Developing Microservices with Dapr

Memi Lavi
www.memilavi.com

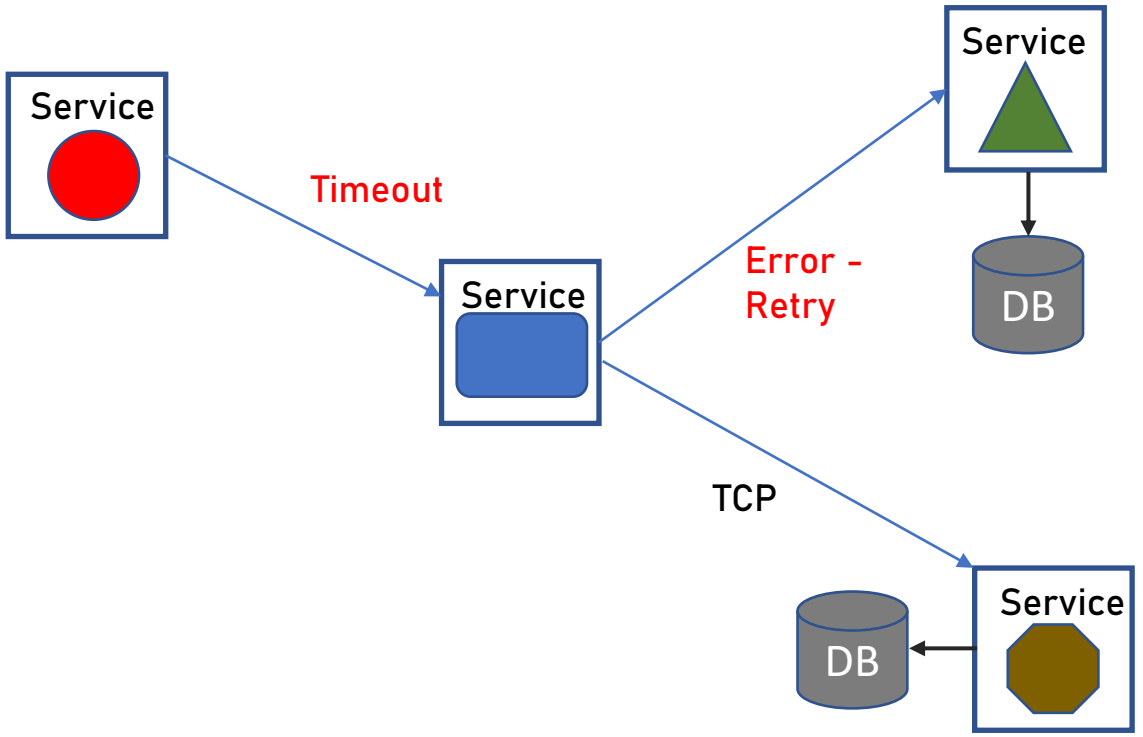
Dapr

- Manages service-to-service communication
- Provides additional services
- Technology, platform and cloud agnostic
- Created by Microsoft
- Integrated into Container Apps
- A type of Service Mesh

Problems Solved by Service Mesh

- Microservices communicate between them a lot
- The communication might cause a lot of problems and challenges:
  - Timeouts
  - Security
  - Retries
  - Monitoring

Problems Solved by Service Mesh



Service Mesh

- Software Components that sit near the service and manage all service-to-service communication
- Provides all communication services
- The service interacts with the service mesh only

Service Mesh Services

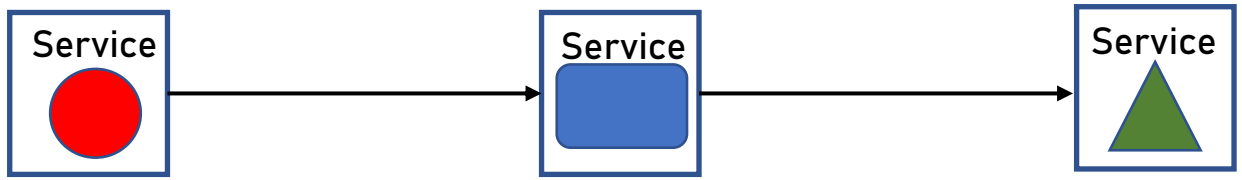
- Protocol conversion
- Communication security
- Authentication
- Reliability (timeouts, retries, health checks, circuit breaking)
- Monitoring
- Service Discovery

