Microsoft

Microsoft Ignite

Microsoft

OAM, dapr, and rudr

The future of cloud native applications

Mark Russinovich

CTO, Microsoft Azure

@markrussinovich

BRK3098

Agenda

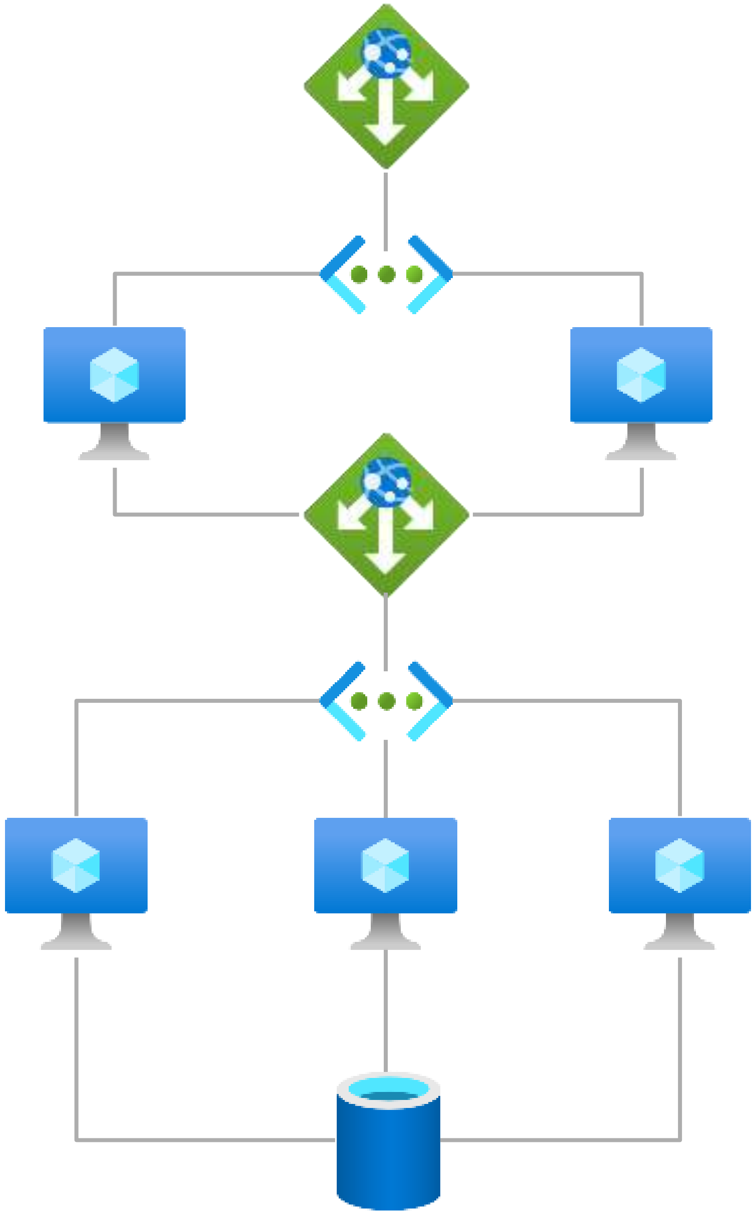
Open Application Model

dapr: Distributed Application Platform

Building Cloud Scale, Hybrid Applications

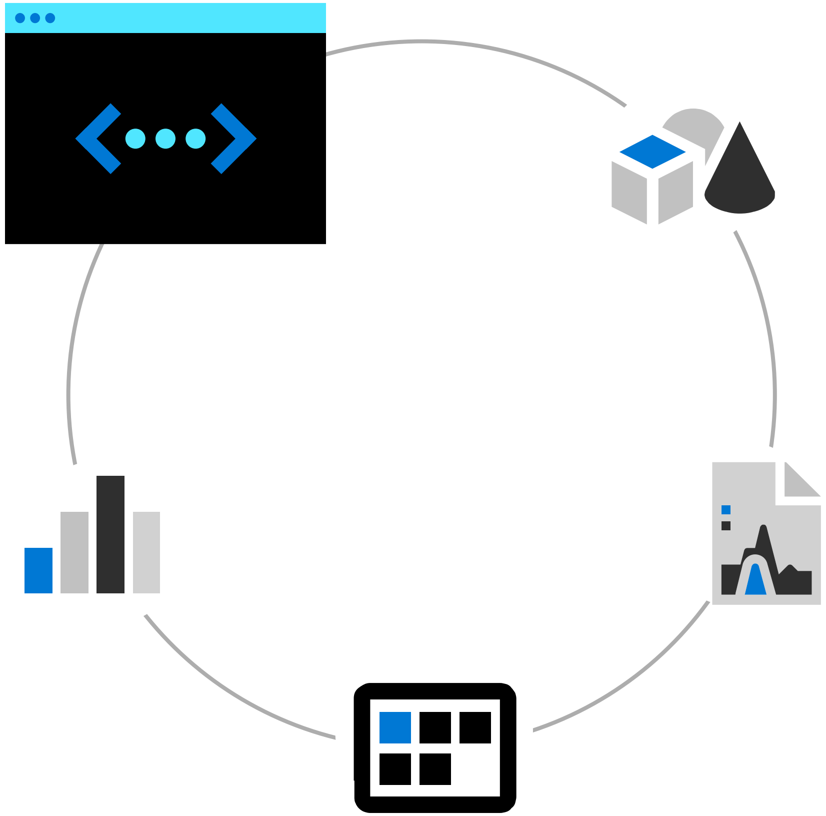
Application Models

Describes the topology of your application and its components



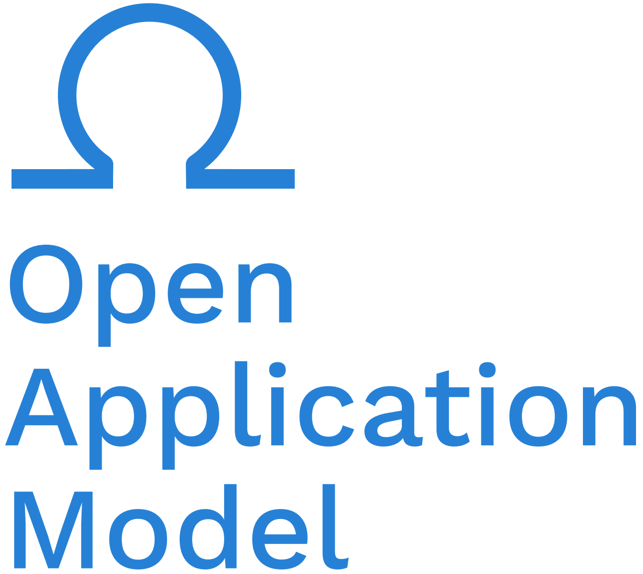
Programming Models

The way developers write their application to interact with other services and data stores



Open Application Model (OAM)

Platform agnostic application model



dapr: Distributed Application Runtime

Building blocks for building scalable distributed apps

Open Application Model

Application model for Cloud and Edge

State of Cloud Native Application Platforms

The cloud is going serverless, but K8s is the infrastructure on-prem and on-edge

App developers need to know and code for each infrastructure they deploy to

Kubernetes for applications

Kubernetes focuses on container infrastructure, not on applications

Application developers need to be experts in Kubernetes APIs

Production use of Kubernetes requires mastery of the broader cloud-native ecosystem

"[Kubernetes] is really hard to get into it and understand how all the parts play together, even for experienced people."

– Software Architect @ Crisp

"A key principle for us when it comes to choosing a platform is that we can maintain the size of our team."

– CTO @ Handled.io

OAM: Platform agnostic application model

The open application model for cloud and edge

Application focused

Focuses on developers and applications, not on container infrastructure

Separation of concerns

Clearly defined roles for application developers, application operators, and infrastructure operators

Cloud + Edge

Standard and consistent application model for cloud, on-prem, and small-edge devices

Application focused

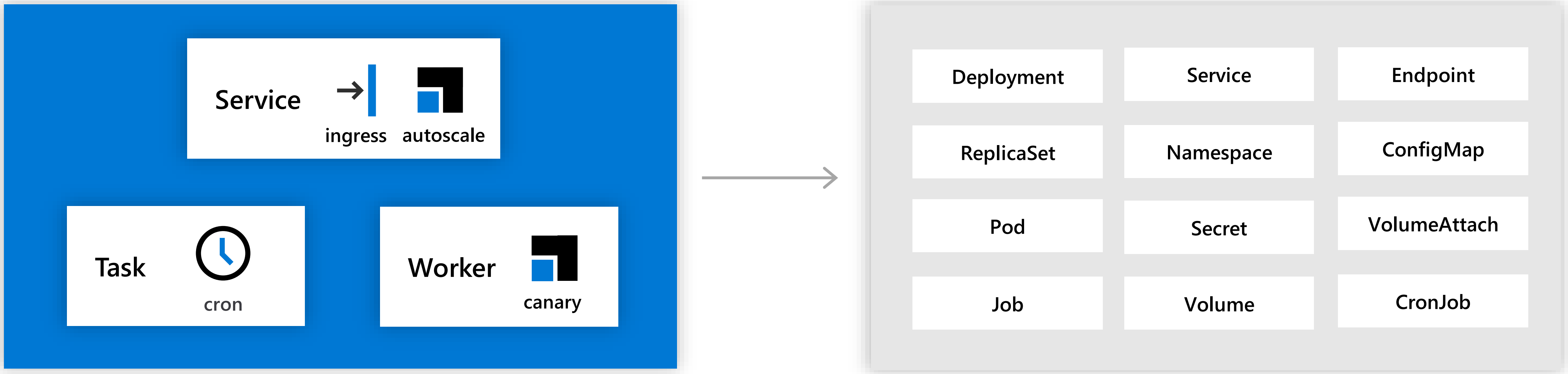
Describes application components and operations as first-class concepts without having to stitch together individual container primitives

Flexible application modeling supports a wide range of application architectures

Small and simple applications are easy, large and complex applications are manageable

Open Application Model

Container infrastructure



Service
ingress
autoscale

Task
cron

Worker
canary

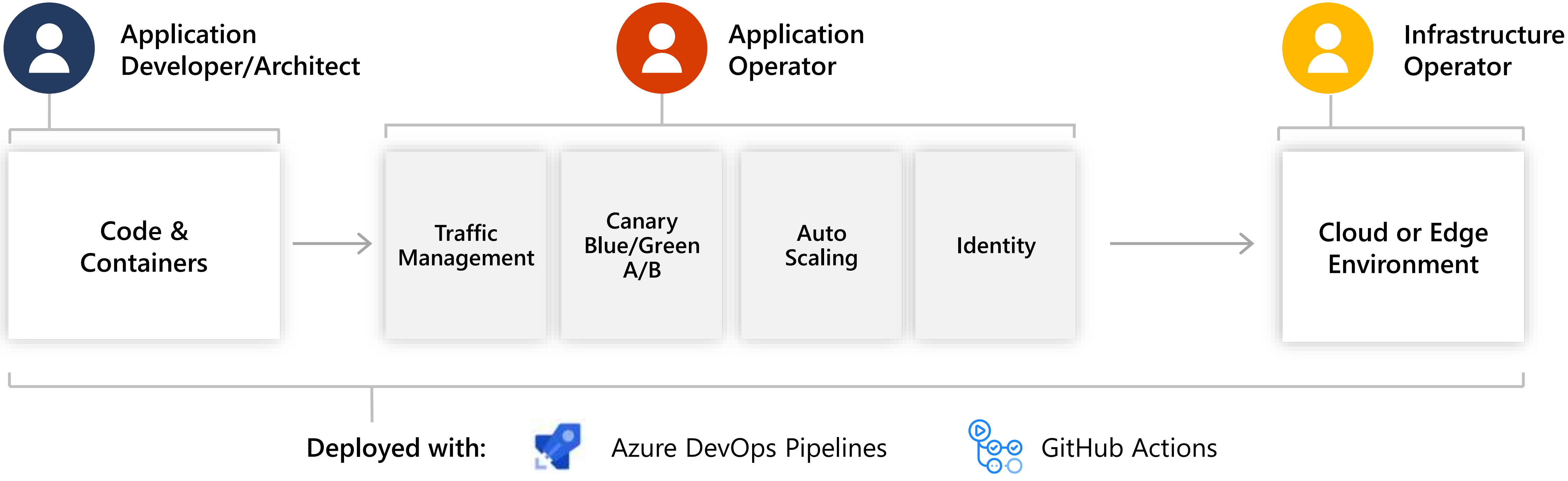
Deployment
Service
Endpoint
ReplicaSet
Namespace
ConfigMap
Pod
Secret
VolumeAttach
Job
Volume
CronJob

Separation of concerns

Allows application developers to focus on their code in a platform-neutral setting to deliver business value

Application operators use powerful and extensible operational traits consistently across platforms and environments

Infrastructure operators can configure their environments to satisfy any unique operating requirements

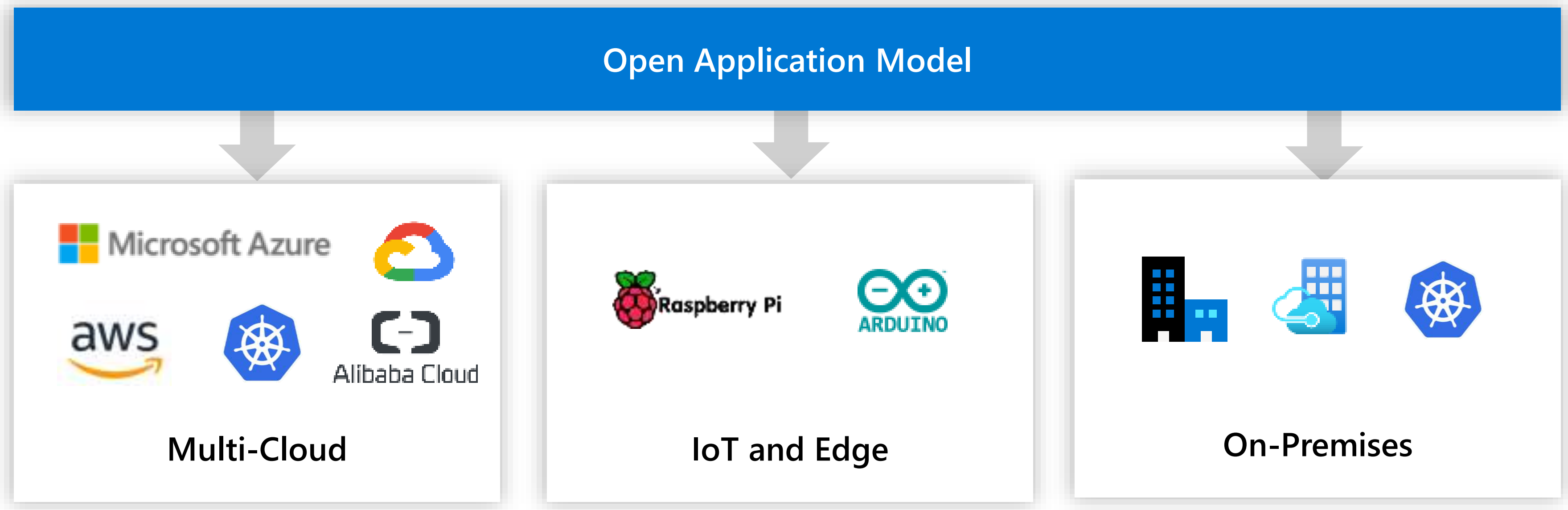


Cloud + Edge

A standard, platform-agnostic application definition for any platform in any environment.

Consistent application modeling for small devices, Kubernetes on prem or cloud, and fully-managed cloud environments.

