# Deploying the first pod and accessing it

kubectl run nginxpod --image=nginx --port 80

kubectl get pods

kubectl describe pod nginxpod

kubectl exec -it nginxpod /bin/sh

kubectl logs nginxpod

# Deep-dive into Master setup

kubectl cluster-info

kubectl cluster-info dump > cluster-dump

kubectl get node worker-node-1.example.com

kubectl describe node worker-node-1.example.com | less

# Look at Status(should be FALSE), Address, Capacity, and Events

kubectl get namespaces

kubectl get pods -A

kubectl get pods -n kube-system

# Look into /etc/kubernetes/ - Config, manifests & pki

kubectl get pods -n kube-system -o wide

service kubelet status

# Kubernetes Dashboard

**Deploying the dashboard**

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

**Verifying the Dashboard resources**

kubectl get pods -n kubernetes-dashboard -o wide

kubectl get deployment -n kubernetes-dashboard -o wide

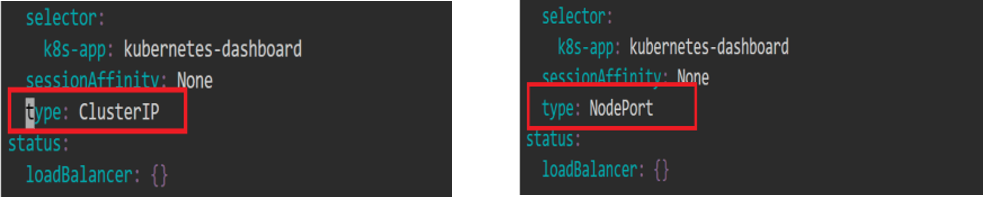
kubectl get svc -n kubernetes-dashboard -o wide

**Editing the Service type of the dashboard**

kubectl edit svc -n kubernetes-dashboard kubernetes-dashboard

**Note:** Change the attribute after entering the deployment

type: ClusterIP (image 1) to NodePort (image 2)



**Verifying the changes**

kubectl get svc -n kubernetes-dashboard -o wide

Note down the service(node-port) port number , here it is 31851

**Checking where the Pod is running**

kubectl get pods -n kubernetes-dashboard -o wide

kubectl get svc -n kubernetes-dashboard -o wide

kubectl get nodes -o wide

**Accessing Kubernetes Dashboard**

Click on the master tab on the lab, and then click on the desktop option.

Open Firefox browser

[**https://localhost**](https://localhost/)**:<<NodePort>>**

Example: https://localhost:31851

Click on Advanced -> Accept Risk and Continue

On the Kubernetes Dashboard,

Select **Token** from the given options and enter the token

In Master Node,

vi ServiceAccount.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

kubectl apply -f ServiceAccount.yaml

vi ClusterRoleBinding.yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kubernetes-dashboard

kubectl apply -f ClusterRoleBinding.yaml

Now we need to find the token we can use to log in. Execute the following command:

kubectl -n kubernetes-dashboard create token admin-user

Copy the token and paste it in Kubernetes Dashboard.

**[OPTIONAL] Cleanup:**

kubectl delete -f<https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.0/aio/deploy/recommended.yaml>

**Reference**:

<https://v1-28.docs.kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/>

<https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md>

# ETCD - Backup

Step 1: Install etcd-client

sudo snap install etcd

sudo apt install etcd-client

sudo chmod a+rw -R /etc/kubernetes/pki

Step 2: backup data

sudo ETCDCTL\_API=3 etcdctl snapshot save etcd\_backup.db \

--endpoints https://127.0.0.1:2379 \

--cert=/etc/kubernetes/pki/etcd/server.crt \

--key=/etc/kubernetes/pki/etcd/server.key \

--cacert=/etc/kubernetes/pki/etcd/ca.crt

Step 3: Verify

sudo ETCDCTL\_API=3 etcdctl --write-out=table snapshot status etcd\_backup.db \

--endpoints https://127.0.0.1:2379 \

--cert=/etc/kubernetes/pki/etcd/server.crt \

--key=/etc/kubernetes/pki/etcd/server.key \

--cacert=/etc/kubernetes/pki/etcd/ca.crt

Reference: https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#restoring-an-etcd-cluster

# **Upgrading Control Plane's Kubernetes Versions**

Reference:

Master Upgrade - <https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/>

Worker Node upgrade - <https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/upgrading-linux-nodes/>

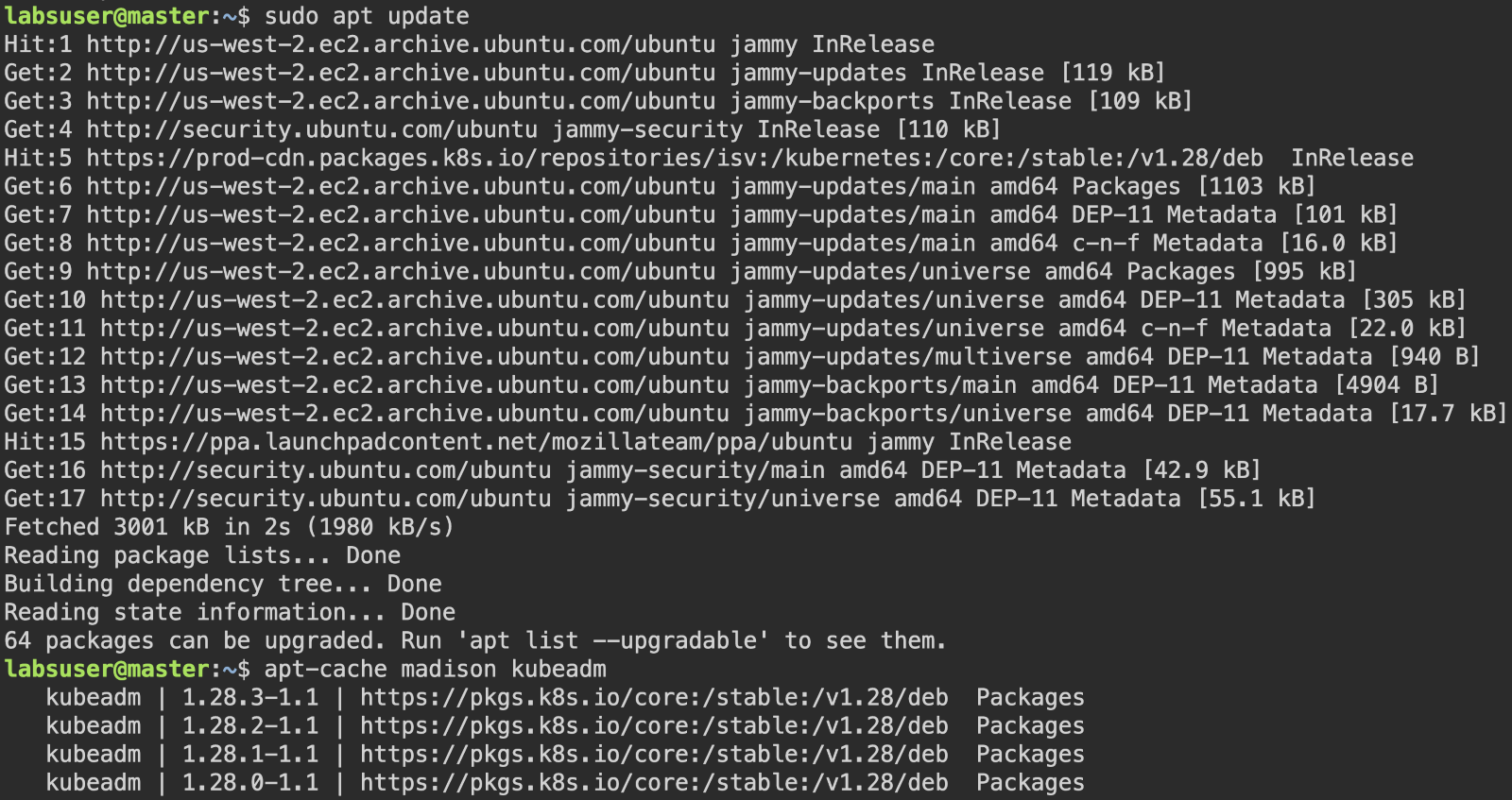
Steps to be followed:

1. Upgrade control plane to v1.28.12
2. Upgrade worker node to v1.28.12
3. Validate upgrade, by creating a pod

**Step 1:** **Upgrade control plane to v1.28.12**

sudo apt update

sudo apt-cache madison kubeadm



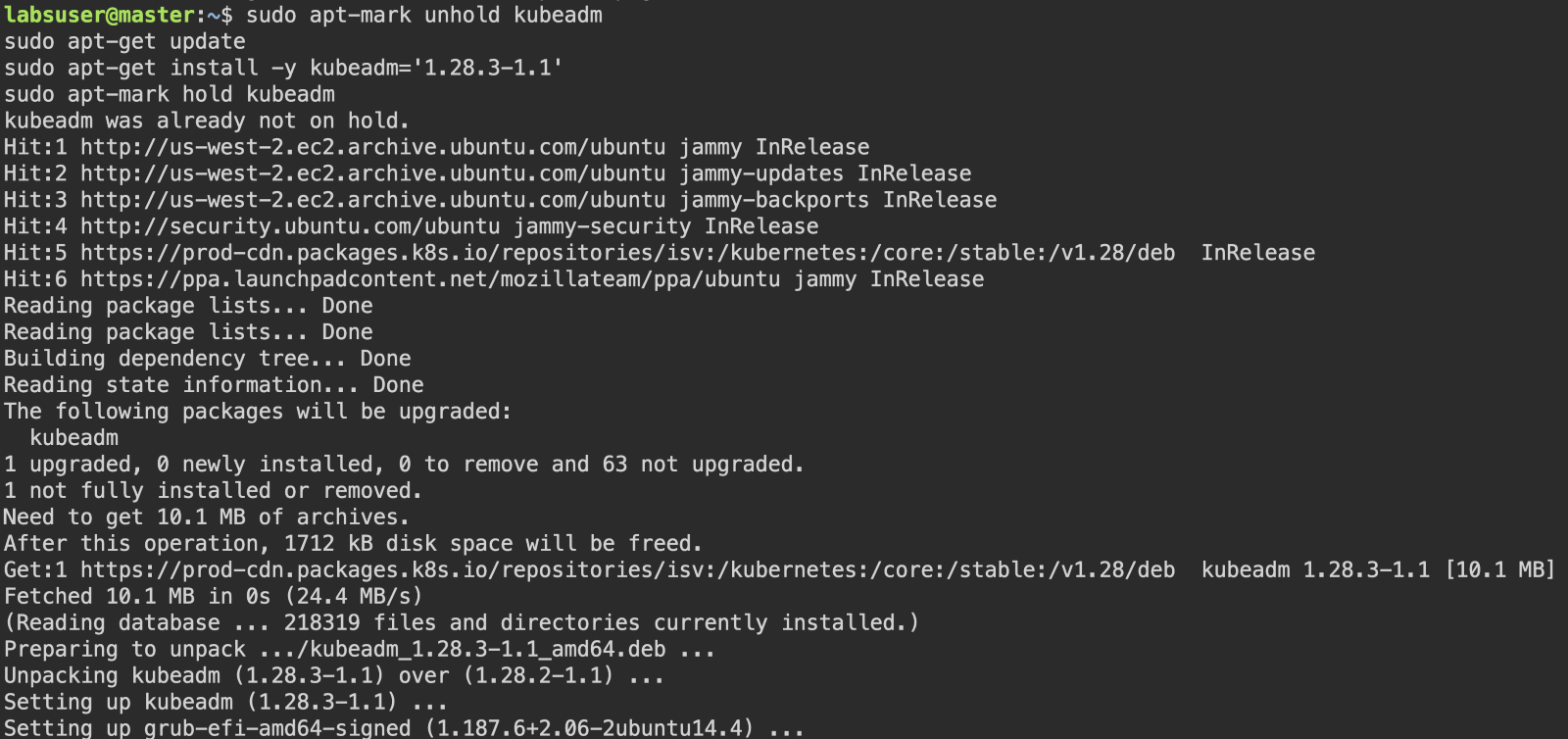
Pick latest version from previous output, We are choosing 1.28.12-1.1 version for upgrade

