distributed application runtime

an intro

https://dapr.io/

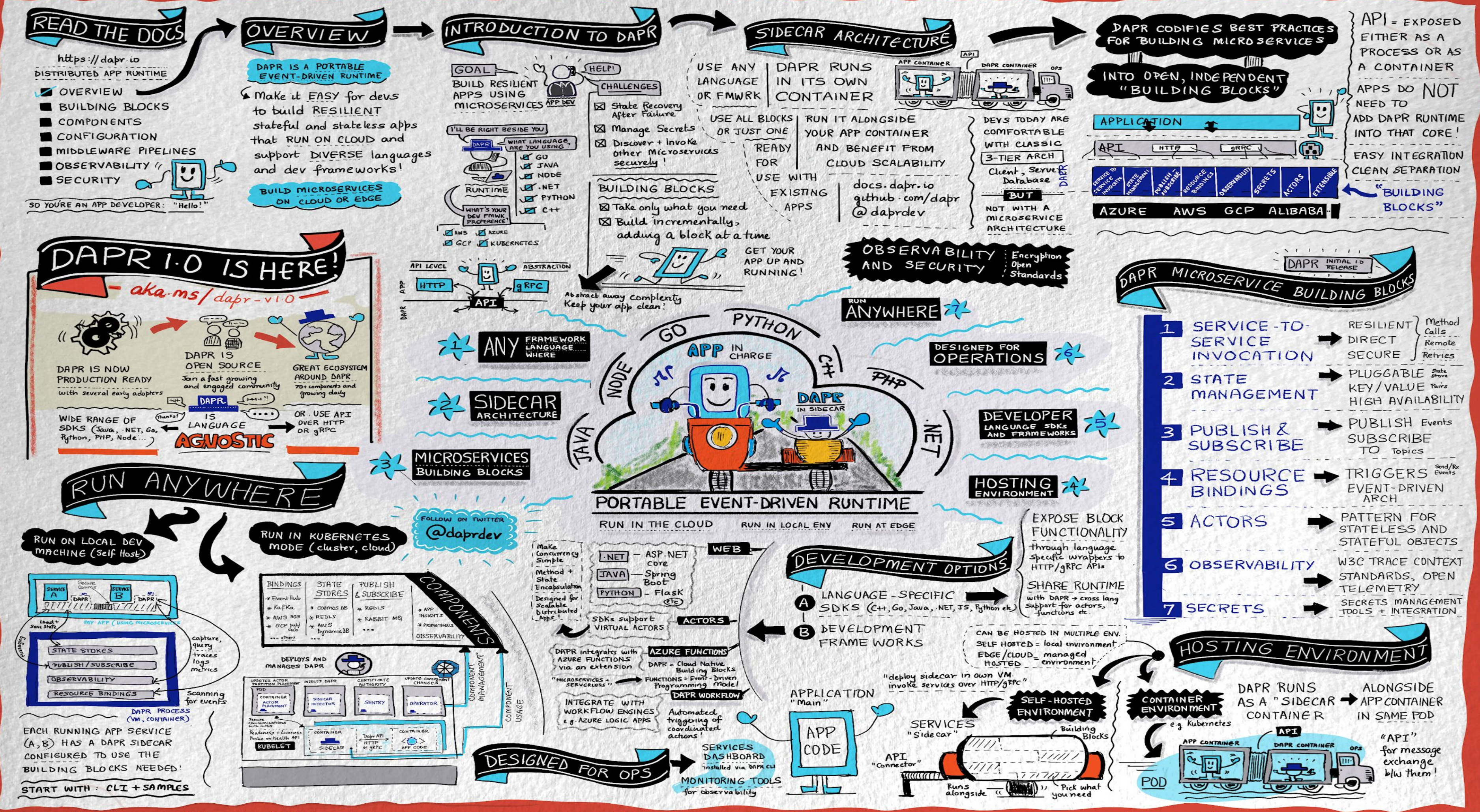
Henrik Binggl | https://github.com/bihe | https://twitter.com/HenrikBinggl