Extendable by design to leverage the native APIs, tools, and unique features of platforms that users know and love



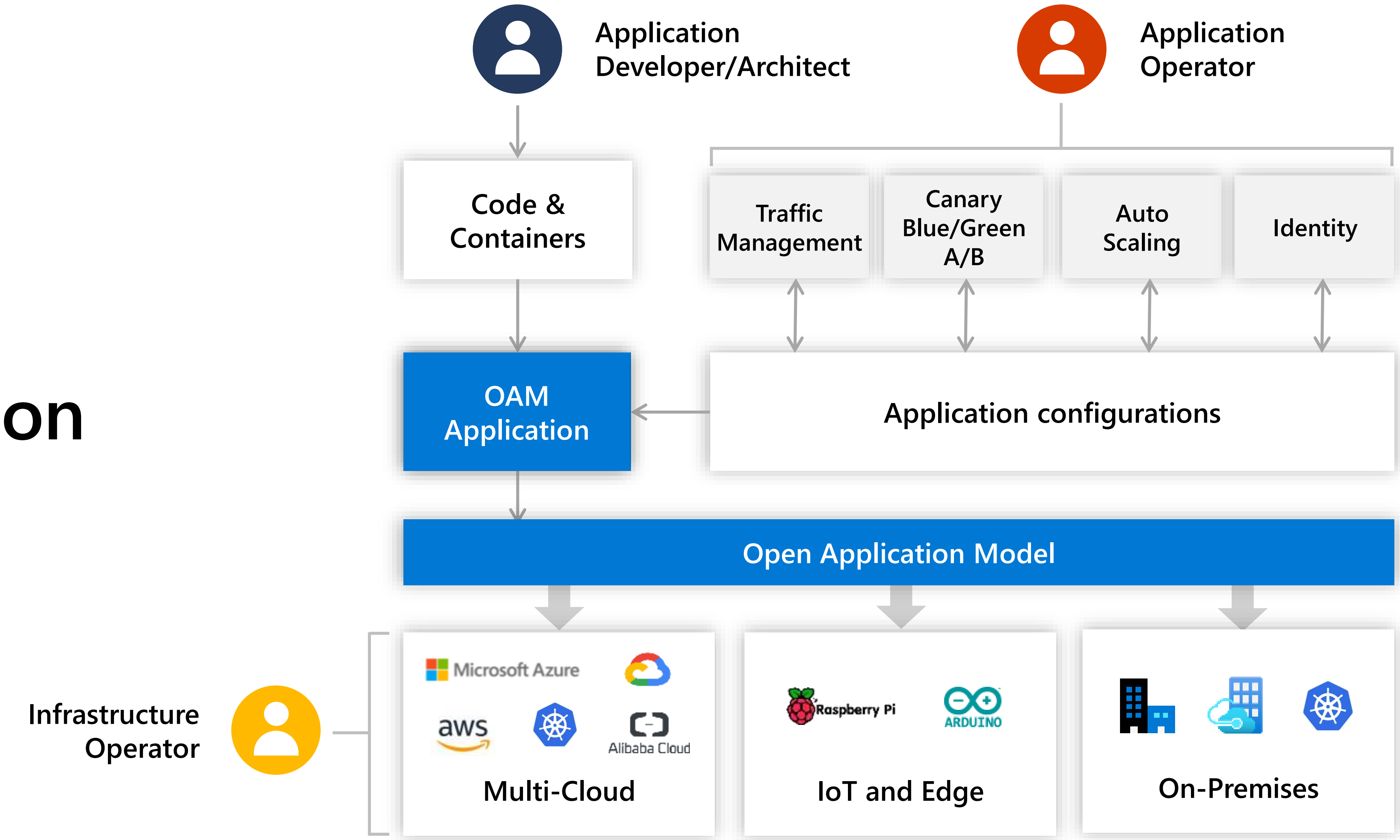
Open Application Model

Multi-Cloud

IoT and Edge

On-Premises

Open Application Model



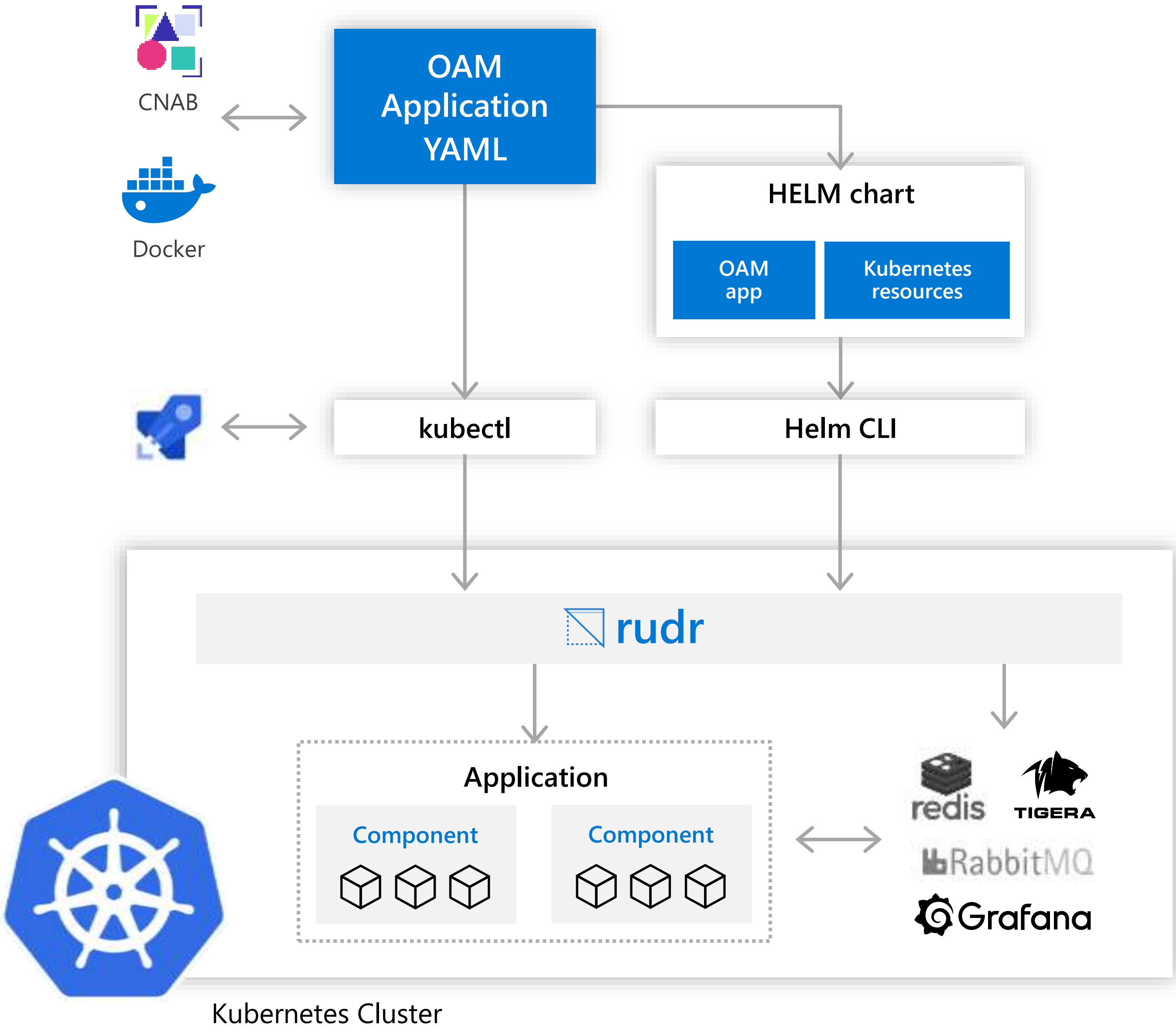
rudr: Open Application Model on Kubernetes

Build and operate cloud-native applications on the leading open source orchestrator

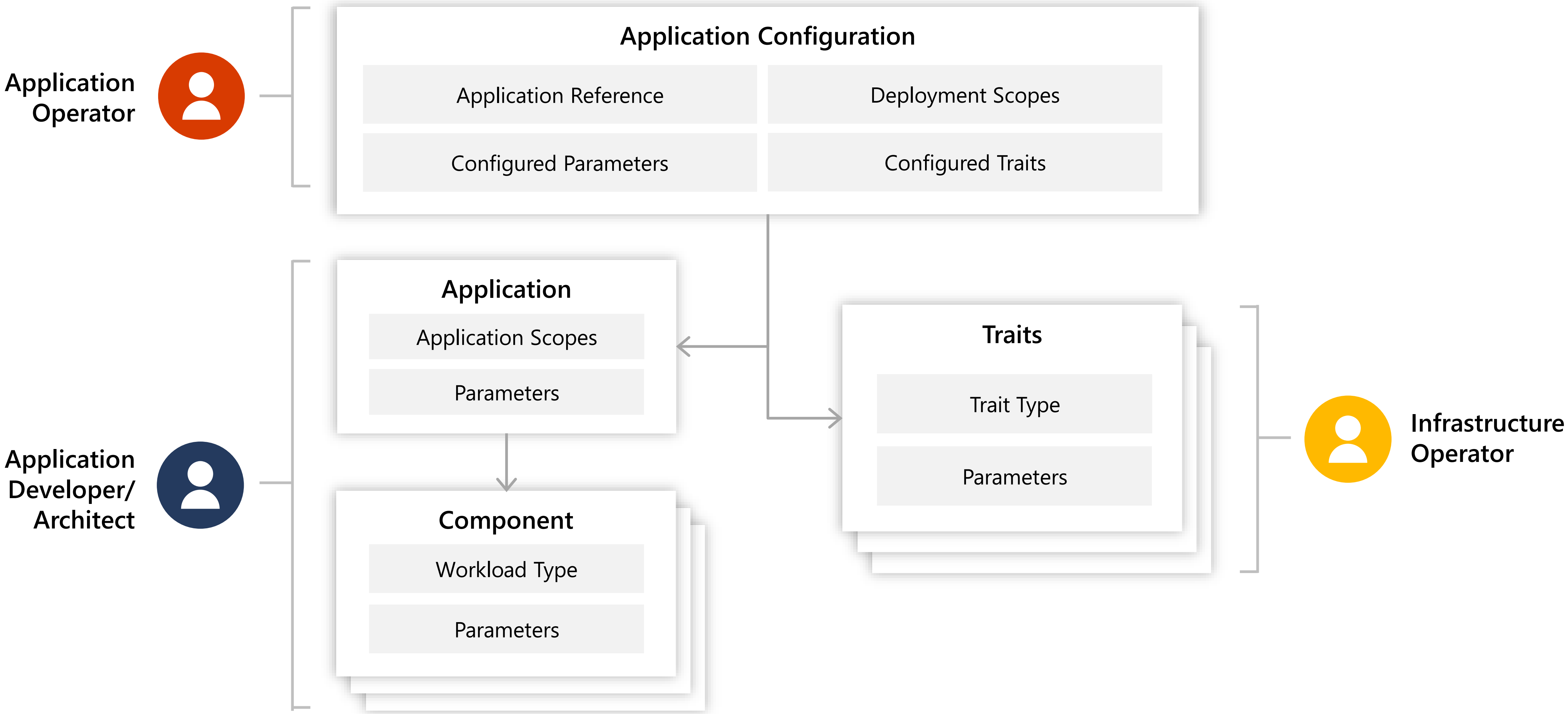
Application developers can focus on business value, not on container primitives and plumbing

CRDs combine high-level application modeling with familiar Kubernetes concepts

Infra operators continue to use familiar Kubernetes infrastructure, APIs, and domain knowledge



Open Application Model



Application Operator

Application Configuration

Application Reference

Deployment Scopes

Configured Parameters

Configured Traits

Application Developer/ Architect

Application

Application Scopes

Parameters

Component

Workload Type

Parameters

Traits

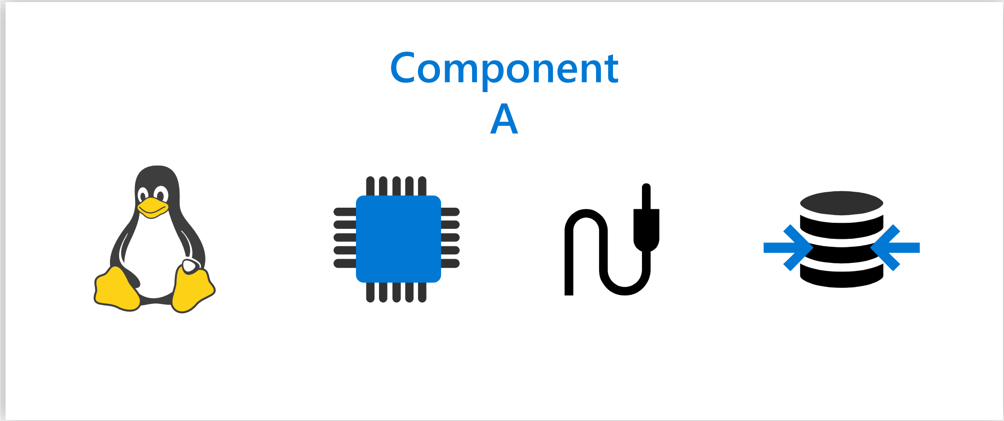
Trait Type

Parameters

Infrastructure Operator

Component

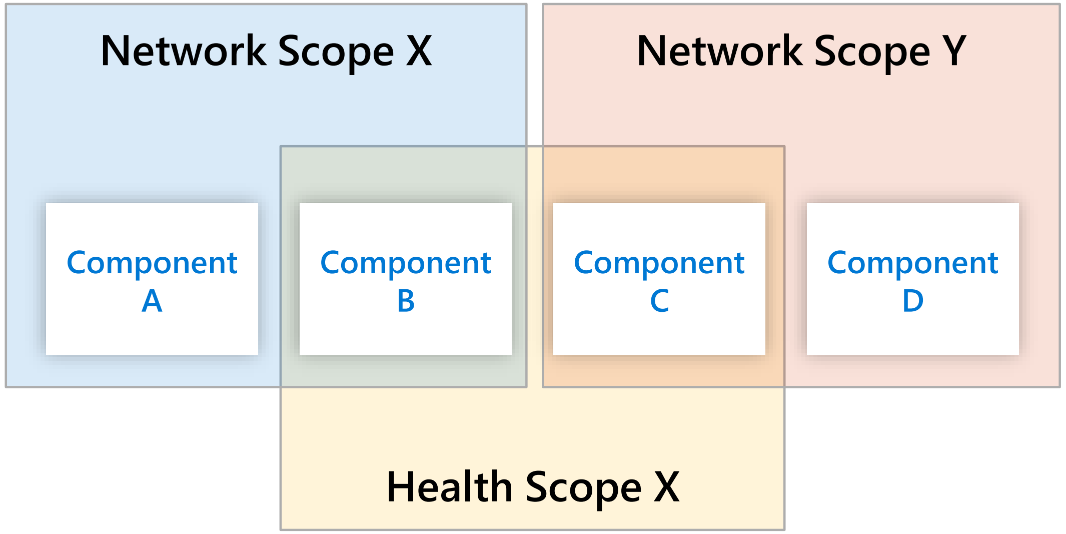
Where developers declare the operational characteristics of the code they deliver *in infrastructure neutral terms*.



```
apiVersion: core.oam.dev/v1alpha1
kind: Component
metadata:
  name: oamfrontend
  version: "1.0.0"
  description: Simple OAM app
spec:
  workloadType: core.oam.dev/v1alpha1.Server
  os: linux
  arch: amd64
  parameters:
    - name: oam_texture
      type: string
      required: true
      default: texture.jpg
  containers:
    - name: frontend
      image: ignite2019/oamhwfrontend:latest
      env:
        - name: OAM_TEXTURE
          value: texture.jpg
          fromParam: oam_texture
      ports:
        - containerPort: 8001
          name: http
          protocol: TCP
```

Application Scope

A way to loosely couple components into groups with common characteristics.



