Infrastructure Explanation

1. Amazon ECR (Elastic Container Registry)

- We created an ECR repo on Amazon and pushed our images on that one for eks cluster to push later.

1. Amazon VPC (Virtual Private Cloud)

- It allows you to control your network environment, including IP address ranges, subnets, route tables, and network gateways.
- In this infrastructure, we create a new Amazon VPC with the specified CIDR block (10.0.0.0/16) using the Vpc resource.
- The VPC is tagged with metadata, such as Environment and Project, for identification and organization purposes.

2. Subnets

- In this infrastructure, we create two subnets: private-subnet and public-subnet.
- The private-subnet is intended for EKS worker nodes and other private resources, while the public-subnet is used for resources accessible from the internet.

3. Security Group & Node Security Group

- In this infrastructure, we create the eks-sg security group using the SecurityGroup resource.
- And then we created a Node security group which helps us attach security group, vpc and eks cluster with each other.
- The security group is associated with the EKS cluster and allows communication with the worker nodes.

4. EKS Cluster (Elastic Kubernetes Service)

- In this infrastructure, we create an EKS cluster named airTek-cluster using the Cluster resource.
- The cluster is configured to use the private subnet for EKS tasks by specifying Node Association to Public Address to false.

5. Nginx Ingress Controller

- The Nginx Ingress Controller is an external load balancer that handles incoming HTTP/HTTPS traffic and routes it to the appropriate Kubernetes services.
- We install the Nginx Ingress Controller on the EKS cluster using Helm with the Helm Chart resource.
- The Nginx Ingress Controller is a critical component for controlling external access to services deployed on the cluster.

6. Kubernetes Namespace

- In this infrastructure, we create a Kubernetes namespace named airTek-prod namespace using the Namespace resource.
- All resources related to the web app and API will be deployed within this namespace for organization and separation.

7. Kubernetes Deployments

- We create two Kubernetes deployments using the Deployment resource: one for the web app and one for the API.
- Each deployment specifies the container image to use and the desired number of replicas to maintain.

8. Kubernetes Services

- We create two Kubernetes services using the Service resource: one for the web app and one for the API.
- The web app service is exposed through an Ingress resource (Nginx Ingress) and accessible via the specified domain (airtek-web-app.com).
- The API service is not exposed externally and can only be accessed through the web app service.

9. Kubernetes Ingress

- Kubernetes Ingress is an API object that manages external access to services within a Kubernetes cluster.
- We create an Ingress resource (webAppIngress) to expose the web app service to the internet using the Nginx Ingress Controller.
- The Ingress rule specifies the host (airtek-web-app.com) and the path (/) to direct traffic to the web app service.

Integration and Flow

- The Amazon ECR helps us in getting docker images of our apps.
- The Amazon VPC provides the networking infrastructure for the EKS cluster, subnets, and security groups.
- The EKS cluster is the managed Kubernetes service responsible for orchestrating and managing containers.
- The Nginx Ingress Controller acts as the external load balancer and manages incoming HTTP/HTTPS traffic.
- The web app and API are deployed as Kubernetes pods within the airTek-dev Kubernetes namespace.
- The web app service is exposed to the internet through the Nginx Ingress Controller using the specified domain (airtek-web-app.com).
- The API service is not directly exposed and is accessible only through the web app service within the Kubernetes cluster.
- The web app communicates with the API by making requests to the API service within the cluster using the internal Kubernetes DNS.
- The security group (eks-sg) allows communication between the EKS cluster and worker nodes.

Overall, this infrastructure provides a scalable, secure, and well-organized environment to deploy a C# based web application and its API using Amazon EKS and Kubernetes resources. The Nginx Ingress Controller efficiently handles external access to the web app while ensuring secure communication between the web app and API within the Kubernetes cluster.