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
APP DEV
HELP!
CHALLENGES
State Recovery After Failure
Manage Secrets
Discover + Invoke other Microservices securely!
I'LL BE RIGHT BESIDE YOU
DAPR
WHAT LANGUAGE ARE YOU USING?
GO
JAVA
NODE
.NET
PYTHON
C++
RUNTIME
WHAT'S YOUR DEV FMWK PREFERENCE?
AWS
AZURE
GCP
KUBERNETES
BUILDING BLOCKS
Take only what you need
Build incrementally, adding a block at a time
GET YOUR APP UP AND RUNNING!
API LEVEL
ABSTRACTION
HTTP
gRPC
API
APP
DAPR
Abstract away complexity
Keep your app clean!
SIDECAR ARCHITECTURE
USE ANY LANGUAGE OR FMWRK
DAPR RUNS IN ITS OWN CONTAINER
APP CONTAINER
API
DAPR CONTAINER
OPS
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
DEVS TODAY ARE COMFORTABLE WITH CLASSIC
3-TIER ARCH
Client, Server, Database
BUT
NOT WITH A MICROSERVICE ARCHITECTURE
DAPR CODIFIES BEST PRACTICES FOR BUILDING MICROSERVICES
INTO OPEN, INDEPENDENT "BUILDING BLOCKS"
APPLICATION
API
HTTP
gRPC
SERVICE TO SERVICE INVOCATION
STATE MANAGEMENT
PUBLISH SUBSCRIBE
RESOURCE BINDINGS
OBSERVABILITY
SECRETS
ACTORS
EXTENSIBLE
"BUILDING BLOCKS"
AZURE
AWS
GCP
ALIBABA
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
DAPR 1.0 IS HERE!
aka.ms/dapr-v1.0
DAPR IS NOW PRODUCTION READY
with several early adopters
DAPR IS OPEN SOURCE
Join a fast growing and engaged community
GREAT ECOSYSTEM AROUND DAPR
70+ components and growing daily
WIDE RANGE OF SDKS (Java, .NET, Go, Python, PHP, Node...)
Thanks!
DAPR
IS LANGUAGE
AGNOSTIC
OR USE API OVER HTTP OR gRPC
OBSERVABILITY AND SECURITY
Encryption
Open Standards
RUN ANYWHERE
1 ANY FRAMEWORK LANGUAGE WHERE
2 SIDECAR ARCHITECTURE
3 MICROSERVICES BUILDING BLOCKS
DESIGNED FOR OPERATIONS 6
DEVELOPER LANGUAGE SDKs AND FRAMEWORKS 5
HOSTING ENVIRONMENT 4
RUN ANYWHERE 7
GO
PYTHON
NODE
C#
PHP
JAVA
.NET
APP IN CHARGE
DAPR IN SIDECAR
PORTABLE EVENT-DRIVEN RUNTIME
RUN IN THE CLOUD
RUN IN LOCAL ENV
RUN AT EDGE
FOLLOW ON TWITTER @daprdev
RUN ANYWHERE
RUN ON LOCAL DEV MACHINE (Self Host)
RUN IN KUBERNETES MODE (cluster, cloud)
SERVICE A
SERVICE B
Secure Comms
DAPR
DAPR
API
Load + Save State
MY APP (USING MICROSERVICES)
Invocation
STATE STORES
PUBLISH / SUBSCRIBE
OBSERVABILITY
RESOURCE BINDINGS
capture, query traces logs metrics
Scanning for events
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
BINDINGS
Event Hub
Kafka
AWS SQS
GCP pub sub
... others
STATE STORES
Cosmos DB
REDIS
AWS DynamoDB
PUBLISH & SUBSCRIBE
REDIS
RABBIT MQ
...
APP INSIGHTS
PROMETHEUS
OBSERVABILITY
COMPONENTS
COMPONENT MANAGEMENT
COMPONENT USAGE
DEPLOYS AND MANAGES DAPR
UPDATES ACTOR PARTITION PLACEMENT
INJECTS DAPR
CERTIFICATE AUTHORITY
UPDATE COMPONENT CHANGES
POD
CONTAINER
ACTOR PLACEMENT
SIDECAR INJECTOR
SENTRY
OPERATOR
Secure communications with mTLS
Readiness + Liveness Probe on health API
KUBELET
CONTAINER
SIDECAR
Dapr API HTTP or gRPC
CONTAINER
APP CODE
DESIGNED FOR OPS
SERVICES DASHBOARD
installed via DAPR CLI
MONITORING TOOLS
for observability
Make Concurrency Simple
Method + State Encapsulation
Designed for Scalable Distributed Apps
.NET - ASP.NET Core
JAVA - Spring Boot
PYTHON - Flask
etc
WEB
SDKs support VIRTUAL ACTORS
ACTORS
DAPR integrates with AZURE FUNCTIONS via an extension
AZURE FUNCTIONS
DAPR = Cloud Native Building Blocks
"MICROSERVICES + SERVERLESS"
FUNCTIONS = Event-Driven Programming Model
DAPR WORKFLOW
INTEGRATE WITH WORKFLOW ENGINES e.g. AZURE LOGIC APPS
Automated triggering of coordinated actions!
DEVELOPMENT OPTIONS
A LANGUAGE-SPECIFIC SDKS (C++, Go, Java, .NET, JS, Python etc)
B DEVELOPMENT FRAME WORKS
EXPOSE BLOCK FUNCTIONALITY
through language specific wrappers to HTTP/gRPC APIs
SHARE RUNTIME
with DAPR → cross lang support for actors, functions etc.
CAN BE HOSTED IN MULTIPLE ENV.
SELF HOSTED = local environment
EDGE / CLOUD HOSTED = managed environment
"deploy sidecar in own VM. invoke services over HTTP/gRPC"
APPLICATION "Main"
APP CODE
SELF-HOSTED ENVIRONMENT
SERVICES "Sidecar"
Building Blocks
API "Connector"
Runs alongside
Pick what you need
DAPR MICROSERVICE BUILDING BLOCKS
DAPR INITIAL 1.0 RELEASE
1 SERVICE-TO-SERVICE INVOCATION → RESILIENT, DIRECT, SECURE — Method Calls, Remote Retries
2 STATE MANAGEMENT → PLUGGABLE State Store, KEY / VALUE Pairs, HIGH AVAILABILITY
3 PUBLISH & SUBSCRIBE → PUBLISH Events, SUBSCRIBE TO Topics
4 RESOURCE BINDINGS → TRIGGERS Send/Rx Events, EVENT-DRIVEN ARCH
5 ACTORS → PATTERN FOR STATELESS AND STATEFUL OBJECTS
6 OBSERVABILITY → W3C TRACE CONTEXT STANDARDS, OPEN TELEMETRY
7 SECRETS → SECRETS MANAGEMENT TOOLS + INTEGRATION
HOSTING ENVIRONMENT
CONTAINER ENVIRONMENT
e.g. Kubernetes
DAPR RUNS AS A "SIDECAR CONTAINER"
ALONGSIDE APP CONTAINER IN SAME POD
APP CONTAINER
API
DAPR CONTAINER
OPS
POD
"API" for message exchange b/w them!

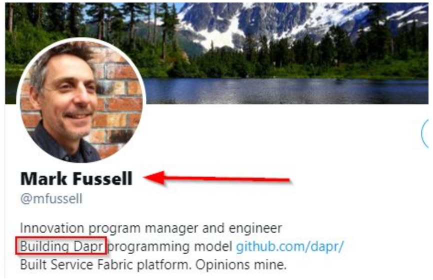
https://aka.ms/dapr-v1.0

Sketchnotes by @nitya

https://azure.cloud-skills.dev

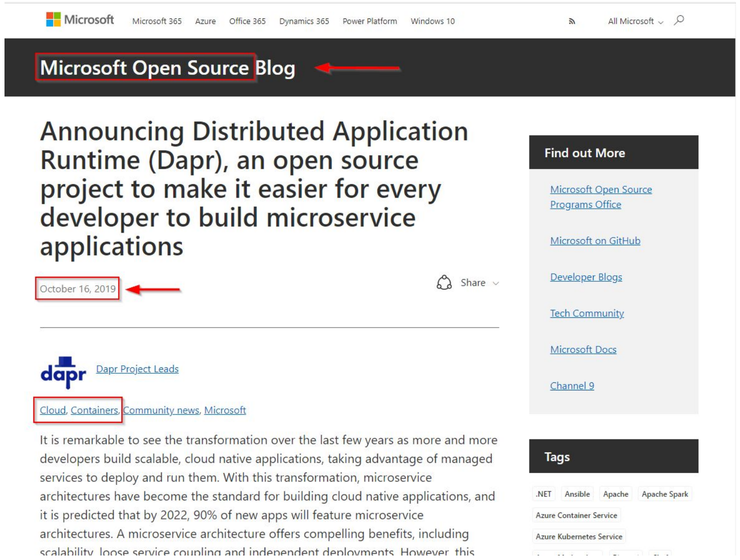
History

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



Mark Fussell
@mfussell

Innovation program manager and engineer
Building Dapr programming model github.com/dapr/
Built Service Fabric platform. Opinions mine.



Microsoft Open Source Blog

Announcing Distributed Application Runtime (Dapr), an open source project to make it easier for every developer to build microservice applications

October 16, 2019

Dapr Project Leads

Cloud, Containers, Community news, Microsoft

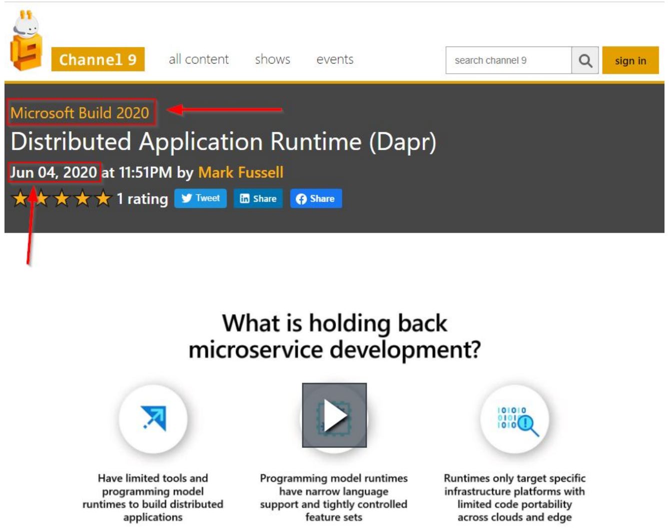
It is remarkable to see the transformation over the last few years as more and more developers build scalable, cloud native applications, taking advantage of managed services to deploy and run them. With this transformation, microservice architectures have become the standard for building cloud native applications, and it is predicted that by 2022, 90% of new apps will feature microservice architectures. A microservice architecture offers compelling benefits, including scalability, loose service coupling and independent deployments. However, this

Find out More

- Microsoft Open Source Programs Office
- Microsoft on GitHub
- Developer Blogs
- Tech Community
- Microsoft Docs
- Channel 9

