



**distributed application runtime**

an intro

<https://dapr.io/>



# TL ; DR ;

- “ Dapr is a **portable**, event-driven runtime that makes it easy for any developer to build resilient, stateless and stateful applications that run on the **cloud and edge** and **embraces the diversity of languages and developer frameworks**.
- “ Dapr codifies the **best practices for building microservice applications** into open, independent, **building blocks** that enable you to build portable applications with the language and framework of your choice.

<https://docs.dapr.io/concepts/overview/>

## READ THE DOCS

<https://dapr.io>  
DISTRIBUTED APP RUNTIME

- OVERVIEW
- BUILDING BLOCKS
- COMPONENTS
- CONFIGURATION
- MIDDLEWARE PIPELINES
- OBSERVABILITY
- SECURITY

SO YOU'RE AN APP DEVELOPER: "Hello!"

## OVERVIEW

DAPR IS A PORTABLE  
EVENT-DRIVEN RUNTIME

- Make it EASY for devs to build RESILIENT stateful and stateless apps that RUN ON CLOUD and support DIVERSE languages and dev frameworks!

BUILD MICROSERVICES ON CLOUD OR EDGE

## INTRODUCTION TO DAPR

GOAL: BUILD RESILIENT APPS USING MICROSERVICES

HELP! CHALLENGES

STATE RECOVERY AFTER FAILURE

MANAGE SECRETS

DISCOVER + INVOKE OTHER MICROSERVICES SECURELY

YOU'LL BE RIGHT BESIDE YOU

WHAT'S YOUR DEV FRAM. PREFERENCE?

GO, JAVA, JET NODE, .NET, PYTHON, C++

AMAZON, AUKU, GCP, KUBERNETES

API LEVEL

ABSTRACTION

API

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

## SIDECAR ARCHITECTURE

USE ANY LANGUAGE OR FWK

DAPR RUNS IN ITS OWN CONTAINER

USE ALL BLOCKS OR JUST ONE

RUN IT ALONGSIDE YOUR APP CONTAINER AND BENEFIT FROM CLOUD SCALABILITY

READY FOR USE WITH EXISTING APPS

docs.dapr.io

github.com/dapr@daprdev

GET YOUR APP UP AND RUNNING!

ABSTRACT AWAY COMPLEXITY. Keep your app clean!

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

OBSERVABILITY AND SECURITY: Encryption, Standards

RUN ANYWHERE

DESIGNED FOR OPERATIONS

DEVELOPER LANGUAGE SDKs AND FRAMEWORKS

HOSTING ENVIRONMENT

EXPOSE BLOCK FUNCTIONALITY through language specific wrappers to HTTP/gRPC API

SHARE RUNTIME with DAPR + cross lang support for actors, functions etc.

CAN BE HOSTED IN MULTIPLE ENV: SELF HOSTED = local environment, EDGE/CLOUD = managed environment

"deploy sidecar in own VM invoke services over HTTP/gRPC"

"Services" Sidecar

APPLICATION "Main"

API "Connector"

RUN alongside "Pick what you need"

SERVICES DASHBOARD

MONITORING TOOLS for observability

DESIGNED FOR OPS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

DAPR CODIFIES BEST PRACTICES FOR BUILDING MICROSERVICES

API - EXPOSED EITHER AS A PROCESS OR AS A CONTAINER

APPS DO NOT NEED TO ADD DAPR RUNTIME INTO THAT CORE!

EASY INTEGRATION

CLEAN SEPARATION

"BUILDING BLOCKS"

APPLICATION

API

HTTP

gRPC

3-TIER ARCH

Client, Service, Database

BUT NOT WITH A MICROSERVICE ARCHITECTURE

AZURE AWS GCP ALIBABA

DAPR MICROSERVICE BUILDING BLOCKS

1 SERVICE-TO-SERVICE INVOCATION

2 STATE MANAGEMENT

3 PUBLISH & SUBSCRIBE

4 RESOURCE BINDINGS

5 ACTORS

6 OBSERVABILITY

7 SECRETS

RESILIENT DIRECT SECURE

PLUGGABLE KEY/VALUE HIGH AVAILABILITY

PUBLISH Events SUBSCRIBE TO Topics

TRIGGERS EVENT-DRIVEN ARCH

PATTERN FOR STATELESS AND STATEFUL OBJECTS

W3C TRACE CONTEXT STANDARDS, OPEN TELEMETRY

SECRETS MANAGEMENT TOOLS + INTEGRATION

HOSTING ENVIRONMENT

DAPR RUNS AS A "SIDECAR" CONTAINER

ALONGSIDE APP CONTAINER IN SAME POD

CONTAINER ENVIRONMENT e.g. Kubernetes

POD

APP CONTAINER

DAPR CONTAINER

OPS

"API" for message exchange b/w them!

