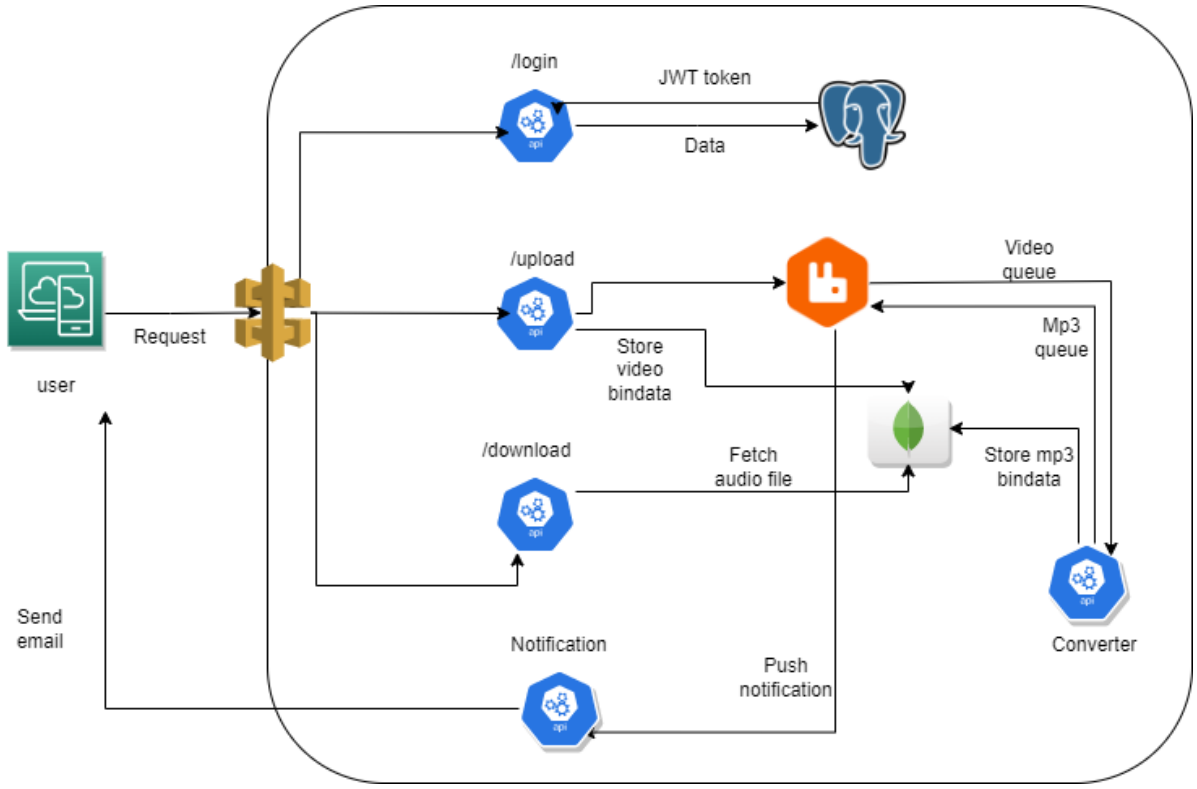
Microservices App monitored with Dynatrace

Converting mp4 videos to mp3 in a microservices-based architecture App + Observability & Monitoring with Dynatrace

Architecture



Deploying a Python-based Microservice Application on AWS EKS

Introduction

This document provides a step-by-step guide for deploying a Python-based microservice application on AWS Elastic Kubernetes Service (EKS). The application comprises four major microservices: `auth-server`, `converter-module`, `database-server` (PostgreSQL and MongoDB), and `notification-server`.

Prerequisites

Before you begin, ensure that the following prerequisites are met:

1. **Create an AWS Account:** If you do not have an AWS account, create one by following the steps here.
2. **Install Helm:** Helm is a Kubernetes package manager. Install Helm by following the instructions provided here.
3. **Python:** Ensure that Python is installed on your system. You can download it from the official Python website.
4. **AWS CLI:** Install the AWS Command Line Interface (CLI) following the official installation guide.

