Construindo aplicações Cloud Native com o Dapr

Walter Silvestre Coan

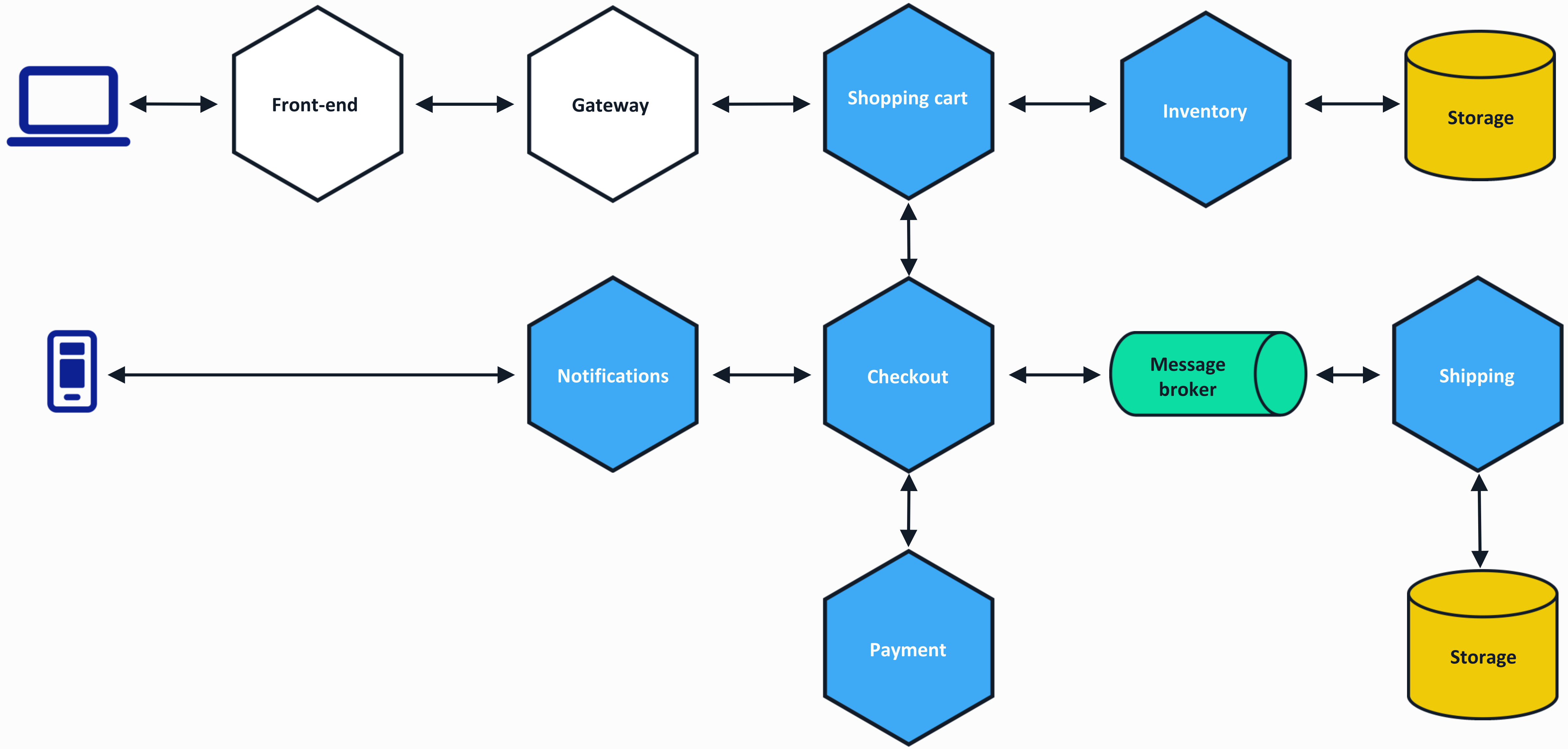
Walter Silvestre Coan

www.linkedin.com/in/waltercoan/

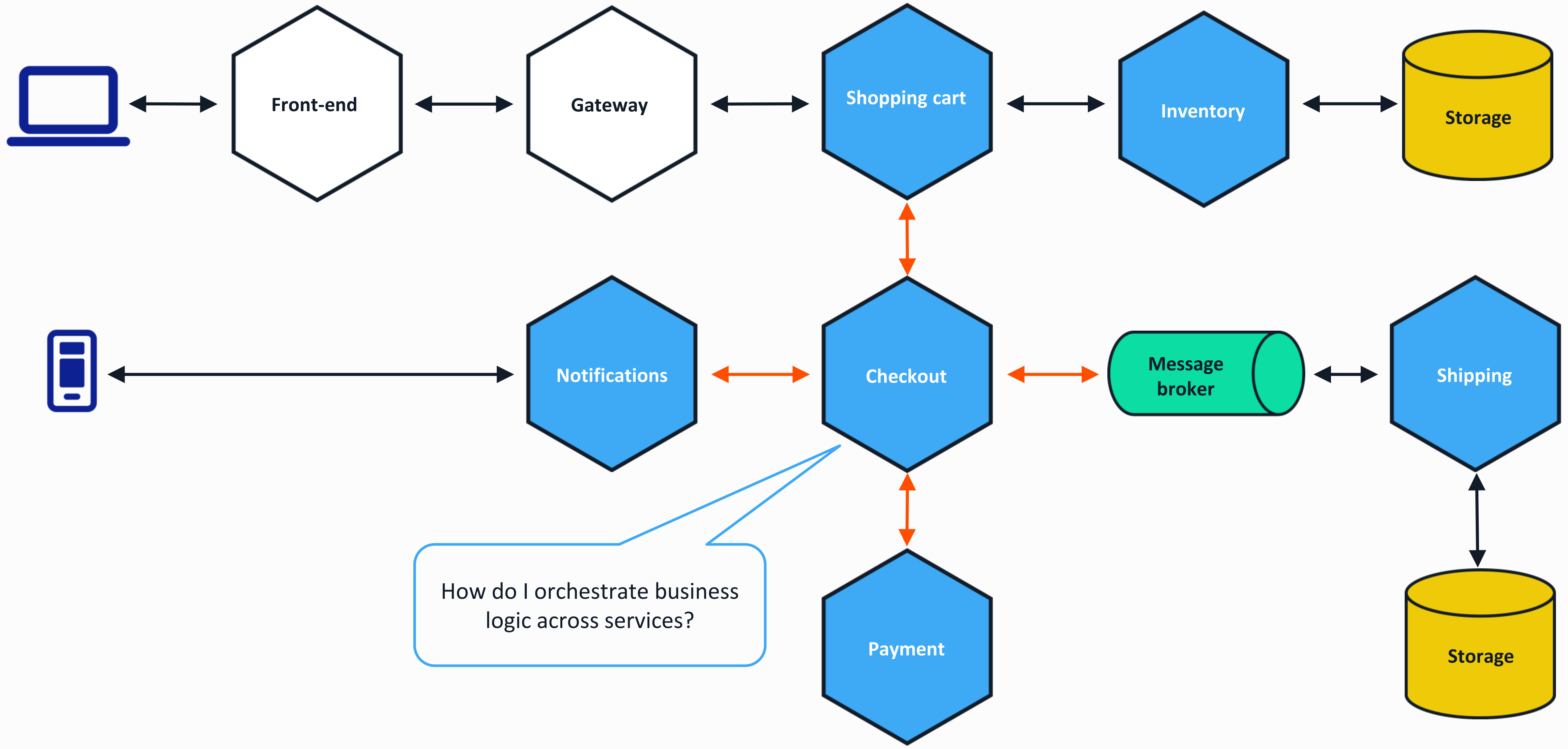
- Microsoft MVP na categoria Internet das Coisas
- Dapr Meteors 2025
- Mestre em Sistemas Distribuídos e Redes de sensores sem fio PUCPR
- Instrutor autorizado Microsoft, AWS, NVIDIA na Ka Solution
- Professor na UNIVILLE

Microsoft MVP 2024 Most Valuable Professional
Microsoft Certified Trainer MCT
Microsoft Certified Azure Solutions Architect Expert
Microsoft Certified Azure Security Engineer Associate
Microsoft Certified Azure Developer Associate
Microsoft Certified Azure Administrator Associate
Microsoft Certified Azure AI Engineer Associate
dapr community supporter
dapr presenter
Microsoft Certified Azure Cosmos DB Developer Specialty
Microsoft Certified Azure IoT Developer Specialty
Microsoft Certified Security, Compliance, and Identity Fundamentals
Microsoft Certified Azure Fundamentals
Microsoft Certified Azure Data Fundamentals
Microsoft Certified Azure AI Fundamentals
kubernetes and Cloud Native Associate
GitHub Copilot Certification Program
aws certified Cloud Practitioner Foundational
aws certified AI Practitioner Foundational
aws certified Developer Associate
aws certified Solutions Architect Associate
aws certified SysOps Administrator Associate
aws certified Solutions Architect Professional
aws Authorized Instructor Champion
aws Academy Educator
2025 dapr meteor

Distributed applications

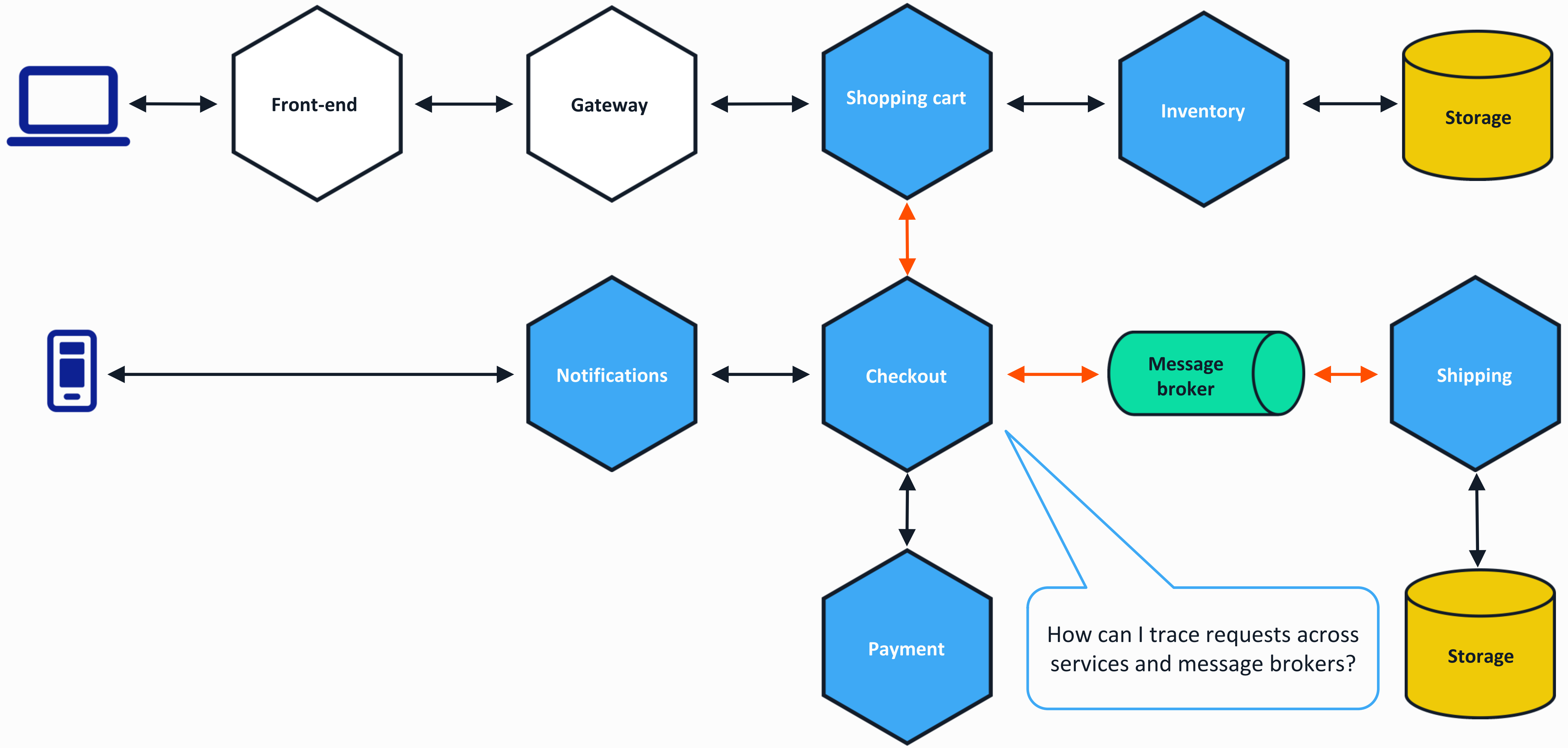


Developer challenges – Service orchestration



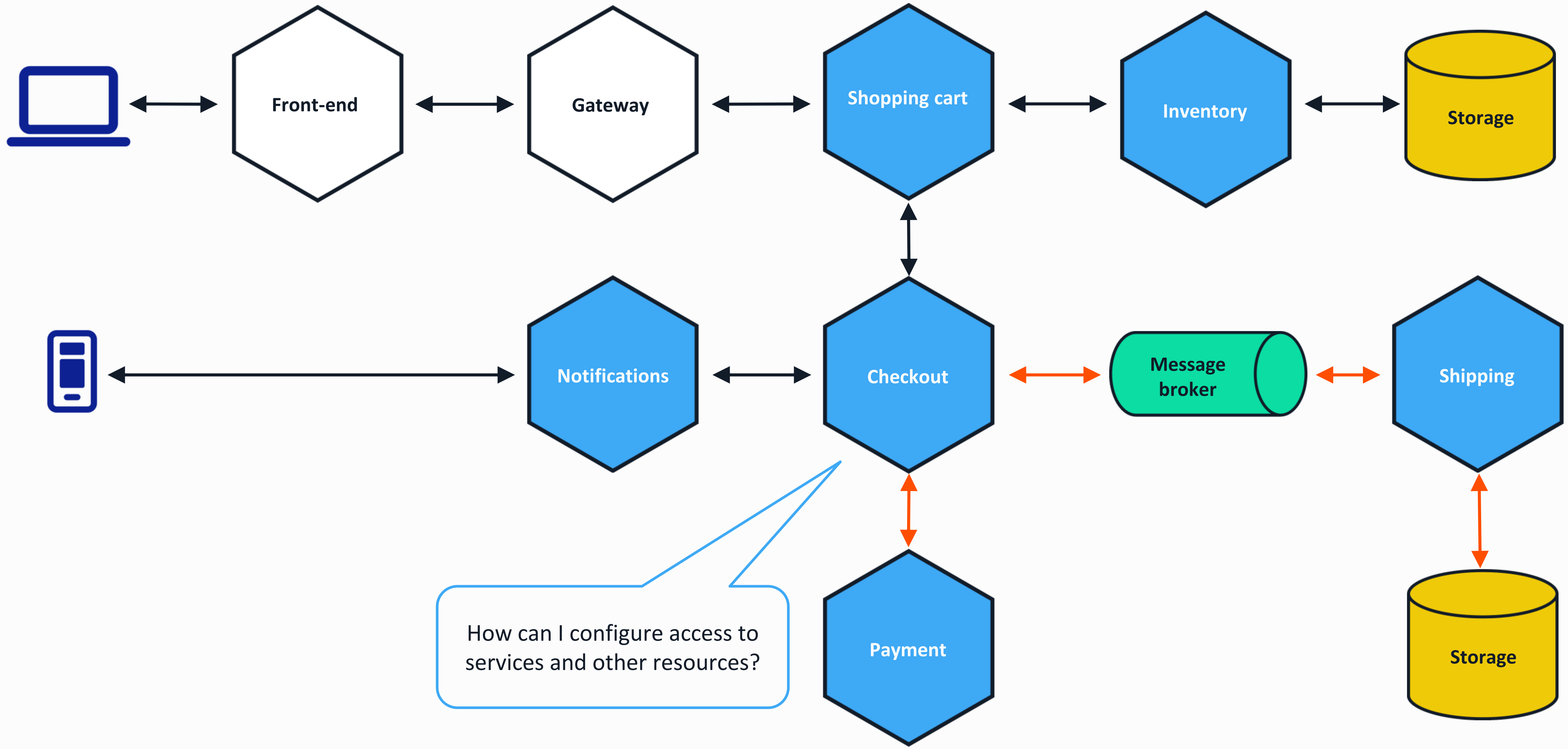
How do I orchestrate business logic across services?

