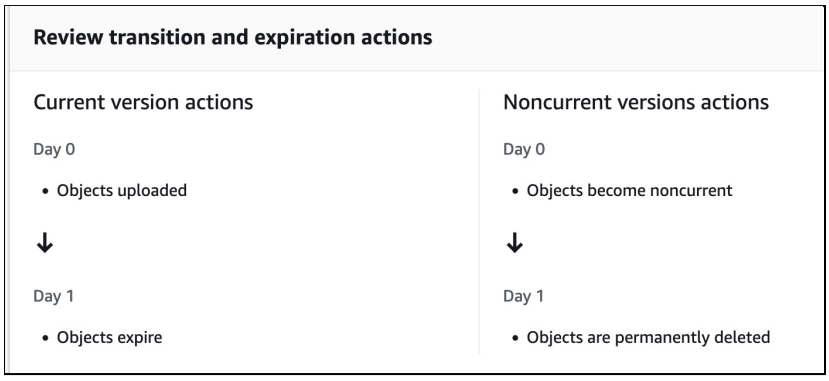
ZenJob Assessment - RAMKANNAN SUBRAMANIAN

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.



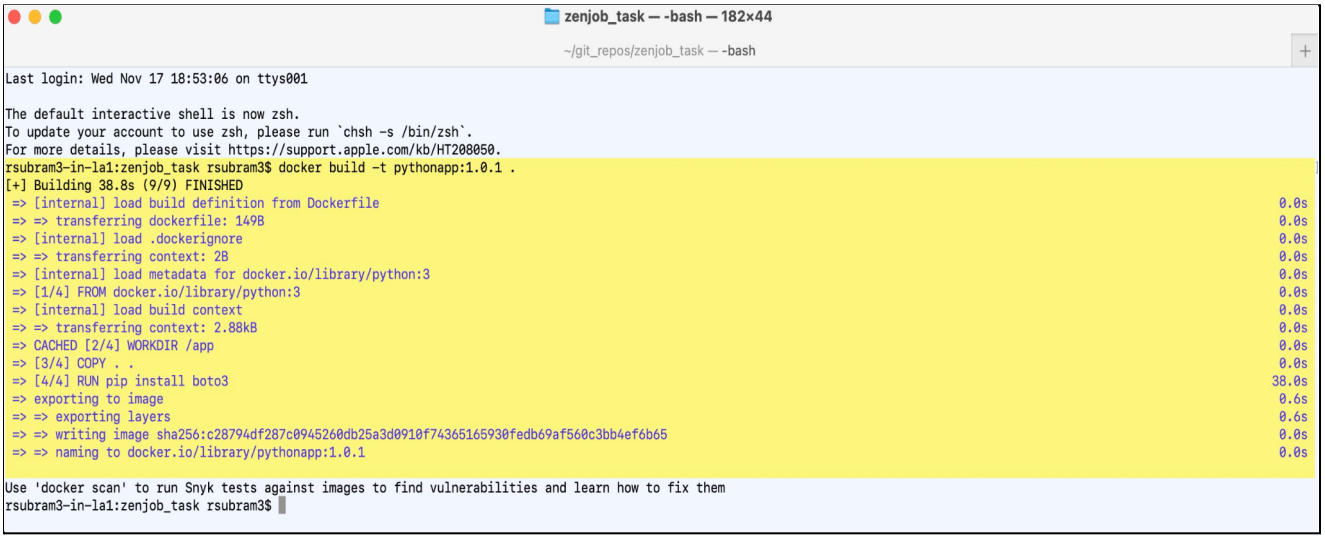
3. Clone the code from git repo ⟶ LINK
4. Connect to the EKS cluster setup in AWS.

5. Run the kubectl cmds 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>
6. Wait for 5 min and check the “LAST SCHEDULE” value.
7. Refresh the s3 bucket of the respective environment to find the file uploaded along with timestamp.

