Microsoft Azure

Dapr + Orleans

Aumente sua caixa de ferramentas com essas duas tecnologias fantásticas!

George Luiz Bittencourt

Sobre mim

SCAN ME

George Luiz Bittencourt

Arquiteto de soluções cloud com mais de 20 anos de experiência em infraestrutura e desenvolvimento de software.

/glzbcrt

/glzbcrt

george.bittencourt@microsoft.com

Orleans

Overview

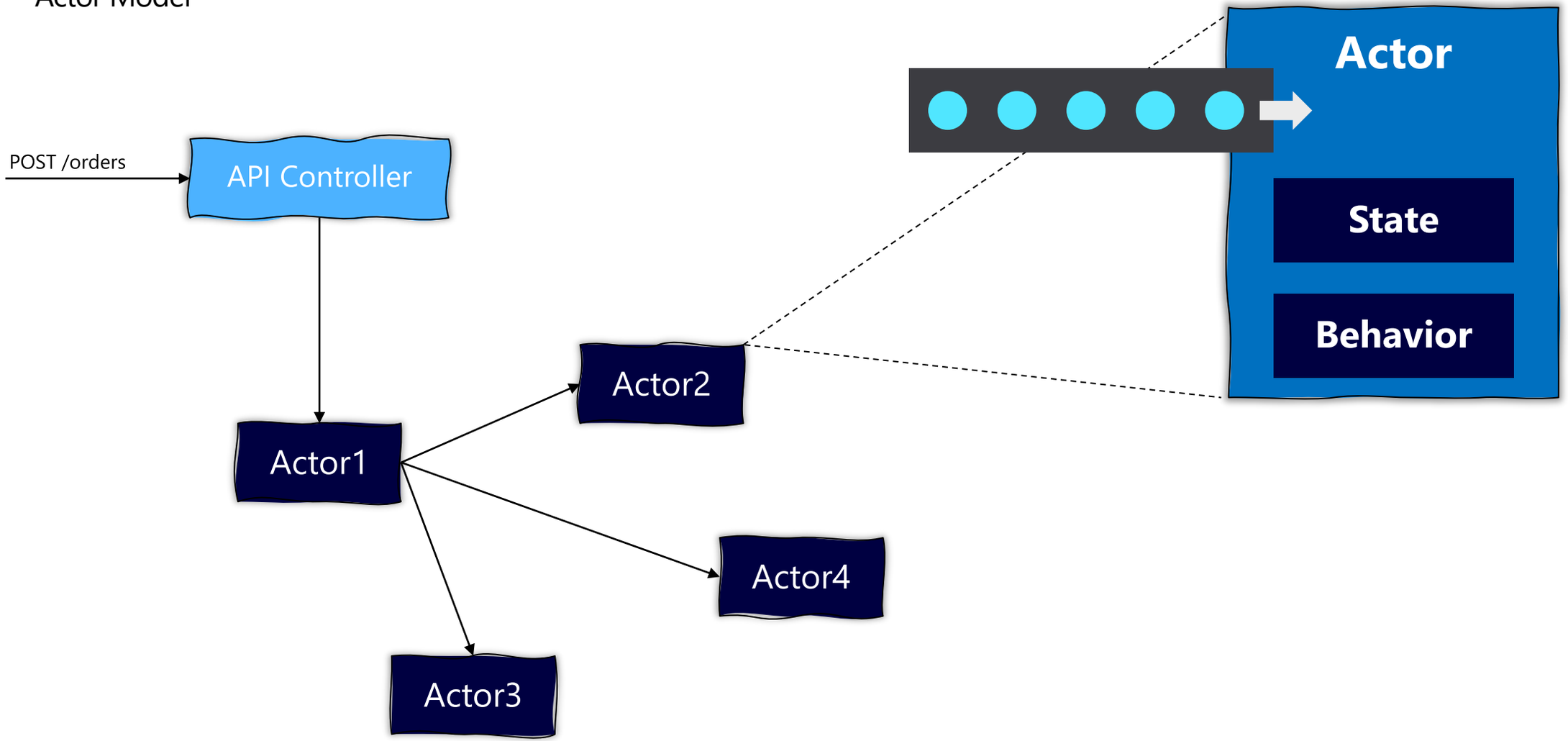
- Framework *cross-plataform* para o desenvolvimento de aplicações distribuídas em .NET.
- É desenvolvido de maneira *open-source* no GitHub sob a licença MIT e tem mais de 9.000 estrelas.
- Conhecido também como **Distributed .NET**.
- É escalável para centenas de milhares de servidores na nuvem.
- Pode ser utilizado em Kubernetes, máquinas virtuais ou ainda serviços PaaS como Azure App Service, Azure Container Apps, etc.
- Pode ser utilizado em outras nuvens ou ainda no ambiente *on-premisses*.
- É disponibilizado através de um conjunto de pacotes NuGet.
- É um concorrente do Akka.NET ou ainda do Akka para Java.

