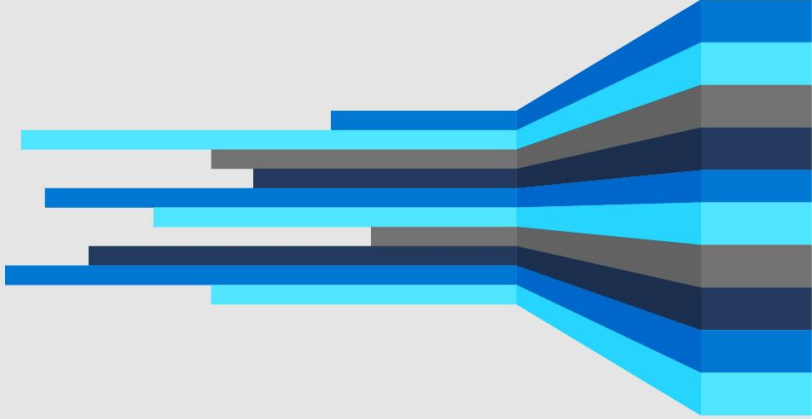
Workshop - Building cloud native applications with Distributed Application Runtime (Dapr)



Shailendra Singh Chauhan
shchauh@microsoft.com

Mark Fussell
mark.fussell@microsoft.com

Lynn Orrell
lyorrell@microsoft.com

Workshop Agenda

- Workshop's structure
  - Workshop's objectives
  - Workshop's contents
- Introduction to Dapr
  - What is Dapr
  - Why Dapr?
  - Dapr building blocks and components
  - Dapr side car architecture
- Hands on labs
- Closing notes

Workshop's Objectives

- Learn about Distributed Application Runtime(Dapr)
- Setup a local environment for Dapr
- Implement different scenarios like state management, pub-sub, service-service invocation, distributed tracing with Dapr

** We will try to do everything in two hours, but in case something is left, you can do it later.

** We will do all hands-on labs in self hosted mode

** We will not use Kubernetes or any other orchestrator as part of this workshop. Ignore the Kubernetes section of each HOL. Though you can do the same samples on Kubernetes or minikube as a stretch goal

Workshop's Contents

Contents for this workshop are hosted at GitHub at

Building cloud native applications with Distributed Application Runtime(Dapr)

https://aka.ms/ndc-dapr-workshop

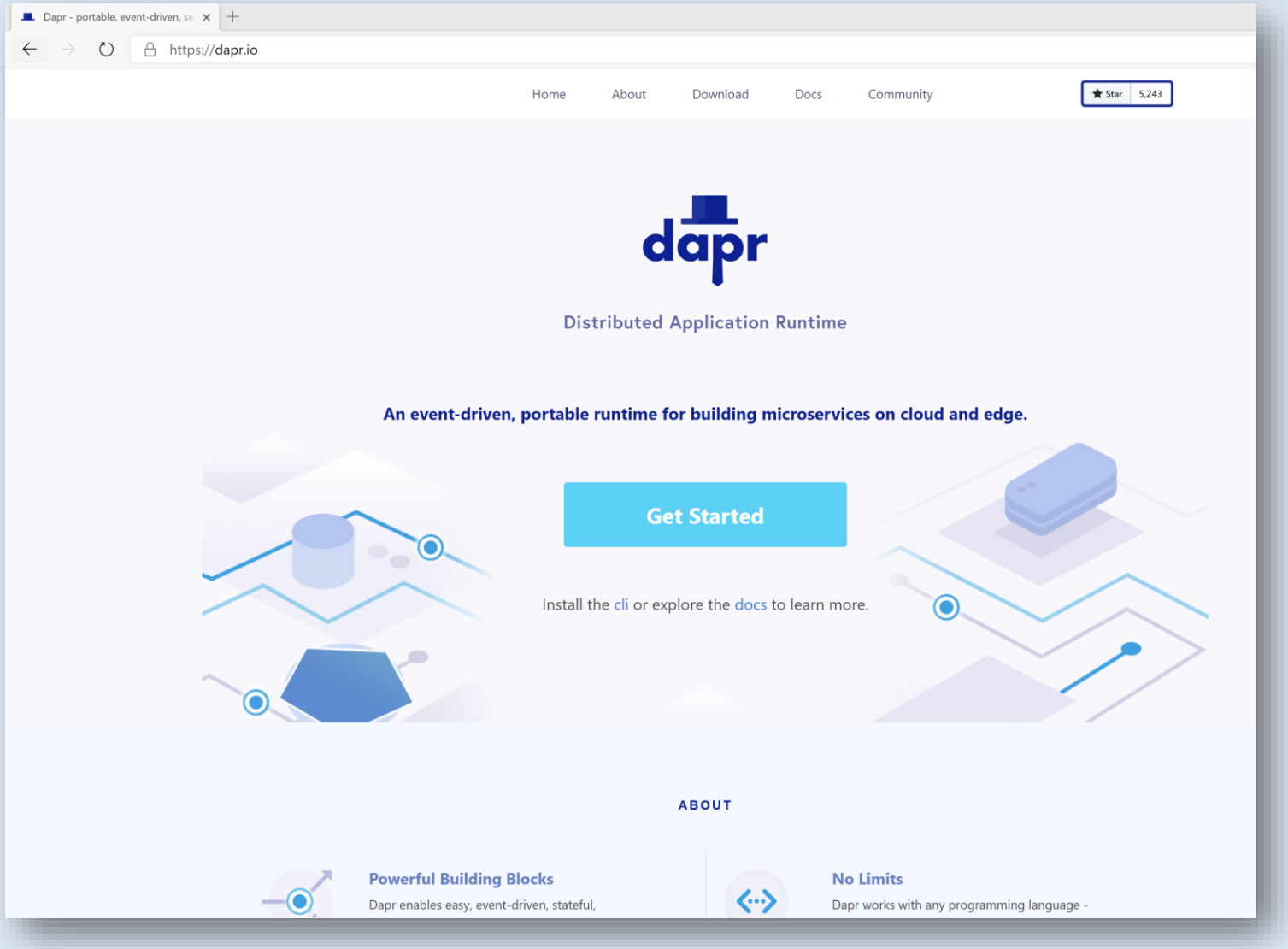
Prerequisites for the workshop

https://aka.ms/ndc-dapr-workshop-prerequisites

Distributed Application Runtime

Dapr is a portable, event-driven runtime that makes it easy for developers to build resilient, microservice applications that run on the cloud and edge

https://dapr.io



Why Dapr?

