Simplifying Microservices with Dapr

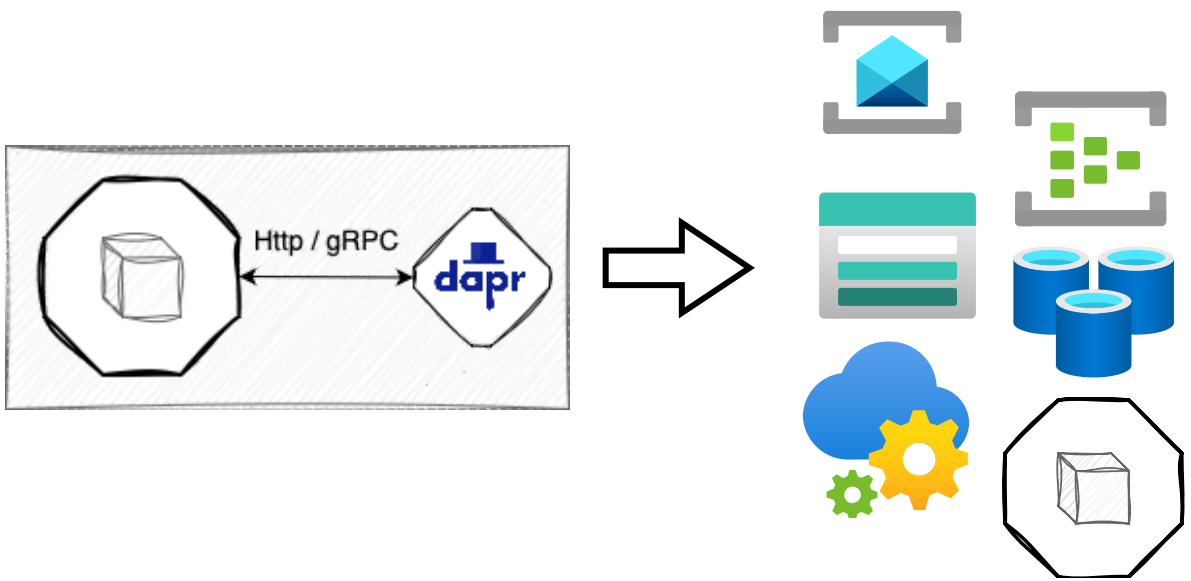
matt sheehan

Github Code

https://github.com/MattSheehanDev/SimplifyingMicroservicesWithDapr

Introducing dapr

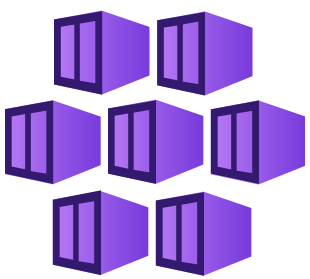
- **D**istributed **Ap**plication **R**untime
- Introduced in Oct. 2019 as a Microsoft open-source project
- Joined Cloud Native Computing Foundation (CNCF) in Nov. 2021 as an incubating project
- Sidecar to enable highly distributed apps



Hosting

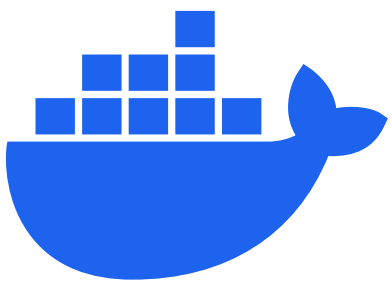
Kubernetes

(GKE, AKS, EKS)



Virtual Machine

Docker container



benefits

Language neutral

Provides flexibility and simplicity

Decreases your API surface area

Codifies best patterns and practices

Provides a standard and declarative configuration

Moves design decisions downstream

building blocks

Configuration & Secrets

Service-to-Service Invocation

State Management

Publish and Subscribe

Resource Bindings

Actors

dapr configuration

- An implementation of a **Building Block** is a *Component*
- **Components** are configured with *Component Specs*

