Dapr

Lego for Microservices

Mark Chmarny
Principal Program Manager
Office of CTO, Azure
@mchmarny

Context

Developer Friction

Runtimes target specific infrastructure, limit portability without rewrite

Platforms have narrow language support, tightly controlled feature sets

Gap between existing code and modern architecture increasingly larger

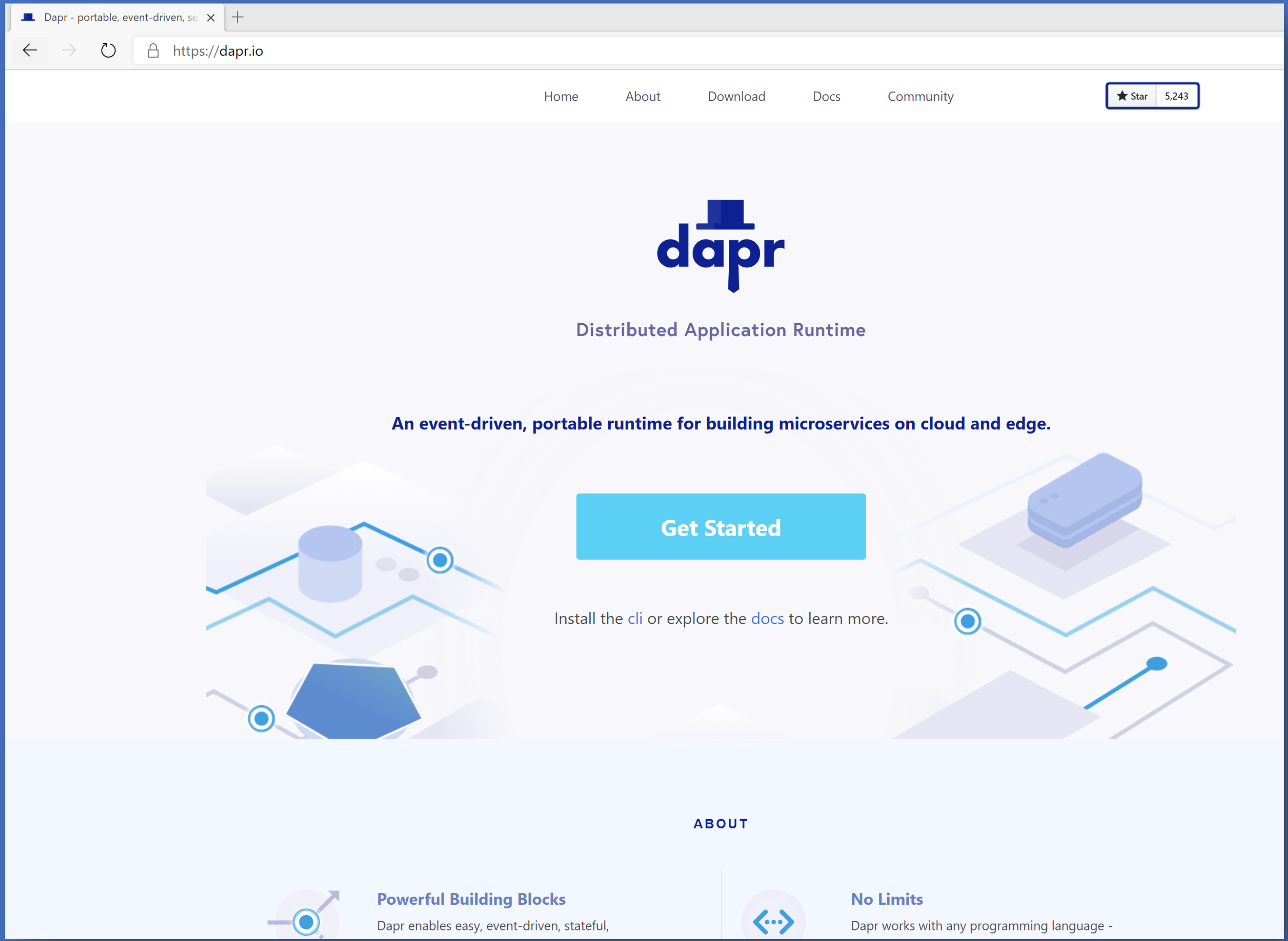
Dapr

dapr

Distributed Application Runtime

dapr.io

Dapr project transitioning to open governance



Distributed Application Runtime

An event-driven, portable runtime for building microservices on cloud and edge.

Install the cli or explore the docs to learn more.

