UDACITY

Project 4

Refactor Udagram App into Microservices and Deploy

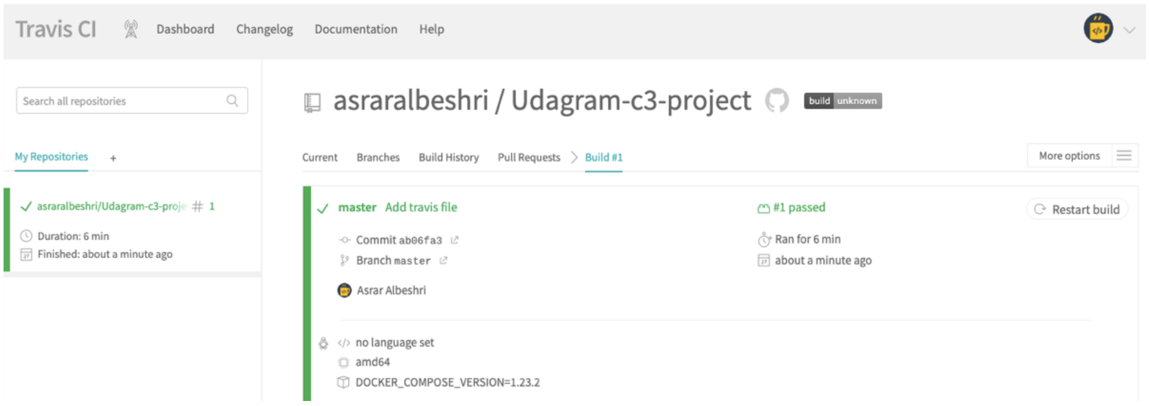
Prepared by:

Asrar Meshal Albeshri

Cloud Developer Nanodegree program

Project Screenshots

1- Travis build result



Travis CI · Dashboard · Changelog · Documentation · Help

Search all repositories

My Repositories +

✓ asraralbeshri/Udagram-c3-proje # 1
Duration: 6 min
Finished: about a minute ago

asraralbeshri / Udagram-c3-project build unknown

Current · Branches · Build History · Pull Requests > Build #1 · More options

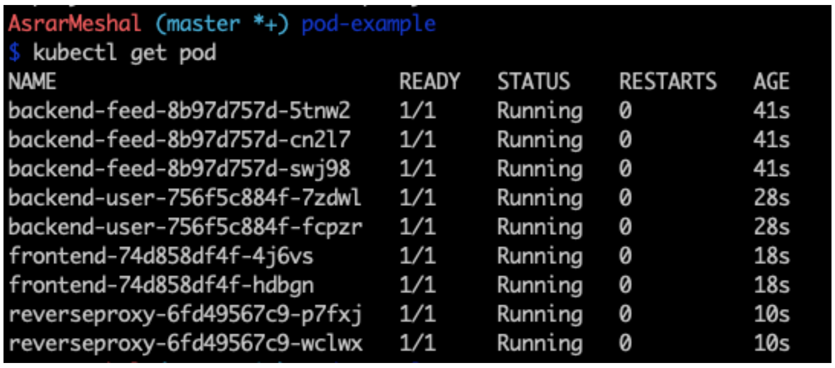
✓ master Add travis file
Commit ab06fa3
Branch master
Asrar Albeshri