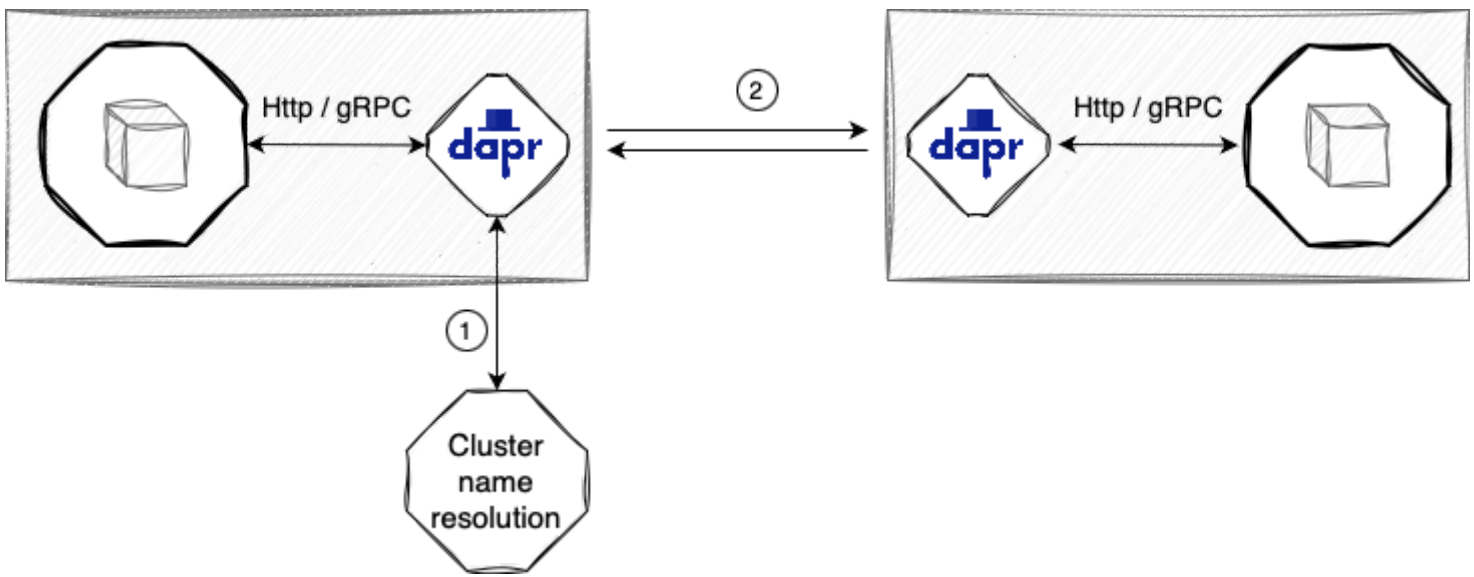
```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: statestore
spec:
  type: state.redis
  version: v1
  metadata:
  - name: redisHost
    value: localhost:6379
  - name: redisPassword
    value: ""
  - name: actorStateStore
    value: "true"
```

```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: ""
```

https://github.com/dapr/components-contrib

dapr service invocation

http://localhost:3500/v1.0/invoke/<service>/method/<path>

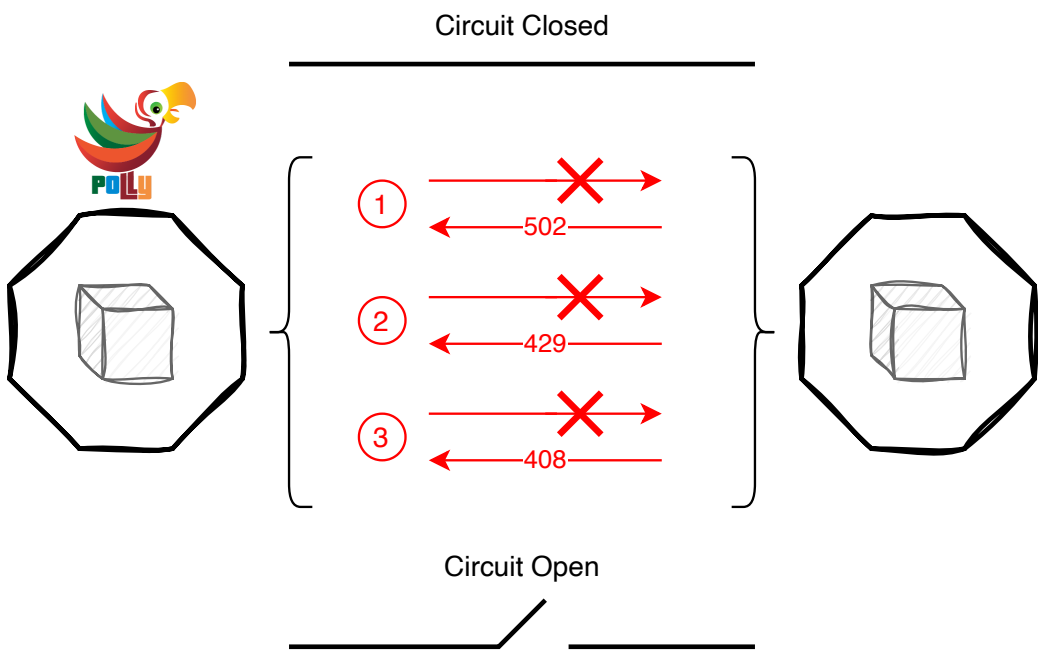


- Service discovery
- HTTP or gRPC
- Supports retries and circuit breaker patterns

