



Start building
applications with
ease using
building block APIs



@marcdviker



dap^rr

≠





Marc Duiker
Sr Dev Advocate

 Diagrid

Azure MVP
Dapr Community
Manager

❤️ pixel art

@marcdijker

3





Dapr

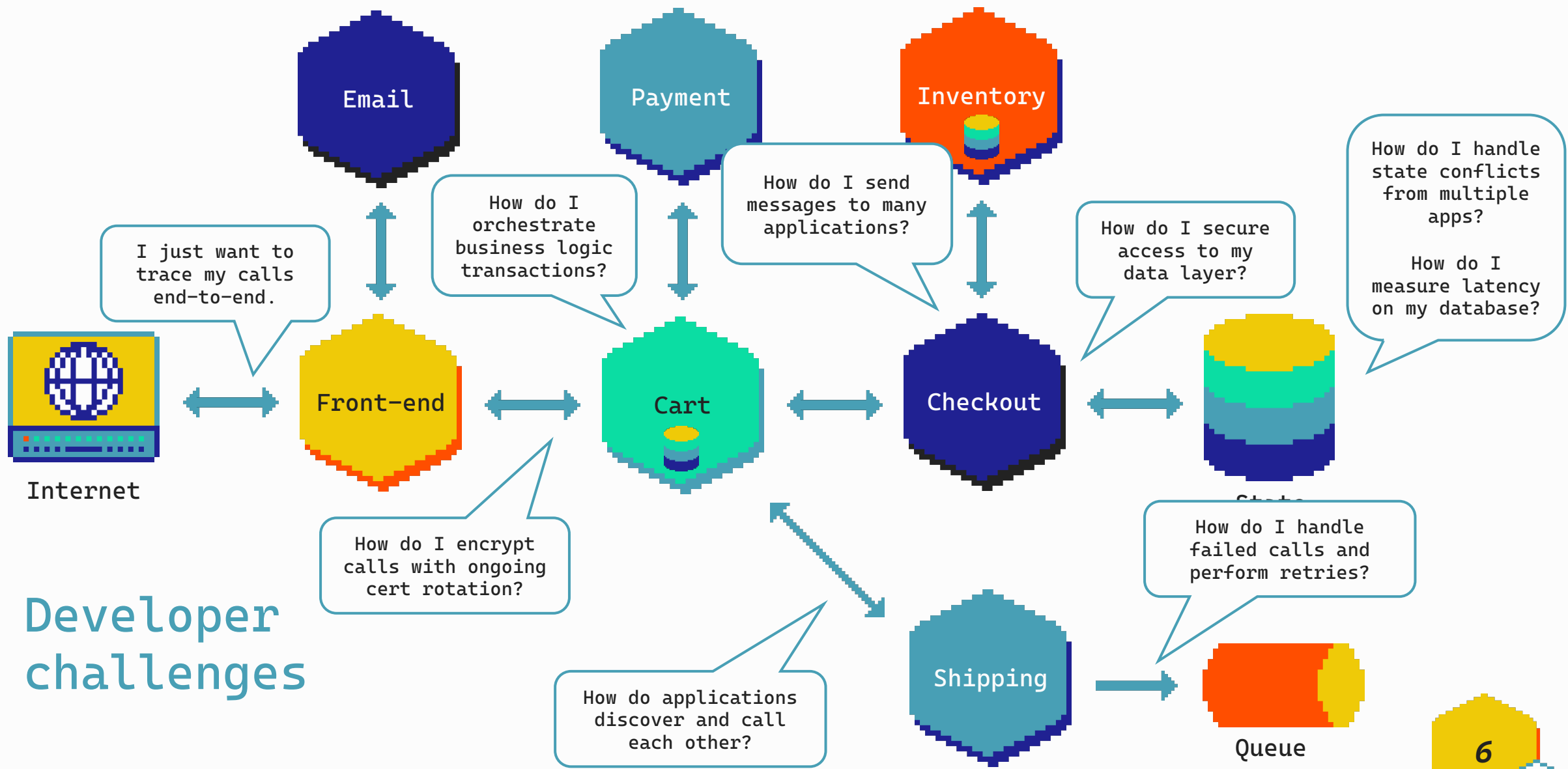
APIs

Components

Demos!

Q&A

Distributed apps



Developer challenges

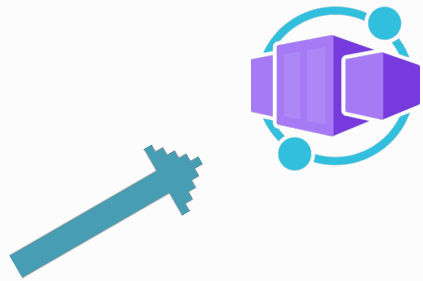
Distributed
application
runtime

dapr

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

Built-in **security**,
resiliency and **observability**
capabilities.

