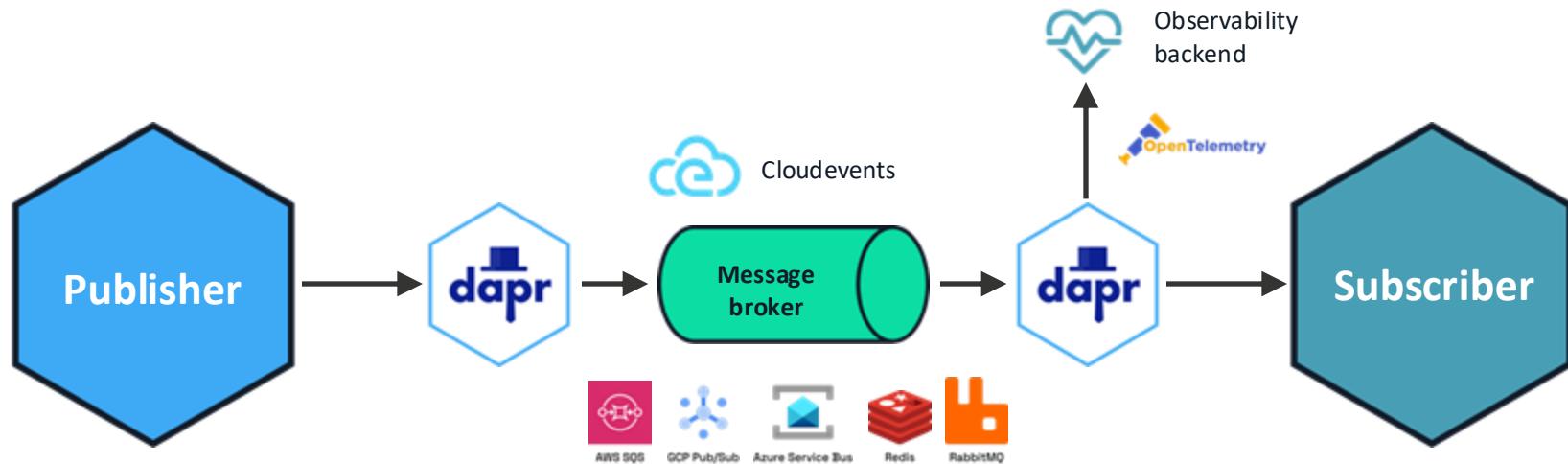
- Platform-agnostic API to send and receive messages
- At-least-once message delivery guarantee
- Integration with various message brokers
- CloudEvents 1.0 specification
- Message content type
- Content-based Routing
- Dead letter topics
- Namespace consumer groups
- Scoping topics

# Competing consumers pattern



- Multiple application instances using a single consumer group
- Same app id = same consumer group
- Dapr delivers each message to only one instance of that application

# Publish / Subscribe



**POST**  
<http://localhost:3500/v1.0/publish/mybroker/order-messages>

**POST**  
<http://localhost:5100/orders>

# Pub/Sub Brokers

## Generic

Component	Status	Component version	Since runtime version
Apache Kafka	Stable	v1	1.5
In-memory	Stable	v1	1.7
JetStream	Beta	v1	1.10
KubeMQ	Beta	v1	1.10
MQTT3	Stable	v1	1.7
Pulsar	Stable	v1	1.10
RabbitMQ	Stable	v1	1.7
Redis Streams	Stable	v1	1.0
RocketMQ	Alpha	v1	1.8
Solace-AMQP	Beta	v1	1.10

## Amazon Web Services (AWS)

Component	Status	Component version	Since runtime version
AWS SNS/SQS	Stable	v1	1.10

## Google Cloud Platform (GCP)

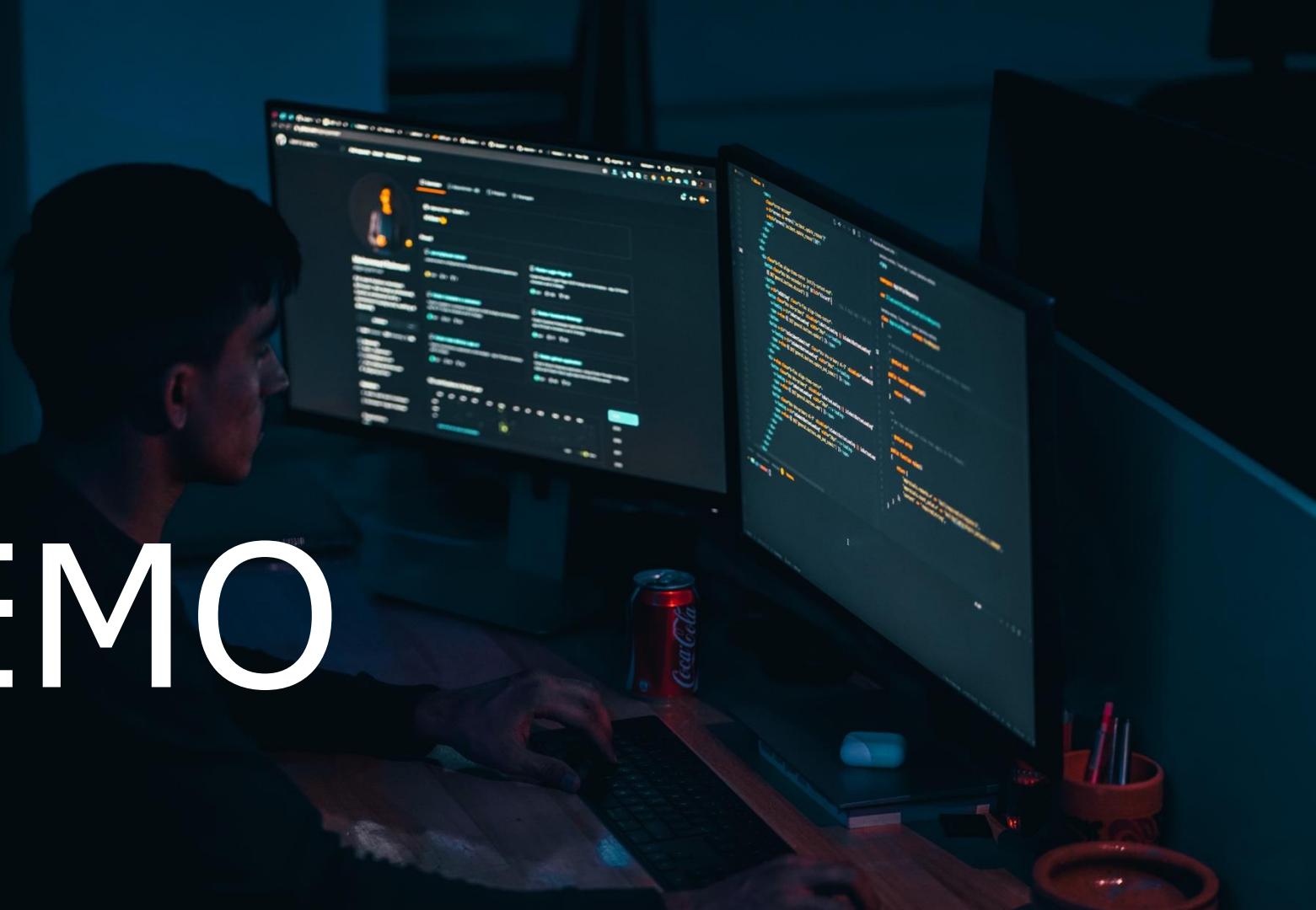
Component	Status	Component version	Since runtime version
GCP Pub/Sub	Stable	v1	1.11

## Microsoft Azure

Component	Status	Component version	Since runtime version
Azure Event Hubs	Stable	v1	1.8
Azure Service Bus Queues	Beta	v1	1.10
Azure Service Bus Topics	Stable	v1	1.0