Service Mesh Services

- Testing (A/B testing, traffic splitting)
- Load balancing

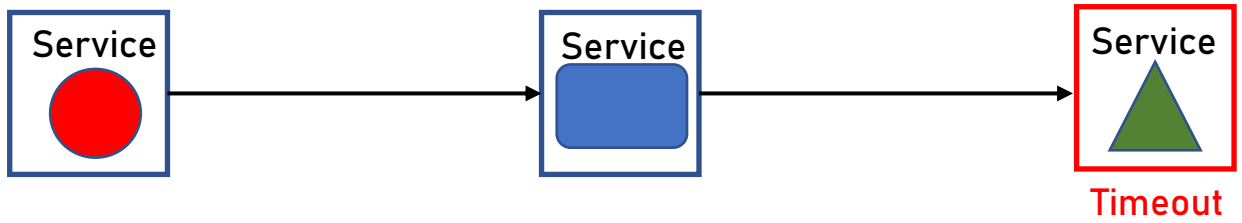
Circuit Breaker

- Prevents cascading failures when a service fails



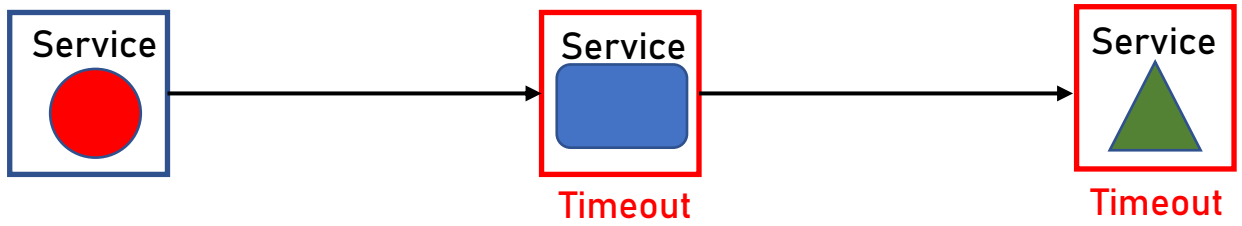
Circuit Breaker

- Prevents cascading failures when a service fails