CLOUD NATIVE
COMPUTING FOUNDATION

Retry & Circuit Breaker Pattern

- Retry Pattern - Retry an operation with the expectation that it will succeed
- Circuit Breaker Pattern - Prevent an operation that is likely to fail



service invocation resiliency

```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: generalresiliencypolicy
version: v1alpha1
scopes:
  - appA
  - appB
spec:
  policies:
    timeouts:
      general: 10s
      longResponse: 60s
    retries:
       general:
        policy: constant
        duration: 5s
        maxRetries: 10
    circuitBreakers:
      serviceCB:
        maxRequests: 1
        timeout: 30s
        trip: consecutiveFailures >= 5
targets:
    apps:
      appA:
        timeout: general
        retry: general
        circuitBreaker: serviceCB
```

dapr service invocation

https://github.com/MattSheehanDev/SimplifyingMicroservicesWithDapr/tree/main/01.service-invocation

dapr configuration secrets

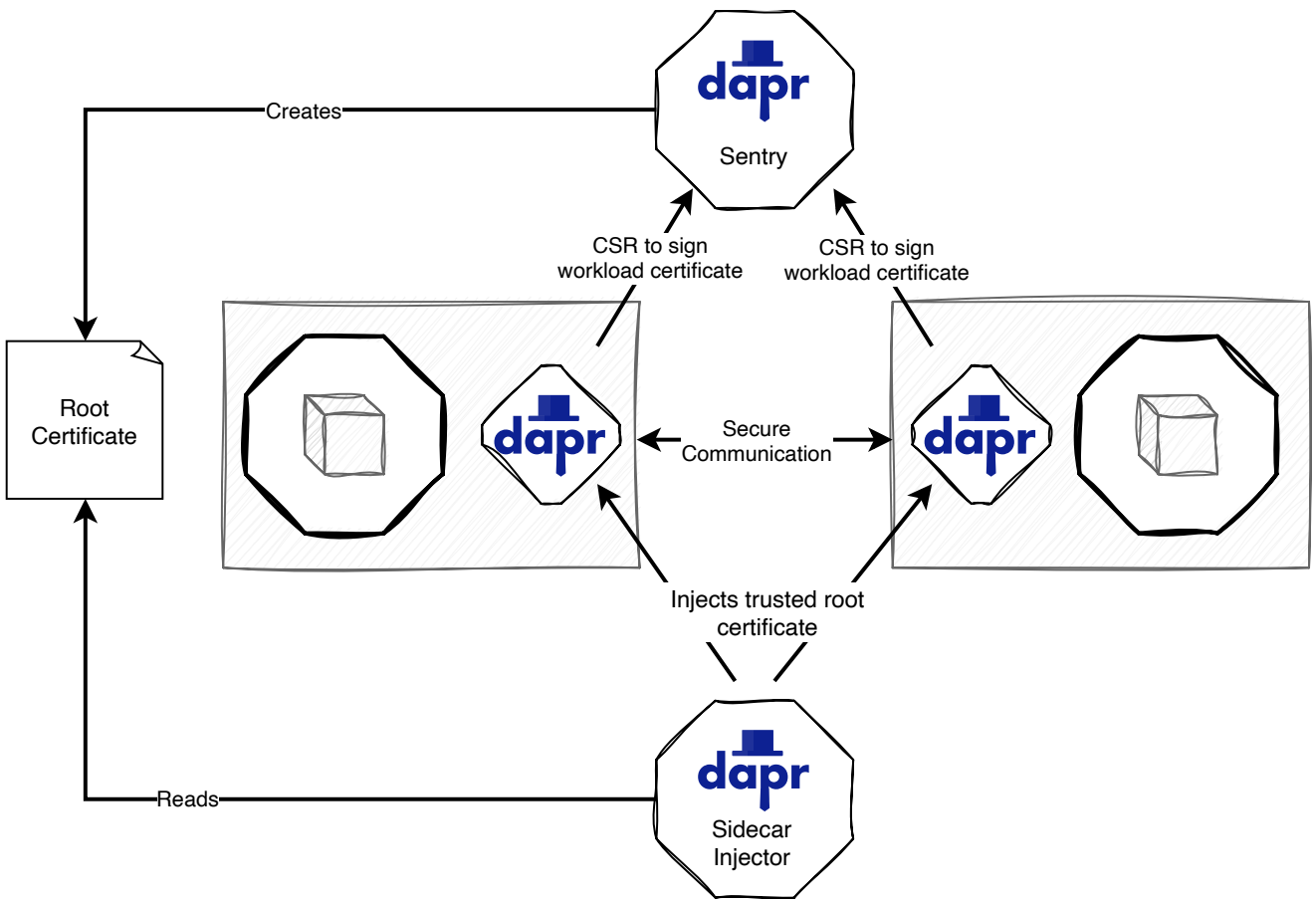
```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: secretstore-file
spec:
  type: secretstores.local.file
  version: v1
  metadata:
  - name: secretsFile
    value: /configuration/secrets.json
  - name: namedSeparator
    value: ":"
  - name: multiValued
    value: "false"
```

