Developing ASP.NET Core Microservices with Dapr

Marc Müller
Principal Consultant

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

MVP Microsoft Most Valuable Professional

4tecture
empower your software solutions

About me:

Marc Müller
Principal Consultant
@muellermarc

Microsoft Most Valuable Professional

4tecture©
empower your software solutions

Our Products:

Multi-Tenant OpenID Connect Identity Provider

ProAuth

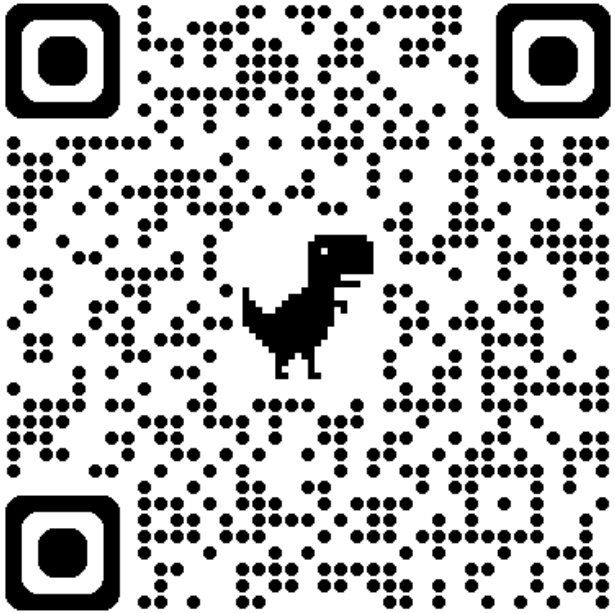
www.proauth.net

Enterprise Application Framework for .NET

ReaFx

www.reafx.net

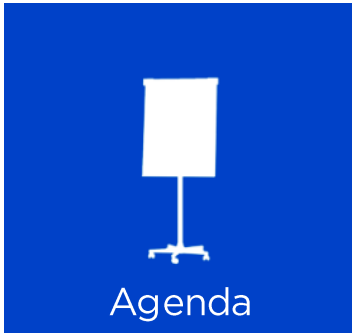
Slide Download



https://www.4tecture.ch/events/wearedevelopers25-dapr

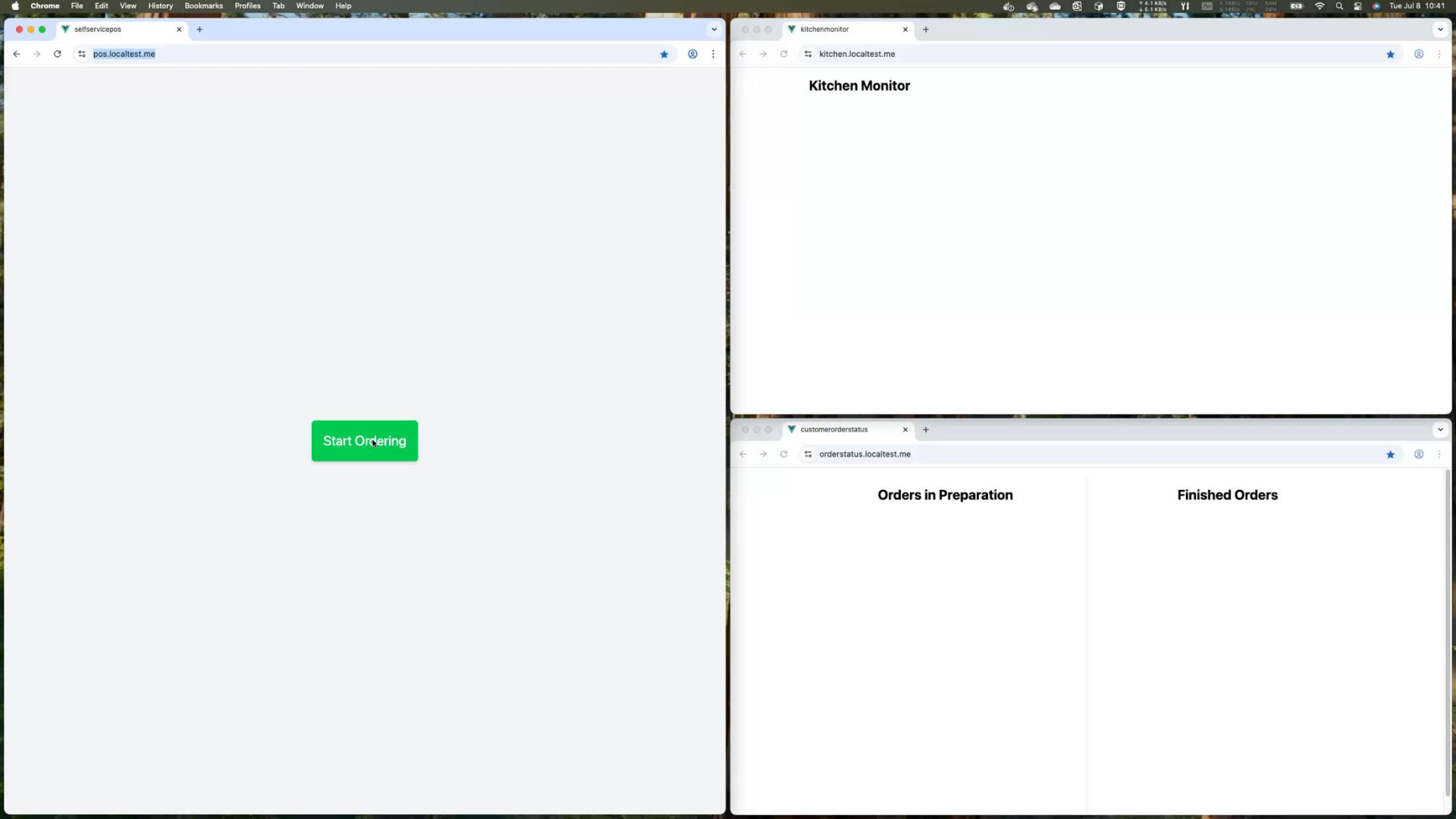
Agenda

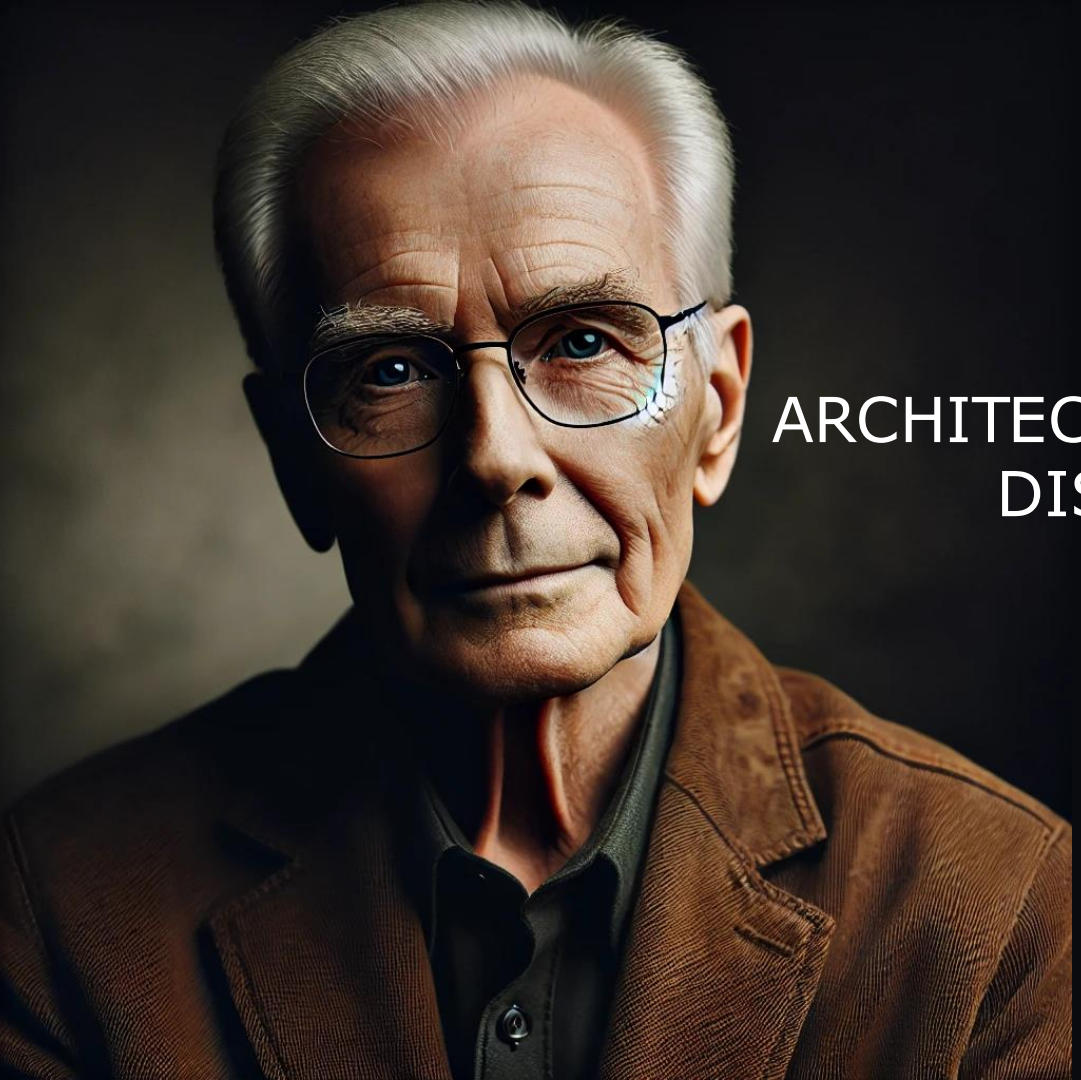
- Intro
- Dapr Basics
- Hands-On!
- Building Blocks
- Conclusion




Dapr
Intro
4tecture
empower your software solutions


DEMO
Fast Food Application


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue Jul 8 10:41
selfservicepos
pos.localtest.me
Start Ordering
kitchenmonitor
kitchen.localtest.me
Kitchen Monitor
customerorderstatus
orderstatus.localtest.me
Orders in Preparation
Finished Orders



ARCHITECTING AND IMPLEMENTING DISTRIBUTED APPLICATIONS IS NOT STRESFUL 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, Tracing

Decoupling

Events

Competing Consumer Pattern

Retry / Poisonous message detection

State Handling

Single state across multiple instances

Stateful services

Virtual Actors

Different Target Environments

Local dev environment vs. cloud environment

Different persistency services

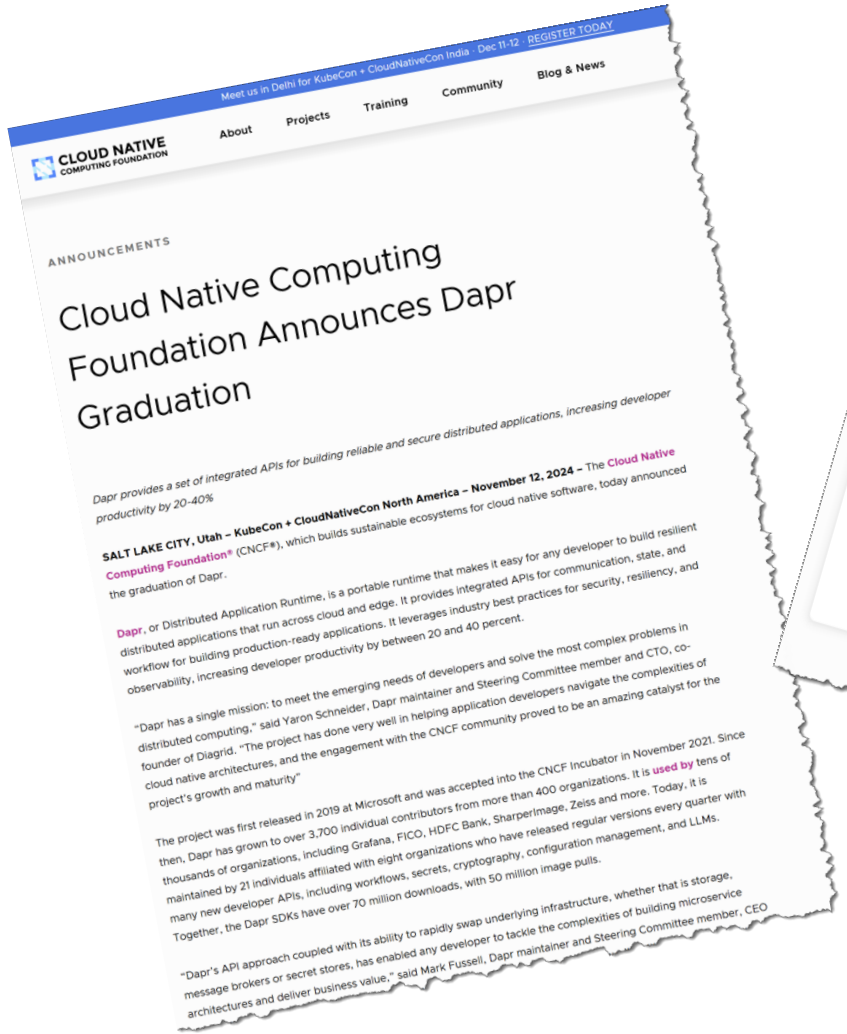
dapr.io

Portable, event-driven runtime

Build connected distributed applications faster

APIs for solving distributed application challenges

Cloud and Edge



Meet us in Delhi for KubeCon + CloudNativeCon India · Dec 11-12 · REGISTER TODAY

CLOUD NATIVE COMPUTING FOUNDATION

About Projects Training Community Blog & News

ANNOUNCEMENTS

Cloud Native Computing Foundation Announces Dapr Graduation

Dapr provides a set of integrated APIs for building reliable and secure distributed applications, increasing developer productivity by 20-40%

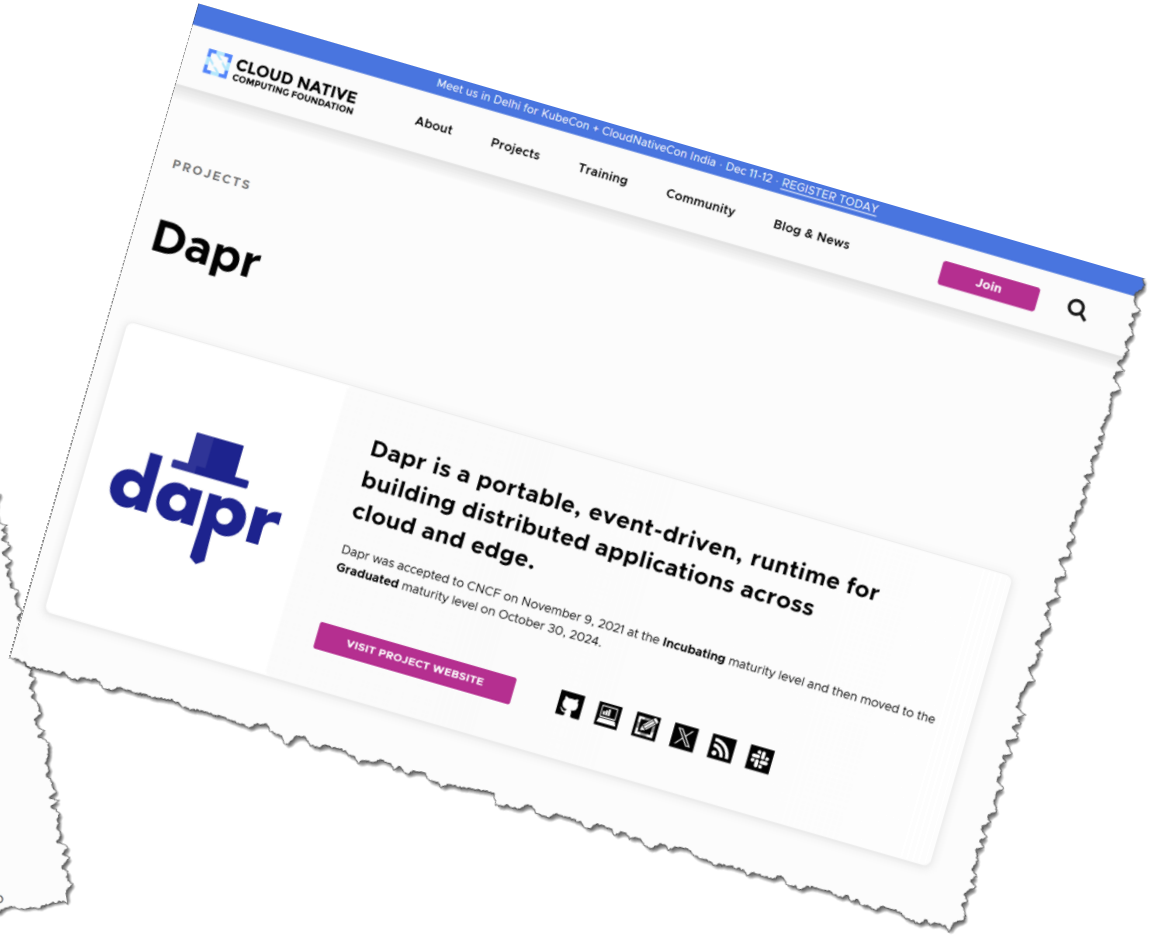
SALT LAKE CITY, Utah – KubeCon + CloudNativeCon North America – November 12, 2024 – The Cloud Native Computing Foundation® (CNCF®), which builds sustainable ecosystems for cloud native software, today announced the graduation of Dapr.

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.

"Dapr has a single mission: to meet the emerging needs of developers and solve the most complex problems in distributed computing," said Yaron Schneider, 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"

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.

"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



Meet us in Delhi for KubeCon + CloudNativeCon India · Dec 11-12 · REGISTER TODAY

CLOUD NATIVE COMPUTING FOUNDATION

About Projects Training Community Blog & News Join

PROJECTS

Dapr

dapr

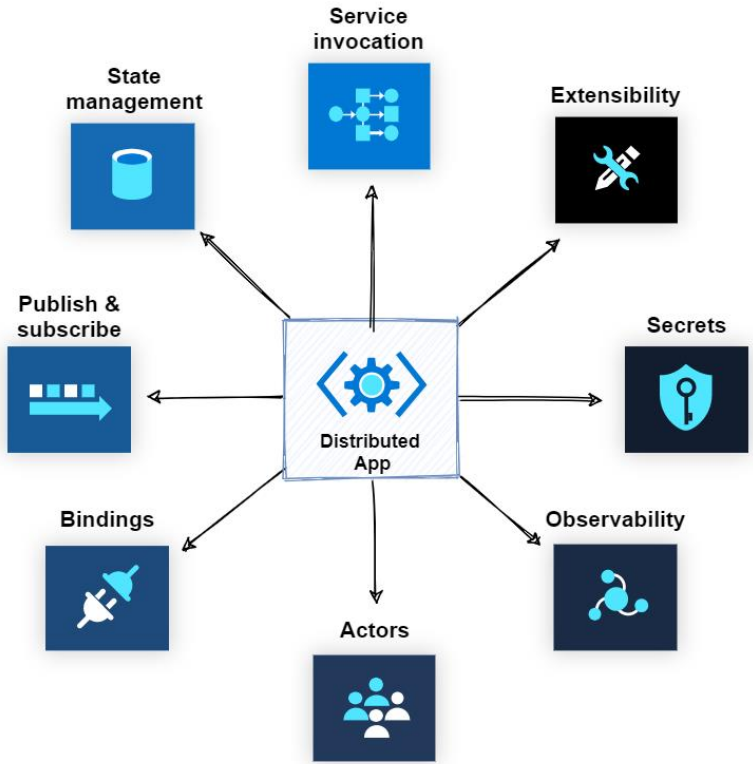
Dapr is a portable, event-driven, runtime for building distributed applications across cloud and edge.

