



# Dapr

## Distributed Application Runtime

Serverless ve Cloud Native - D

Gökçenur ÇINAR

Hüseyin Çambaşı

İhsan Baran SÖNMEZ

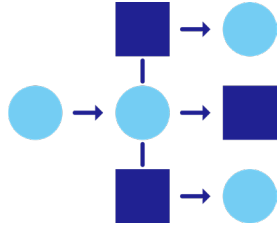
Nazlı Ece Çavlan

# dapr

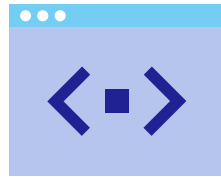
# State of enterprise developers



Must deploy scale-out apps for flexibility, cost, and efficiency



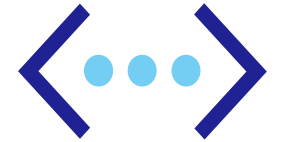
Must develop resilient, scalable, microservice-based apps that interact with services



Want to focus on building applications, not learning infrastructure



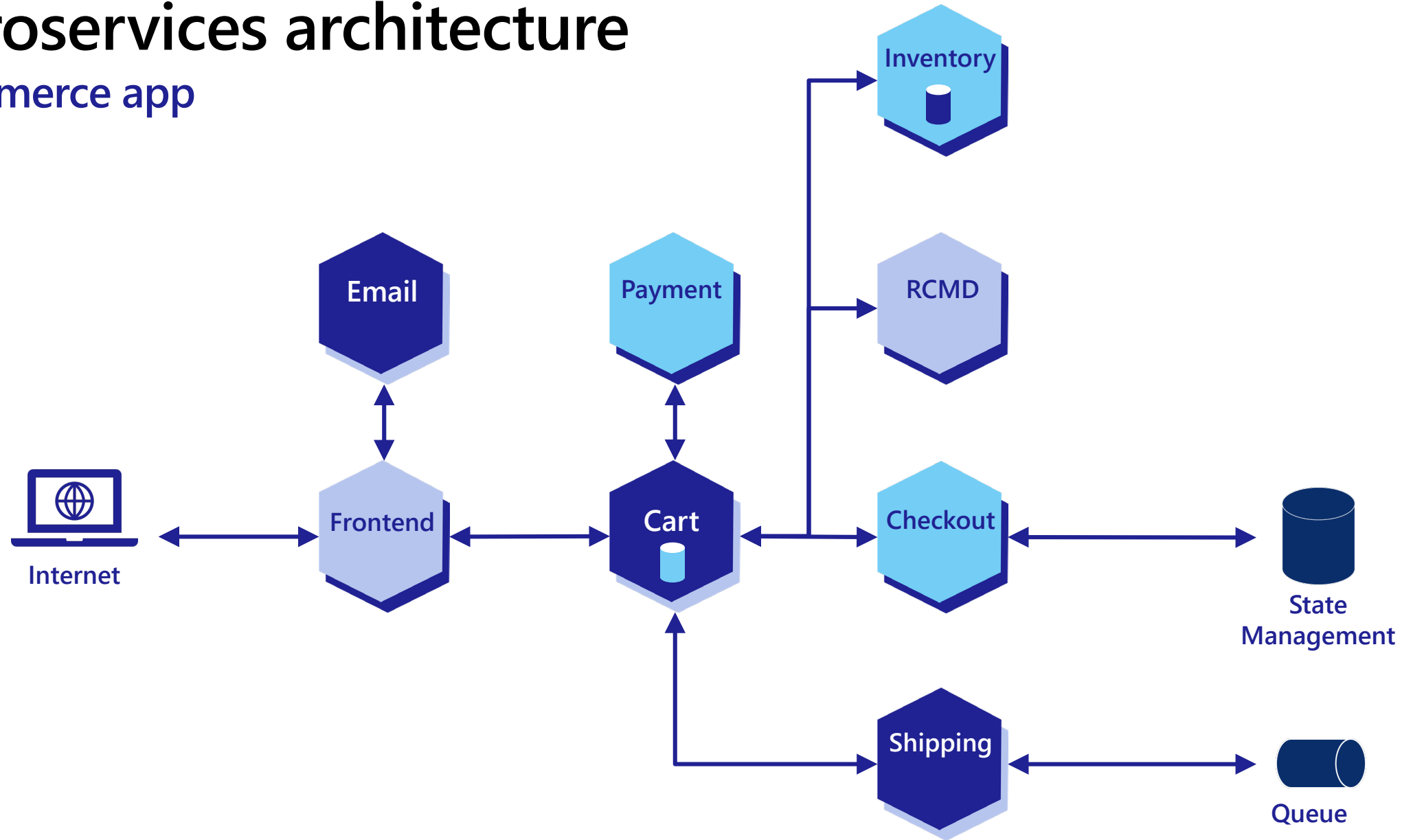
Trending toward serverless platforms with simple code to cloud pipelines



Use multiple languages and frameworks during development

# Microservices architecture

E-commerce app





# Distributed Application Runtime

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

[dapr.io](https://dapr.io)

The screenshot shows the Dapr website homepage. At the top, there is a navigation bar with the Dapr logo, links for Home, Testimonials, Docs, Blog, GitHub, and Discord, a star count of 17,575, and a 'Get Started' button. The main content area features the headline 'APIs for building portable and reliable microservices' and a sub-headline 'Leverage industry best practices and focus on your application's logic.' Below this is another 'Get Started' button. A diagram illustrates microservices interacting via 'Invoke', 'Store', 'Publish', and 'Subscribe' actions. The footer displays logos for Bosch, Zeiss, Alibaba Cloud, Ignition Group, Roadwork, amop.com, Legentic, and Man. The bottom section is titled 'Build connected distributed applications faster' and contains two columns of text describing Dapr's capabilities.

Home Testimonials Docs Blog GitHub Discord ☆ Star 17,575 [Get Started](#)

## APIs for building portable and reliable microservices

Leverage industry best practices and focus on your application's logic.

[Get Started](#)

Invoke Store Publish Subscribe

**BOSCH** **ZEISS** **Alibaba Cloud** **IGNITION GROUP** **Roadwork** **高德地图 amop.com** **LEGENTIC** **Man**

### Build connected distributed applications faster

The Distributed Application Runtime (Dapr) provides APIs that simplify microservice connectivity. Whether your communication pattern is service to service invocation or pub/sub messaging, Dapr helps you write resilient and secured microservices.