RUN alongside "Pick what you need"

SERVICES DASHBOARD

MONITORING TOOLS for observability

DESIGNED FOR OPS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

## DAPR 1.0 IS HERE!

- aka.ms/dapr-v1.0

DAPR IS NOW PRODUCTION READY

Join a fast growing and engaged community

GREAT ECOSYSTEM AROUND DAPR

700+ components and growing daily

WIDE RANGE OF SDKs (Go, Java, .NET, GCP, Python, PHP, Node...)

IS LANGUAGE AGNOSTIC

OR USE API OVER HTTP OR gRPC

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

RUN ON LOCAL DEV MACHINE (Self Host)

RUN IN KUBERNETES MODE (cluster, cloud)

COMPONENTS

STATE STORES

PUBLISH & SUBSCRIBE

OBSERVABILITY

RESOURCE BINDINGS

DAPR PROCESS (VM, CONTAINER)

EACH RUNNING APP SERVICE (A, B) HAS A DAPR SIDECAR CONFIGURED TO USE THE BUILDING BLOCKS NEEDED.

START WITH: CLI + SAMPLES

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

RUN ON LOCAL DEV MACHINE (Self Host)

RUN IN KUBERNETES MODE (cluster, cloud)

COMPONENTS

STATE STORES

PUBLISH & SUBSCRIBE

OBSERVABILITY

RESOURCE BINDINGS

DAPR PROCESS (VM, CONTAINER)

EACH RUNNING APP SERVICE (A, B) HAS A DAPR SIDECAR CONFIGURED TO USE THE BUILDING BLOCKS NEEDED.

START WITH: CLI + SAMPLES

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

RUN ON LOCAL DEV MACHINE (Self Host)

RUN IN KUBERNETES MODE (cluster, cloud)

COMPONENTS

STATE STORES

PUBLISH & SUBSCRIBE

OBSERVABILITY

RESOURCE BINDINGS

DAPR PROCESS (VM, CONTAINER)

EACH RUNNING APP SERVICE (A, B) HAS A DAPR SIDECAR CONFIGURED TO USE THE BUILDING BLOCKS NEEDED.

START WITH: CLI + SAMPLES

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

RUN ON LOCAL DEV MACHINE (Self Host)

RUN IN KUBERNETES MODE (cluster, cloud)

COMPONENTS

STATE STORES

PUBLISH & SUBSCRIBE

OBSERVABILITY

RESOURCE BINDINGS

DAPR PROCESS (VM, CONTAINER)

EACH RUNNING APP SERVICE (A, B) HAS A DAPR SIDECAR CONFIGURED TO USE THE BUILDING BLOCKS NEEDED.

START WITH: CLI + SAMPLES

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

RUN ON LOCAL DEV MACHINE (Self Host)

RUN IN KUBERNETES MODE (cluster, cloud)

COMPONENTS

STATE STORES

PUBLISH & SUBSCRIBE

OBSERVABILITY

RESOURCE BINDINGS

DAPR PROCESS (VM, CONTAINER)

EACH RUNNING APP SERVICE (A, B) HAS A DAPR SIDECAR CONFIGURED TO USE THE BUILDING BLOCKS NEEDED.

START WITH: CLI + SAMPLES

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

RUN ON LOCAL DEV MACHINE (Self Host)

RUN IN KUBERNETES MODE (cluster, cloud)

COMPONENTS

STATE STORES

PUBLISH & SUBSCRIBE

OBSERVABILITY

RESOURCE BINDINGS

DAPR PROCESS (VM, CONTAINER)

EACH RUNNING APP SERVICE (A, B) HAS A DAPR SIDECAR CONFIGURED TO USE THE BUILDING BLOCKS NEEDED.

START WITH: CLI + SAMPLES

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

SIDECAR ARCHITECTURE

MICROSERVICES BUILDING BLOCKS

GO

PYTHON

ANY FRAMEWORK LANGUAGE WHERE

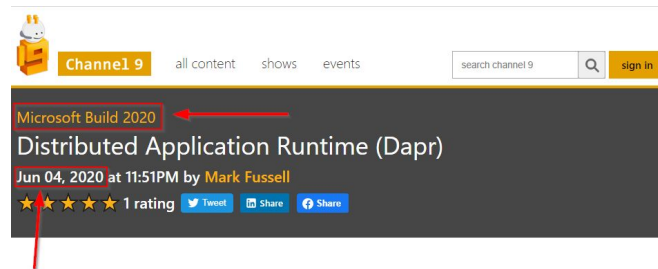
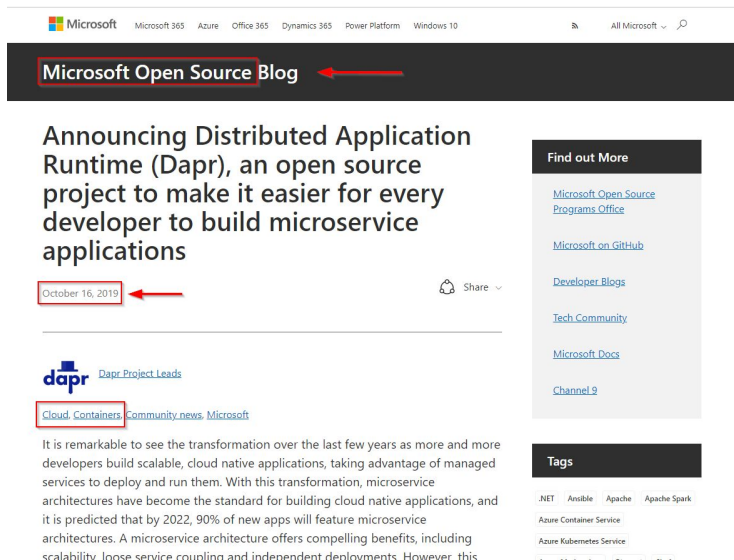
RUN ON LOCAL DEV MACHINE (Self Host)

