Int.J.HIT.TRANSC:ECCN.Vol.10: Special Issue (2025) Page 86-90 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN


ORIGINAL CONTRIBUTION

# Automating AWS Infrastructure Deployment Using the CLI in Cloud Computing Technologies

[2]Ashis Kumar Rawani, [2]Amit Kumar Bhowmik, [2]Achintya Mondal, [2]Srijita Bhowmik, [2]Subhadeep Barik, [1]Jayanta Kumar Bag, [1]Dipak Samanta, [1]Chanchal Kumar De

*[1]Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal*

*[2,]UG student, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal*

**ABSTRACT**

This report outlines the process of running an Amazon EC2 instance using the AWS Command Line Interface (CLI). It covers essential prerequisites, including the installation of the AWS CLI and configuration of AWS credentials. The report details step-by-step instructions for selecting an Amazon Machine Image (AMI), choosing an instance type, creating a key pair, launching the EC2 instance, and verifying its status. Additionally, this report outlines the process of deploying a server less application using AWS services specifically AWS Lambda, Dynamo DB, and API Gateway through the AWS CLI. Detailed step-by-step instructions are provided for creating each service and for connecting them to build a cohesive server less architecture using the CLI. Best practices for server less deployment are also discussed. By leveraging the AWS CLI, developers can streamline the deployment process, enhance scalability, and minimize operational overhead, ultimately leading to more efficient and cost-effective server less applications.

**KEYWORDS:** AWS Command Line Interface (CLI), Amazon EC2 Instance, Server less Architecture, AWS Lambda, Scalability

## 1. INTRODUCTION

As cloud computing has rapidly evolved, so too have the methods for managing and deploying cloud infrastructure. Amazon Web Services (AWS) is at the forefront of this transformation, providing an expansive suite of services that cater to diverse business and technical requirements [1, 2]. The AWS Management Console, a user-friendly web-based interface, is often the first choice for setting up and configuring resources. However, managing large-scale, dynamic, or production grade environments through a graphical interface can become time-consuming, error-prone, and operationally inefficient. Automation of infrastructure deployment addresses these challenges by leveraging AWS Command Line Interface (CLI) and Software Development Kits (SDKs), which allow programmatic control over resources, reducing human intervention and enabling repeatable, consistent, and secure infrastructure setup [3, 4]. This report explores the approach of computing service models along with issues faced in the contemporary scenario. automating infrastructure on AWS by launching EC2 instances using AWS CLI with Identity and Access Management (IAM)

*Corresponding Address: jkbag2008@gmail.com

DOI: 10.5281/zenodo.14591981

profiles and creating a server less web application using AWS services such as Lambda, API Gateway, DynamoDB, and S3 [5, 6]. By automating these tasks, we achieve higher efficiency and scalability while improving security. With AWS CLI and SDKs, users can define, configure, and manage AWS resources through code rather than through manual inputs, which is ideal for agile workflows, continuous deployment, and scaling of cloud infrastructure [7,8]. The work detailed in this report includes scripting and configuration management to launch EC2 instances and setting up a fully functional serverless web application. We discuss the motivation, objectives, and key highlights of the work, demonstrating the relevance and advantages of automating AWS infrastructure management. Small code snippets are provided to illustrate this automation.

## HIGHLIGHTS OF THE WORK

**i) Automated AWS Resource Provisioning:**
Efficiently launch and configure AWS services like EC2 instances, S3 buckets, and IAM roles using AWS CLI scripts.

**ii) Infrastructure-as-Code (IaC):**
Demonstrate the integration of AWS CLI with IaC principles for repeatable and consistent deployments.

**iii) Serverless Architecture Automation:**
Deploy and manage serverless applications, including AWS Lambda functions and API Gateway, using CLI commands.

**iv) Security and Access Management:**
Automate the creation and assignment of IAM roles and policies to ensure secure infrastructure management.

**v) Cost Monitoring and Optimization:**
Utilize CLI commands to track resource usage and implement cost-saving measures.

**vi) DevOps Integration:**
Align AWS CLI automation with DevOps workflows for seamless CI/CD pipeline integration.

**vii) Real-World Use Cases:**
Provide practical examples and scripts for automating cloud operations, ensuring hands-on applicability.

**viii) Time and Error Reduction:**
Highlight the efficiency of automation in reducing deployment time and minimizing configuration errors.

Rest of this paper is as follows: Section 2 briefly explains the method of AWS infrastructure automation. Subsequently, section 3 describes the method of automating EC2 in-stance launch using AWS CLI. In section 4, the results and discussion are illustrated briefly. The paper is concluded in the final section 5.