Orleans

Quem está usando?

Orleans

Actor Model



Orleans

Actor Model → Grains

Orleans

Criando um Grain

Tipos de identidades disponíveis:

- *IGrainWithGuidKey*
- *IGrainWithIntegerKey*
- *IGrainWithStringKey*
- *IGrainWithGuidCompoundKey*
- *IGrainWithIntegerCompoundKey*

```
public interface IUrlShortenerGrain : IGrainWithStringKey
{
    2 references
    Task SetUrl(string fullUrl);

    2 references
    Task<string> GetUrl();
}
```

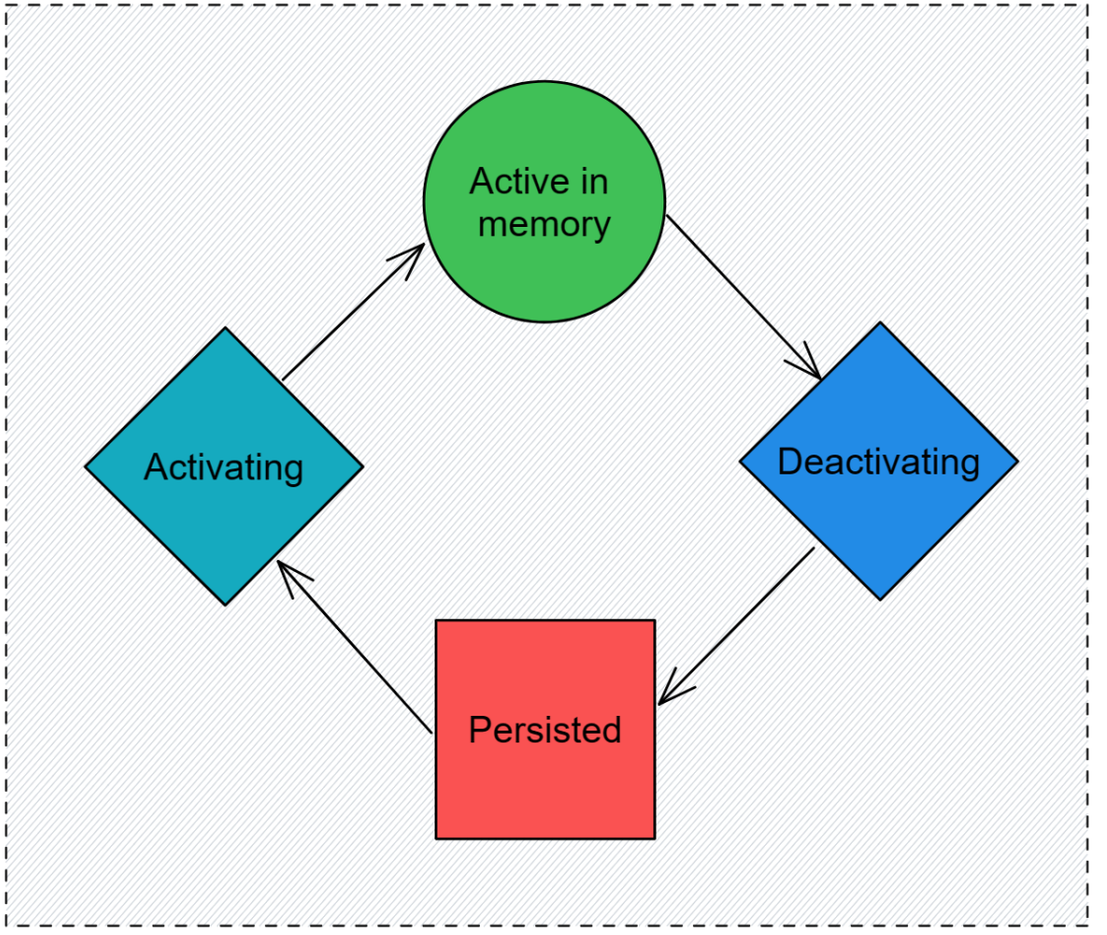
```
public sealed class UrlShortenerGrain : Grain, IUrlShortenerGrain
{
    private readonly IPersistentState<UrlDetails> _state;

    0 references
    public UrlShortenerGrain(
        [PersistentState(
        stateName: "url",
        storageName: "urls")]
        IPersistentState<UrlDetails> state) => _state = state;

    2 references
    public async Task SetUrl(string fullUrl)
    {
        _state.State = new()
        {
            ShortenedRouteSegment = this.GetPrimaryKeyString(),
            FullUrl = fullUrl
        };

        await _state.WriteStateAsync();
    }

    2 references
    public Task<string> GetUrl() =>
        Task.FromResult(_state.State.FullUrl);
}
```

