Dapr

Kubernetes Deep Dive

What’s in this module?

What is Dapr?

Installing Dapr

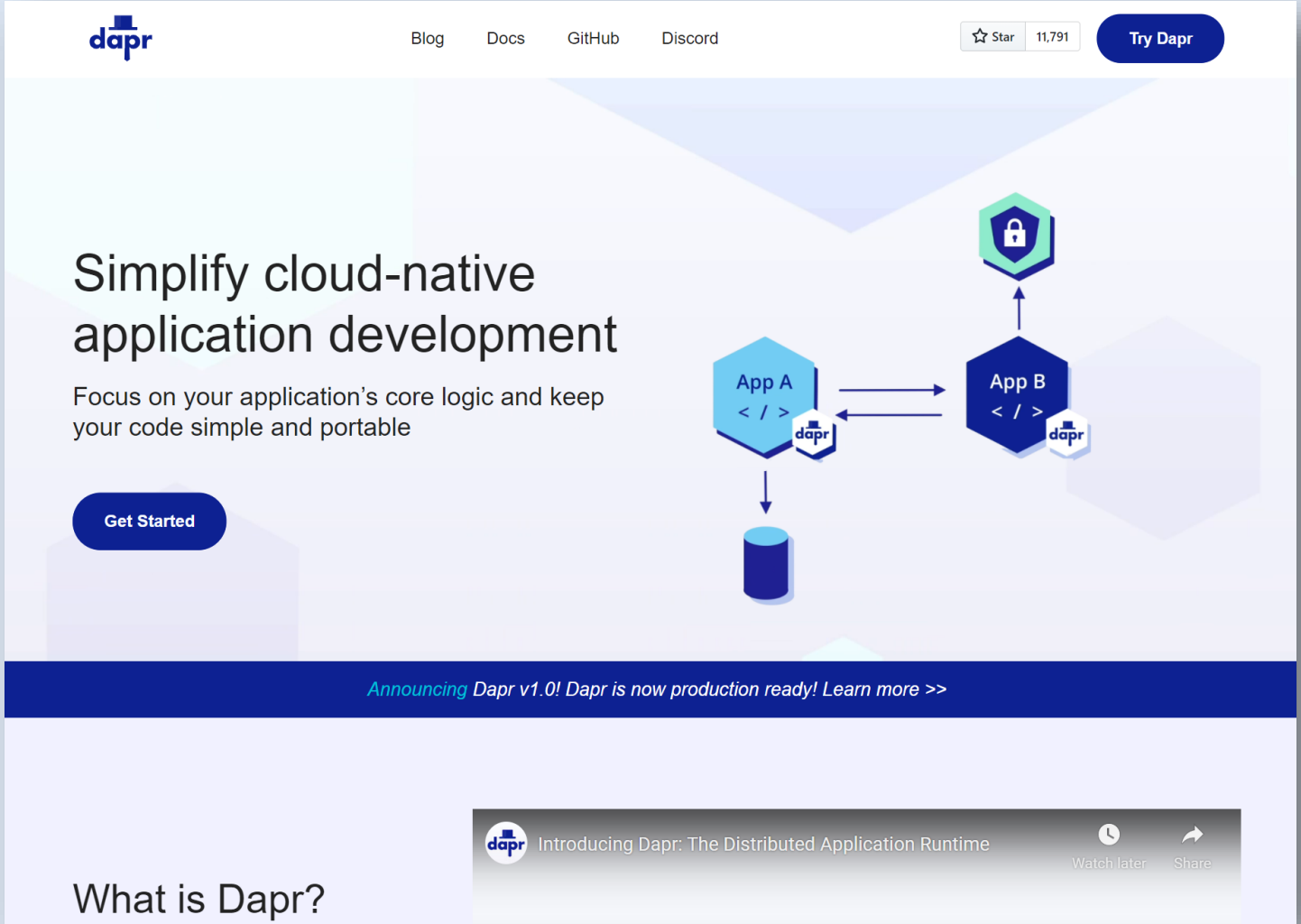
Using Dapr

What is Dapr?

