TOYOTA
CONNECTED INDIA

Build a Replicable Serverless Python NLP App from Scratch

Empower Sustainable Mobility and Deliver Elevated Customer Experiences

Table of Contents

I. Introduction

- What is our Goal?
- Motivation for the Talk
- Why AWS CLI
- Why Taskfiles
- GitHub Codes supporting this talk

What is our Goal?

> ***Automation in infrastructure provisioning*** *is key to reducing the development time.*
> *No one else can do this better than a developer so that DevOps engineer does less heavy-lifting at the time-crunched release window period*

- Create Infrastructure provisioning codes as & when you write your application development codes
- Make every small configuration component of your provisioned services traceable, and version-controlled

In this talk, we will ...

> *Cover how to build a Python NLP App in Cloud using **replicable infrastructure provisioning codes***

> *Give transferable learnings on building **Serverless Apps in any cloud infra** (here, AWS Cloud infra is chosen).*

> *Be using a combination of **CLI commands and Yaml Taskfiles** to provision the AWS Infrastructure.*

What is in it for you?

By the end of this talk, a developer will

- start loving the potent combination of **AWS CLI** + **Taskfile** for provisioning replicable cloud infrastructure
- understand how this AWS CLI approach helps in-depth understanding of the cloud Services employed
- reduce the "hell month" for DevOps Engineers during releases (refer **adopting DevOps mindset**)

Motivation for the Talk

- Is DevOps an afterthought for a developer?
- Does a developer have the same rigor in `application code development` applied to their `infrastructure provisioning codes`?

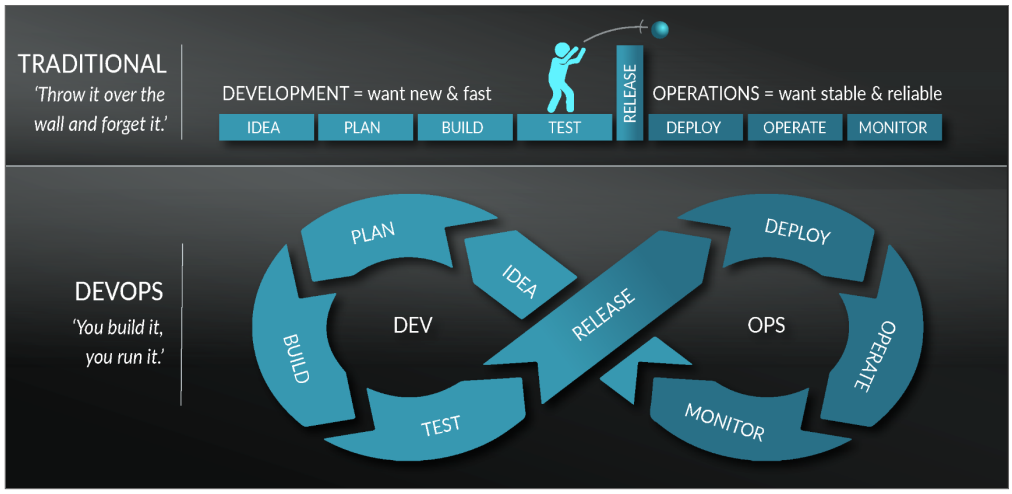


Image Source: https://modelthinkers.com/mental-model/devops-mindset

Why Dev needs to adopt DevOps Principles

- More nimble & responsive product delivery !

> *The right DevOps culture ultimately makes you deliver better products faster.*

Source: https://stackoverflow.blog/2020/06/10/the-rise-of-the-devops-mindset/

> *"It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change." Charles Darwin*

Source: https://faun.pub/some-popular-devops-quotes-and-what-i-learned-from-it-using-in-my-day-to-day-development-7b299ced7884

Why AWS CLI?

The reason why we chose AWS CLI is that

- it is a unified interface to all AWS services and
- it mimics the console way of creating AWS services with the right* level of abstraction
    - meaning, you have control over the CRUD operations
- it is not different from a traditional bash commands that we are highly familiar with

*arguable personal opinion

- What you give in arguments is what is provisioned for you !

```
/path/where/lambda_codes/located % cat
aws_cli_command_for_lambda_creation.bash
#!/bin/bash
aws lambda create-function \
--function-name $1 \
--zip-file fileb://${1}.zip \
--runtime python3.8 \
--role $2 \
--handler lambda_function.lambda_handler \
--timeout 60 \
--memory-size 256 \
--layers $3 \
--architectures x86_64
```

- And, it gets executed with a just `task <task_name>`

```
/path/where/lambda_codes/located % cat
Taskfile.yml
...
...
tasks:
  create_lambda_name:
    cmds:
      - zip -r ${LAMBDA_FUNCTION_NAME}.zip
lambda_function.py
      - bash
aws_cli_command_for_lambda_creation.bash
$LAMBDA_FUNCTION_NAME $IAM_ROLE_ARN $SPACY_LAYER
```

Why Taskfiles?

