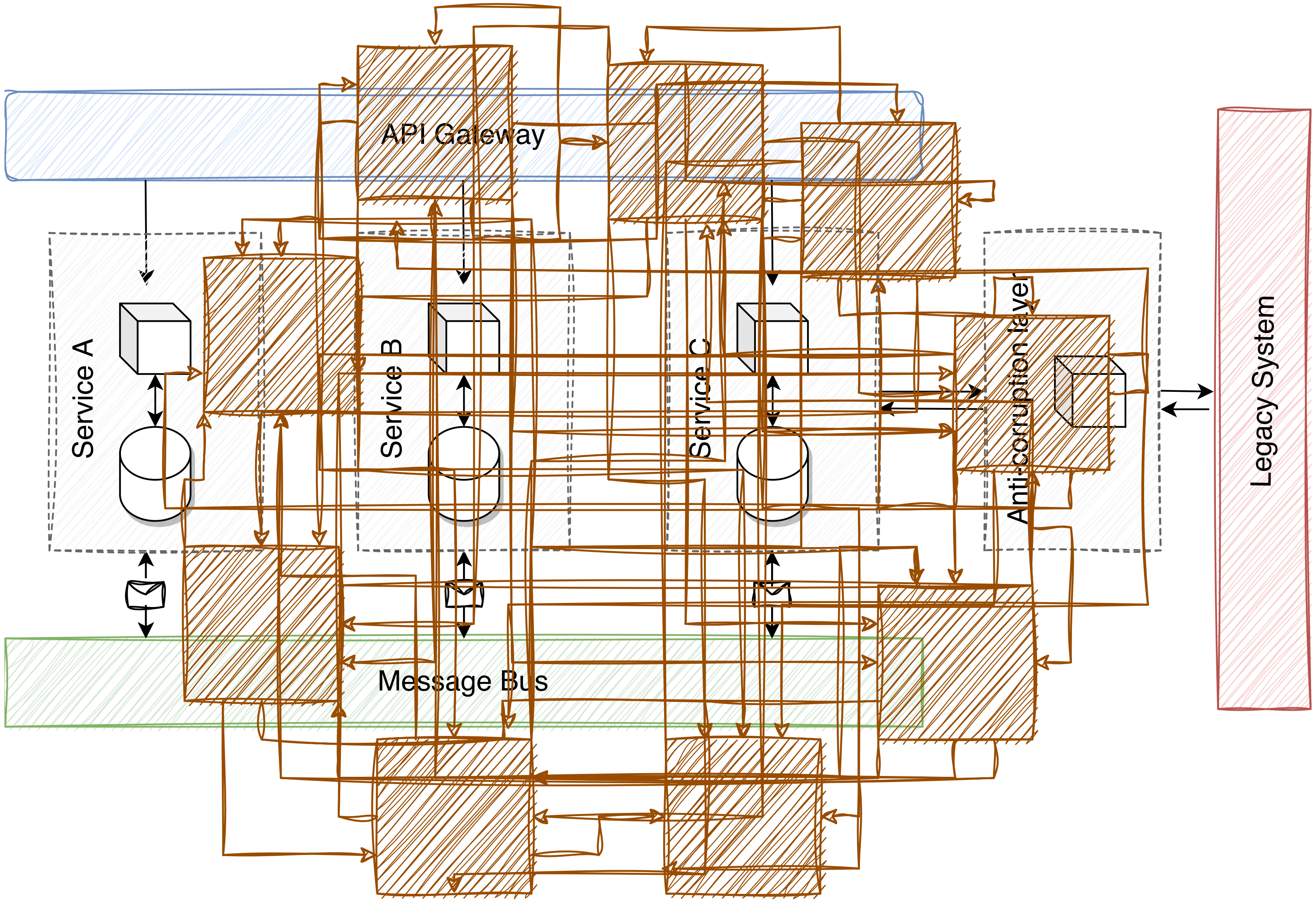
Simplifying Microservices with Dapr

matt sheehan

Distributed ~~Architecture~~ Ball of Mud



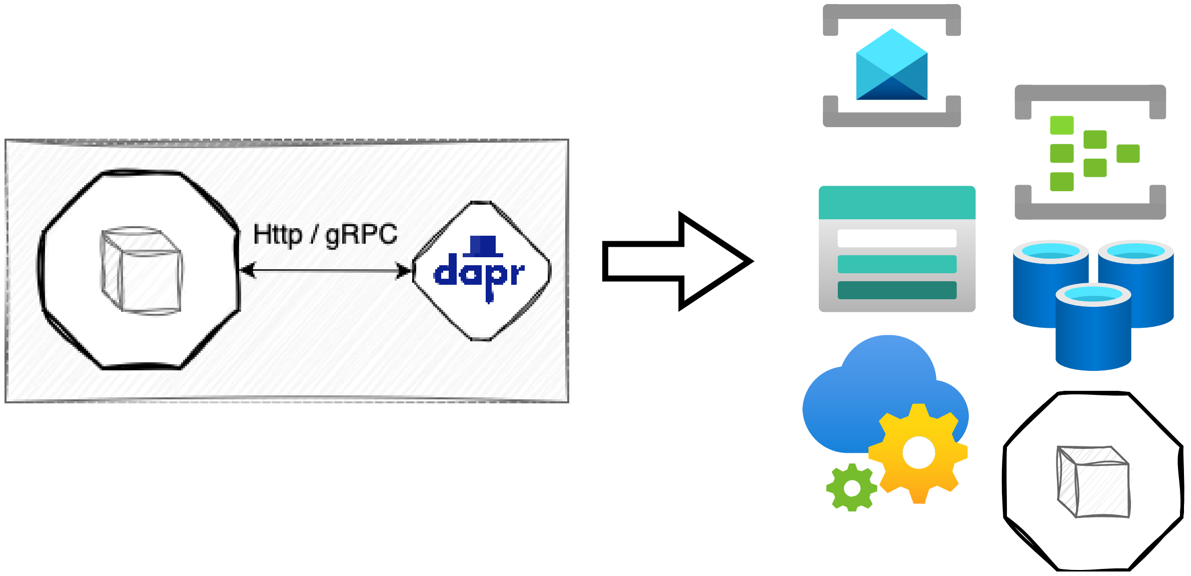
Microservice Challenges

- Reducing coupling to the platform and the backing services
- Removing vendor lock-in
- Discovering and calling other services
- Handling failures

dapr to the rescue!

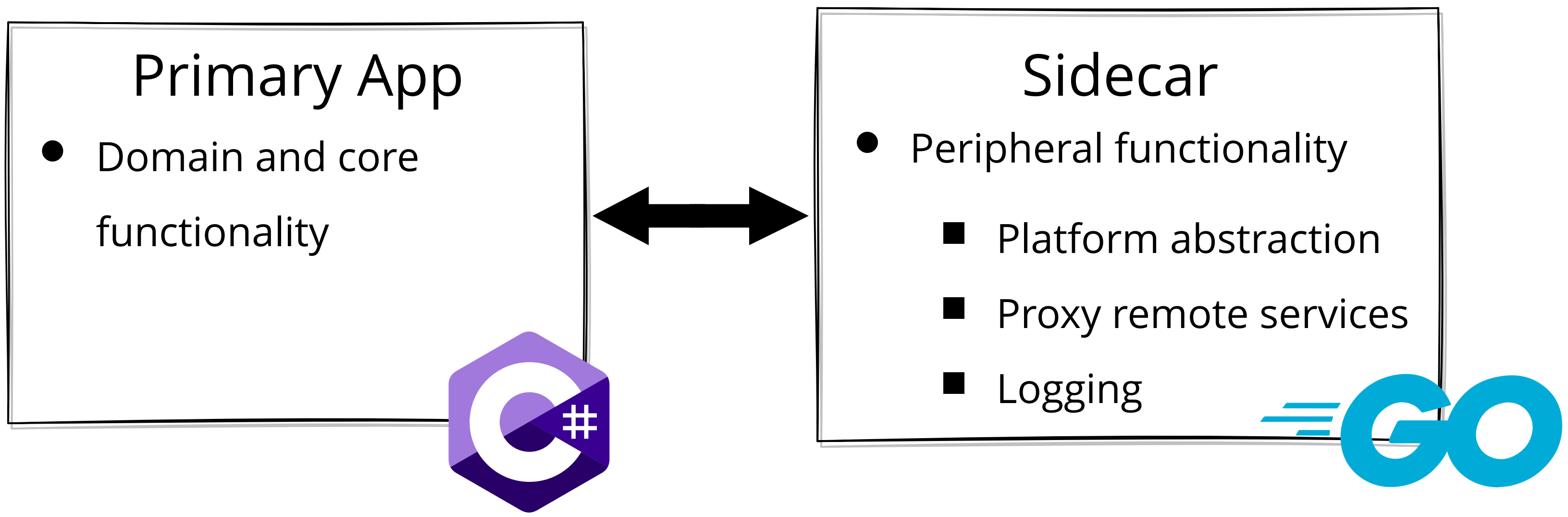
Introducing dapr

- **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
- Codifies best patterns and practices