Network Scope X

Network Scope Y

Component A

Component B

Component C

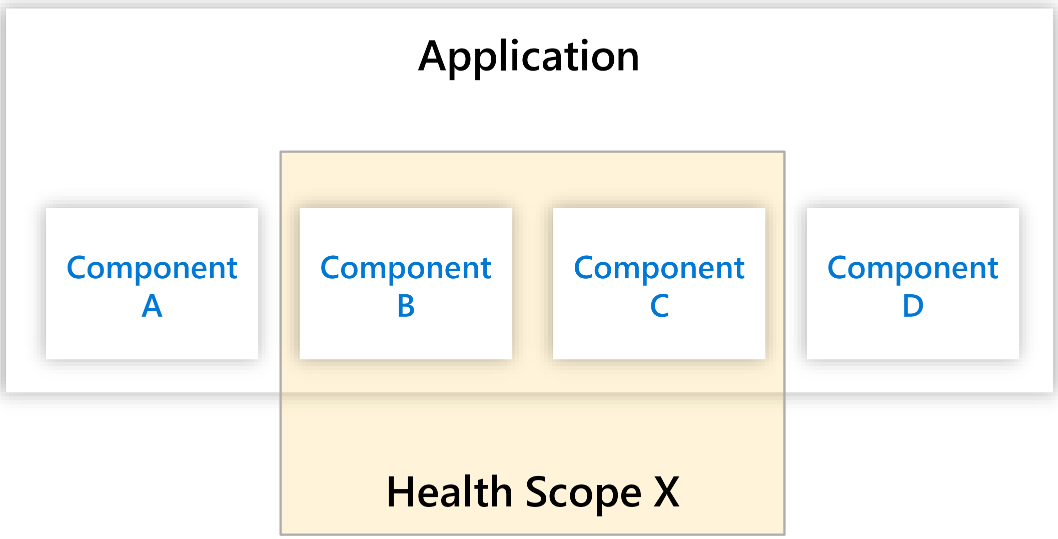
Component D

Health Scope X

```
apiVersion: core.oam.dev/v1alpha1
kind: ApplicationScope
metadata:
  name: network
  annotations:
    version: v1.0.0
    description: "network boundary that a
group of components reside in"
spec:
  type: core.oam.dev/v1.NetworkScope
  allowComponentOverlap: false
  parameters:
    - name: network-id
      description: The id of the network
      type: string
      required: Y
    - name: subnet-id
      description: The id of the subnet
      type: string
      required: Y
    - name: internet-gateway-type
      description: The type of the gateway.
      type: string
      required: N
```

Application

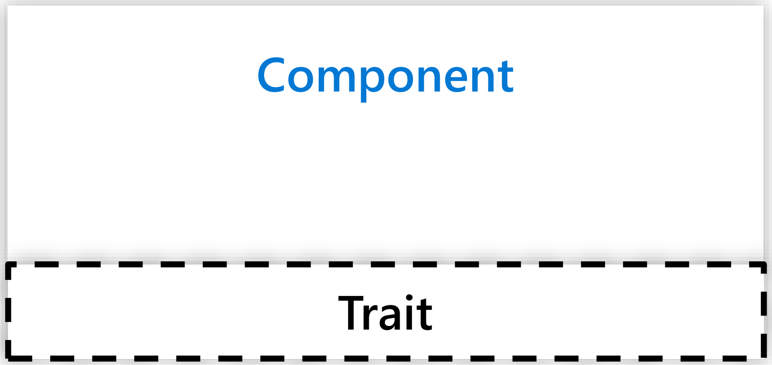
Where developers group components together into a single, deployable unit and specifies cross-component info, such as health scopes.



```
apiVersion: core.oam.dev/v1alpha1
kind: Application
metadata:
name: oam-helloworld-app
spec:
  components:
    - name: oamfrontend
    - name: oambackend
      traits:
        - name: scaler
          parameterValues:
            - name: min
              value: 1
            - name: max
              value: 50
  scopes:
    - name: oam-be-fe-metrics
      type: core.oam.dev/v1.HealthScope
      parameters:
        - name: metrics-endpoint
          protocol: https
          path: /metrics
```

Trait

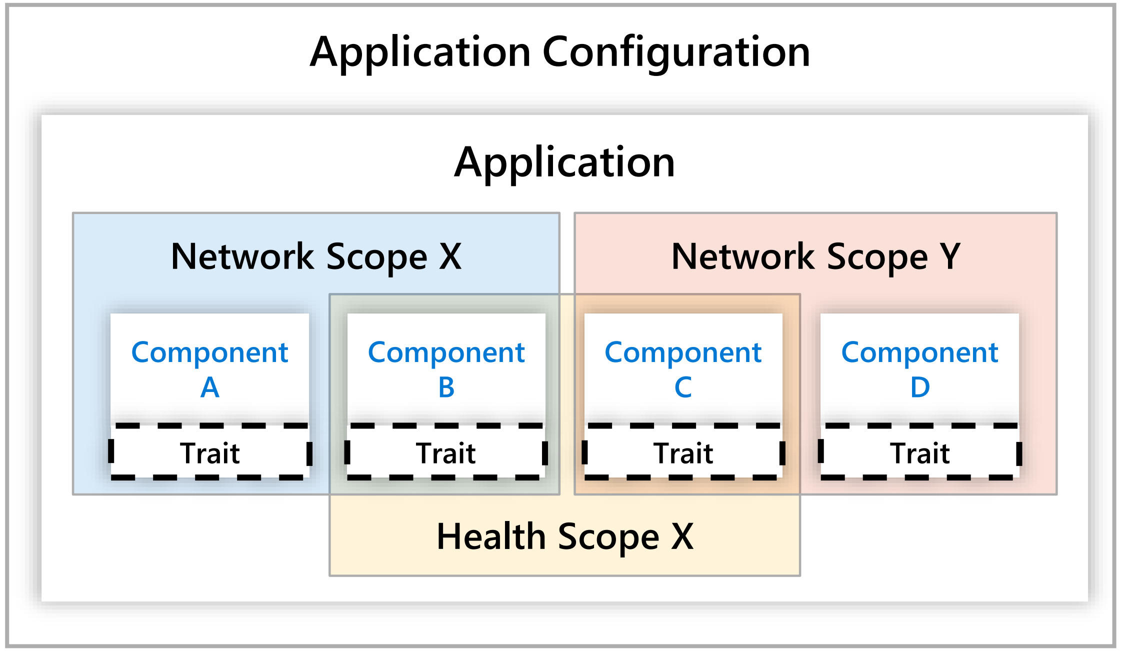
For assigning operational features to instances of components.



```
apiVersion: core.oam.dev/v1alpha1
kind: Trait
metadata:
  name: ManualScaler
  annotations:
    version: v1.0.0
  spec:
  appliesTo:
    - core.oam.dev/v1alpha1.Server
    - core.oam.dev/v1alpha1.Worker
    - core.oam.dev/v1alpha1.Task
  properties:
    type: object
    properties: |
      {"$schema": "http://json-
schema.org/draft-07/schema#",
        "type": "object",
        "required": ["replicaCount],
        "properties": {
          "replicaCount": {
            "type": "integer",
         "minimum": 0 }}}
```

Application Configuration

Defines a configuration of an application, its traits, and additional scopes, such as network scopes.



```yaml
apiVersion: core.oam.dev/v1alpha1
kind: ApplicationConfiguration
metadata:
  name: oam-helloworld
spec:
  components:
    - name: oamfrontend
      instanceName: oam-fe1
      parameterValues:
        - name: oam_texture
          value: aks
      traits:
        - name: ingress
          parameterValues:
            - name: hostname
              value: aks.azureocto.com
            - name: path
              value: /
            - name: service_port
              value: 8001
    - name: oambackend
      instanceName: oam-be1
```

DEMO

Deploying an OAM application to rudr

dapr

Distributed Application Runtime

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

State of Enterprise Developers

Being asked to develop resilient, scalable, microservice-based apps

Functions and Actors are powerful programming models

They write in many languages

They want to leverage existing code

What is holding back serverless development?

Frequently need to incrementally migrate from existing and legacy code

Serverless runtimes have narrow language support with tightly controlled feature sets

Serverless runtimes don’t have composable and incrementally adoptable equivalents that can run anywhere

Introducing Dapr

A portable, event-driven, serverless runtime for building distributed applications across cloud and edge

{ } Sidecar Architecture

Developer first, standard APIs used from any programming language or framework

Microservice Building Blocks

Make it easy for developers to create microservice applications without being an expert in distributed systems, including migrating existing code

Cloud + Edge

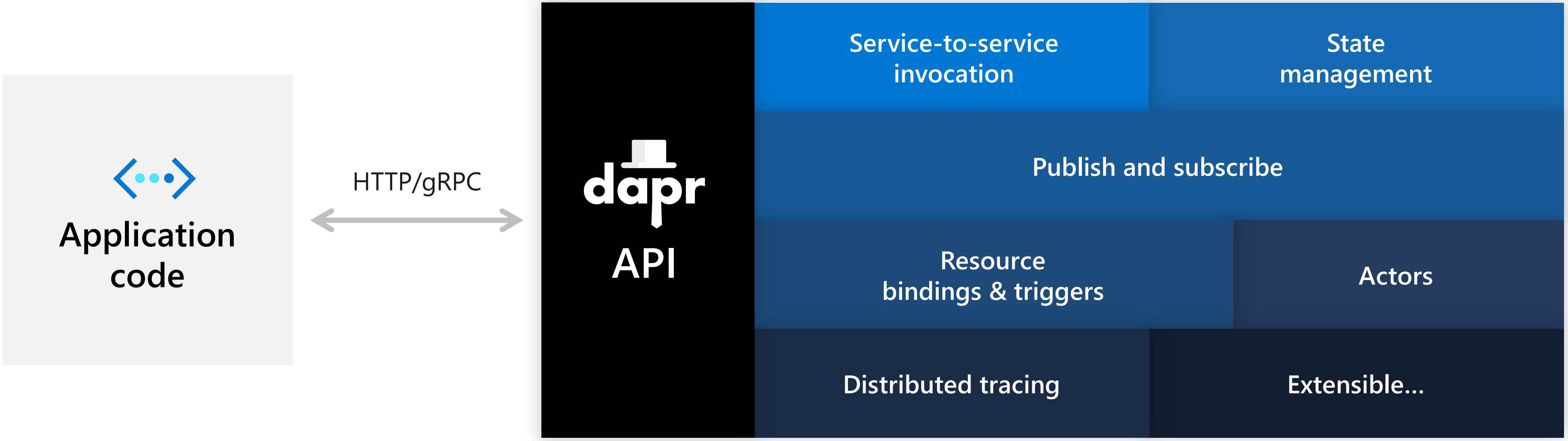
Runs on multiple environments for cloud, on-prem, and small-edge including any Kubernetes

Sidecar architecture

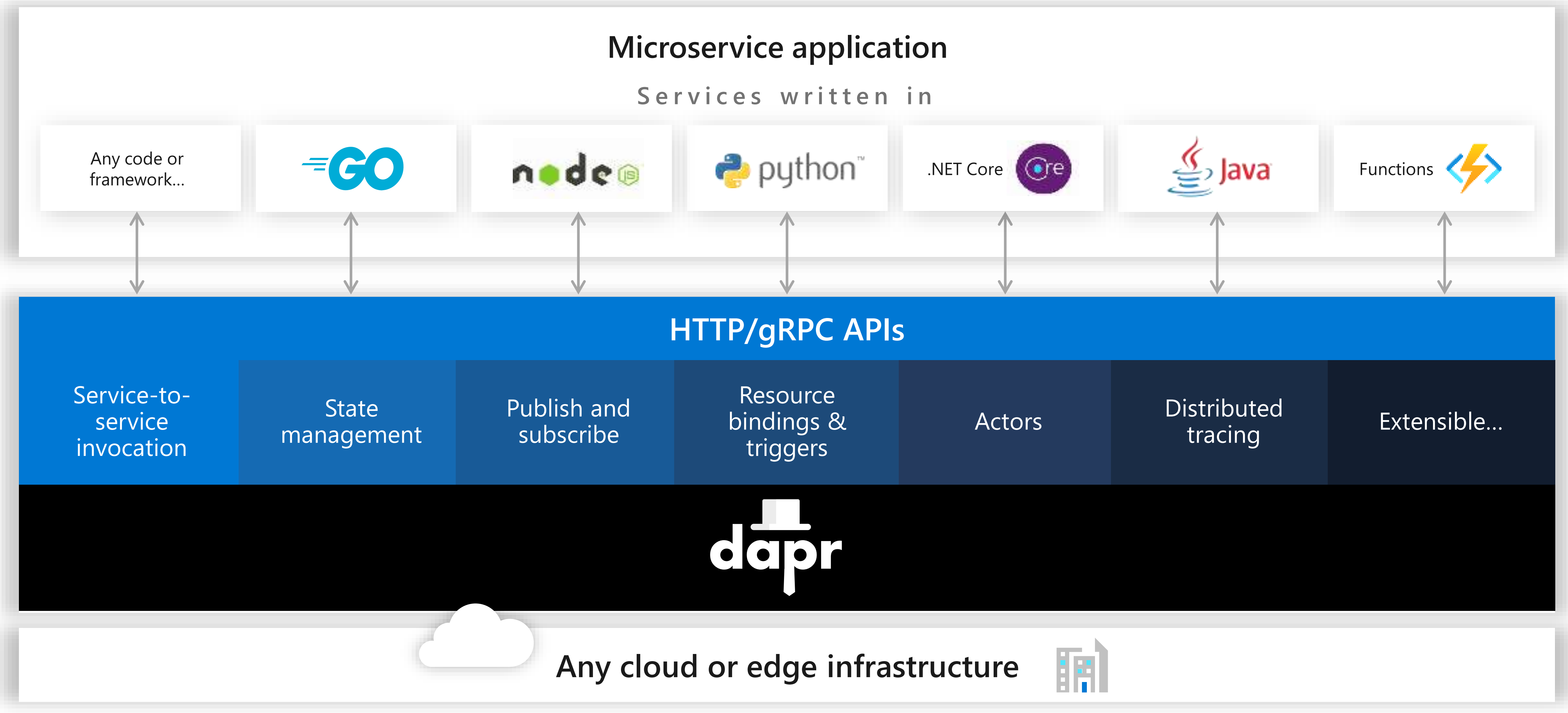
Standard APIs accessed over http/gRPC protocols from user service code

`e.g. http://localhost:3500/v1.0/invoke/myapp/method/neworder`

Dapr runs as local “side-car library” dynamically loaded at runtime for each service



