1. Basics of Postman

1. What is Postman? Why is it used?

Postman is an API (Application Programming Interface) development and testing tool. It's an API client that allows you to easily create, send, and test HTTP requests. It provides a user-friendly interface to manage and organize requests, making it a popular choice for developers and testers.

Why it's used:

- **API Development:** Developers use it to build and test their own APIs.
- **API Testing:** Testers use it to validate API functionality, performance, and security.
- **API Documentation:** It helps in creating and maintaining API documentation.
- **Collaboration:** Teams can share collections and environments, enabling seamless collaboration.
- **Automation:** It supports writing scripts for pre-request and test automation, and can be integrated into CI/CD pipelines.

2. Difference between Postman and cURL?

Feature	Postman	cURL
Type	GUI (Graphical User Interface) tool	Command-line tool
Ease of Use	User-friendly, intuitive interface. Easy for beginners.	Requires knowledge of command-line syntax. Can be complex.
Functionality	Full-fledged API development platform. Includes features like collections, environments, runners, mocks, etc.	Primarily a tool for transferring data. Focused on sending requests.
Collaboration	Excellent for team collaboration (workspaces, sharing collections).	Difficult to share and manage complex request sequences.

Scripting	Built-in JavaScript execution environment for pre-request and test scripts.	Can be used within shell scripts, but lacks a dedicated scripting environment.
Use Case	Best for interactive development, testing, and team collaboration.	Ideal for quick one-off requests, scripting, and CI/CD pipelines where a GUI is not available.

Example:

- **Postman:** You open the app, enter the URL, select `GET`, and click 'Send'.
- **cURL:** You type `curl -X GET "https://api.example.com/data"` in your terminal.

3. How do you install and set up Postman?

- **Download:** Go to the official Postman website and download the installer for your operating system (Windows, macOS, or Linux).
- **Install:** Run the installer. On Windows, it's a simple executable. On macOS, you drag the app to the Applications folder.
- **Sign Up/Log In:** When you first open Postman, it will prompt you to create a free account or sign in. This is recommended for syncing your work and collaborating with a team.
- **Start a Workspace:** Once logged in, you can create a new workspace (e.g., "My First API Project") to start organizing your work.

4. What is a workspace in Postman?

A workspace is a private or shared space where you organize your API development work. It's like a project folder. Workspaces help you manage your collections, environments, and other elements related to a specific project or a set of related APIs.

- **Personal Workspace:** Visible only to you.
- **Team Workspace:** Shared with your team members, allowing for real-time collaboration.

5. What is a collection in Postman?

A collection is a group of saved API requests. It's a fundamental feature for organizing and documenting your work. You can group related requests (e.g., all requests for the "User Management" API) into a single collection.

Key benefits:

- **Organization:** Keeps your requests tidy and easy to find.
- **Sharing:** You can easily share an entire collection with your team.
- **Automation:** Collections can be run sequentially using the Collection Runner.

- **Documentation:** You can add descriptions to collections and requests to document your API.

Example: A collection named "E-commerce API" might contain requests like "Get all products," "Add a new product," "Update product details," and "Delete product."

6. What is an environment in Postman?

An environment is a set of variables that you can use in your Postman requests. It allows you to switch between different setups without changing your requests. For example, you can have a "Development" environment with `dev.api.example.com` and a "Production" environment with `prod.api.example.com`.

Example:

- **Environment:** `Development`
  - `url`: `https://dev.api.example.com`
  - `api_key`: `dev_key_123`
- **Request URL:** `{{url}}/users?key={{api_key}}`

This single request can be executed against different environments just by selecting the desired environment from the dropdown menu.

7. Difference between global variables, environment variables, and collection variables in Postman?

- **Global Variables:** Have the widest scope. They are available in all workspaces and collections. Use them for variables that are consistent across your entire Postman usage (e.g., a username or password for a test account that is used everywhere).
- **Environment Variables:** Limited to the selected environment. They are used for environment-specific data like API URLs, authentication tokens, or test data for a specific environment (e.g., `dev`, `staging`, `production`).
- **Collection Variables:** Limited to a specific collection. They are ideal for data that is specific to a group of requests, regardless of the environment (e.g., a `user_id` that is used by multiple requests within a "User Management" collection).

Scope Hierarchy: Global > Environment > Collection > Data Variables (from Collection Runner) > Local (Script-level). When a variable with the same name exists at multiple levels, Postman uses the value from the narrowest scope.

8. What is a request in Postman?

A request is a single API call to a specific endpoint. In Postman, a request is composed of several parts:

- **URL:** The endpoint you are calling (e.g., `https://api.example.com/users`).
- **Method:** The HTTP method (`GET`, `POST`, `PUT`, `DELETE`, etc.).
- **Headers:** Metadata for the request (e.g., `Content-Type`, `Authorization`).
- **Body:** The data you are sending with the request (for `POST`, `PUT`, etc.).

- **Authorization:** The type of authentication you are using (e.g., Bearer Token, API Key).
- **Query Params:** Key-value pairs appended to the URL (`?key1=value1&key2=value2`).

9. What are request methods supported in Postman?

Postman supports all standard HTTP request methods. The most common ones are:

- `GET`: Retrieves data from a server.
- `POST`: Submits data to a server to create a new resource.
- `PUT`: Updates an existing resource on the server.
- `PATCH`: Applies partial modifications to a resource.
- `DELETE`: Deletes a resource from the server.
- `HEAD`: Retrieves the headers of a resource without the body.
- `OPTIONS`: Describes the communication options for the target resource.

10. What are request headers and body in Postman?

- **Request Headers:** Key-value pairs that contain metadata about the request. They provide information to the server about the request itself.
  - `Content-Type`: Specifies the format of the request body (e.g., `application/json`).
  - `Authorization`: Contains credentials to authenticate the request.
  - `Accept`: Specifies the type of content the client can handle in the response.
- **Request Body:** The actual data that is sent to the server. This is typically used for `POST` and `PUT` requests. Postman supports various body types:
  - **`form-data`:** Used for sending form data, including files.
  - **`x-www-form-urlencoded`:** Used for simple key-value pairs, similar to a URL query string.
  - **`raw`:** Allows you to send raw data, such as JSON, XML, or plain text.
  - **`binary`:** Used for sending non-textual data like images or executables.
  - **`GraphQL`:** Specifically for sending GraphQL queries.