## 2. AWS INFRASTRUCTURE AUTOMATION:

The roadmap for AWS infrastructure automation is implemented in the following steps [9,10]:

i) Installation Process of AWS CLI
ii) Install AWS CLI following the instructions provided by AWS.
iii) Creating an IAM User and Providing Permissions
iv) Create an IAM user and configure permissions necessary for access- ing EC2, Lambda, Dynamo DB, and API Gateway.
v) Launching an EC2 Instance Using Command Line Interface
vi) Creating Key Pair: Generate a key pair to securely access the EC2 instance.
vii) Configuring Security Group: Set up a security group with necessary inbound and outbound rules.
viii) Running Instance: Launch an EC2 instance using the CLI com- mand.
ix) Providing Name: Assign a name to the instance for easy identifica- tion.
x) Starting/Stopping Instance: Use CLI commands to start and stop the instance as needed.
xi) Terminating Instance: Terminate the instance when no longer needed.
xii) Launching Lambda through CLI (Optional)
xiii) Configure and launch AWS Lambda functions directly via CLI for additional automation capabilities.

## 3. AUTOMATING EC2 IN-STANCE LAUNCH USING AWS CLI

The AWS Command Line Interface (CLI) is a

tool that lets you interact with AWS services from the terminal. It provides a unified command-line interface for managing AWS services [11,12].

***A. Tools and Resources Needed***

i) AWS CLI: The main tool for automation. Version 2 or the latest AWS CLI should be installed.
ii) IAM Role/Policy: Set up IAM roles with appropriate permissions (EC2 launch privileges) mapped to the user who will run the CLI commands.
iii) Key Pair: A key pair for securing SSH access to the EC2 instance.
iv) Security Group: Define security groups for access control (allowing SSH and HTTP/HTTPS access as needed).
v) AWS Account: An active AWS account with sufficient permissions.
vi) EC2 AMI ID: The specific Amazon Machine Image (AMI) ID to launch the EC2 instance.
vii) Programming Environment: A local terminal or cloud-based environment to execute AWS CLI commands.

viii)Text Editor: Any preferred text editor for scripting, such as Visual Studio Code or Sublime Text.

***B. Procedure/Steps to Implement***

**Step 1: Install and Configure AWS CLI:**
• Install the AWS CLI tool from the AWS website.
• Configure AWS CLI with IAM credentials (Access Key ID and Secret Ac- cess Key) by using:
• AWS configure
• Set the default region (e.g., us-east-1) and output format (json).

**Step 2: Create and Map an IAM User Profile**
• Create IAM Role: Navigate to the IAM Console → Roles → Create Role.
• Choose AWS Service as the trusted entity and select EC2.
• Attach policies (such as Amazon EC2 Full Access) to provide the role with permissions to launch and manage EC2 instances
• Assign a name to the role and finish creation.

**Assign Role to IAM User:**
• Navigate to IAM → Users → Add Permissions for the specific user.
• Attach the necessary policies or add the previously created IAM role to the user, allowing EC2 management privileges.

**Step 3: Write AWS CLI Commands for Key Pair and Security Group**
Using the AWS CLI, launch the key pair with the following command:
**aws ec2 create-key-pair –key-name keypair1 –query 'keymaterial' – output text > keypair1.ppk**

**Step 4: Write AWS CLI Commands to Create Security Group:**
Using the AWS CLI, launch the with the following command: **aws ec2 create-security-group –group-name securitygroup1 – description "Security- group1**"
•--image-id: The AMI ID to be used.
•--instance-type: The type of EC2 instance to launch.
•--key-name: The key pair name for SSH access.
•--security-groups: The security group for access control.

**Step 5: Write AWS CLI Commands to Launch EC2 Instance:**
Using the AWS CLI, launch the EC2 instance with the following command:
**aws ec2 run-instances –image-id ami-0035f356066b106a8 –count 1 – instance-type t2. micro –key-name keypair1 –security-groups securitygroup1**

**Step 6: Manage EC2 Instance Tags and States:**
•Assign Name: aws ec2 create-tags --resources i-0e3e1e0c18607104f—tags Key=Name,Value=achha_sa_naam
•Stop Instance:
i-0e3e1e0c18607104faws ec2 stop-instances --instance-ids i-0e3e1e0c18607104f
•Start Instance:
i-0e3e1e0c18607104faws ec2 start-instances --instance-ids i-0e3e1e0c18607104f
•Terminate Instance:
i-0e3e1e0c18607104faws ec2 terminate-instances --instance-ids i-0e3e1e0c18607104f

***C. Algorithm of the Codes***

**Key Pair Creation:**
Command: aws ec2 create-key-pair
Subcommands:
•--key-name keypair1: Specifies the name of the key pair to be created.