# DEMO

PubSub



Solution Files

FastFoodDelivery - 26 projects

- Common - 3 projects
  - FastFood.Common
  - FastFood.Common.Unit.Tests
  - FastFood.Observability.Common
- Services - 22 projects
  - Finance - 4 projects
  - frontedncustomerorderstatus - 2 projects
  - frontendkitchenmonitor - 2 projects
  - frontendservicecipes - 2 projects
  - kitchen - 4 projects
    - KitchenService
    - Dependencies
    - Properties
    - chart
    - Controllers
    - Entities
    - Helpers
    - Observability
    - Services
      - IKitchenService.cs
      - KitchenService.cs
  - appsettings.json
  - appsettings.Development.json
- Dockerfile
- Program.cs
- Start-Selfhosted.ps1
- KitchenService.Common
- KitchenService.Common.Unit.Tests
- KitchenService.Unit.Tests
- order - 8 projects
  - OrderService
    - Dependencies
  - Properties
  - chart
  - Controllers
    - DeadLetterHandlerController.cs
    - OrderActorController.cs
    - OrderController.cs
    - OrderStateController.cs
    - OrderUpdateEventHandlerController.cs
    - OrderWorkflowController.cs
    - SecretDemoController.cs
  - Observability
  - Services
    - IOrderEventRouter.cs
    - IOrderProcessingService.cs
    - IOrderEventRouter.cs
    - OrderEventRoutingTarget.cs
    - OrderProcessingServiceActor.cs
    - OrderProcessingServiceState.cs
    - OrderProcessingServiceWorkflow
  - Storage
  - Workflow

```
public class KitchenService : IKitchenService
{
    public Task<KitchenOrderItem> GetPendingItems()
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        return Task.FromResult(_mockStorage.Values.SelectMany(o:KitchenOrder => o.Items).Where(i:KitchenOrderItem => i.State == KitchenOrderItemState.AwaitingPreparation));
    }

    public Task<KitchenOrderItem> SetItemAsFinished(Guid id)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var item = _mockStorage.Values.SelectMany(o:KitchenOrder => o.Items).FirstOrDefault(i:KitchenOrderItem => i.Id == id);
        if (item != null)
        {
            item.State = KitchenOrderItemState.Finished;
            item.FinishedAt = DateTimeOffset.UtcNow;
            _observability.OrderItemPreparationDuration.Record((item.FinishedAt - item.CreatedAt).TotalSeconds);

            _daprClient.PublishEventAsync(FastFoodConstants.PubSubName, topicName: "kitchenitemfinished", new KitchenItemFinishedEvent(){ OrderId = item.OrderId, ItemId = item.Id });

            var order = _mockStorage[item.OrderId];
            if (order.Items.All(i:KitchenOrderItem => i.State == KitchenOrderItemState.Finished))
            {
                order.FinishedAt = DateTimeOffset.UtcNow;
                _observability.OrderPreparationDuration.Record((order.FinishedAt - order.CreatedAt).TotalSeconds);
                _mockStorage.Remove(order.Id);
            }
        }

        return Task.FromResult(item);
    }

    activity?.setStatus(ActivityStatusCode.Error, description: "Item not found");
    throw new InvalidOperationException("Item not found");
}
}
```

```
builder.Services.AddControllers()
    .AddJsonOptions(options =>
{
    options.JsonSerializerOptions.ConfigureJsonSerializerOptions();
})
    .AddDapr();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddHealthChecks();

var app :WebApplication = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCloudEvents();

app.UseObservability(observabilityOptions);

app.MapControllers();
app.MapSubscribeHandler();

app.MapHealthChecks( pattern: "/healthz");

app.Run();
```

FastFoodDelivery > Services > order > OrderService > Controllers > OrderUpdateEventHandlerController.cs > OrderUpdateEventHandlerController

```
12     public class OrderUpdateEventHandlerController : ControllerBase
13     {
14         private async Task<IOrderProcessingService> GetOrderProcessingServiceByOrderId(Guid orderId)
15         {
16             switch(serviceType)
17             {
18                 case ServiceType.FastFood:
19                     return await _fastFoodProcessingService;
20                 case ServiceType.Order:
21                     return await _orderProcessingService;
22                 case ServiceType.Finance:
23                     return await _financeProcessingService;
24                 default:
25                     throw new ArgumentException("Unknown service type");
26             }
27         }
28     }
29
30     [HttpPost(template: "orderitemfinished")]
31     [Topic(FastFoodConstants.PubSubName, name: FastFoodConstants.EventNames.KitchenItemFinished, DeadLetterTopic = FastFoodConstants.EventNames.DeadLetterKitchenItemFinished)]
32     public async Task<ActionResult> OrderItemFinished(KitchenItemFinishedEvent itemEvent, [FromServices] DaprClient daprClient)
33     {
34         try
35         {
36             if (_failForDemo)
37             {
38                 _logger.LogWarning("Processing failed for demo purposes");
39                 throw new Exception("Processing failed for demo purposes");
40             }
41
42             await (await GetOrderProcessingServiceByOrderId(itemEvent.OrderId)).FinishedItem(itemEvent.OrderId, itemEvent.ItemId);
43             _logger.LogInformation("Finished item event received for item {ItemId} in order {OrderId}", itemEvent.ItemId, itemEvent.OrderId);
44             return Ok();
45         }
46         catch(Exception ex)
47         {
48             _logger.LogError(ex, message: "Error processing item finished for order {OrderId} and item {ItemId}", itemEvent.OrderId, itemEvent.ItemId);
49             return StatusCode(500);
50         }
51     }
52
53     [HttpPost(template: "orderstartprocessing")]
54     [Topic(FastFoodConstants.PubSubName, name: FastFoodConstants.EventNames.KitchenOrderStartProcessing, DeadLetterTopic = FastFoodConstants.EventNames.DeadLetterKitchenOrderStartProcessing)]
55     public async Task<ActionResult> OrderStartProcessing(KitchenOrderStartProcessingEvent orderProcessingEvent, [FromServices] DaprClient daprClient)
56     {
57         try
58         {
59             if (_failForDemo)
60             {
61                 _logger.LogWarning("Processing failed for demo purposes");
62                 throw new Exception("Processing failed for demo purposes");
63             }
64
65             await (await GetOrderProcessingServiceByOrderId(orderProcessingEvent.OrderId)).StartProcessing(orderProcessingEvent.OrderId);
66             _logger.LogInformation("Order start processing event received for order {OrderId}", orderProcessingEvent.OrderId);
67             return Ok();
68         }
69         catch(Exception ex)
70         {
71             _logger.LogError(ex, message: "Error processing order start processing for order {OrderId}", orderProcessingEvent.OrderId);
72             return StatusCode(500);
73         }
74     }
75
76     [HttpPost(template: "orderprocessing")]
77     [Topic(FastFoodConstants.PubSubName, name: FastFoodConstants.EventNames.KitchenOrderProcessing, DeadLetterTopic = FastFoodConstants.EventNames.DeadLetterKitchenOrderProcessing)]
78     public async Task<ActionResult> OrderProcessing(KitchenOrderProcessingEvent orderProcessingEvent, [FromServices] DaprClient daprClient)
79     {
80         try
81         {
82             if (_failForDemo)
83             {
84                 _logger.LogWarning("Processing failed for demo purposes");
85                 throw new Exception("Processing failed for demo purposes");
86             }
87
88             await (await GetOrderProcessingServiceByOrderId(orderProcessingEvent.OrderId)).Process(orderProcessingEvent.OrderId);
89             _logger.LogInformation("Order processing event received for order {OrderId}", orderProcessingEvent.OrderId);
90             return Ok();
91         }
92         catch(Exception ex)
93         {
94             _logger.LogError(ex, message: "Error processing order processing for order {OrderId}", orderProcessingEvent.OrderId);
95             return StatusCode(500);
96         }
97     }
98 }
```

29:5 2 4 LF UTF-8 4 spaces

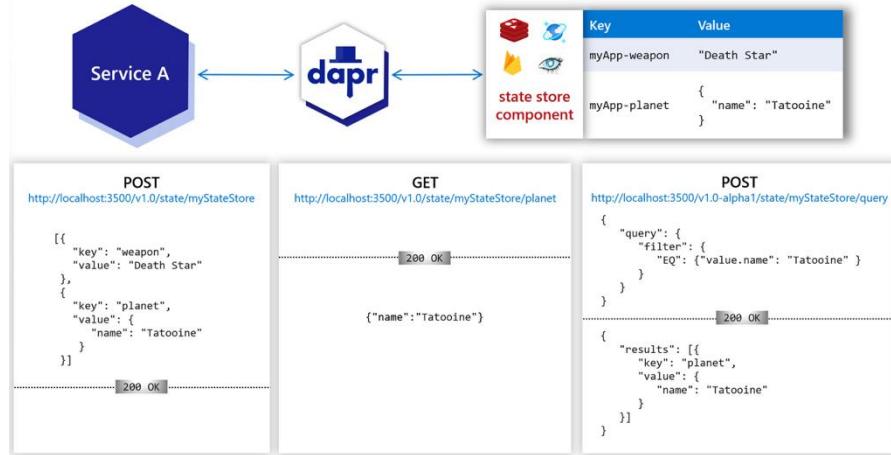
A close-up, low-angle shot of several rowers in a racing shell. Their hands are gripping yellow oars, which are angled downwards. The water is visible at the bottom, creating a sense of motion. The rowers are wearing blue and red athletic gear.