By letting Dapr's sidecar take care of the complex challenges such as service discovery, message broker integration, encryption, observability, and secret management, you can focus on business logic and keep your code simple.

# 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`
- Slim mode does executable deployment (no Docker images)



## Virtual/Physical Machines

- Self-deploy Dapr control plane per machine
- Deploy Hashicorp Consul per machine
- Run any application with Dapr sidecar using `dapr run`
- Dapr Installer Package allows for offline/remote deployments with no network connectivity



## kubernetes

- Get started with `dapr init -k`
- Integrated 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

# 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

# Dapr APIs

## 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



Observability



Secrets



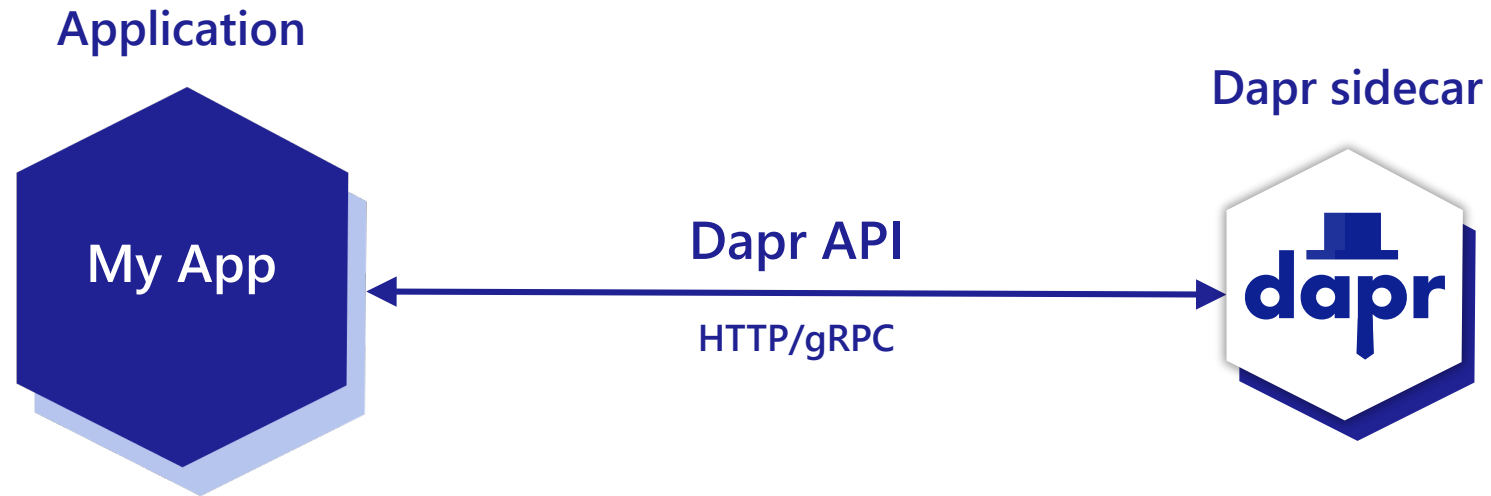
Configuration

## Any cloud or edge infrastructure



virtual or physical machines

# 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/order`

`GET http://localhost:3500/v1.0/secrets/vault/password42`

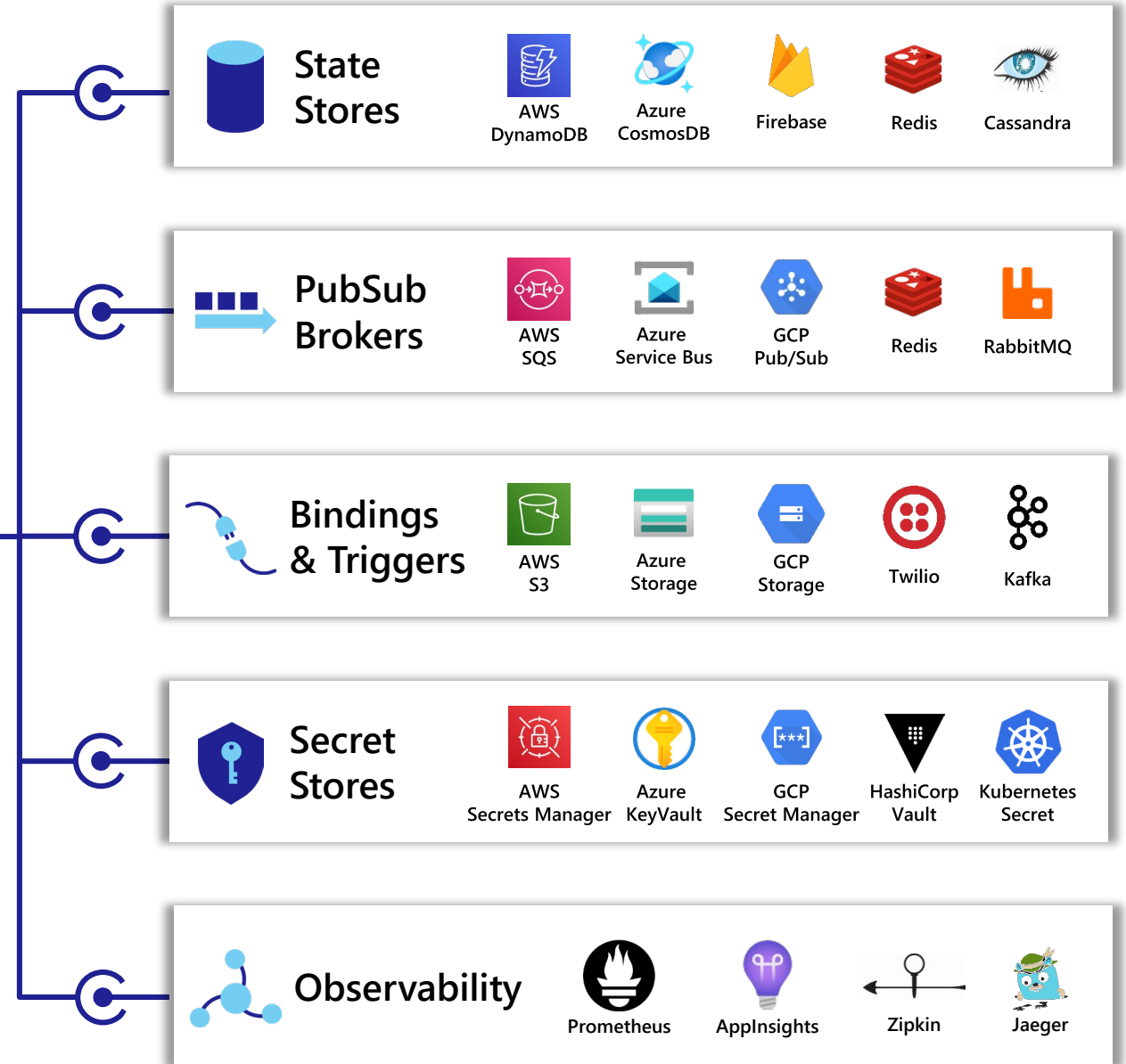
# Dapr components



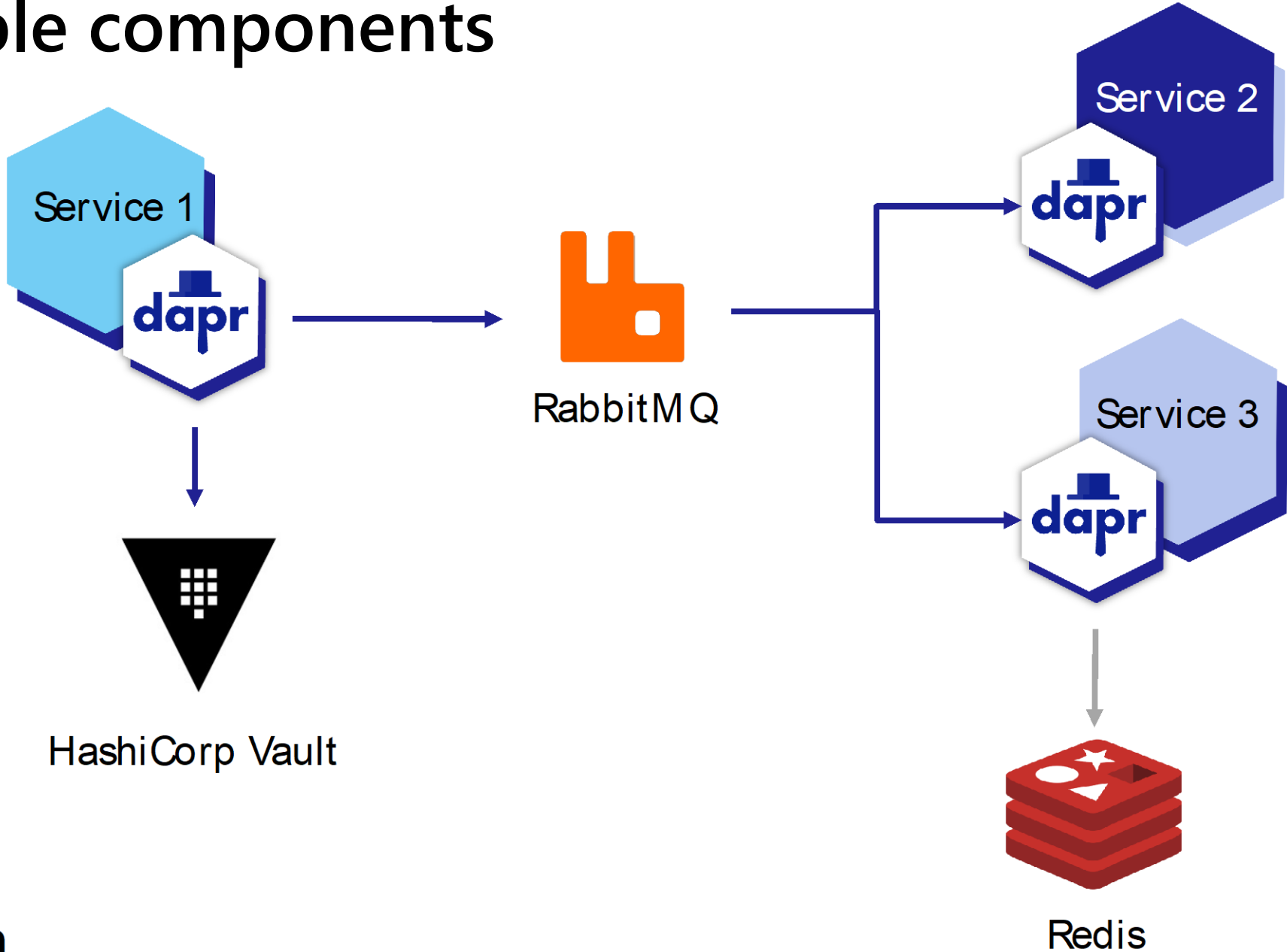
Swappable YAML files with resource connection details

Over 97 components available

Create components for your resource at:  
[github.com/dapr/components-contrib](https://github.com/dapr/components-contrib)