- The Taskfiles make the execution of steps so easy. You could repeat it multiple times for various projects.

```
# how to create IAM policy and roles
/path/where/IAM_Taskfile/is/located/ % task
create_policy && task create_role && task
attach_role_to_policy

# how lambda function is created
/path/where/LAMBDA_Taskfile/is/located/ % task
create_lambda_name && task
update_lambda_environment

# how to test the lambda
/path/where/Testing_Taskfile/is/located/ % task
run_test_event_1

{
    "StatusCode": 200, "ExecutedVersion":
"$LATEST"
},
{
  "output_bucket_name": "pycon-$USER-nlp-output-
bkt", "file_key": "email_1.txt",
  "message": "PII Redaction Pipeline worked
successfully"
}
```

Alternate Options:

- As a developer you could also choose other modes of creation of Cloud Services such as **AWS CDK**, **AWS CloudFormation** and **AWS Terraform** and others.
    - There is a bit of a learning curve involved in all the above frameworks.
- Want to know more about AWS Toolkits? Refer **aws-toolkit-aws-cli-sdk-and-cdk-6feab9e746b8**

Link to the GitHub Code

https://github.com/Toyota-Connected-India/serverless_nlp_app

II. The NLP Pipeline

- Goal of the NLP Pipeline
- Sample Input and Output
- Pipeline Architecture
- Workflow of the Pipeline `Tasks`

Broadly, there are two major types of NLP Pipelines:

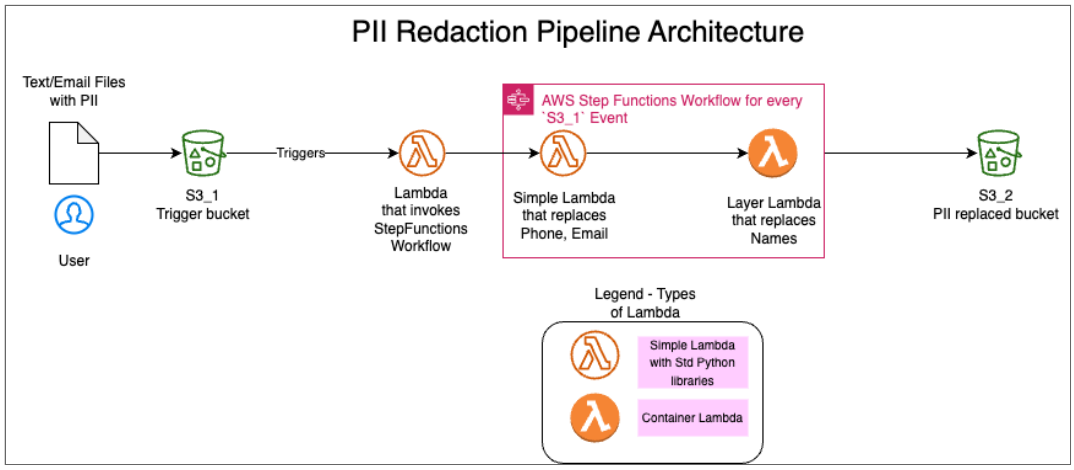
1. **NLP Pipeline in ML World**: Text pre-processing >> Feature Embeddings >> Model Training and Prediction on Numericalized Embeddings
2. **NLP Pipeline in Data Engineering World**: Transforming raw text data into useful outputs in a sequence of steps.

If an NLP Pipeline could be defined in above 2 major ways, the second definition of Data Engineering based pipeline, is what we will accomplish in this talk.

Goal of the NLP Pipeline

- The pipeline will be capable of redacting sensitive information such as email addresses, phone numbers, and names (PII) from email bodies.
    - We will use examples from Enron Email data to test the pipeline.

Pipeline Architecture



Note: This pipeline is intentionally made simple. Real-world Serverless Pipelines could be much more complex

Sample Input and Output

Sample Input

```
Date: Tue, 20 Jun 2000 06:49:00 -0700 (PDT)
From: elizabeth.sager@enron.com
To: rmadden@gibbs-bruns.com
Subject: Revised TVA letter

Bob

Here is a revised draft.  Please review and give me comments.
Thanks
Elizabeth
+1 1234567890
```

Output of Simple Lambda (phone and email redacted)

```
Date: Tue, 20 Jun 2000 06:49:00 -0700 (PDT)
From: <EMAIL>
To: <EMAIL>
Subject: Revised TVA letter

Bob

Here is a revised draft.  Please review and give me comments.
Thanks
Elizabeth
<PHONE>
```

Output of Layer Lambda (names redacted)

```
Date: Tue, 20 Jun 2000 06:49:00 -0700 (PDT)
From: <EMAIL>
To: <EMAIL>
Subject: Revised TVA letter

<NAME>

Here is a revised draft.  Please review and give me comments.
Thanks
<NAME>
<PHONE>
```

TOYOTA

Empower Sustainable Mobility and Deliver Elevated Customer Experiences

Workflow of the Tasks

1. Setting up S3 Buckets

```
- Create the S3 Trigger bucket (s3_1 in pic), intermediary S3 bucket
and Output S3 Bucket (s3_2 in pic)
```

GitHub Codes Reference

---

2. Create a "Simple Lambda"

```
- With no special/ extra packages, in a standard Py3.8 lambda env,
    - create a lambda that replaces Phone and Email and
    - test it with a sample csv file
```