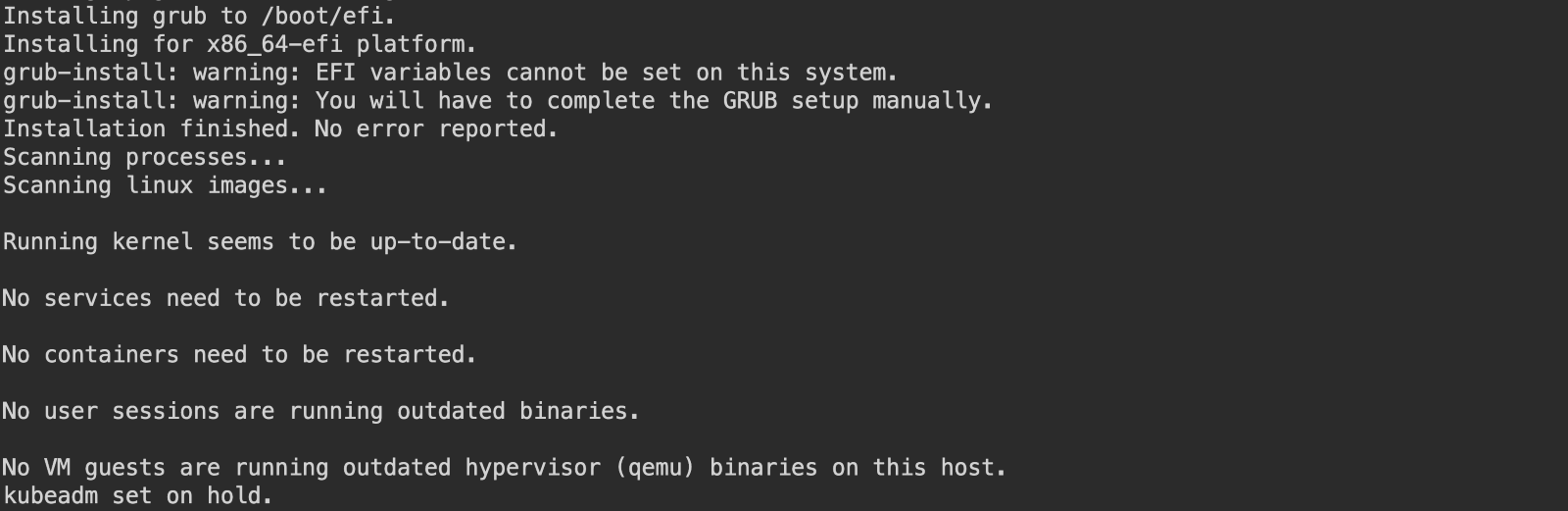
sudo apt-mark unhold kubeadm

sudo apt-get update

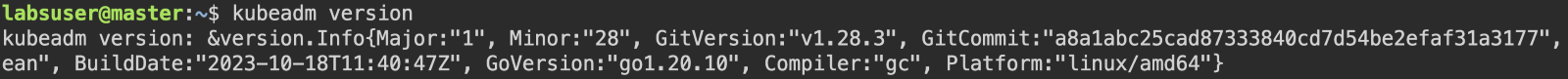
sudo apt-get install -y kubeadm='1.28.12-1.1'