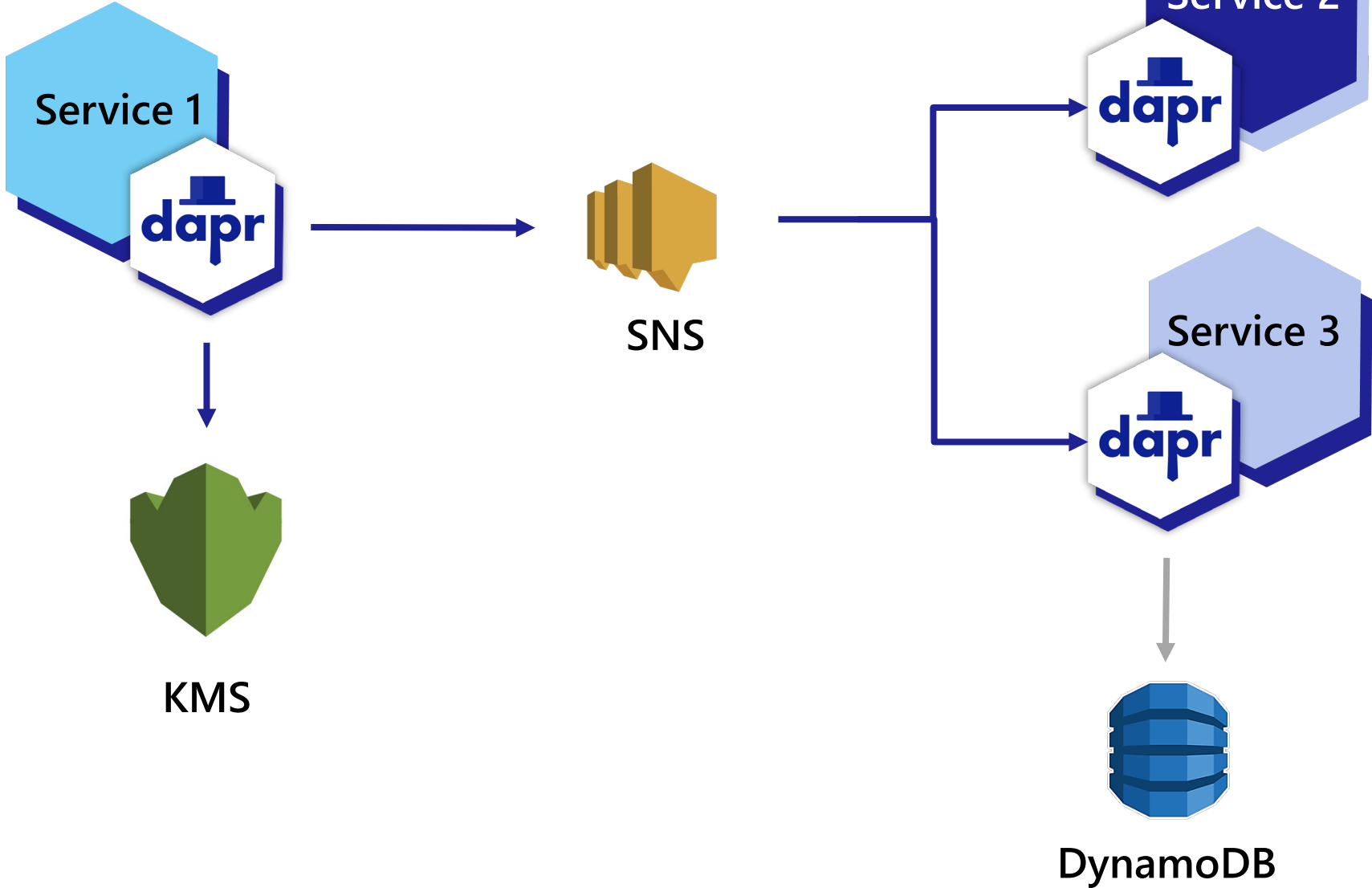


# Pluggable components

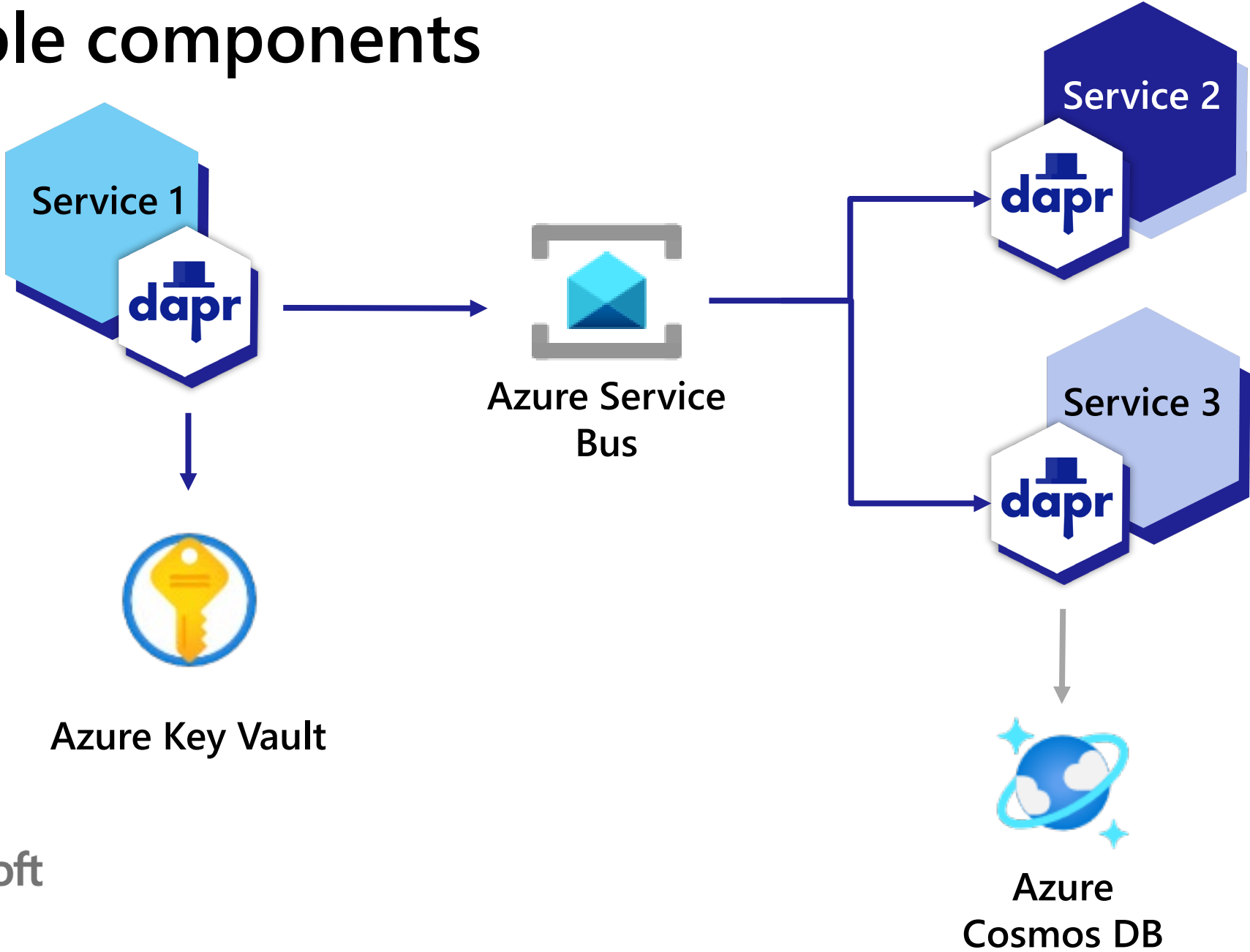


On-prem

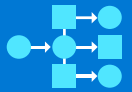
# Pluggable components



# Pluggable components



# Dapr building blocks



## Service-to-service invocation

---

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



## State management

---

Create long running, stateless and stateful services



## Publish and subscribe

---

Secure, scalable messaging between services



## Bindings (input/output)

---

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



## Actors

---

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



## Observability

---

See and measure the message calls across components and networked services



## Secrets

---

Securely access secrets from your application

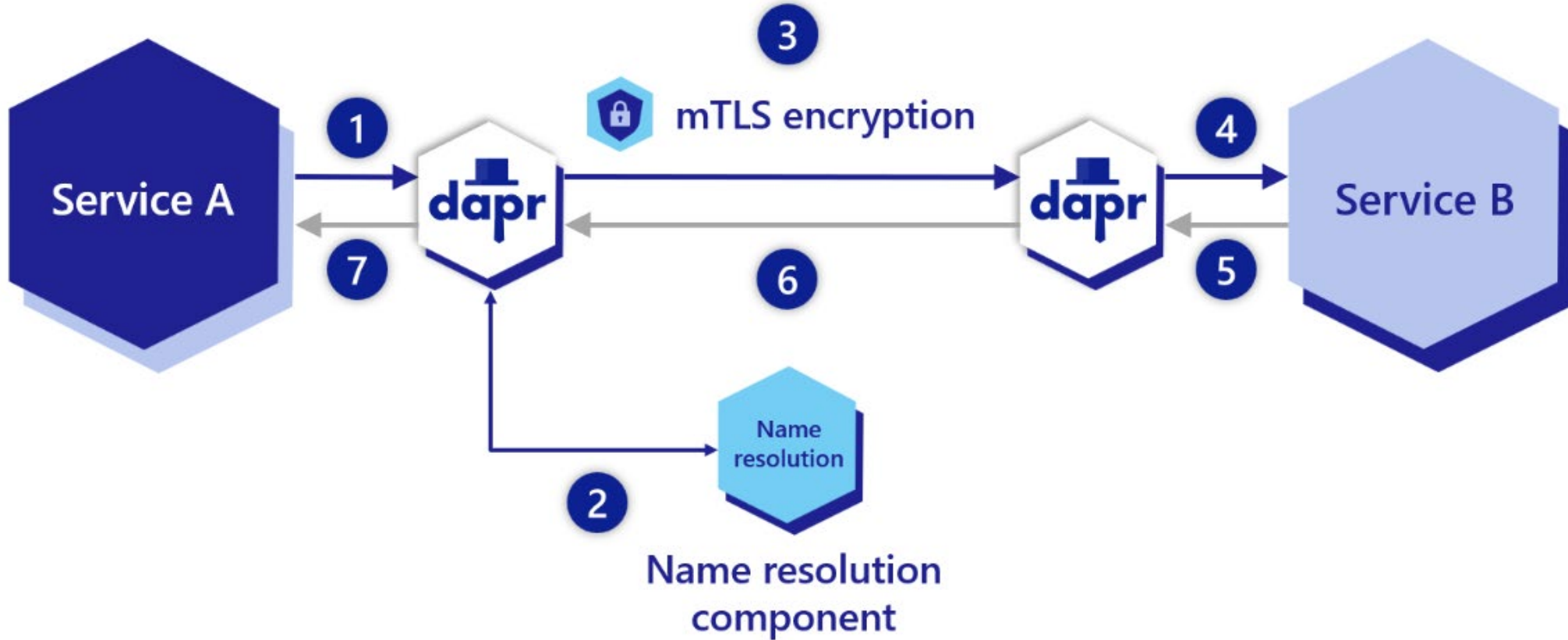


## Configuration

---

Access application configuration and be notified of updates

# Service invocation



Service-to-Service Security -> mTLS

Service-to-Service Communication -> HTTP or gRPC

Sidecar-to-Sidecar Communication -> gRPC



# State management



orderstore



Redis Cache

key	Field	value
myapp  order1	data	"myorder"
myapp  order1	version	1

**POST**

<http://localhost:3500/v1.0/state/orderstore>

```
[{  
  "key": "order1",  
  "value": "myorder"  
}]
```

**GET**

<http://localhost:3500/v1.0/state/orderstore/order1>

```
"myorder"
```

# Dapr state API

## Save state

POST /v1.0/state/orderstore

## Retrieve state

GET /v1.0/state/orderstore/myorder1

## Delete state

DELETE /v1.0/state/orderstore/myorder1

## Get bulk state

POST /v1.0/state/orderstore/bulk

## Submit multiple state transactions

POST /v1.0/state/corpdb/transaction

```
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: orderstore
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
```