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.

VISIT PROJECT WEBSITE

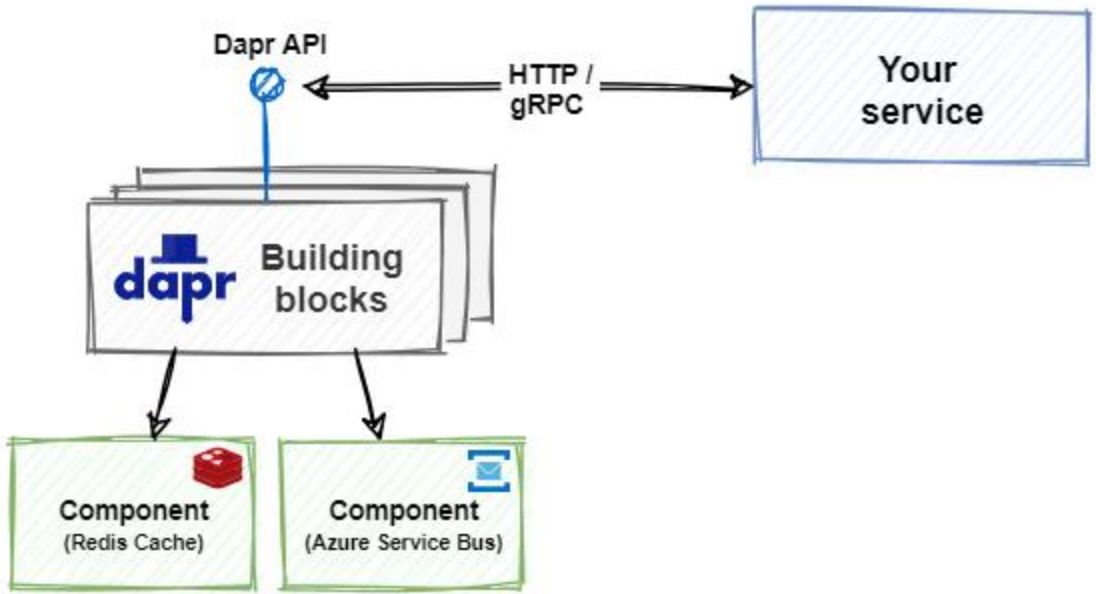
Source: https://www.cncf.io/announcements/2024/11/12/cloud-native-computing-foundation-announces-dapr-graduation/ and https://www.cncf.io/projects/dapr/

Dapr Building Blocks



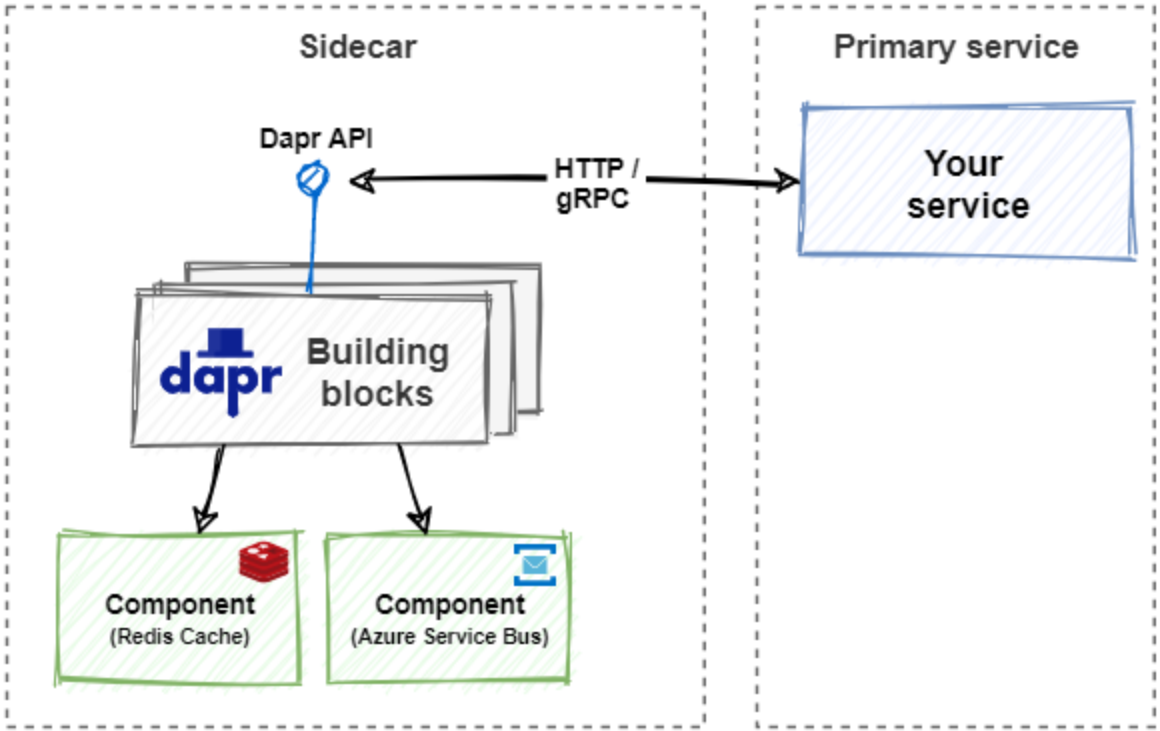
https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet

Dapr Building Blocks Abstraction



https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet

Dapr Sidecar Architecture



https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet


Dapr Basics
4tecture
empower your software solutions

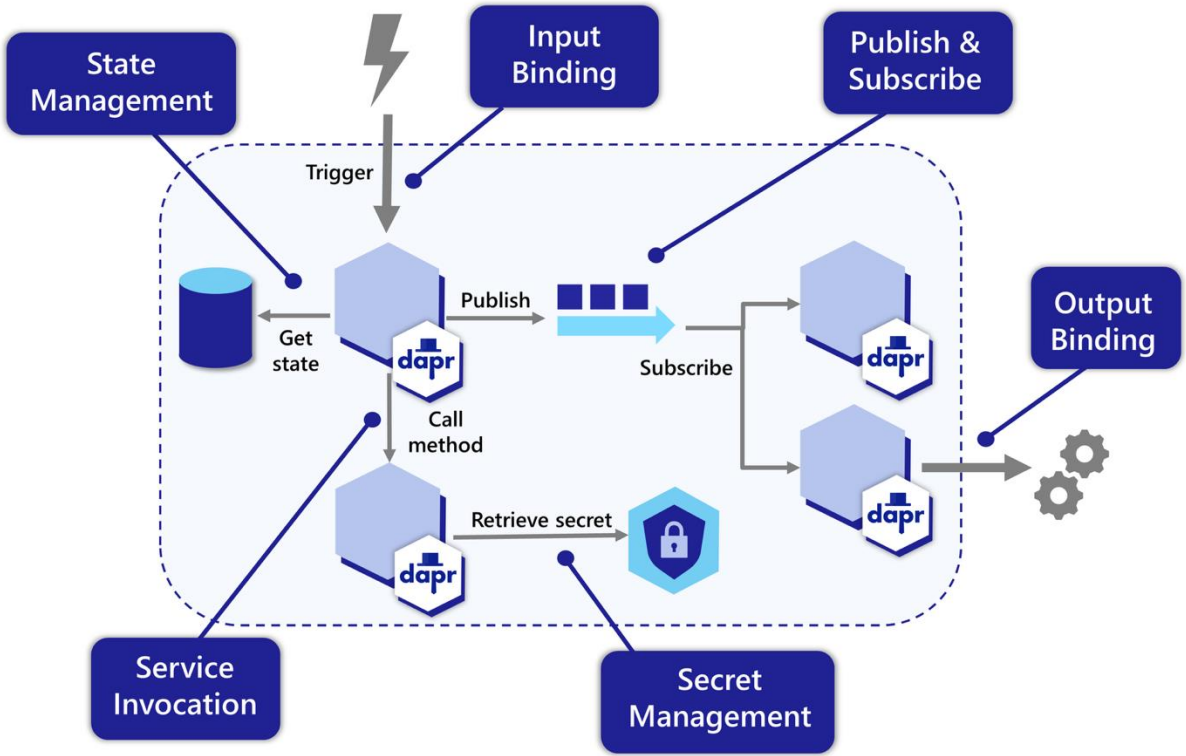
Dapr

- 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



https://dapr.io/

Single API - Multiple Components

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

daprClient.SaveStateAsync("statestore", "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

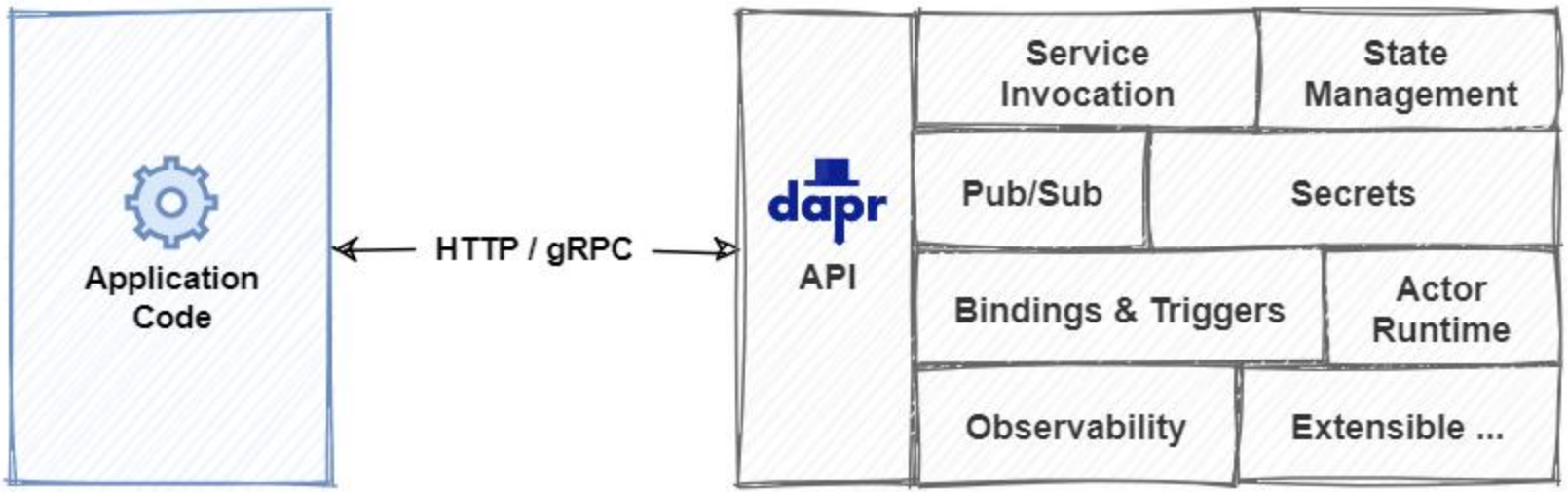
https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet


The power of side-cars!

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

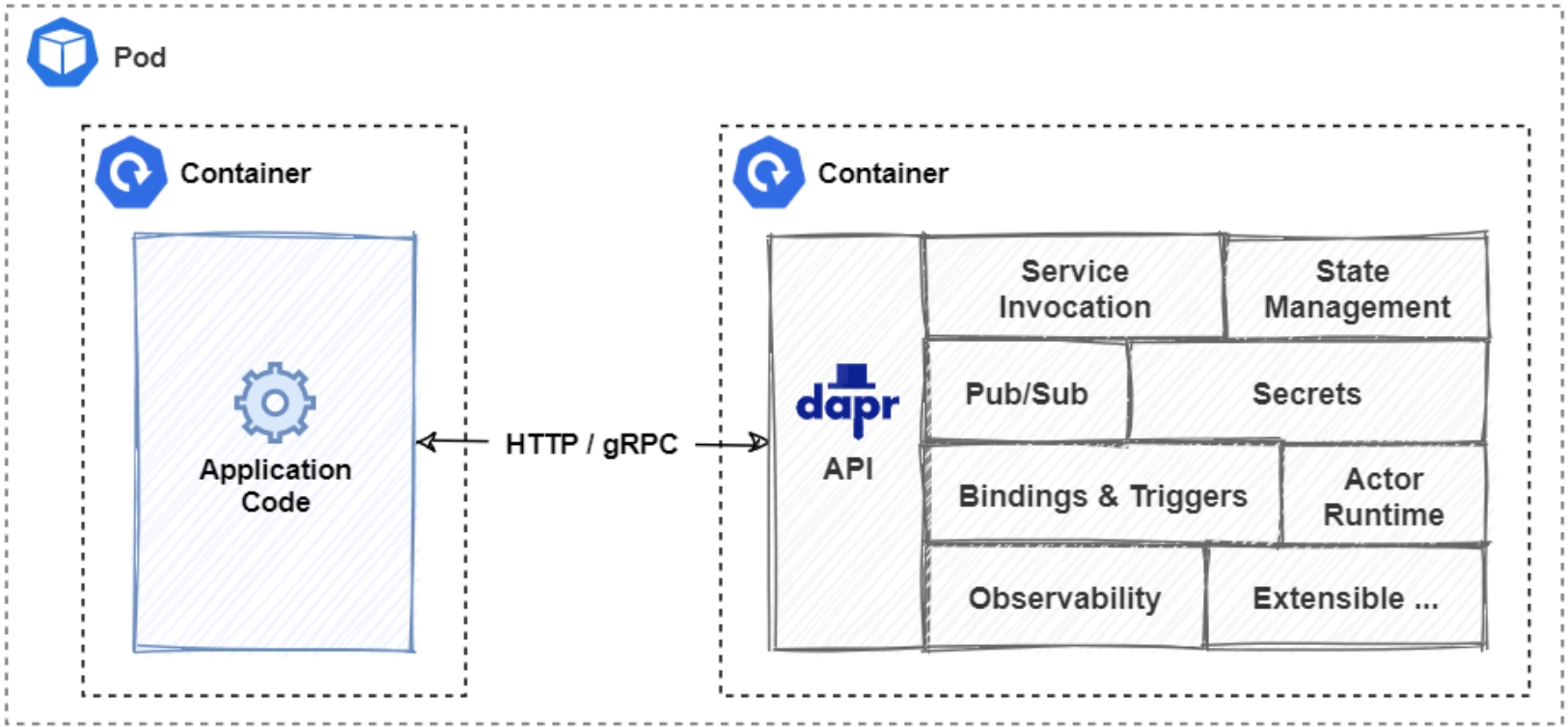
Speeds up application development by providing an integrated set of APIs for communication, state, and workflow.

Self-hosted Sidecar



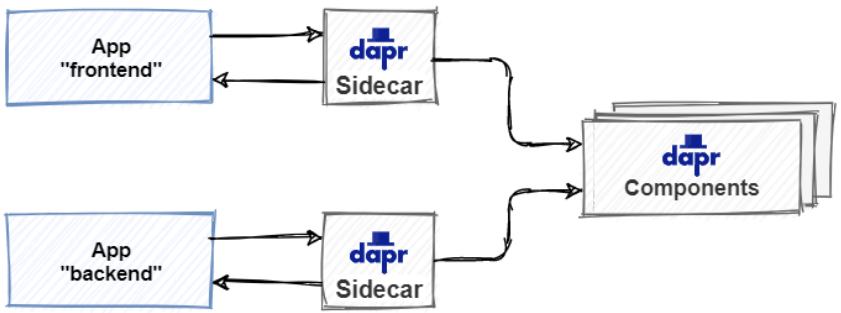
https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet

Kubernetes-hosted Sidecar



https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet

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

https://learn.microsoft.com/en-us/dotnet/architecture/dapr-for-net-developers/dapr-at-20000-feet