Ambassador Pattern

Sidecar Pattern



Advantages

- Provides a homogenous interface for platform services
- Simplifies the plumbing and glue
- Independent from primary application runtime environment and language

dapr benefits

Provides flexibility and simplicity

Decreases your API surface area

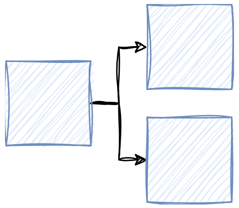
Language neutral

Provides a standard and declarative configuration

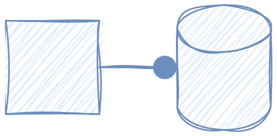
Codifies best patterns and practices

dapr building blocks

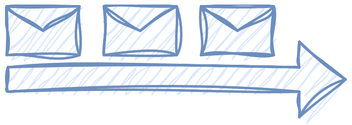
Service-to-Service Invocation



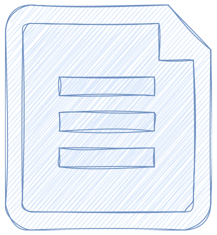
State Management



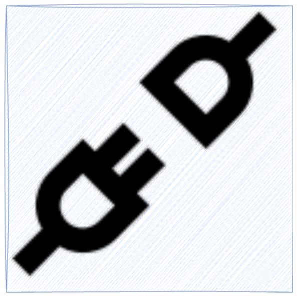
Publish and Subscribe



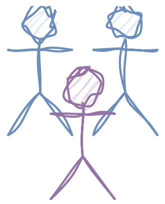
Configuration & Secrets



Input and Output Bindings



Actors



dapr configuration

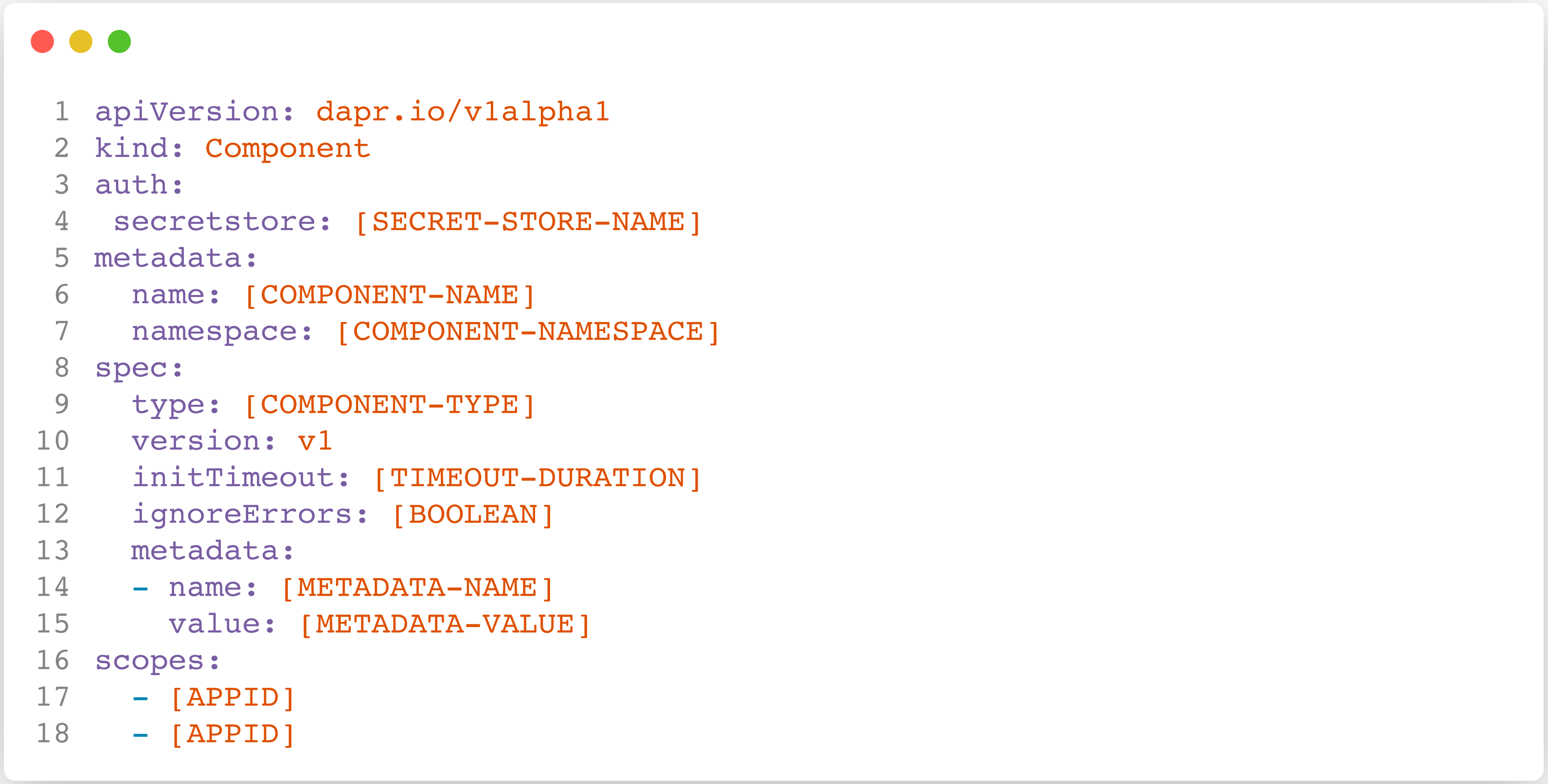
- An implementation of a **Building Block** is a *Component*
- **Components** are configured with *Component Specs*

```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 configuration

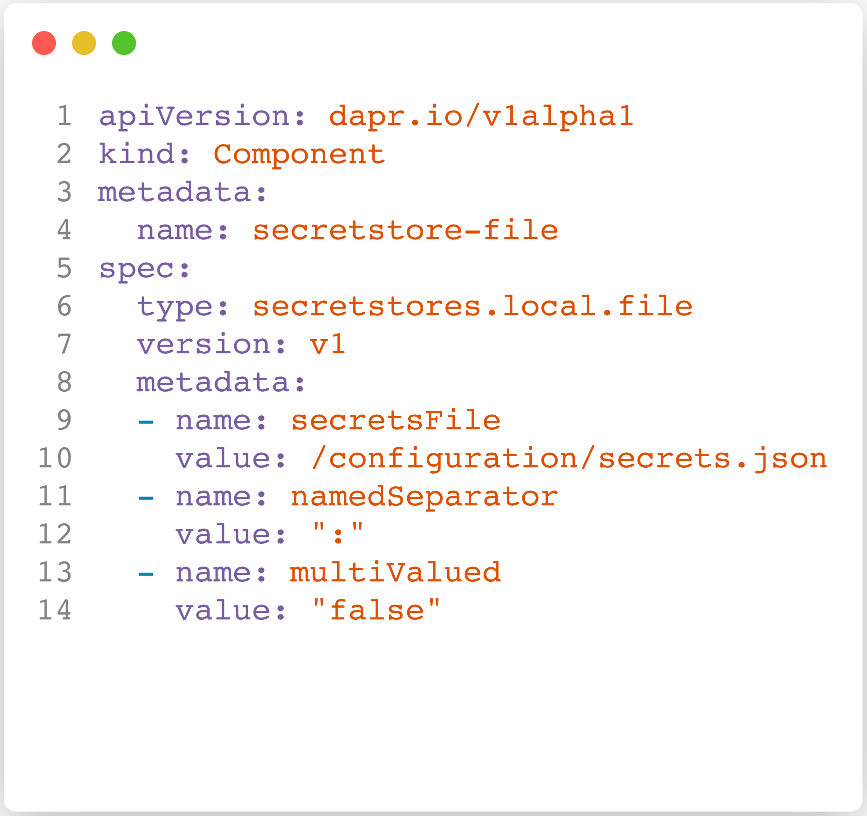


