Horizontal pod autoscaling

- This is a feature in k8s that adjusts the number of pods in the deployment, replica sets based on observed metrics
- For this to work we need to enable metrics server in k8s Refer Here
- Target Metrics: HPA requires to specify a target for metric which you want ot scale on
- Refer Here for official docs
- Refer Here for the changes to implement hpa

Node Autoscaling in AKS

- This feature automatically increases/decreases number of nodes in your aks cluster based on resource requirements Refer Here

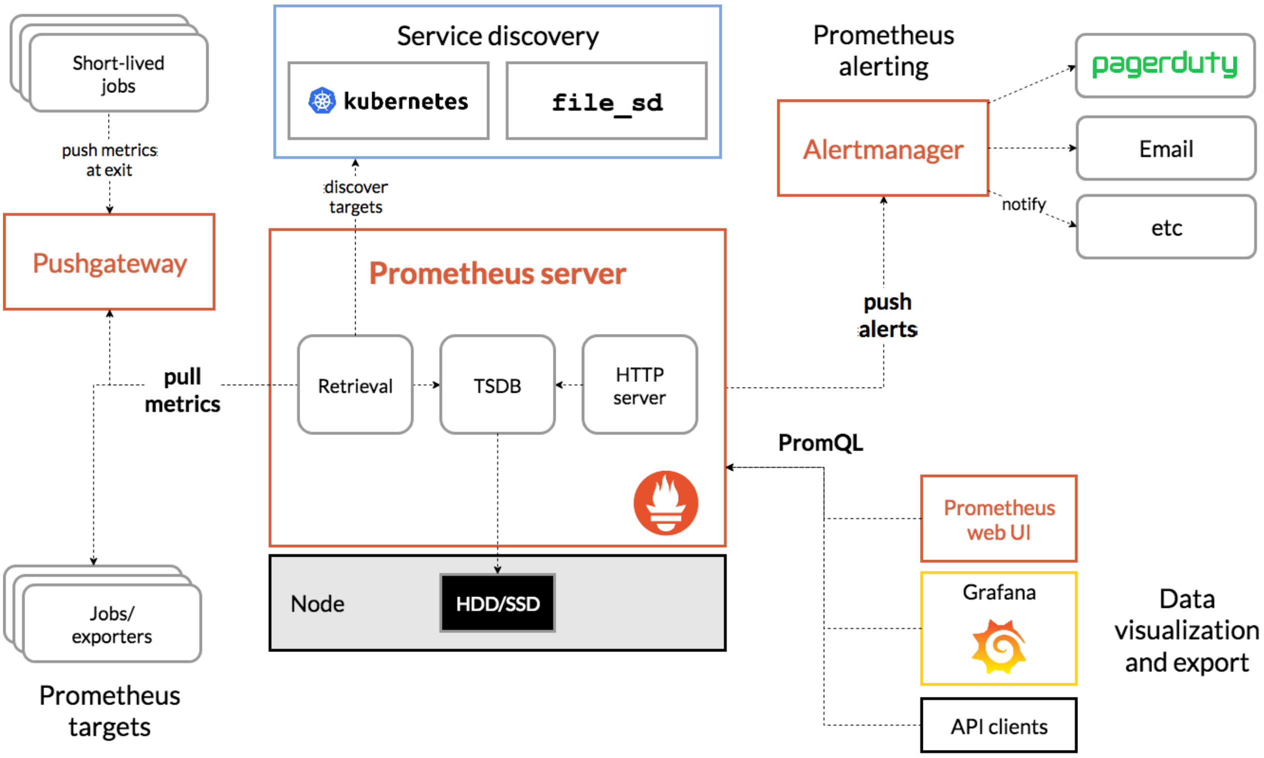
Node Autoscaling in EKS

- Refer Here for autoscaling options
- AWS Options
  - Cluster Autoscaler
  - Karpenter
- Exercise:
  - Write a k8s deployment and service with cluster ip for running
    - spring petclinic
      - 8080
    - gameoflife
      - 8080

Prometheus

- Refer Here for official site of Prometheus

- Architecture



- Setting up prometheus and grafana using helm Refer Here
- Grafana is visualization platform Refer Here

Prometheus and Grafana Setup on Kubernetes

Prerequisites

- A running Kubernetes cluster
- kubectl configured to interact with your cluster
- Helm installed on your local machine

Step 1: Install Prometheus

1. **Add the Prometheus Helm repository:**

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

2. **Install Prometheus using Helm:**

```
helm install prometheus prometheus-community/prometheus
```

3. **Verify the installation:**

```
kubectl get pods -l "release=prometheus"
```

Step 2: Install Grafana

1. **Add the Grafana Helm repository:**

   ```
   helm repo add grafana https://grafana.github.io/helm-charts
   helm repo update
   ```

2. **Install Grafana using Helm:**

   ```
   helm install grafana grafana/grafana
   ```

3. **Verify the installation:**

   ```
   kubectl get pods -l "release=grafana"
   ```

4. **Retrieve the admin password for Grafana:**

   ```
   kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo
   ```

Step 3: Accessing Prometheus and Grafana

1. **Port-forward to access Prometheus:**

   ```
   kubectl port-forward deploy/prometheus-server 9090
   ```

   Now, you can access Prometheus by navigating to `http://localhost:9090` in your web browser.

2. **Port-forward to access Grafana:**

   ```
   kubectl port-forward deploy/grafana 3000
   ```

   Now, you can access Grafana by navigating to `http://localhost:3000` in your web browser.