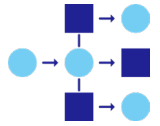
# State management features

- Time-to-Live

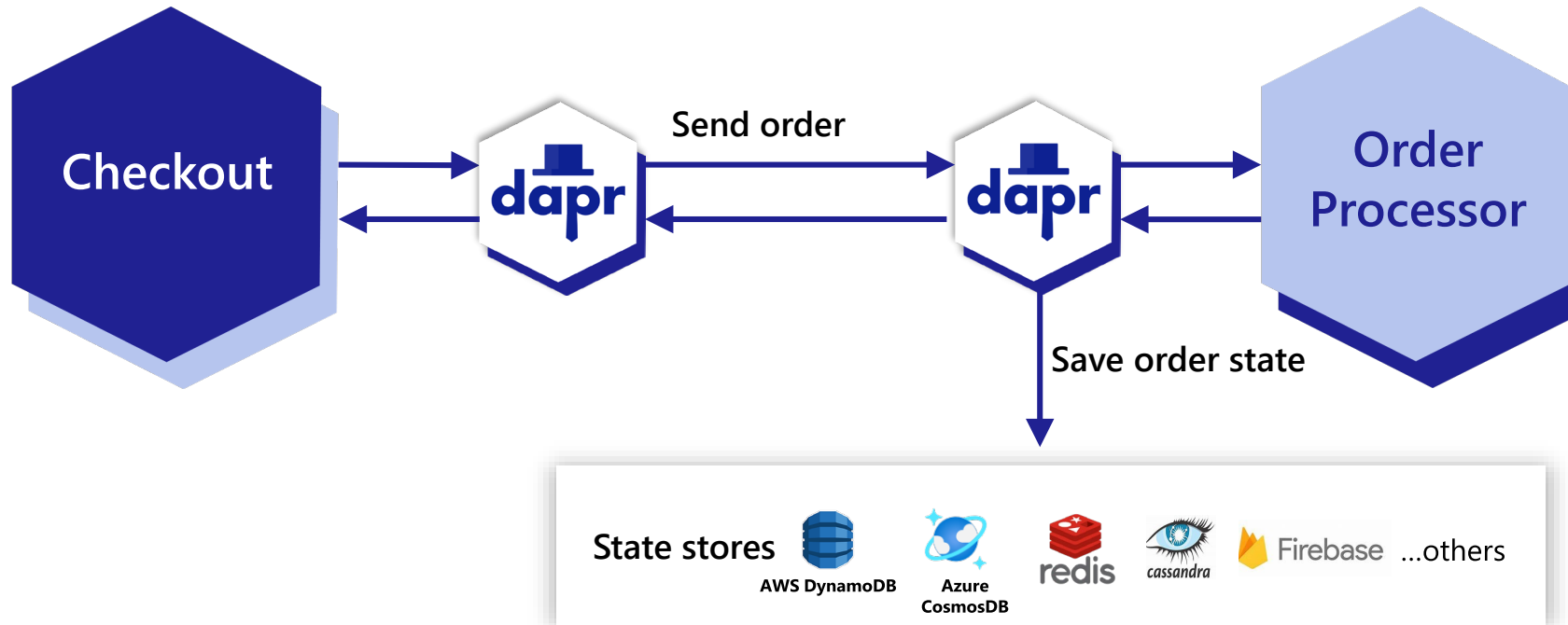
We may want the declared state to be valid for a certain time.

- Secret Store

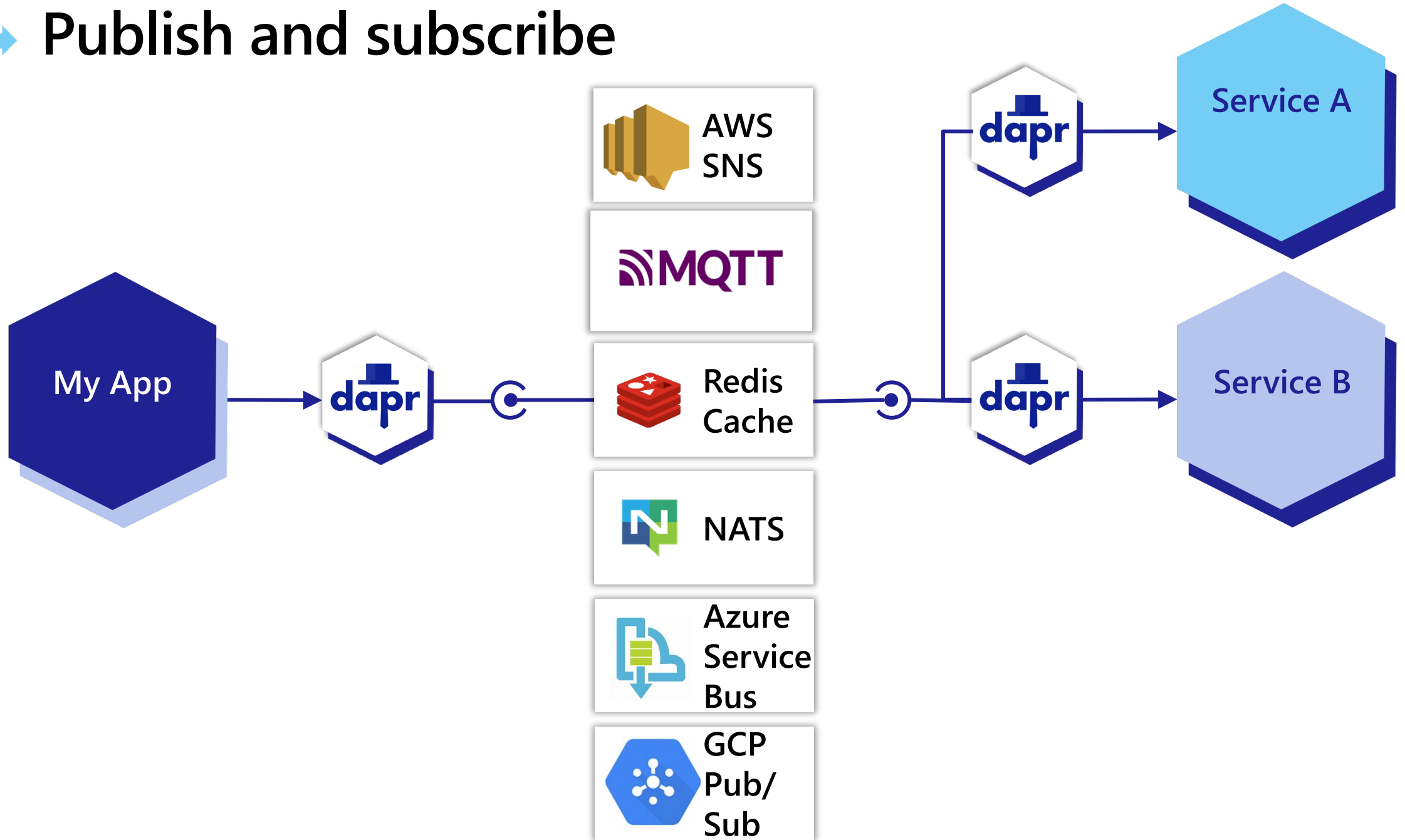
Defining important information directly as a plain string in the definition of components can cause serious security problems.