Dapr is a framework for
building distributed
applications across
cloud and edge.




 Microsoft Azure

 Google Cloud



 Alibaba Cloud

 **kubernetes**

 virtual or physical machines

Dapr project

**Submitted
to CNCF
Nov 2021**

**Incubation
maturity
level**

**10th largest
CNCF project**

App Definition and Development

Database **Streaming & Messaging** **Application Definition & Image Build** **Continuous Integration & Delivery**

Platform

Serverless

Members

CD Foundation Landscape

Observability and Analysis

Logging

Tracing

Chaos Engineering

Continuous Optimization

Automation & Configuration **Container Registry** **Security & Compliance** **Key Management**

Runtime

Cloud Native Storage **Container Runtime** **Cloud Native Network**

Scheduling & Orchestration **Coordination & Service Discovery** **Remote Procedure Call** **Service Proxy** **API Gateway** **Service Mesh**

Orchestration & Management

Kubernetes Certified Service Provider **Kubernetes Training Partner** **Certified CNFs**

@marcduiker

CLOUD NATIVE LANDSCAPE

CLOUD NATIVE
COMMITTEE FOR
PROMOTING
ADOPTION

l.cncf.io

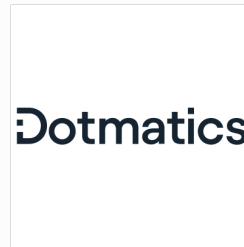
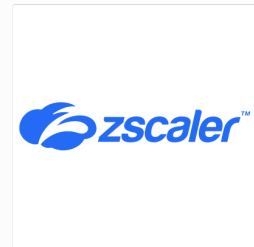
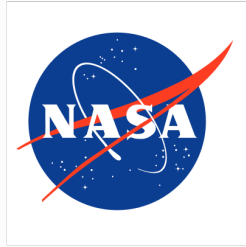
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

13

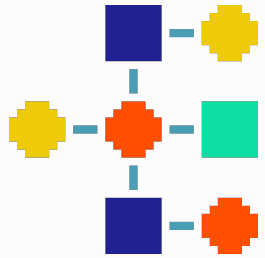
Contributing organizations



Dapr users



Building block APIs



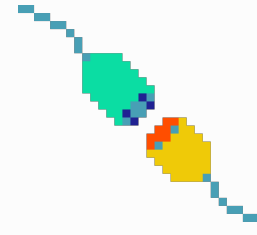
Service invocation



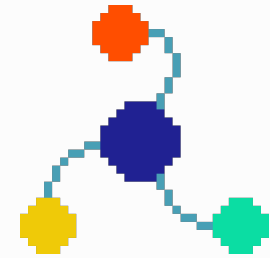
State Management



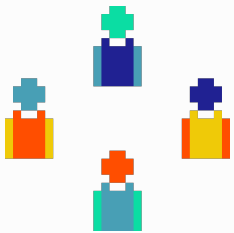
Publish & subscribe



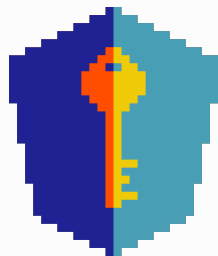
Bindings (input & output)