Dapr Building Blocks

# State Management

4tecture®  
empower your software solutions

# State Management



- Configurable state store behavior (default eventually consistent, last-write-wins concurrency pattern)
- Optimistic concurrency with ETag
- Time to live (TTL)
- State encryption
- Querying state

# State Stores

## Generic

Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version	Since runtime version
Aerospike	✓	□	✓	□	□	□	Alpha	v1	1.0
Apache Cassandra	✓	□	□	✓	□	□	Stable	v1	1.9
CockroachDB	✓	✓	✓	✓	✓	✓	Stable	v1	1.10
Couchbase	✓	□	✓	□	□	□	Alpha	v1	1.0
etcd	✓	✓	✓	✓	✓	□	Beta	v2	1.12
Hashicorp Consul	✓	□	□	□	□	□	Alpha	v1	1.0
Hazelcast	✓	□	□	□	□	□	Alpha	v1	1.0
In-memory	✓	✓	✓	✓	✓	□	Stable	v1	1.9
JetStream KV	✓	□	□	□	□	□	Alpha	v1	1.7
Memcached	✓	□	□	✓	□	□	Stable	v1	1.9
MongoDB	✓	✓	✓	✓	✓	✓	Stable	v1	1.0
MySQL & MariaDB	✓	✓	✓	✓	✓	□	Stable	v1	1.10
Oracle Database	✓	✓	✓	✓	✓	□	Beta	v1	1.7
PostgreSQL v1	✓	✓	✓	✓	✓	✓	Stable	v1	1.0
PostgreSQL v2	✓	✓	✓	✓	✓	□	Stable	v2	1.13
Redis	✓	✓	✓	✓	✓	✓	Stable	v1	1.0
RethinkDB	✓	□	□	□	□	□	Beta	v1	1.9
SQLite	✓	✓	✓	✓	✓	□	Stable	v1	1.11
Zookeeper	✓	□	✓	□	□	□	Alpha	v1	1.0

## Amazon Web Services (AWS)

Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version	Since runtime version
AWS DynamoDB	✓	✓	✓	✓	✓	□	Stable	v1	1.10

## Cloudflare

Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version	Since runtime version
Cloudflare Workers KV	✓	□	□	✓	□	□	Beta	v1	1.10

## Google Cloud Platform (GCP)

Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version	Since runtime version
GCP Firestore	✓	□	□	□	□	□	Stable	v1	1.11

## Microsoft Azure

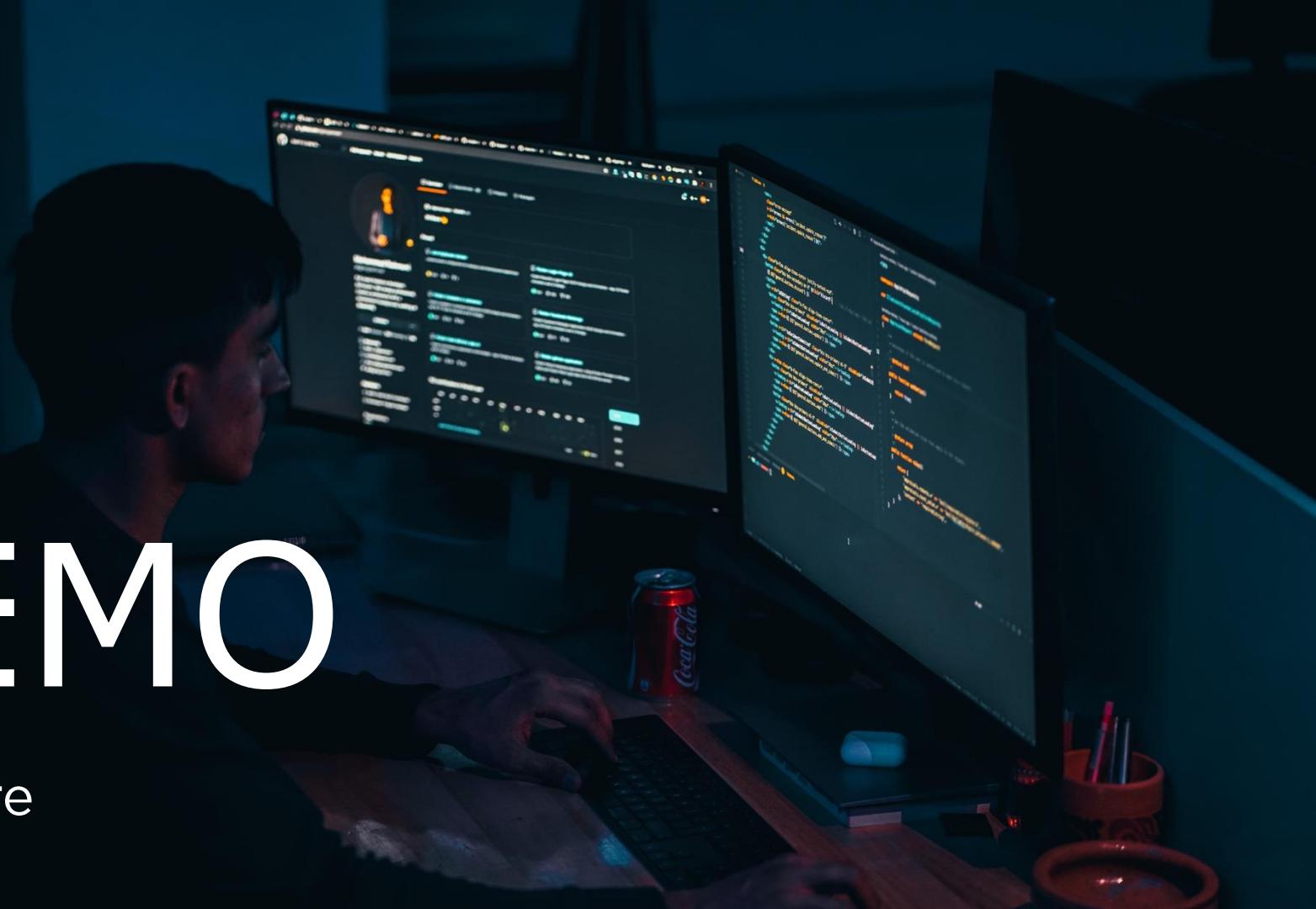
Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version	Since runtime version
Azure Blob Storage	✓	□	✓	□	□	□	Stable	v2	1.13
Azure Cosmos DB	✓	✓	✓	✓	✓	✓	Stable	v1	1.0
Azure Table Storage	✓	□	✓	□	□	□	Stable	v1	1.9
Microsoft SQL Server	✓	✓	✓	✓	✓	✓	Stable	v1	1.5

## Oracle Cloud

Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version	Since runtime version
Autonomous Database (ATP and ADW)	✓	✓	✓	✓	✓	✓	Alpha	v1	1.7
Object Storage	✓	□	✓	✓	✓	□	Alpha	v1	1.6