•--query 'keymaterial': Extracts only the key material (private key) from the response.
•--output text: Outputs the key material in plain text format.
•> keypair1.ppk: Redirects the output to a file named keypair1.ppk.

**Security Group Creation:**
Command: aws ec2 create-security-group
Subcommands:
•--group-name securitygroup1: Specifies the name of the security group to be created.
•--description "Securitygroup1": Provides a description for the security group.

**Instance Creation:**
Command: aws ec2 run-instances
Subcommands:
•--image-id ami-0035f356066b106a8: Specifies the AMI ID (Amazon Ma- chine Image) to use for the instance.
•--count 1: Specifies the number of instances to create.
•--instance-type t2. micro: Specifies the instance type (e.g., t2. micro).
•--key-name keypair1: Specifies the key pair to use for the instance.
•--security-groups securitygroup1: Specifies the security group to associate with the instance.

**Tagging Instance:**
Command: aws ec2 create-tags
Subcommands:
•--resources i-0e3e1e0c18607104f: Specifies the resource ID (instance ID) to tag.
•--tagsKey=Name,Value=achha_sa_naam:
Specifies the key-value pair of tags to add to the instance.

**Starting Instance:**
Command: aws ec2 start-instances
Subcommands:
•--instance-ids i-0e3e1e0c18607104f: Specifies the instance ID(s) to start.

**Stopping Instance:**
Command: aws ec2 stop-instances
Subcommands:
•--instance-ids i-0e3e1e0c18607104f: Specifies the instance ID(s) to stop.

**Terminating Instance:**
Command: aws ec2 terminate-instances
Subcommands:
•--instance-ids i-0e3e1e0c18607104f: Specifies the instance ID(s) to terminate.

## 4. RESULTS AND DISCUSSION

This research work "Automating AWS Infrastructure with CLI" yield significant results across two primary dimensions: infrastructure automation using AWS CLI and SDKs. Automated EC2 Instance Deployment with AWS CLI: -One of the main objectives was to automate the deployment of EC2 instances, which are essential for hosting and running applications on AWS. By using AWS CLI, instances could be provisioned consistently without manual interaction through the AWS Management Console. The CLI commands included creating and configuring instances, associating IAM roles, and setting up security groups. Lambda functions were developed to handle the backend logic, processing requests received from the API Gateway. Each function triggered automatically based on incoming events, demonstrating the true "event-driven" nature of serverless architecture. Here's a sample Lambda
function written in Python:

```
import json
def lambdahandler(event, context) :
return
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
```

The advantages of automating AWS infrastructure with CLI are listed below:

**Efficient Request Routing with API Gateway:**
API Gateway was configured to act as a front-door entry point, routing HTTP requests to appropriate Lambda functions.This configuration provided a scalable API endpoint, which could handle a high number of concurrent requests without additional configuration.

**Data Storage and Retrieval with DynamoDB:**
The use of DynamoDB provided a NoSQL database solution that automatically scaled based on traffic. DynamoDB was chosen for its high availability and low latency, which complemented the serverless design by offering a managed database service.

**Cost Optimization:**
The pay-as-you-go model inherent in serverless applications proved cost-effective, as charges accrued only when resources were actively being used. For instance, Lambda and API Gateway billed for request handling and compute time

only, allowing the application to operate with minimal idle costs.

## 5. CONCLUSION

Launching an EC2 instance using the AWS CLI offers a powerful, streamlined approach to managing cloud resources efficiently. By following the outlined steps, users can deploy instances directly from the command line, allowing for greater control and flexibility in cloud operations. This method is particularly advantageous for automating workflows, scaling deployments, and managing resources in a consistent, repeatable man- ner. Moreover, using the CLI eliminates the need to navigate the AWS Management Con- sole, saving time and facilitating integration with scripts and automation tools. Overall, mastering the AWS CLI for EC2 deployments is an essential skill for cloud practitioners, providing a versatile foundation for more advanced AWS automation and infrastructure management.

## REFERENCES

[1] https://docs.aws.amazon.com/cli/.
[2] https://aws.amazon.com/architecture/well-architected
[3] https://aws.amazon.com/blogs/developer
[4] https://stackoverflow.com.
[5] https://dev.to.
[6] https://aws.amazon.com/aws-cost-management
[7] https://aws.amazon.com/cloudformation
[8] https://aws.amazon.com/security
[9] https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html.
[10] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ iam-roles-for-amazon-ec2.html#attach-instance-profile.
[11] Nikit Swaraj, “AWS Automation Cookbook A practical guide to automating AWS infrastructure”, Packt Publishingcbs,2017.
[12] Online courses on platforms like Udemy and Coursera, focusing on AWS CLI and infrastructure-as-code (IaC).