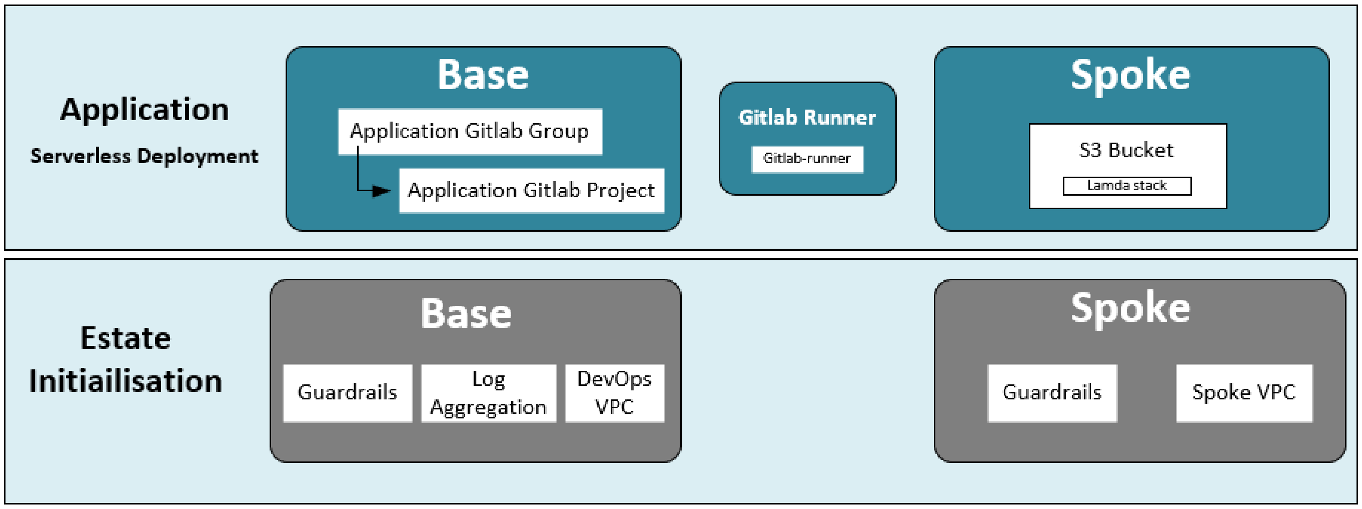
Deploy Serverless template

This guide will help you to various step to deploy the ***serverless*** template. This template follows the HUb and Spoke model. This template creates a Gitlab repository, which contains a Python ‘hello world’ script deployed to an S3 bucket in AWS, which is run via the Lambda platform. The Lambda function attaches to the API Gateway and can be called via a https request.

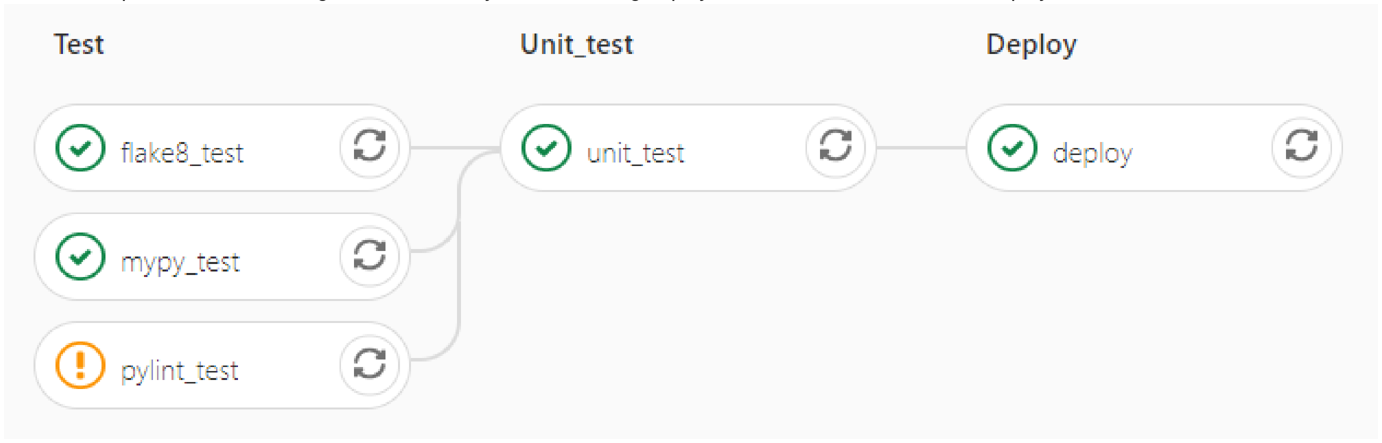
Overview

The steps include;

1. Creating an S3 bucket with a Terraform workspace
2. Create a Gitlab project to house the Lambda python application
3. Load variables in the Gitlab project to connect it to the AWS account
4. Publish the Application via the AWS account via the CICD pipeline



The CICD Pipeline includes a range of tests to verify the code being deployed. This is what a successful deployment will look like.



