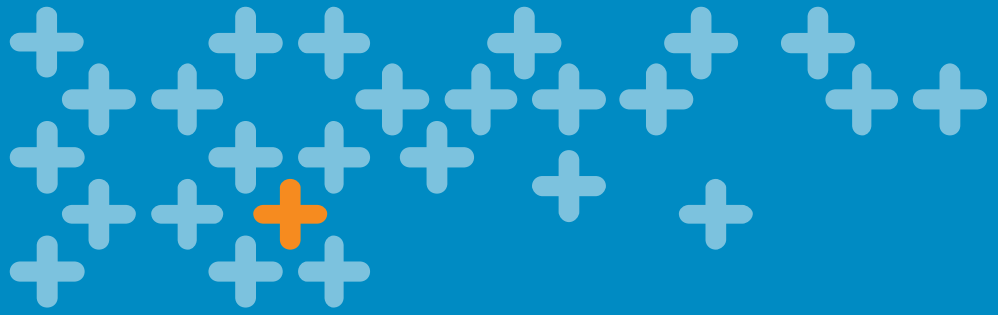


Building Microservices with Dapr

Jakob Ehn

@jakobehn

https://blog.ehn.nu

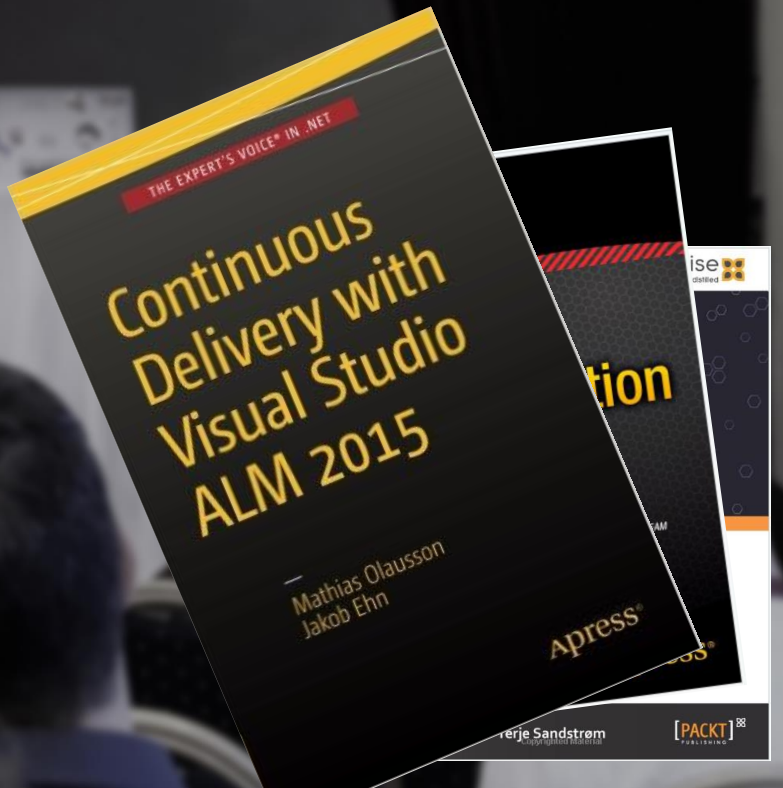
jakob.ehn@activesolution.se

active SOLUTION

Jakob Ehn
Microsoft Azure MVP

https://blog.ehn.nu

@jakobehn

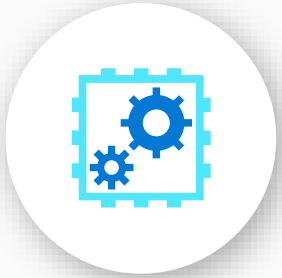


MVP Microsoft® Most Valuable Professional

What’s hard about microservices?

Need to support lots of different integration points

(cache, message queues, 3rd party APIs, secret stores)



This results in coupling in your code.

Coupled to specific service and SDK

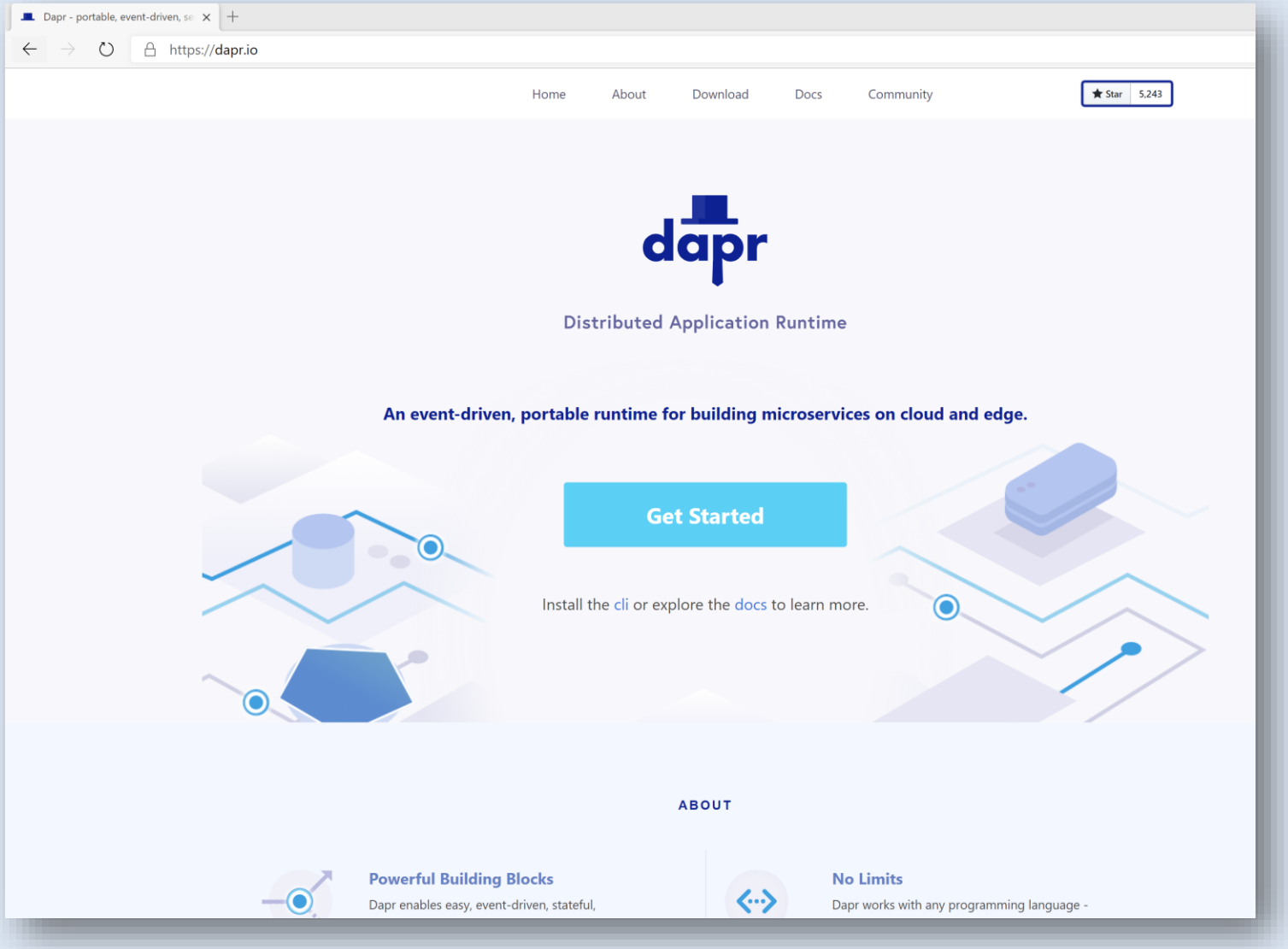
Lots of different targets to support tracing, configuration, and secret management