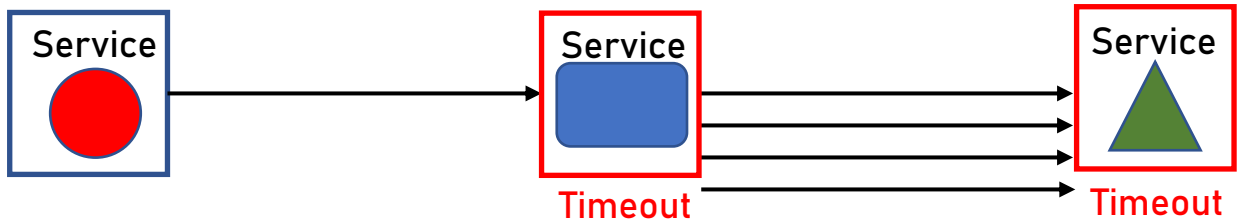
Circuit Breaker

- Prevents cascading failures when a service fails



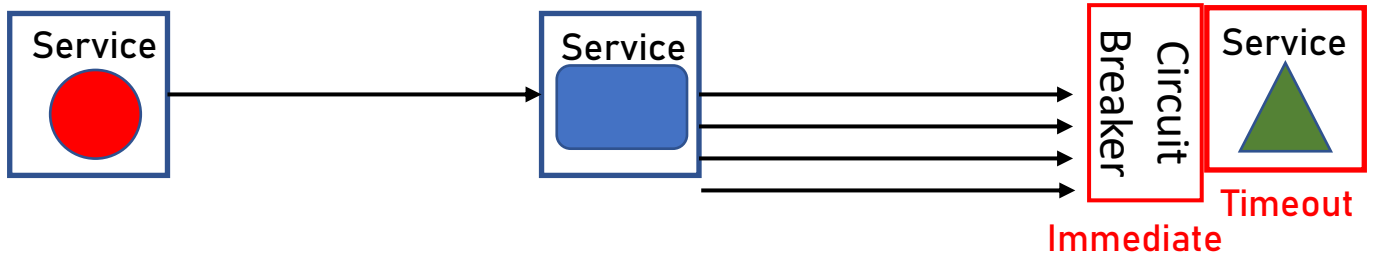
Circuit Breaker

- Prevents cascading failures when a service fails



Circuit Breaker

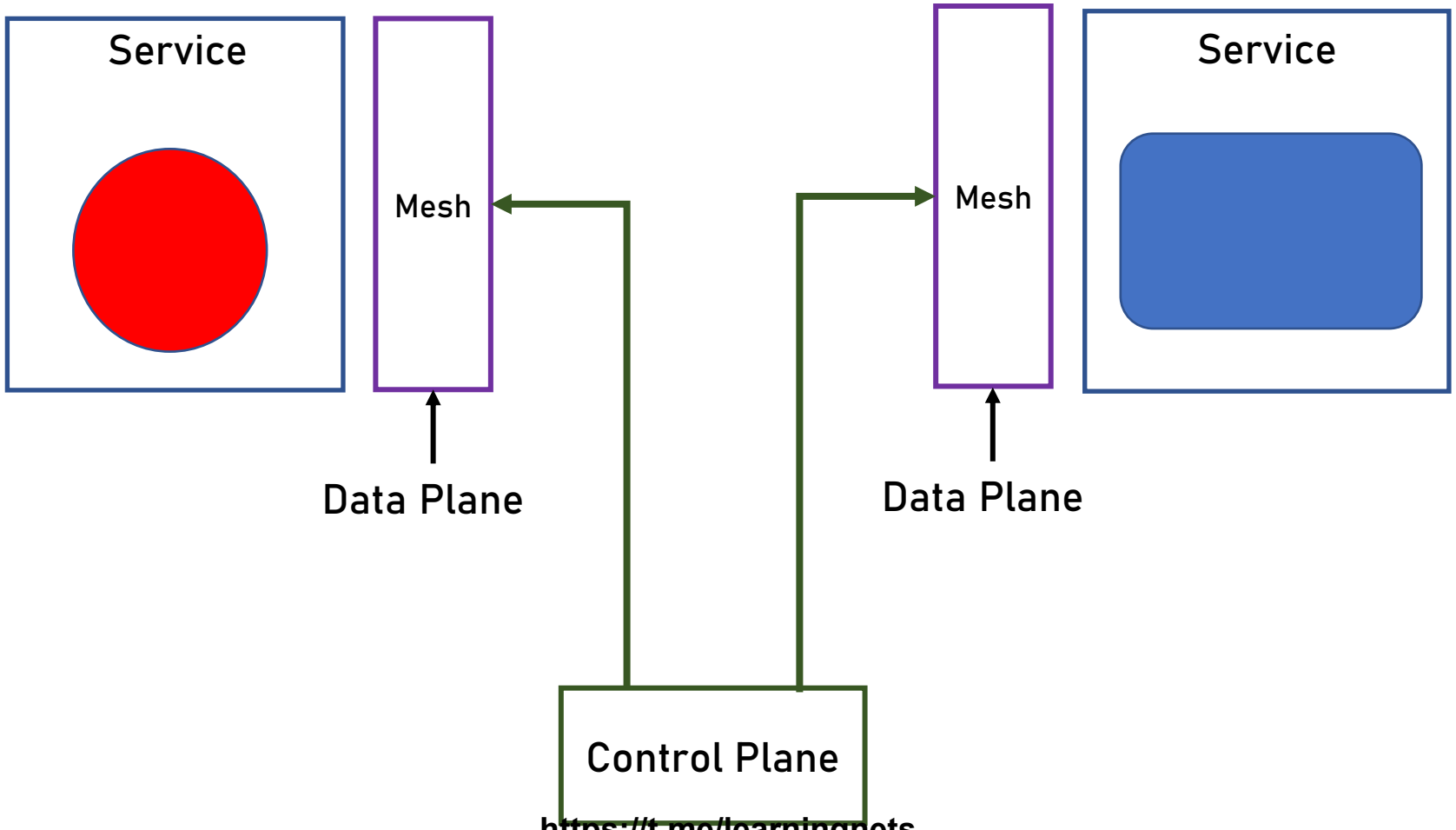
- Prevents cascading failures when a service fails