sudo apt-mark hold kubeadm

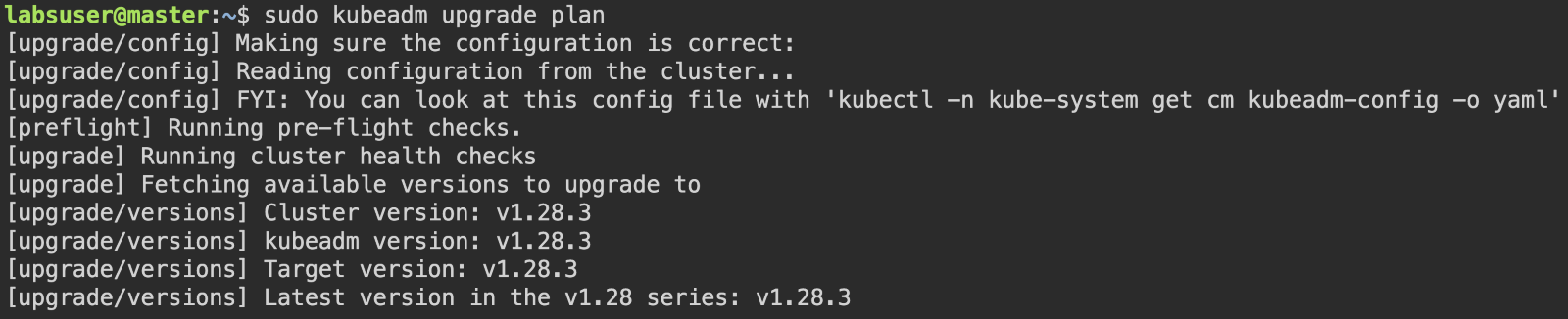




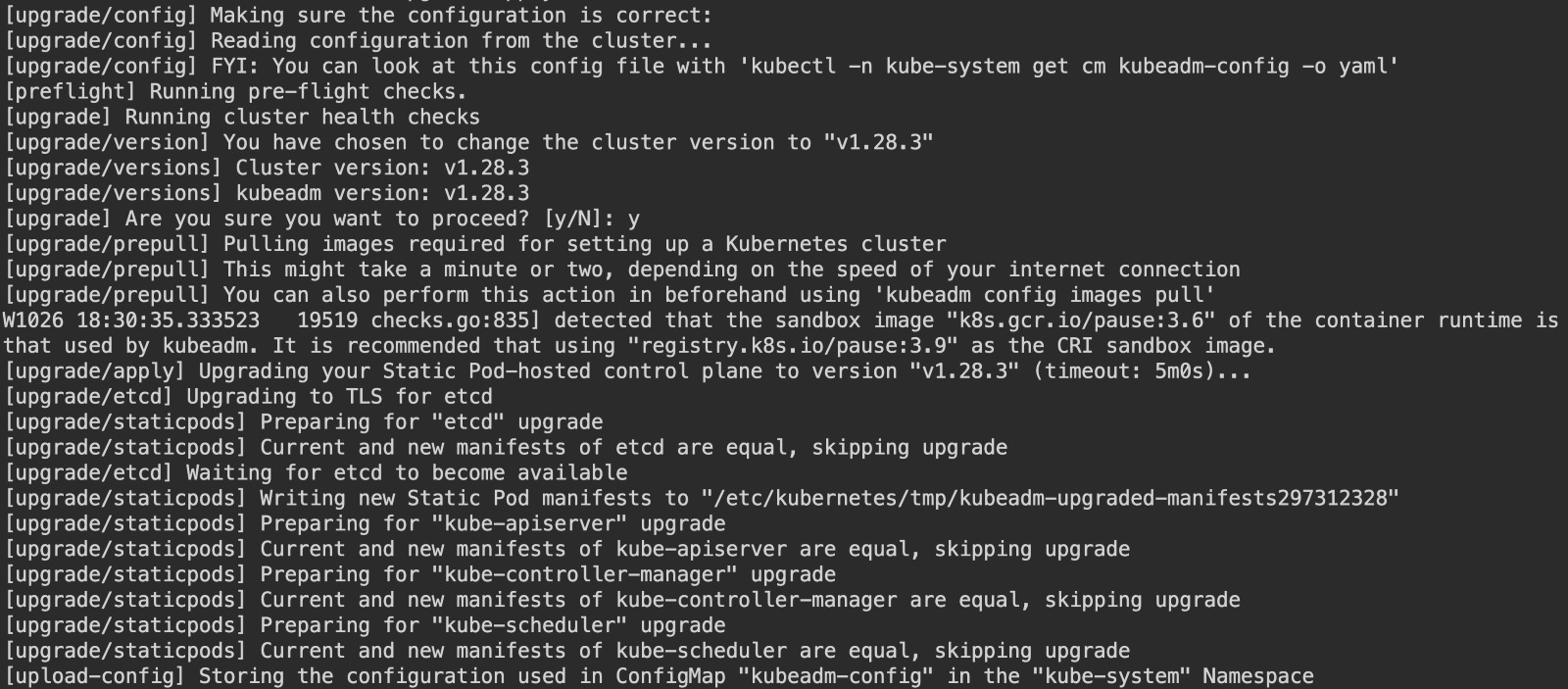
kubeadm version

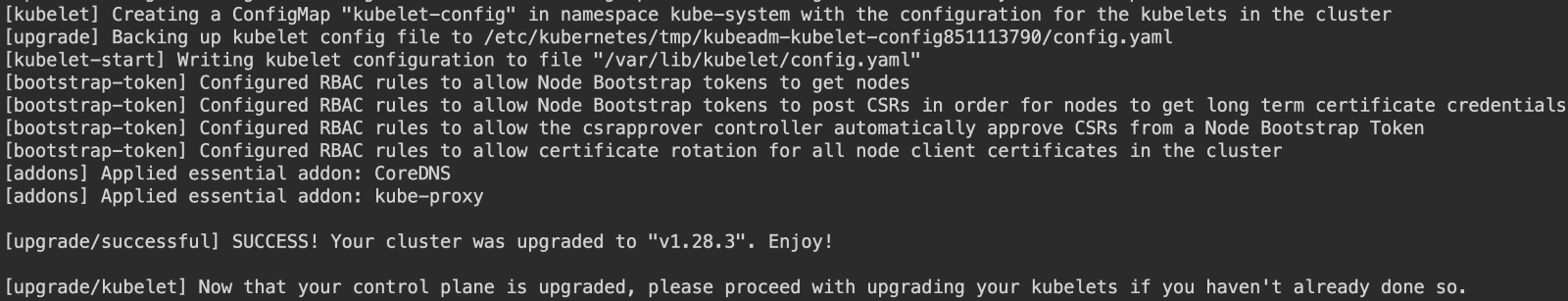


sudo kubeadm upgrade plan

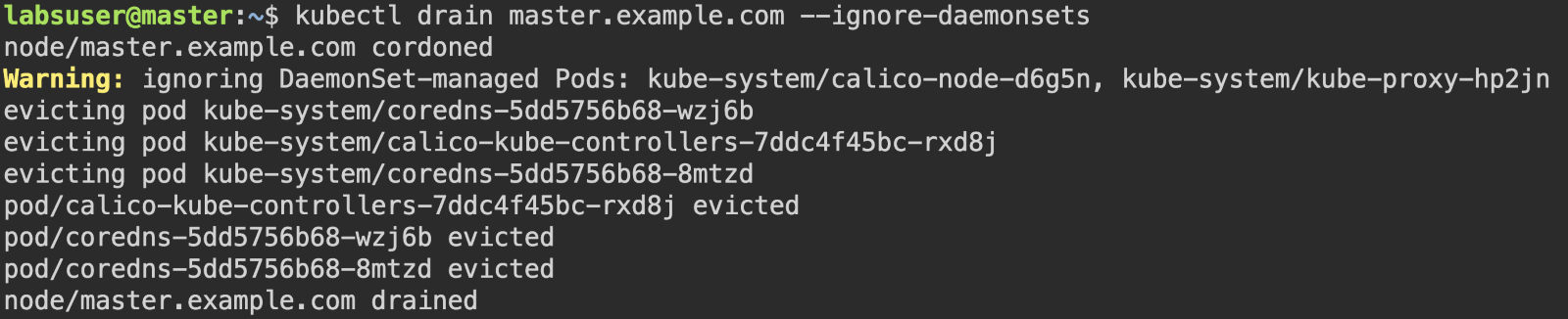


sudo kubeadm upgrade apply v1.28.12





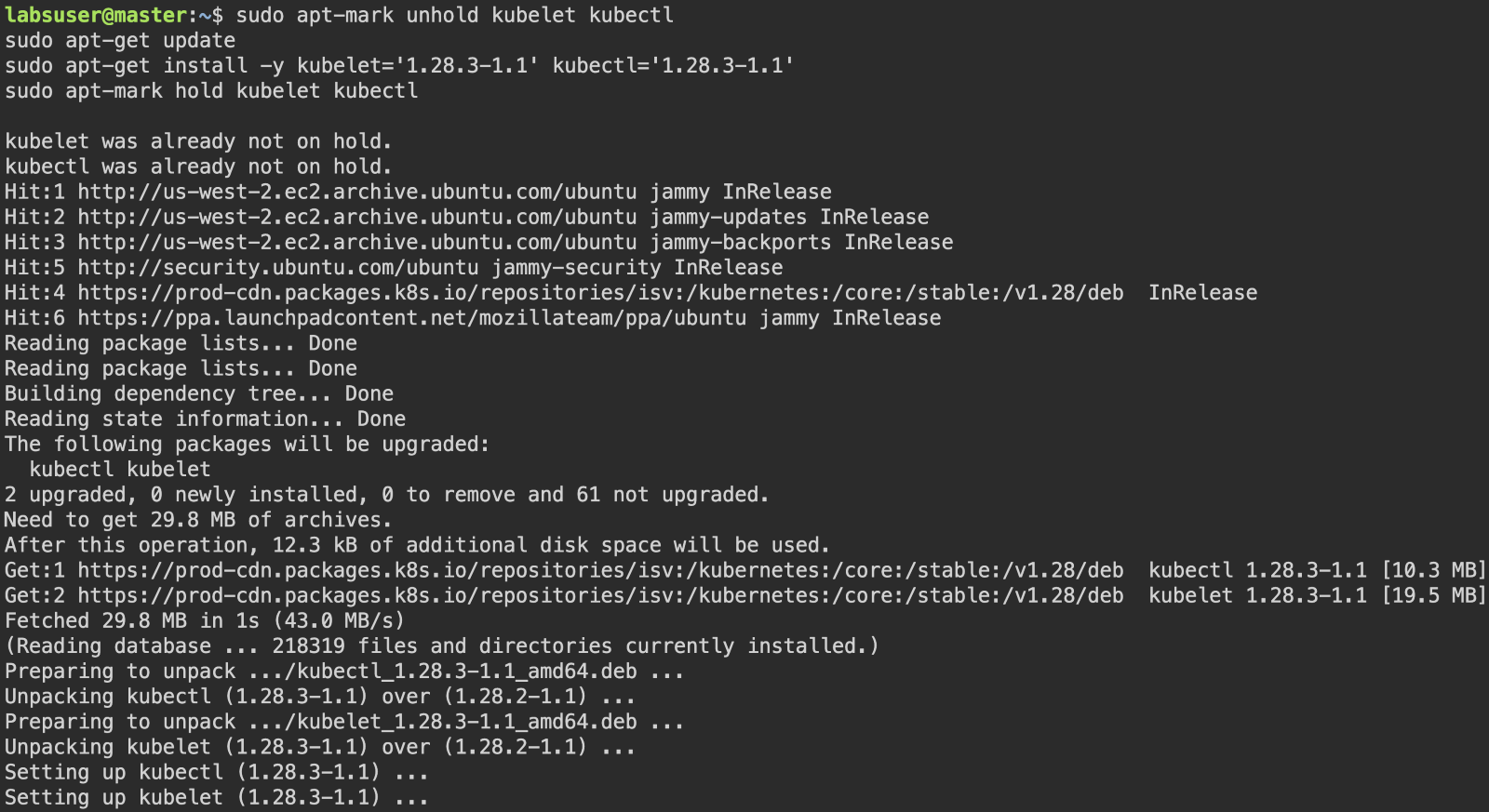
kubectl drain master.example.com --ignore-daemonsets



