10th International Scientific Conference
Technics, Informatics, and Education – TIE 2024
20-22 September 2024

Session: *Educational Technology*
Professional paper
DOI: 10.46793/TIE24.185B

LocalStack: A Practical Approach to Teaching Cloud Computing

Žarko Bogićević[1] [0000-0002-0831-1300] and Marjan Milošević[2] [0000-0003-4730-1292]
[1] globaldatanet GmbH, Hamburg, Germany
[2] University of Kragujevac, Faculty of Technical Sciences, Čačak, Serbia
* zarko1993@hotmail.com

Abstract: *Cloud computing has become the core of modern software development and deployment. LocalStack provides an efficient, cost-effective platform for developers and students to simulate complex cloud architectures. In this paper, LocalStack has been explored as a cloud service emulator as it offers local deployment of cloud-based services that would typically require Amazon Web Services (AWS) and accounts. The research has focused on the deployment of LocalStack, its capability of emulating AWS cloud services, the educational benefits for students learning cloud technologies and Infrastructure-as-Code practices using Terraform and LocalStack with the goal of demonstrating how it reduces barriers to cloud education with local development, debugging and testing.*

Keywords: *cloud computing, localstack, simulation, hands-on approach*

1. INTRODUCTION

Cloud computing has become an omnipresent technology, running applications of all kinds, especially with the rapid development of artificial intelligence, data science, and the Internet of Things. The Association for Computing Machinery (ACM) reserved an important place for cloud computing in their curriculum recommendations [1] and there are even complete master programs, with majors in this very field.

While for many other computer-related disciplines the traditional computer lab equipment or even students' laptops can be sufficient, cloud computing requires specific resources, responding to the essential features of the cloud computing model: on-demand access, available pool of resources, rapid provision [2].

The precise definition of resources required for the cloud computing course depends on the course objectives. On the other hand, there is also the other way around: as the subject is heavily reliant on expensive technology, the learning objectives are influenced by the institution's capacity to support various features requiring high computing power. Establishing a realistic testbed is a serious challenge and educators have taken various approaches regarding this matter. The most popular way is to use a well-known provider, such as AWS, Google Cloud, IBM Cloud, etc. These providers offer free tier subscriptions, allowing the implementation of less demanding cloud scenarios. Some also have education programs, such as AWS Academy (former AWSEducate) [3], providing both access to real cloud (with $100 credit) and learning resources. Inclining towards one particular provider is a two-sided medal: it provides specific knowledge and hands-on experience, but (a) the resources need to be used very carefully not to surpass the limit and (b) sticking to one provider leads to vendor lock-in, which arguably may not be a good pedagogy practice. The challenges teachers may face, including the provider choice, are described in [4]. To provide students with a universal way of interacting with cloud resources (e.g. virtual machines), researchers and practitioners came up with solutions running aggregating multi-cloud access. Such examples are CloudMesh [5] and EasyCloud [6].

Alternatives to using commercial providers are considered in [7], and include OpenStackc [8] and Chameleon. OpenStack enables composing its own cloud. Chameleon provides free access to cloud resources distributed by the University of Chicago and Texas Advanced Computing Center and runs on OpenStack.

Composing fully private hardware infrastructure providing features needed for teaching cloud computing and executing demanding parallel processing is also an option, advocated in [9]. Although the authors' calculation of expenses is in favor of the private, physical cluster scenario, there are certain circumstances, that might come up as very problematic and contribute to the increased total cost – such as administration of such cluster, additional expenses (space, cooling systems, etc.).

Lastly, there are cloud simulation software, such as CloudSim [10]. This simulator framework provides rich possibilities for simulating and testing cloud

scalability, energy efficiency, load-balancing algorithms, and much more. However, it requires knowledge of Application Programming Interfaces (API) and the learning curve can be tedious to follow. Other simulators are mostly based on CloudSim. A comprehensive preview is given in [11].

In this paper, we discussed LocalStack, a local cloud emulator, and its prospects for use in teaching cloud computing.

2. LOCALSTACK ARCHITECTURE

LocalStack relies on Docker to provide isolated and reproducible environments for its execution. Each AWS service is typically run as a separate process within a Docker container, allowing LocalStack to emulate several services simultaneously. The architecture can be seen in Figure 1.

