Cloud Native Summit 2023

Dependency Injection for Serverless Applications

Thor Chen
Principal Software Engineer @ Objective Corporation

Outline

1. **What is Dependency Injection?**
2. **Why Serverless applications need it?**
3. **How to use it?**

What is Dependency Injection?

Dependency Injection (DI):

- Is a software design pattern
- Solves the problem of how code components obtain their dependencies

What is Dependency Injection?

Dependency Injection (DI):

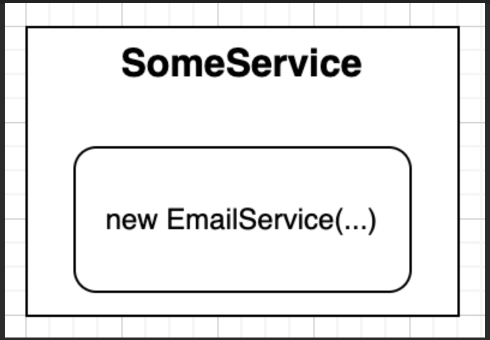
- Is a software design pattern
- Solves the problem of how code components obtain their dependencies

In this talk:

- Examples are written in **TypeScript**
- We use a library **Awilix**
- Terms are based on **AWS**

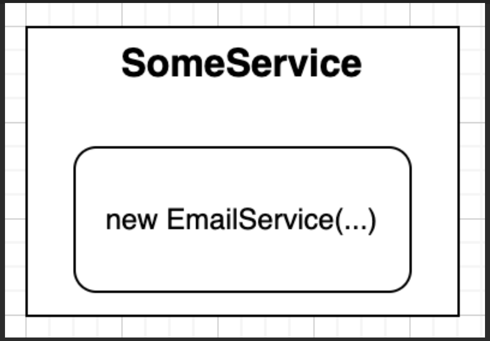
What is Dependency Injection?

```
class SomeService {

  public async doSomething() {
    // ...
    const emailService = new EmailService();
    await emailService.sendMail();
  }

}
```



What is Dependency Injection?

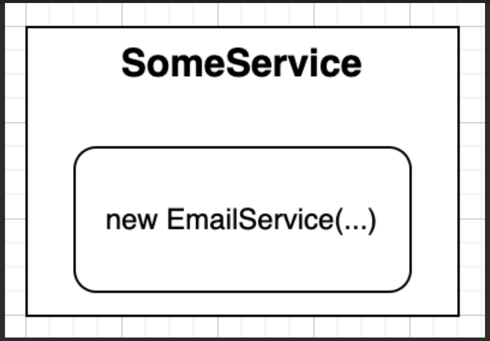
```
class SomeService {

  public async doSomething() {
    // ...
    const emailService = new EmailService();
    await emailService.sendMail();
  }

}
```



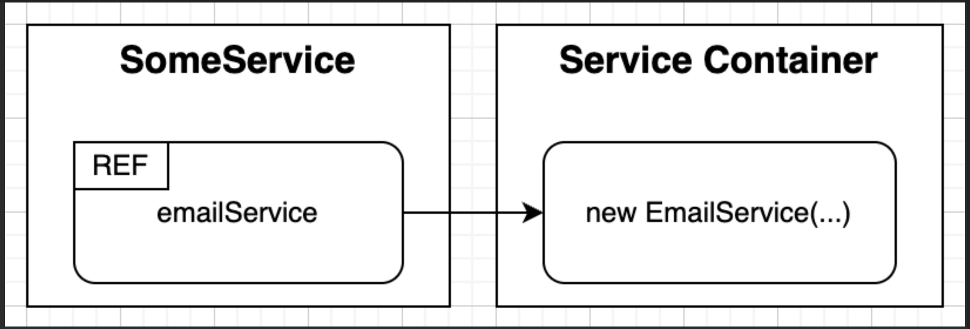
```
class SomeService {

  private readonly emailService: EmailService;

  constructor(dependencies: { emailService: EmailService }) {
    this.emailService = dependencies.emailService;
  }

  public async doSomething() {
    // ...
    await this.emailService.sendMail();
  }

}
```

What is Dependency Injection?

```
class SomeService {

  public async doSomething() {
    // ...
    const emailService = new EmailService();
    await emailService.sendMail();
  }

}
```



```
class SomeService {

  private readonly emailService: EmailService;

  constructor(dependencies: { emailService: EmailService }) {
    this.emailService = dependencies.emailService;
  }

  public async doSomething() {
    // ...
    await this.emailService.sendMail();
  }

}
```



Benefits of Dependency Injection?