Dapr: Build apps using any language with any framework



Microservice application

Services written in

Any code or framework…

.NET Core

Functions

HTTP/gRPC APIs

Service-to-service invocation

State management

Publish and subscribe

Resource bindings & triggers

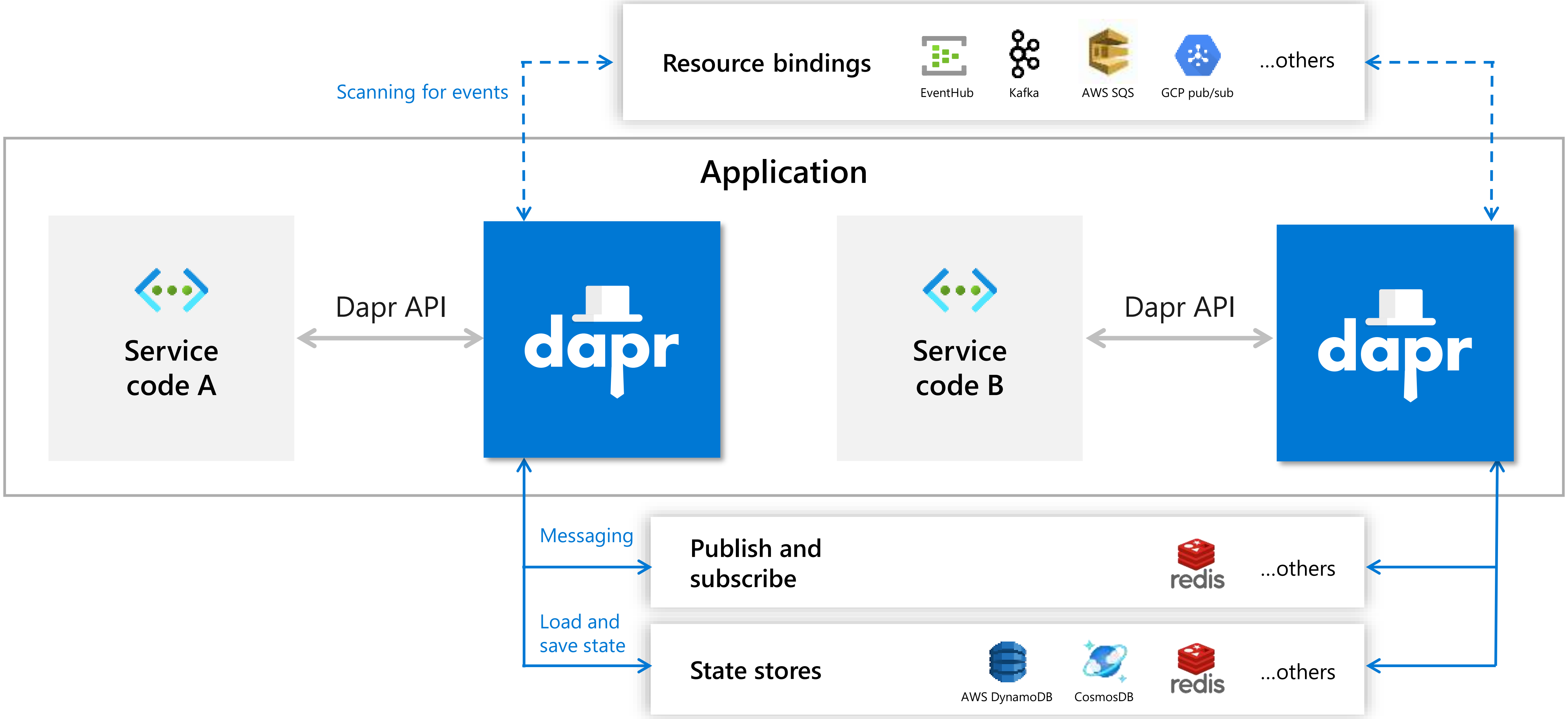
Actors

Distributed tracing

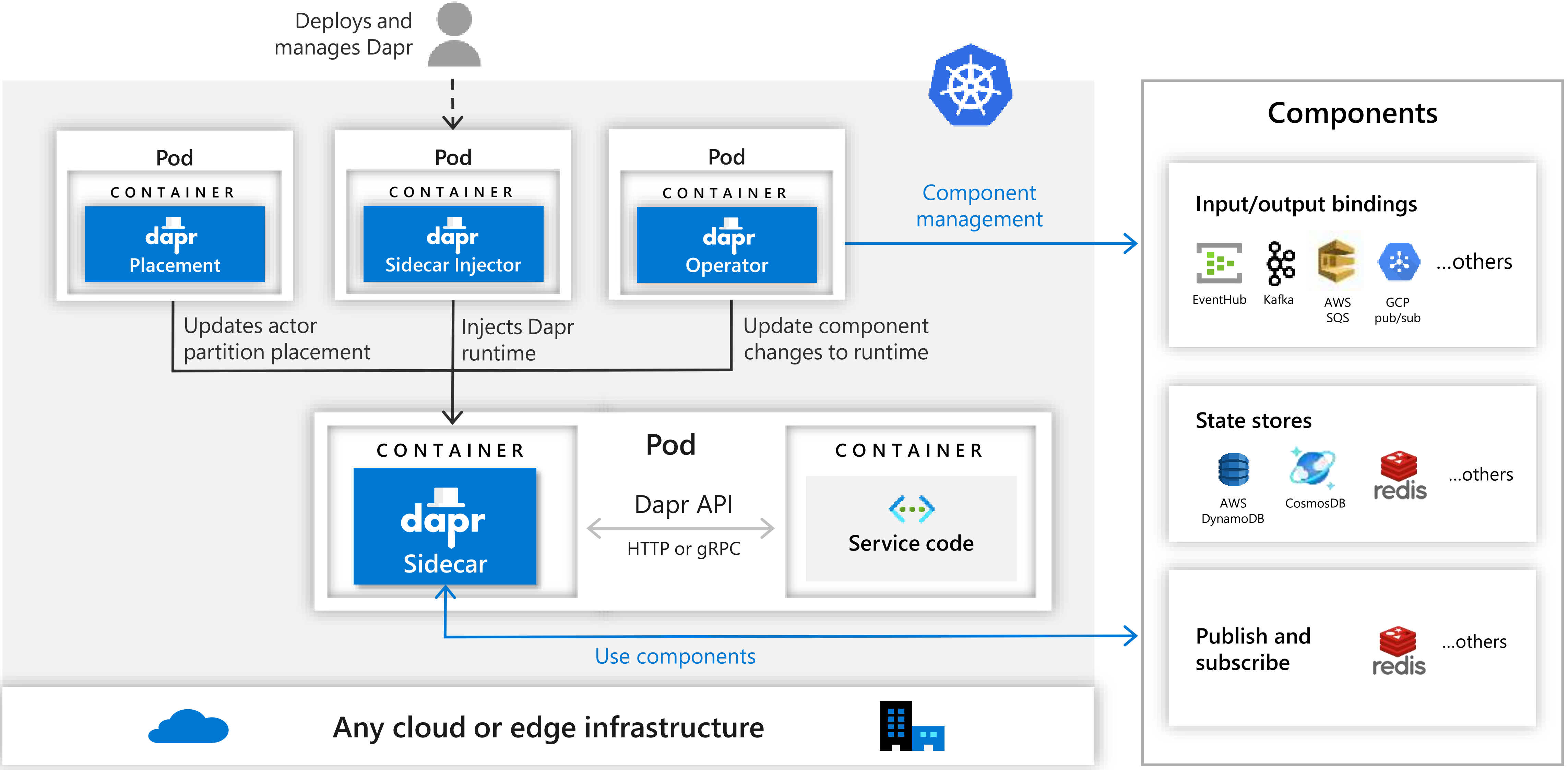
Extensible…

Any cloud or edge infrastructure

Dapr self-hosted



Dapr Kubernetes-hosted



Microservice Building Blocks

State Management

Create long running, stateless and stateful services

Service Invocation & Fault Handling

Perform direct, secure, service-to-service method calls

Resource Bindings

Trigger code through events from a large array of input and output bindings to external resources including databases and queues

Publish & Subscribe

Secure, scalable messaging between services

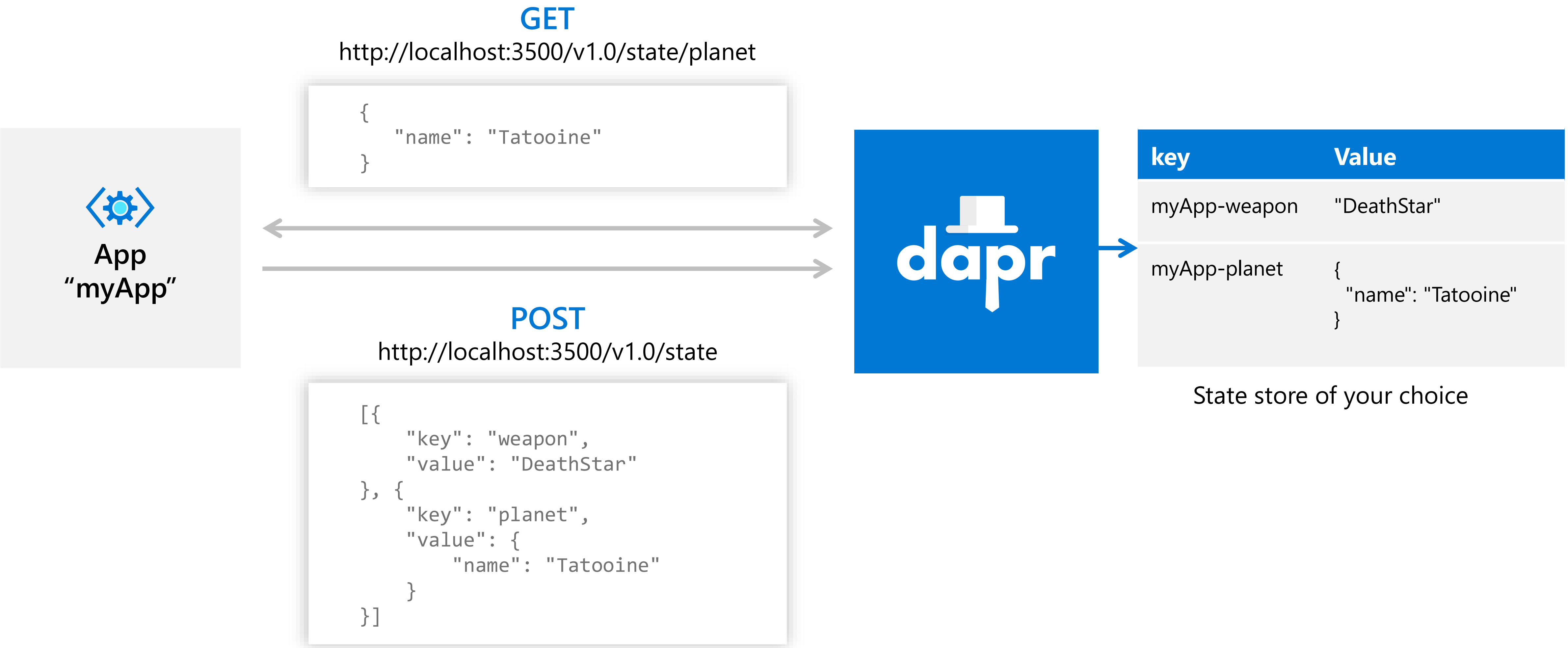
Actors

Encapsulate code and data in reusable actor objects as a common microservices design pattern

Distributed Tracing & Diagnostics

See and measure the message calls across components and networked services

State management



GET

http://localhost:3500/v1.0/state/planet

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

POST

http://localhost:3500/v1.0/state

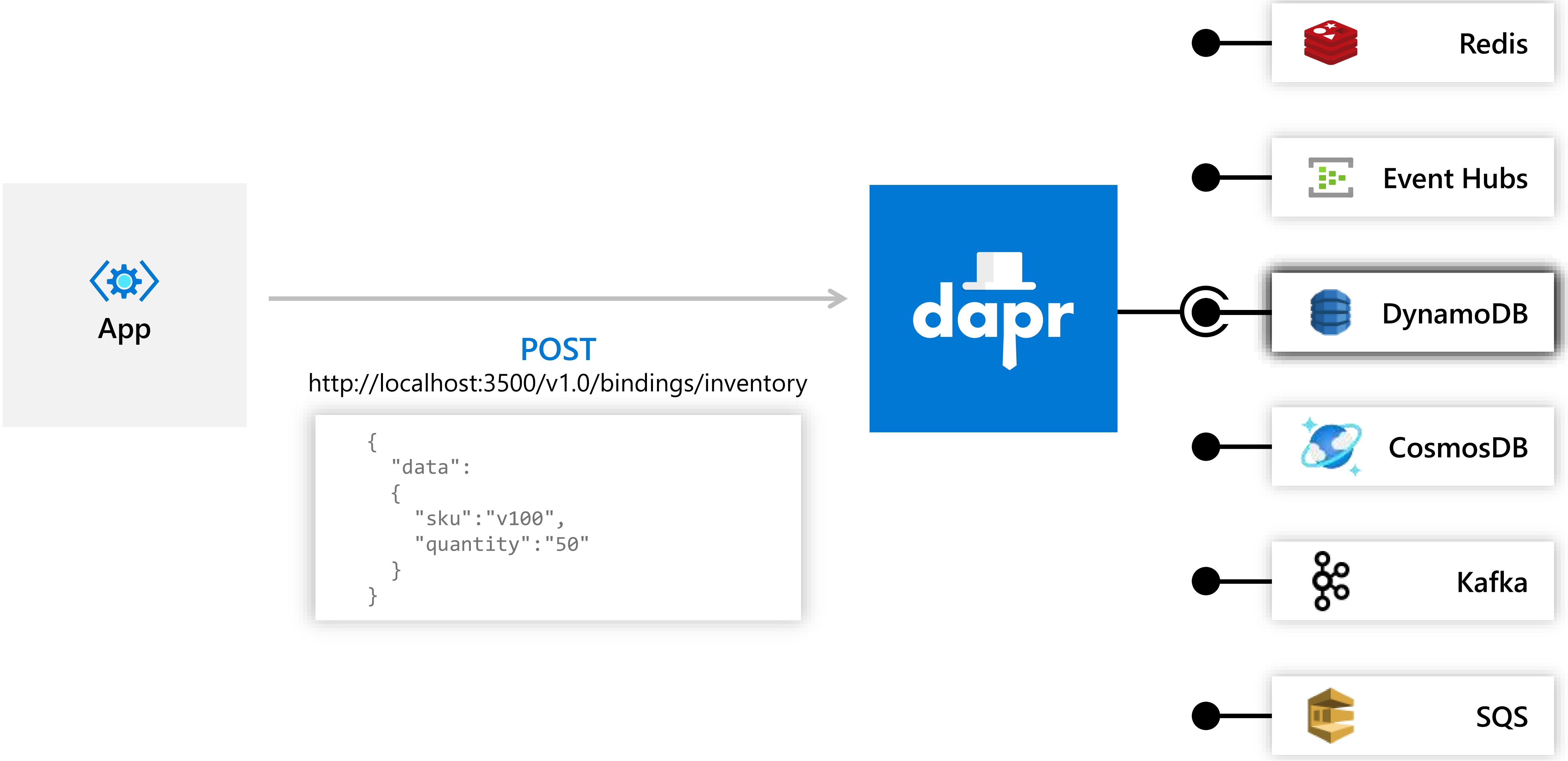
```
[{
    "key": "weapon",
    "value": "DeathStar"
}, {
    "key": "planet",
    "value": {
        "name": "Tatooine"
    }
}]
```

key	Value
myApp-weapon	"DeathStar"

myApp-planet	{
	"name": "Tatooine"
	}

State store of your choice

Output bindings



App

POST

http://localhost:3500/v1.0/bindings/inventory

```
{
  "data":
  {
    "sku":"v100",
    "quantity":"50"
  }
}
```