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: statestore
auth:
  secretStore: secretstore-file
spec:
  type: state.redis
  version: v1
  metadata:
  - name: redisHost
    value: localhost:6379
  - name: redisPassword
    secretRef:
      name: secretName
      value: secretKey
  - name: actorStateStore
    value: "true"
```

http://localhost:3500/v1.0/secrets/<secretstore>/<secretname>

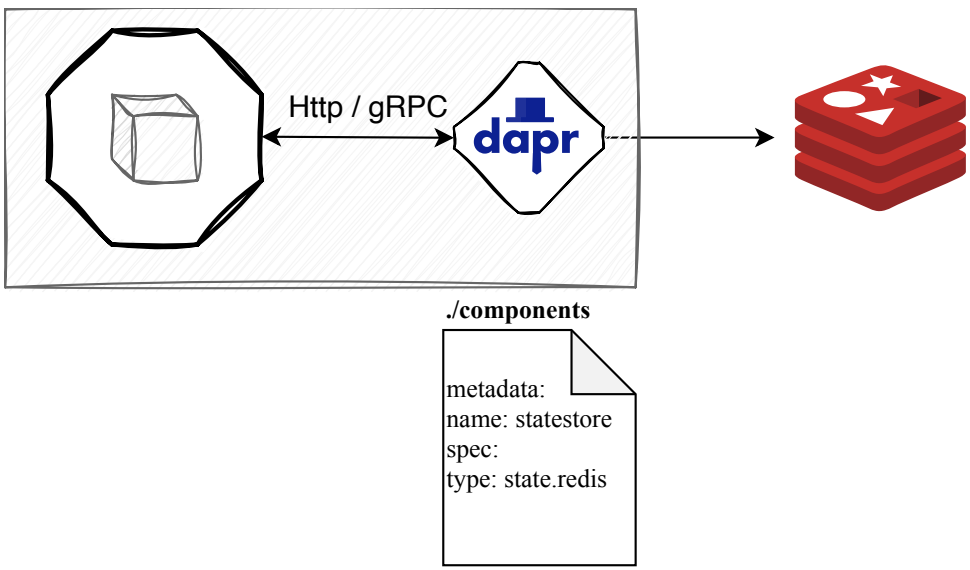
dapr mTLS

- Certificates to prove both the client and server identity
- Dapr Sentry acts as a Certificate Authority
- Dapr Sidecar Injector adds CA cert as trusted root cert



dapr state management

http://localhost:3500/v1.0/state/<storename>/<key>



- Consistency
  - Strong vs Eventual
- Concurrency
  - Last-write wins
  - First-write wins (Optimistic Concurrency Control)