Service Mesh

- In short:
  - Service's developers need not handle communication aspects when using Service Mesh
  - Focus on the business, not the plumbing

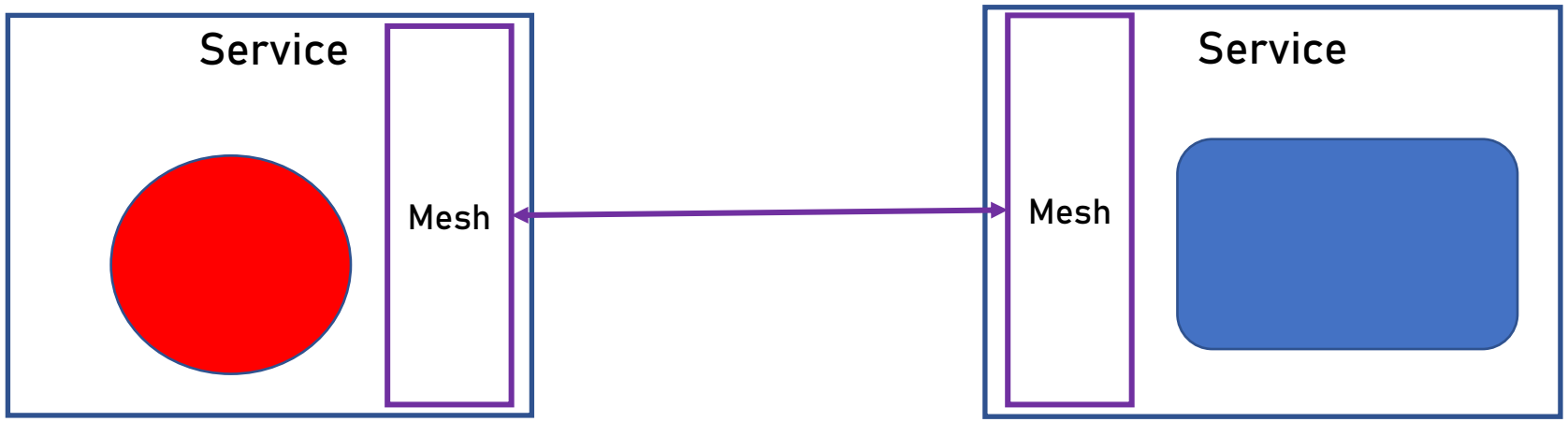
Service Mesh Architecture



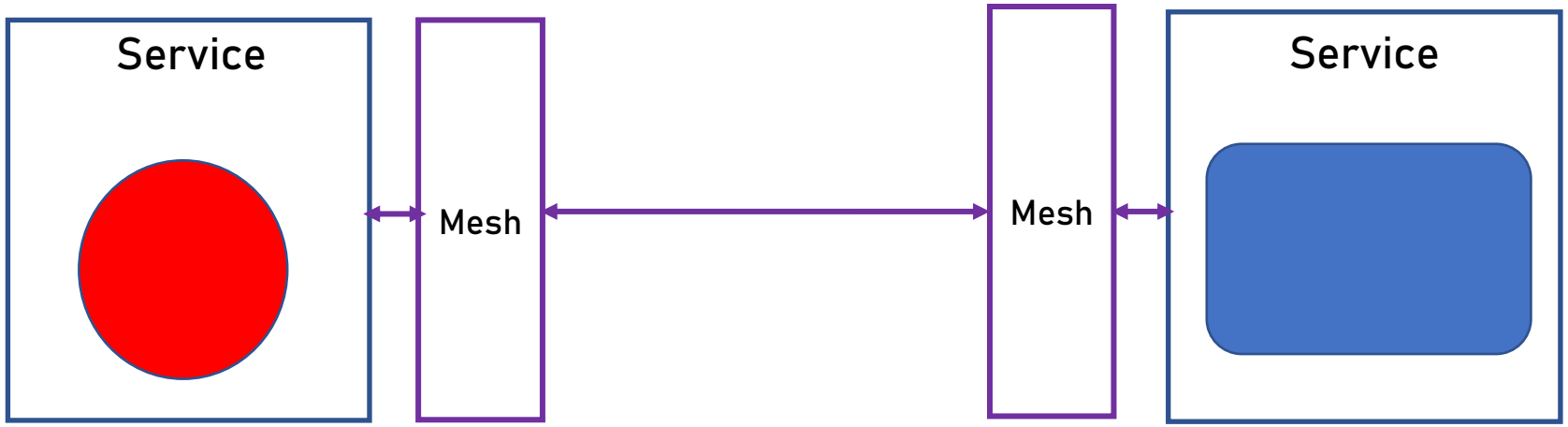
Types of Service Mesh

- Two main types:
  - In-Process
  - Sidecar

In-Process



Sidecar



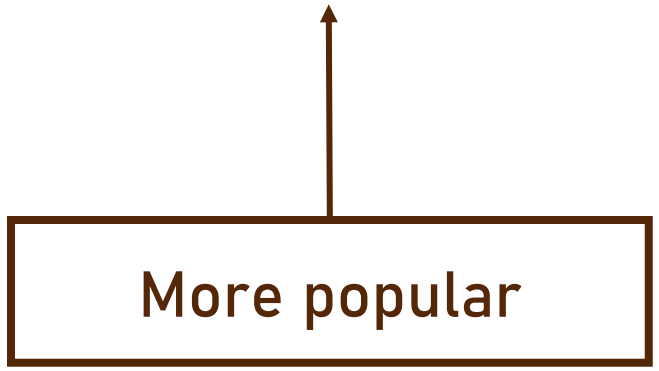
In-Process vs Sidecar

In-Process

- Performance

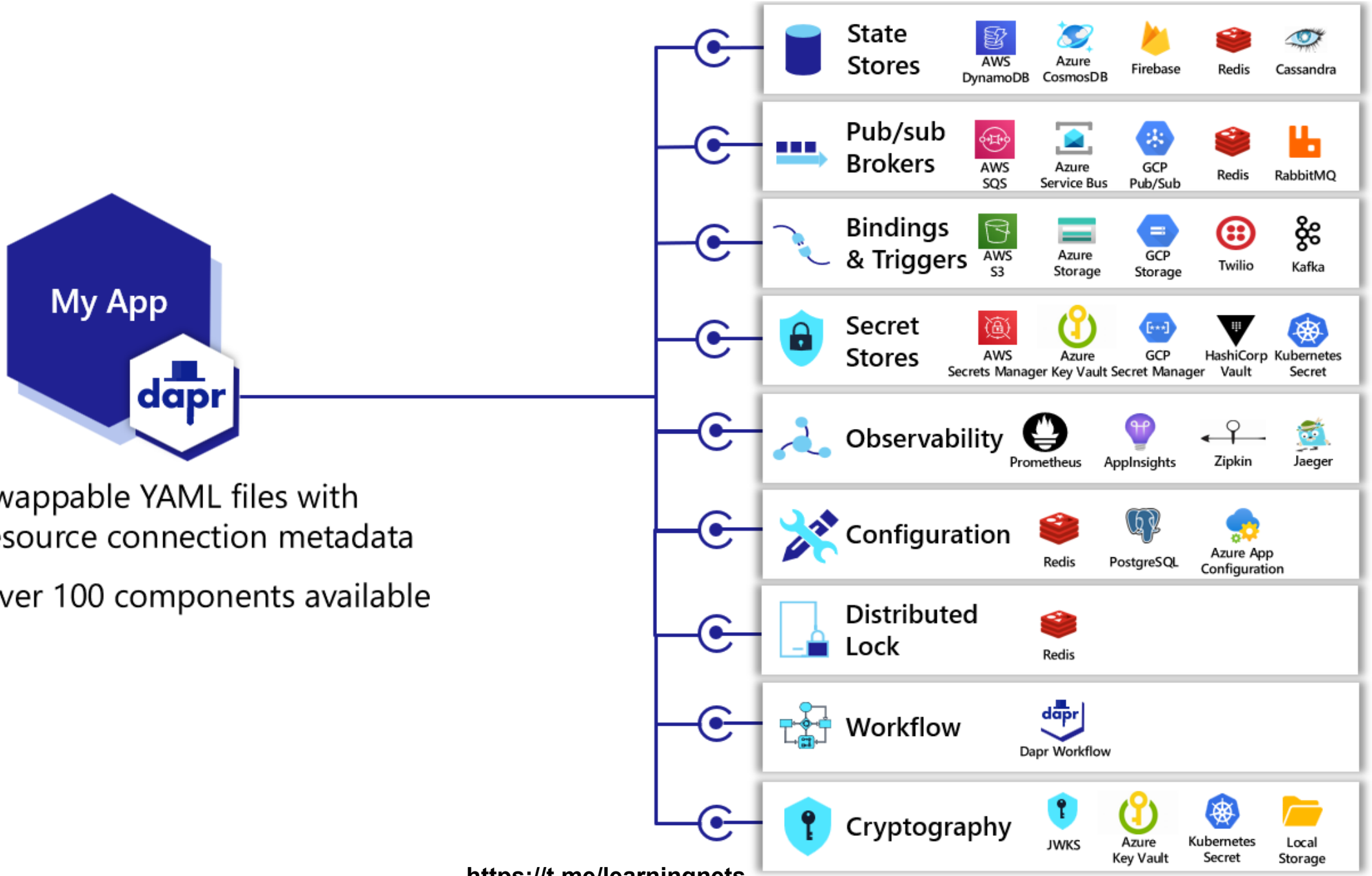
Sidecar

- Platform agnostic
- Code agnostic



Dapr Components

- Add more capabilities to Dapr
- In addition to the Service Mesh capabilities
- Represented as Components
- Each component is responsible for a specific capability