Need to handle things like transient failures and distributed tracing

Distributed Application Runtime

Portable, event-driven, runtime for building distributed applications across cloud and edge

https://dapr.io

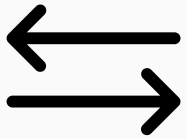


Dapr Goals

Best-Practices
Building Blocks

Any Language
or Framework

Community Driven
Vendor Neutral

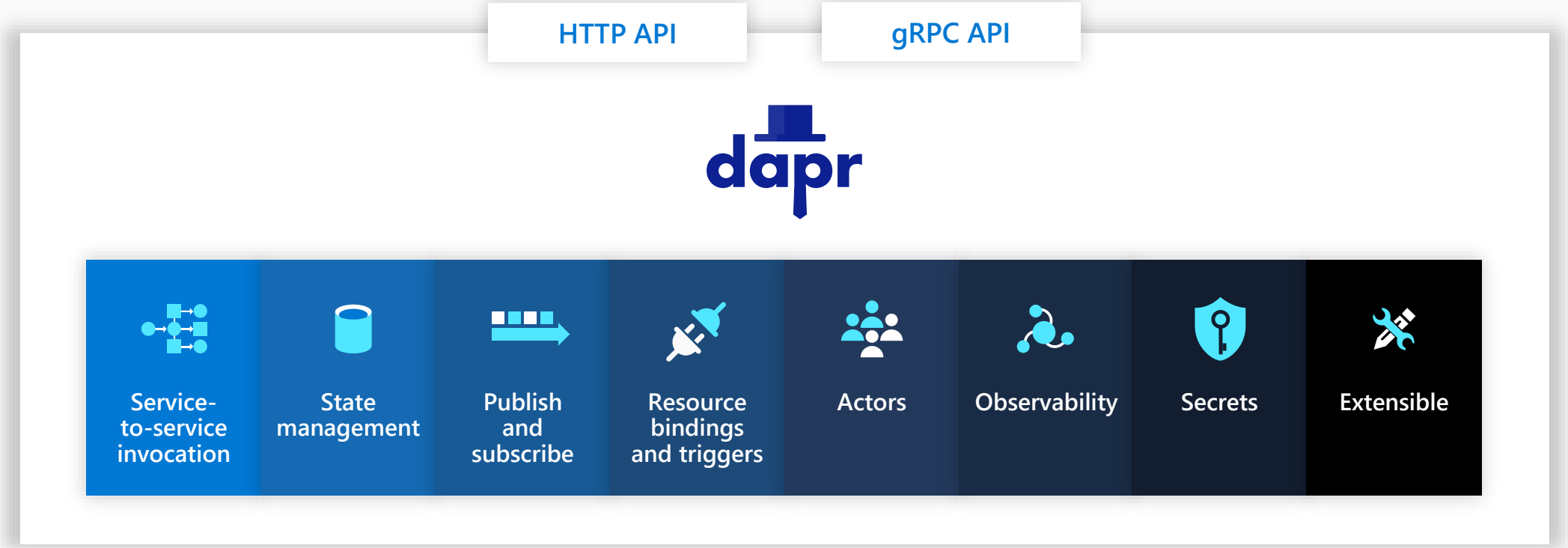


Consistent, Portable,
Open APIs

Platform Agnostic
Cloud + Edge

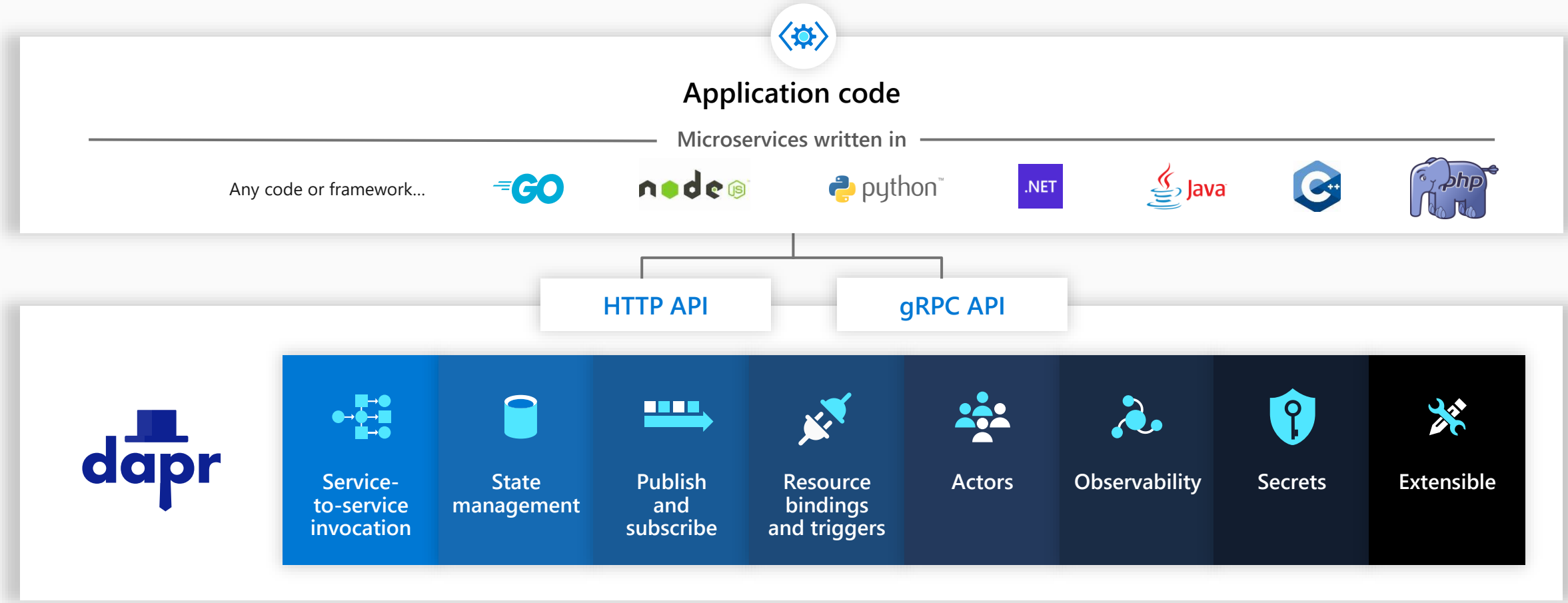
Extensible and
Pluggable Components

Microservice building blocks

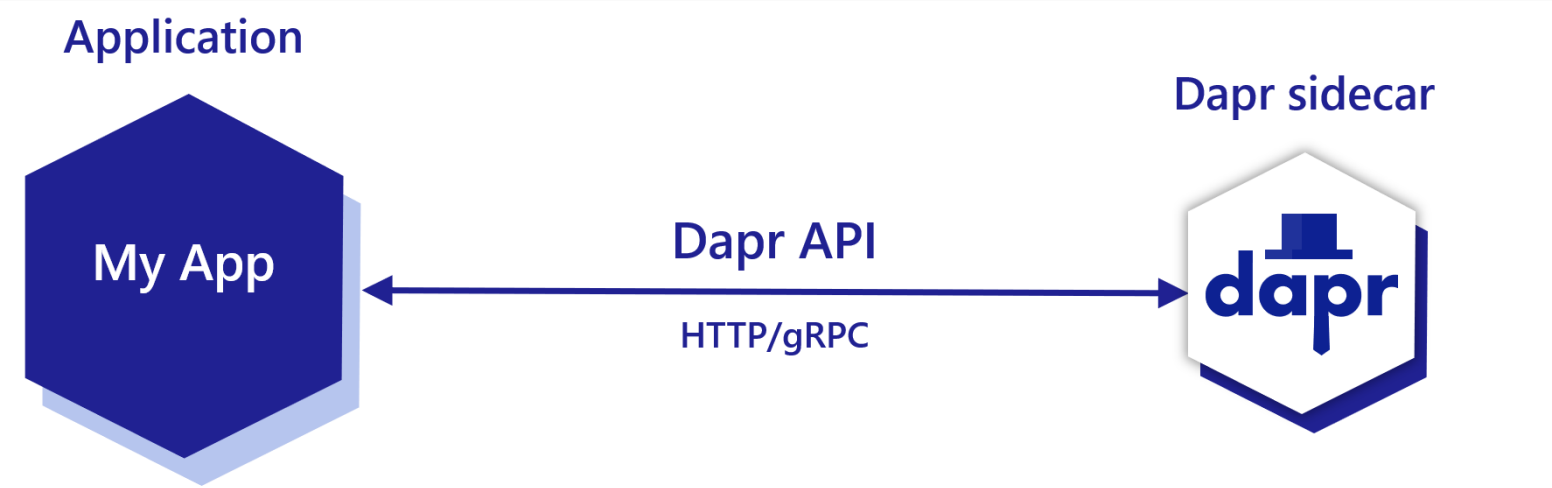


