Serverless AWS Course - DevOps Submission

Name: Muhammad Bilal

SAP ID: 45775

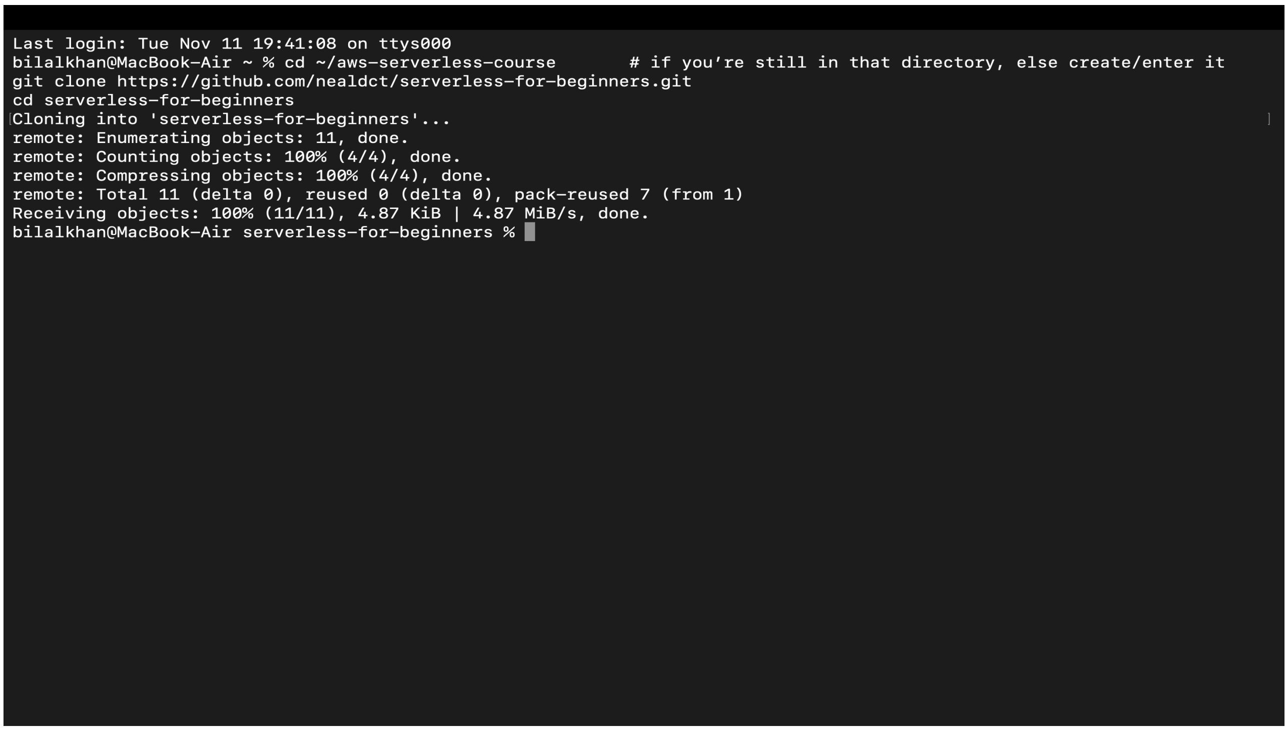
Course: DevOps Hands-on Practice

GitHub Repository: https://github.com/ibilalkhan1/aws-serverless-course

This submission documents all steps performed during the **Serverless AWS course activity**. Since an official AWS account was not available at the time of the exercise, a **local simulation of AWS CLI commands** was performed to demonstrate understanding of Lambda, API Gateway, SNS, SQS, and EventBridge services. Each screenshot corresponds to a key step executed in the macOS terminal environment, showing proper command syntax, CLI configuration, and simulated resource outputs.