Distributed Application Runtime

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

dapr.io

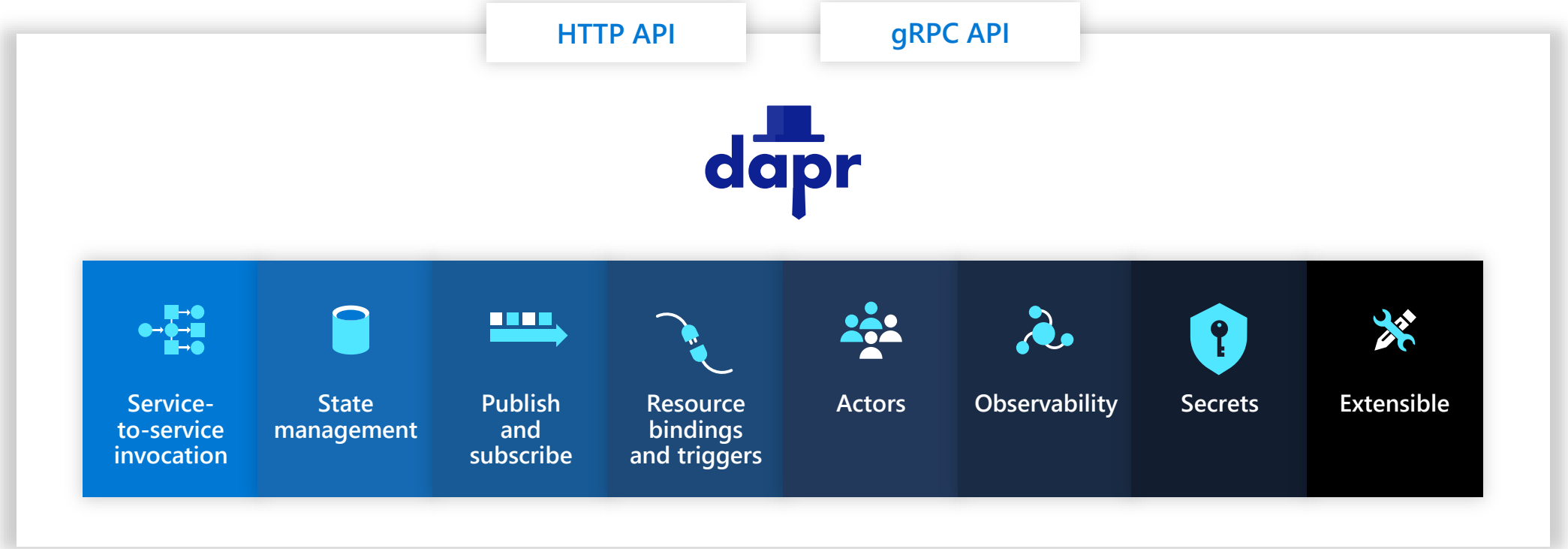


Dapr Goals

- Best-practices building blocks
- Any language or framework
- Consistent, portable, open APIs
- Adopt standards
- Extensible and pluggable components
- Platform agnostic cloud + edge

