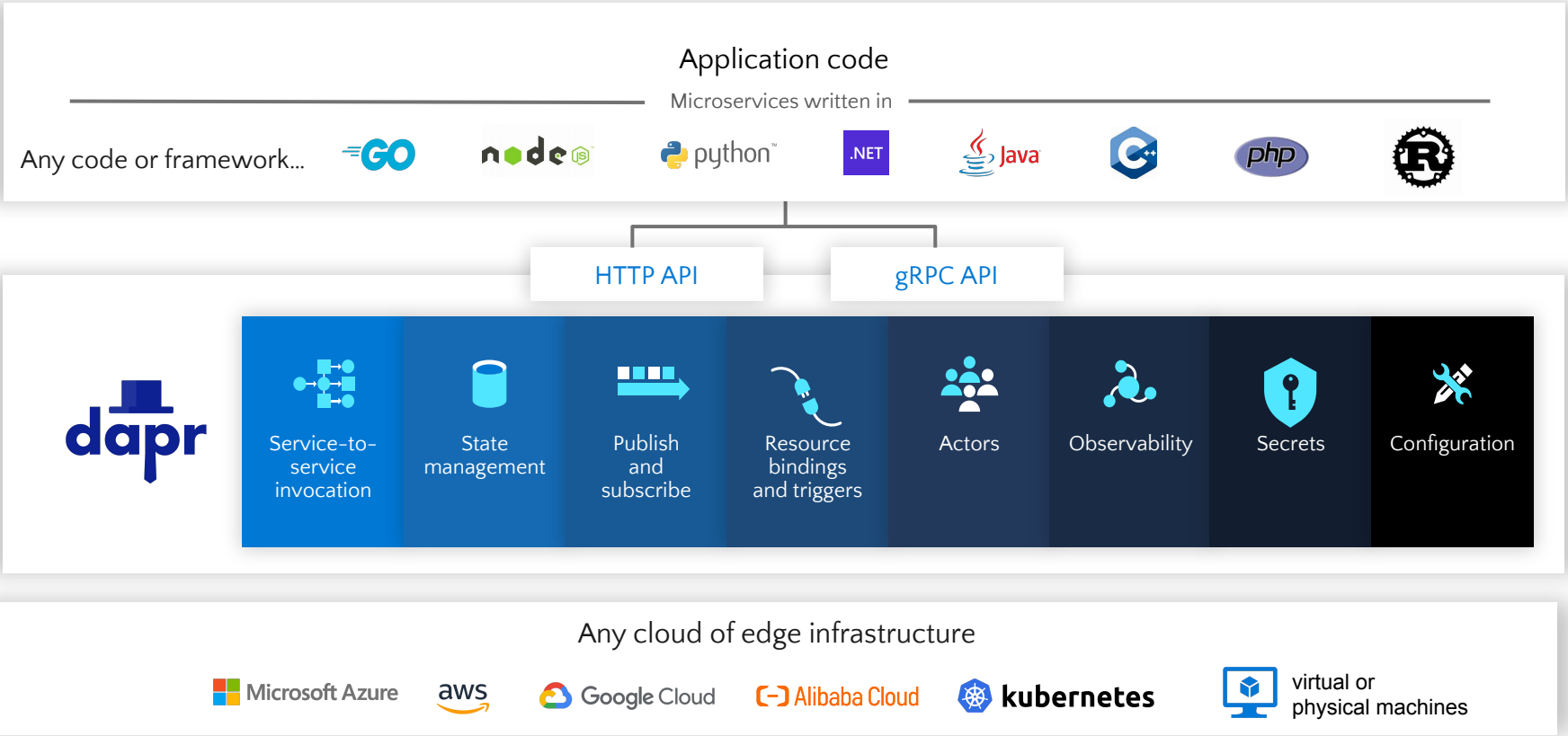
KubeCon CloudNativeCon Europe 2022

Making your apps and infrastructure services resilient with Dapr

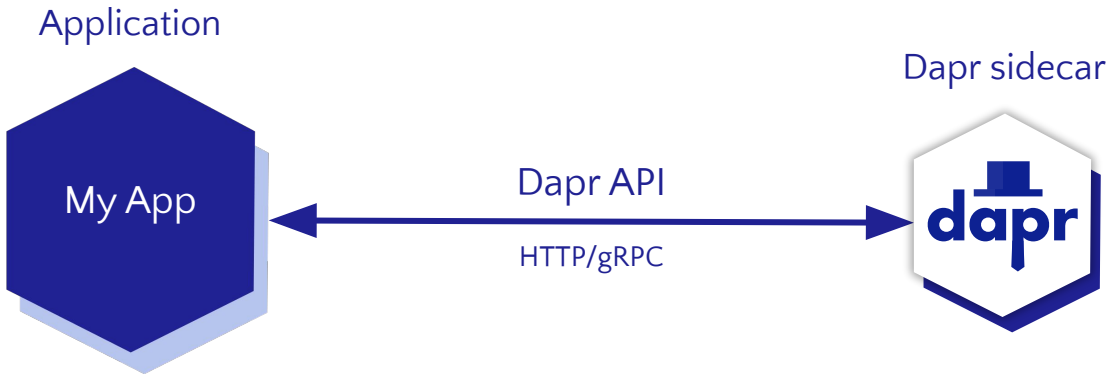
Yaron Schneider - Co-Founder / CTO @ Diagrid

Hal Spang - Senior Software Engineer @ Microsoft

What is Dapr?

