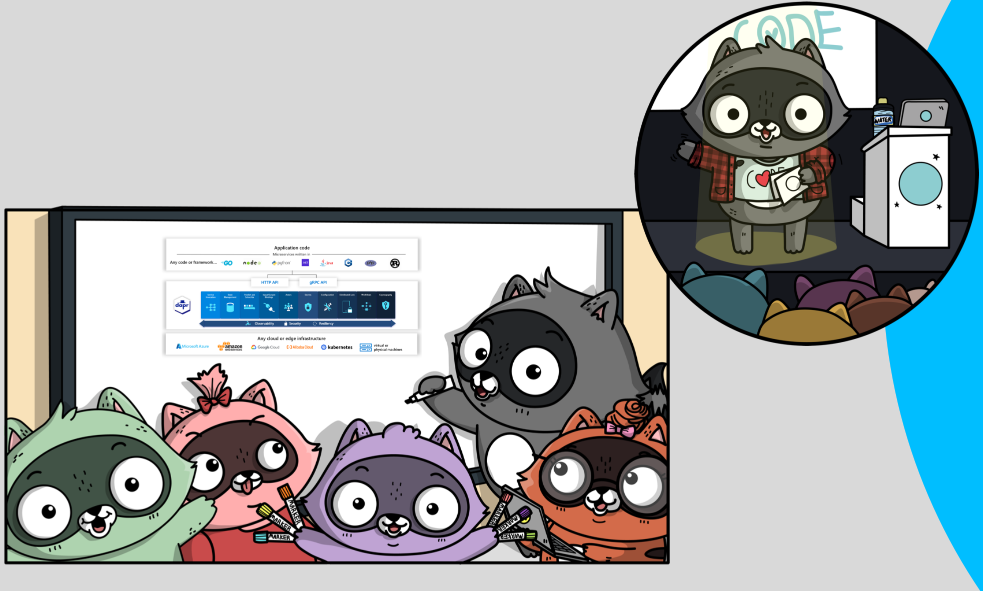
University of Stuttgart

Institute of Software Engineering (ISTE)
Software Quality and Architecture Group (SQA)

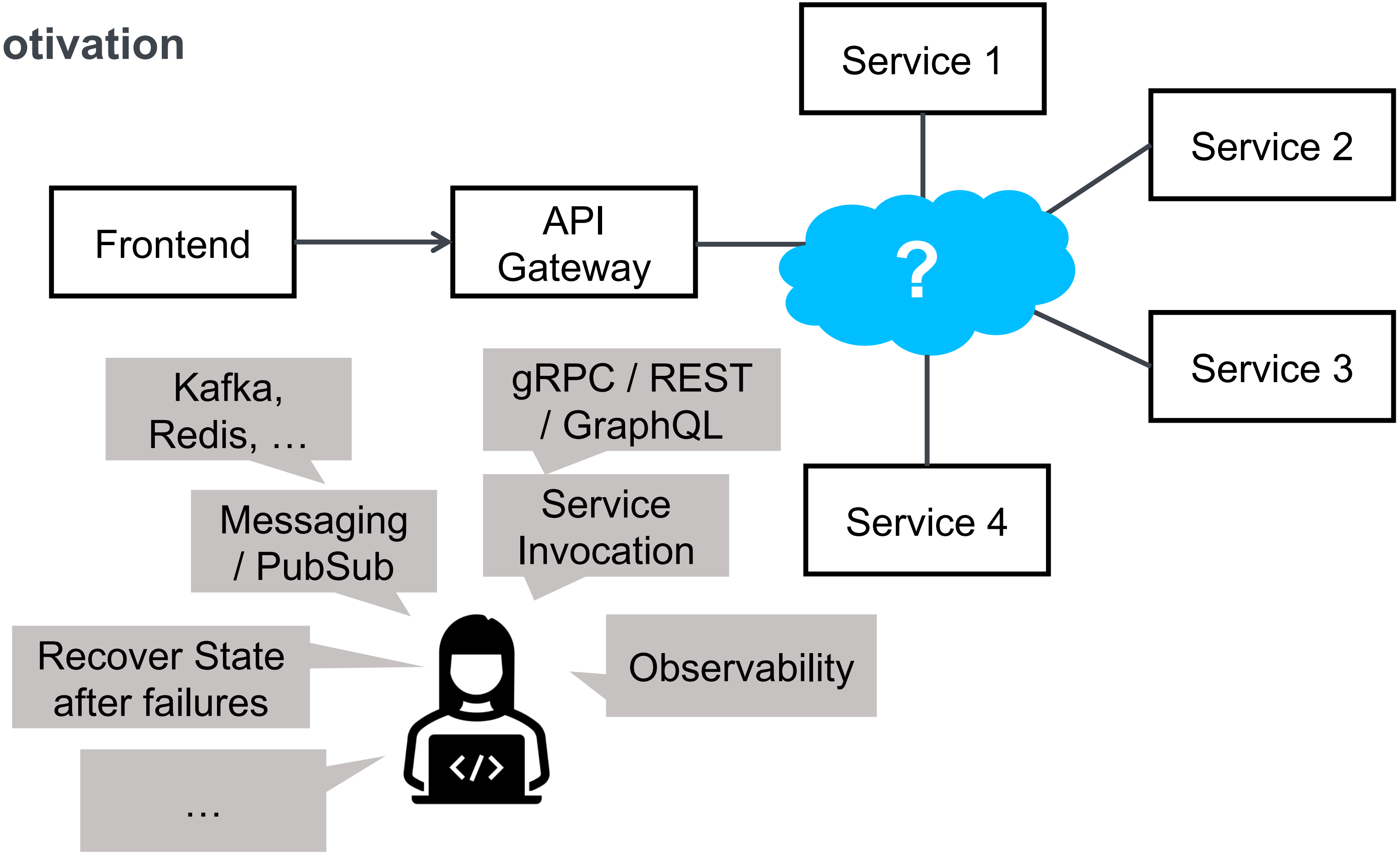


Developing Microservice Architectures with the Distributed Application Runtime (DAPR): Is DAPR Ready?