grain base class

Orleans

Grain - State

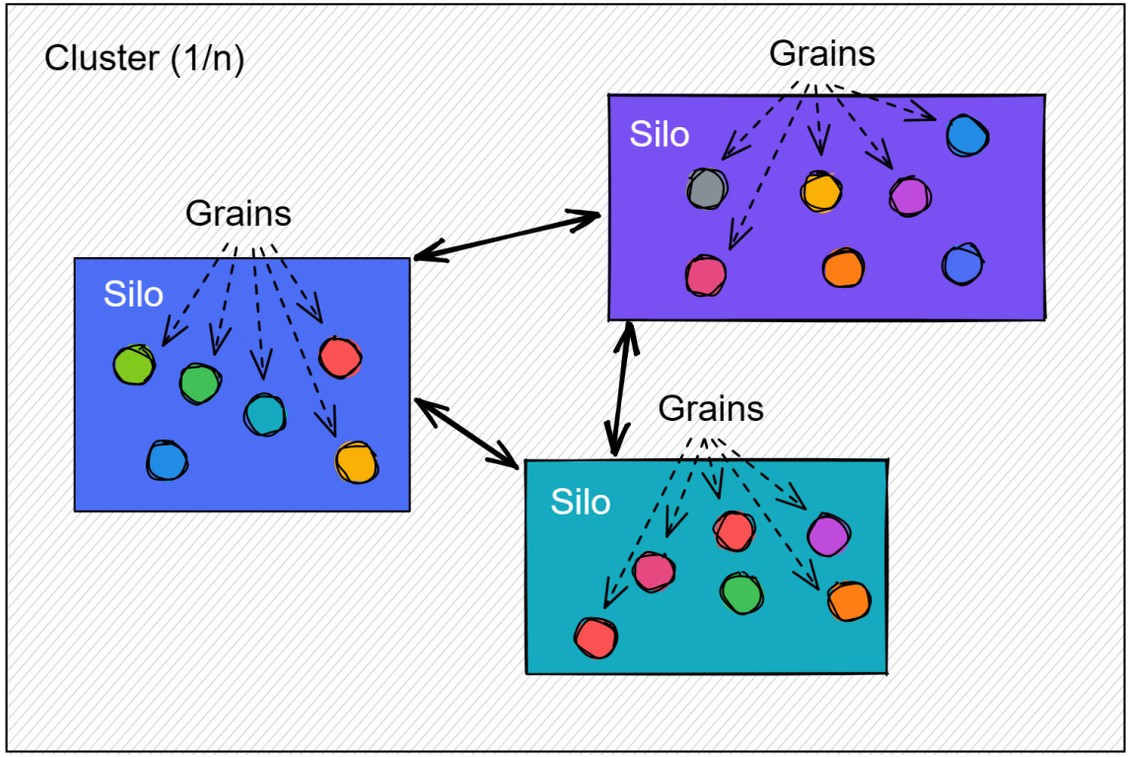
- Grains podem ser voláteis ou persistentes.
- O grain é mantido em memória enquanto estiver em uso.
- A criação do grain é feita de forma automática sob demanda a partir da camada de persistência adotada.
- Grains não utilizados são removidos da memória, liberando recursos.
- A interface de armazenamento é extensível e hoje existe suporte para:
  - Azure Blob, Table e Cosmos DB
  - ADO.NET
  - AWS Dynamo DB
- Alterações não são automaticamente gravadas. É necessário executar o método WriteStateAsync.