Tags

.NET · Ansible · Apache · Apache Spark · Azure Container Service · Azure Kubernetes Service



Microsoft Build 2020

Distributed Application Runtime (Dapr)

Jun 04, 2020 at 11:51PM by Mark Fussell

★★★★★ 1 rating

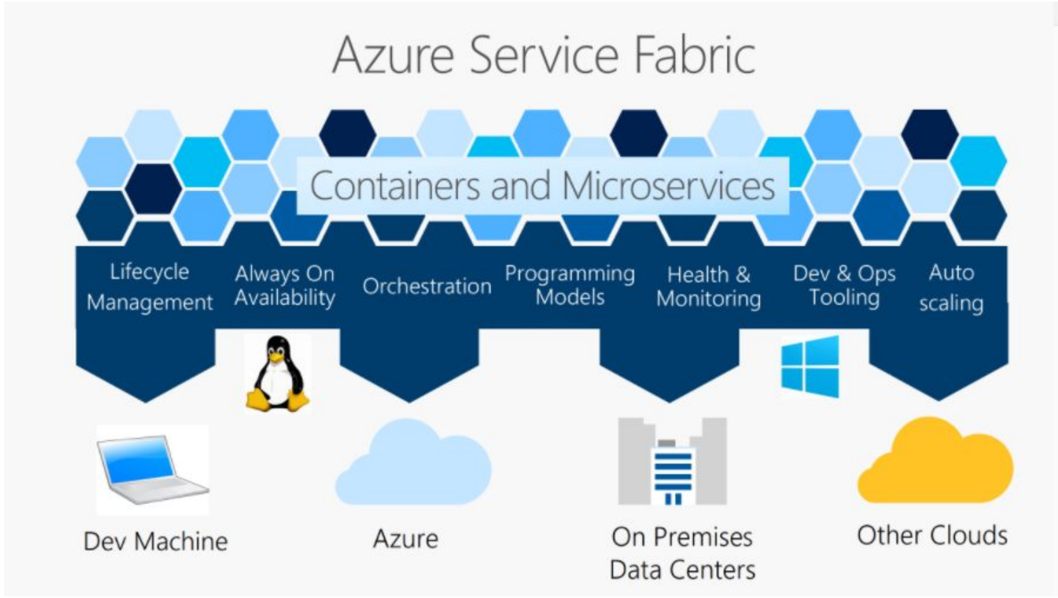
What is holding back microservice development?

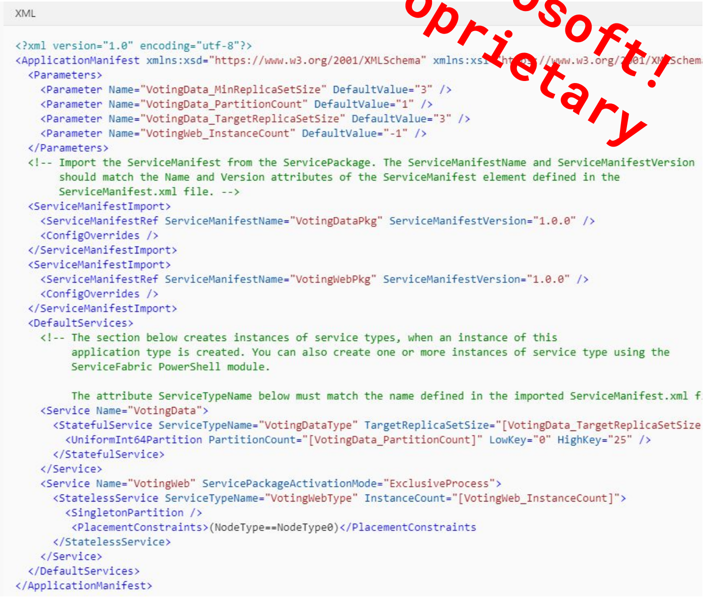
- Have limited tools and programming model runtimes to build distributed applications
- Programming model runtimes have narrow language support and tightly controlled feature sets
- Runtimes only target specific infrastructure platforms with limited code portability across clouds and edge

(detour) Service Fabric

- Microsoft-specific technology not really widely used

old Microsoft proprietary!



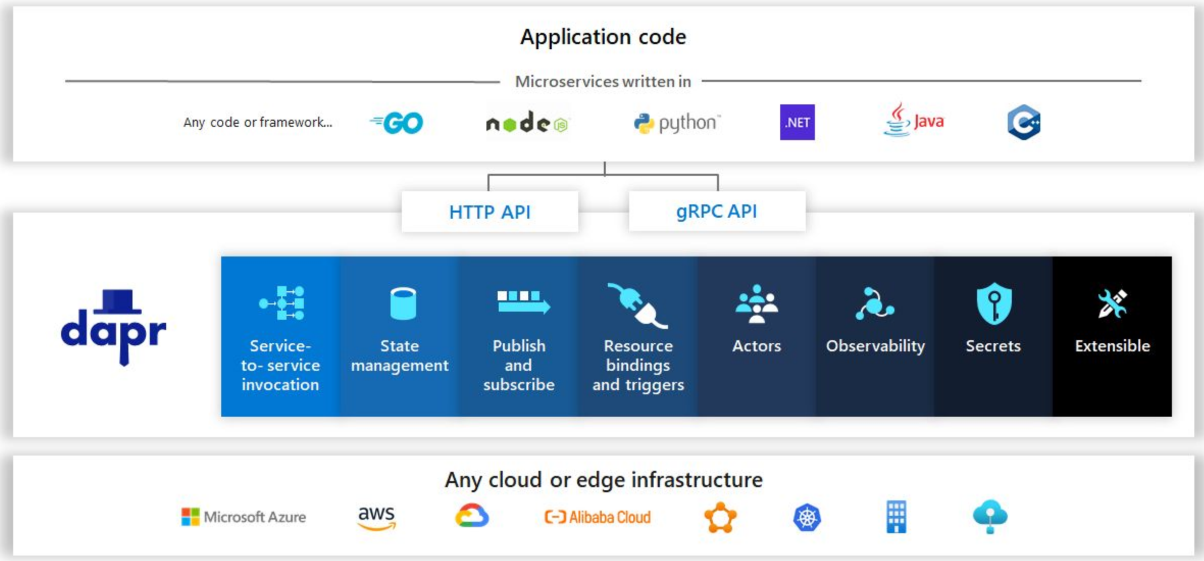

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<ApplicationManifest xmlns:xsd="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchem
  <Parameters>
    <Parameter Name="VotingData_MinReplicaSetSize" DefaultValue="3" />
    <Parameter Name="VotingData_PartitionCount" DefaultValue="1" />
    <Parameter Name="VotingData_TargetReplicaSetSize" DefaultValue="3" />
    <Parameter Name="VotingWeb_InstanceCount" DefaultValue="-1" />
  </Parameters>
  <!-- Import the ServiceManifest from the ServicePackage. The ServiceManifestName and ServiceManifestVersion
       should match the Name and Version attributes of the ServiceManifest element defined in the
       ServiceManifest.xml file. -->
  <ServiceManifestImport>
    <ServiceManifestRef ServiceManifestName="VotingDataPkg" ServiceManifestVersion="1.0.0" />
    <ConfigOverrides />
  </ServiceManifestImport>
  <ServiceManifestImport>
    <ServiceManifestRef ServiceManifestName="VotingWebPkg" ServiceManifestVersion="1.0.0" />
    <ConfigOverrides />
  </ServiceManifestImport>
  <DefaultServices>
    <!-- The section below creates instances of service types, when an instance of this
         application type is created. You can also create one or more instances of service type using the
         ServiceFabric PowerShell module.