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	✓		✓	

https://docs.dapr.io/developing-applications/sdks/

Runtimes

Self-Hosted

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

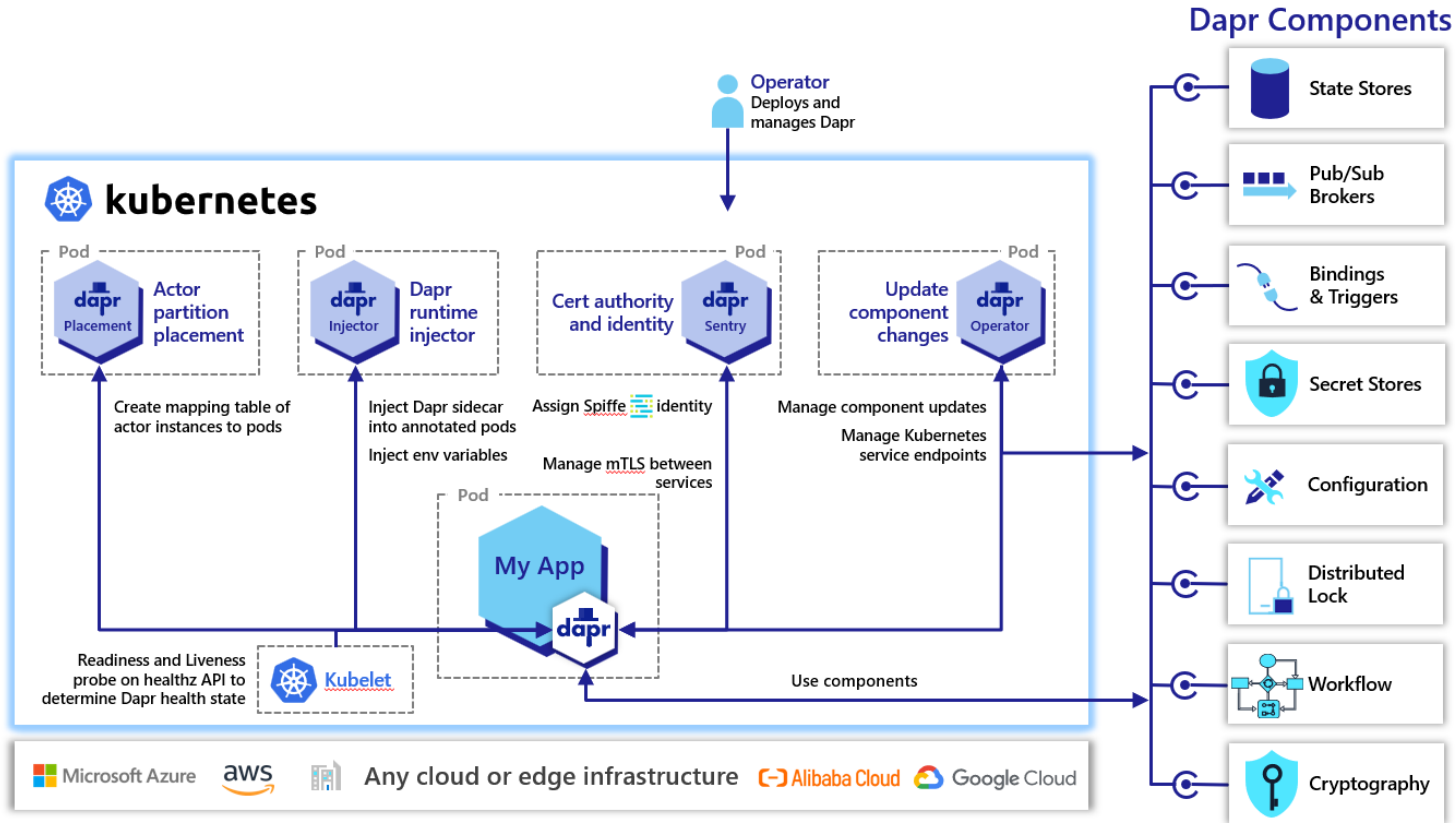
Kubernetes

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

Serverless

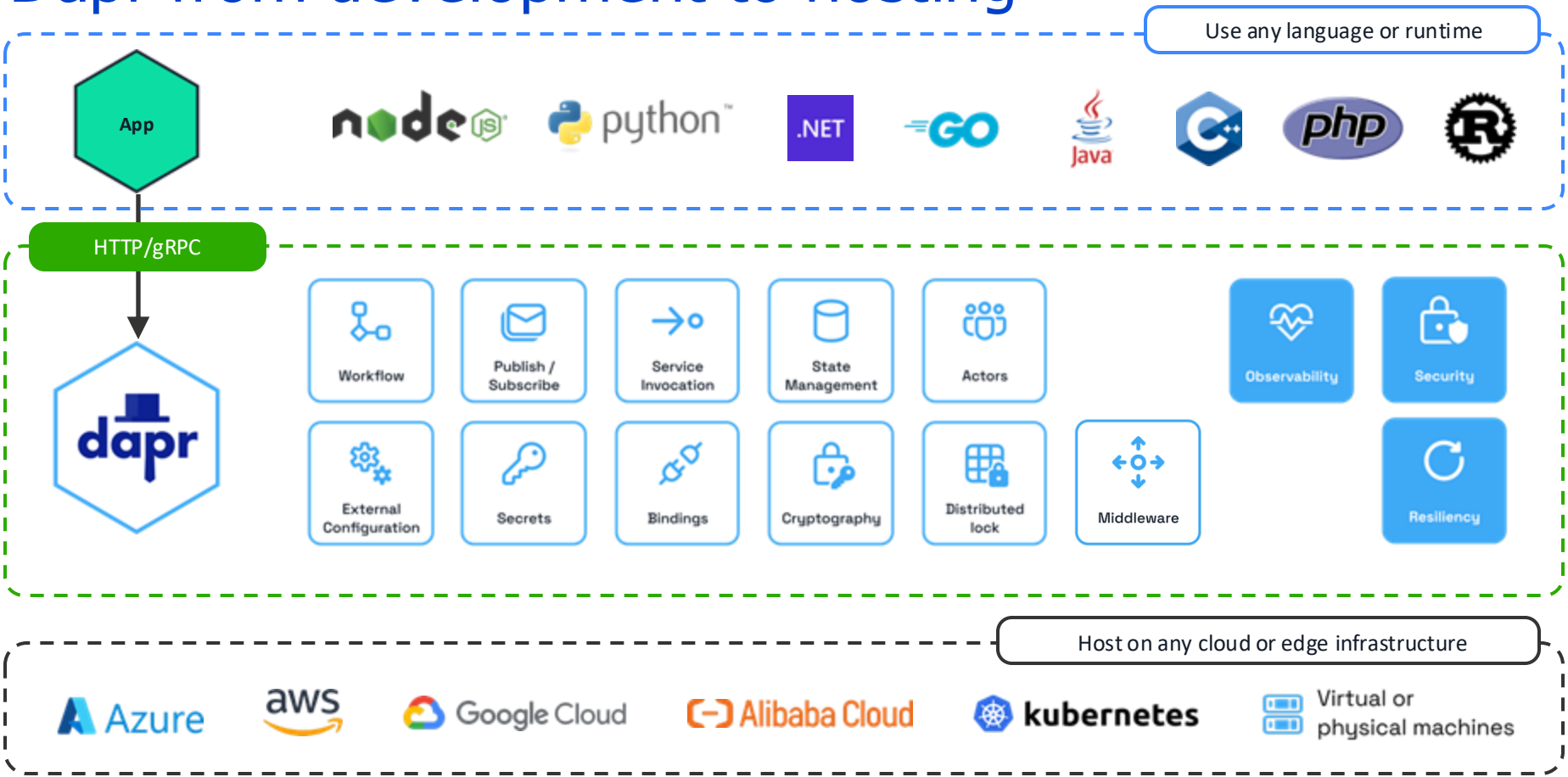
- Integrated in Azure
- i.e. Container Apps

