**🧪 Lab Guide: ETCD Backup and Restore in a Kubernetes Cluster (Static Pods)**

**🧾 Prerequisites**

* Kubernetes control plane set up with static pods (kube-apiserver, etcd, kube-controller-manager, kube-scheduler).
* Root or sudo access to the control plane node.
* etcdctl CLI installed (apt install etcd-client).
* Docker or container runtime access (for checking containers if using Docker).

**🔍 Step 1: Locate Static Pod Manifests**

Static pods are managed directly by kubelet, and their manifests are stored on disk.

**✅ Check kubelet configuration:**

bash

CopyEdit

cat /var/lib/kubelet/config.yaml | grep staticPodPath

Expected output:

yaml

CopyEdit

staticPodPath: /etc/kubernetes/manifests

All static pod definitions like etcd.yaml, kube-apiserver.yaml are in this directory.

**🔧 Step 2: Verify etcd Pod and Configuration**

**✅ Check running etcd process:**

bash

CopyEdit

ps -ef | grep etcd

From the output, note the data directory used by etcd, usually:

ini

CopyEdit

--data-dir=/var/lib/etcd

**💾 Step 3: Perform ETCD Backup**

**✅ Set ETCD API version:**

bash

CopyEdit

export ETCDCTL\_API=3

**✅ Execute snapshot backup:**

bash

CopyEdit

etcdctl --endpoints=https://127.0.0.1:2379 \

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

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

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

snapshot save /root/myclust.db

**📝 Verify backup file:**

bash

CopyEdit

ls -lh /root/myclust.db

**🎯 Step 4: Simulate Disaster (Delete All User Deployments)**

**⚠️ This is a destructive operation.**

bash

CopyEdit

kubectl get deployments --all-namespaces --no-headers | \

awk '$1 != "kube-system" {print "kubectl delete deployment " $2 " -n " $1}' | bash

**♻️ Step 5: Restore ETCD Snapshot**

**✅ Identify existing etcd data directory:**

bash

CopyEdit

ps -ef | grep etcd

# Look for --data-dir, usually /var/lib/etcd

**✅ Do NOT manually create the restore directory:**

bash

CopyEdit

etcdctl snapshot restore /root/myclust.db --data-dir /var/lib/etcd-new

This will auto-create /var/lib/etcd-new and populate it.

**🔧 Step 6: Reconfigure etcd Static Pod to Use New Data Directory**

**✅ Edit etcd manifest:**

bash

CopyEdit

vi /etc/kubernetes/manifests/etcd.yaml

**🔁 Modify the volume mount to point to the new data directory:**

yaml

CopyEdit

volumes:

- name: etcd-certs

hostPath:

path: /etc/kubernetes/pki/etcd

type: DirectoryOrCreate

- name: etcd-data

hostPath:

path: /var/lib/etcd-new

type: DirectoryOrCreate

You only need to change the hostPath.path under the volume etcd-data.

**⏱️ Step 7: Wait for etcd Pod to Restart**

Since you're editing a static pod manifest, kubelet will automatically restart the etcd pod using the new data directory.

**✅ Monitor containers:**

bash

CopyEdit

watch -n 1 docker ps -a

Look for a new etcd container being created with /var/lib/etcd-new.

**✅ Step 8: Validate Cluster State**

**✅ Check etcd health:**

bash

CopyEdit

etcdctl --endpoints=https://127.0.0.1:2379 \

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

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

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

endpoint health

**✅ Verify resources are restored:**

bash

CopyEdit

kubectl get all --all-namespaces

**🧹 Optional: Cleanup & Revert Data Directory (if needed)**

Once validated, you may:

* Remove old data: rm -rf /var/lib/etcd
* Rename restored dir: mv /var/lib/etcd-new /var/lib/etcd
* Update manifest again to revert to original path.