dapr

Redis

Event Hubs

DynamoDB

CosmosDB

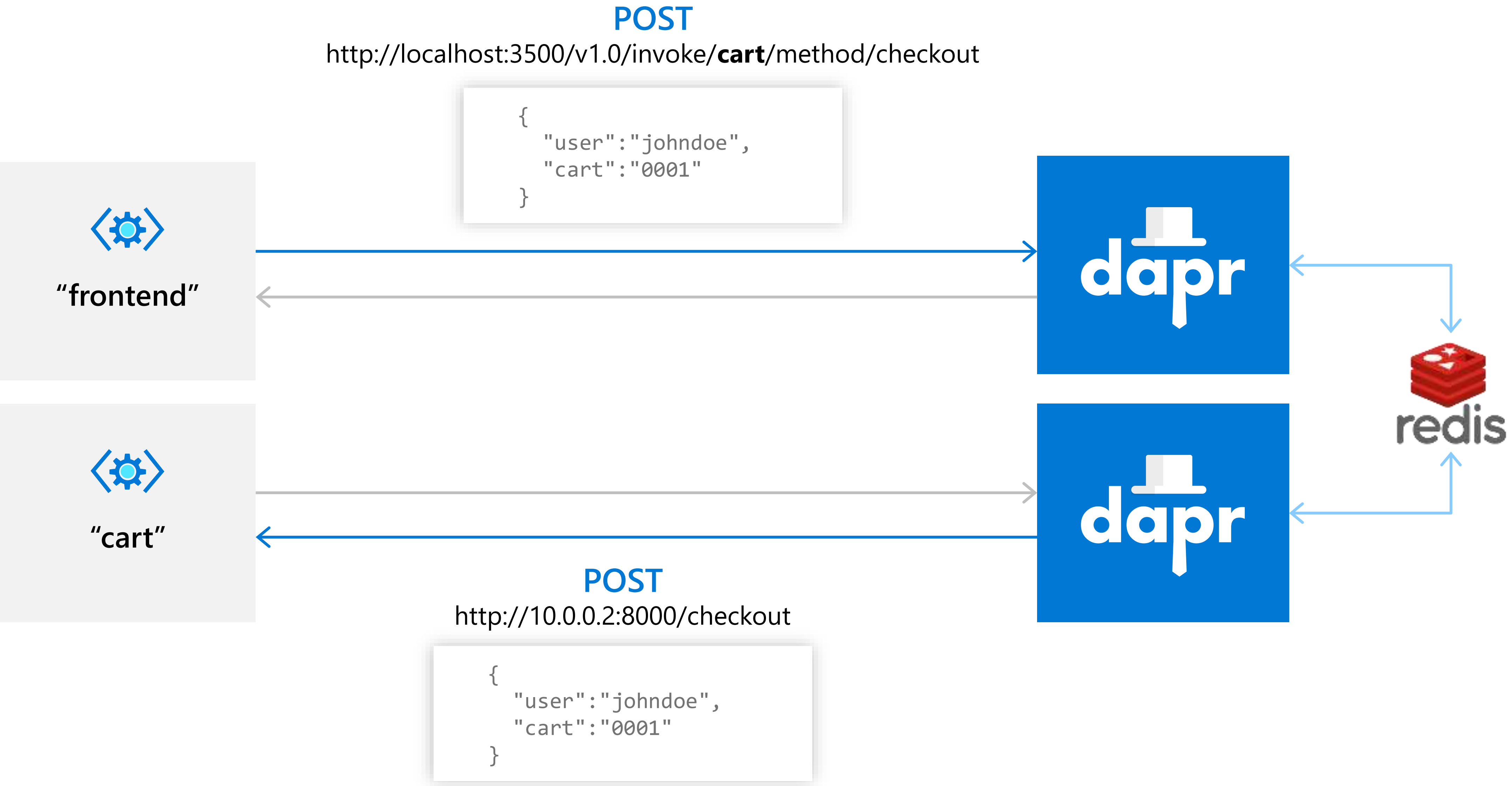
Kafka

SQS

DEMO

Dapr State Management and Bindings

Service Invocation



POST

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

```
{
  "user":"johndoe",
  "cart":"0001"
}
```

“frontend”

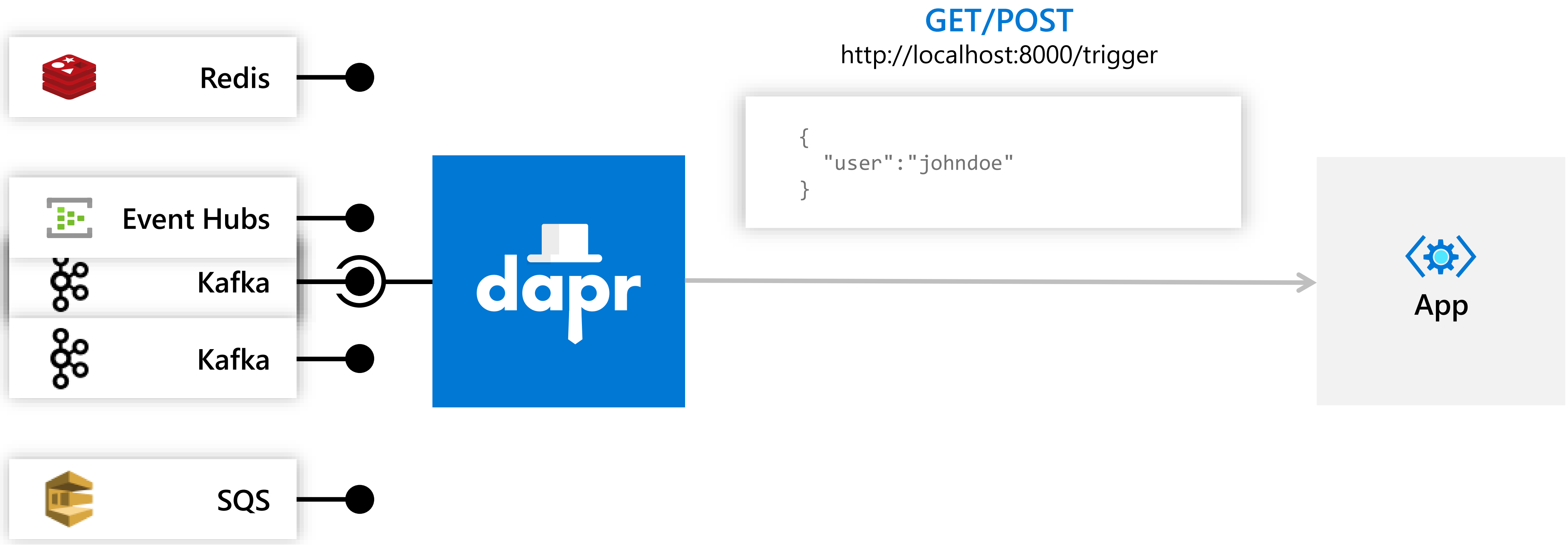
“cart”

POST

http://10.0.0.2:8000/checkout

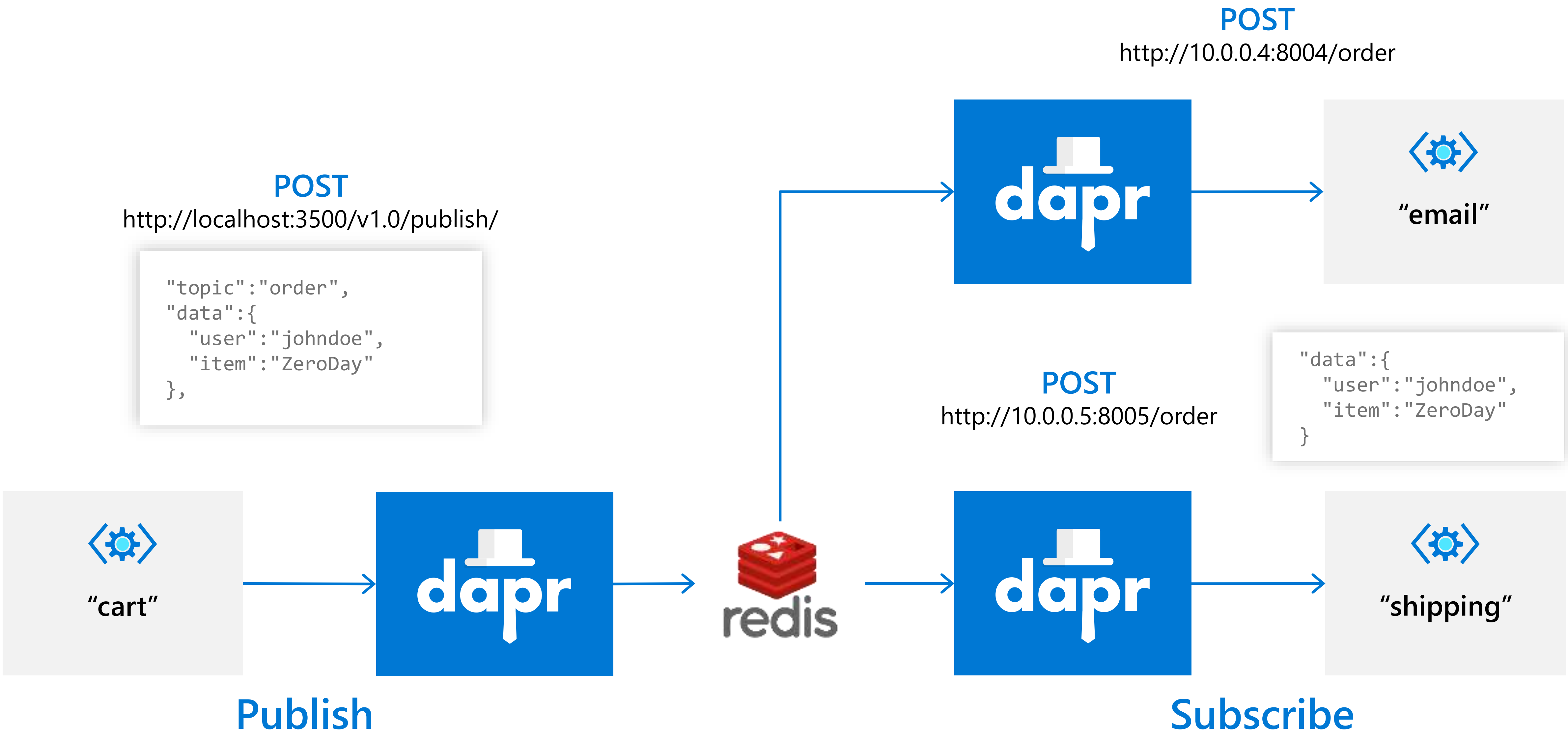
```
{
  "user":"johndoe",
  "cart":"0001"
}
```

Input bindings



```
{
  "user":"johndoe"
}
```

Publishing & Subscribing



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

```
"topic":"order",
"data":{
  "user":"johndoe",
  "item":"ZeroDay"
},
```

POST
http://10.0.0.4:8004/order

POST
http://10.0.0.5:8005/order

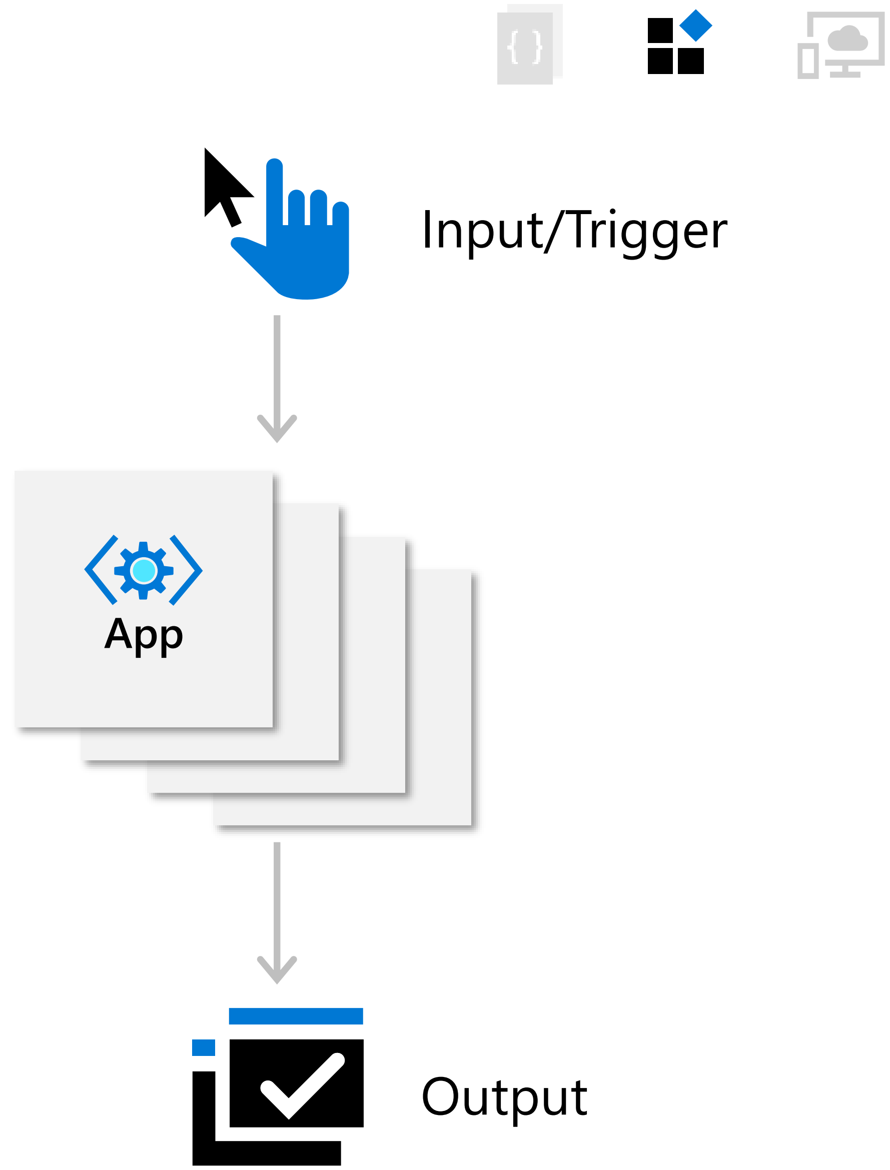
```
"data":{
  "user":"johndoe",
  "item":"ZeroDay"
}
```