state management resiliency

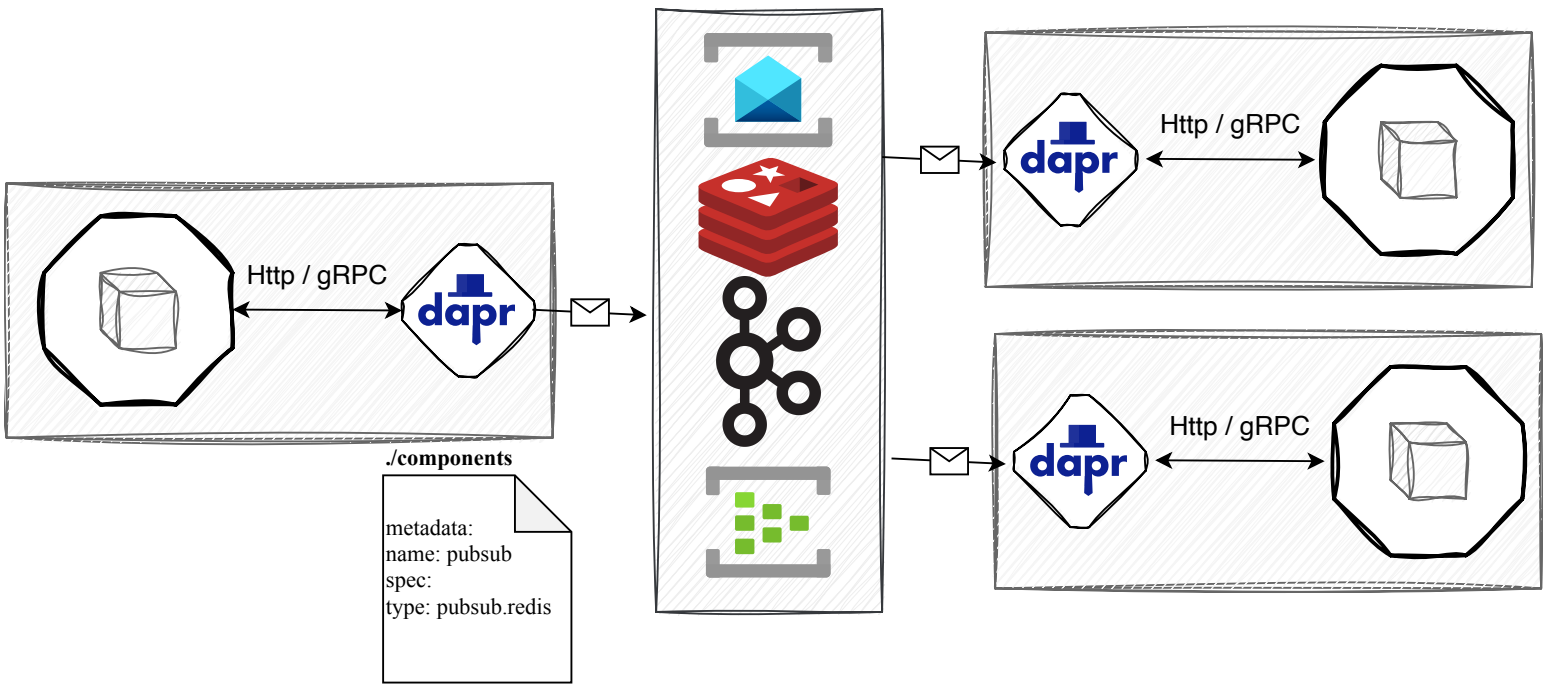
```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: statestoreresiliencypolicy
version: v1alpha1
scopes:
  - appA
spec:
  policies:
    timeouts:
      general: 5s
    retries:
      retryForever:
        policy: exponential
        maxInterval: 15s
        maxRetries: -1
    circuitBreakers:
      componentCB:
        maxRequests: 1
        timeout: 30s
        trip: consecutiveFailures >= 5
targets:
    components:
      statestore:
        outbound:
          timeout: general
          retry: retryForever
          circuitBreaker: componentCB
```

dapr state management

https://github.com/MattSheehanDev/SimplifyingMicroservicesWithDapr/tree/main/02.state-management

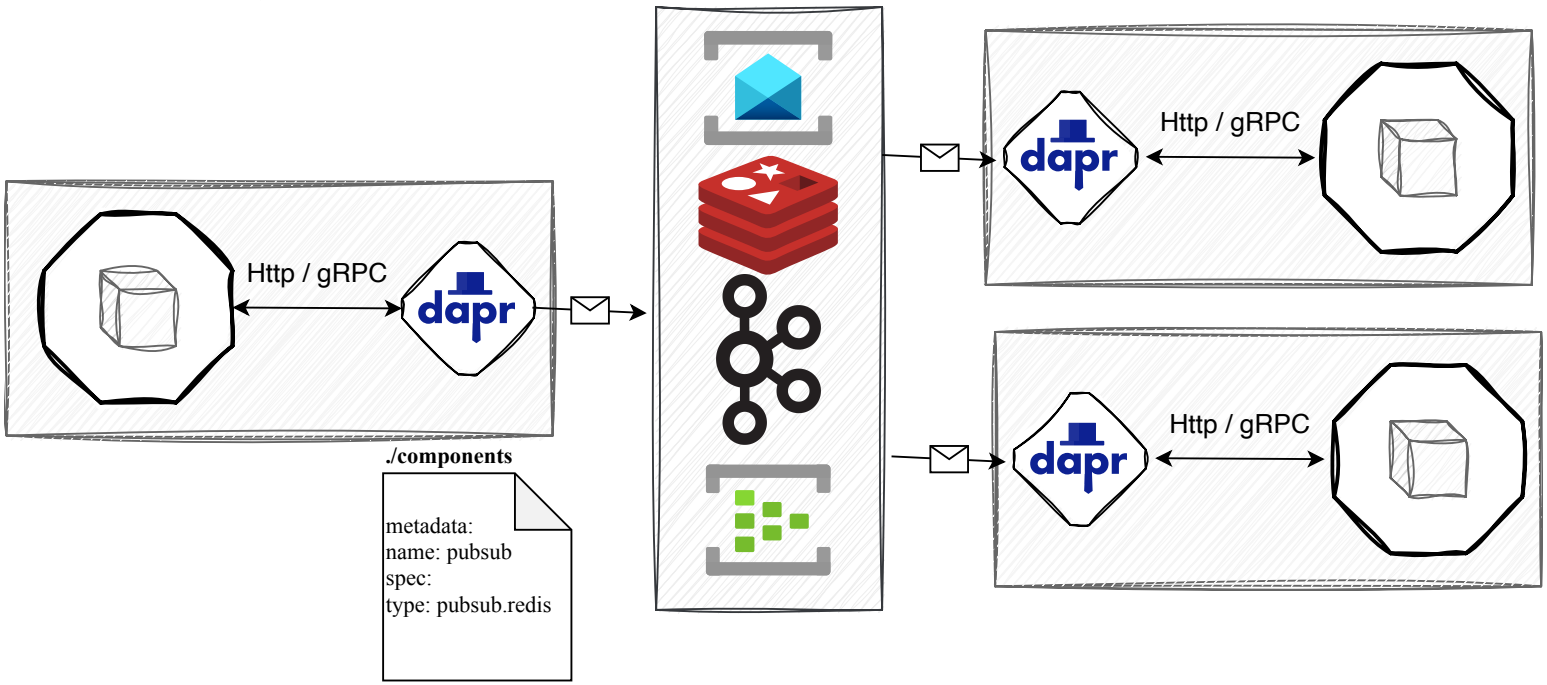
dapr publish/subscribe