#1 passed
Ran for 6 min
about a minute ago

Restart build

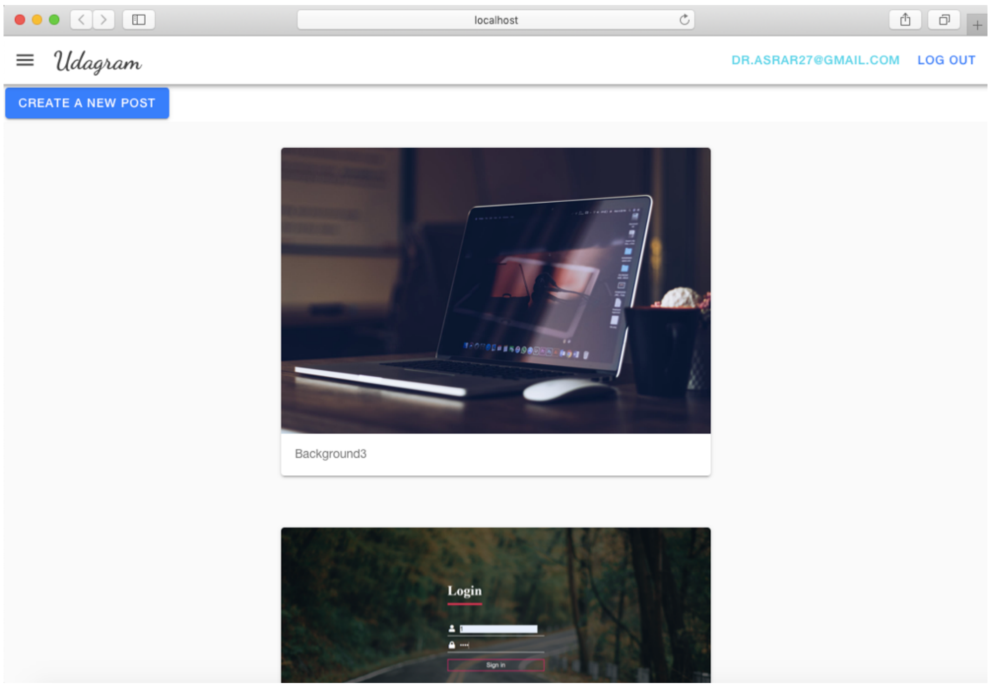
no language set
amd64
DOCKER_COMPOSE_VERSION=1.23.2

2- Cluster Pods

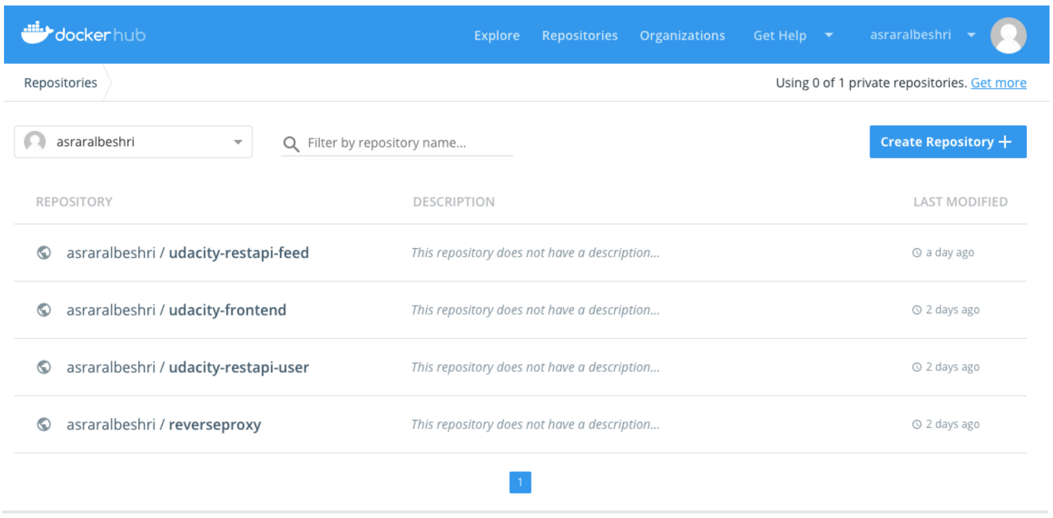


```
AsrarMeshal (master *+) pod-example
$ kubectl get pod
NAME                              READY   STATUS    RESTARTS   AGE
backend-feed-8b97d757d-5tnw2      1/1     Running   0          41s
backend-feed-8b97d757d-cn2l7      1/1     Running   0          41s
backend-feed-8b97d757d-swj98      1/1     Running   0          41s
backend-user-756f5c884f-7zdwl     1/1     Running   0          28s
backend-user-756f5c884f-fcpzr     1/1     Running   0          28s
frontend-74d858df4f-4j6vs         1/1     Running   0          18s
frontend-74d858df4f-hdbgn         1/1     Running   0          18s
reverseproxy-6fd49567c9-p7fxj     1/1     Running   0          10s
reverseproxy-6fd49567c9-wclwx     1/1     Running   0          10s
```