---

2. Working with APIs

11. How do you test a GET API in Postman?

1. Open a new request tab.
2. Set the request method to `GET`.
3. Enter the API endpoint URL in the address bar.
4. If there are query parameters, enter them in the "Params" tab below the URL bar.
5. Click the "Send" button.
6. The response will appear in the lower pane, showing the status code (e.g., `200 OK`), response time, size, and the response body.

Example:

- **Method:** `GET`
- **URL:** `https://api.example.com/users?id=123`

12. How do you test a POST API with a JSON body in Postman?

1. Open a new request tab.
2. Set the request method to `POST`.
3. Enter the API endpoint URL.
4. Go to the "Body" tab.
5. Select the "raw" option.
6. From the dropdown menu (which defaults to "Text"), select "JSON".
7. Enter your JSON payload in the text area.
8. Click "Send".

Example:

- **Method:** `POST`
- **URL:** `https://api.example.com/users`
- **Body (raw, JSON):**
- JSON

```json
{
  "name": "John Doe",
  "email": "john.doe@example.com"
}
```

-
-

13. How do you test file upload API in Postman?

1. Open a new request tab.
2. Set the request method to `POST` or `PUT`.
3. Enter the API endpoint URL.
4. Go to the "Body" tab.
5. Select the `form-data` option.
6. In the key-value editor, hover over the key input field and a dropdown will appear. Select "File" from the dropdown.
7. Click the "Select Files" button next to the value field.
8. Choose the file(s) you want to upload from your computer.
9. Click "Send".

14. How to send form-data and x-www-form-urlencoded data?

- **`x-www-form-urlencoded`:**
  - Go to the "Body" tab.
  - Select `x-www-form-urlencoded`.
  - Add key-value pairs in the table. Postman will automatically encode them.
  - This is typically used for simple form submissions.

- **form-data:**
    - Go to the "Body" tab.
    - Select form-data.
    - This is more versatile. You can add key-value pairs.
    - For the value, you can select "Text" (default) or "File".
    - This is required for file uploads and for sending mixed data types (text and files) in a single request.

15. How to pass query parameters and path variables in Postman?

- **Query Parameters:**
    - Used to filter or paginate results (/users?status=active).
    - Enter them directly in the URL after ? or use the "Params" tab.
    - When you type ?status=active in the URL, Postman automatically populates the "Params" tab. Conversely, adding key-value pairs in the "Params" tab updates the URL.
- **Path Variables:**
    - Used to identify a specific resource (/users/123).
    - Define them using a colon followed by the variable name in the URL (/users/:userId).
    - Postman's URL bar is a smart text field. When you type https://api.example.com/users/123, the "Params" tab will not show 123. Path variables are part of the resource path itself. If you use variables (e.g., {{userId}}), Postman will show them in a Path Variables section in the Params tab.

16. How to test secured APIs with API key authentication in Postman?

1. Go to the "Authorization" tab of your request.
2. Select the "API Key" type.
3. Enter the key's name (e.g., x-api-key).
4. Enter the key's value.
5. Specify where the key should be sent: "Header", "Query Params", etc.
6. Click "Send". Postman will automatically add the key to the specified location.

17. How to test OAuth2 or JWT authentication in Postman?

- **OAuth 2.0:**
    1. Go to the "Authorization" tab.
    2. Select "OAuth 2.0".
    3. Click "Get New Access Token".
    4. Fill in the required fields (Auth URL, Access Token URL, Client ID, Client Secret, etc.).
    5. Postman will handle the entire OAuth flow (authorization code grant, etc.) and get an access token for you.
    6. The access token will be automatically added to the request header as a Bearer Token.

- **JWT (JSON Web Token):**
    1. JWTs are often received from a separate authentication endpoint (e.g., a login API).
    2. In a `POST` request to the login endpoint, get the JWT from the response body.
    3. In the "Tests" tab of the login request, save the JWT to an environment variable: `pm.environment.set("jwt_token", pm.response.json().token);`
    4. In all subsequent requests that require authentication, go to the "Authorization" tab, select "Bearer Token", and set the token value to `{{jwt_token}}`.

18. How do you test HTTPS APIs with SSL certificates in Postman?

1. Go to `Postman > Settings > Certificates` (or the gear icon in the header).
2. Click "Add Certificate".
3. Specify the host for which you are adding the certificate.
4. Upload the client certificate file (`.crt` or `.pem`) and the private key file (`.key`).
5. If the private key is encrypted, provide the passphrase.
6. Postman will automatically use this certificate for requests made to the specified host.

19. How to set request timeouts in Postman?

- Go to `Postman > Settings > General`.
- You will see an option for "Request timeout in ms".
- Enter the desired timeout value in milliseconds (e.g., `5000` for 5 seconds).
- A value of `0` means no timeout.
- This is a global setting that applies to all requests.

20. How to send multiple requests in Postman?

- **Manually:** Open multiple tabs and send each request individually.
- **Collection Runner:** The most common way.
    1. Click the "Runner" button in the bottom-right corner of the Postman window or the "Run" button on a collection's overview page.
    2. Select the collection and the environment you want to use.
    3. You can reorder the requests, select a data file for data-driven testing, and set iteration count and delays.
    4. Click "Run [Collection Name]". Postman will execute all requests in the specified order and provide a summary of the results.

---

3. Postman Automation & Scripts

21. What are Pre-request Scripts in Postman?

Pre-request scripts are JavaScript code that runs *before* a request is sent. They are executed in a sandbox environment and are useful for preparing data or variables for your request.

Common use cases:

- Generating a dynamic timestamp.
- Calculating a hash or a signature for authentication.
- Setting an environment variable.
- Pulling data from an external source or another variable.

Example:

To generate a random user ID before a POST request:

```JavaScript
// Generate a random number between 1 and 1000
const randomId = Math.floor(Math.random() * 1000) + 1;
// Set a collection variable named "userId"
pm.collectionVariables.set("userId", randomId);
```

The request body can then use `{{userId}}`.

22. What are Tests in Postman?

Tests are JavaScript code that runs *after* a request has been sent and a response has been received. They are used to validate the response and ensure the API is working as expected.

Common use cases:

- Verifying the HTTP status code (e.g., `200 OK`).
- Checking if the response body contains a specific value.
- Validating the data type or structure of the response.
- Storing a value from the response for use in a subsequent request.

Example:

```JavaScript
// Test if the status code is 201 (Created)
pm.test("Status code is 201", function () {
  pm.response.to.have.status(201);
});

// Test if the response body contains the string "Success"
pm.test("Body matches string", function () {
  pm.expect(pm.response.text()).to.include("Success");
});
```

23. How do you write assertions in Postman tests? (e.g., `pm.response.to.have.status(200)`)

Postman uses the `Chai.js` assertion library, which provides a clean and readable syntax for writing tests. The core object is `pm.expect()`.