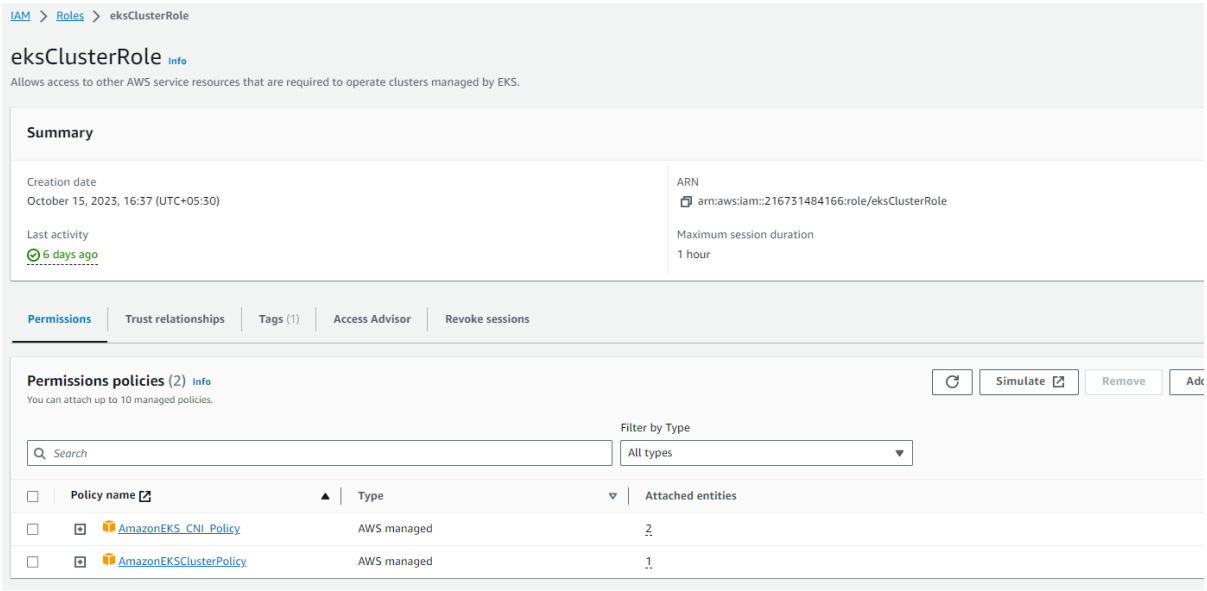
5. **Install kubectl:** Install the latest stable version of `kubectl` on your system. You can find installation instructions here.

6. **Databases:** Set up PostgreSQL and MongoDB on your local development machine