Best-practices building blocks

Any language or framework

Consistent, portable, open APIs

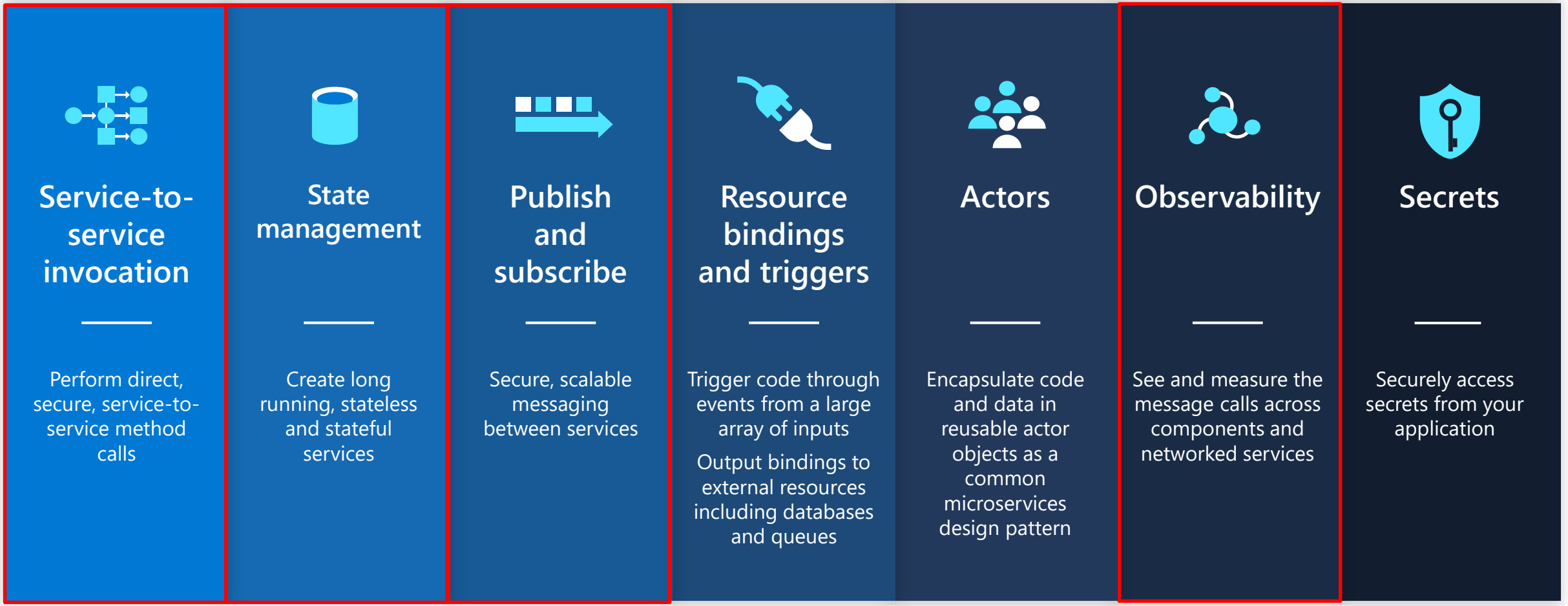
Adopt standards

Extensible and pluggable components

Platform agnostic cloud + edge

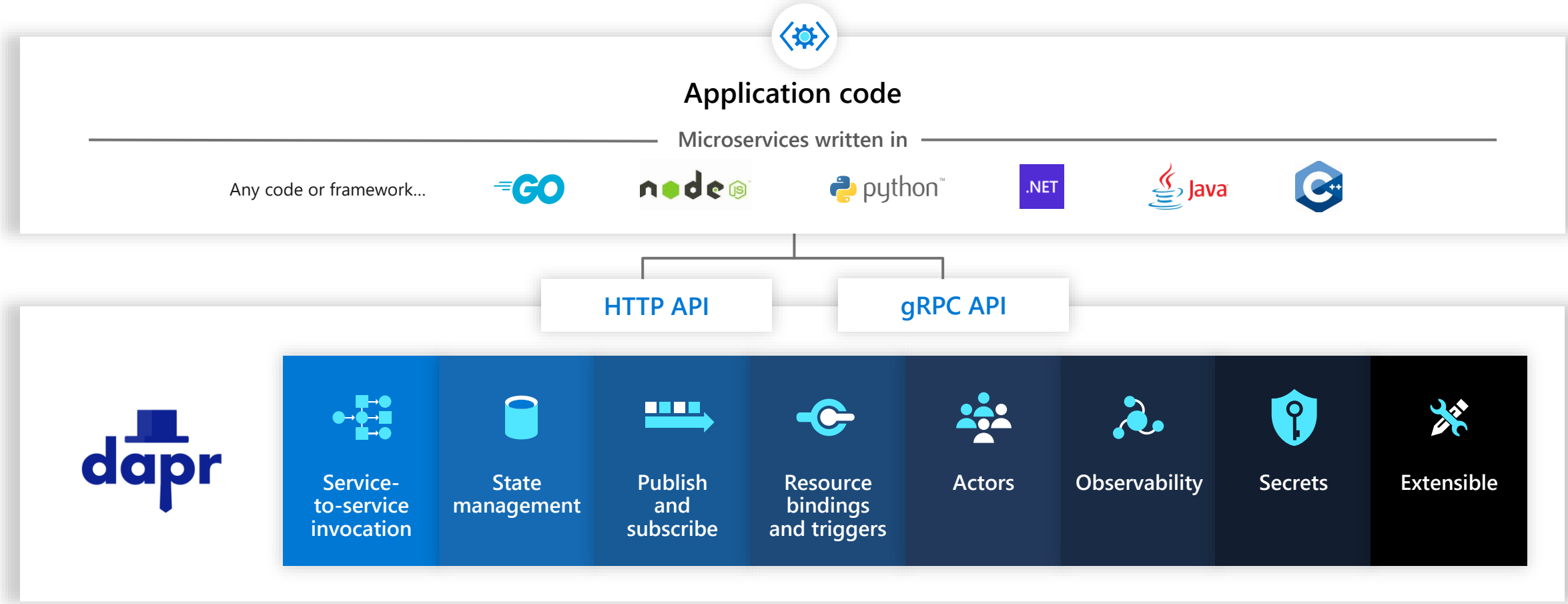
Community driven vendor neutral

Dapr building blocks

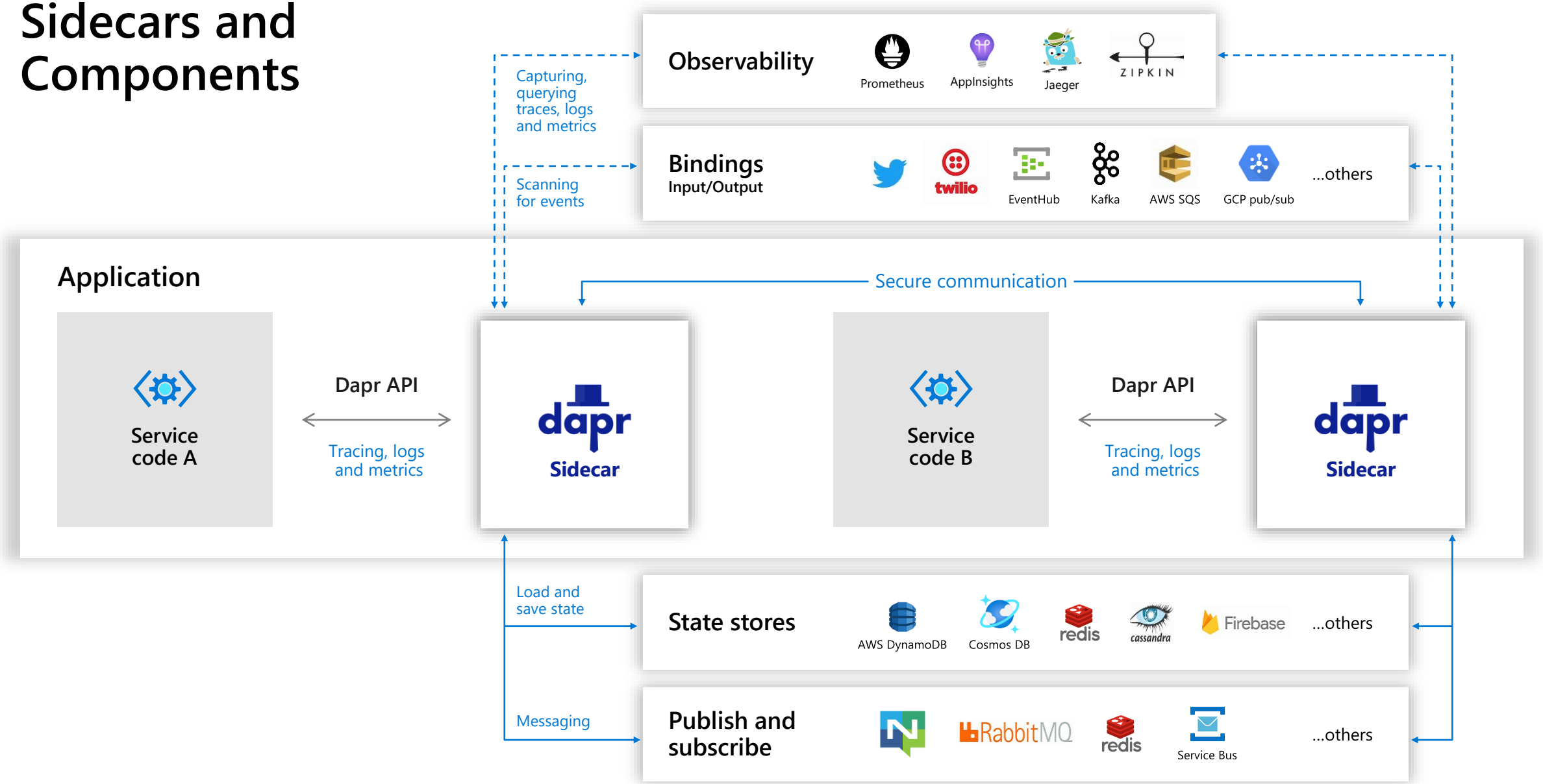


Using dapr building blocks

- 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

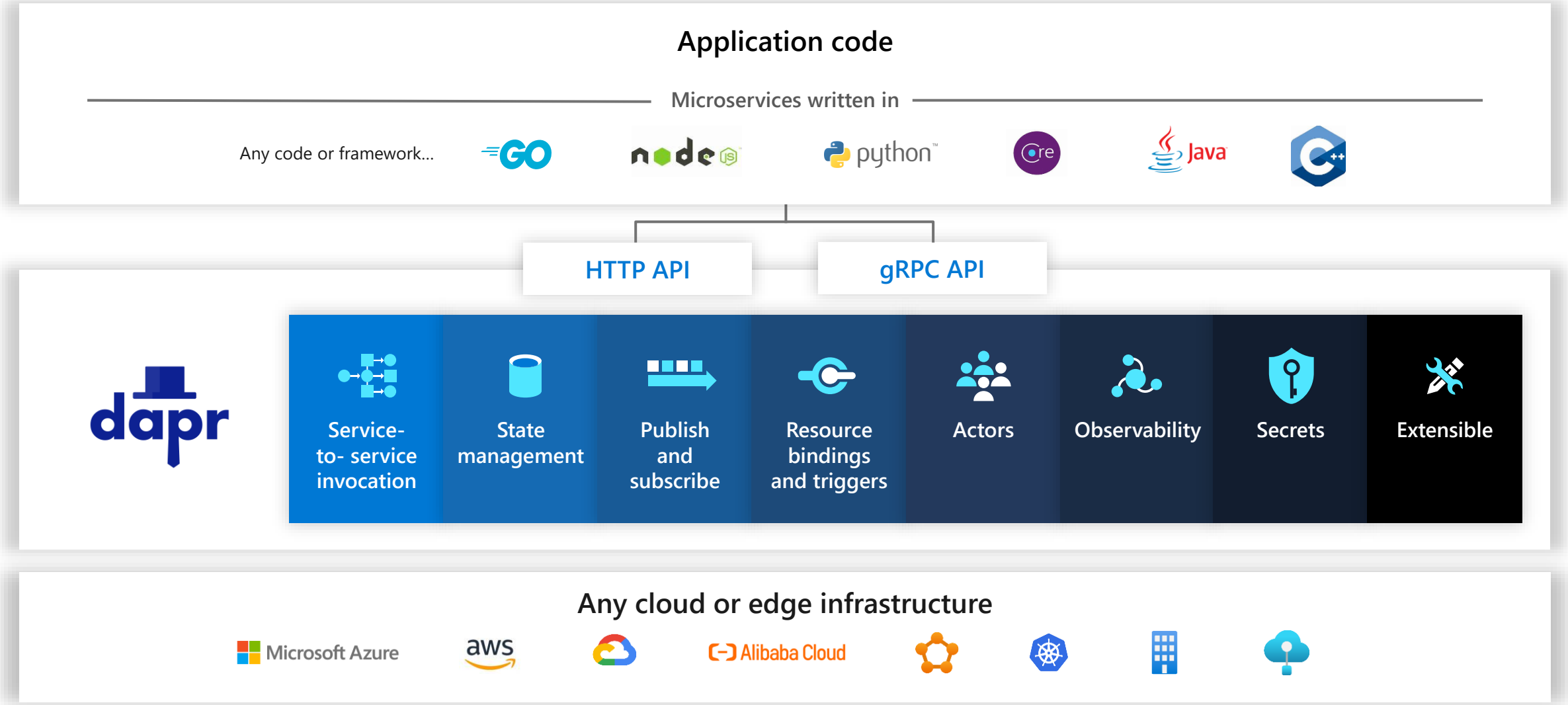


Sidecars and Components



Dapr: Distributed Application Runtime

Build apps using any language with any framework



Hands on labs

Let’s get started !