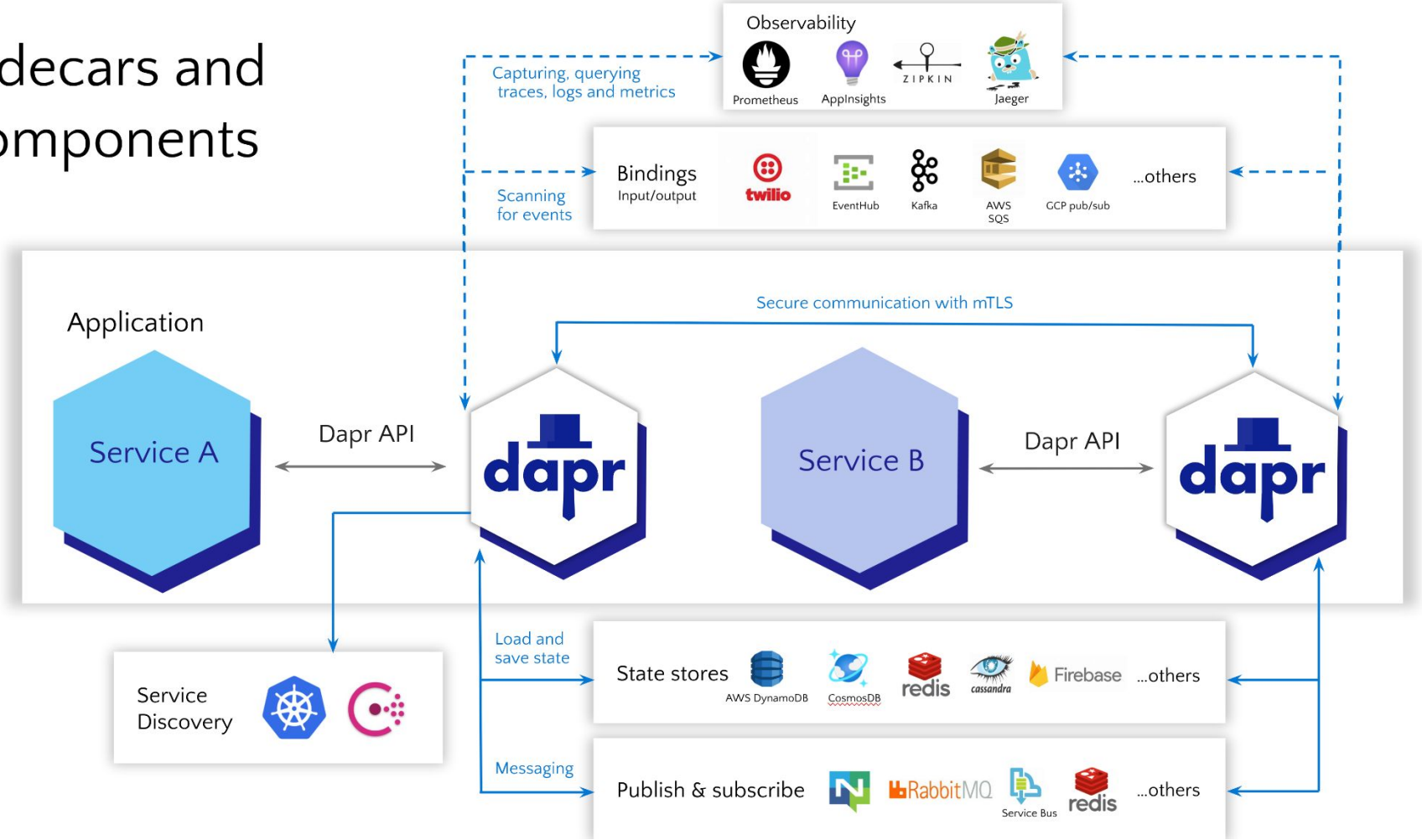

Application code
Microservices written in
Any code or framework...
HTTP API
gRPC API
dapr
Service-to-service invocation
State management
Publish and subscribe
Resource bindings and triggers
Actors
Observability
Secrets
Configuration
