**Case Study 1: High Availability Microservices Deployment**

**Scenario:**

A growing **e-commerce startup** faces frequent website crashes during peak traffic. They need a scalable solution.

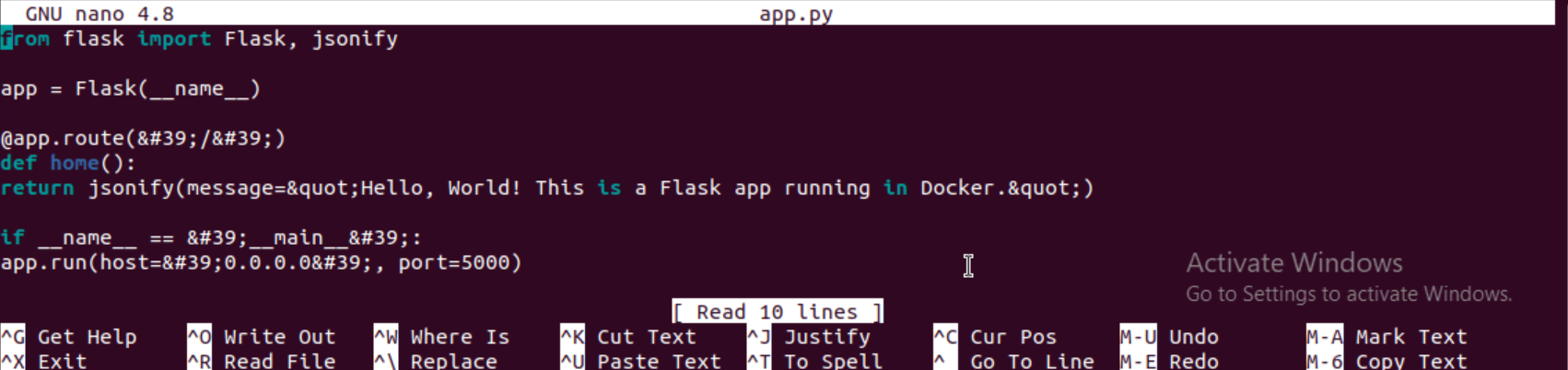
**Solution:**

Deploy **Product & Cart services** with Kubernetes  
Implement **Horizontal Pod Autoscaler (HPA)** to scale dynamically.

**Plugins Installation:**

* **Docker** (for building images)
* **Kubernetes cluster** (with master-vm, worker1-vm, worker2-vm)
* **Metrics Server** (for auto-scaling)