Microservice building blocks

State management: key/value

Get

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

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

App
“myApp”

Post

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

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

dapr

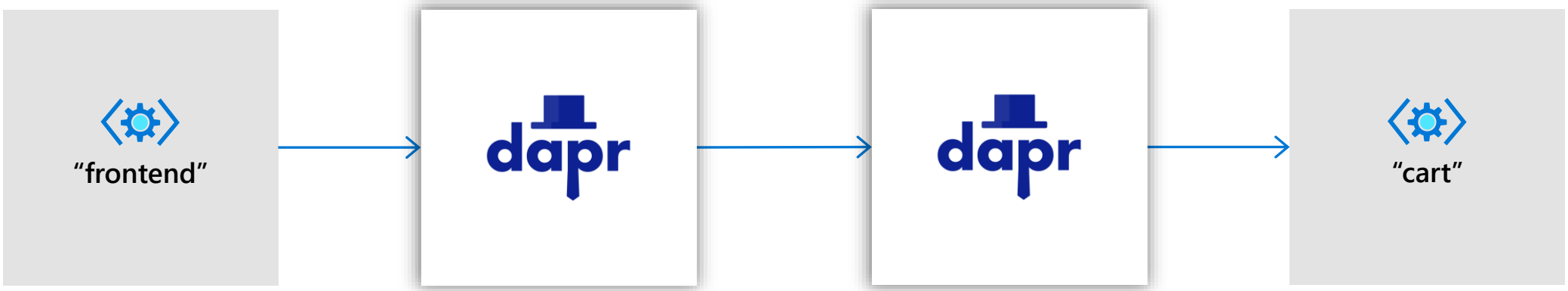
key	value
myApp-weapon	"DeathStar"
myApp-planet	{ "name": "Tatooine" }

State stores

AWS DynamoDB Cosmos DB redis cassandra Firebase ...others

Microservice building blocks

Service invocation



Post

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

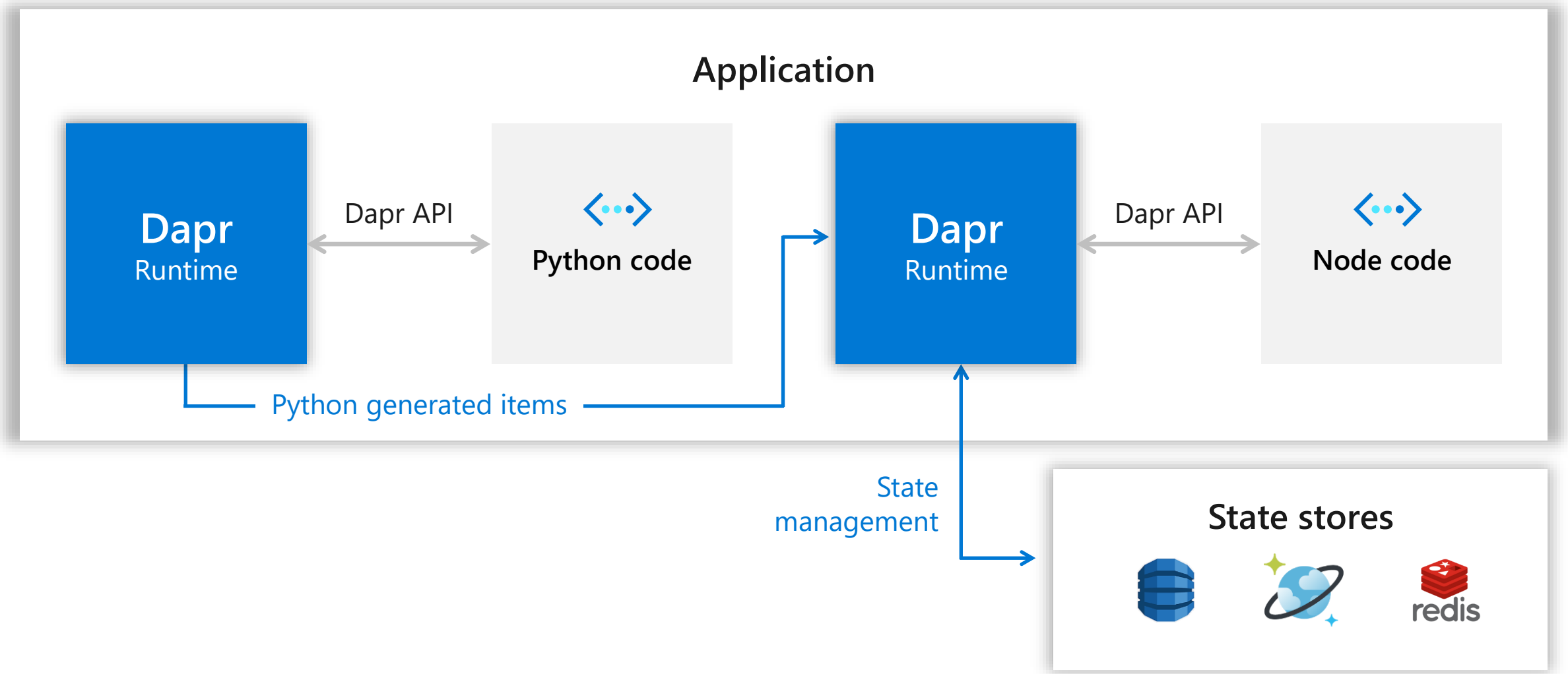
```
{
  "user":"johndoe",
  "cart":"0001"
}
```

Post

http://10.0.0.2:8000/checkout

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

HOL 1 – Hello World – Service to Service invocation and State Management



HOL 1

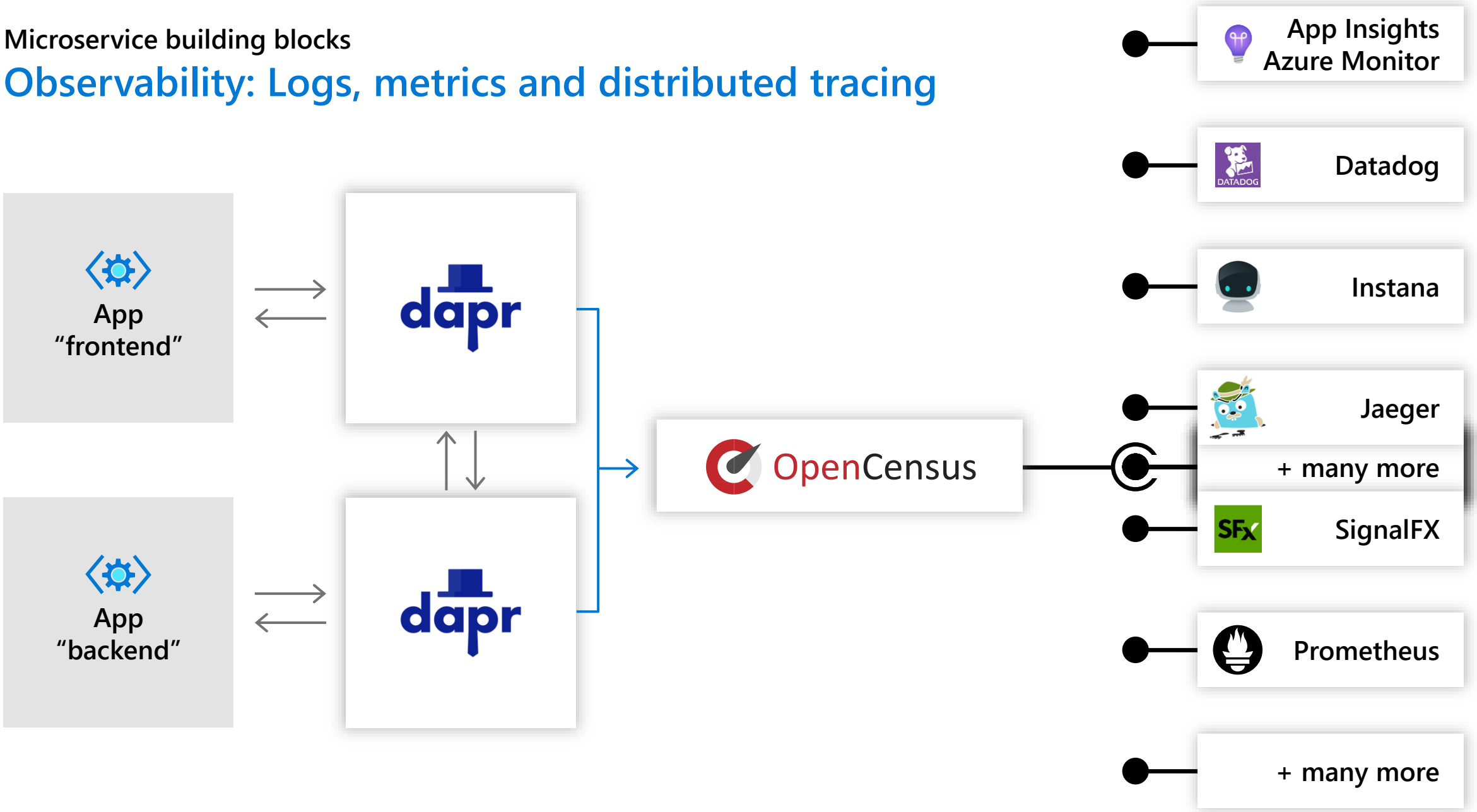
HOL # 1	HOL 1 - Hello Dapr - Implenting state management and service-service invocation
Scenario	- State management - Service-service invocation
Prerequisites	- Docker - Dapr - Git - Visual Studio Code or Visual Studio
Software needed	- Node - Python