sudo apt-mark unhold kubelet kubectl

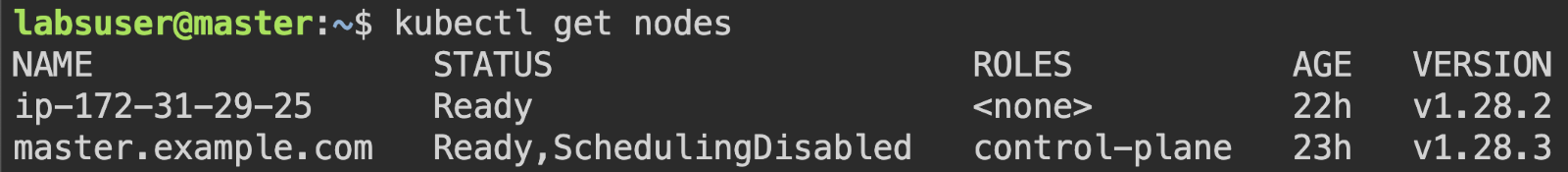
sudo apt-get update

sudo apt-get install -y kubelet='1.28.12-1.1' kubectl='1.28.12-1.1'   
sudo apt-mark hold kubelet kubectl





kubectl get nodes

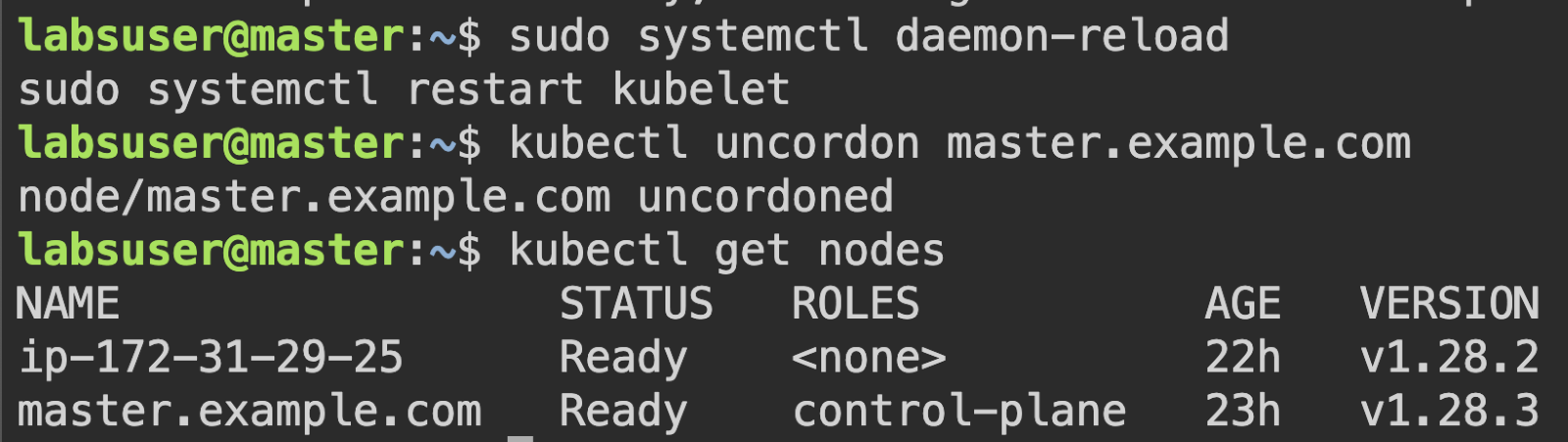


sudo systemctl daemon-reload

sudo systemctl restart kubelet

kubectl uncordon master.example.com