Developer challenges – Distributed tracing



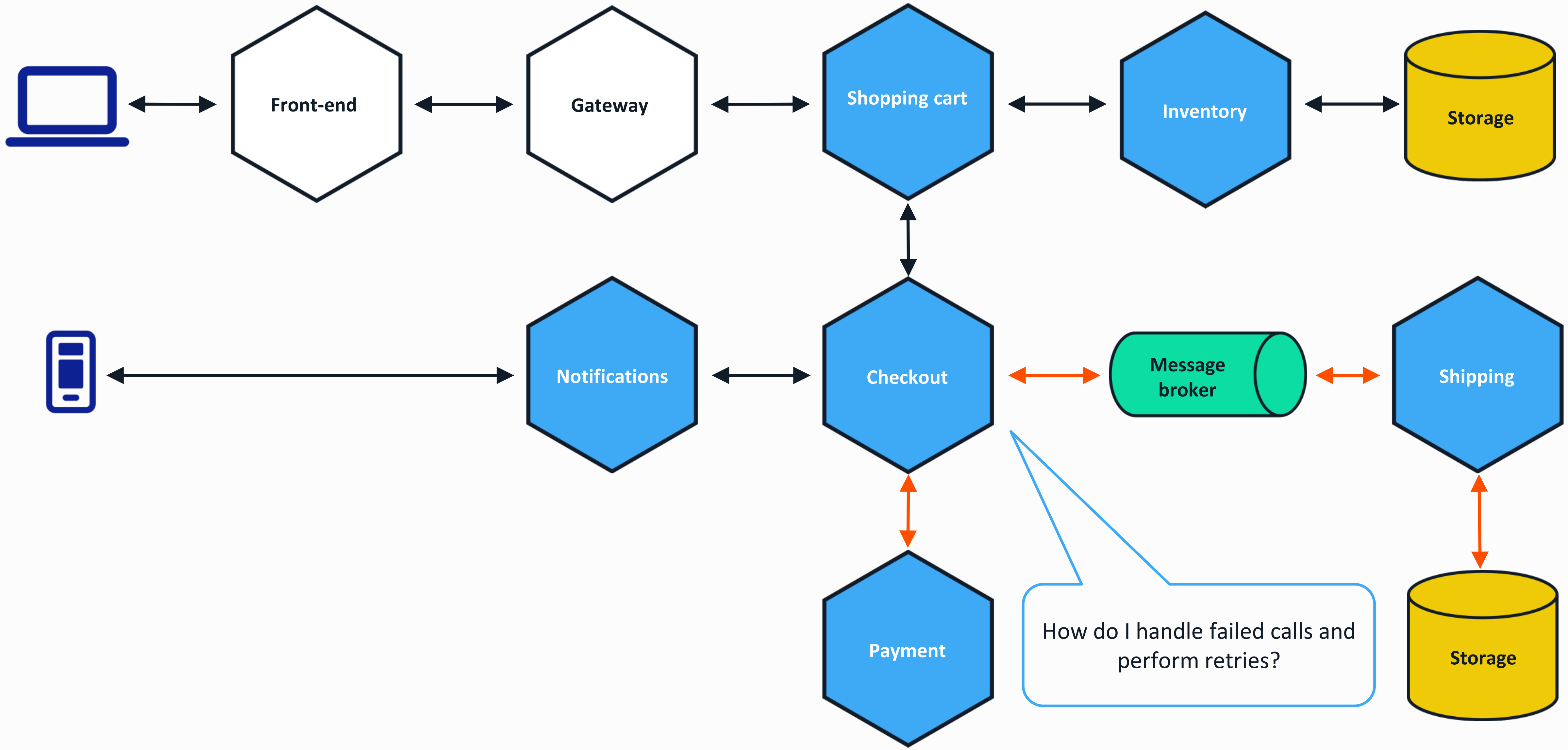
How can I trace requests across services and message brokers?

Developer challenges – Access control



How can I configure access to services and other resources?

Developer challenges – Resiliency



How do I handle failed calls and perform retries?

Distributed

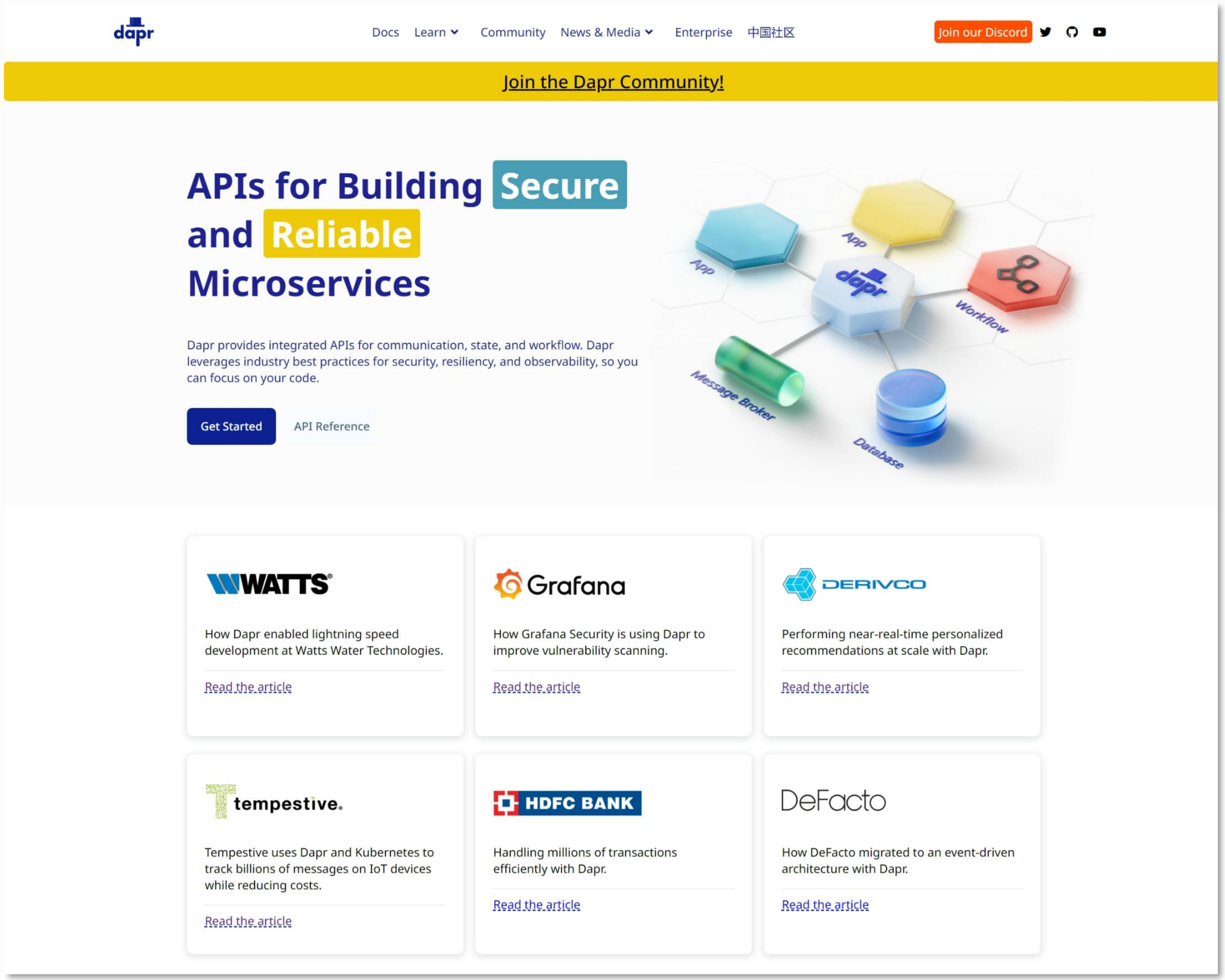
Application

Runtime

dapr.io

CLOUD NATIVE COMPUTING FOUNDATION

Graduated project



Docs Learn Community News & Media Enterprise 中国社区 Join our Discord

Join the Dapr Community!

APIs for Building Secure and Reliable Microservices

Dapr provides integrated APIs for communication, state, and workflow. Dapr leverages industry best practices for security, resiliency, and observability, so you can focus on your code.

Get Started API Reference

WATTS
How Dapr enabled lightning speed development at Watts Water Technologies.
Read the article

Grafana
How Grafana Security is using Dapr to improve vulnerability scanning.
Read the article

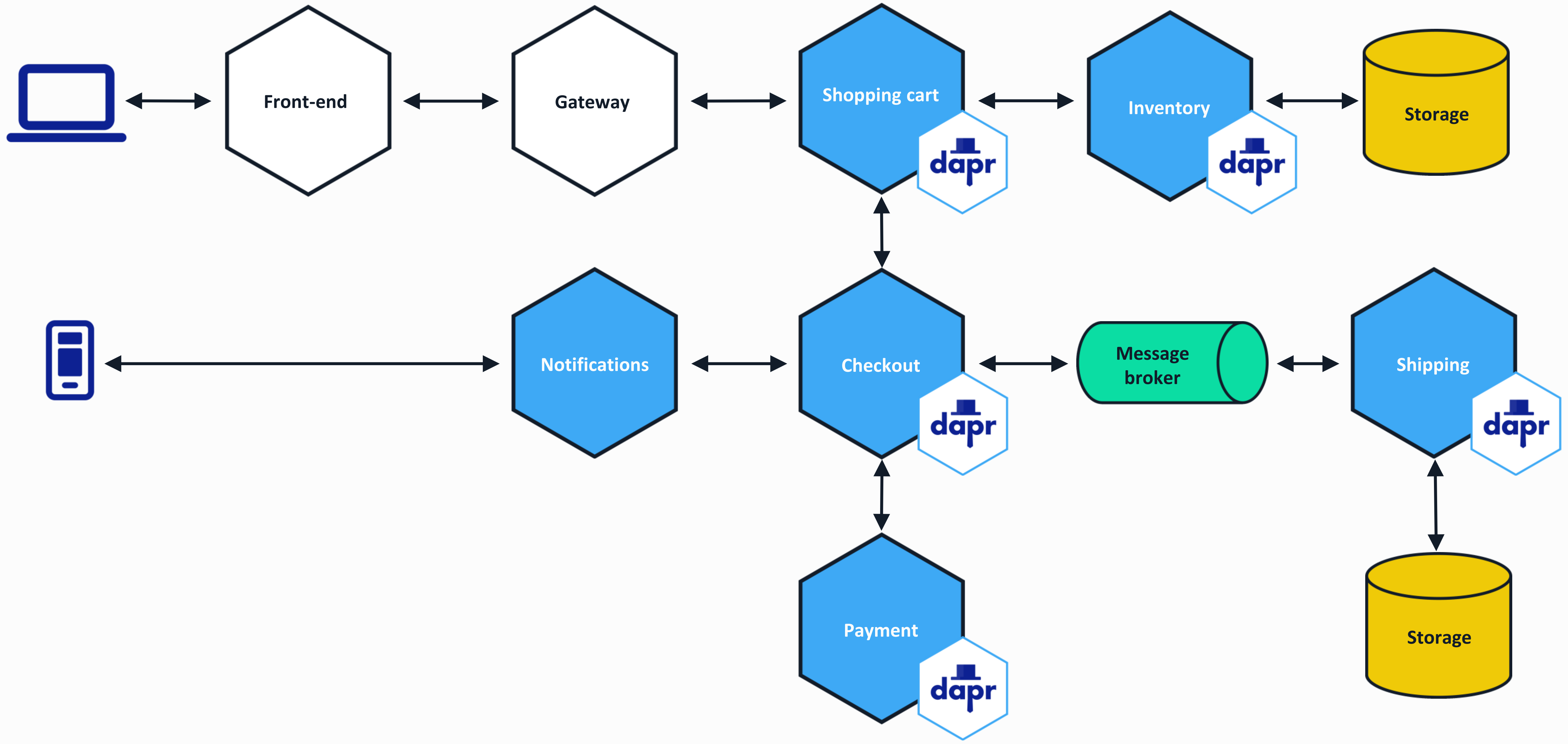
DERIVCO
Performing near-real-time personalized recommendations at scale with Dapr.
Read the article

tempestive.
Tempestive uses Dapr and Kubernetes to track billions of messages on IoT devices while reducing costs.
Read the article

HDFC BANK
Handling millions of transactions efficiently with Dapr.
Read the article

DeFacto
How DeFacto migrated to an event-driven architecture with Dapr.
Read the article

Dapr uses a sidecar pattern



The Dapr sidecar provides built-in security, resiliency and observability capabilities.

Speeds up application development by providing an integrated set of APIs for communication, state, and workflow.

State of enterprise developers

Must develop resilient, scalable, distributed apps that interact with services.

Want to focus on writing code, not learning infrastructure.

Trending toward serverless platforms with simple code to cloud pipelines.

Use multiple languages and frameworks during development.

What is holding back distributed app development?

Limited tools and runtimes to build distributed applications.

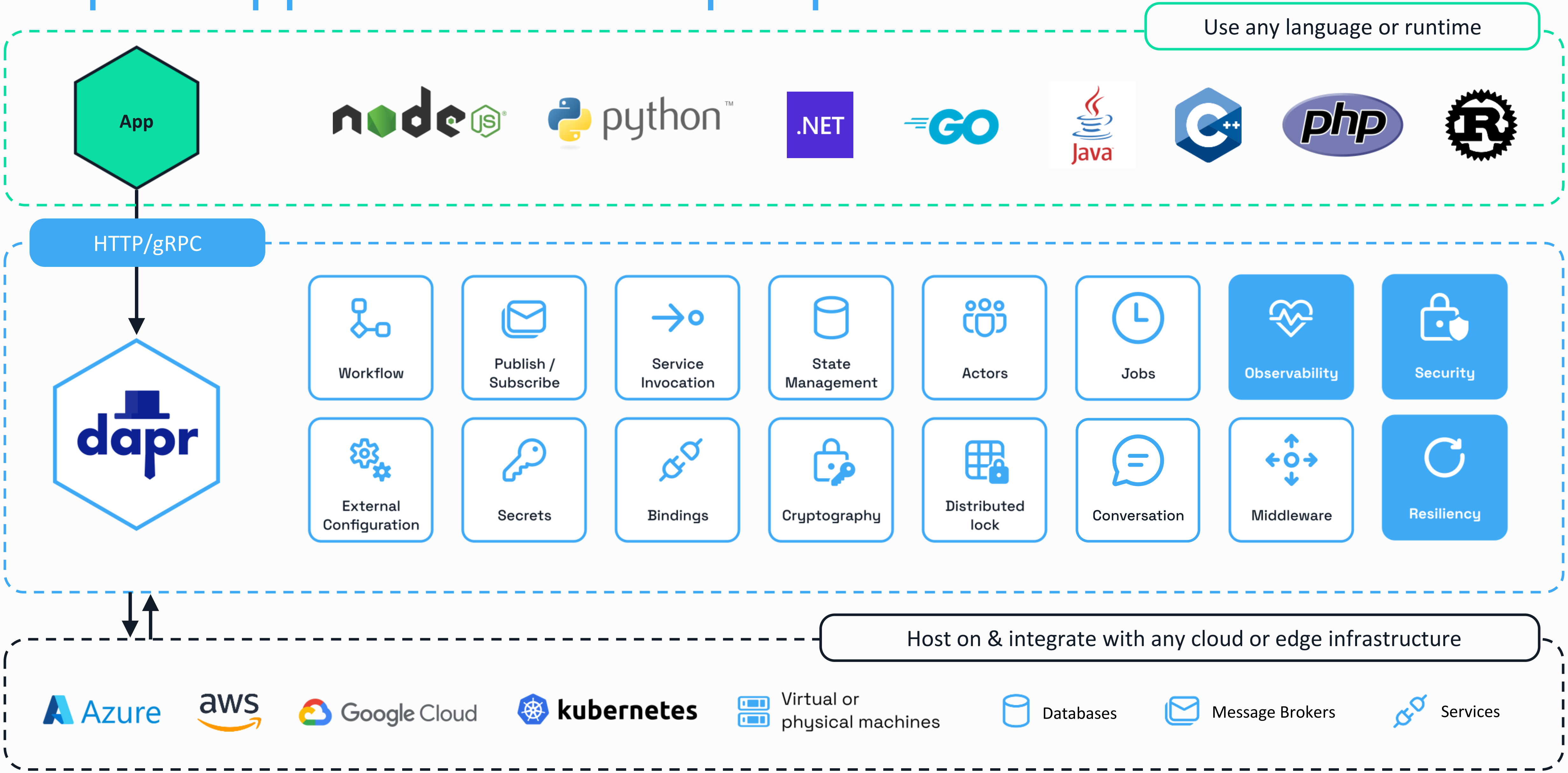
Runtimes have limited language support and tightly controlled feature sets.