Any cloud of edge infrastructure
Microsoft Azure
aws
Google Cloud
Alibaba Cloud
kubernetes
virtual or physical machines

Sidecar

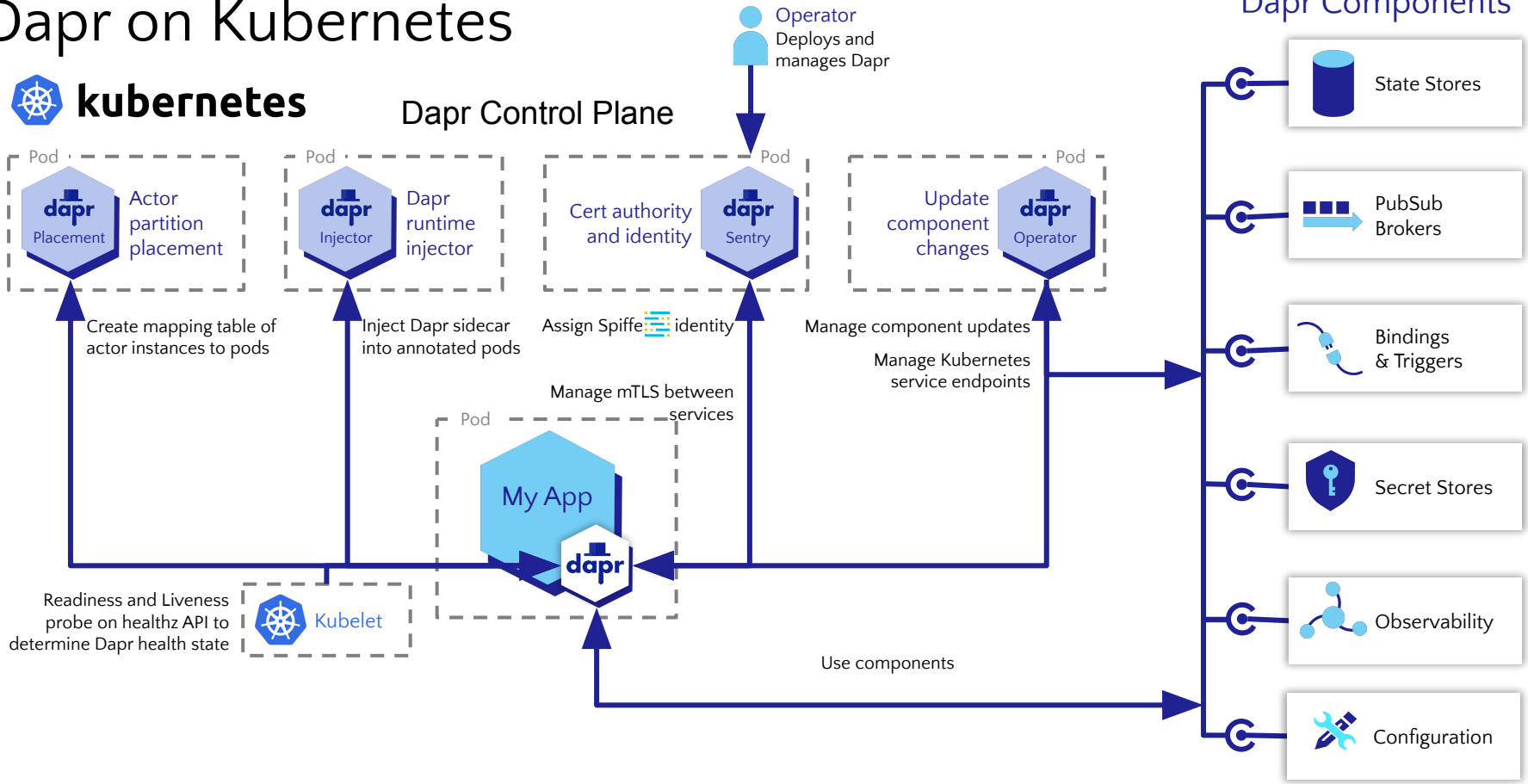



```
POST http://localhost:3500/v1.0/invoke/cart/method/neworder

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
```