3- Application home page



4- Docker hub images

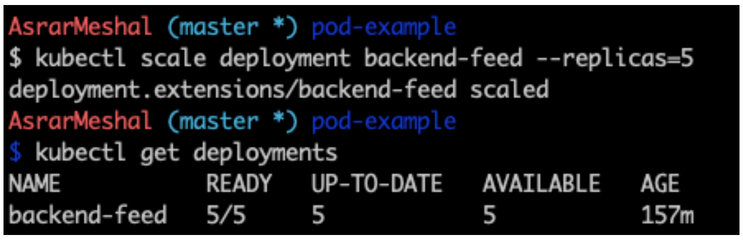


REPOSITORY	DESCRIPTION	LAST MODIFIED
asraralbeshri / udacity-restapi-feed	This repository does not have a description...	a day ago
asraralbeshri / udacity-frontend	This repository does not have a description...	2 days ago
asraralbeshri / udacity-restapi-user	This repository does not have a description...	2 days ago
asraralbeshri / reverseproxy	This repository does not have a description...	2 days ago

Some details about the project

Scaling the deployment

I scale feed deployment from 3 to 5 (as shown in the following picture).

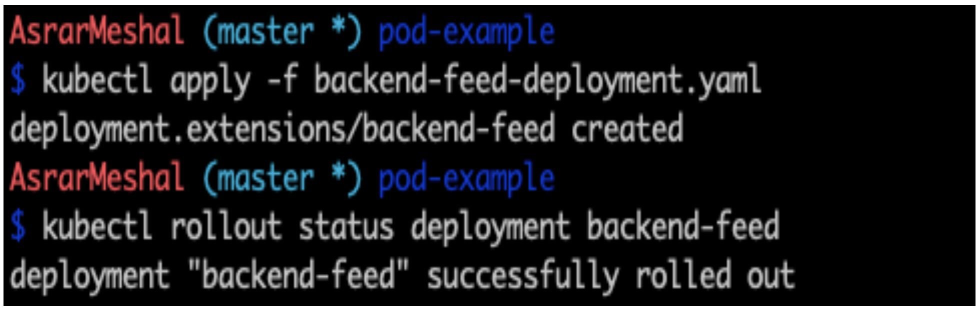


```
AsrarMeshal (master *) pod-example
$ kubectl scale deployment backend-feed --replicas=5
deployment.extensions/backend-feed scaled
AsrarMeshal (master *) pod-example
$ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
backend-feed   5/5     5            5           157m
```

Rolling update and roll back

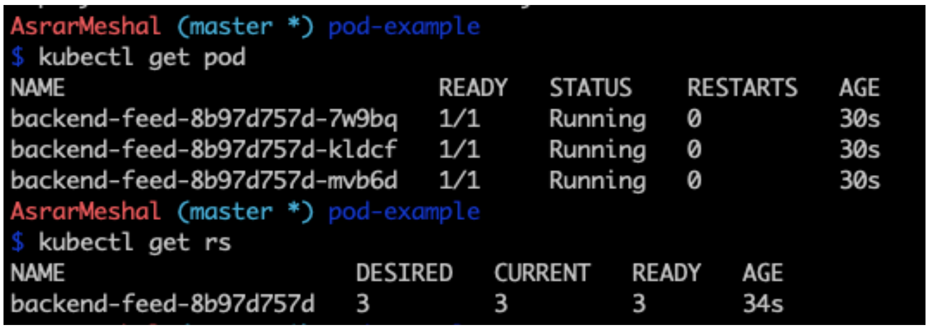
I will explain this section with feed deployment, for example.

1- Creates backend feed deployment and rollout it.



```
AsrarMeshal (master *) pod-example
$ kubectl apply -f backend-feed-deployment.yaml
deployment.extensions/backend-feed created
AsrarMeshal (master *) pod-example
$ kubectl rollout status deployment backend-feed
deployment "backend-feed" successfully rolled out
```