Dapr in Kubernetes

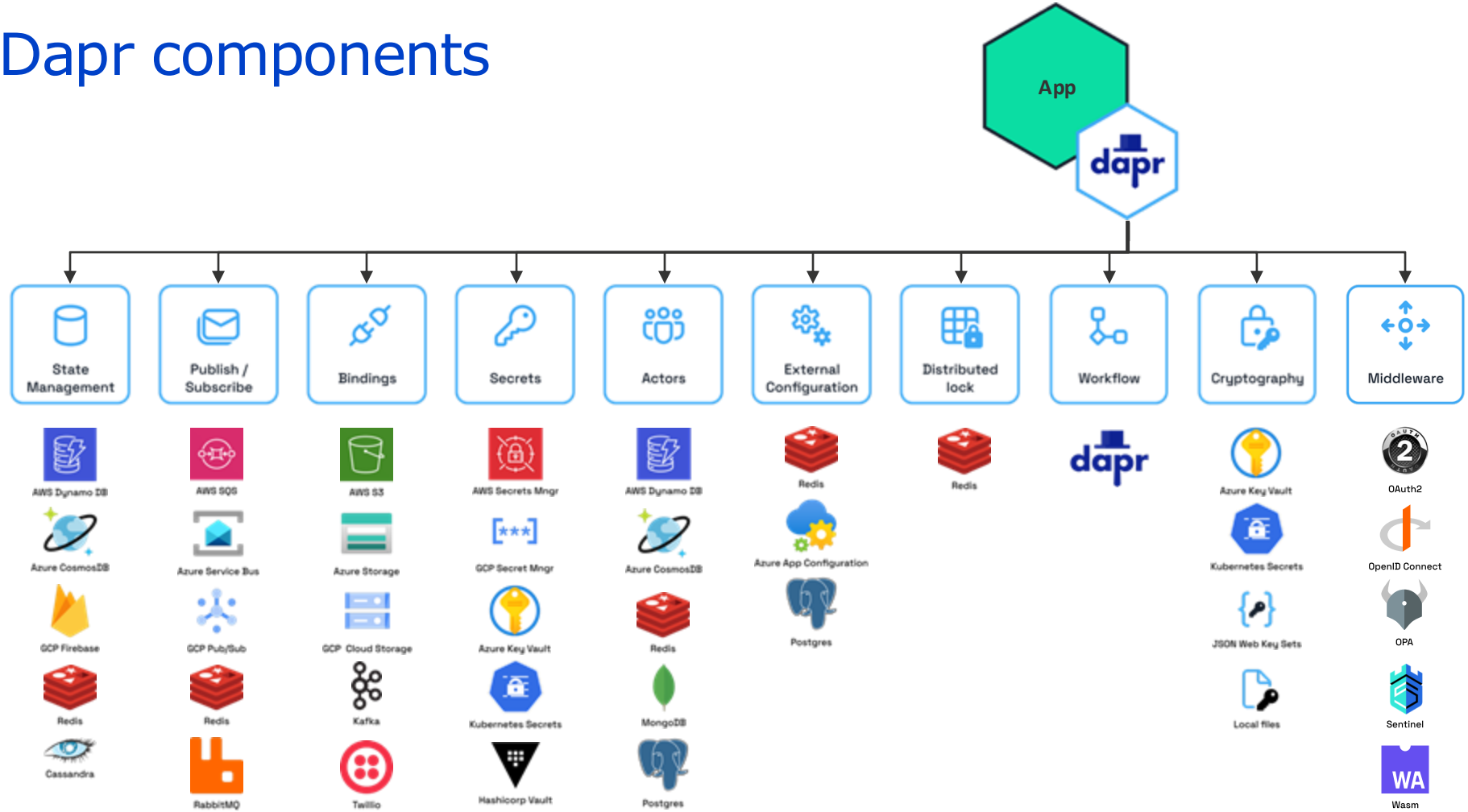


https://docs.dapr.io/operations/hosting/kubernetes/kubernetes-overview/

Dapr from development to hosting

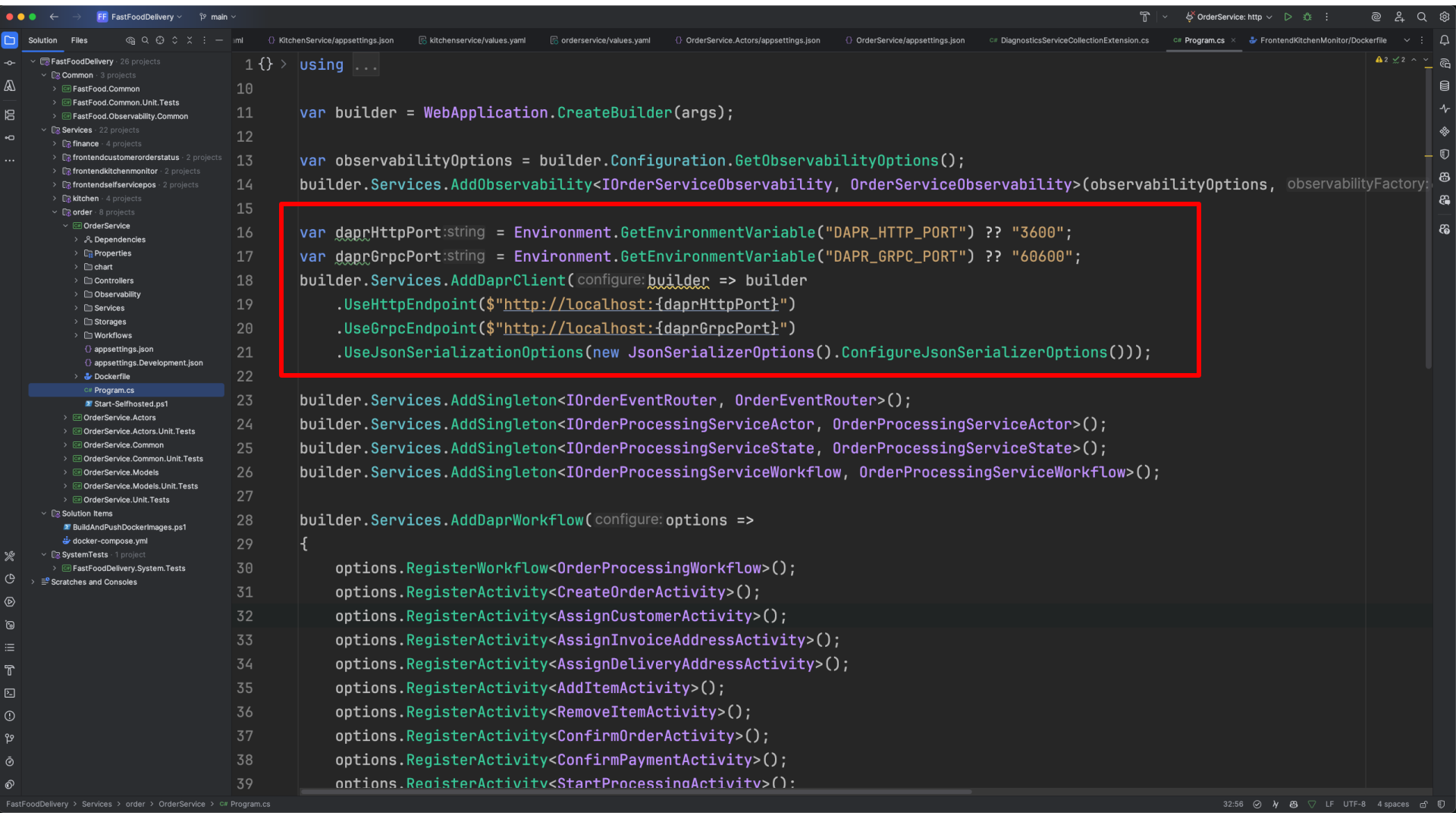


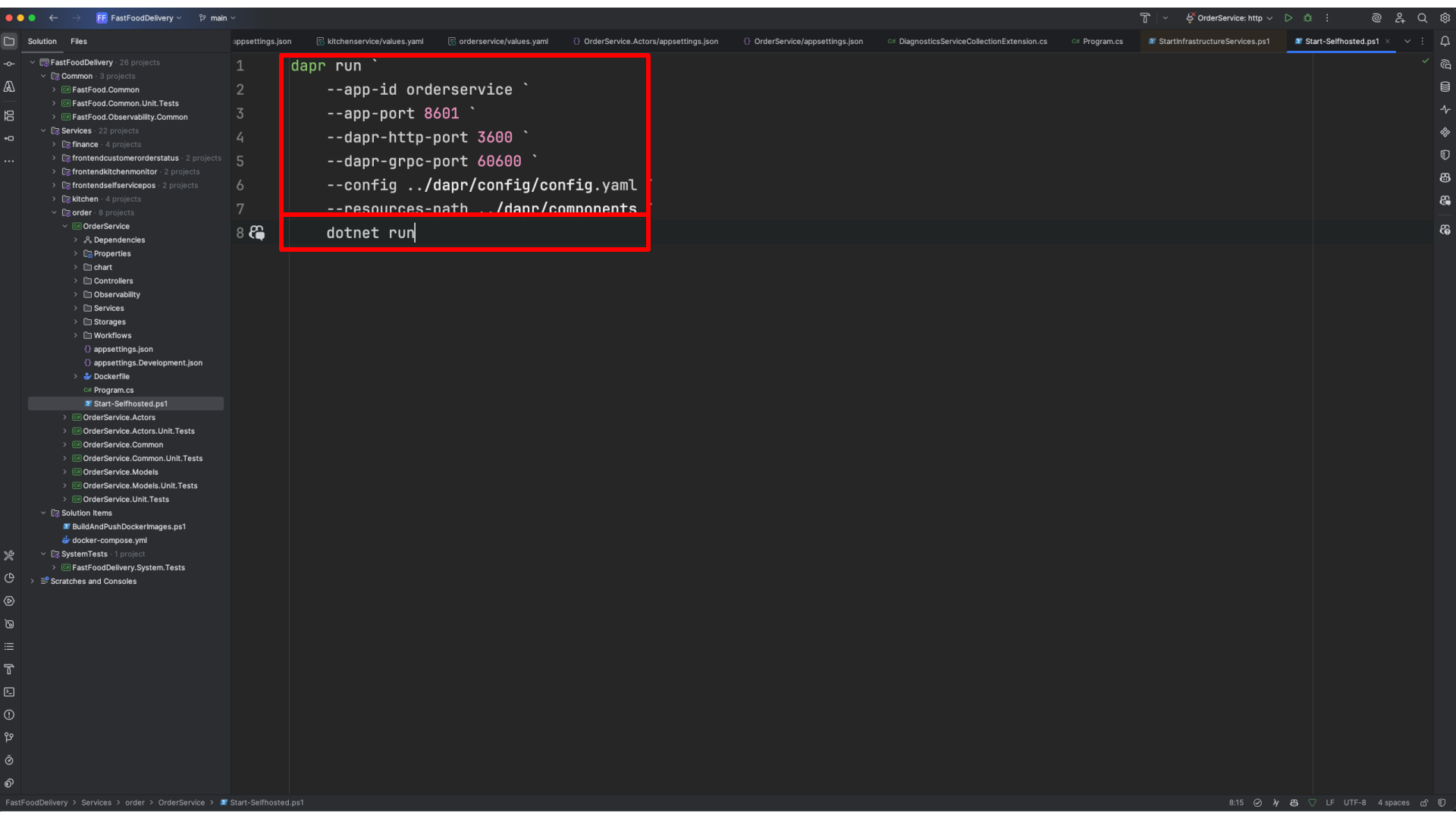
Dapr components

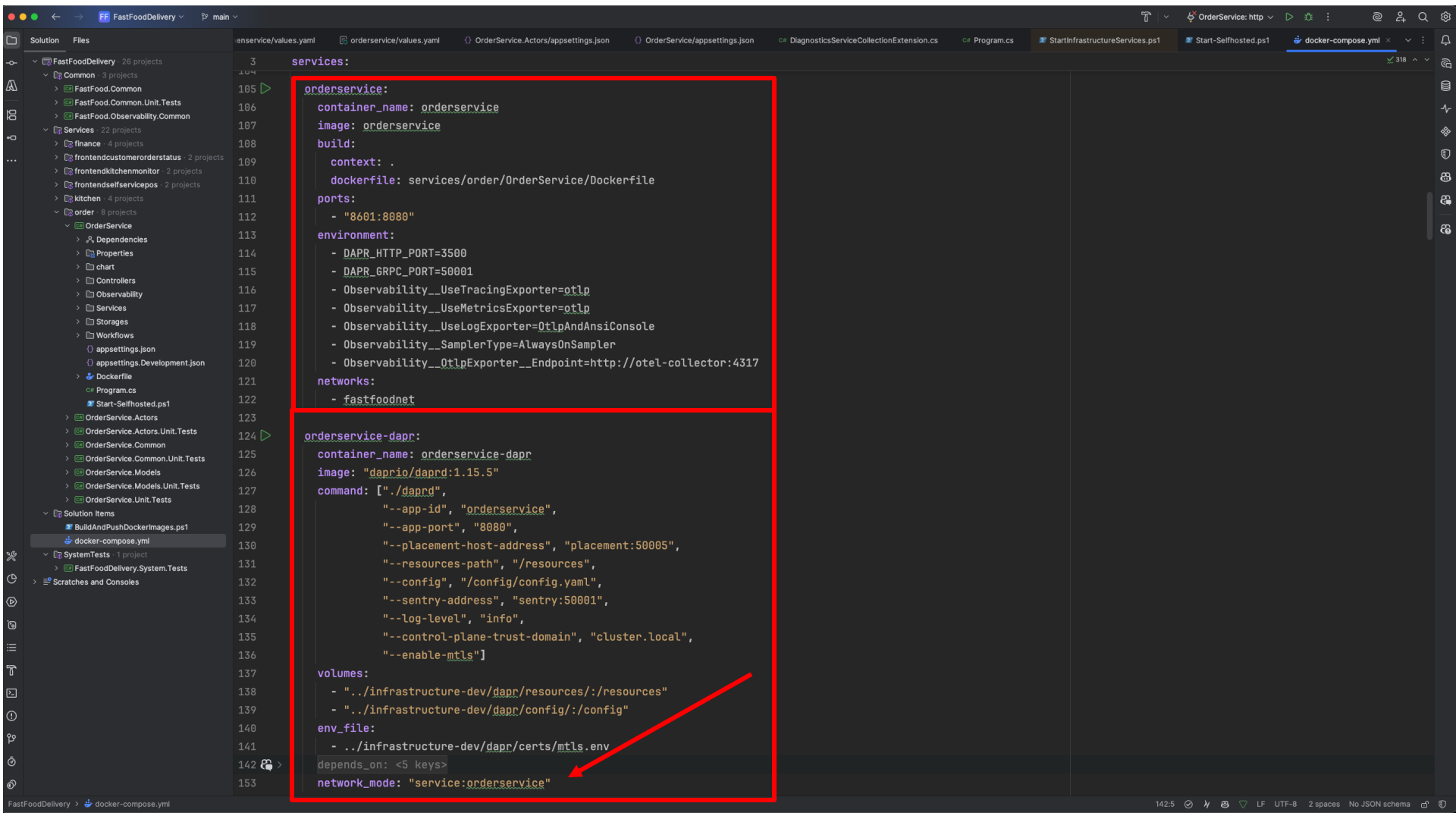


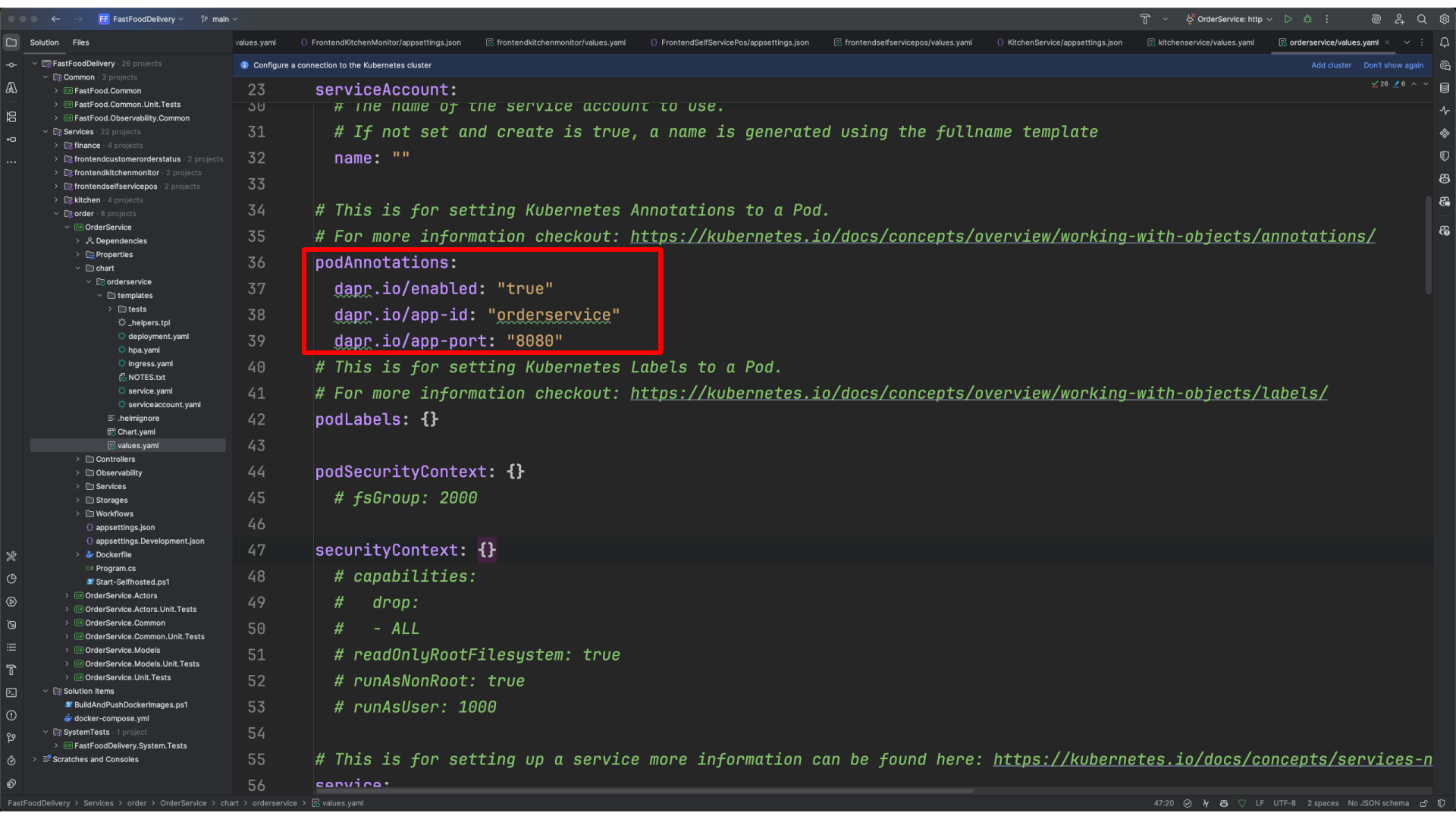

Hands-on!
4tecture
empower your software solutions

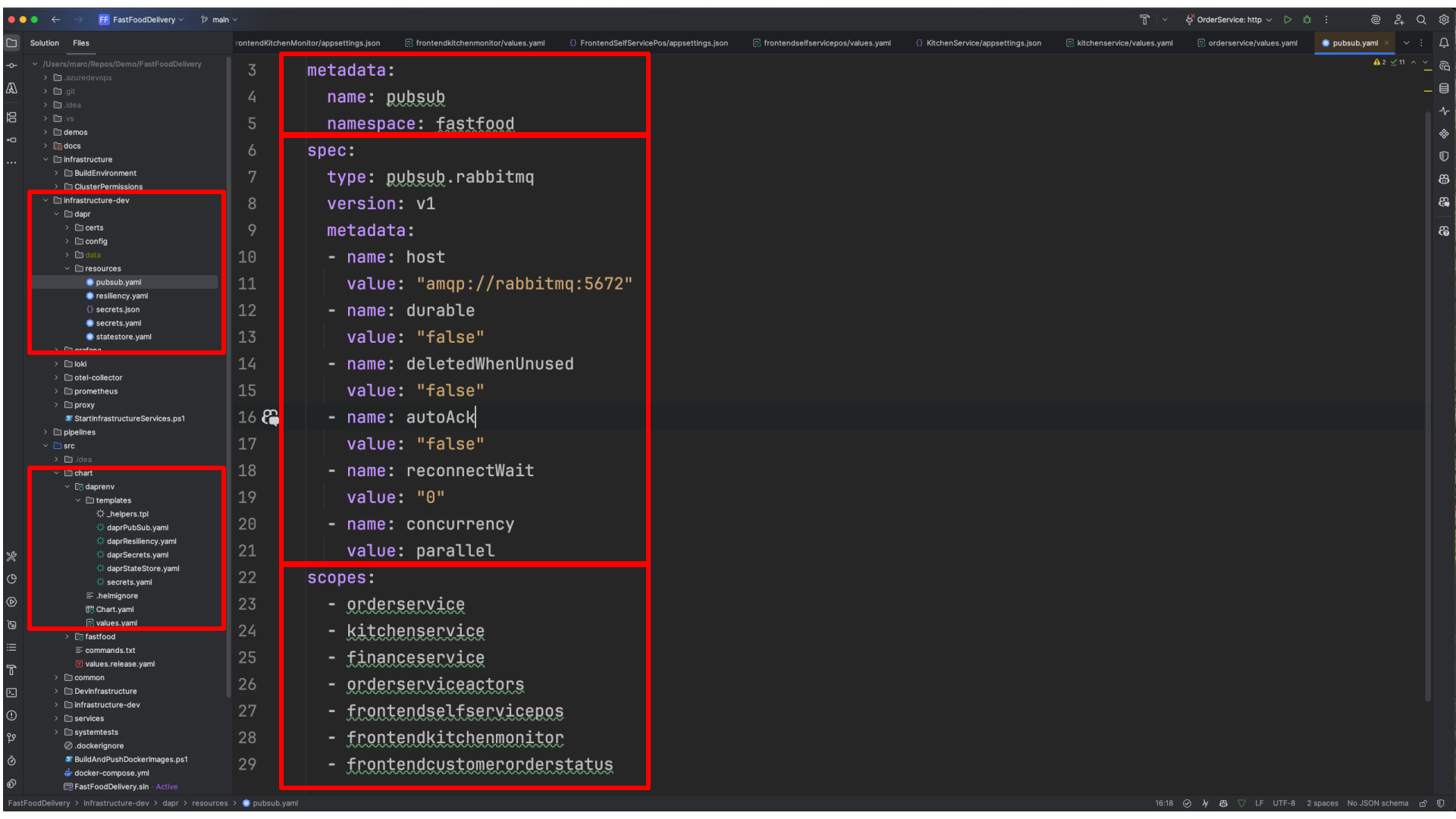

DEMO
Setup


FastFoodDelivery
main
OrderService: http
Solution
Files
KitchenService/appsettings.json
kitchenservice/values.yaml
orderservice/values.yaml
OrderService.Actors/appsettings.json
OrderService/appsettings.json
DiagnosticsServiceCollectionExtension.cs
Program.cs
FrontendKitchenMonitor/Dockerfile
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
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
Observability
Services
Storages
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
using ...
var builder = WebApplication.CreateBuilder(args);
var observabilityOptions = builder.Configuration.GetObservabilityOptions();
builder.Services.AddObservability<IOrderServiceObservability, OrderServiceObservability>(observabilityOptions, observabilityFactory:
var daprHttpPort:string = Environment.GetEnvironmentVariable("DAPR_HTTP_PORT") ?? "3600";
var daprGrpcPort:string = Environment.GetEnvironmentVariable("DAPR_GRPC_PORT") ?? "60600";
builder.Services.AddDaprClient(configure:builder => builder
.UseHttpEndpoint($"http://localhost:{daprHttpPort}")
.UseGrpcEndpoint($"http://localhost:{daprGrpcPort}")
.UseJsonSerializationOptions(new JsonSerializerOptions().ConfigureJsonSerializerOptions()));
builder.Services.AddSingleton<IOrderEventRouter, OrderEventRouter>();
builder.Services.AddSingleton<IOrderProcessingServiceActor, OrderProcessingServiceActor>();
builder.Services.AddSingleton<IOrderProcessingServiceState, OrderProcessingServiceState>();
builder.Services.AddSingleton<IOrderProcessingServiceWorkflow, OrderProcessingServiceWorkflow>();
builder.Services.AddDaprWorkflow(configure:options =>
{
options.RegisterWorkflow<OrderProcessingWorkflow>();
options.RegisterActivity<CreateOrderActivity>();
options.RegisterActivity<AssignCustomerActivity>();
options.RegisterActivity<AssignInvoiceAddressActivity>();
options.RegisterActivity<AssignDeliveryAddressActivity>();
options.RegisterActivity<AddItemActivity>();
options.RegisterActivity<RemoveItemActivity>();
options.RegisterActivity<ConfirmOrderActivity>();
options.RegisterActivity<ConfirmPaymentActivity>();
options.RegisterActivity<StartProcessingActivity>();
FastFoodDelivery › Services › order › OrderService › Program.cs
32:56
LF
UTF-8
4 spaces


FastFoodDelivery
main
OrderService: http
Solution
Files
appsettings.json
kitchenservice/values.yaml
orderservice/values.yaml
OrderService.Actors/appsettings.json
OrderService/appsettings.json
DiagnosticsServiceCollectionExtension.cs
Program.cs
StartInfrastructureServices.ps1
Start-Selfhosted.ps1
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
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
Observability
Services
Storages
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
1 dapr run `
2 --app-id orderservice `
3 --app-port 8601 `
4 --dapr-http-port 3600 `
5 --dapr-grpc-port 60600 `
6 --config ../dapr/config/config.yaml `
7 --resources-path ../dapr/components `
8 dotnet run
FastFoodDelivery › Services › order › OrderService › Start-Selfhosted.ps1
8:15
LF
UTF-8
4 spaces


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:50005",
"--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"


FastFoodDelivery
main
OrderService: http
Solution
Files
values.yaml
FrontendKitchenMonitor/appsettings.json
frontendkitchenmonitor/values.yaml
FrontendSelfServicePos/appsettings.json
frontendselfservicepos/values.yaml
KitchenService/appsettings.json
kitchenservice/values.yaml
orderservice/values.yaml
Configure a connection to the Kubernetes cluster
Add cluster
Don't show again
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
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
orderservice
templates
tests
_helpers.tpl
deployment.yaml
hpa.yaml
ingress.yaml
NOTES.txt
service.yaml
serviceaccount.yaml
.helmignore
Chart.yaml
values.yaml
Controllers
Observability
Services
Storages
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
23 serviceAccount:
30 # The name of the service account to use.
31 # If not set and create is true, a name is generated using the fullname template
32 name: ""
33
34 # This is for setting Kubernetes Annotations to a Pod.
35 # For more information checkout: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/
36 podAnnotations:
37 dapr.io/enabled: "true"
38 dapr.io/app-id: "orderservice"
39 dapr.io/app-port: "8080"
40 # This is for setting Kubernetes Labels to a Pod.
41 # For more information checkout: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/
42 podLabels: {}
43
44 podSecurityContext: {}
45 # fsGroup: 2000
46
47 securityContext: {}
48 # capabilities:
49 # drop:
50 # - ALL
51 # readOnlyRootFilesystem: true
52 # runAsNonRoot: true
53 # runAsUser: 1000
54
55 # This is for setting up a service more information can be found here: https://kubernetes.io/docs/concepts/services-n
56 service:
FastFoodDelivery › Services › order › OrderService › chart › orderservice › values.yaml
47:20
LF
UTF-8
2 spaces
No JSON schema

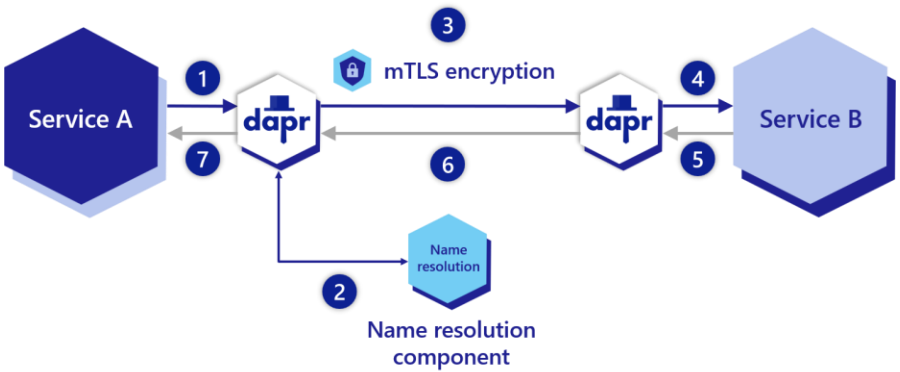

FastFoodDelivery
main
OrderService: http
Solution
Files
rontendKitchenMonitor/appsettings.json
frontendkitchenmonitor/values.yaml
FrontendSelfServicePos/appsettings.json
frontendselfservicepos/values.yaml
KitchenService/appsettings.json
kitchenservice/values.yaml
orderservice/values.yaml
pubsub.yaml
/Users/marc/Repos/Demo/FastFoodDelivery
.azuredevops
.git
.idea
.vs
demos
docs
infrastructure
BuildEnvironment
ClusterPermissions
infrastructure-dev
dapr
certs
config
data
resources
pubsub.yaml
resiliency.yaml
secrets.json
secrets.yaml
statestore.yaml
loki
otel-collector
prometheus
proxy
StartInfrastructureServices.ps1
pipelines
src
.idea
chart
daprenv
templates
_helpers.tpl
daprPubSub.yaml
daprResiliency.yaml
daprSecrets.yaml
daprStateStore.yaml
secrets.yaml
.helmignore
Chart.yaml
values.yaml
fastfood
commands.txt
values.release.yaml
common
DevInfrastructure
infrastructure-dev
services
systemtests
.dockerignore
BuildAndPushDockerImages.ps1
docker-compose.yml
FastFoodDelivery.sln · Active
3 metadata:
4 name: pubsub
5 namespace: fastfood
6 spec:
7 type: pubsub.rabbitmq
8 version: v1
9 metadata:
10 - name: host
11 value: "amqp://rabbitmq:5672"
12 - name: durable
13 value: "false"
14 - name: deletedWhenUnused
15 value: "false"
16 - name: autoAck
17 value: "false"
18 - name: reconnectWait
19 value: "0"
20 - name: concurrency
21 value: parallel
22 scopes:
23 - orderservice
24 - kitchenservice
25 - financeservice
26 - orderserviceactors
27 - frontendselfservicepos
28 - frontendkitchenmonitor
29 - frontendcustomerorderstatus
FastFoodDelivery › infrastructure-dev › dapr › resources › pubsub.yaml
16:18
LF
UTF-8
2 spaces
No JSON schema

Dapr Building Blocks

Service Invocation and Configuration

4tecture
empower your software solutions

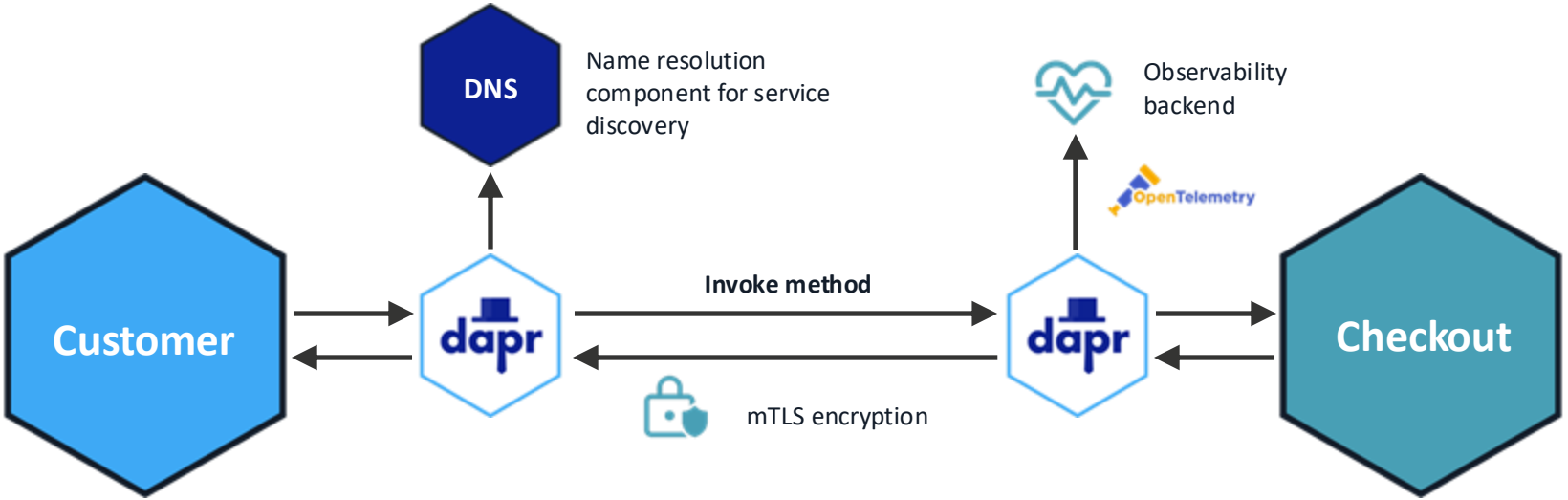
Service Invocation



- 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

https://docs.dapr.io/developing-applications/building-blocks/service-invocation/service-invocation-overview/