Microservice building blocks

- Standard APIs accessed over HTTP/gRPC protocols from user service code
  http://localhost:3500/v1.0/invoke/cart/method/neworder
  http://localhost:3500/v1.0/state/inventory/item67
- Runs as local “side car library” dynamically loaded at runtime for each service

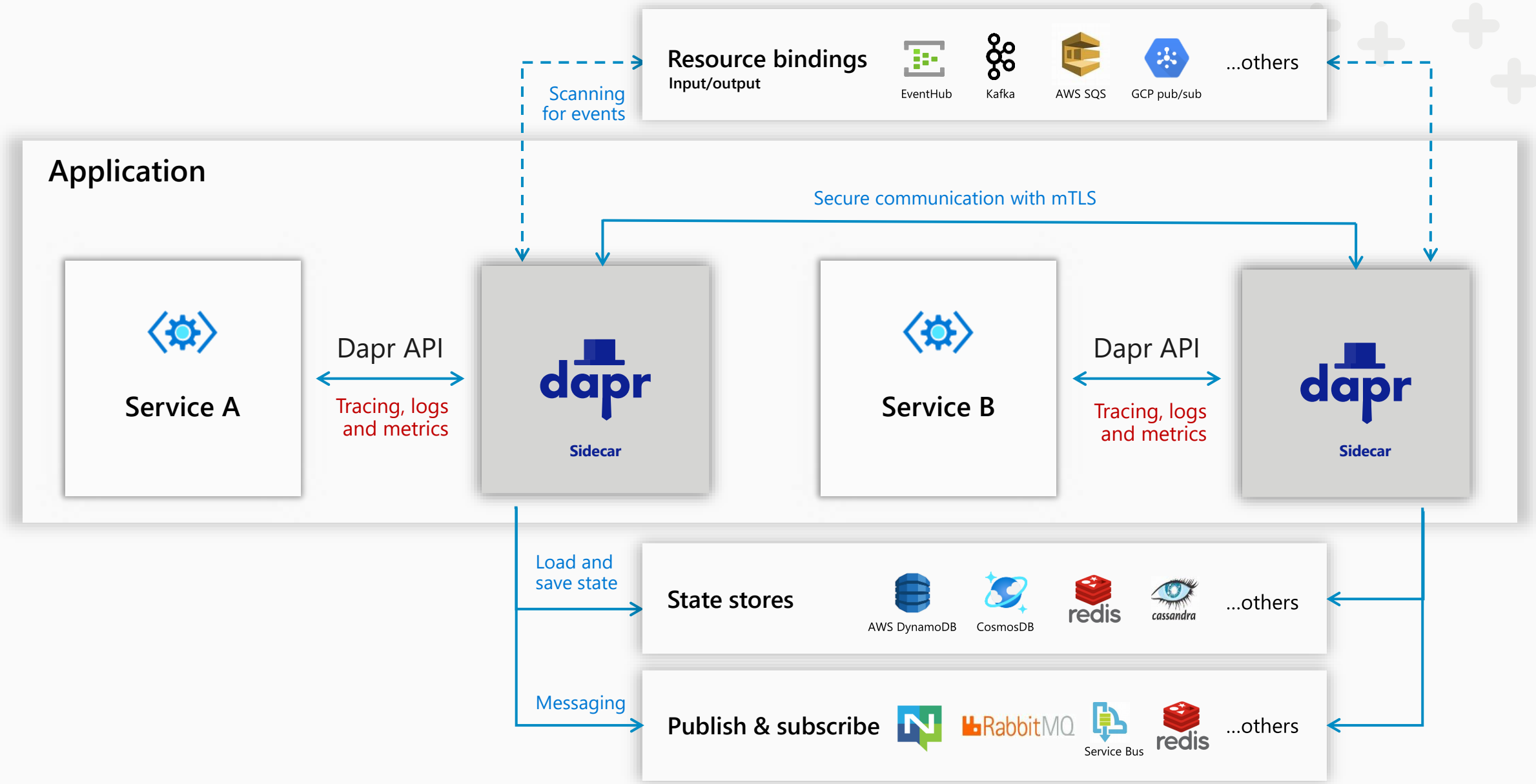


Dapr Sidecar Model

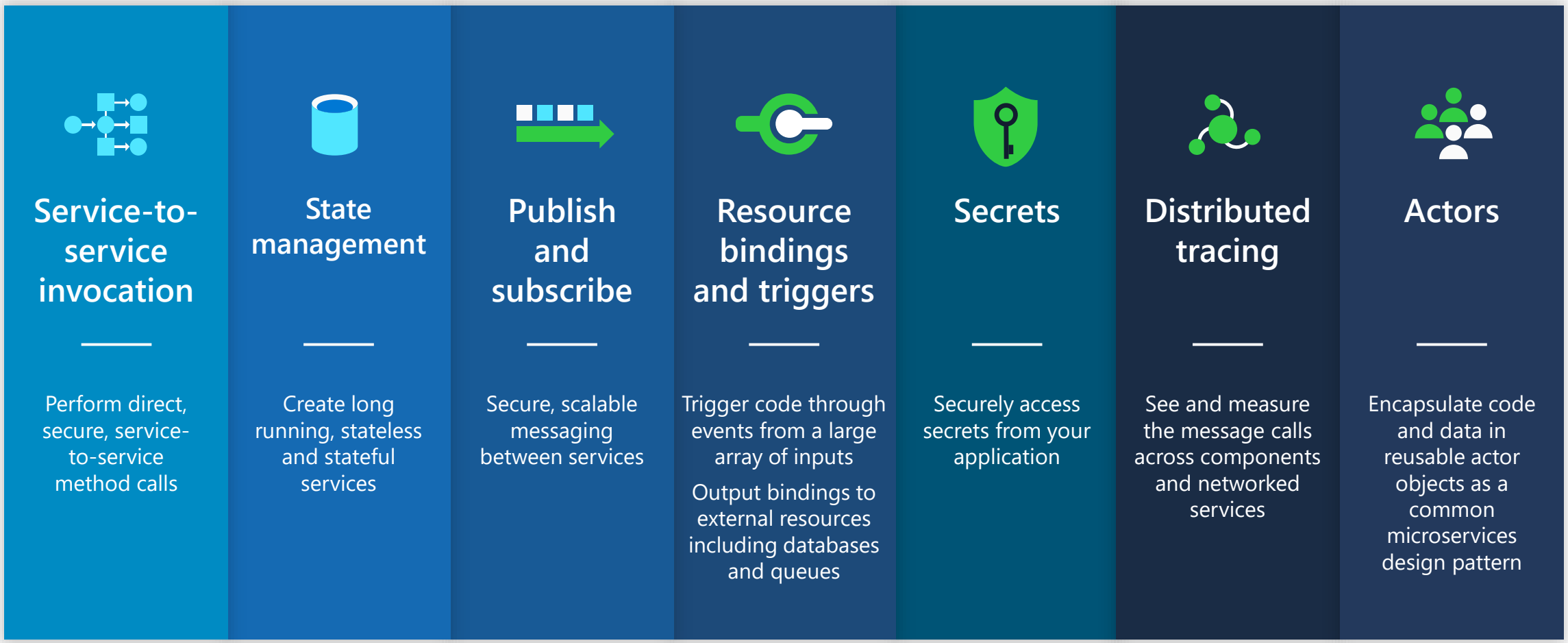



```
POST http://localhost:3500/v1.0/invoke/cart/method/neworder

GET http://localhost:3500/v1.0/state/inventory/item67

POST http://localhost:3500/v1.0/publish/shipping/orders

GET http://localhost:3500/v1.0/secrets/keyvault/password
```