Benefits of Dependency Injection?

1. **It makes the dependency very explicit**

```
class SomeService {
  public async doSomething() {
    const emailService = new EmailService();
  }
}
```

🤔 Dependency is not clear until read implementation details

Benefits of Dependency Injection?

1. **It makes the dependency very explicit**

```
class SomeService {
  public async doSomething() {
    const emailService = new EmailService();
  }
}
```

🤔 Dependency is not clear until read implementation details

```
class SomeService {
  constructor(dependencies: { emailService: EmailService }) {
    this.emailService = dependencies.emailService;
  }
}
```

👍 Dependency is clear when defined in the constructor

Benefits of Dependency Injection?

1. **It makes the dependency very explicit**

2. **It encourages separation of concerns**

```
class SomeService {
  public async doSomething() {
    const emailService = new EmailService();
  }
}
```

🤔 SomeService had to know how to instantiate EmailService

Benefits of Dependency Injection?

1. **It makes the dependency very explicit**

2. **It encourages separation of concerns**

```
class SomeService {
  public async doSomething() {
    const emailService = new EmailService();
  }
}
```

🤔 SomeService had to know how to instantiate EmailService

```
class SomeService {
  constructor(dependencies: { emailService: EmailService }) {
    this.emailService = dependencies.emailService;
  }
}
```

👍 SomeService don't need to know how to instantiate EmailService

Benefits of Dependency Injection?

1. **It makes the dependency very explicit**

2. **It encourages separation of concerns**

3. **It empowers writing code based on contracts**

```
interface EmailService { ... }

class EmailServiceCloud implements EmailService { ... }

class EmailServiceLocal implements EmailService { ... }

class EmailServiceMock implements EmailService { ... }
```

Benefits of Dependency Injection?

1. **It makes the dependency very explicit**

2. **It encourages separation of concerns**

3. **It empowers writing code based on contracts**

```
interface EmailService { ... }

// When the code is running on cloud, we send real emails
class EmailServiceCloud implements EmailService { ... }

// When the code is running locally, we don't send the email, but just log the email-sending action
class EmailServiceLocal implements EmailService { ... }

// In unit tests, we completely skip the email-sending process
class EmailServiceMock implements EmailService { ... }
```

Why do Serverless applications need Dependency Injection?

Why do Serverless applications need Dependency Injection?

Serverless is great:

- Cloud providers (e.g., AWS) **handle the server management entirely**
- Resources can be **scaled automatically as needed**
- Computation process is generally **event-driven**

👍

Why do Serverless applications need Dependency Injection?

However:

- It is **challenging** to set up a **local development environment**
- It is **tricky** to write **unit tests**

Why do Serverless applications need Dependency Injection?

However:

- It is **challenging** to set up a **local development environment**
- It is **tricky** to write **unit tests**
- **Naive coding practice** makes the code base **very hard to maintain**

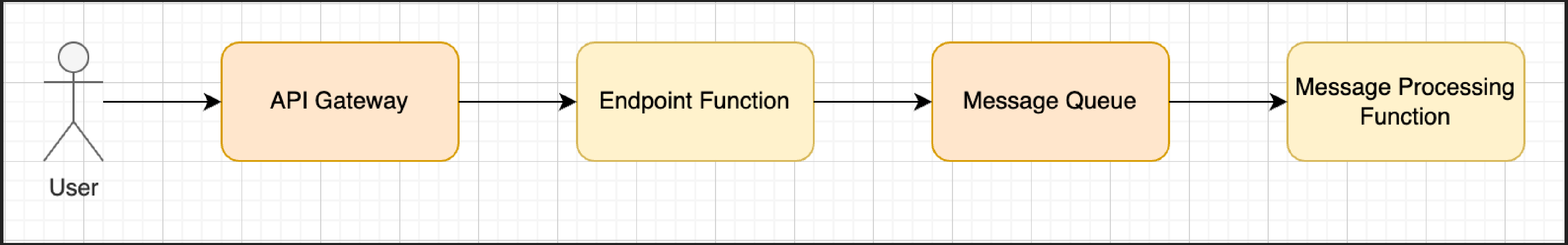
```
// The lines below are repeated everywhere in the code base

if (process.env.IS_LOCAL) {
  // ...
} else if (process.env.JEST_WORKER_ID) {
  // ...
} else {
  // ...
}
```

Why do Serverless applications need Dependency Injection?