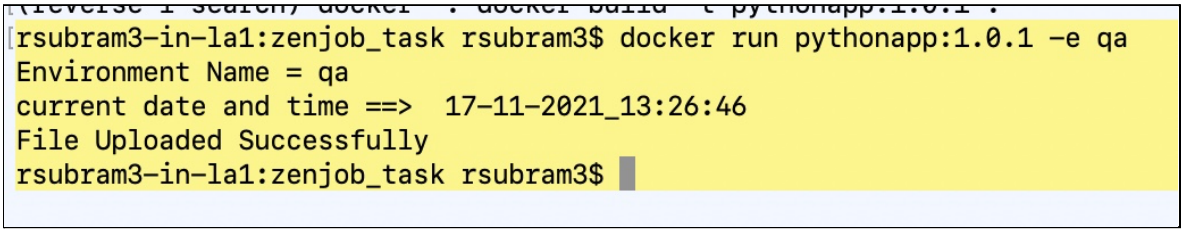
Files

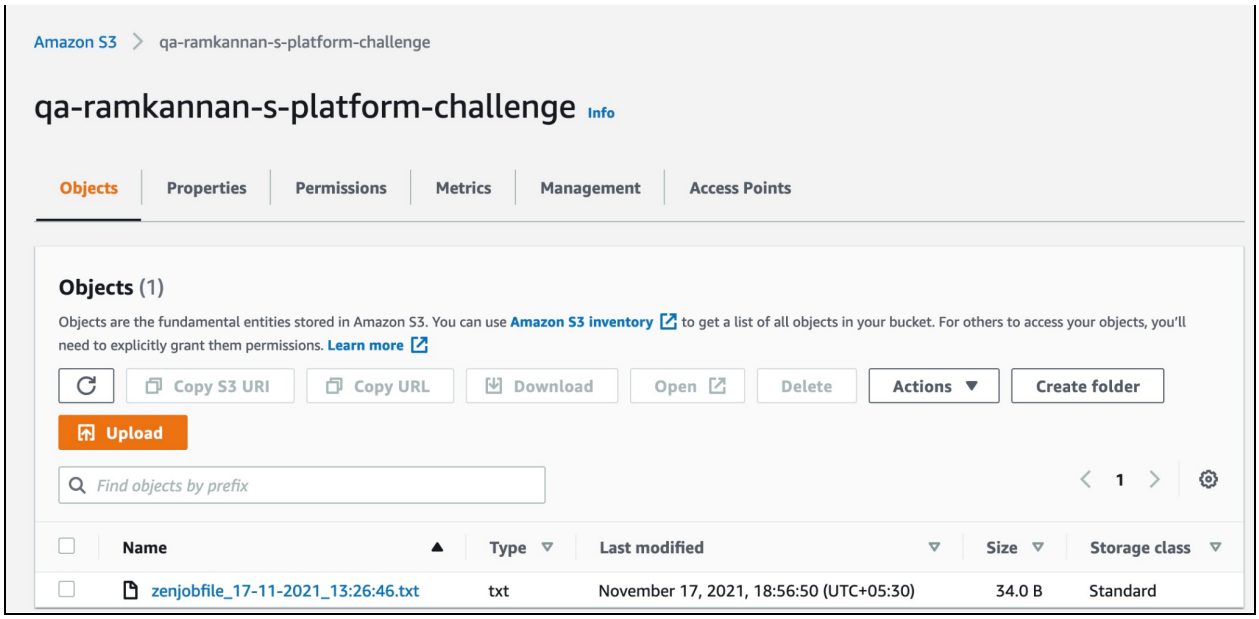
Dockerfile →

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.