ABOUT

Powerful Building Blocks

Dapr enables easy, event-driven, stateful,

No Limits

Dapr works with any programming language -

GitHub | github.com/dapr

1
Year

11
Releases

10M
Image Pulls

70+
Components

>450
Contributors

7.8K
GitHub Stars

Dapr Design Principles

Any Language
Any Framework

Opt-in
Building Blocks

Extensible
and Pluggable

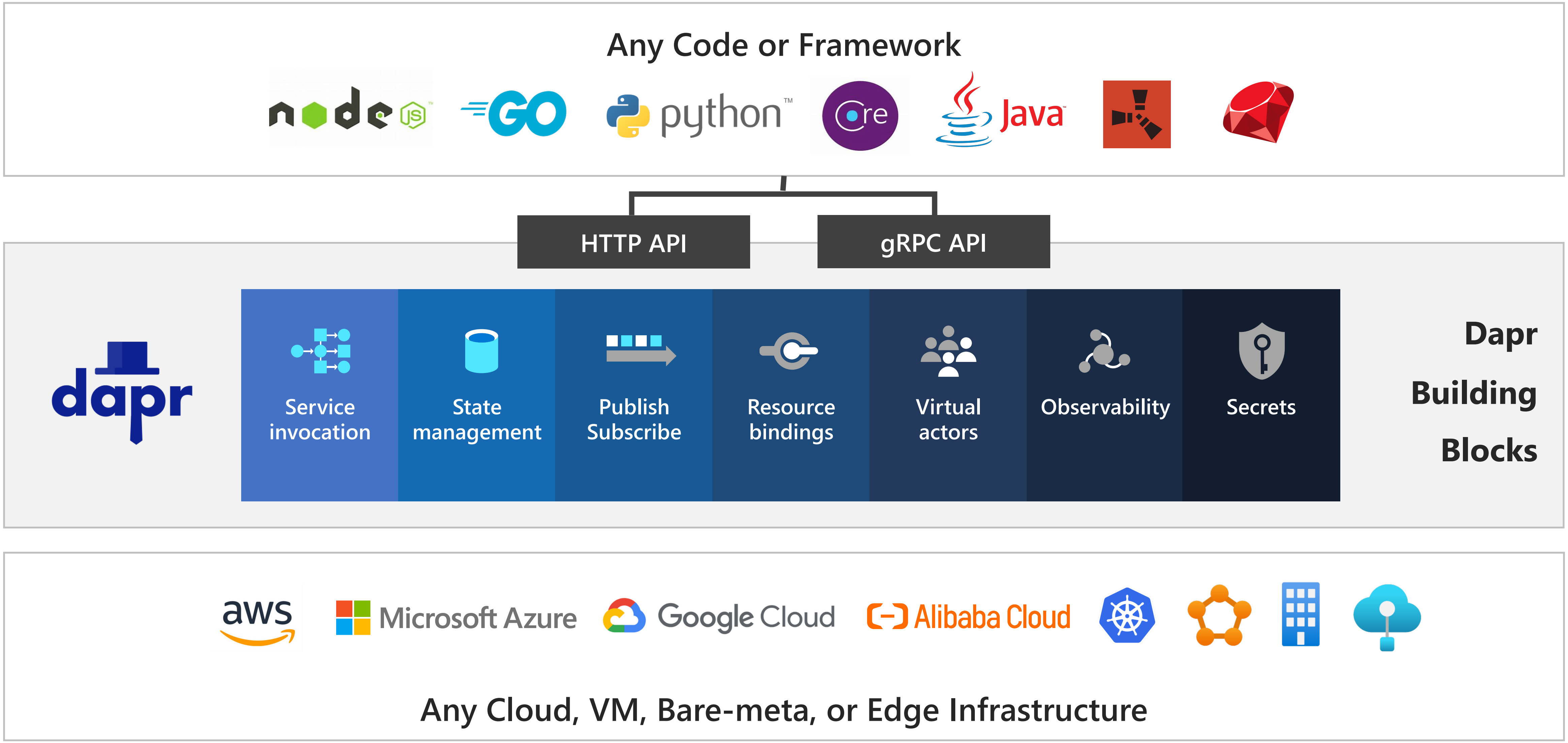
Consistent, Portable,
Open APIs

Platform Agnostic
VM, Cloud, BM, Edge

Idiomatic, Based on
Common Practices

Dapr Logical Layers

Distributed Application Runtime



Any Code or Framework

HTTP API

gRPC API

Service invocation

State management

Publish Subscribe

Resource bindings

Virtual actors

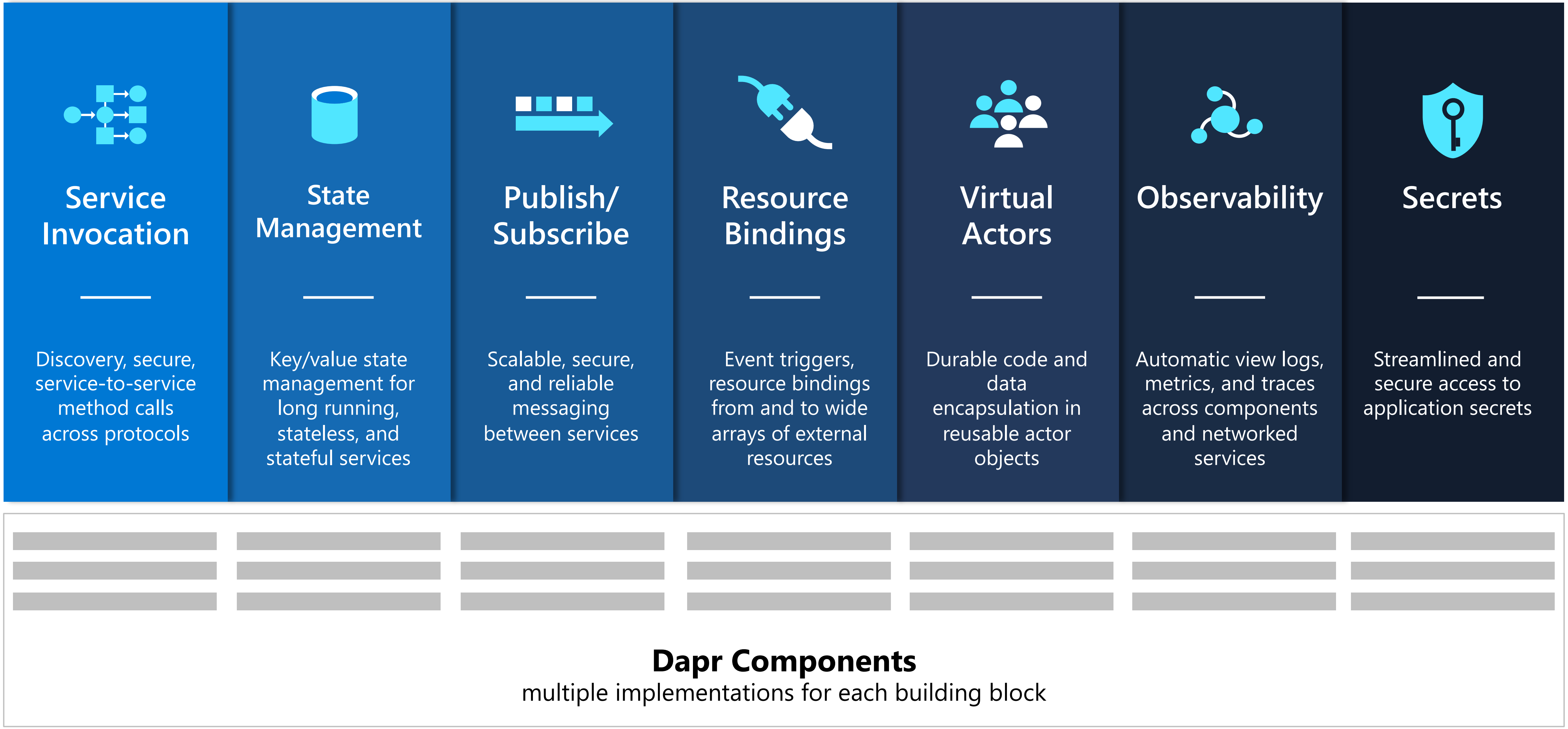
Observability

Secrets

Dapr Building Blocks

Any Cloud, VM, Bare-meta, or Edge Infrastructure

Dapr Building Blocks



Service Invocation

Discovery, secure, service-to-service method calls across protocols

State Management

Key/value state management for long running, stateless, and stateful services

Publish/ Subscribe

Scalable, secure, and reliable messaging between services

Resource Bindings

Event triggers, resource bindings from and to wide arrays of external resources

Virtual Actors

Durable code and data encapsulation in reusable actor objects

Observability

Automatic view logs, metrics, and traces across components and networked services