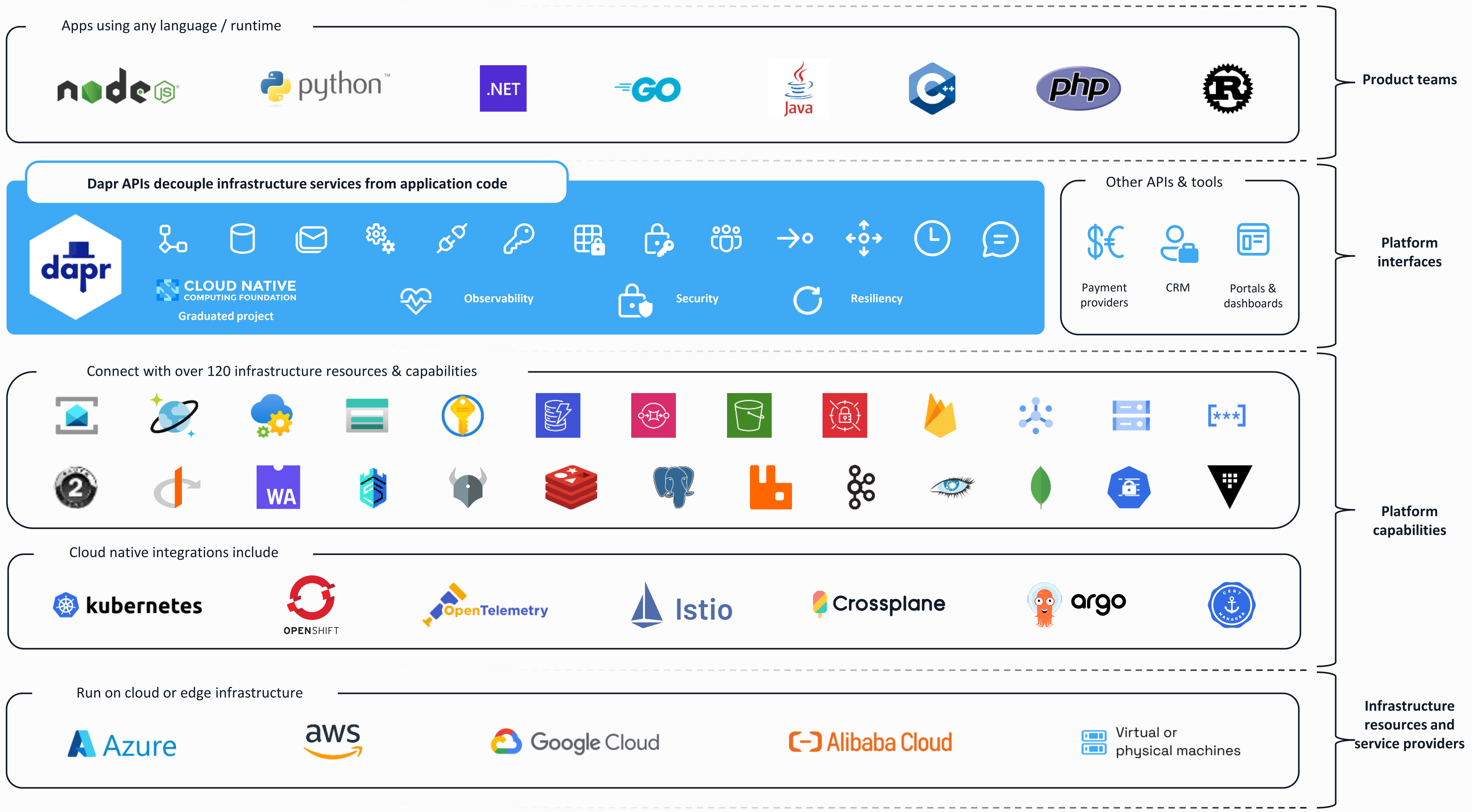
Runtimes only target specific infrastructure platforms with limited portability.

Dapr Goals

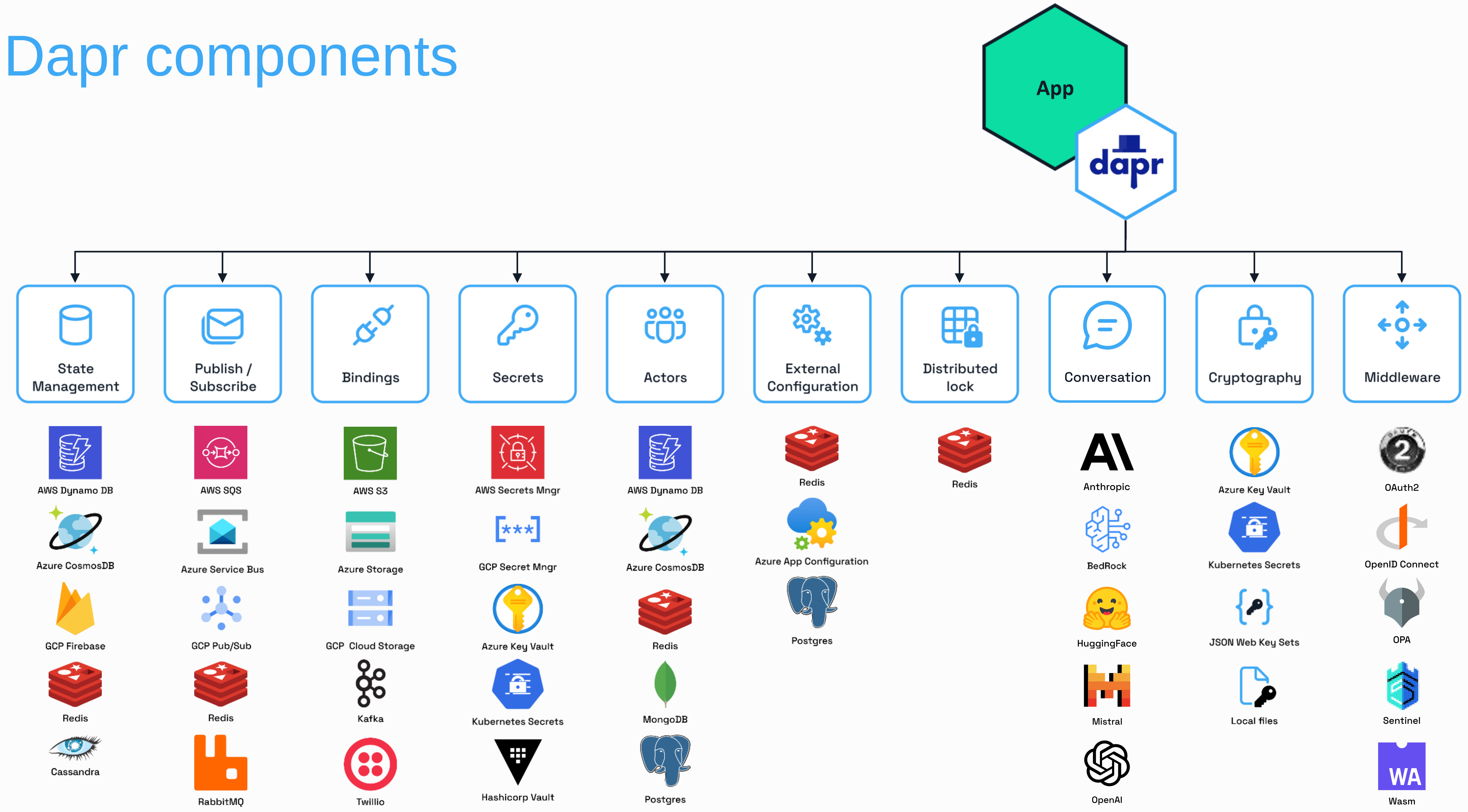
- Provide an integrated set of APIs
- Includes best practices & standards
- Extensible and pluggable
- Any language or framework
- Platform agnostic
- Community driven, vendor neutral

Dapr – Application developer platform

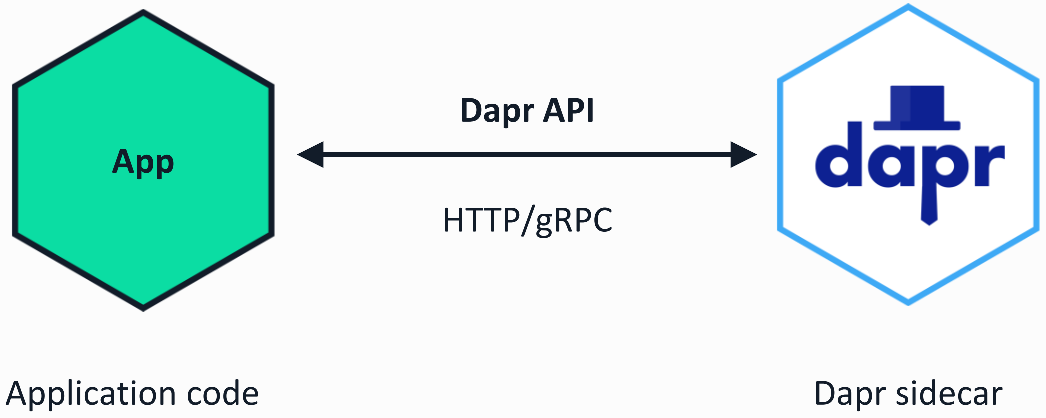



Apps using any language / runtime
node js
python
.NET
GO
Java
php
Product teams
Dapr APIs decouple infrastructure services from application code
dapr
CLOUD NATIVE COMPUTING FOUNDATION
Graduated project
Observability
Security
Resiliency
Other APIs & tools
Payment providers
CRM
Portals & dashboards
Platform interfaces
Connect with over 120 infrastructure resources & capabilities
WA
Cloud native integrations include
kubernetes
OPENSHIFT
OpenTelemetry
Istio
Crossplane
argo
Platform capabilities
Run on cloud or edge infrastructure
Azure
aws
Google Cloud
Alibaba Cloud
Virtual or physical machines
Infrastructure resources and service providers

Dapr components

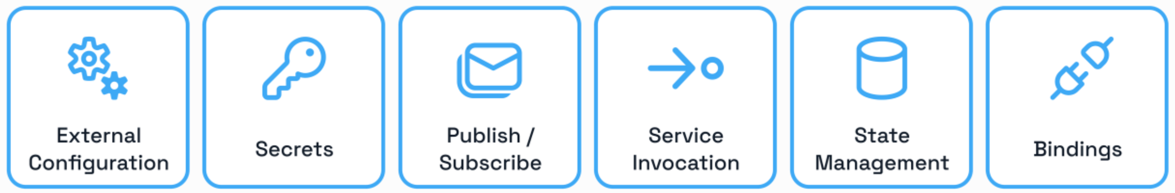


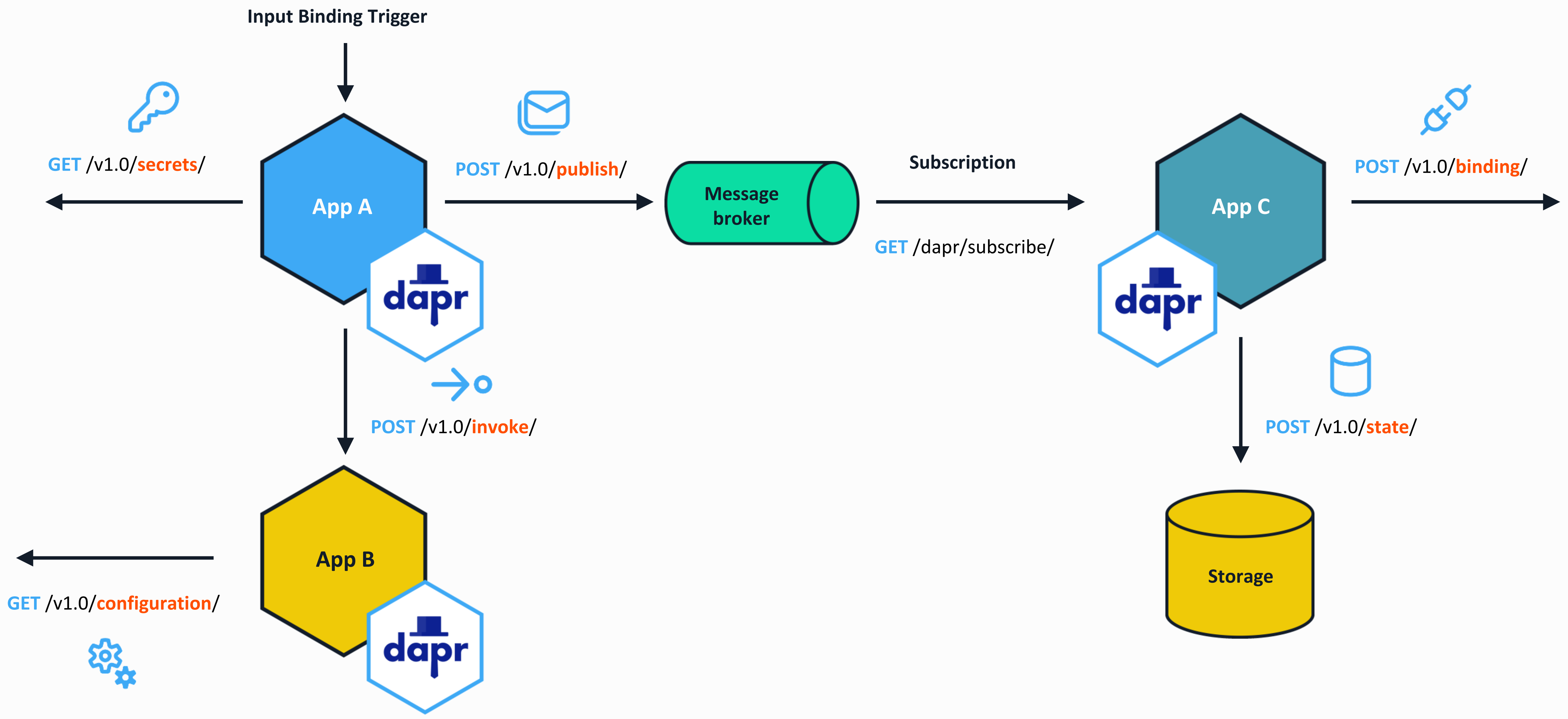
Sidecar pattern and the Dapr API



POST	http://localhost:3500/v1.0/invoke/cart/method/order
GET	http://localhost:3500/v1.0/state/inventory/item50
POST	http://localhost:3500/v1.0/publish/mybroker/order-messages
GET	http://localhost:3500/v1.0/secrets/vault/dbaccess
POST	http://localhost:3500/v1.0/workflows/dapr/businessprocess/start