Common Assertions:

- **Status Code:**
- JavaScript

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

-
-
- **Response Body:**
- JavaScript

```
const responseBody = pm.response.json();
pm.test("Response body has a property 'id'", function () {
    pm.expect(responseBody).to.have.property('id');
});

pm.test("The 'name' property is 'John Doe'", function () {
    pm.expect(responseBody.name).to.eql("John Doe");
});
```

-
-
- **Headers:**
- JavaScript

```
pm.test("Content-Type header is application/json", function () {
    pm.response.to.have.header("Content-Type", "application/json");
});
```

-
-

24. How to extract values from API response in Postman?

You can extract values from the response body or headers and store them in variables for later use.

- **From JSON Body:**
- JavaScript

```
// Parse the JSON response
const responseBody = pm.response.json();
// Get the 'id' from the response
const userId = responseBody.id;
// Set an environment variable
pm.environment.set("newlyCreatedUserId", userId);
```

- **From Headers:**
- JavaScript

```
const sessionId = pm.response.headers.get("X-Session-ID");
pm.environment.set("sessionId", sessionId);
```

- **From XML Body:**
- JavaScript

```
const xmlResponse = xml2Json(pm.response.text());
const orderId = xmlResponse.order.orderId;
pm.environment.set("orderId", orderId);
```

25. How to pass values between requests using variables?

This is a fundamental concept for building workflows.

1. **Request 1 (`POST /users`):** Creates a new user.
    - **Tests Tab:** Extract the newly created user's ID from the response and save it to an environment variable.
    - JavaScript

```
const responseData = pm.response.json();
pm.environment.set("newUserId", responseData.id);
```

2. **Request 2 (`GET /users/:id`):** Retrieves the user created in Request 1.
    - URL: Use the environment variable in the URL. https://api.example.com/users/{{newUserId}}

When Request 2 is sent, Postman will substitute `{{newUserId}}` with the value set by the tests in Request 1.

26. What is `pm` object in Postman scripting?

The `pm` object is the central object in Postman's sandbox environment. It provides access to all the Postman-specific functions and data, and is the entry point for all scripting.

Key properties and methods:

- `pm.request`: Information about the current request.
- `pm.response`: Information about the response (only in tests).
- `pm.environment`: Access to environment variables (`.get()`, `.set()`).
- `pm.collectionVariables`: Access to collection variables.
- `pm.globals`: Access to global variables.
- `pm.test()`: Defines a test.
- `pm.expect()`: The assertion library entry point.
- `pm.sendRequest()`: To send a new, unsaved request from a script.
- `pm.variables`: Provides a hierarchical view of variables (e.g., `pm.variables.get("variable_name")`).

27. Difference between `pm.environment.set()` and `pm.collectionVariables.set()`?

The difference is in the scope of the variable you are setting.

- `pm.environment.set("varName", "value")`: Sets a variable in the **current active environment**. This variable will be available to any request that uses this environment. It's used for data that is specific to a particular environment (e.g., `dev` or `prod`).
- `pm.collectionVariables.set("varName", "value")`: Sets a variable that is available only within the **current collection**. This variable is independent of the environment and is useful for data that is relevant to all requests within that collection.

28. How to use JavaScript in Postman tests?

Postman's scripting environment is based on Node.js and uses standard JavaScript (ES5). You can write any JavaScript code within the "Pre-request Script" and "Tests" tabs.

Postman provides:

- **The `pm` object** for Postman-specific functions.
- **The `Chai.js` library** for assertions (`pm.expect()`).
- **Helper libraries:** `lodash`, `moment`, `uuid`, etc.
- **`console.log()`** for debugging.

Example:

```javascript
JavaScript

// Using a standard JS function and a helper library
```

```
const moment = require('moment');
const today = moment().format('YYYY-MM-DD');

pm.environment.set("current_date", today);
console.log("Current date is: " + pm.environment.get("current_date"));

// If response status is OK, parse the body
if (pm.response.to.have.status(200)) {
    const data = pm.response.json();
    // Loop through an array in the response
    data.items.forEach(item => {
        console.log("Item ID: " + item.id);
    });
}
```

29. How to log messages in Postman console?

Use `console.log()`, `console.info()`, `console.warn()`, and `console.error()` within your scripts.

1. Open the Postman Console by clicking the "Console" icon in the bottom-left corner.
2. Add a `console.log()` statement to your Pre-request or Test script.
3. Send the request.
4. The output of your `console.log()` will appear in the Postman Console, helping you debug your script.

Example:

```JavaScript
console.log("Starting script...");
pm.test("Response is successful", () => {
    pm.expect(pm.response.status).to.eql(200);
    console.log("Test passed!");
});
```

30. How to debug Postman scripts?

- **Postman Console:** The primary debugging tool. Use `console.log()` statements to print variable values, test assertions, and track the flow of your script.
- **`console.log(pm.request)`:** To inspect the details of the request being sent.
- **`console.log(pm.response)`:** To inspect the raw response object, including headers and status.
- **Variable Quick Look:** Hover over a variable in a script to see its current value.
- **Environment Viewer:** Click the "eye" icon next to the environment selector to view all current variables and their values.
- **Breakpoints (Experimental):** As of a certain version, Postman has introduced an experimental feature to set breakpoints in scripts for step-by-step debugging. Check the latest Postman documentation for details.

4. Collections & Workflows

31. What is a Postman Collection Runner?

The Collection Runner is a feature that allows you to run all the requests in a collection in a specific order. It provides a user interface to configure the run, view the results, and perform automated testing.

Features:

- **Run Sequence:** Execute requests sequentially.
- **Iterations:** Run the entire collection multiple times.
- **Data Files:** Use external data (CSV or JSON) for data-driven testing.
- **Delays:** Add delays between requests to simulate real-world scenarios.
- **Test Results:** Displays a summary of all test assertions (passed/failed).

32. How do you run multiple requests as a sequence?

1. Open the Collection Runner from the bottom-right menu or from the collection's "Run" tab.
2. Select the collection you want to run.
3. Drag and drop the requests in the desired order.
4. Specify the environment and the number of iterations.
5. Click "Run [Collection Name]".

The Runner will execute the requests in the order they are listed, and any variables set in a script will be available to subsequent requests.

