

# ZenJob Assessment - RAMKANNAN SUBRAMANIAN

## ZenJob Assessment - RAMKANNAN SUBRAMANIAN

[Git Repo Location - LINK](#)

[Steps](#)

[Files](#)

[Dockerfile →](#)

[Python script →](#)

[YAML Files →](#)

[S3 terraform Files →](#)

[EKS terraform Files →](#)

[Improvements in the existing System](#)

## [Git Repo Location - LINK](#)

### Steps

1. Run the terraform script to create a s3 bucket and eks cluster.

[NOTE - we cannot use caps for creating S3 buckets. Hence, "qa-RAMKANNAN-S-platform-challenge" is not a possible naming convention for creating S3 Bucket.]

    terraform init

    terraform plan

    terraform apply -auto-approve

2. Make sure to check if the terraform script has added a LifeCycle Rule that will make sure files are maintained only for 24hrs.

Review transition and expiration actions	
Current version actions	Noncurrent versions actions
Day 0	Day 0
• Objects uploaded	• Objects become noncurrent
↓	↓
Day 1	Day 1
• Objects expire	• Objects are permanently deleted

3. Clone the code from git repo → [LINK](#)
4. Connect to the EKS cluster setup in AWS.

- Run the kubectl cmd to create namespaces and cronjobs respectively :-

```
kubectl apply -f namespaces.yaml
kubectl get ns
kubectl apply -f cron-k8s-qa.yaml -n qa
kubectl apply -f cron-k8s-staging.yaml -n staging
kubectl get cronjob zenjob-pythonapp -n <env>
```

- Wait for 5 min and check the “LAST SCHEDULE” value.

- Refresh the s3 bucket of the respective environment to find the file uploaded along with timestamp.

## Files

**Dockerfile** →

The Dockerfile here runs the python app after installing boto3. The dynamic environment name is passed when we run the image as a container or in a kubernetes job.

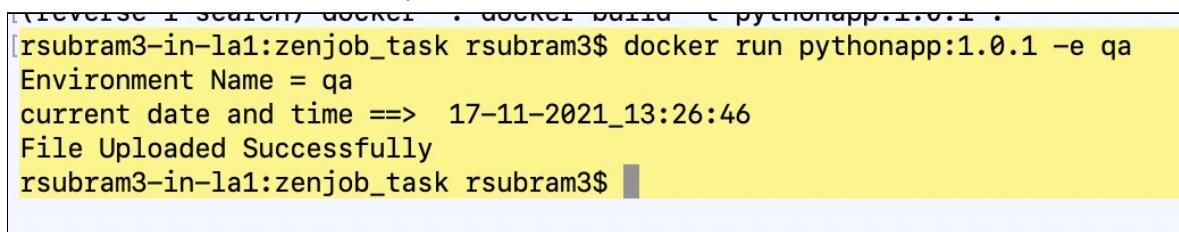
Docker build is running fine and tested locally.



```
Last login: Wed Nov 17 18:53:06 on ttys001
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
rsubram3-in-la1:zenjob_task rsubram3$ docker build -t pythonapp:1.0.1 .
[+] Building 38.8s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 149B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3
=> [1/4] FROM docker.io/library/python:3
=> [internal] load build context
=> => transferring context: 2.88kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY .
=> [4/4] RUN pip install boto3
=> exporting to image
=> => exporting layers
=> => writing image sha256:c28794df287c0945260db25a3d0910f74365165930fedb69af560c3bb4ef6b65
=> => naming to docker.io/library/pythonapp:1.0.1

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
rsubram3-in-la1:zenjob_task rsubram3$
```

Use docker Run to test if the file uploaded to s3 from local.



```
[rsubram3-in-la1:zenjob_task rsubram3$ docker run pythonapp:1.0.1 -e qa
Environment Name = qa
current date and time ==> 17-11-2021_13:26:46
File Uploaded Successfully
rsubram3-in-la1:zenjob_task rsubram3$
```

Amazon S3 > qa-ramkannan-s-platform-challenge

## qa-ramkannan-s-platform-challenge [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

**Objects (1)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Copy S3 URI](#)  [Copy URL](#)  [Download](#)  [Open](#) [Delete](#) [Actions ▾](#) [Create folder](#)

[Upload](#)

Find objects by prefix

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">zenjobfile_17-11-2021_13:26:46.txt</a>	txt	November 17, 2021, 18:56:50 (UTC+05:30)	34.0 B	Standard