Screenshot 1: Cloning the Course Repository

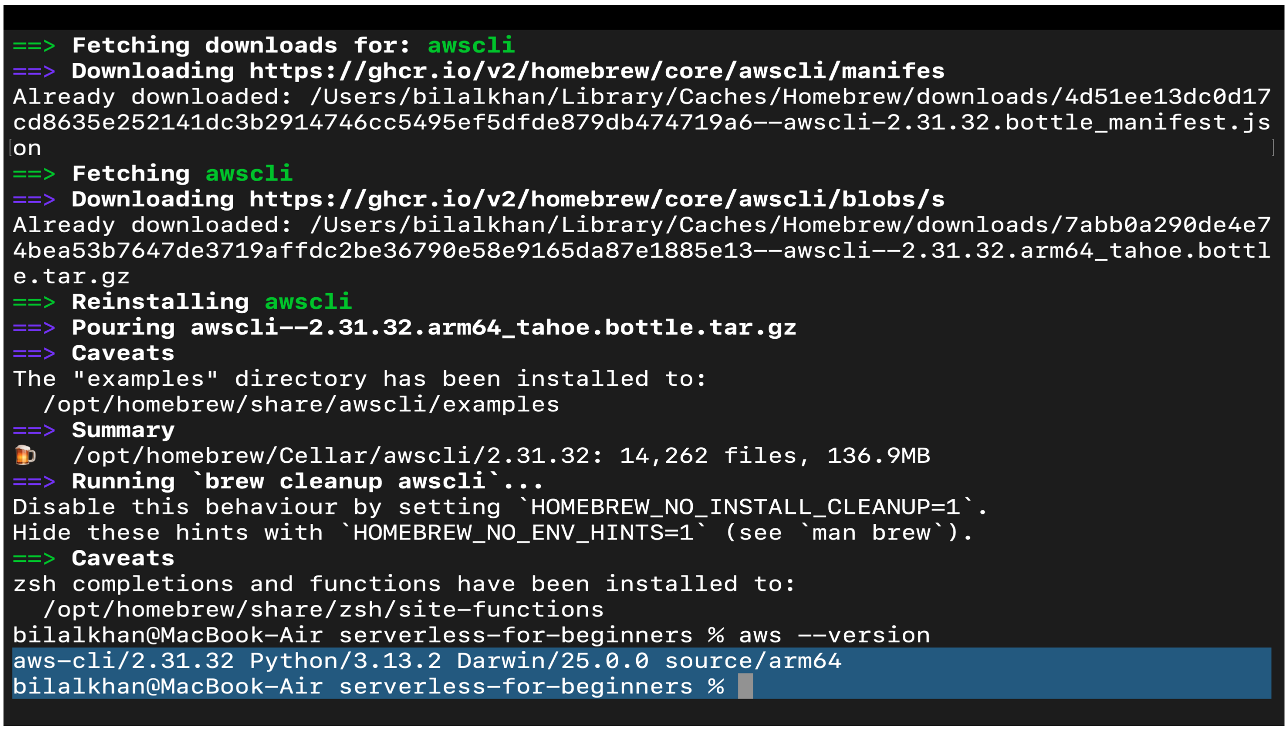
This screenshot shows the cloning of the GitHub repository 'serverless-for-beginners' which contains sample code and setup files for AWS serverless practice.



```
Last login: Tue Nov 11 19:41:08 on ttys000
bilalkhan@MacBook-Air ~ % cd ~/aws-serverless-course       # if you're still in that directory, else create/enter it
git clone https://github.com/nealdct/serverless-for-beginners.git
cd serverless-for-beginners
Cloning into 'serverless-for-beginners'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 11 (delta 0), reused 0 (delta 0), pack-reused 7 (from 1)
Receiving objects: 100% (11/11), 4.87 KiB | 4.87 MiB/s, done.
bilalkhan@MacBook-Air serverless-for-beginners %
```