```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
auth:
 secretstore: [SECRET-STORE-NAME]
metadata:
  name: [COMPONENT-NAME]
  namespace: [COMPONENT-NAMESPACE]
spec:
  type: [COMPONENT-TYPE]
  version: v1
  initTimeout: [TIMEOUT-DURATION]
  ignoreErrors: [BOOLEAN]
  metadata:
  - name: [METADATA-NAME]
    value: [METADATA-VALUE]
scopes:
  - [APPID]
  - [APPID]
```

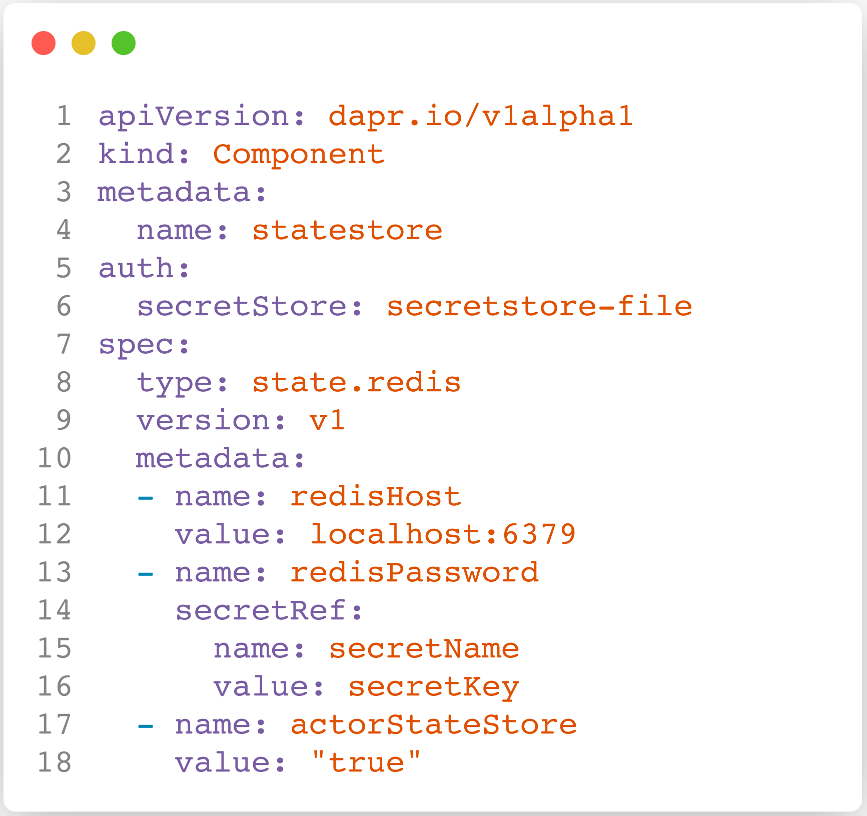
```
$HOME/.dapr/components
```

```
dapr run --app-id "<app-id>" --app-port "5005" --dapr-http-port "5050" --resources-path "./components" -- dotnet run ...
```

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"
```

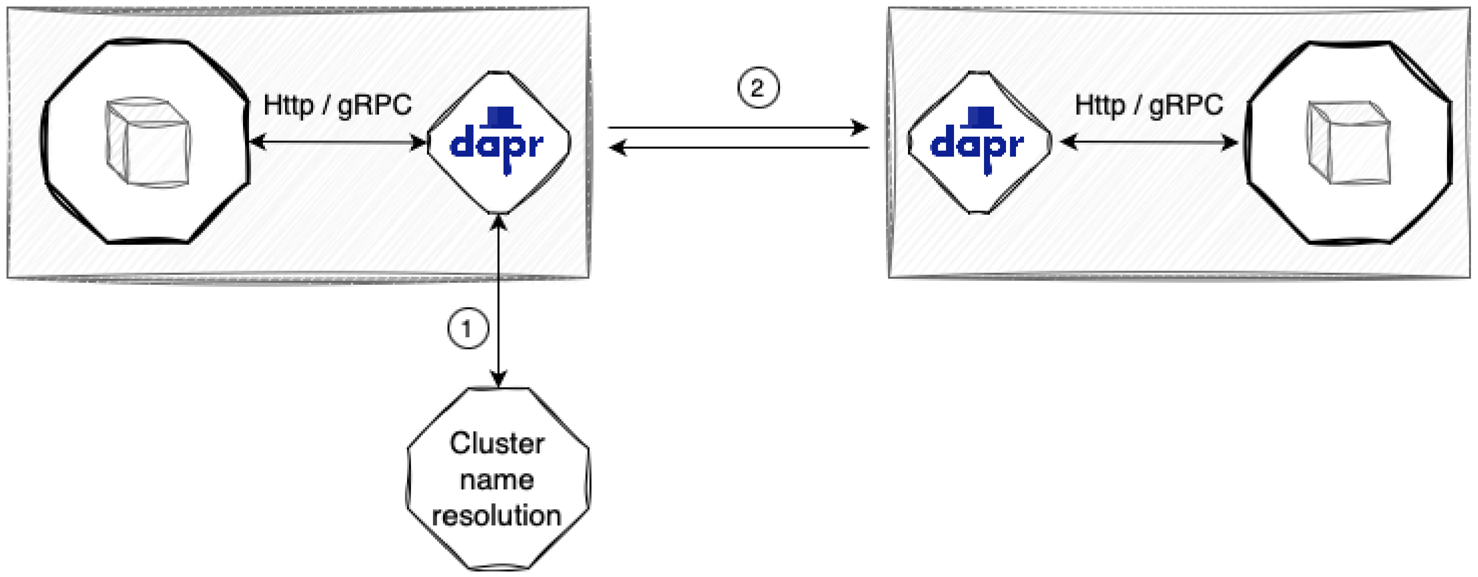


```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: statestore
auth:
  secretStore: secretstore-file
spec:
  type: state.redis
  version: v1
  metadata:
  - name: redisHost
    value: localhost:6379
  - name: redisPassword
    secretRef:
      name: secretName
      value: secretKey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

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

dapr service invocation

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

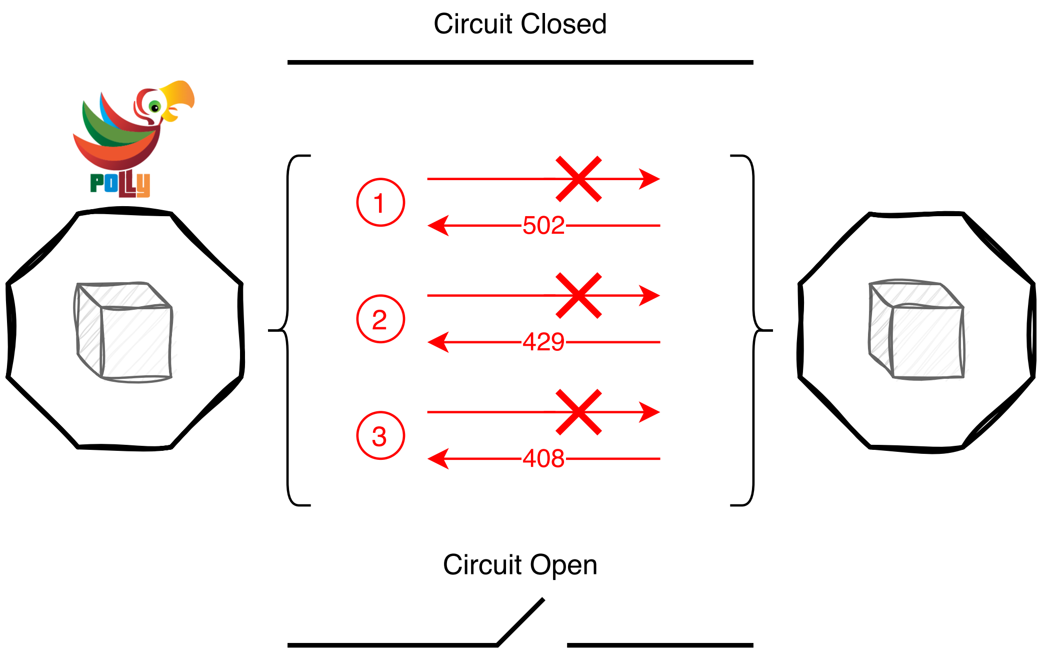


- Service discovery
- HTTP or gRPC
- Supports retries and circuit breaker patterns

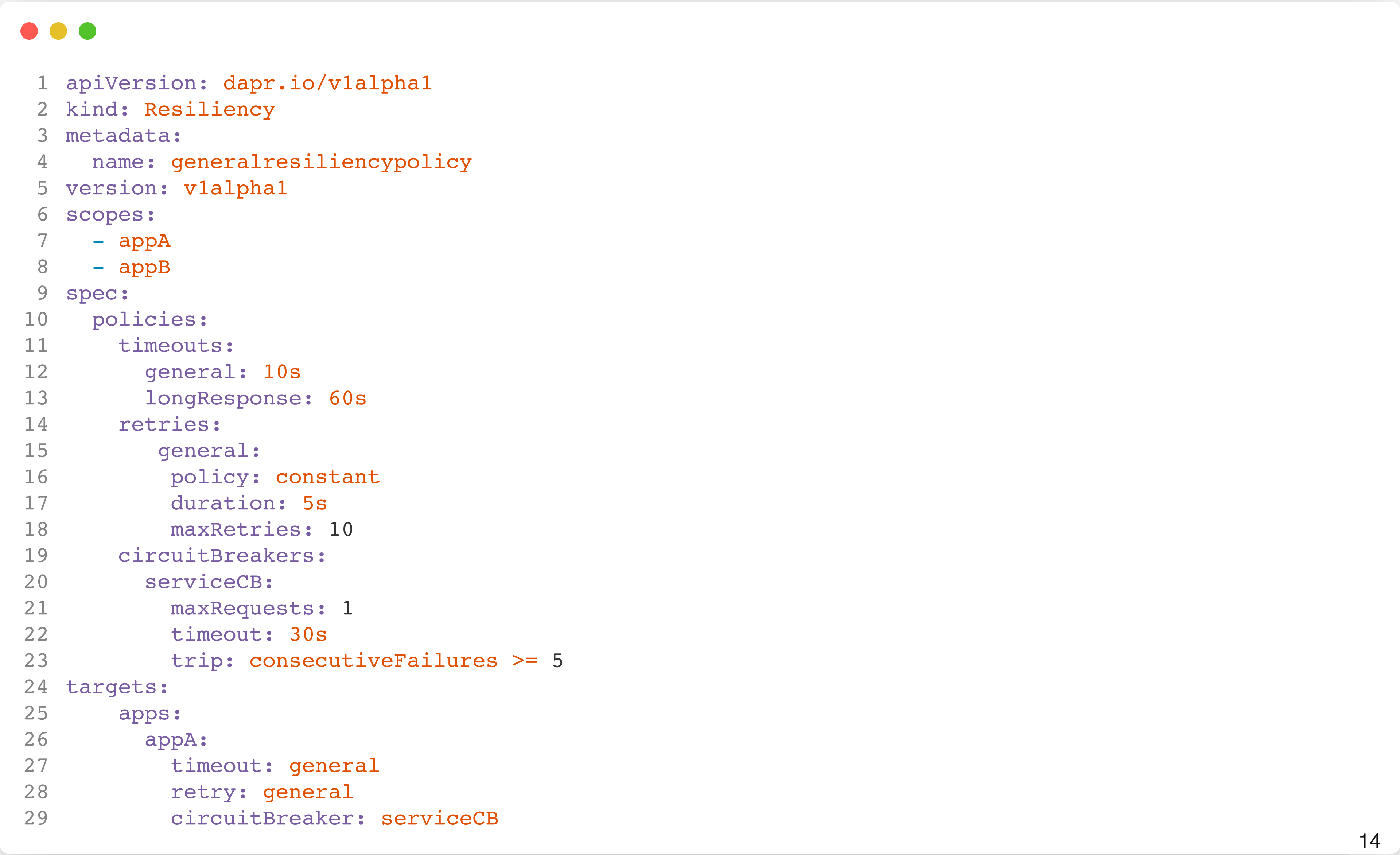
CLOUD NATIVE COMPUTING FOUNDATION

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

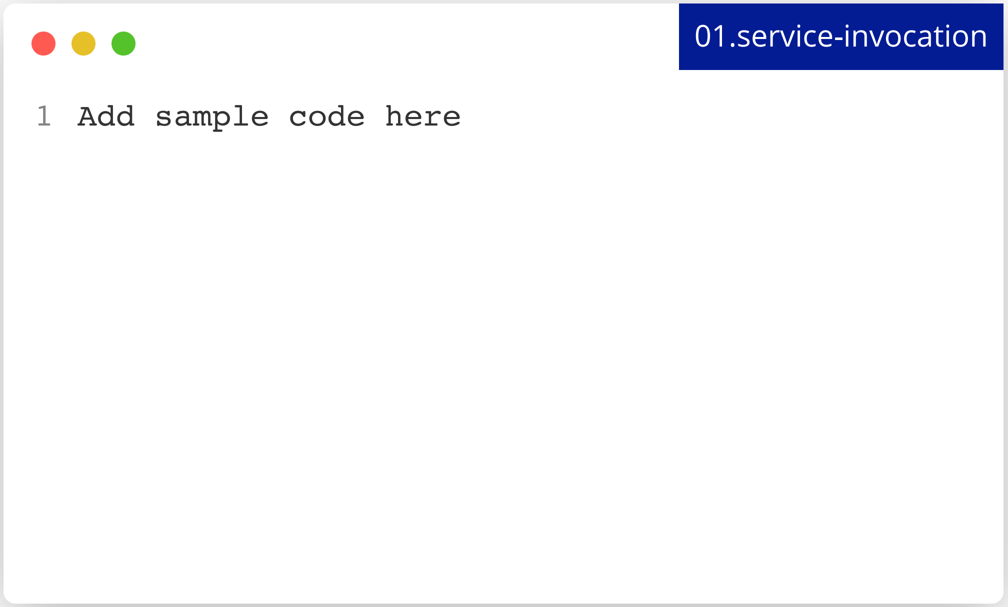


service invocation resiliency