Sidecar and component architecture



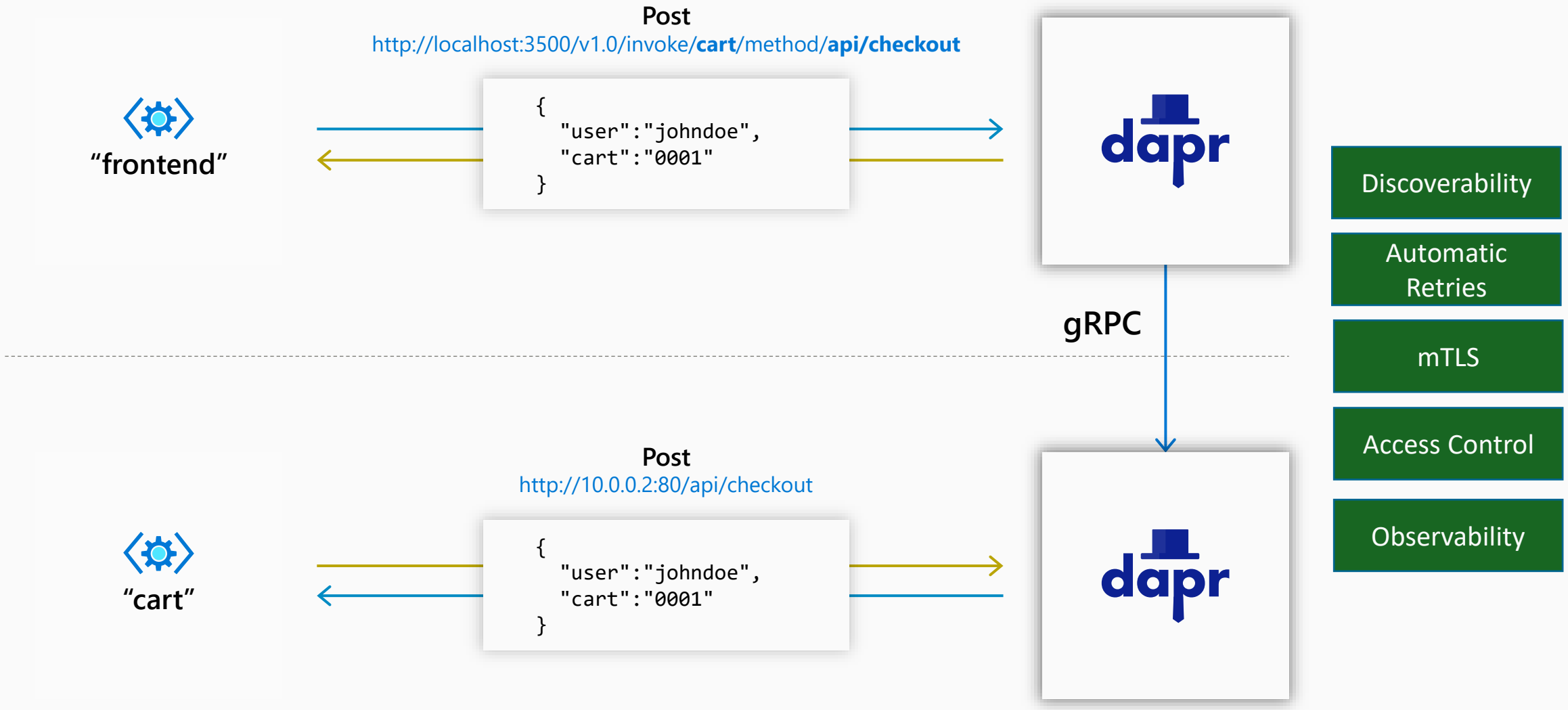
Microservice building blocks



Use Dapr components

Microservice building blocks

Service invocation



Demo

- ”Daprize” an application
- Service invocation

Microservice building blocks

State management: key/value

Optimistic concurrency

Automatic Retries

State transactions

App "myApp"

Get

http://localhost:3500/v1.0/state/<**store-name**>**/planet**

```
{
   "name": "Tatooine"
}
```

Post

http://localhost:3500/v1.0/state**/<store-name>**

```
[{
    "key": "weapon",
    "value": "DeathStar"
}, {
    "key": "planet",
    "value": {
        "name": "Tatooine"
    }
}]
```