33. What is data-driven testing in Postman?

Data-driven testing is a testing strategy where you run the same request or collection multiple times with different sets of data. In Postman, you can achieve this using the Collection Runner and a CSV or JSON data file.

Steps:

1. Create a CSV or JSON file with your test data.
   - **CSV:** `user_id,username,email`
   - **JSON:** `[ { "user_id": 1, ... }, { "user_id": 2, ... } ]`
2. In your Postman request, use variables in your URL, body, or headers that match the column headers/keys in your data file (e.g., `{{user_id}}`).
3. In the Collection Runner, select the number of iterations and upload the data file.
4. Postman will run the collection once for each row/object in your data file, substituting the variable values.

34. How to use external CSV/JSON files with Collection Runner?

1. In the Collection Runner window, go to the "Data" section.
2. Click "Select File" and choose your CSV or JSON file.
3. Postman will show a preview of the data.

4. The "Iterations" count will automatically adjust to the number of rows or objects in your data file.
5. Variables in your requests (e.g., `{{username}}`) will automatically be populated from the `username` column in your data file during each iteration.

35. What is Postman Monitor?

A Postman Monitor is a scheduled run of a collection in the cloud. It allows you to continuously check the health and performance of your APIs. Monitors run on Postman's cloud servers at specified intervals (e.g., every 5 minutes, every hour).

Use cases:

- Uptime monitoring.
- Performance monitoring (checking response times).
- Functional regression testing.
- Alerting when tests fail.

36. What is Postman Newman?

Newman is a command-line collection runner for Postman. It allows you to run a Postman collection from the terminal. Since it's a command-line tool, it's perfect for integrating Postman tests into CI/CD pipelines (e.g., Jenkins, GitHub Actions).

Key features:

- Runs collections from the command line.
- Can be installed as an npm package (`npm install -g newman`).
- Supports various report formats (HTML, JSON, JUnit XML).
- Can be configured with different environments and data files.

37. Difference between Collection Runner and Newman?

Feature	Collection Runner	Newman
Interface	GUI (Graphical User Interface)	CLI (Command Line Interface)
Execution	Locally within the Postman application	From the terminal/command line

Integration	Manual, interactive testing	Automated, ideal for CI/CD pipelines
Reporting	A visual summary within the app	Generates customizable reports (HTML, JUnit)
Use Case	Interactive runs, debugging, quick validation	Automated testing in a CI/CD environment

38. How do you integrate Newman with Jenkins/GitHub Actions?

General Steps:

1. **Export Collection & Environment:** Export your Postman collection and environment files from the Postman app as JSON files.
2. **Set up CI/CD pipeline:**
   - **Jenkins:** Create a new Jenkins pipeline job.
   - **GitHub Actions:** Create a `.github/workflows/main.yml` file.
3. **Install Newman:** Add a step to install Newman globally.
   - `npm install -g newman`
4. **Run Newman:** Add a step to execute Newman with your files.
   - `newman run your_collection.json -e your_environment.json -r cli,html`
   - This command runs the collection with the environment and generates both a CLI output and an HTML report.
5. **Publish Reports (Optional):** Configure your CI/CD tool to publish the generated HTML or JUnit reports for easy access.

39. How to export/import collections in Postman?

- **Export:**
  1. Right-click on a collection in the left sidebar.
  2. Select "Export".
  3. Choose a file format (Collection v2.1 is recommended).
  4. Save the collection as a JSON file.
- **Import:**
  1. Click the "Import" button at the top-left of Postman.
  2. Select "File" and choose your exported JSON file.
  3. You can also paste the raw JSON code or a URL.
  4. Postman will import the collection and its requests.

40. How to share collections with a team?

- **Team Workspace (Recommended):** The easiest and most efficient way.
  1. Create or switch to a Team Workspace.

    2. Move your collection into the Team Workspace.
    3. Changes are automatically synced in real-time for all team members.
- **Export/Import (Manual):**
    1. Export the collection as a JSON file.
    2. Share the file via email, Slack, or a shared drive.
    3. Team members import the file. This is a manual process and can lead to version control issues.
- **Sharing via Link:**
    1. Right-click on a collection and select "Share".
    2. Choose "Via link" and create a public link.
    3. This is useful for sharing a read-only version of a collection.

---

5. Postman Advanced Features

41. What is Postman Mock Server?

A Postman Mock Server is a feature that allows you to simulate a real API endpoint without having a live backend. You can create mock responses based on saved examples in your collection.

How it works:

1. Create a request and save an "Example" response with the desired body and status code.
2. From the collection's "Mocks" tab, create a new Mock Server.
3. Postman generates a unique URL for the mock server.
4. When you send a request to the mock URL, Postman will return the example response you defined.

Use cases:

- Frontend developers can start building UI even if the backend is not ready.
- API consumers can test their application against a stable API.
- For testing error conditions or edge cases without needing a live server.

42. What is Postman Interceptor?

Postman Interceptor is a browser extension (for Chrome) that allows you to capture network requests and cookies from your web browser and send them to the Postman desktop app.

Use cases:

- Testing APIs that require an active session or cookies set by a web application.
- Debugging front-end applications' API calls.
- Generating Postman requests directly from a browser session.

43. How to capture cookies in Postman?

- **Automatic:** Postman automatically handles and sends cookies that are set by the server in the `Set-Cookie` header. You can view the cookies in the "Cookies" tab in the response pane.
- **Manual:** You can manually set cookies for a request by going to the "Cookies" link below the URL bar. This opens a cookie manager where you can add, edit, or delete cookies for a specific domain.
- **Interceptor:** Use the Postman Interceptor to capture cookies from your browser and use them in your Postman requests.

44. How to clear cookies and cache in Postman?

- **Cookies:** In the Postman application, click the "Cookies" link below the URL bar. A modal will pop up where you can manage and clear cookies for different domains.
- **Cache:** Postman doesn't have a specific "cache" clearing feature in the same way a browser does. However, if you're experiencing issues, you can:
    - Restart the Postman app.
    - Sign out and sign back in to clear local sync data.
    - For persistent issues, you may need to clear Postman's local application data folder (path varies by OS).

45. What is Postman API Network?

The Postman API Network is a public repository of APIs, collections, and workspaces. It's a directory where developers and companies can share their APIs publicly.

Benefits:

- **Discovery:** Users can discover and explore popular APIs.
- **Examples:** You can import collections from the network to quickly get started with an API.
- **Collaboration:** Companies can share official APIs and collections with their user base.