```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
```

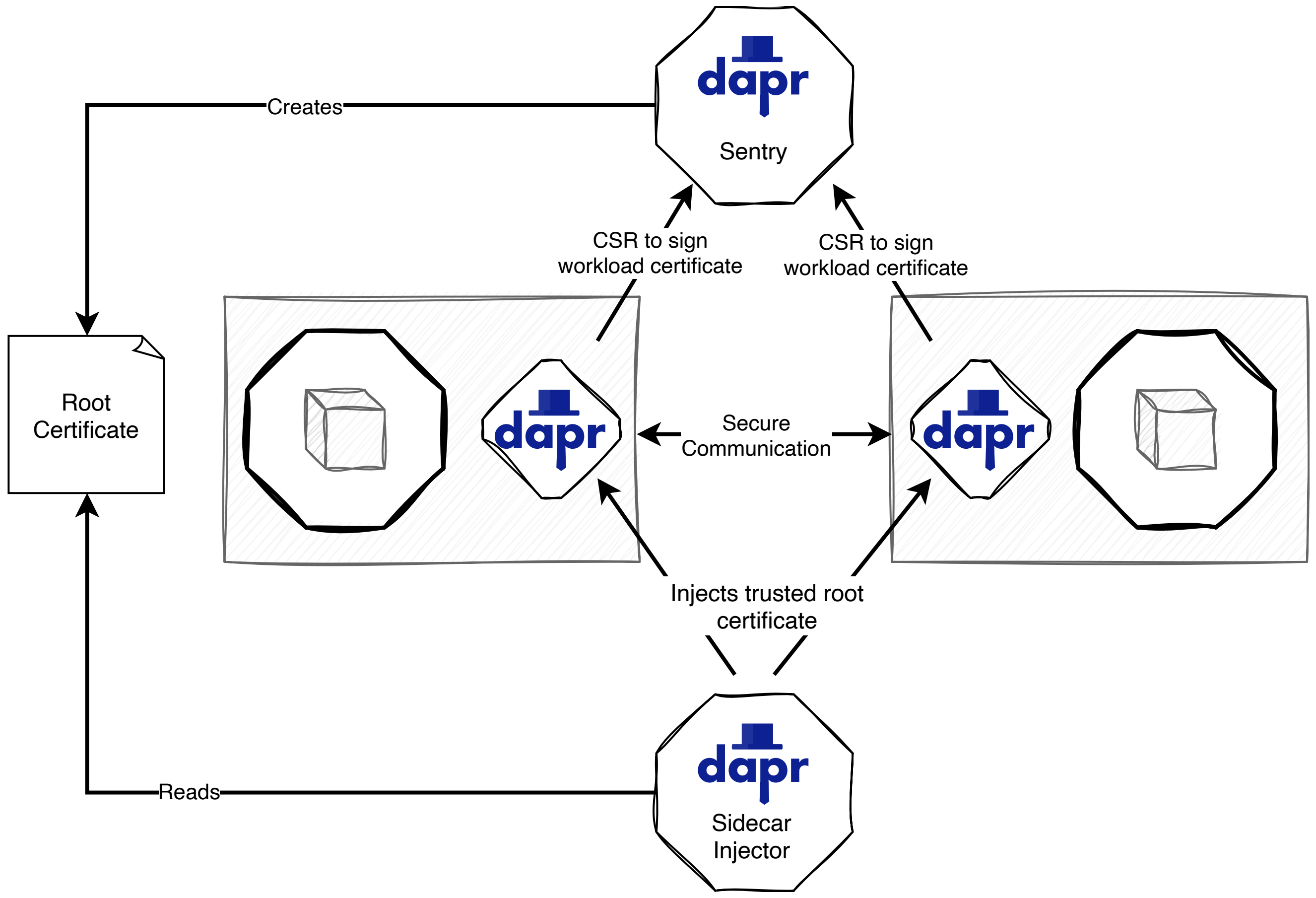
dapr service invocation


01.service-invocation

