MERN-STACK

MERN STACK : Content

- What is MERN?
- MERN Components
- Server setup - Basic Hello World
- Server-Less Hello World
- Questionnaire

Introduction to MERN

MERN is a popular technical stack used for building web applications,

It Stands for: M-MongoDB,E-Express.js,R-React and Node.js

The main importance of using MERN Stack in web application development is

- Full-Stack JavaScript Development
- Rapid Development
- Scalability
- Open Source and Community Support
- Versatility
- Integration Capabilities
- High Demand for MERN Stack Developers

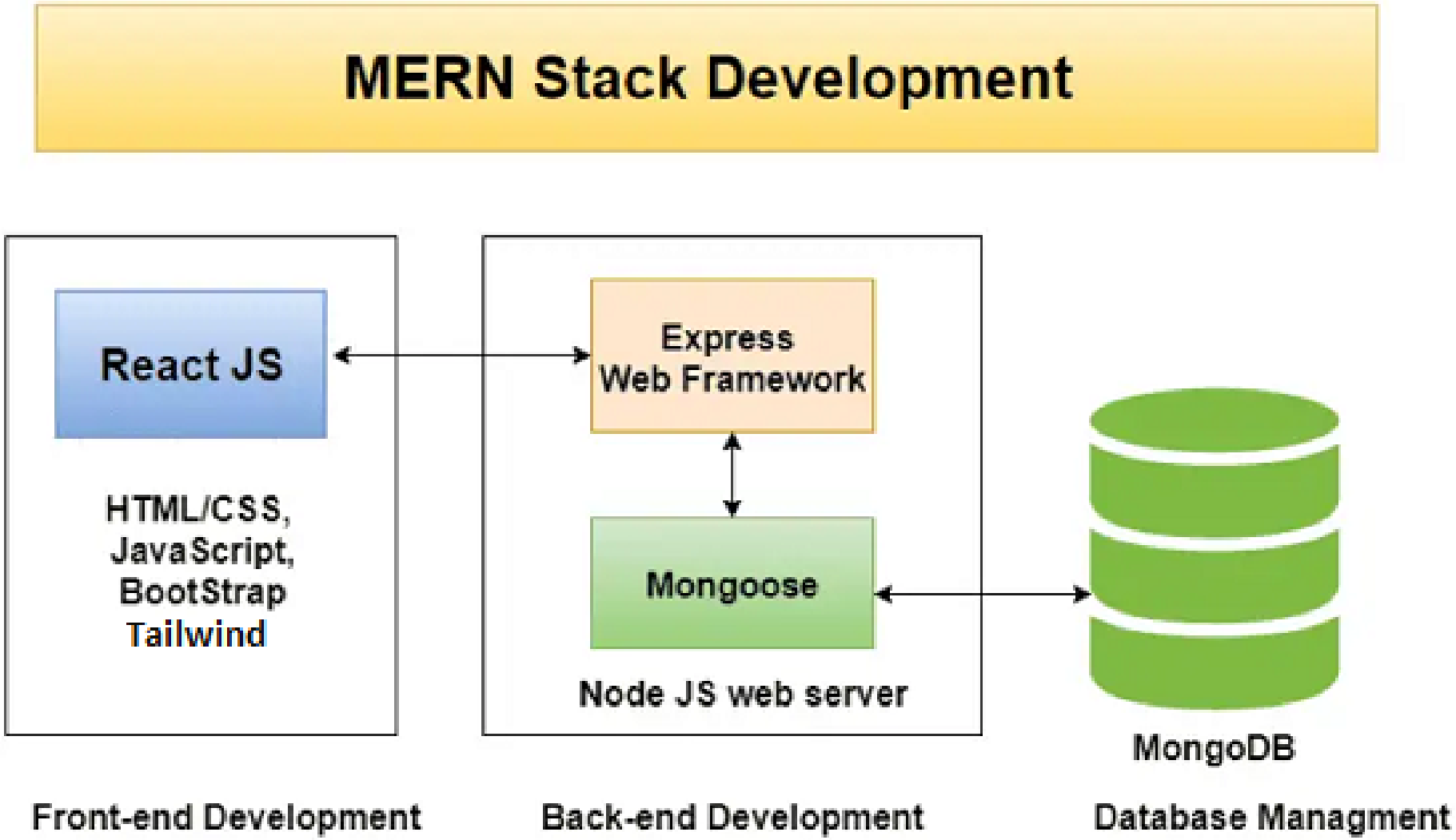
Introduction to MERN

Full Stack Development:
The term Full-Stack Development is used to describe the process of developing both the front-end (client-side) and back-end (server-side) of a web application.
`The main key components of Full Stack Development are:`

