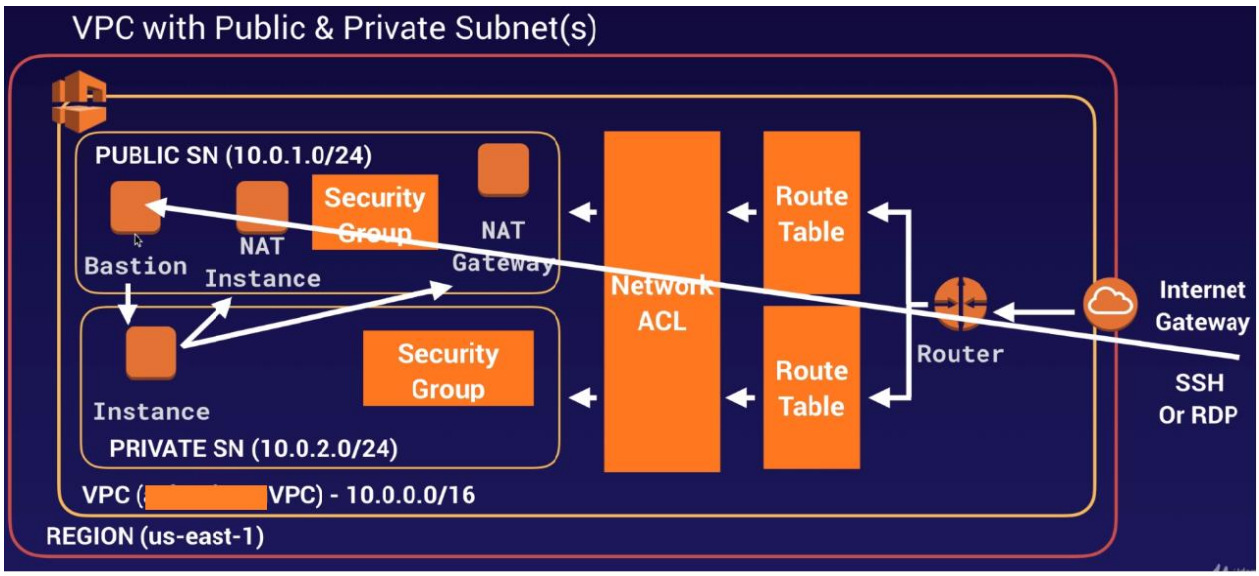
Amit Goswami
Capstone | Udacity Microservices Project NanoDegree

Capstone Micro services Project(Cloud Developer) Nanodegree

Overview: The purpose of the cloud developer capstone project is to give you a chance to combine what you've learned throughout the program. This project will be an important part of your portfolio that will help you achieve your Cloud-related career goals.

PROJECT PROPOSAL (SCOPE): Capstone Project is a cloud application developed for Udacity Cloud Engineering Nanodegree. It allows users to register and log into a web client, post photos to the feed, and process photos using an image filtering micro service. I will be using the AWS for deploying the micro services project. Will be using IAM, S3, RDS-Postgres, VPC, Subnets, Internet Gatway(IGW), EC2, ELB, Dockers,Kubectl using KOPS to deploy the micro services on AWS Cloud.

Here goes the AWS deployment diagram and will be replicating the same for the project deployment



The project is split into four services:

1. The Ionic Client- Ionic client web application which consumes the RestAPI Backend.
2. RESTful API Service, a Node-Express server which does user authentication and registration.
3. RESTful API Service, a Node-Express server which is used to list feeds and upload feed images to an AWS S3 bucket.
4. RESTful API Service, a Node-Express server which runs a simple script to process images.

.

Amit Goswami
Capstone | Udacity Microservices Project NanoDegree

PROJECT AWS ENVIRONMENT SETUP FOR DEPLOYMENT

- STEP-01 : Create the VPC - CapstoneVPC (10.0.0.0/16). Create the three subnets:
    - Capstone_publicsubnet01 (10.0.1.0/24); Availability Zone: us-east-1a
    - Capstone_privatesubnet02 (10.0.2.0/24); Availability Zone: us-east-1b
    - Capstone_publicsubnet03 (10.0.3.0/24); Availability Zone: us-east-1c
- STEP-02 : Create the Internet Gateway (IGW) : CapstoneIGW. Attach the VPC (CapstoneVPC) to the CapstoneIGW.
- STEP-03 : Create the route table - Capstone_mypublicroute and associate it with the CapstoneVPC. Every subnet we create is by default associated with the main route table. Edit routes in the custom route table to add the route to internet via CapstoneIGW. After this edit the "subnet associations" and add the public subnets to it.
- STEP-04 : Create the RDS-Postgress-"capstonerds" database in the Capstone_privatesubnet02.
- STEP-05 : Create the S3 Bucket(FILESTORES) in the Capstone_publicsubnet01.
    - File stores allow for archiving data. In AWS, the file store is called S3, and the archive resource is called “glacier”.
    - Content Delivery Network (CDN): are a network of proxy servers that are placed closer to end users to deliver data and compute. CDNs reduce latency for end users.
    - SignedURLs allow clients to send and receive data by directly communicating with the file store. This saves the server from using its bandwidth to serve as the intermediary that transmits data to and from the client. This is faster for clients as well.
    - Buckets: a simple directory-like system in which to store data
    - Bucket CORS Policy : You'll need this policy to create a bucket where we can use the SignedURL pattern.

```
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>GET</AllowedMethod>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedMethod>HEAD</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
</CORSRule>
</CORSConfiguration>
```