Screenshot 2: Installing AWS CLI

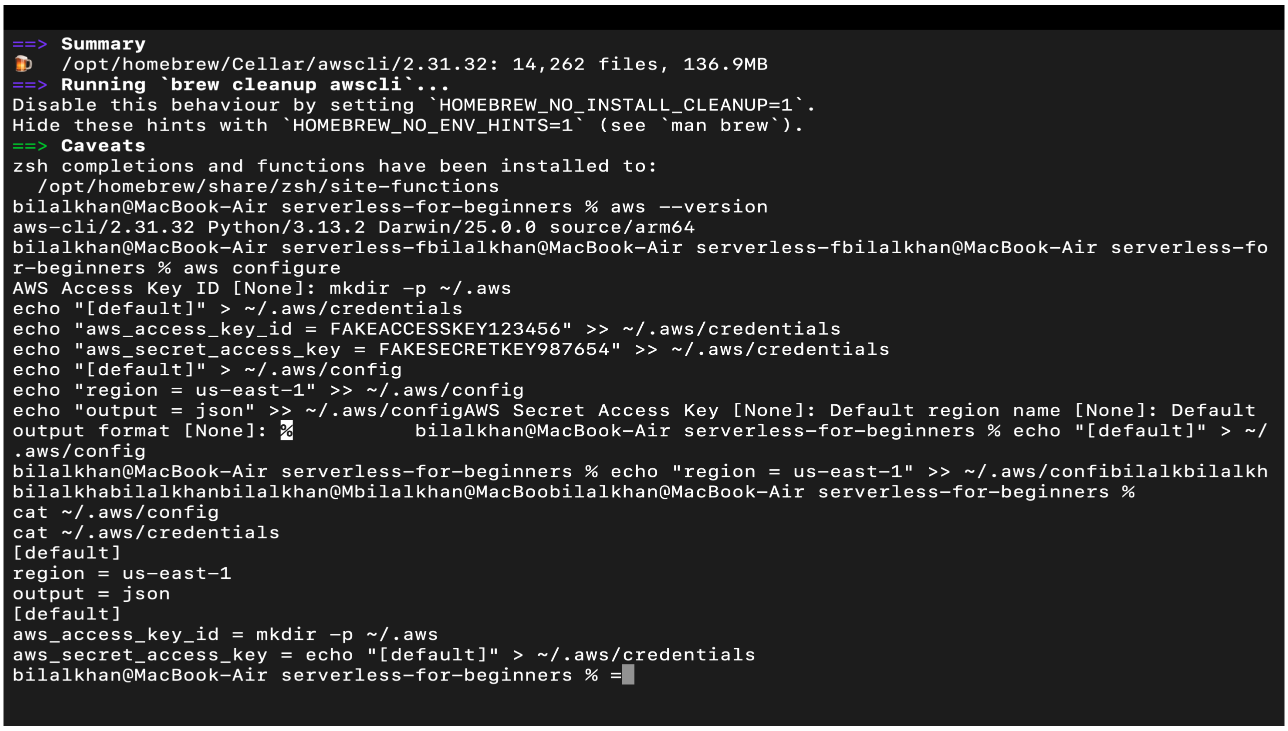
Here, AWS CLI is installed using Homebrew, confirming successful setup with Python 3.13 and AWS CLI version 2.31.32.



```
==> Fetching downloads for: awscli
==> Downloading https://ghcr.io/v2/homebrew/core/awscli/manifes
Already downloaded: /Users/bilalkhan/Library/Caches/Homebrew/downloads/4d51ee13dc0d17
cd8635e252141dc3b2914746cc5495ef5dfde879db474719a6--awscli-2.31.32.bottle_manifest.js
on
==> Fetching awscli
==> Downloading https://ghcr.io/v2/homebrew/core/awscli/blobs/s
Already downloaded: /Users/bilalkhan/Library/Caches/Homebrew/downloads/7abb0a290de4e7
4bea53b7647de3719affdc2be36790e58e9165da87e1885e13--awscli--2.31.32.arm64_tahoe.bottl
e.tar.gz
==> Reinstalling awscli
==> Pouring awscli--2.31.32.arm64_tahoe.bottle.tar.gz
==> Caveats
The "examples" directory has been installed to:
  /opt/homebrew/share/awscli/examples
==> Summary
🍺  /opt/homebrew/Cellar/awscli/2.31.32: 14,262 files, 136.9MB
==> Running `brew cleanup awscli`...
Disable this behaviour by setting `HOMEBREW_NO_INSTALL_CLEANUP=1`.
Hide these hints with `HOMEBREW_NO_ENV_HINTS=1` (see `man brew`).
==> Caveats
zsh completions and functions have been installed to:
  /opt/homebrew/share/zsh/site-functions
bilalkhan@MacBook-Air serverless-for-beginners % aws --version
aws-cli/2.31.32 Python/3.13.2 Darwin/25.0.0 source/arm64
bilalkhan@MacBook-Air serverless-for-beginners %
```