46. How to fork and merge collections in Postman?

- **Forking:** When you fork a collection, you create a copy of it in your own workspace. This is useful for making changes without affecting the original collection.
    1. Go to the collection's overview.
    2. Click the "Fork" button.
    3. Select the destination workspace and give the fork a name.
- **Merging:** Once you've made changes to your forked collection, you can "merge" them back into the original collection.
    1. Go to the forked collection's overview.
    2. Click the "Pull requests" tab.
    3. Click "Create Pull Request".
    4. Review the changes and submit the pull request. The owner of the original collection can then review and merge the changes.

47. What are Postman Workspaces (Personal, Team, Public)?

- **Personal Workspace:** The default workspace. Only you can see and access the collections, environments, and other elements within it.
- **Team Workspace:** A shared space where team members can collaborate in real time. Changes are synced instantly. Used for active team projects.
- **Public Workspace:** A workspace where all collections and environments are publicly visible. This is used for sharing APIs with the public (similar to a public repository on GitHub).

48. What is Postman CLI?

Postman CLI is a command-line interface for Postman that allows you to run collections and other Postman workflows directly from your terminal. It is the modern replacement for Newman and offers enhanced capabilities.

Key features:

- Runs collections.
- Integrates with Postman's cloud platform.
- Supports more advanced features like running against different environments and data files.
- Can generate reports.

49. How does Postman handle GraphQL queries?

1. Open a new request.
2. Set the method to `POST`.
3. Go to the "Body" tab and select the "GraphQL" option.
4. Postman provides two text areas:
    - **Query:** Where you write your GraphQL query or mutation.
    - **GraphQL Variables:** A JSON object where you can define variables for your query.
5. Postman automatically sets the `Content-Type` header to `application/json` and sends the query and variables in the correct JSON format.

50. How to use gRPC APIs in Postman?

1. Create a new gRPC request.
2. Enter the gRPC server address.
3. Postman supports importing `.proto` files to define the services and messages. Click "Import a .proto file".
4. Once imported, Postman will show a list of services and methods. Select the method you want to call.
5. Postman will generate a template for the request payload based on the `.proto` definition.
6. Fill in the payload and click "Invoke".

---

6. Collaboration & Version Control

51. How do you collaborate with a team in Postman?

- **Team Workspaces:** This is the primary method. Team members are invited to a shared workspace where they can create, edit, and share collections, environments, and mock servers.
- **Version Control:** Use Postman's built-in version control or integrate with Git (GitHub/GitLab) to manage changes to collections and environments.
- **Comments:** Use comments on collections, folders, and requests to provide context and feedback.
- **Pull Requests:** Team members can fork a collection, make changes, and then submit a pull request for review.

52. What is the difference between personal and team workspaces?

- **Personal:** Private to a single user. Best for personal projects, testing, and learning. No collaboration features.
- **Team:** Shared among multiple users. All members can view, edit, and contribute. Changes are synced in real-time. This is the standard for professional team-based projects.

53. How does version control work in Postman collections?

Postman has a built-in versioning system for collections.

- **Change Log:** Every time a change is saved to a collection, it creates a new version. You can view the history and compare versions.
- **Forking and Merging:** You can create a fork (a copy) of a collection, make changes, and then create a pull request to merge your changes back into the original. This is similar to the Git workflow.
- **Restoring:** You can revert to a previous version of a collection.

54. How do you fork and pull changes in Postman?

- **Fork:**
    1. On a collection that belongs to a team workspace or is public, click the "Fork" button.
    2. Select your workspace and a name for the fork.
    3. A copy of the collection is created in your workspace.
- **Pull:**
    1. After making changes to your forked collection, go to the collection's overview.
    2. Go to the "Pull requests" tab.
    3. Click "Create Pull Request".
    4. Postman will compare your fork with the parent collection.
    5. Once the pull request is created, the owner of the parent collection can review and merge the changes.

55. How to integrate Postman with GitHub/GitLab?

1. **Integrations:** Go to Integrations from your Postman workspace menu.
2. **Select Git Provider:** Choose GitHub or GitLab.
3. **Configure:** Provide the necessary credentials (API token).
4. **Sync:** Select the Postman collection you want to sync and the Git repository you want to link it to.
5. Postman will sync the collection to your Git repository, allowing you to manage Postman artifacts as code (.json files). You can then use Git for version control, branching, and pull requests.

56. How to sync Postman collections with API repositories?

This is achieved through Postman's Git integrations. By linking a Postman collection to a Git repository, any changes made in the Postman app are automatically pushed to the repository, and changes from the repository are pulled into Postman. This enables a source-of-truth approach where your Postman collections are version-controlled alongside your API code.

57. What is Postman API?

The Postman API is a RESTful API that allows you to programmatically access and manage your Postman data (collections, environments, mocks, etc.).

Use cases:

- **Automate workflows:** Automatically create new collections or environments.
- **Integrate with external tools:** Trigger a Postman monitor run from a CI/CD pipeline.
- **Programmatic management:** Update environment variables or collection data via a script.

Example: Using the API to run a collection from a script.

58. How do you generate API documentation in Postman?

1. Fill out the descriptions for your collection and individual requests and examples. You can use Markdown for rich formatting.
2. On the collection's overview page, click the "Documentation" button.
3. Click "Publish".
4. Postman will generate a public or private web page with your API documentation. The documentation is automatically updated when you save changes in your collection.

59. How to publish APIs in Postman API Network?

1. Ensure your API collection is well-documented with clear descriptions and examples.
2. Go to the collection's overview page.
3. Click "Publish" in the documentation section.
4. Select "Publish to the Postman API Network".
5. Fill in the required information (API name, description, tags, etc.).

6. Once approved by Postman, your API will be listed in the public network for others to discover and use.

60. How to manage API lifecycle in Postman?

Postman provides a comprehensive set of features to manage the entire API lifecycle:

- **Design & Development:** Start with an API schema (OpenAPI, GraphQL). Postman can generate a collection, mock server, and documentation from the schema.
- **Testing:** Use collections and the Collection Runner for functional, regression, and data-driven testing.
- **Publishing & Documentation:** Generate and publish professional API documentation.
- **Monitoring:** Use Postman Monitors for continuous API health checks.
- **CI/CD:** Integrate with Newman or Postman CLI to automate testing in your pipeline.
- **Discovery & Consumption:** Share APIs with the team or publish them to the Postman API Network.

---

7. Postman Performance & Security

61. How to test API performance in Postman?

Postman's built-in tools are more for functional testing, but you can get basic performance metrics.