Deploying a Kubernetes Cluster with KOPS

Kops (Kubernetes Operations), it's an open-source free tool which helps us to easily deploy and manage a HA (High Availability) Kubernetes cluster on different cloud providers.

- Step1 : Create the AWS EC2 Instance and set the development environemnt.
- Step2 : Configure AWS CLI. And configure AWS account by using command "AWS Configure".

```
curl https://s3.amazonaws.com/aws-cli/awscli-bundle.zip -o awscli-bundle.zip
apt install unzip python
unzip awscli-bundle.zip
#sudo apt-get install unzip - if you dont have unzip in your system
./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

- Step3 : Install kops and kubectl. kopsis the tool we need to create the Kubernetes cluster on AWS. kubectl is the cli we use to manage the cluster once it's up and running. ##### Install Kops on Linux

```
•          curl -Lo kops
   https://github.com/kubernetes/kops/releases/download/$(curl -s
   https://api.github.com/repos/kubernetes/kops/releases/latest | grep
   tag_name | cut -d '"' -f 4)/kops-linux-amd64
•          chmod +x ./kops
•          sudo mv ./kops /usr/local/bin/
•      ##### Install Kubeclt on Linux
•          curl -Lo kubectl https://storage.googleapis.com/kubernetes-
   release/release/$(curl -s https://storage.googleapis.com/kubernetes-
   release/release/stable.txt)/bin/linux/amd64/kubectl
•         chmod +x ./kubectl
•         sudo mv ./kubectl /usr/local/bin/kubectl
```

- Step4 : Create an IAM user/role with Route53, EC2, IAM and S3 full access.

- Step5: Create a Route53 private hosted zone (you can create Public hosted zone if you have a domain) Real domain in Route53 : It is now possible to use kops without a real domain. Instead of using a Route53 domain, we can create a cluster using a subdomain of k8s.local, like chat.k8s.local. A cluster will be created with a load-balancer pointing to our masters. Kops needs a real domain and valid zone setup into AWS Route53. I know, this can be a blocking step, especially if you just want to just try kops on AWS. Unfortunately, it doesn't seem to be a way to around this. I've personally changed my "amit-goswami.com" domain name servers to Route53 time ago. It was super easy. I just had to download the zone file from GoDaddy and import it into Route53, telling GoDaddy to use the Route53 name servers.

Importing my personal registered domain name "amit-goswami.com" a Zone File from GODADDY.com and cofiguring ROUTE53:

```
 https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-creating-import.html
```

- Step6 : Create an S3 bucket : S3 bucket to store the cluster state: We just need to create an S3 bucket which kops will use to save the cluster's state files. I've called my bucket like a subdomain, state.app.amit-goswami.com

- Step7 : Creating the Kubernetes cluster using KOPS (Create kubernetes cluster definitions on S3 bucket) * Existing VPC and Internet Gateway will be re-used : Use kops create cluster with the --subnets argument for your existing subnets: * Create ssh key to login into servers. #ssh-keygen -f .ssh/id_rsa

```
    ```
     export KOPS_STATE_STORE=s3://state.app.amit-goswami.com
     export CLUSTER_NAME=capstone.amit-goswami.com
     export VPC_ID=vpc-01a5884a775b4a474 # replace with your VPC id
     export NETWORK_CIDR=10.0.0.0/16 # replace with the cidr for the VPC ${VPC_ID}
     export SUBNET_ID=subnet-0b683542993fc8e8a # replace with your subnet id
     export SUBNET_CIDR=10.0.1.0/24 # replace with your subnet CIDR
     export SUBNET_IDS=subnet-0c6468c90331aa4dd,subnet-0b683542993fc8e8a # replace with your comma separated subnet ids

     kops create cluster --zones=us-east-1a,us-east-1c --name=${CLUSTER_NAME} --vpc=${VPC_ID} --subnets=${SUBNET_IDS} --master-count 3 --master-size=m3.medium --node-count 2 --node-size=t2.medium --ssh-public-key=~/.ssh/id_rsa.pub --dns-zone=amit-goswami.com --dns private --yes