Hands on labs

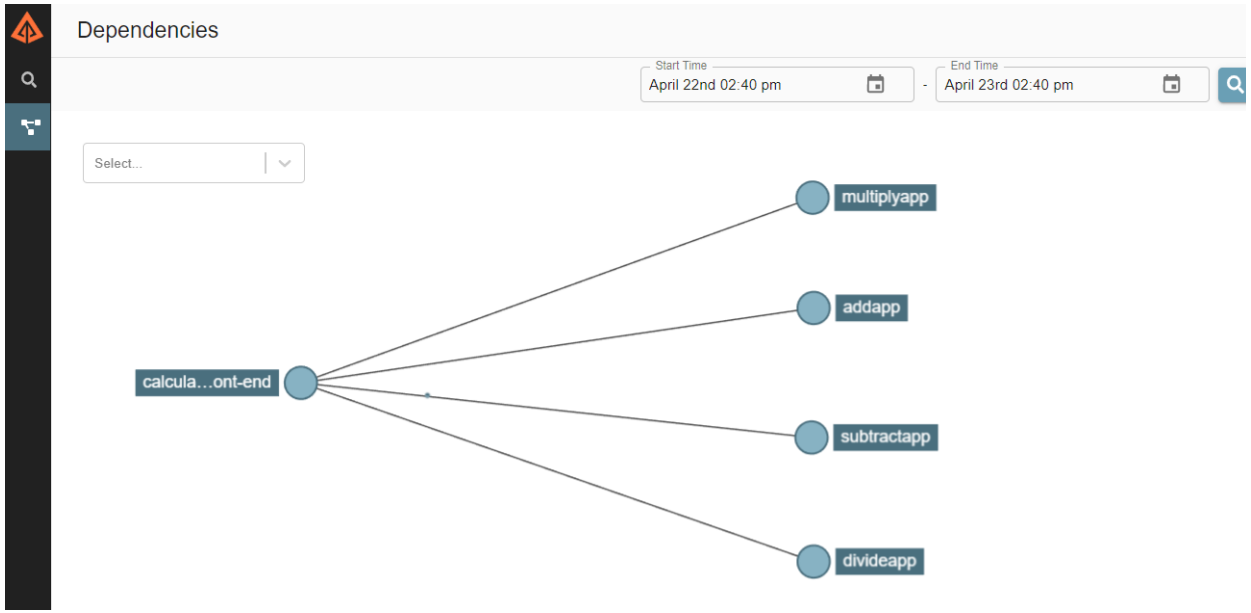
aka.ms/ndc-dapr-workshop

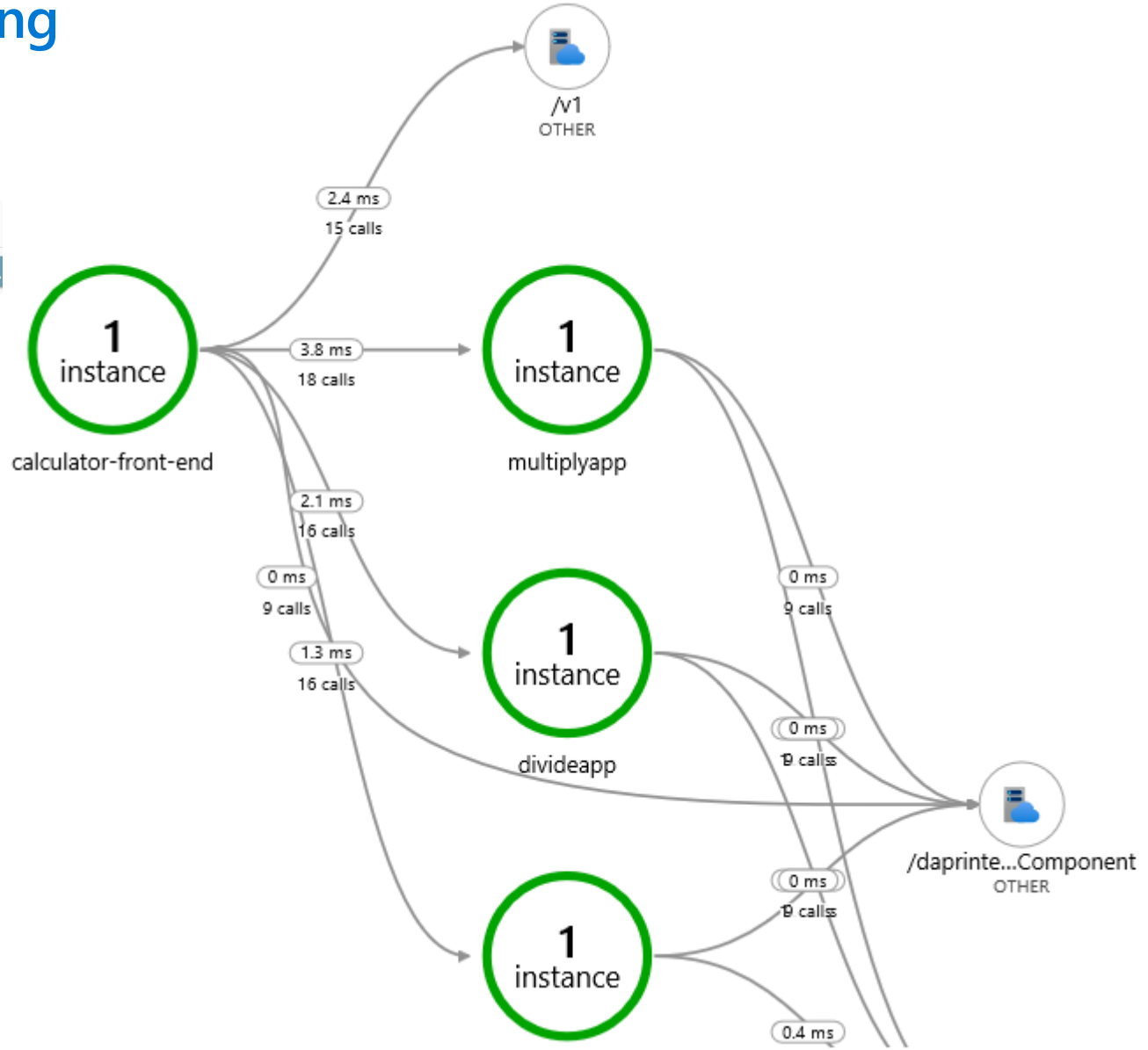
Microservice building blocks

Observability: Logs, metrics and distributed tracing



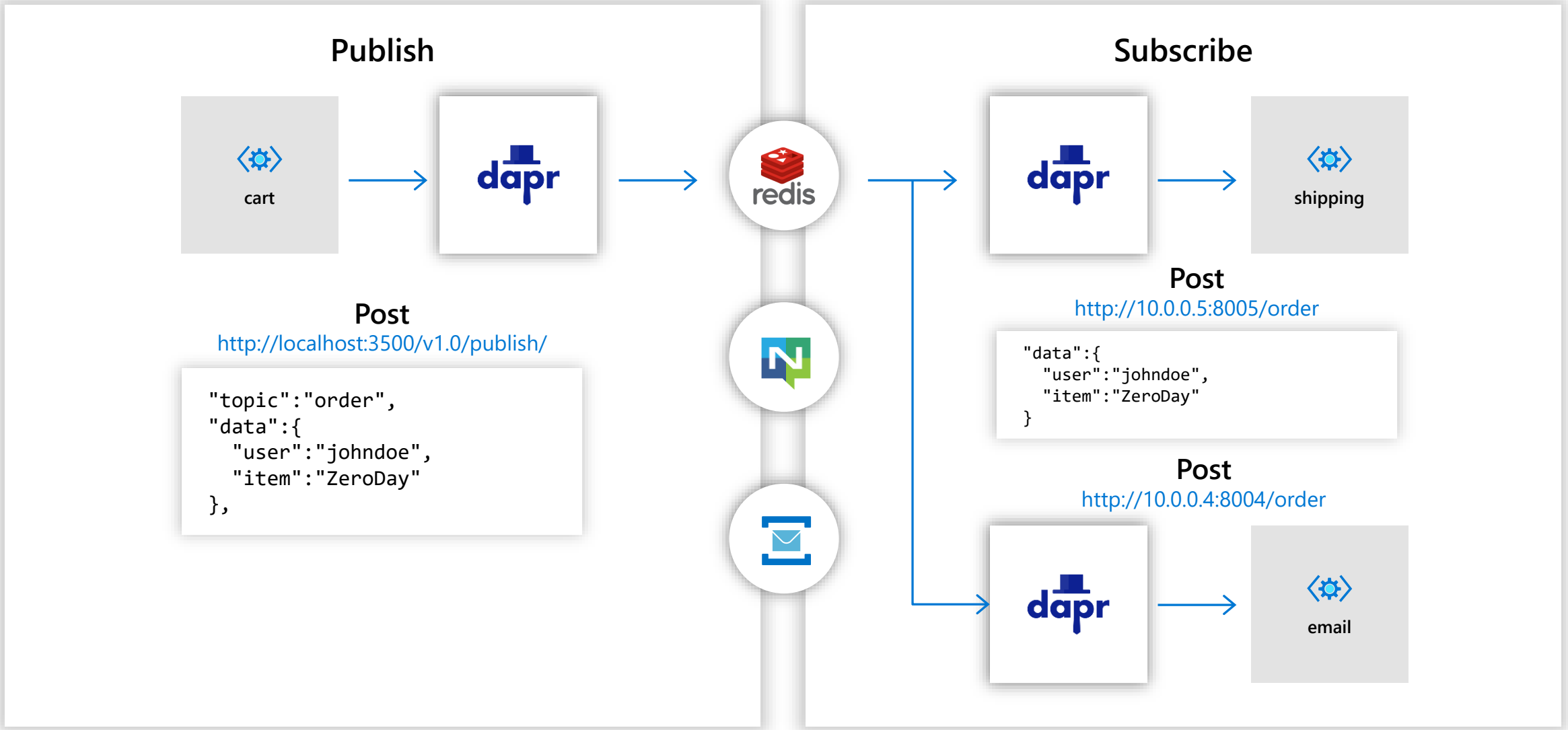
HOL 2 – Observability: Distributed Tracing