```
Add sample code here
```

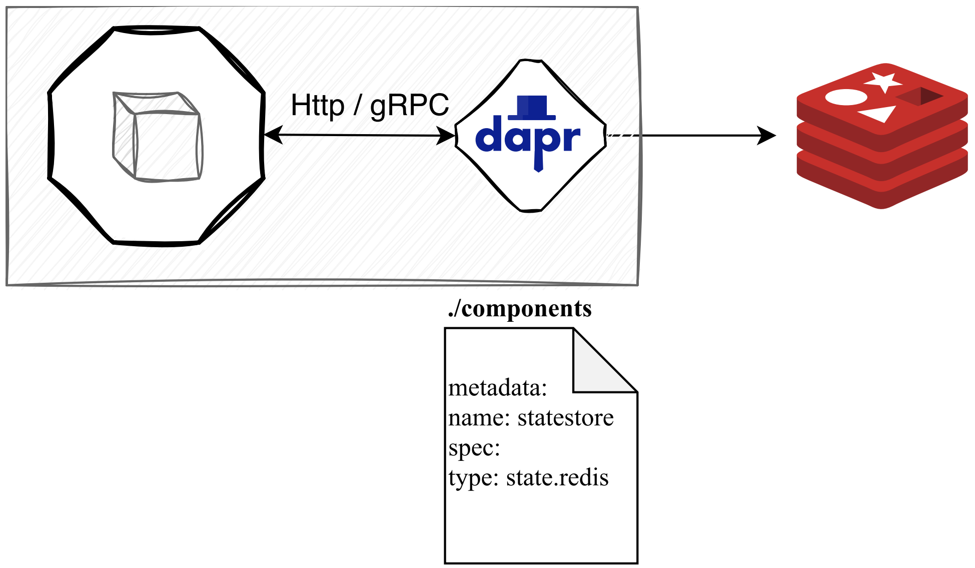
dapr mTLS

- 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
Sentry
Creates
CSR to sign workload certificate
CSR to sign workload certificate
Root Certificate
dapr
Secure Communication
dapr
Injects trusted root certificate
dapr
Sidecar Injector
Reads

dapr state management

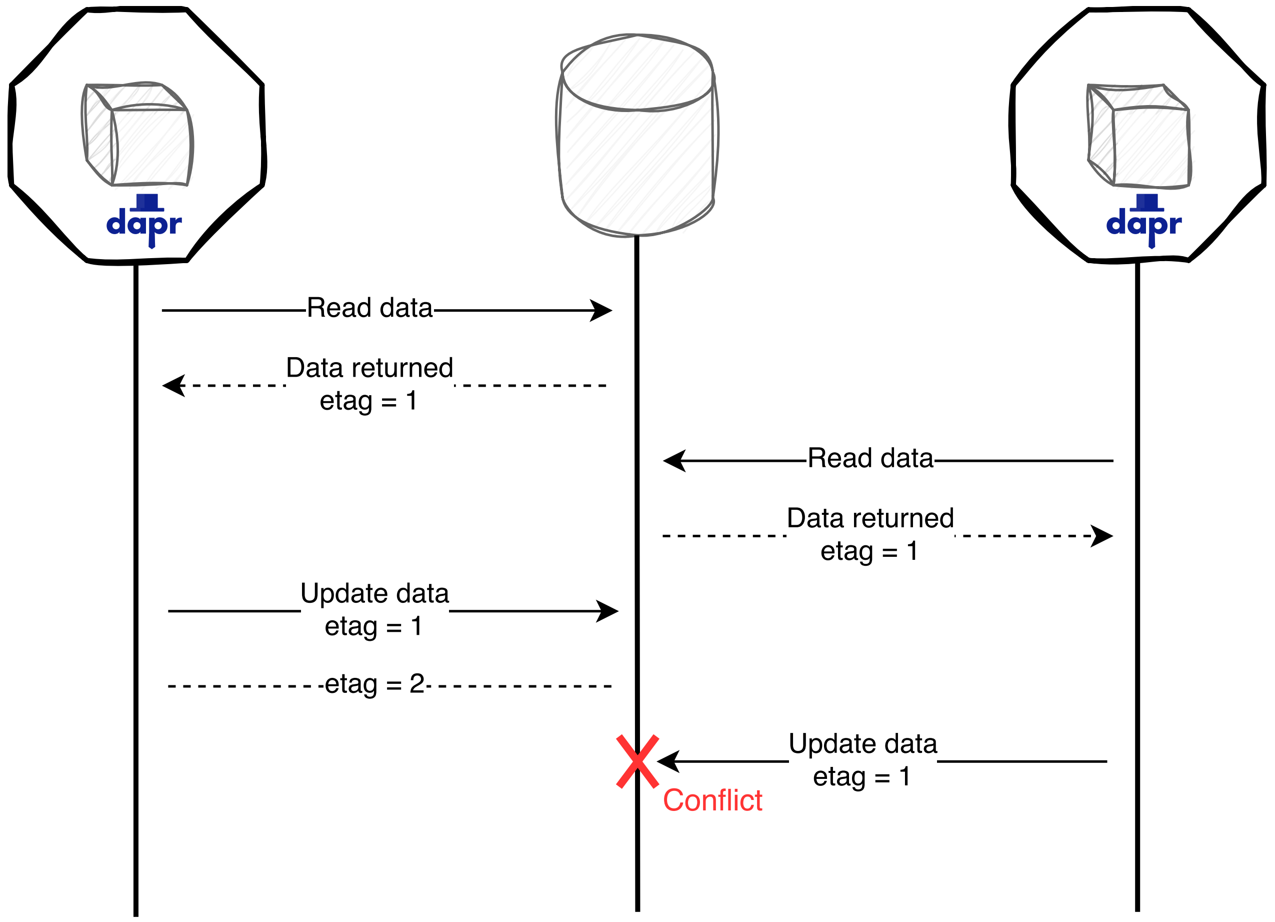
http://localhost:3500/v1.0/state/<storename>/<key>



```
metadata:
name: statestore
spec:
type: state.redis
```

- Consistency
  - Strong vs Eventual
- Concurrency
  - Last-write wins *Default*
  - First-write wins *Optimistic Concurrency Control*

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

02.state-management

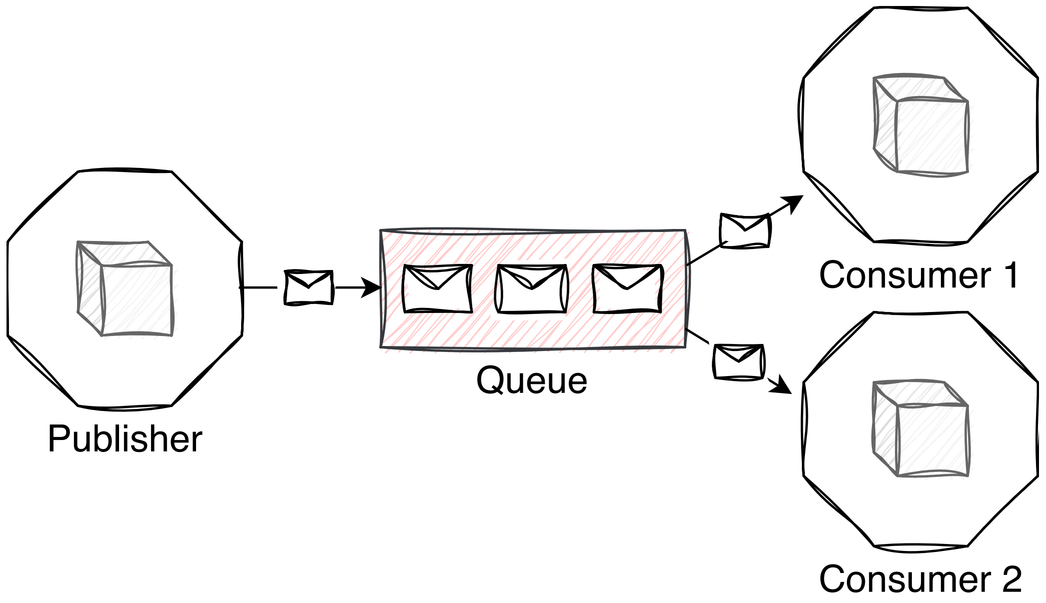
```
# name of state store component
const stateStore = "statestore"

app.MapGet("/getShoppingCart/{id}", async (
    [FromRoute] string id,
    [FromServices] DaprClient daprClient) =>
{
    return await daprClient.GetStateAsync(stateStore, id);
});