    ```
```

- Step8 : Create kubernetes cluser : kops update cluster capstone.amit-goswami.com --yes

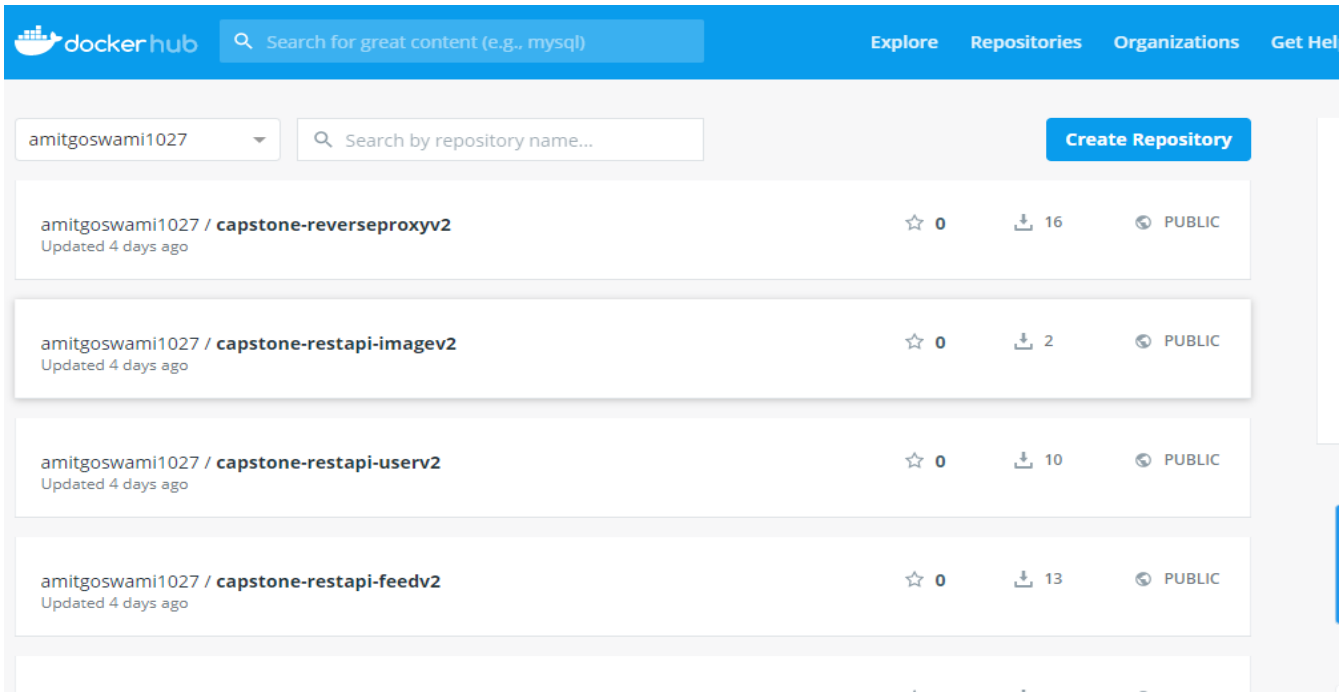
- Step9 : Validate the cluster : kops validate cluster --state "s3://state.app.amit-goswami.com" --name capstone.amit-goswami.com

- Step10 : To list nodes : kubectl get nodes

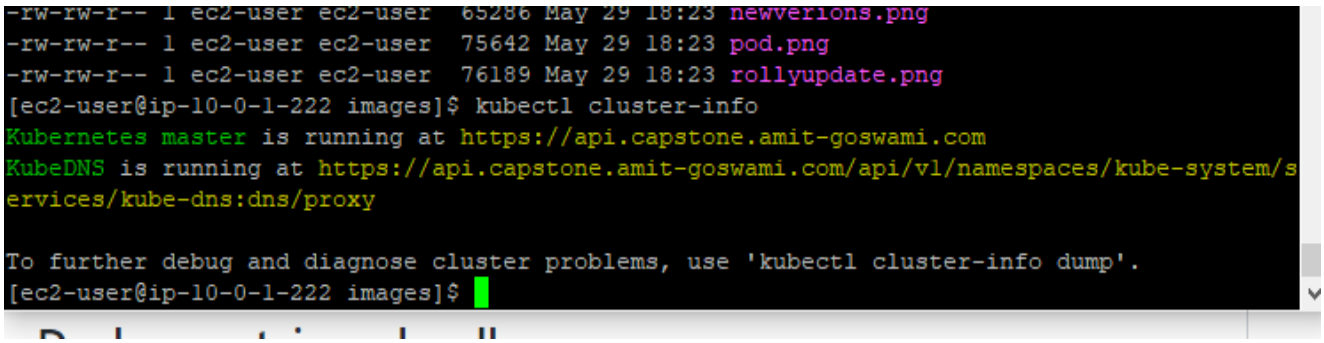
- Step11 : Kubernetes API and Security Group. The Kubernetes API is by default exposed on the internet. At the end it's the only way we can easily connect to our cluster (without using VPN connections to our VPC).

Amit Goswami
Capstone | Udacity Microservices Project NanoDegree

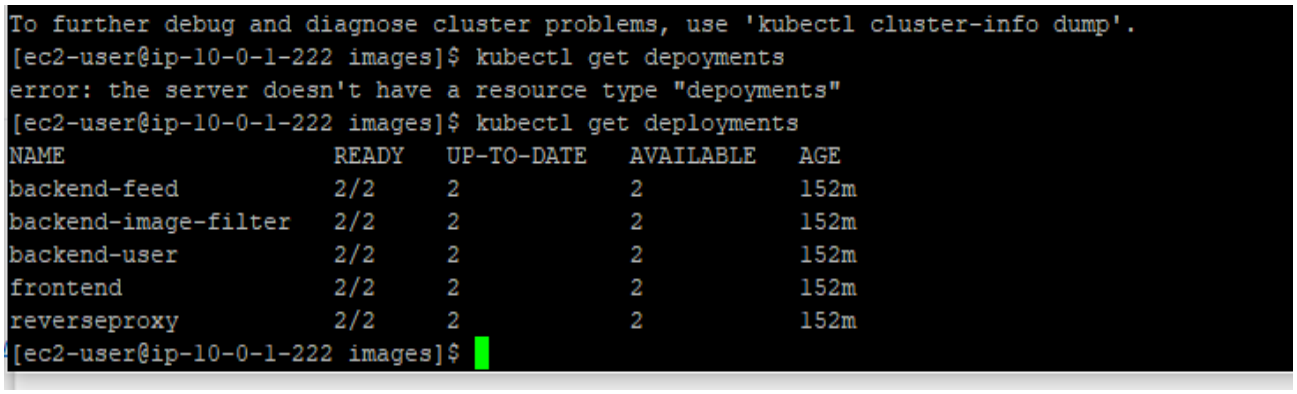
DOCKER HUB IMAGES DETAILS



Kubernetes Cluster deployment details



Kubernetes Deployment of Microservices



Kubernetes Deployment of services and ELB configuration to access from the outside the cluster.