Functions with Dapr

Event driven

Stateless

Easy replication/scaling



DEMO

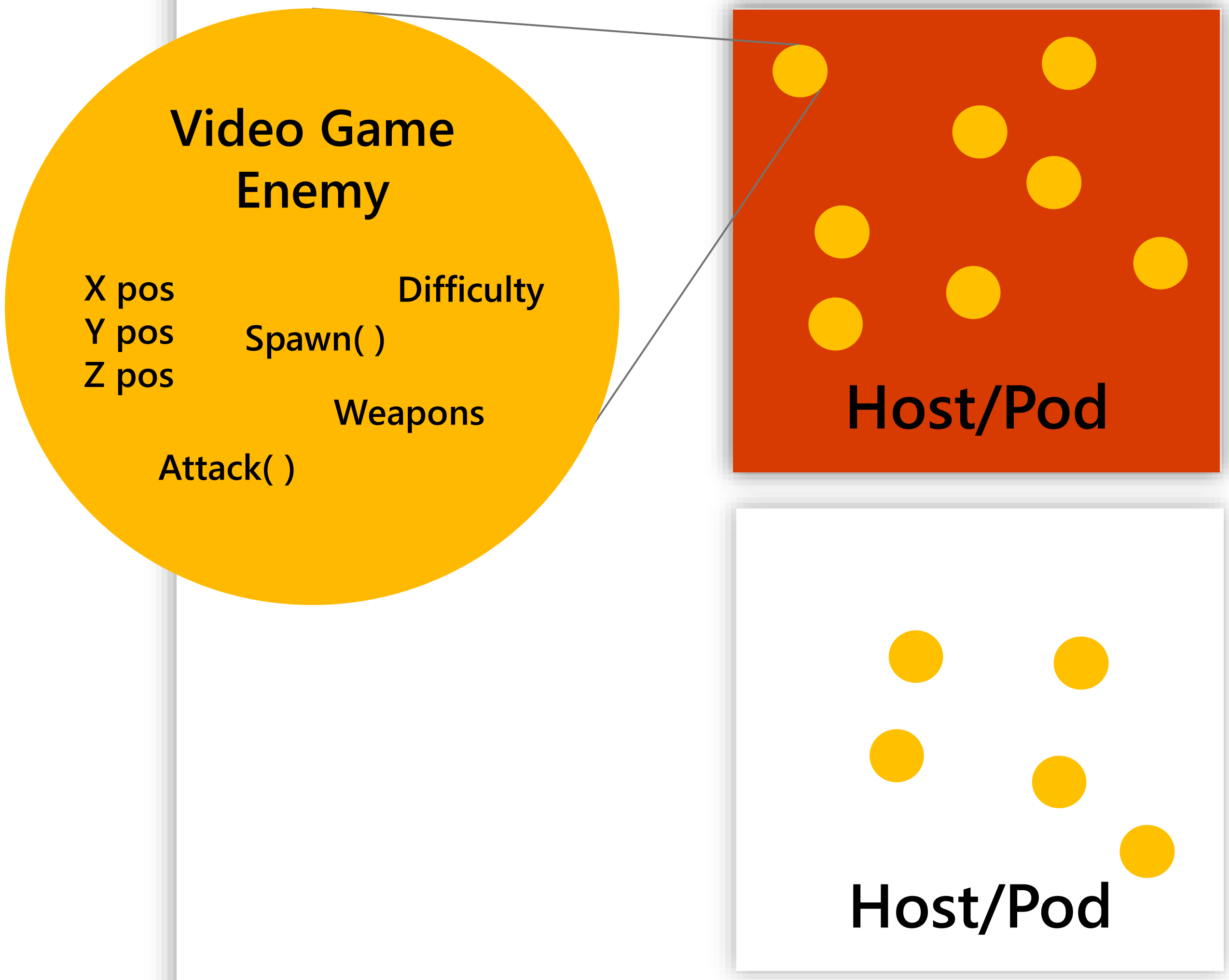
Functions with Dapr

Virtual Actors with Dapr

Stateful, objects of storage and compute

Dapr Actor Features:

- Distribution & failover
- Turn-based concurrency
- State management
- Timers
- Reminders



Video Game Enemy

X pos
Y pos
Z pos

Difficulty

Spawn( )

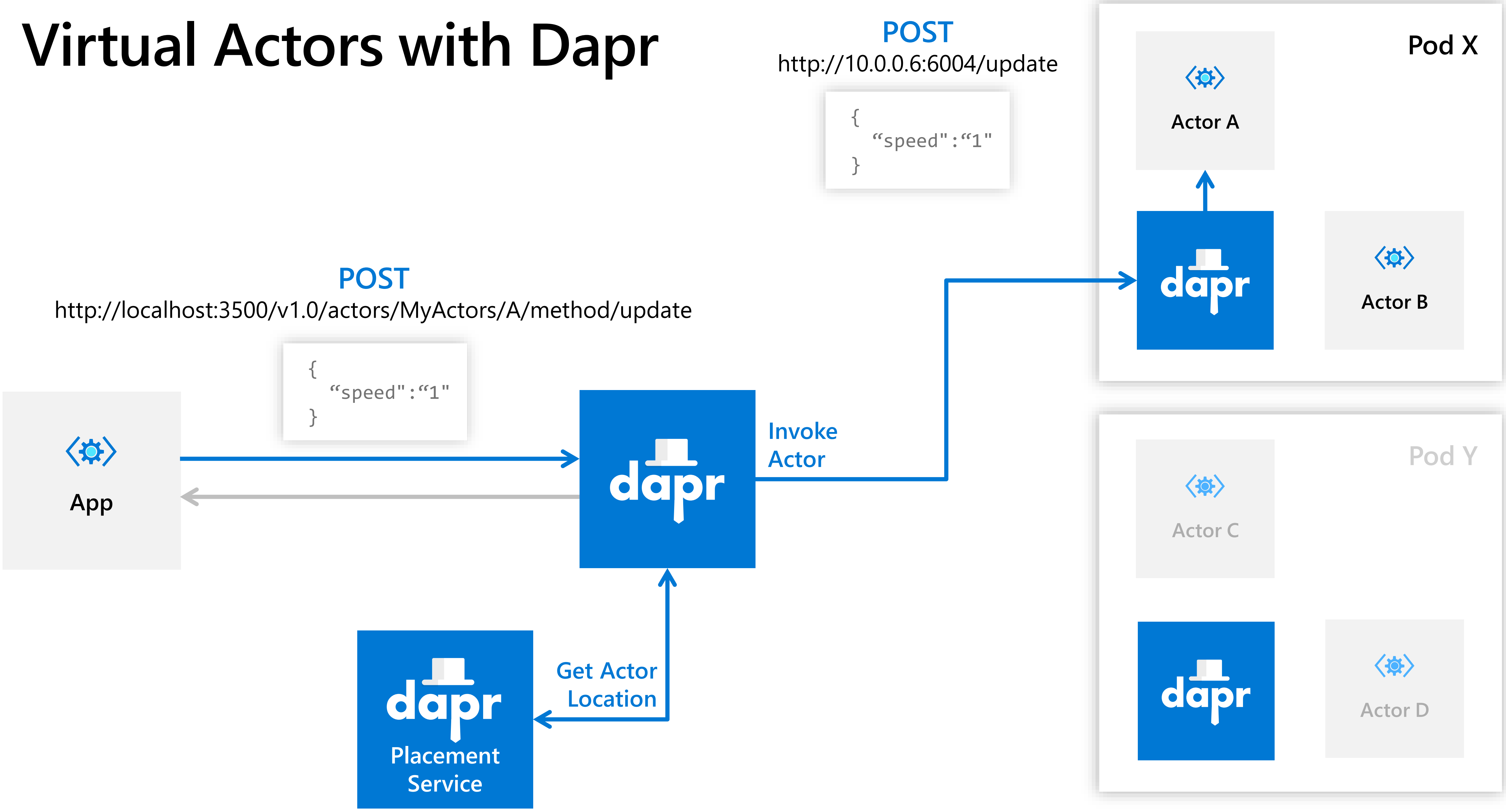
Weapons

Attack( )

Host/Pod

Host/Pod

Virtual Actors with Dapr



POST

http://localhost:3500/v1.0/actors/MyActors/A/method/update

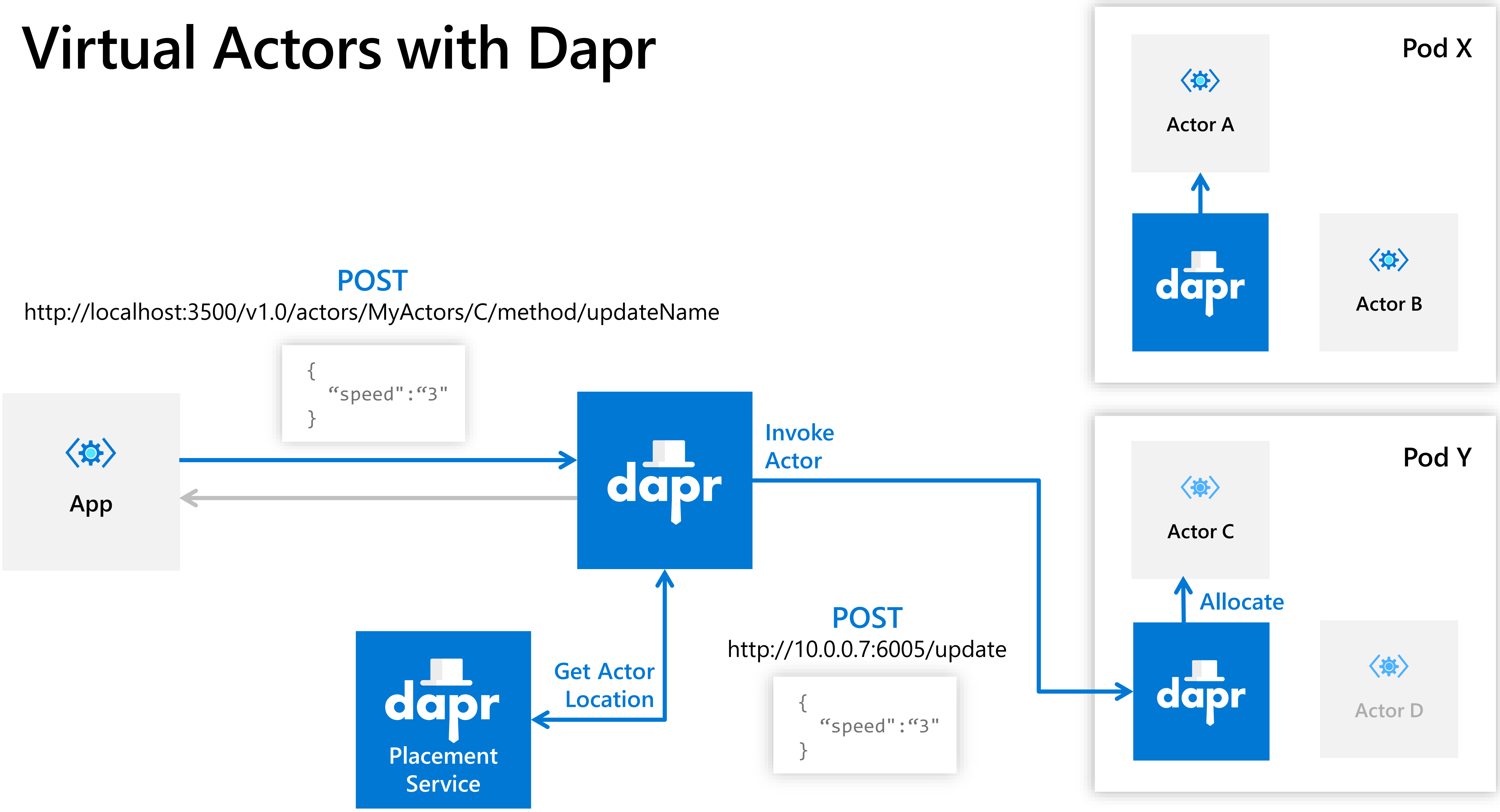
```
{
  “speed":“1"
}
```

POST

http://10.0.0.6:6004/update

```
{
  “speed":“1"
}
```

Virtual Actors with Dapr



POST
http://localhost:3500/v1.0/actors/MyActors/C/method/updateName

```
{
  “speed":“3"
}
```

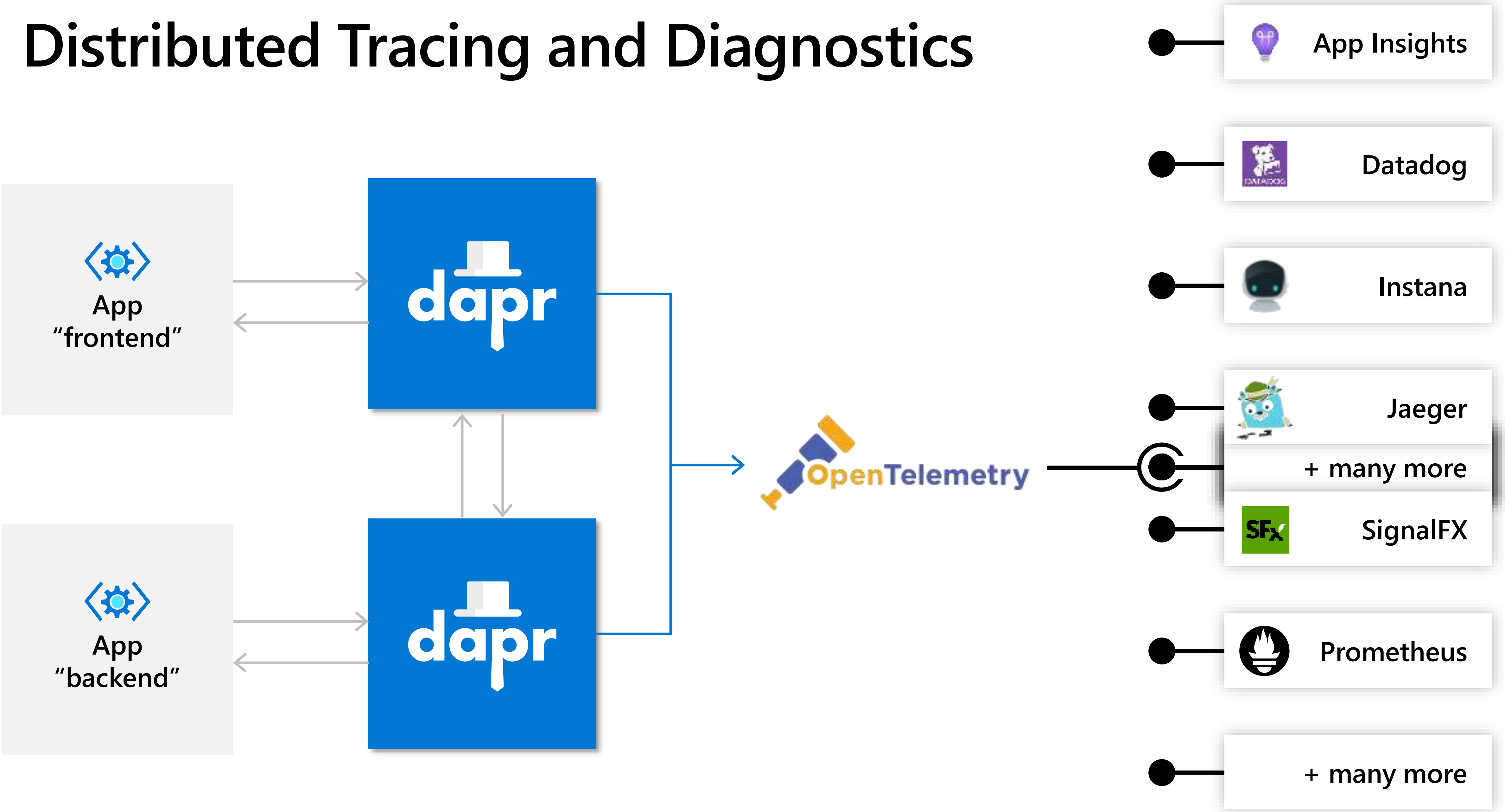
POST
http://10.0.0.7:6005/update

```
{
  “speed":“3"
}
```