dapr

key	value
myApp-weapon	"DeathStar"
myApp-planet	{ "name": "Tatooine" }

State store of your choice

AWS DynamoDB, CosmosDB, redis, cassandra, ...others

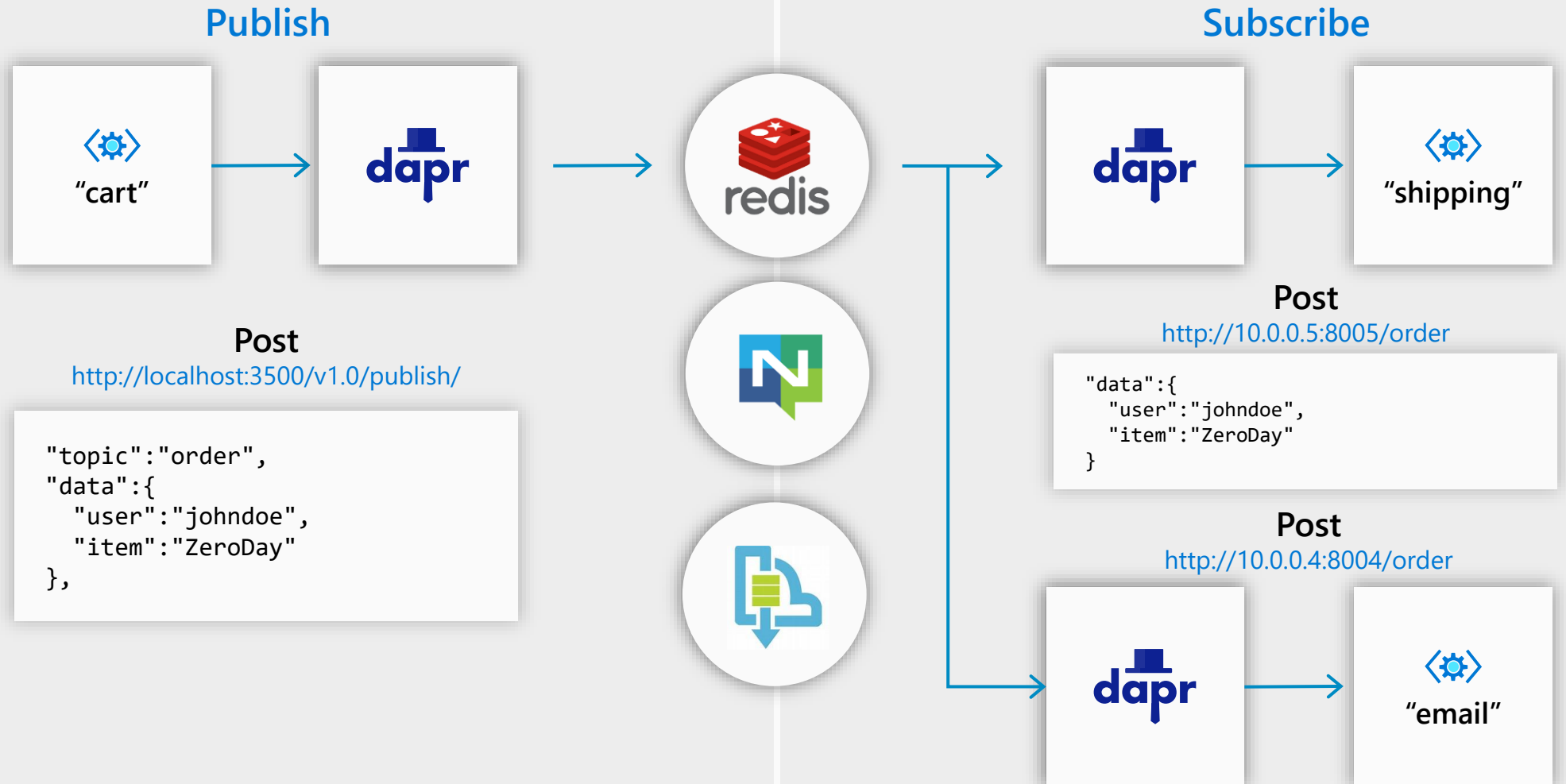
Microservice building blocks

Publish and subscribe

At-least-once guarantee

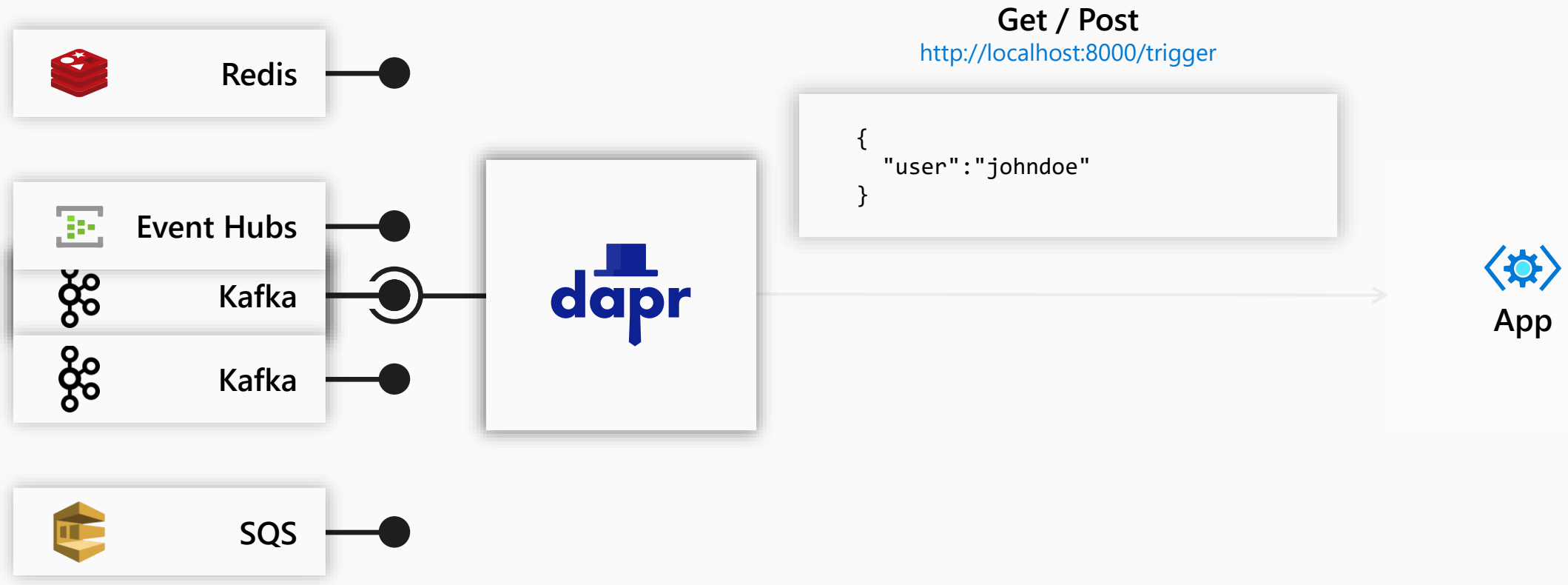
Competing consumer

Message Time-to-Live



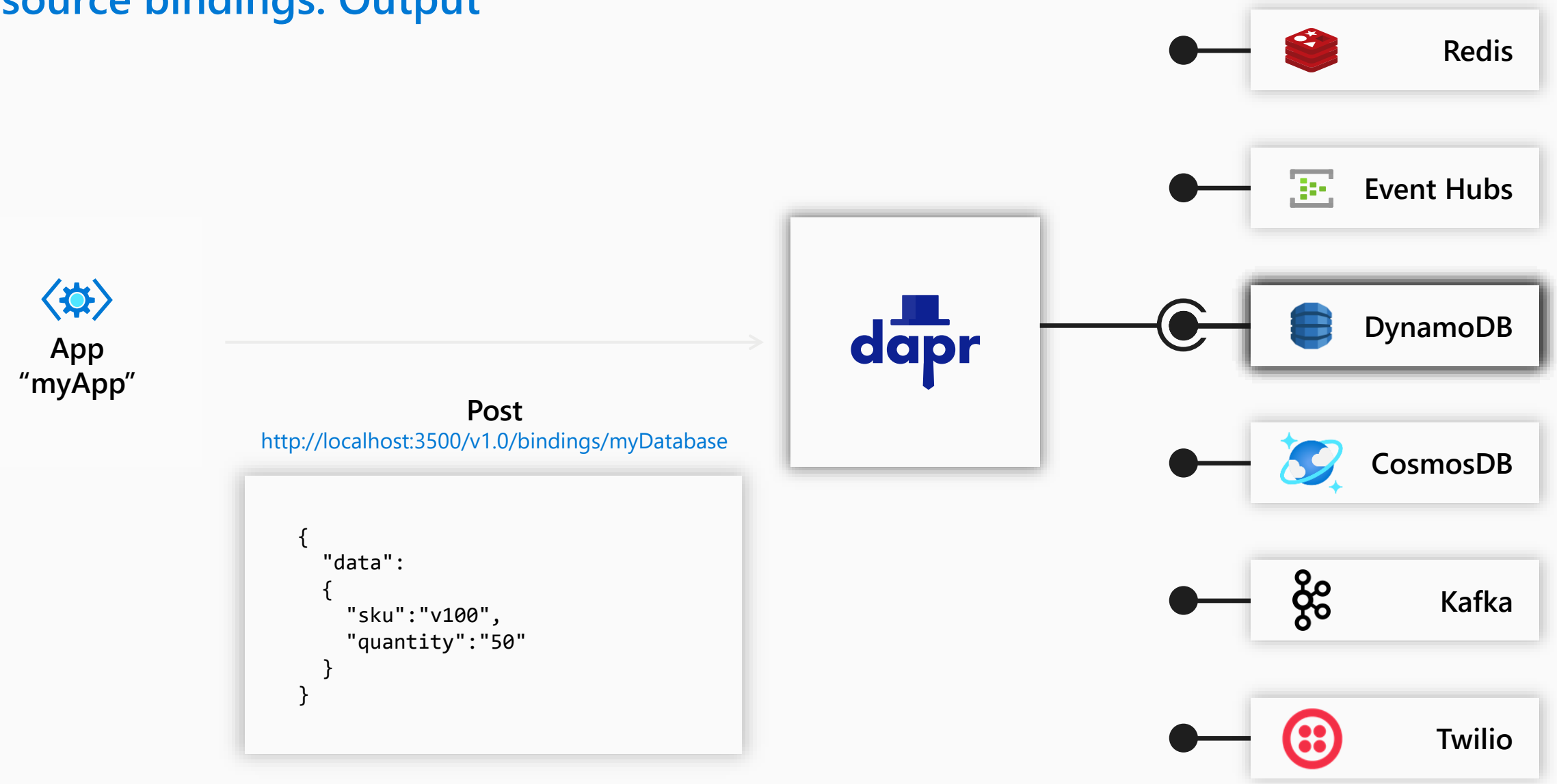
Microservice building blocks

Resource triggers: Input