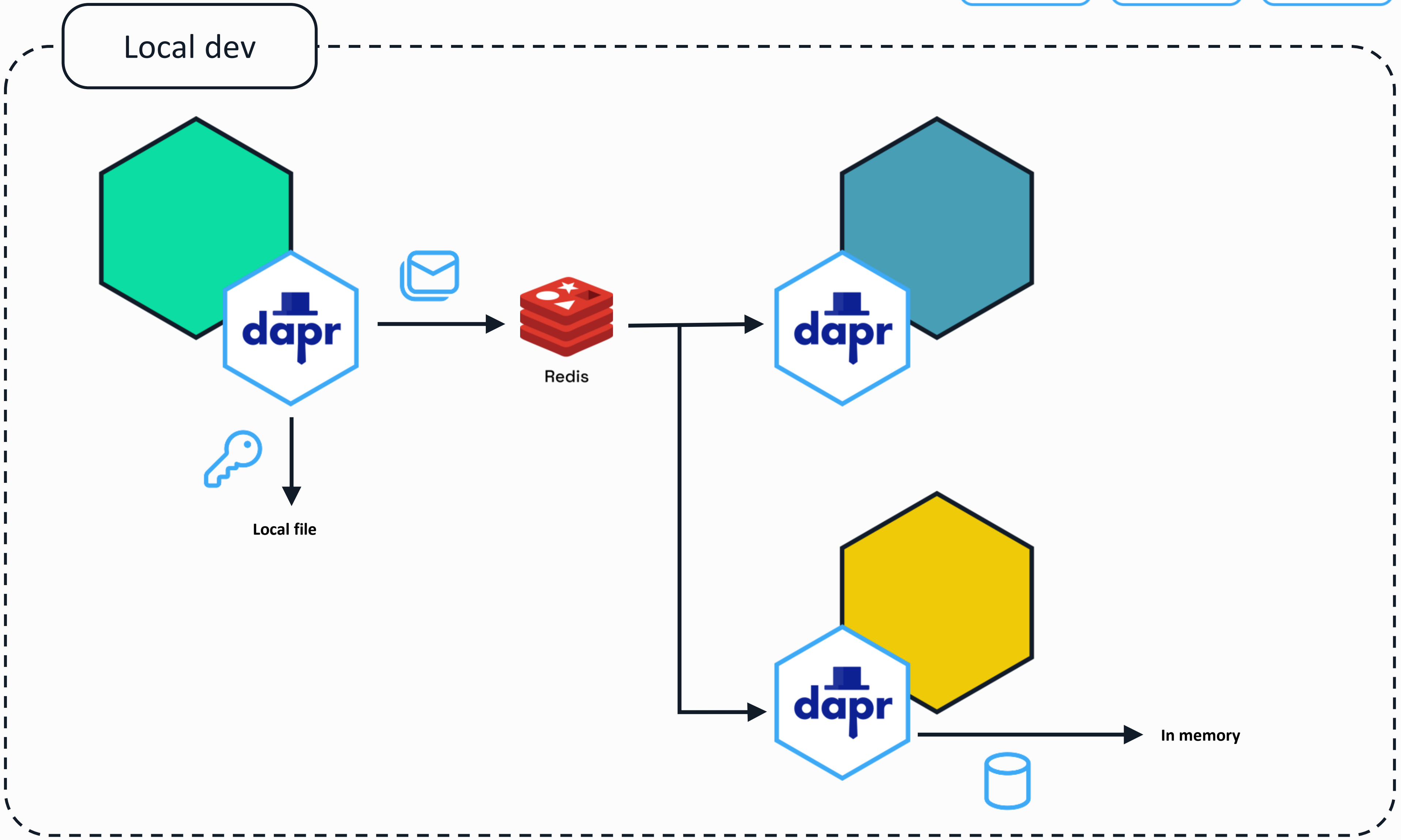
Using the Dapr APIs





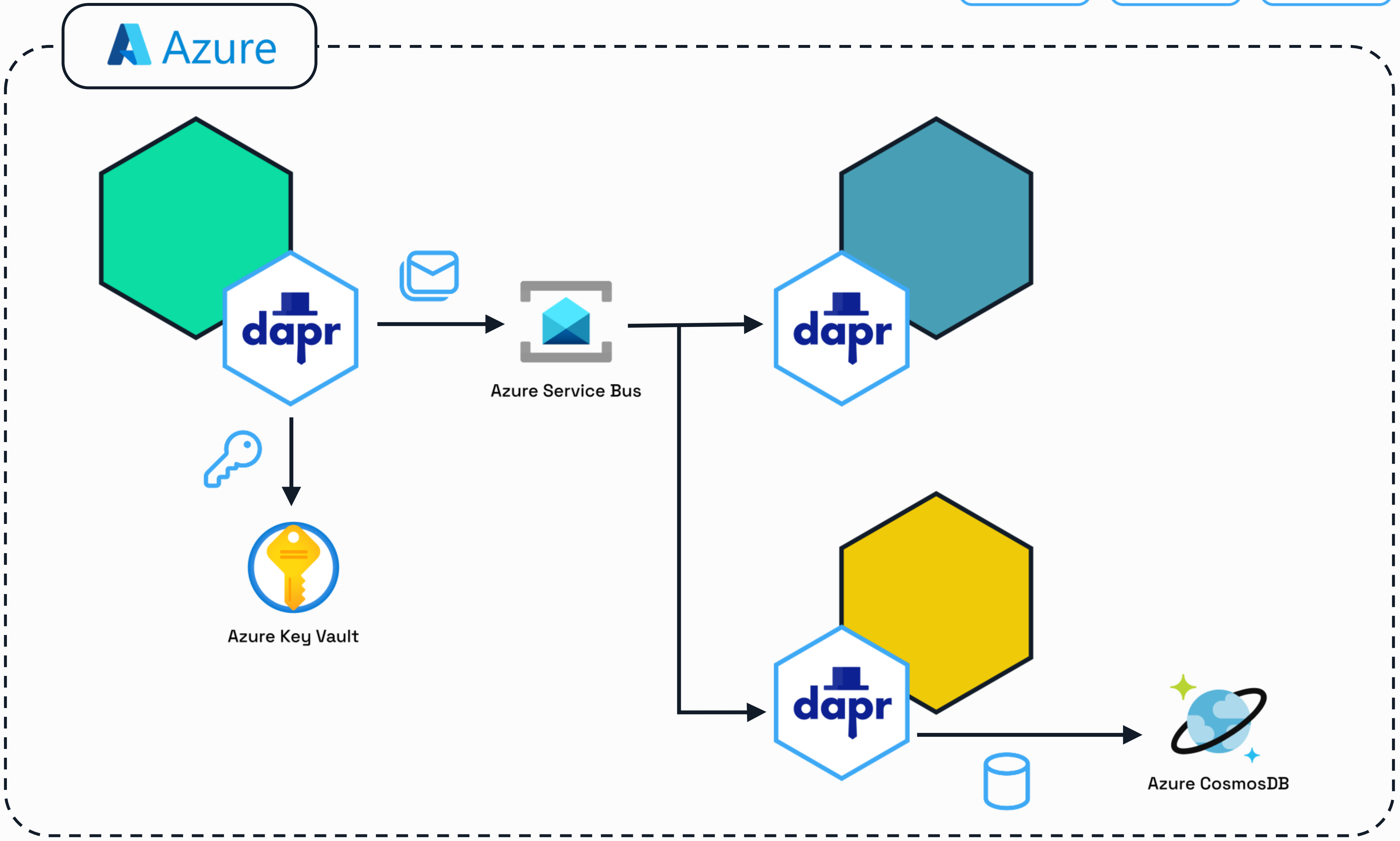
Swappable component model

Secrets
Publish / Subscribe
State Management



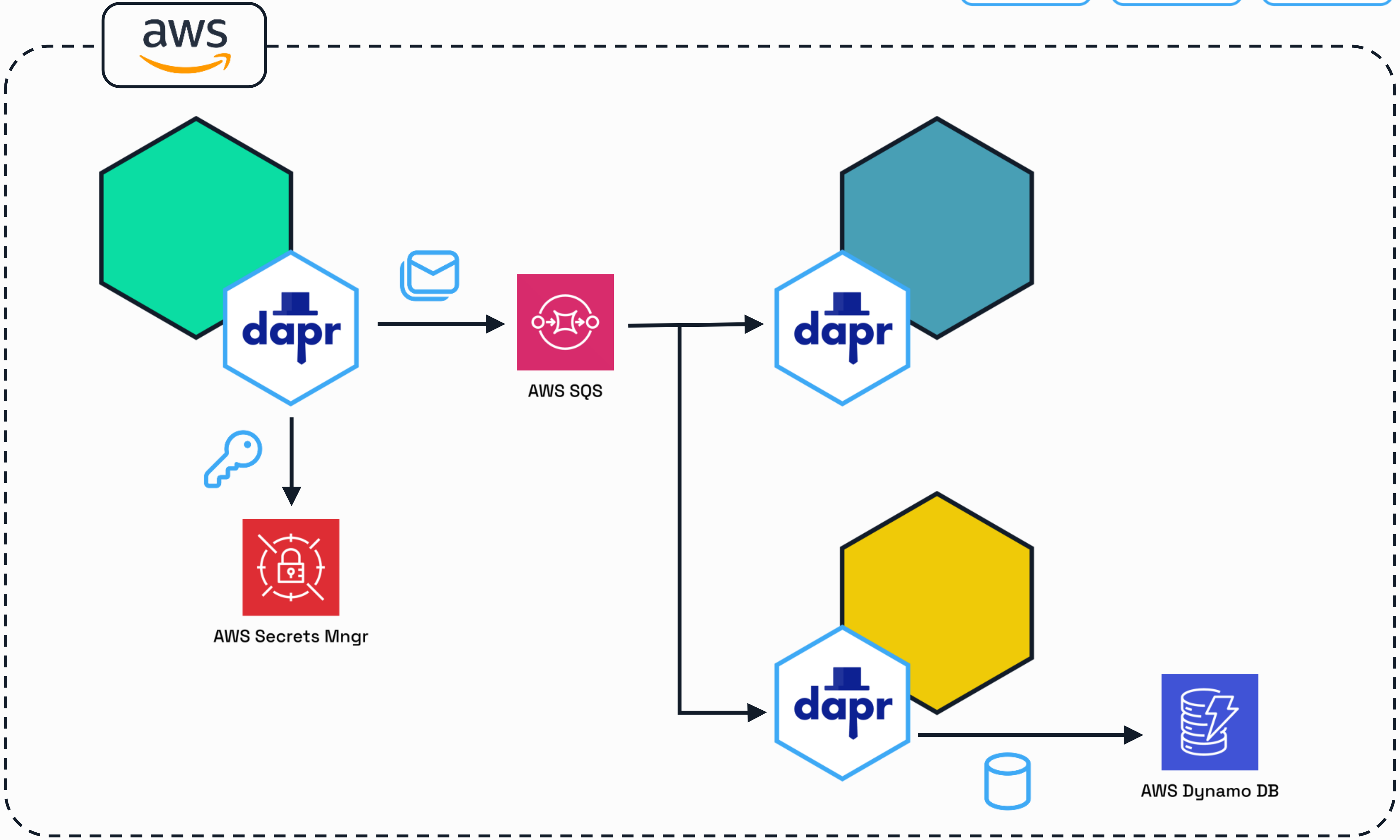
Swappable component model

Secrets
Publish / Subscribe
State Management



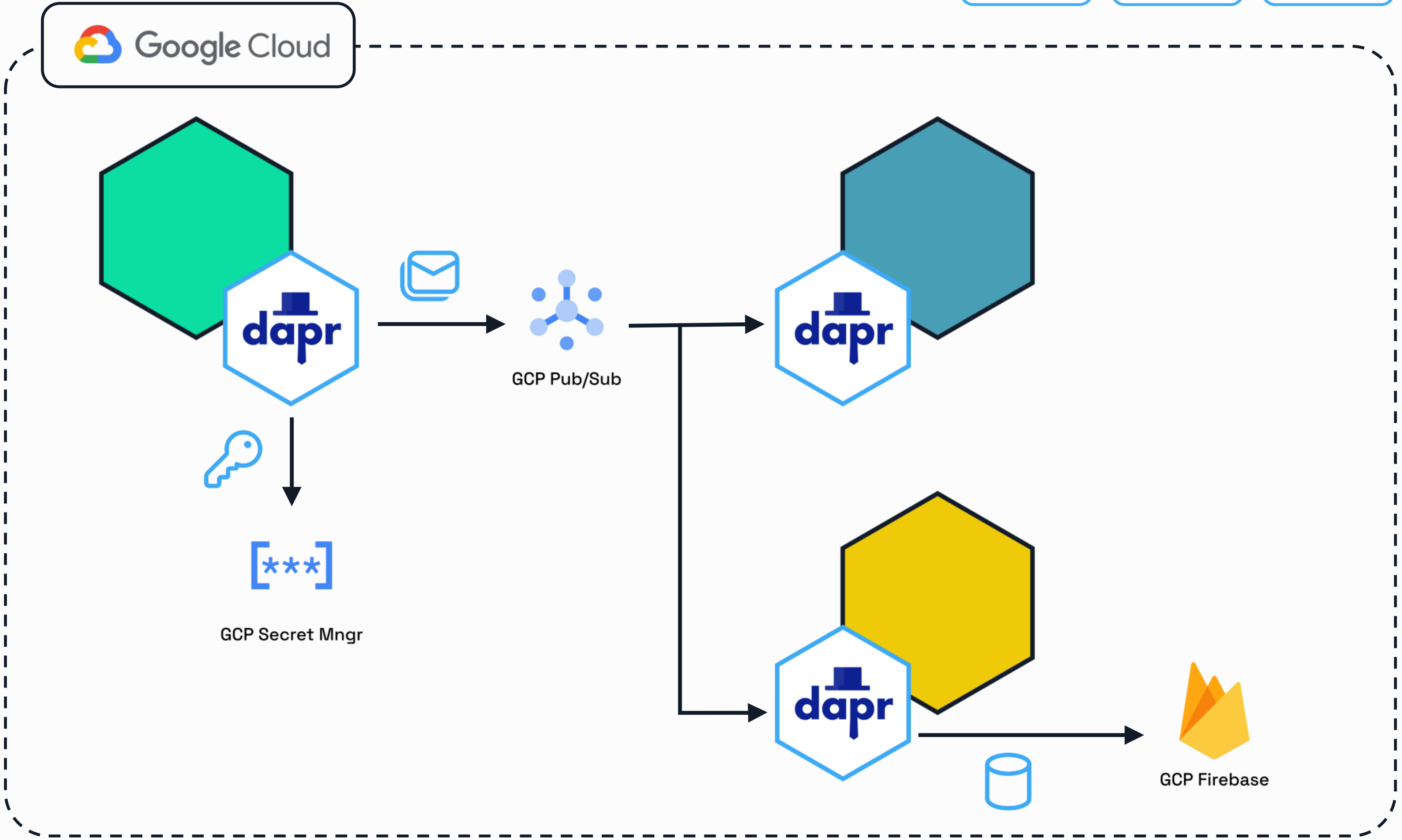
Swappable component model

Secrets
Publish / Subscribe
State Management



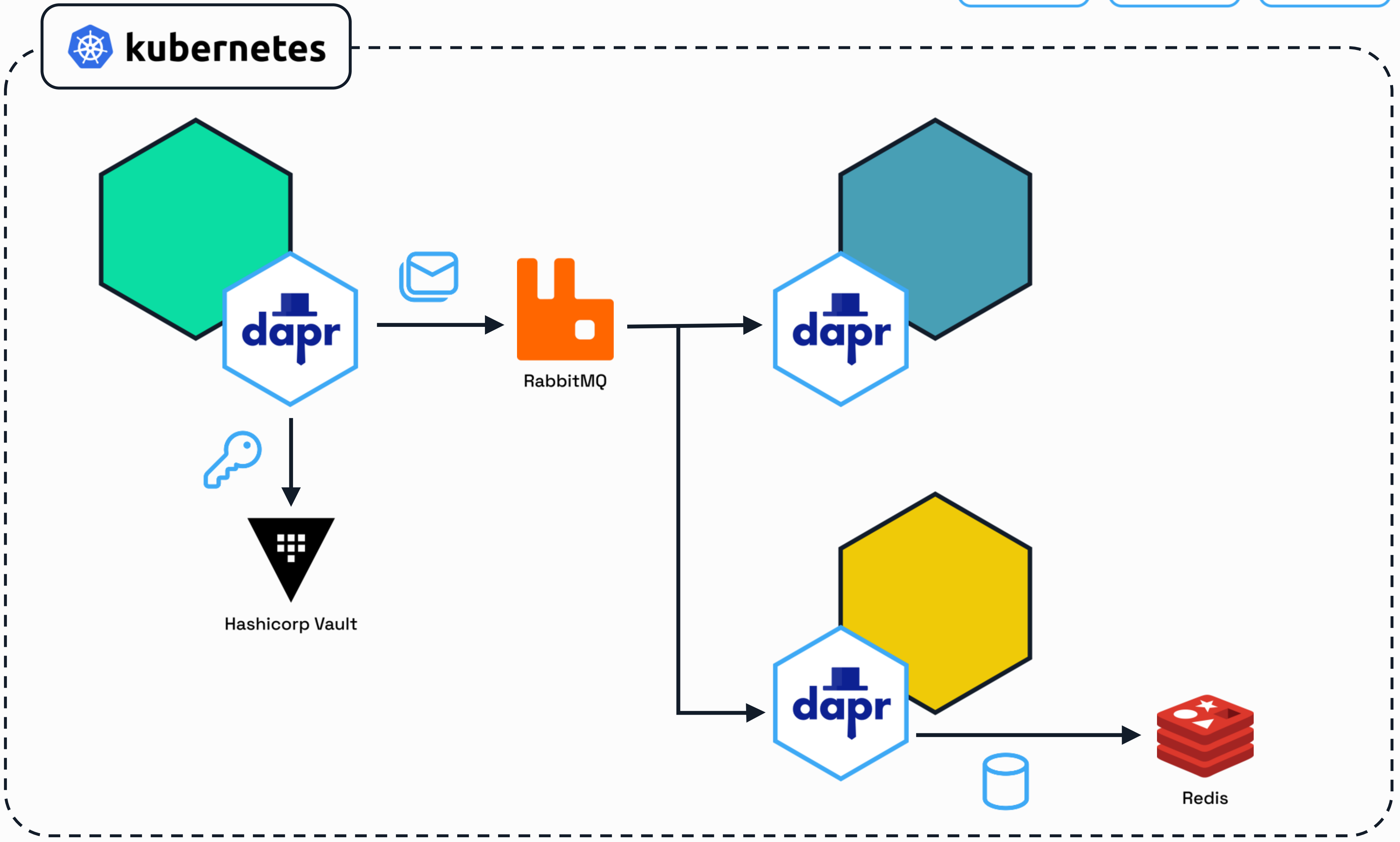
Swappable component model

Secrets
Publish / Subscribe
State Management



Swappable component model

Secrets
Publish / Subscribe
State Management



Use Dapr anywhere

Azure Container Apps

Diagrid Catalyst

Microsoft Azure

Google Cloud

aws

Alibaba Cloud

kubernetes

Virtual or
physical machines

Dapr community

14/157
Largest CNCF project

CLOUD NATIVE
COMPUTING FOUNDATION

120
Community components

700k+
Docker Hub pulls/month

300k
Unique doc views/month

31k
GitHub stars

3.9k
Contributors

8k
Discord members

Dapr contributors

intel® Diagrid tetrate sysdig ZEISS

SAP BOSCH vmware® HashiCorp new relic.

HUAWEI Microsoft amazon Alibaba Cloud INNOVIAN

Dapr users

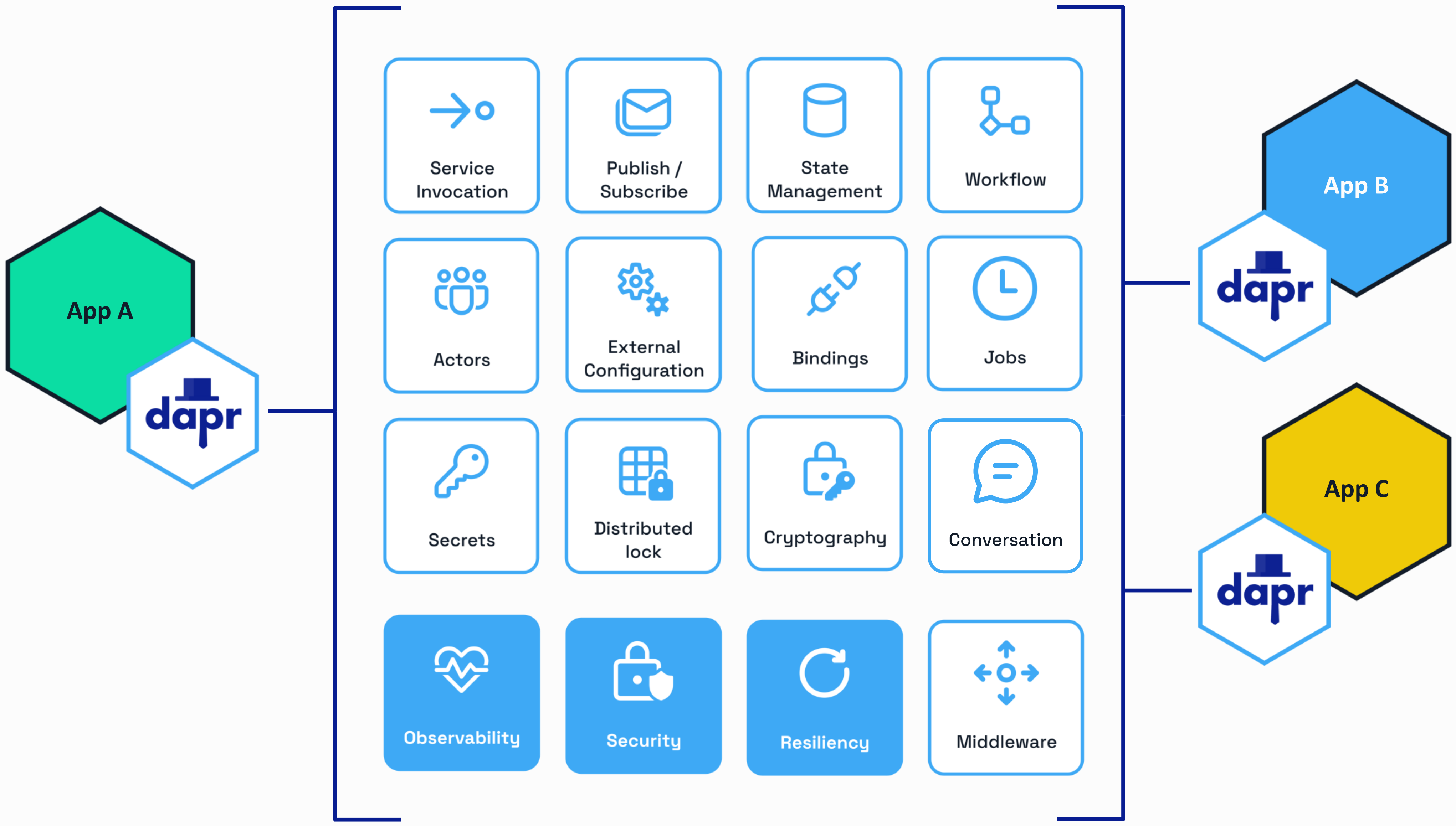
IBM pwc Microsoft intel Alibaba Cloud ZEISS NASA

HDFC BANK CISCO Rakuten Lufthansa FUJITSU Grafana zscaler

BOSCH DeFacto VONAGE at bay Dotmatics wortell IGNITION GROUP

Composabl DERIVCO WATTS tempestive.

Dapr APIs & Cross cutting concerns



Service
Invocation

Service Invocation API

https://docs.dapr.io/developing-applications/building-blocks/service-invocation/

Service Invocation

Service
Invocation

