**Day 1:**

1. Use the reference of YouTube[YT] :

S1: A new user account is created in Gmail/YT

S2: The user logs in to YT using the account

S3: Create a subscription service in YT

S4 : User :: Create a channel

S5 :: user --> channel --> uploads a video

What are the publish + subscribe services that will be utilized?

**Solution:**

Account Creation (S1):

Publisher: User Registration Service

Subscribers: User Database, Email Notification Service, Authentication Service

User Login (S2):

Publisher: Authentication Service

Subscribers: User Profile Service, Session Management Service, User Activity Tracker

Subscription Creation (S3):

Publisher: Subscription Management Service

Subscribers: Channel Owner's Dashboard, Subscription Notification Service, User Subscription Data

Channel Creation (S4):

Publisher: Channel Creation Service

Subscribers: Channel Metadata Storage, User Dashboard, Channel Analytics Service

Video Upload (S5):

Publisher: Video Upload Service

Subscribers: Video Processing System, Content Delivery Network (CDN) Service, Video Analytics Platform

**Day 2: 04/08/2023 [Delivery is due post assessment on 05/08]**

1. Refer FB + LinkedIn API's :: Try some of the API's in Postman

- Which API referred

- What did you understand from the APIs

- What are the parameters + endpoints

- What are the business utilities of the APIs

**Solution:**

Facebook API

* Graph API: Graph API is the primary way to interact with Facebook's data. It allows you to access data about users, pages, groups, events, and more.
* Messenger API: The Messenger API allows you to build chatbots and other conversational experiences on Facebook Messenger.
* Instagram API: The Instagram API allows you to access data about users, posts, and other content on Instagram.

LinkedIn API

* LinkedIn API: The LinkedIn API allows you to access data about users, profiles, companies, and jobs on LinkedIn.
* Sales Navigator API: The Sales Navigator API allows you to access data about leads and prospects on LinkedIn.
* Talent Solutions API: The Talent Solutions API allows you to access data about job postings and candidates on LinkedIn.

Business Utilities:

* Data collection: These APIs can be used to collect data about users, products, and other entities.
* Data analysis: These APIs can be used to analyze data and identify trends.
* Communication: These APIs can be used to communicate with users, send notifications, and build chatbots.
* Marketing: These APIs can be used to target ads, track leads, and measure the effectiveness of marketing campaigns.

2. To practice some of the quick NodeJS code bases :

https://docs.solace.com/API/Messaging-APIs/NodeJS-API/node-js-home.htm

https://tutorials.solace.dev/nodejs

https://github.com/SolaceSamples/solace-samples-nodejs

3. Github: https://www.freecodecamp.org/news/git-and-github-for-beginners/

4. DB Schema --> Structure of the tables : Read | Read | Read :: Practice

5. Refer --> https://docs.solace.com/Cloud/service-class-limits.htm

6. Read and explain: What is an endpoint [NOT API endpoints --> Queue endpoint + topic endpoint]

**Solution:**

An endpoint is a network address that identifies a specific resource or service. In the context of APIs, an endpoint is the address of a specific resource that can be accessed using the API.

For example, the following is an endpoint for the Facebook Graph API:

https://graph.facebook.com/v2.12/me

This endpoint can be used to get information about the current user.

Endpoints are typically identified by a URL. However, they can also be identified by other means, such as a hostname or a port number.

The term "endpoint" is also used to refer to the endpoint of a network connection. In this context, an endpoint is a device or a service that is connected to a network.

For example, a computer that is connected to the internet is an endpoint. The internet itself is also an endpoint.

7. What is High Availability Architecture? What happens in High Availability Architecture

**Solution:**

High availability (HA) architecture is a design approach that ensures that critical applications and systems are available to users even when there are failures in the underlying infrastructure. HA architectures typically use redundant components and failover mechanisms to ensure that there is always a backup available in case of a failure.