**Step 1:** Building & Containerizing the Flask Application.

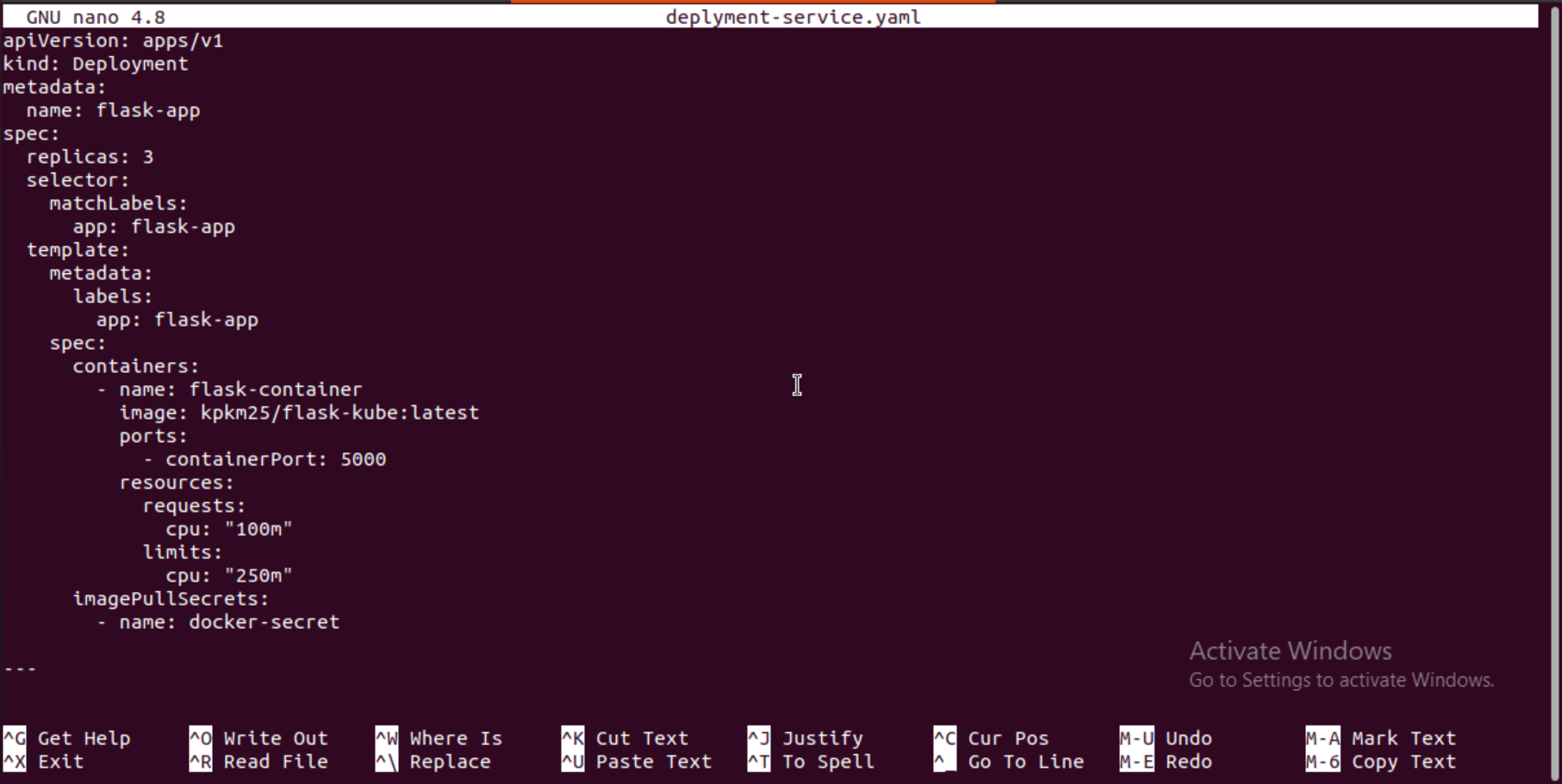


**Step 2:** Dockerfile

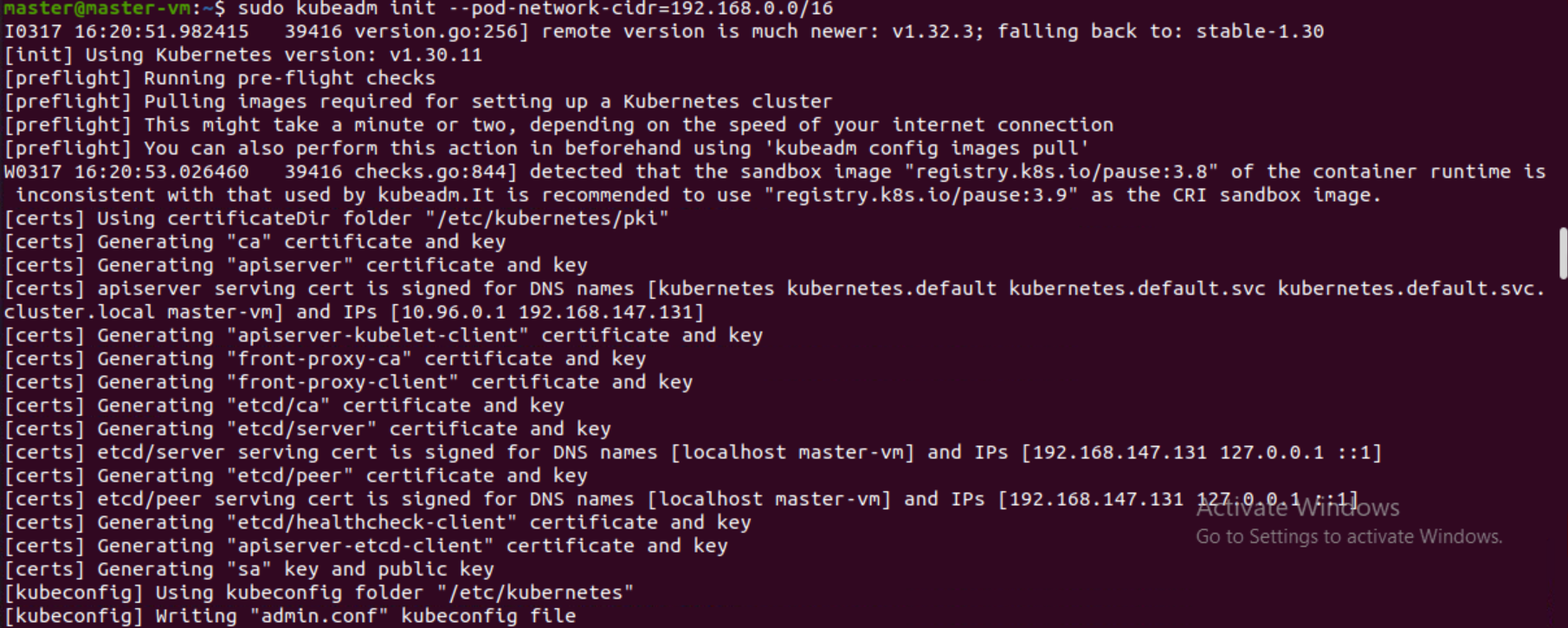