Dependency Injection:

- Provides an easy way to **swap implementations** for individual pieces accordingly
- Helps structure the project code in a **modular** way

Example Setup



Example Setup

```typescript
// ConfigService.ts
export interface ConfigService {
  getMessageQueueUrl(): string;
}

// MessageQueueService.ts
export interface MessageQueueService {
  sendMessage(args: SendMessageArgs): Promise<string>;
}

// TodoService.ts
export interface TodoService {
  createTodo(content: string): Promise<string>;
}
```

Example Setup

```typescript
export type ServiceContainerCradle = {
  configService: ConfigService;
  messageQueueService: MessageQueueService;
  todoService: TodoService;
};

const serviceContainer = awilix.createContainer<ServiceContainerCradle>();

serviceContainer.register({ configService: awilix.asClass(ConfigServiceCloud) });
serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceCloud) });
serviceContainer.register({ todoService: awilix.asClass(TodoServiceImpl) });

const isLocal = Boolean(process.env.IS_LOCAL) || Boolean(process.env.IS_OFFLINE);
if (isLocal) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceLocal) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceLocal) });
}

const isTest = Boolean(process.env.JEST_WORKER_ID);
if (isTest) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceMock) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceMock) });
}
```

Example Setup

```
export type ServiceContainerCradle = {
  configService: ConfigService;
  messageQueueService: MessageQueueService;
  todoService: TodoService;
};

const serviceContainer = awilix.createContainer<ServiceContainerCradle>();

serviceContainer.register({ configService: awilix.asClass(ConfigServiceCloud) });
serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceCloud) });
serviceContainer.register({ todoService: awilix.asClass(TodoServiceImpl) });

const isLocal = Boolean(process.env.IS_LOCAL) || Boolean(process.env.IS_OFFLINE);
if (isLocal) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceLocal) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceLocal) });
}

const isTest = Boolean(process.env.JEST_WORKER_ID);
if (isTest) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceMock) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceMock) });
}
```

Example Setup

```
export type ServiceContainerCradle = {
  configService: ConfigService;
  messageQueueService: MessageQueueService;
  todoService: TodoService;
};

const serviceContainer = awilix.createContainer<ServiceContainerCradle>();

serviceContainer.register({ configService: awilix.asClass(ConfigServiceCloud) });
serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceCloud) });
serviceContainer.register({ todoService: awilix.asClass(TodoServiceImpl) });

const isLocal = Boolean(process.env.IS_LOCAL) || Boolean(process.env.IS_OFFLINE);
if (isLocal) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceLocal) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceLocal) });
}

const isTest = Boolean(process.env.JEST_WORKER_ID);
if (isTest) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceMock) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceMock) });
}
```

Example Setup

```
export type ServiceContainerCradle = {
  configService: ConfigService;
  messageQueueService: MessageQueueService;
  todoService: TodoService;
};

const serviceContainer = awilix.createContainer<ServiceContainerCradle>();

serviceContainer.register({ configService: awilix.asClass(ConfigServiceCloud) });
serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceCloud) });
serviceContainer.register({ todoService: awilix.asClass(TodoServiceImpl) });

const isLocal = Boolean(process.env.IS_LOCAL) || Boolean(process.env.IS_OFFLINE);
if (isLocal) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceLocal) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceLocal) });
}

const isTest = Boolean(process.env.JEST_WORKER_ID);
if (isTest) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceMock) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceMock) });
}
```

Example Setup

```
export type ServiceContainerCradle = {
  configService: ConfigService;
  messageQueueService: MessageQueueService;
  todoService: TodoService;
};

const serviceContainer = awilix.createContainer<ServiceContainerCradle>();

serviceContainer.register({ configService: awilix.asClass(ConfigServiceCloud) });
serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceCloud) });
serviceContainer.register({ todoService: awilix.asClass(TodoServiceImpl) });

const isLocal = Boolean(process.env.IS_LOCAL) || Boolean(process.env.IS_OFFLINE);
if (isLocal) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceLocal) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceLocal) });
}

const isTest = Boolean(process.env.JEST_WORKER_ID);
if (isTest) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceMock) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceMock) });
}
```