Orleans

Silo

- Um silo é outro conceito fundamental do Orleans.
- Um silo pode hospedar um ou mais grains.
- Um grupo de silos é chamado de cluster.
- Grains interagem entre si trocando mensagens.



Orleans

Benefícios

- É um framework já testado em produção com grandes volumes e por ser *open-source* não tem *lock-in*.
- Aumenta a produtividade do desenvolvedor, já que muita coisa já está desenvolvida, testada e é somente necessário utilizar.
- Utiliza o paradigma orientada a objetos, já amplamente conhecido e utilizado.
- Não existem problemas de concorrência no nível do grain, pois o Orleans se encarrega de gerenciar isso. O mesmo objeto nunca é acessado ao mesmo tempo por mais de uma thread.
- O grain somente é ativado quando necessário e removido quando não estiver mais em uso.
- A localização do grain é transparente. É criada uma classe proxy para isso.
- Escalabilidade transparente.
- A programação assíncrona é mandatória e força o programador a desenvolvedor código *non-blocking* aumentando assim a escalabilidade da solução.

Microsoft Azure

DEMO

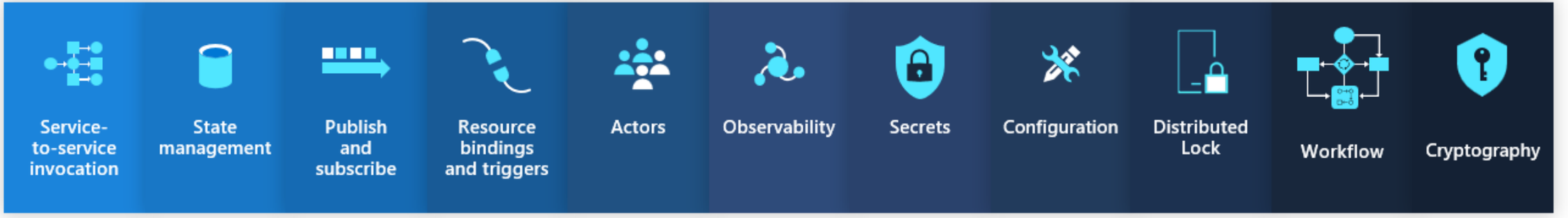
Dapr

Overview

- Sigla: **D**istributed **Ap**plication **R**untime
- Projeto *open-source* criado pela Microsoft com licença Apache e mais de 21 mil estrelas.
- É um projeto incubado na CNCF.
- Adota o pattern de *side-car*.
- Contém onze *building blocks* que podem ser utilizados para os mais diversos fins.
- Sua API pode ser utilizada tanto via HTTP quanto via gRPC.