# DEMO

State Store



Solution Files

FastFoodDelivery - 26 projects

- Common - 3 projects
  - FastFood.Common
  - FastFood.Common.Unit.Tests
  - FastFood.Observability.Common
- Services - 22 projects
  - finance - 4 projects
  - frontendcustomerorderstatus - 2 projects
  - frontendkitchenmonitor - 2 projects
  - frontendserviceipos - 2 projects
  - kitchen - 4 projects
  - order - 8 projects
    - OrderService
      - Dependencies
      - Properties
      - chart
      - Controllers
        - DeadLetterHandlerController.cs
        - OrderActorController.cs
        - OrderController.cs
        - OrderStateController.cs
        - OrderUpdateEventHandlerController.cs
        - OrderWorkflowController.cs
        - SecretDemoController.cs
    - Observability
    - Services
      - IDelegateEventRouter.cs
      - IOrderProcessingService.cs
      - OrderEventRouter.cs
      - OrderEventRoutingTarget.cs
      - OrderProcessingServiceActor.cs
      - OrderProcessingServiceState.cs
      - OrderProcessingServiceWorkflow.cs
    - Storage
    - Workflows
      - appsettings.json
      - appsettings.Development.json
  - Dockerfile
  - Program.cs
  - Start-Selfhosted.ps1
  - OrderService.Actors
  - OrderService.Actors.Unit.Tests
  - OrderService.Common
  - OrderService.Common.Unit.Tests
  - OrderService.Models
  - OrderService.Models.Unit.Tests
  - OrderService.Unit.Tests
  - Solution Items
    - BuildAndPushDockerImages.ps1
    - docker-compose.yml
  - SystemTests - 1 project
    - FastFoodDelivery.System.Tests
  - Scratches and Consoles

Program.cs KitchenService.cs OrderProcessingServiceState.cs

```
public partial class OrderProcessingServiceState : IOrderProcessingServiceState
{
    public async Task AssignDeliveryAddress(Guid orderid, Address address)
    {
        throw new InvalidOperationException("Order is not in the correct state to assign a delivery address");
    }

    public async Task AddItem(Guid orderid, OrderItem item)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var order = await _daprClient.GetStateAsync<Order>(FastFoodConstants.StateStoreName, key: GetStateId(orderid));
        if (order.State == OrderState.Creating)
        {
            order.Items?.Add(item);

            await _daprClient.SaveStateAsync(FastFoodConstants.StateStoreName, key: GetStateId(order.Id), order);
            await _daprClient.PublishEventAsync(FastFoodConstants.PubSubName, topicName: FastFoodConstants.EventNames.OrderUpdated, data: order.ToDto());
        }
        else
        {
            throw new InvalidOperationException("Order is not in the correct state to add an item");
        }
    }

    public async Task RemoveItem(Guid orderid, Guid itemId)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var order = await _daprClient.GetStateAsync<Order>(FastFoodConstants.StateStoreName, key: GetStateId(orderid));
        if (order.State == OrderState.Creating)
        {
            var itemToRemove = order.Items?.FirstOrDefault(i => i.Id == itemId);
            if (itemToRemove != null && order.Items != null)
            {
                order.Items.Remove(itemToRemove);
            }
        }
    }
}
```

0+1 usages Marc Müller

OrderProcessingServiceState.cs

FastFoodDelivery Services order OrderService Services OrderProcessingServiceState.cs OrderProcessingServiceState

128.6 1 LF UTF-8 4 spaces

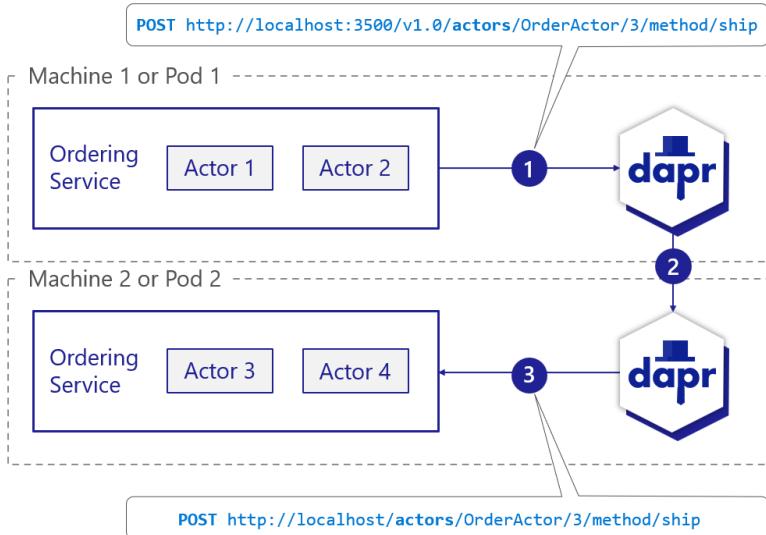
A close-up, low-angle shot of several rowers in a boat. Their hands are gripping yellow and black oars, which are angled downwards. The water is visible at the bottom, creating a sense of motion. The rowers are wearing blue athletic gear.

Dapr Building Blocks

# Virtual Actors

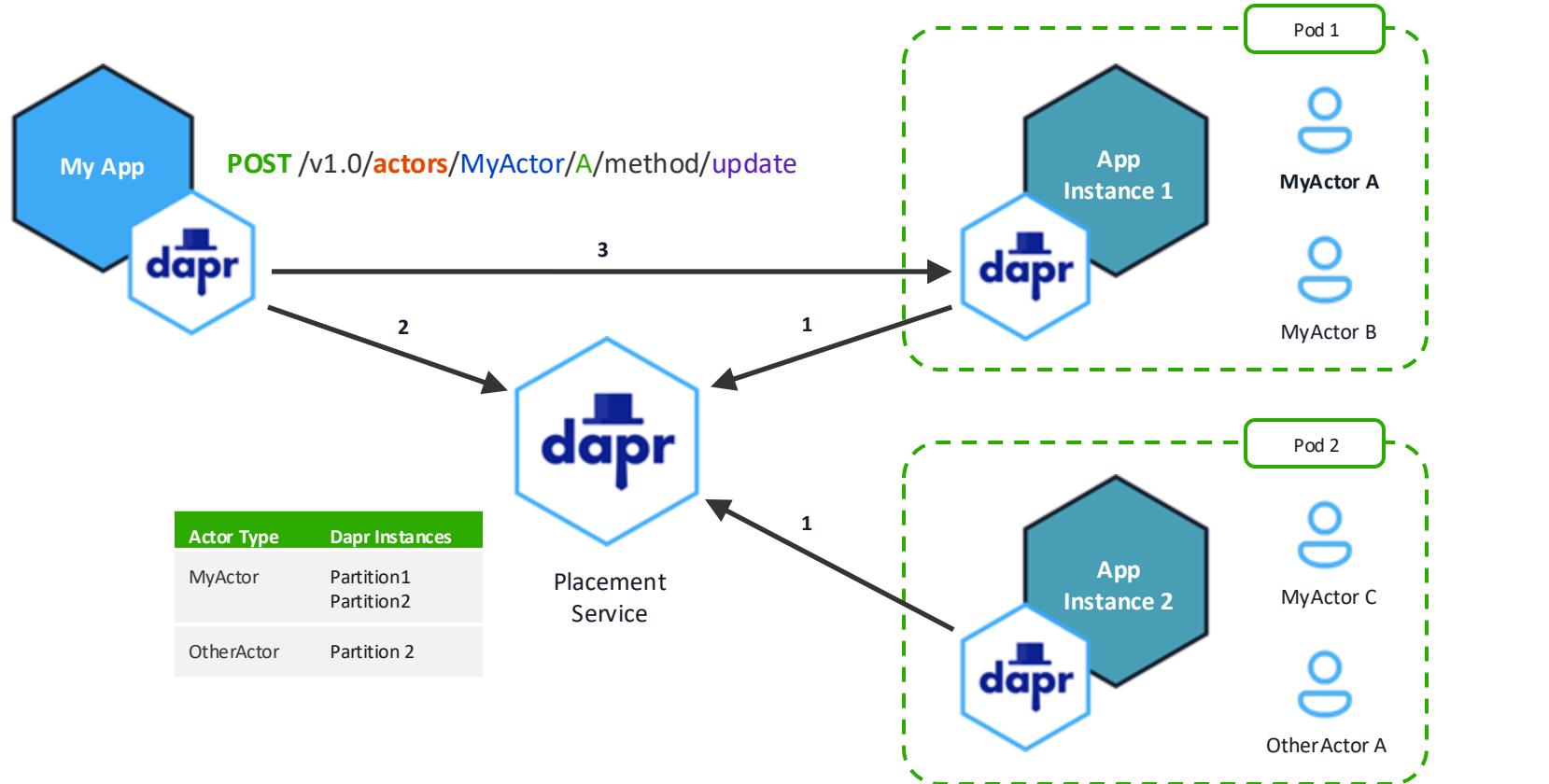
4tecture®  
empower your software solutions