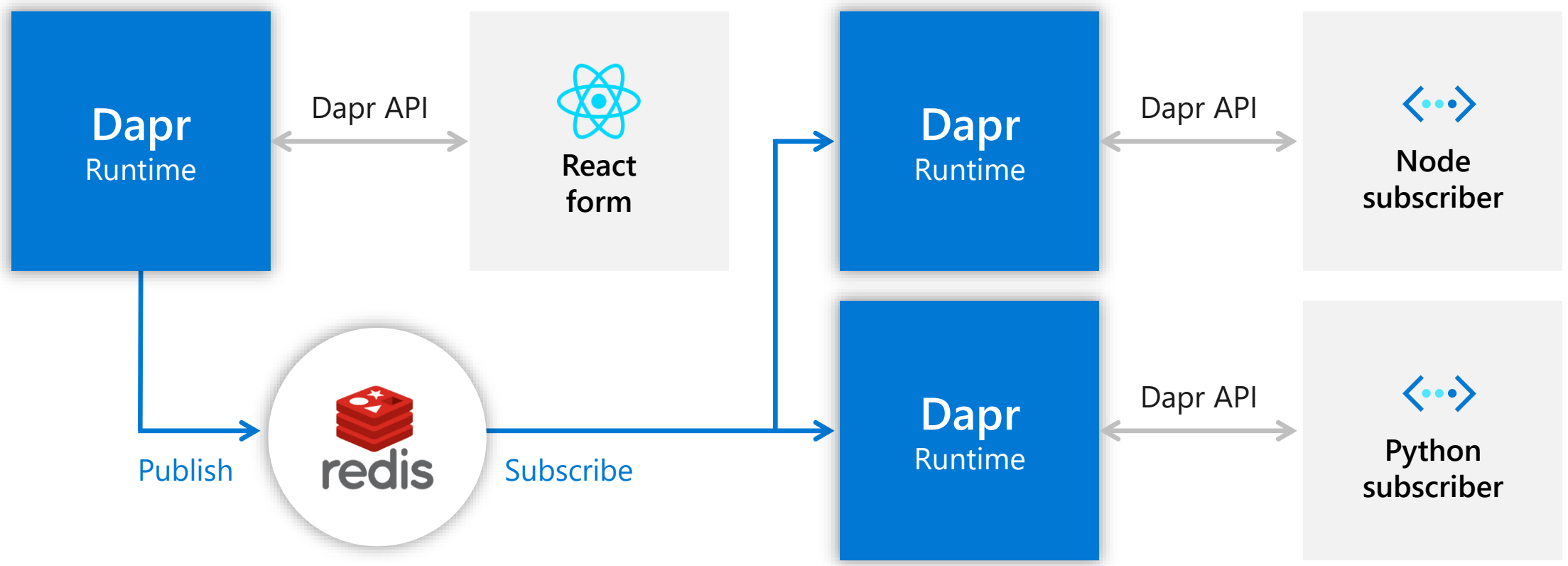


Microservice building blocks

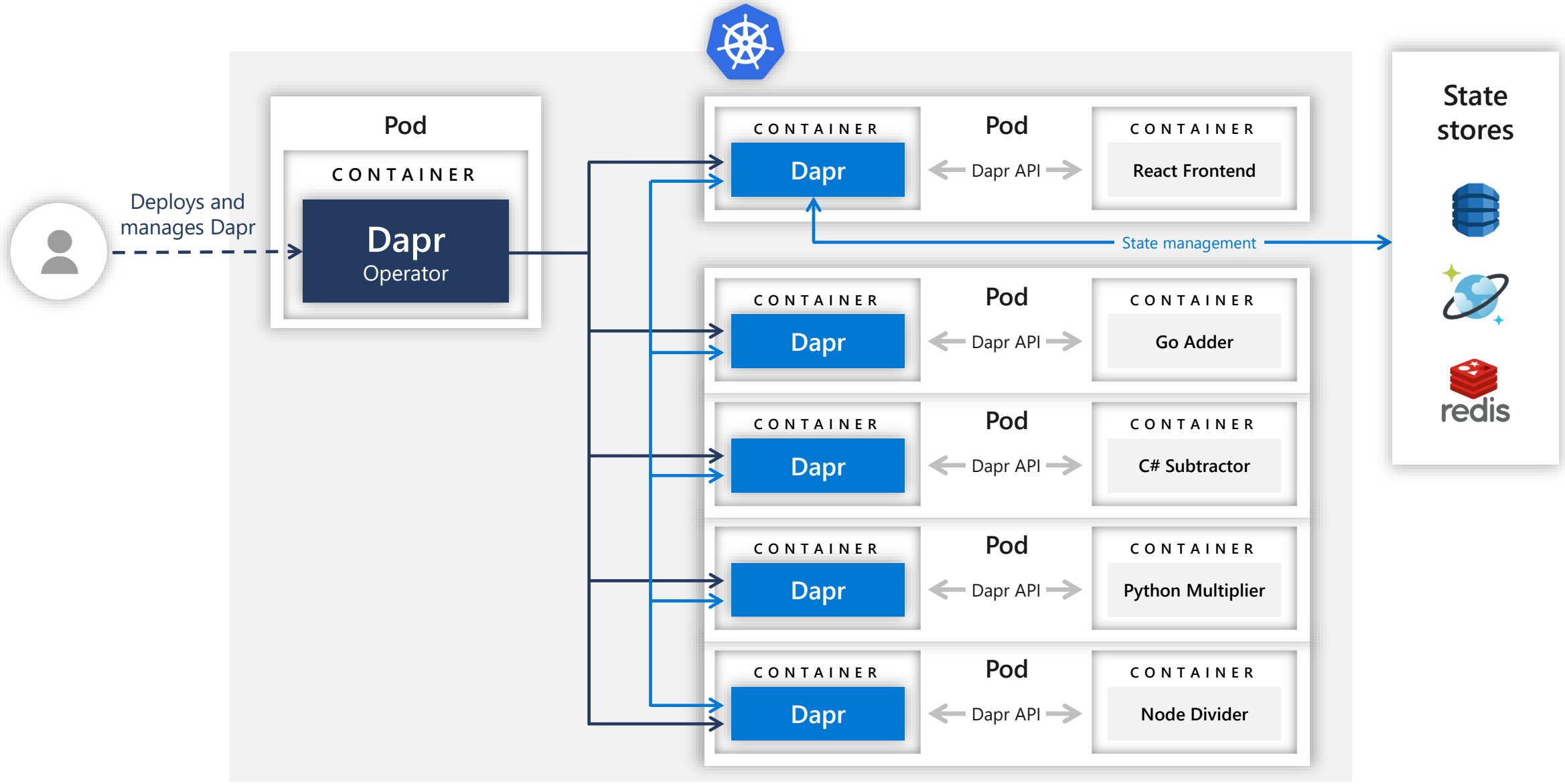
Publish and Subscribe



HOL 3 – Pub/Sub


Dapr
Runtime
Dapr API
React
form
Dapr
Runtime
Dapr API
Node
subscriber
Publish
redis
Subscribe
Dapr
Runtime
Dapr API
Python
subscriber