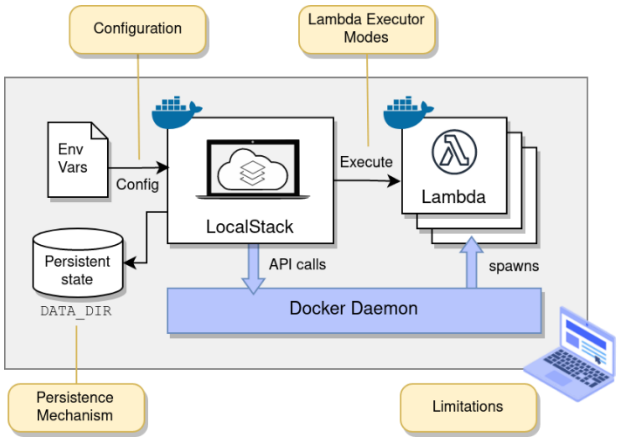

Figure 1. *LocalStack Architecture [12]*

LocalStack has several key components [12, 13]:

- Edge service – this serves as the main entry point for all requests to LocalStack. It listens to the default port (4566) and routes requests to appropriate services. It ensures request handling, routing, and forwarding to backend services.
- Service Emulators – these components are used to emulate specific AWS services with each service emulator mimicking the behavior corresponding to that AWS service, providing the same APIs and responses. There are plenty of emulated services, but the most common ones are covered with some less commonly available as well.
- Persistent storage – there is support for persistent storage for certain services that require it, meaning that data can be retained after restarts. Local directories are used to act as volumes to store data for services such as S3, DynamoDB, etc.
- Configuration and extensions – it is highly configurable, allowing certain services to be enabled and disabled, adjusting ports, various runtime options, etc. It also supports extensions and plugins allowing adding of new features.

Upon startup, LocalStack initializes the Docker environment and sets up the containers for AWS services. This may require downloading additional docker images if they are needed. The Edge service routes the incoming requests to these services which are being emulated based on the targeted service and API. Each service listens to its port and responds to API requests. These emulators use a combination of:

- Mock implementations – mock objects replicate the behavior of AWS services, for example, the local filesystem is used to emulate S3 buckets and objects.
- Local databases – some services use local databases to store and respond to queries (SQLite for example)
- In-memory data structures – for services that don't require persistent storage, an in-memory data structure is used to maintain the state.

When a restart occurs LocalStack will pull its configuration and data from local directories or Docker volumes. This is very useful for students, who often shut down and start their computers.

LocalStack is designed to use extensions so it can be used to add plugins, custom scripts (configuration or data population scripts), and various configuration options and settings with environment variables and configuration files.

To interact with LocalStack there are several options:

- AWS Software Development Kit (SDK) – perfect for development purposes and easy debugging apps.
- AWS Command Line Interface (CLI) – the standard approach for accessing AWS through the command line tools.
- Third-party tools – for example, Terraform, as a popular Infrastructure as Code (IaC) solution

All of this allows developers to easily do local development and testing without the need to deploy or access AWS directly, which also allows complex integration testing between multiple services in a reproducible environment and significant cost reduction that would normally be involved when developing and testing on AWS.

3. USING LOCALSTACK

LocalStack offers an application called LocalStack Desktop. This application supports Windows, Mac, and Linux and its main purpose is to provide an effortless way to view logs, interact with the containers, and use a resource browser from within the application. The main page with the running containers is shown in Figure 2.

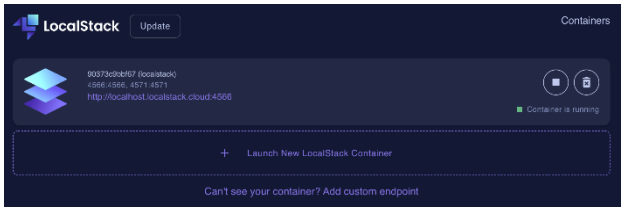

Figure 2. *LocalStack Desktop container management*

Logs can be seen on a container basis as shown in Figure 3.

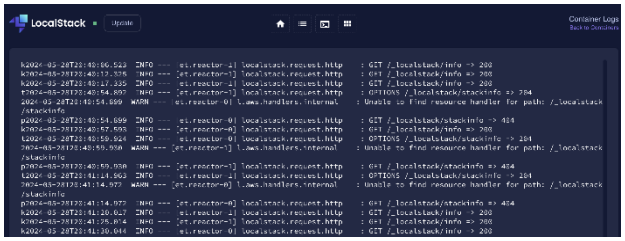

Figure 3. *LocalStack Desktop container log inspection*

Also, the Resource browser (Figure 4.) provides an overview of available resources, where it can be seen that some of them are free, while others require a pro plan. It can be used to view, manage, and deploy AWS services locally.

It is worth noting that the LocalStack in a paid version is available for individual developers, teams, and multiple teams, and the first two options are billed annually. The pro version offers support and advanced options. There is also a 14-day free trial available for the pro version before purchase. However, the community version is licensed under the Apache 2.0 open-source version, which enables smooth commercial use.