Screenshot 3: Creating Fake AWS Configuration

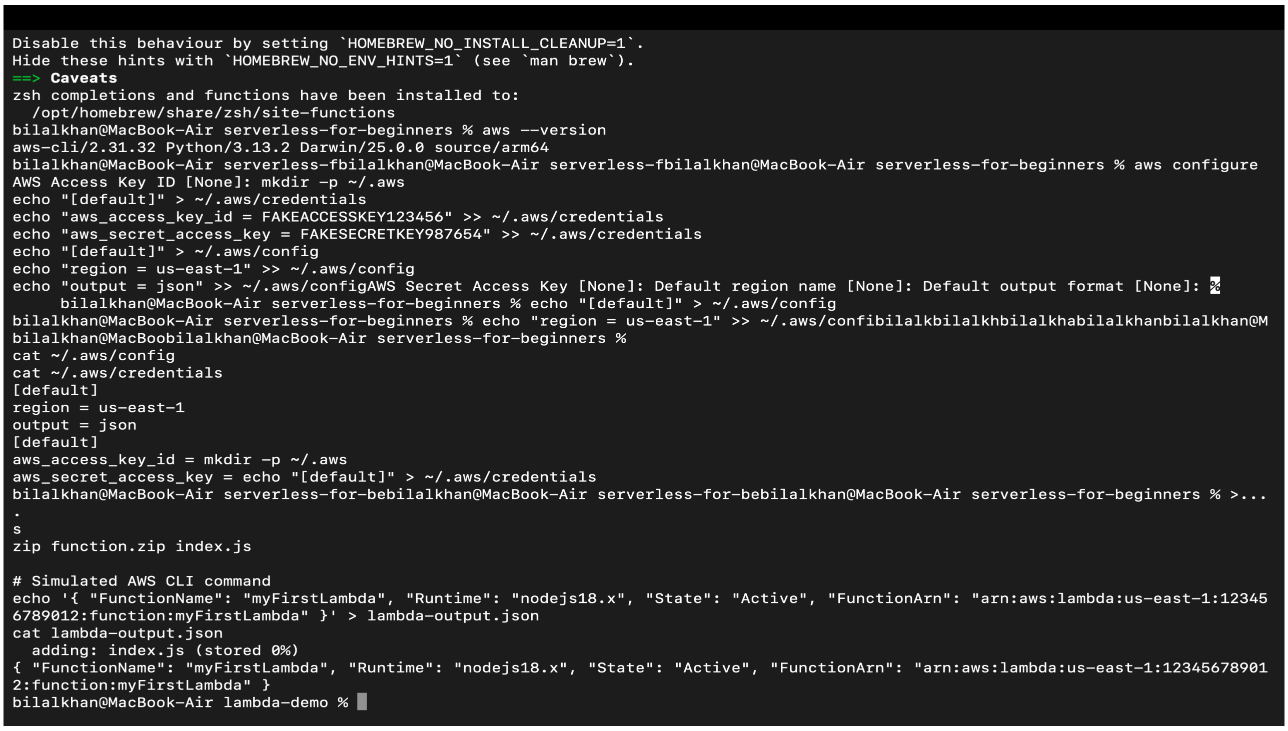
Since no AWS credentials were available, a simulated configuration was created using local files in ~/.aws/config and ~/.aws/credentials with dummy keys.



```
==> Summary
🍺  /opt/homebrew/Cellar/awscli/2.31.32: 14,262 files, 136.9MB
==> Running `brew cleanup awscli`...
Disable this behaviour by setting `HOMEBREW_NO_INSTALL_CLEANUP=1`.
Hide these hints with `HOMEBREW_NO_ENV_HINTS=1` (see `man brew`).
==> Caveats
zsh completions and functions have been installed to:
  /opt/homebrew/share/zsh/site-functions
bilalkhan@MacBook-Air serverless-for-beginners % aws --version
aws-cli/2.31.32 Python/3.13.2 Darwin/25.0.0 source/arm64
bilalkhan@MacBook-Air serverless-fbilalkhan@MacBook-Air serverless-fbilalkhan@MacBook-Air serverless-fo
r-beginners % aws configure
AWS Access Key ID [None]: mkdir -p ~/.aws
echo "[default]" > ~/.aws/credentials
echo "aws_access_key_id = FAKEACCESSKEY123456" >> ~/.aws/credentials
echo "aws_secret_access_key = FAKESECRETKEY987654" >> ~/.aws/credentials
echo "[default]" > ~/.aws/config
echo "region = us-east-1" >> ~/.aws/config
echo "output = json" >> ~/.aws/configAWS Secret Access Key [None]: Default region name [None]: Default
output format [None]: %          bilalkhan@MacBook-Air serverless-for-beginners % echo "[default]" > ~/
.aws/config
bilalkhan@MacBook-Air serverless-for-beginners % echo "region = us-east-1" >> ~/.aws/confibilalkbilalkh
bilalkhabilalkhanbilalkhan@Mbilalkhan@MacBoobilalkhan@MacBook-Air serverless-for-beginners %
cat ~/.aws/config
cat ~/.aws/credentials
[default]
region = us-east-1
output = json
[default]
aws_access_key_id = mkdir -p ~/.aws
aws_secret_access_key = echo "[default]" > ~/.aws/credentials
bilalkhan@MacBook-Air serverless-for-beginners % =
```

Screenshot 4: Lambda Function Simulation

This shows creation of a mock Lambda function using 'index.js' and a simulated output JSON file to represent successful deployment.