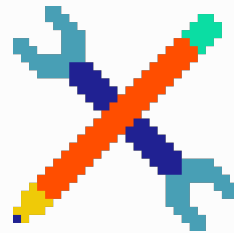
Observability



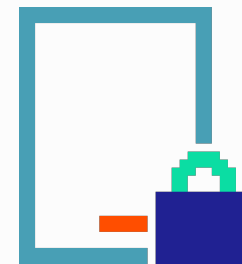
Actors



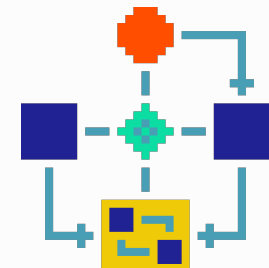
Secret Stores



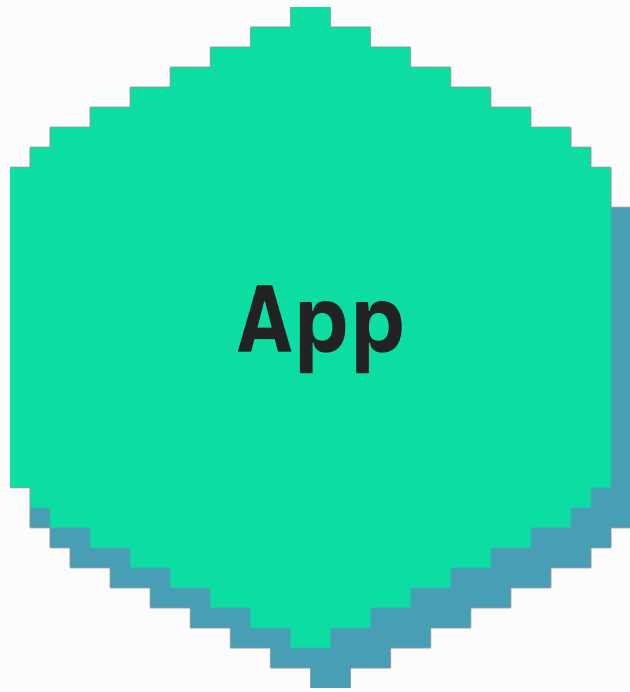
External Configuration



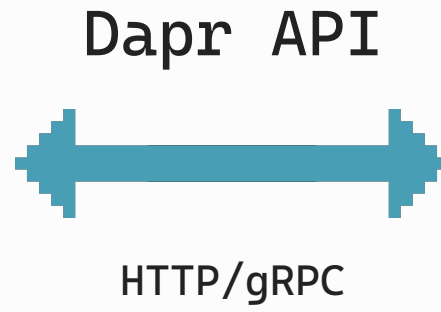
Distributed Lock



Workflows

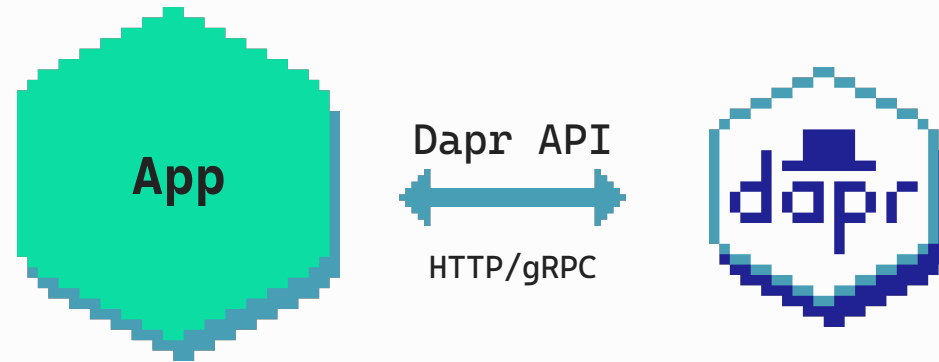


Application



Dapr sidecar





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

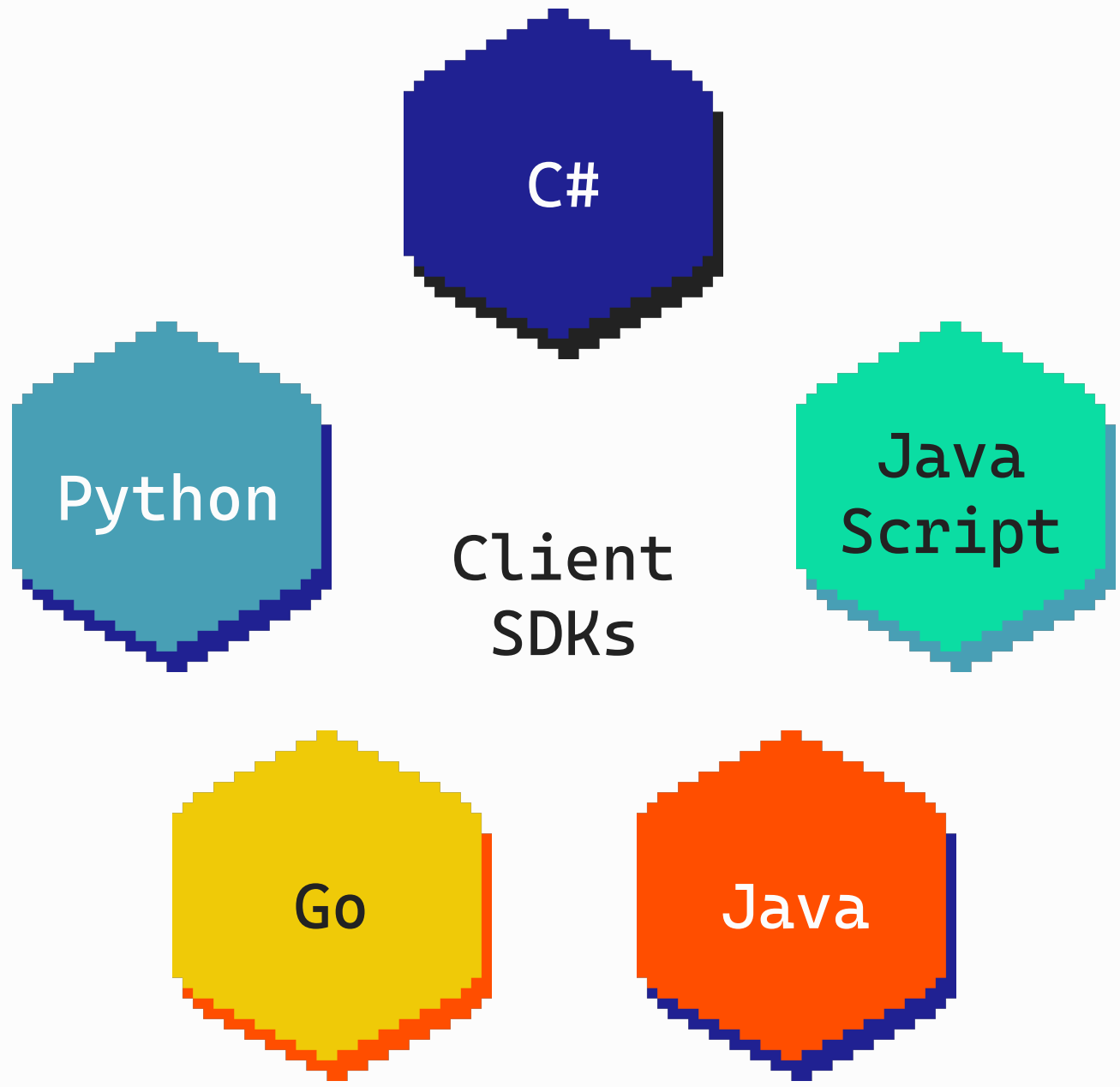
GET `http://localhost:3500/v1.0/state/inventory/item50`