R



# History

- originated within Microsoft
- announced **2019**, broader audience **Microsoft Build 2020**
  - IMHO: possibly because nobody wanted to use [Service Fabric](#)?



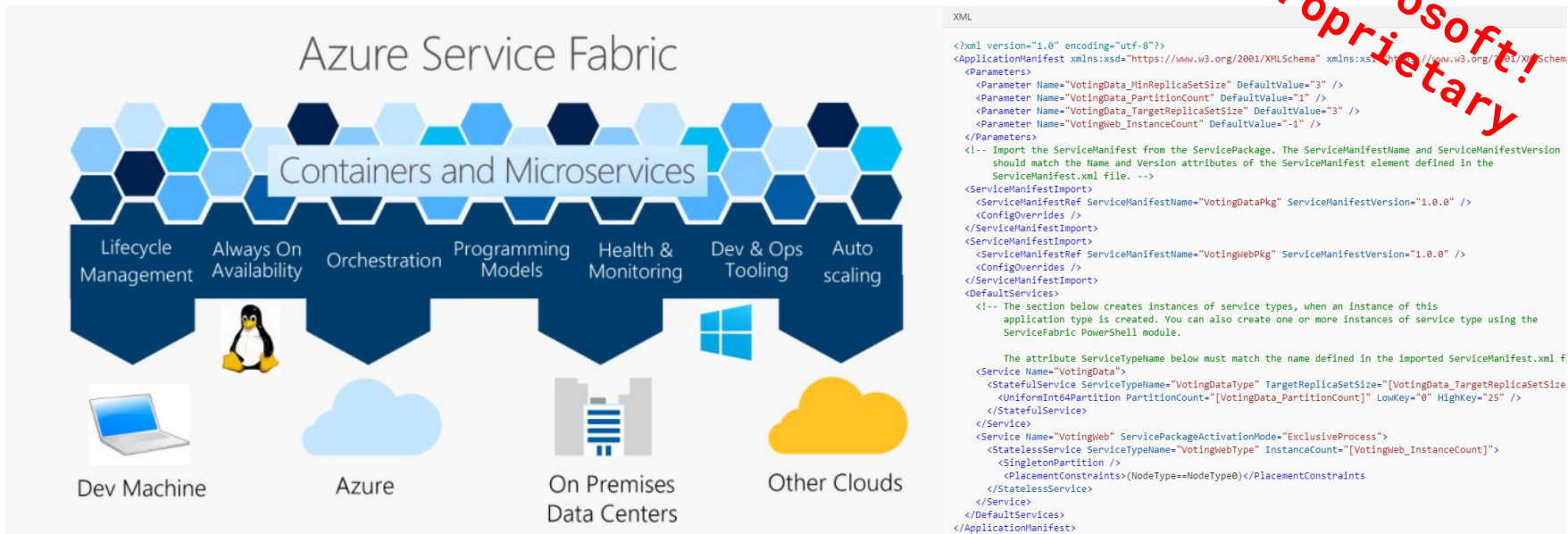
## What is holding back microservice development?