   Use `admin` as the username and the password retrieved in the previous step.

Step 4: Configure Prometheus as a Data Source in Grafana

1. **Login to Grafana:**

   Navigate to `http://localhost:3000` and log in with the credentials.

2. **Add Prometheus as a Data Source:**

   - Click on the **gear icon** (Configuration) in the left sidebar.
   - Click **Data Sources**.
   - Click **Add data source**.
   - Select **Prometheus**.
   - In the HTTP URL field, enter `http://prometheus-server.default.svc.cluster.local:9090`.
   - Click **Save & Test** to verify the connection.

Step 5: Creating a Dashboard in Grafana

1. **Import a Dashboard:**

   - Click on the **plus icon** in the left sidebar and select **Import**.
   - Enter a dashboard ID from Grafana's Dashboard repository, for example, `6417` for a Kubernetes cluster monitoring dashboard.
   - Click **Load**.
   - Select the Prometheus data source you added earlier.
   - Click **Import**.

2. **Customize and View Metrics:**

   - You can now customize the dashboard to fit your needs and view various metrics collected by Prometheus.

Service Mesh

- A Service mesh is a dedicated infrastructure layer to manage service to service communications within microservice architecture.

- Popular Services Mesh

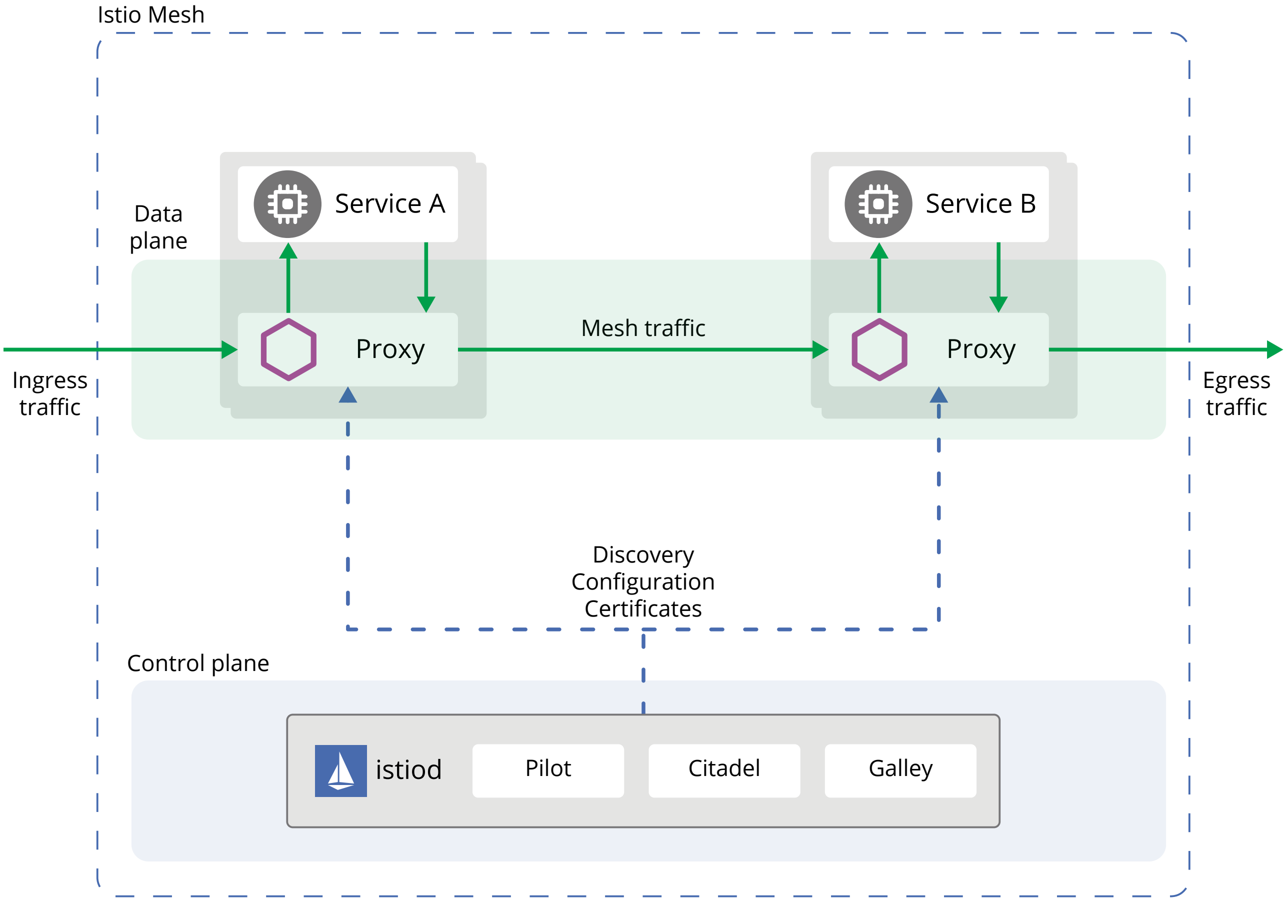
  - Istio
  - Linkerd
  - Open Service Mesh

- Concepts:

  - Traffic Management:
    - Load balancing
    - Traffic splitting
    - Circuit Breaking
  - Security:
    - Mutual TLS (mTLS)

        - Authentication and Authorization
    - Observability
        - Metric Collections
        - Distributed Tracing
        - Logging
    - Policy Enforcment
        - Rate Limiting

- istio architecture



- Example Refer Here

- Refer Here for linkerd