- **Response Time:** The response time for each request is displayed in the response pane.
- **Postman Monitors:** Monitors can track response times over time, helping you identify performance degradation.
- **Postman CLI (Newman):** You can use Newman to run collections with a specified number of iterations and a delay between requests to simulate load. While not a dedicated load testing tool, it can provide baseline performance data.
- **Dedicated Tools:** For serious performance testing, it's better to use a dedicated tool like JMeter, LoadRunner, or k6, which can generate a high number of concurrent requests.

62. How to set request delays or throttling in Postman?

- **Collection Runner:** In the Collection Runner settings, you can set a "delay" in milliseconds. This will pause for the specified duration between each request in the run. This is useful for simulating a user workflow or preventing rate-limiting during testing.
- **Postman Monitors:** You can also set a delay in a monitor's configuration.

63. How to simulate concurrent requests in Postman?

Postman's primary function is to send requests sequentially. To simulate concurrency, you can:

- **Collection Runner with Multiple Instances:** Run multiple Collection Runner windows simultaneously. This is a manual approach.
- **Newman:** Use Newman to run the same collection multiple times from different command-line instances.
- **Scripts:** Write a test script that uses `pm.sendRequest` to fire off multiple requests in parallel. This is a more advanced technique.

For true, high-volume concurrent load testing, a dedicated load testing tool is a better choice.

64. How to test for rate-limiting in APIs with Postman?

Rate-limiting is a server-side control that limits the number of requests a user can make within a certain timeframe.

1. **Set up:** Create a collection with a single request to the API endpoint you want to test.
2. **Run with Iterations:** Use the Collection Runner.
3. **Configure:** Set the number of iterations to a high number (e.g., 100).
4. **Run:** Execute the collection.
5. **Analyze:** Watch the Postman Console or the Runner results. You should see successful responses initially, followed by `429 Too Many Requests` status codes once the rate limit is exceeded.

65. How to manage secrets securely in Postman (API keys, passwords)?

- **Environment Variables:** Store secrets in environment variables. Do not share the environment file itself.
- **Postman Vault:** The most secure way. Postman Vault is a feature for storing sensitive information.
    - Secrets stored in the Vault are encrypted locally and in the cloud.
    - They are not synced with the collection or environment, so they cannot be accidentally shared.
    - Only team members with access can use the secrets.
    - You reference them in your requests using a special syntax (e.g., `vault:secret-name`).

66. Difference between storing secrets in environment vs Postman Vault?

Feature	Environment Variables	Postman Vault
Security	Not encrypted, can be exported and shared.	Encrypted and stored securely. Not part of the shared collection.

Sharing	Values can be shared with the team via the environment.	Values are tied to a user and cannot be accidentally shared with the team.
Use Case	Secrets for a specific environment that are safe to share (e.g., dev API key).	Highly sensitive secrets (e.g., production passwords, private keys).
Visibility	Visible to all team members in a shared workspace.	Visible only to the user who created it, or specific team members.

67. How to use Postman with HTTPS and SSL certificates?

- **HTTPS:** Postman supports HTTPS by default. Just use `https://` in your request URL.
- **Self-Signed Certificates:** Go to `Postman > Settings > General` and turn off `SSL certificate verification`. This is not recommended for production but can be useful for testing against a local dev server with a self-signed certificate.
- **Client Certificates:** As mentioned in Q18, you can add client-side SSL certificates in the `Settings > Certificates` section to authenticate your requests.

68. How to handle CSRF tokens in Postman?

CSRF (Cross-Site Request Forgery) tokens are used to prevent malicious requests.

1. **Get the Token:** The first request (e.g., a `GET` request to a login page) will return a CSRF token, usually in a cookie or a hidden input field in the response body.
2. **Extract the Token:** In the "Tests" script of the first request, extract the token.
   - From a cookie: `const csrfToken = pm.cookies.get('csrf_token');`
   - From the response body (if it's in a JSON response): `const csrfToken = pm.response.json().csrfToken;`
3. **Store the Token:** Save the token to an environment or collection variable.
   - `pm.environment.set("csrf_token", csrfToken);`
4. **Use the Token:** In the subsequent requests that require the token, add a header with the token's name and its value as `{{csrf_token}}`.

69. How to capture network traffic using Postman Proxy?

1. **Enable Proxy:** In Postman, go to `Settings > Proxy > Interceptor Proxy`.
2. **Set Proxy Port:** Configure the port (e.g., 5555).
3. **Configure Client:** On your client machine or browser, set the proxy settings to use `localhost` and the configured port.
4. **Capture Traffic:** All HTTP/HTTPS traffic from your client will now be routed through the Postman proxy. Postman will capture the requests and display them in the "Proxy

History" in the sidebar. You can then save these requests as new ones in your collection.

70. What are security best practices when using Postman?

- **Do not hardcode secrets:** Never hardcode passwords, API keys, or tokens in your requests. Use environment variables or Postman Vault.
- **Use Postman Vault for sensitive data:** Use the Vault for production credentials or private keys.
- **Review shared collections:** When importing collections from external sources, review the scripts to ensure they don't contain malicious code.
- **Use Team Workspaces for collaboration:** Do not share sensitive environments via email.
- **Manage access:** Use Postman's role-based access control to limit who can modify collections and environments.
- **SSL Verification:** Keep SSL certificate verification enabled unless you have a specific, temporary reason to disable it.

---

8. API Documentation & Reporting

71. How do you create API documentation in Postman?

Postman automates documentation generation from your collections.

1. **Add Descriptions:** For each request, folder, and the collection itself, add a detailed description in the description tab. Use Markdown for formatting.
2. **Save Examples:** For each request, save examples of successful or error responses. This provides a clear example of what the user can expect.
3. **Publish:** Click the "Documentation" button on the collection overview and click "Publish".

72. How to generate documentation automatically from collections?

The process is largely automatic.

- Postman uses the information from your collection (requests, URLs, methods, headers, descriptions, and saved examples) to generate a well-structured and searchable web page.
- Every time you update your collection and save it, the published documentation is automatically updated.

73. How to share API documentation publicly and privately?

- **Private:** You can share the documentation with specific team members or via a private link that requires a Postman account to view.
- **Public:** You can generate a public link that anyone on the internet can access. This is ideal for external-facing APIs.