HOL 4 – Distributed Calculator



Next Steps

Distributed Application Runtime

Build apps using any language with any framework

Dapr language SDKs

Application code

Microservices written in

Any code or framework... GO node python .Net Java C++

HTTP API | gRPC API

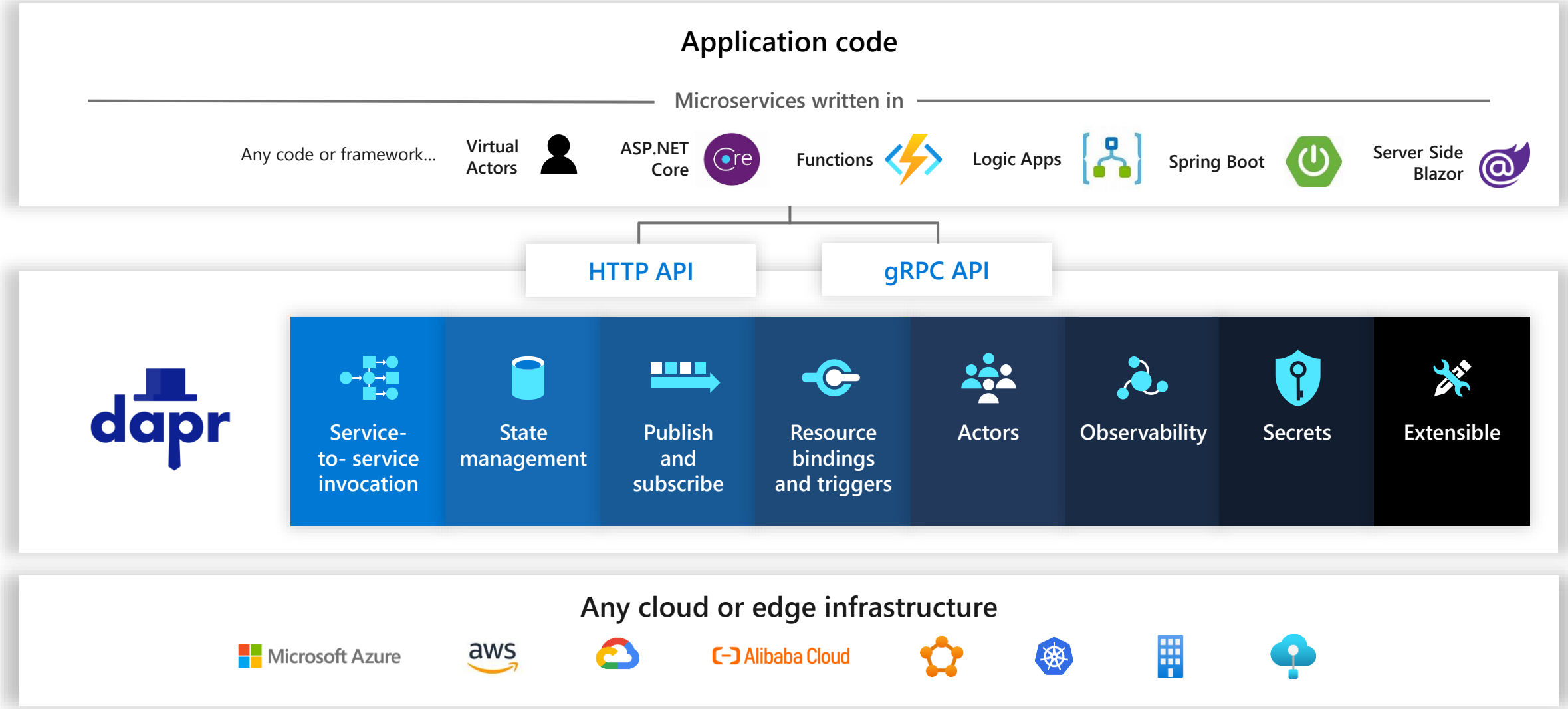
dapr

Service-to- service invocation | State management | Publish and subscribe | Resource bindings and triggers | Actors | Observability | Secrets | Extensible