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

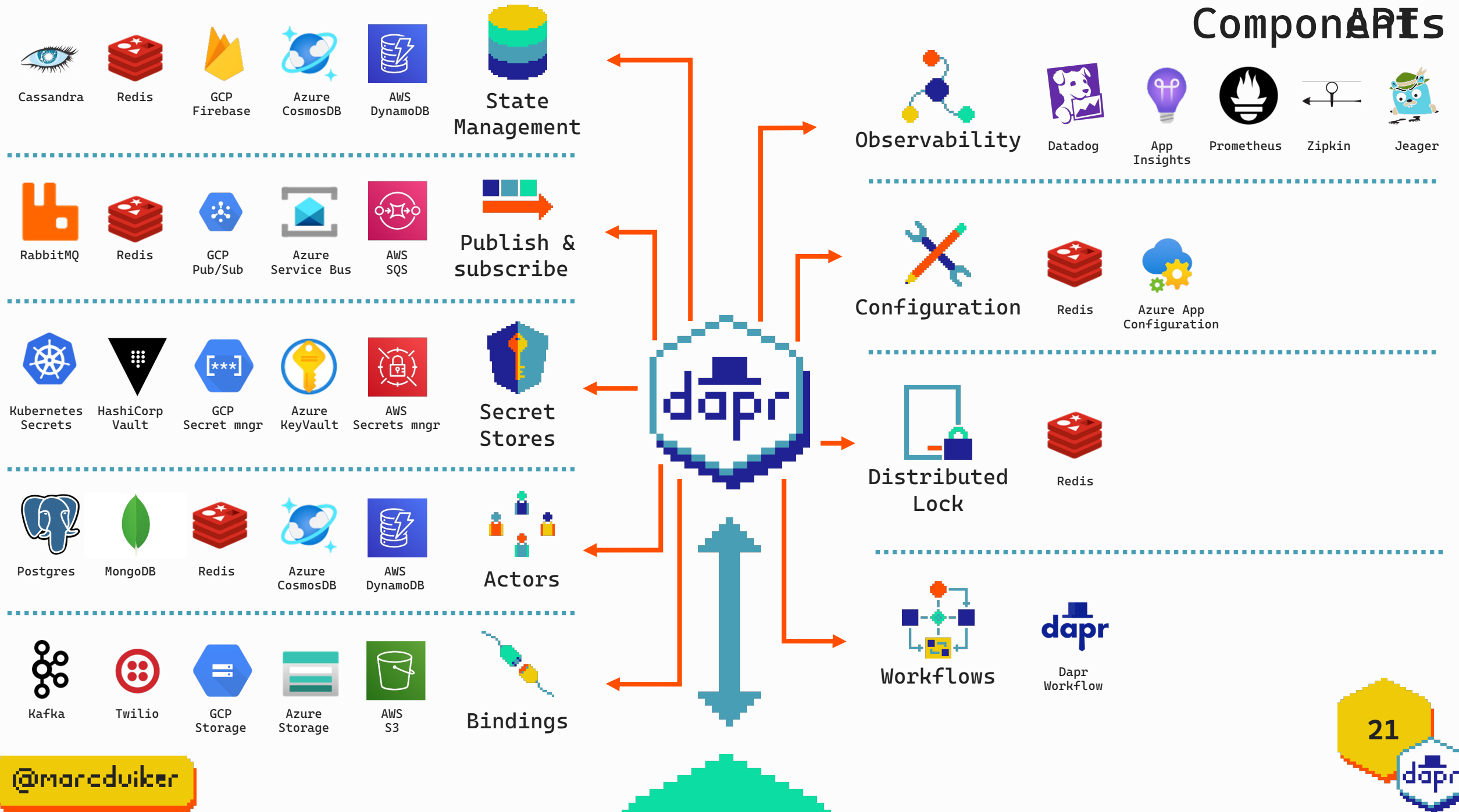
GET `http://localhost:3500/v1.0/secrets/vault/password42`