**Step 3:** Deploying Flask App on Kubernetes.

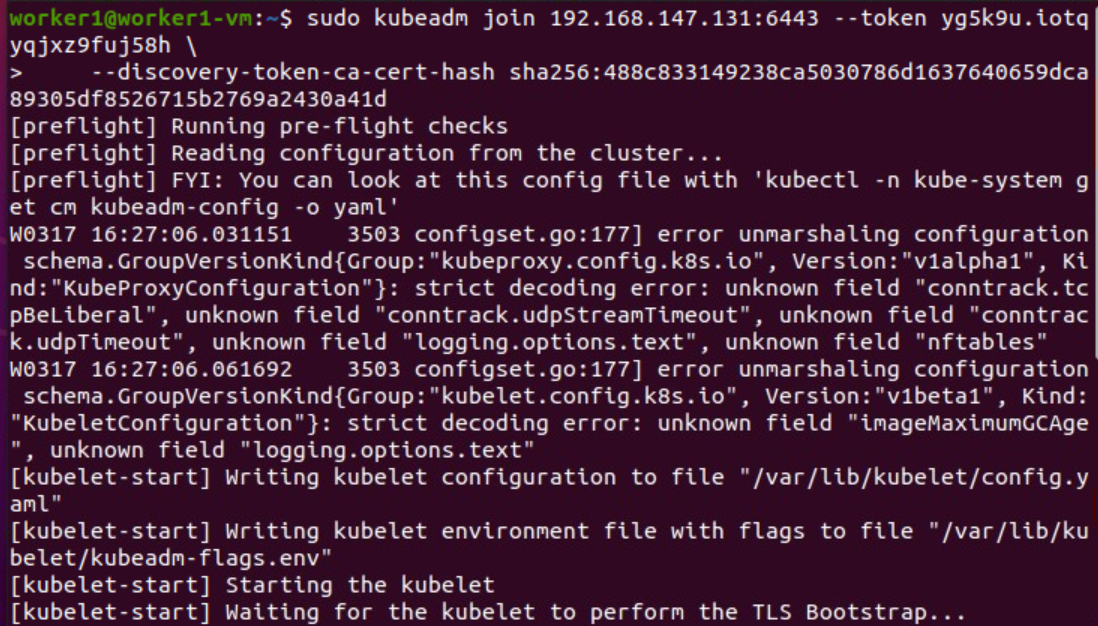
**-Deployment & Service YAML (deployment-service.yaml)**