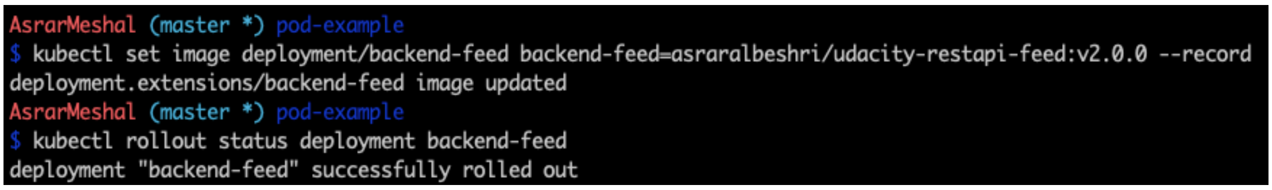
2- Displays the output of replicas (rs) and pods.



```
AsrarMeshal (master *) pod-example
$ kubectl get pod
NAME                             READY   STATUS    RESTARTS   AGE
backend-feed-8b97d757d-7w9bq     1/1     Running   0          30s
backend-feed-8b97d757d-kldcf     1/1     Running   0          30s
backend-feed-8b97d757d-mvb6d     1/1     Running   0          30s
AsrarMeshal (master *) pod-example
$ kubectl get rs
NAME                     DESIRED   CURRENT   READY   AGE
backend-feed-8b97d757d   3         3         3       34s
```

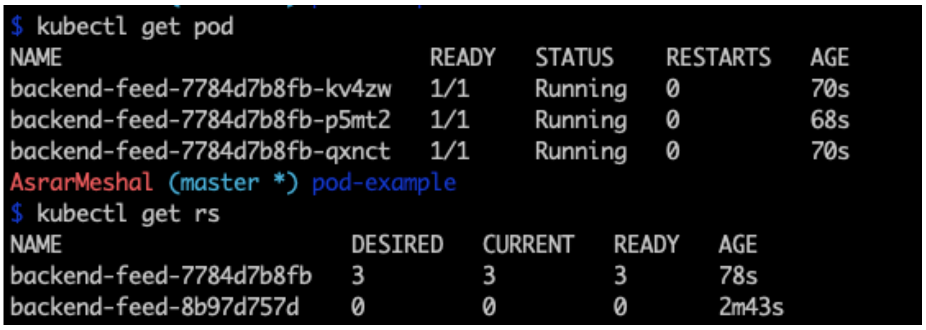
After that, I update the deployment and display the rolling update status.

1- Update image by using the following command and see rollout status



```
AsrarMeshal (master *) pod-example
$ kubectl set image deployment/backend-feed backend-feed=asraralbeshri/udacity-restapi-feed:v2.0.0 --record
deployment.extensions/backend-feed image updated
AsrarMeshal (master *) pod-example
$ kubectl rollout status deployment backend-feed
deployment "backend-feed" successfully rolled out
```

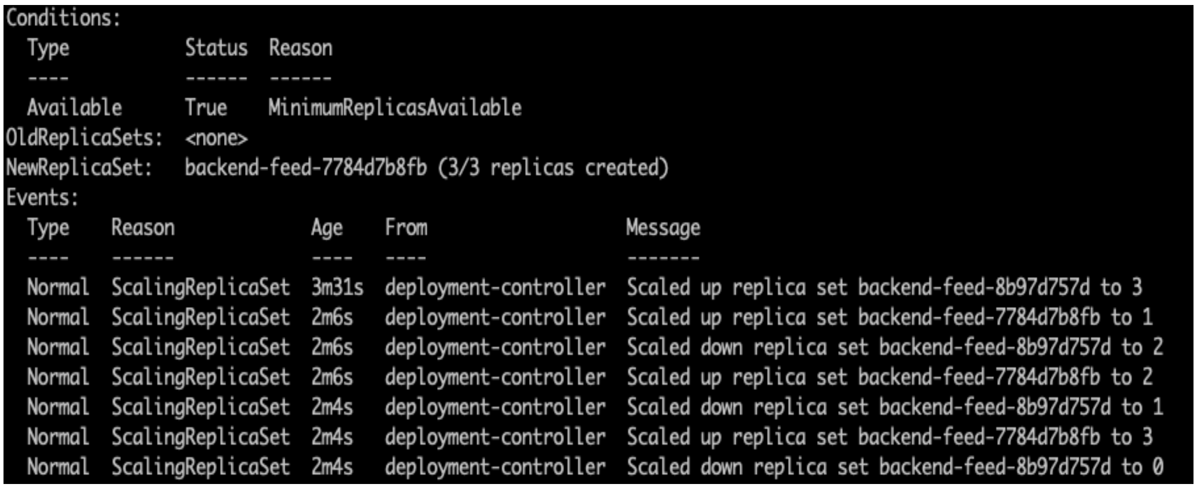
2- Displays rs and pods



```
$ kubectl get pod
NAME                             READY   STATUS    RESTARTS   AGE
backend-feed-7784d7b8fb-kv4zw    1/1     Running   0          70s
backend-feed-7784d7b8fb-p5mt2    1/1     Running   0          68s
backend-feed-7784d7b8fb-qxnct    1/1     Running   0          70s
AsrarMeshal (master *) pod-example
$ kubectl get rs
NAME                      DESIRED   CURRENT   READY   AGE
backend-feed-7784d7b8fb   3         3         3       78s
backend-feed-8b97d757d    0         0         0       2m43s
```