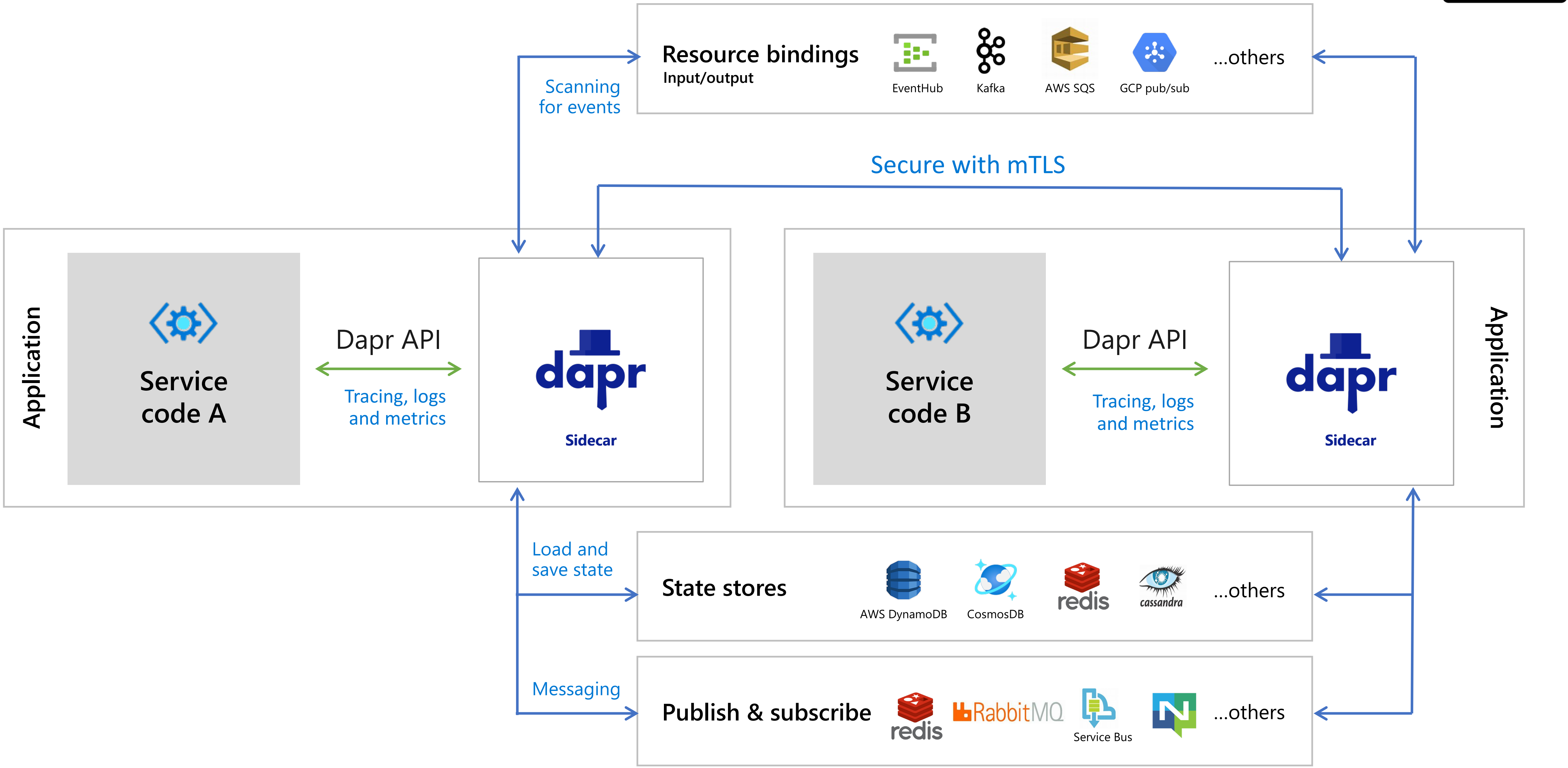
Secrets

Streamlined and secure access to application secrets

Dapr Components

multiple implementations for each building block

Dapr Sidecar Architecture



Resource bindings
Input/output

EventHub

Kafka

AWS SQS

GCP pub/sub

...others

Scanning for events

Secure with mTLS

Application

Service code A

Dapr API

Tracing, logs and metrics

dapr

Sidecar

Service code B

Dapr API

Tracing, logs and metrics

dapr

Sidecar

Application

Load and save state

State stores

AWS DynamoDB

CosmosDB

redis

cassandra

...others

Messaging

Publish & subscribe

redis

RabbitMQ

Service Bus

...others

Dapr Sidecar

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″
…
```

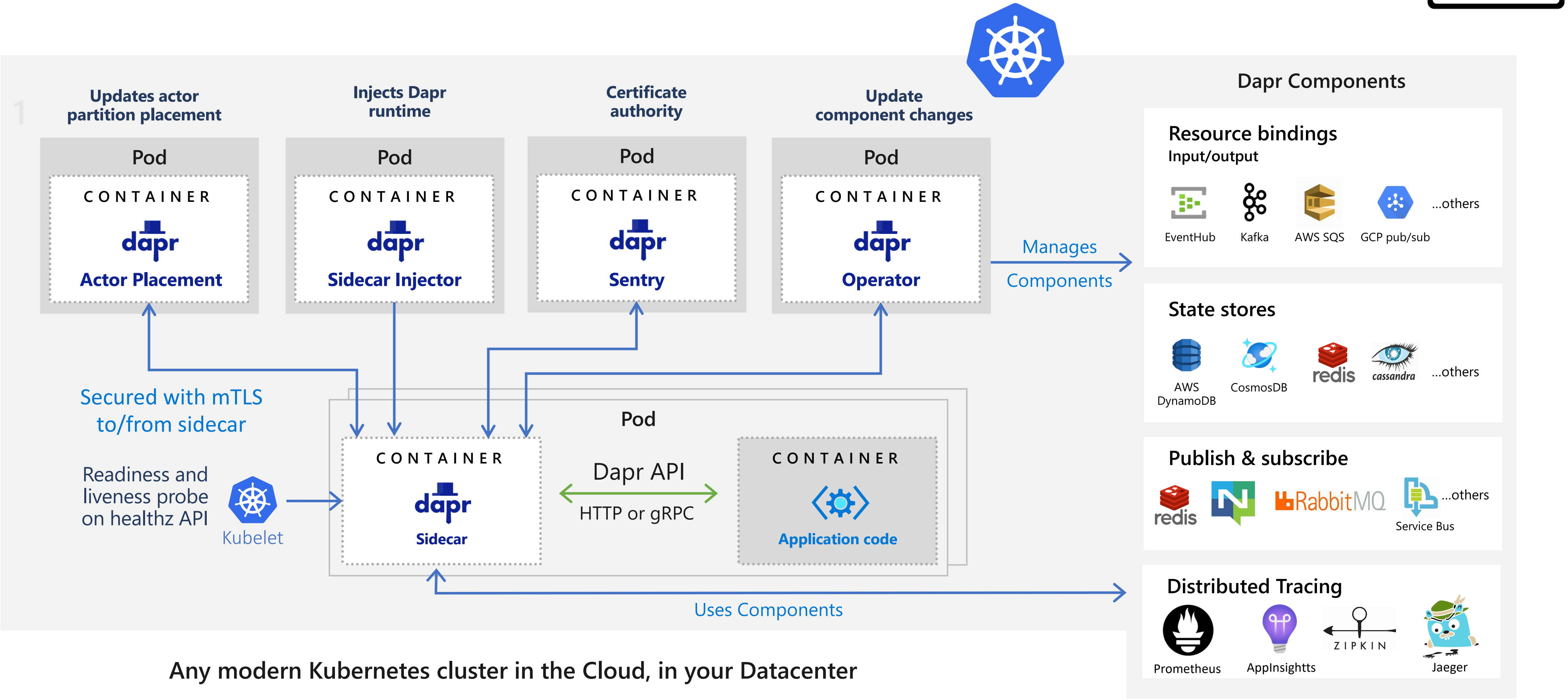
Standalone (self-hosted)

```
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 \
         node app.js
```

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

Dapr on Kubernetes

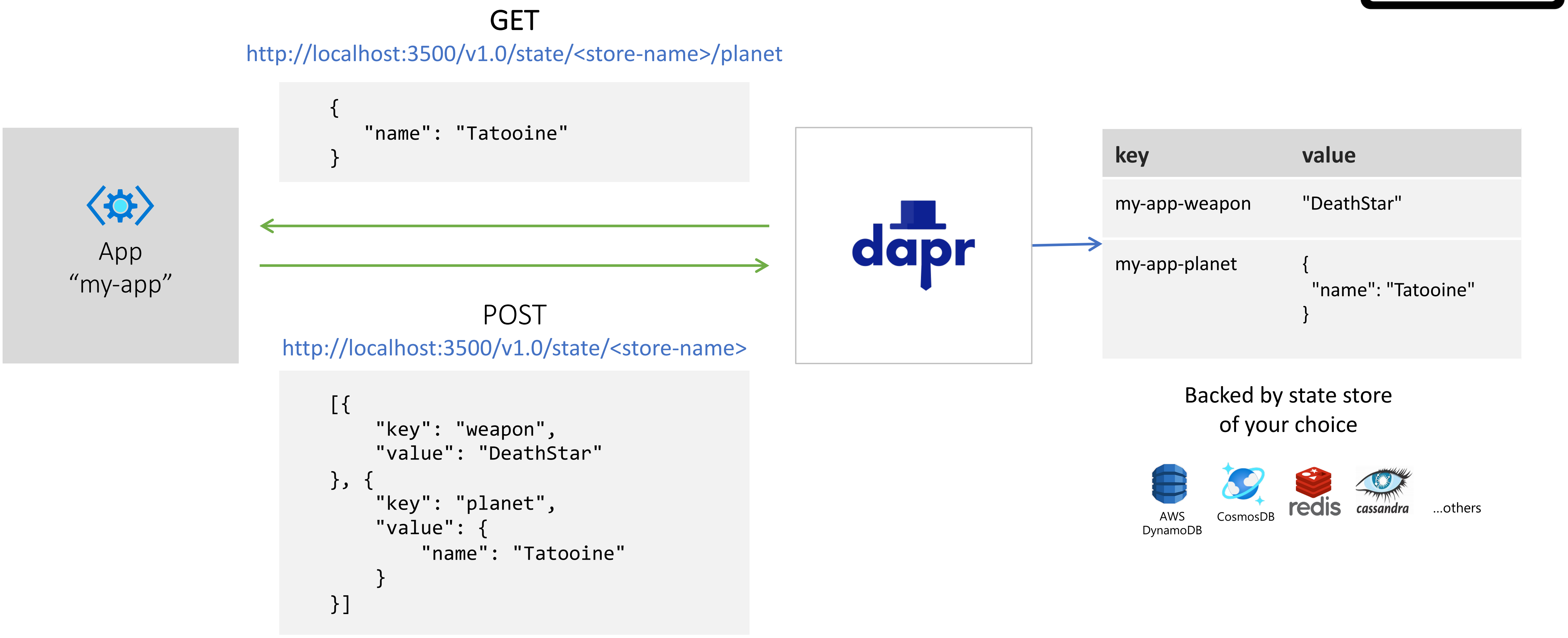


Any modern Kubernetes cluster in the Cloud, in your Datacenter

Dapr Building Blocks

Dapr building blocks

Key/value state management



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"
    }
}]
```

key	value
my-app-weapon	"DeathStar"

my-app-planet	{
	"name": "Tatooine"
	}

Backed by state store
of your choice

Dapr building blocks

Key/value state management

```
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: corpdb
spec:
  type: state.mongodb
  metadata:
  - name: host
    value: headless.mongo.svc.cluster.local:27017
  - name: username
    value: user1
  - name: password
    secretKeyRef:
      name: mongo-secret
      key: mongodb-password
  - name: databaseName
    value: orders
  - name: collectionName
    value: processed
```