There are many different ways to implement HA architecture, but some common techniques include:

* Redundancy: This involves having multiple copies of critical components, such as servers, storage devices, and network links. If one component fails, the other components can take over to keep the application or system running.
* Failover: This involves having a mechanism in place to automatically switch to a backup component in the event of a failure. For example, if a server fails, a failover mechanism can automatically switch to a backup server.
* Load balancing: This involves distributing traffic across multiple servers to improve performance and availability. If one server fails, the load can be redistributed to the remaining servers to keep the application or system running.

HA architectures are typically used for critical applications and systems that cannot afford to be unavailable. For example, banks, hospitals, and government agencies often use HA architectures to ensure that their critical applications are always available.

Here are some of the benefits of using HA architecture:

* Increased availability: HA architecture can help to ensure that critical applications and systems are available even when there are failures in the underlying infrastructure.
* Improved performance: HA architecture can help to improve performance by distributing traffic across multiple servers.
* Reduced downtime: HA architecture can help to reduce downtime by automatically switching to a backup component in the event of a failure.

Here are some of the challenges of using HA architecture:

* Cost: HA architecture can be more expensive than traditional architecture.
* Complexity: HA architecture can be more complex to design and implement than traditional architectures.
* Maintenance: HA architecture requires more maintenance than traditional architecture.

Overall, HA architecture is a valuable tool for ensuring the availability of critical applications and systems. However, it is important to weigh the benefits and challenges of HA architecture before deciding whether or not to implement it.

8. Write a NodeJS code for :

a. Create a dummy calculator :: Addition/Subtraction/Division [check for division by 0]

b. parse a sample XML file using NodeJS and display the response in browser

**Solution:**

1. const http = require('http');

const fs = require('fs');

const xml2js = require('xml2js');

// Create a simple calculator

const calculator = {

add: (a, b) => a + b,

subtract: (a, b) => a - b,

divide: (a, b) => {

if (b === 0) {

throw new Error('Division by zero is not allowed.');

}

return a / b;

},

};

// Create an HTTP server

const server = http.createServer((req, res) => {

if (req.url === '/calculator') {

const result = calculator.add(10, 5); // You can change the operation here

res.writeHead(200, { 'Content-Type': 'text/plain' });

res.end(`Result: ${result}`);

} else if (req.url === '/xml') {

fs.readFile('sample.xml', 'utf-8', (err, data) => {

if (err) {

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Internal Server Error');

} else {

xml2js.parseString(data, (xmlErr, result) => {

if (xmlErr) {

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Error parsing XML');

} else {

res.writeHead(200, { 'Content-Type': 'application/xml' });

res.end(JSON.stringify(result));

}

});

}

});

} else {

res.writeHead(404, { 'Content-Type': 'text/plain' });

res.end('Not Found');

}

});

// Start the server

server.listen(3000, () => {

console.log('Server is running on http://localhost:3000');

});

b.

<books>

<book>

<title>Book 1</title>

<author>Author 1</author>

</book>

<book>

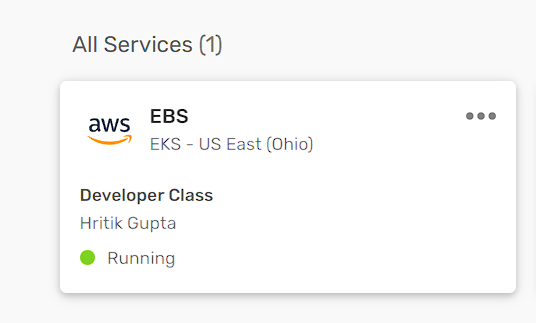
<title>Book 2</title>

<author>Author 2</author>

</book>

</books>

9. Create a new Solace free a/c --> Create a new EBS service :: Deploy it in AWS --> Ohio region + Custom name for Message-VPN and Cluster



Day 3: 07/08/2023

1. Read a text file + HTML file + JSON file using async method in NodeJS [1 program or 3 different programs]

**Solution:** Node.js file

const fs = require('fs');