Infrastructure Provisioning

Step 1: Clone the template for ‘Spoke’ Infrastructure

Template repository: https://gitlab.com/mc-estate-infra/blueprints-repo/serverless-blueprint/serverless-spoke.git

- Clone the repository **your_name-serverless-spoke**
- Create a new repository to house your project
- Inside the folder execute the following commands

```
rm -rf .git
git init
git remote add origin https://gitlab.com/mc-estate-infra/spikes
/your_project/<your_application>-serverless-spoke.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Step 2: Set up Spoke Terraform Spoke Workspace

- Log in to Terraform cloud
- Create a new workspace linked to your project, name it ***your_application*-serverless-spoke**
- Configure Terraform variables

Variable	Details	Example
application_name	App Name	
bucket_name	Name of S3 bucket	Same as App Name

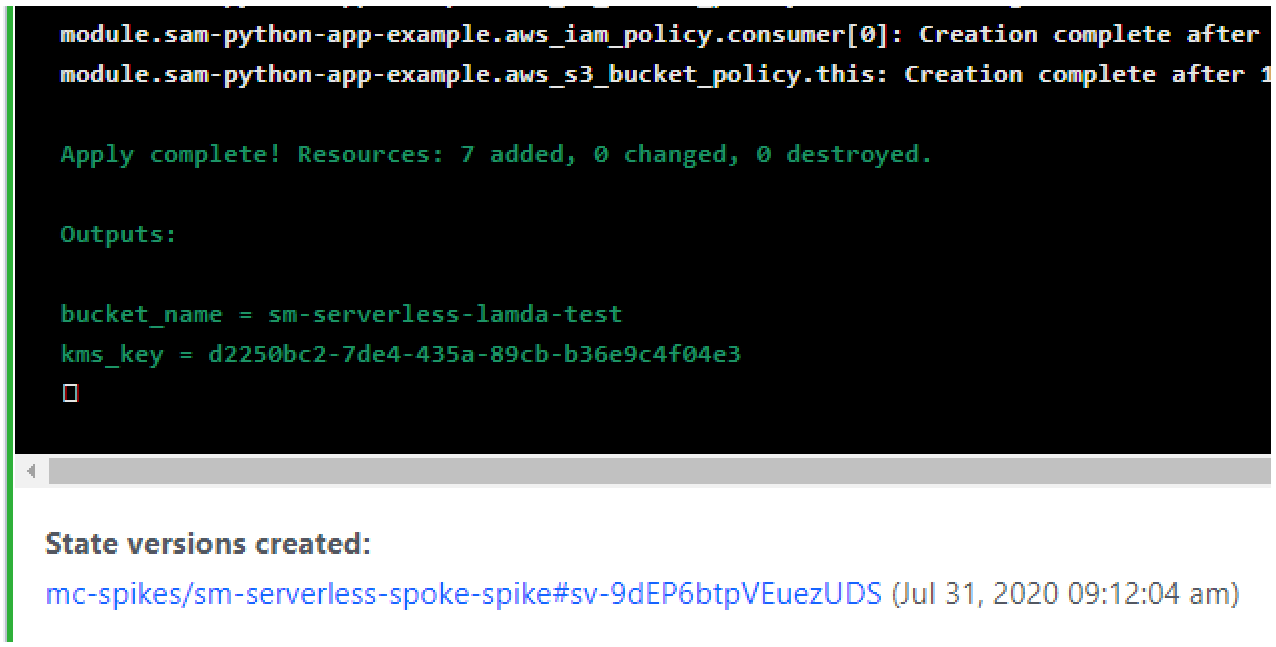
custom_tags	Accounting an tracking	`{`
		`}`

webex_teams_bearer_token	Notifications	
webex_teamId	Notifications	

aws_account	AWS Credentials	`{`
		` aws_account = “…”`
		` aws_key = “..”`
		` region = “..”`
		`}`

force_s3_destroy	Delete S3 bucket on destroy	`true`

- Set SSH key `service_terraform` in the workspace settings to pull nested dependencies from gitlab
- Queue a Terraform plan for the workspace
- Review the plan and apply changes
- After the apple an S3 bucket will be applied, note down the **bucket name,** and **kms key** for the next steps