POST
/v1.0/state/**corpdb**

GET
/v1.0/state/**corpdb**/<key>

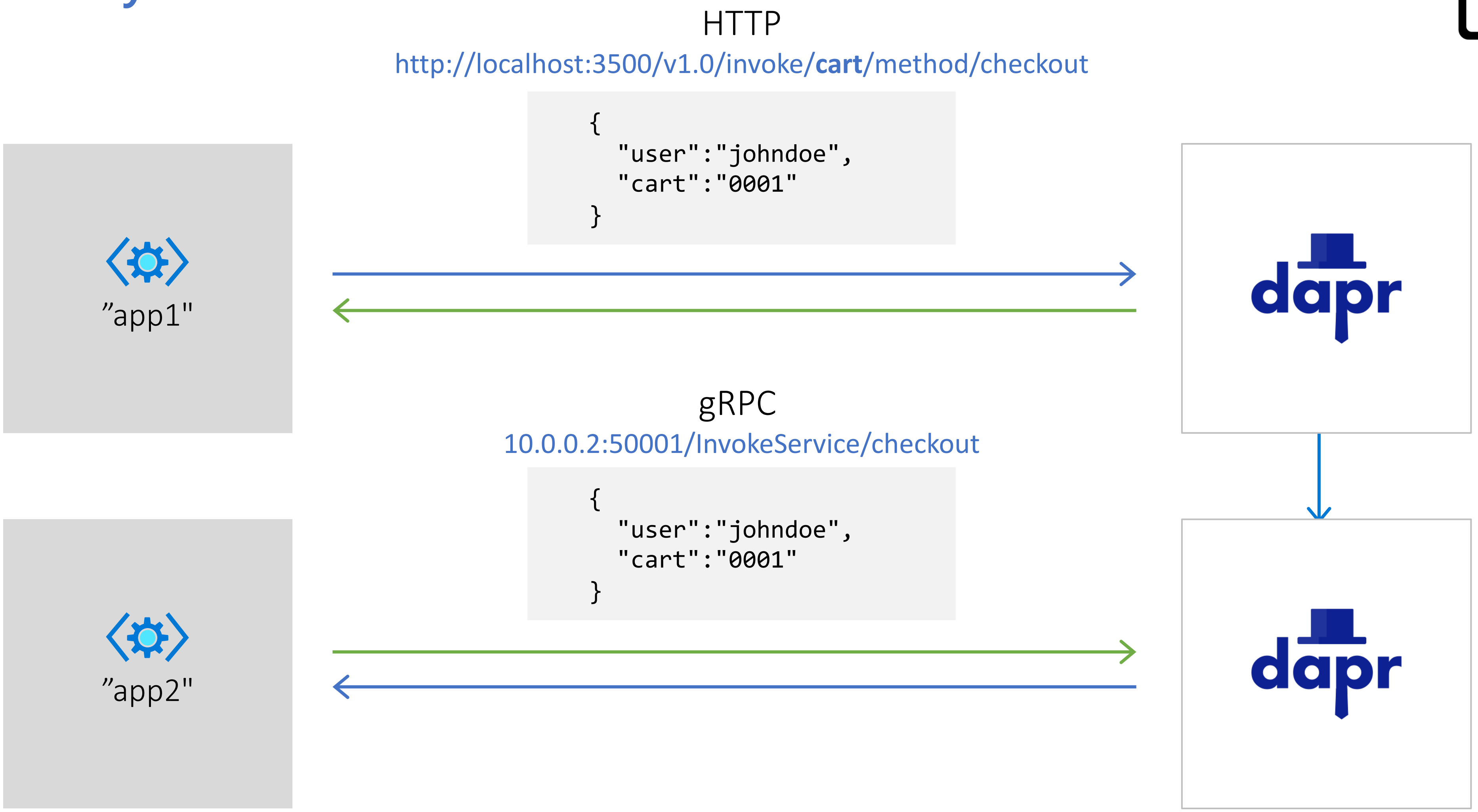
DELETE
/v1.0/state/**corpdb**/<key>

POST (Get bulk)
/v1.0/state/**corpdb**/bulk

POST / PUT (Save multiple)
/v1.0/state/**corpdb**/transaction

Dapr building blocks

Service discovery and invocation


HTTP
http://localhost:3500/v1.0/invoke/**cart**/method/checkout
”app1"
dapr
gRPC
10.0.0.2:50001/InvokeService/checkout
”app2"
dapr

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

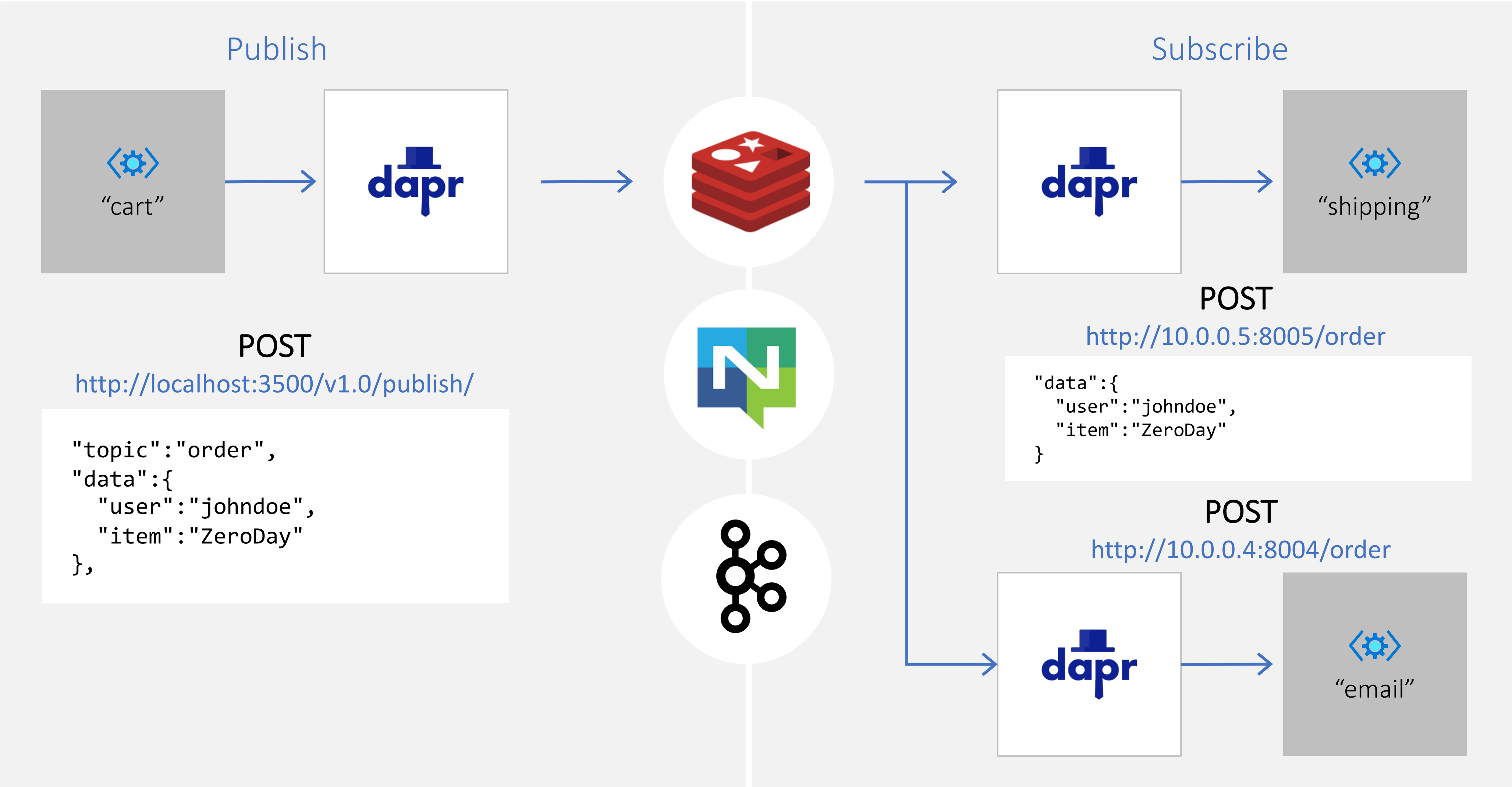
```
{
  "user":"johndoe",
  "cart":"0001"
}
```

```
curl -H "Content-Type: application/json” -d '{ "arg1": 10, "arg2": 23}' \
     http://localhost:3500/v1.0/invoke/myService/method/myMethod?param1=a&parm2=1
```

Dapr building blocks

Publish and subscribe

cloudevents



Publish

“cart”

dapr

POST

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

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

Subscribe

dapr

“shipping”

POST

http://10.0.0.5:8005/order

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

POST

http://10.0.0.4:8004/order

dapr

“email”

Dapr building blocks

Publish and subscribe

```
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: orders
spec:
  type: pubsub.redis
  metadata:
  - name: redisHost
    value: leader.redis.svc.cluster.local:6379
  - name: redisPassword
    secretKeyRef:
      name: redis-secret
      key: password
  - name: allowedTopics
    value: ”processed,audit"
```