```
[ec2-user@ip-10-0-1-222 images]$ kubectl get services
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP
                                       PORT(S)          AGE
backend-feed            ClusterIP      100.65.194.37    <none>
                                       8080/TCP         154m
backend-image-filter    ClusterIP      100.68.131.154   <none>
                                       8080/TCP         154m
backend-user            ClusterIP      100.67.229.98    <none>
                                       8080/TCP         154m
frontend                LoadBalancer   100.66.222.239   a17ea872508c740488e9b73f9ada8adf-120
1684481.us-east-1.elb.amazonaws.com    8100:31944/TCP   154m
frontend1               LoadBalancer   100.67.234.177   a554ce0af3ab24996a6de96e85ab5130-409
319829.us-east-1.elb.amazonaws.com     80:31958/TCP     120m
kubernetes              ClusterIP      100.64.0.1       <none>
                                       443/TCP          4h37m
reverseproxy            LoadBalancer   100.68.67.103    a779752e349f24a7b811866b8d5d4f59-767
756617.us-east-1.elb.amazonaws.com     8080:32684/TCP   154m
[ec2-user@ip-10-0-1-222 images]$
```

All running pods

```
[ec2-user@ip-10-0-1-222 images]$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
backend-feed-d5bf48bfd-hrr8q            1/1     Running   7          155m
backend-feed-d5bf48bfd-mc98t            1/1     Running   0          155m
backend-image-filter-64d44dc684-w6h6n   1/1     Running   0          155m
backend-image-filter-64d44dc684-xqn26   1/1     Running   0          155m
backend-user-7545b79fbf-8wz6l           1/1     Running   3          155m
backend-user-7545b79fbf-vh2wg           1/1     Running   5          155m
frontend-655ffdd55c-7xcqw               1/1     Running   0          155m
frontend-655ffdd55c-s9ktm               1/1     Running   0          155m
pod-example                             1/1     Running   0          155m
reverseproxy-67698bdcfb-nqb75           1/1     Running   14         155m
reverseproxy-67698bdcfb-vvbcv           1/1     Running   0          155m
[ec2-user@ip-10-0-1-222 images]$
```

Amit Goswami

Capstone | Udacity Microservices Project NanoDegree

Application rendering at : a17ea872508c740488e9b73f9ada8adf-1201684481.us-east-1.elb.amazonaws.com .