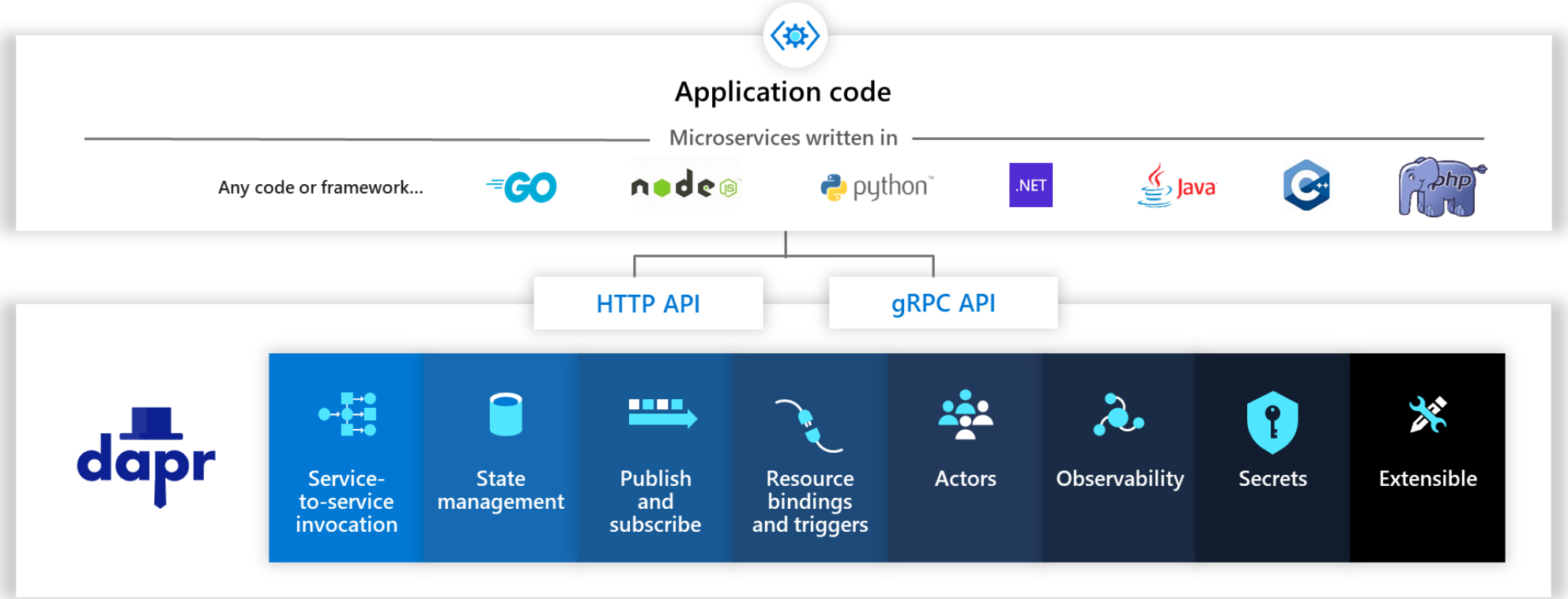
- Community driven, vendor neutral

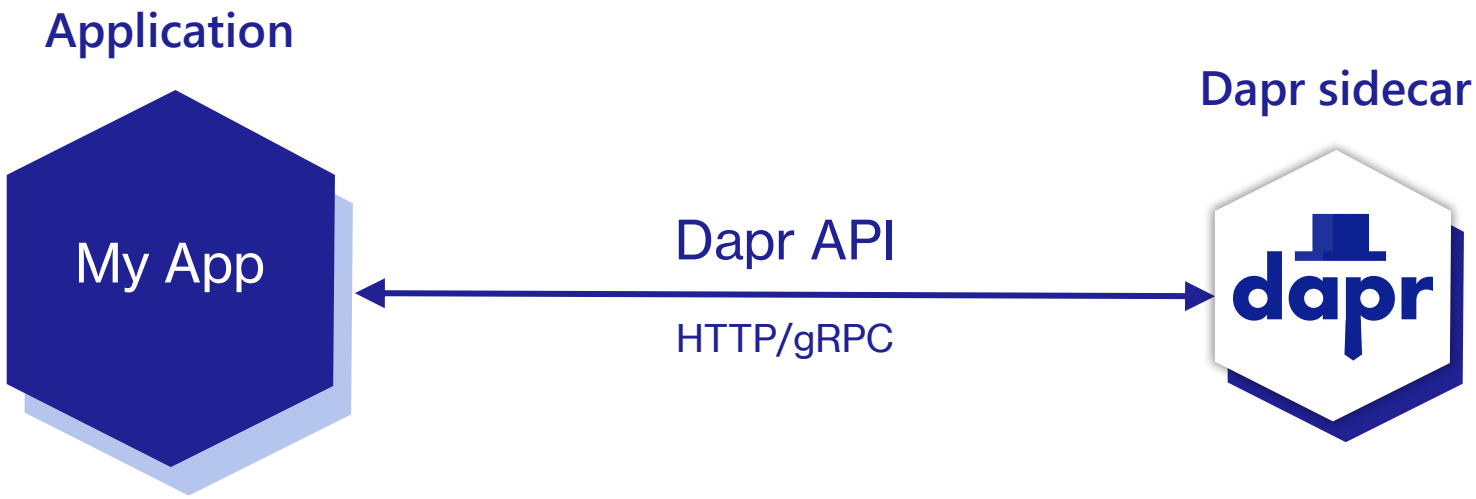
Building blocks



Any language or framework

- Standard APIs accessed over http/gRPC protocols from user service code
- Runs as local "side car library" dynamically loaded at runtime for each service



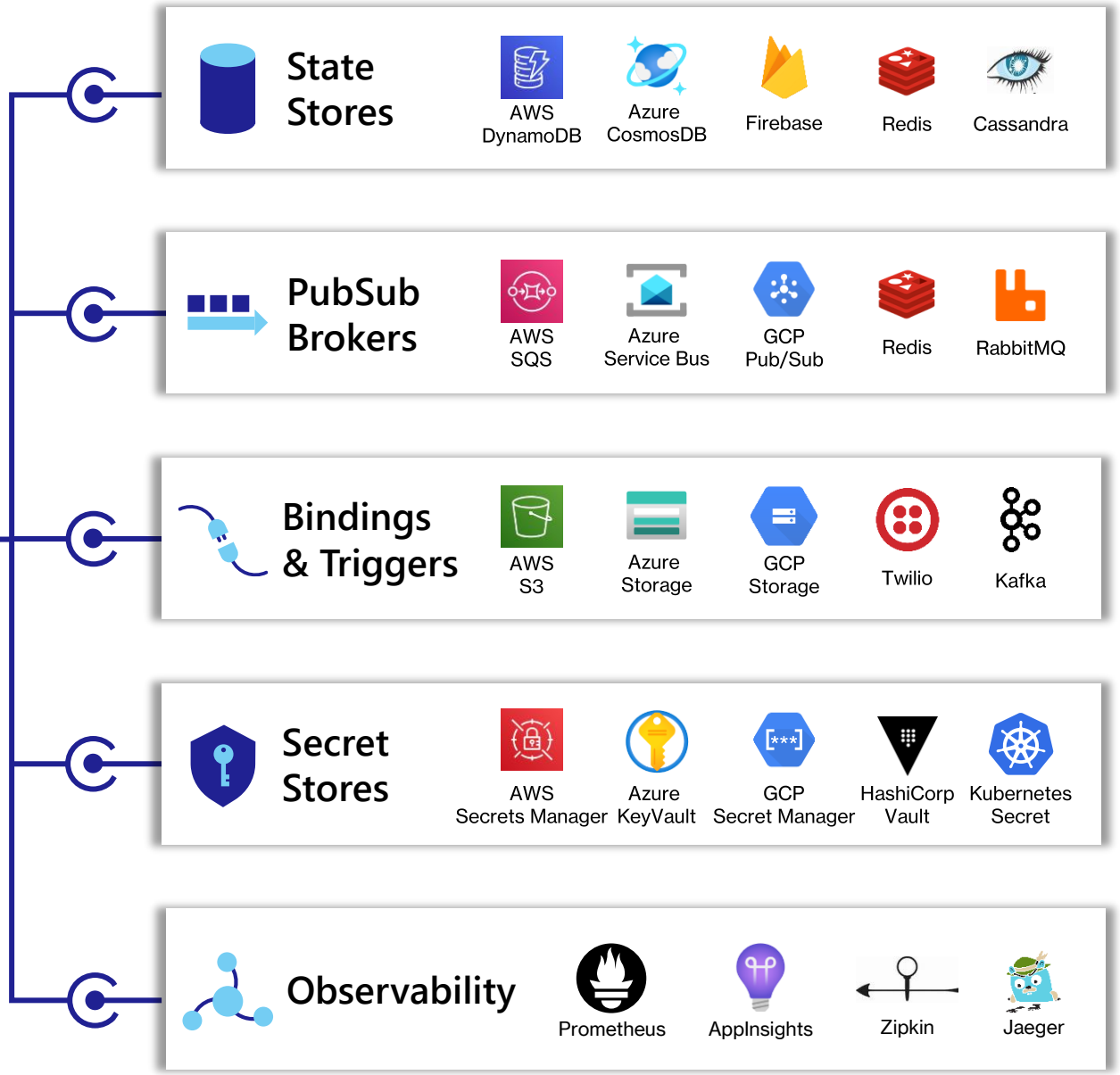



```
POST http://localhost:3500/v1.0/invoke/cart/method/neworder

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
```

Swappable YAML files with resource connection details

Over 70 components available

Create components for your resource at: github.com/dapr/components-contrib



Installing Dapr

Dapr hosting environments

Self-hosted

- Get started with `dapr init`
- Easy setup with Docker images
  - Sets up placement, Zipkin, Redis
  - `slim-init` available without Docker
- Run any application with Dapr sidecar using `dapr run`

kubernetes

- Get started with `dapr init -k`
- Fully managed Dapr control plane
  - Deploys dashboard, placement, operator, sentry, and injector pods
- Automatically inject Dapr sidecar into all annotated pods
- Upgrade with `dapr upgrade` or Helm