app.MapPost("/updateShoppingCart", async (
    Cart cart,
    [FromServices] DaprClient daprClient) =>
{
    await daprClient.SaveStateAsync(stateStore, cart.Id, cart);
});
```

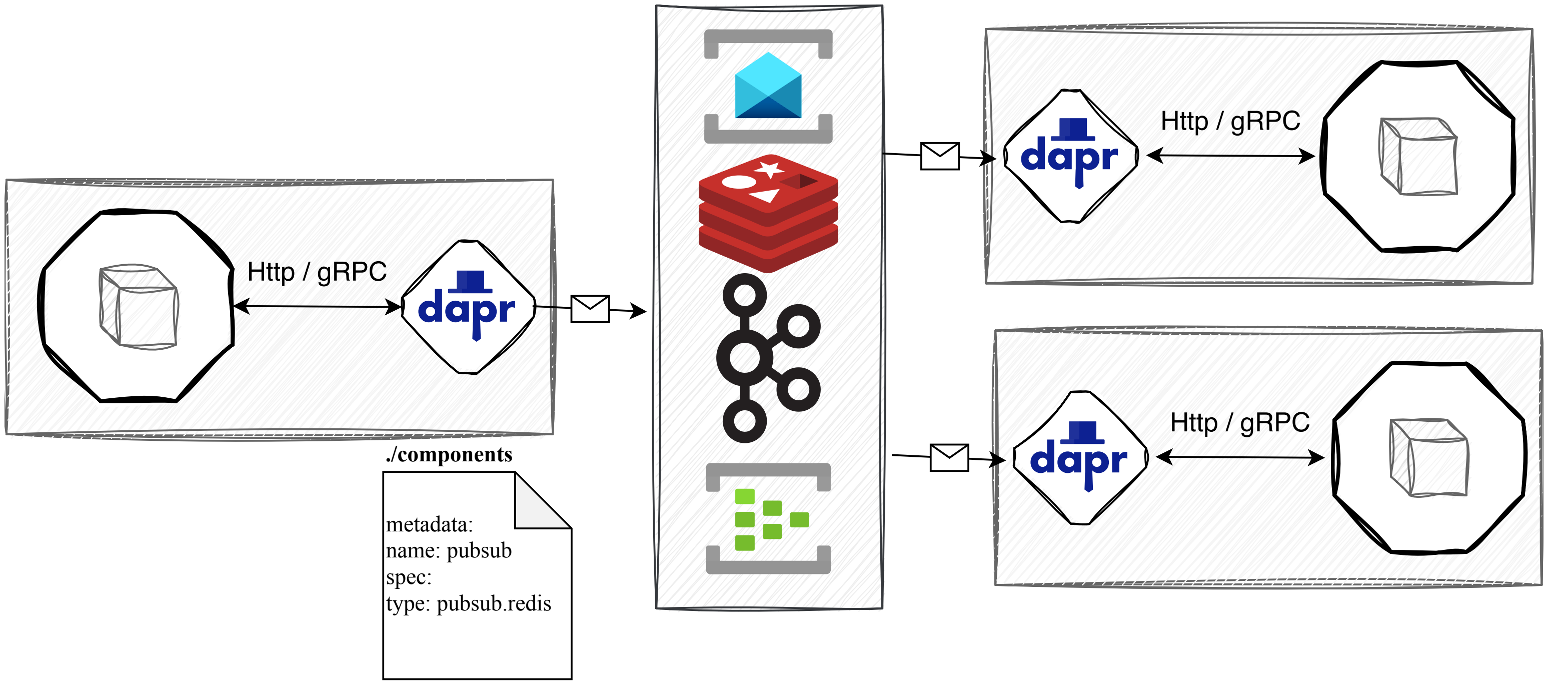
Publisher/Subscriber Pattern

- Decouples senders from consumers
- Improve responsiveness of the sender
- Improve reliability
- Allows for deferred processing



dapr publish/subscribe

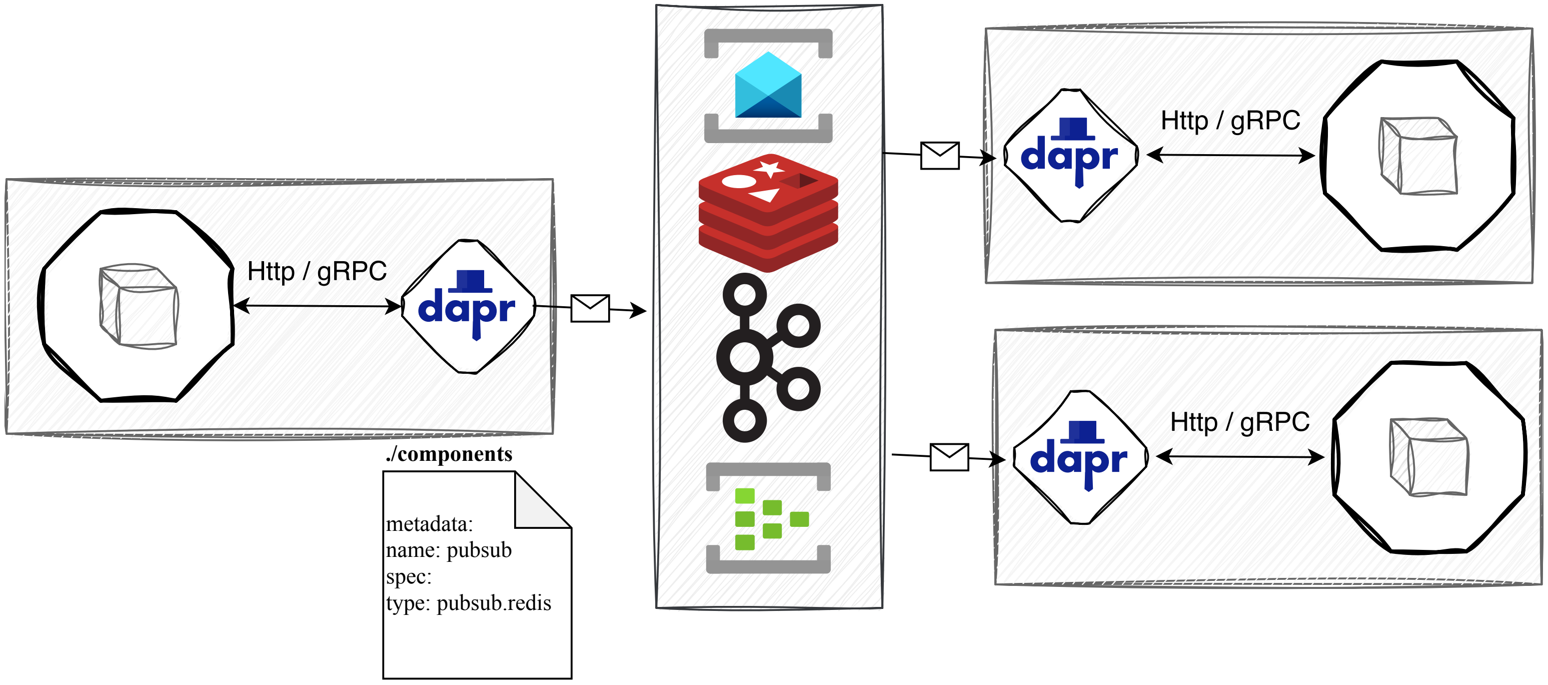
`http://localhost:3500/v1.0/publish/<pubsubname>/<topic>`