GI AK MSDO Treffen

© Sandro Speth

Motivation



Service 1

Service 2

Frontend

API Gateway

?

Service 3

Kafka, Redis, …

gRPC / REST / GraphQL

Service 4

Messaging / PubSub

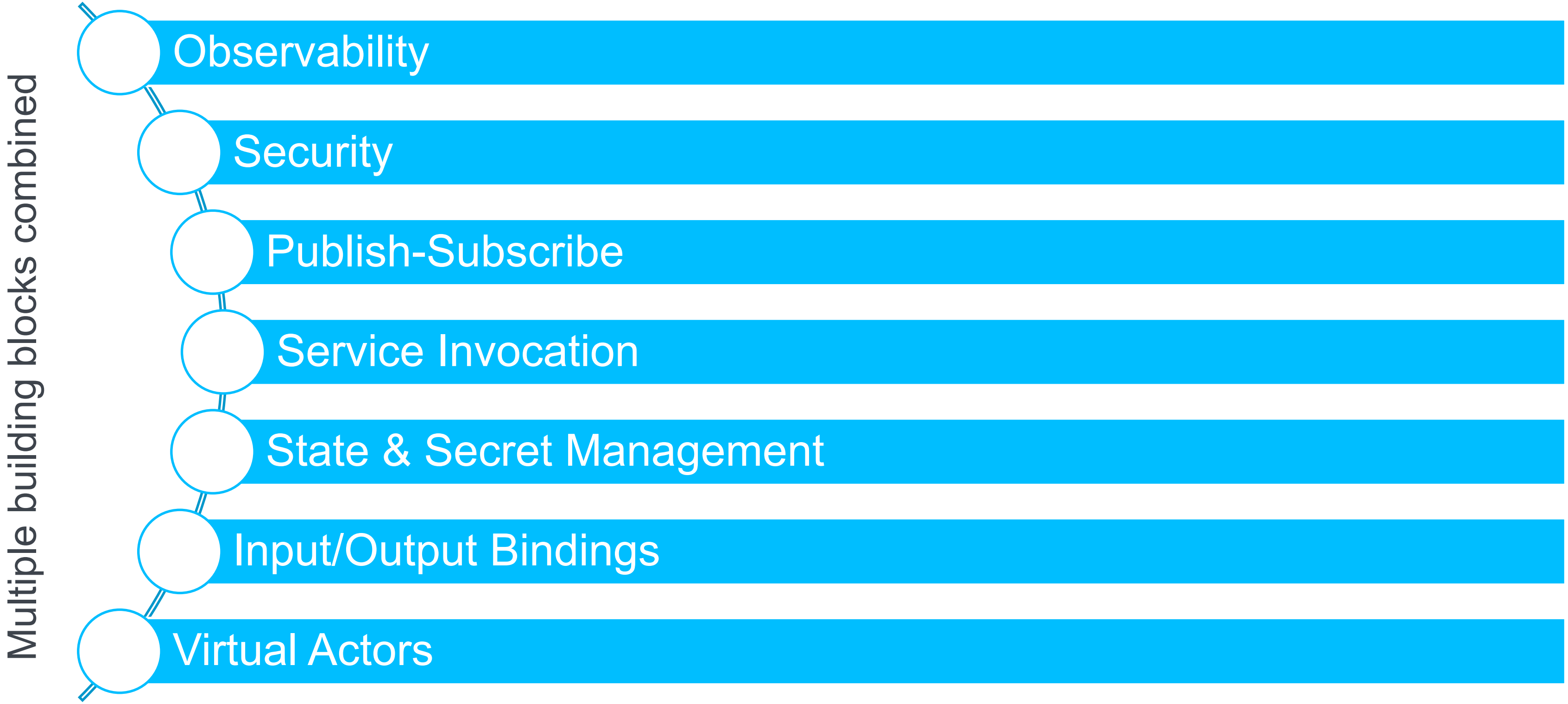
Service Invocation

Recover State after failures

Observability

…

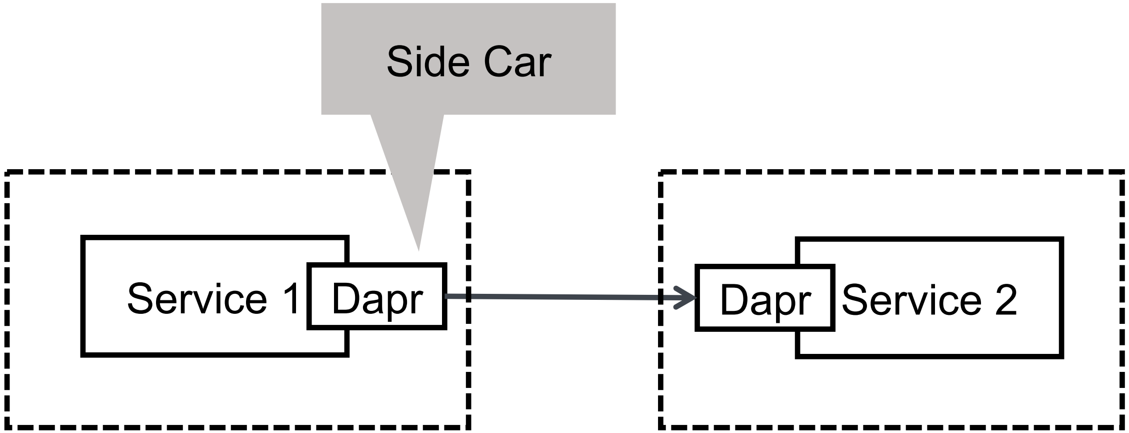
Dapr – Overview



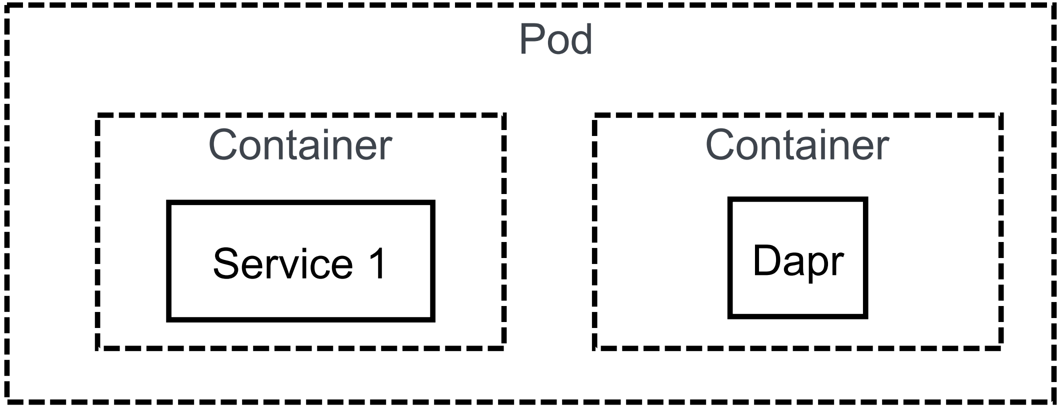
Multiple building blocks combined

- Observability
- Security
- Publish-Subscribe
- Service Invocation
- State & Secret Management
- Input/Output Bindings
- Virtual Actors