# Actors



- Virtual Actor pattern
- Stateful, long running objects with identity
- Encapsulate state and behavior within a distributed system
- Actor state store
- Actor timers and reminders

# Actor placement



# DEMO

Virtual Actors



The screenshot shows a Visual Studio IDE interface with the following details:

- Solution Explorer:** On the left, it displays the solution structure for "FastFoodDelivery". The "order" project is selected. Inside "order", the "OrderService" folder contains several files: Program.cs, OrderService.cs, OrderProcessingServiceState.cs, and OrderActor.cs. The OrderActor.cs file is currently open.
- Code Editor:** The main window shows the code for the OrderActor class. A red box highlights the class definition:

```
namespace OrderPlacement_Actors;  
public class OrderActor : Actor, IOrderActor, IRemindable
```
- Toolbars and Status Bar:** The top bar includes standard icons for file operations. The status bar at the bottom right shows the time as 13:14 and the file path as "FastFoodDelivery > Services > order > OrderService.Actors > Actors > OrderActor.cs".

```
FastFoodDelivery > Services > order > OrderService.Actors > Program.cs > <top-level-entry-point>
```

```
18     .UseJsonSerializationOptions(new JsonSerializerOptions().ConfigureJsonSerializerOptions()));
19
20     builder.Services.AddActors(configure: options => { options.Actors.RegisterActor<OrderActor>(); });
21
22
23     builder.Services.AddControllers()
24         .AddJsonOptions(options => { options.JsonSerializerOptions.ConfigureJsonSerializerOptions(); })
25         .AddDapr();
26
27 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
28 builder.Services.AddEndpointsApiExplorer();
29 builder.Services.AddSwaggerGen();
30 builder.Services.AddHealthChecks();
31
32 var app: WebApplication = builder.Build();
33
34 // Configure the HTTP request pipeline.
35 if (app.Environment.IsDevelopment())
36 {
37     app.UseSwagger();
38     app.UseSwaggerUI();
39 }
40
41 app.UseCloudEvents();
42
43 app.UseObservability(observabilityOptions);
44
45 app.MapControllers();
46 app.MapSubscribeHandler();
47
48 app.MapActorsHandlers();
49
50 app.MapHealthChecks(pattern: "/healthz");
51
52 app.Run();
```

Solution Files

OrderService/Program.cs KitchenService.cs OrderProcessingServiceState.cs OrderActor.cs OrderService.Actors/Program.cs

```
public class OrderActor : Actor, IOrderActor, IRemindable
{
    public async Task<Order> AssignDeliveryAddress(Address address)
    {
        activity?.SetStatus(ActivityStatusCode.Error, description: "Order is not in the correct state to assign a delivery address");
        throw new InvalidOperationException("Order is not in the correct state to assign a delivery address");
    }

    public 0+1 usages  Marc Müller
    public async Task<Order> AddItem(OrderItem item)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var order = await StateManager.GetStateAsync<Order>("order");
        if (order.State == OrderState.Creating)
        {
            if (order.Items == null || order.Items.Count == 0)
            {
                order.Items = new List<OrderItem>();
            }
            order.Items?.Add(item);

            await StateManager.SetStateAsync("order", order);
            await _publisherClient.PublishEventAsync(FastFoodConstants.PubSubName, topicName: FastFoodConstants.EventNames.OrderUpdated, data: order.ToDto());
        }
        return order;
    }

    activity?.SetStatus(ActivityStatusCode.Error, description: "Order is not in the correct state to add an item");
    throw new InvalidOperationException("Order is not in the correct state to add an item");
}

public 0+1 usages  Marc Müller
public async Task<Order> RemoveItem(Guid itemId)
{
    using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
}
```

FastFoodDelivery > Services > order > OrderService.Actors > Actors > OrderActor.cs > OrderActor

13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122

Solution Files

OrderService/Program.cs KitchenService.cs OrderProcessingServiceState.cs OrderActor.cs OrderProcessingServiceActor.cs OrderService.Actors/Program.cs

```
public class OrderProcessingServiceActor : IOrderProcessingServiceActor
{
    public async Task<Order> GetOrder(Guid orderid)
    {
        ...
    }

    [0+1 usages  ↳ Marc Müller]
    public async Task AddItem(Guid orderid, OrderItem item)
    {
        using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
        var actorId = new ActorId(orderid.ToString());
        var proxy = ActorProxy.Create<IOrderActor>(actorId, actorType: OrderActorName); ①
        var orderResult :Order = await proxy.AddItem(item); ②
    }
}

[0+1 usages  ↳ Marc Müller]
public async Task AssignDeliveryAddress(Guid orderid, Address address)
{
    using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
    var actorId = new ActorId(orderid.ToString());
    var proxy = ActorProxy.Create<IOrderActor>(actorId, actorType: OrderActorName);
    var orderResult :Order = await proxy.AssignDeliveryAddress(address);
}

[0+1 usages  ↳ Marc Müller]
public async Task AssignInvoiceAddress(Guid orderid, Address address)
{
    using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
    var actorId = new ActorId(orderid.ToString());
    var proxy = ActorProxy.Create<IOrderActor>(actorId, actorType: OrderActorName);
    var orderResult :Order = await proxy.AssignInvoiceAddress(address);
}
```

OrderService: http://localhost:5000

118:1 1 LF UTF-8 4 spaces

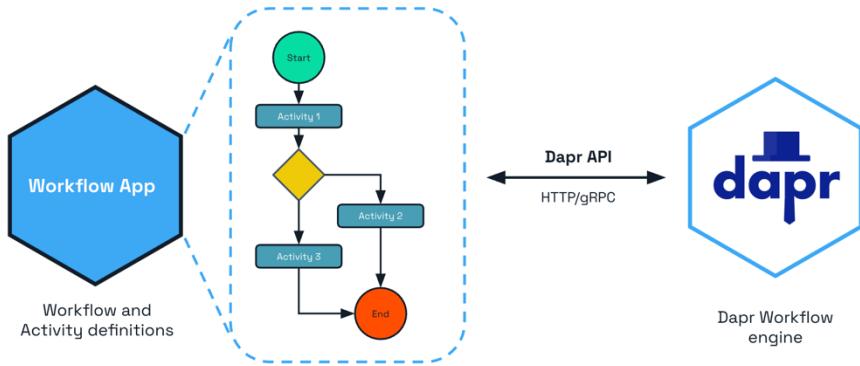
A close-up, low-angle shot of several rowers in a racing shell. Their hands are gripping yellow and black oars, which are angled downwards. The water is visible at the bottom, creating a sense of motion. The rowers are wearing blue and red athletic gear.