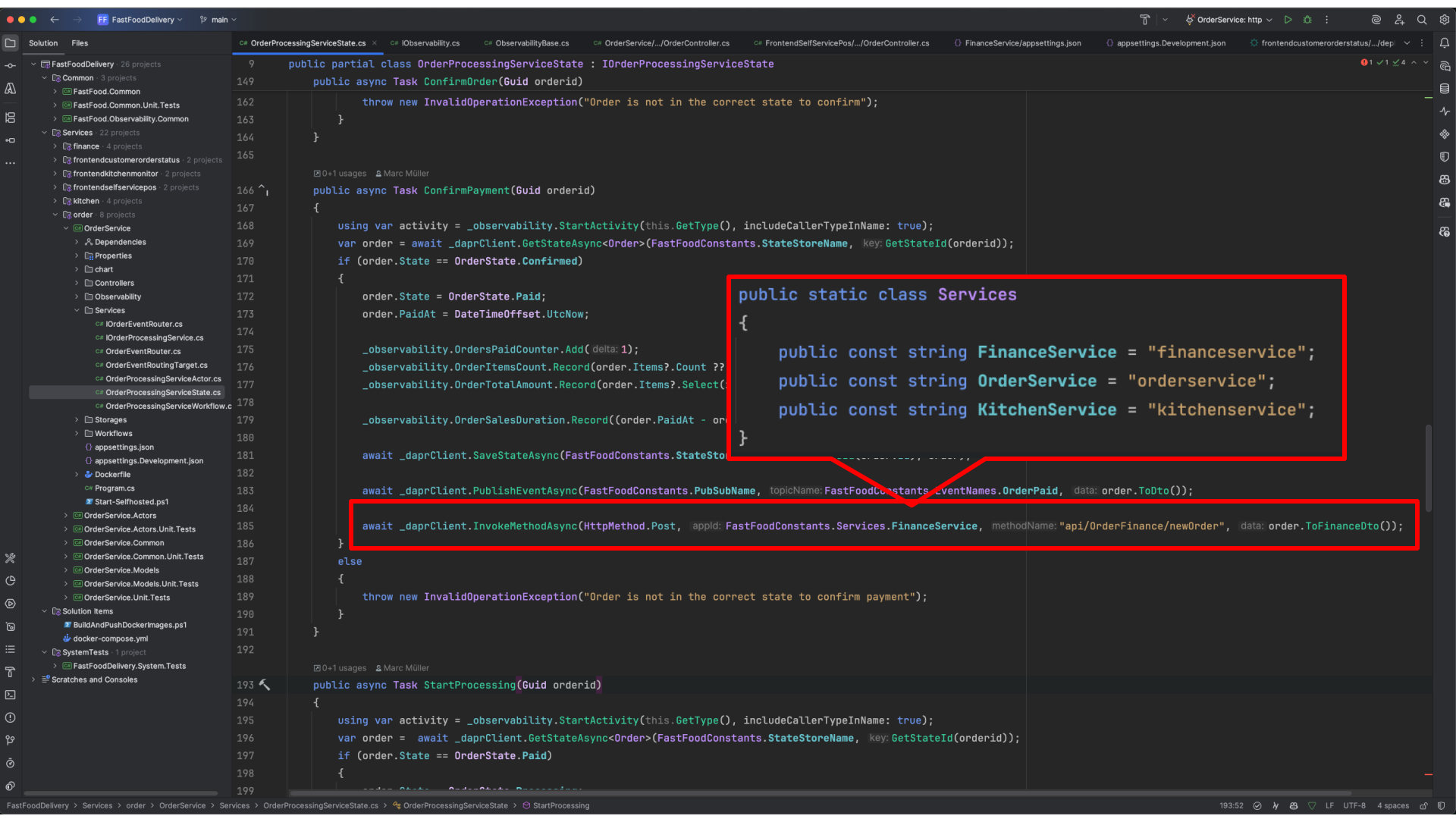
Service Invocation



POST
http://localhost:3500/v1.0/**invoke**/checkout/method/order

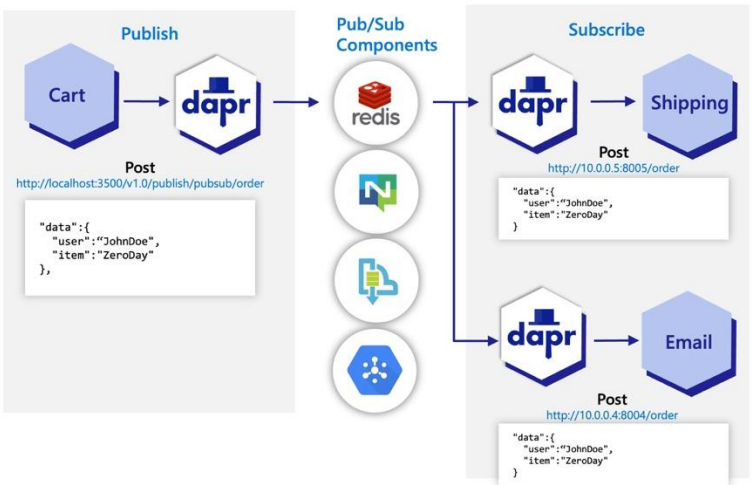
POST
http://localhost:5100/order


DEMO
Service Invocation


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


Dapr Building Blocks
Publish & Subscribe
4tecture
empower your software solutions

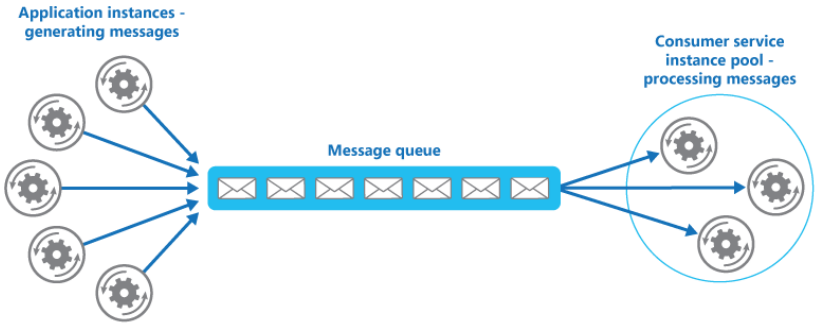
Publish & Subscribe



- 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

https://docs.dapr.io/developing-applications/building-blocks/pubsub/pubsub-overview/

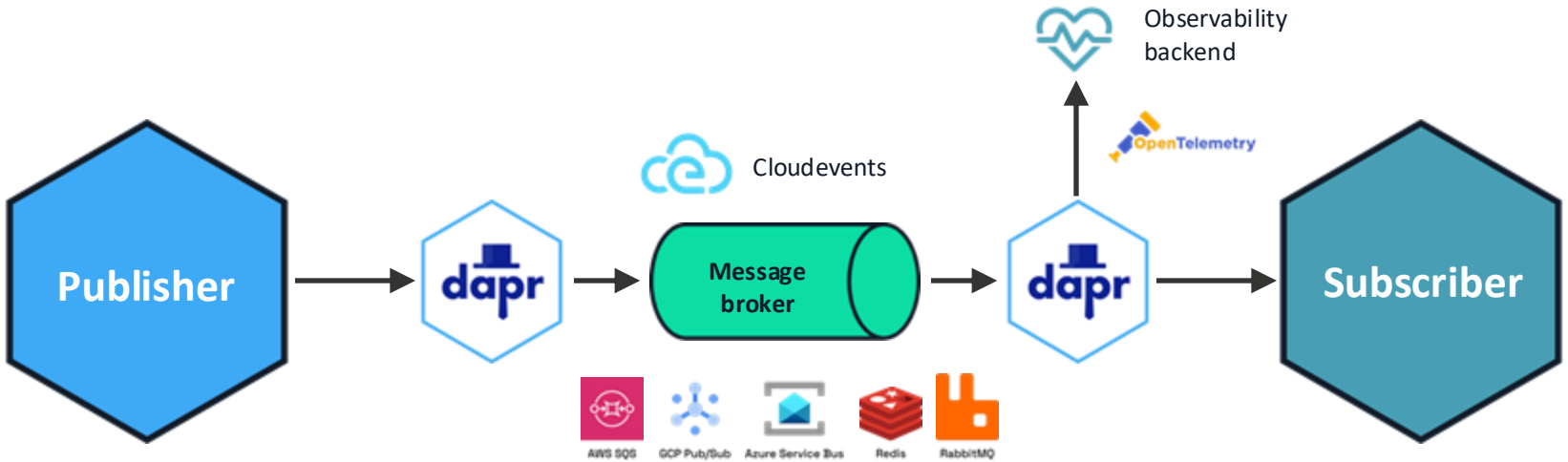
Competing consumers pattern



- 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

https://docs.dapr.io/developing-applications/building-blocks/pubsub/pubsub-overview/

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)

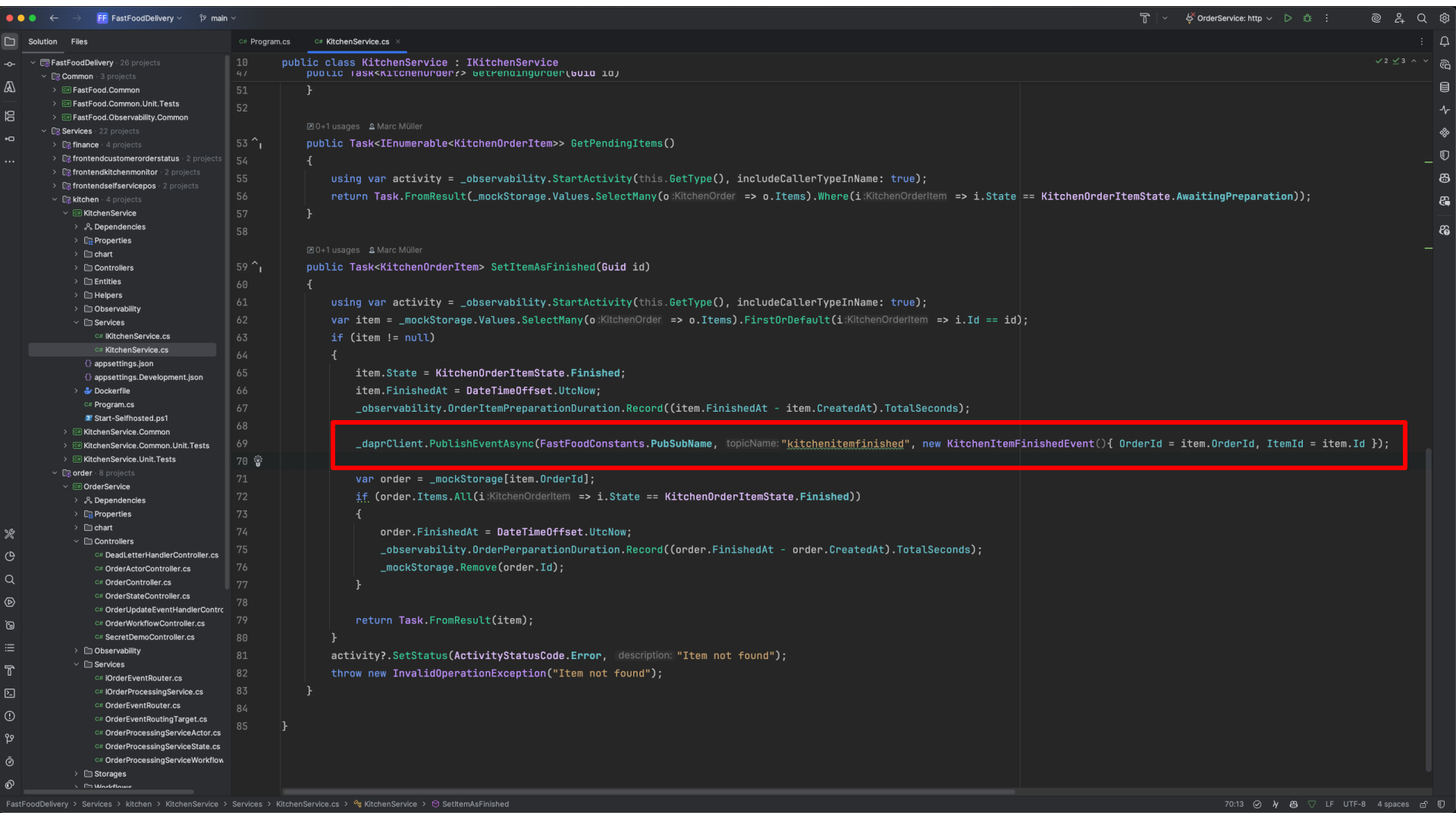
Component	Status	Component version	Since runtime version
GCP Pub/Sub	Stable	v1	1.11

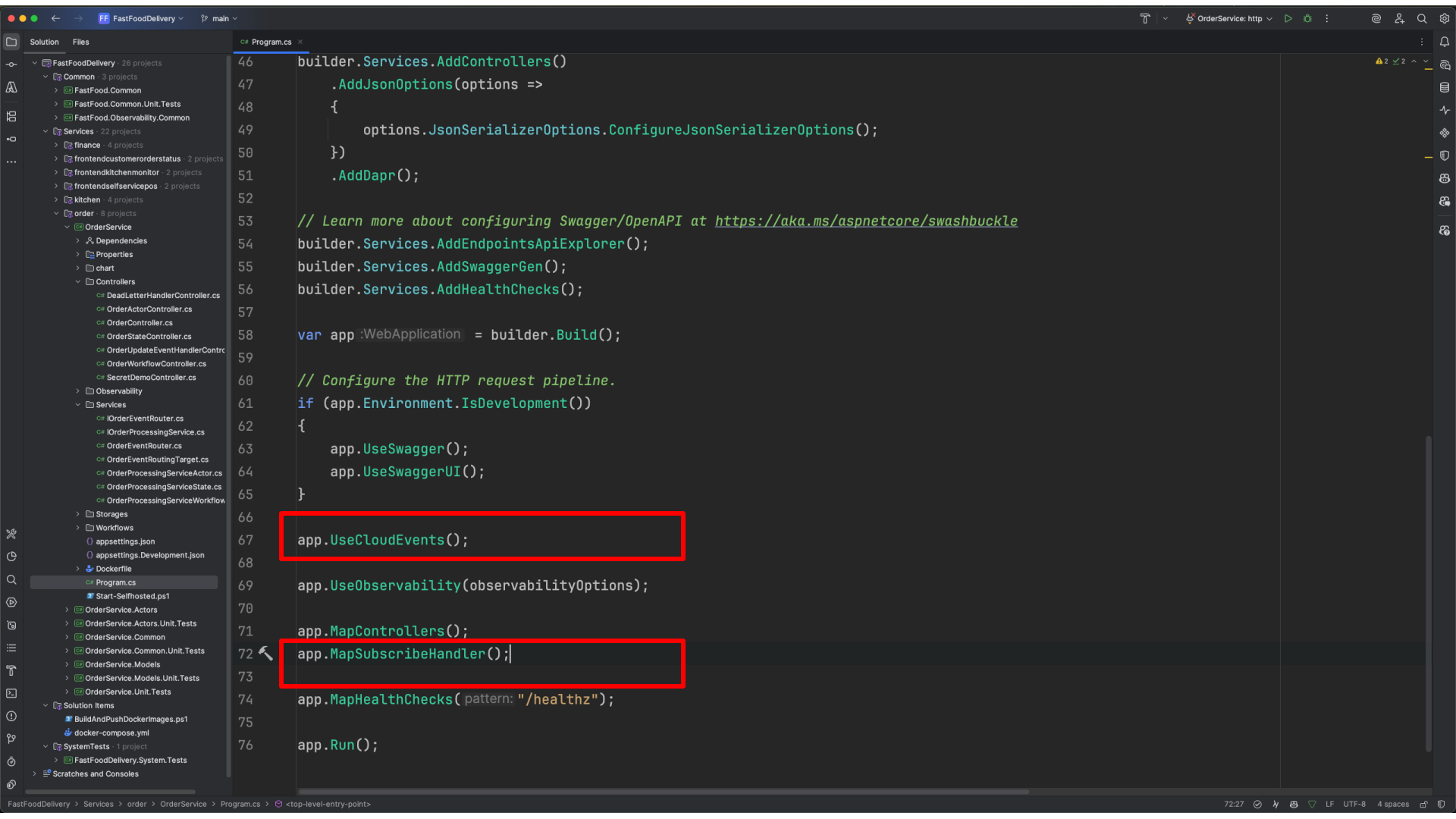
Microsoft Azure

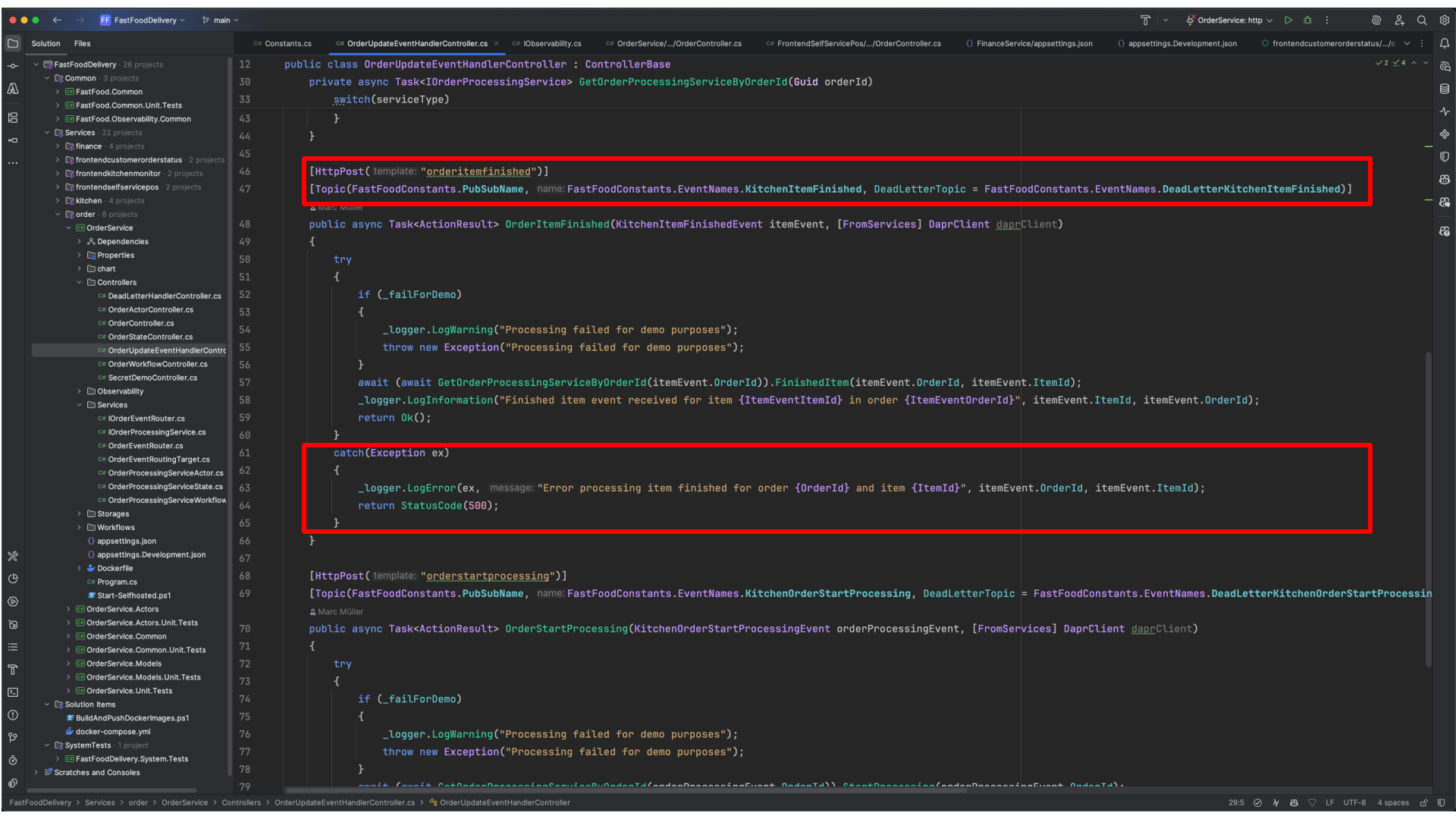
Component	Status	Component version	Since runtime version
Azure Event Hubs	Stable	v1	1.8
Azure Service Bus Queues	Beta	v1	1.10
Azure Service Bus Topics	Stable	v1	1.0

https://docs.dapr.io/reference/components-reference/supported-pubsub/


DEMO
PubSub


FastFoodDelivery
main
OrderService: http
Solution
Files
Program.cs
KitchenService.cs
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
frontendselfservicepos · 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
OrderUpdateEventHandlerContrc
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
Workflows
public class KitchenService : IKitchenService
public Task<KitchenOrder?> GetPendingOrder(Guid id)
}
0+1 usages Marc Müller
public Task<IEnumerable<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));
}
0+1 usages Marc Müller
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.OrderPerparationDuration.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");
}
}
FastFoodDelivery › Services › kitchen › KitchenService › Services › KitchenService.cs › KitchenService › SetItemAsFinished
70:13
LF
UTF-8
4 spaces