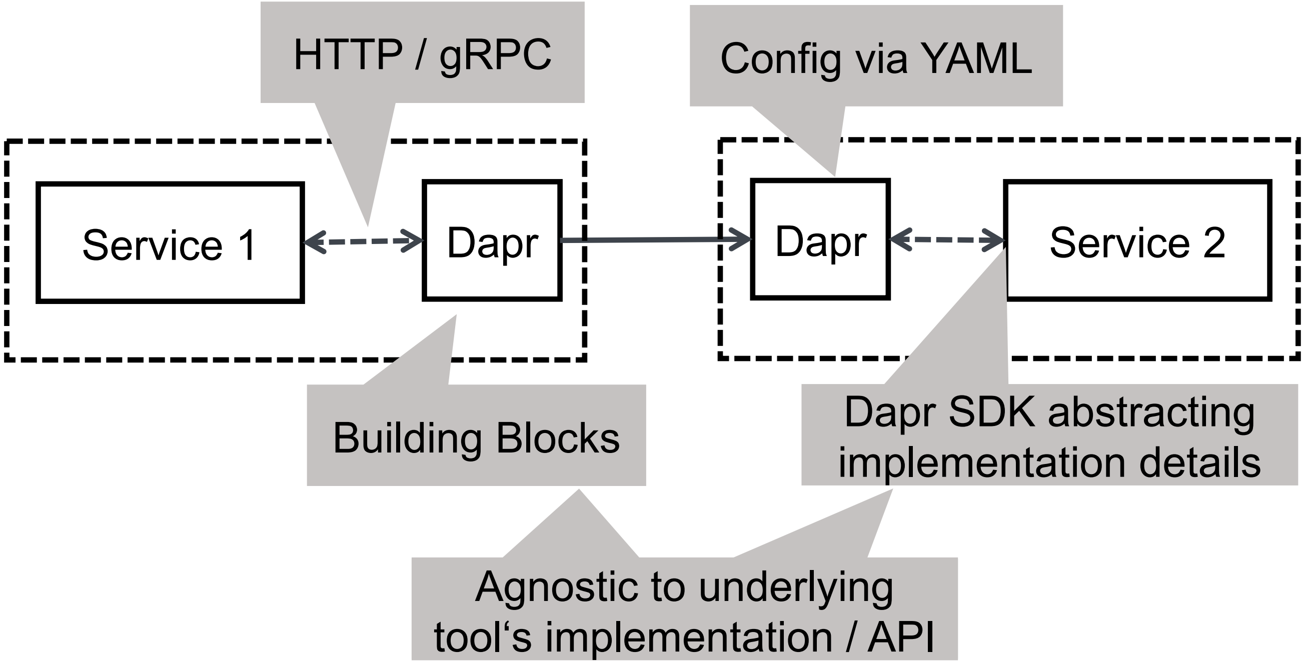
Dapr – Sidecar



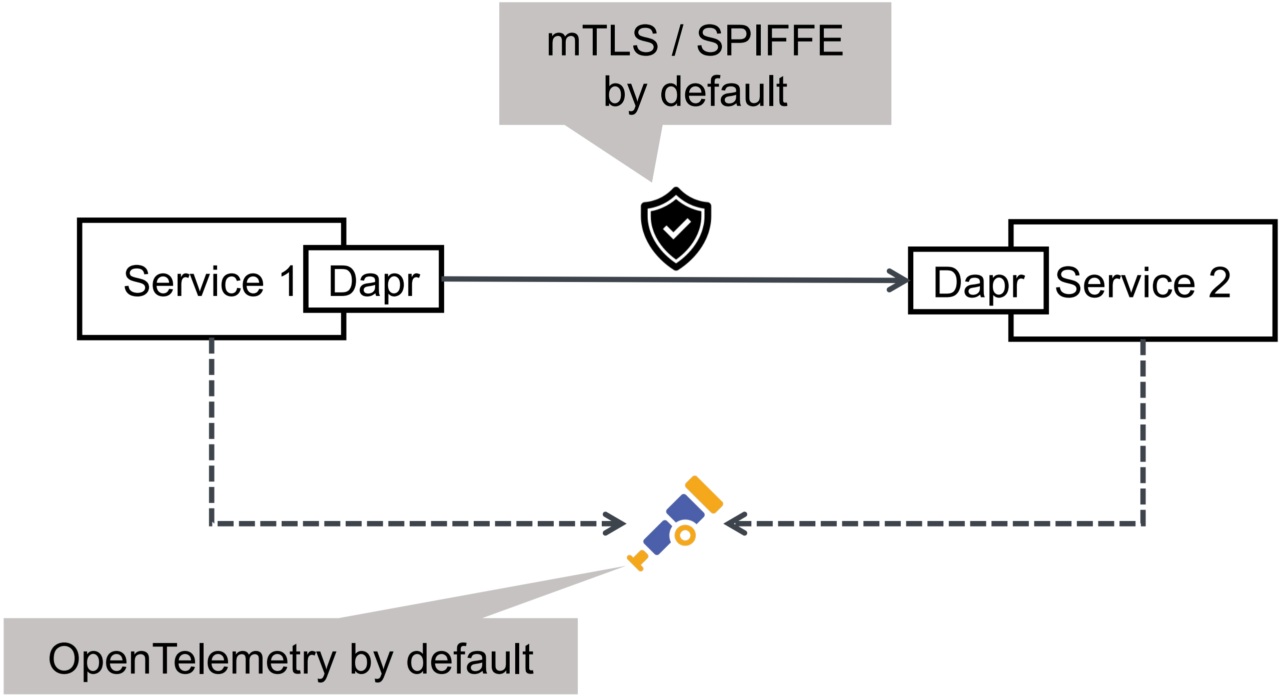
Dapr – Sidecar on Kubernetes



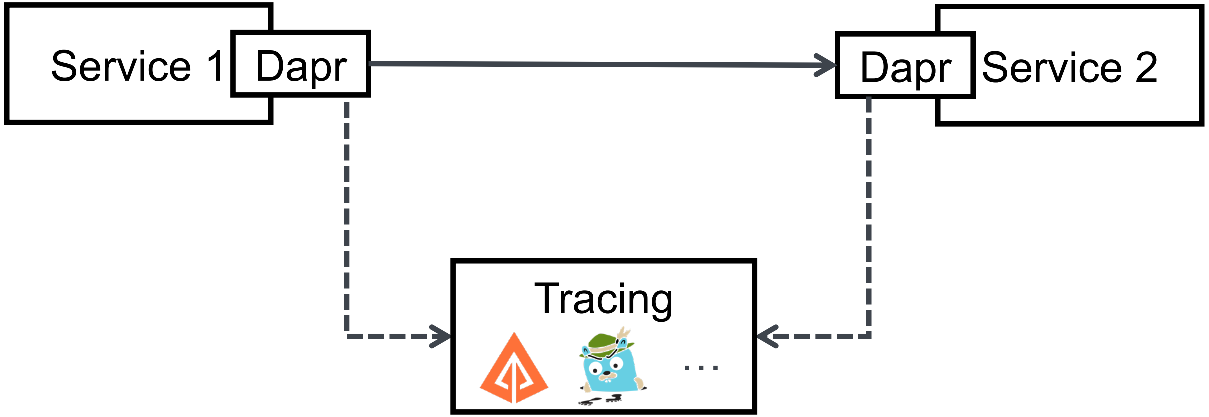
Dapr – Sidecar


HTTP / gRPC
Config via YAML
Service 1
Dapr
Dapr
Service 2
Building Blocks
Dapr SDK abstracting implementation details
Agnostic to underlying tool‘s implementation / API

