Microsoft

Reactor

Let's Get Technical - Vietnam

Build a Workflow Application in minutes with Dapr

Friday, August 11 | 4PM GMT+7 | LIVE

Thang Chung

Technical Manager, NashTech
Microsoft MVP

https://aka.ms/lets-get-technical-vn

Let's Get Technical – Vietnam is a 55-minute virtual session where we showcase the latest innovative tech trends specific to Vietnam

Reactors Code of Conduct

Microsoft is dedicated to empowering every person and every organization on the planet to achieve more. This includes Microsoft Reactor events where we seek to provide a respectful, friendly, professional experience for everyone, regardless of gender, sexual orientation, physical appearance, disability, age, race, or religion.

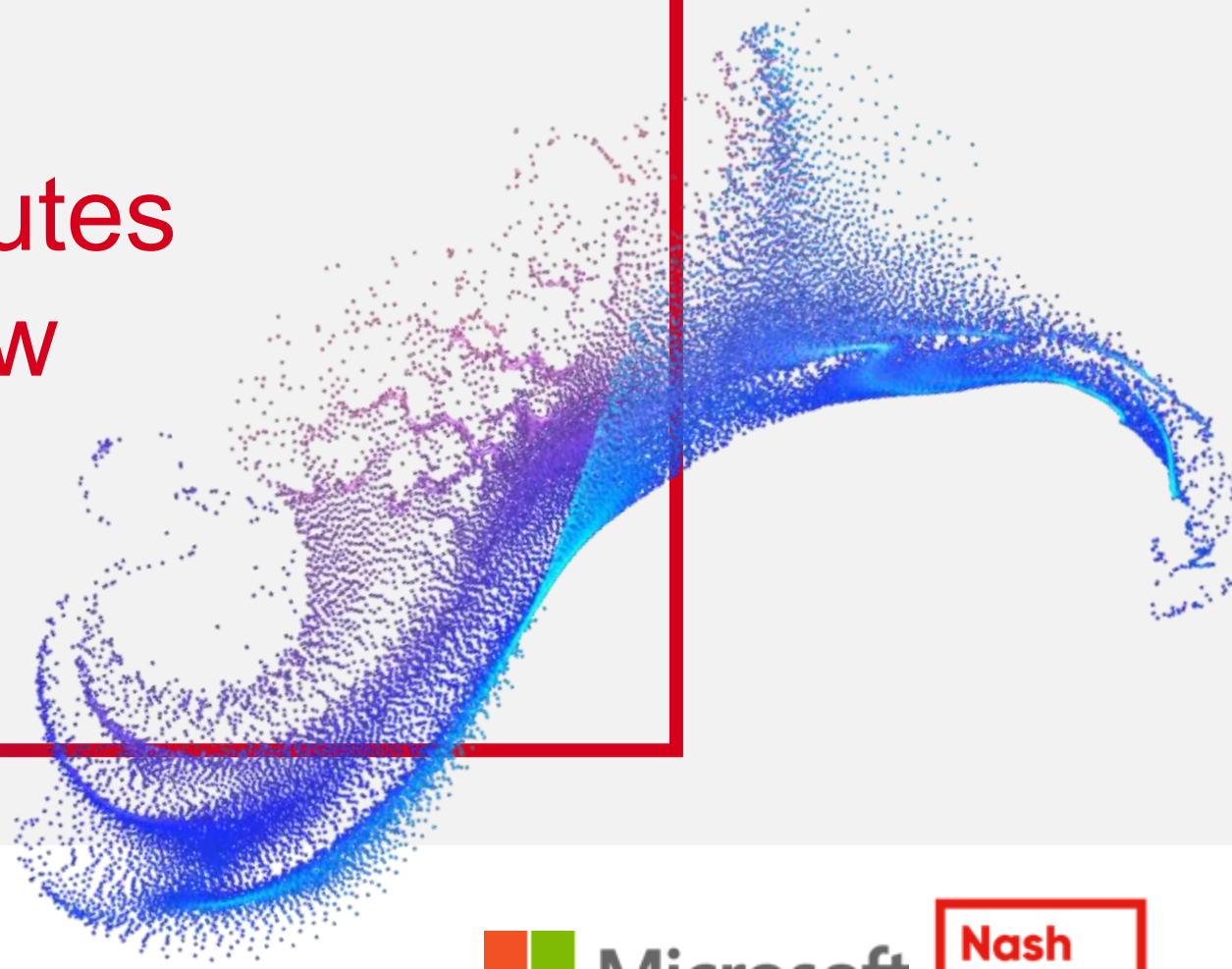
We do not tolerate any behavior that is harassing or degrading to any individual, in any form. Individuals are responsible for knowing and abiding by these standards. We encourage everyone to assist in creating a welcoming and safe environment.

Quy tắc ứng xử trên Microsoft Reactor

Microsoft cam kết hỗ trợ mọi người và mọi tổ chức trên hành tinh đạt được nhiều thành tựu hơn. Điều này bao gồm các sự kiện trên Microsoft Reactor, nơi chúng tôi tìm cách cung cấp trải nghiệm thân thiện, chuyên nghiệp cho mọi người, bất kể giới tính, xu hướng tình dục, ngoại hình, khuyết tật, tuổi tác, chủng tộc hoặc tôn giáo.

Chúng tôi không dung thứ cho bất kỳ hành vi quấy rối hoặc hạ thấp phẩm giá nào đối với bất kỳ cá nhân nào, dưới mọi hình thức. Các cá nhân có trách nhiệm biết và tuân thủ các tiêu chuẩn này. Chúng tôi khuyến khích mọi người hỗ trợ tạo ra một môi trường thân thiện và an toàn.

Build a Stateful Application in Minutes with Dapr Workflow



Thang Chung

Technical Manager @ NashTech VN

Microsoft Azure MVP

August 11, 2023

Microsoft Reactor

Nash Tech.

