Author: Madhurima Rawat

Setting Up a Virtual Machine in a Cloud Environment

To set up a virtual machine using LocalStack, users can simulate a cloud environment locally. While LocalStack primarily emulates AWS services, it allows for the replication of functionalities such as managing EC2 instances within a local environment.

LocalStack

LocalStack is a local AWS cloud stack that provides a seamless development and testing environment. Here's a concise breakdown of its features and benefits:

- **AWS Service Simulation**: Mimics AWS services like S3, EC2, Lambda, DynamoDB, and more.
- **Local Development**: Enables developers to test and build cloud applications on their local machine without needing internet connectivity.
- **Cost-Free Environment**: No AWS account or associated costs are required for running services locally.
- **API Compatibility**: Provides APIs that replicate AWS functionalities, allowing easy integration with existing tools and workflows.
- **Isolated Testing**: Facilitates isolated development and testing before deploying to the actual AWS cloud.
- **Automation and CI/CD**: Perfect for automating workflows, testing infrastructure, and integrating with continuous integration/continuous deployment pipelines.

Introduction to LocalStack and S3

What is LocalStack?

LocalStack is a fully functional local AWS cloud stack that enables you to develop and test your cloud and Serverless apps offline. It spins up an easy-to-use testing environment on your local machine that has the same APIs and works the same way as the real AWS cloud environment.

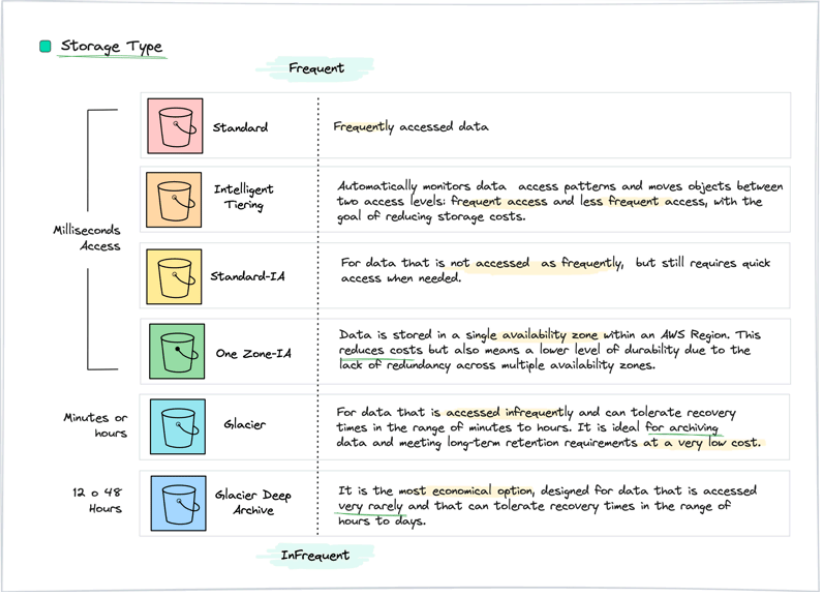
What is S3?

Amazon S3 (Simple Storage Service) is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is designed to make web-scale computing easier for developers. S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.

What is AWS S3?

Use Cases for S3

S3 Storage Types



Prerequisites

Docker Setup

The error may also indicate that **Docker** is not installed or configured correctly on the system. Since LocalStack relies on Docker to simulate AWS services locally, follow these steps to resolve the issue:

1. Install Docker on the System

- Download and install Docker Desktop for Windows from the official Docker website.
- During installation, ensure the following options are selected:
  - "Install required Windows components for WSL 2" (if using WSL 2).
  - Restart the machine if prompted to complete the installation.

2. Verify Docker Installation

- Open a command prompt or PowerShell and run the following command to check if Docker is installed:

```
docker --version
```

- If Docker is successfully installed, a version number will appear, for example: `Docker version 24.0.2, build abcdef`.

3. Start Docker Desktop

- Launch Docker Desktop and wait until it indicates that "Docker is running."

4. Enable WSL 2 Integration (If Required)

- If using Windows Subsystem for Linux (WSL), enable WSL 2 by following these steps:
  i. Open Docker Desktop settings.
  ii. Navigate to **General** and check the option "Use the WSL 2-based engine."
  iii. Apply the changes and restart Docker Desktop.

5. Retry Running LocalStack

- Once Docker is running, execute the following command in the terminal:

```
localstack start
```

- LocalStack should now start successfully.