Dapr Building Blocks

# Workflows

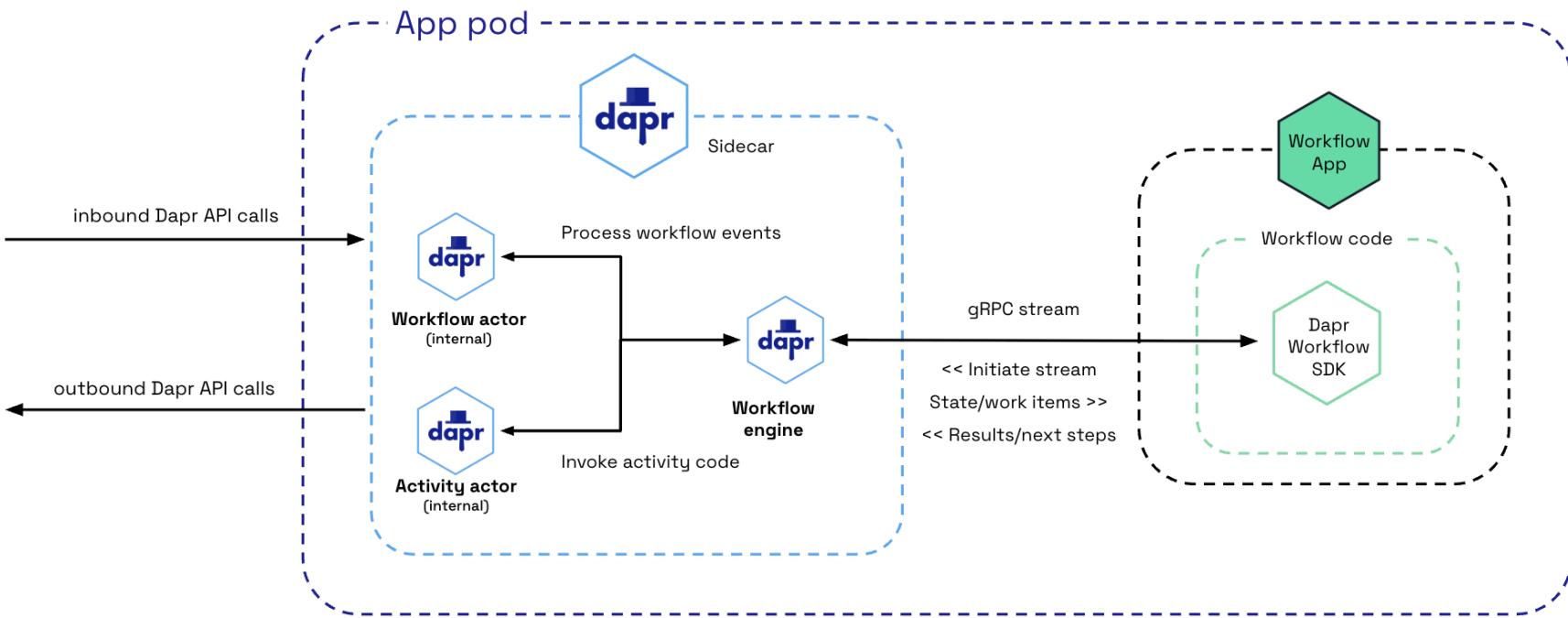
4tecture®  
empower your software solutions

# Workflows



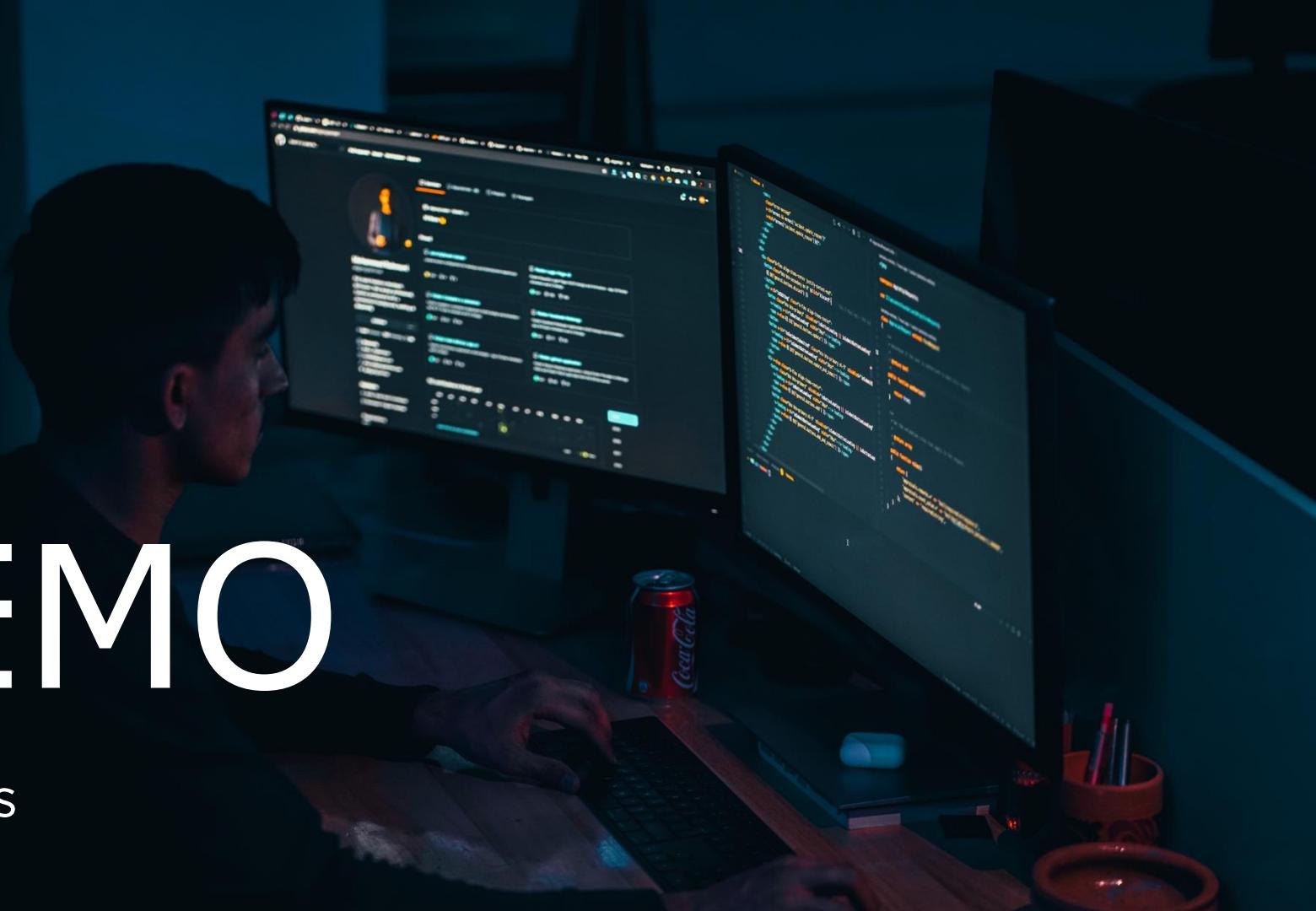
- Built-In workflow runtime
- Activities execute custom logic
- External event waiters
- Child Workflows
- Durable timers

# Workflows the hood



# DEMO

Workflows



FastFoodDelivery main

OrderService/Program.cs KitchenService.cs OrderProcessingServiceState.cs OrderActor.cs OrderProcessingServiceActor.cs OrderProcessingWorkflow.cs OrderService.Actors/Program.cs

FastFoodDelivery • 26 projects  
Common • 3 projects  
FastFood.Common  
FastFood.Common.Unit.Tests  
FastFood.Observability.Common  
Services • 22 projects  
finance • 4 projects  
frontendcustomerorderstat • 2 projects  
frontentkitchenservice • 2 projects  
frontendservicepos • 2 projects  
kitchen • 4 projects  
order • 8 projects  
OrderService  
Dependencies  
Properties  
chart  
Controllers  
Observability  
Services  
Storage  
Workflows  
Activities  
AddItemActivity.cs  
AssignCustomerActivity.cs  
AssignDeliveryAddressActivity.cs  
AssignInvoiceAddressActivity.cs  
ConfirmOrderActivity.cs  
ConfirmPaymentActivity.cs  
CreateOrderActivity.cs  
ItemFinishedActivity.cs  
OrderServedActivity.cs  
RemoveItemActivity.cs  
StartProcessingActivity.cs  
Events  
Extensions  
OrderProcessingWorkflow.cs  
appsettings.json  
appsettings.Development.json  
Dockerfile  
Program.cs  
Start-Selfhosted.ps1  
OrderService.Actors  
OrderService.Actors.Unit.Tests  
OrderService.Common  
OrderService.Common.Unit.Tests  
OrderService.Models  
OrderService.Models.Unit.Tests  
OrderService.Unit.Tests  
BuildAndPushDockerImages.ps1  
docker-compose.yml  
SystemTests 1 project  
FastFoodDelivery.System.Tests  
Scratches and Console