Low Level Step-by-Step Guide

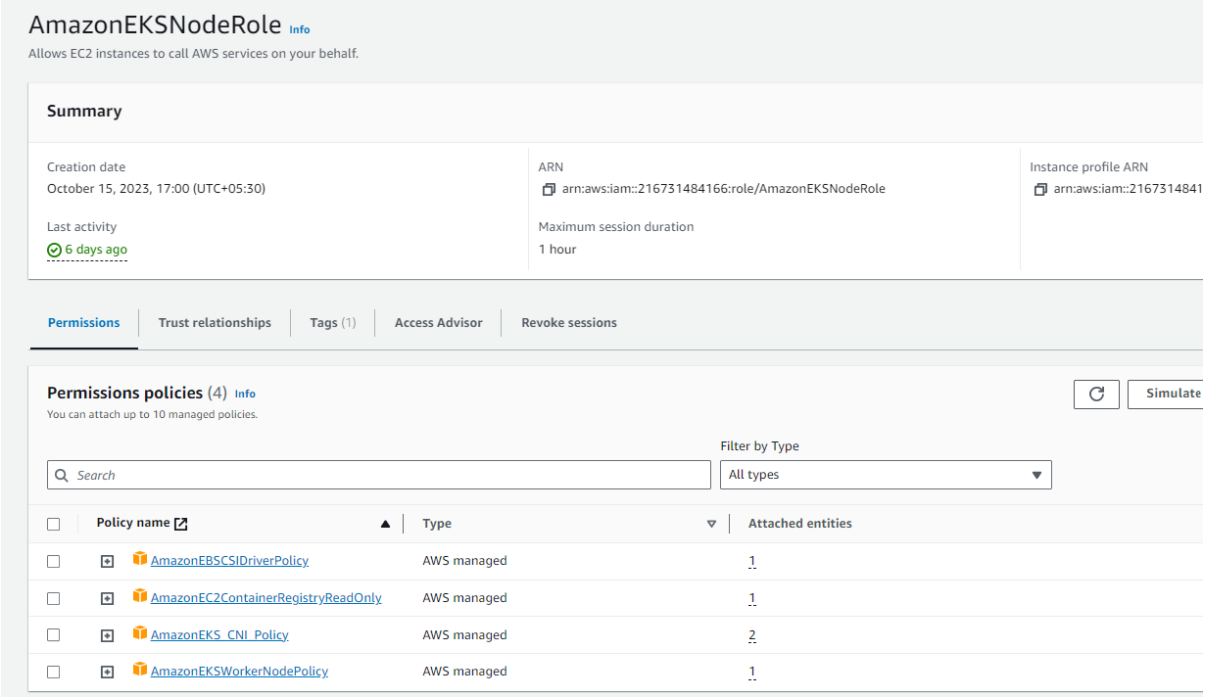
Cluster Creation

1. **Log in to AWS Console:**
    - Access the AWS Management Console with your AWS account credentials.
2. **Create eksCluster IAM Role**
    - Follow the steps mentioned in this documentation using root user
    - After creating it will look like this:



- Please attach `AmazonEKS_CNI_Policy` explicitly if it is not attached by default

3. **Create Node Role - AmazonEKSNodeRole**
    - Follow the steps mentioned in this documentation using root user
    - Please note that you do NOT need to configure any VPC CNI policy mentioned after step 5.e under Creating the Amazon EKS node IAM role
    - Simply attach the following policies to your role once you have created `AmazonEKS_CNI_Policy`, `AmazonEBSCSIDriverPolicy`, `AmazonEC2ContainerRegistryReadOnly` incase it is not attached by default
    - Your AmazonEKSNodeRole will look like this:



4. **Open EKS Dashboard:**
    - Navigate to the Amazon EKS service from the AWS Console dashboard.
5. **Create EKS Cluster:**
    - Click "Create cluster."
    - Choose a name for your cluster.
    - Configure networking settings (VPC, subnets).
    - Choose the `eksCluster` IAM role that was created above
    - Review and create the cluster.
6. **Cluster Creation:**
    - Wait for the cluster to provision, which may take several minutes.
7. **Cluster Ready:**
    - Once the cluster status shows as "Active," you can now create node groups.

Node Group Creation

1. In the "Compute" section, click on "Add node group."
2. Choose the AMI (default), instance type (e.g., t3.large), and the number of nodes=1.

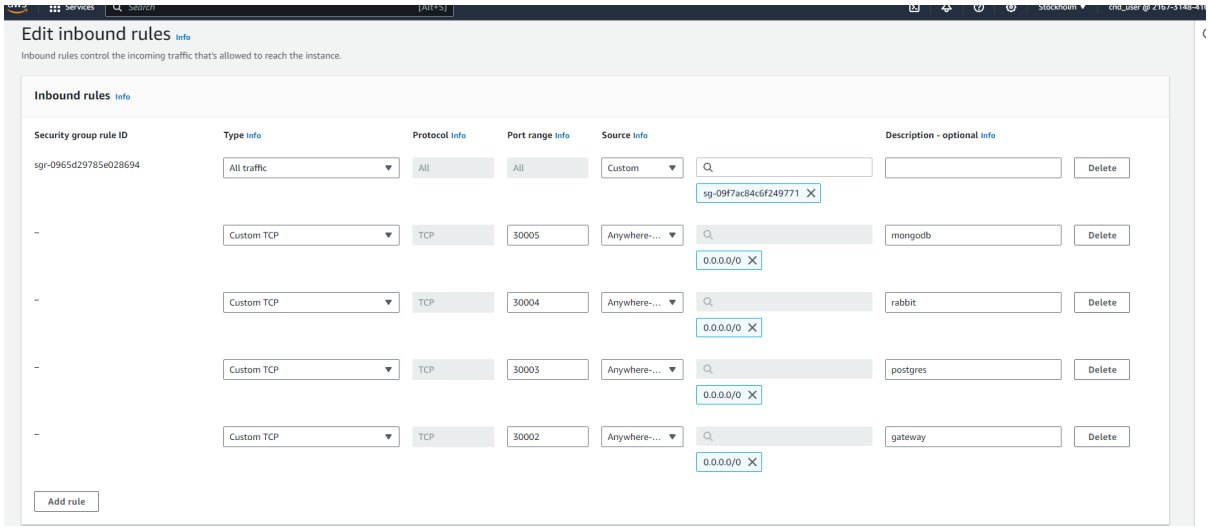
Bear in mind that different types of AWS VMs have different max k8s pods that can be spinned up on them. This will affect heavily the amount of replicas that you can spin up per each of the microservices. The replicas parameter can be found, for example, in src/converter-service/manifest/converter-deploy.yaml under "replicas" key.