# (detour) Service Fabric

- Microsoft-specific technology not really widely used

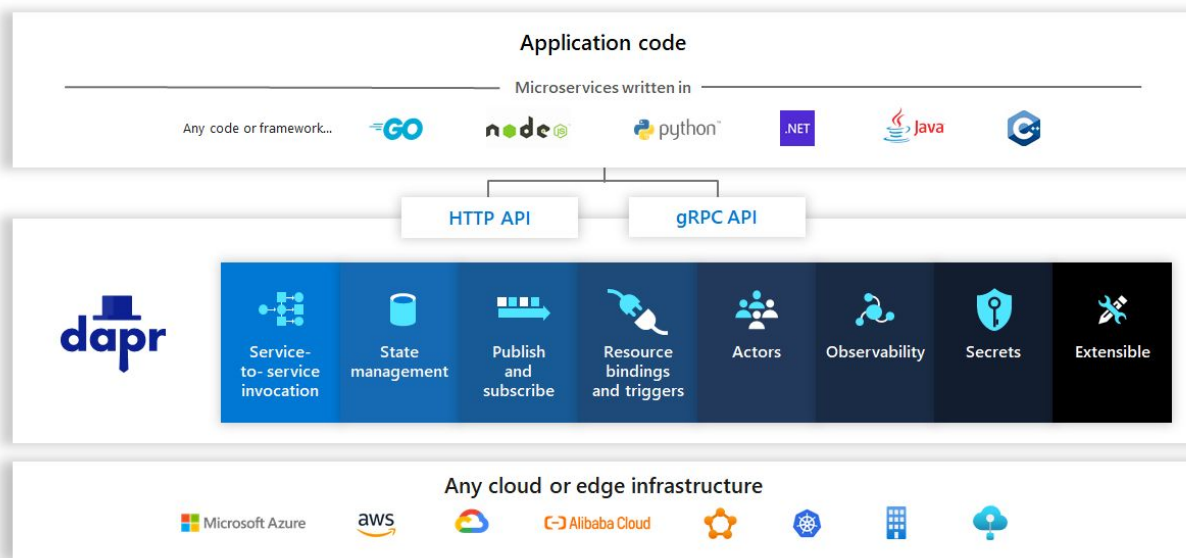
old Microsoft!  
proprietary



<https://docs.microsoft.com/en-us/azure/service-fabric/>

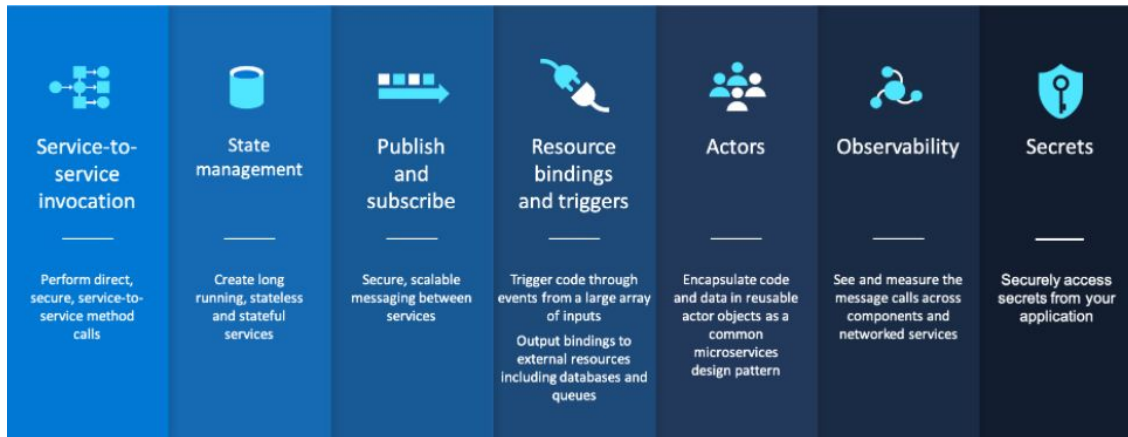
# Overview

- **Building-Blocks** to somehow standardize/simplify cloud development
- No custom SDKs **only HTTP/gRPC (\*)**
- Can be used with **any language** and with **multiple environments** (systems/cloud providers)



# Building Blocks

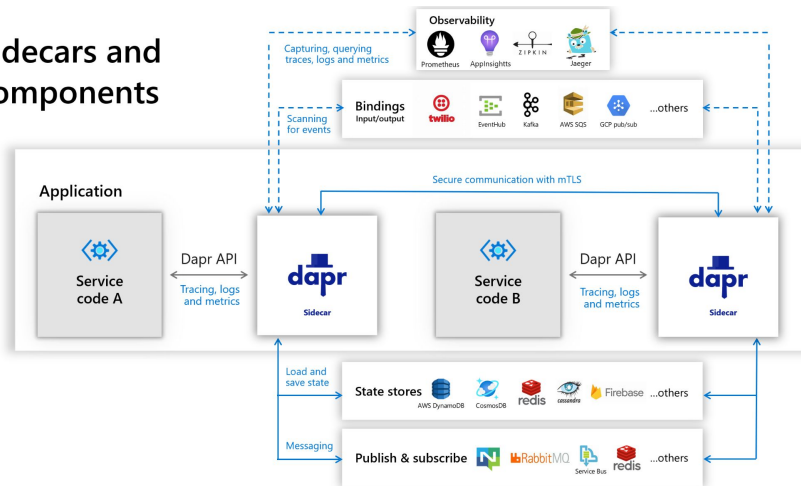
- **Typical components/services** needed for app development
  - <https://docs.dapr.io/reference/components-reference/>
  - “young project” - many components Alpha/Beta status!
  - cannot hide Microsoft Background (Azure components GA)
- Simple use/integration because of **HTTP/gRPC**
- Hide cloud platform-/technology-details - **abstraction layer**