LocalStack Installation

1. **Install LocalStack**:
   - LocalStack can be installed via pip:

     ```
     pip install localstack
     ```

   - Alternatively, it can be pulled from Docker:

     ```
     docker pull localstack/localstack
     ```

2. **Start LocalStack**:
   - For running LocalStack with Docker:

     ```
     docker run -d -p 4566:4566 -p 4510-4559:4510-4559 localstack/localstack
     ```

   - If LocalStack is installed locally:

     ```
     localstack start
     ```

3. **Install AWS CLI**:
   - If the AWS CLI is not already installed:

     ```
     pip install awscli
     ```

4. **Configure AWS CLI for LocalStack**:
    - The AWS CLI can be configured using default credentials, as LocalStack does not require actual AWS credentials:

      ```
      aws configure
      ```

        - Access Key ID: `test`
        - Secret Access Key: `test`
        - Region: `us-east-1`
    - To configure the AWS CLI to connect to LocalStack:

      ```
      export AWS_ENDPOINT_URL=http://localhost:4566
      ```

    During the `aws configure` setup, the AWS CLI will prompt for configuration details, including the **output format**, which controls the display style of results from AWS CLI commands.

Common Output Formats:

1. **JSON** (default): A structured data format, ideal for automation and scripting.

   Example:

   ```
   {
     "Key": "Value",
     "AnotherKey": "AnotherValue"
   }
   ```

2. **Table**: A human-readable table format, ideal for quick reviews.

   Example:

   ```
   +------------+----------------+
   |   Key      |    Value       |
   +------------+----------------+
   | Key1       | Value1         |
   | Key2       | Value2         |
   +------------+----------------+
   ```

3. **Text**: A simple plain-text format, suitable for scripts or command-line parsing.

   Example:

```
Key1 Value1
Key2 Value2
```

Choosing the Output Format

During the `aws configure` setup, users can:

- Accept the **default** format by pressing **Enter** to select `json`.
- Alternatively, specify a different format by typing one of the following:
  - `json`
  - `table`
  - `text`

Example:

```
Default output format [json]: table
```

How to Change the Output Format Later

If a user wishes to change the output format after the initial configuration, they can:

1. Manually edit the AWS CLI configuration file:
   - Open `~/.aws/config` (on Linux/macOS) or `%USERPROFILE%\.aws\config` (on Windows).
   - Modify the `output` setting:

     ```
     [default]
     output = text
     ```

2. Use the `--output` flag when running commands:

   ```
   aws s3 ls --output table
   ```

Resolving the `Failed to connect to localhost port 4566` Error

When encountering the error `Failed to connect to localhost port 4566`, it typically indicates that no service is running on that port. To resolve this issue:

1. **Check if LocalStack is Running:**

- To verify if LocalStack is running, users can check the container or process status by executing the following command:

```
docker ps
```

2. **Verify Port Mapping and LocalStack Startup:**

   - If using Docker, ensure the correct ports are mapped. If running LocalStack locally, users should confirm its successful startup by checking the logs or re-running the following command:

```
localstack start
```

Limitations in LocalStack

- LocalStack provides a basic simulation of AWS EC2 but does not support actual VM provisioning.
- For working with real VMs locally, users should consider using hypervisors such as **VirtualBox**, **VMware**, or combining LocalStack with these tools for a hybrid setup.

This setup is ideal for students looking to interact with a simulated cloud environment, allowing them to practice API calls and configurations without incurring any cloud service costs.

Error Encountered:

```
C:\Windows\System32>pip install localstack
WARNING: Ignoring invalid distribution
-treamlit (c:\users\rawat\appdata\local\programs\python\
python310\lib\
site-packages)
Collecting localstack
  Using cached localstack-4.0.3.tar.gz (5.7 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting localstack-core (from localstack)
  Using cached localstack_core-4.0.3-py3-none-any.whl.metadata (5.4 kB)
Collecting localstack-ext==4.0.3 (from localstack)
  Using cached localstack_ext-4.0.3.tar.gz (6.2 MB)
ERROR: Could not install packages due to an OSError:
[Errno 2] No such file or directory:
'C:\\Users\\rawat\\AppData\\Local\\Temp\\pip-install-
q450br39\\localstack-ext_771d2730449b461b8323becea
3af2ce5\\localstack/pro/core/services/lambda_/invocation/
endpoint_injection/java/SdkV2DisableCertificateValidation/
src/main/java/cloud/localstack/HttpClientTransformer.java'
```

```
HINT: This error might have occurred since this system
does not have Windows Long Path support enabled.
You can find information on how to enable this at
https://pip.pypa.io/warnings/enable-long-paths
```

The error indicates that **Windows Long Path Support** is not enabled, which is necessary because some files in the `localstack` installation exceed the default path length limit of 260 characters on Windows.