- Front-End Development
- Back-End Development
- Database Management
- Deployment

MERN Components

How does the MERN stack work:



MERN Components

MERN is composed of four primary parts, each of which plays a unique role in the creation of web applications:

- **MongoDB:** MongoDB is a NoSQL database that stores data in a flexible, JSON-like format called BSON. It allows for schema-less data storage, which can adapt to various data structures and formats. This flexibility makes it ideal for handling complex and dynamic data requirements.
  Features:
  1. Schema-less design
  2. Scalability
  3. Document-based storage
  4. Built-in replication and high availability

MERN Components

- **Express.js:** Express.js simplifies the creation of server-side applications and APIs. It provides robust features for routing, middleware, and handling HTTP requests and responses, making it easier to build and manage server-side logic.
  Features:
  1. Lightweight
  2. Middleware support for handling requests
  3. Routing for defining endpoints
- **React:** React is used for building dynamic and interactive user interfaces. It allows developers to create reusable UI components and manage the state of applications efficiently.
  Features:
  1. Component-based architecture
  2. Declarative syntax for building UIs
  3. Hooks and context for managing state and side effects

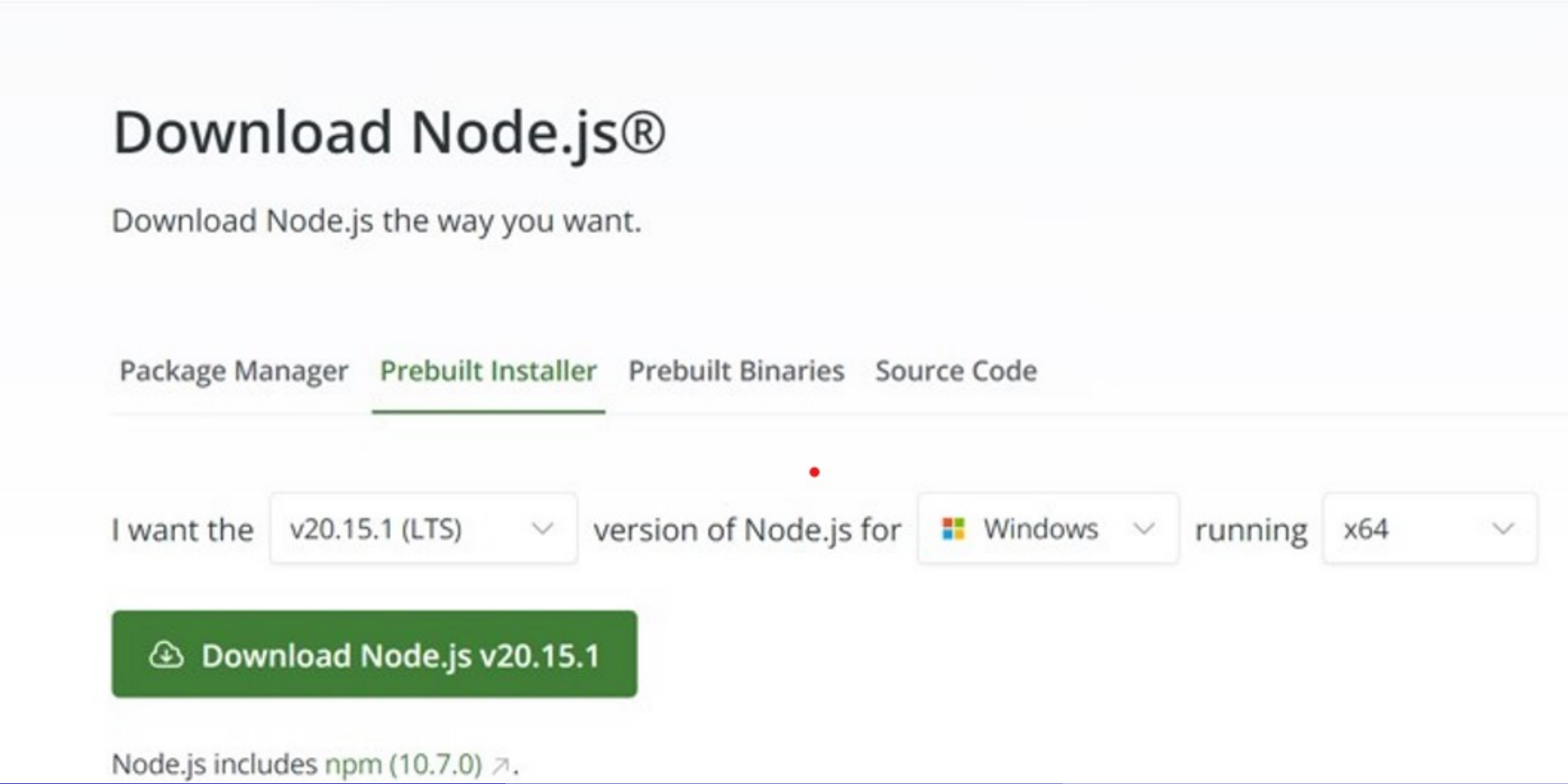
MERN Components

- **Node.js:** Node.js allows JavaScript to be used on the server side. It provides the runtime environment for executing JavaScript code outside of a browser, enabling server-side logic, API development, and interactions with databases.
  Features:
  1. Non-blocking, asynchronous I/O operations
  2. Event-driven architecture
  3. Scalability and high performance
  4. npm (Node Package Manager) for managing packages and dependencies

Basic - Hello World

Pre-Requisities:

- NodeJS, Install the Nodejs using the following link: https://nodejs.org/en/download/prebuilt-installer