```
http://localhost:3500/v1.0/publish/<pubsubname>/<topic>
```



- At-least-once message delivery
- Content-based routing
- Cloud Events v1.0 spec

dapr publish/subscribe



2xx	Message is processed
404	Message is dropped
5xx	Message is retried

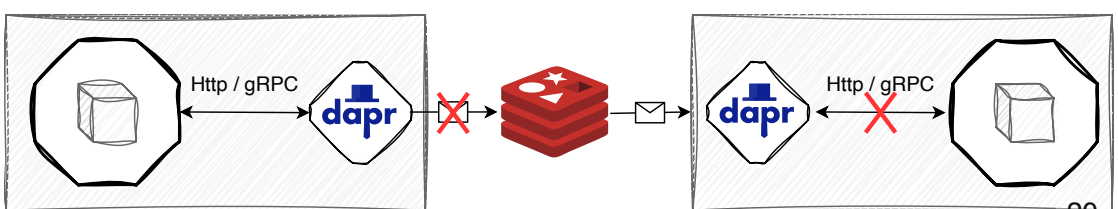
Cloud Events

- Enables content-based routing
- Enables message de-duplication
- Improves interoperability

```json
{
  "id": "ade1a5ef-ba37-4a12-935f-6ca6c5c1254a",
  "source": "app_name",
  "specversion": "1.0",
  "type": "com.dapr.event.sent",
  "topic": "topic_name",
  "pubsubname": "pub_sub_name",
  "data": {
    // serialized message data
  },
  "datacontenttype": "application/json; charset=utf-8",
  "time": "2024-01-08T13:00:00Z",
  "traceid": "00-113ad9c4e42b27583ae98ba698d54255-e3743e35ff56f219-01",
  "tracestate": "",
  "traceparent": "00-113ad9c4e42b27583ae98ba698d54255-e3743e35ff56f219-01"
}
```

dapr resiliency

```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: resiliencypolicy
version: v1alpha1
spec:
  policies:
    timeouts:
      general: 5s
    retries:
      constantRetry:
        policy: constant
        duration: 5s
        maxRetries: 10
    circuitBreakers:
      pubsubCB:
        maxRequests: 1
        interval: 8s
        timeout: 45s
        trip: consecutiveFailures > 5
targets:
    components:
      pubsub:
        outbound:
          retry: constantRetry
          circuitBreaker: pubsubCB
        inbound:
          timeout: general
          retry: constantRetry
          circuitBreaker: pubsubCB
```

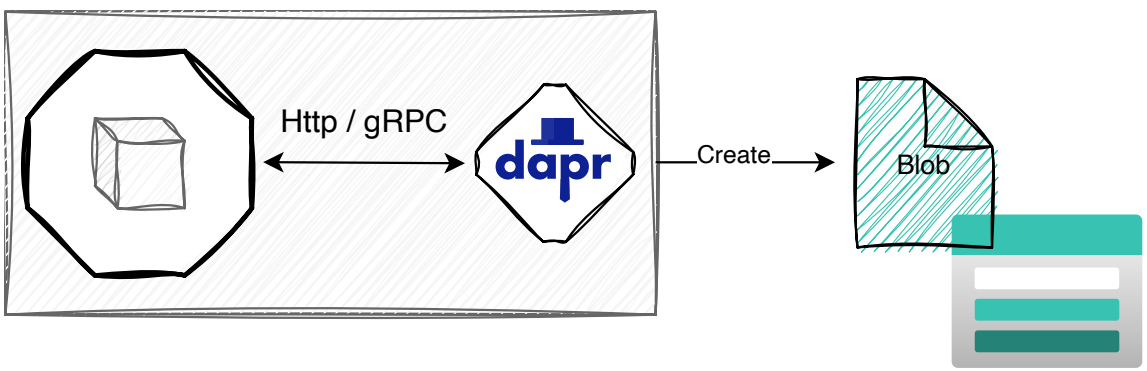


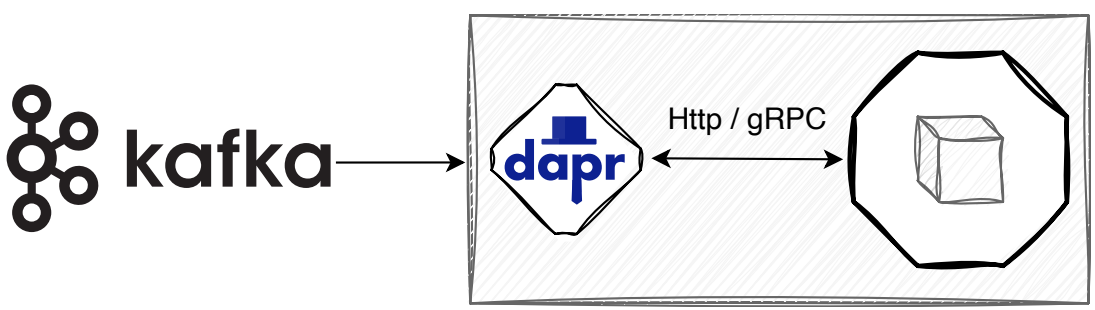
dapr publish/subscribe

https://github.com/MattSheehanDev/SimplifyingMicroservicesWithDapr/tree/main/03.publish-subscribe

dapr bindings

- Outbound binding - invoke external resource
- Input binding - trigger based on an event
- Can depend on the runtime instead of a library





binding resiliency