Agenda

1. The problems
2. Workflow process and engine
3. Workflow patterns
4. Demo
5. Q&A

About me

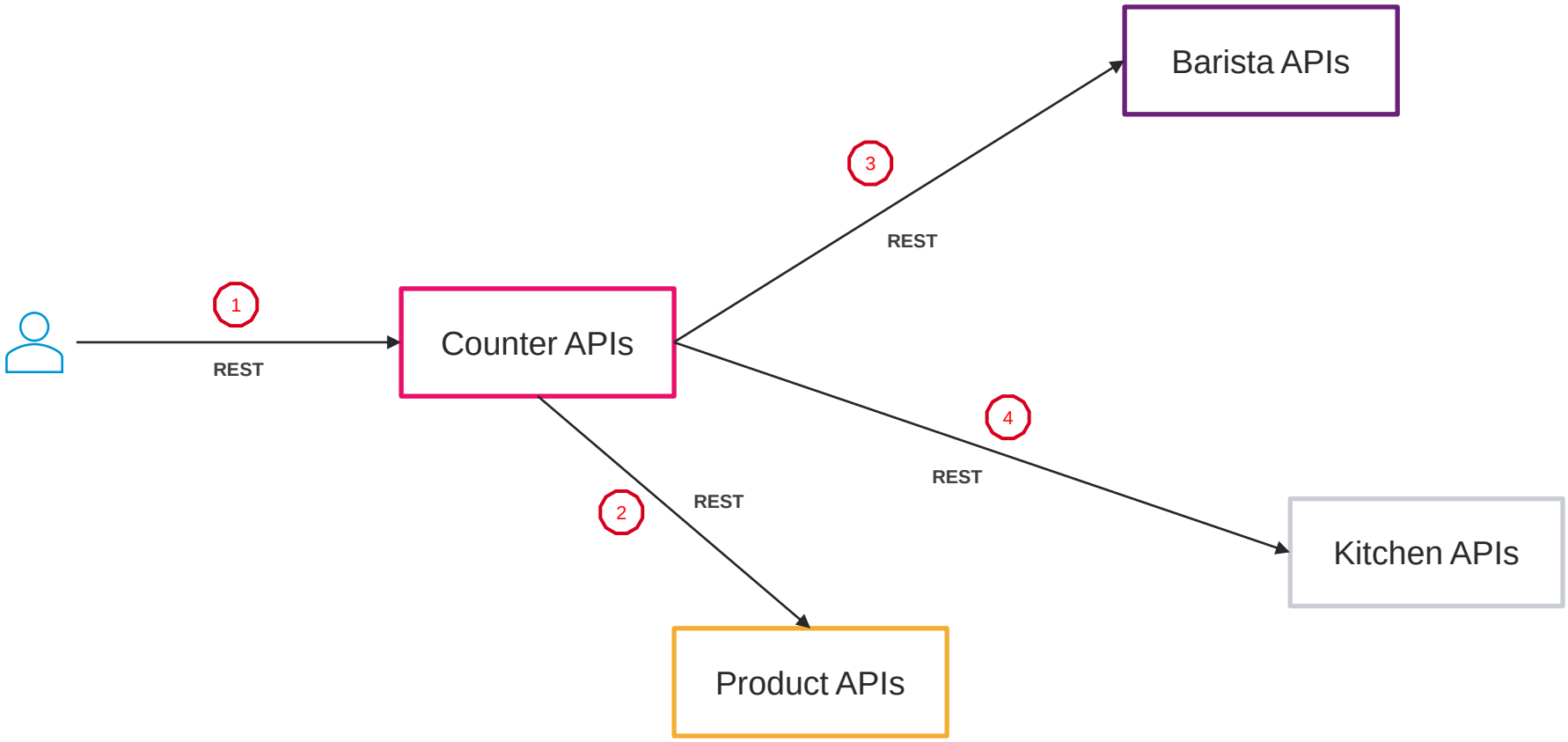
Thang Chung

- Technical Manager at NashTech VN
- Microsoft Azure MVP since 2020
- Creator of Vietnam Microservices Group on Facebook (11.2k participants).
  - https://www.facebook.com/groups/645391349250568
- Experience: ~16 years in software consult, design, development, and deployment software for outsourcing, product, and startup companies.
- Expertise in cloud computing, cloud-native platform, serverless, and WebAssembly/WASI.
- GitHub: https://github.com/thangchung
- Twitter: @thangchung

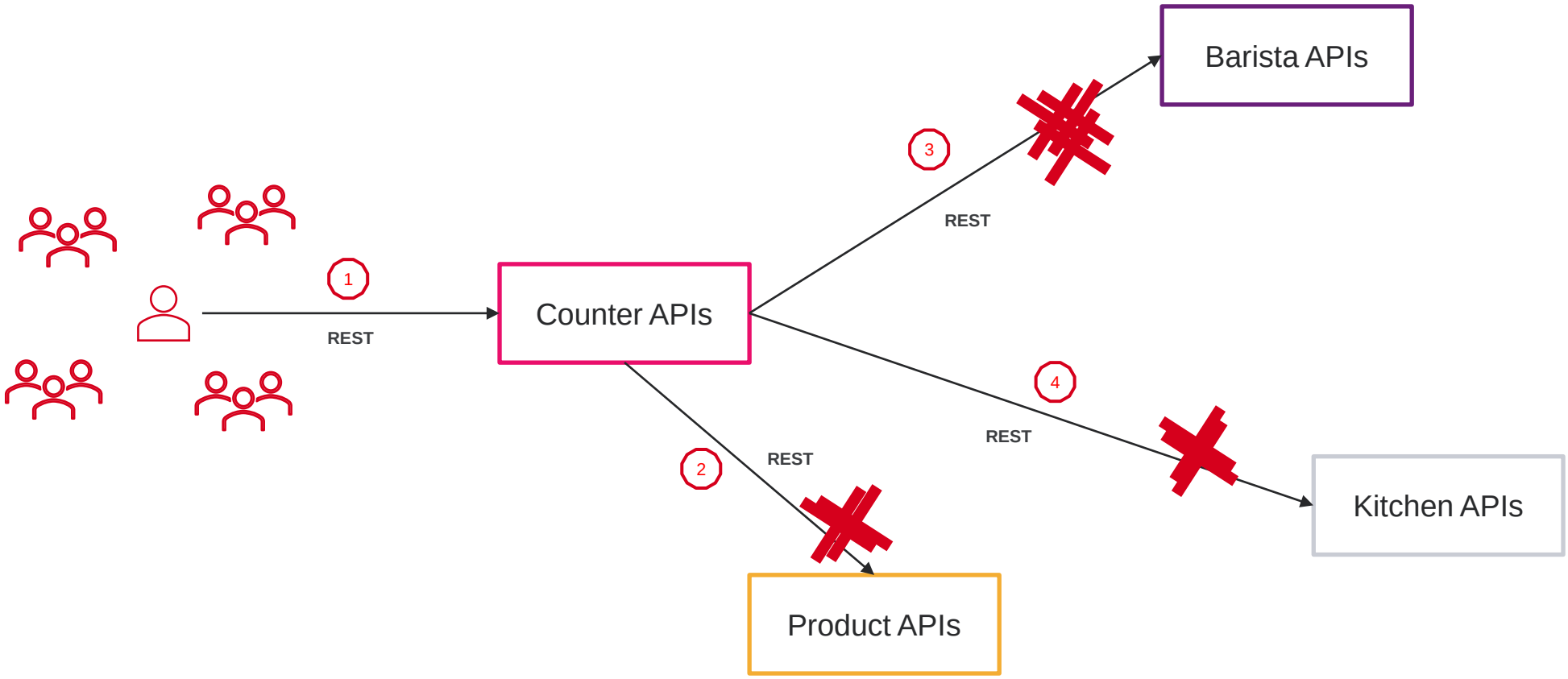
01

The problems

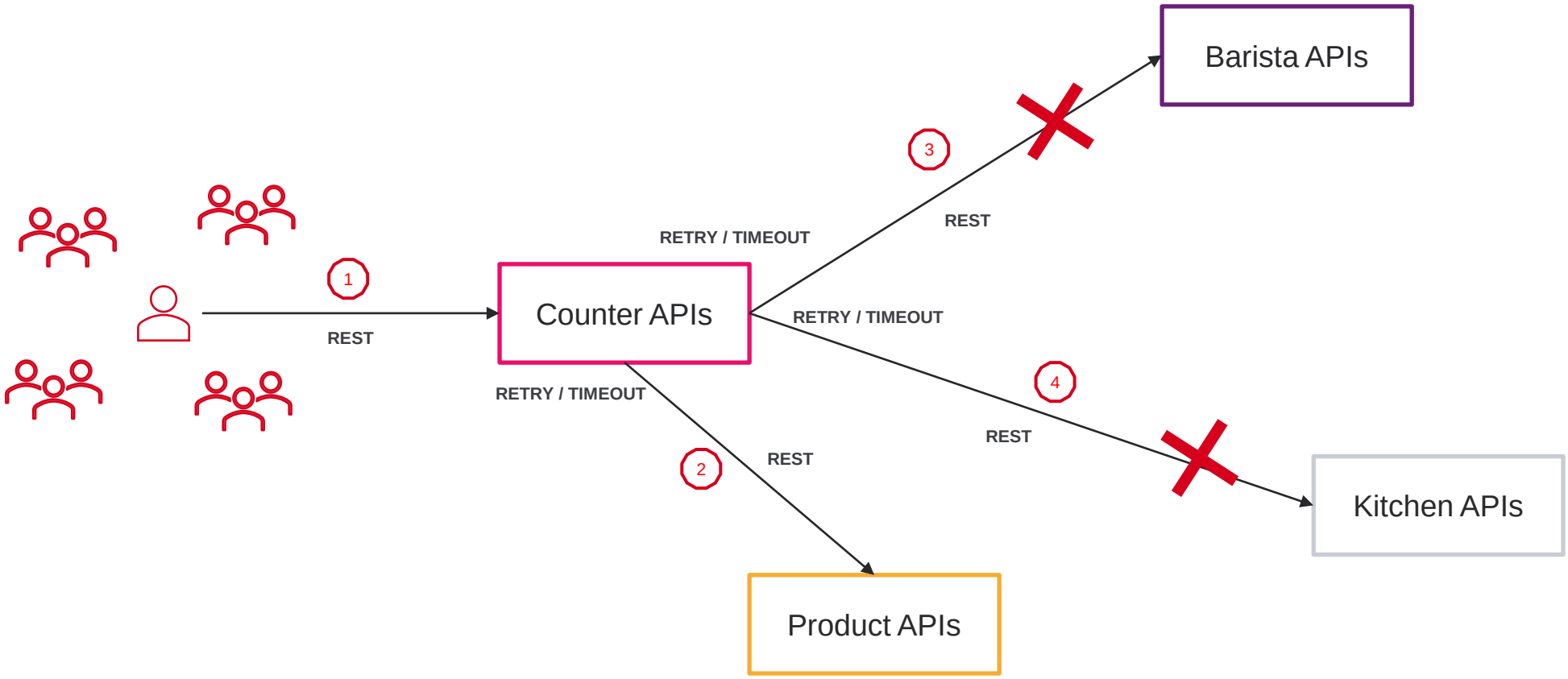
The story – simple and beautiful software architecture