# Implementation

- Development in the open <https://github.com/dapr/dapr> / [Community Calls - YouTube](#)
- [v1.2.0](#) current, [MIT License](#), go lang based (♥)
- Dapr logic/building-blocks implemented as sidecars
- Expose HTTP/gRPC API for invocation ([dapr API](#) / [dapr GRPC](#))
- Dapr sidecar takes care to interact with other services/components (abstraction)
- Support for microservice patterns

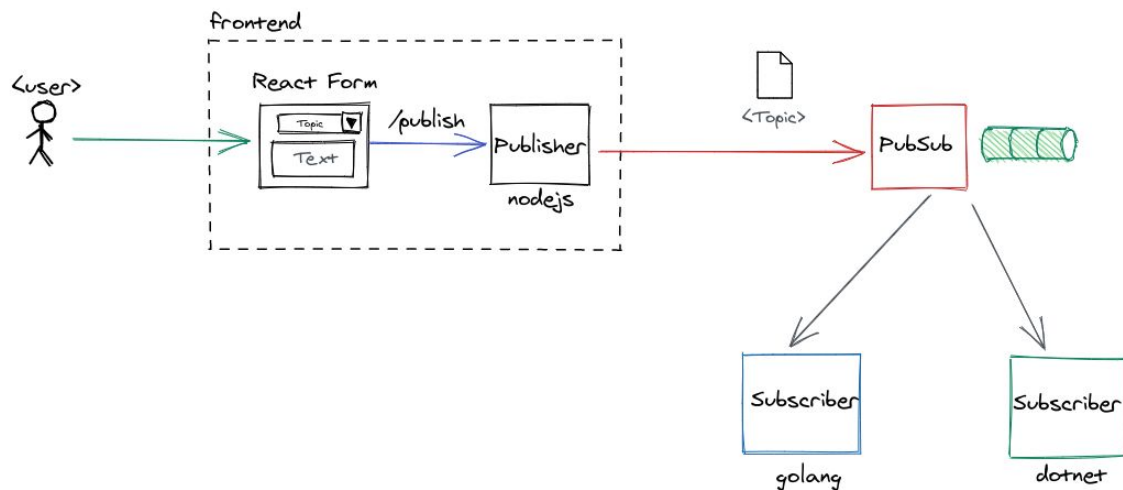
## Sidecars and components





# DEMO

- Dev example with simple pubsub logic
- dapr CLI
- k8s with dapr (use redis)
- k8s with dapr (use azure service-bus)



# Dev example - PupSub

- **expressjs** -> simple POST to <http://localhost:3500/v1.0/publish/pubsubname/<TOPIC>>

```
const daprPort = process.env.DAPR_HTTP_PORT || 3500;
const daprUrl = `http://localhost:${daprPort}/v1.0`;
const port = 8080;
const pubsubName = 'pubsub';

app.post('/publish', (req, res) => {
  console.log("Publishing: ", req.body);
  const publishUrl = `${daprUrl}/publish/${pubsubName}/${req.body.messageType}`;
  request( { uri: publishUrl, method: 'POST', json: req.body } );
  res.sendStatus(200);
});
```

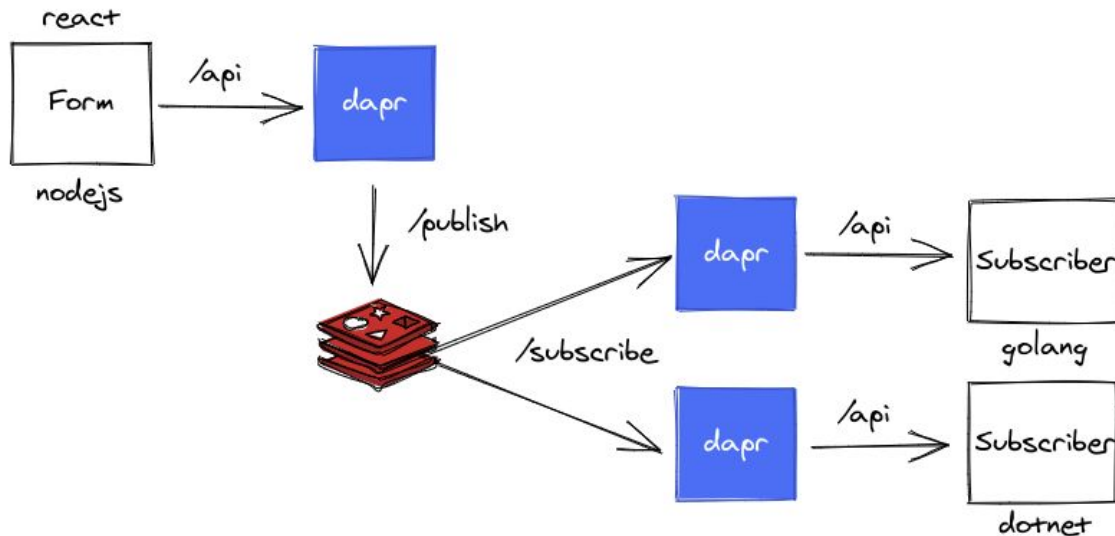
- **golang** -> register/consume (/dapr/subscribe) by HTTP-handler (\* [programmatic-subscription](#))

```
func subscribe() http.HandlerFunc {
  return func(w http.ResponseWriter, r *http.Request) {
    sub := []Subscribe{
      {
        PubSubName: pubSubName,
        Topic:      "ALL",
        Route:      "receive_all",
      },
      {
        PubSubName: pubSubName,
        Topic:      "Topic1",
        Route:      "receive_b",
      },
    },
  }
}
```

```
func procMessage(route string) http.HandlerFunc {
  return func(w http.ResponseWriter, r *http.Request) {
    msg, err := getMessage(r.Body)
    defer r.Body.Close()
    if err != nil {
      http.Error(w, err.Error(), 500)
      return
    }
    log.Printf("📄 via '%s', for '%s' with message '%s'", route, msg.Topic, msg.Data.Message)
    w.WriteHeader(http.StatusOK)
  }
}
```