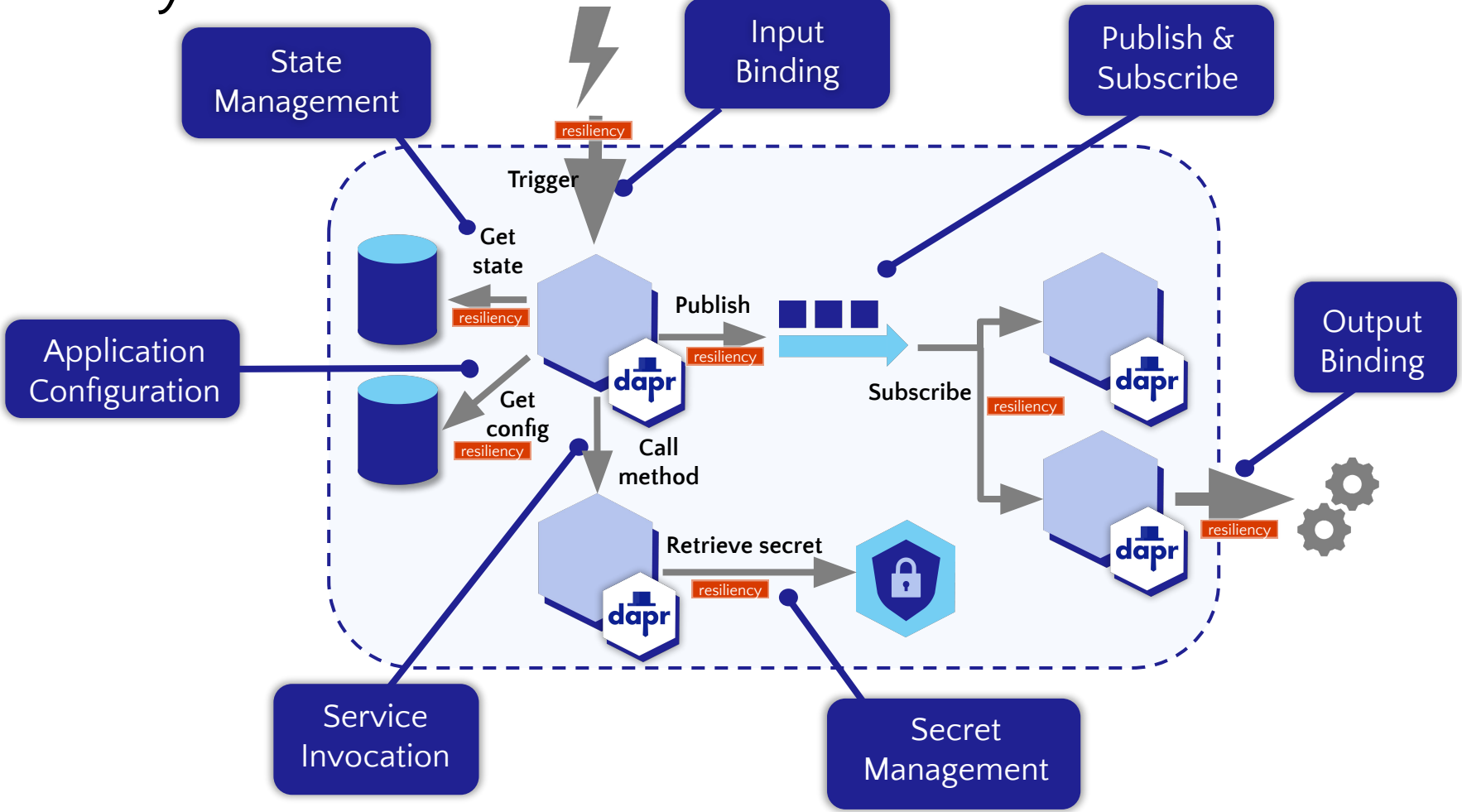
Sidecars and components



Dapr on Kubernetes



Resiliency

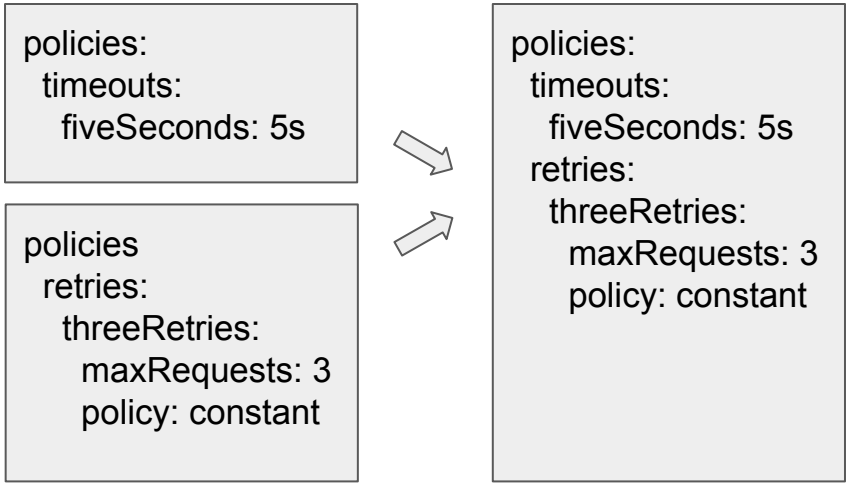

State Management
Input Binding
Publish & Subscribe
resiliency
Trigger
Get state
resiliency
Publish
resiliency
Application Configuration
Get config
resiliency
Call method
Subscribe
resiliency
Output Binding
Retrieve secret
resiliency
resiliency
Service Invocation
Secret Management
dapr

Resiliency Configuration YAML

```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: resiliency
spec:

  policies:
    # Timeouts are simple named durations.
    timeouts:
      fast: 2s

    # Retries are named templates for and are instantiated for life of the operation.
    retries:
      pubsubRetry:
        policy: constant
        duration: 5s
        maxRetries: 10

    # Circuit breakers are automatically instantiated per component, service endpoint, and application route.
    # using these settings as a template. See logic below under `buildingBlocks`.
    # Circuit breakers maintain counters that can live as long as the Dapr sidecar.
    circuitBreakers:
      pubsubCB:
        maxRequests: 1
        interval: 30s
        timeout: 60s
        trip: consecutiveFailures > 1