         The attribute ServiceTypeName below must match the name defined in the imported ServiceManifest.xml f
    <Service Name="VotingData">
      <StatefulService ServiceTypeName="VotingDataType" TargetReplicaSetSize="[VotingData_TargetReplicaSetSize
        <UniformInt64Partition PartitionCount="[VotingData_PartitionCount]" LowKey="0" HighKey="25" />
      </StatefulService>
    </Service>
    <Service Name="VotingWeb" ServicePackageActivationMode="ExclusiveProcess">
      <StatelessService ServiceTypeName="VotingWebType" InstanceCount="[VotingWeb_InstanceCount]">
        <SingletonPartition />
         <PlacementConstraints>(NodeType==NodeType0)</PlacementConstraints
      </StatelessService>
    </Service>
  </DefaultServices>
</ApplicationManifest>
```

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

Overview

- **Building-Blocks** to somehow standardize/simplify cloud development
- Can be used with **any language** and with **multiple environments** (systems/cloud providers)



any language

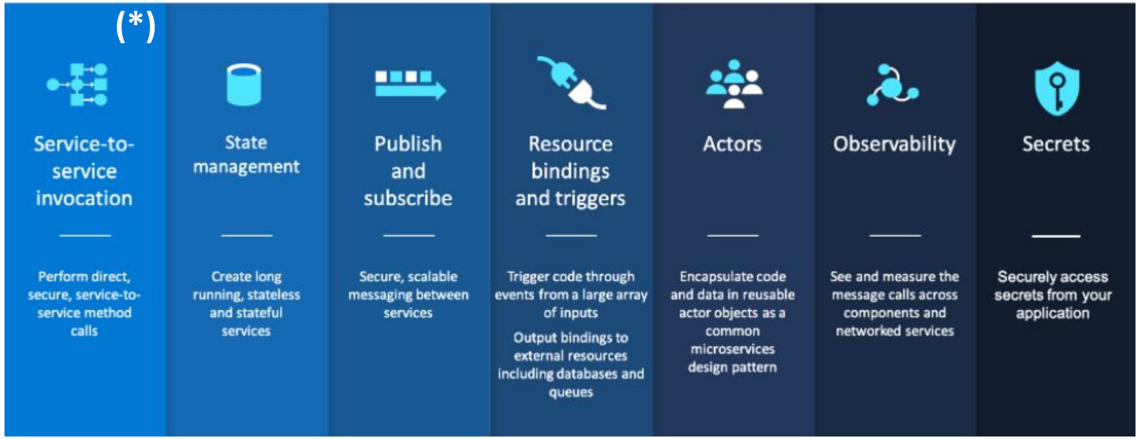
[Dapr (github.com)](https://github.com)

Open Source

multi cloud

Building Blocks

- Typical **components/services** needed for app development
  - cannot hide Microsoft Background (Azure components GA)
- **Simple** use/integration because of **HTTP/gRPC -> any language**
- **Abstract** cloud platform-/technology-details **-> multi cloud**



(*)

Service-to-service invocation
Perform direct, secure, service-to-service method calls

State management
Create long running, stateless and stateful services

Publish and subscribe
Secure, scalable messaging between services

Resource bindings and triggers
Trigger code through events from a large array of inputs
Output bindings to external resources including databases and queues

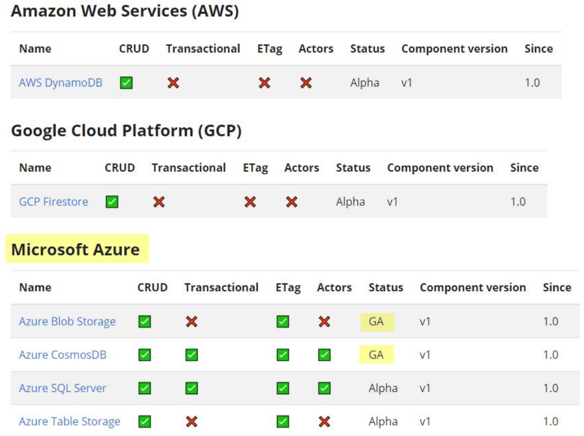
Actors
Encapsulate code and data in reusable actor objects as a common microservices design pattern

Observability
See and measure the message calls across components and networked services

Secrets
Securely access secrets from your application

https://docs.dapr.io/concepts/building-blocks-concept/

State Stores



Amazon Web Services (AWS)

Name	CRUD	Transactional	ETag	Actors	Status	Component version	Since
AWS DynamoDB	✓	✗	✗	✗	Alpha	v1	1.0

Google Cloud Platform (GCP)

Name	CRUD	Transactional	ETag	Actors	Status	Component version	Since
GCP Firestore	✓	✗	✗	✗	Alpha	v1	1.0

Microsoft Azure

Name	CRUD	Transactional	ETag	Actors	Status	Component version	Since
Azure Blob Storage	✓	✗	✓	✗	GA	v1	1.0
Azure CosmosDB	✓	✓	✓	✓	GA	v1	1.0
Azure SQL Server	✓	✓	✓	✓	Alpha	v1	1.0
Azure Table Storage	✓	✗	✓	✗	Alpha	v1	1.0

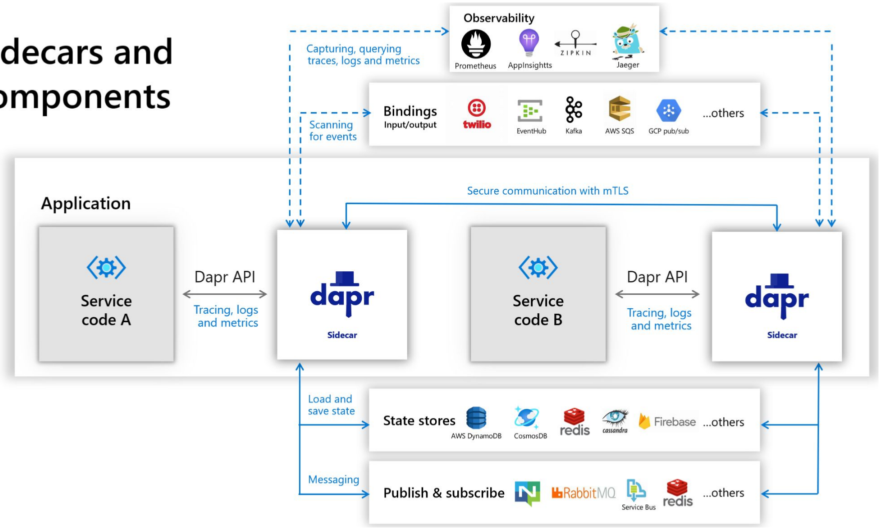
https://docs.dapr.io/reference/components-reference/

(*) Dapr is not a service mesh

Implementation

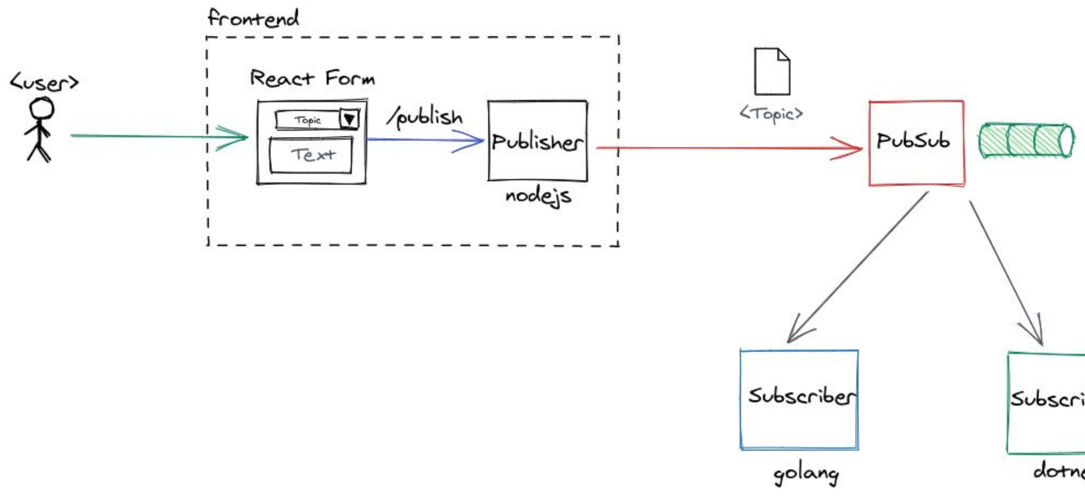
- Development **in the open** https://github.com/dapr/dapr / [Community Calls - YouTube]()
- [v1.2.0]() current, [MIT License](), **golang based** (❤)
- Dapr logic/building-blocks **implemented as sidecars**
- Expose **HTTP/gRPC API** for invocation ([dapr API]() / [dapr GRPC]())
- Support for **microservice patterns**

Sidecars and components



DEMO

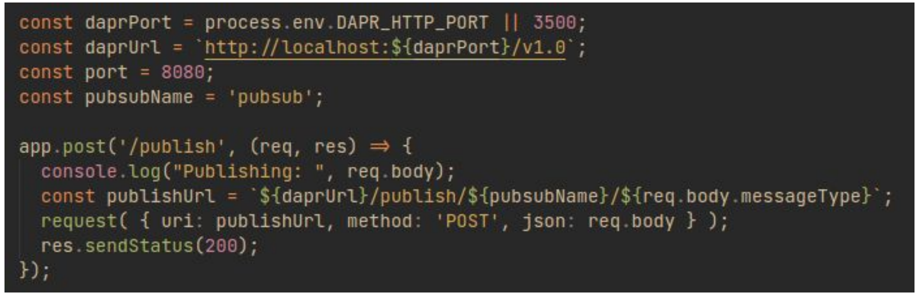
- Dev example with simple **pubsub logic**
- dapr CLI
- k8s with dapr (use redis)
- k8s with dapr (use azure service-bus)



DEMO

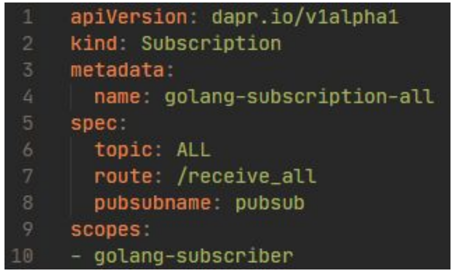
Dev example - PupSub

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

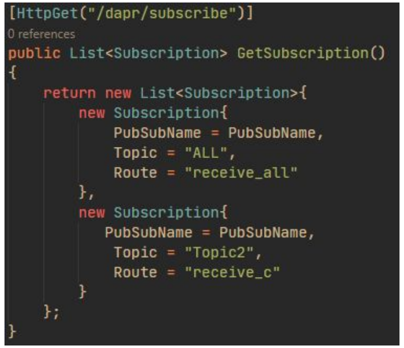