Here are the steps to resolve this:

Steps to Enable Windows Long Path Support

1. **Enable Long Paths in the Windows Registry:**
   - Open the **Run dialog** (`Win + R`), type `regedit` , and press Enter.
   - Navigate to the following key:

     ```
     HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem
     ```

   - Look for the `LongPathsEnabled` entry. If it does not exist, create a new `DWORD` (32-bit) value named `LongPathsEnabled` .
   - Set its value to `1` .
2. **Enable Long Paths Using Group Policy (for Pro, Education, or Enterprise Editions):**
   - Open the **Run dialog** (`Win + R`), type `gpedit.msc` , and press Enter.
   - Navigate to:

     ```
     Computer Configuration > Administrative Templates > System > Filesystem
     ```

   - Double-click on **Enable Win32 long paths**.
   - Set it to **Enabled**, then click **OK**.
3. **Restart the System:**
   - After making the necessary changes, users should restart their computer to apply them.

Retry Installing `localstack`

Once Long Path Support is enabled, users can retry the installation with the following command:

```
pip install localstack
```

Additional Tips

- **Upgrade Pip**: It's recommended to ensure that the latest version of pip is being used:

```
python -m pip install --upgrade pip
```

- **Use a Virtual Environment**: To avoid making system-wide changes, users can create a virtual environment:

```
python -m venv venv
venv\Scripts\activate
pip install localstack
```

- **Check Python Compatibility**: Ensure that the Python version is compatible with the `localstack` version being installed.

Issue with the `export` Command on Windows

The `export` command is used in Unix-based systems like Linux and macOS to set environment variables. However, it is not recognized in Windows Command Prompt (CMD). In CMD, the `set` command should be used instead to set environment variables.

To resolve the issue, replace the `export` command with the `set` command in CMD:

```
set AWS_ENDPOINT_URL=http://localhost:4566
```

Notes:

1. If using PowerShell, the environment variable can be set with the following command:

```
$env:AWS_ENDPOINT_URL = "http://localhost:4566"
```

2. For a persistent environment variable across sessions, use the `setx` command to set it permanently:

```
setx AWS_ENDPOINT_URL "http://localhost:4566"
```

   Caution: The variable set with `setx` will only be available in new CMD or PowerShell sessions.

To verify the environment variable is set correctly, run the following:

```
echo %AWS_ENDPOINT_URL%
```

This should display:

```
http://localhost:4566
```

1. Ensure the Service is Running

- If LocalStack or another AWS emulator is being used, it's essential to confirm that it has started successfully.
  - For **LocalStack**:

    ```
    localstack start
    ```

  - For **Docker** (if LocalStack is running via Docker):

    ```
    docker run -p 4566:4566 -d localstack/localstack
    ```

    After executing this, the following command can be used to confirm the container is running:

    ```
    docker ps
    ```

2. Verify Port Availability

- To check if another service is using port `4566`, the following command should be run:

  ```
  netstat -ano | find "4566"
  ```

  - If no output is returned, it indicates the port is free.
  - If output is returned, it's necessary to identify which process is using the port by executing:

    ```
    tasklist /FI "PID eq <PID>"
    ```

    Replace `<PID>` with the process ID found from the `netstat` output.

3. Check Firewall or Antivirus Settings

- It should be ensured that no firewall or antivirus software is blocking connections to port `4566`.

- Temporarily disabling the firewall or antivirus can help in testing.

4. Confirm localhost Works

- To check if `localhost` is working correctly on the system, the following command can be used:

  ```
  ping localhost
  ```

- If there is no response, replacing `localhost` with `127.0.0.1` is recommended:

  ```
  set AWS_ENDPOINT_URL=http://127.0.0.1:4566
  ```

5. Validate `curl` Setup

- It is crucial to confirm that `curl` is installed and functioning correctly by running:

  ```
  curl --version
  ```

- If `curl` is not found, it can be downloaded from curl's official website or enabled through Windows Features.