GitHub Codes Reference

---

3. Create a "Layer Lambda"

- 3.A. Create a Spacy Layer inside a `amazon/aws-lambda-python:3.8` and publish as a layer
- 3.B. Create a "Layer Lambda" that identifies & replaces names using a SpaCy pre-trained model

GitHub Codes Reference

---

4. Create a StepFunctions Pipeline

- 4.A. Create a Stepfunctions State Machine Json | **GitHub Codes Reference**
- 4.B. Create a StepFunctions Invoke Lambda and Test it | **GitHub Codes Reference**

5.Test the Whole Pipeline with a simple task

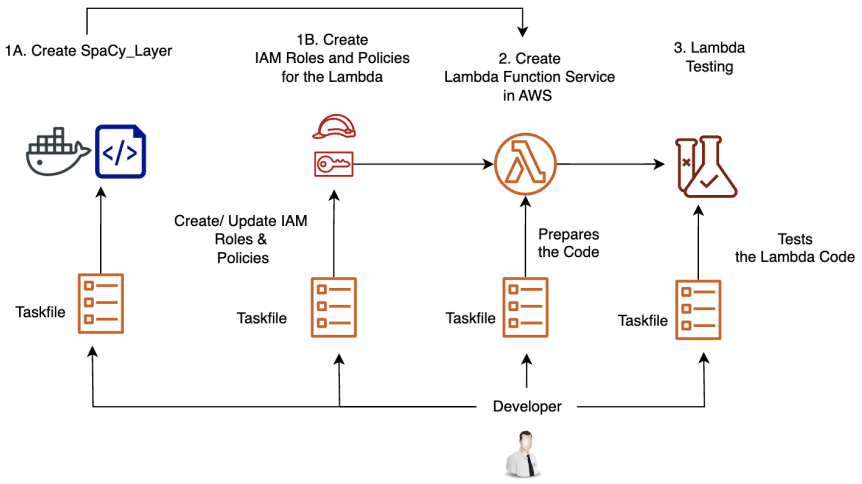
COMMAND

```
# set up the temporary AWS credentials
## task executes the task from `Taskfile.yml` in
the `/path/to/serverless_nlp_app`
/path/to/serverless_nlp_app/src/aws/2.stepfunctio
ns_invoke_lambda/c.testing  % task
run_test_event_1
```

GitHub Codes Reference

III. Infrastructure Provisioning Workflow

Step 3 - Creating a Layer Lambda from Scratch



Sample IAM `Taskfile.yml`

```yaml
version: '3'

env:
  final_lambda_policy_json_file: final_iam_policy_for_lambda.json
  template_lambda_policy_json_file: template_iam_policy_for_lambda.json
  policy_name: rws3_cloudwatch_layer_lambda
  role_name:
    sh: basename $(dirname $PWD)
  lambda_function_name:
    sh: basename $(dirname $PWD)

# get $region and $account_id from the hidden file below
# the below file is not pushed to git, but it looks like below
# region="your-region"
# account_id="account_id"

dotenv: ['.env_for_IAM_policy']

tasks:
  create_policy:
    cmds:
      - aws iam create-policy --policy-name $policy_name --policy-document
file://${final_lambda_policy_json_file}

  create_role:
    env:
      trust_policy_json: trust-policy.json
    cmds:
      - aws iam create-role --role-name $role_name --assume-role-policy-document
file://${trust_policy_json}

  attach_policy_to_role:
    cmds:
      - aws iam attach-role-policy --role-name $role_name --policy-arn
"arn:aws:iam::${account_id}:policy/${policy_name}"
```

Sample Lambda `Taskfile.yml`

```yaml
version: '3'

env:
  IAM_ROLE_ARN:
    sh: (cat ../a.create_iam_role_and_policy/final_role_arn.txt | cut -d= -f2)
  LAMBDA_FUNCTION_NAME:
    sh: basename $(dirname $PWD)
  ENVIRONMENT_JSON: environment.json

dotenv: ['.env_for_creating_lambda']


tasks:
  create_lambda_name:
    cmds:
      - zip -r ${LAMBDA_FUNCTION_NAME}.zip lambda_function.py
      - bash aws_cli_command_for_lambda_creation.bash $LAMBDA_FUNCTION_NAME $IAM_ROLE_ARN
$SPACY_LAYER
    silent: false

  update_lambda_function_code:
    cmds:
      - zip -r ${LAMBDA_FUNCTION_NAME}.zip lambda_function.py
      - aws lambda update-function-code --function-name ${LAMBDA_FUNCTION_NAME} --zip-file
fileb://./${LAMBDA_FUNCTION_NAME}.zip
    silent: false
```

IV. Conclusion

- Treat your infrastructure provisioning codes with the same rigor as your application development codes.
    - This is getting easier with `AWS CLI` + `Taskfile.yml` approach
- As many companies adopt a cloud-first approach, ensuring its developers have a `DevOps Mindset` ensures better software development cycle

Appendix: Link to the GitHub codes

https://github.com/Toyota-Connected-India/serverless_nlp_app