Dapr

Building Blocks

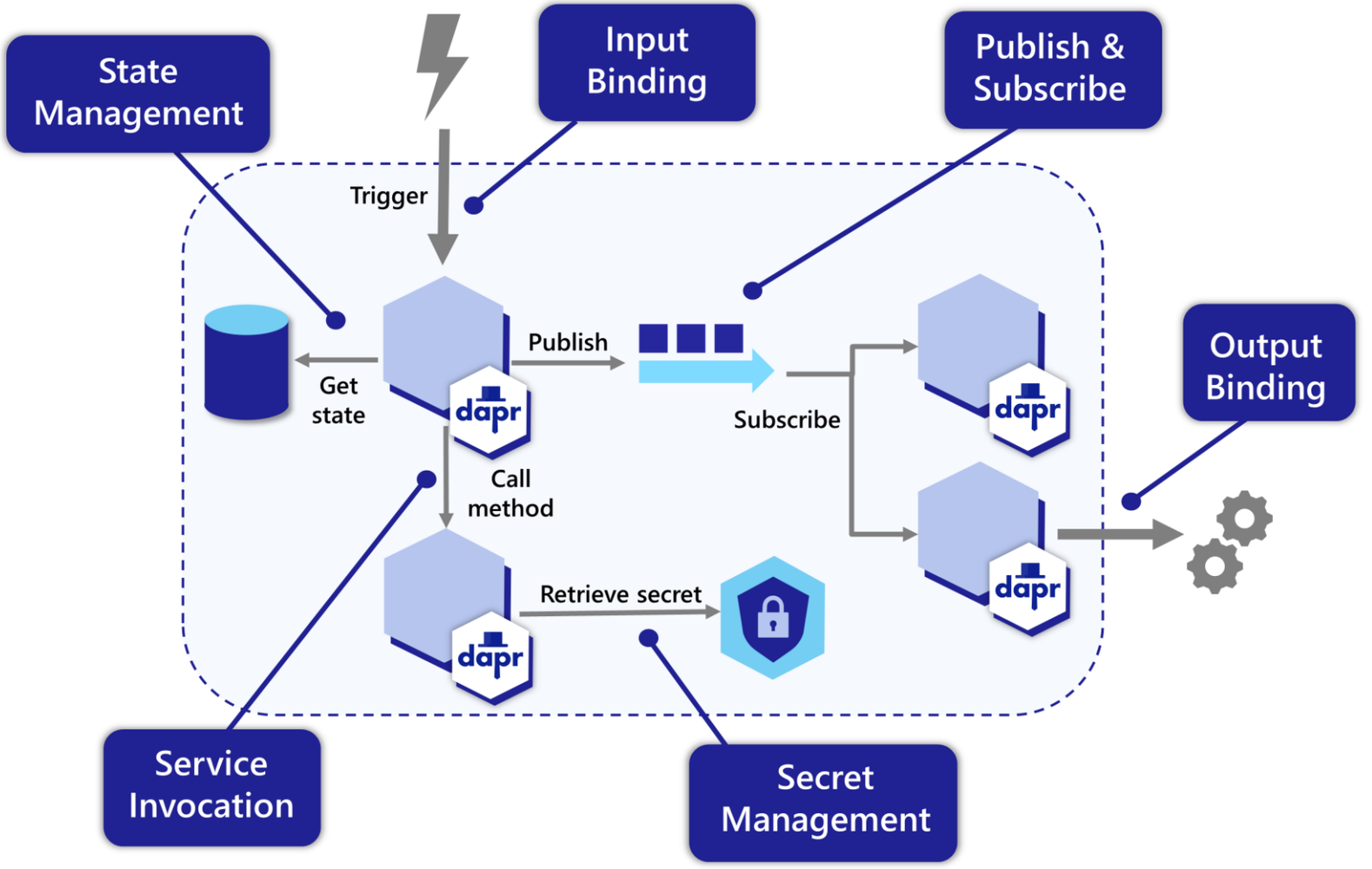


Microsoft Azure

DEMO

Dapr

Overview

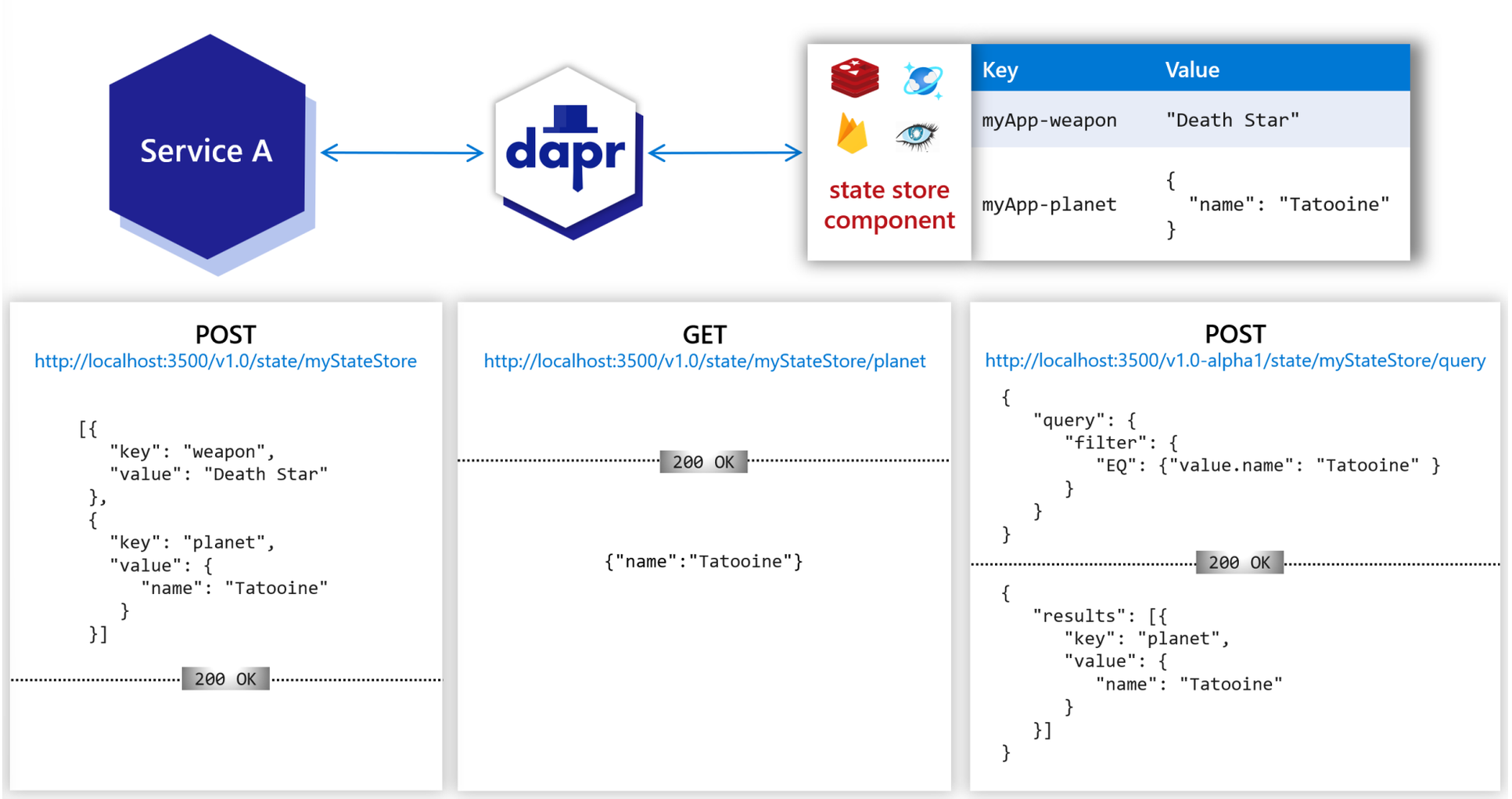


Dapr

SDKs

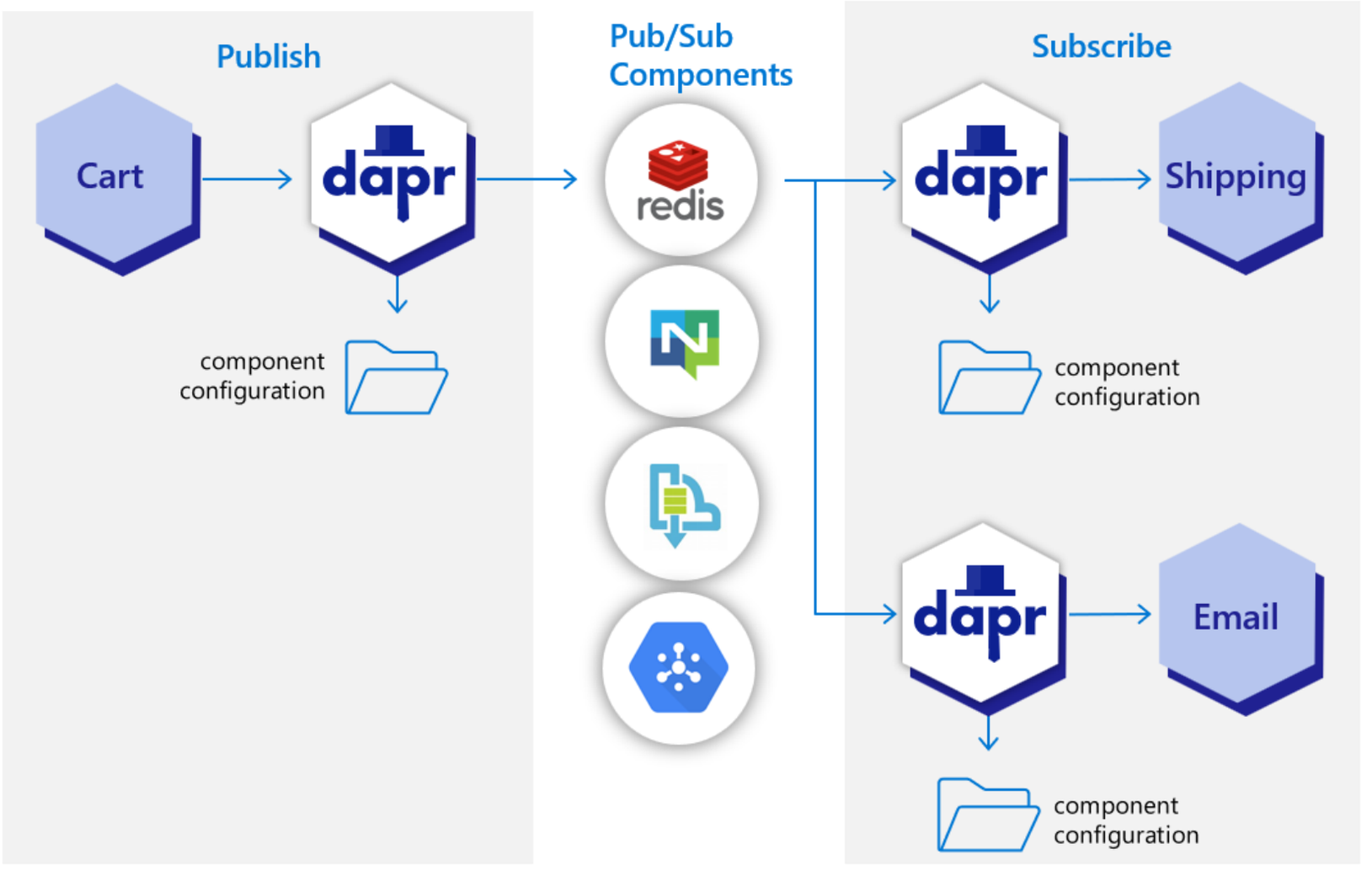
Dapr

Building Blocks – State Managment



Dapr

Building Blocks – Pub/Sub



Dapr

Componentes

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: events
  namespace: reactor
spec:
  type: pubsub.azure.servicebus.topics
  version: v1
  metadata:
  - name: connectionString
    value: "Endpoint=sb://glzbcrt.servicebus.windows.net/;SharedAccessKeyName=send;SharedAccessKey=n4Em4S<REDACTED>bG8N5kM=;EntityPath=customerevents"
---
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: customer
  namespace: reactor
spec:
  type: state.azure.cosmosdb
  version: v1
  metadata:
  - name: url
    value: https://reactorg.documents.azure.com:443/
  - name: masterKey
    value: gXsLozWKwp<REDACTED>jzFiD6gACDbVXFaAQ==
  - name: database
    value: reactor
  - name: collection
    value: customer
---
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: order
  namespace: reactor
spec:
  type: state.azure.cosmosdb
  version: v1
  metadata:
  - name: url
    value: https://reactorg.documents.azure.com:443/
  - name: masterKey
    value: gXsLozWKwpSlTXGRihvF1h<REDACTED>goajzFiD6gACDbVXFaAQ==
  - name: database
    value: reactor
  - name: collection
    value: order
```

Microsoft Azure

DEMO

Microsoft Azure

EOF

/glzbcrt

/glzbcrt

george.bittencourt@microsoft.com

SCAN ME