****

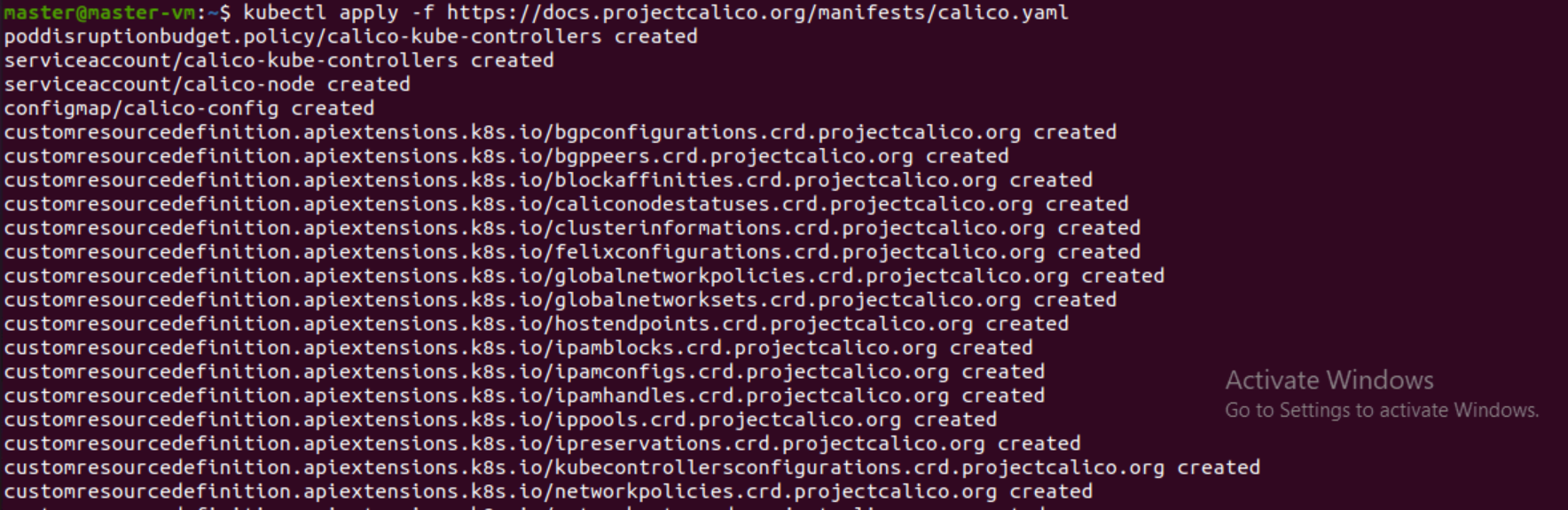
**Step 4:** Initially connecting the Master Node with the worker node 1. Using the below Token.



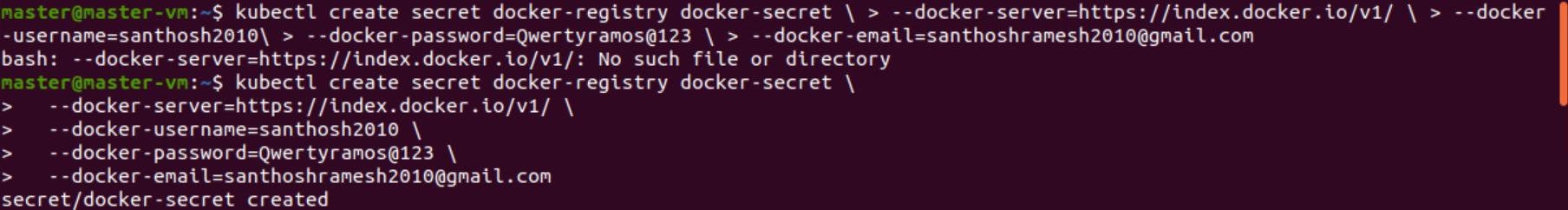
**Step 5:** Worker Node 1 is connected.



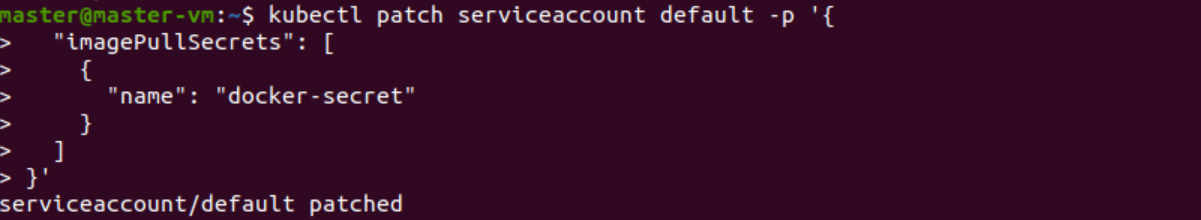
**Step 6:** Applying the Deployment**.**