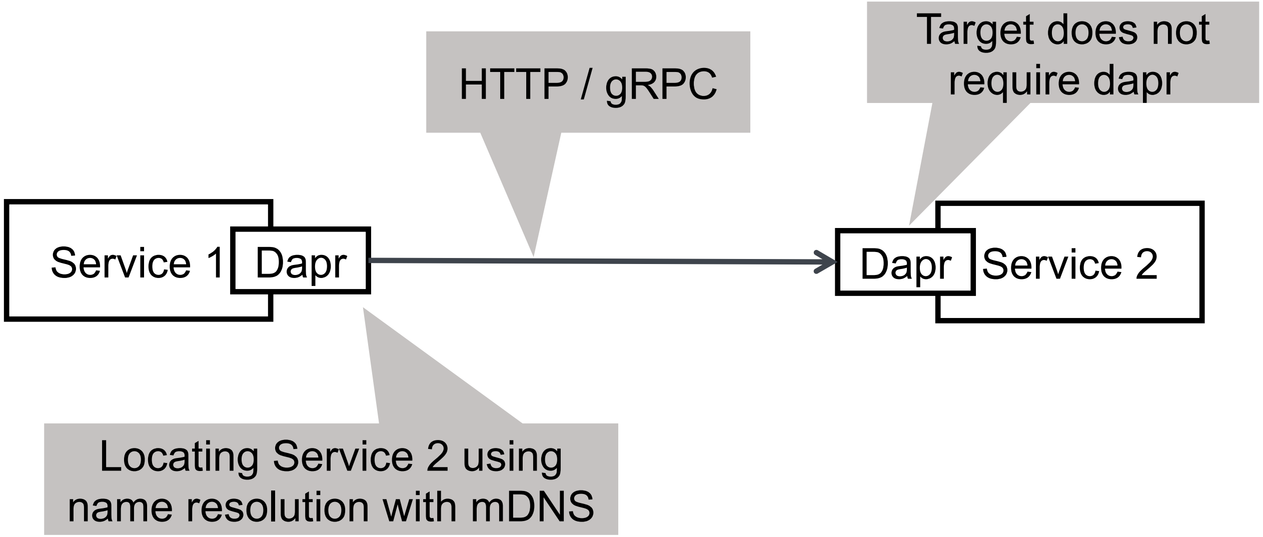
Dapr – Observability & Security



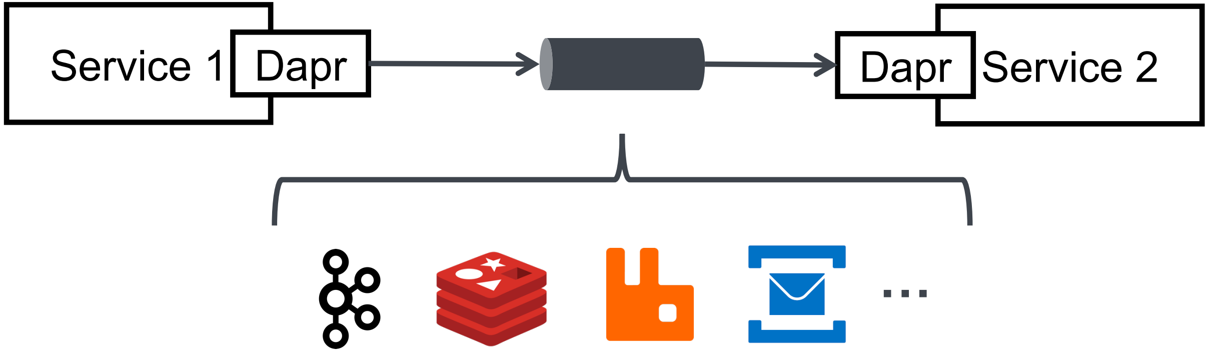
Dapr – Observability & Security



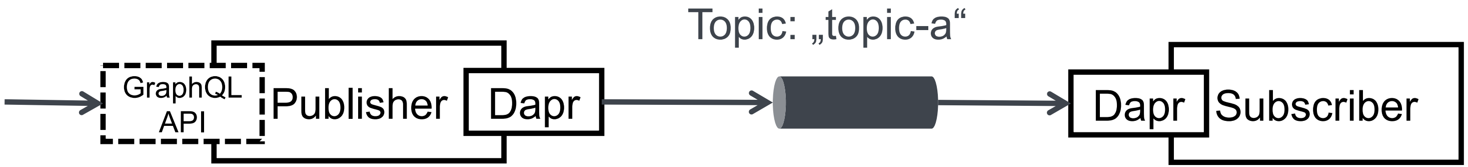
Dapr – Service Invocation


Target does not require dapr
HTTP / gRPC
Service 1
Dapr
Dapr