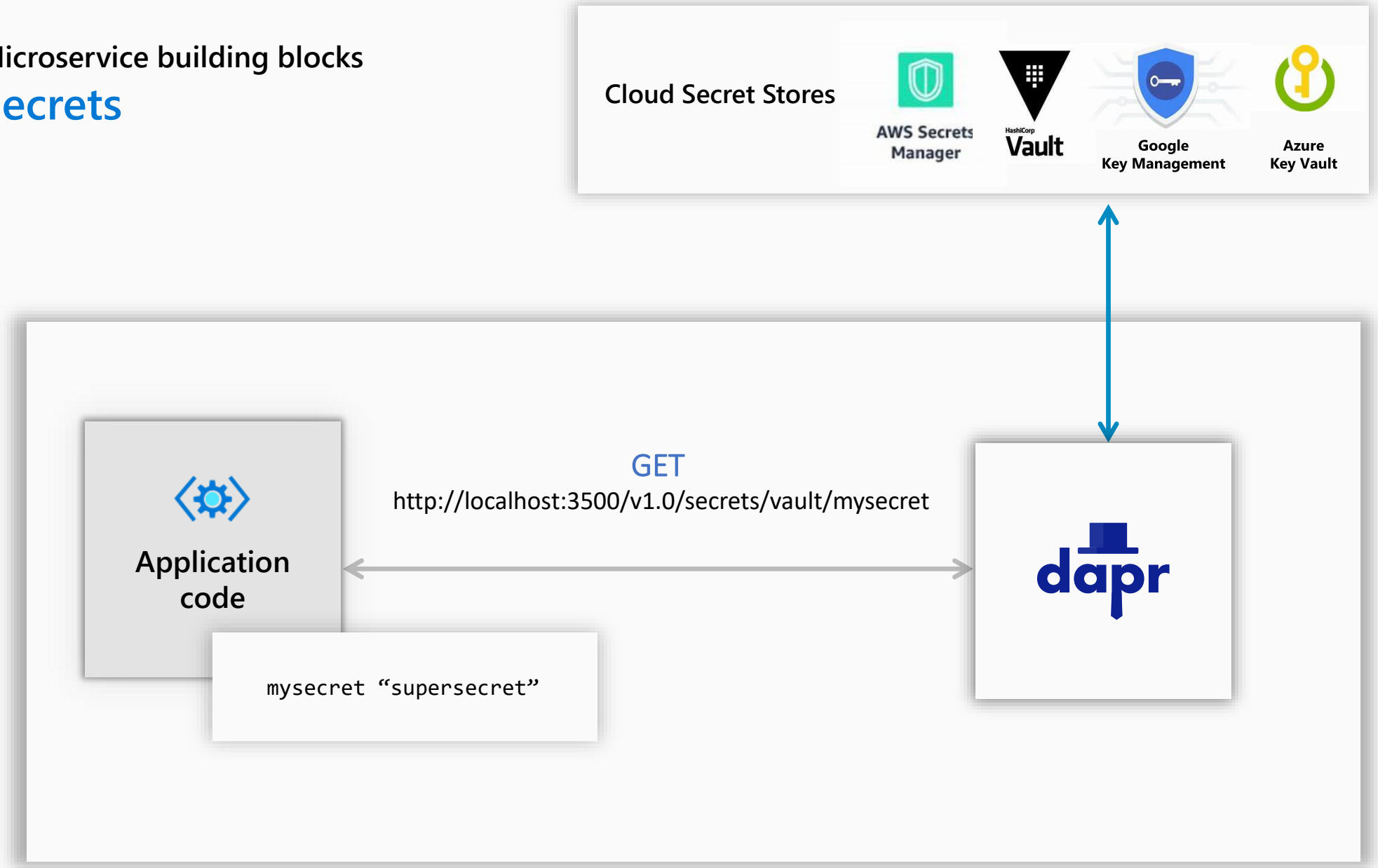
Microservice building blocks

Resource bindings: Output



Microservice building blocks

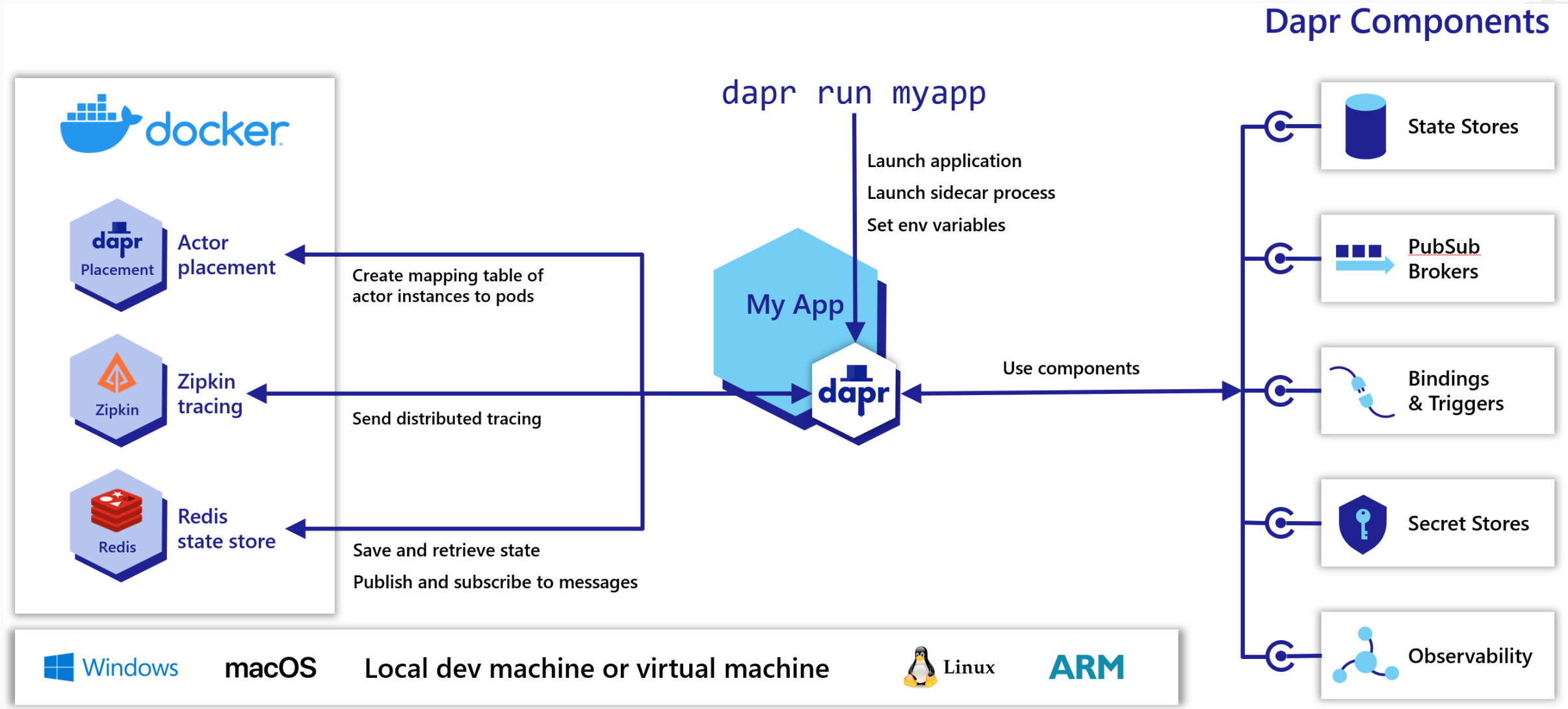
Secrets



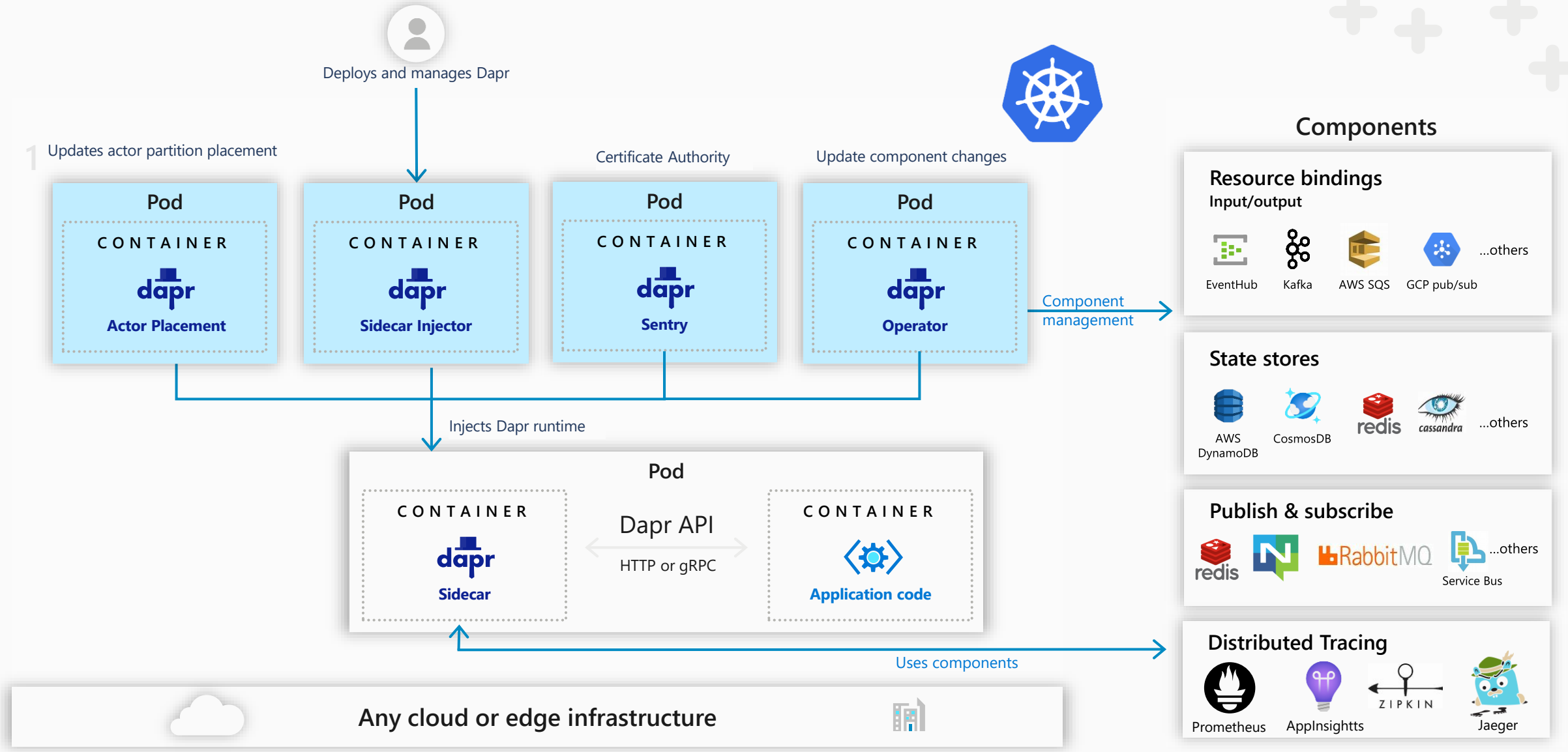
Demo

- State management
- Responding to events (Input bindings)
- Communicate with external resources (Output bindings)
- Secret management

Running Dapr Self-hosted



Dapr Kubernetes hosted



Running Dapr

Standalone (self-hosted)

```
dapr run --app-id my-app \