POST
/v1.0/publish/**orders/processed**

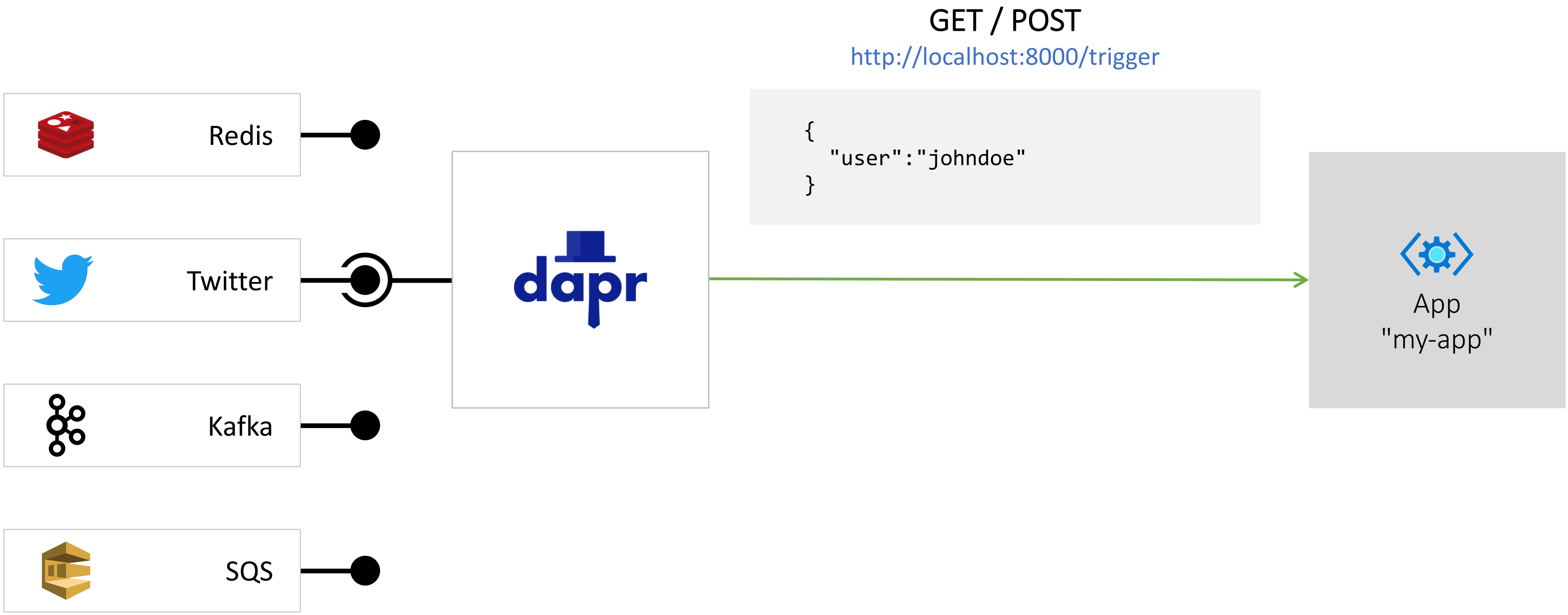
GET (programmatic discover)
/dapr/subscribe

Supports both programmatic and declarative subscriptions

POST (to the app)
/any/path/defined/in/subscription

Dapr building blocks

Resource bindings: input



Redis

Twitter

Kafka

SQS

dapr

GET / POST

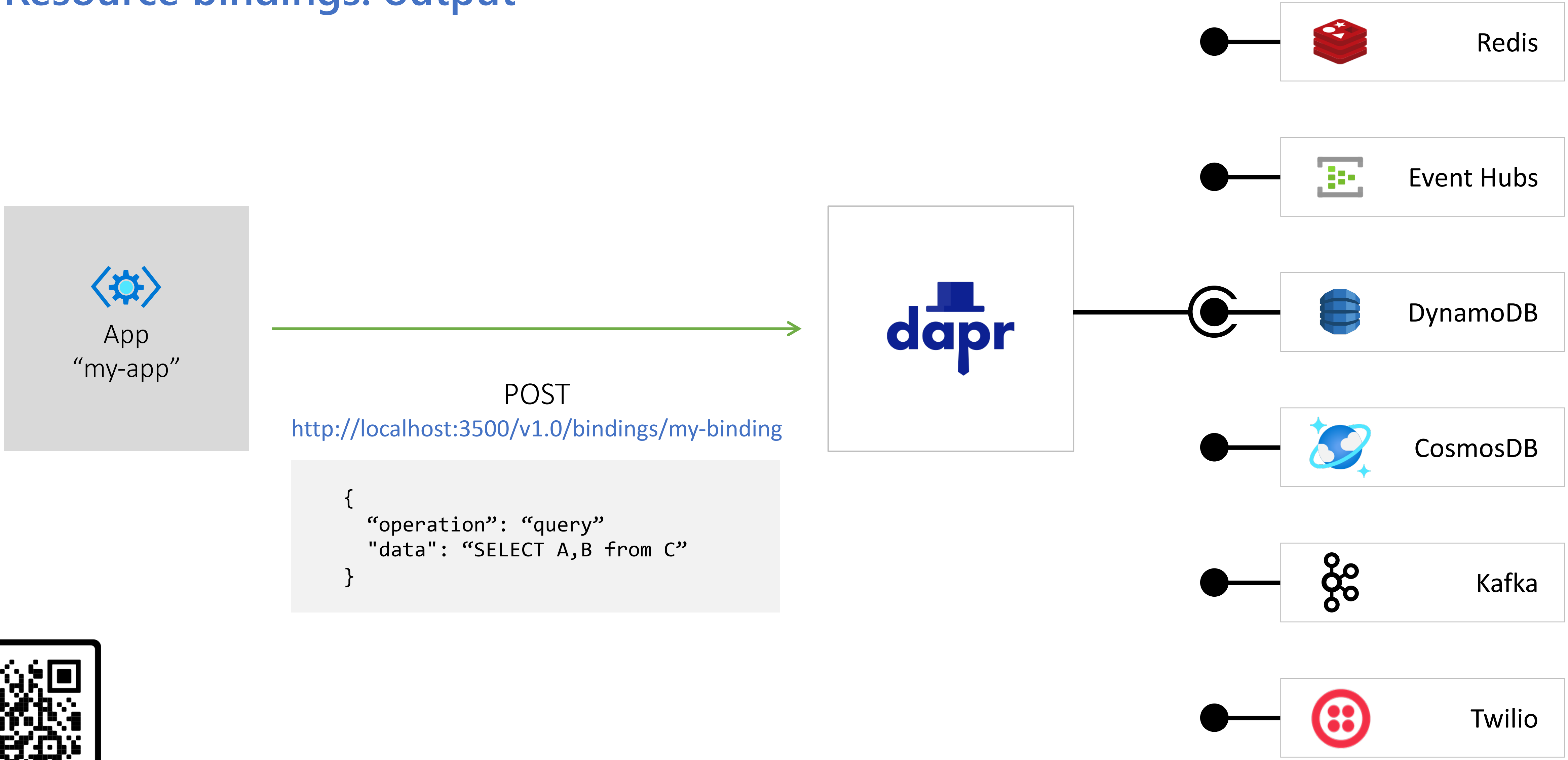
http://localhost:8000/trigger

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

App
"my-app"

Dapr building blocks

Resource bindings: output



POST
http://localhost:3500/v1.0/bindings/my-binding

```
{
  “operation”: “query”
  "data": “SELECT A,B from C”
}
```