t3.medium machine has max pods = 17, which will not be enough for our case. Learn more about thresholds here: https://github.com/awslabs/amazon-eks-ami/blob/master/files/eni-max-pods.txt

3. Click "Create node group."

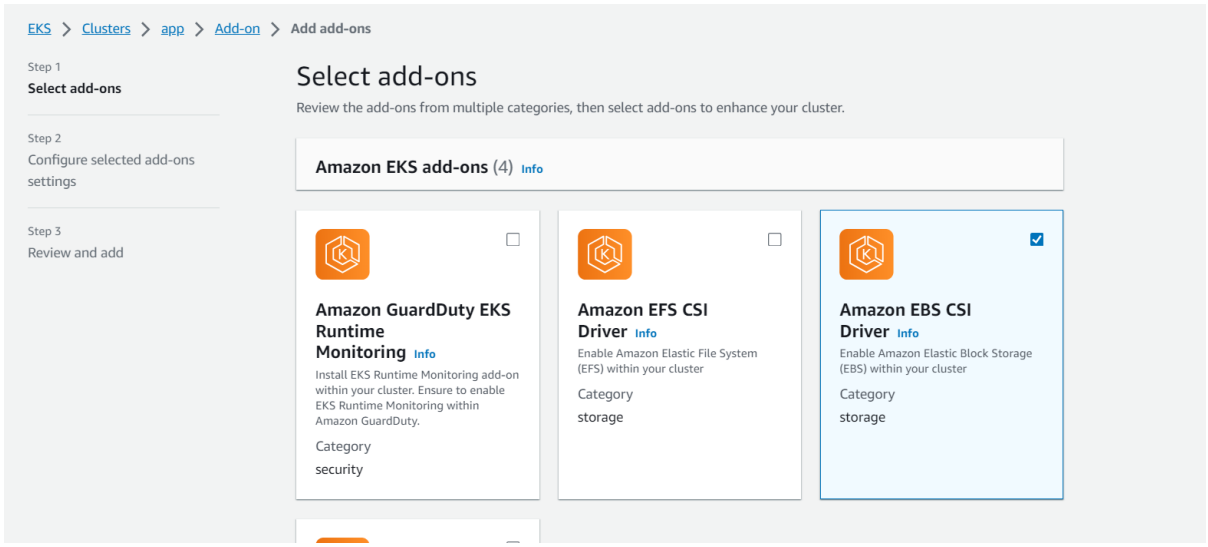
Adding inbound rules in Security Group of Nodes

NOTE: Ensure that all the necessary ports are open in the node security group.