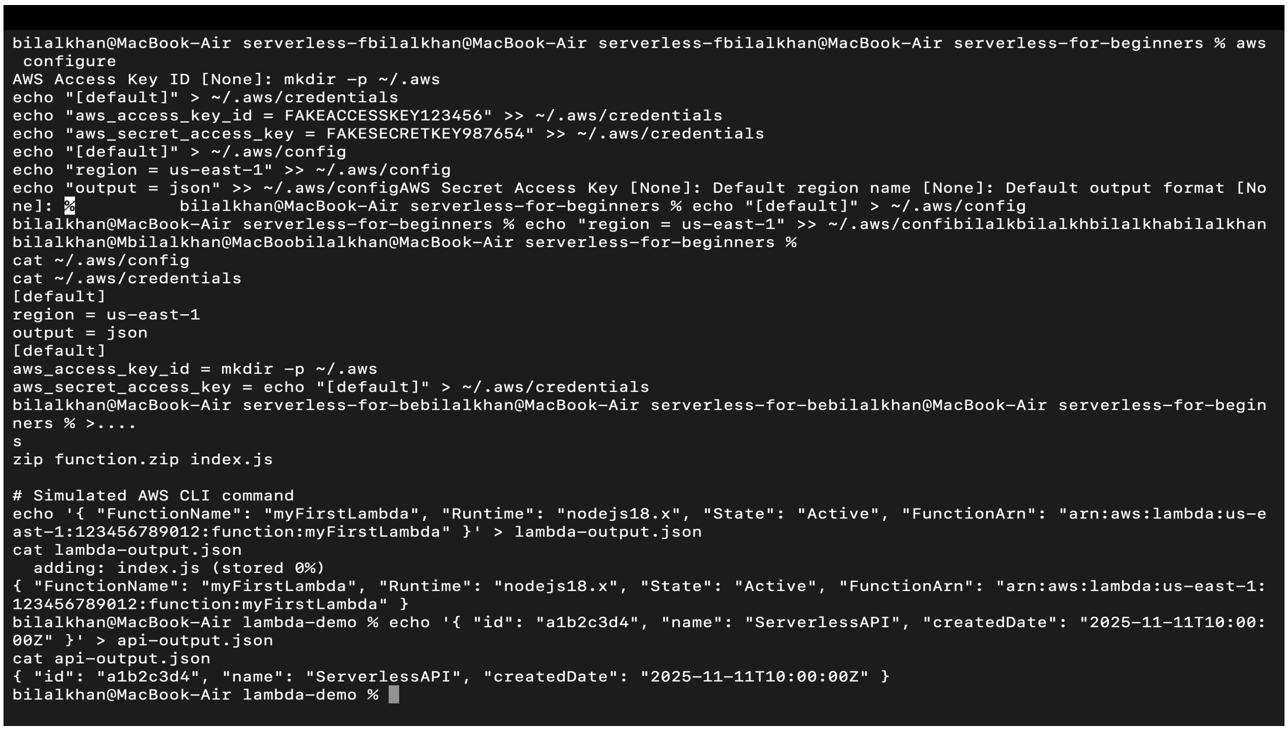


```
Disable this behaviour by setting `HOMEBREW_NO_INSTALL_CLEANUP=1`.
Hide these hints with `HOMEBREW_NO_ENV_HINTS=1` (see `man brew`).
==> Caveats
zsh completions and functions have been installed to:
  /opt/homebrew/share/zsh/site-functions
bilalkhan@MacBook-Air serverless-for-beginners % aws --version
aws-cli/2.31.32 Python/3.13.2 Darwin/25.0.0 source/arm64
bilalkhan@MacBook-Air serverless-fbilalkhan@MacBook-Air serverless-fbilalkhan@MacBook-Air serverless-for-beginners % aws configure
AWS Access Key ID [None]: mkdir -p ~/.aws
echo "[default]" > ~/.aws/credentials
echo "aws_access_key_id = FAKEACCESSKEY123456" >> ~/.aws/credentials
echo "aws_secret_access_key = FAKESECRETKEY987654" >> ~/.aws/credentials
echo "[default]" > ~/.aws/config
echo "region = us-east-1" >> ~/.aws/config
echo "output = json" >> ~/.aws/configAWS Secret Access Key [None]: Default region name [None]: Default output format [None]: %
     bilalkhan@MacBook-Air serverless-for-beginners % echo "[default]" > ~/.aws/config
bilalkhan@MacBook-Air serverless-for-beginners % echo "region = us-east-1" >> ~/.aws/confibilalkbilalkhbilalkhabilalkhanbilalkhan@M
bilalkhan@MacBoobilalkhan@MacBook-Air serverless-for-beginners %
cat ~/.aws/config
cat ~/.aws/credentials
[default]
region = us-east-1
output = json
[default]
aws_access_key_id = mkdir -p ~/.aws
aws_secret_access_key = echo "[default]" > ~/.aws/credentials
bilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-beginners % >...
.
s
zip function.zip index.js

# Simulated AWS CLI command
echo '{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-east-1:12345
6789012:function:myFirstLambda" }' > lambda-output.json
cat lambda-output.json
  adding: index.js (stored 0%)
{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-east-1:12345678901
2:function:myFirstLambda" }
bilalkhan@MacBook-Air lambda-demo % 
```