# Service invocation & state management

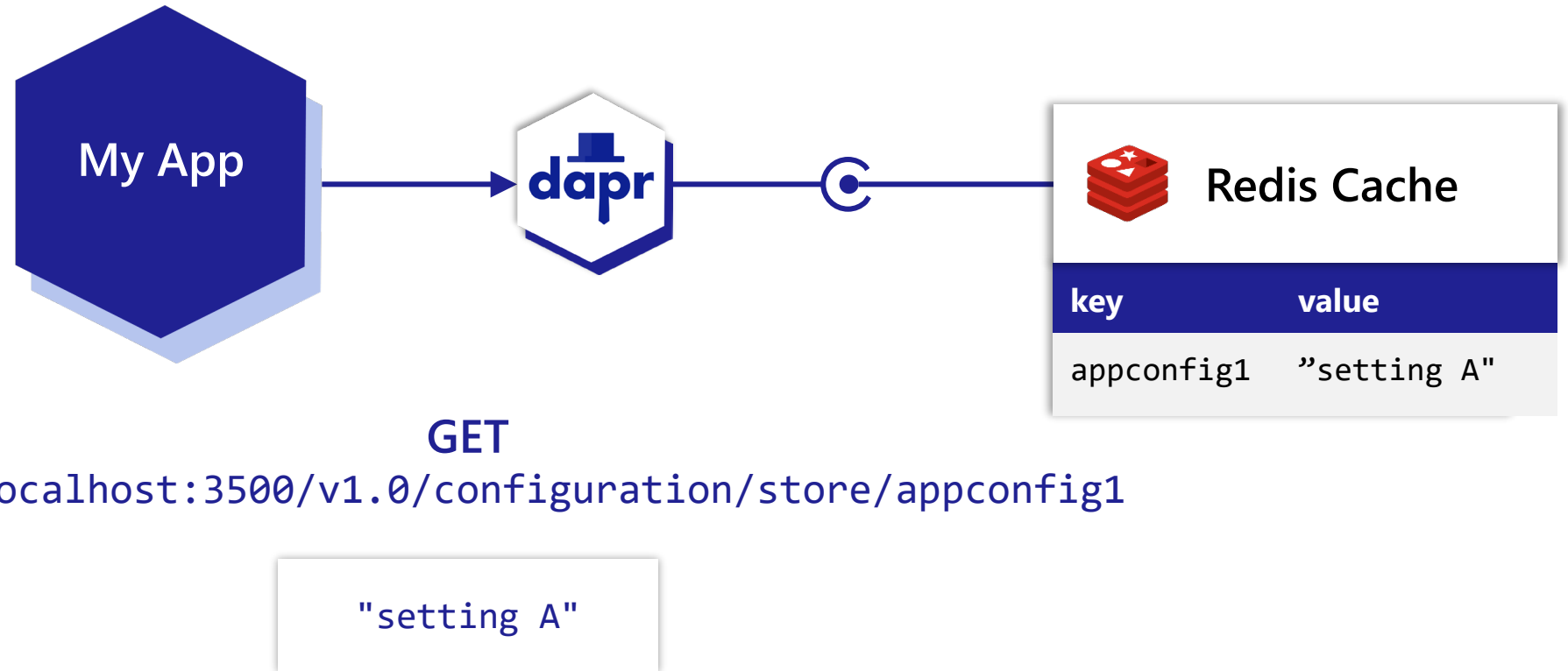


# Publish and subscribe



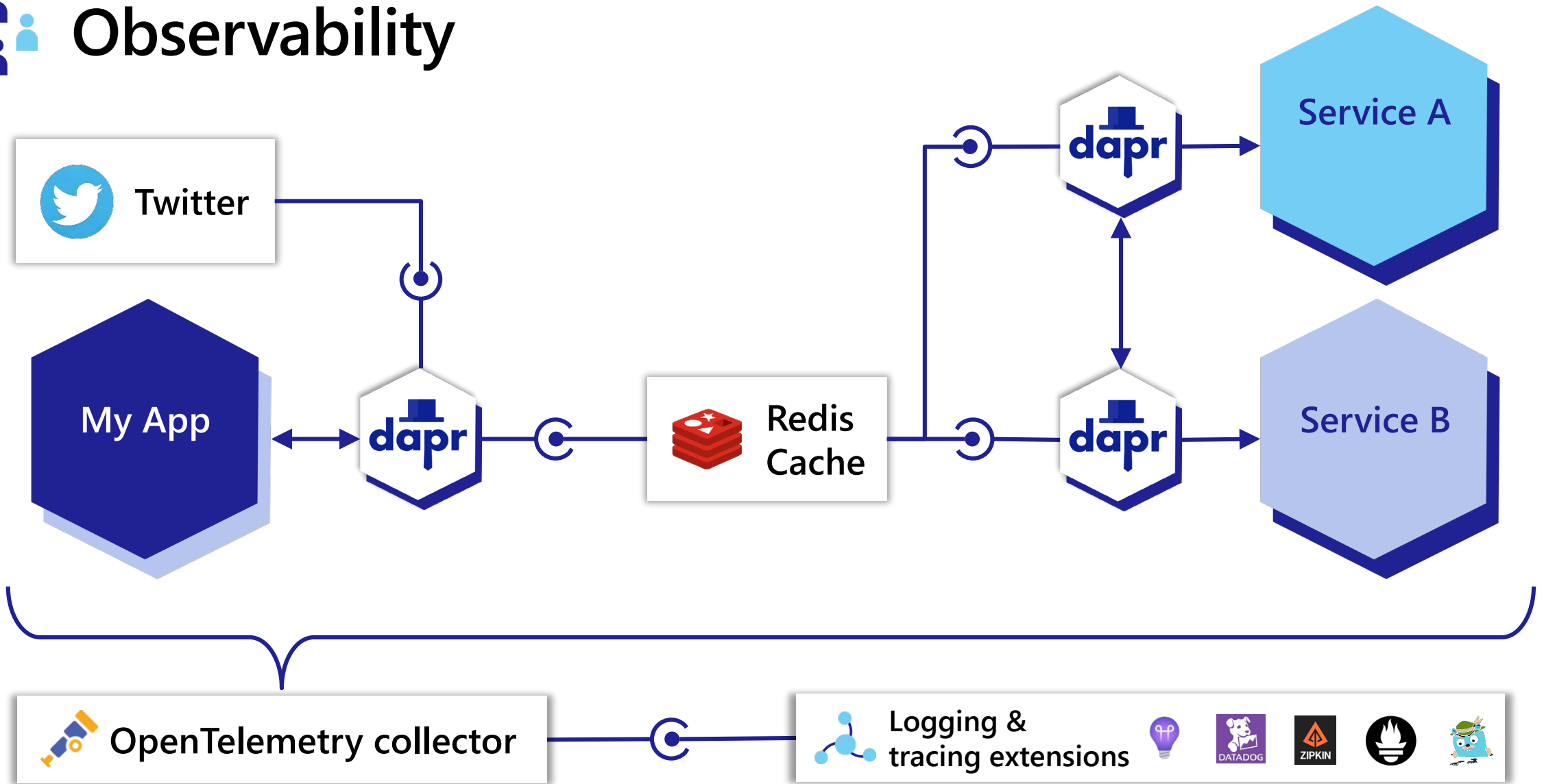


# Configuration





# Observability

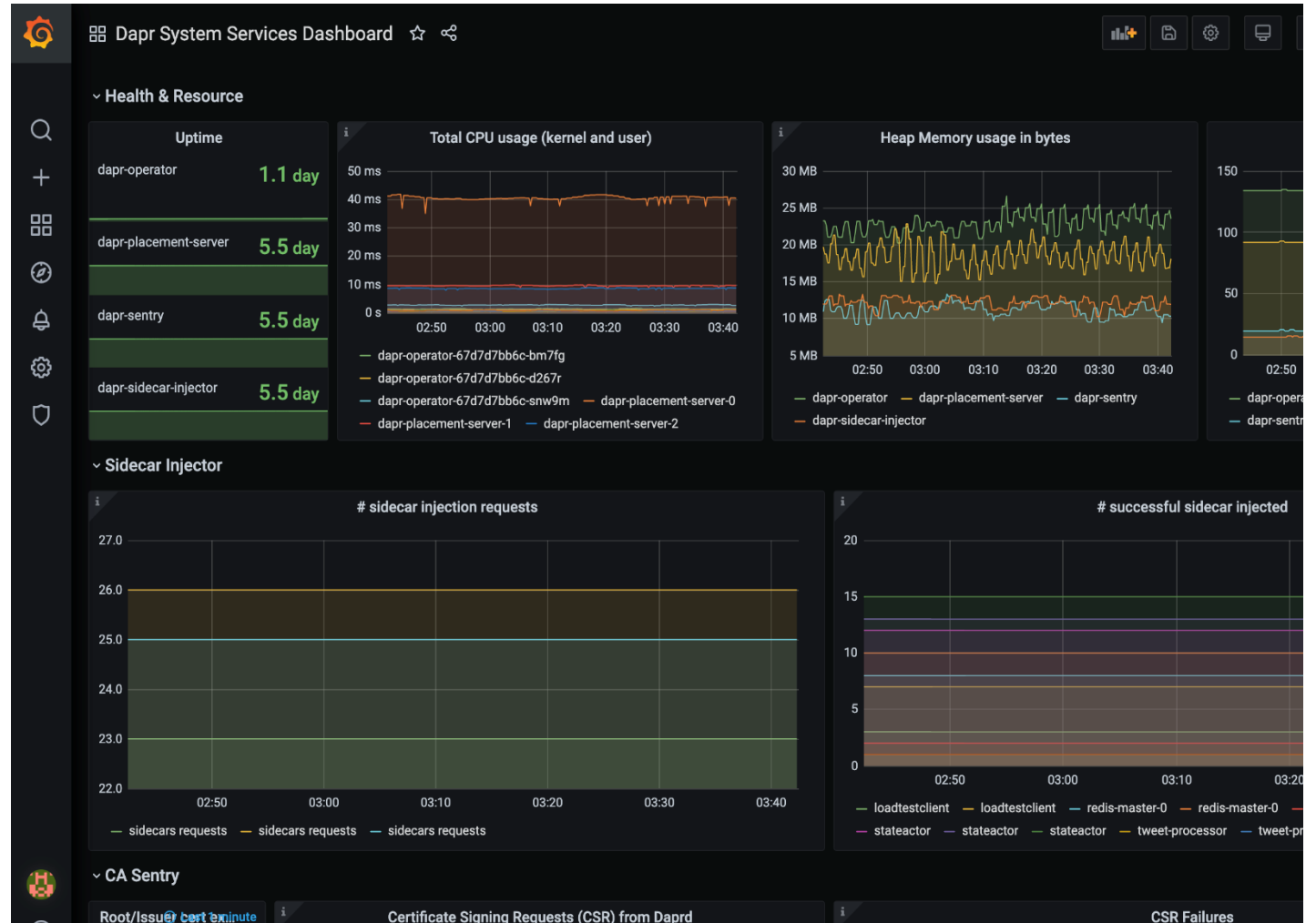


# Metrics

Built-in monitoring capabilities to understand the behavior of the Dapr sidecar and system services

## Dapr Metrics features:

- ✓ Call latency
- ✓ CPU/memory usage
- ✓ Error rates
- ✓ Sidecar injection failures
- ✓ System health



# Resiliency

- Resiliency patterns can be applied across Dapr APIs
  - Retries
  - Timeouts
  - Circuit breakers
- Declarative and decoupled from application code
- Available across all component types, service invocation and actors.

```
1  apiVersion: dapr.io/v1alpha1
2  kind: Resiliency
3  metadata:
4    name: resiliency
5  spec:
6
7    policies:
8      timeouts:
9        general: 5s
10
11     retries:
12       fiveRetries:
13         policy: constant
14         duration: 5s
15         maxRetries: 5
16
17     circuitBreakers:
18       simpleCB:
19         maxRequests: 1
20         timeout: 10s
21         trip: consecutiveFailures >= 2
22
23     targets:
24       apps:
25         processor:
26           timeout: general
27           retry: fiveRetries
28           circuitBreaker: simpleCB
29
```

# Questions?



Get started at <http://dapr.io>



Contribute at [github.com/dapr](https://github.com/dapr)

**dap<sup>r</sup>**