Python script →

This python script uses boto3 library to Upload file to s3 bucket. The timestamp is added to the filename which is uploaded to s3..

```
rsubram3-in-la1:zenjob_task rsubram3$ python3 file_upload_s3.py -e qa
Environment Name = qa
current date and time ==> 17-11-2021_18:58:37
File Uploaded Successfully
rsubram3-in-la1:zenjob_task rsubram3$
```

Amazon S3 > qa-ramkannan-s-platform-challenge

qa-ramkannan-s-platform-challenge [Info](#)

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[C](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#)

[Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">zenjobfile_17-11-2021_13:26:46.txt</a>	txt	November 17, 2021, 18:56:50 (UTC+05:30)	34.0 B	Standard
<input type="checkbox"/>	<a href="#">zenjobfile_17-11-2021_18:58:37.txt</a>	txt	November 17, 2021, 18:58:42 (UTC+05:30)	34.0 B	Standard

## YAML Files →

Added 2 yaml files in which the difference is only the “env name” (i.e qa/staging).

The Yaml file is a cron job which will run every 5min in kubernetes. This is tested in my kubernetes cluster and below is the output.

```
rsubram3-in-la1:zenjob_task rsubram3$ kubectl apply -f cron-k8s-staging.yaml -n staging
cronjob.batch/zenjob-pythonapp created
rsubram3-in-la1:zenjob_task rsubram3$ kubectl get cronjob zenjob-pythonapp -n staging
NAME          SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
zenjob-pythonapp  */5 * * * *  False       0        <none>    19s
rsubram3-in-la1:zenjob_task rsubram3$
```

Additionally, I have created namespaces.yaml which will create a namespace in kubernetes cluster if not present.

## S3 terraform Files →

the terraform files for s3 bucket creation are placed inside this folder. It is a direct tf file with hard-coded secret key values and here it created 2 buckets by reading the list.

```

~/git_repos/zenjob_task/s3_tf -- bash
}

# aws_s3_bucket.buckets_cleanup[1] will be created
+ resource "aws_s3_bucket" "buckets_cleanup" {
  + acceleration_status      = (known after apply)
  + acl                      = "private"
  + arn                      = (known after apply)
  + bucket                   = "staging-ramkannan-s-platform-challenge"
  + bucket_domain_name       = (known after apply)
  + bucketRegionalDomainName = (known after apply)
  + force_destroy            = true
  + hostedZoneId             = (known after apply)
  + id                       = (known after apply)
  + region                   = (known after apply)
  + requestPayer              = (known after apply)
  + tags_all                 = (known after apply)
  + websiteDomain             = (known after apply)
  + websiteEndpoint           = (known after apply)

  + lifecycle_rule {
    + enabled = true
    + id      = "clean-up-1-day"

    + expiration {
      + days = 1
    }

    + noncurrentVersionExpiration {
      + days = 1
    }
  }

  + versioning {
    + enabled    = (known after apply)
    + mfaDelete = (known after apply)
  }
}
}

Plan: 2 to add, 0 to change, 0 to destroy.

```

Even performed a terraform apply for creating these buckets and its successful along with its LifeCycle Rule configured :-

```

~/git_repos/zenjob_task/s3_tf --bash

+ hosted_zone_id          = (known after apply)
+ id                      = (known after apply)
+ region                  = (known after apply)
+ request_payer           = (known after apply)
+ tags_all                = (known after apply)
+ website_domain          = (known after apply)
+ website_endpoint         = (known after apply)

+ lifecycle_rule {
  + enabled = true
  + id      = "clean-up-1-day"

  + expiration {
    + days = 1
  }

  + noncurrent_version_expiration {
    + days = 1
  }
}

+ versioning {
  + enabled   = (known after apply)
  + mfa_delete = (known after apply)
}
}

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_s3_bucket.buckets_cleanup[1]: Creating...
aws_s3_bucket.buckets_cleanup[0]: Creating...
aws_s3_bucket.buckets_cleanup[1]: Still creating... [10s elapsed]
aws_s3_bucket.buckets_cleanup[0]: Still creating... [10s elapsed]
aws_s3_bucket.buckets_cleanup[1]: Creation complete after 20s [id=staging-ramkannan-s-platform-challenge]
aws_s3_bucket.buckets_cleanup[0]: Creation complete after 20s [id=qa-ramkannan-s-platform-challenge]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
rsubram3-in-la1:s3_tf rsubram3$ 