```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: resiliencypolicy
version: v1alpha1
spec:
  policies:
    timeouts:
      general: 5s
    retries:
      important:
        policy: constant
        duration: 5s
        maxRetries: 30
    circuitBreakers:
      bindingCB:
        maxRequests: 1
        interval: 8s
        timeout: 45s
        trip: consecutiveFailures > 8
targets:
    components:
      blobStorageBinding:
        outbound:
          timeout: general
          retry: important
          circuitBreaker: bindingCB
        inbound:
          timeout: general
          retry: important
```

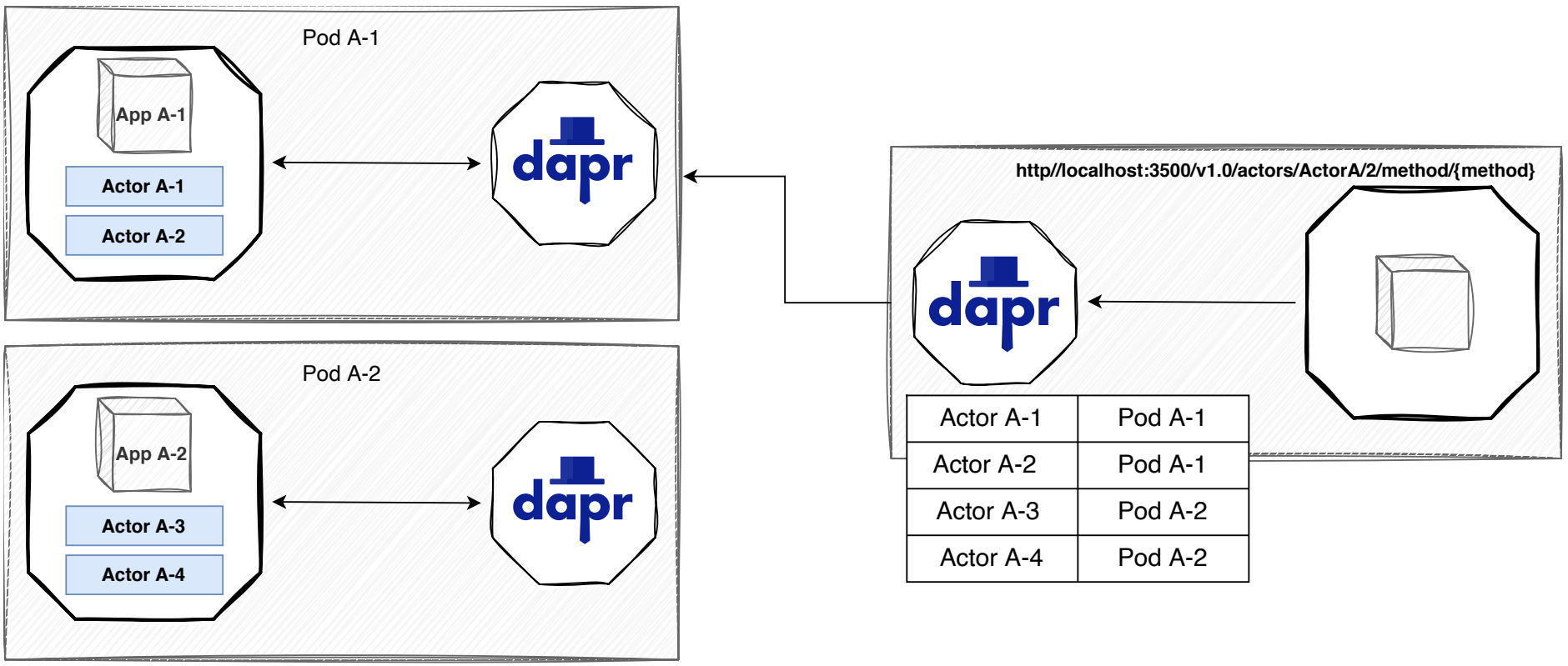
dapr bindings

https://github.com/MattSheehanDev/SimplifyingMicroservicesWithDapr/tree/main/04.bindings

dapr actors

- Manage concurrency
  - Turn-based concurrency access **per-actor**
- External state store
  - Require state store that supports transactions and etags
- Managed lifetimes
  - Timers & Reminders
- Placement and routing
  - Dapr placement service broadcasts current actors to all sidecars

dapr actors


Pod A-1