POST `http://localhost:3500/v1.0-beta1/workflows/dapr/businessprocess/start`





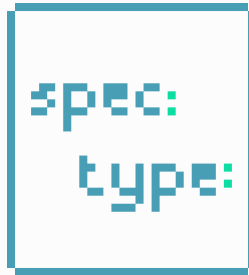
Component





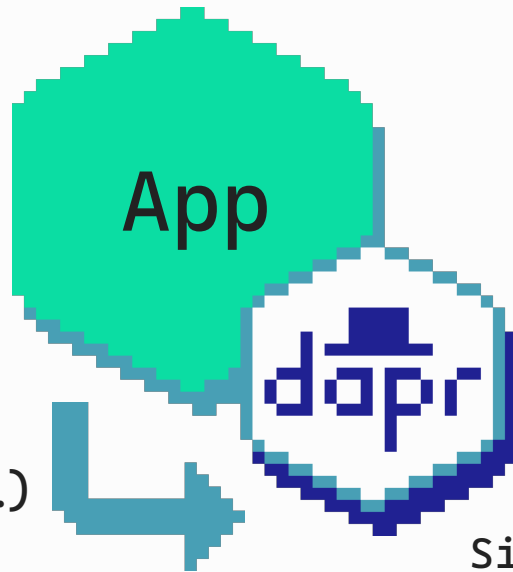
CosmosDB





yaml

```
metadata:  
  name: mystorage  
spec:  
  type: state.azure.cosmosdb  
metadata:  
  - name:  
    value:
```



```
SaveStateAsync(  
  "mystorage", ...)
```

```
GetStateAsync(  
  "mystorage", ...)
```

Sidecar



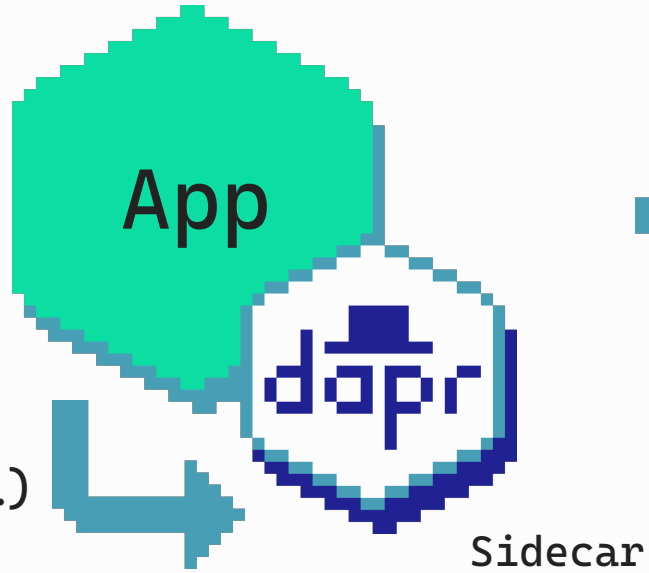
CosmosDB



```
spec:  
  type:
```

yaml

```
metadata:  
  name: mystorage  
spec:  
  type: state.redis  
metadata:  
  - name:  
    value:
```



```
SaveStateAsync(  
  "mystorage", ...)
```

```
GetStateAsync(  
  "mystorage", ...)
```



Redis



Dapr APIs are cloud agnostic.
Do they implement the lowest
common denominator?

Dapr APIs are **basic, by design.**
Dapr **adds a lot of**
functionality.