16:1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

```
public class OrderProcessingWorkflow : Workflow<Guid, Order>
{
    public override async Task<Order> RunAsync(WorkflowContext context, Guid orderId)
    {
        Order order = null;
        try
        {
            // Create order
            order = await context.CallActivityAsync<Order>(nameof(CreateOrderActivity), orderId);

            while (order.State == OrderState.Creating)
            {
                var createOrderWrapperEvent = await context.WaitForExternalEventAsync<CreateOrderWrapperEvent>(CreateOrderWrapperEvent.Name);
                if (createOrderWrapperEvent.SubEventType == CreateOrderWrapperEvent.CreateOrderSubEventType.AssignCustomerEvent)
                {
                    order = await context.CallActivityAsync<Order>(nameof(AssignCustomerActivity), createOrderWrapperEvent.AssignCustomerEvent);
                }
                else if (createOrderWrapperEvent.SubEventType == CreateOrderWrapperEvent.CreateOrderSubEventType.AssignInvoiceAddressEvent)
                {
                    order = await context.CallActivityAsync<Order>(nameof(AssignInvoiceAddressActivity), createOrderWrapperEvent.AssignInvoiceAddressEvent);
                }
                else if (createOrderWrapperEvent.SubEventType == CreateOrderWrapperEvent.CreateOrderSubEventType.AssignDeliveryAddressEvent)
                {
                    order = await context.CallActivityAsync<Order>(nameof(AssignDeliveryAddressActivity), createOrderWrapperEvent.AssignDeliveryAddressEvent);
                }
                else if (createOrderWrapperEvent.SubEventType == CreateOrderWrapperEvent.CreateOrderSubEventType.AddItemEvent)
                {
                    order = await context.CallActivityAsync<Order>(nameof(AddItemActivity), createOrderWrapperEvent.AddItemEvent);
                }
                else if (createOrderWrapperEvent.SubEventType == CreateOrderWrapperEvent.CreateOrderSubEventType.RemoveItemEvent)
                {
                    order = await context.CallActivityAsync<Order>(nameof(RemoveItemActivity), createOrderWrapperEvent.RemoveItemEvent);
                }
                else if (createOrderWrapperEvent.SubEventType == CreateOrderWrapperEvent.CreateOrderSubEventType.ConfirmOrderEvent)
                {
                    order = await context.CallActivityAsync<Order>(nameof(ConfirmOrderActivity), createOrderWrapperEvent.ConfirmOrderEvent);
                }
                else
                {
                    context.SetCustomStatus("No event received");
                }
            }
        }
    }
}
```

FastFoodDelivery > Services > order > OrderService > Workflows > Activities > AddItemActivity.cs > AddItemActivity

```
namespace OrderPlacement.Workflows;

public partial class AddItemActivity : WorkflowActivity<AddItemEvent, Order>
{
    private readonly IOrderStorage _orderStorage;
    private readonly ILogger<AddItemActivity> _logger;
    private readonly DaprClient _daprClient;

    public AddItemActivity(IOrderStorage orderStorage, DaprClient daprClient, ILogger<AddItemActivity> logger)
    {
        _orderStorage = orderStorage;
        _daprClient = daprClient;
        _logger = logger;
    }

    public override async Task<Order> RunAsync(WorkflowActivityContext context, AddItemEvent input)
    {
        var order = await _orderStorage.GetOrderById(input.OrderId);
        if (order != null && order.State == OrderState.Creating)
        {
            var existingItem = order.Items?.FirstOrDefault(i => i.Id == input.Item.Id);
            if (existingItem != null)
            {
                // item already exists, idempotent operation
                return order;
            }
            else
            {
                order.Items.Add(input.Item);
                await _orderStorage.UpdateOrder(order);
                await _daprClient.PublishEventAsync(FastFoodConstants.PubSubName, topicName: FastFoodConstants.EventNames.OrderUpdated, data: order.ToDto());
                LogAddedItem(context.InstanceId, order.Id, input.Item.Id);
            }
        }
        else
        {
            LogAddedItemFailed(context.InstanceId, input.OrderId, input.Item.Id);
        }
    }
}
```

11:4 usages & Marc Müller

12: Marc Müller

13: Marc Müller

14: Marc Müller

15: Marc Müller

16: Marc Müller

17: Marc Müller

18: Marc Müller

19: Marc Müller

20: Marc Müller

21: Marc Müller

22: Marc Müller

23: Marc Müller

24: Marc Müller

25: Marc Müller

26: Marc Müller

27: Marc Müller

28: Marc Müller

29: Marc Müller

30: Marc Müller

31: Marc Müller

32: Marc Müller

33: Marc Müller

34: Marc Müller

35: Marc Müller

36: Marc Müller

37: Marc Müller

38: Marc Müller

39: Marc Müller

40: Marc Müller

41: Marc Müller

42: Marc Müller

43: Marc Müller

44: Marc Müller

45: Marc Müller

46: Marc Müller

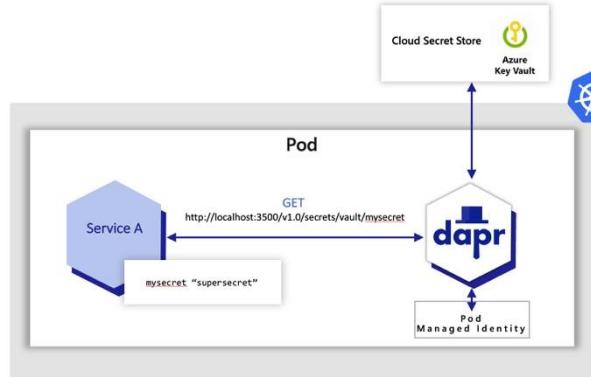
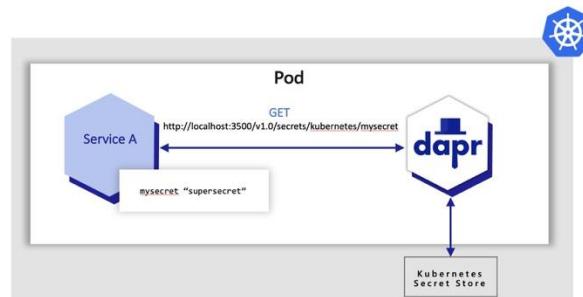
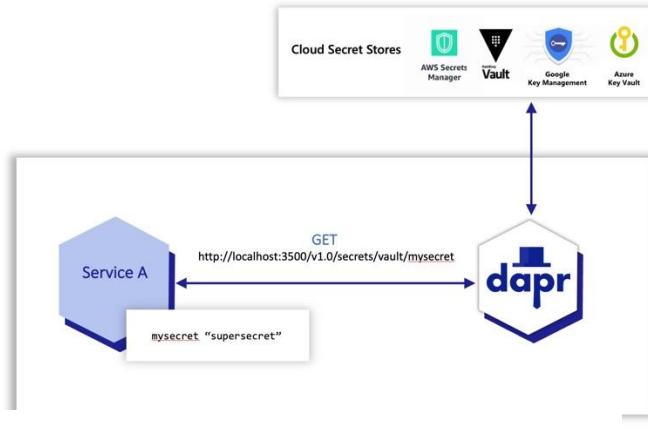
A close-up, slightly blurred photograph of several rowers in a boat. They are wearing blue and red athletic gear. Their oars are extended, pointing towards the bottom right of the frame. The water is visible at the bottom.

Dapr Building Blocks

# Secret Management

4tecture®  
empower your software solutions

# Secret Management



- Access secret stores through generic Dapr API
- Secret scoping (limit access)

A close-up, low-angle shot of several rowers in a racing shell. Their hands are gripping yellow and black oars, which are angled downwards. The water is visible at the bottom, creating a sense of motion. The rowers are wearing blue and red athletic gear.