The service invocation API allows synchronous communication between services.

- Service discovery via name resolution components
- Invoke HTTP and gRPC services consistently
- Configurable resiliency policies
- Built-in distributed tracing & metrics
- Access control policies & mTLS
- Chain pluggable middleware components

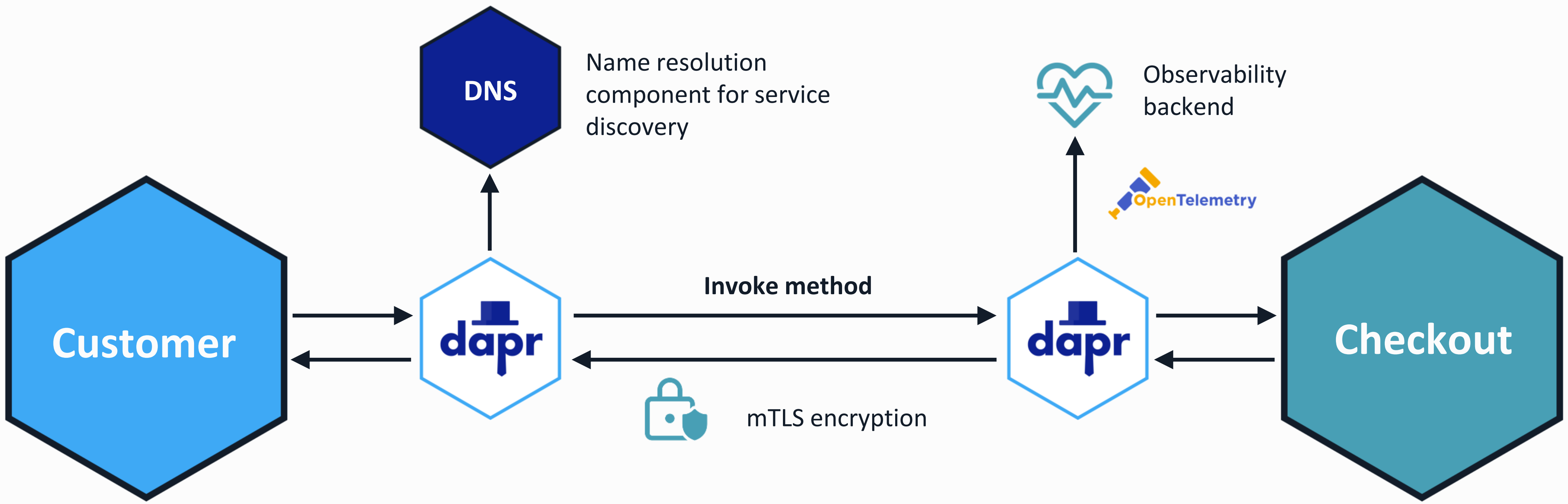
Service Invocation

Service Invocation

POST
http://localhost:3500/v1.0/**invoke**/checkout/method/order

POST
http://localhost:5100/order

Service Invocation



POST
http://localhost:3500/v1.0/invoke/checkout/method/order

POST
http://localhost:5100/order

Service Invocation in .NET

Service
Invocation

```
var order = new Order(orderId);

var client = DaprClient.CreateInvokeHttpClient(appId: "order-processor");

var response = await client.PostAsJsonAsync("/orders", order);
```

Service Invocation in Python

Service
Invocation

```
base_url = os.getenv('BASE_URL', 'http://localhost') + ':' +
              os.getenv('DAPR_HTTP_PORT', '3500')

headers = {'dapr-app-id': 'order-processor', 'content-type': 'application/json'}

order = {'orderId': orderId}

result = requests.post(
         url='%s/orders' % (base_url),
         data=json.dumps(order),
         headers=headers
```

Service
Invocation

Service Invocation Demo

https://docs.dapr.io/getting-started/quickstarts/serviceinvocation-quickstart/

Publish /
Subscribe

Publish / Subscribe API

https://docs.dapr.io/developing-applications/building-blocks/pubsub/

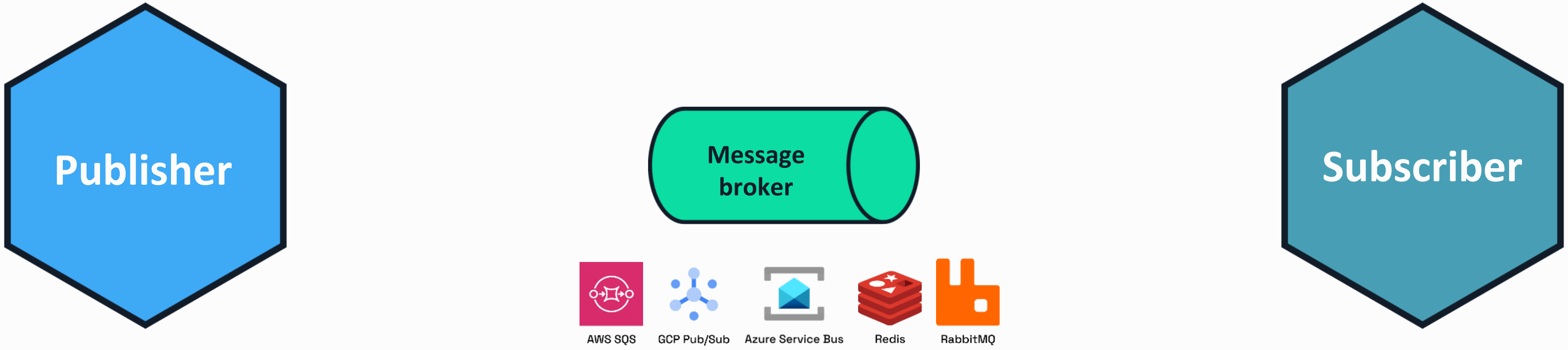
Publish / Subscribe

Publish /
Subscribe

The publish subscribe API allows asynchronous communication between services.