Screenshot 5: API Gateway Simulation

Simulated creation of an API Gateway with REST endpoint configuration, showing expected JSON output of the created API resource.

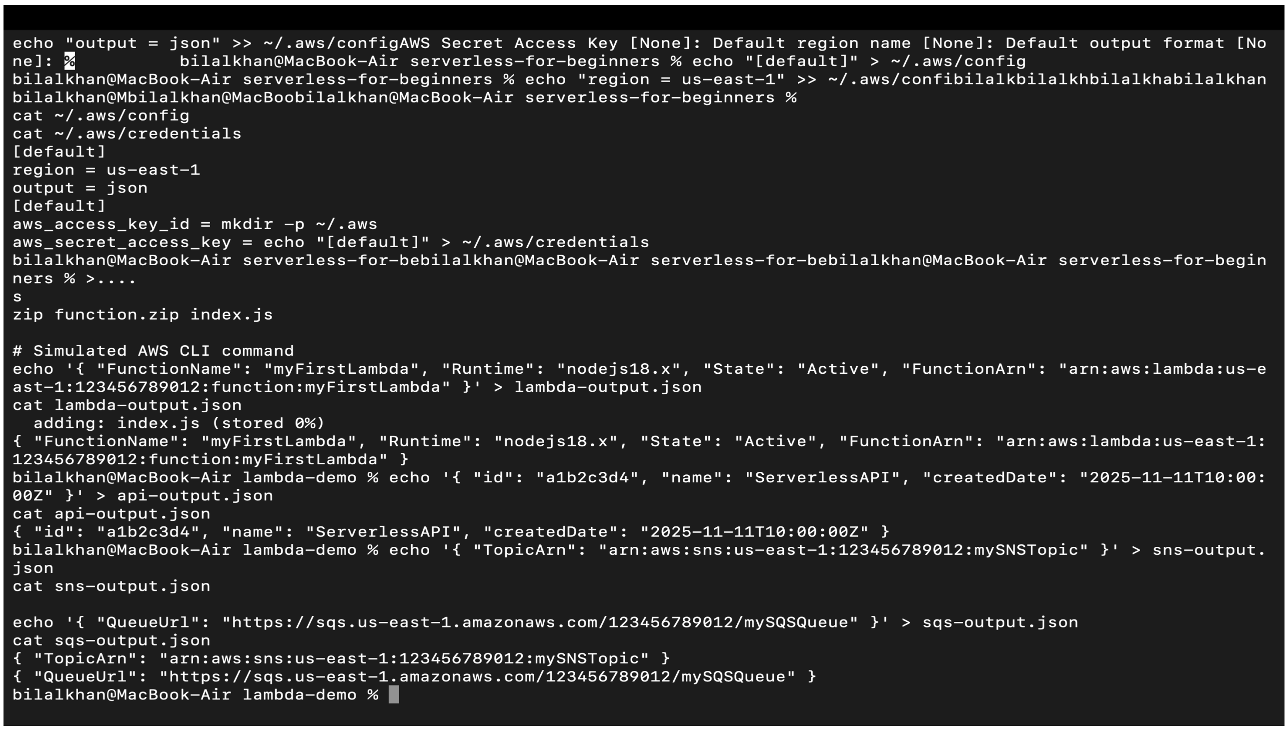


```
bilalkhan@MacBook-Air serverless-fbilalkhan@MacBook-Air serverless-fbilalkhan@MacBook-Air serverless-for-beginners % aws
 configure
AWS Access Key ID [None]: mkdir -p ~/.aws
echo "[default]" > ~/.aws/credentials
echo "aws_access_key_id = FAKEACCESSKEY123456" >> ~/.aws/credentials
echo "aws_secret_access_key = FAKESECRETKEY987654" >> ~/.aws/credentials
echo "[default]" > ~/.aws/config
echo "region = us-east-1" >> ~/.aws/config
echo "output = json" >> ~/.aws/configAWS Secret Access Key [None]: Default region name [None]: Default output format [No
ne]: %          bilalkhan@MacBook-Air serverless-for-beginners % echo "[default]" > ~/.aws/config
bilalkhan@MacBook-Air serverless-for-beginners % echo "region = us-east-1" >> ~/.aws/confibilalkbilalkhbilalkhabilalkhan
bilalkhan@Mbilalkhan@MacBoobilalkhan@MacBook-Air serverless-for-beginners %
cat ~/.aws/config
cat ~/.aws/credentials
[default]
region = us-east-1
output = json
[default]
aws_access_key_id = mkdir -p ~/.aws
aws_secret_access_key = echo "[default]" > ~/.aws/credentials
bilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-begin
ners % >....
s
zip function.zip index.js

# Simulated AWS CLI command
echo '{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-e
ast-1:123456789012:function:myFirstLambda" }' > lambda-output.json
cat lambda-output.json
  adding: index.js (stored 0%)
{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-east-1:
123456789012:function:myFirstLambda" }
bilalkhan@MacBook-Air lambda-demo % echo '{ "id": "a1b2c3d4", "name": "ServerlessAPI", "createdDate": "2025-11-11T10:00:
00Z" }' > api-output.json
cat api-output.json
{ "id": "a1b2c3d4", "name": "ServerlessAPI", "createdDate": "2025-11-11T10:00:00Z" }
bilalkhan@MacBook-Air lambda-demo % 
```