         --app-protocol grpc \
         --app-port 50105 \
         --components-path ./config \
         go run main.go
```

```
dapr run --app-id my-app \
         --app-protocol http \
         --app-port 3000 \
         --components-path ./config \
         dotnet run
```

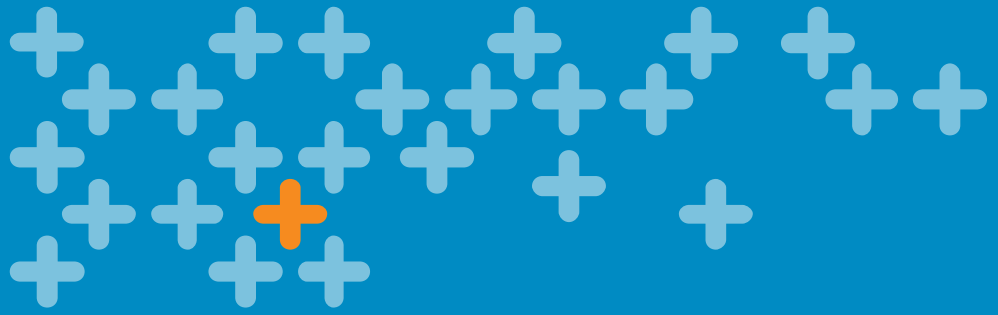
```
dapr run --app-id my-app \
         --app-protocol http \
         --app-port 5678 \
         --components-path ./config \
         ./my-exe
```