- Concepts
- Getting started
- Developing applications
- Operations
- Reference
 - Dapr API
 - Dapr CLI
 - Arguments and annotations
 - Environment variables
 - Component specs
 - State stores**
 - Aerospike
 - AWS
 - DynamoDB
 - Azure

Generic

Component	CRUD	Transactional	ETag	TTL	Actors	Query	Status	Component version
Aerospike	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Alpha	v1
Apache Cassandra	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stable	v1
CockroachDB	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Stable	v1
Couchbase	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Alpha	v1
Hashicorp Consul	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Alpha	v1
Hazelcast	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Alpha	v1
In-memory	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Developer-only	v1
JetStream KV	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Alpha	v1
Memcached	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stable	v1

[Edit this page](#)
[Create documentation issue](#)

[Concepts](#)[Getting started](#)[Developing applications](#)[Building blocks](#)[Service invocation](#)[State management](#)[Publish & subscribe](#)[Overview](#)[How to: Publish & subscribe to topics](#)[Messages with Cloudevents](#)[Developing applications](#) / [Building blocks](#) / [Publish & subscribe](#) / [Messages with Cloudevents](#)

Publishing & subscribing messages with Cloudevents

Learn why Dapr uses CloudEvents, how they work in Dapr pub/sub, and how to create CloudEvents.

To enable message routing and provide additional context with each message, Dapr uses the [CloudEvents 1.0 specification](#) as its message format. Any message sent by an application to a topic using Dapr is automatically wrapped in a CloudEvents envelope, using the [Content-Type header value](#) for `datacontenttype` attribute.

Dapr uses CloudEvents to provide additional context to the event payload, enabling features like:

- Tracing
- Deduplication by message Id
- Content-type for proper deserialization of event data

[Concepts](#)[Getting started](#)[Developing applications](#)[Operations](#)[Observability](#)[Hosting options](#)[Configuration](#)[Components](#)[Security](#)[Resiliency](#)**Overview**[Policies](#)[Targets](#)[Support](#)[Performance and scalability](#)[Troubleshooting](#)[Reference](#)[Operations](#) / [Resiliency](#) / [Overview](#)

Overview

Configure Dapr retries, timeouts, and circuit breakers

Dapr provides a capability for defining and applying fault tolerance resiliency policies via a [resiliency spec](#). Resiliency specs are saved in the same location as components specs and are applied when the Dapr sidecar starts. The sidecar determines how to apply resiliency policies to your Dapr API calls. In self-hosted mode, the resiliency spec must be named `resiliency.yaml`. In Kubernetes Dapr finds the named resiliency specs used by your application. Within the resiliency spec, you can define policies for popular resiliency patterns, such as:

- [Timeouts](#)
- [Retries/back-offs](#)
- [Circuit breakers](#)

Policies can then be applied to [targets](#), which include:

- [Apps](#) via service invocation
- [Components](#)
- [Actors](#)

Concepts

- Overview
- Building blocks
- Components
- Configuration
- Resiliency
- Observability

Security

- Dapr services**
- Service meshes
- Terminology
- FAQs

- Getting started
- Developing applications
- Operations

[Concepts](#) / [Security](#)

Security

How Dapr is designed with security in mind

Security is fundamental to Dapr. This article describes the security features and capabilities when using Dapr in a distributed application. These can be divided into:

- Secure communication with service invocation and pub/sub APIs.
- Security policies on components and applied through configuration.
- Operational security practices.
- State security, focusing on data at rest.

An example application is used to illustrate many of the security features available in Dapr.

Secure communication

Dapr provides end-to-end security with the service invocation API, with the

Integrated APIs with
flexibility at the *component*
level.

When to use Dapr?