```
const daprPort = process.env.DAPR_HTTP_PORT || 3500;
const daprUrl = `http://localhost:${daprPort}/v1.0`;
const port = 8080;
const pubsubName = 'pubsub';

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);
});
```

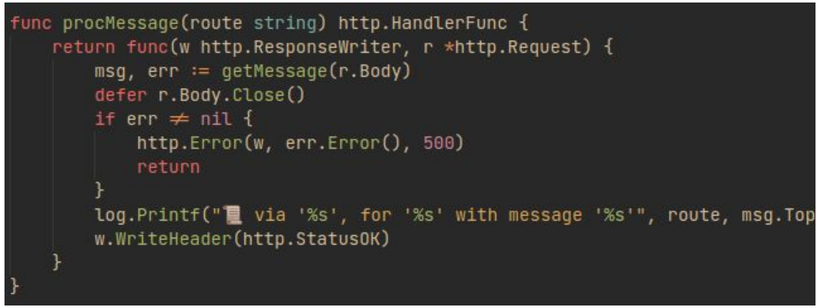
- **subscribers** -> register/consume either by **/dapr/subscribe** or **yaml**



```yaml
apiVersion: dapr.io/v1alpha1
kind: Subscription
metadata:
  name: golang-subscription-all
spec:
  topic: ALL
  route: /receive_all
  pubsubname: pubsub
scopes:
- golang-subscriber
```



```
[HttpGet("/dapr/subscribe")]
0 references
public List<Subscription> GetSubscription()
{
    return new List<Subscription>{
        new Subscription{
            PubSubName = PubSubName,
            Topic = "ALL",
            Route = "receive_all"
        },
        new Subscription{
           PubSubName = PubSubName,
            Topic = "Topic2",
            Route = "receive_c"
        }
    };
}
```



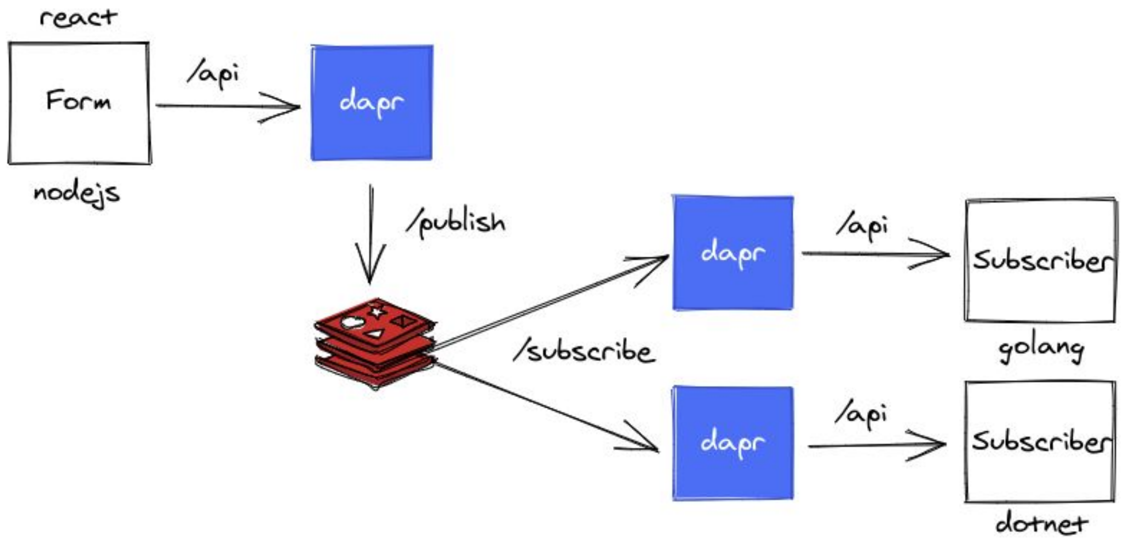
```
func procMessage(route string) http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        msg, err := getMessage(r.Body)
        defer r.Body.Close()
        if err ≠ nil {
            http.Error(w, err.Error(), 500)
            return
        }
        log.Printf("📜 via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