FastFoodDelivery
main
OrderService: http
Solution
Files
Program.cs
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
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
DeadLetterHandlerController.cs
OrderActorController.cs
OrderController.cs
OrderStateController.cs
OrderUpdateEventHandlerContro
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
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
46 builder.Services.AddControllers()
47 .AddJsonOptions(options =>
48 {
49 options.JsonSerializerOptions.ConfigureJsonSerializerOptions();
50 })
51 .AddDapr();
52
53 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
54 builder.Services.AddEndpointsApiExplorer();
55 builder.Services.AddSwaggerGen();
56 builder.Services.AddHealthChecks();
57
58 var app :WebApplication = builder.Build();
59
60 // Configure the HTTP request pipeline.
61 if (app.Environment.IsDevelopment())
62 {
63 app.UseSwagger();
64 app.UseSwaggerUI();
65 }
66
67 app.UseCloudEvents();
68
69 app.UseObservability(observabilityOptions);
70
71 app.MapControllers();
72 app.MapSubscribeHandler();
73
74 app.MapHealthChecks(pattern: "/healthz");
75
76 app.Run();
FastFoodDelivery › Services › order › OrderService › Program.cs › <top-level-entry-point>
72:27
LF
UTF-8
4 spaces

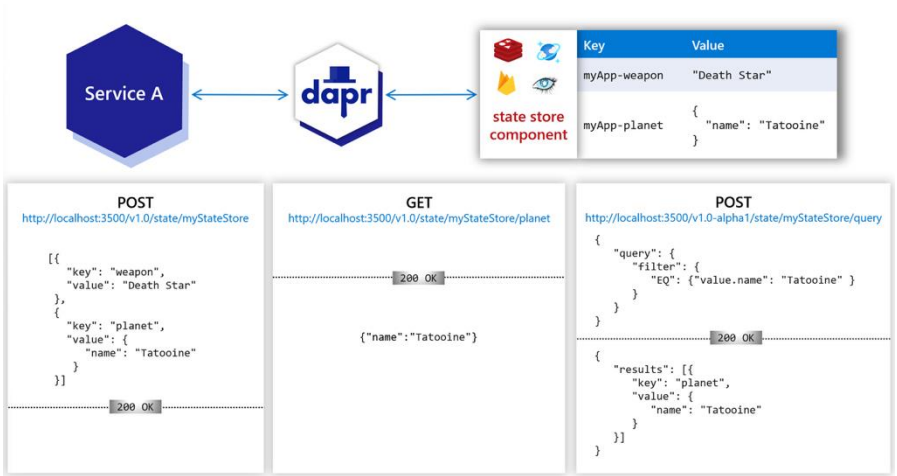

FastFoodDelivery
main
OrderService: http
Solution
Files
Constants.cs
OrderUpdateEventHandlerController.cs
IObservability.cs
OrderService/.../OrderController.cs
FrontendSelfServicePos/.../OrderController.cs
FinanceService/appsettings.json
appsettings.Development.json
frontendcustomerorderstatus/.../c
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
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
DeadLetterHandlerController.cs
OrderActorController.cs
OrderController.cs
OrderStateController.cs
OrderUpdateEventHandlerContro
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
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
public class OrderUpdateEventHandlerController : ControllerBase
private async Task<IOrderProcessingService> GetOrderProcessingServiceByOrderId(Guid orderId)
switch(serviceType)
}
}
[HttpPost(template: "orderitemfinished")]
[Topic(FastFoodConstants.PubSubName, name: FastFoodConstants.EventNames.KitchenItemFinished, DeadLetterTopic = FastFoodConstants.EventNames.DeadLetterKitchenItemFinished)]
Marc Müller
public async Task<ActionResult> OrderItemFinished(KitchenItemFinishedEvent itemEvent, [FromServices] DaprClient daprClient)
{
try
{
if (_failForDemo)
{
_logger.LogWarning("Processing failed for demo purposes");
throw new Exception("Processing failed for demo purposes");
}
await (await GetOrderProcessingServiceByOrderId(itemEvent.OrderId)).FinishedItem(itemEvent.OrderId, itemEvent.ItemId);
_logger.LogInformation("Finished item event received for item {ItemEventItemId} in order {ItemEventOrderId}", itemEvent.ItemId, itemEvent.OrderId);
return Ok();
}
catch(Exception ex)
{
_logger.LogError(ex, message: "Error processing item finished for order {OrderId} and item {ItemId}", itemEvent.OrderId, itemEvent.ItemId);
return StatusCode(500);
}
}
[HttpPost(template: "orderstartprocessing")]
[Topic(FastFoodConstants.PubSubName, name: FastFoodConstants.EventNames.KitchenOrderStartProcessing, DeadLetterTopic = FastFoodConstants.EventNames.DeadLetterKitchenOrderStartProcessin
Marc Müller
public async Task<ActionResult> OrderStartProcessing(KitchenOrderStartProcessingEvent orderProcessingEvent, [FromServices] DaprClient daprClient)
{
try
{
if (_failForDemo)
{
_logger.LogWarning("Processing failed for demo purposes");
throw new Exception("Processing failed for demo purposes");
}
FastFoodDelivery › Services › order › OrderService › Controllers › OrderUpdateEventHandlerController.cs › OrderUpdateEventHandlerController
29:5
LF
UTF-8
4 spaces

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

https://docs.dapr.io/developing-applications/building-blocks/state-management/state-management-overview/

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

https://docs.dapr.io/reference/components-reference/supported-state-stores/


DEMO
State Store

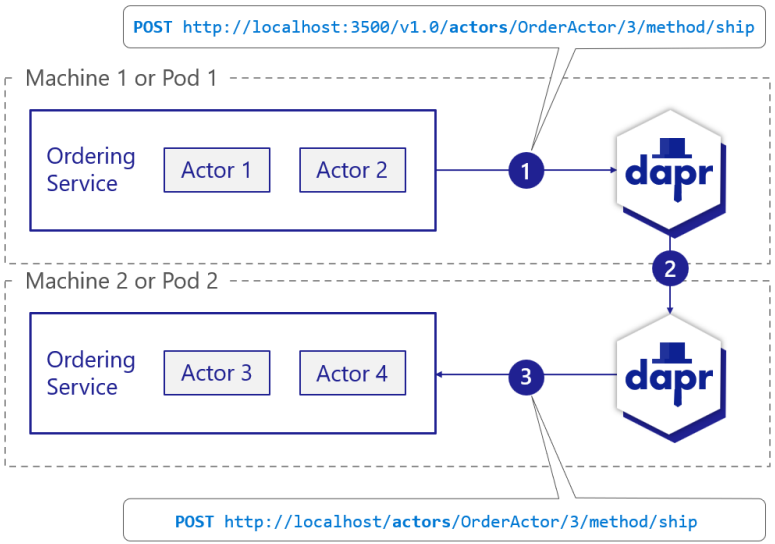
```
public partial class OrderProcessingServiceState : IOrderProcessingServiceState
    public async Task AssignDeliveryAddress(Guid orderid, Address address)
            throw new InvalidOperationException("Order is not in the correct state to assign a delivery address");
        }
    }

    0+1 usages  Marc Müller
    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");
        }
    }

    0+1 usages  Marc Müller
    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 :OrderItem  => i.Id == itemId);
            if (itemToRemove != null && order.Items != null)
            {
                order.Items.Remove(itemToRemove);
            }
```

Dapr Building Blocks

Virtual Actors

4tecture
empower your software solutions

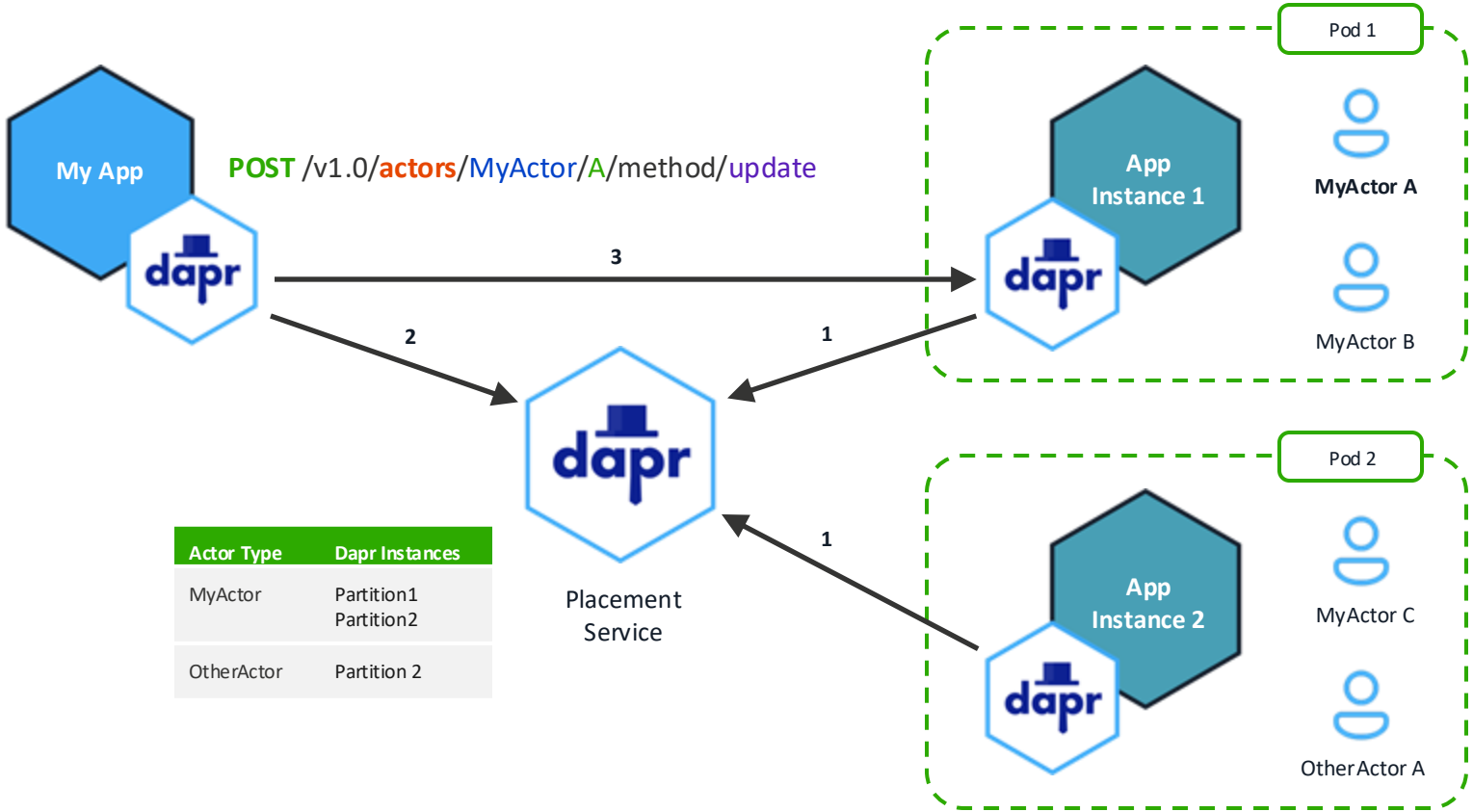
Actors



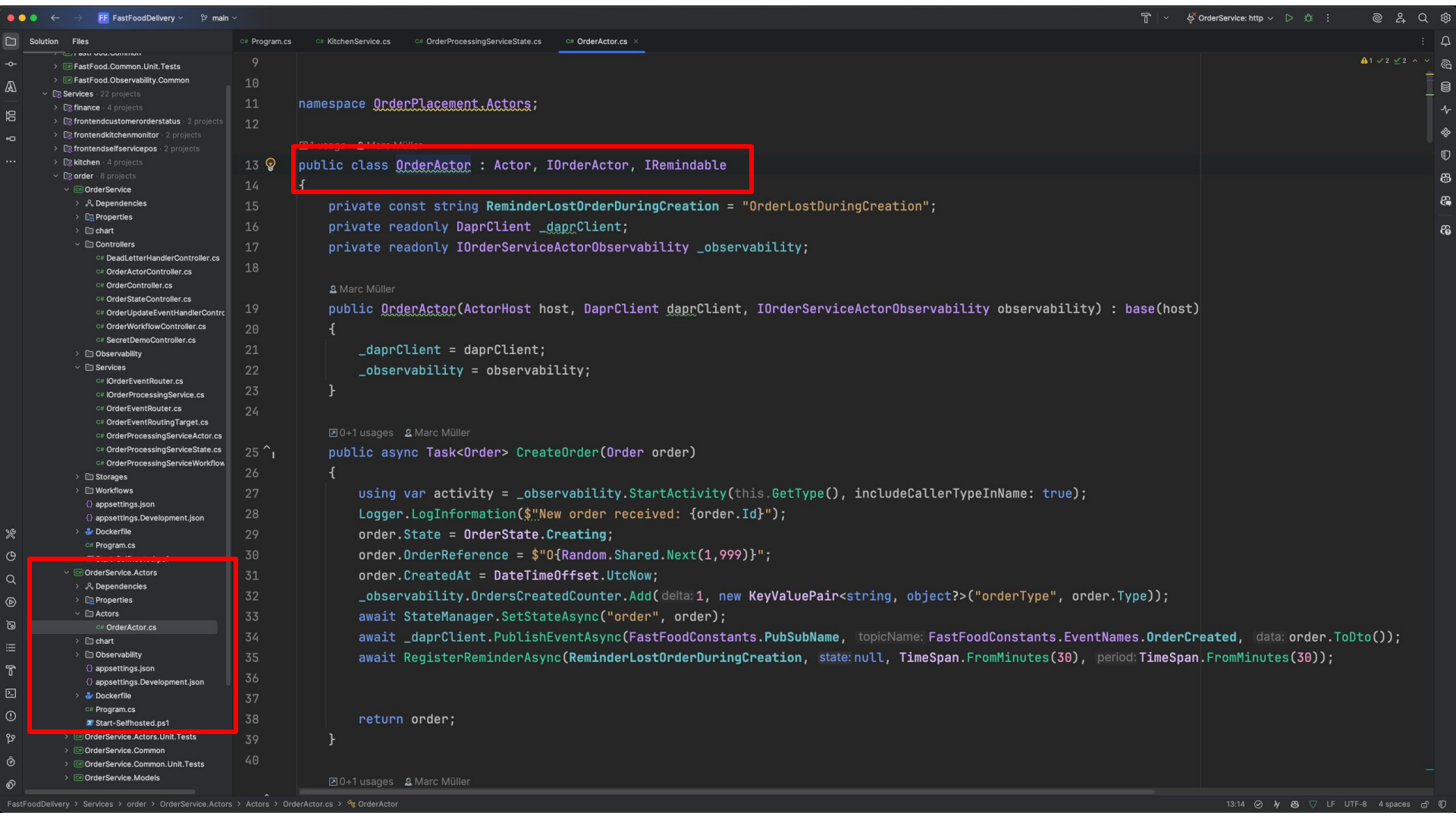
- Virtual Actor pattern
- Stateful, long running objects with identity
- Encapsulate state and behavior within a distributed system
- Actor state store
- Actor timers and reminders

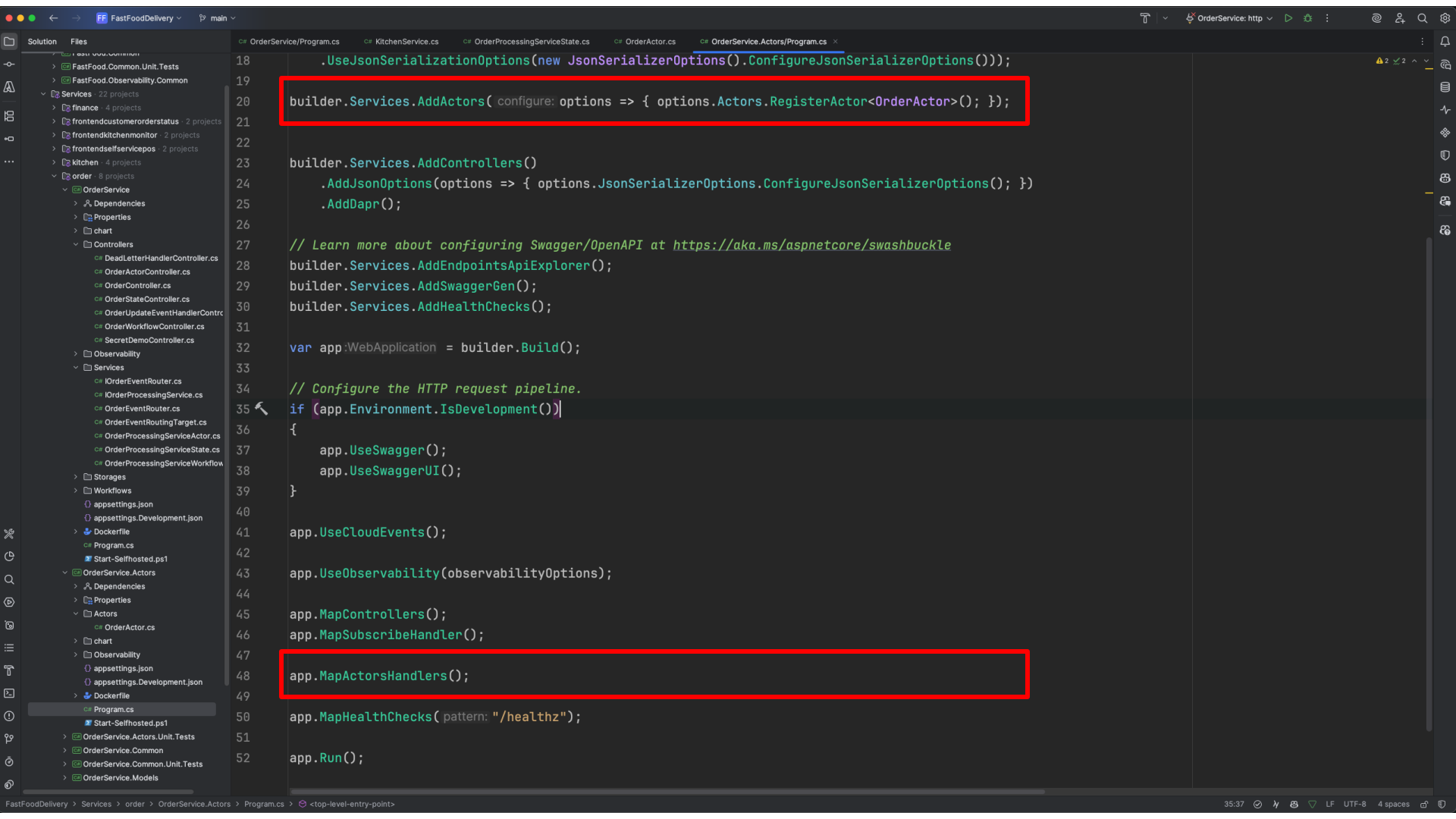
https://docs.dapr.io/developing-applications/building-blocks/actors/actors-overview/