Install the Dapr CLI

- Runs on Linux, Windows and MacOS
- Dapr releases: https://github.com/dapr/cli/releases

```
wget -q https://raw.githubusercontent.com/dapr/cli/master/install/install.sh -O - | /bin/bash
```

(installs to /usr/local/bin)

```
powershell -Command
  "iwr -useb https://raw.githubusercontent.com/dapr/cli/master/install/install.ps1 | iex"
```

(installs to C:\dapr)

```
      __
  ____/ /__ ____  ____
 / __  / _ '/ _ \/ ___/
/ /_/ / /_/ / /_/ / /
\__,_/\__,_/ .___/_/
           /_/

=============================
Distributed Application Runtime

Usage:
  dapr [command]

Available Commands:
  build-info     Print build info of Dapr CLI and runtime
  completion     Generates shell completion scripts
  components     List all Dapr components. Supported platforms: Kubernetes
  configurations List all Dapr configurations. Supported platforms: Kubernetes
  dashboard      Start Dapr dashboard. Supported platforms: Kubernetes and self-hosted
  help           Help about any command
  init           Install Dapr on supported hosting platforms. Supported platforms: Kubernetes and self-hosted
  invoke         Invoke a method on a given Dapr application. Supported platforms: Self-hosted
  list           List all Dapr instances. Supported platforms: Kubernetes and self-hosted
  logs           Get Dapr sidecar logs for an application. Supported platforms: Kubernetes
  mtls           Check if mTLS is enabled. Supported platforms: Kubernetes
  publish        Publish a pub-sub event. Supported platforms: Self-hosted
  run            Run Dapr and (optionally) your application side by side. Supported platforms: Self-hosted
  status         Show the health status of Dapr services. Supported platforms: Kubernetes
  stop           Stop Dapr instances and their associated apps. Supported platforms: Self-hosted
  uninstall      Uninstall Dapr runtime. Supported platforms: Kubernetes and self-hosted
  upgrade        Upgrades or downgrades a Dapr control plane installation in a cluster. Supported platforms: Kubernete
```

Initialize Dapr

- Run **dapr init**
  - Fetches sidecar binaries
  - Runs a few containers: redis, zipkin, placement
  - Creates a default components folder
- Verify Dapr version

  dapr --version
- Verify containers

  docker container ls

```
gbaeke  ~  .dapr  $  tree -L 2
.
├── bin
│   ├── daprd
│   ├── dashboard
│   └── web
├── components
│   ├── pubsub.yaml
│   └── statestore.yaml
└── config.yaml

3 directories, 5 files
```

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: pubsub
spec:
  type: pubsub.redis
  version: v1
  metadata:
  - name: redisHost
    value: localhost:6379
  - name: redisPassword
    value: ""
```

Dashboard

Configuration: daprConfig

Summary | Configuration

Summary

Name	daprConfig
Kind	Configuration
Created	2021-10-20 17:23.22
Age	20h
Tracing Enabled	true
Sampling Rate	1
Metrics Enabled	true
MTLS Enabled	false

Dashboard

Dapr Components

	Name	Type	Age	Created
	pubsub	pubsub.redis	20h	2021-10-20 17:23.22
	statestore	state.redis	20h	2021-10-20 17:23.22

Dashboard

Scope: All

Overview

Dapr Applications

Name	Age	Actions
myapp	6m	Stop

Follow along

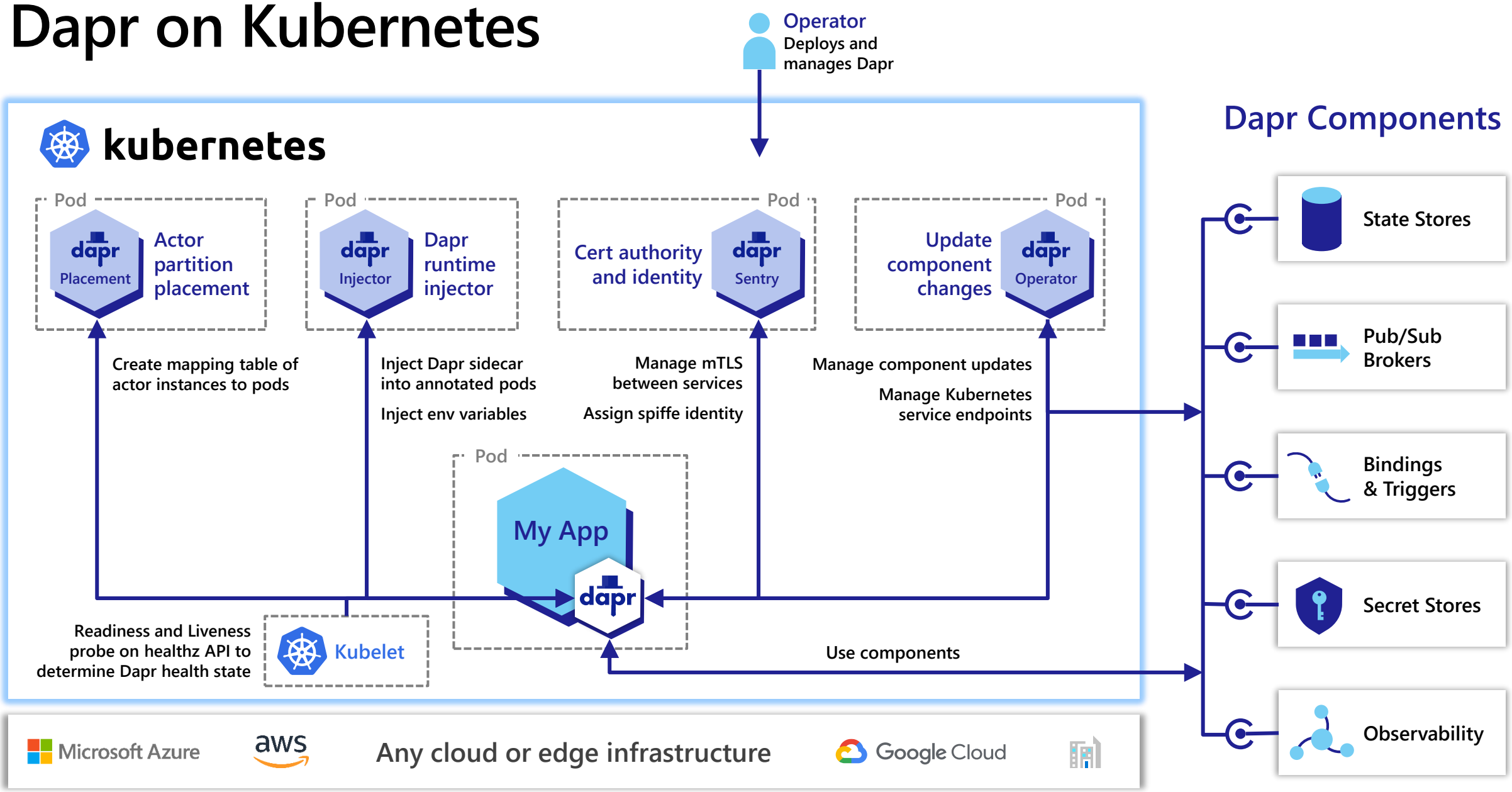
- Installing dapr
- Initialize dapr
- Running the sidecar
- Saving state in Redis

Install on Kubernetes

- On a development cluster: **dapr init --kubernetes --wait**
- In production, use Helm:

```
helm upgrade --install dapr dapr/dapr \
--version=1.4 \
--namespace dapr-system \
--create-namespace \
--set global.ha.enabled=true \
--wait
```

Dapr on Kubernetes



No default components on K8S

- You need to decide which components to use and create them with YAML
- For example: Redis
  - Install Redis in Kubernetes – or – use Azure Redis Cache
  - Create a secret to hold the Redis password
  - Create a state store component for Redis

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: statestore
  namespace: default
spec:
  type: state.redis
  version: v1
  metadata:
  - name: redisHost
    value: redis-master.default.svc.cluster.local:6379
  - name: redisPassword
    secretKeyRef:
      name: redis
      key: redis-password
```

Follow along

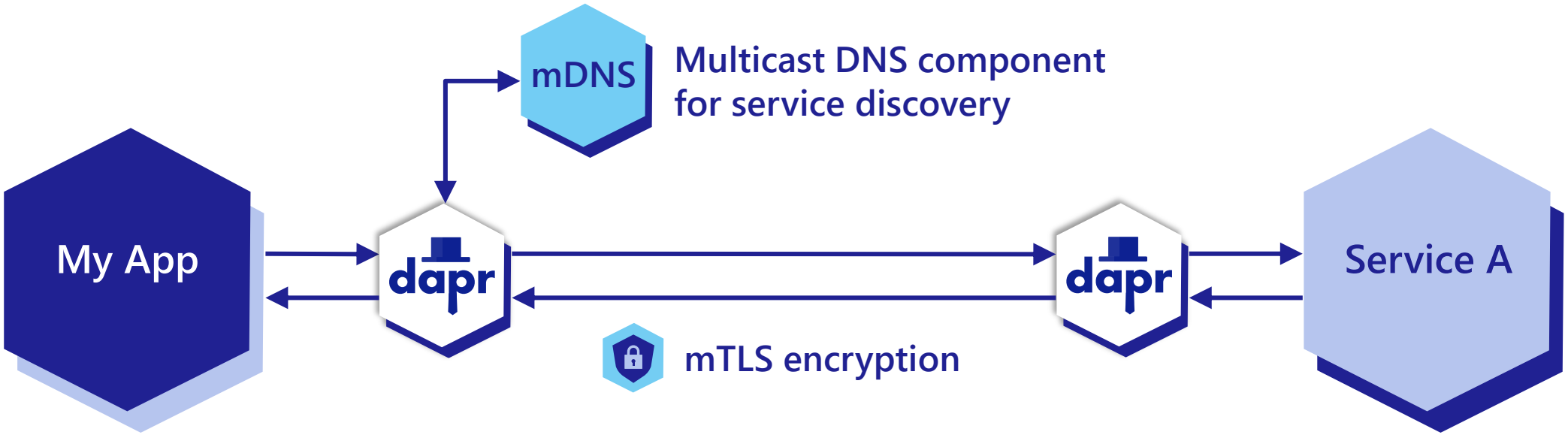
- Installing Dapr on K8S
- Check what was deployed
- View the dashboard

Using Dapr

Coming up...