```
metadata:
name: pubsub
spec:
type: pubsub.redis
```

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

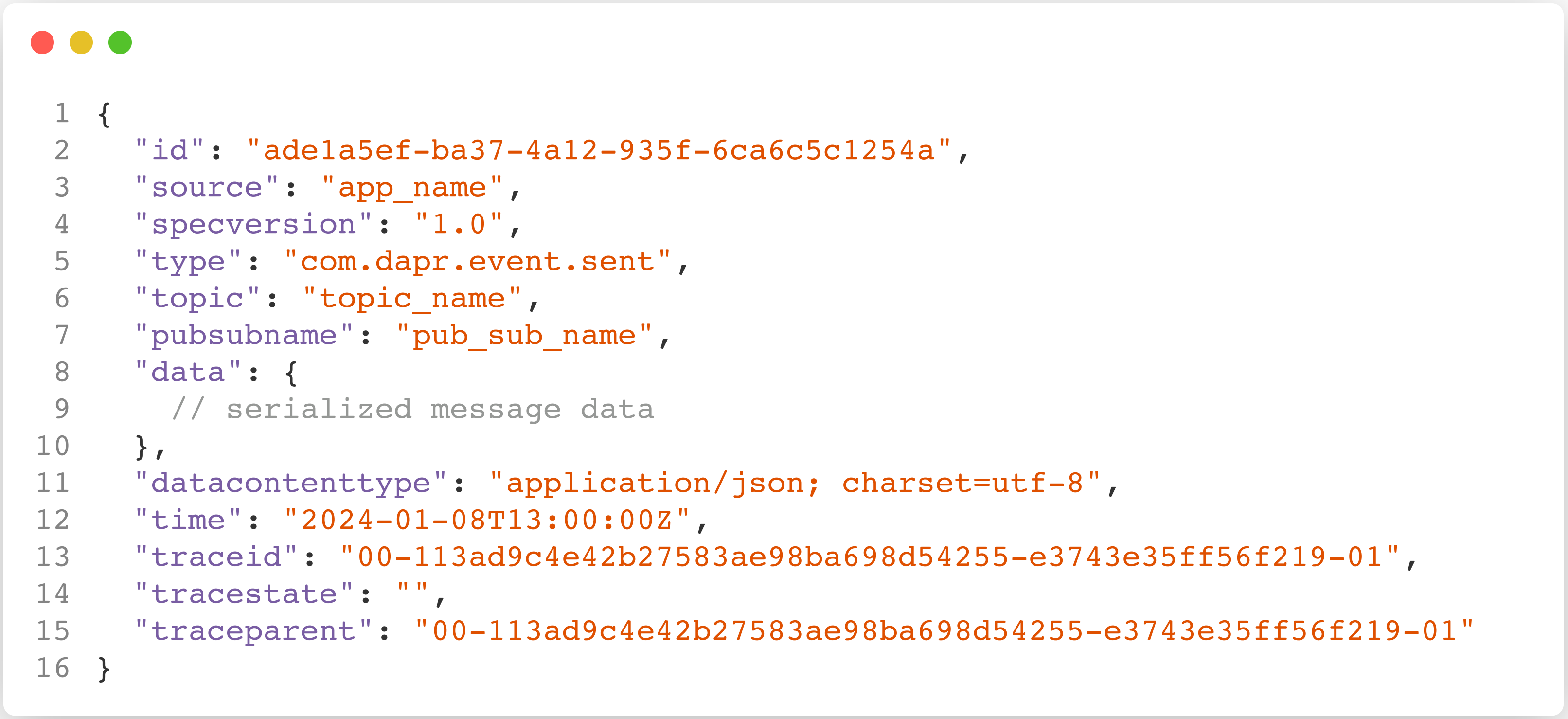
dapr publish/subscribe



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

Cloud Events

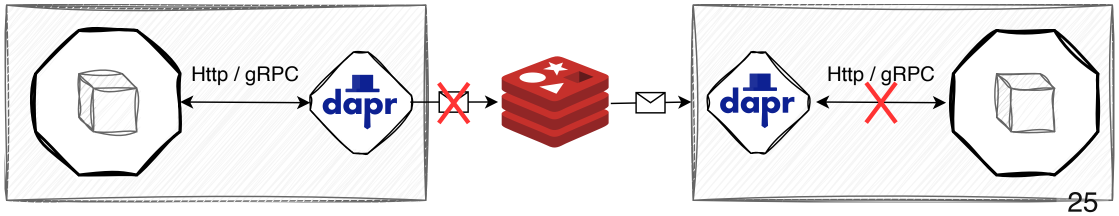
- Enables content-based routing
- Enables message de-duplication
- Improves interoperability



```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
```



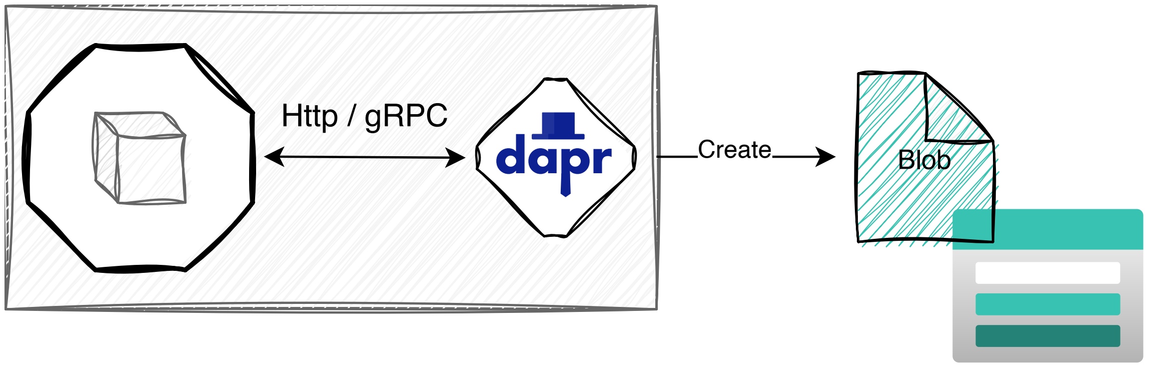
dapr publish/subscribe

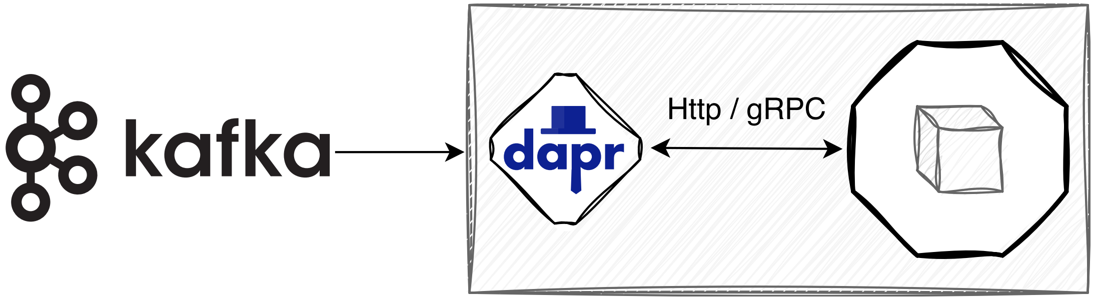
03.publish-subscribe