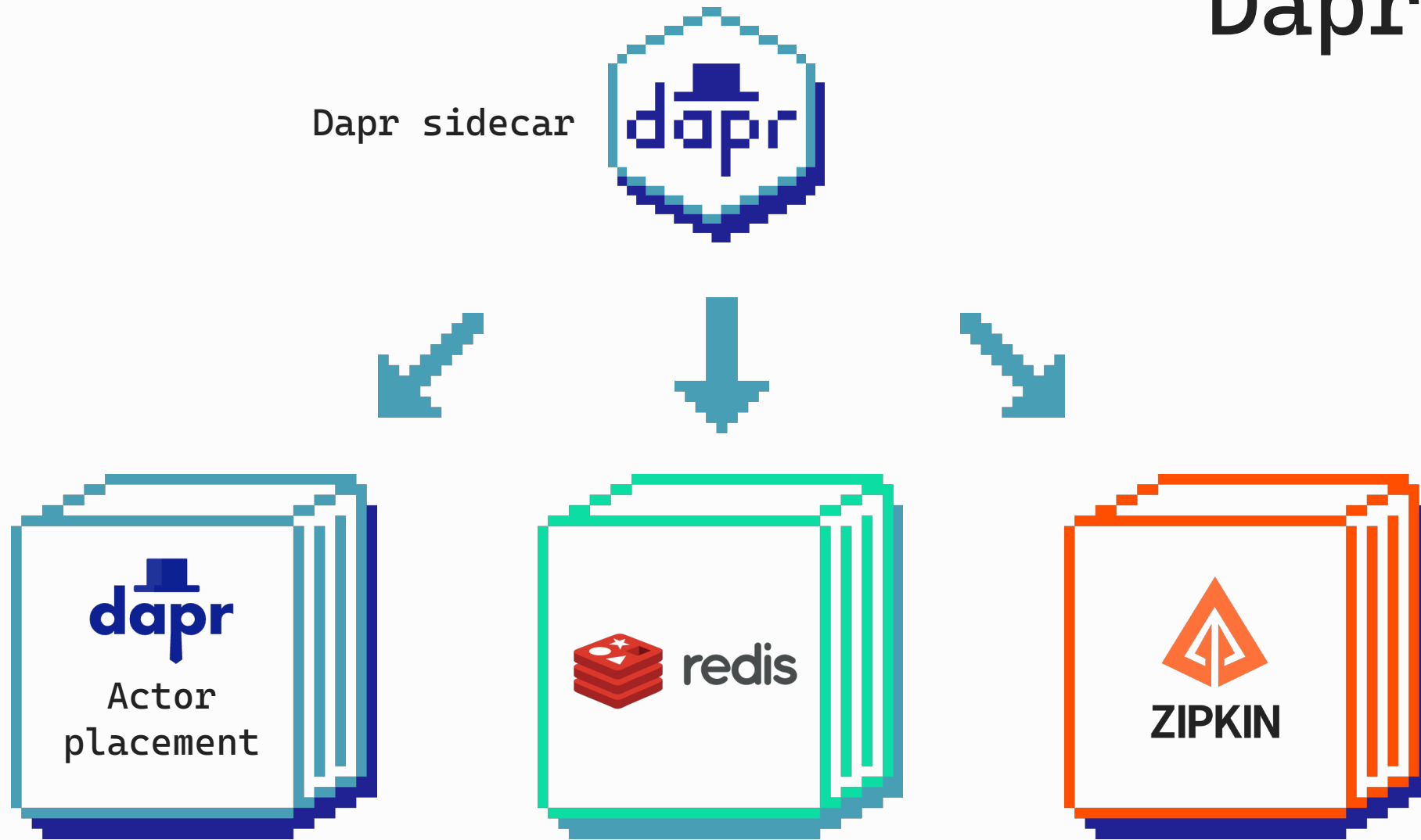
You are in a large organisation
with many different teams or
tech stacks.

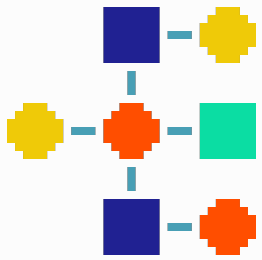
Greenfield projects: postpone
your architecture decisions.