# Dev example - PupSub

- `dapr run --app-id golang-subscriber --app-port 3000 ./golang-subscriber`
- `dapr run --app-id dotnet-subscriber --app-port 5000 ./output/dotnet-subscriber`
- `dapr run --app-id react-form --app-port 8080 npm run start`



```
make dap-run
workstation scope=dapr.runtime.actor type=log ver=1.1.2
== APP == info: Microsoft.Hosting.Lifetime[0]
== APP == Now listening on: http://localhost:5000
== APP == info: Microsoft.Hosting.Lifetime[0]
== APP == Now listening on: https://localhost:5001
== APP == info: Microsoft.Hosting.Lifetime[0]
== APP == Application started. Press Ctrl+C to shut down.
== APP == info: Microsoft.Hosting.Lifetime[0]
== APP == Hosting environment: Production
== APP == info: Microsoft.Hosting.Lifetime[0]
== APP == Content root path: /home/henrik/Development/dapr/pub-sub/dotnet-subscriber
INFO[0000] app is subscribed to the following topics: [Topic2 ALL] through pubsub=pubsub app_id=dotnet-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] dapr initialized. Status: Running. Init Elapsed 285.870498ms app_id=dotnet-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] placement tables updated, version: 0 app_id=dotnet-subscriber instance=henrik-workstation scope=dapr.runtime.actor.internal.placement type=log ver=1.1.2
[+] Updating metadata for app command: ./output/dotnet-subscriber
[+] You're up and running! Both Dapr and your app logs will appear here.
== APP == info: dotnet_subscriber.Controllers.PubSubController[0]
== APP == message 'hello, dapr world!' via '/receive_all' for 'ALL'
```

```
make dap-run
rpc.internal type=log ver=1.1.2
INFO[0000] enabled gRPC metrics middleware app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
rpc.internal type=log ver=1.1.2
INFO[0000] internal gRPC server is running on port 39695 app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] application protocol: http. waiting on port 3000. This will block until the app is listening on that port. app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] application discovered on port 3000 app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
== APP == 2021/05/29 14:31:24 "GET http://127.0.0.1:3000/dapr/config HTTP/1.1" from 127.0.0.1:47190 - 404 198 in 7.171µs
INFO[0000] application configuration loaded app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] actor runtime started. actor idle timeout: 1h0m0s. actor scan interval: 30s app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime.actor type=log ver=1.1.2
== APP == 2021/05/29 14:31:24 "GET http://127.0.0.1:3000/dapr/subscribe HTTP/1.1" from 127.0.0.1:47190 - 200 122B in 41.13µs
INFO[0000] app is subscribed to the following topics: [Topic1 ALL] through pubsub=pubsub app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] dapr initialized. Status: Running. Init Elapsed 3.983924ms app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] placement tables updated, version: 0 app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime.actor.internal.placement type=log ver=1.1.2
[+] Updating metadata for app command: ./golang-subscriber
[+] You're up and running! Both Dapr and your app logs will appear here.
== APP == 2021/05/29 14:32:48 via '/receive_all', for 'ALL' with message 'hello, dapr world!'
== APP == 2021/05/29 14:32:48 "POST http://127.0.0.1:3000/receive_all HTTP/1.1" from 127.0.0.1:47358 - 200 0B in 112.072µs
```

React App localhost:8080

Pub-Sub Sample

Select Message Type

ALL

Enter message

hello, dapr world!