Dapr building blocks

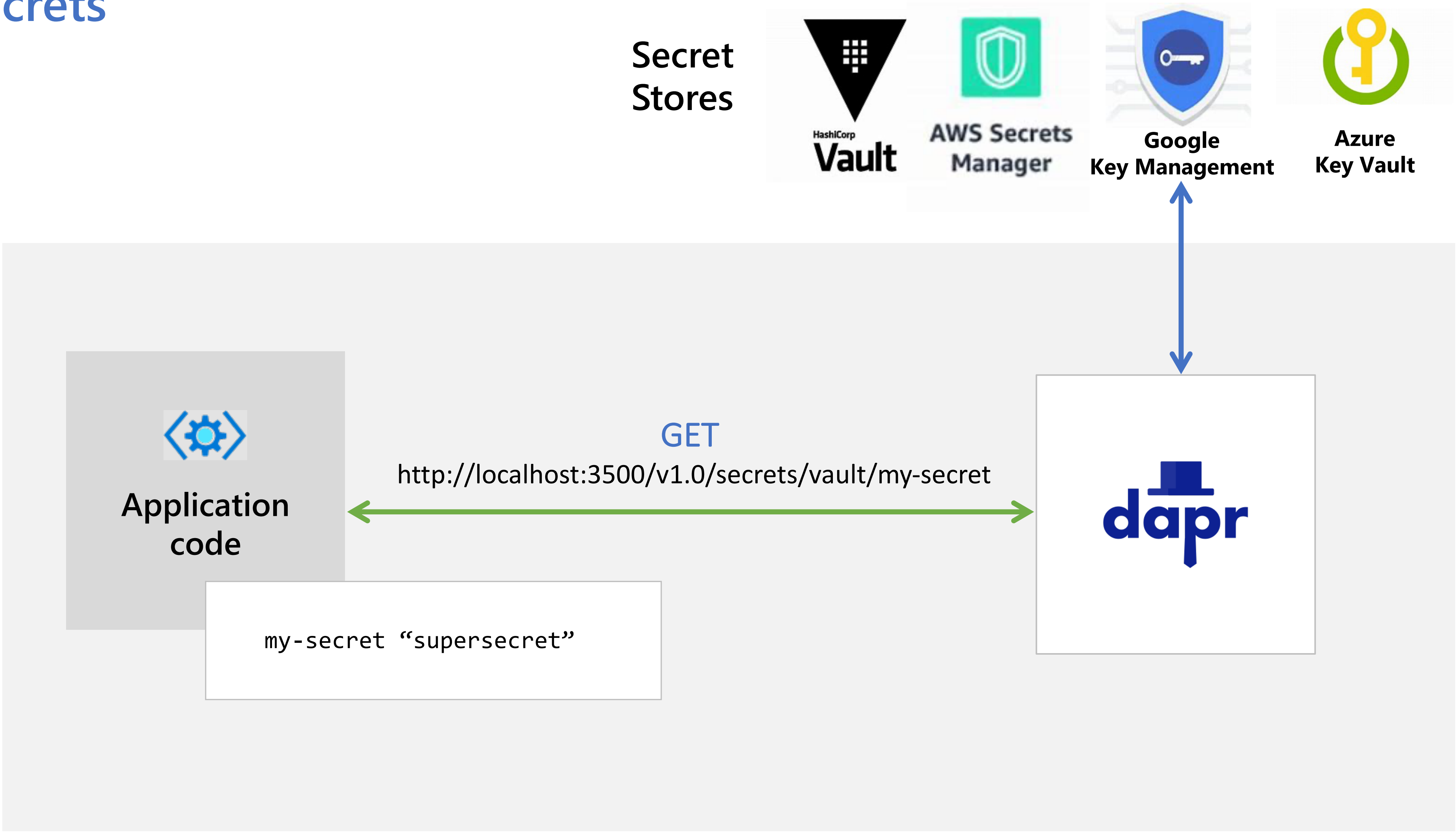
Resource bindings

```
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: my-kafka
spec:
  type: bindings.kafka
  metadata:
  - name: brokers
    value: "http://localhost:5050"
  - name: topics
    value: ”submitted"
  - name: publishTopic
    value: ”processed"
  - name: consumerGroup
    value: "group1"
```

OPTIONS/POST (Input)
/my-kafka

POST/PUT (Output)
/v1.0/bindings/**my-kafka**

Dapr building blocks
Secrets



```
my-secret “supersecret”
```

Dapr building blocks

Secrets

```
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: my-secrets
spec:
  type: secretstores.hashicorp.vault
  metadata:
  - name: vaultAddr
    value: https://127.0.0.1:8200
  - name: caCert
    value: "ca_cert"
  - name: caPath
    value: ”/certs/cert.pem"
  - name: caPem
    value: "/certs/ca.pem"
…
```

GET

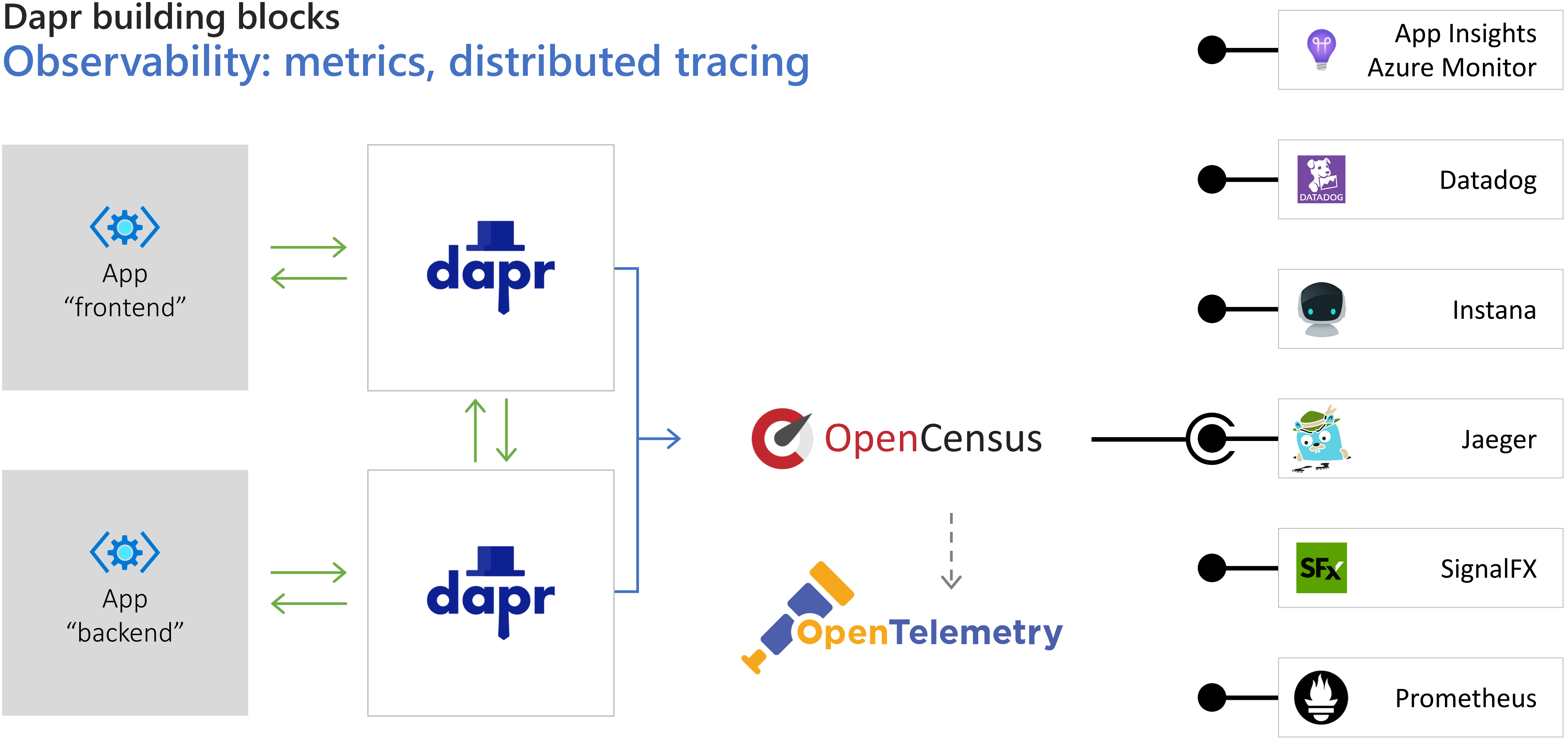
/v1.0/secrets/**my-secrets/redis-password**

Some secret stores support parameters in query (for metadata (e.g. version)