- Integrates with many message brokers and queues
- Guaranteed at least one delivery
- Use declarative or programmatic subscriptions
- Use content-based message routing
- Set dead-letter topics and resiliency policies
- Limit publish and subscribe access by using scopes

Publish / Subscribe

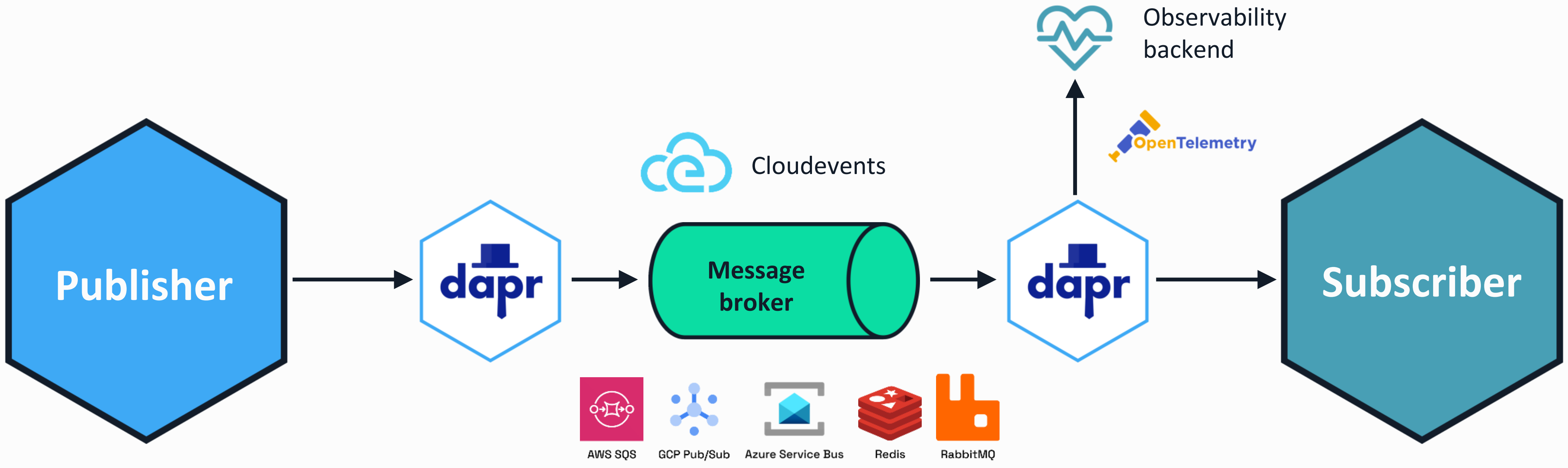


POST
http://localhost:3500/v1.0/publish/mybroker/order-messages

POST
http://localhost:5100/orders

Publish / Subscribe

Publish / Subscribe

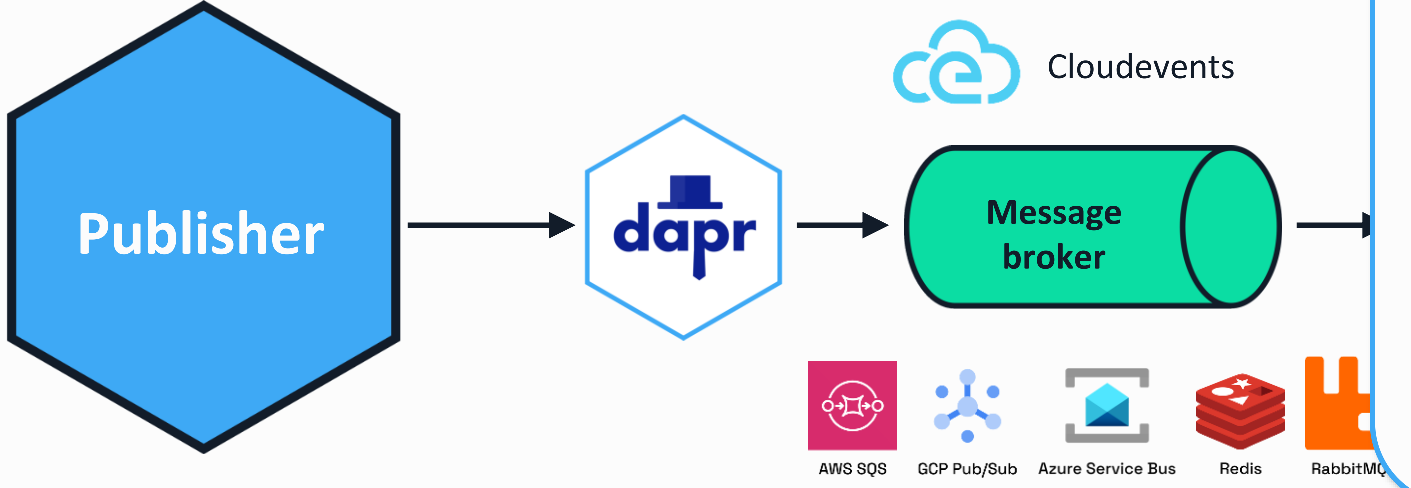


POST
http://localhost:3500/v1.0/publish/mybroker/order-messages

POST
http://localhost:5100/orders

Publish / Subscribe Component

Publish / Subscribe



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

POST
http://localhost:3500/v1.0/**publish**/mybroker/order-messages

POST
http://localhost:5100/orders

Publish / Subscribe with .NET SDK

Publish /
Subscribe

Publish

```
var order = new Order(orderId);

using var client = new DaprClientBuilder().Build();

await client.PublishEventAsync("orderpubsub", "orders", order);
```

Subscribe

```
app.UseCloudEvents();

app.MapSubscribeHandler();

app.MapPost("/orders", [Topic("orderpubsub", "orders")] (Order order) =>
{
    return Results.Ok(order);
});
```

Publish / Subscribe with Python SDK

Publish / Subscribe

Publish

```
with DaprClient() as client:
    order = {'orderId’: orderId}

    result = client.publish_event(
            pubsub_name='orderpubsub',
            topic_name='orders',
            data=json.dumps(order),
            data_content_type='application/json',
    )
```

Publish / Subscribe with Python SDK

Publish / Subscribe

Subscribe

```
@app.route('/dapr/subscribe', methods=['GET'])
def subscribe():
    subscriptions = [{
        'pubsubname': 'orderpubsub',
        'topic': 'orders',
        'route': 'orders'
    }]
    return jsonify(subscriptions)

@app.route('/orders', methods=['POST'])
def orders_subscriber():
    event = from_http(request.headers, request.get_data())
    return json.dumps({'success': True}), 200, {
        'ContentType': 'application/json'}
```

Publish /
Subscribe

Publish / Subscribe Demo

https://docs.dapr.io/getting-started/quickstarts/pubsub-quickstart/

State
Management

State Management API (Key/Value)

https://docs.dapr.io/developing-applications/building-blocks/state-management/

State Management

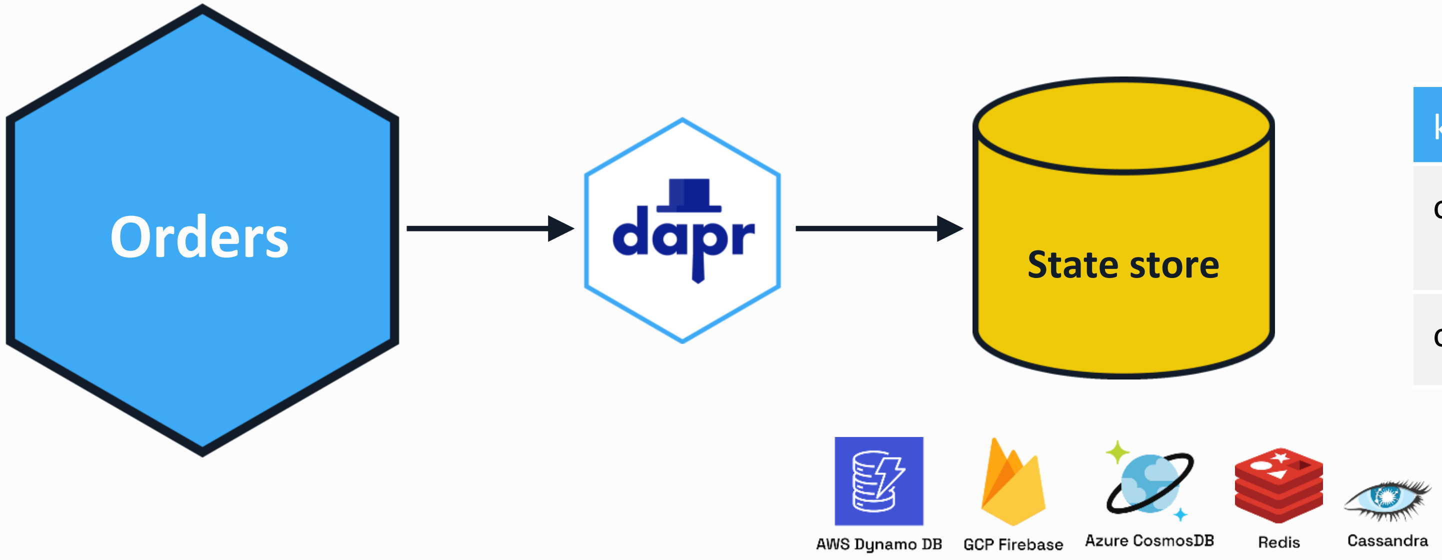
State Management

The state management API allows key/value pair storage across many supported state stores.

- Integrates with many state stores
- Configurable concurrency and consistency behaviors
- Use bulk operations
- Use resiliency policies
- Limit access by using scopes

State Management (Key/Value)

State Management



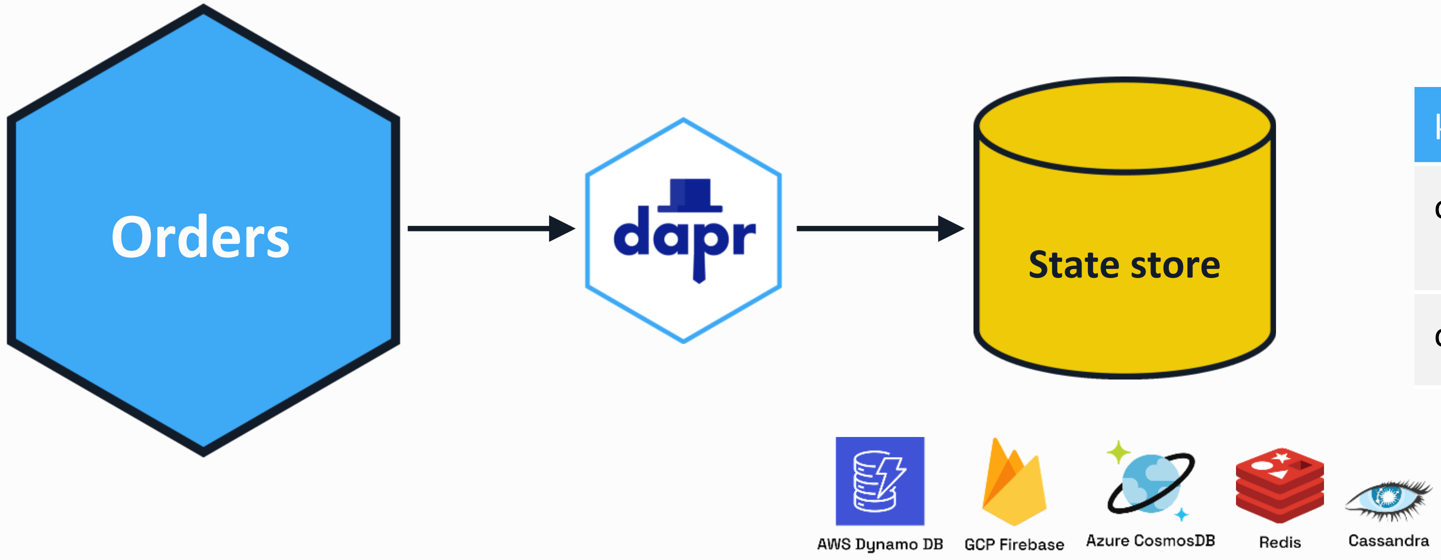
key	**field**	value
orders||order1	data	"{orderId:1}"
orders||order1	version	1

POST
http://localhost:3500/v1.0/**state**/mystatestore

```
[{
  "key": "order1",
  "value": "{orderId: 1}"
}]
```

State Management (Key/Value)

State
Management



key	field	value
orders||order1	data	"{orderId:1}"
orders||order1	version	1

GET
http://localhost:3500/v1.0/state/mystatestore/order1

State Management with .NET SDK

State
Management

```
var order = new Order(orderId);

await client.SaveStateAsync(
       DAPR_STORE_NAME,
       orderId.ToString(),
       order.ToString());

var result = await client.GetStateAsync<string>(
       DAPR_STORE_NAME,
       orderId.ToString());

await client.DeleteStateAsync(
       DAPR_STORE_NAME,
       orderId.ToString());
```

State Management with Python SDK

State
Management

```
with DaprClient() as client:
        order = {'orderId': orderId}

        client.save_state(DAPR_STORE_NAME, orderId, str(order))

        result = client.get_state(DAPR_STORE_NAME, orderId)

        client.delete_state(store_name=DAPR_STORE_NAME, key=orderId)
```