Submit

```
make dap-run
ime.grpc.internal type=log ver=1.1.2
INFO[0000] internal gRPC Server is running on port 41865 app_id=react-form instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] application protocol: http. waiting on port 8080. This will block until the app is listening on that port. app_id=react-form instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
== APP ==
== APP == > react-form@1.0.0 start /home/henrik/Development/dapr/pub-sub/react-form
== APP == > node server.js
== APP == Listening on port 8080!
INFO[0000] application discovered on port 8080 app_id=react-form instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] actor runtime started. actor idle timeout: 1h0m0s. actor scan interval: 30s app_id=react-form instance=henrik-workstation scope=dapr.runtime.actor type=log ver=1.1.2
ERROR[0000] error getting topics from app: invalid character '<' looking for beginning of value app_id=react-form instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] dapr initialized. Status: Running. Init Elapsed 212.186236ms app_id=react-form instance=henrik-workstation scope=dapr.runtime type=log ver=1.1.2
INFO[0000] placement tables updated, version: 0 app_id=react-form instance=henrik-workstation scope=dapr.runtime.actor.internal.placement type=log ver=1.1.2
[+] Updating metadata for app command: npm run start
[+] You're up and running! Both Dapr and your app logs will appear here.
== APP == Publishing: { messageType: 'ALL', message: 'hello, dapr world!' }
```

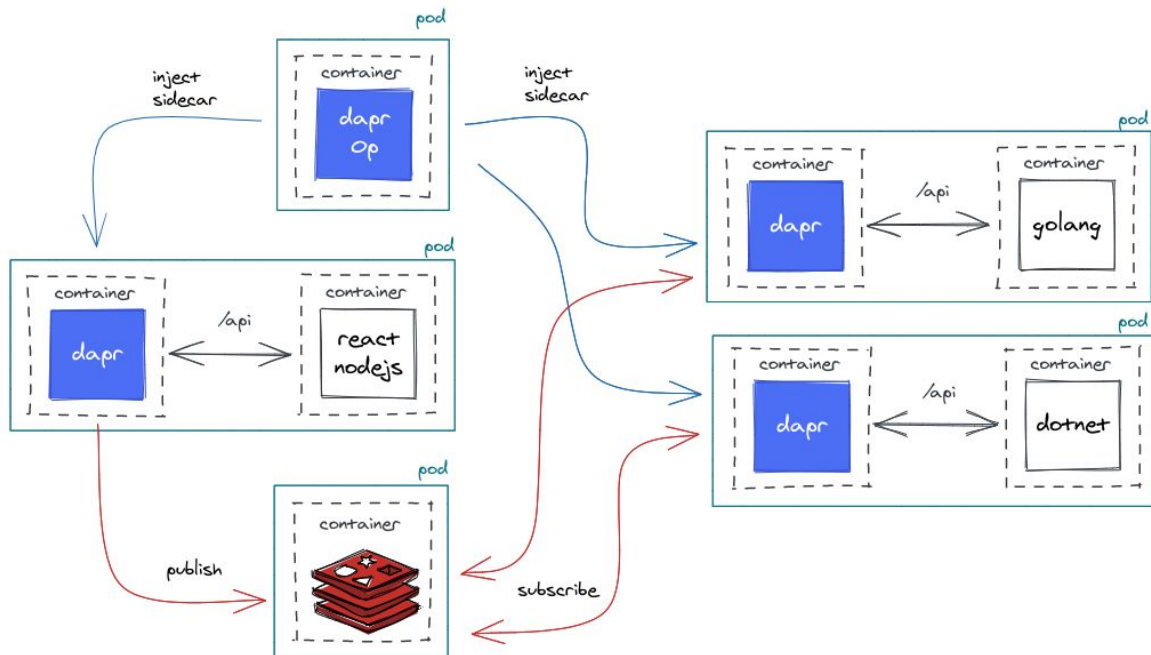
# Dev example - PupSub @ k8s

- k8s deployment with dapr sidecars

```

1  apiVersion: dapr.io/v1alpha1
2  kind: Component
3  metadata:
4    name: pubsub
5  spec:
6    type: pubsub.redis
7    version: v1
8    metadata:
9      - name: redisHost
10        value: redis-master:6379
11      - name: redisPassword
12        secretKeyRef:
13          name: redis
14          key: redis-password
15  auth:
16    secretStore: kubernetes

```





```
> kubectll logs -f --selector app=dotnet-subscriber -c dotnet-subscriber
info: Microsoft.Hosting.Lifetime[0]
Now listening on: http://[::]:5000
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
Content root path: /opt/dotnet-subscriber
info: dotnet_subscriber.Controllers.PubSubController[0]
message 'hello, dapr world!' via '/receive_all' for 'ALL'
info: dotnet_subscriber.Controllers.PubSubController[0]
message 'hello, dapr world!' via '/receive_all' for 'ALL'
```

## Pub-Sub Sample

Select Message Type

ALL

Enter message

hello, dapr world!