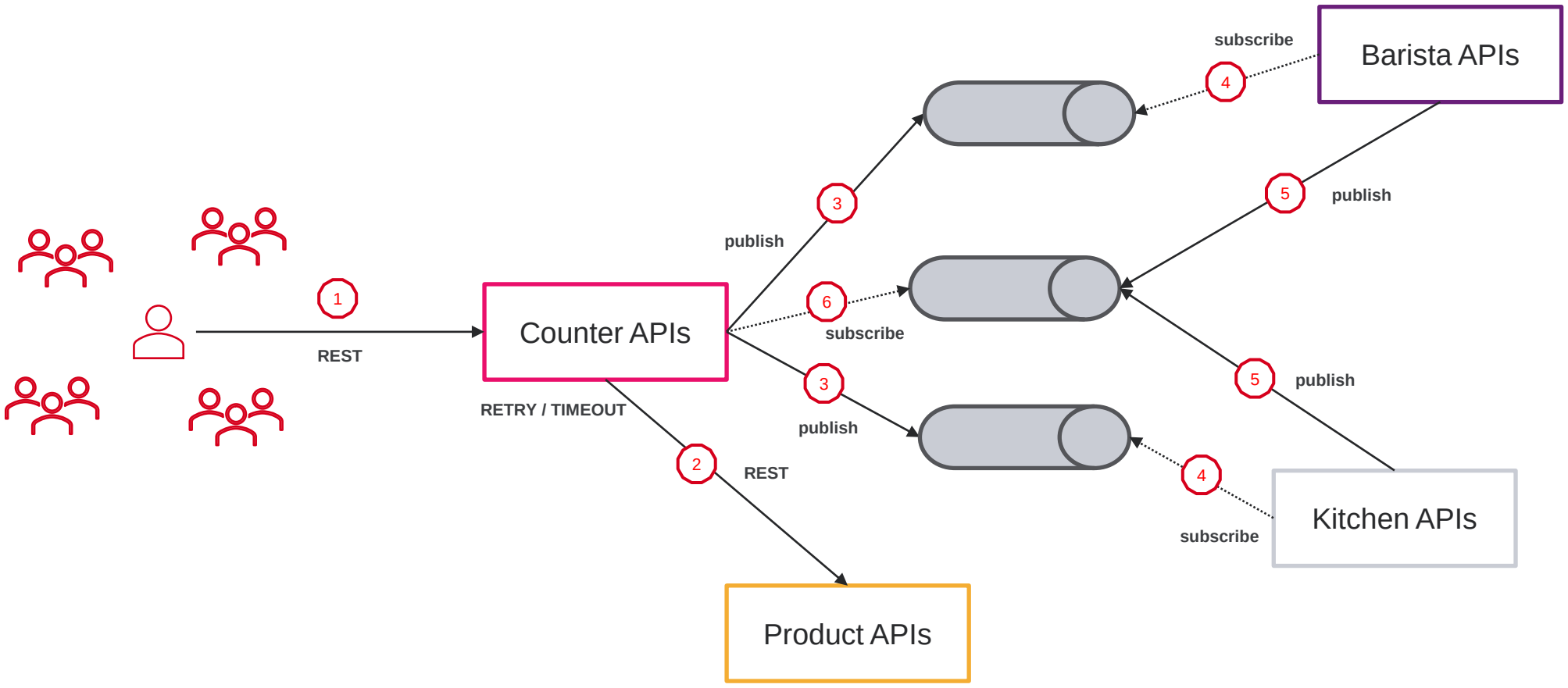
The story – more network traffics



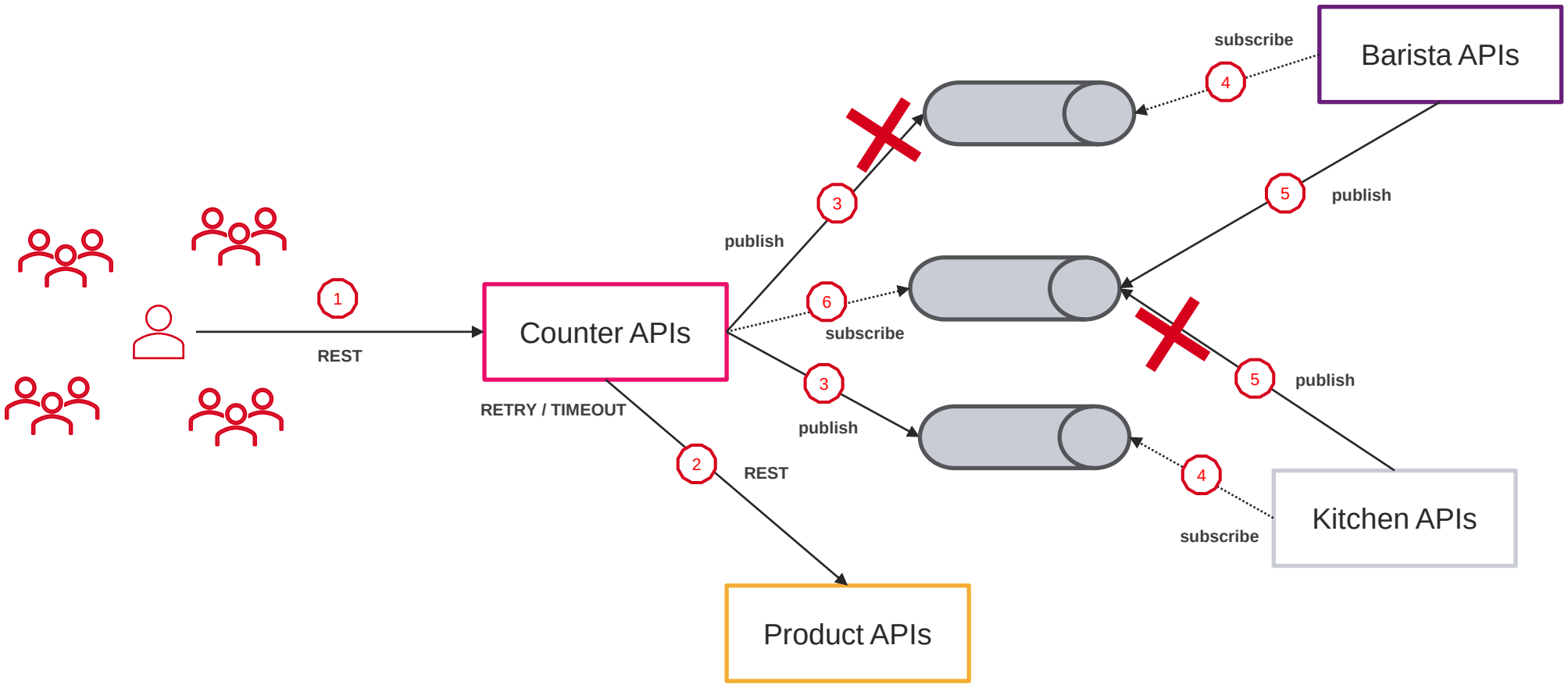
The story – add resiliency (+retry/+timeout)