State
Management

State Management (Key/Value) Demo

https://docs.dapr.io/getting-started/quickstarts/statemanagement-quickstart/

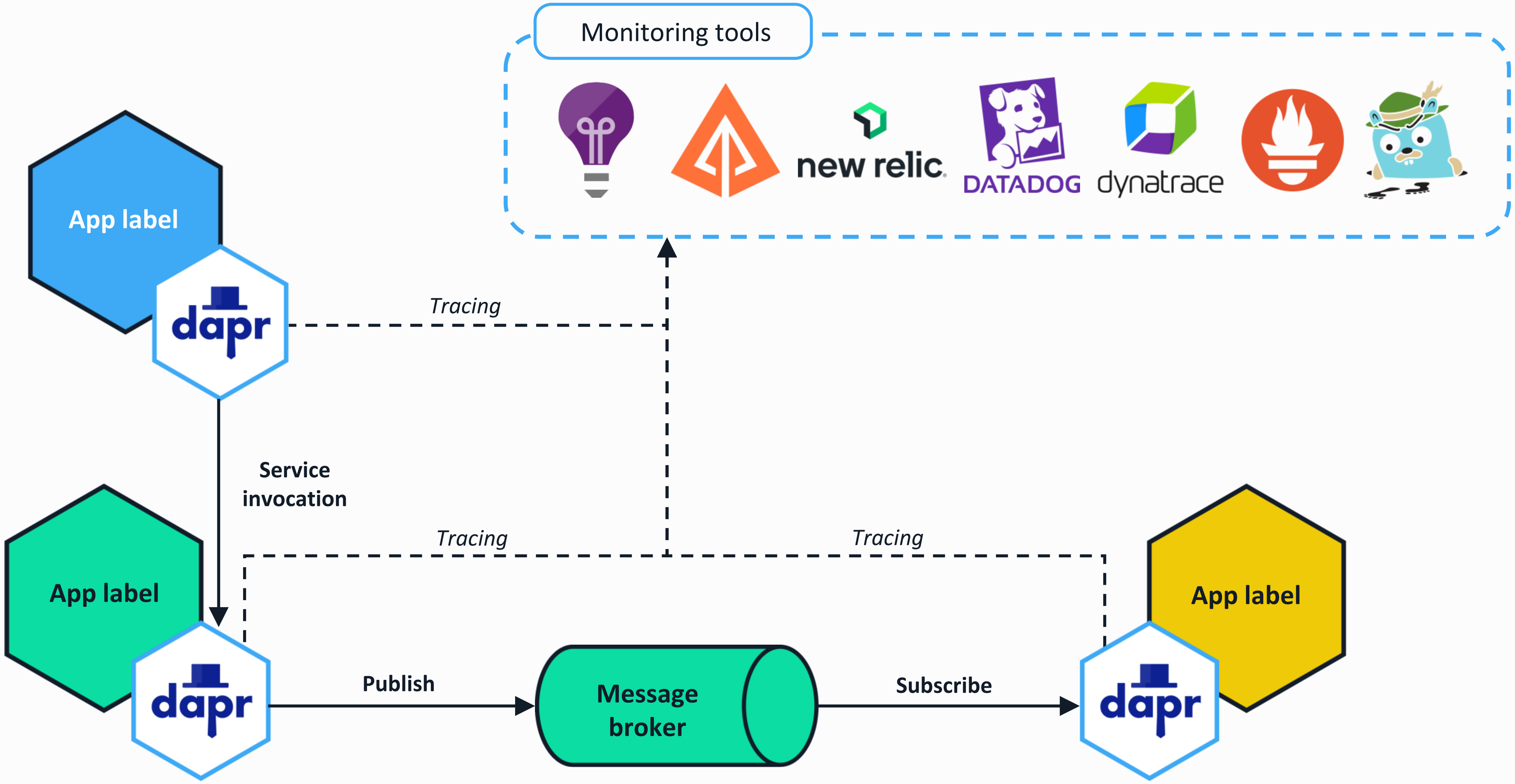
Observability

Observability

https://docs.dapr.io/concepts/observability-concept/

Distributed tracing

Observability



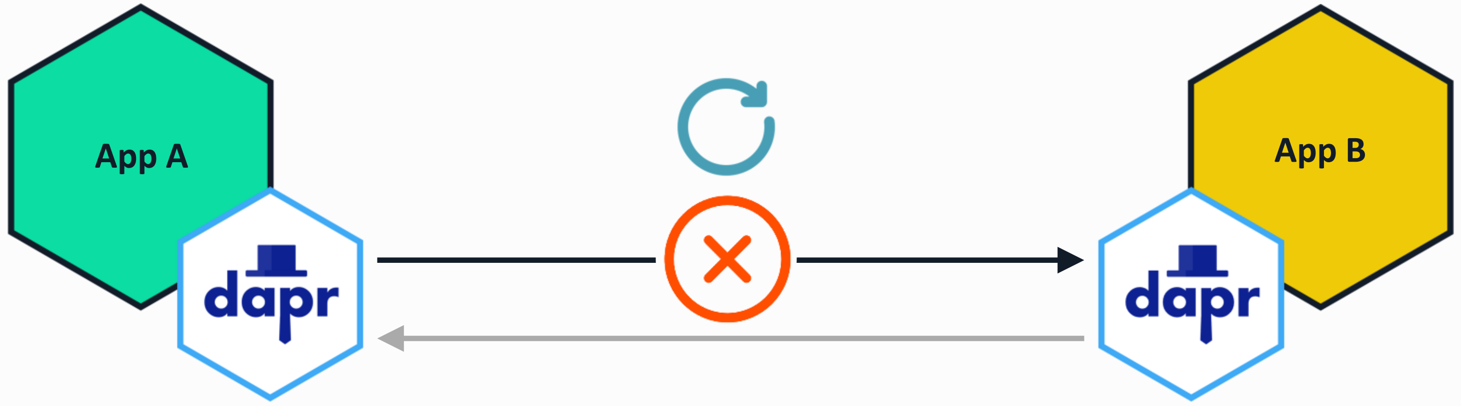
Resiliency

Resiliency

https://docs.dapr.io/concepts/resiliency-concept/

Service invocation resiliency

The built-in service invocation retries are always performed with a backoff interval of 1 second up to a threshold of 3 times.

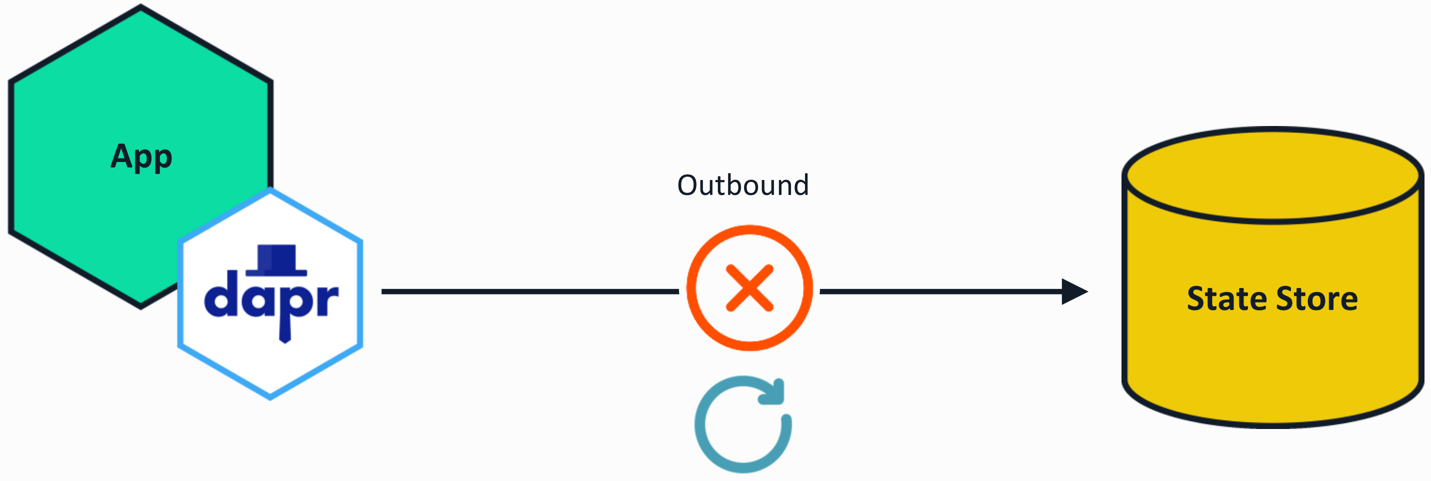


Additionally, service invocation resiliency polices for *retries*, *timeouts* and *circuit breakers* can be applied.

Outbound component resiliency

Resiliency

Component resiliency polices can be applied to outbound component calls. For example, calls to a state store.

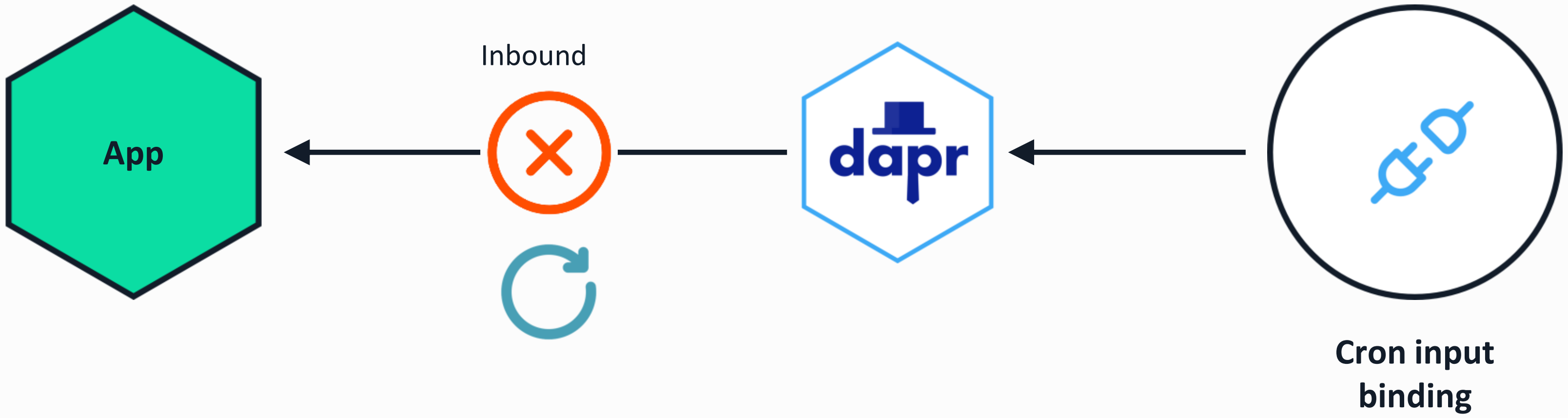


Additionally, some components have retry capabilities built-in. The policies are configured on a per component basis.

Inbound component resiliency

Resiliency

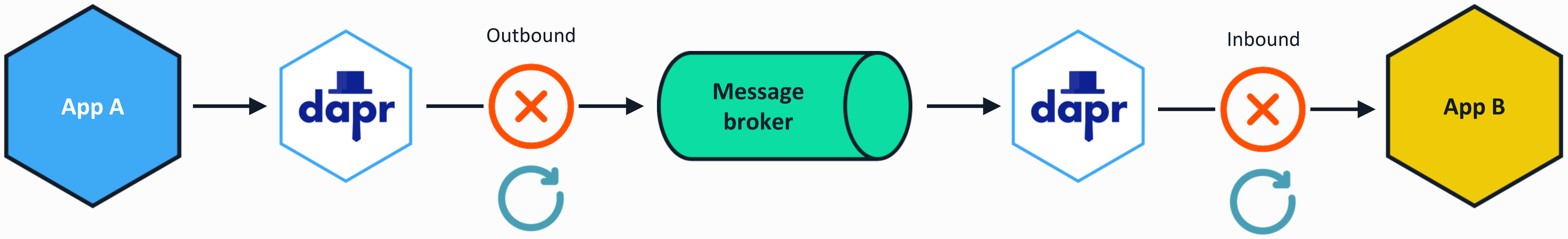
Resiliency polices can be applied to inbound component calls. For example, pub/sub subscriptions and input bindings.



Pub/Sub resiliency

Outbound component resiliency policies for can be applied to message publishing.

Inbound component resiliency polices can be applied to subscriptions when delivering messages.



Additionally, many pub/sub components have *retry* capabilities built-in. The policies are configured on a per component basis.

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.

```
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: myresiliency
scopes:
  - order-processor

spec:
  policies:
    retries:
      retryForever:
        policy: constant
        duration: 5s
        maxRetries: -1

    circuitBreakers:
      simpleCB:
        maxRequests: 1
        timeout: 5s
        trip: consecutiveFailures >= 5

  targets:
    components:
      statestore:
        outbound:
          retry: retryForever
          circuitBreaker: simpleCB
```

Hosting modes

Hosting modes

Self-hosted

Run **`dapr init`** to install Docker images.

Run any app with a Dapr side car using dapr run.

Virtual/Physical machines

Self-deploy Dapr control plane and Hashicorp Consul per machine.

Use the Dapr Installer Bundle for airgapped environments.

Run any app with a Dapr side car using **`dapr run`**.

Kubernetes

Run **`dapr init -k`** to install Dapr (or use Helm). Integrated Dapr control plane.

Deploys placement, operator, sentry and injector pods.

Automatically injects a Dapr sidecar into all annotated pods.

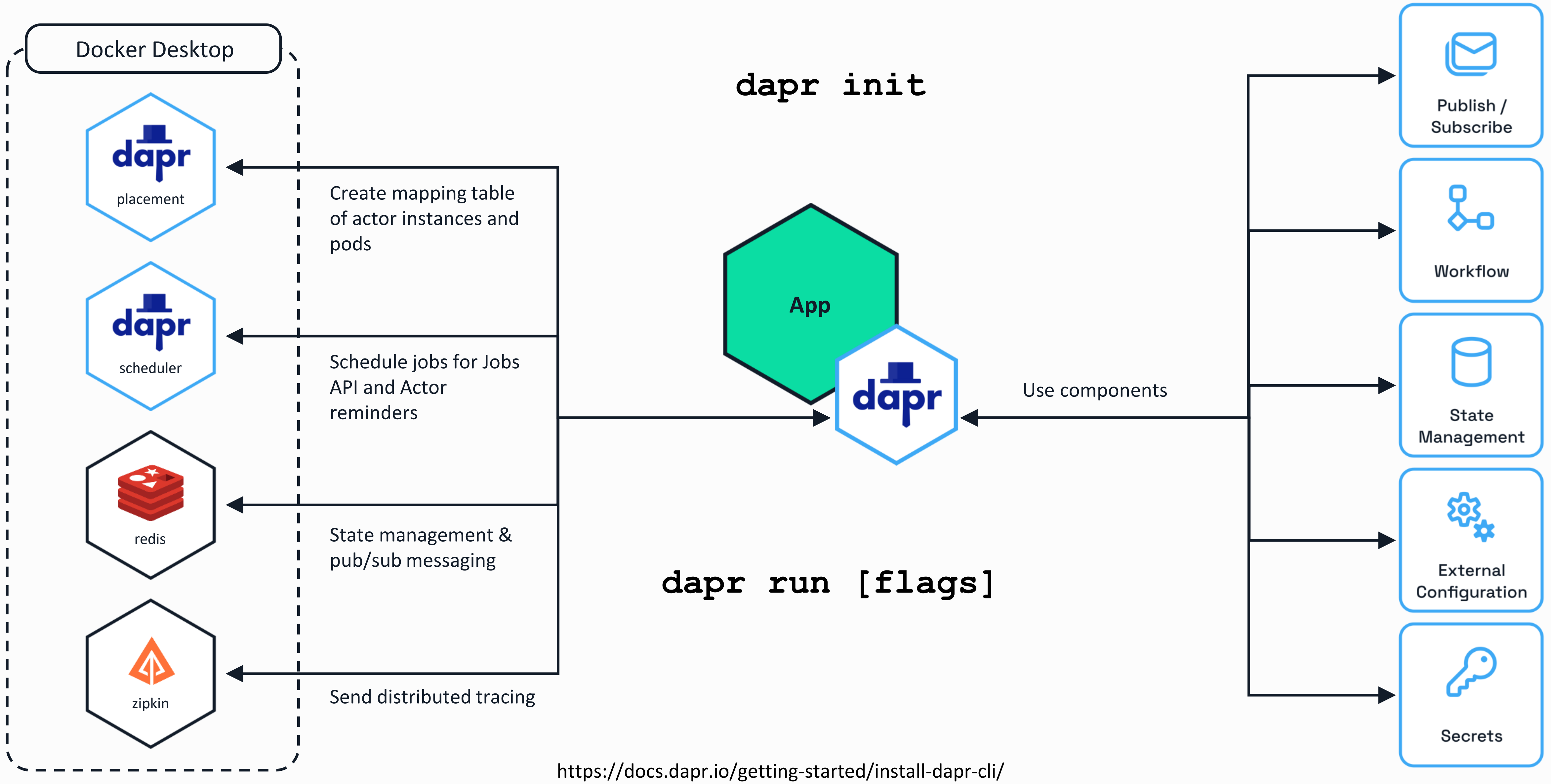
Hosting modes

Serverless

The Dapr side car is hosted by a provider.