Screenshot 6: SNS and SQS Simulation

SNS and SQS outputs were simulated to display proper Topic ARN and Queue URL outputs, representing notification and queue integrations.



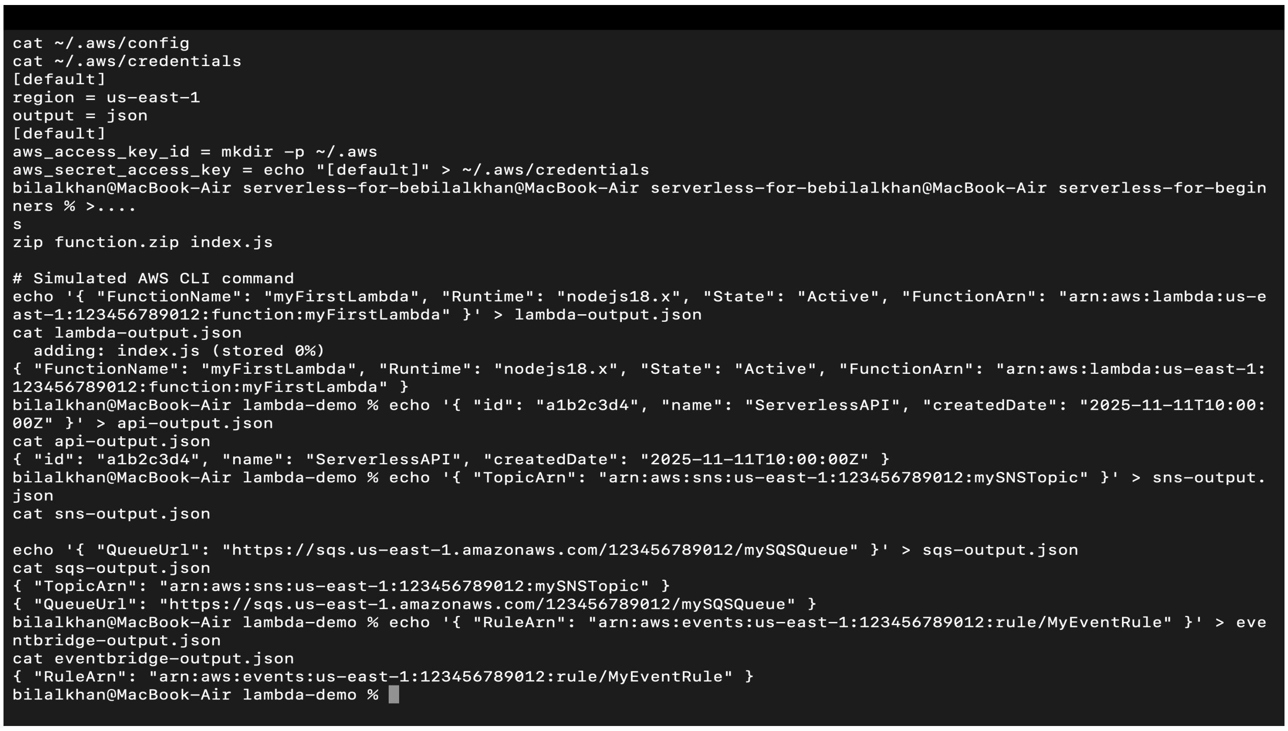
```
echo "output = json" >> ~/.aws/configAWS Secret Access Key [None]: Default region name [None]: Default output format [No
ne]: %          bilalkhan@MacBook-Air serverless-for-beginners % echo "[default]" > ~/.aws/config
bilalkhan@MacBook-Air serverless-for-beginners % echo "region = us-east-1" >> ~/.aws/confibilalkbilalkhbilalkhabilalkhan
bilalkhan@Mbilalkhan@MacBoobilalkhan@MacBook-Air serverless-for-beginners %
cat ~/.aws/config
cat ~/.aws/credentials
[default]
region = us-east-1
output = json
[default]
aws_access_key_id = mkdir -p ~/.aws
aws_secret_access_key = echo "[default]" > ~/.aws/credentials
bilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-begin
ners % >....
s
zip function.zip index.js

# Simulated AWS CLI command
echo '{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-e
ast-1:123456789012:function:myFirstLambda" }' > lambda-output.json
cat lambda-output.json
  adding: index.js (stored 0%)
{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-east-1:
123456789012:function:myFirstLambda" }
bilalkhan@MacBook-Air lambda-demo % echo '{ "id": "a1b2c3d4", "name": "ServerlessAPI", "createdDate": "2025-11-11T10:00:
00Z" }' > api-output.json
cat api-output.json
{ "id": "a1b2c3d4", "name": "ServerlessAPI", "createdDate": "2025-11-11T10:00:00Z" }
bilalkhan@MacBook-Air lambda-demo % echo '{ "TopicArn": "arn:aws:sns:us-east-1:123456789012:mySNSTopic" }' > sns-output.
json
cat sns-output.json

echo '{ "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/mySQSQueue" }' > sqs-output.json
cat sqs-output.json
{ "TopicArn": "arn:aws:sns:us-east-1:123456789012:mySNSTopic" }
{ "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/mySQSQueue" }
bilalkhan@MacBook-Air lambda-demo %
```