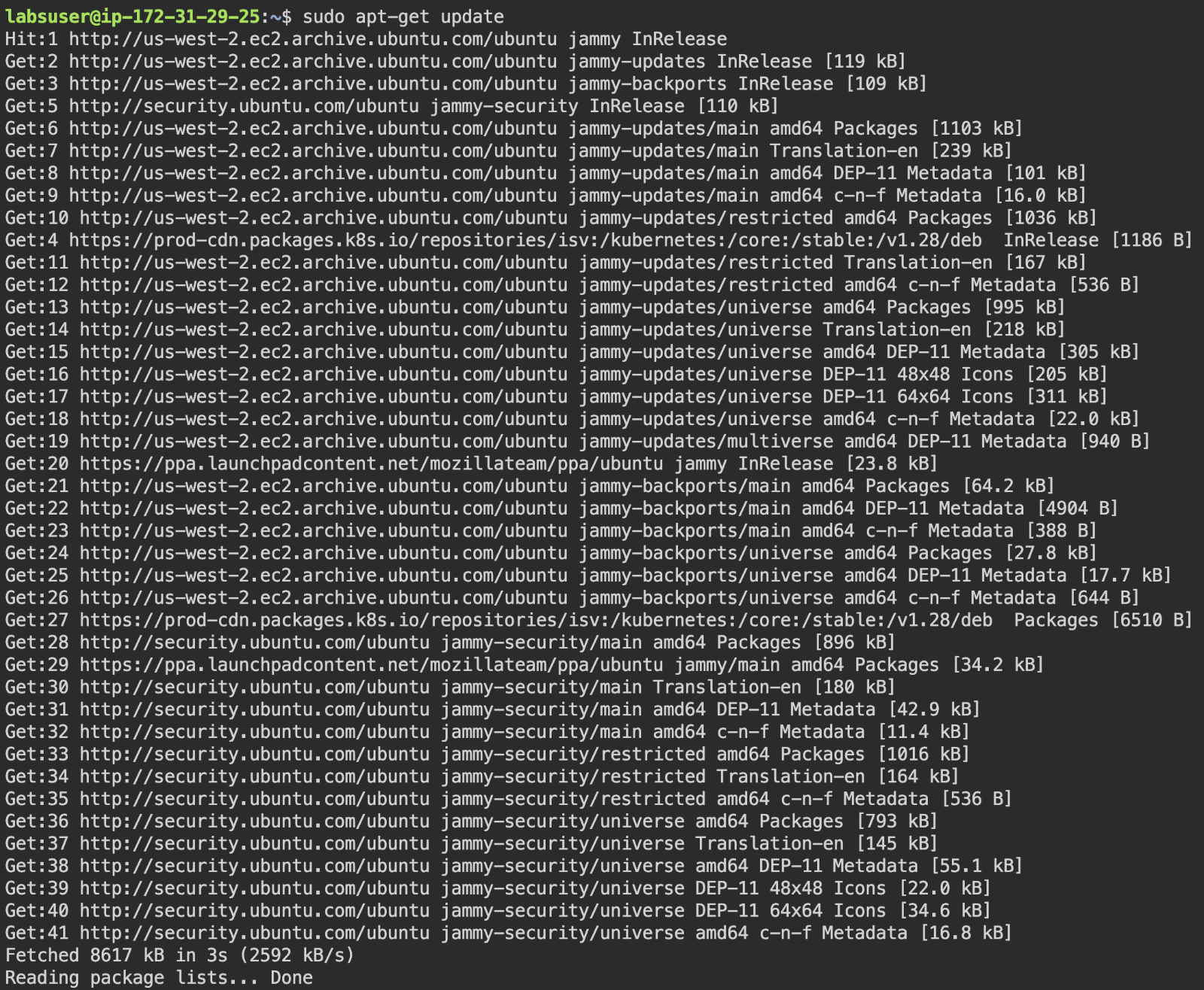
kubectl get nodes



**Step 2:** **Upgrade worker node to v1.28.12**

**On the Worker node,**

sudo apt-get update

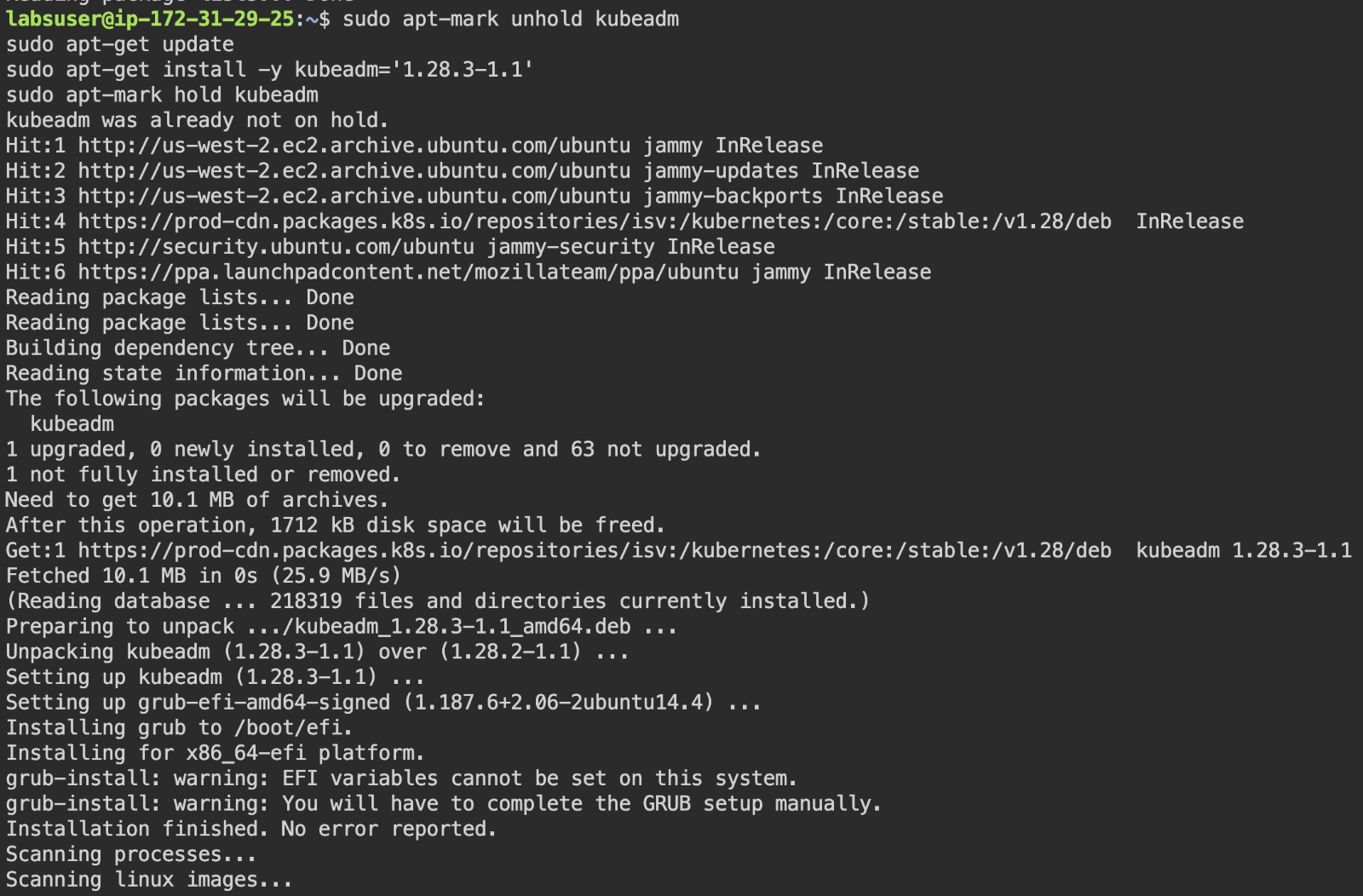


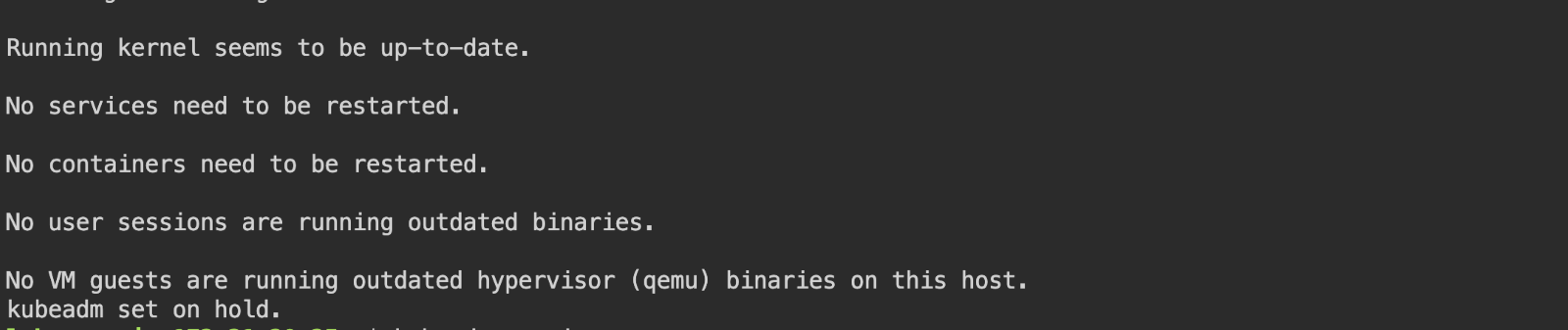
sudo apt-mark unhold kubeadm

sudo apt-get update

sudo apt-get install -y kubeadm='1.28.12-1.1'