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

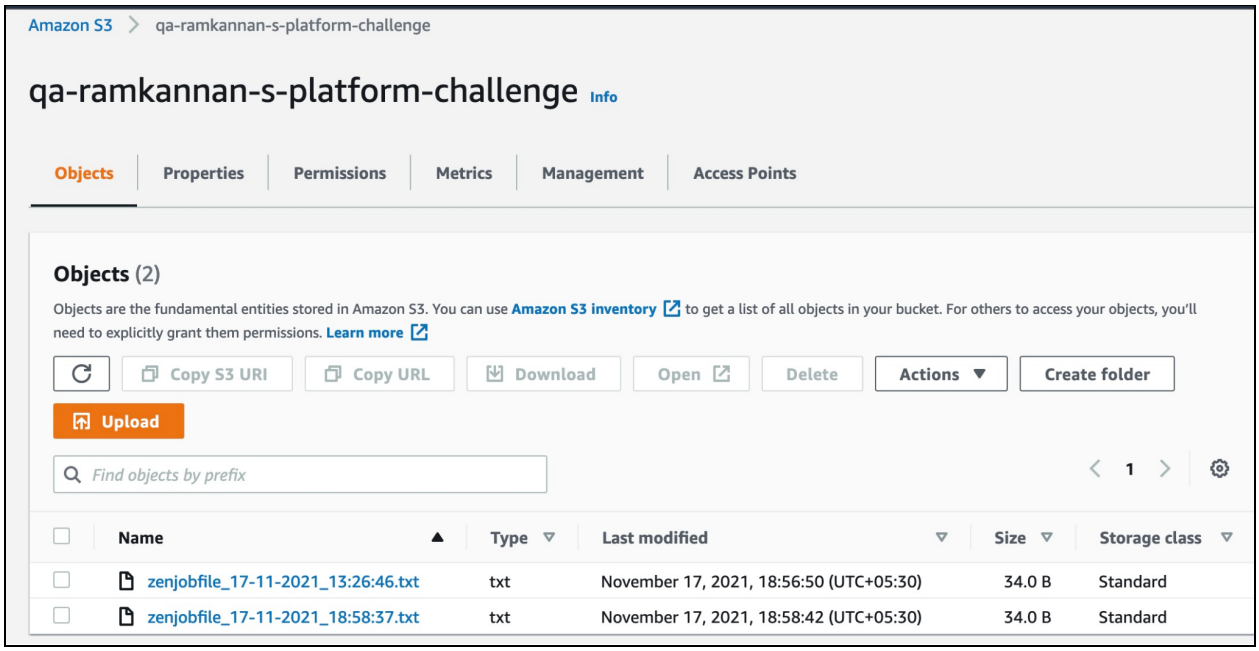




Python script →

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$
```



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 →

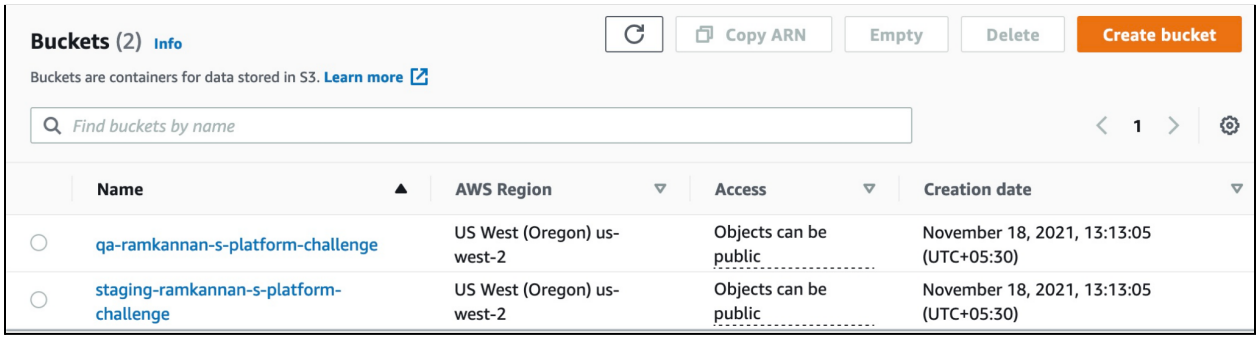
the terraform files for s3 bucket creation are placed inside this folder. It is a direct tf file with hard-coded secret key values and here it created 2 buckets by reading the list.

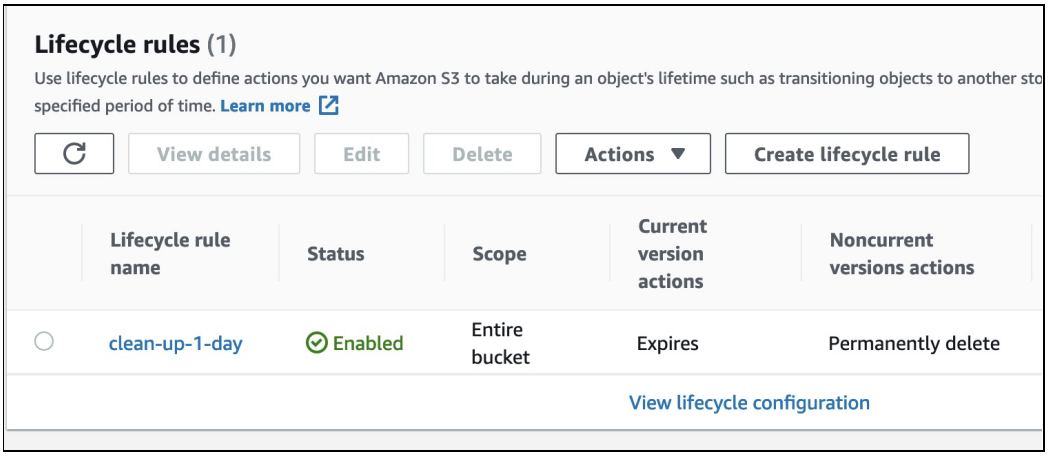
```
~/git_repos/zenjob_task/s3_tf — -bash
  }