3- Identifies if the rolling update is working correctly by using the following command.

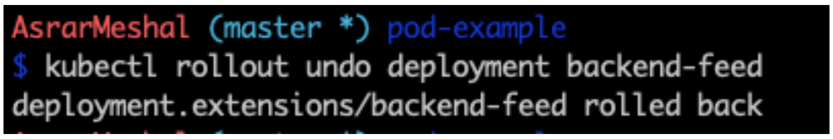
kubectl describe deployments



```
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   backend-feed-7784d7b8fb (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m31s  deployment-controller  Scaled up replica set backend-feed-8b97d757d to 3
  Normal  ScalingReplicaSet  2m6s   deployment-controller  Scaled up replica set backend-feed-7784d7b8fb to 1
  Normal  ScalingReplicaSet  2m6s   deployment-controller  Scaled down replica set backend-feed-8b97d757d to 2
  Normal  ScalingReplicaSet  2m6s   deployment-controller  Scaled up replica set backend-feed-7784d7b8fb to 2
  Normal  ScalingReplicaSet  2m4s   deployment-controller  Scaled down replica set backend-feed-8b97d757d to 1
  Normal  ScalingReplicaSet  2m4s   deployment-controller  Scaled up replica set backend-feed-7784d7b8fb to 3
  Normal  ScalingReplicaSet  2m4s   deployment-controller  Scaled down replica set backend-feed-8b97d757d to 0
```

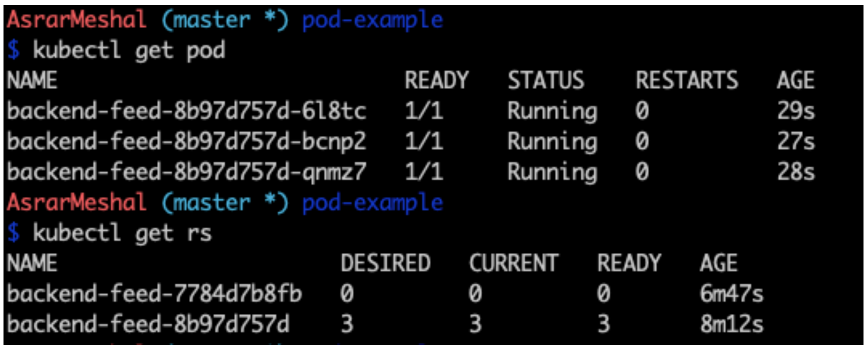
Finally, rolling back to the previous version using the following steps:

1- Use the following command to roll back to the previous version.



```
AsrarMeshal (master *) pod-example
$ kubectl rollout undo deployment backend-feed
deployment.extensions/backend-feed rolled back
```

2- Displays rs and pods to check the deployment is rolling back successfully.