DEMO

Actors with Dapr

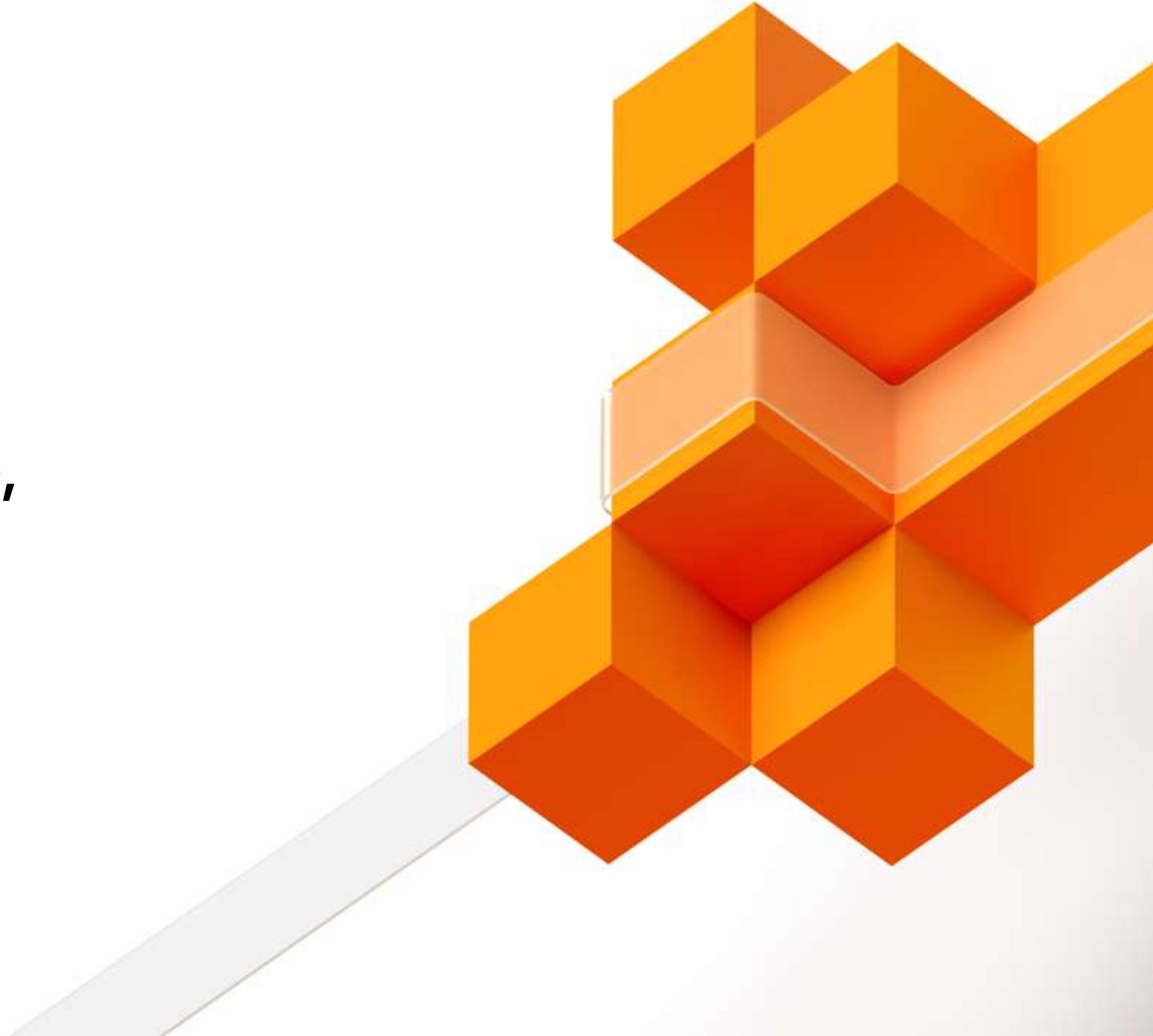
Distributed Tracing and Diagnostics



DEMO

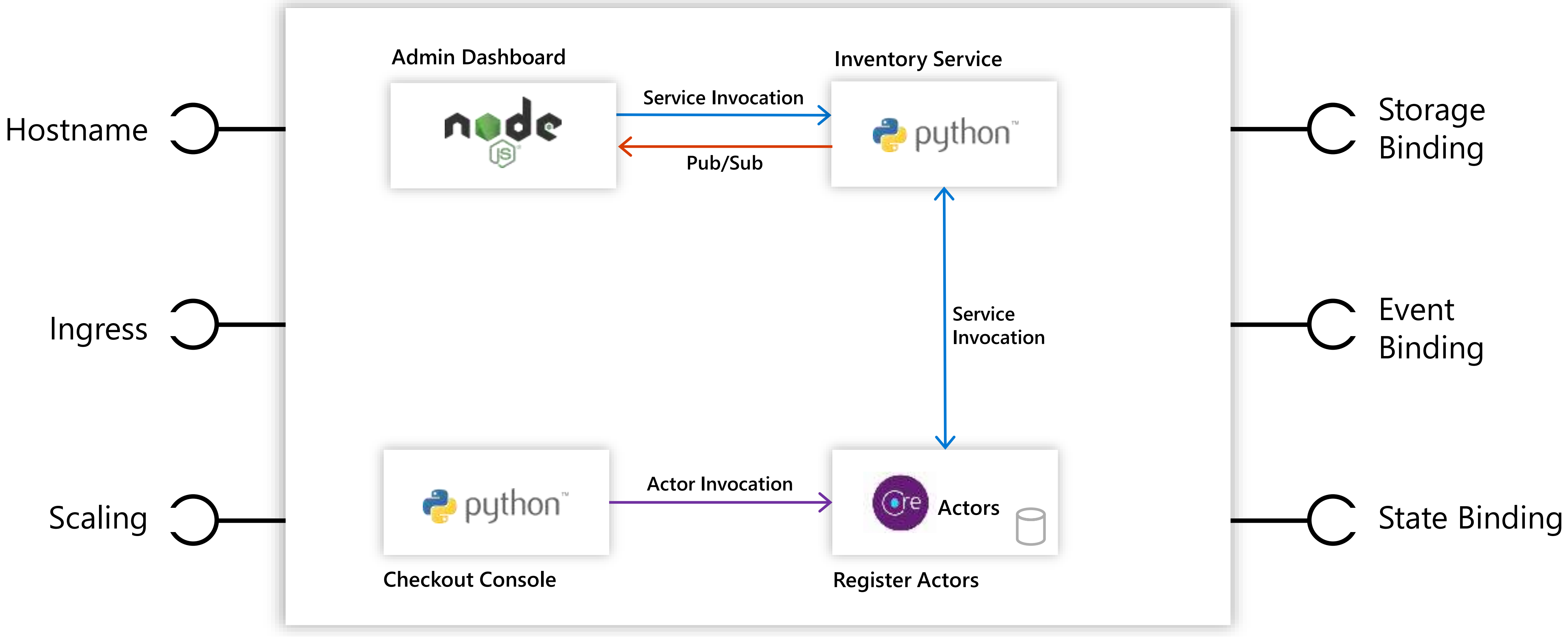
Diagnostics with Dapr

Building Cloud Scale, Hybrid Applications



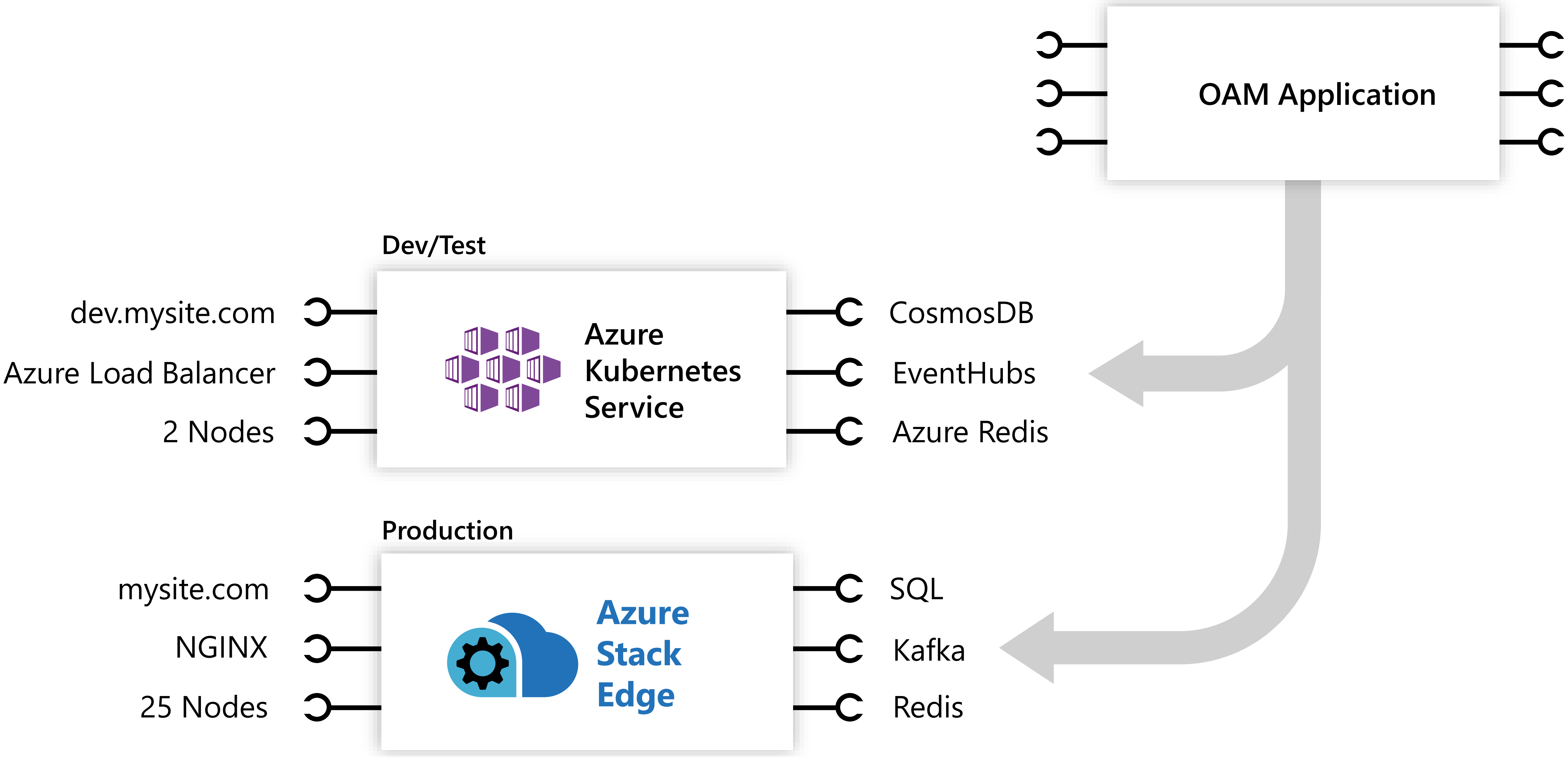
Retail PoS Application

Built with Stateless and Stateful Services



Retail PoS Application

Built with Stateless and Stateful Services



OAM Application

Dev/Test

dev.mysite.com

Azure Load Balancer

2 Nodes

Azure Kubernetes Service

CosmosDB

EventHubs

Azure Redis

Production

mysite.com

NGINX

25 Nodes

Azure Stack Edge

SQL

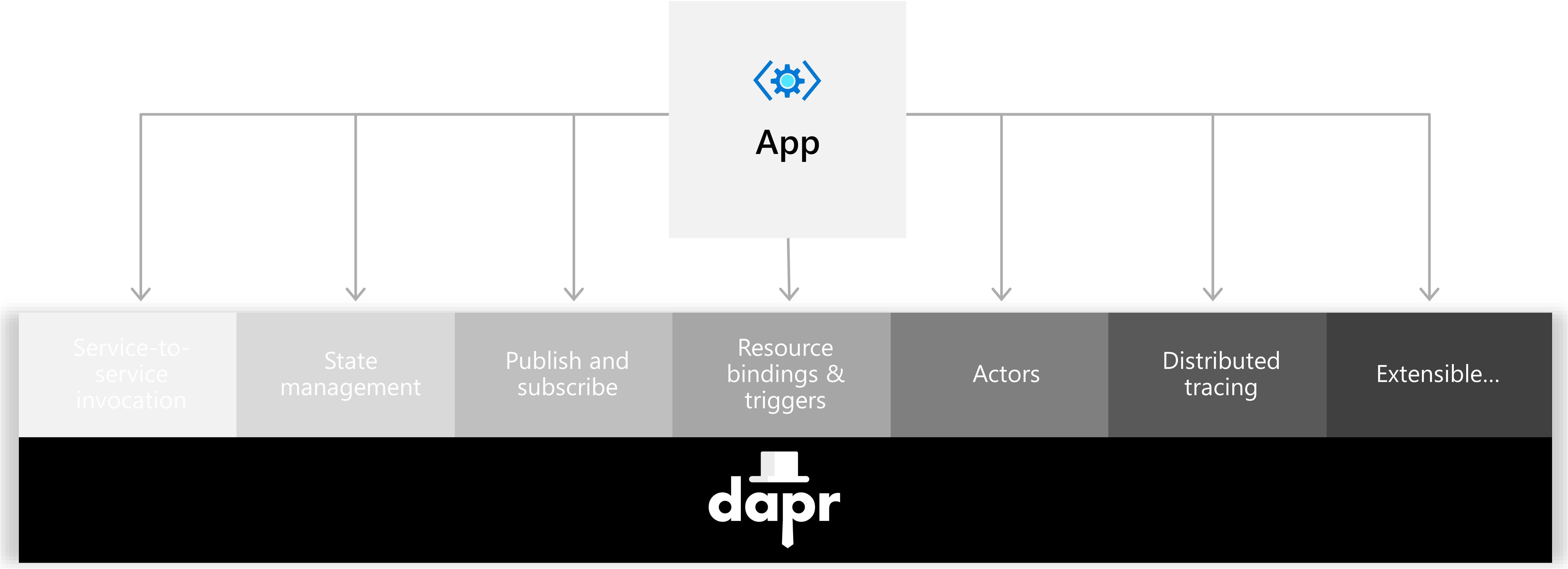
Kafka

Redis

DEMO

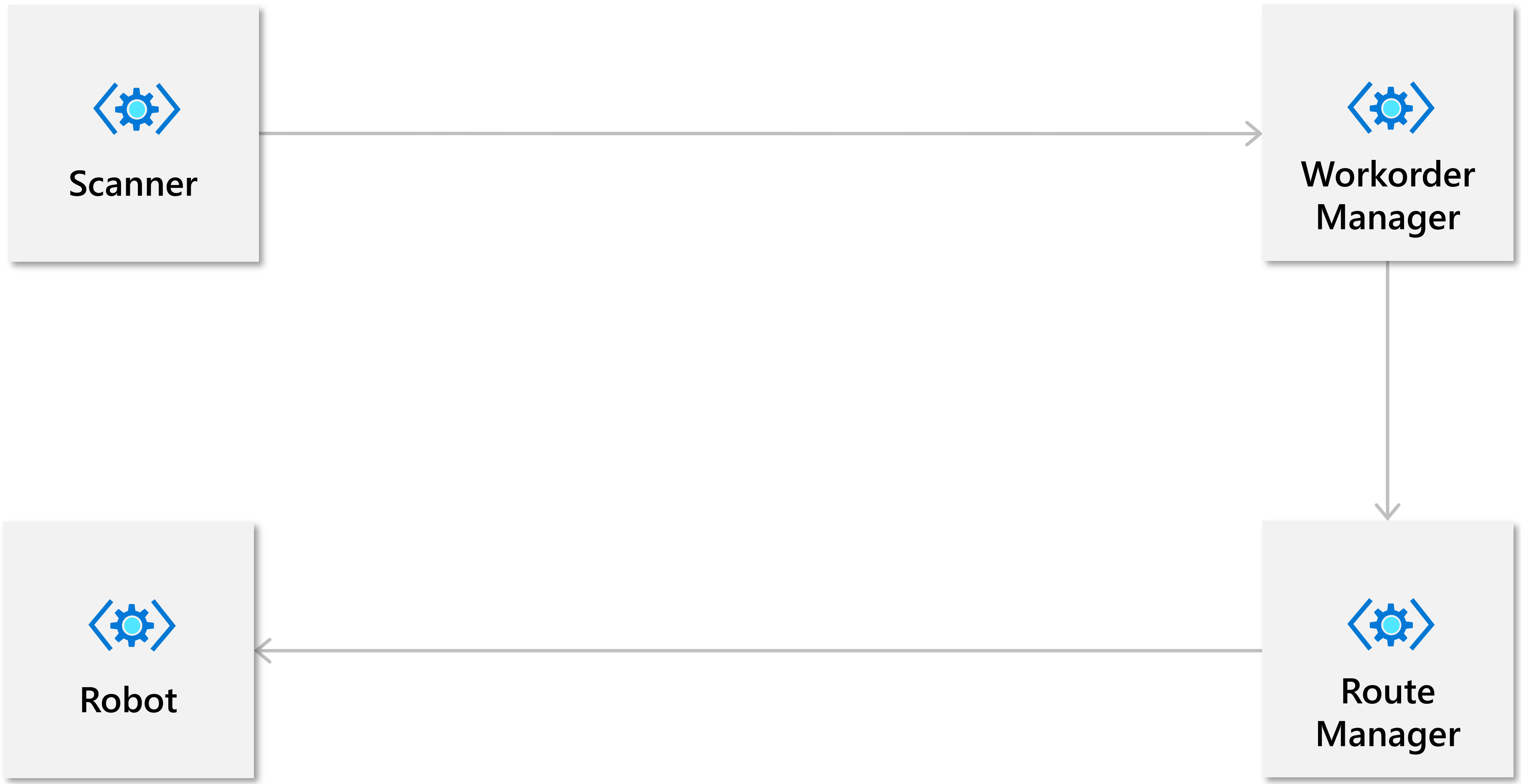