Service 2
Locating Service 2 using name resolution with mDNS

Dapr – PubSub

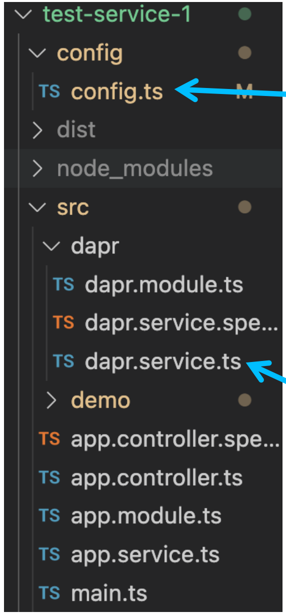


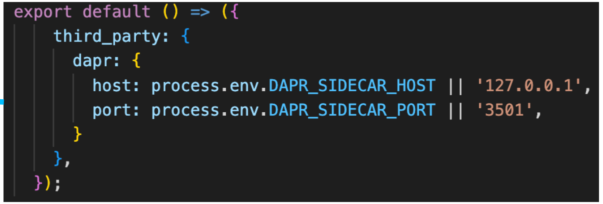
Our first „tech prototype“



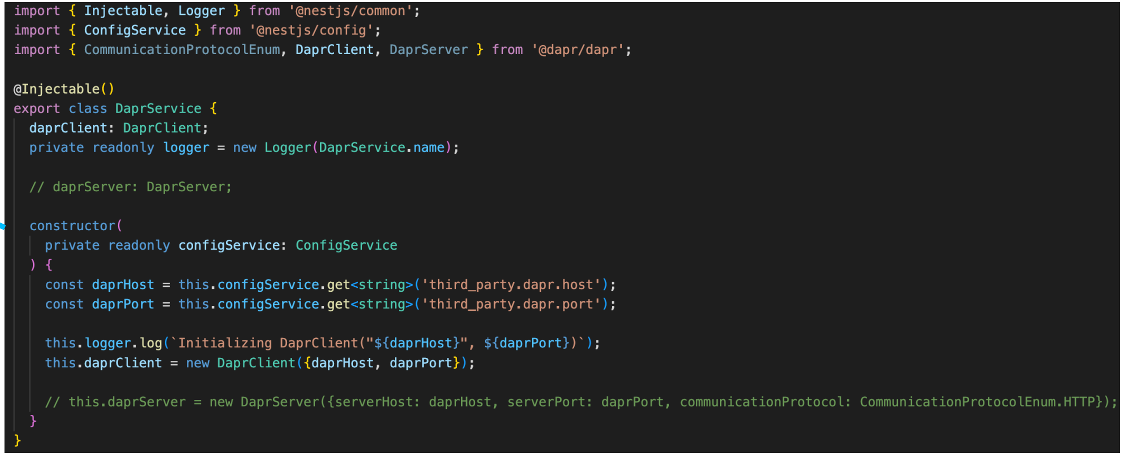
Simple example but required some hours of work back then…