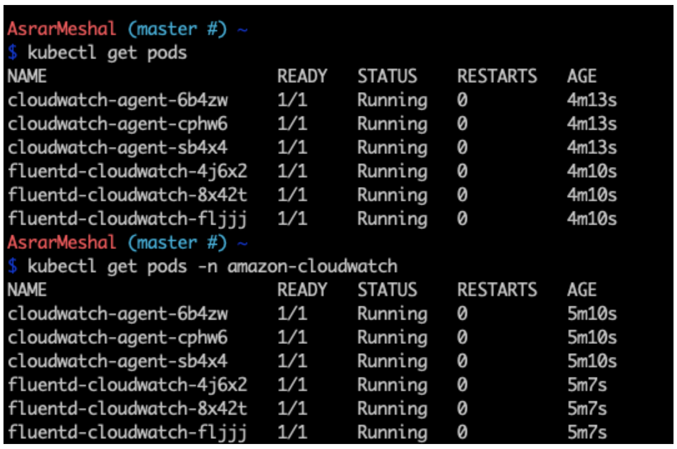


```
AsrarMeshal (master *) pod-example
$ kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
backend-feed-8b97d757d-6l8tc    1/1     Running   0          29s
backend-feed-8b97d757d-bcnp2    1/1     Running   0          27s
backend-feed-8b97d757d-qnmz7    1/1     Running   0          28s
AsrarMeshal (master *) pod-example
$ kubectl get rs
NAME                       DESIRED   CURRENT   READY   AGE
backend-feed-7784d7b8fb    0         0         0       6m47s
backend-feed-8b97d757d     3         3         3       8m12s
```

AWS CloudWatch service

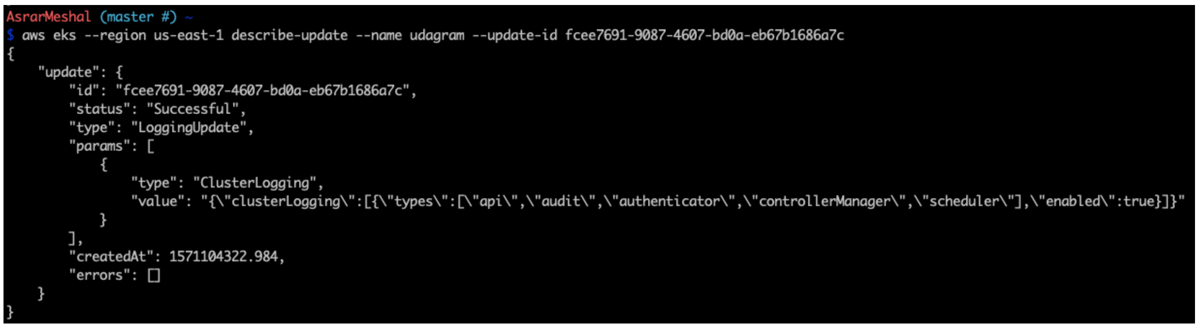
I use CloudWatch service to monitor the cluster.

First, I will create a namespace for this service named amazon-Cloudwatch, then I configure fluentd to collect logs and sent it to AWS Cloudwatch.



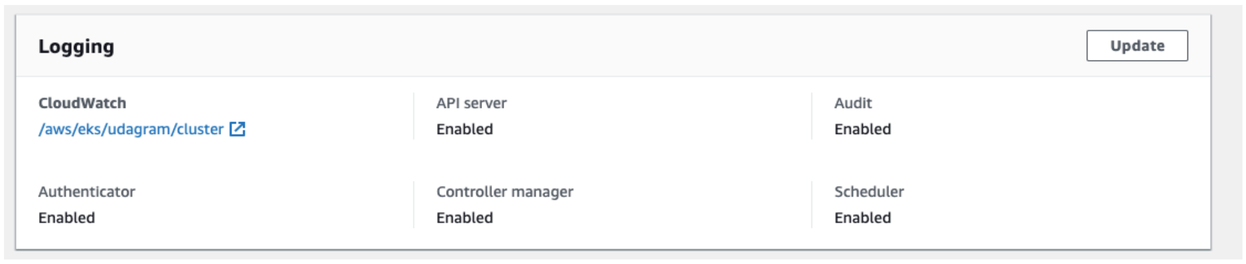
```
AsrarMeshal (master #) ~
$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
cloudwatch-agent-6b4zw     1/1     Running   0          4m13s
cloudwatch-agent-cphw6     1/1     Running   0          4m13s
cloudwatch-agent-sb4x4     1/1     Running   0          4m13s
fluentd-cloudwatch-4j6x2   1/1     Running   0          4m10s
fluentd-cloudwatch-8x42t   1/1     Running   0          4m10s
fluentd-cloudwatch-fljjj   1/1     Running   0          4m10s
AsrarMeshal (master #) ~
$ kubectl get pods -n amazon-cloudwatch
NAME                       READY   STATUS    RESTARTS   AGE
cloudwatch-agent-6b4zw     1/1     Running   0          5m10s
cloudwatch-agent-cphw6     1/1     Running   0          5m10s
cloudwatch-agent-sb4x4     1/1     Running   0          5m10s
fluentd-cloudwatch-4j6x2   1/1     Running   0          5m7s
fluentd-cloudwatch-8x42t   1/1     Running   0          5m7s
fluentd-cloudwatch-fljjj   1/1     Running   0          5m7s
```