sudo apt-mark hold kubeadm

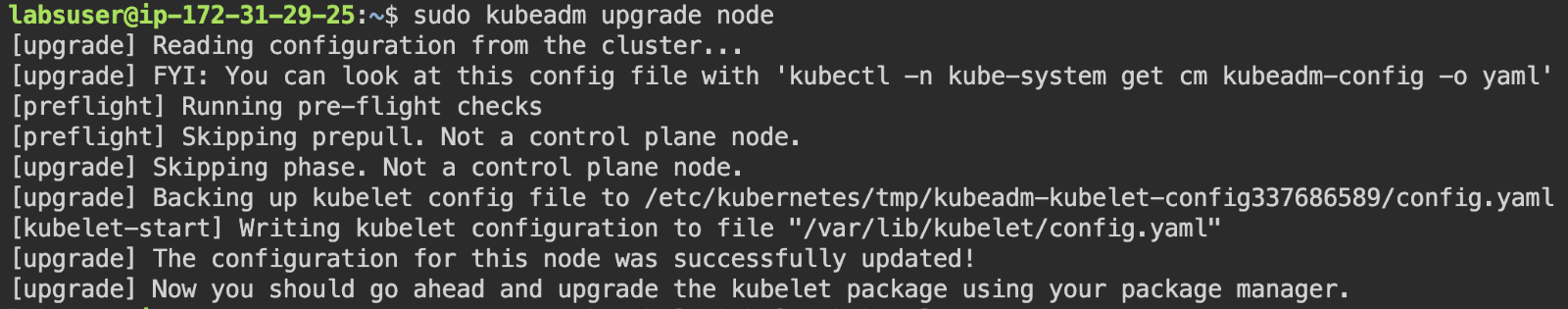




kubeadm version



sudo kubeadm upgrade node

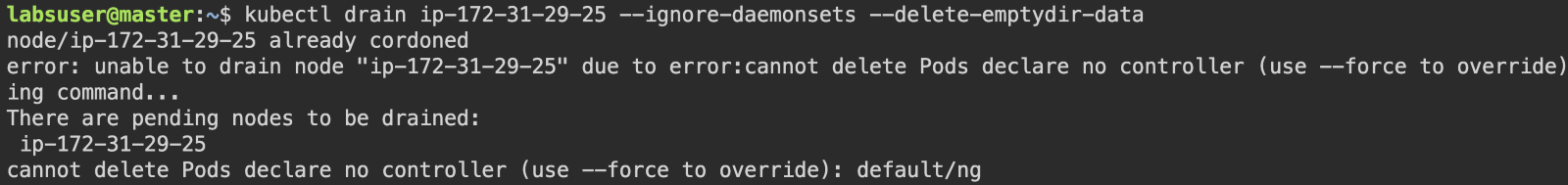


**On the Master node,**

kubectl drain worker-node-1.example.com --ignore-daemonsets --delete-emptydir-data

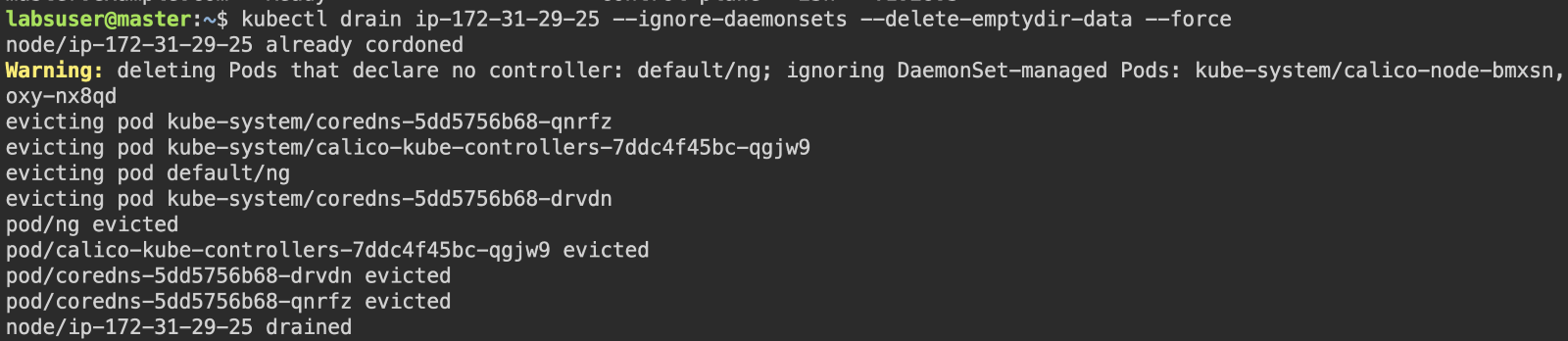
Note: While executing the above drain command, if there are pods cannot be deleted,

Error message: *cannot delete Pods declare no controller (use --force to override):*



Use the below command,

kubectl drain worker-node-1.example.com --ignore-daemonsets --delete-emptydir-data - - force