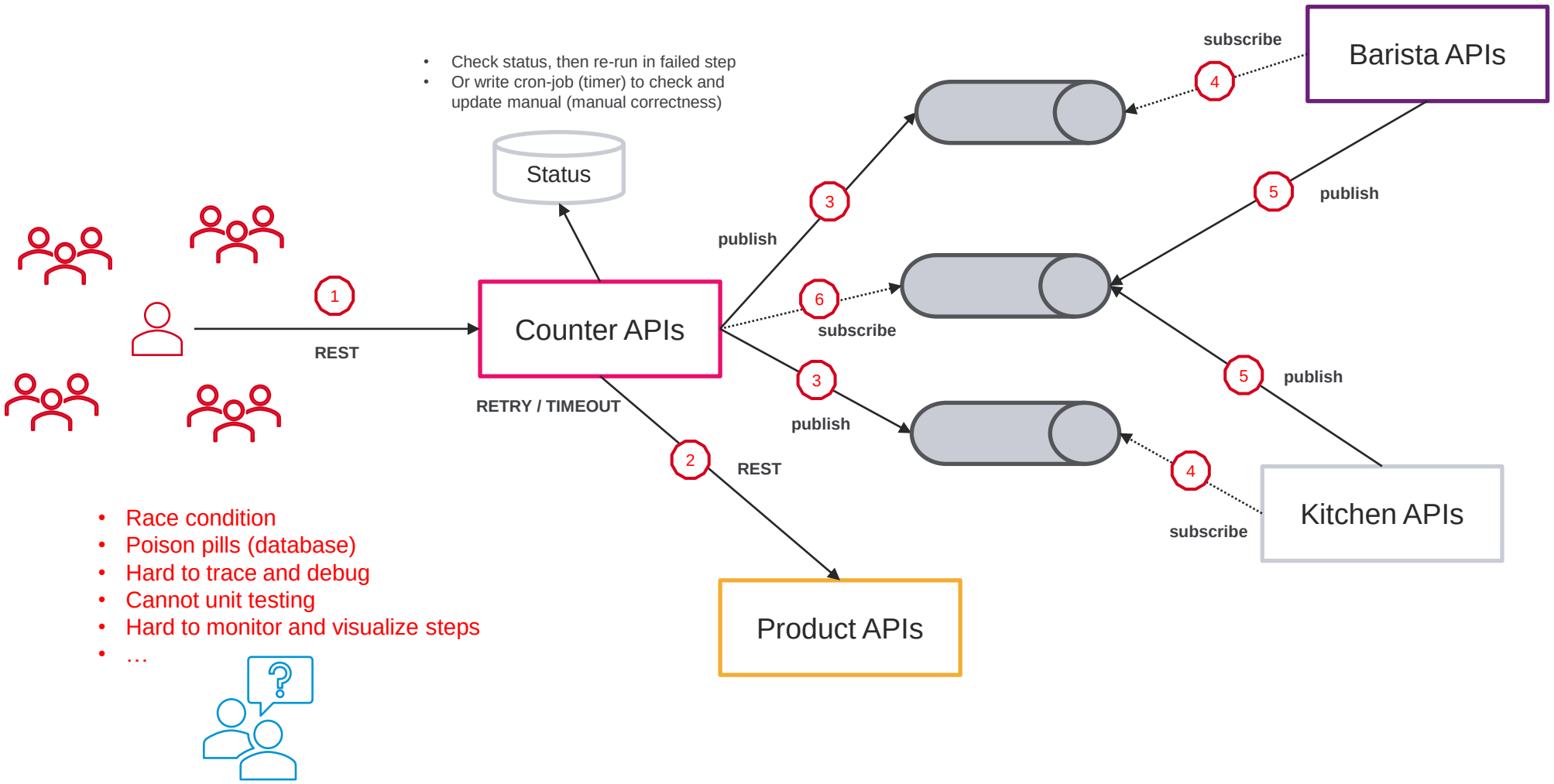
The story – adding queues to increase reliability



The story – failed so not know which step to re-run



The story – store the status in storage to manual check



- Race condition
- Poison pills (database)
- Hard to trace and debug
- Cannot unit testing
- Hard to monitor and visualize steps
- …

02

Enter the world of the workflow process and engine

Workflow definition

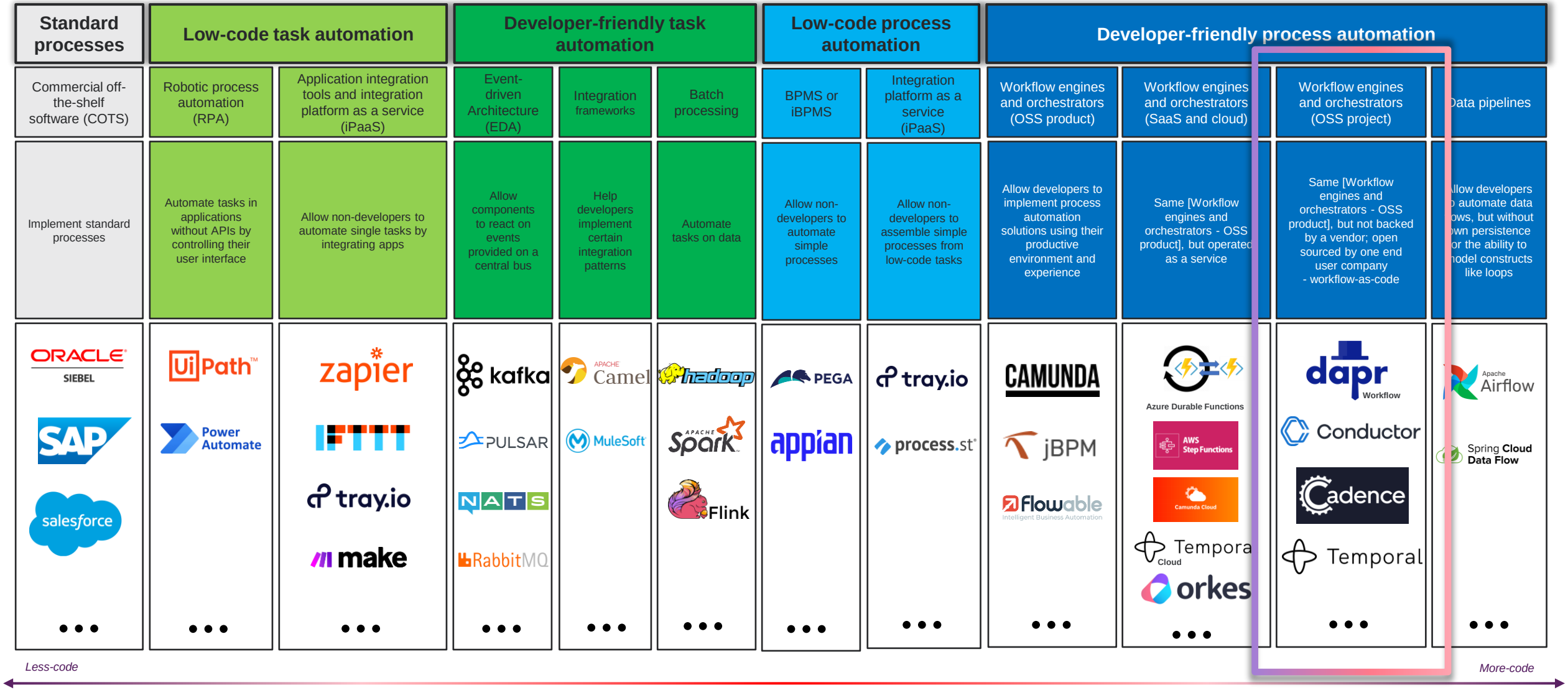
- A workflow is a system for managing **repetitive processes and tasks** which occur in a particular order. They are the mechanism by which people and enterprises accomplish their work, whether manufacturing a product, providing a service, processing information or any other value-generating activity.

- Within business process management, a workflow can be defined as a simple series of individual tasks, while a business process is considered more complex, **consisting of multiple workflows, information systems, data, people and their activity patterns**. A workflow is distinguished by its simplicity and repeatability, and it is generally visualized with diagram or checklist.