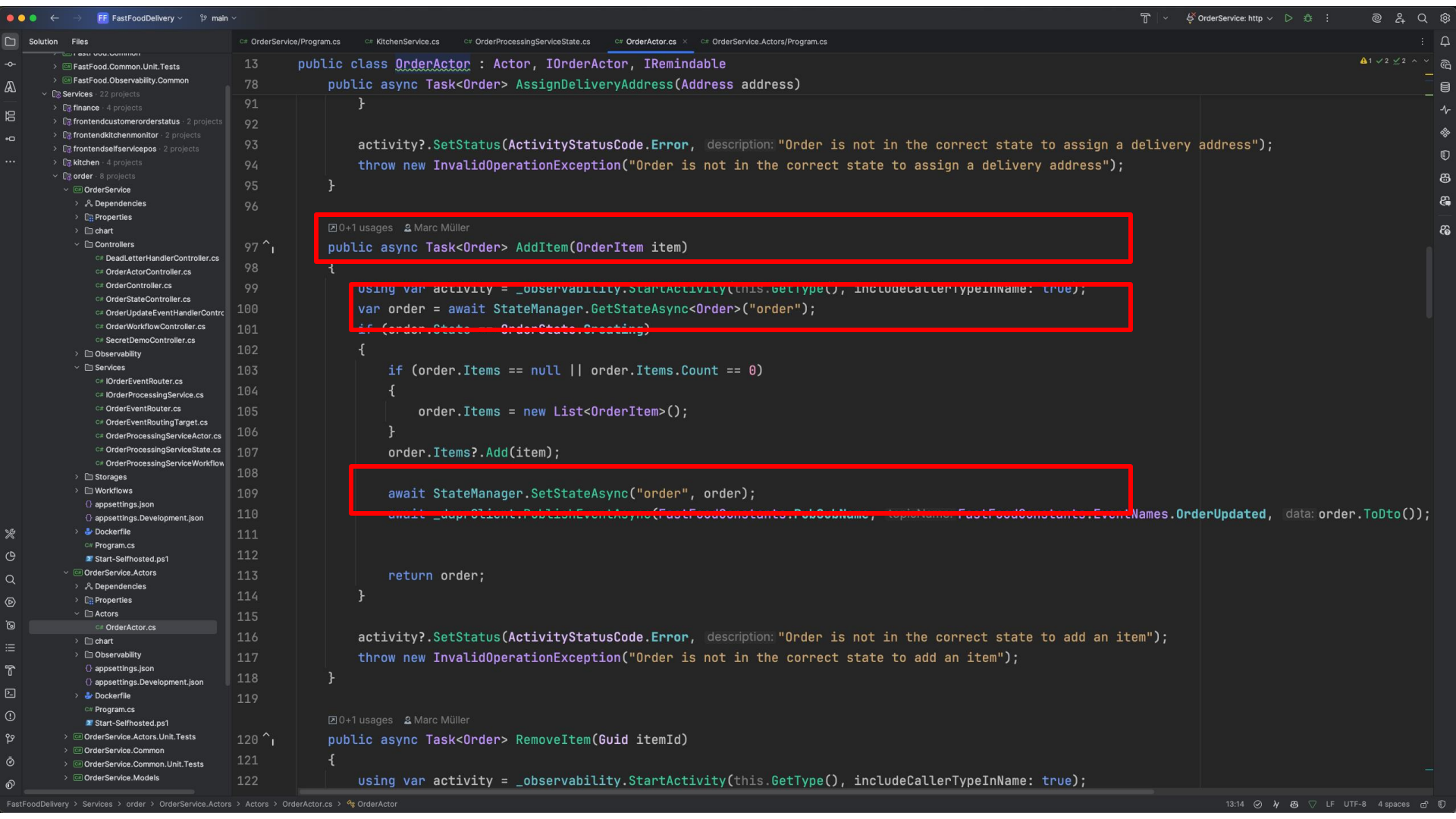
Actor placement

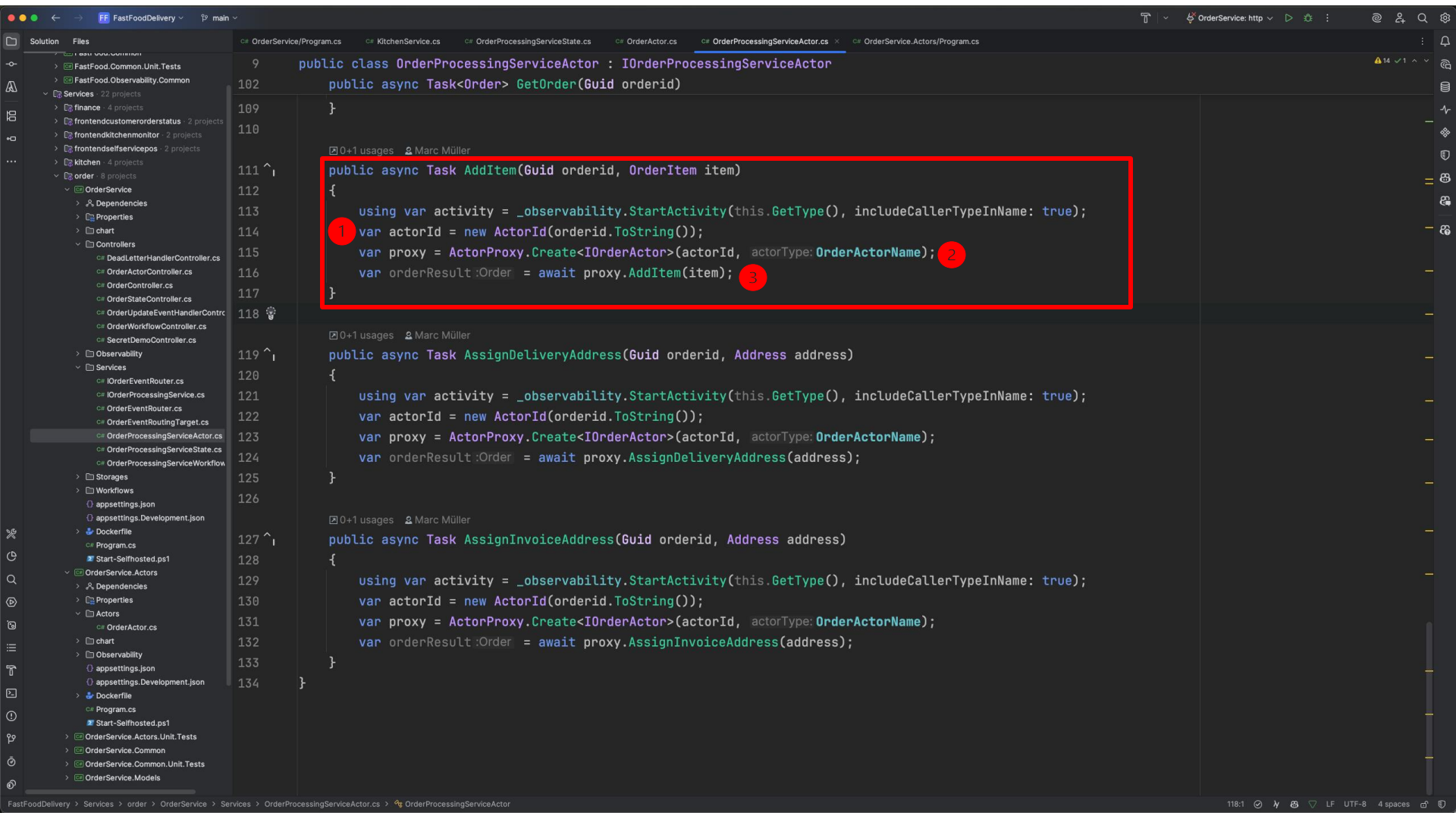



DEMO
Virtual Actors


FastFoodDelivery
main
OrderService: http
Solution
Files
Program.cs
KitchenService.cs
OrderProcessingServiceState.cs
OrderActor.cs
FastFood.Common.Unit.Tests
FastFood.Observability.Common
Services · 22 projects
finance · 4 projects
frontendcustomerorderstatus · 2 projects
frontendkitchenmonitor · 2 projects
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
DeadLetterHandlerController.cs
OrderActorController.cs
OrderController.cs
OrderStateController.cs
OrderUpdateEventHandlerContro
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
Workflows
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
OrderService.Actors
Dependencies
Properties
Actors
OrderActor.cs
chart
Observability
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors.Unit.Tests
OrderService.Common
OrderService.Common.Unit.Tests
OrderService.Models
9
10
11 namespace OrderPlacement.Actors;
12
13 public class OrderActor : Actor, IOrderActor, IRemindable
14 {
15 private const string ReminderLostOrderDuringCreation = "OrderLostDuringCreation";
16 private readonly DaprClient _daprClient;
17 private readonly IOrderServiceActorObservability _observability;
18
Marc Müller
19 public OrderActor(ActorHost host, DaprClient daprClient, IOrderServiceActorObservability observability) : base(host)
20 {
21 _daprClient = daprClient;
22 _observability = observability;
23 }
24
0+1 usages Marc Müller
25 public async Task<Order> CreateOrder(Order order)
26 {
27 using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
28 Logger.LogInformation($"New order received: {order.Id}");
29 order.State = OrderState.Creating;
30 order.OrderReference = $"0{Random.Shared.Next(1,999)}";
31 order.CreatedAt = DateTimeOffset.UtcNow;
32 _observability.OrdersCreatedCounter.Add(delta: 1, new KeyValuePair<string, object?>("orderType", order.Type));
33 await StateManager.SetStateAsync("order", order);
34 await _daprClient.PublishEventAsync(FastFoodConstants.PubSubName, topicName: FastFoodConstants.EventNames.OrderCreated, data: order.ToDto());
35 await RegisterReminderAsync(ReminderLostOrderDuringCreation, state: null, TimeSpan.FromMinutes(30), period: TimeSpan.FromMinutes(30));
36
37
38 return order;
39 }
40
0+1 usages Marc Müller
FastFoodDelivery › Services › order › OrderService.Actors › Actors › OrderActor.cs › OrderActor
13:14
LF
UTF-8
4 spaces


FastFoodDelivery
main
OrderService: http
Solution
Files
OrderService/Program.cs
KitchenService.cs
OrderProcessingServiceState.cs
OrderActor.cs
OrderService.Actors/Program.cs
FastFood.Common.Unit.Tests
FastFood.Observability.Common
Services · 22 projects
finance · 4 projects
frontendcustomerorderstatus · 2 projects
frontendkitchenmonitor · 2 projects
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
DeadLetterHandlerController.cs
OrderActorController.cs
OrderController.cs
OrderStateController.cs
OrderUpdateEventHandlerContrc
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
Workflows
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors
Dependencies
Properties
Actors
OrderActor.cs
chart
Observability
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors.Unit.Tests
OrderService.Common
OrderService.Common.Unit.Tests
OrderService.Models
.UseJsonSerializationOptions(new JsonSerializerOptions().ConfigureJsonSerializerOptions()));
builder.Services.AddActors(configure: options => { options.Actors.RegisterActor<OrderActor>(); });
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.MapActorsHandlers();
app.MapHealthChecks(pattern: "/healthz");
app.Run();
FastFoodDelivery › Services › order › OrderService.Actors › Program.cs › <top-level-entry-point>
35:37
LF
UTF-8
4 spaces


FastFoodDelivery
main
OrderService: http
Solution
Files
OrderService/Program.cs
KitchenService.cs
OrderProcessingServiceState.cs
OrderActor.cs
OrderService.Actors/Program.cs
FastFood.Common.Unit.Tests
FastFood.Observability.Common
Services · 22 projects
finance · 4 projects
frontendcustomerorderstatus · 2 projects
frontendkitchenmonitor · 2 projects
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
DeadLetterHandlerController.cs
OrderActorController.cs
OrderController.cs
OrderStateController.cs
OrderUpdateEventHandlerContro
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
Workflows
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors
Dependencies
Properties
Actors
OrderActor.cs
chart
Observability
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors.Unit.Tests
OrderService.Common
OrderService.Common.Unit.Tests
OrderService.Models
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");
}
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 _daprClient.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");
}
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
LF
UTF-8
4 spaces

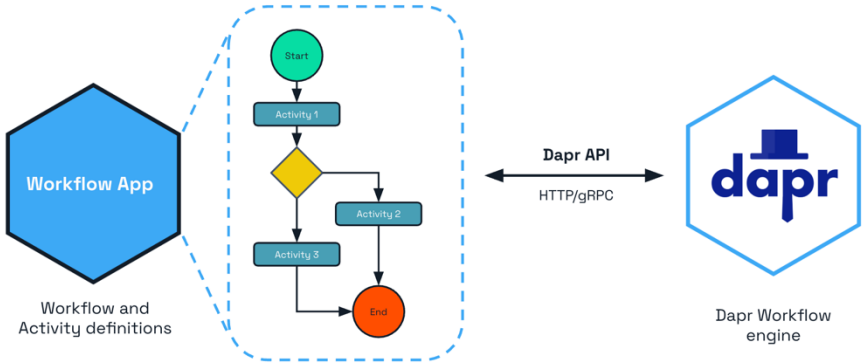

FastFoodDelivery
main
OrderService: http
Solution
Files
OrderService/Program.cs
KitchenService.cs
OrderProcessingServiceState.cs
OrderActor.cs
OrderProcessingServiceActor.cs
OrderService.Actors/Program.cs
FastFood.Common.Unit.Tests
FastFood.Observability.Common
Services · 22 projects
finance · 4 projects
frontendcustomerorderstatus · 2 projects
frontendkitchenmonitor · 2 projects
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
DeadLetterHandlerController.cs
OrderActorController.cs
OrderController.cs
OrderStateController.cs
OrderUpdateEventHandlerContro
OrderWorkflowController.cs
SecretDemoController.cs
Observability
Services
IOrderEventRouter.cs
IOrderProcessingService.cs
OrderEventRouter.cs
OrderEventRoutingTarget.cs
OrderProcessingServiceActor.cs
OrderProcessingServiceState.cs
OrderProcessingServiceWorkflow
Storages
Workflows
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors
Dependencies
Properties
Actors
OrderActor.cs
chart
Observability
appsettings.json
appsettings.Development.json
Dockerfile
Program.cs
Start-Selfhosted.ps1
OrderService.Actors.Unit.Tests
OrderService.Common
OrderService.Common.Unit.Tests
OrderService.Models
9 public class OrderProcessingServiceActor : IOrderProcessingServiceActor
102 public async Task<Order> GetOrder(Guid orderid)
109 }
110
0+1 usages Marc Müller
111 public async Task AddItem(Guid orderid, OrderItem item)
112 {
113 using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
114 var actorId = new ActorId(orderid.ToString());
115 var proxy = ActorProxy.Create<IOrderActor>(actorId, actorType: OrderActorName);
116 var orderResult :Order = await proxy.AddItem(item);
117 }
118
1
2
3
0+1 usages Marc Müller
119 public async Task AssignDeliveryAddress(Guid orderid, Address address)
120 {
121 using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
122 var actorId = new ActorId(orderid.ToString());
123 var proxy = ActorProxy.Create<IOrderActor>(actorId, actorType: OrderActorName);
124 var orderResult :Order = await proxy.AssignDeliveryAddress(address);
125 }
126
0+1 usages Marc Müller
127 public async Task AssignInvoiceAddress(Guid orderid, Address address)
128 {
129 using var activity = _observability.StartActivity(this.GetType(), includeCallerTypeInName: true);
130 var actorId = new ActorId(orderid.ToString());
131 var proxy = ActorProxy.Create<IOrderActor>(actorId, actorType: OrderActorName);
132 var orderResult :Order = await proxy.AssignInvoiceAddress(address);
133 }
134 }
FastFoodDelivery › Services › order › OrderService › Services › OrderProcessingServiceActor.cs › OrderProcessingServiceActor
118:1
LF
UTF-8
4 spaces