Retail Point of Sale (PoS) Application

Incrementally adoptable



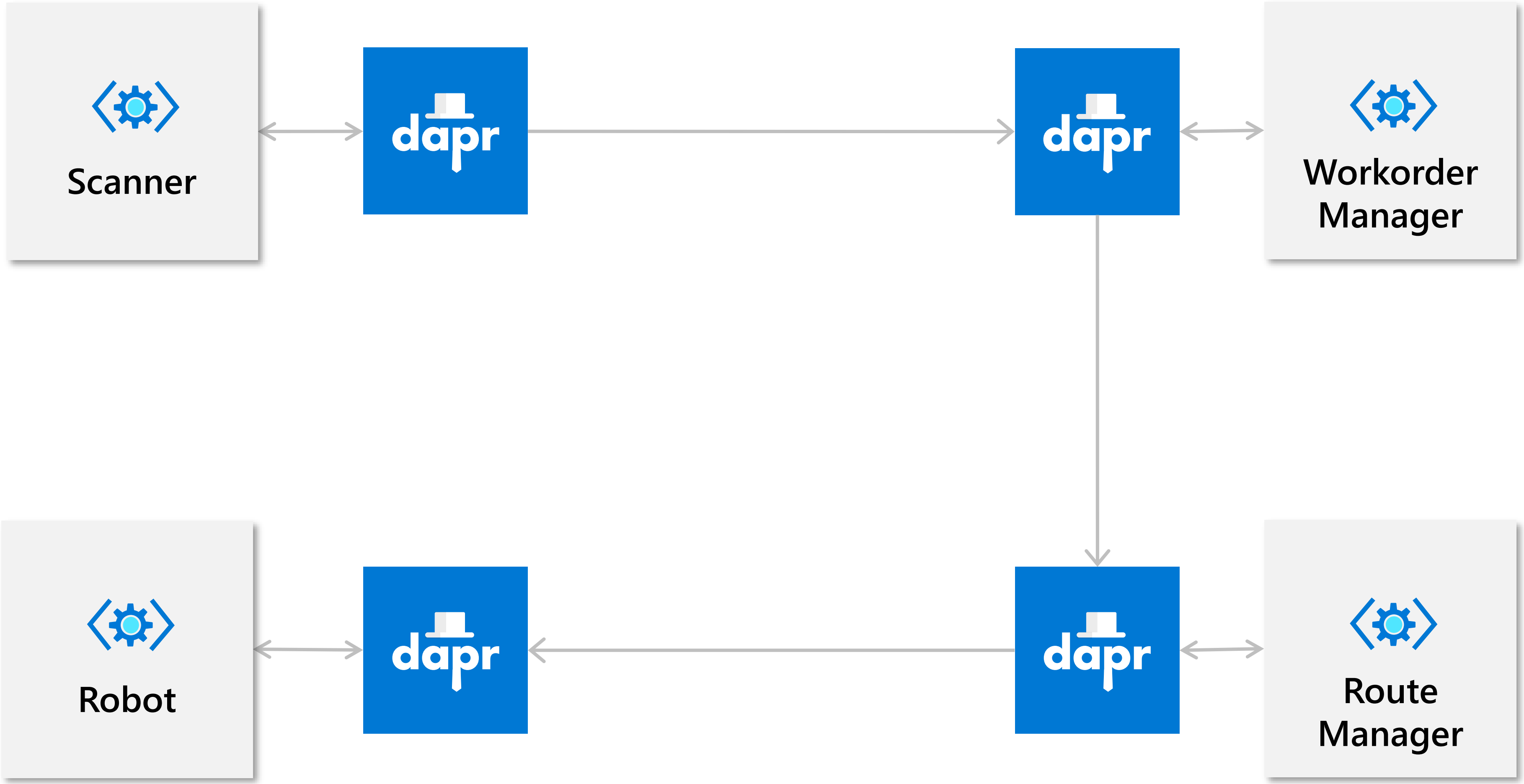
Warehouse Robotics

Incremental extensions to a legacy system



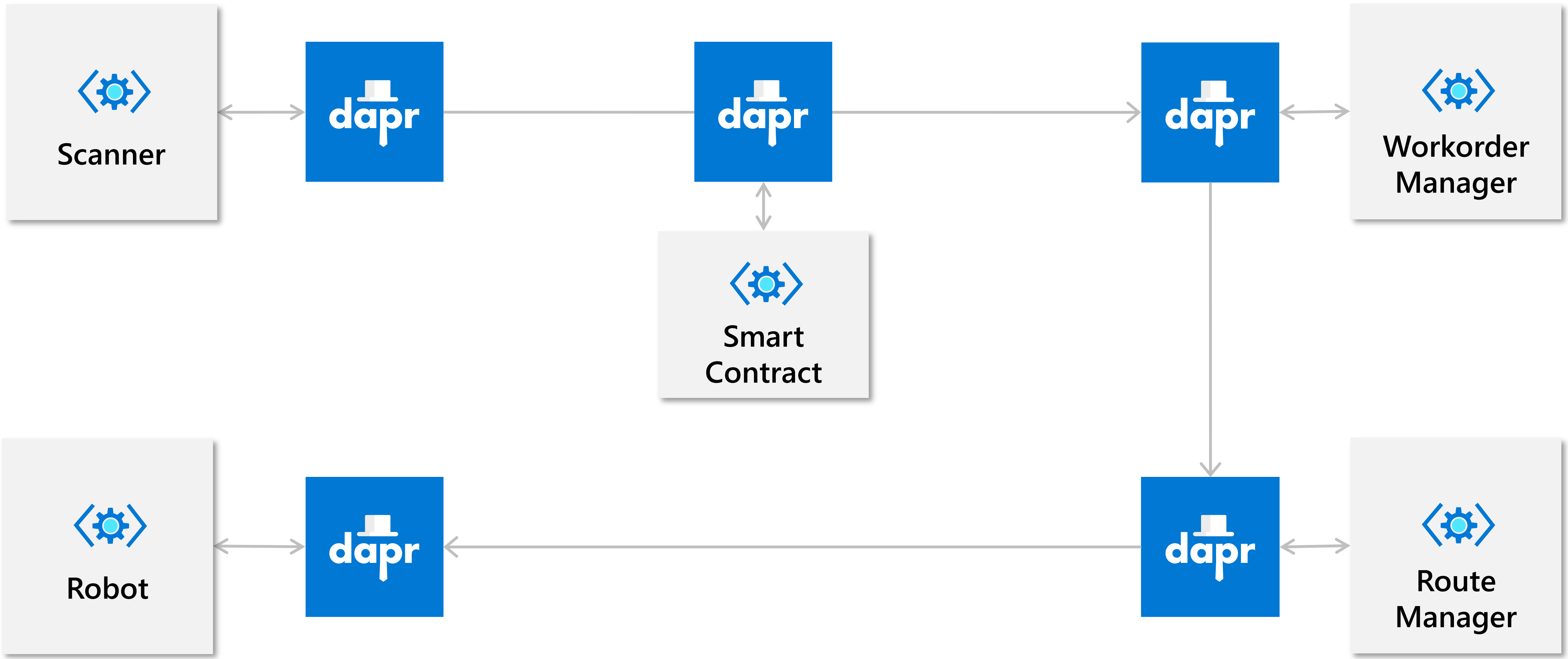
Warehouse Robotics

Incremental extensions to a legacy system



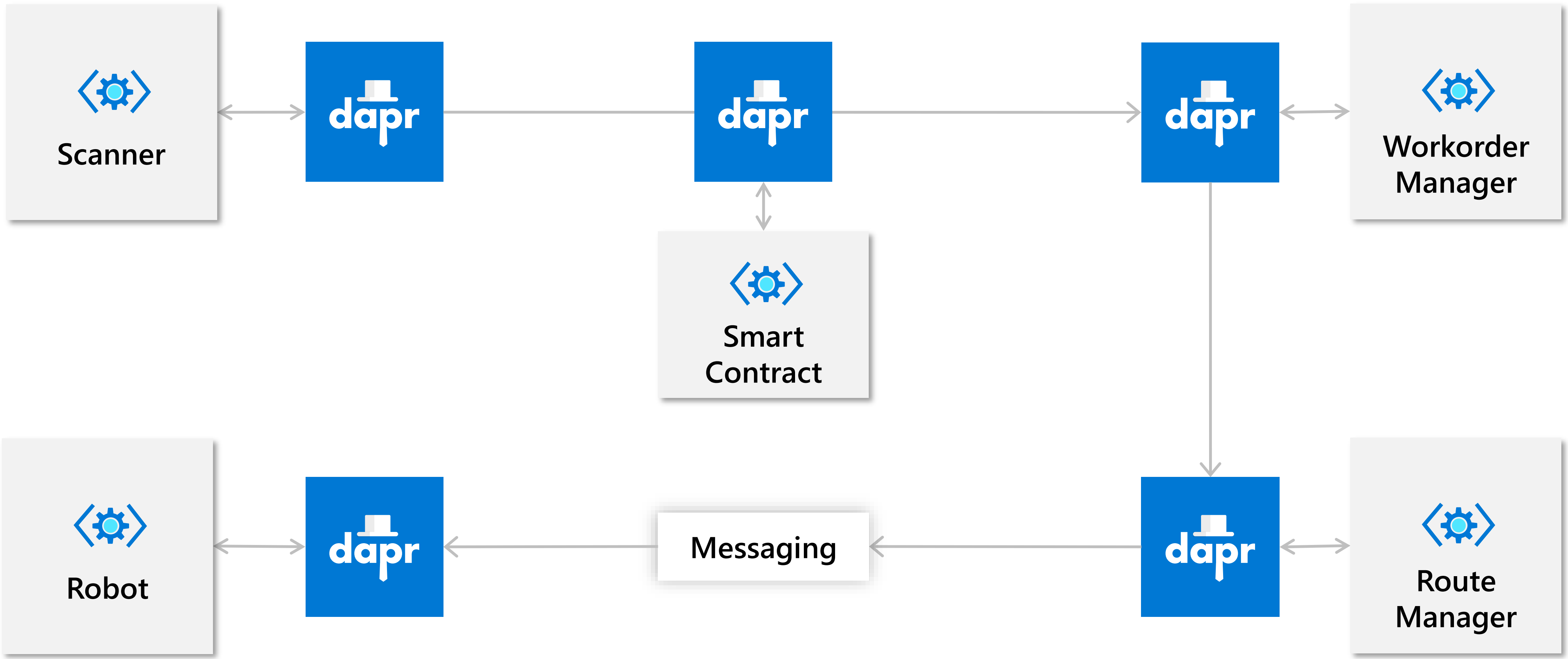
Warehouse Robotics

Incremental extensions to a legacy system



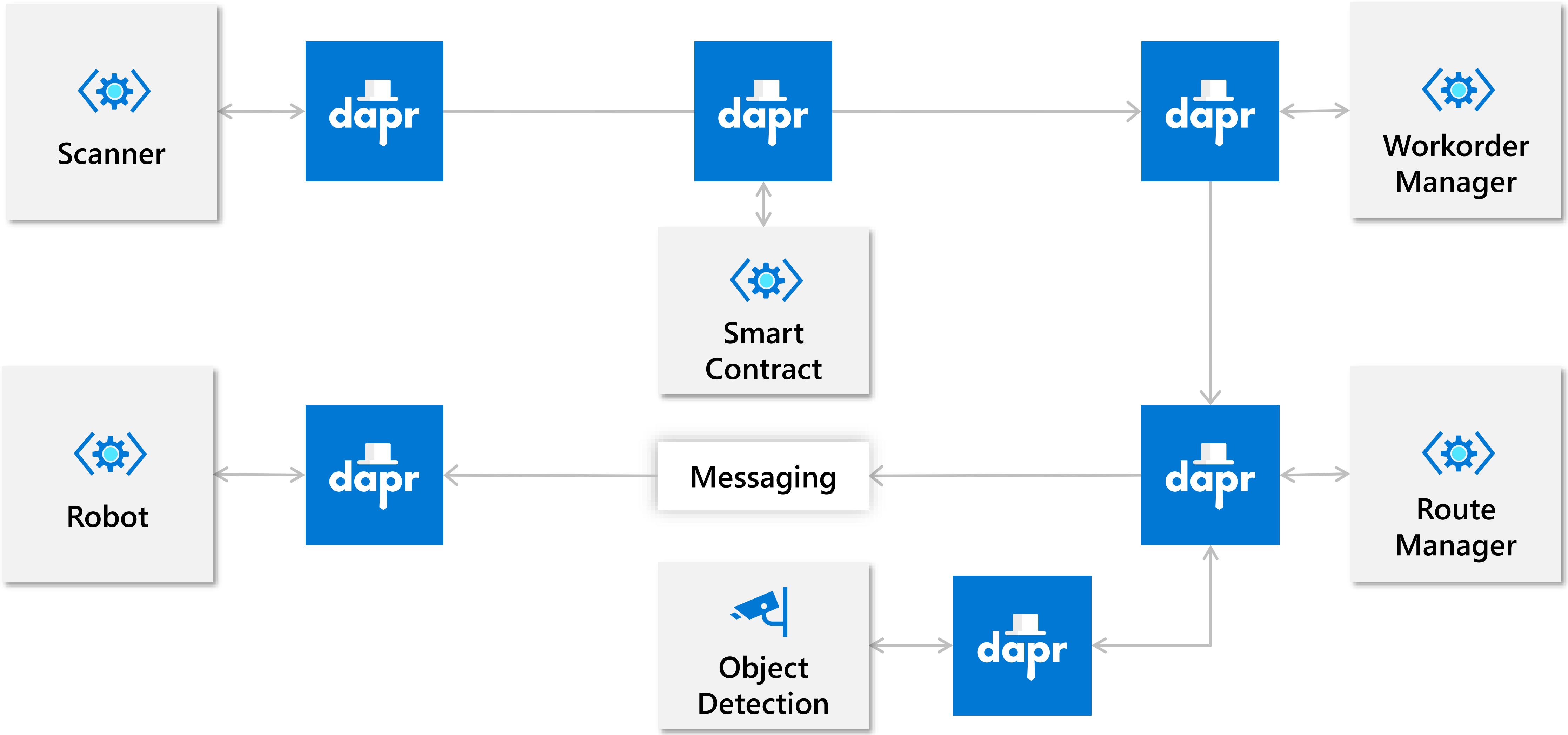
Warehouse Robotics

Incremental extensions to a legacy system



Warehouse Robotics

Incremental extensions to a legacy system



DEMO

Warehouse Robotics Orchestration

Learn more and contribute

Open
Application
Model

openappmodel.io

dapr

dapr.io

Thank you

Microsoft