// Read text file

fs.readFile('example.txt', 'utf8', (err, textData) => {

if (err) {

console.error('Error reading text file:', err);

return;

}

console.log('Text File Content:');

console.log(textData);

});

// Read HTML file

fs.readFile('example.html', 'utf8', (err, htmlData) => {

if (err) {

console.error('Error reading HTML file:', err);

return;

}

console.log('HTML File Content:');

console.log(htmlData);

});

// Read JSON file

fs.readFile('example.json', 'utf8', (err, jsonData) => {

if (err) {

console.error('Error reading JSON file:', err);

return;

}

console.log('JSON File Content:');

console.log(JSON.parse(jsonData));

});

**Example.txt:**

This is an example text file.

It contains some random text for demonstration purposes.

Feel free to modify this text as needed.

**Example.html:**

<!DOCTYPE html>

<html>

<head>

<title>Example HTML File</title>

</head>

<body>

<h1>This is an example HTML file.</h1>

<p>It contains some random HTML content for demonstration purposes.</p>

<p>Feel free to modify this HTML as needed.</p>

</body>

</html>

**Example.json**

{

"title": "Example JSON File",

"description": "This is an example JSON file.",

"content": "It contains some random JSON data for demonstration purposes. Feel free to modify this JSON as needed."

}

2. Practice with different use cases :

- Try Me : Pub-Sub scenario

- Get the hands cleared on Solace CLI Access

**COMPLETED**

Day 4: 08/08/2023

1. Practice NodeJS code for sync + async - 3 each

2. Create a calculator in NodeJS using async :: Add + Subtract + Multiply + Divide + Percentage

**Solution:** const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

function getInput(prompt) {

return new Promise((resolve) => {

rl.question(prompt, (answer) => {

resolve(answer);

});

});

}

async function add() {

const num1 = parseFloat(await getInput('Enter the first number: '));

const num2 = parseFloat(await getInput('Enter the second number: '));

console.log(`Result: ${num1 + num2}`);

}

async function subtract() {

const num1 = parseFloat(await getInput('Enter the first number: '));

const num2 = parseFloat(await getInput('Enter the second number: '));

console.log(`Result: ${num1 - num2}`);

}

async function multiply() {

const num1 = parseFloat(await getInput('Enter the first number: '));

const num2 = parseFloat(await getInput('Enter the second number: '));

console.log(`Result: ${num1 \* num2}`);

}

async function divide() {

const num1 = parseFloat(await getInput('Enter the first number: '));

const num2 = parseFloat(await getInput('Enter the second number: '));

if (num2 === 0) console.log('Error: Cannot divide by zero');

else console.log(`Result: ${num1 / num2}`);

}

async function calculatePercentage() {

const num = parseFloat(await getInput('Enter the number: '));

const percentage = parseFloat(await getInput('Enter the percentage: '));

console.log(`Result: ${(num \* percentage) / 100}`);

}

async function main() {

while (true) {

console.log('\nCalculator Menu:');

console.log('1. Add');

console.log('2. Subtract');

console.log('3. Multiply');

console.log('4. Divide');

console.log('5. Calculate Percentage');

console.log('6. Exit');

const choice = await getInput('Enter your choice: ');

switch (choice) {

case '1':

await add();

break;

case '2':

await subtract();

break;

case '3':

await multiply();

break;

case '4':

await divide();

break;

case '5':

await calculatePercentage();

break;

case '6':

rl.close();

return;

default:

console.log('Invalid choice. Please try again.');

}

}

}

main();

3. Solace CLI --> show session :: What is the default unit of the Timeout?

**Solution:** The default unit of the Timeout for the show session command in the Solace CLI is minutes. The timeout value can be specified as an integer from 0 to 43200, where 0 means that the timeout is disabled. For example, to set the timeout to 15 minutes, you would use the following command:

solace show session timeout 15

4. Practice :: Commands in Solace CLI

5. Practice : Queue + Topic Subscriptions in WebGUI

6. Try the Queue creation using Solace CLI