74. How to generate reports in Postman?

Postman itself doesn't have a "report generation" feature for a single run. However, there are two primary ways to get reports:

- **Collection Runner:** After a run, the Collection Runner provides a summary of tests passed/failed. This can be exported as a JSON file.
- **Newman:** When you run a collection with Newman, you can use reporters to generate various report formats, such as HTML or JUnit XML.
    - `newman run collection.json -r html`
    - This generates a detailed, easy-to-read HTML report.

75. How to customize API documentation in Postman?

- **Markdown:** Use Markdown in your descriptions to add rich formatting, tables, code blocks, and images.
- **Examples:** Create multiple examples for each request to show different response scenarios (e.g., `200 OK`, `404 Not Found`, `500 Server Error`).
- **Themes:** You can choose a theme for your documentation page.
- **Domain:** You can use a custom domain for your documentation URL (available on some Postman plans).

76. What are examples of auto-generated documentation endpoints?

The documentation automatically generated by Postman is not itself an endpoint that you can query. It's a static web page hosted by Postman. The URL would look something like `https://documenter.getpostman.com/view/[team_id]/[collection_id]/[version]`.

77. How to publish Postman collections as APIs?

By linking a Postman collection to an API within Postman's API Builder.

1. Go to the "APIs" tab.
2. Create a new API.
3. Select "Connect to collection" and link your existing collection.
4. This connects the collection to the API definition, allowing you to manage the entire API lifecycle from a single view.

78. How to use Postman's built-in examples feature?

1. Send a request to an endpoint.
2. Once you get a response, click the "Save Response" button on the top-right of the response pane.
3. Select "Save as example".
4. Give the example a name (e.g., `200 OK - User Found`).
5. You can then edit the example's status code, headers, and body.

This feature is crucial for creating realistic mock servers and comprehensive documentation.

79. Difference between examples and mock responses in Postman?

- **Examples:** A saved instance of a request's response. They are stored within the collection itself and are used for documentation and to define what a mock server should return. They are static representations.
- **Mock Responses:** The actual response returned by a Postman Mock Server when a request hits its unique URL. The mock server uses the saved "Examples" to determine which response to send back.

80. How to export API schema in Postman (OpenAPI/Swagger)?

1. In the "APIs" tab, select the API you want to export.
2. Click "Define" and go to the "API Definition" section.
3. Click the "Download" button.
4. Select the desired format (`OpenAPI 3.0`, `Swagger 2.0`, etc.) and the version.
5. Postman will export the API definition as a JSON or YAML file.

---

9. Integration with CI/CD & Tools

81. How to run Postman tests in Jenkins?

1. **Install Node.js:** Make sure Node.js is installed on your Jenkins agent.
2. **Install Newman:** In the Jenkins job's build step, run `npm install -g newman`.
3. **Add Build Step:** Add a "Execute shell" or "Windows batch command" build step.
4. Execute Newman: In the command line, run:
   newman run your_collection.json -e your_environment.json -r junit --reporter-junit-export "newman-report.xml"
5. **Publish JUnit Report:** Add a "Publish JUnit test result report" post-build action and point it to the generated XML file.

82. How to run Postman tests in GitHub Actions?

1. **Create Workflow:** Create a new `.github/workflows/main.yml` file.
2. **Define Job:** Define a job that checks out your code and sets up Node.js.
3. **Install Newman:** Use a `run` step to install Newman.
4. **Run Newman:** Use a `run` step to execute Newman.
5. YAML

```
- name: Run Postman Tests
  run: newman run postman/MyCollection.json --env-var "my_secret=${{ secrets.MY_API_KEY }}"
--reporters cli,junit --reporter-junit-export "newman-report.xml"
```

6.
7.
8. **Publish Report:** Use a GitHub Action like `EnricoMi/publish-unit-test-result-action` to publish the JUnit report.

83. How to run Postman tests in GitLab CI/CD?

The process is very similar to Jenkins and GitHub Actions.

1. **Create .gitlab-ci.yml:** Create this file in your repository.
2. **Define Stage:** Define a test stage.
3. Install Newman: Use a script command to install Newman.
   npm install -g newman
4. Run Newman: Run the Newman command in a script.
   newman run MyCollection.json -e MyEnvironment.json --reporters junit --reporter-junit-export "report.xml"
5. Artifacts: Configure your job to save the report.xml file as an artifact.
   artifacts: reports: - report.xml

84. Difference between Newman and Postman CLI?

- **Newman:** The original, community-supported command-line runner. It's widely used and very stable for running collections and generating reports.
- **Postman CLI:** Postman's newer, official command-line tool. It is designed to be more tightly integrated with the Postman platform. It can also manage and run Postman's APIs and other workflows beyond just collections. Postman is encouraging users to migrate to the Postman CLI for a more seamless experience with their cloud platform.

85. How to integrate Postman with Swagger/OpenAPI?

Postman has first-class support for OpenAPI (formerly Swagger).

- **Import:** You can import an OpenAPI file directly into Postman to automatically generate a collection, environment, and documentation.
- **Export:** You can design your API in Postman and then export it as an OpenAPI file.
- **Sync:** Postman can be configured to sync an API's definition with a Git repository. Any changes to the OpenAPI file in Git will be reflected in Postman, and vice versa.

86. How to export Postman collection to Swagger?

This is not a direct feature. Postman collections are not in the OpenAPI/Swagger format. Instead, you can:

1. **Design an API:** Create an API definition in Postman's API Builder, using OpenAPI/Swagger syntax.
2. **Generate a Collection:** Use the "Generate Collection" feature from the API Builder.
3. **Export the API:** From the API Builder, you can then export the API definition as a JSON or YAML file in OpenAPI format.

87. How to import Swagger/OpenAPI specs into Postman?

1. Click the "Import" button.
2. Select the "Link" tab and paste the URL of the OpenAPI spec, or select the "File" tab and upload the spec file.
3. Postman will parse the spec and give you options to:
   - Generate a Postman Collection.
   - Add a Mock Server.

    - Add a Postman Monitor.
    - Generate API documentation.
4. Choose your options and click "Import".

88. How to run Postman tests in Docker containers?

1. **Pull Newman image:** Use a command like `docker pull postman/newman_alpine33`.
2. **Mount directory:** Mount your local directory containing the collection and environment files into the Docker container.
3. **Run Newman:** Execute the container with the `docker run` command.
4. Bash

```
docker run --network=host -v "$(pwd):/etc/newman" postman/newman_alpine33 run
"mycollection.json" -e "myenv.json"
```

5.
    - `--network=host` is important if your API is running on `localhost`.
    - `-v` mounts the current directory to `/etc/newman` inside the container.