Also, I enable Amazon EKS control plane logging, to provide audit and diagnostic logs directly from the Amazon EKS control plane to CloudWatch Logs (as shown in the following picture).



```
AsrarMeshal (master #) ~
$ aws eks --region us-east-1 describe-update --name udagram --update-id fcee7691-9087-4607-bd0a-eb67b1686a7c
{
    "update": {
        "id": "fcee7691-9087-4607-bd0a-eb67b1686a7c",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1571104322.984,
        "errors": []
    }
}
```

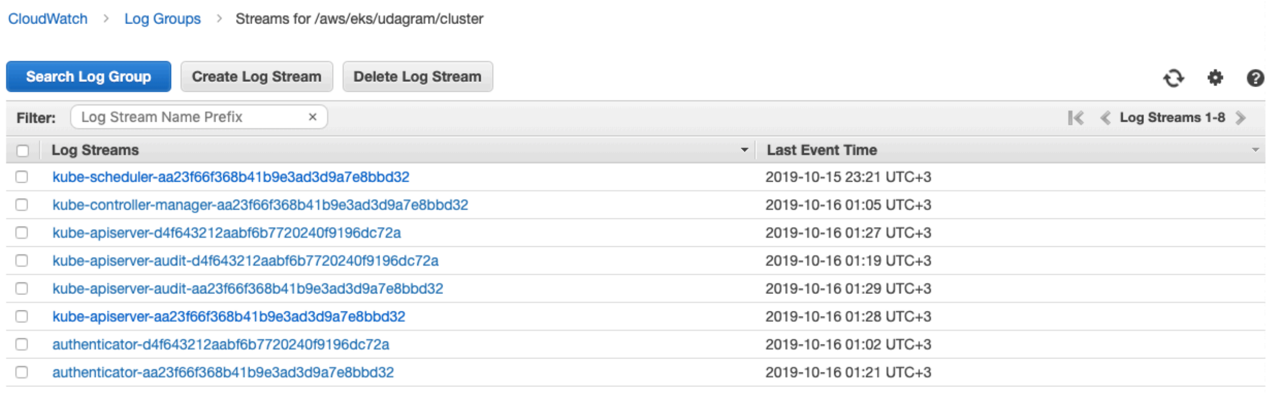
The following pictures show that the cluster logs are created and sent successfully.



Logging

Update

CloudWatch /aws/eks/udagram/cluster	API server Enabled	Audit Enabled
Authenticator Enabled	Controller manager Enabled	Scheduler Enabled



CloudWatch > Log Groups > Streams for /aws/eks/udagram/cluster

Log Streams	Last Event Time
kube-scheduler-aa23f66f368b41b9e3ad3d9a7e8bbd32	2019-10-15 23:21 UTC+3
kube-controller-manager-aa23f66f368b41b9e3ad3d9a7e8bbd32	2019-10-16 01:05 UTC+3
kube-apiserver-d4f643212aabf6b7720240f9196dc72a	2019-10-16 01:27 UTC+3
kube-apiserver-audit-d4f643212aabf6b7720240f9196dc72a	2019-10-16 01:19 UTC+3
kube-apiserver-audit-aa23f66f368b41b9e3ad3d9a7e8bbd32	2019-10-16 01:29 UTC+3
kube-apiserver-aa23f66f368b41b9e3ad3d9a7e8bbd32	2019-10-16 01:28 UTC+3
authenticator-d4f643212aabf6b7720240f9196dc72a	2019-10-16 01:02 UTC+3
authenticator-aa23f66f368b41b9e3ad3d9a7e8bbd32	2019-10-16 01:21 UTC+3