      stateCB:
        maxRequests: 1
        interval: 30s
        timeout: 60s
        trip: consecutiveFailures > 1

  # This section specifies default policies for:
  # * service invocation
  # * requests to components
  # * actor invocation
  targets:
    components:
      asb:
        outbound:
          retry: pubsubRetry
          circuitBreaker: pubsubCB
        inbound:
          timeout: fast
          retry: pubsubRetry
          circuitBreaker: pubsubCB

      cosmosdb-state:
        outbound:
          circuitBreaker: stateCB
```

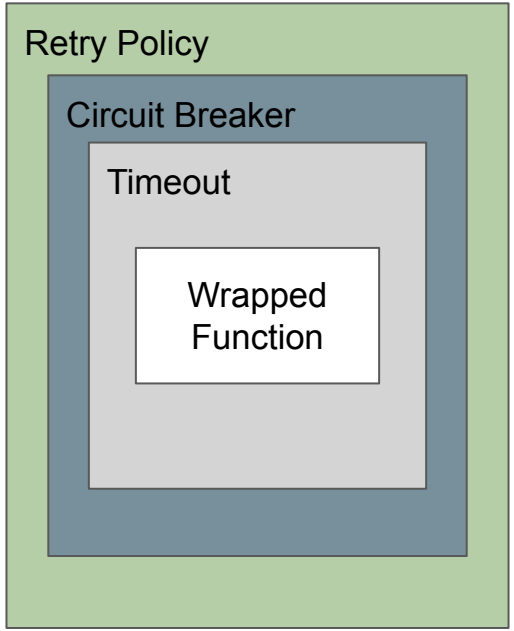
Resiliency as CRD

- In kubernetes Resiliency is defined as a CRD
- Allows for multiple policies to be defined
- Dapr merges all found policies into single configuration



Resiliency Policies

- **Timeouts**
  - Allows for the cancellation of requests after a given duration
- **Retries**
  - Allows for the generic retrying of a request or operation
  - Two supported retry types, constant and exponential
  - Can specify errors which are retryable and permanent
- **Circuit Breakers**
  - Allows for broken/breaking systems to be cut-off from requests
  - Helps reduce traffic and requests to allow for recovery time