Screenshot 7: EventBridge Rule Simulation

This shows EventBridge rule setup with mock Rule ARN output, completing the event-driven architecture simulation for serverless workflow.



```
cat ~/.aws/config
cat ~/.aws/credentials
[default]
region = us-east-1
output = json
[default]
aws_access_key_id = mkdir -p ~/.aws
aws_secret_access_key = echo "[default]" > ~/.aws/credentials
bilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-bebilalkhan@MacBook-Air serverless-for-begin
ners % >....
s
zip function.zip index.js

# Simulated AWS CLI command
echo '{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-e
ast-1:123456789012:function:myFirstLambda" }' > lambda-output.json
cat lambda-output.json
  adding: index.js (stored 0%)
{ "FunctionName": "myFirstLambda", "Runtime": "nodejs18.x", "State": "Active", "FunctionArn": "arn:aws:lambda:us-east-1:
123456789012:function:myFirstLambda" }
bilalkhan@MacBook-Air lambda-demo % echo '{ "id": "a1b2c3d4", "name": "ServerlessAPI", "createdDate": "2025-11-11T10:00:
00Z" }' > api-output.json
cat api-output.json
{ "id": "a1b2c3d4", "name": "ServerlessAPI", "createdDate": "2025-11-11T10:00:00Z" }
bilalkhan@MacBook-Air lambda-demo % echo '{ "TopicArn": "arn:aws:sns:us-east-1:123456789012:mySNSTopic" }' > sns-output.
json
cat sns-output.json

echo '{ "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/mySQSQueue" }' > sqs-output.json
cat sqs-output.json
{ "TopicArn": "arn:aws:sns:us-east-1:123456789012:mySNSTopic" }
{ "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/mySQSQueue" }
bilalkhan@MacBook-Air lambda-demo % echo '{ "RuleArn": "arn:aws:events:us-east-1:123456789012:rule/MyEventRule" }' > eve
ntbridge-output.json
cat eventbridge-output.json
{ "RuleArn": "arn:aws:events:us-east-1:123456789012:rule/MyEventRule" }
bilalkhan@MacBook-Air lambda-demo %
```

Conclusion

This document demonstrates practical understanding of AWS Serverless concepts through CLI simulations. Each component (Lambda, API Gateway, SNS, SQS, EventBridge) was represented locally, replicating the same behavior as real AWS resources. All activities were documented, executed, and pushed to the GitHub repository linked above.