```
add code here
```

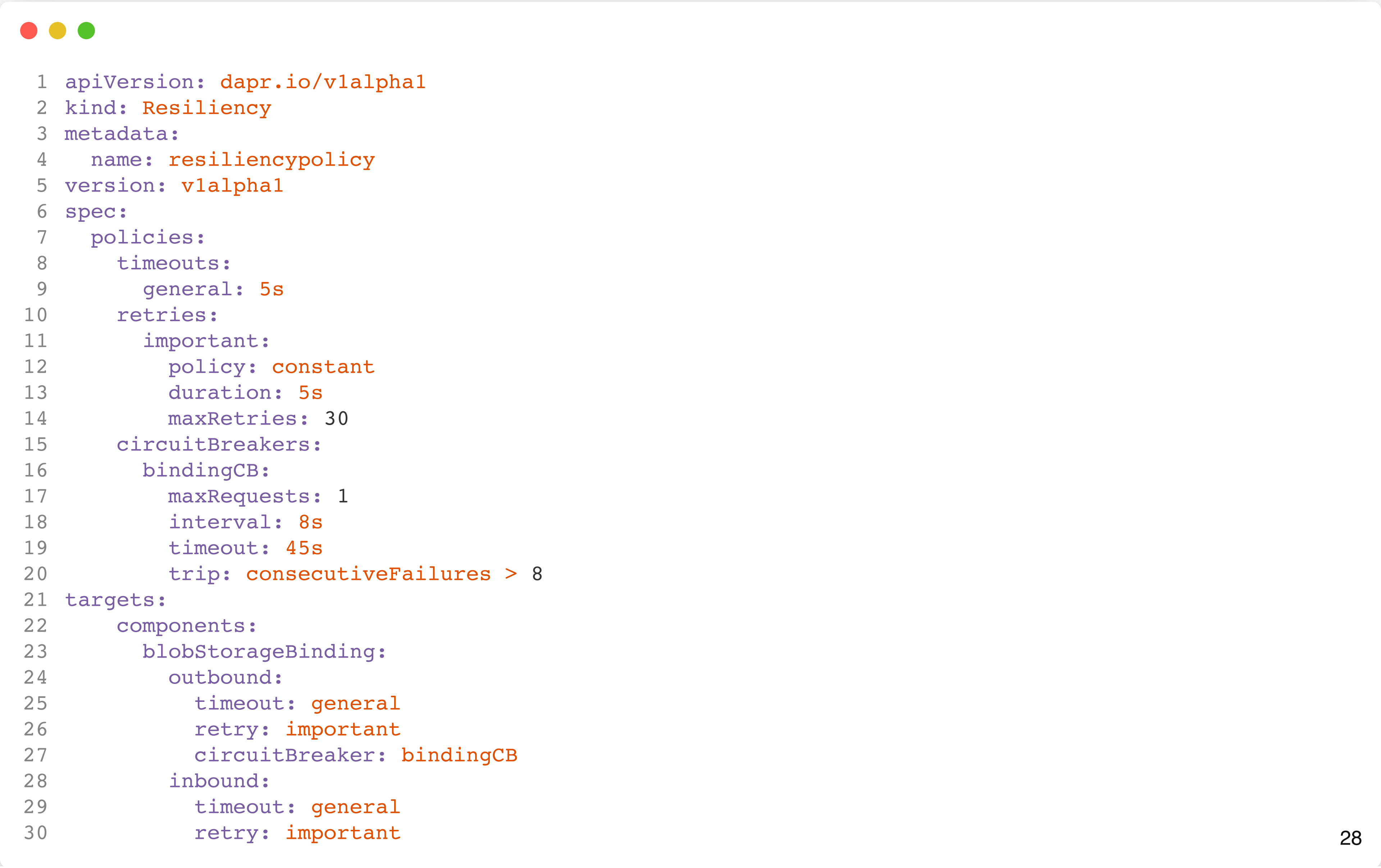
dapr bindings

- 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

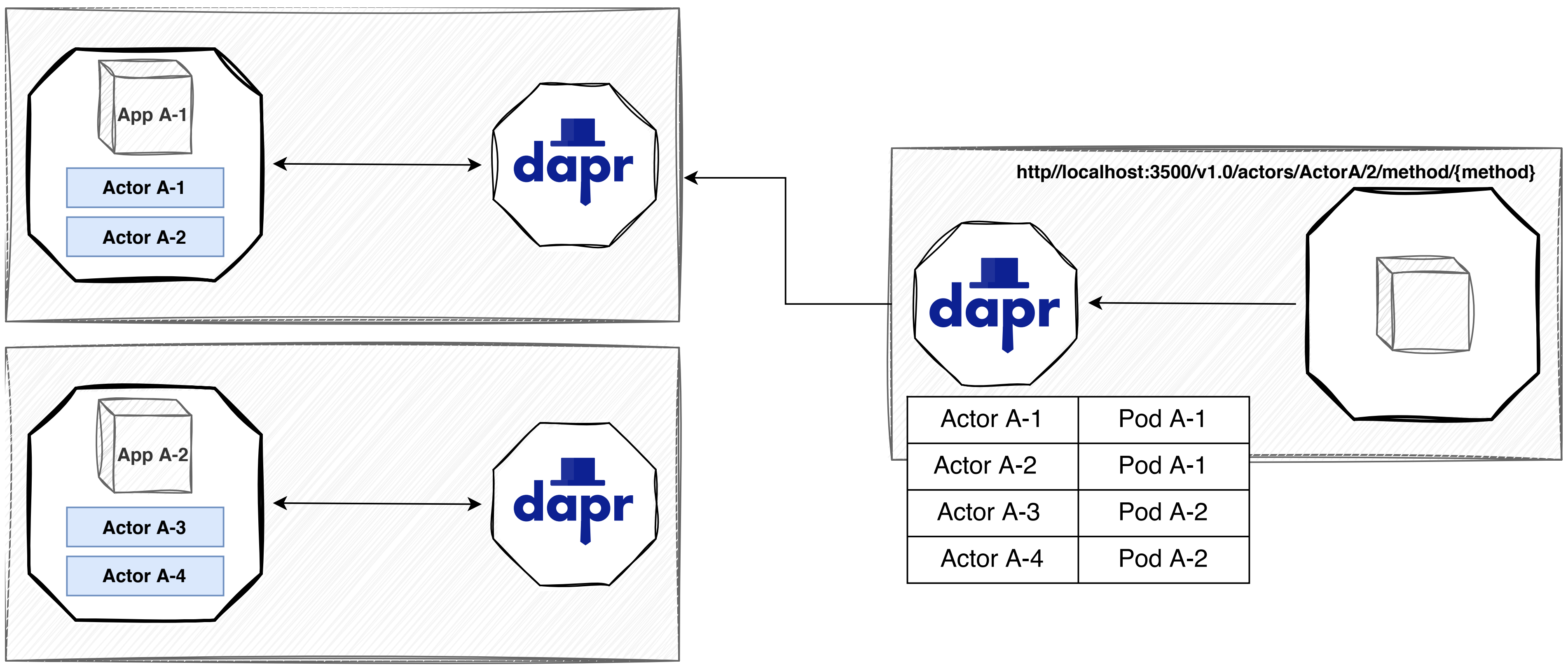
04.bindings

```
todo
```

dapr actors

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

dapr actors



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
```

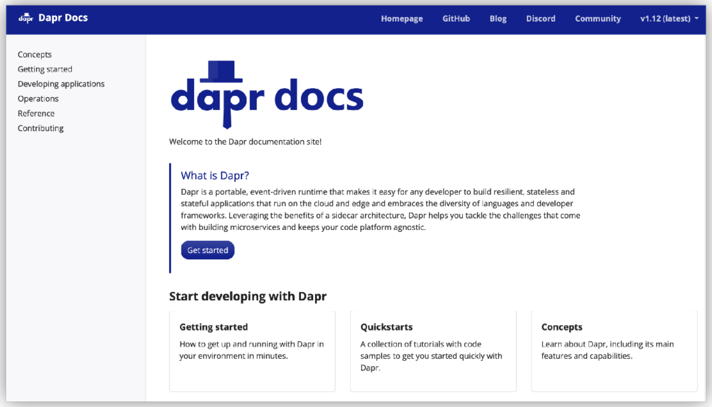
Hosting

Kubernetes
(GKE, AKS, EKS)

Virtual Machine

Docker container

Further Reading



Dapr Docs — Homepage · GitHub · Blog · Discord · Community · v1.12 (latest)

Concepts
Getting started
Developing applications
Operations
Reference
Contributing

dapr docs

Welcome to the Dapr documentation site!

What is Dapr?

Dapr is a portable, event-driven runtime that makes it easy for any developer to build resilient, stateless and stateful applications that run on the cloud and edge and embraces the diversity of languages and developer frameworks. Leveraging the benefits of a sidecar architecture, Dapr helps you tackle the challenges that come with building microservices and keeps your code platform agnostic.

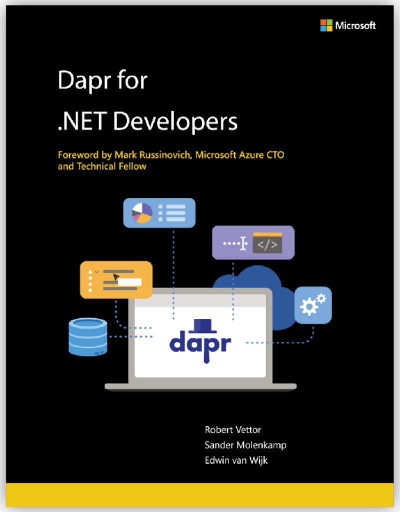
Get started

Start developing with Dapr

Getting started
How to get up and running with Dapr in your environment in minutes.

Quickstarts
A collection of tutorials with code samples to get you started quickly with Dapr.

Concepts
Learn about Dapr, including its main features and capabilities.

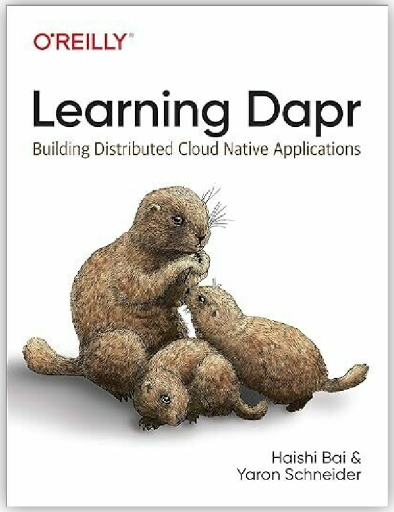


Microsoft

Dapr for .NET Developers

Foreword by Mark Russinovich, Microsoft Azure CTO and Technical Fellow

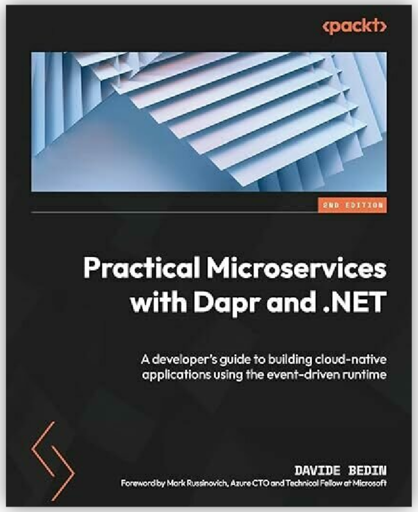
Robert Vettor
Sander Molenkamp
Edwin van Wijk



O'REILLY

Learning Dapr
Building Distributed Cloud Native Applications

Haishi Bai & Yaron Schneider



packt

2ND EDITION

Practical Microservices with Dapr and .NET

A developer's guide to building cloud-native applications using the event-driven runtime

DAVIDE BEDIN
Foreword by Mark Russinovich, Azure CTO and Technical Fellow at Microsoft

https://github.com/dapr/samples

https://github.com/dapr/quickstarts