Example Setup

```
export type ServiceContainerCradle = {
  configService: ConfigService;
  messageQueueService: MessageQueueService;
  todoService: TodoService;
};

const serviceContainer = awilix.createContainer<ServiceContainerCradle>();

serviceContainer.register({ configService: awilix.asClass(ConfigServiceCloud) });
serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceCloud) });
serviceContainer.register({ todoService: awilix.asClass(TodoServiceImpl) });

const isLocal = Boolean(process.env.IS_LOCAL) || Boolean(process.env.IS_OFFLINE);
if (isLocal) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceLocal) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceLocal) });
}

const isTest = Boolean(process.env.JEST_WORKER_ID);
if (isTest) {
  serviceContainer.register({ configService: awilix.asClass(ConfigServiceMock) });
  serviceContainer.register({ messageQueueService: awilix.asClass(MessageQueueServiceMock) });
}
```

Example Setup

```
// TodoServiceImpl.ts

export class TodoServiceImpl implements TodoService {
  private readonly configService: ConfigService;
  private readonly messageQueueService: MessageQueueService;

  constructor(dependencies: Pick<ServiceContainerCradle, "configService" | "messageQueueService">) {
    this.configService = dependencies.configService;
    this.messageQueueService = dependencies.messageQueueService;
  }

  public async createTodo(content: string): Promise<string> {
    const id = crypto.randomUUID();

    // ...

    await this.messageQueueService.sendMessage({
      messageQueueUrl: this.configService.getMessageQueueUrl(),
      messageBody: JSON.stringify({ type: "TODO_CREATED", id }),
    });

    return id;
  }
}
```

Example Setup

```
// TodoServiceImpl.ts

export class TodoServiceImpl implements TodoService {
  private readonly configService: ConfigService;
  private readonly messageQueueService: MessageQueueService;

  constructor(dependencies: Pick<ServiceContainerCradle, "configService" | "messageQueueService">) {
    this.configService = dependencies.configService;
    this.messageQueueService = dependencies.messageQueueService;
  }

  public async createTodo(content: string): Promise<string> {
    const id = crypto.randomUUID();

    // ...

    await this.messageQueueService.sendMessage({
      messageQueueUrl: this.configService.getMessageQueueUrl(),
      messageBody: JSON.stringify({ type: "TODO_CREATED", id }),
    });

    return id;
  }
}
```

Example Setup

```
// TodoServiceImpl.ts

export class TodoServiceImpl implements TodoService {
  private readonly configService: ConfigService;
  private readonly messageQueueService: MessageQueueService;

  constructor(dependencies: Pick<ServiceContainerCradle, "configService" | "messageQueueService">) {
    this.configService = dependencies.configService;
    this.messageQueueService = dependencies.messageQueueService;
  }

  public async createTodo(content: string): Promise<string> {
    const id = crypto.randomUUID();

    // ...

    await this.messageQueueService.sendMessage({
      messageQueueUrl: this.configService.getMessageQueueUrl(),
      messageBody: JSON.stringify({ type: "TODO_CREATED", id }),
    });

    return id;
  }
}
```

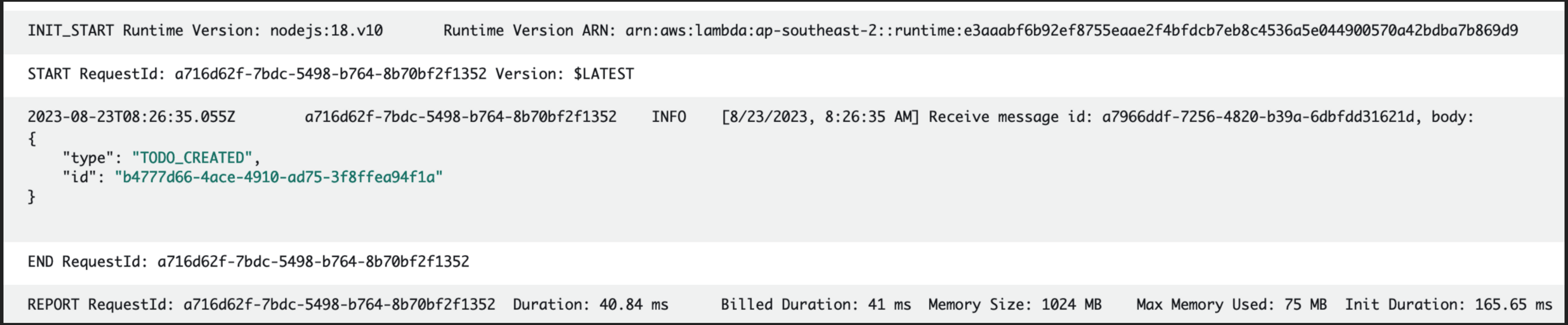
- on **cloud**: use *AWS SQS*
- on **local**: use *ElasticMQ* to simulate SQS
- in **tests**: log a message and *do nothing*

