Année de Formation 2022/2023
Examen de Fin de Module
M206 Création d'une application Cloud native
Correction : M. Bahae Eddine Halim

Exercice 1: QCM (10 pts)

1. Qu'est-ce que le cloud native?
B. Un ensemble de pratiques et de technologies permettant de concevoir, développer et déployer des applications adaptées aux environnements cloud.

2. Quel est le terme décrivant un service cloud accessible uniquement à un nombre limité de personnes?
C. Cloud privé.

3. Quelle fonctionnalité du cloud computing permet au service de changer de taille ou de volume afin de répondre aux besoins des utilisateurs?
B. Scalabilité.

4. Quelles sont les meilleures pratiques de développement pour les applications cloud native?
B. Adopter une architecture basée sur les microservices.

5. Lequel des éléments suivants est une plateforme cloud développée par Amazon?
B. AWS.

6. Quelle option fournit des machines virtuelles, un stockage virtuel, une infrastructure virtuelle et d'autres actifs matériels?
C. PaaS.

7. Qu'est-ce qu'une image Docker?
B. Un modèle de base pour créer des conteneurs.

8. Quel système d'exploitation est compatible avec Docker?
C. Linux, Windows et macOS.

9. Sur quel langage est basé Node.js?
B. Javascript.

10. Quelle instruction exécute le code du fichier "test.js"?
A. node test.js.

Exercice 2: Expliquer les codes suivants (10 pts)

Code 1:

```
FROM node:16
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "index.js"]
```

Ce code est un fichier Dockerfile qui décrit la création d'une image Docker pour une application Node.js. Voici les étapes qu'il effectue :
- Il utilise l'image de base `node:16`.
- Il définit le répertoire de travail dans le conteneur comme étant `/app`.
- Il copie le fichier `package*.json` (le fichier package.json et package-lock.json) dans le répertoire de travail.
- Il exécute la commande `npm install` pour installer les dépendances de l'application.
- Il copie tous les fichiers du répertoire local dans le répertoire de travail du conteneur.
- Il expose le port 3000 pour que l'application puisse être accessible depuis l'extérieur du conteneur.
- Il définit la commande par défaut à exécuter lorsque le conteneur est démarré, c'est-à-dire `node index.js`.

Code 2:

```
const MongoClient = require('mongodb').MongoClient;
const url = 'mongodb://localhost:27017';
const dbName = 'mybd';

MongoClient.connect(url, function(err, client) {
  console.log("Connexion réussie avec Mongo");
  db = client.db(dbName);
});
```

Ce code est écrit en JavaScript et utilise le module `mongodb` pour se connecter à une base de données MongoDB. Voici ce qu'il fait :
- Il importe le module `MongoClient` à partir de la bibliothèque `mongodb`.
- Il définit l'URL de connexion à MongoDB en utilisant l'adresse `localhost` et le port `27017`.
- Il spécifie le nom de la base de données à laquelle se connecter (`mybd`).
- Il utilise la méthode `connect` de `MongoClient` pour établir une connexion avec la base de données MongoDB.
- S'il n'y a pas d'erreur de connexion, il affiche le message "Connexion réussie avec Mongo" dans la console.
- Il attribue l'objet de base de données (`db`) à la base de données spécifiée.

Bahae Eddine Halim

Exercice 3: (10 pts)

Considérez la ressource `stagiaires.json`. Chaque stagiaire dispose des champs (id, idgroupe, nom, numero, moyenne).

1. Développer les opérations CRUD pour l'entité stagiaire (4 requêtes) (6 pts):
   - Créer un stagiaire: POST /stagiaires
   - Lire tous les stagiaires: GET /stagiaires
   - Mettre à jour un stagiaire: PUT /stagiaires/{id}
   - Supprimer un stagiaire: DELETE /stagiaires/{id}

2. Développer la route permettant d'afficher les stagiaires d'un groupe via son id (id du groupe). (2 pts)
   - GET /groupes/{idgroupe}/stagiaires