6. Restart System or Docker

- A system restart and/or restarting Docker could help resolve any lingering issues.
- After restarting, the desired service (e.g., LocalStack) should be started again.

7. Manually Test Another Port

- To manually test connectivity to `localhost`, a simple web server can be run using the following command:

  ```
  python -m http.server 4566
  ```

  Then, the server can be accessed with:

  ```
  curl http://localhost:4566
  ```

  - If this works, it suggests the issue lies with the service running on port `4566`.
  - If it doesn't work, there may be a network configuration issue.

Troubleshooting Docker Desktop Sign-Out Issues

If Docker Desktop keeps signing out automatically, several common causes and fixes should be considered:

1. Enable Trusted Root Certificates

- Docker Desktop may fail to authenticate if the required root certificates are missing or untrusted.
- **Solution**:
  i. The system's date and time should be confirmed to be correct.
  ii. The certificates can be updated by running the following command:

```
certutil -generateSSTFromWU roots.sst
```

  iii. The `roots.sst` file can then be double-clicked to install the certificates.

2. Clear Docker Credentials

- Cached credentials might be causing the sign-out issues.
- **Solution**:
  i. Navigate to the Docker configuration folder:

```
C:\Users\<YourUsername>\.docker
```

  ii. The `config.json` file should be deleted.
  iii. Docker Desktop should be restarted, and the user should sign in again.

If Docker is not running, it should automatically troubleshoot and restart itself.

LocalStack Container Image Download Process

The following output indicates that **LocalStack is attempting to pull the container image** **`localstack/localstack`** to run in Docker mode, which suggests that the setup is working up to this point. Here's what's happening:

```
C:\Users\rawat>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
Pulling container image localstack/localstack
```

What's Happening?

1. **LocalStack CLI Detected Docker Mode**:
    - `[11:20:31] starting LocalStack in Docker mode` confirms that LocalStack is trying to run using Docker.
2. **Container Image Not Found Locally**:
    - `[11:25:13] container image not found on host` indicates that Docker is downloading the `localstack/localstack` image from Docker Hub since it's not found locally.
3. **Pulling the Container Image**:
    - `⠸ Pulling container image localstack/localstack` confirms that the image download process is ongoing.

What to Expect Next

- If the internet connection is stable, Docker should complete the download in a few minutes.
- After the image is downloaded, LocalStack will initialize, and the services (e.g., S3, DynamoDB) will start. This will be confirmed by further logs.

How to Verify It's Working

1. After the image download is complete, the user can check if the container is running with the following command:

    ```
    docker ps
    ```

    The container should be listed with the name `localstack`.
2. To verify that LocalStack is running, the endpoint should be accessed:

    ```
    curl http://localhost:4566
    ```

    A response should be returned, confirming that LocalStack is active.

If Pulling Takes Too Long

- Check the **internet speed** and Docker's ability to pull images:

```
docker pull localstack/localstack
```

- If the pull succeeds, proceed to rerun the LocalStack command:

```
python "C:\Users\rawat\AppData\Local\Programs\Python\Python310
\Scripts\\localstack" start
```

After successfully pulling the image, run the following to verify:

```
C:\Users\rawat>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
localstack/localstack latest b686f3948f42 4 hours ago 1.18GB
hello-world latest 74cc54e27dc4 31 hours ago 10.1kB
```

If issues persist after pulling `hello-world`, follow these steps:

1. Start Docker Desktop.
2. Then, execute:

```
localstack start
```

Steps to Set Up a Virtual Machine in LocalStack

1. **Simulate EC2 Service**:

   LocalStack emulates a limited set of EC2 functionalities. The goal is to create mock resources like key pairs, security groups, and instances.

2. **Create a Key Pair**:

   Use the AWS CLI to generate a key pair:

```
aws ec2 create-key-pair --key-name local-key --endpoint-
url=%AWS_ENDPOINT_URL%
```

   The output will include the generated public/private key pair.

3. **Create a Security Group**:

   Create a security group to define network rules:

```
aws ec2 create-security-group --group-name
local-sg --description "Local Security Group"
--endpoint-url=%AWS_ENDPOINT_URL%
```

4. **Run an Instance**:

   Launch a mock EC2 instance using:

   ```
   aws ec2 run-instances
     --image-id ami-12345678
     --count 1
     --instance-type t2.micro
     --key-name local-key
     --security-group-ids sg-12345678
     --endpoint-url=%AWS_ENDPOINT_URL%
   ```

   Replace `ami-12345678` with an example AMI ID that is recognized by LocalStack.