Dapr Building Blocks

Workflows

4tecture
empower your software solutions

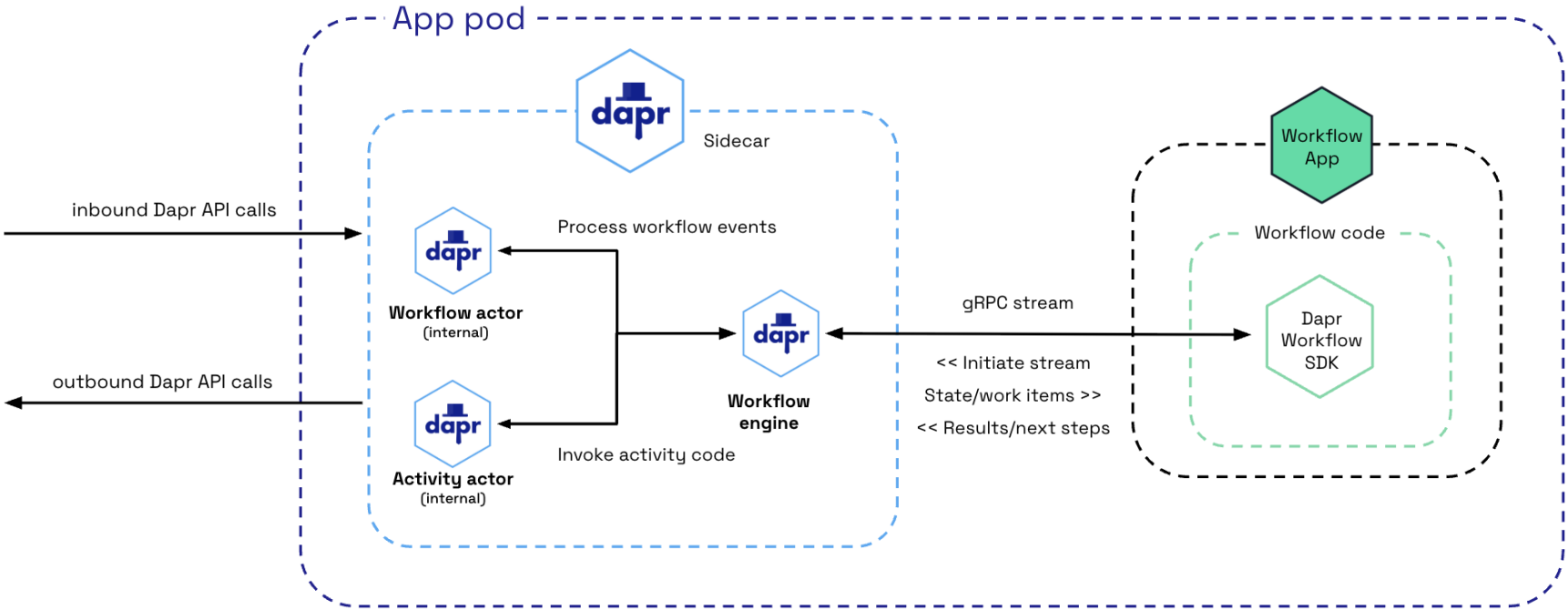
Actors



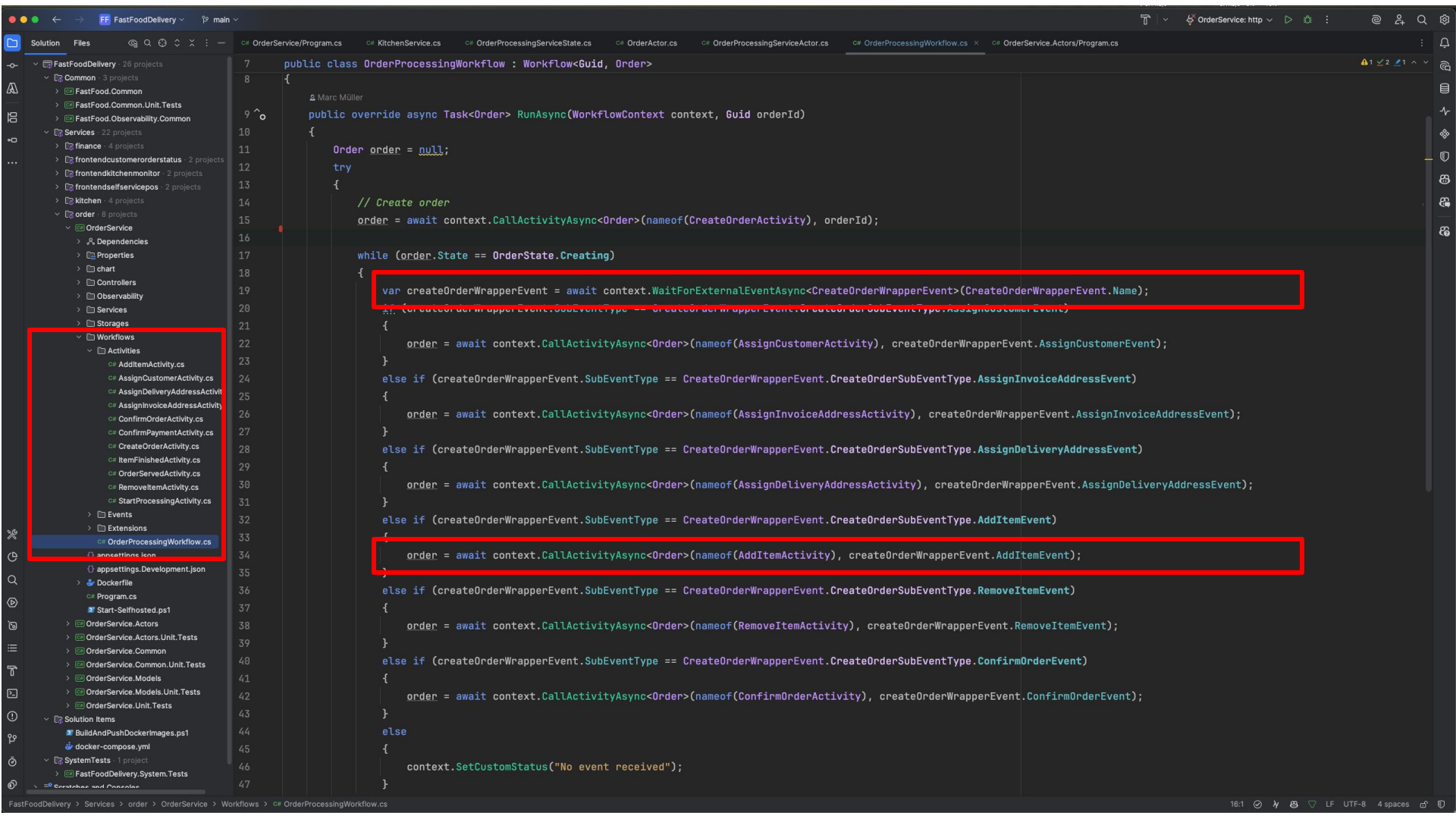
- Built-In workflow runtime
- Activities execute custom logic
- External event waiters
- Child Workflows
- Durable timers

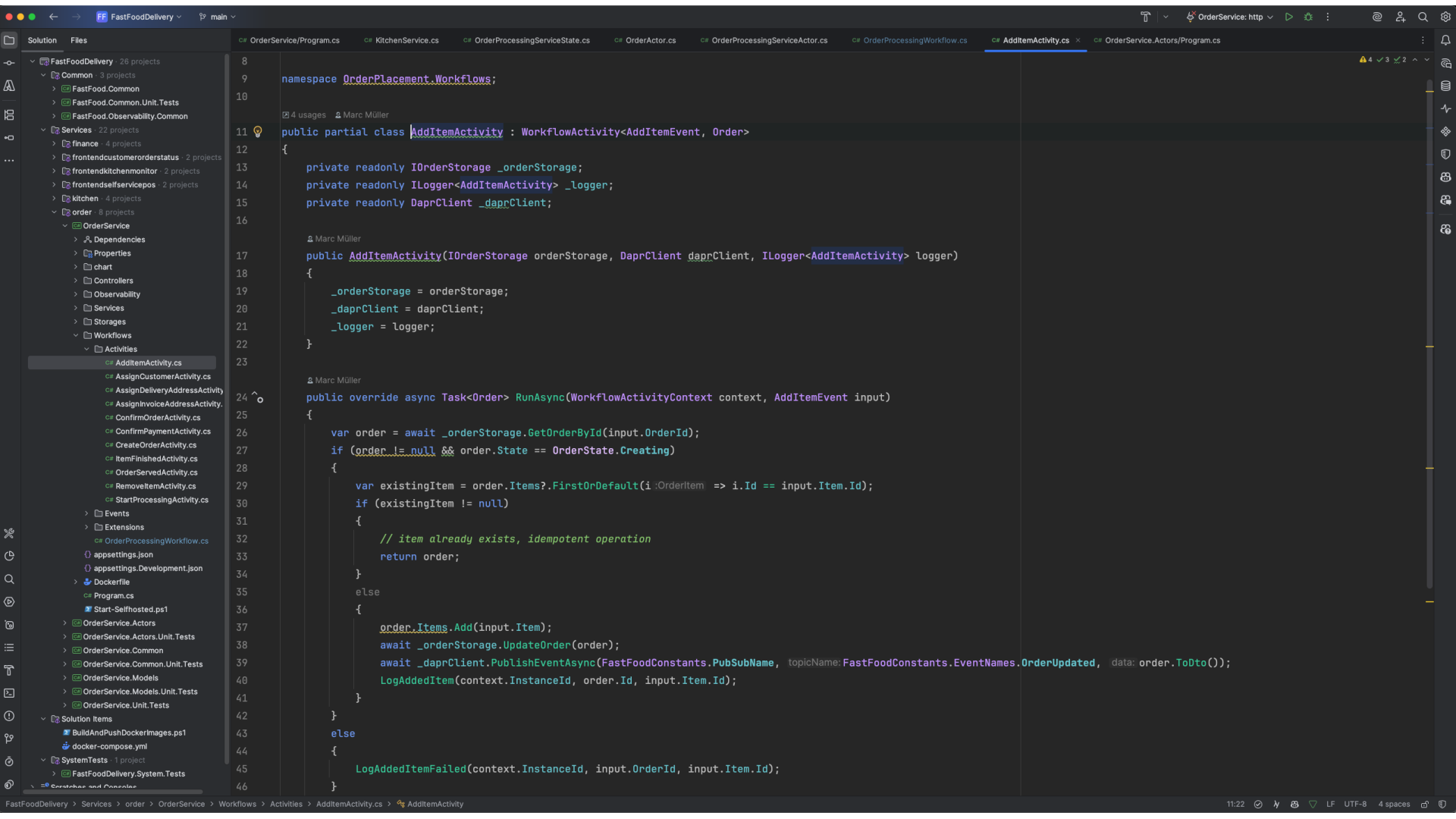
https://docs.dapr.io/developing-applications/building-blocks/workflow/workflow-overview/

Workflows the hood




DEMO
Virtual Actors


FastFoodDelivery
main
OrderService: http
Solution
Files
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
frontendcustomerorderstatus · 2 projects
frontendkitchenmonitor · 2 projects
frontendselfservicepos · 2 projects
kitchen · 4 projects
order · 8 projects
OrderService
Dependencies
Properties
chart
Controllers
Observability
Services
Storages
Workflows
Activities
AddItemActivity.cs
AssignCustomerActivity.cs
AssignDeliveryAddressActivit
AssignInvoiceAddressActivity
ConfirmOrderActivity.cs
ConfirmPaymentActivity.cs
CreateOrderActivity.cs
ItemFinishedActivity.cs
OrderServedActivity.cs
RemoveItemActivity.cs
StartProcessingActivity.cs
Events
Extensions
OrderProcessingWorkflow.cs
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
public class OrderProcessingWorkflow : Workflow<Guid, Order>
{
Marc Müller
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);
{
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 › OrderProcessingWorkflow.cs
16:1
LF
UTF-8
4 spaces



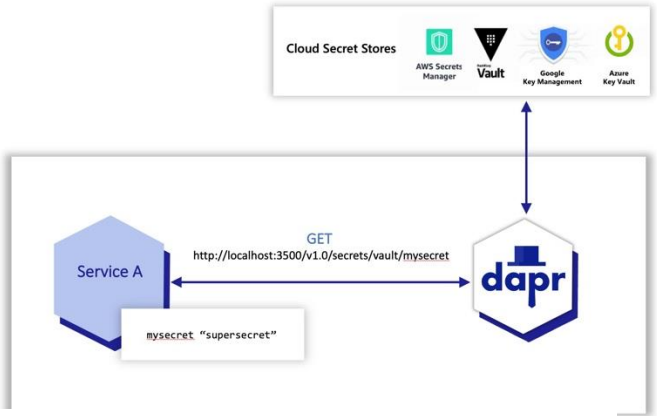

```
namespace OrderPlacement.Workflows;

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

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

    Marc Müller
    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 :OrderItem => 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);
        }
```

Dapr Building Blocks

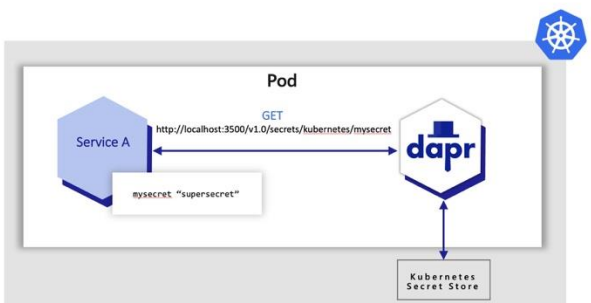
Secret Management

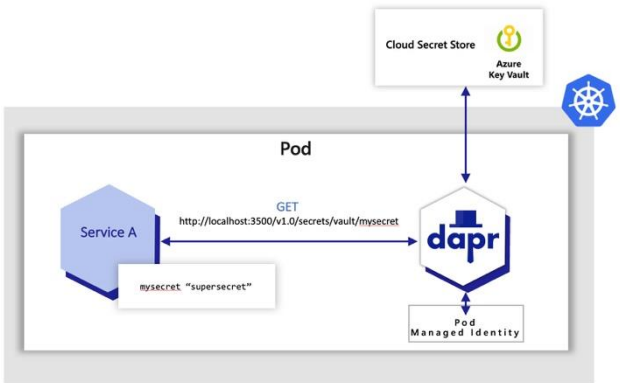
4tecture
empower your software solutions

Secret Management



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





https://docs.dapr.io/developing-applications/building-blocks/secrets/secrets-overview/

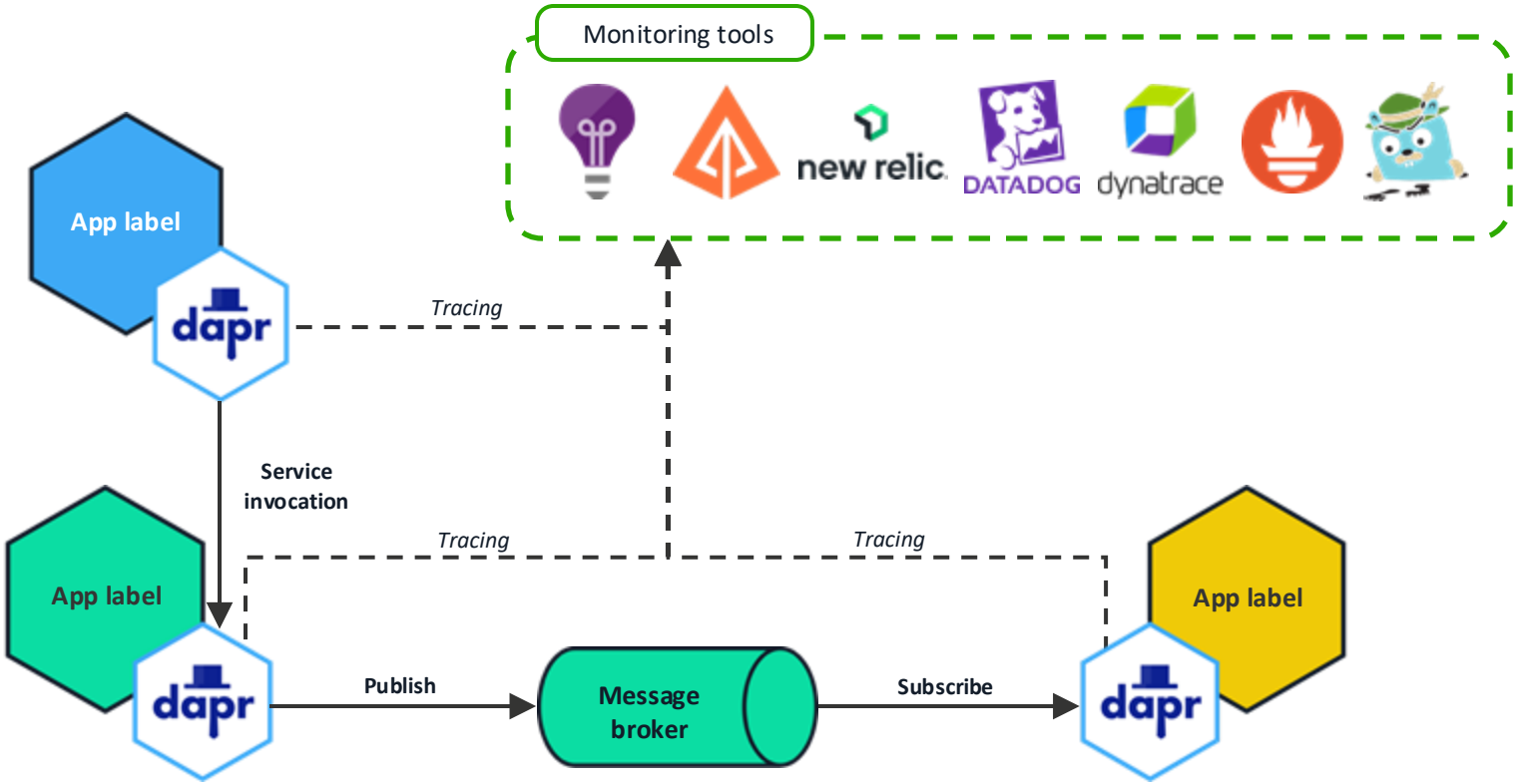
Dapr Building Blocks

Observability

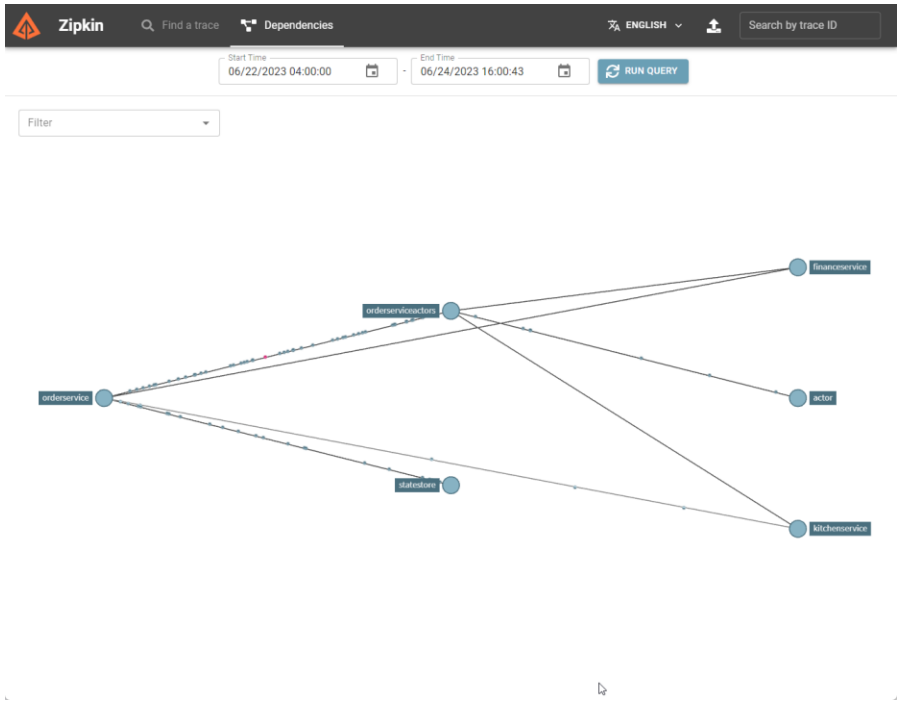
4tecture
empower your software solutions

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

https://docs.dapr.io/developing-applications/building-blocks/observability/tracing-overview/

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.

```yaml
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.)


.NET Microservices with Dapr
Q & A
4tecture
empower your software solutions

Thank you for your attention!

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

4tecture GmbH
Industriestrasse 25
CH-8604 Volketswil
www.4tecture.ch

Marc Müller
Principal Consultant

www.powerofdevops.com

4tecture©
empower your software solutions

ProAuth

ReaFx


4tecture©
empower your software solutions