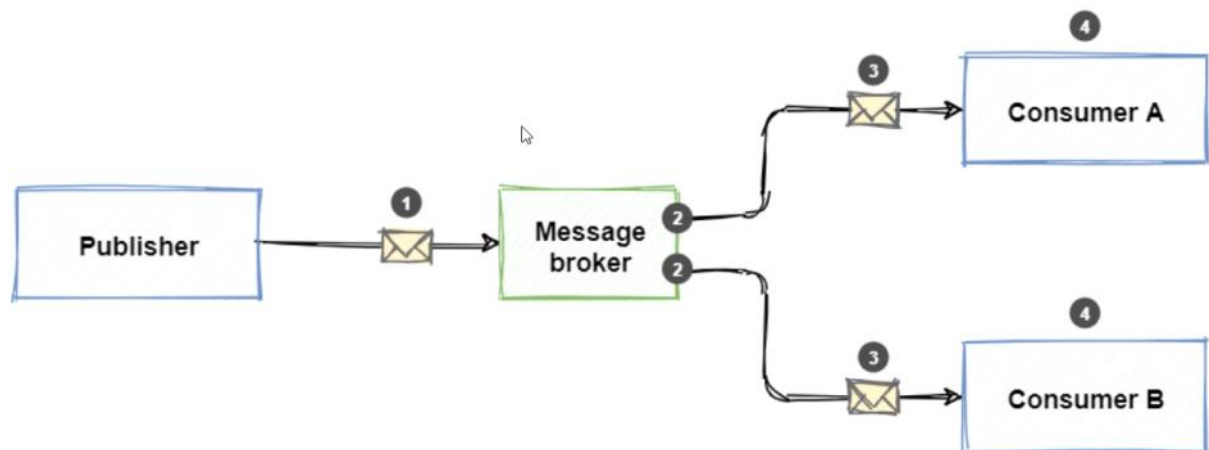


第08期-发布和订阅构建块

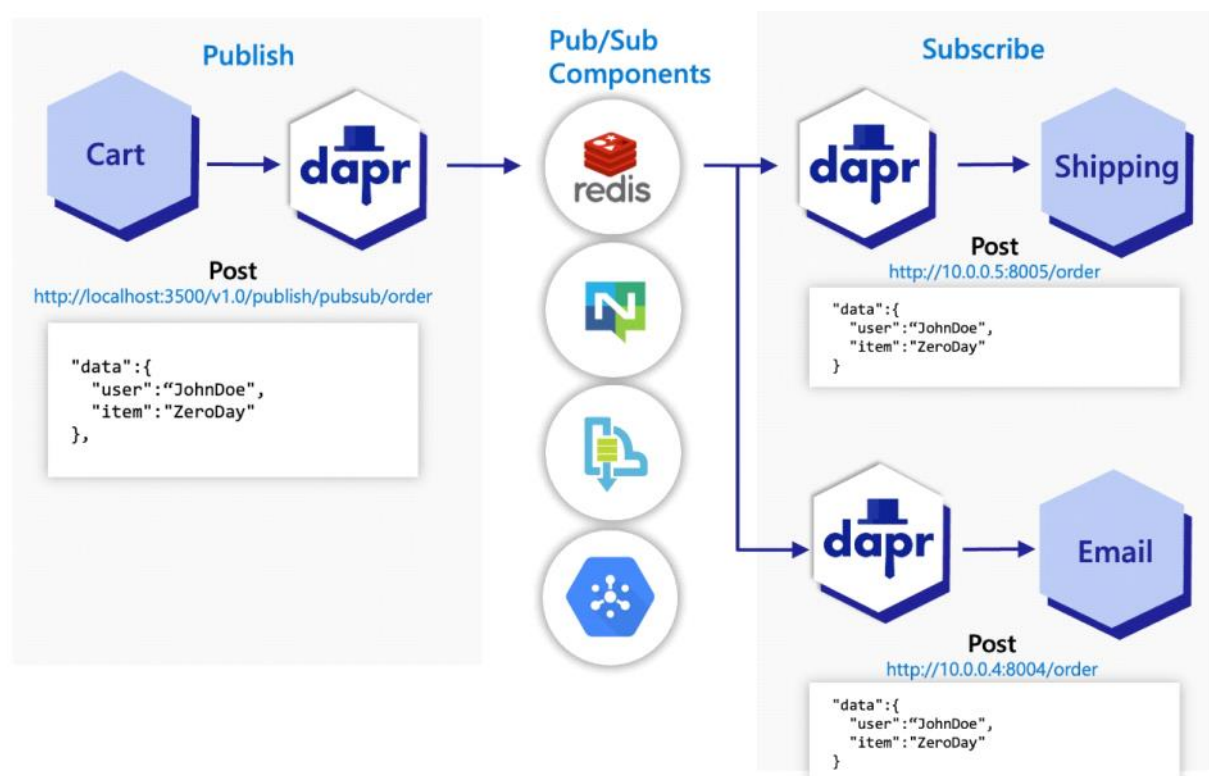
2022年7月14日 10:20

发布和订阅



1. 发布服务器将消息发送到消息代理。
2. 订阅服务器绑定到消息代理上的订阅。
3. 消息代理将消息的副本转发给相关的订阅。
4. 订阅服务器从其订阅使用消息。