3. Développer la route permettant d'afficher le groupe d'un stagiaire donné via son id. (2 pts)
   - GET /stagiaires/{id}/groupe

```
const express = require('express');
const app = express();
app.use(express.json());

let stagiaires = [
  { id: 1, idgroupe: 1, nom: 'Stagiaire 1', numero: 'S001', moyenne: 15 },
  { id: 2, idgroupe: 1, nom: 'Stagiaire 2', numero: 'S002', moyenne: 14 },
  { id: 3, idgroupe: 2, nom: 'Stagiaire 3', numero: 'S003', moyenne: 16 },
  // ...
];

// Récupérer tous les stagiaires
app.get('/stagiaires', (req, res) => {
  res.json(stagiaires);
});

// Récupérer un stagiaire par son id
app.get('/stagiaires/:id', (req, res) => {
  const stagiaire = stagiaires.find(stagiaire => stagiaire.id === parseInt(req.params.id));
  if (!stagiaire) {
    return res.status(404).json({ message: 'Stagiaire non trouvé' });
  }
  res.json(stagiaire);
});

// Créer un nouveau stagiaire
```

```
app.post('/stagiaires', (req, res) => {
  const nouveauStagiaire = req.body;
  stagiaires.push(nouveauStagiaire);
  res.status(201).json(nouveauStagiaire);
});

// Mettre à jour un stagiaire par son id
app.put('/stagiaires/:id', (req, res) => {
  const stagiaire = stagiaires.find(stagiaire => stagiaire.id === parseInt(req.params.id));
  if (!stagiaire) {
    return res.status(404).json({ message: 'Stagiaire non trouvé' });
  }
  stagiaire.nom = req.body.nom;
  stagiaire.numero = req.body.numero;
  stagiaire.moyenne = req.body.moyenne;
  res.json(stagiaire);
});

// Supprimer un stagiaire par son id
app.delete('/stagiaires/:id', (req, res) => {
  const stagiaireIndex = stagiaires.findIndex(stagiaire => stagiaire.id ===
parseInt(req.params.id));
  if (stagiaireIndex === -1) {
    return res.status(404).json({ message: 'Stagiaire non trouvé' });
  }
  stagiaires.splice(stagiaireIndex, 1);
  res.sendStatus(204);
});

// Récupérer les stagiaires d'un groupe par son id
app.get('/groupes/:id/stagiaires', (req, res) => {
  const groupeId = parseInt(req.params.id);
  const stagiairesDuGroupe = stagiaires.filter(stagiaire => stagiaire.idgroupe === groupeId);
  res.json(stagiairesDuGroupe);
});

// Récupérer le groupe d'un stagiaire par son id
app.get('/stagiaires/:id/groupe', (req, res) => {
  const stagiaire = stagiaires.find(stagiaire => stagiaire.id === parseInt(req.params.id));
  if (!stagiaire) {
    return res.status(404).json({ message: 'Stagiaire non trouvé' });
  }
  const groupeId = stagiaire.idgroupe;
  // Supposons que vous avez une liste de groupes
```

```
  const groupes = [
    { id: 1, nom: 'Groupe 1' },
    { id: 2, nom: 'Groupe 2' },
    // ...
  ];
  const groupe = groupes.find(groupe => groupe.id === groupeId);
  if (!groupe) {
    return res.status(404).json({ message: 'Groupe non trouvé' });
  }
  res.json(groupe);
});

// Port d'écoute du serveur
const port = 3000;
app.listen(port, () => {
  console.log(`Serveur en écoute sur le port ${port}`);
});
```

Exercice 4: (10 pts)

Selon vous, quelle solution de stockage de données choisir pour une entreprise ? Quels sont les avantages et les inconvénients du cloud computing?

Le choix de la solution de stockage de données pour une entreprise dépend de plusieurs facteurs, tels que les besoins spécifiques de l'entreprise, la taille de l'entreprise, les exigences en matière de sécurité et de conformité, le budget disponible, etc. Voici quelques options courantes avec leurs avantages et leurs inconvénients :

1. Stockage local (serveurs physiques) :

Avantages :
- Contrôle direct sur l'infrastructure de stockage.
- Pas de dépendance à Internet.
- Latence potentielle plus faible.

Inconvénients :
- Coûts initiaux élevés pour l'achat et la maintenance du matériel.
- Limité en termes d'évolutivité et de flexibilité.
- Risque de perte de données en cas de défaillance matérielle ou de catastrophe.

2. Stockage sur site avec virtualisation :

Avantages :
- Utilisation plus efficace des ressources matérielles.
- Possibilité de créer des environnements virtuels isolés.

- Évolutivité et flexibilité accrues par rapport aux serveurs physiques.

Inconvénients :
- Coûts de matériel et de maintenance continus.
- Nécessite une expertise en virtualisation.
- Risque de perte de données en cas de défaillance matérielle.

3. Cloud privé :

Avantages :
- Contrôle direct sur l'infrastructure de stockage.
- Évolutivité et flexibilité élevées.
- Meilleure disponibilité des données avec des options de redondance.

Inconvénients :
- Coûts initiaux élevés pour la mise en place de l'infrastructure.
- Besoin d'une expertise pour la gestion du cloud privé.
- Possibilité de dépendance à un fournisseur spécifique.

4. Cloud public :

Avantages :
- Coûts initiaux réduits

avec un modèle de paiement à l'utilisation.
- Évolutivité et flexibilité élevées.
- Maintenance et gestion gérées par le fournisseur de services.

Inconvénients :
- Moins de contrôle sur l'infrastructure de stockage.
- Dépendance à une connexion Internet fiable.
- Préoccupations en matière de sécurité et de confidentialité des données.

5. Hybrid Cloud (Combinaison de stockage local et de cloud) :

Avantages :
- Flexibilité pour répartir les charges de travail entre les infrastructures locales et le cloud.
- Possibilité de profiter des avantages du stockage local et du cloud public/privé.

Inconvénients :
- Complexité accrue de la gestion et de l'intégration des deux environnements.
- Nécessite une planification et une architecture soigneuses.

Les avantages et les inconvénients du cloud computing en général sont :
Avantages :
- Évolutivité et flexibilité élevées pour s'adapter à la croissance de l'entreprise.
- Réduction des coûts initiaux grâce à un modèle de paiement à l'utilisation.

- Accès facile aux ressources informatiques à la demande.
- Mise à jour et maintenance automatiques des infrastructures par les fournisseurs de services.

Inconvénients :
- Dépendance à une connexion Internet fiable.
- Préoccupations en matière de sécurité et de confidentialité des données.
- Limitations potentielles en termes de personnalisation et de contrôle de l'infrastructure.
- Risques de dépendance à un fournisseur spécifique et de verrouillage du système.