```
module.sam-python-app-example.aws_iam_policy.consumer[0]: Creation complete after
module.sam-python-app-example.aws_s3_bucket_policy.this: Creation complete after 1

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

bucket_name = sm-serverless-lamda-test
kms_key = d2250bc2-7de4-435a-89cb-b36e9c4f04e3
```

State versions created:

mc-spikes/sm-serverless-spoke-spike#sv-9dEP6btpVEuezUDS (Jul 31, 2020 09:12:04 am)

Load all the variables, and apple the Workspace. Take note of the **bucket_name** and **kms_key** for the project deployment.

Step 3: Clone the Template for 'Base'

Template Repository: https://gitlab.com/mc-estate-infra/blueprints-repo/serverless-blueprint/serverless-base.git

- Clone the repository **your_name-serverless-spoke**
- Create a new repository to house your project
- Inside the folder execute the following commands

```
rm -rf .git
git init
git remote add origin https://gitlab.com/mc-estate-infra/spikes
/your_project/<your_application>-serverless-base.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Step 4: Set up Spoke Terraform Base Workspace

- Log in to Terraform cloud
- Create a new workspace linked to your project, name it ***your_application*-serverless-base**
- Configure Terraform variables

Variable	Details	Example
application_name	Used to create project name	
gitlab_token	Your API token from Gitlab	
parent_group_id	Location for project	`8041986` (this is spike group)

- Set SSH key `service_terraform` in the workspace settings to pull nested dependencies from gitlab
- Queue a Terraform plan for the workspace
- Review the plan and apply changes
- After the apple an S3 bucket will be applied, note down the **gitlab_application_url,** this is where the project is located.

Step 5: Load Variables into Gitlab Project

In the Gitlab Project, under;

Settings CI / CD Expand 'Variables'

Create these Variables

Variable	Details	Example
APP_NAME	Output from Step 2	Keep Bucket and App_Name the same
AWS_ACCESS_KEY_ID	From your account	
AWS_DEFAULT_REGION	AWS Region	ap-southeast-2
AWS_SECRET_ACCESS_KEY	From your account	
KMS_KEY	Output from Step 2	

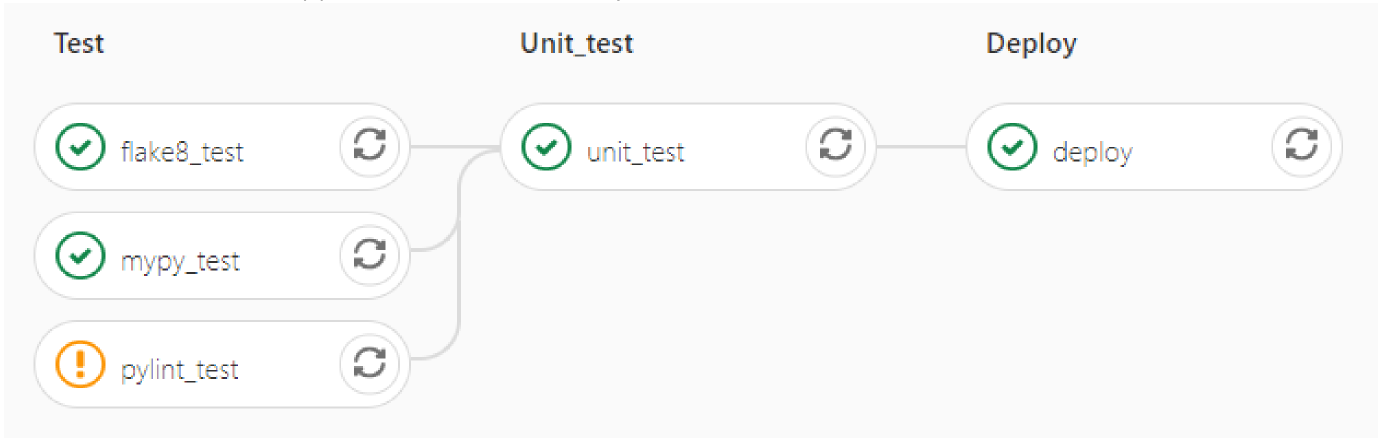
Step 6: Clone the Application

Template Repository: https://gitlab.com/mc-estate-infra/blueprints-repo/serverless-blueprint/serverless-app-lambda-python.git

- Clone the repository **your_name-serverless-app**
- Create a new repository to house your project
- Inside the folder execute the following commands, publish to the output from Step 4

```
rm -rf .git
git init
git remote add origin https://gitlab.com/min-au-apps/spikes/your_project
/<your_application>-serverless-app.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Once Published, check that the pipeline has been run successfully.


Test
Unit_test
Deploy
flake8_test
mypy_test
pylint_test
unit_test
deploy