src: https://github.com/bihe/dapr-intro

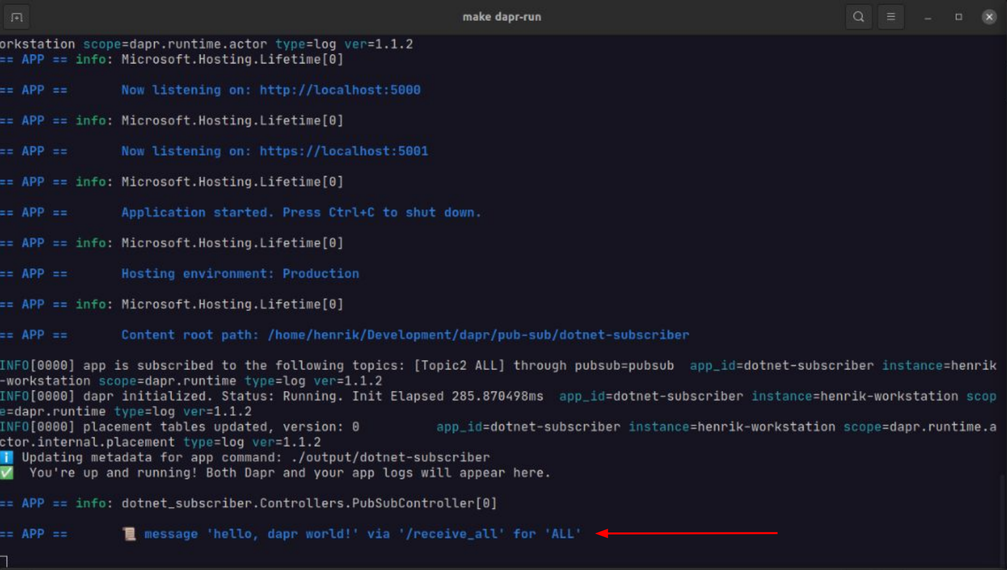
DEMO/cli

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

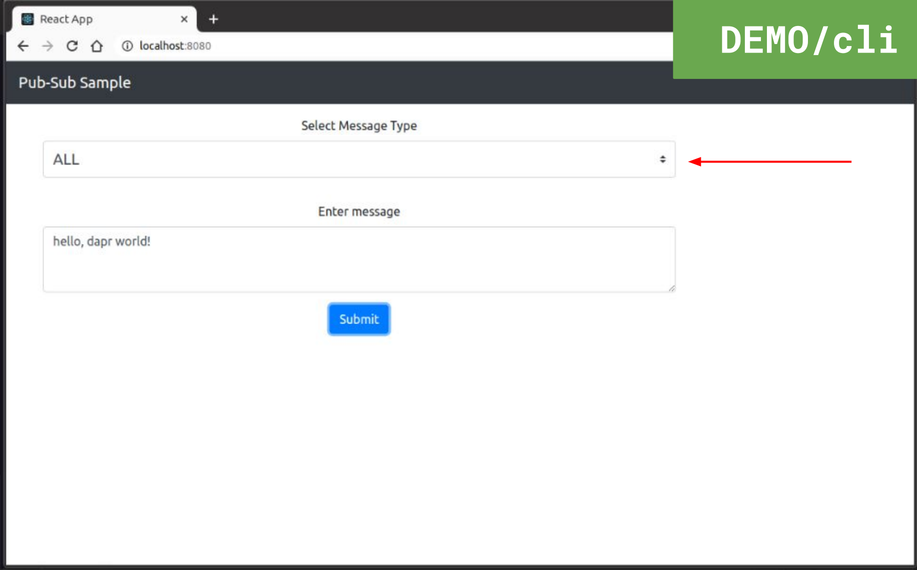


DEMO/cli



make dapr-run

```
orkstation 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 scop
e=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.a
ctor.internal.placement type=log ver=1.1.2
ℹ️ 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'
```



React App

localhost:8080

Pub-Sub Sample

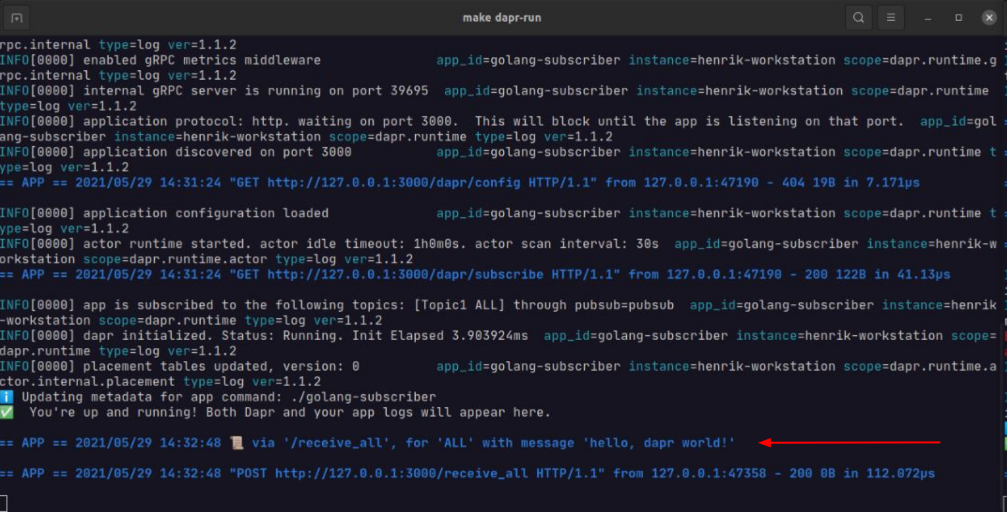
Select Message Type

ALL

Enter message

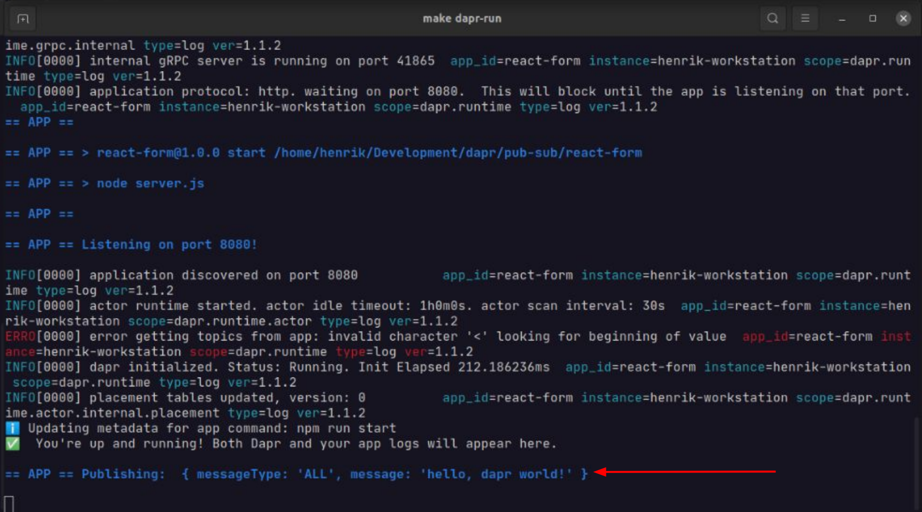
hello, dapr world!

