Deploying a machine learning app on Amazon Web Services

Step 1: Model Creation

1. **Create and Save the Model:**
   - Develop your machine learning model, train it, and save it as a pickle file (e.g., `model.pkl`).

Step 2: App Development

1. **Develop the front end App:**
   - Create a Python file (e.g., `app.py`) to load the model and create the application.

Step 3: Dependency Management

1. **Generate `requirements.txt`:**
   - Run `pip freeze > requirements.txt` to capture the necessary Python dependencies for your project.

Step 4: Containerization

1. **Construct Dockerfile:**
   - Create a `Dockerfile` specifying how to build your app. Include necessary dependencies and instructions to run the app.
2. **Build and Push Docker Image:**
   - Build the Docker image: `docker build -t your-image-name .`
   - Tag the image: `docker tag your-image-name username/your-image-name:tag`
   - Push the image to Docker Hub: `docker push username/your-image-name:tag`

Step 5: AWS Configuration

1. **Configure AWS CLI:**
   - Run `aws configure` to set up your AWS credentials. Make sure to generate keys by logging in as IAM root user and clicking on security credentials.
     AWS CLI Installation

Step 6: Kubernetes Setup

1. **Install kubectl and eksctl:**
   - Follow the official documentation to install `kubectl` and `eksctl` on your local machine.
     kubectl Installation eksctl Installation

Step 7: EKS Cluster Creation

1. **Create an EKS Cluster:**
   - Use `eksctl` to create an Amazon EKS cluster:
   - `eksctl create cluster --name demo-cluster --version 1.28 --region ap-south-1 --nodegroup-name linux-nodes --node-type t2.micro --nodes 1`

Step 8: App Deployment

1. **Define Kubernetes Deployment and Service:**
   - Create a `.yaml` file (e.g., `flask-app-deployment.yaml`) defining the deployment and service configurations for your app.
2. **Deploy the Application:**
   - Apply the configuration to deploy the app: `kubectl apply -f flask-app-deployment.yaml`

step 9: Debugging And Sanity Check

- `kubectl get pod` to check the status of your pods.
- `kubectl get services` to get the external IP where the app will be hosted.
- `kubectl describe pods` verbose status of your pods which can be used for debugging if the pods are not running.
- `eksctl delete cluster --name <name-of-cluster>` this will delete the Kubernetes cluster from aws and best part is it removes all the associated EC2 instances as well.

Step 9: Version Control

1. **Implement Version Control:**
   - Set up a Git repository for your project.
   - Use DVC for data and code versioning.

Step 10: Continuous Integration and Deployment (CI/CD)

1. **Set Up AWS CodePipeline:**
   - Configure AWS CodePipeline with AWS CodeBuild for CI/CD.
   - Create a `buildspec.yml` file defining build and deployment steps.
   - Trigger the pipeline on code changes.

Step 11: Infrastructure as Code (IaC)

1. **Use Terraform for IaC:**
   - Write Terraform configurations in `main.tf` to manage AWS resources such as EKS cluster, VPC, and security groups.

References

deployment