****

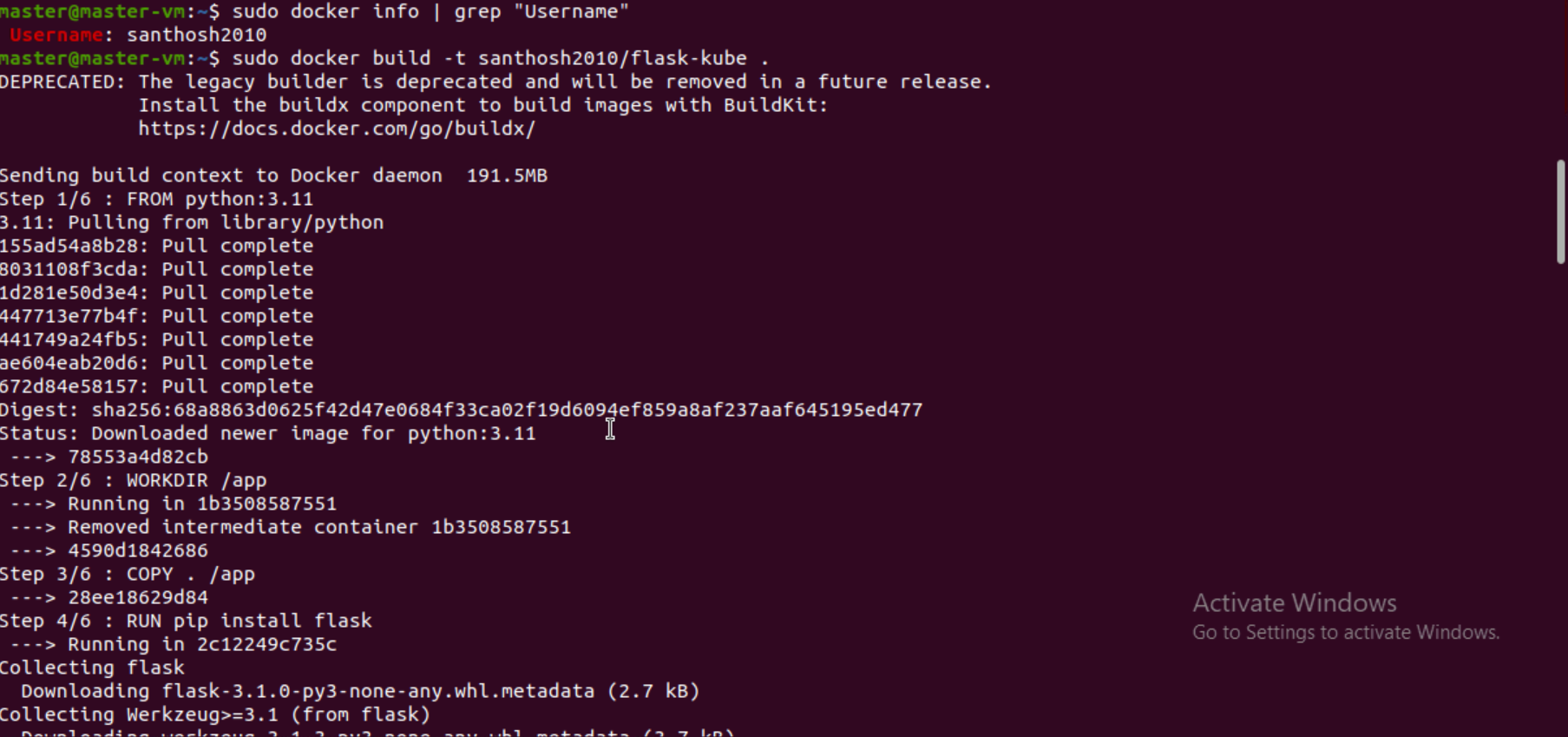
**Step 7**: Fixing Docker Hub Rate Limits (Authentication Issue).

****

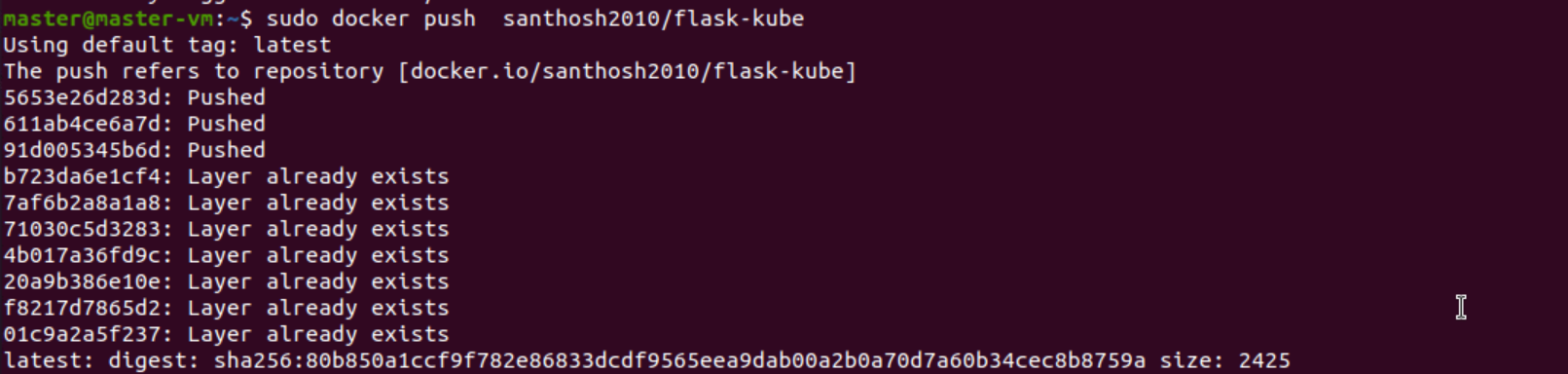
**Step 8:** Patch Default Service Account.