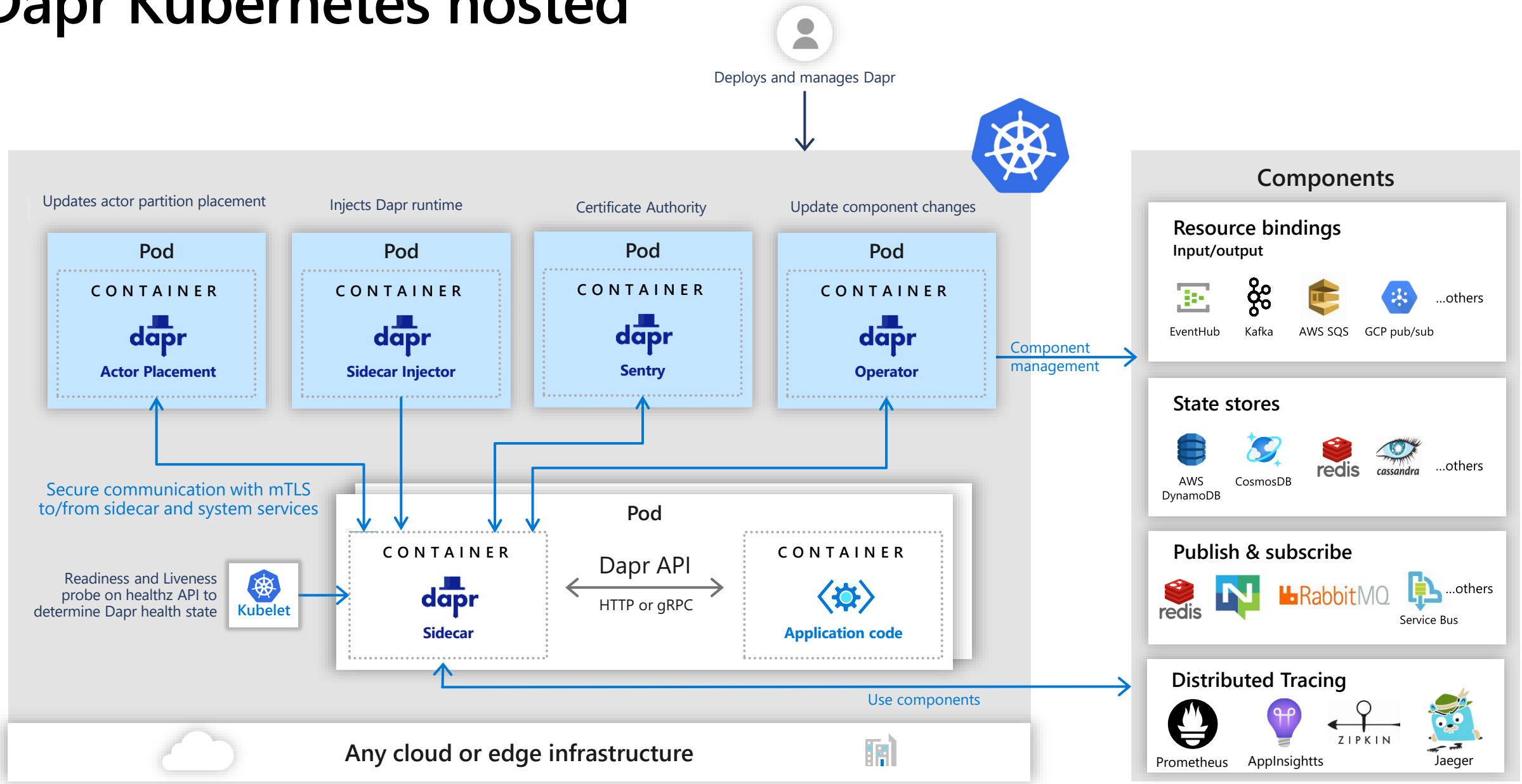
Any cloud or edge infrastructure

Microsoft Azure | aws | Alibaba Cloud

Integration with developer frameworks



Dapr Kubernetes hosted



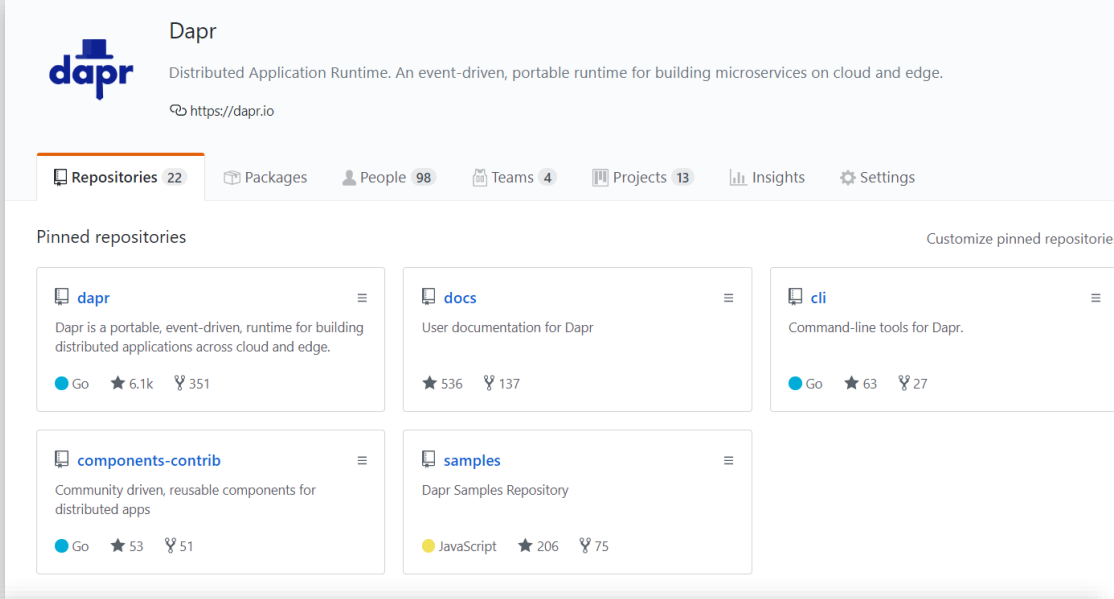
Join the community

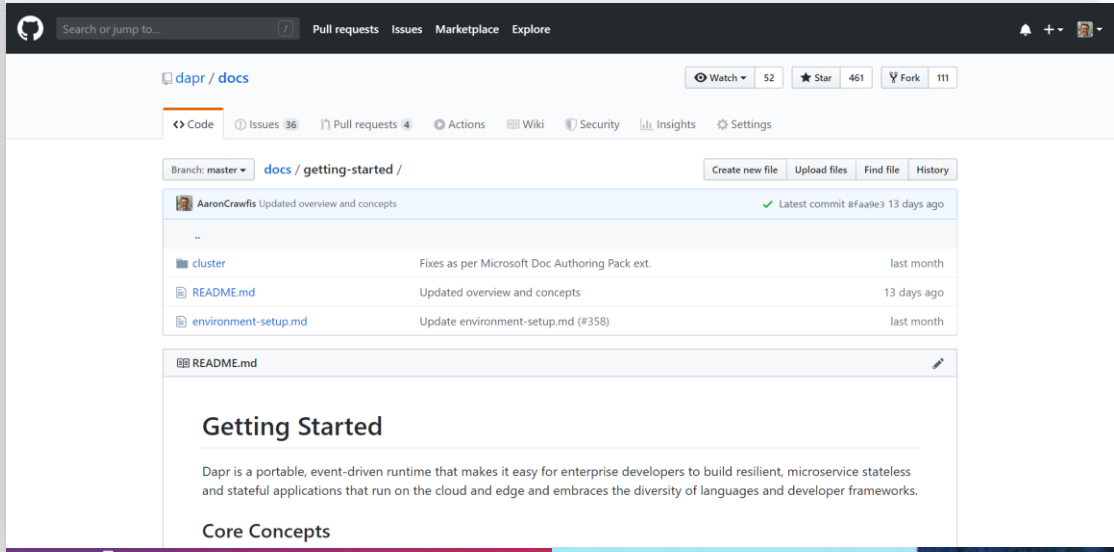
GitHub GITTER

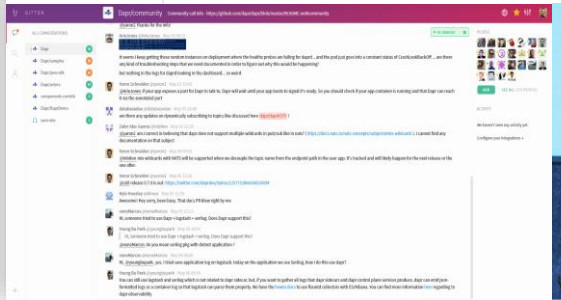
github.com/dapr

aka.ms/dapr-community

@daprdev







Thanks for joining !!

Please fill out survey*

Challenges for Developers

Being asked to develop resilient, scalable, microservice-based apps that interact with services

Use multiple languages and frameworks during development

Focus on building apps not infrastructure

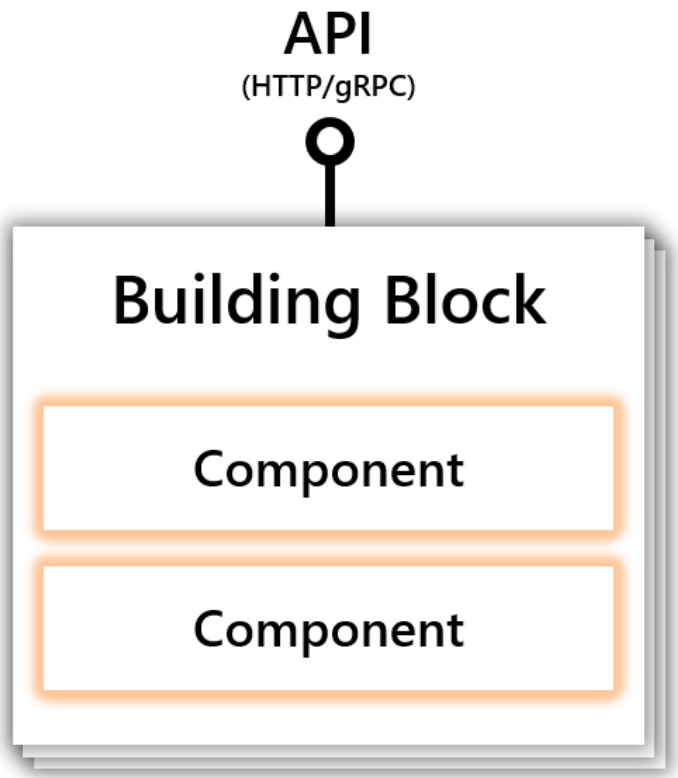
Challenges with Microservices applications

Have limited tools and programming model runtimes to build distributed applications

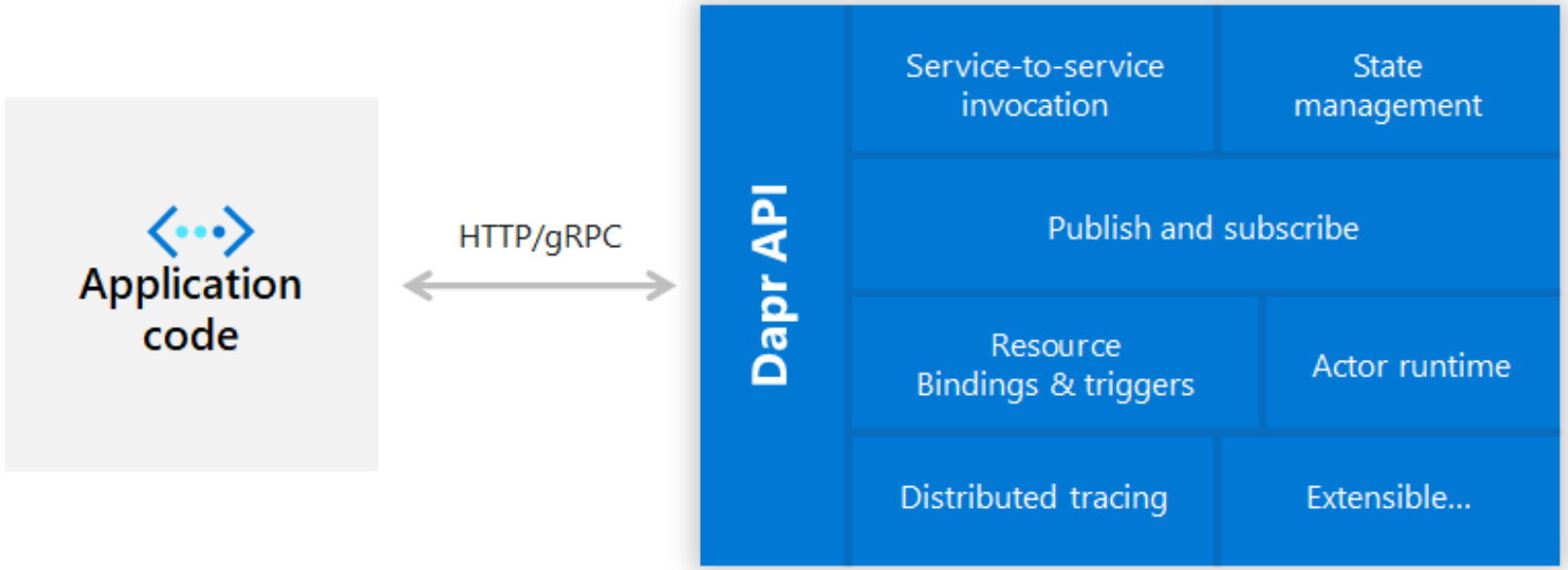
Programming model runtimes have narrow language support and tightly controlled feature sets

Runtimes only target specific infrastructure platforms with limited code portability across clouds and edge