- Workflow management software assists in simplifying and optimizing a business process within an organization. It largely does this by **coordinating interactions among different stakeholders or between individuals and information systems**. Workflow management systems route tasks to the appropriate employee at the right time, providing the pertinent information and nudge to expedite work along the overall process. It also **supports manual and automated tasks** through document management for activities, like expense reports.

https://www.ibm.com/topics/workflow

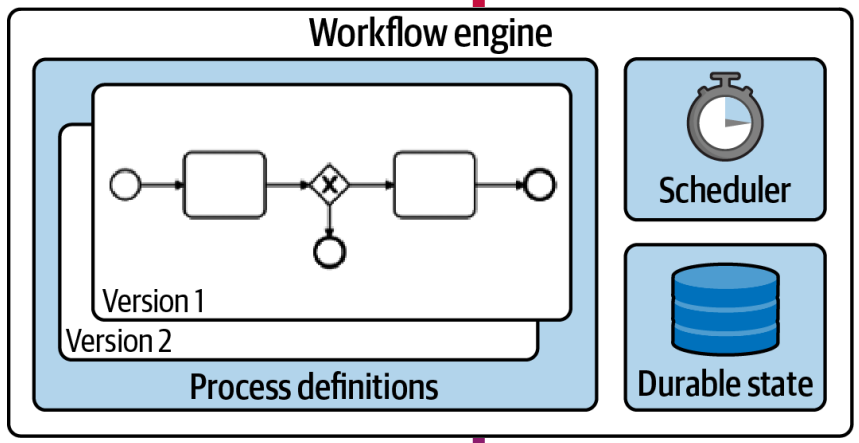
The process automation landscape



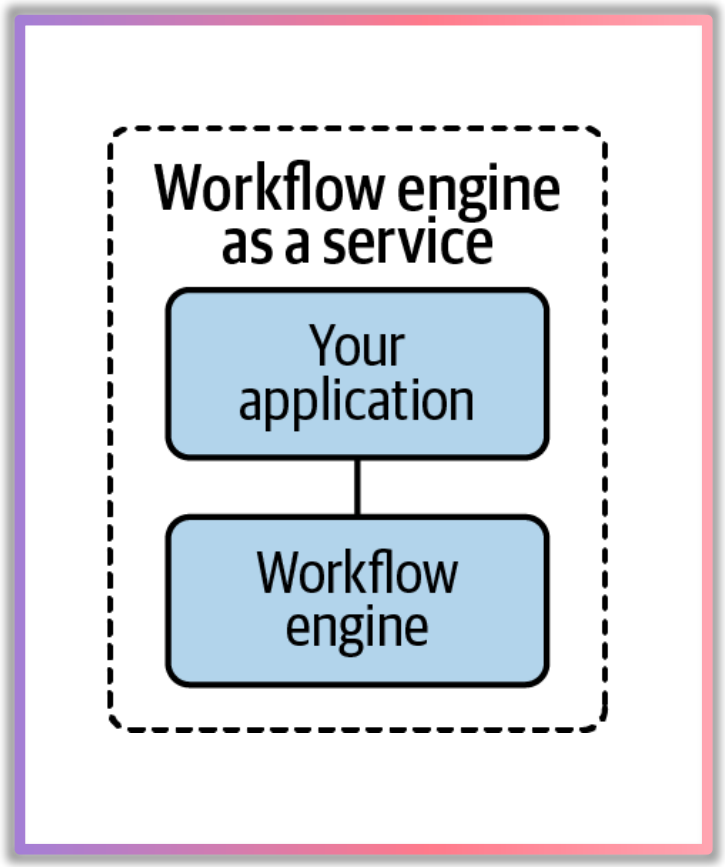
Ref: Understanding the process automation landscape

Workflow architecture (1/2)

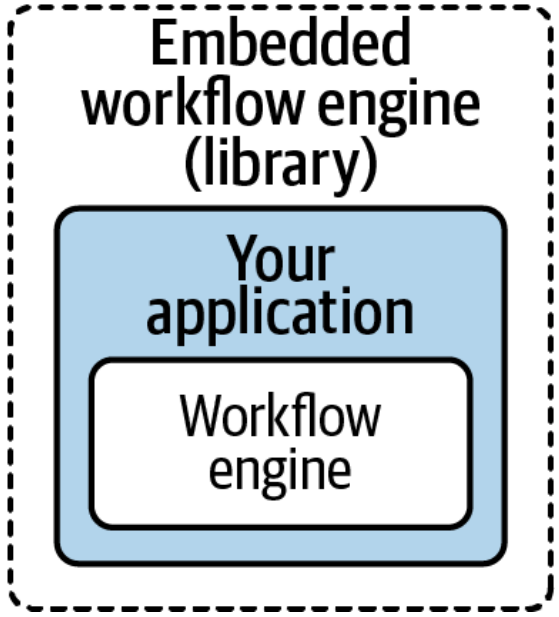
- Durable state (persistence)
- Scheduling
- Versioning
- Visibility (Process Modeling Languages)
- Audit data(Event Sourcing)
- Tooling (e.g., tracing, transformation…)



Workflow architecture (2/2)



Microservices, Serverless,
or Enterprise System

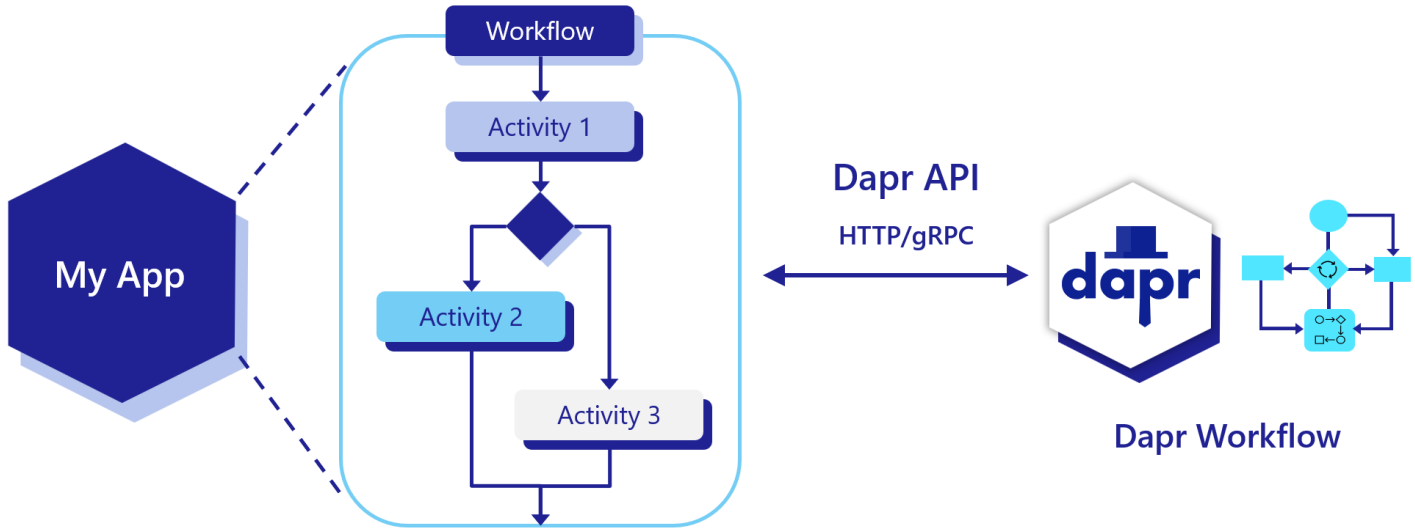


Monolithic App
or Modular Monolith App

Dapr workflow architecture

Core concepts

- Workflows
  - Workflow identity
  - Workflow replay
  - Infinite loops and eternal workflows (continue-as-new API)
- Workflow Activities (ensure at-least-once)
- Child workflows
- Durable timers
- Retry policies
- External events



Rule: Code should be deterministic

- I/O
- Timer & Random
- Schedule background threads

Rule 1: I/O

```
// DON'T DO THIS!
string configuration = Environment.GetEnvironmentVariable("MY_CONFIGURATION")!;
string data = await new HttpClient().GetStringAsync("https://example.com/api/data");
```

```
// Do this!!
string configuation = workflowInput.Configuration; // imaginary workflow input argument
string data = await context.CallActivityAsync<string>("MakeHttpCall", "https://example.com/api/data");
```

Rule 2: Timer & Random

```
// DON'T DO THIS!
DateTime currentTime = DateTime.UtcNow;
Guid newIdentifier = Guid.NewGuid();
string randomString = GetRandomString();
```

```
// Do this!!
DateTime currentTime = context.CurrentUtcDateTime;
Guid newIdentifier = context.NewGuid();
string randomString = await context.CallActivityAsync<string>("GetRandomString");
```

Rule 3: Schedule background threads

```
// DON'T DO THIS!
Task t = Task.Run(() => context.CallActivityAsync("DoSomething"));
await context.CreateTimer(5000).ConfigureAwait(false);
```