Kubernetes

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  labels:
    app: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
      annotations:
        dapr.io/enabled: "true"
        dapr.io/app-id: "my-appr"
        dapr.io/app-protocol: ”http"
        dapr.io/app-port: ”8080”
…
```

Demo

- Publish/Subscribe
- Deploy to Kubernetes
- Distributed Tracing



Thank you

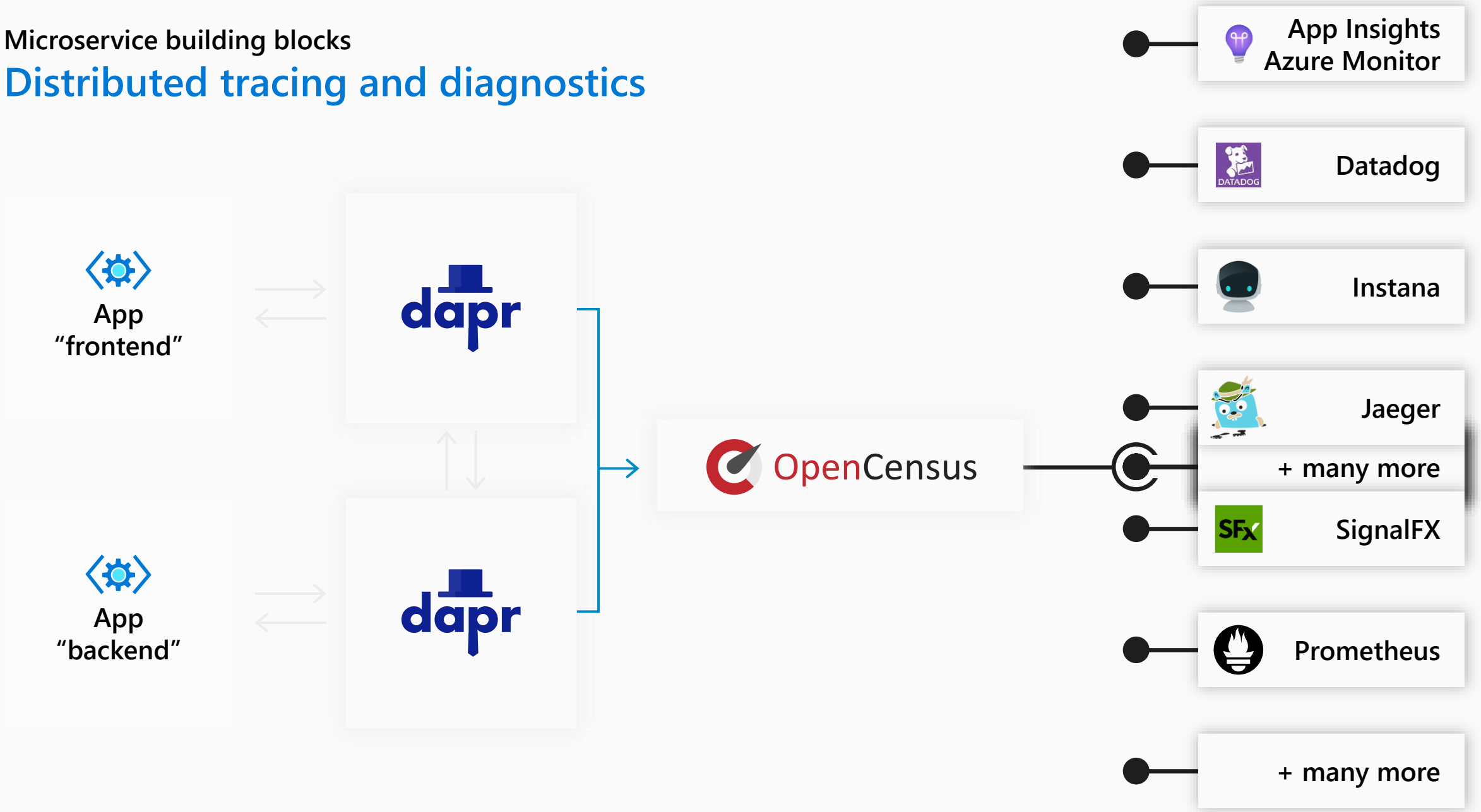
Jakob Ehn

@jakobehn

https://blog.ehn.nu

Microservice building blocks

Distributed tracing and diagnostics



Distributed Application Runtime

Build apps using any language with any framework

Dapr language SDKs

Application code

Microservices written in

Any code or framework...






HTTP API | gRPC API


Service-to- service invocation


State management


Publish and subscribe


Resource bindings and triggers


Actors


Distributed tracing


Secrets


Extensible

Any cloud or edge infrastructure








SDK Languages

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

Client SDK:
The Dapr client allows you to invoke Dapr building block APIs and perform actions such as:

- Invoke methods on other services
- Store and get state
- Publish and subscribe to message topics
- Interact with external resources through input and output bindings
- Get secrets from secret stores

Server extensions:
The Dapr service extensions allow you to create services that can:

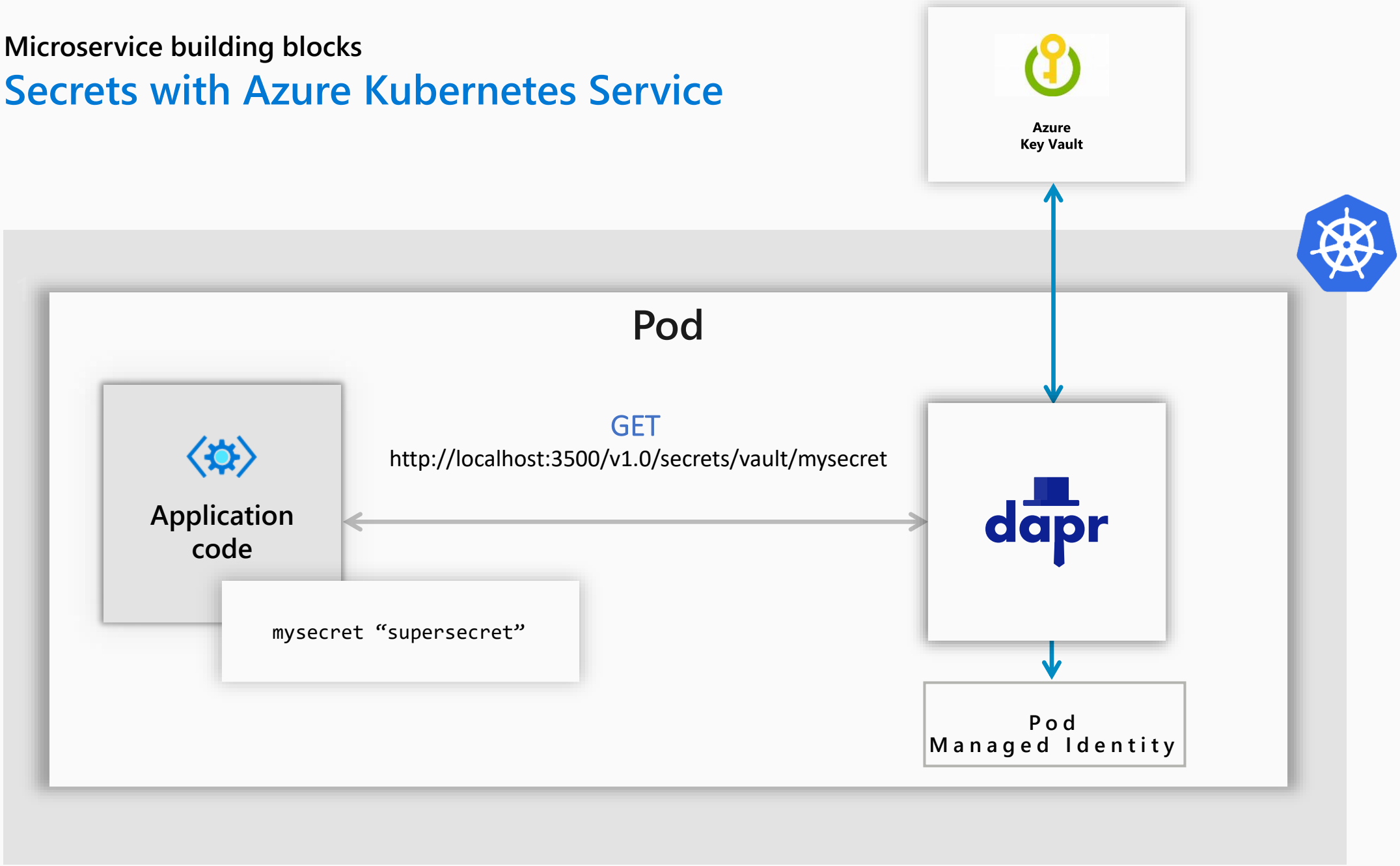
- Be invoked by other services
- Subscribe to topics

Actor SDK:
The Dapr Actor SDK allows you to build virtual actors with:

- Methods that can be invoked by other services
- State that can be stored and retrieved
- Timers with callbacks

Microservice building blocks

Secrets with Azure Kubernetes Service



Microservice building blocks

Secrets with Kubernetes