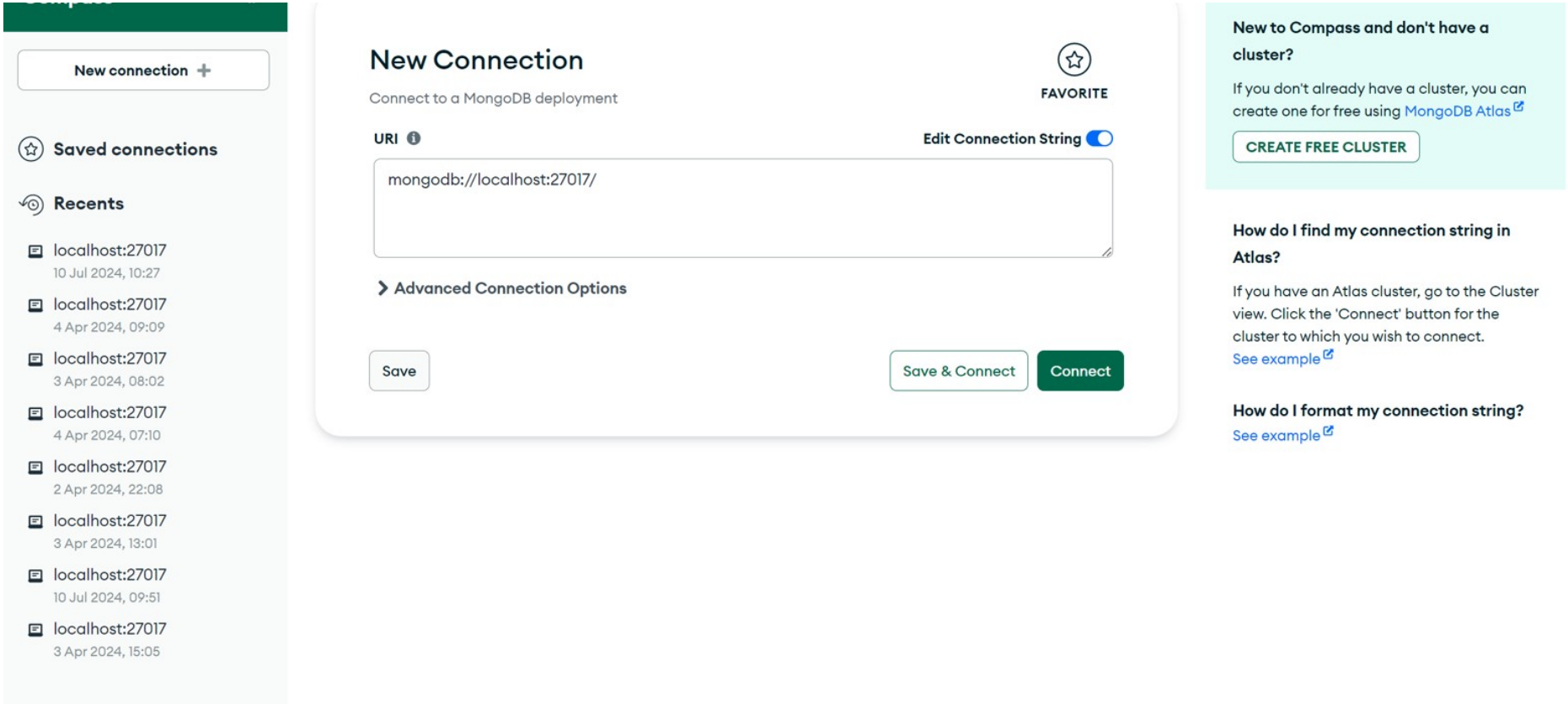
Basic - Hello World

Pre-Requisities:(Contd...)

MongoDB Community Server:(Gives us MongoDB and Compass(GUI))

Download Link:

https://www.mongodb.com/try/download/community-kubernetes-operator



Basic - Hello World

Pre-Requisities:(Contd...)

MongoDB Shell:(CLI)

Download Link:

`https://www.mongodb.com/try/download/shell`

Note: Download package msi

```
mongosh mongodb://127.0.0.

Microsoft Windows [Version 10.0.22631.3810]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vijay>mongosh
Current Mongosh Log ID: 668e4c1dd4714999ffe8e3e7
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.2.11
Using MongoDB:          7.0.12
Using Mongosh:          2.2.11

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-07-10T09:35:13.150+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test>
```

Basic - Hello World

Basic Hello World Application:

Creating an application using the MERN stack (MongoDB, Express, React, Node.js) is a simple example to demonstrate how the stack's components interact to build a full-stack web application and involves deploying each component separately.

Step by Step to create MERN Stack Application:

- **Set Up Your MongoDB Database(Optional):** We setup MongoDB database by installing MongoDB locally or using MongoDB Atlas(Multi-Cloud Database Service).
- Create a folder HelloWorld in the local repository, where we store all the applications in the machine.
- Open HelloWorld folder created above using VS Code Editor.

Basic - Hello World

- **Create the React Frontend:** Open the terminal and run the following command to create front end app:
  - npx create-react-app frontend
  - cd frontend
  - Modify the App.js to display a static message Hello World on the browser as Shown below:

```
import logo from ./logo.svg';
import ./App.css';
function App() {
  return (
    <div>
      <h1>Hello World</h1>
    </div>
  );
}
export default App;
```

- Run the app using the following command: npm start

Basic - Hello World

- **Create the Node.js Backend:** Open the new terminal and run the following commands to create back end app:
  - Create a new directory called backend under the root directory(HelloWorld).
  - cd backend
  - Run the command npm init -y which will create a package.json file, It is the application's configuration file.
  - Install the modules express,mongoose and cors using the following command

    npm install express mongoose cors

    In the above command
    - Express, is a back end web application framework for building RESTful APIs using Node.js
    - Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, Used to interact with MongoDB without writing complex queries.
    - CORS(cross-origin resource sharing) is a node.js module, enables secure communication between applications hosted on different origins.