The resource browser is meant to provide an interface that is similar to the AWS Management Console. It is not meant to replace it, but rather as an interface to replicate some of its features.

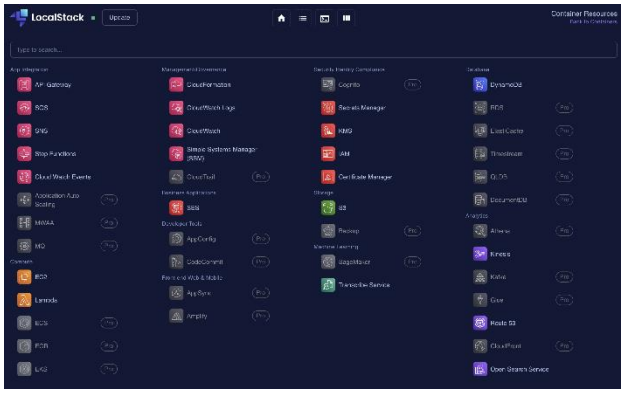

Figure 4. *Resource Browser*

Executing CLI commands requires modifications to the AWS CLI, those include setting up the Access Key ID and Secret Access Key, configuring the region to us-east-1, and overriding the default AWS endpoint to the LocalStack deployment on the local machine. After the configuration is completed standard AWS CLI commands can be used with LocalStack's deployment.

When integrating with the AWS SDK the same rules apply. The configuration remains the same, the only difference is the endpoint that the code is connecting to. Two examples in Figure 5. and Figure 6. can be seen for Java and Python.

```java
// Configure the S3 client to use LocalStack
S3Client s3Client = S3Client.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(StaticCredentialsProvider.create(
                AwsBasicCredentials.create("test", "test")))
        .endpointOverride(URI.create("http://localhost:4566"))
        .build();
```

Figure 5. *Java SDK endpoint override*

```python
# Configure the S3 client to use LocalStack
s3_client = boto3.client(
    's3',
    region_name='us-east-1',
    aws_access_key_id='test',
    aws_secret_access_key='test',
    endpoint_url='http://localhost:4566'
)
```

Figure 6. *Python SDK endpoint override*

This opens up for easy use with IaC tools such as Terraform. Terraform is an open-source IaC tool developed by Hashicorp that allows users to define, provision, and manage cloud infrastructure using declarative configuration language which is referred to as HashiCorp Configuration Language or HCL for short. This approach to writing IaC allows the use of version control, peer review, automation, etc. Terraform uses provider plugins which allow it to work with various providers and switch environments easier. To use Terraform with LocalStack it needs to be configured in a similar way as the AWS CLI and AWS SDK. However, for Terraform, the endpoint override needs to be done on a provider level. This means that all of the endpoints that are going to be used need to be configured in the provider and the credentials should be configured there as well. This configuration can be seen in Figure 7.

After the configuration is completed, the resources can be created and commands such as terraform init, plan and apply can be freely used with LocalStack.

```
provider "aws" {
  access_key                  = "test"
  secret_access_key           = "test"
  region                      = "us-east-1"
  s3_force_path_style         = true
  skip_credentials_validation = true
  skip_requesting_account_id  = true
  endpoints {
    apigateway     = "http://localhost:4566"
    cloudformation = "http://localhost:4566"
    cloudwatch     = "http://localhost:4566"
    dynamodb       = "http://localhost:4566"
    es             = "http://localhost:4566"
    firehose       = "http://localhost:4566"
    iam            = "http://localhost:4566"
    kinesis        = "http://localhost:4566"
    lambda         = "http://localhost:4566"
    route53        = "http://localhost:4566"
    redshift       = "http://localhost:4566"
    s3             = "http://localhost:4566"
    secretsmanager = "http://localhost:4566"
    ses            = "http://localhost:4566"
    sns            = "http://localhost:4566"
    sqs            = "http://localhost:4566"
    ssm            = "http://localhost:4566"
    stepfunctions  = "http://localhost:4566"
    sts            = "http://localhost:4566"
  }
}
```

Figure 7. *Terraform provider configuration for LocalStack*

4. LOCALSTACK IN EDUCATION

Implementing LocalStack in cloud computing classes offers significant benefits that can enhance student's learning experiences. Some of the major benefits include:

- Cost efficiency – there are no AWS charges and costs associated with service usage, as well as no budget limitations as those in AWS Cloud Academy. There is also the benefit of unlimited usage as the whole stack runs locally allowing unlimited experimentation and learning without worries of quotas and charges.
- Real-world skills in a safe environment – students gain hands-on experience that can be used in real-world cloud environments (AWS SDK, CLI, IaC) all while remaining in a safe sandbox environment.
- Feedback – a faster feedback loop for infrastructure changes would allow for a more rapid learning cycle in which students can quickly test their code and configurations. This would also allow students to do local debugging of cloud applications, potentially making it easier to identify and resolve problems.
- Flexibility – cloud applications can be developed and tested entirely on local machines and offline. Due to LocalStacks containerized nature, there are benefits of it working on various operating systems. LocalStack can also be included in Continuous Integration/Continuous Deployment (CI/CD) pipelines for teaching DevOps practices along with a Git-based approach for versioning IaC.
- Enhanced learning – professors and teaching assistants can assign more creative assignments and are not limited to the predefined scenarios in AWS Academy, allowing for more engaging learning as well as projects more tailored to the curriculum.

When implementing LocalStack into the curriculum it would be recommended to have a few starter projects to have students lean into it in an easy-going manner, with more challenging tasks planned as the course progresses. Having documentation and tutorials available for all tooling involved is a must – LocalStack, AWS SDK, AWS CLI, and Terraform (IaC). There are also plenty of community resources on various GitHub repositories, forum posts, etc. With this approach, a peer-reviewed feedback loop can be implemented which would foster collaboration and collective learning, as well as simulating working in a team, within a project in a company.

5. CONCLUSION

LocalStack is a very powerful, flexible, and convenient way to emulate AWS services in a local environment. The Docker-based architecture, service emulation, and extensibility make it a very desirable tool for developers and testers working with AWS infrastructure and applications.

Implementing LocalStack in a cloud computing curriculum could provide students with a cost-effective, low-risk, and practical learning environment. A setup like this would prepare students for real-world roles by offering hands-on experience using industry-standard tools and best practices, which in turn would equip them for their future careers.

ACKNOWLEDGEMENTS

This study was supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia, and these results are parts of the Grant No. 451-03-66/2024-03/200132 with the University of Kragujevac - Faculty of Technical Sciences Čačak.

REFERENCES

[1] Sabin M., Alrumaih H., Impagliazzo J., Lunt M. B., Zhang M., et. al., (2017). Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology. 10.1145/3173161.

[2] Mell P., Grance T., (2011), The NIST Definition of Cloud Computing, 2011. Available at: https://nvlpubs.nist.gov/nistpubs/legacy/sp/n

istspecialpublication800-145.pdf [July 10. 2024]
[3] Milosevic M., Bogicevic Z., Ristic O., (2022). *Implementing the AWS Academy curriculum into a cloud computing course*, 9th International Scientific Conference Technics and informatics in education – TIE 2022, 278-282.
[4] Ali A., Smith T. D., Leslie A. T., (2018), *Issues and challenges facing the teaching of cloud computing for the first time*, Issues in Informational Systems, Vol 19, Issue 4, 187-195.
[5] G. von Laszewski, B. Abdul-Wahid, F. Wang, H. Lee, G. C. Fox, W. Chang, (2017) *Cloudmesh in support of the NIST big data architecture framework*, Technical report, Indiana University.
[6] C. Anglano, M. Canonico, M. Guazzone, (2021), *An educational toolkit for teaching cloud computing*, SIGCOMM Comput. Commun. Rev., Vol 51, Issue 4, 36–46. Available at: https://doi.org/10.1145/3503954.3503959 [July 10, 2024]
[7] C. Anglano, M. Canonico, M. Guazzone, (2020), *Teaching Cloud Computing: Motivations, Challenges and Tools*, IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2020, pp. 300-306.
[8] *OpenStack – Open Source Cloud Software.* Available at*:* https://www.openstack.org/
[9] Eickholt J., Shrestha S., (2017), *Teaching Big Data and Cloud Computing with a Physical Cluster*, ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). Association for Computing Machinery, 177–181.
[10] Goyal T., Singh A., Agrawal A., (2012), *Cloudsim: Simulator for could computing infrastructure and modeling*, Procedia Engineering, Vol 38, 3566 – 3572.
[11] Mansouri N., Ghafari R., Mohammad Hasani Zade B., (2020), *Cloud computing simulators: A comprehensive review*, Simulation Modelling Practice and Theory, Vol 104, DOI:10.1016/j.simpat.2020.102144
[12] LocalStack official documentation. Available at: https://docs.localstack.cloud/references/ [July 10. 2024]
[13] LocalStack project codebase. Available at: https://github.com/localstack/localstack [July 10. 2024]