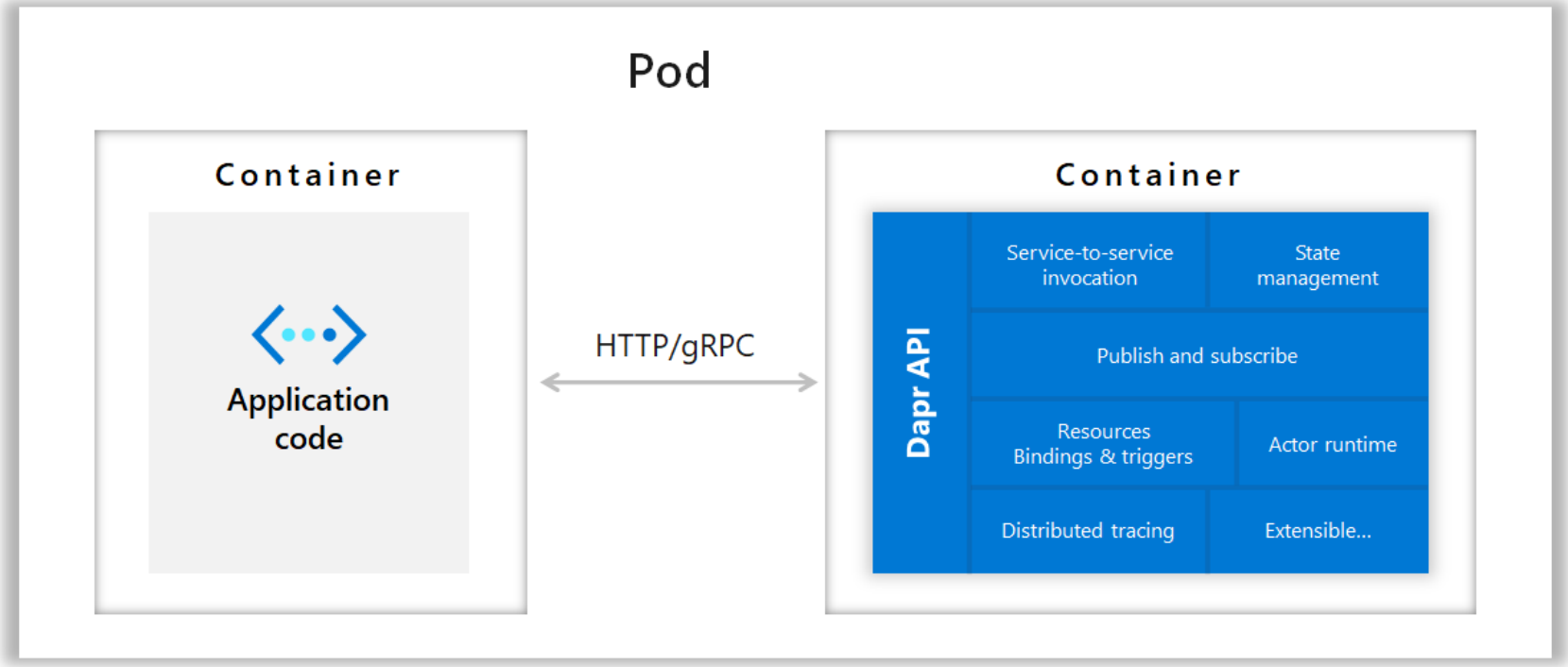
Dapr Building blocks



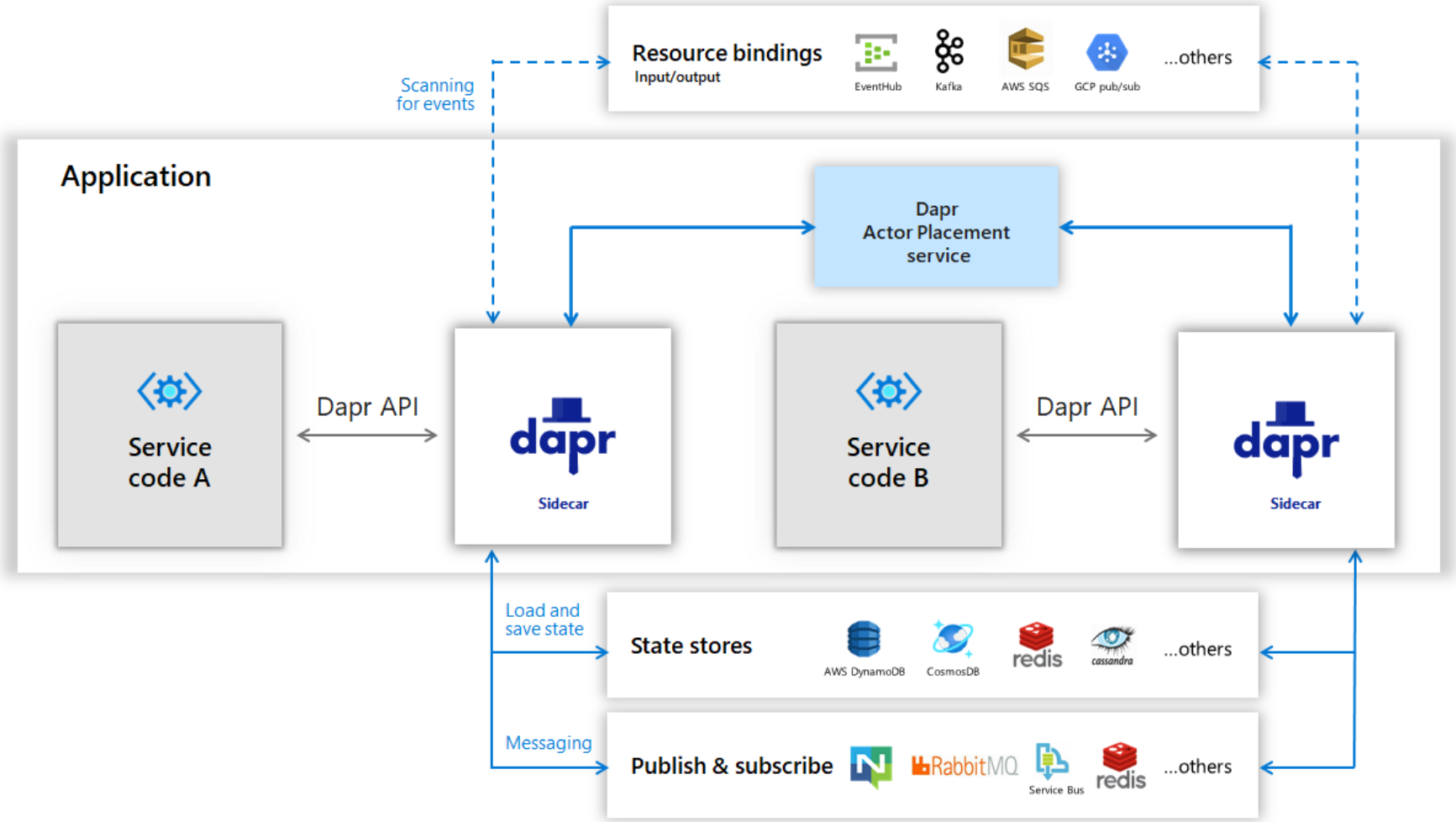
Self hosted mode



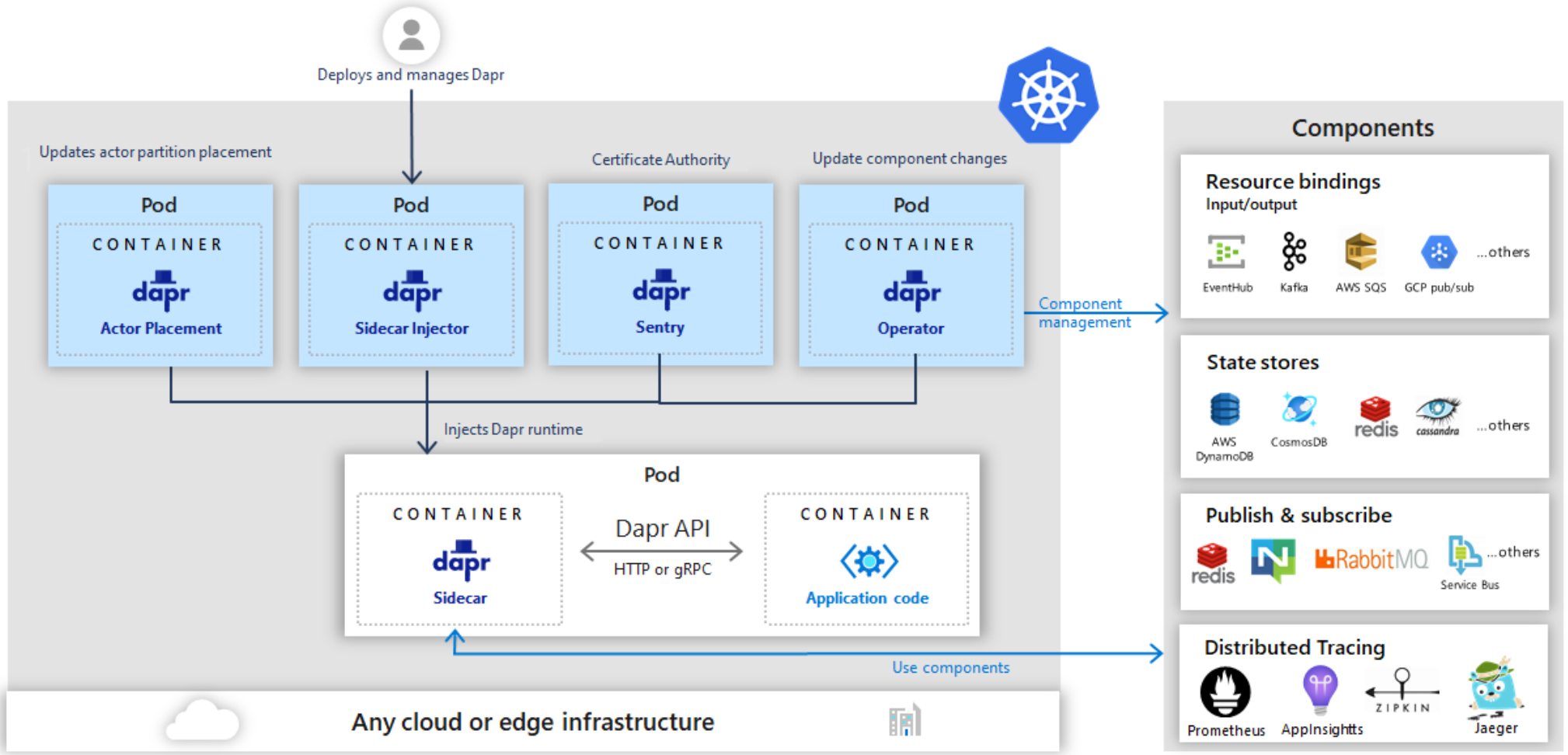
Kubernetes mode



Running Dapr on a local developer machine in self hosted mode



Running Dapr in Kubernetes mode



```
annotations:
  dapr.io/enabled: "true"
  dapr.io/id: "nodeapp"
  dapr.io/port: "3000"
  dapr.io/config: "tracing"
```

Dapr building block endpoints

Building Block	Endpoint	Description
Service-to-Service Invocation	/v1.0/invoke	Service invocation enables applications to communicate with each other through well-known endpoints in the form of http or gRPC messages. Dapr provides an endpoint that acts as a combination of a reverse proxy with built-in service discovery, while leveraging built-in distributed tracing and error handling.
State Management	/v1.0/state	Application state is anything an application wants to preserve beyond a single session. Dapr provides a key/value-based state API with pluggable state stores for persistence.
Publish and Subscribe	/v1.0/publish /v1.0/subscribe	Pub/Sub is a loosely coupled messaging pattern where senders (or publishers) publishes messages to a topic, to which subscribers subscribe. Dapr supports the pub/sub pattern between applications.
Resource Bindings	/v1.0/bindings	A binding provides a bi-directional connection to an external cloud/on-premise service or system. Dapr allows you to invoke the external service through the Dapr binding API, and it allows your application to be triggered by events sent by the connected service.
Actors	/v1.0/actors	An actor is an isolated, independent unit of compute and state with single-threaded execution. Dapr provides an actor implementation based on the Virtual Actor pattern which provides a single-threaded programming model and where actors are garbage collected when not in use. See * Actor Overview
Observability	NA	Dapr system components and runtime emit metrics, logs, and traces to debug, operate and monitor Dapr system services, components and user applications.
Secrets	/v1.0/secrets	Dapr offers a secrets building block API and integrates with secret stores such as Azure Key Vault and Kubernetes to store the secrets. Service code can call the secrets API to retrieve secrets out of the Dapr supported secret stores.

Visual Studio Extensions used

- VS Code Icons
- Azure
- C#
- Node
- Python
- Go
- REST Client