Basic - Hello World

Create the Node.js Backend:(Contd...)

- Create a new file 'index.js' and write the following code in it.

```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');
const app = express();
app.use(cors());
// MongoDB connection string
const uri = 'mongodb://localhost:27017/usermanagent
';
mongoose.connect(uri, { useNewUrlParser: true,
useUnifiedTopology: true });
mongoose.connection.on('connected', () => {
    console.log('Mongoose connected to MongoDB');
  });
  mongoose.connection.on('error', (err) => {
    console.error('Mongoose connection error:', err)
;
  });
```

Basic - Hello World

```
  app.get('/', (req, res) => {
  res.json({ message: 'Hello World'});
});
const port = process.env.PORT || 5000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

- Run the backend application using the command node index.js, Default backend will be running on port 5000, which we mentioned in index.js, we can change it, but we have to make sure the port is not being used by any other application.
- Connect Frontend and Backend: Update the App.js using the following code in the frontend application to connect backend and fetch Hello World from the backend API.

```
import logo from './logo.svg';
import './App.css';
```

Basic - Hello World

```
import React, { useEffect, useState } from 'react';
function App() {
const [message, setMessage] = useState('');
useEffect(() => {
    fetch('http://10.4.29.45:5000/')
      .then((response) => response.json())
      .then((data) => setMessage(data.message));
  }, []);
  return (
    <div>
      <h1>{message}</h1>
    </div>
  );
}
export default App;
```

Now restart the frontend application to get Hello World from Backend API

Basic - Hello World

Here the ipv4 address entered above is the IP address of local machine, which can obtained using the comman prompt as show below

Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::c994:5e48:1c35:c232%9
   IPv4 Address. . . . . . . . . . . : 10.4.29.45
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.4.29.1

C:\Windows\system32>
```

Server Less Hello World

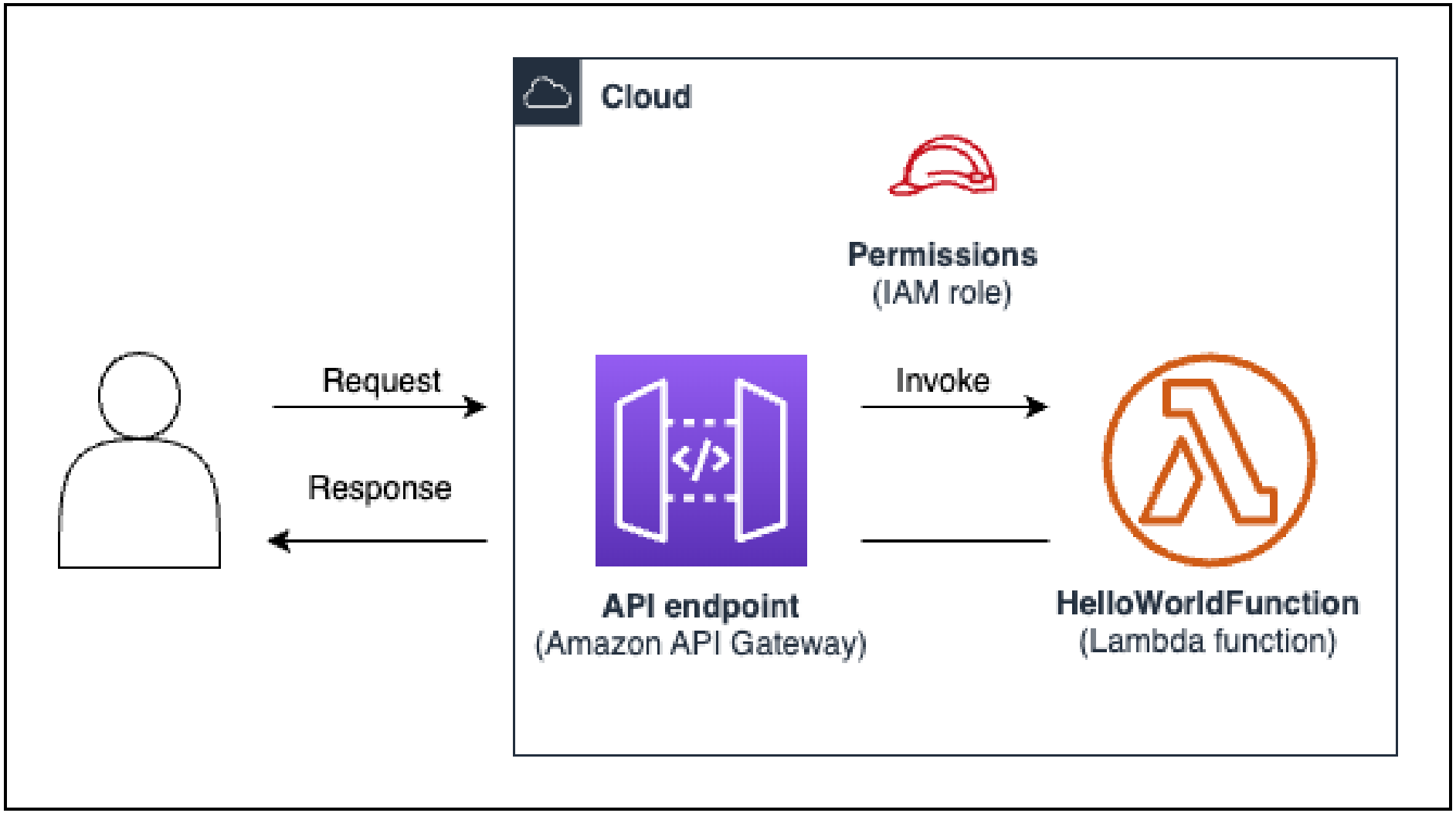
To create a Server less hello world application we use AWS Serverless Application Model Command Line Interface.