Our first „tech prototype“



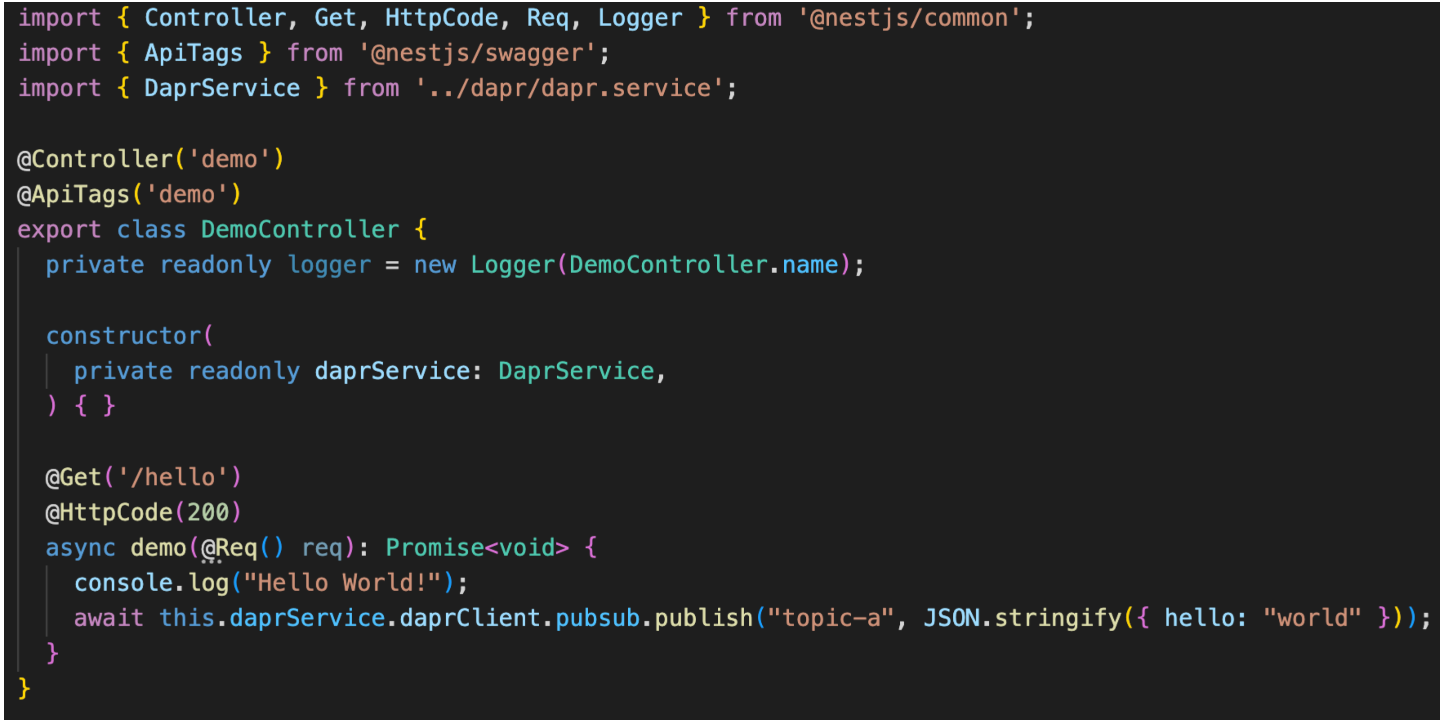


```
export default () => ({
    third_party: {
      dapr: {
        host: process.env.DAPR_SIDECAR_HOST || '127.0.0.1',
        port: process.env.DAPR_SIDECAR_PORT || '3501',
      }
    },
  });
```



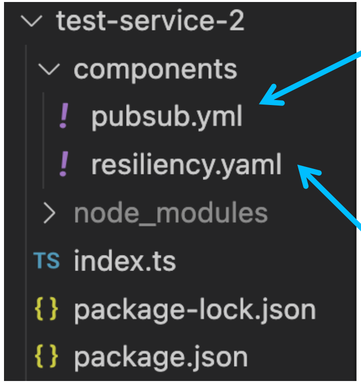
```
import { Injectable, Logger } from '@nestjs/common';
import { ConfigService } from '@nestjs/config';
import { CommunicationProtocolEnum, DaprClient, DaprServer } from '@dapr/dapr';

@Injectable()
export class DaprService {
  daprClient: DaprClient;
  private readonly logger = new Logger(DaprService.name);

  // daprServer: DaprServer;

  constructor(
    private readonly configService: ConfigService
  ) {
    const daprHost = this.configService.get<string>('third_party.dapr.host');
    const daprPort = this.configService.get<string>('third_party.dapr.port');

    this.logger.log(`Initializing DaprClient("${daprHost}", ${daprPort})`);
    this.daprClient = new DaprClient({daprHost, daprPort});

    // this.daprServer = new DaprServer({serverHost: daprHost, serverPort: daprPort, communicationProtocol: CommunicationProtocolEnum.HTTP});
  }
}
```

Our first „tech prototype“

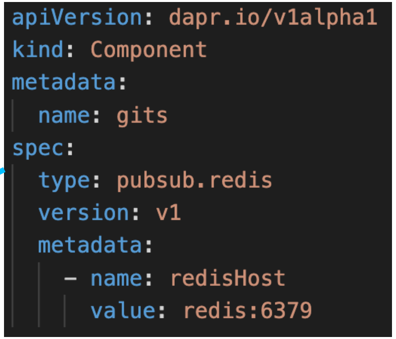


```
import { Controller, Get, HttpCode, Req, Logger } from '@nestjs/common';
import { ApiTags } from '@nestjs/swagger';
import { DaprService } from '../dapr/dapr.service';

@Controller('demo')
@ApiTags('demo')
export class DemoController {
  private readonly logger = new Logger(DemoController.name);

  constructor(
    private readonly daprService: DaprService,
  ) { }

  @Get('/hello')
  @HttpCode(200)
  async demo(@Req() req): Promise<void> {
    console.log("Hello World!");
    await this.daprService.daprClient.pubsub.publish("topic-a", JSON.stringify({ hello: "world" }));
  }
}
```