Dapr 中的发布与订阅



事件消息格式

Dapr 使用 CloudEvents 1.0 规范作为消息格式，CloudEvents 是一种以通用格式描述事件数据的规范，以提供跨服务、平台和系统的互操作性。

```
{
  "specversion" : "1.0",
  "type" : "xml.message",
  "source" : "https://example.com/message",
  "subject" : "Test XML Message",
  "id" : "id-1234-5678-9101",
  "time" : "2020-09-23T06:23:21Z",
  "datacontenttype" : "text/xml",
  "data" : "<note><to>User1</to><from>user2</from><message>hi</message>
</note>"
}
```

消费者群体和竞争性消费者模式

当同一个应用程序的多个实例(相同的 ID) 订阅主题时，Dapr 只将每个消息传递给该应用程序的一个实例。这通常称为相互竞争的消费者模式。

同样，如果两个不同的应用程序(不同的 ID) 订阅同一主题，那么 Dapr 将每个消息仅传递到每个应用程序的一个实例。

消息生存时间

Dapr 可以在每个消息上设置超时。表示如果消息未从 Pub/Sub 组件读取，则消息将被丢弃。这是为了防止未读消息的积累。

当前支持的发布订阅组件

<https://docs.dapr.io/zh-hans/reference/components-reference/supported-pubsub>

设置发布订阅组件

explorer "%USERPROFILE%\.dapr\components"

pubsub.yaml


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

发布测试主题

<https://docs.dapr.io/zh-hans/reference/cli/dapr-publish>

```
dapr publish --publish-app-id helloclient --pubsub mypubsub --topic
newOrder --data '{"OrderId": "123", "ProductId": "456", "Amount": 22}'
```

在 .NET 平台上使用 SDK 发布主题

```
<PackageReference Include="Dapr.Client" Version="1.8.0" />

using var daprClient = new
DaprClientBuilder().UseGrpcEndpoint("http://localhost:60001").Build();

while (true)
{
    var data = new
    {
        OrderId = Random.Shared.Next().ToString(),
        ProductId = Random.Shared.Next().ToString(),
        Amount = Random.Shared.Next(1,10)
    };

    await daprClient.PublishEventAsync("mypubsub", "newOrder", data);

    await Task.Delay(TimeSpan.FromSeconds(3));
}
```

订阅主题

申明方式订阅主题

subscription.yaml

```
apiVersion: dapr.io/v1alpha1
```



```
kind: Subscription
metadata:
  name: myevent-subscription
spec:
  topic: newOrder
  route: /api/orders
  pubsubname: mypubsub
scopes:
- helloservice
```

编程方式订阅

```
[Route("dapr/subscribe")]
[ApiController]
public class DaprSubscribeController : ControllerBase
{
    public class DaprSubscribeOutput
    {
        [JsonPropertyName("pubsubname")]
        public string? PubSubName { get; set; }

        [JsonPropertyName("topic")]
        public string? Topic { get; set; }

        [JsonPropertyName("route")]
        public string? Route { get; set; }
    }

    [HttpGet]
    public ActionResult<DaprSubscribeOutput[]> Get()
    {
        return Ok(new DaprSubscribeOutput[]
        {
            new DaprSubscribeOutput
            {
                PubSubName="mypubsub",
                Topic="newOrder",
                Route="/api/orders"
            }
        });
    }
}
```

关于 CloudEvent 封装

<https://github.com/dapr/dotnet-sdk/blob/master/src/Dapr.Client/CloudEvent.cs>

```
[ApiController]
[Route("api/[controller]")]
public class OrdersController : ControllerBase
{
    [HttpPost]
    public ActionResult PostOrder(CloudEvent<OrderData> @event)
    {
        Console.WriteLine(System.Text.Json.JsonSerializer.Serialize(@event.Data));
    }
}
```



```
        return Ok();  
    }  
}
```

在 ASP .NET Core 中使用 SDK 订阅主题

```
<PackageReference Include="Dapr.AspNetCore" Version="1.8.0" />
```

自动注入 DaprClient 和相关模型绑定

```
builder.Services.AddControllers().AddDapr(configureClient =>  
{ configureClient.UseGrpcEndpoint("http://localhost:50001"); });
```

使用中间件拦截事件请求

```
app.UseCloudEvents();  
app.MapControllers();  
app.MapSubscribeHandler();
```

参考资料

<https://docs.dapr.io/zh-hans/developing-applications/building-blocks/pubsub/pubsub-overview>

https://docs.dapr.io/zh-hans/reference/api/pubsub_api

<https://docs.microsoft.com/zh-cn/dotnet/architecture/dapr-for-net-developers/publish-subscribe>