The sample Hello World application implements a basic API backend. It consists of the following resources:

- Amazon API Gateway – API endpoint that you will use to invoke your function.
- AWS Lambda – Function that processes the HTTP API GET request and returns a hello world message.
- AWS Identity and Access Management (IAM) role – Provisions permissions for the services to securely interact.

Server Less Hello World

The following diagram shows the components of this application:



Server Less Hello World

Prerequisites

- AWS SAM prerequisites
  - Step 1: Sign up for an AWS account
  - Step 2: Create an IAM user account
  - Step 3: Create an access key ID and secret access key
  - Step 4: Install the AWS CLI
  - Step 5: Use the AWS CLI to configure AWS credentials
- Install the AWS SAM CLI

Server Less Hello World

Here is the step by step process of creating Serverless Hello World using AWS SAM

- Step 1: Initialize the sample Hello World application
- Step 2: Build your application
- Step 3: Deploy your application to the AWS Cloud
- Step 4: Run your application
- Step 5: Interact with your function in the AWS Cloud
- Step 6: Modify and sync your application to the AWS Cloud
- Step 7: (Optional) Test your application locally
- Step 8: Delete your application from the AWS Cloud

Below is the link to follow to Deploy the Hello World application With AWS SAM

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-getting-started-hello-world.html

Questionnaire

Short Answer Questions

1. How does the MERN stack facilitate full-stack web development?
2. Explain how Express.js is used within the MERN stack.
3. How does Node.js contribute to the MERN stack's functionality?
4. What is serverless computing, and how does it differ from traditional server-based models?
5. What are the benefits of using a serverless architecture for deploying simple applications?

Questionnaire

Long Answer Questions

1. Explain the MERN stack and its components. How does the combination of MongoDB, Express.js, React, and Node.js work together to facilitate full-stack web development?
2. Discuss the role of each component in the MERN stack. How does MongoDB serve as a NoSQL database, and what are the benefits of using it over traditional SQL databases?
3. How does the MERN stack support the development of single-page applications (SPAs)? Discuss the flow of data between the front-end and back-end components, and how the stack's components enable a seamless user experience.
4. What is serverless computing, and how does it differ from traditional server-based architecture? Discuss the advantages and challenges of using a serverless model, particularly in the context of scalability, cost management, and application deployment.

Thank you