89. How to schedule Postman monitors for automated testing?

1. From the Postman app, go to your collection's overview and click the "Monitors" tab.
2. Click "Create a monitor".
3. Select the collection and environment to use.
4. Choose the frequency and time for the monitor to run (e.g., every 15 minutes, daily at 3 AM).
5. Set up any email alerts for test failures.
6. Click "Create Monitor". Postman's cloud service will now run your collection automatically on the schedule you defined.

90. How to generate HTML/JUnit reports from Postman?

This is done with Newman, not the Postman desktop app.

- HTML: Use the newman-reporter-html reporter.
  npm install -g newman-reporter-html
  newman run mycollection.json -r html --reporter-html-export "report.html"
- JUnit: Use the built-in junit reporter.
  newman run mycollection.json -r junit --reporter-junit-export "report.xml"
- Multiple Reporters: You can use multiple reporters in a single run.
  newman run mycollection.json -r cli,html,junit

---

10. Real-World & Best Practices

91. How do you test REST APIs with Postman?

Postman is built for testing REST APIs.

- **GET:** Test with query parameters and verify response codes and bodies.
- **POST/PUT/PATCH:** Use the "Body" tab with JSON, form-data, etc. and verify the `201 Created` or `200 OK` status and the response body.
- **DELETE:** Verify the `204 No Content` or `200 OK` status.
- **Tests:** Use Postman's tests to assert status codes, response times, headers, and body content.
- **Workflows:** Use variables and the Collection Runner to chain requests together to test multi-step workflows (e.g., login, then get user data, then delete the user).

92. How do you test GraphQL APIs with Postman?

- **Request Method:** Use `POST` as all GraphQL queries and mutations are sent via a POST request.
- **Body Type:** Select "GraphQL" in the "Body" tab.
- **Query & Variables:** Write your query in the `Query` pane and your variables (if any) in the `GraphQL Variables` pane.
- **Headers:** Set the `Content-Type` header to `application/json`. Postman handles this automatically.
- **Tests:** Use the same `pm.response.json()` and `pm.expect()` methods to validate the GraphQL response body.

93. How do you test SOAP APIs in Postman?

- **Request Method:** Use `POST`.
- **URL:** The SOAP endpoint URL.
- **Headers:**
  - `Content-Type`: `text/xml` or `application/soap+xml`.
  - `SOAPAction`: The name of the SOAP method you are calling.
- **Body:** Select the "raw" body type and paste your XML-formatted SOAP envelope.
- **Tests:** Use `xml2Json()` to parse the XML response and then write assertions against the parsed JSON object.

94. How to handle dynamic tokens in Postman requests?

Dynamic tokens (e.g., access tokens, session IDs) are a common part of API security.

1. **Authentication Request:** Create a request (e.g., a `POST` to a login endpoint) that returns the token in the response body or header.
2. **Script to Extract & Store:** In the "Tests" tab of the authentication request, write a script to extract the token and store it in an environment variable.
3. JavaScript

```javascript
const responseData = pm.response.json();
pm.environment.set("accessToken", responseData.token);
```

4.
5.

6. **Subsequent Requests:** In all other requests that need the token, go to the "Authorization" tab, select "Bearer Token", and set the token value to `{{accessToken}}`. This automatically adds the `Authorization: Bearer <token>` header.

95. How to organize requests in Postman collections for large projects?

- **Folders:** Use folders to group related requests. For example, a "User API" collection could have folders for "Users", "Products", and "Orders".
- **Naming Conventions:** Use clear, consistent naming for collections, folders, and requests (e.g., `POST Create User`, `GET All Products`).
- **Variables:** Use environment and collection variables to avoid hardcoding values and to make requests reusable across different environments.
- **Documentation:** Use the description fields for context, examples, and markdown.

96. What are best practices for naming requests and collections?

- **Collections:** Use a descriptive name that reflects the API or project (`E-commerce API`, `User Management Service`).
- **Folders:** Group requests by resource or functionality (`/users`, `/products`, `Authentication`).
- **Requests:** Use a consistent naming convention that includes the HTTP method and the resource being acted on (e.g., `GET All Users`, `POST Create a New User`, `PUT Update User by ID`).
- **Variables:** Use clear, lowercase names with underscores (`api_key`, `base_url`).

97. How do you debug API failures in Postman?

- **Postman Console:** The first place to look. It shows the full request and response headers, body, cookies, and network information.
- **Status Codes:** Check the HTTP status code (e.g., `404 Not Found`, `500 Internal Server Error`). This gives you a high-level idea of the problem.
- **Response Body:** Check the response body for a detailed error message from the server.
- **Headers:** Look at the request and response headers for clues (e.g., `Content-Type`, `Authorization`).
- **Pre-request & Test Scripts:** Use `console.log()` in your scripts to print variable values and debug the flow.
- **URL & Parameters:** Double-check the URL, path variables, and query parameters for typos.

98. How to simulate error responses in Postman?

- **Mock Server:** The best way. Create a request with an example for an error response (e.g., status `400 Bad Request` with a JSON body explaining the error). The mock server will return this when you hit its URL.
- **Manual:** Simply change the request to intentionally cause an error (e.g., use a non-existent URL, an invalid API key, or malformed JSON).

99. How to set up Postman for microservices testing?

1. **Workspaces:** Create a team workspace for your microservices project.
2. **Collections per Service:** Create a separate collection for each microservice (e.g., User Service API, Payment Service API).
3. **Shared Environments:** Create a shared environment that contains common variables like base URLs for different services (e.g., user_service_url, payment_service_url).
4. **Workflows:** Use the Collection Runner to test complex workflows that span multiple services (e.g., creating a user in Service A, then a payment in Service B for that user).
5. **Monitors:** Set up monitors for critical microservices to ensure they are always up and running.

100. What are best practices for using Postman in a team environment?

- **Use Team Workspaces:** All collaboration should happen here.
- **Share Collections and Environments:** Share all project-related collections and environments with the team.
- **Use Version Control:** Fork collections, create pull requests, and integrate with Git for robust version control.
- **Document Everything:** Use descriptions for collections, folders, and requests. This makes it easier for new team members to get up to speed.
- **Avoid Hardcoding:** Use variables for all URLs, keys, and dynamic data.
- **Write Automated Tests:** Add tests to every request to ensure API quality and prevent regressions.
- **Leverage Postman API:** For advanced teams, use the Postman API to automate workflows like updating collections or triggering runs from external systems.
- **Maintain Clear Naming:** Enforce a consistent naming convention for requests and variables.