- Service invocation
- State Management
- Pub/sub
- Bindings

Service invocation



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

```
{"data":"Hello World"}
```

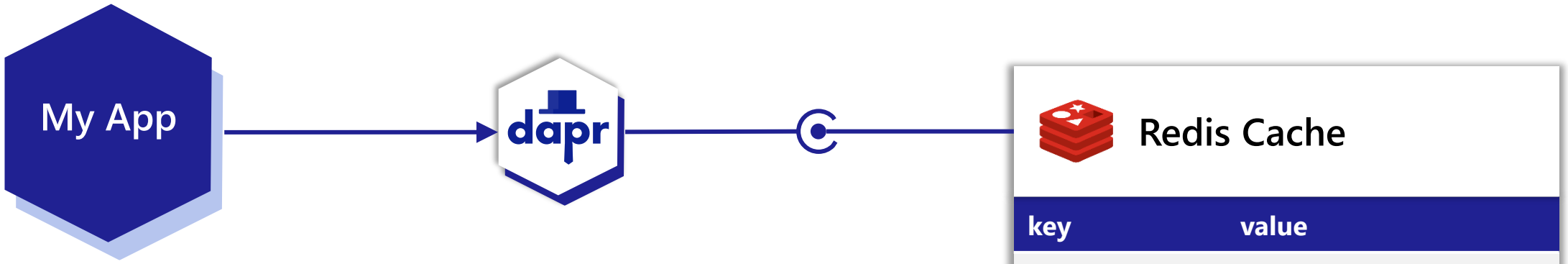
POST
`http://10.0.0.2:8000/neworder`

```
{"data":"Hello World"}
```

```csharp
public async Task<VehicleInfo> GetVehicleInfo(string licenseNumber)
    {
        return await _httpClient.GetFromJsonAsync<VehicleInfo>(
            $"http://localhost:6002/vehicleinfo/{licenseNumber}");
    }
```

```csharp
public async Task<VehicleInfo> GetVehicleInfo(string licenseNumber)
{
    return await _httpClient.GetFromJsonAsync<VehicleInfo>(
        $"http://localhost:3601/v1.0/invoke/vehicleregistrationservice/method/vehicleinfo/{licenseNumber}");
}
```

State management

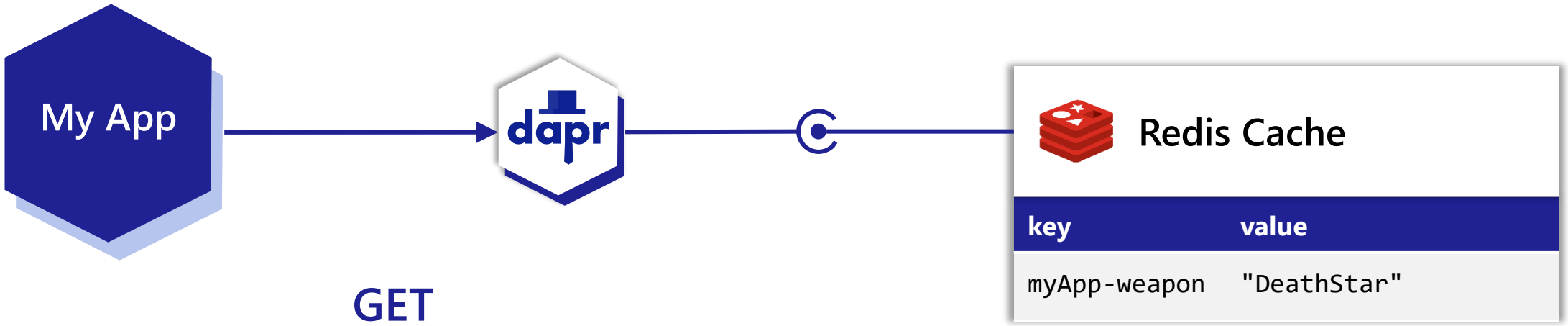


POST

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

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

State management

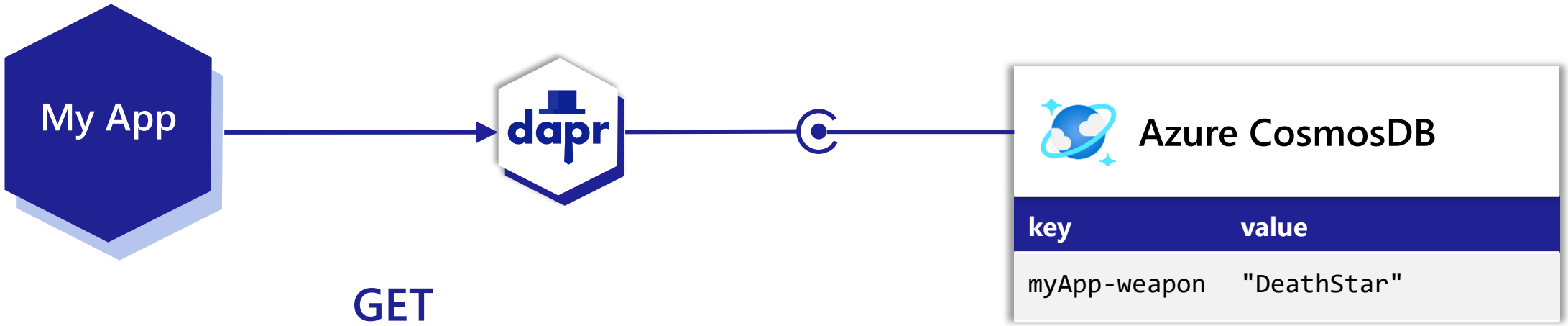


GET

`http://localhost:3500/v1.0/state/corpdb/weapon`

```
"DeathStar"
```

State management



GET

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

```
"DeathStar"
```

Dapr state API

Save state

```
POST /v1.0/state/corpdb
```

Retrieve state

```
GET /v1.0/state/corpdb/mystate
```

Delete state

```
DELETE /v1.0/state/corpdb/mystate
```

Get bulk state

```
POST /v1.0/state/corpdb/bulk
```

Submit multiple state transactions

```
POST /v1.0/state/corpdb/transaction
```

corpdb-redis.yaml

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: corpdb
spec:
  type: state.redis
  version: v1
  metadata:
  - name: redisHost
    value: redis-master.default.svc.cluster.local:6379
  - name: redisPassword
    secretKeyRef:
      name: redis-secret
      key: redis-password
```

Dapr state API

Save state