Submit

DEMO/k8s

```
> kubectll logs -f --selector app=golang-subscriber -c golang-subscriber
up and running @ :3000
2021/05/29 14:37:07 "GET http://127.0.0.1:3000/dapr/config HTTP/1.1" from 127.0.0.1:46106 - 404 198 in 7.15µs
2021/05/29 14:37:07 "GET http://127.0.0.1:3000/dapr/subscribe HTTP/1.1" from 127.0.0.1:46106 - 200 122B in 77.291µs
2021/05/29 14:44:15 via '/receive_all', for 'ALL' with message 'hello, dapr world!'
2021/05/29 14:44:15 "POST http://127.0.0.1:3000/receive_all HTTP/1.1" from 127.0.0.1:54246 - 200 0B in 124.671µs
2021/05/29 14:49:25 via '/receive_all', for 'ALL' with message 'hello, dapr world!'
2021/05/29 14:49:25 "POST http://127.0.0.1:3000/receive_all HTTP/1.1" from 127.0.0.1:60356 - 200 0B in 71.481µs
```

```
> kubectll logs -f --selector app=react-form -c react-form
> react-form@1.0.0 server
> nodeemon server.js
```

```
[nodemon] 2.0.7
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node server.js'
Listening on port 8080!
Publishing: { messageType: 'ALL', message: 'hello, dapr world!' }
Publishing: { messageType: 'ALL', message: 'hello, dapr world!' }
```

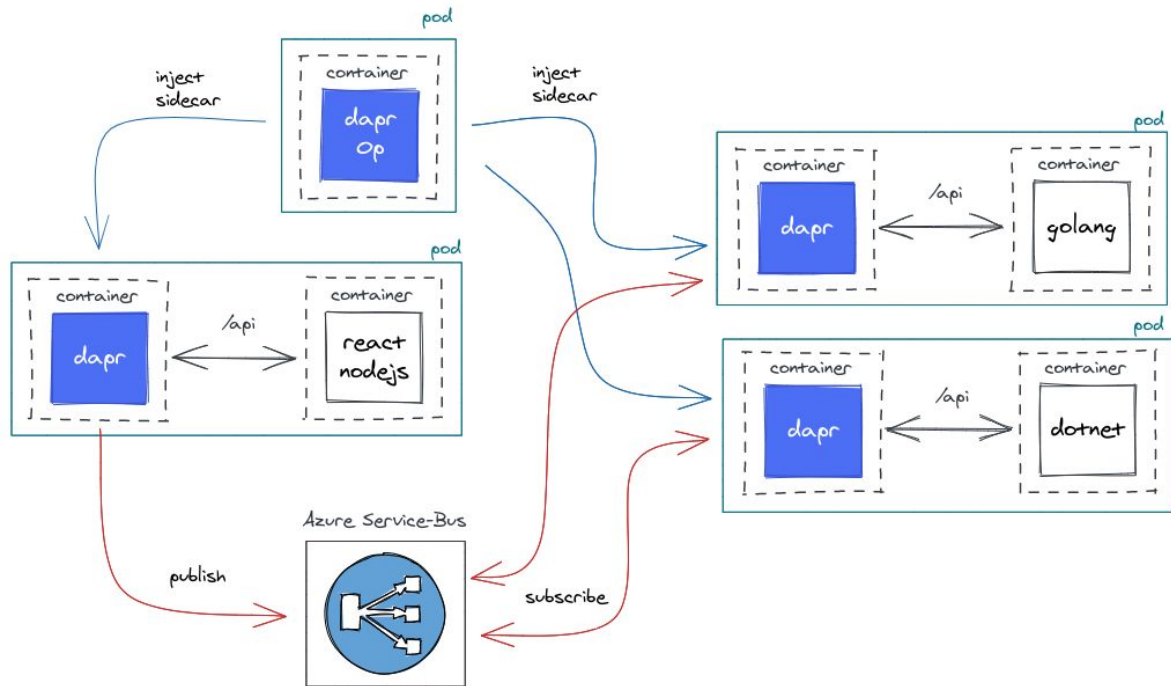
# Dev example - PupSub @ k8s

- swap redis with Azure Service-Bus

```

1  apiVersion: dapr.io/v1alpha1
2  kind: Component
3  metadata:
4    name: pubsub
5  spec:
6    type: pubsub.azure.servicebus
7    version: v1
8    metadata:
9      - name: connectionString # Require
10        secretKeyRef:
11          name: az-sb
12          key: connstr
13  auth:
14    secretStore: kubernetes

```



# Dapr examples / use-cases

- [How Alibaba is using Dapr | Dapr Blog](#)
- [Running Dapr in production at Roadwork | Dapr Blog](#)
- [Microsoft Customer Story-ZEISS accelerates cloud-first development on Azure and streamlines order processing](#)
- [Microsoft Customer Story-Ignition Group speeds development and payment processing using Dapr and Azure](#)

# Links

- <https://docs.dapr.io/getting-started/>
- <https://github.com/dapr/quickstarts/tree/master/pub-sub>
- <https://docs.dapr.io/developing-applications/building-blocks/pubsub/pubsub-overview/>
- <https://docs.dapr.io/reference/components-reference/supported-pubsub/setup-azure-servicebus/>
- PubSub with Azure Service Bus: <https://www.youtube.com/watch?v=umrU1frZqKk>
- <https://github.com/bihe/dapr-intro>
- <https://blog.dapr.io/posts/2021/03/02/a-visual-guide-to-dapr/>