Dapr Building Blocks

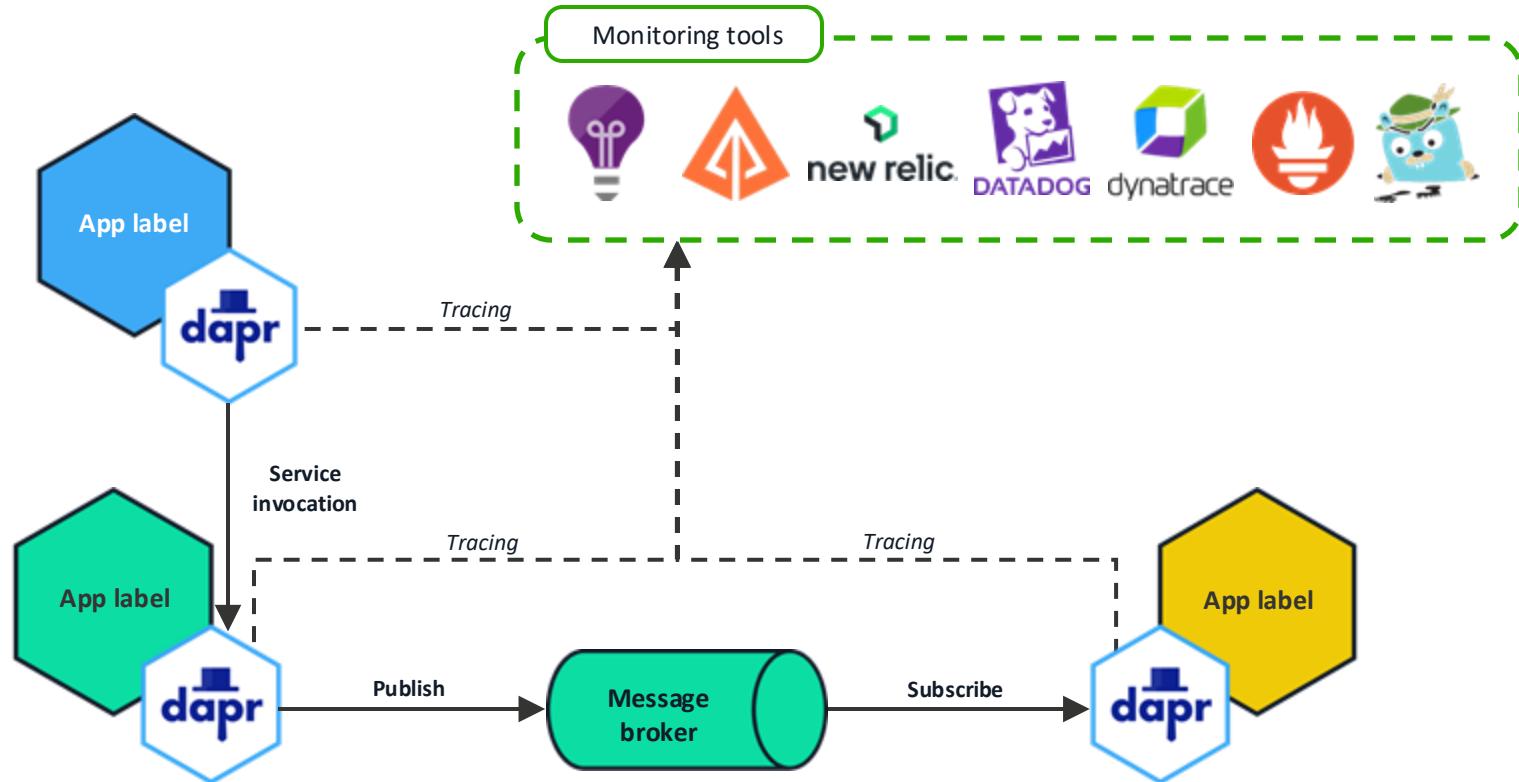
# Observability

4tecture®  
empower your software solutions

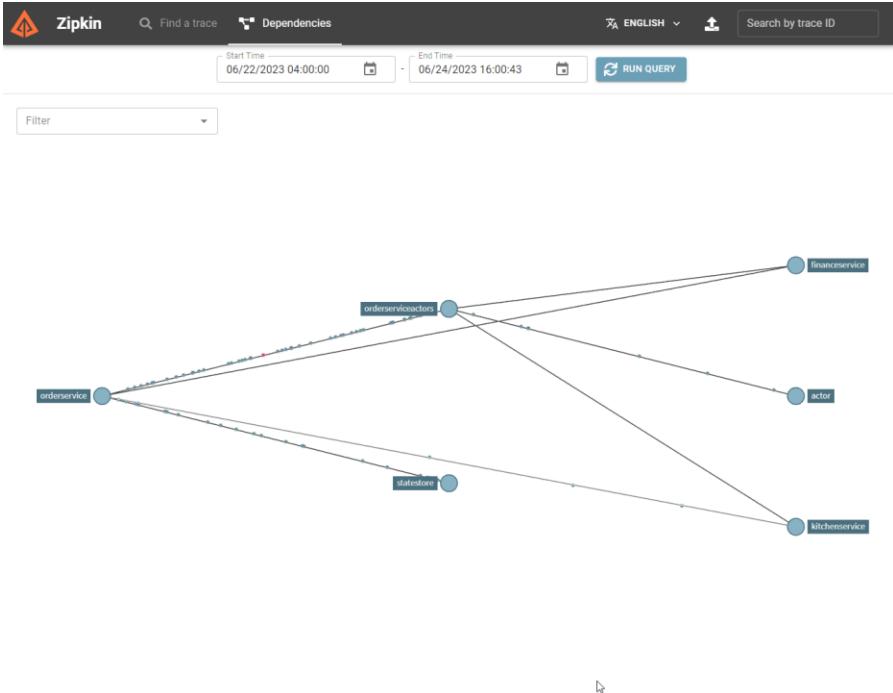
# Distributed tracing



Observability



# Observability



- Distributed tracing
- Open Telemetry (OTEL) and Zipkin protocols
- Used with service invocation and pub/sub APIs
  
- Sidecar health
- App health checks
  - Unsubscribing Pub/Sub
  - Stop input binings
  - Short-circuiting all service-invocation requests

A close-up, low-angle shot of several rowers in a boat. They are wearing blue athletic gear and red and white striped athletic gear. Each rower is holding a oar with a yellow handle. The water is visible at the bottom, and the background is blurred.

Dapr Building Blocks

# Resiliency

4tecture®  
empower your software solutions

# Resiliency

Resiliency patterns can be applied across Dapr APIs:

- Retries
- Timeouts
- Circuit breakers

Declarative and decoupled from application code.

Available across all component types, service invocation, and actors.

```
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: myresiliency
scopes:
  - order-processor

spec:
  policies:
    retries:
      retryForever:
        policy: constant
        duration: 5s
        maxRetries: -1

  circuitBreakers:
    simpleCB:
      maxRequests: 1
      timeout: 5s
      trip: consecutiveFailures >= 5

  targets:
    components:
      statestore:
        outbound:
          retry: retryForever
          circuitBreaker: simpleCB
```



.NET Microservices with Dapr

# Conclusion

# Pros / Cons

## Advantages

- Develop faster
- Best practices
- Portability
- Focus on your logic

## Disadvantages

- Additional hop / network overhead
- Common API – less features

# Conclusion

- Suitable for most teams and applications
- Base your development on proven best practices
- Focus on your application, not on re-inventing the wheel
- Ideal, if portability is key (different environments / clouds, local, etc.)

A close-up, low-angle shot of several rowers in a boat. Their legs and the oars they are using are visible. The oars have yellow handles and black blades. The water is slightly choppy, creating small white caps. The rowers are wearing blue athletic gear.

.NET Microservices with Dapr

# Q & A

# Thank you for your attention!

If you have any questions do not hesitate to contact us:

4ecture GmbH  
Industriestrasse 25  
CH-8604 Volketswil  
[www.4ecture.ch](http://www.4ecture.ch)

Marc Müller  
Principal Consultant

[www.powerofdevops.com](http://www.powerofdevops.com)

A background photograph showing four hands of different skin tones and nail polish colors (red, white, pink) holding four interlocking wooden puzzle pieces. The pieces are light brown, dark brown, red, and green.

4 tecture<sup>©</sup>  
empower your software solutions