```

Buckets (2) <a href="#">Info</a>		<a href="#">C</a>	<a href="#">Copy ARN</a>	<a href="#">Empty</a>	<a href="#">Delete</a>	<a href="#">Create bucket</a>
Buckets are containers for data stored in S3. <a href="#">Learn more</a>						
<input type="text"/> <a href="#">Find buckets by name</a> <span style="float: right;">◀ 1 ▶ ⚙</span>						
Name	AWS Region	Access	Creation date			
qa-ramkannan-s-platform-challenge	US West (Oregon) us-west-2	Objects can be public	November 18, 2021, 13:13:05 (UTC+05:30)			
staging-ramkannan-s-platform-challenge	US West (Oregon) us-west-2	Objects can be public	November 18, 2021, 13:13:05 (UTC+05:30)			

Lifecycle rules (1)				
Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class or deleting objects after a specified period of time. <a href="#">Learn more</a>				
<a href="#">View details</a>		<a href="#">Edit</a>	<a href="#">Delete</a>	<a href="#">Actions ▾</a>
Lifecycle rule name	Status	Scope	Current version actions	Noncurrent versions actions
<a href="#">clean-up-1-day</a>	 Enabled	Entire bucket	Expires	Permanently delete
<a href="#">View lifecycle configuration</a>				

Review transition and expiration actions	
Current version actions	Noncurrent versions actions
Day 0 <ul style="list-style-type: none"><li>Objects uploaded</li></ul> ↓	Day 0 <ul style="list-style-type: none"><li>Objects become noncurrent</li></ul> ↓
Day 1 <ul style="list-style-type: none"><li>Objects expire</li></ul>	Day 1 <ul style="list-style-type: none"><li>Objects are permanently deleted</li></ul>

## EKS terraform Files →

This uses the "[Terraform module](#) to create an Elastic Kubernetes (EKS) cluster and associated worker instances on AWS". The terraform plan works fine but the vpc and subnet ids has to updated with respect to the AWS account.

~/git\_repos/zenjob\_task/eks\_cluster — -bash

```
# module.eks.aws_eks_cluster.this[0] will be created
+ resource "aws_eks_cluster" "this" {
+   arn          = (known after apply)
+   certificate_authority = (known after apply)
+   created_at    = (known after apply)
+   endpoint      = (known after apply)
+   id           = (known after apply)
+   identity     = (known after apply)
+   name         = "zenjob_cluster"
+   platform_version = (known after apply)
+   role_arn     = (known after apply)
+   status        = (known after apply)
+   tags_all     = (known after apply)
+   version       = "1.21"

+   kubernetes_network_config {
+     service_ipv4_cidr = (known after apply)
+   }

+   timeouts {
+     create = "30m"
+     delete = "15m"
+     update = "60m"
+   }

+   vpc_config {
+     cluster_security_group_id = (known after apply)
+     endpoint_private_access  = false
+     endpoint_public_access   = true
+     public_access_cidrs      = [
+       "0.0.0.0/0",
+     ]
+     security_group_ids       = (known after apply)
+     subnet_ids               = [
+       "subnet-abcd012",
+       "subnet-bcde012a",
+       "subnet-fghi345a",
+     ]
+     vpc_id                  = (known after apply)
+   }
}
```

```

~/git_repos/zenjob_task/eks_cluster -- bash
}

# module.eks.aws_launch_configuration.workers[0] will be created
+ resource "aws_launch_configuration" "workers" {
  + arn = (known after apply)
  + associate_public_ip_address = false
  + ebs_optimized = true
  + enable_monitoring = true
  + iam_instance_profile = (known after apply)
  + id = (known after apply)
  + image_id = "ami-0eb873de16468e55e"
  + instance_type = "m4.large"
  + key_name = (known after apply)
  + name = (known after apply)
  + name_prefix = "zenjob_cluster-0"
  + security_groups = (known after apply)
  + user_data_base64 = (known after apply)

  + ebs_block_device {
    + delete_on_termination = (known after apply)
    + device_name = (known after apply)
    + encrypted = (known after apply)
    + iops = (known after apply)
    + no_device = (known after apply)
    + snapshot_id = (known after apply)
    + throughput = (known after apply)
    + volume_size = (known after apply)
    + volume_type = (known after apply)
  }

  + metadata_options {
    + http_endpoint = "enabled"
    + http_put_response_hop_limit = (known after apply)
    + http_tokens = "optional"
  }
}

```

**Plan:** 24 to add, 0 to change, 0 to destroy.

#### Changes to Outputs:

+ eks\_cluster\_name = (known after apply)

## Improvements in the existing System

1. AWS SecretKey is hard-coded in python script. It can be passed as a dynamic argument by storing in Kubernetes Secrets.
2. We currently have 2 yaml files. We can consolidate into one if we run kubectl cmd from CI-CD tools like Jenkins/Spinnaker or use helm in which we can set environment name in values.yaml for instance.
3. Terraform modules to be used with proper structure & adding terraform provider for version, region, backup (s3) for storing tfstate files.