  # 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)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = true
      + 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.
```

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$
```



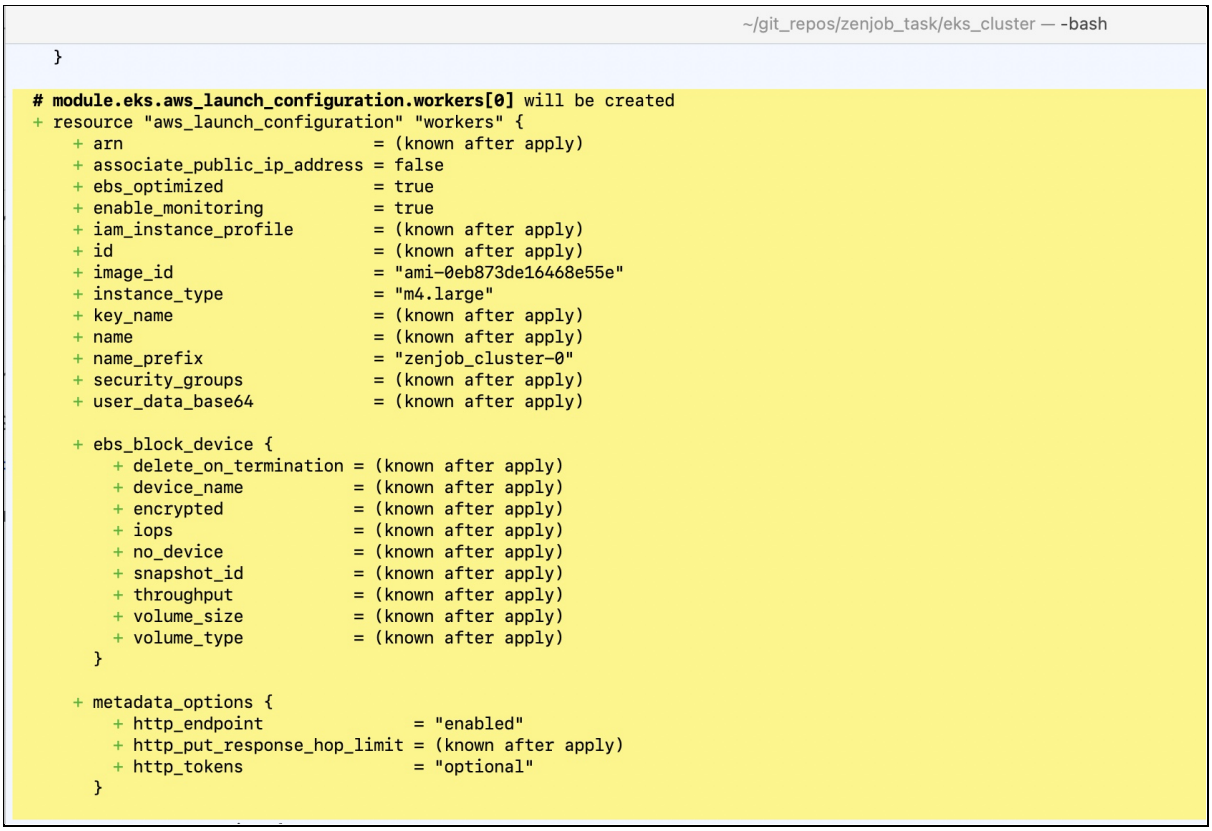


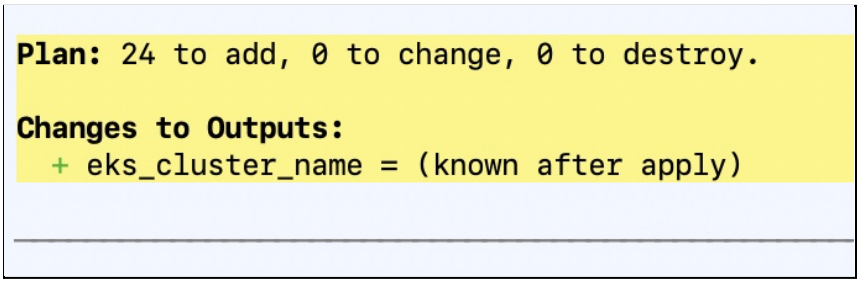
EKS terraform Files →

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-abcde012",
            + "subnet-bcde012a",
            + "subnet-fghi345a",
          ]
        + vpc_id                    = (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.