   Example output:

   - Security Group ID: sg-2cd410ccd533c7f8b
   - Image ID: ami-a2678d778fc6

Command:

```
aws ec2 run-instances --image-id ami-a2678d778fc6 --count 1
--instance-type t2.micro --key-name local-key
--security-group-ids sg-2cd410ccd533c7f8b
--endpoint-url=%AWS_ENDPOINT_URL%
```

Explanation of Parameters

1. **`aws ec2 run-instances`**
   - This is the AWS CLI command to launch one or more Amazon EC2 instances.
2. **`--image-id ami-a2678d778fc6`**
   - **What it is:** The unique ID of the Amazon Machine Image (AMI) you want to use.
   - **Why it matters:** An AMI is like a template that defines what the instance will look like, including its operating system, software, and configuration.
   - **Example:** If you want to run an Ubuntu server, you select an AMI ID for an Ubuntu image.
3. **`--count 1`**
   - **What it is:** The number of EC2 instances to create.
   - **Why it matters:** You can launch multiple instances at once. In this case, `1` means you're creating a single instance.

4. **`--instance-type t2.micro`**
    - **What it is:** The type of EC2 instance to launch.
    - **Why it matters:** Instance types determine the amount of CPU, memory, and networking performance available.
    - **Example:** `t2.micro` is a small, low-cost instance type suitable for lightweight tasks or free-tier usage.
5. **`--key-name local-key`**
    - **What it is:** The name of the key pair to use for SSH access to your instance.
    - **Why it matters:** A key pair ensures secure access to the instance. You'll need the private key file associated with this name to log in.
6. **`--security-group-ids sg-2cd410ccd533c7f8b`**
    - **What it is:** The ID of the security group to associate with the instance.
    - **Why it matters:** Security groups act as firewalls for your instance, controlling which traffic is allowed to enter or leave.
    - **Example:** You might configure it to allow SSH (port 22) or HTTP (port 80) traffic.
7. **`--endpoint-url=%AWS_ENDPOINT_URL%`**
    - **What it is:** Specifies a custom endpoint URL for your AWS service.
    - **Why it matters:** This is useful when working with a local AWS emulator (e.g., LocalStack) or custom AWS environments.
    - **Example:** `%AWS_ENDPOINT_URL%` expands to the URL you set earlier, such as `http://localhost:4566`.

What Happens When You Run This Command?

- The AWS CLI will create a single EC2 instance based on the AMI (`ami-a2678d778fc6`).
- The instance will be of type `t2.micro` , suitable for low-resource tasks.
- The instance will use the `local-key` key pair for SSH access.
- The security group (`sg-2cd410ccd533c7f8b`) will control the traffic to and from the instance.
- The endpoint URL will be used to connect to the specified AWS service.

Example Use Case

- We want to set up a small server (like an Ubuntu instance) locally for testing, using our custom AWS endpoint URL (`http://localhost:4566`) with specific security and access configurations.

5. **List Instances**:
    - Verify the instance creation:

      ```
      aws ec2 describe-instances --endpoint-url=%AWS_ENDPOINT_URL%
      ```

Creating an S3 Bucket in LocalStack

Commands Breakdown:

1. **List Existing Buckets:**

   ```
   C:\Users\rawat>aws --endpoint-url=
   http://localhost:4566 --region us-east-1 s3 ls
   ```

    - **Purpose:** This command is used to list the existing S3 buckets in your LocalStack environment, running at `localhost:4566`.
    - **Output:** No output is shown initially, meaning there are no buckets created at that point.

2. **Create a New S3 Bucket:**

   ```
   C:\Users\rawat>aws --endpoint-url=http://localhost:4566
   --region us-east-1 s3 mb s3://my-test-bucket
   ```

    - **Purpose:** This command creates a new S3 bucket named `my-test-bucket` in the LocalStack environment.
    - **Output:** `make_bucket: my-test-bucket` — This confirms the successful creation of the `my-test-bucket`.

3. **List Buckets Again:**

   ```
   C:\Users\rawat>aws --endpoint-url=http://localhost:4566
   --region us-east-1 s3 ls
   ```

    - **Purpose:** This command lists the existing S3 buckets again, this time after creating the new bucket.
    - **Output:**

      ```
      2025-01-23 13:09:41 my-test-bucket
      ```

      — This confirms that the `my-test-bucket` is now listed and was successfully created.