App A-1
Actor A-1
Actor A-2
dapr
Pod A-2
App A-2
Actor A-3
Actor A-4
dapr
http//localhost:3500/v1.0/actors/ActorA/2/method/{method}
dapr
Actor A-1 Pod A-1
Actor A-2 Pod A-1
Actor A-3 Pod A-2
Actor A-4 Pod A-2

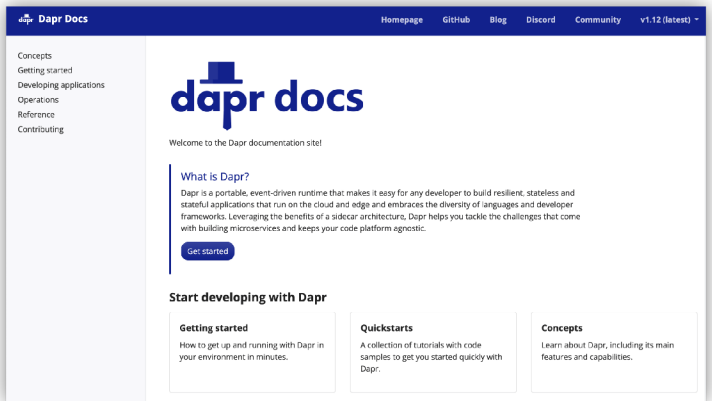
actor resiliency

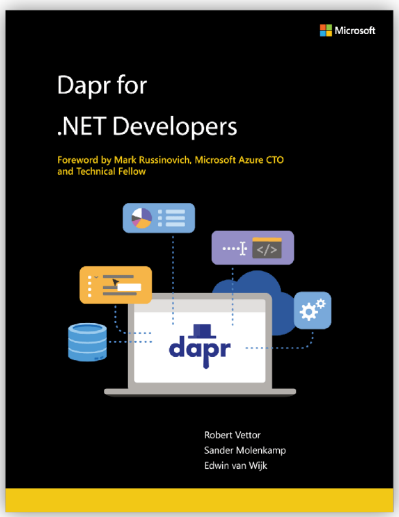
```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: resiliencypolicy
version: v1alpha1
scopes:
  - appB
spec:
  policies:
    timeouts:
      general: 5s
    retries:
      retry10:
        policy: exponential
        maxInterval: 15s
        maxRetries: 10
    circuitBreakers:
      serviceCB:
        maxRequests: 1
        timeout: 30s
        trip: consecutiveFailures >= 5
targets:
    actors:
      actorA:
        timeout: general
        retry: retry10
        circuitBreaker: serviceCB
```

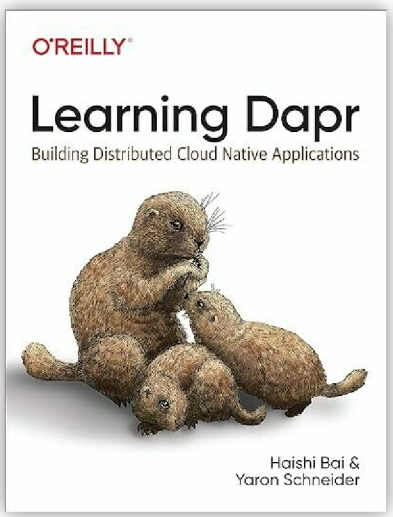
dapr actors

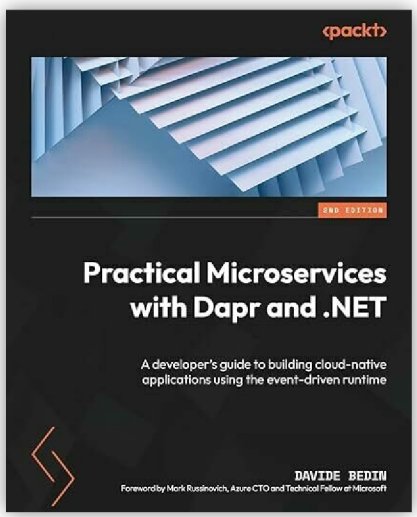
https://github.com/MattSheehanDev/SimplifyingMicroservicesWithDapr/tree/main/04.actors

Further Reading









https://github.com/dapr/samples

https://github.com/dapr/quickstarts