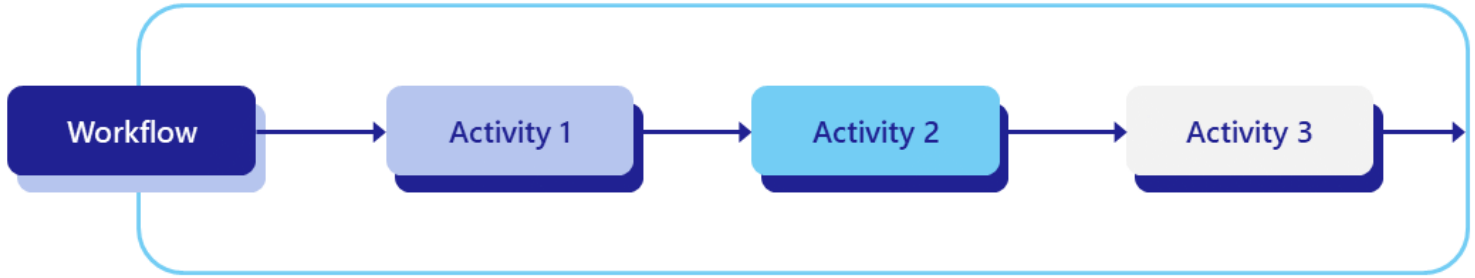
```
// Do this!!
Task t = context.CallActivityAsync("DoSomething");
await context.CreateTimer(5000).ConfigureAwait(true);
```

Workflow patterns

Workflow patterns (1/5)

Task chaining

- In the task chaining pattern, multiple steps in a workflow are run in succession, and the output of one step may be passed as the input to the next step.
- Problems solved:
  - What happens if one of the microservices are unavailable for an extended period of time?
  - Can failed steps be automatically retried?
  - If not, how do you facilitate the rollback of previously completed steps, if applicable?
  - Implementation details aside, is there a way to visualize the workflow so that other engineers can understand what it does and how it works?

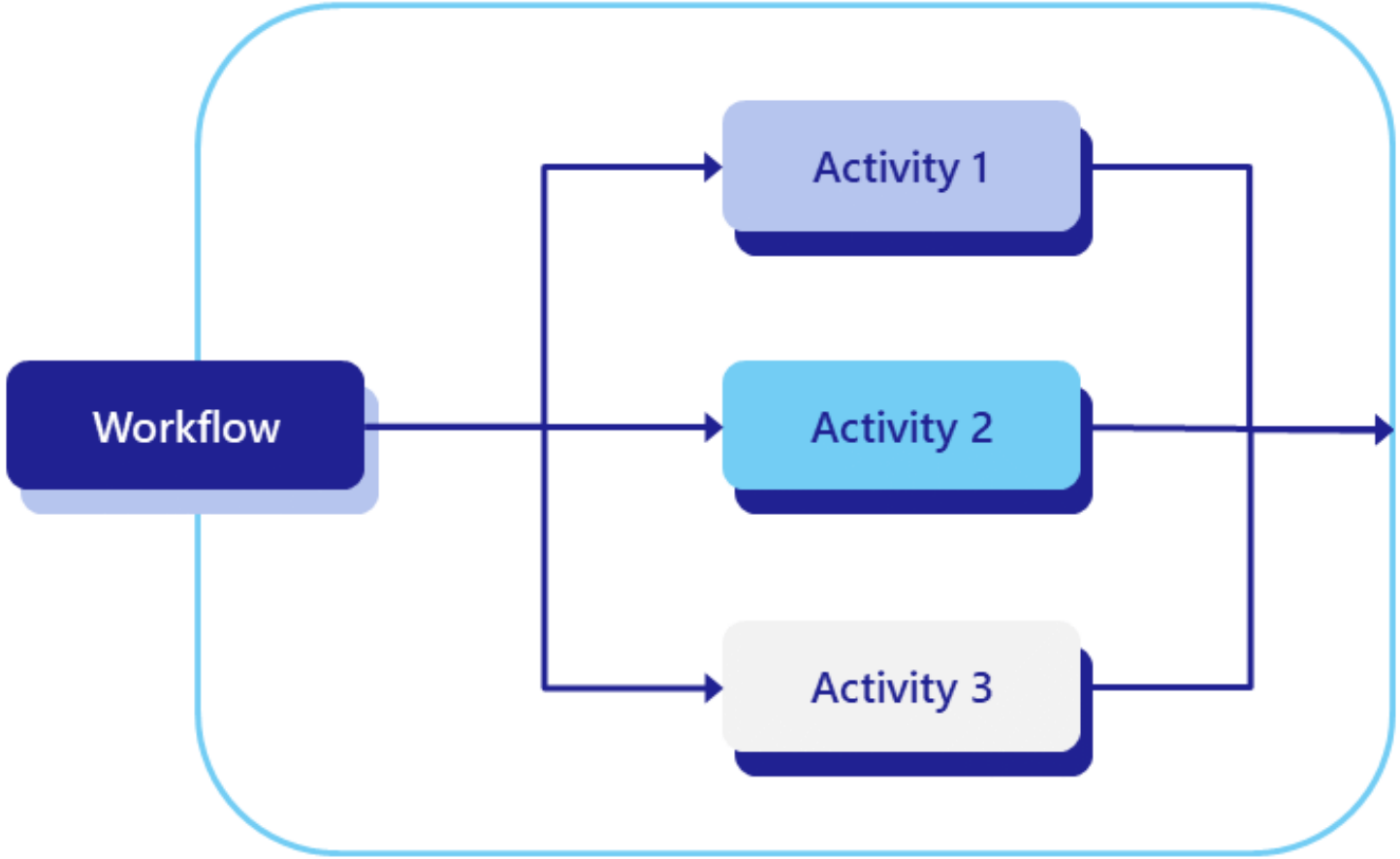


https://docs.dapr.io/developing-applications/building-blocks/workflow/workflow-patterns/#task-chaining

Workflow patterns (2/5)

Fan-out / Fan-in

- In the fan-out/fan-in design pattern, you execute multiple tasks simultaneously across potentially multiple workers, wait for them to finish, and perform some aggregation on the result.
- Problems solved:
  - How do you control the degree of parallelism?
  - How do you know when to trigger subsequent aggregation steps?
  - What if the number of parallel steps is dynamic?

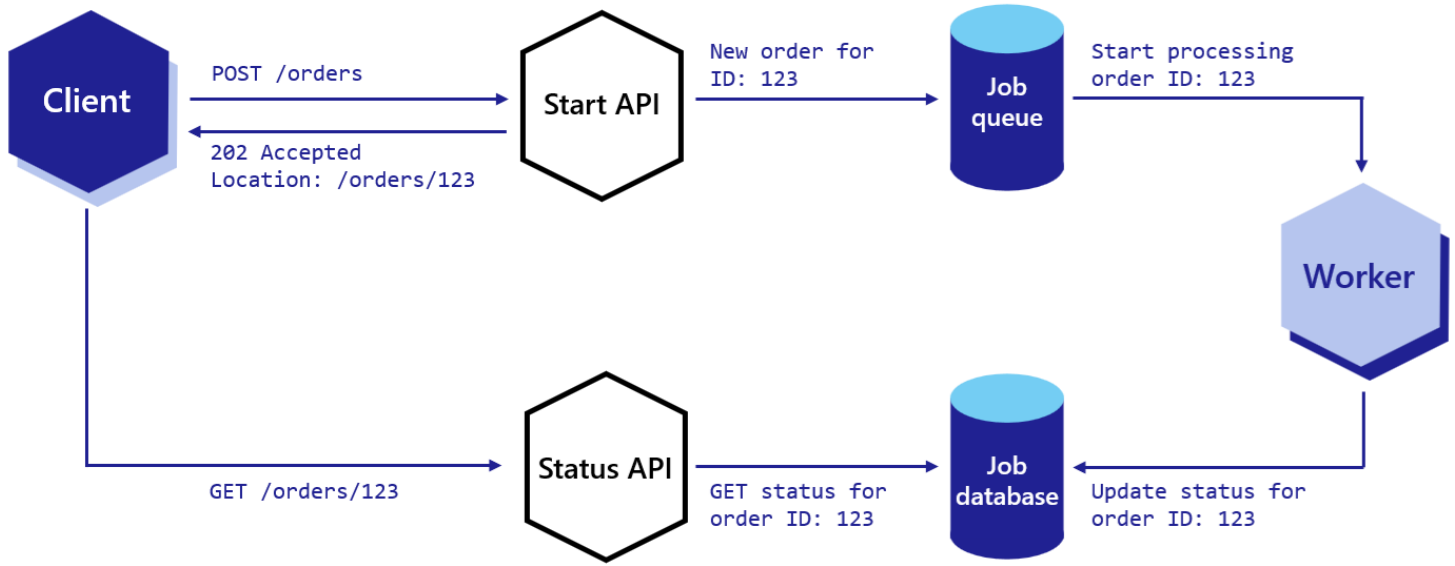


https://docs.dapr.io/developing-applications/building-blocks/workflow/workflow-patterns/#fan-outfan-in