```
POST /v1.0/state/corpdb
```

Retrieve state

```
GET /v1.0/state/corpdb/mystate
```

Delete state

```
DELETE /v1.0/state/corpdb/mystate
```

Get bulk state

```
POST /v1.0/state/corpdb/bulk
```

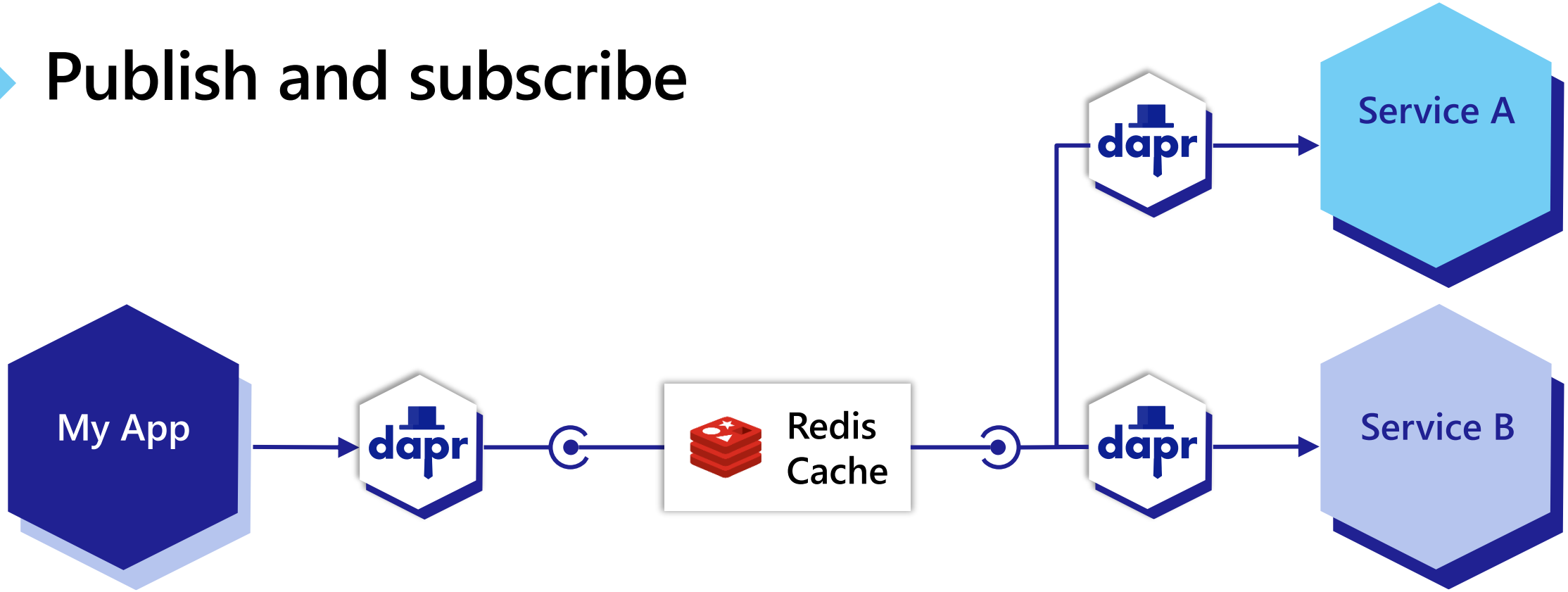
Submit multiple state transactions

```
POST /v1.0/state/corpdb/transaction
```

corpdb-cosmosdb.yaml

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: corpdb
spec:
  type: state.azure.cosmosdb
  version: v1
  metadata:
  - name: url
    value: corpdb.documents.azure.com
  - name: masterKey
    secretKeyRef:
      name: master-key
      key: cosmos-key
  - name: database
    value: orders
  - name: collection
    value: processed
```

Publish and subscribe



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

POST
http://10.0.0.2:8000/orders
http://10.0.0.4:8000/factory/orders

```
{"data":"Hello World"}
```

```
{"data":"Hello World"}
```

Dapr pub/sub API

App-to-sidecar

Publish a message

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

Sidecar-to-app

Get app subscriptions

```
GET /dapr/subscribe
```

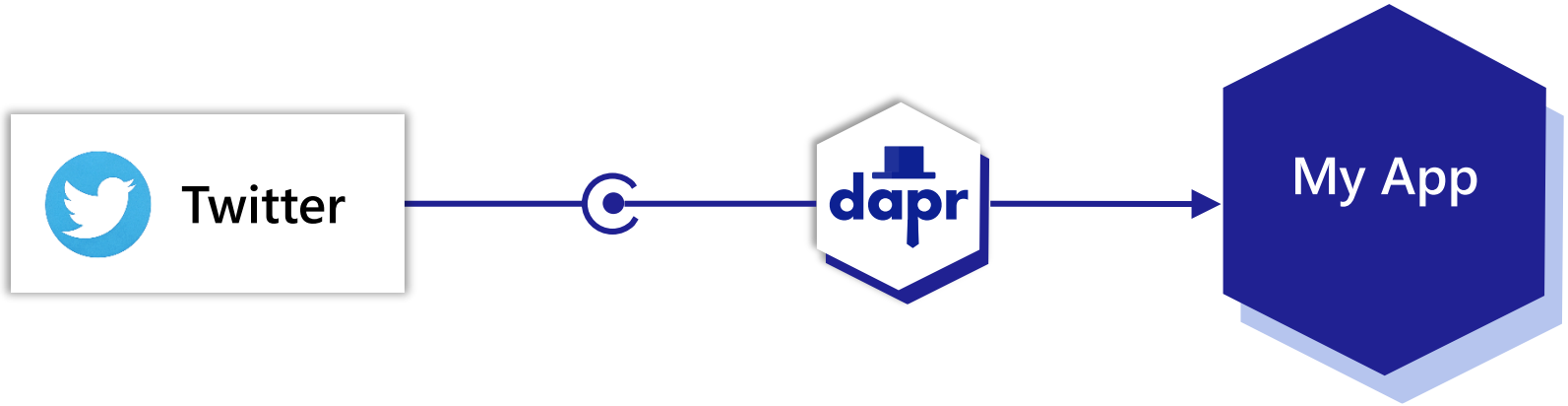
Publish to app

```
POST /order-processing
```