**📌 Summary**

| **Step** | **Task** | **Command/Description** |
| --- | --- | --- |
| 1 | Find static pod path | cat /var/lib/kubelet/config.yaml |
| 2 | Verify etcd process and data-dir | `ps -ef |
| 3 | Backup etcd | etcdctl snapshot save |
| 4 | Simulate failure | Delete all deployments |
| 5 | Restore snapshot | etcdctl snapshot restore |
| 6 | Update manifest | Change hostPath in etcd.yaml |
| 7 | Watch restart | watch -n 1 docker ps -a |
| 8 | Verify cluster state | kubectl get all |

**🧪 Lab Guide: Kubernetes Upgrade to v1.30 with kubeadm**

**🧾 Prerequisites**

* A functional Kubernetes cluster bootstrapped using kubeadm
* Current version: v1.29.x
* Target version: v1.30.0
* Internet access from nodes (for fetching packages)
* Root/sudo access on all nodes

**📍 Cluster Topology**

| **Node** | **Role** | **Current Version** | **Target Version** |
| --- | --- | --- | --- |
| master | Control Plane | v1.29.x | v1.30.0 |
| w1 | Worker Node | v1.29.x | v1.30.0 |
| w2 | Worker Node | v1.29.x | v1.30.0 |

**🔄 Step 1: Prepare the APT Repository (All Nodes)**

Kubernetes now distributes deb packages via pkgs.k8s.io.

**✅ Configure Kubernetes v1.30 APT Repo:**

bash

CopyEdit

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | \

sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF

deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /

EOF

**🧠 Step 2: Upgrade the Control Plane Node (master)**

**🔹 Upgrade kubeadm:**

bash

CopyEdit

sudo apt-mark unhold kubeadm

sudo apt-get update

sudo apt-get install -y kubeadm=1.30.0-1.1

sudo apt-mark hold kubeadm

**🔍 Step 3: Check Upgrade Plan:**

bash

CopyEdit

kubeadm upgrade plan --ignore-preflight-errors=CoreDNSUnsupportedPlugins,CoreDNSMigration

This will:

* Confirm cluster is upgradeable
* Show current vs target versions
* List necessary component changes

**🚀 Step 4: Apply the Upgrade:**

bash

CopyEdit

sudo kubeadm upgrade apply v1.30.0 --ignore-preflight-errors=CoreDNSUnsupportedPlugins,CoreDNSMigration

This command will:

* Upgrade etcd if required
* Apply kube-apiserver, controller-manager, scheduler, and coredns upgrades
* Backup manifests and certs before replacing

**🔁 Step 5: Upgrade kubelet and kubectl:**

bash

CopyEdit

sudo apt-mark unhold kubelet kubectl

sudo apt-get update

sudo apt-get install -y kubelet=1.30.0-1.1 kubectl=1.30.0-1.1

sudo apt-mark hold kubelet kubectl

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl restart kubelet

**✅ Verify Master Node Version:**

bash

CopyEdit

kubectl get nodes

# should show master at v1.30.0

**🧱 Step 6: Upgrade Worker Nodes (Repeat for each)**

**🔹 Step 6.1: Upgrade kubeadm on Worker Node w1:**

bash

CopyEdit

sudo apt-mark unhold kubeadm

sudo apt-get update

sudo apt-get install -y kubeadm=1.30.0-1.1

sudo apt-mark hold kubeadm

**🔒 Step 6.2: Drain Node from Master:**

bash

CopyEdit

kubectl drain w1 --ignore-daemonsets

**🚀 Step 6.3: Upgrade Node Components:**

bash

CopyEdit

sudo kubeadm upgrade node

**🔁 Step 6.4: Upgrade kubelet and kubectl:**

bash

CopyEdit

sudo apt-mark unhold kubelet kubectl

sudo apt-get update

sudo apt-get install -y kubelet=1.30.0-1.1 kubectl=1.30.0-1.1

sudo apt-mark hold kubelet kubectl

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl restart kubelet

**🔓 Step 6.5: Uncordon Node:**

bash

CopyEdit

kubectl uncordon w1

Repeat steps 6.1–6.5 for w2.