Workflow patterns (3/5)

Async HTTP APIs (Asynchronous Request-Reply pattern)

1. A client sends a request to an HTTP API endpoint (the start API)
2. The start API writes a message to a backend queue, which triggers the start of a long-running operation
3. Immediately after scheduling the backend operation, the start API returns an HTTP 202 response to the client with an identifier that can be used to poll for status
4. The status API queries a database that contains the status of the long-running operation
5. The client repeatedly polls the status API either until some timeout expires or it receives a “completion” response



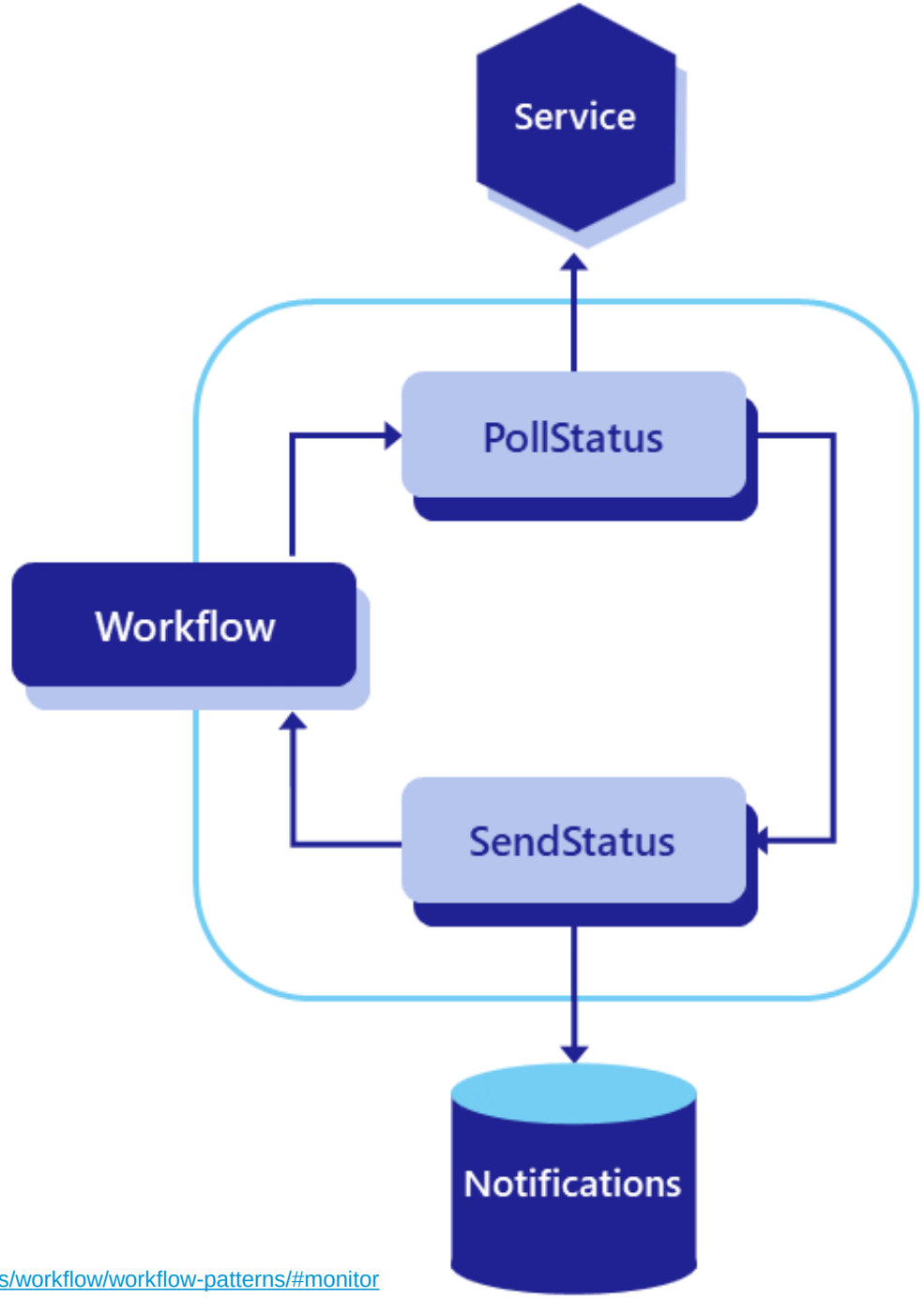
https://docs.dapr.io/developing-applications/building-blocks/workflow/workflow-patterns/#async-http-apis

Workflow patterns (4/5)

Monitor

1. Checks the status of a system
2. Takes some action based on that status - e.g., send a notification
3. Sleeps for some period of time
4. Repeat

- Careful about infinite while-loops (e.g., while(1))
- Eternal workflows with continue-as-new API

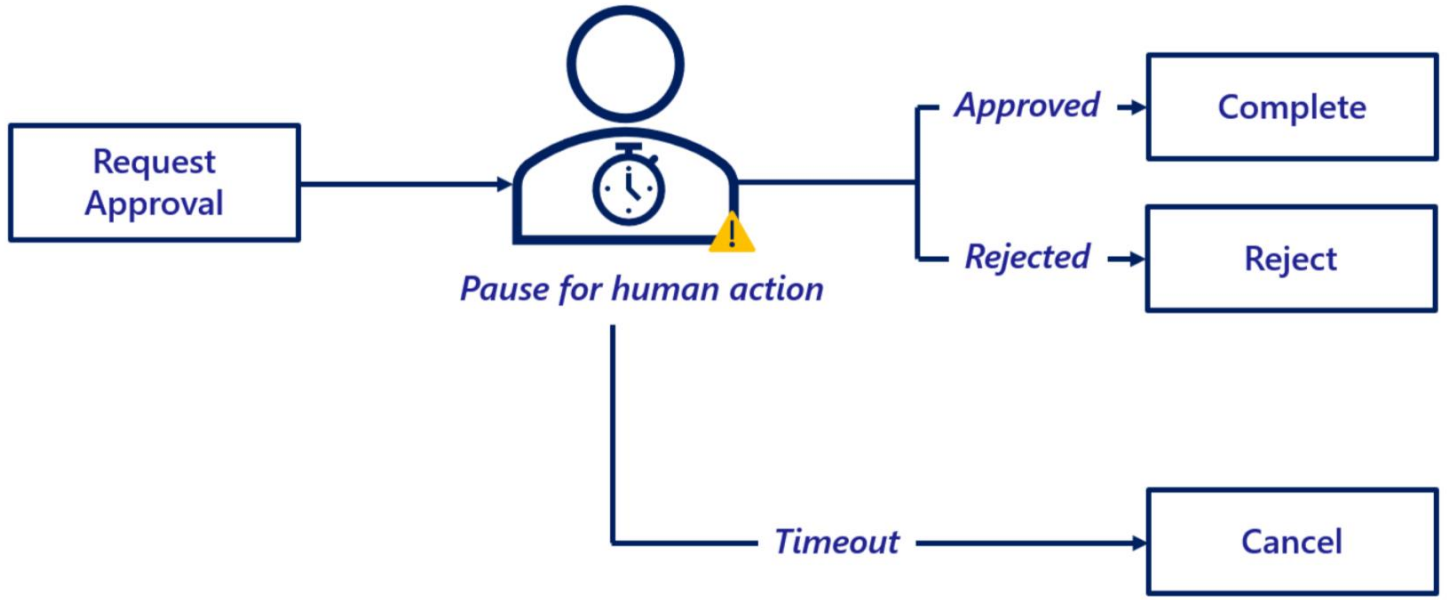


https://docs.dapr.io/developing-applications/building-blocks/workflow/workflow-patterns/#monitor

Workflow patterns (5/5)

External system interaction

- In some cases, a workflow may need to pause and wait for an external system to perform some action (RaiseEvent API).
  - For example, a workflow may need to pause and wait for a payment to be received
- Another very common scenario is when a workflow needs to pause and wait for a human (WaitForExternalEvent API).
  - For example, when approving a purchase order