Resiliency Policies - Retries

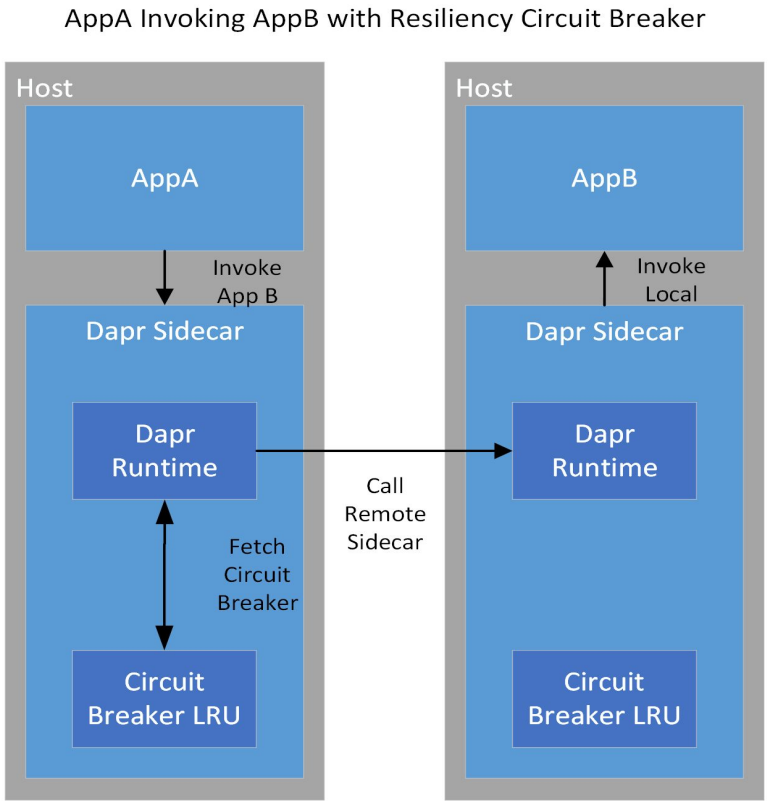
Constant Policies

- **maxRetries** - The maximum number of attempts to make for a request
- **duration** - The time in-between retries

Exponential Policies

- **maxRetries** - The maximum number of attempts to make for a request
- **initialInterval** - The starting time between retries
- **randomizationFactor** - Jitter used to offset requests
- multiplier - Growth rate of the retry interval
- **maxInterval** - The maximum duration between retries
- **maxElapsedTime** - The maximum time spent over all retries

Resiliency Policies - Circuit Breakers



- **maxRequests** - The maximum number of requests allowed while the breakers is in the half-open state
- **interval** - The cyclical period that errors are evaluated in, if not specified the evaluation period is continuous
- **timeout** - The time in which the circuit breaker will remain open after breaking
- **trip** - The criteria that errors are evaluated against to trigger state changes

Resiliency Targets

- Can be defined as Applications, Actors, and Components
- A target maps the policies to be used when calling **into** a system

```yaml
apps:
  appA:
    timeout: general
    retry: serviceRetry
    circuitBreaker: serviceCB
```

```yaml
components:
  pubsub:
    outbound:
      retry: pubsubRetry
      circuitBreaker: pubsubCB
    inbound:
      timeout: general
      retry: pubsubRetry
      circuitBreaker: pubsubCB
```

```yaml
actors:
  myActorType:
    timeout: general
    retry: actorRetry
    circuitBreaker: actorCB
    circuitBreakerScope: type
    circuitBreakerCacheSize: 5000
```

Target Examples

Calling AppA:

- **Retry**: generalRetry
- **Timeout**: fast
- **Circuit Breaker**: appACB

Calling AppB:

- **Retry**: appBRetry
- **Timeout**: slow
- **Circuit Breaker**: None

```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: resiliency
spec:

  policies:
    timeouts:
      fast: 2s
      slow: 10s

    retries:
      generalRetry:
        policy: constant
        duration: 5s
        maxRetries: 10

      appBRetry:
        policy: exponential
        maxInterval: 20s

    circuitBreakers:
      appACB:
        maxRequests: 1
        interval: 30s
        timeout: 60s
        trip: consecutiveFailures > 1

  targets:
    apps:
      appA:
        retry: generalRetry
        timeout: fast
        circuitBreaker: appACB

      appB:
        retry: appBRetry
        timeout: slow

```

Demo