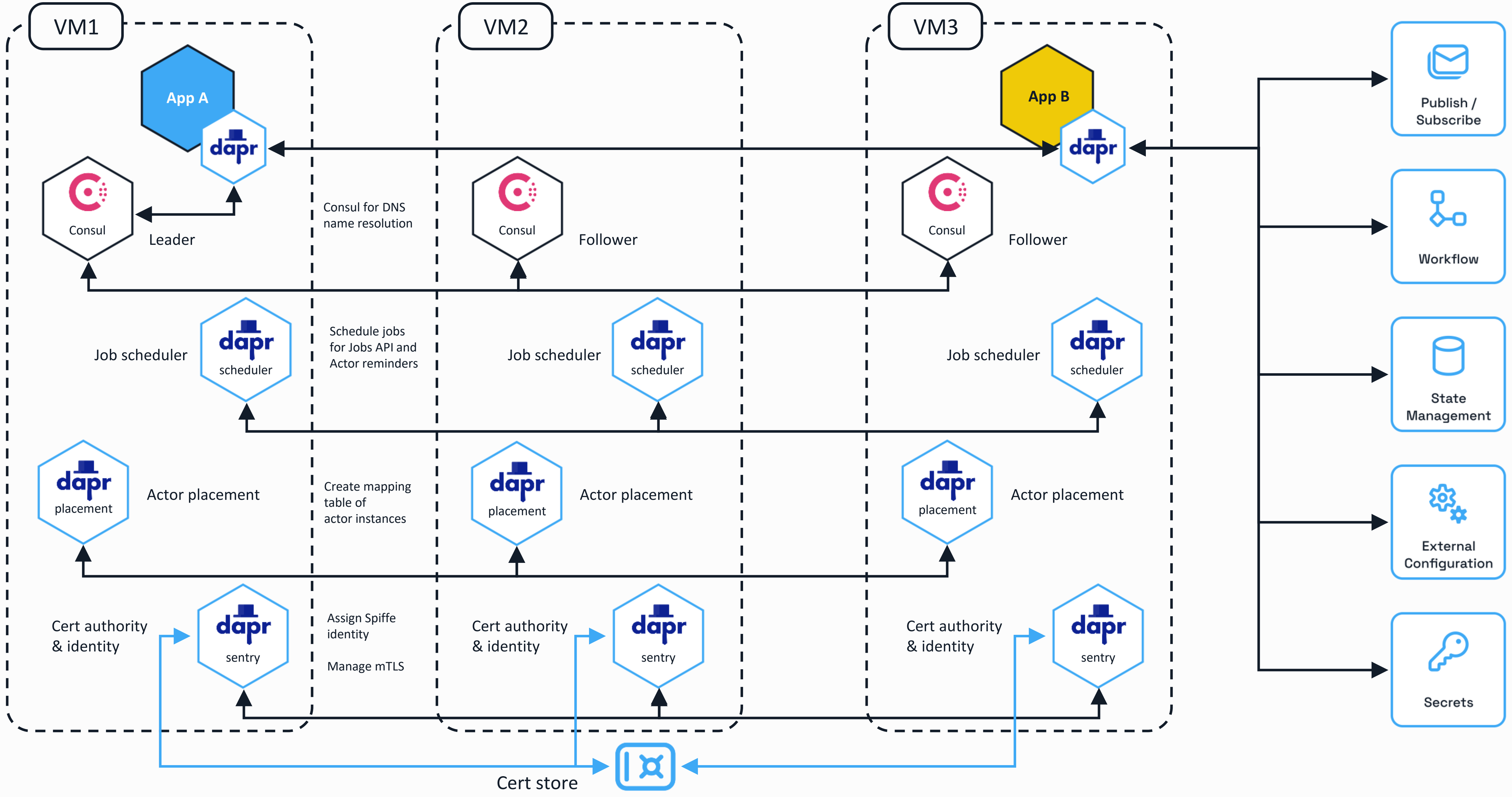
You only manage your applications.

Local development with the Dapr CLI

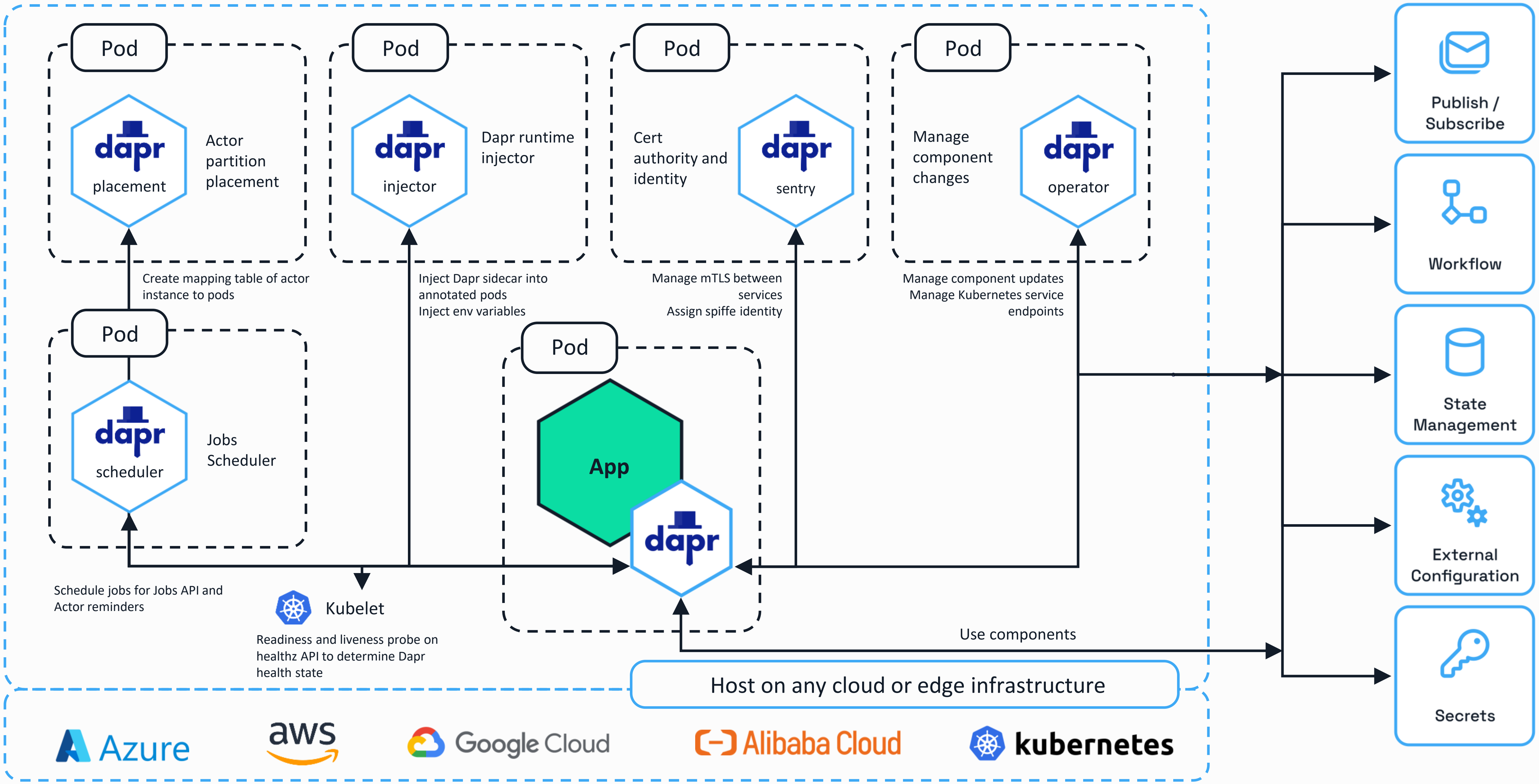


https://docs.dapr.io/getting-started/install-dapr-cli/

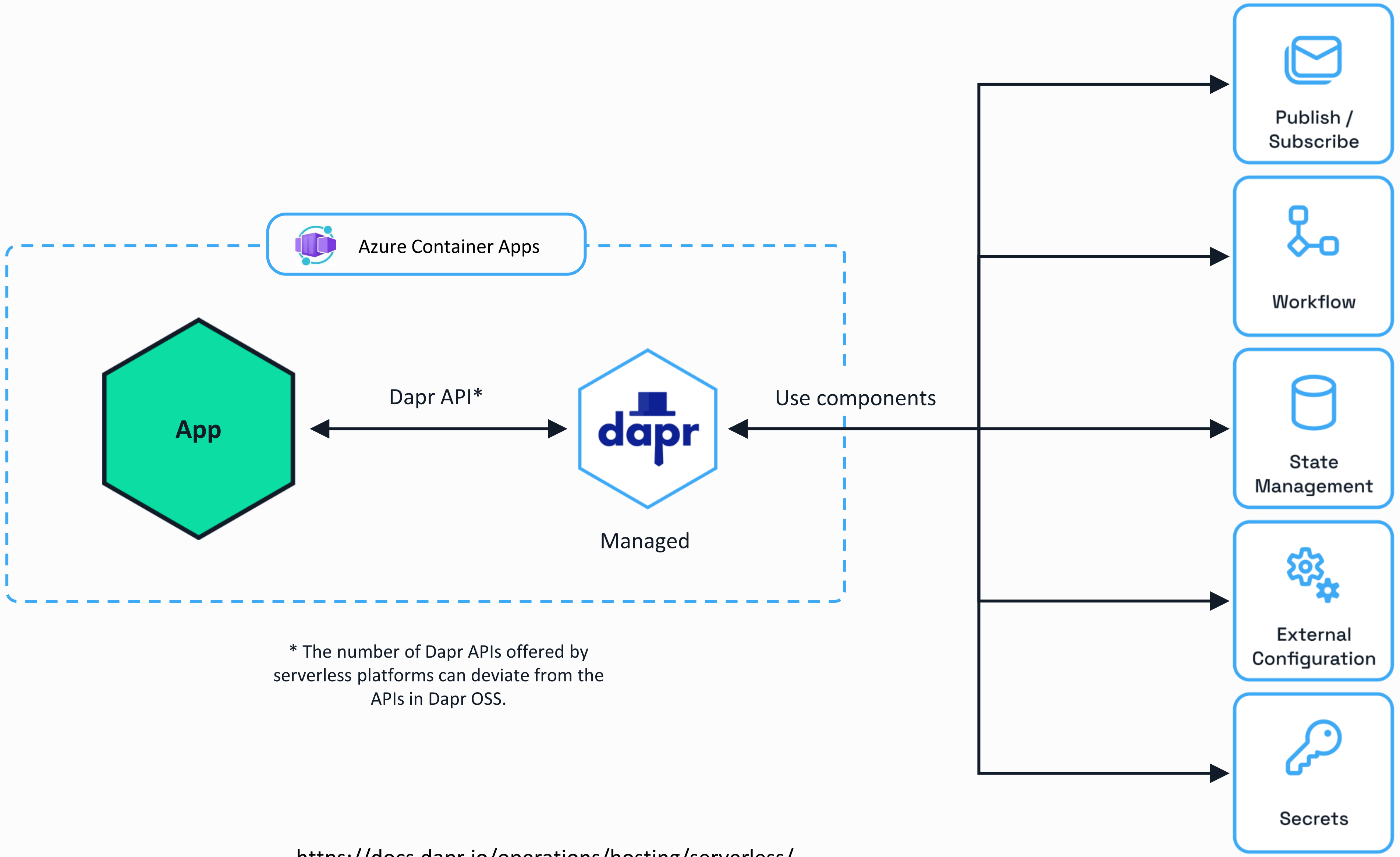
Dapr in self-hosted mode on VMs



Dapr on Kubernetes


Pod
dapr
placement
Actor partition placement
Pod
dapr
injector
Dapr runtime injector
Pod
Cert authority and identity
dapr
sentry
Pod
Manage component changes
dapr
operator
Create mapping table of actor instance to pods
Inject Dapr sidecar into annotated pods
Inject env variables
Manage mTLS between services
Assign spiffe identity
Manage component updates
Manage Kubernetes service endpoints
Pod
dapr
scheduler
Jobs Scheduler
Pod
App
dapr
Schedule jobs for Jobs API and Actor reminders
Kubelet
Readiness and liveness probe on healthz API to determine Dapr health state
Use components
Host on any cloud or edge infrastructure
Azure
aws
Google Cloud
Alibaba Cloud
kubernetes
Publish / Subscribe
Workflow
State Management
External Configuration
Secrets

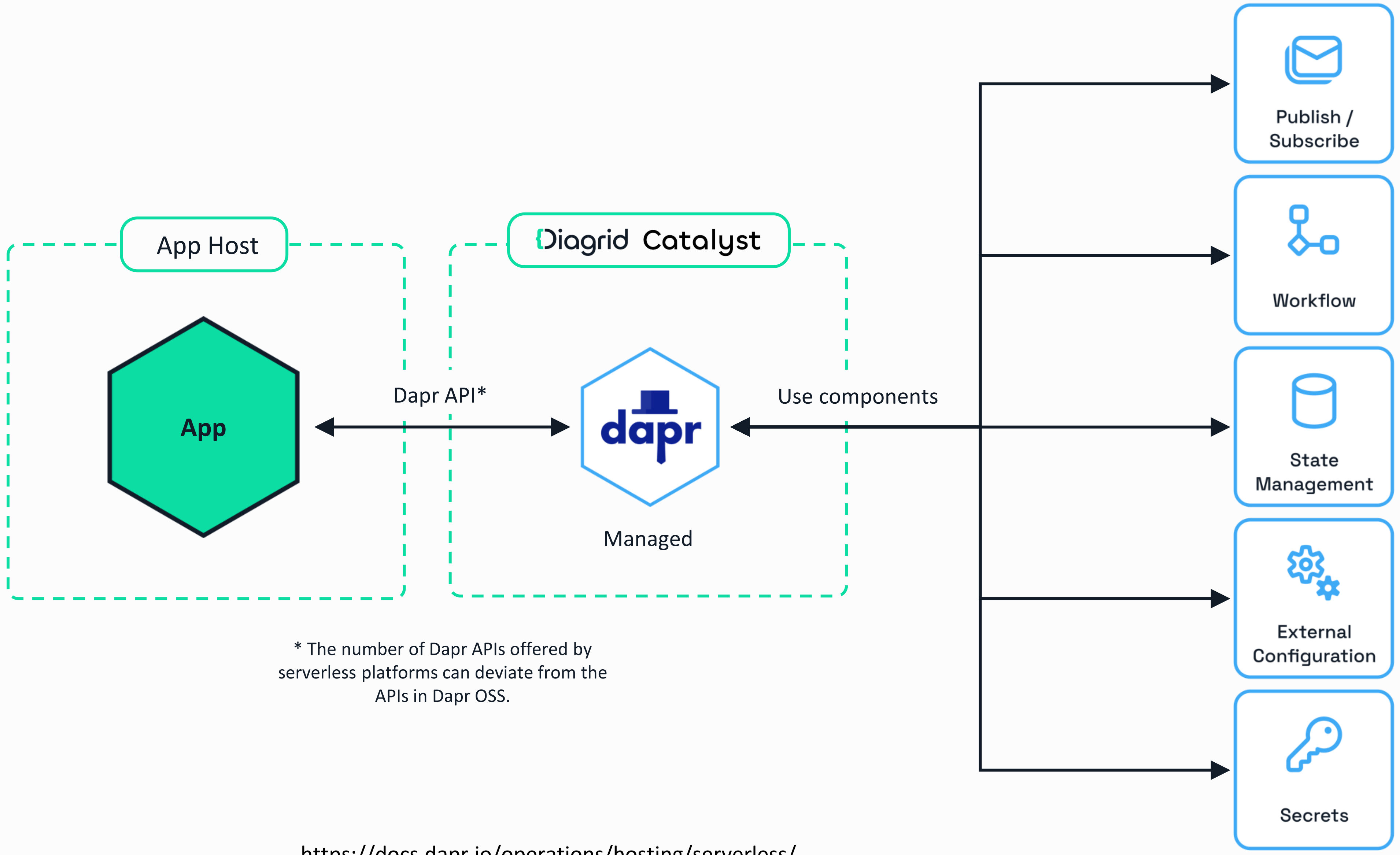
Serverless



* The number of Dapr APIs offered by serverless platforms can deviate from the APIs in Dapr OSS.

https://docs.dapr.io/operations/hosting/serverless/

Serverless



* The number of Dapr APIs offered by serverless platforms can deviate from the APIs in Dapr OSS.

https://docs.dapr.io/operations/hosting/serverless/

Dapr Resources

dapr.io

bit.ly/dapr-youtube

bit.ly/dapr-quickstarts

bit.ly/dapr-discord

@daprdev

@daprdev.bsky.social

dapr.io

Claim the Dapr Community Supporter badge!

bit.ly/dapr-supporter