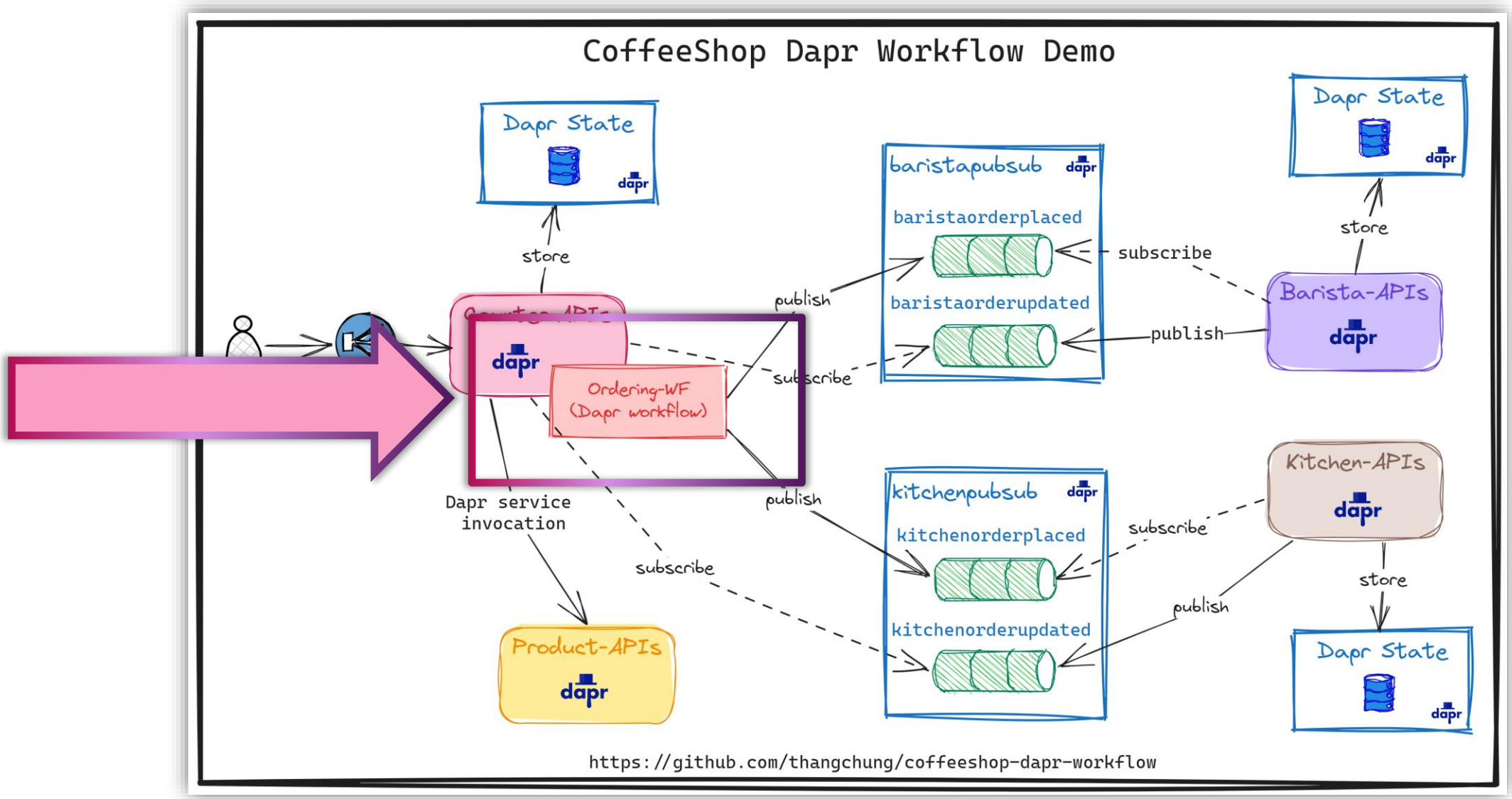


https://docs.dapr.io/developing-applications/building-blocks/workflow/workflow-patterns/#external-system-interaction

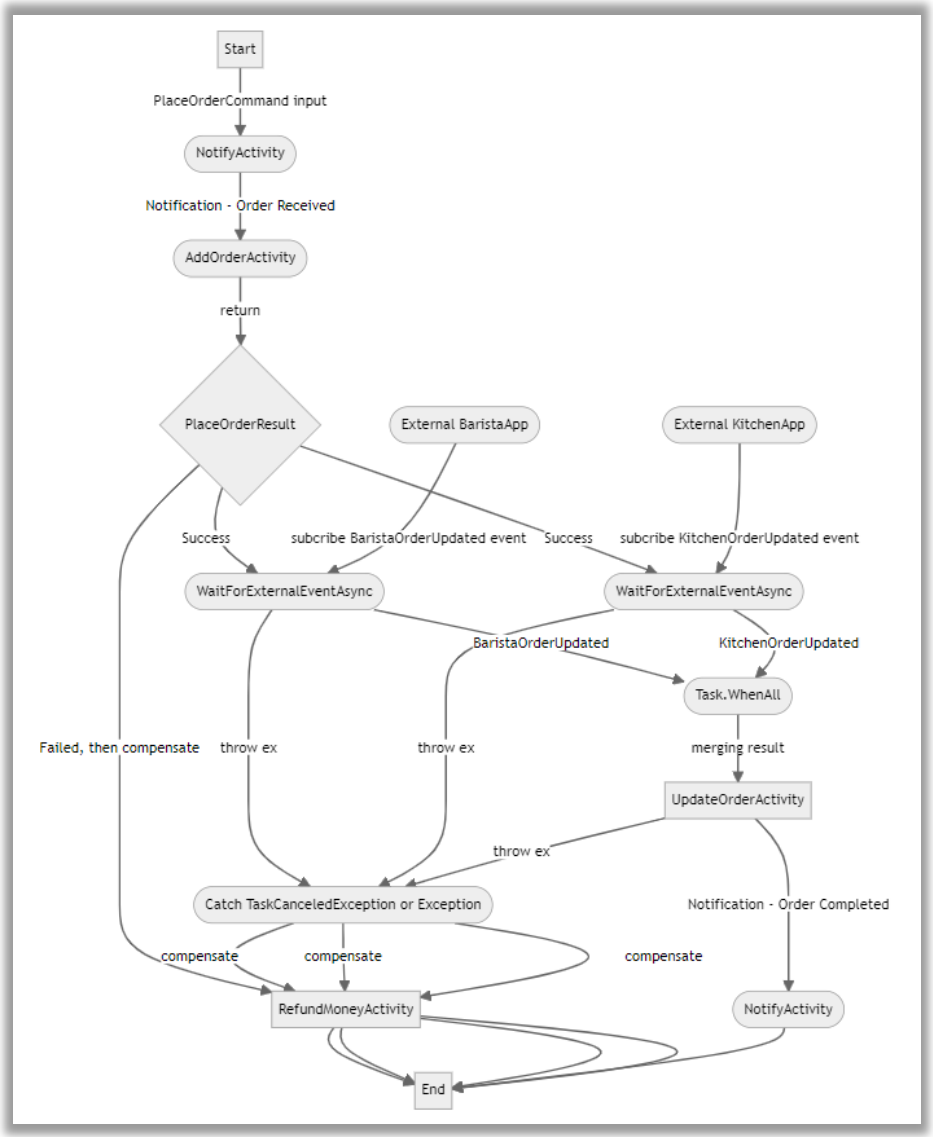
Demo Time

“Talk is cheap. Show me the code” - Linus Torvalds

DEMO: CoffeeShop on Dapr Workflow (1/2)



DEMO: CoffeeShop on Dapr Workflow (2/2)



Resources

https://aka.ms/lets-get-technical-dapr

Microsoft

Reactor

Let's Get Technical- Vietnam

A 55-minute virtual session where we showcase the latest innovative tech trends specific to Vietnam

Join us for this free live virtual event!

- Friday, August 4, 4PM GMT+7 LIVE | Create your own Copilot today!
- Friday, August 11, 4PM GMT+7 LIVE | Build a Workflow Application in minutes with Dapr
- Friday, August 18, 4PM GMT+7 LIVE | Demystify OAuth 2.0 and Integrate web apps to Azure Active Directory

Our speakers

Phi Huynh
Technical Manager, NashTech
Microsoft MVP

Thang Chung
Technical Manager, NashTech
Microsoft MVP

Thien Nguyen
Technical Manager, NashTech
Microsoft MVP

Register now for this free event!

https://aka.ms/lets-get-technical-vn-edm

Thank you

Nash Tech.