Detailed Explanation:

1. **LocalStack's Purpose:**
    - LocalStack emulates AWS services locally, providing an environment for developers to simulate AWS without needing an actual AWS account or incurring costs. It mimics APIs for various services such as S3, EC2, Lambda, etc., making it a useful tool for development and testing.
2. **CLI Commands and Output:**
    - The AWS CLI interacts directly with LocalStack through the endpoint `localhost:4566`, which is where the LocalStack service is running. Every time you run commands like `aws s3 mb` or `aws s3 ls`, LocalStack processes these commands as if they were sent to the actual AWS cloud.
    - After running the `aws s3 mb` command, LocalStack creates the `my-test-bucket` bucket in its emulated environment. The successful creation is confirmed by the output: `make_bucket: my-test-bucket`.
    - When you list the buckets with `aws s3 ls`, it shows the newly created bucket as `my-test-bucket`, confirming the operation was successful.
3. **Why the `localhost:4566` Page Remains Empty:**
    - `localhost:4566` is not intended to serve a graphical user interface (GUI) like the AWS Management Console. Instead, it acts as an endpoint for API requests from tools like the AWS CLI or SDKs.
    - LocalStack emulates AWS services by providing HTTP-based API endpoints. These APIs handle requests and respond accordingly, but LocalStack does not serve a web-based dashboard. Therefore, accessing `localhost:4566` via a browser will result in a blank or empty page, which is completely normal.
    - To interact with LocalStack, users should rely on the AWS CLI, SDKs, or other infrastructure management tools (like Terraform). These tools send API calls to LocalStack, which then processes and responds with the requested information or performs actions such as creating S3 buckets.
4. **Why the S3 Bucket Is Created:**
    - The `aws s3 mb` command in LocalStack simulates the creation of an S3 bucket, just as it would in AWS. In your case, the bucket `my-test-bucket` was created locally within LocalStack.
    - Even though LocalStack is emulating AWS S3 locally, the functionality is designed to be very similar to the actual AWS S3 service. Therefore, the `aws s3 ls` command shows the bucket because LocalStack has processed the request successfully and maintains an internal record of the bucket.

5. **Key Takeaways:**
    - **LocalStack is functional:** The successful creation and listing of the `my-test-bucket` confirm that LocalStack is operating correctly and the AWS CLI is interacting with it as expected.
    - **No graphical interface is provided at `localhost:4566`:** LocalStack is designed to work through API calls, not through a web dashboard.
    - **Use CLI for interaction:** For operations like creating buckets, listing them, or interacting with other AWS services, the AWS CLI or other tools should be used to send requests to LocalStack's API endpoints.

Additional Information:

- **LocalStack's API works correctly:** The creation and listing of the S3 bucket (`my-test-bucket`) show that LocalStack's simulation of AWS services is functioning as expected.
- **No GUI at `localhost:4566`:** It's important to understand that LocalStack is API-driven, so there is no web interface at `localhost:4566` . Users should interact with LocalStack via command-line tools or SDKs to simulate AWS service operations locally.

Useful Resources for Working with LocalStack

1. Articles on LocalStack and AWS S3

- **Learning AWS S3 on Localhost: Best Practices with Boto3 and LocalStack**
  A practical guide to learning and using AWS S3 locally with LocalStack, focusing on best practices and integration with Python's Boto3 library.
  Read on Dev.to

- **A Technical Introduction to LocalStack**
  A comprehensive introduction to LocalStack, its purpose, and how to simulate AWS services locally for testing and development.
  Read on Dev.to

2. Official Docker Documentation

- **Docker Guide for LocalStack**
  Learn how to use Docker to run LocalStack efficiently, with step-by-step guidance for setup and configuration.
  Explore Docker Documentation

3. LocalStack Tutorials

- **LocalStack Official Tutorials**
  Get started with LocalStack through hands-on tutorials, covering everything from setup to

practical usage scenarios.
Visit LocalStack Tutorials

4. Blog Post on LocalStack Setup and Usage

- **Getting Started with LocalStack: Overview, Setup, and Practical Usage Guide**
  A detailed guide on LocalStack's setup, configuration, and real-world use cases for developers.
  Read on Ruan.dev