Submit



make dapr-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.g
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=gol
ang-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 t
ype=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 19B in 7.171µs

INFO[0000] application configuration loaded              app_id=golang-subscriber instance=henrik-workstation scope=dapr.runtime t
ype=log ver=1.1.2
INFO[0000] actor runtime started. actor idle timeout: 1h0m0s. actor scan interval: 30s  app_id=golang-subscriber instance=henrik-w
orkstation 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 3.903924ms  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.a
ctor.internal.placement type=log ver=1.1.2
ℹ️ 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
```

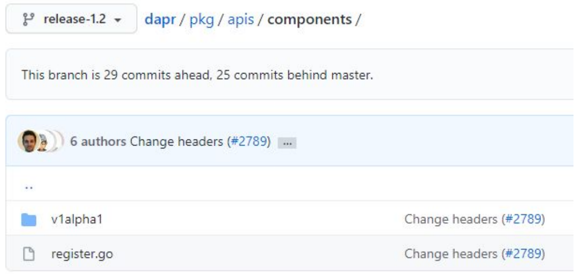


make dapr-run

```
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.run
time 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 ==

== APP == Listening on port 8080!

INFO[0000] application discovered on port 8080           app_id=react-form instance=henrik-workstation scope=dapr.runt
ime type=log ver=1.1.2
INFO[0000] actor runtime started. actor idle timeout: 1h0m0s. actor scan interval: 30s  app_id=react-form instance=hen
rik-workstation scope=dapr.runtime.actor type=log ver=1.1.2
ERRO[0000] error getting topics from app: invalid character '<' looking for beginning of value  app_id=react-form inst
ance=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.runt
ime.actor.internal.placement type=log ver=1.1.2
ℹ️ 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

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: pubsub
spec:
  type: pubsub.redis
  version: v1
  metadata:
  - name: redisHost
    value: redis-master:6379
  - name: redisPassword
    secretKeyRef:
      name: redis
      key: redis-password
auth:
  secretStore: kubernetes
```



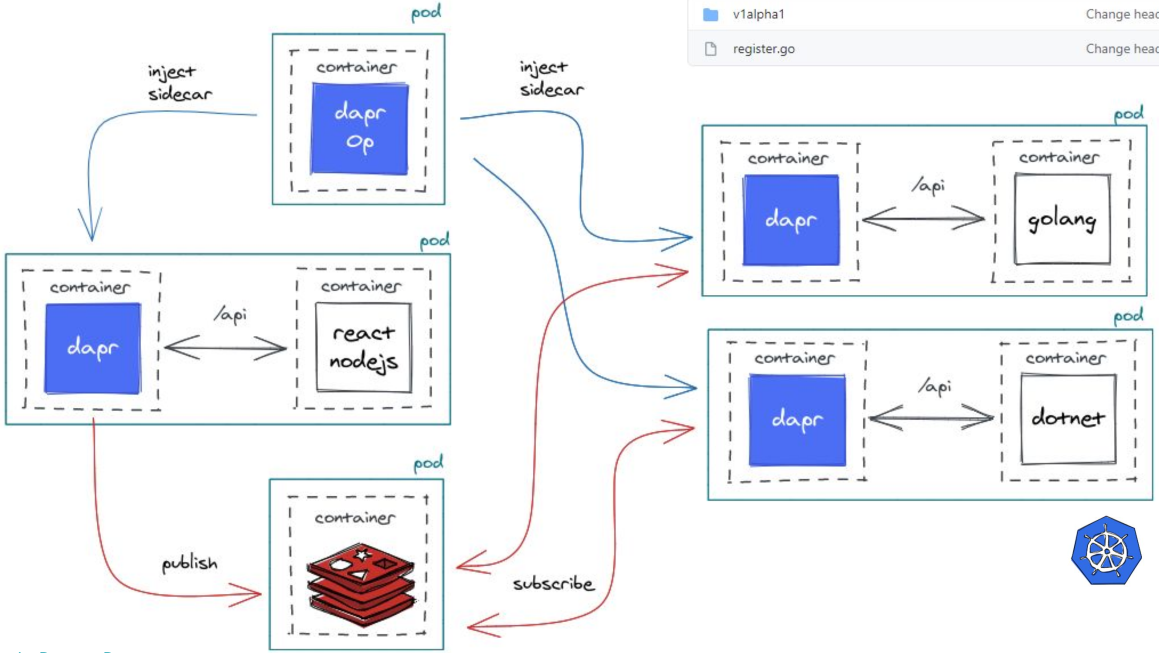
release-1.2 dapr / pkg / apis / components /

This branch is 29 commits ahead, 25 commits behind master.

6 authors Change headers (#2789) ...

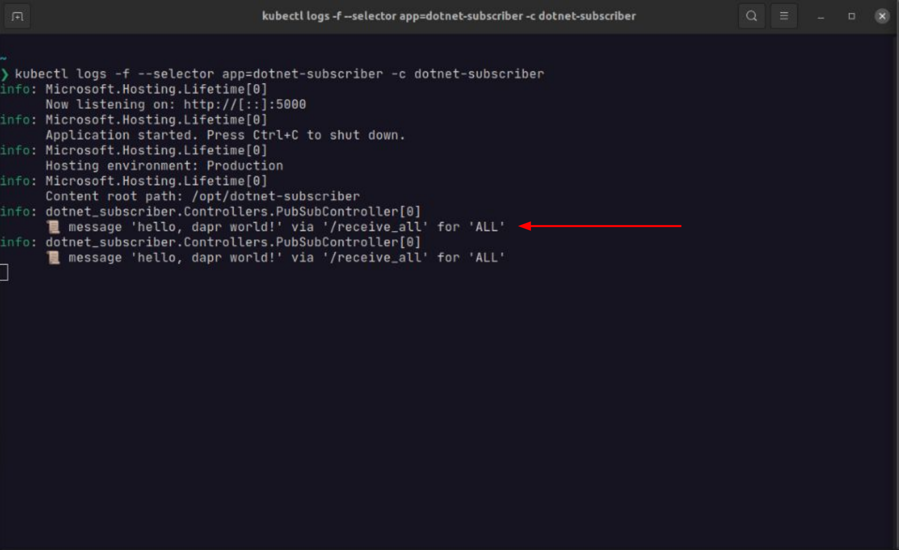
..

v1alpha1	Change headers (#2789)
register.go	Change headers (#2789)

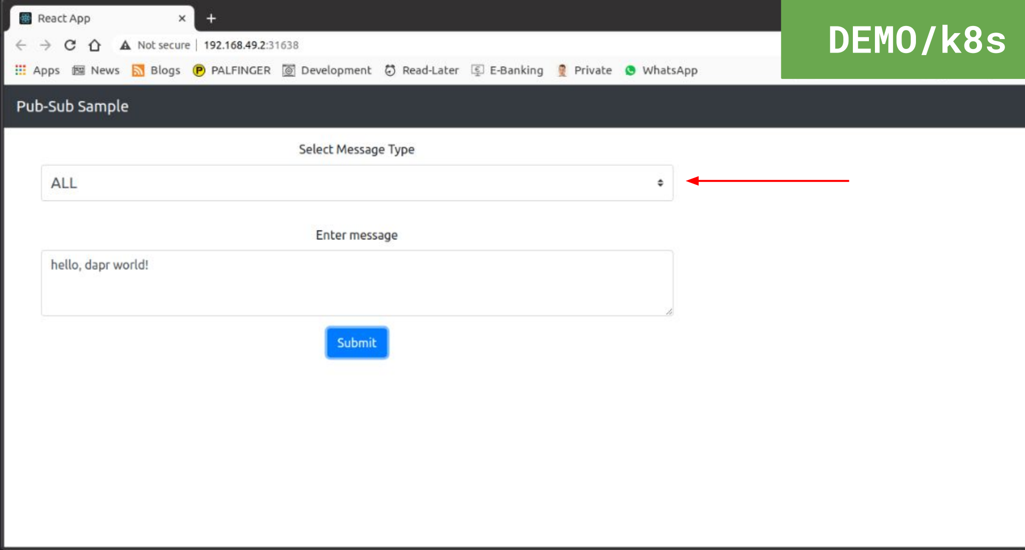


[Redis Streams | Dapr Docs](#) / [Component schema | Dapr Docs](#)

DEMO/k8s