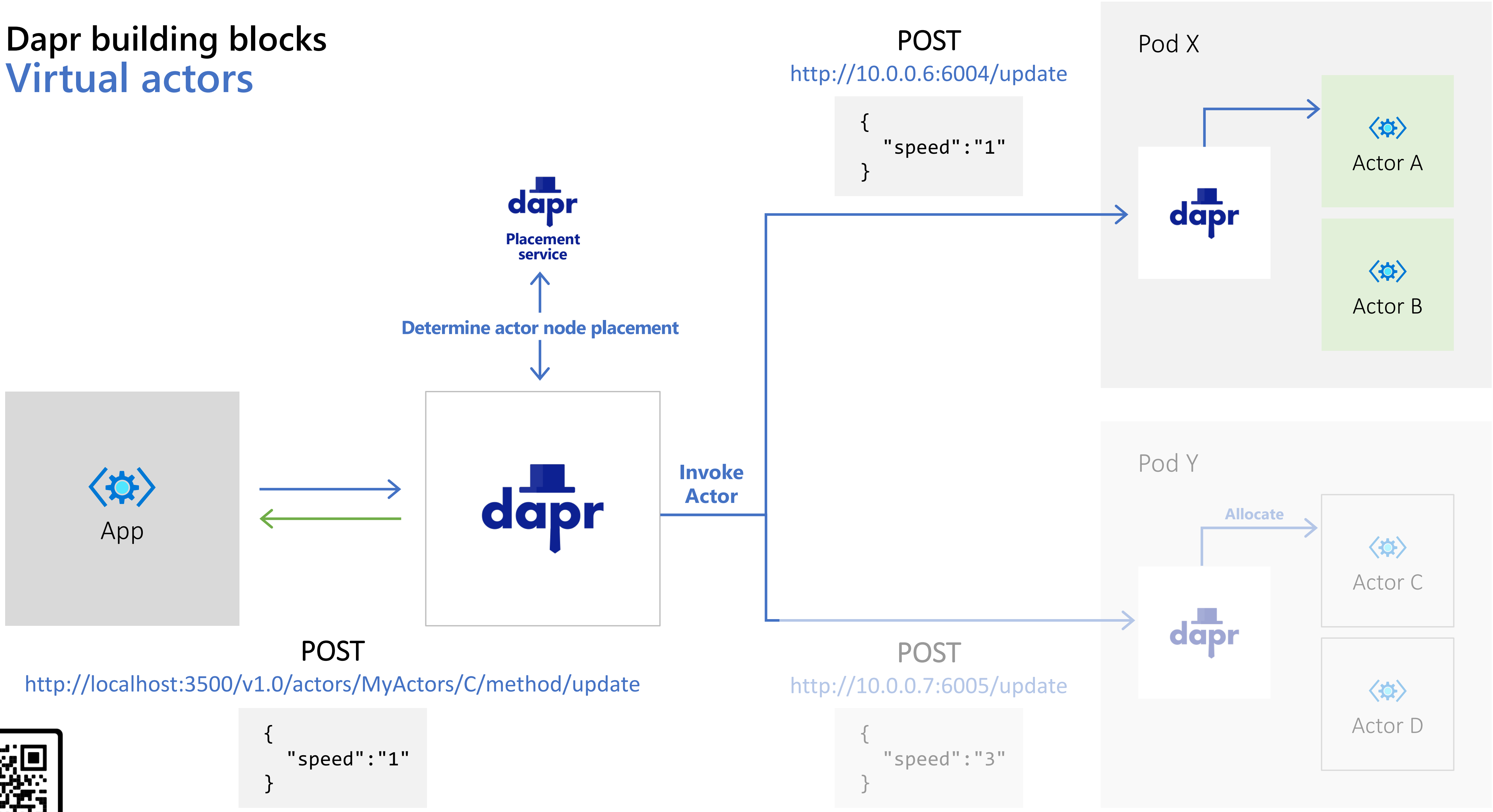
And in config

```
- name: redisPassword
  secretKeyRef:
    name: my-secrets
    key: redis-password
```

Dapr building blocks
Observability: metrics, distributed tracing



Dapr building blocks
Virtual actors



Placement service

Determine actor node placement

App

Invoke Actor

POST

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

```
{
   "speed":"1"
}
```

POST

http://10.0.0.6:6004/update

```
{
   "speed":"1"
}
```

Pod X

Actor A

Actor B

Pod Y

Allocate

Actor C

Actor D

POST

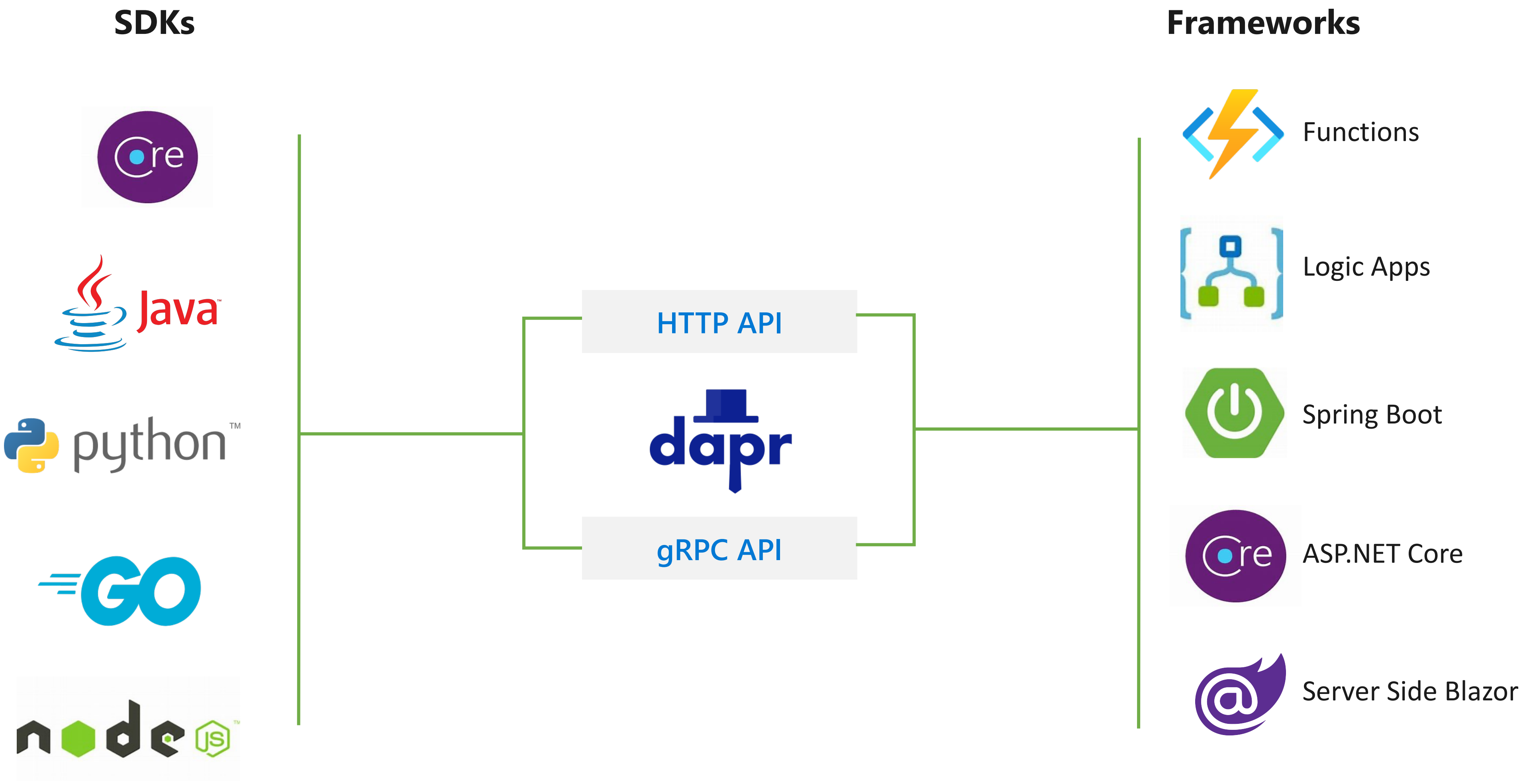
http://10.0.0.7:6005/update

```
{
   "speed":"3"
}
```