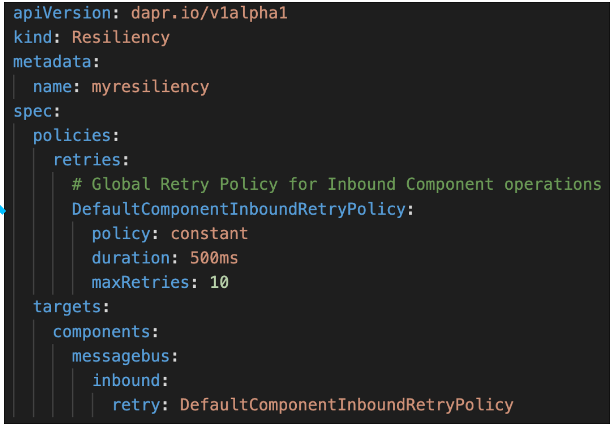
Our first „tech prototype“



```
test-service-2
  components
    pubsub.yml
    resiliency.yaml
  node_modules
  index.ts
  package-lock.json
  package.json
```

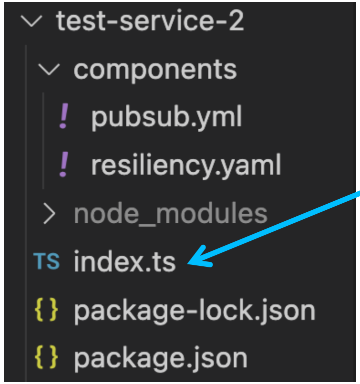


```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: gits
spec:
  type: pubsub.redis
  version: v1
  metadata:
    - name: redisHost
      value: redis:6379
```

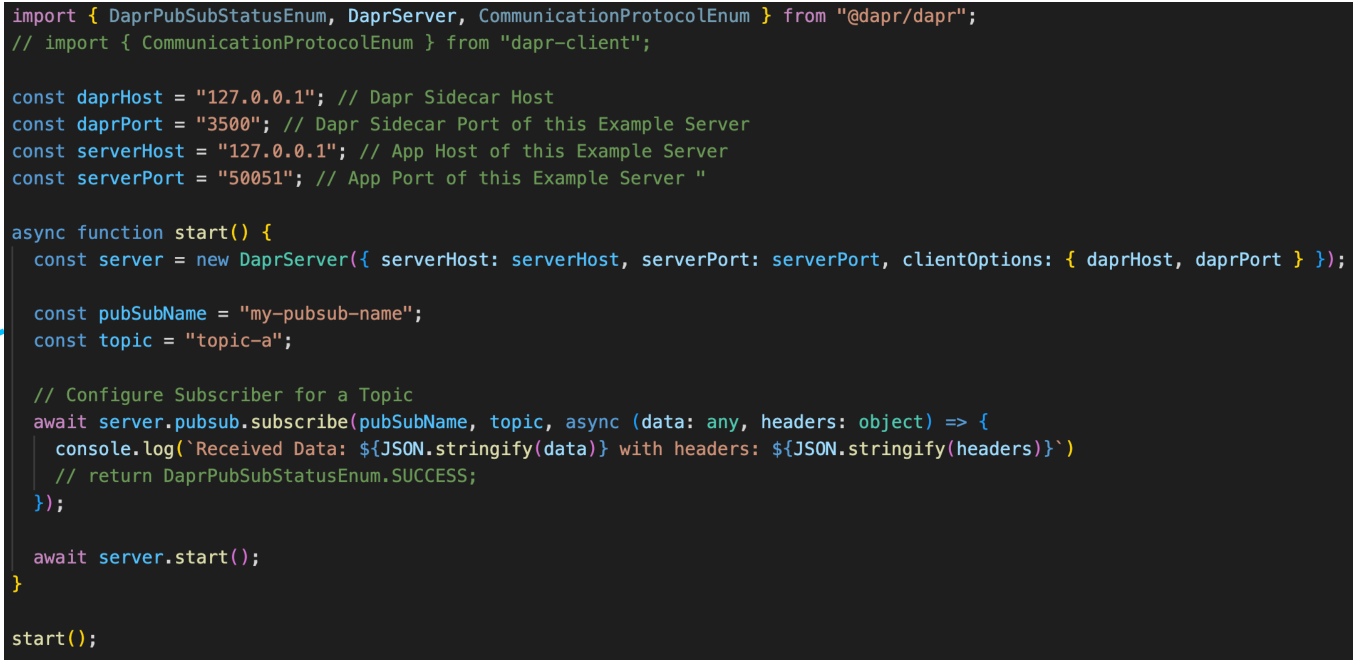


```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: myresiliency
spec:
  policies:
    retries:
      # Global Retry Policy for Inbound Component operations
      DefaultComponentInboundRetryPolicy:
        policy: constant
        duration: 500ms
        maxRetries: 10
  targets:
    components:
      messagebus:
        inbound:
          retry: DefaultComponentInboundRetryPolicy
```

Our first „tech prototype“



```
∨ test-service-2
  ∨ components
      ! pubsub.yml
      ! resiliency.yaml
  > node_modules
  TS index.ts
  {} package-lock.json
  {} package.json
```



```typescript
import { DaprPubSubStatusEnum, DaprServer, CommunicationProtocolEnum } from "@dapr/dapr";
// import { CommunicationProtocolEnum } from "dapr-client";

const daprHost = "127.0.0.1"; // Dapr Sidecar Host
const daprPort = "3500"; // Dapr Sidecar Port of this Example Server
const serverHost = "127.0.0.1"; // App Host of this Example Server
const serverPort = "50051"; // App Port of this Example Server "

async function start() {
  const server = new DaprServer({ serverHost: serverHost, serverPort: serverPort, clientOptions: { daprHost, daprPort } });

  const pubSubName = "my-pubsub-name";
  const topic = "topic-a";

  // Configure Subscriber for a Topic
  await server.pubsub.subscribe(pubSubName, topic, async (data: any, headers: object) => {
    console.log(`Received Data: ${JSON.stringify(data)} with headers: ${JSON.stringify(headers)}`)
    // return DaprPubSubStatusEnum.SUCCESS;
  });

  await server.start();
}

start();
```