```
~
❯ kubectl 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

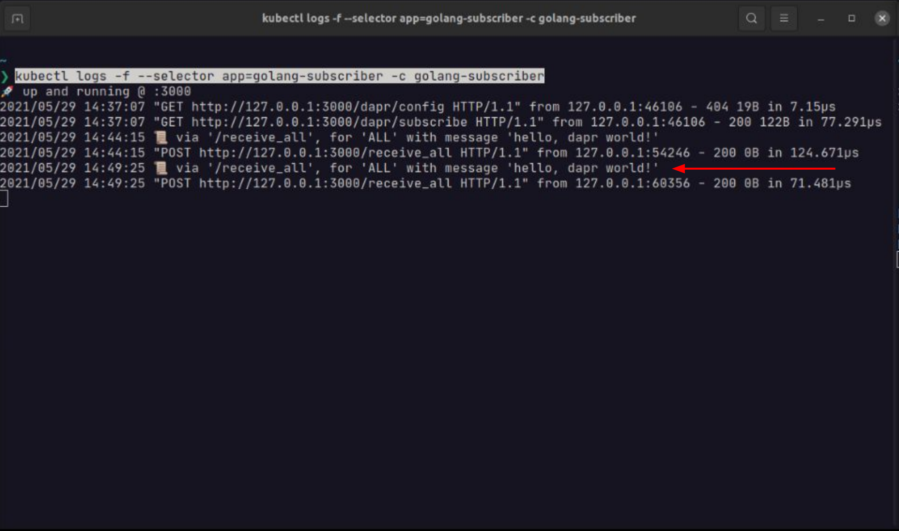
Select Message Type

ALL

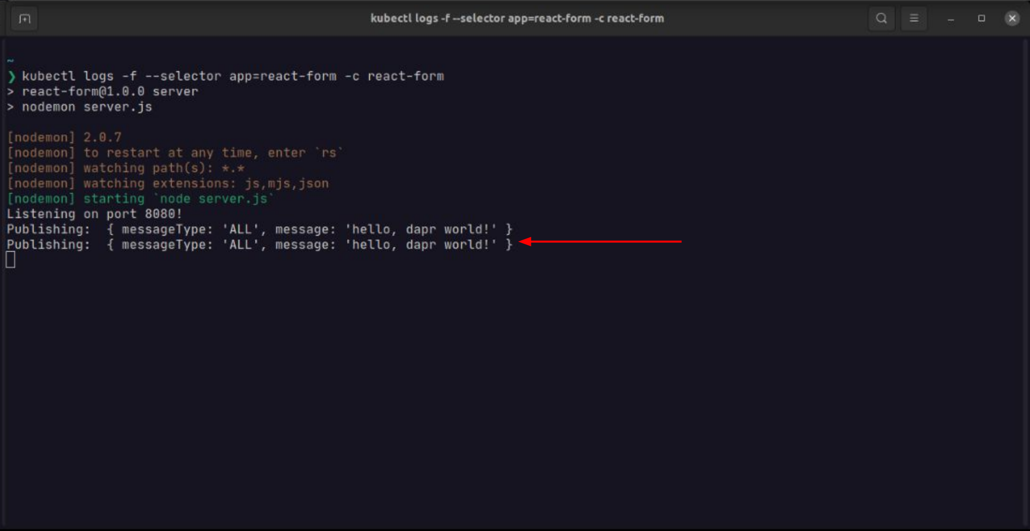
Enter message

hello, dapr world!

Submit



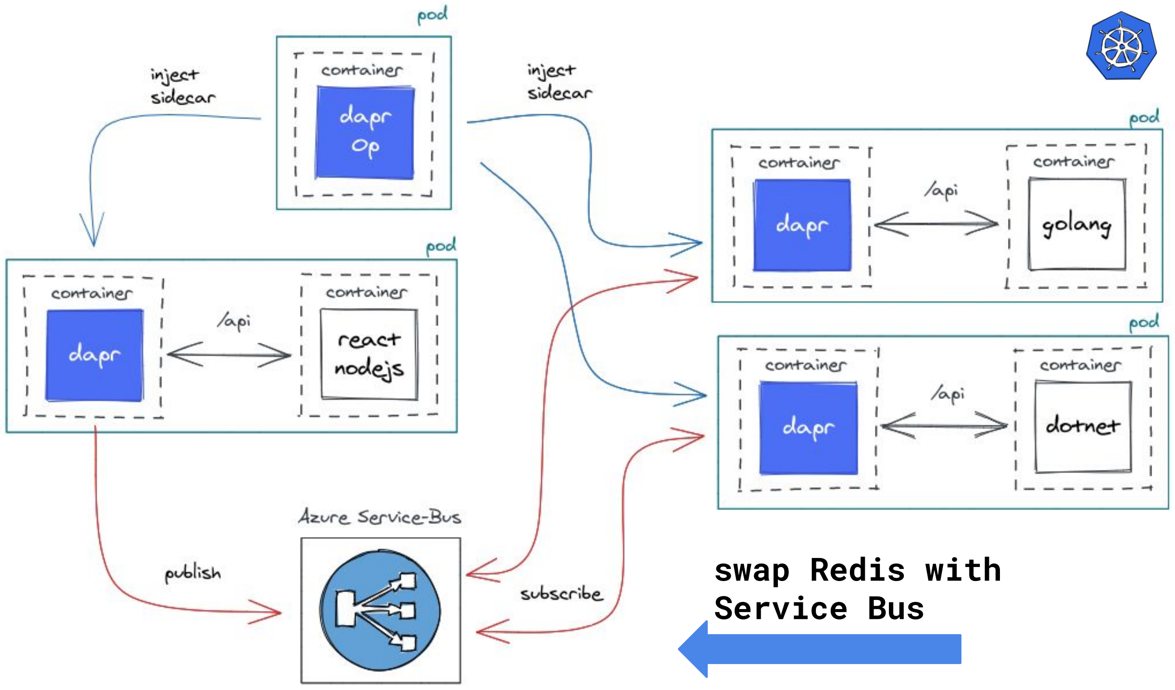
```
~
❯ kubectl logs -f --selector app=golang-subscriber -c golang-subscriber
🚀 up and running @ :3000
2021/05/29 14:37:07 "GET http://127.0.0.1:3000/dapr/config HTTP/1.1" from 127.0.0.1:46106 - 404 19B in 7.15µs
2021/05/29 14:37:07 "GET http://127.0.0.1:3000/dapr/subscribe HTTP/1.1" from 127.0.0.1:46106 - 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
```



```
~
❯ kubectl logs -f --selector app=react-form -c react-form
> react-form@1.0.0 server
> nodemon 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

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: pubsub
spec:
  type: pubsub.azure.servicebus
  version: v1
  metadata:
  - name: connectionString # Requir
    secretKeyRef:
      name: az-sb
      key: connstr
auth:
  secretStore: kubernetes
```



swap Redis with Service Bus

[Azure Service Bus | Dapr Docs](#) / [How-To: Publish a message and subscribe to a topic | Dapr Docs](#)

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=umrUlfrZqKk
- https://github.com/bihe/**dapr-intro**
- https://blog.dapr.io/posts/2021/03/02/**a-visual-guide-to-dapr**/