SDKs & Frameworks

Dapr SDKs & Frameworks

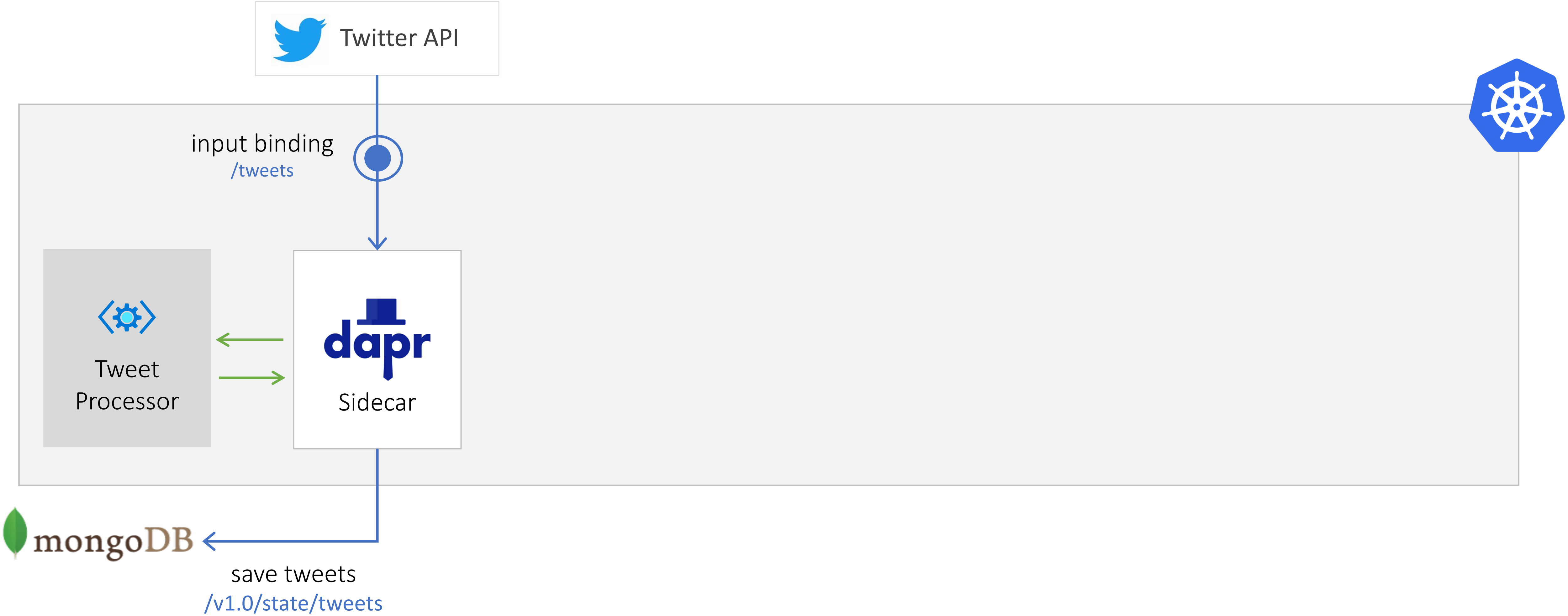
Language-specific Convenience Layers



Demo

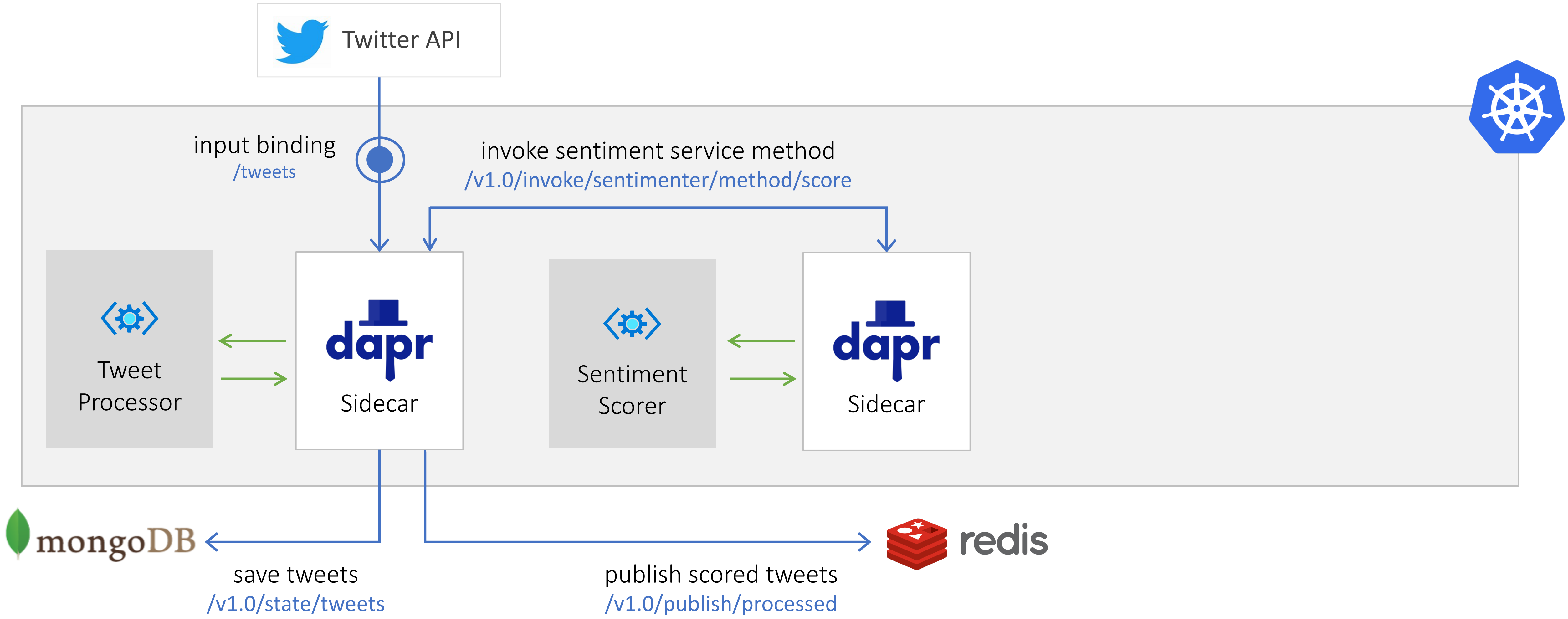
Demo

Event processing pipeline



Demo

Event processing pipeline



Twitter API

input binding
/tweets

invoke sentiment service method
/v1.0/invoke/sentimenter/method/score

Tweet Processor

Sidecar

Sentiment Scorer

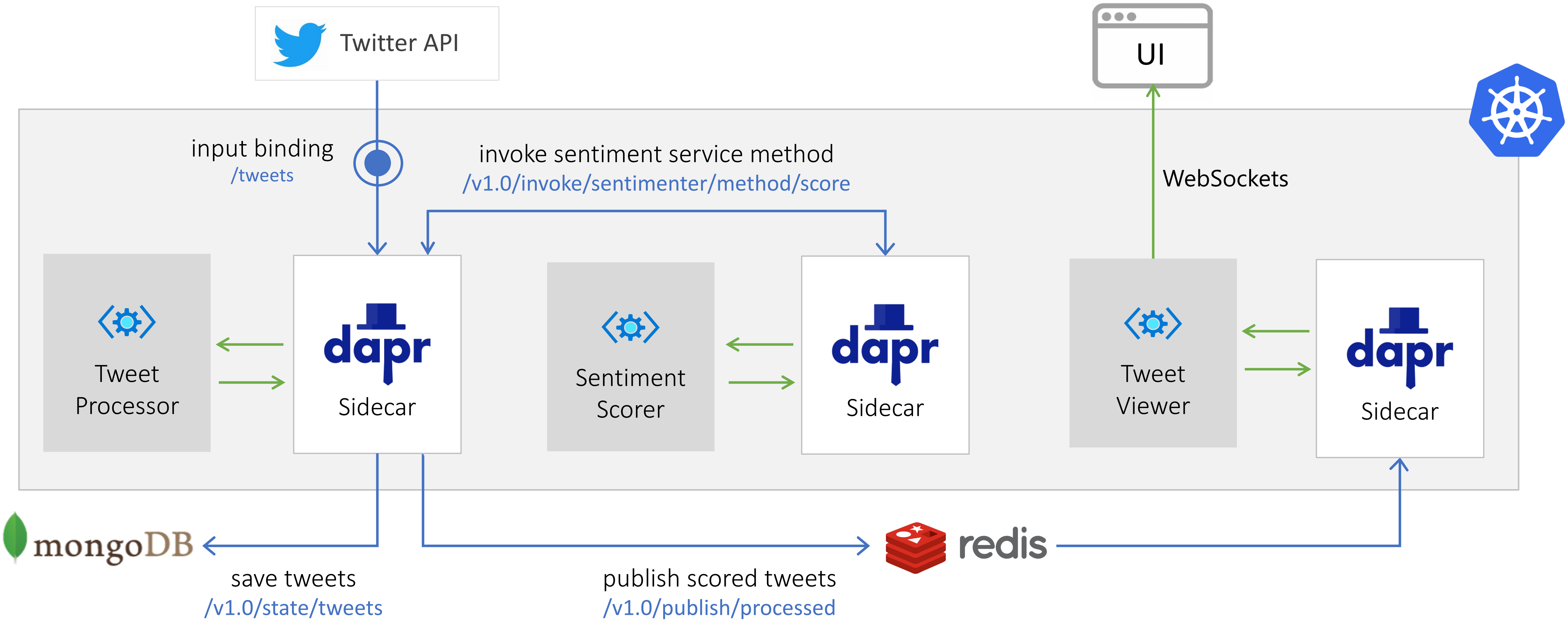
Sidecar

save tweets
/v1.0/state/tweets

publish scored tweets
/v1.0/publish/processed

Demo

Event processing pipeline



Twitter API

UI

input binding
/tweets

invoke sentiment service method
/v1.0/invoke/sentimenter/method/score

WebSockets

Tweet Processor

Sidecar

Sentiment Scorer

Sidecar

Tweet Viewer

Sidecar

save tweets
/v1.0/state/tweets

publish scored tweets
/v1.0/publish/processed

Event Processing Pipeline

Demo

bit.ly/cncf-dapr

Dapr Integrations

Dapr Areas of Focus

- v0.11
  - Stable APIs
  - Access control, service identity
  - External security audit (CNCF-certified company)
  - Kicked off project transition to open governance
- Next, most likely, v1.0 RC by EOY
  - Address friction from real-world use-cases
  - Test infra (performance, conformance, long running tests…)
  - Start seating the Technical and Steering Committee (TSC)
  - Ensure general readiness for production grade workloads

Dapr

dapr.io

Project

github.com/dapr

Chat

gitter.im/dapr

Twitter

@daprdev

Videos

bit.ly/dapr-videos

Anything Else

mchmarny@microsoft.com