orders.yaml

```yaml
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"
```

Input triggers

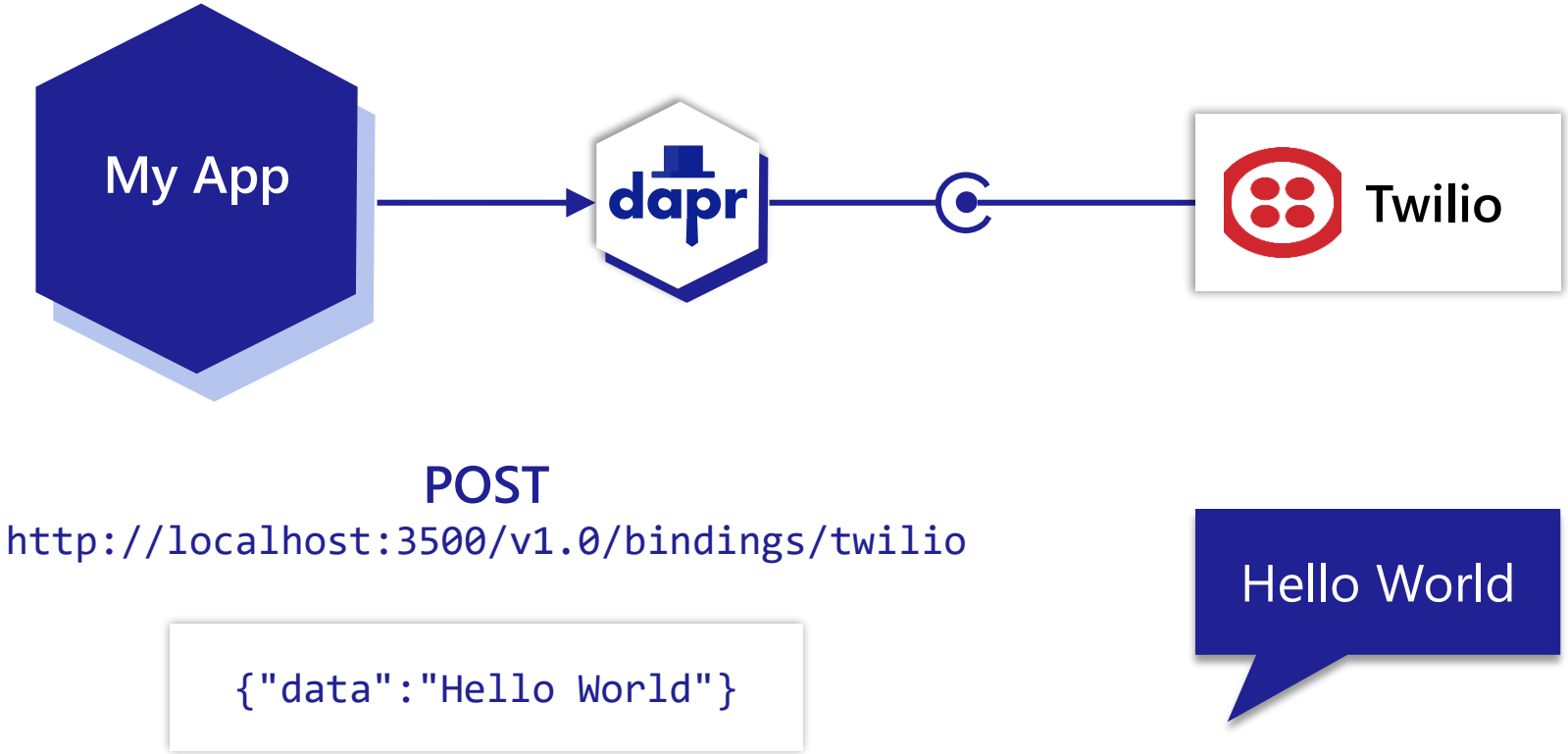


POST

`http://10.0.0.2:8000/newtweet`

```
{"data":"📣 We are excited
to announce the …"}
```

Output bindings



Dapr bindings API

App-to-sidecar

Invoke an output binding

`POST/PUT /v1.0/bindings/twitter`

Sidecar-to-app

Trigger an app

`OPTIONS/POST /new-tweet`

twitter.yaml

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: twitter
spec:
  type: bindings.twitter
  version: v1
  metadata:
  - name: consumerKey
    secretKeyRef:
      name: twitter-secret
      key: consumerKeys
  - name: consumerSecret
    secretKeyRef:
      name: twitter-secret
      key: consumerSecret
  - name: accessToken
    secretKeyRef:
      name: twitter-secret
      key: accessToken
  - name: accessSecret
    secretKeyRef:
      name: twitter-secret
      key: accessSecret
```

Let's try it!

- Pub sub with Node.js
- Using an output binding (Azure storage)
- Secrets with Key Vault and AAD Pod Identity (optional)