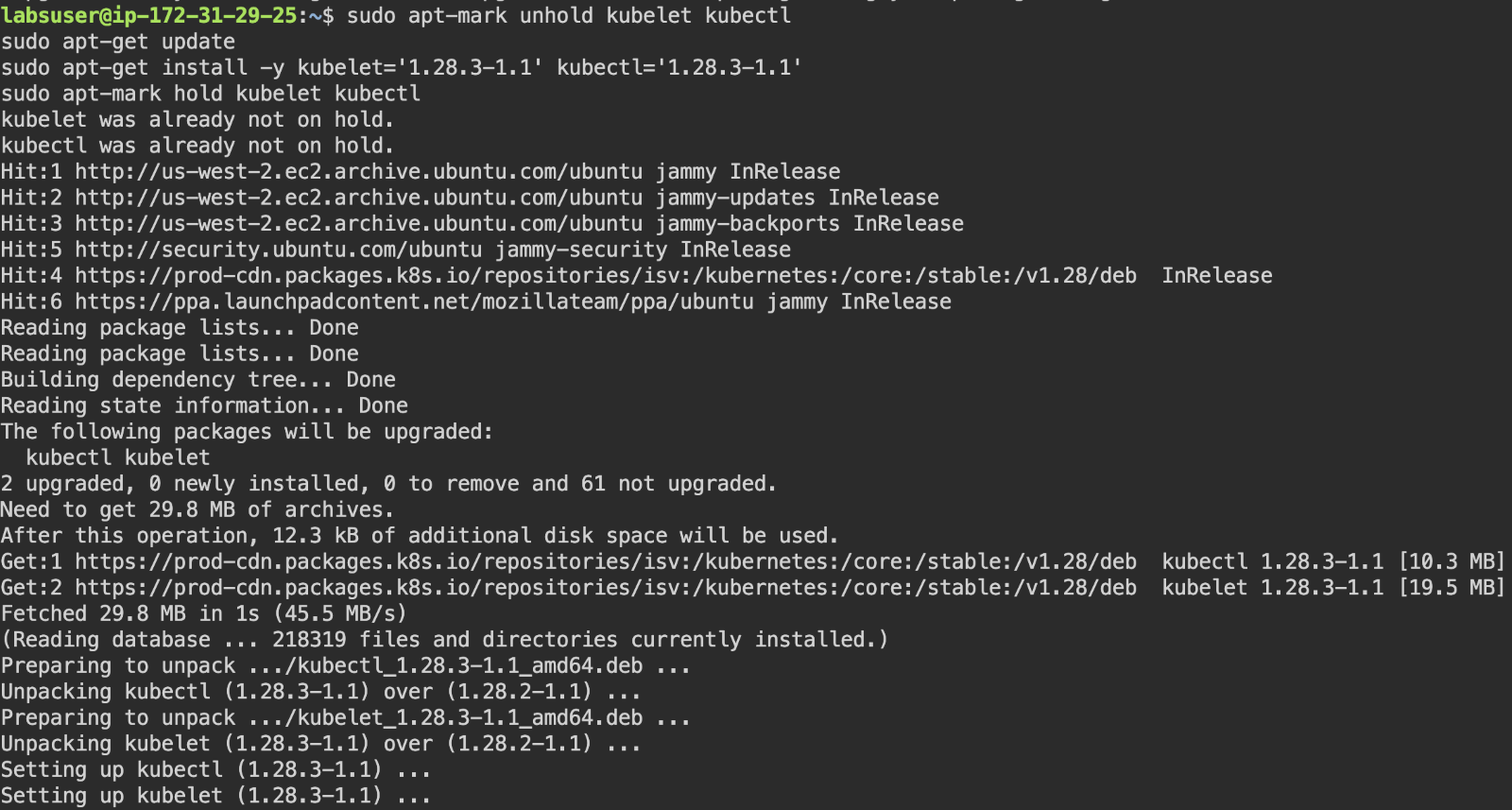
**On the Worker node,**

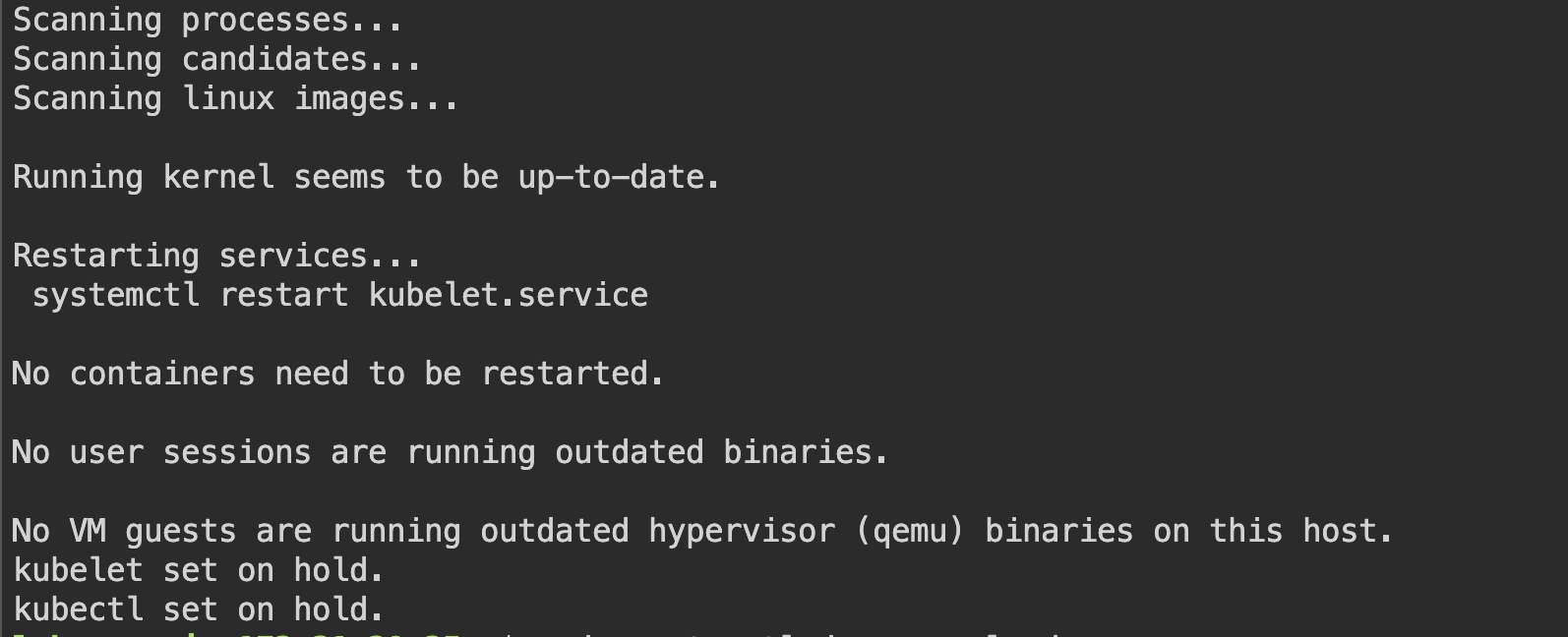
sudo apt-mark unhold kubelet kubectl

sudo apt-get update

sudo apt-get install -y kubelet='1.28.12-1.1' kubectl='1.28.12-1.1'

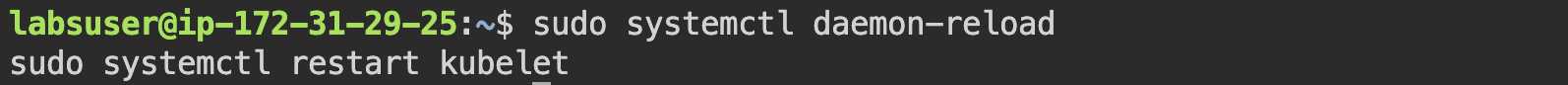
sudo apt-mark hold kubelet kubectl





sudo systemctl daemon-reload

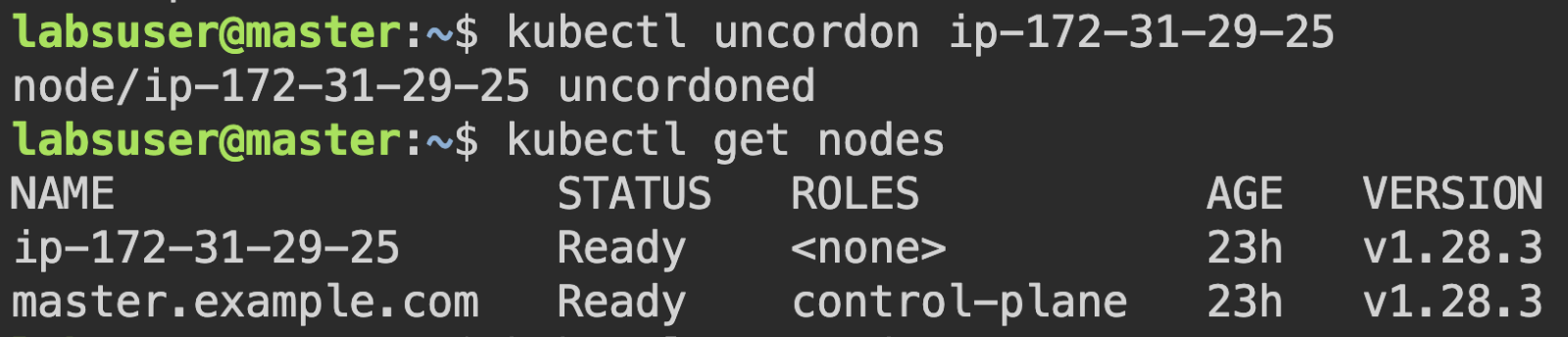
sudo systemctl restart kubelet



**On the Master node,**

kubectl uncordon worker-node-1.example.com

kubectl get nodes

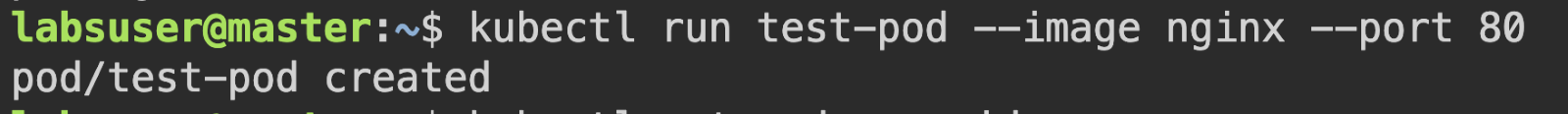


Cluster and Worker nodes are successfully upgraded to v1.28.3.

**Step 3:** **Validate cluster upgrade, by creating pod**

**On the Master node,**

kubectl run test-pod --image nginx --port 80



kubectl get pods -o wide