API demos

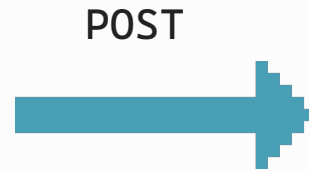
github.com/dapr/quickstarts

Dapr CLI



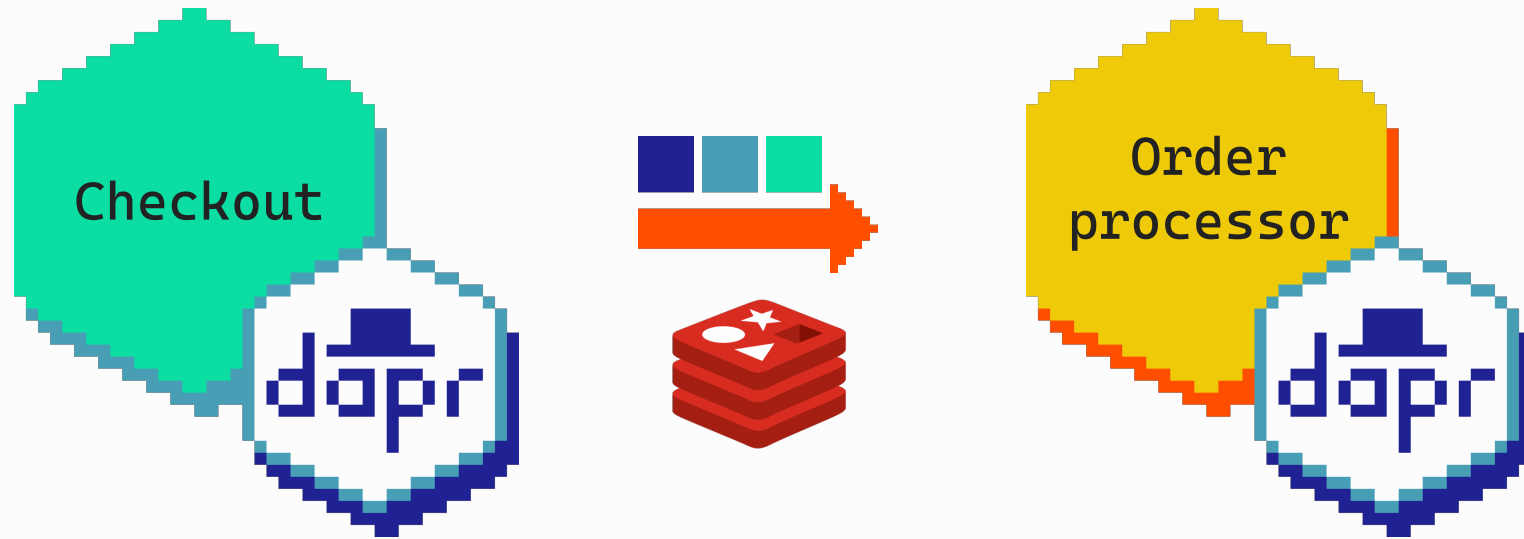


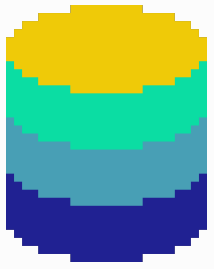
Service invocation



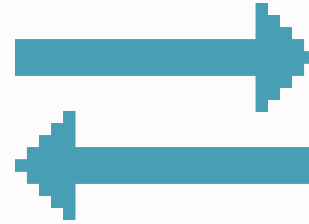
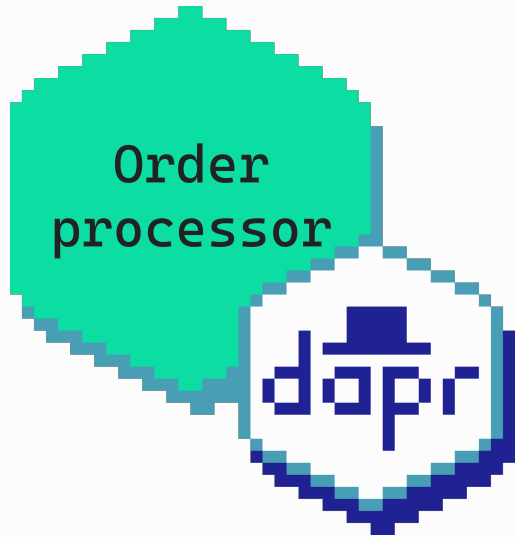


Publish &
subscribe



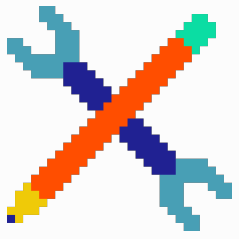


State
Management

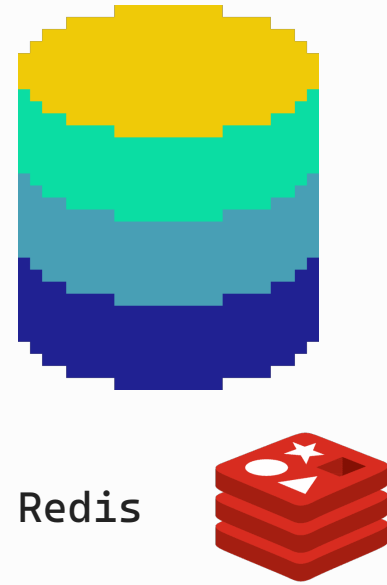
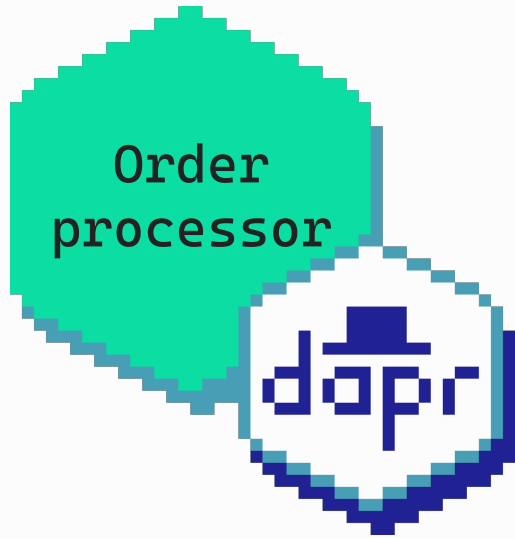


Redis





External
Configuration





<https://pages.diagrid.io/download-the-state-of-dapr-2023-report>



Claim this Holopin badge!



github.com/dapr/quickstarts



Join the Dapr Discord!

bit.ly/dapr-discord