Enable EBS CSI Addon

1. enable addon `ebs csi` this is for enabling pvcs once cluster is created



Deploying your application on EKS Cluster

1. Clone the code from this repository.
2. Set the cluster context:

```
aws eks update-kubeconfig --name <cluster_name> --region <aws_region>
```

Commands

Here are some essential Kubernetes commands for managing your deployment:

MongoDB

To install MongoDB, set the database username and password in `values.yaml`, then navigate to the MongoDB Helm chart folder and run:

```
cd Helm_charts/MongoDB
helm install mongo .
```

Connect to the MongoDB instance using:

```
mongosh 'mongodb://<username>:<pwd>@<nodeip>:30005/mp3s?authSource=admin'
```

PostgreSQL

Set the database username and password in `values.yaml`. Install PostgreSQL from the PostgreSQL Helm chart folder and initialize it with the queries in `init.sql`. For PowerShell users:

```
cd ..
cd Postgres
helm install postgres .
```

Connect to the Postgres database and copy all the queries from the "init.sql" file.

```
psql 'postgres://<username>:<pwd>@<nodeip>:30003/authdb'
```

Run sql commands from Postgres/init.sql

Check the results:

\d select * from auth_user;

Ctrl + D

RabbitMQ

Deploy RabbitMQ by running:

```
helm install rabbitmq .
```

Ensure you have created two queues in RabbitMQ named `mp3` and `video`. To create queues, visit `<nodeIp>:30004` and use default username `guest` and password `guest`

NOTE: Ensure that all the necessary ports are open in the node security group.

Notification Configuration

For configuring email notifications and two-factor authentication (2FA), follow these steps:

1. Go to your Gmail account and click on your profile.
2. Click on "Manage Your Google Account."
3. Navigate to the "Security" tab on the left side panel.
4. Enable "2-Step Verification."
5. Search for the application-specific passwords. You will find it in the settings.
6. Click on "Other" and provide your name.
7. Click on "Generate" and copy the generated password.
8. Paste this newly generated password in `src/notification-service/manifest/secret.yaml` along with your email.

Apply the manifest file for each microservice:

- **Auth Service:**

```
cd auth-service/manifest
kubectl apply -f .
```

- **Gateway Service:**

```
cd gateway-service/manifest
kubectl apply -f .
```

- **Converter Service:**

```
cd converter-service/manifest
kubectl apply -f .
```

- **Notification Service:**

```
cd notification-service/manifest
kubectl apply -f .
```

Application Validation

After deploying the microservices, verify the status of all components by running:

```
kubectl get all
```

Run the application through the following API calls:

API Definition

- **Login Endpoint**

Use password that we inserted for our user into psql database!

```
curl -X POST http://nodeIP:30002/login -u <email>:<password>
```

Expected output: success!

- **Upload Endpoint**

```
cd into assets/
```

```
 curl -X POST -F 'file=@./video.mp4' -H 'Authorization: Bearer <JWT
Token>' http://nodeIP:30002/upload
```

Check if you received the ID on your email.

- **Download Endpoint**

```
 curl --output video.mp3 -X GET -H 'Authorization: Bearer <JWT Token>'
"http://nodeIP:30002/download?fid=<Generated fid>"
```

Connecting Dynatrace Monitoring & Observability Platform

Log in to your Dynatrace account & Go to Apps. Look for "Kubernetes" app & click "Connect automatically via Dynatrace Operator" there.

Create & save "Dynatrace Operator token" & "Data ingest token"
dt0c01.MOCK2NVRVMXRJZDZI3DTERR2.P3VBZBJ4QLTQJ2LPGMRIE2RP3AEWWUFIUN6HWLY3G627WMMKFB7J22ANN6TCS7PZ

dt0c01.BZDVSSPNUMAOZDKWYFNTAGV7.FSHRCWFP5MTR2SALPPX6QKEE7JW5JLJ6672FKK54AYJ2ICLN7LLR25ROK5YIJXBF

Save dynakube.yaml to the root folder of our project

Run a set of `kubectl` commands in your terminal:

```
kubectl create namespace dynatrace
kubectl apply -f https://github.com/Dynatrace/dynatrace-
operator/releases/download/v0.14.2/kubernetes.yaml
kubectl -n dynatrace wait pod --for=condition=ready --
selector=app.kubernetes.io/name=dynatrace-
operator,app.kubernetes.io/component=webhook --timeout=300s
kubectl apply -f dynakube.yaml
```

Check out "ActiveGates" section in "Deployment status"

Destroying the Infrastructure

To clean up the infrastructure, follow these steps:

1. **Delete the Node Group:** Delete the node group associated with your EKS cluster.
2. **Delete the EKS Cluster:** Once the nodes are deleted, you can proceed to delete the EKS cluster itself.