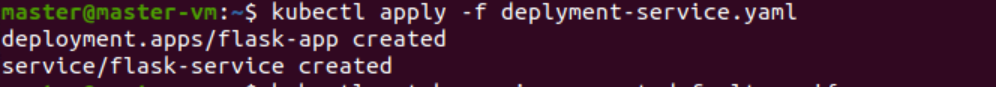
**Step 9:** Building the Docker Image.



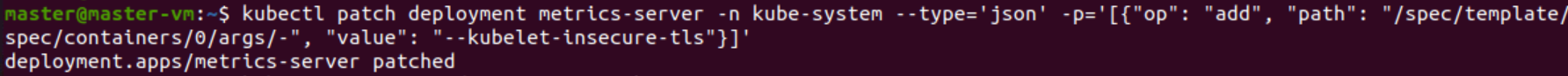
**Step 10:** Pushing the Docker Image.

****

**Step 11:** Installing & Troubleshooting Metrics Server.

****

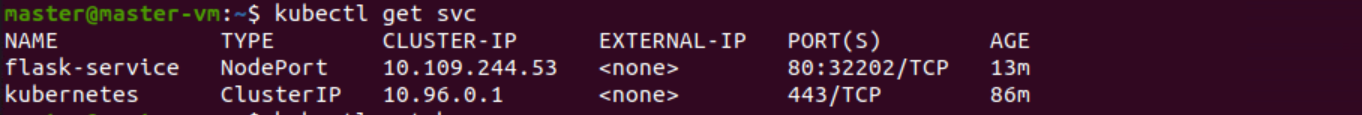
**Step 12:** Installing & Troubleshooting Metrics Server.

****

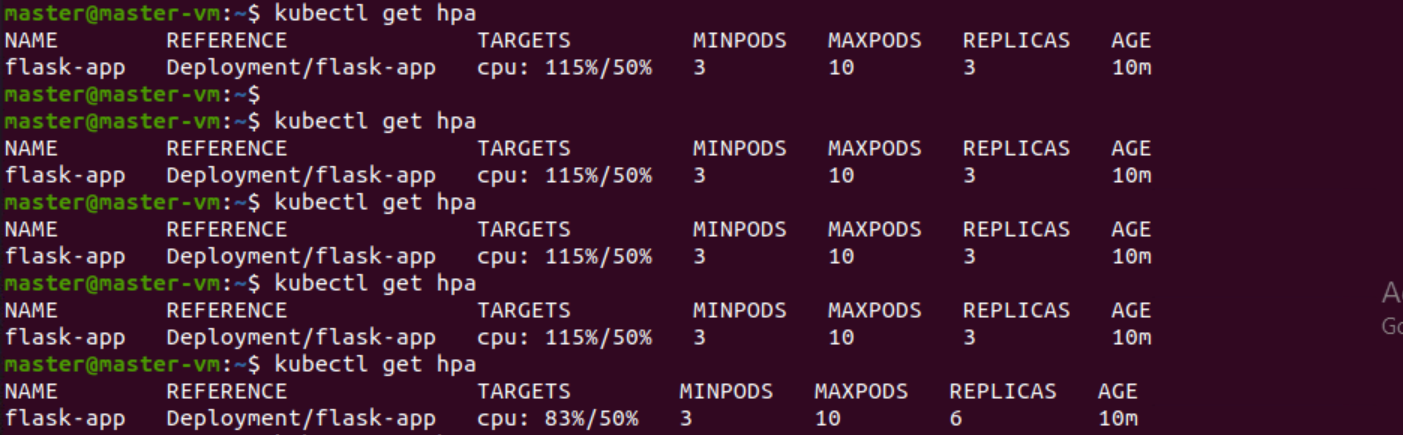
**Step 13:** Enabling HPA (Horizontal Pod Auto-scaler).

****

**Step 14:** Seeing the Service of the Deployment.



**Step 15:** Simulating Load for HPA.



**Step 16:** Checking the Scaling.