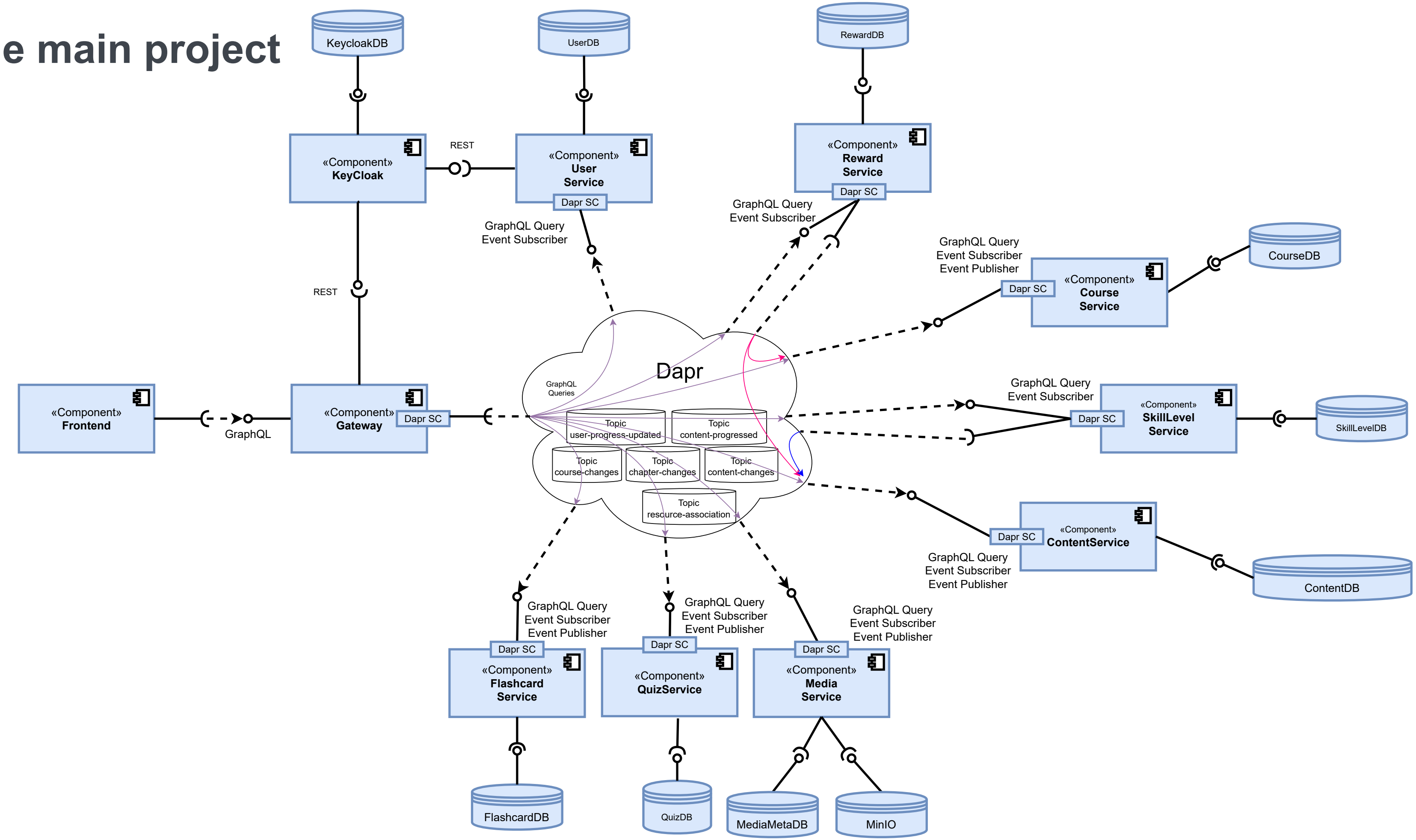
Running service 2: dapr run --app-id test-service-2 --dapr-http-port 3500 --app-protocol http --app-port 50051 -- npm start

Our first experience

- Running apps more difficult … but managable if you know the commands
- Docs are screwed up … but surely only in this case and only for JS/TS
  - Correct docs instead of docs.dapr.io./… under https://v1-11.docs.dapr.io/developing-applications/sdks/js/js-client/
- ChatGPT knows dapr 🥳 … but several old versions before breaking changes 🤡

Nevertheless, once you have the right docs and know what to do it‘s really nice!

The main project



Experience & Discussion

- Documentation
  - Simple example well documented
  - More special stuff lacks documentation
  - .Net well documented, others less
- Devs on Discord server and GitHub extremely fast and helpful
- Parts of the lib differ in „how to use“
- Configuring usually fairly simple
- Service invocation also usable with GraphQL

Is Dapr ready?

Yes!

… if you are Microsoft or work esp. on .Net

And otherwise?

Maybe wait 1-2 more years if you do not want a hard learning curve

University of Stuttgart
Institute of Software Engineering (ISTE)
Software Quality and Architecture Group (SQA)

Thank you!

Sandro Speth

e-mail sandro.speth@iste.uni-stuttgart.de
phone +49 (0) 711 685-61693
www. iste.uni-stuttgart.de/sqa/team/Speth

University of Stuttgart
Institute of Software Engineering,
Software Quality and Architecture Group
Universitätsstraße 38,
70569 Stuttgart
Room 1.336

X @spethso
in /in/sandro-Speth
▶ @SandroSpeth