Example Setup

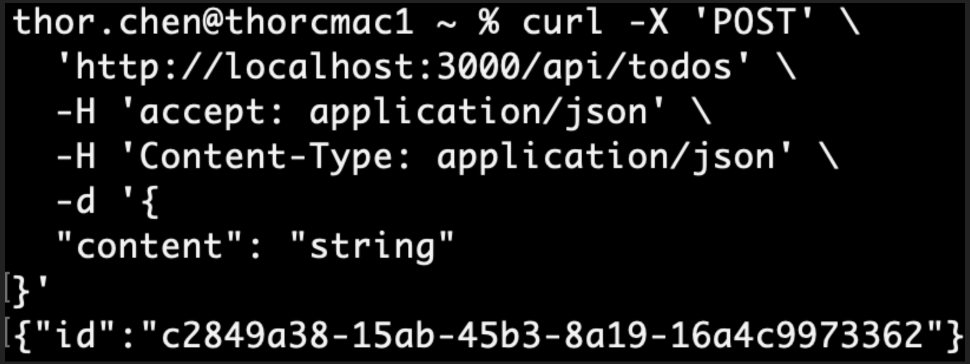
```
// TodoServiceImpl.test.ts

it("should create a new todo and put the message into queue", async () => {
  const configService = serviceContainer.cradle.configService;
  const messageQueueService = serviceContainer.cradle.messageQueueService;

  const todoService = new TodoServiceImpl({ configService, messageQueueService });

  const sendMessageSpy = jest.spyOn(messageQueueService, "sendMessage");

  await todoService.createTodo("test content");

  expect(sendMessageSpy).toHaveBeenCalledWith({
    messageQueueUrl: configService.getMessageQueueUrl(),
    messageBody: JSON.stringify({
      type: "TODO_CREATED",
      id: "test-uuid",
    }),
  });
});
```

Example Setup

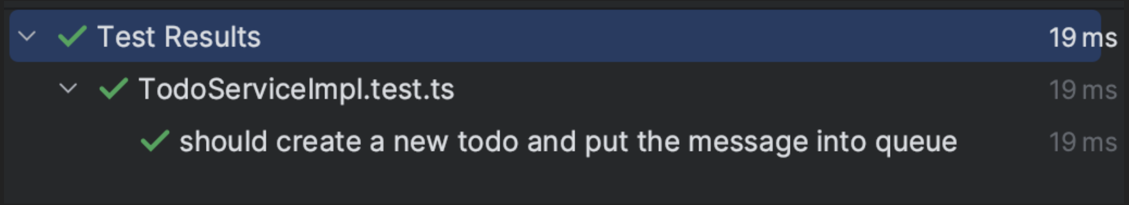
https://github.com/zzdjk6/serverless-dependency-injection-example



```
INIT_START Runtime Version: nodejs:18.v10       Runtime Version ARN: arn:aws:lambda:ap-southeast-2::runtime:e3aaabf6b92ef8755eaae2f4bfdcb7eb8c4536a5e044900570a42bdba7b869d9

START RequestId: a716d62f-7bdc-5498-b764-8b70bf2f1352 Version: $LATEST

2023-08-23T08:26:35.055Z        a716d62f-7bdc-5498-b764-8b70bf2f1352    INFO    [8/23/2023, 8:26:35 AM] Receive message id: a7966ddf-7256-4820-b39a-6dbfdd31621d, body:
{
    "type": "TODO_CREATED",
    "id": "b4777d66-4ace-4910-ad75-3f8ffea94f1a"
}

END RequestId: a716d62f-7bdc-5498-b764-8b70bf2f1352

REPORT RequestId: a716d62f-7bdc-5498-b764-8b70bf2f1352  Duration: 40.84 ms      Billed Duration: 41 ms  Memory Size: 1024 MB    Max Memory Used: 75 MB  Init Duration: 165.65 ms
```



```
thor.chen@thorcmac1 ~ % curl -X 'POST' \
  'http://localhost:3000/api/todos' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "content": "string"
}'
{"id":"c2849a38-15ab-45b3-8a19-16a4c9973362"}
```



```
✓ Test Results                                                  19 ms
  ✓ TodoServiceImpl.test.ts                                     19 ms
    ✓ should create a new todo and put the message into queue   19 ms
```

Recap

1. **What is Dependency Injection**
2. **Why Serverless applications need it**
3. **How to use it**

Thanks!