- Capabilities are implemented using different underlying mechanism
- Can be switched without changing the component spec



Component Schema

- YAML file defining the component
- Specifies the component type, authentication, metadata, scopes and more

Component Schema

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
auth:
 secretstore: [SECRET-STORE-NAME]
metadata:
  name: [COMPONENT-NAME]
  namespace: [COMPONENT-NAMESPACE]
spec:
  type: [COMPONENT-TYPE]
  version: v1
  initTimeout: [TIMEOUT-DURATION]
  ignoreErrors: [BOOLEAN]
  metadata:
  - name: [METADATA-NAME]
    value: [METADATA-VALUE]
scopes:
  - [APPID]
  - [APPID]
```

Dapr in Container Apps

- Dapr is deeply integrated into Container Apps
- Different configuration between service invocation (Service Mesh) and components

Service Invocation

- Enabled on the container app
- Almost no change to code
- Allows tracing
- Easy to configure

Dapr Components

- Defined on the Environment scope
- Attached to relevant container apps
- Have slightly leaner schema

Dapr Components Schema

```
componentType: [COMPONENT-TYPE]
version: v1
initTimeout: [TIMEOUT-DURATION]
ignoreErrors: [BOOLEAN]
metadata:
  - name: [METADATA-NAME]
    value: [METADATA-VALUE]
```

Dapr Component Scope

- Components can define scopes
- Basically app Id assigned to the container apps
- Only container app with an app Id defined as scope can use the component