**✅ Step 7: Post-Upgrade Validation**

**🔍 Check All Node Versions:**

bash

CopyEdit

kubectl get nodes -o wide

Expected:

bash

CopyEdit

NAME STATUS ROLES AGE VERSION

master Ready control-plane 3d23h v1.30.0

w1 Ready <none> 3d22h v1.30.0

w2 Ready <none> 3d22h v1.30.0

**🧪 Validate CoreDNS, kube-proxy:**

bash

CopyEdit

kubectl get pods -n kube-system -l k8s-app=kube-dns

kubectl get ds -n kube-system -l k8s-app=kube-proxy

**📌 Summary Checklist**

| **Component** | **Upgrade Method** | **Command** |
| --- | --- | --- |
| APT Repo | Configure manually | tee /etc/apt/sources.list.d/... |
| kubeadm | Upgrade individually | apt install kubeadm=... |
| Plan Upgrade | Dry run | kubeadm upgrade plan |
| Apply Upgrade | Control Plane | kubeadm upgrade apply |
| kubelet, kubectl | Upgrade & restart | apt install, systemctl restart |
| Workers | Drain, upgrade, uncordon | kubectl drain/uncordon, kubeadm upgrade node |

lab guide for deploying the **kube-prometheus-stack** using Helm, configuring Prometheus and Grafana, and setting up alerting with Microsoft Teams integration.

**🧰 Prerequisites**

* **Kubernetes Cluster**: Ensure you have a Kubernetes cluster with at least one t3.large node to handle the resource demands of Prometheus and Grafana.
* **Helm 3**: Helm must be installed on your system.

**🛠️ Step 1: Install Helm**

Download and install Helm:[kubernetestraining.io](https://kubernetestraining.io/blog/deploying-the-kube-prometheus-stack-a-comprehensive-guide-to-kubernetes-monitoring?utm_source=chatgpt.com)

bash

CopyEdit

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

./get\_helm.sh

Verify the installation:[UMH Blog+1Grafana Labs Community Forums+1](https://learn.umh.app/course/how-to-install-grafana-alerts-in-microsoft-teams/?utm_source=chatgpt.com)

bash

CopyEdit

helm version

**📦 Step 2: Add Prometheus Community Helm Repository**

Add the Prometheus Community Helm repository:[GitHub+4kubernetestraining.io+4bootvar+4](https://kubernetestraining.io/blog/deploying-the-kube-prometheus-stack-a-comprehensive-guide-to-kubernetes-monitoring?utm_source=chatgpt.com)

bash

CopyEdit

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

helm repo update

**🚀 Step 3: Install kube-prometheus-stack**

Install the kube-prometheus-stack Helm chart version 45.7.1 into the monitoring namespace:

bash

CopyEdit

helm install prometheus prometheus-community/kube-prometheus-stack \

--version 45.7.1 \

--namespace monitoring \

--create-namespace \

--set kubeEtcd.enabled=false

This command deploys Prometheus, Grafana, Alertmanager, and related components.[Grafana Labs Community Forums](https://community.grafana.com/t/grafana-alerting-with-ms-office-teams/147159?utm_source=chatgpt.com)

**🔍 Step 4: Verify Deployment**

Check the status of the deployed pods:

bash

CopyEdit

kubectl get pods -n monitoring

List the services:[Grafana Labs Community Forums+6UMH Blog+6Medium+6](https://learn.umh.app/course/how-to-install-grafana-alerts-in-microsoft-teams/?utm_source=chatgpt.com)

bash

CopyEdit

kubectl get svc -n monitoring

**🌐 Step 5: Expose Prometheus and Grafana Services**

**🔧 Modify Prometheus Service**

Edit the prometheus-operated service to change its type to NodePort:[GitHub](https://github.com/prometheus-community/helm-charts/issues/3401?utm_source=chatgpt.com)

bash

CopyEdit

kubectl edit svc -n monitoring prometheus-operated

In the editor, change:

yaml

CopyEdit

spec:

type: ClusterIP

clusterIP: None

clusterIPs:

- None

To:

yaml

CopyEdit

spec:

type: NodePort

Save and exit the editor.

Check the updated service:

bash

CopyEdit

kubectl get svc -n monitoring prometheus-operated

Note the NodePort assigned (e.g., 31585).

**🔧 Modify Grafana Service**

Similarly, edit the prometheus-grafana service:

bash

CopyEdit

kubectl edit svc -n monitoring prometheus-grafana

Change the service type to NodePort as shown above.

Retrieve the NodePort:

bash

CopyEdit

kubectl get svc -n monitoring prometheus-grafana

**🔐 Step 6: Access Grafana Dashboard**

Obtain the Grafana admin password:

bash

CopyEdit

kubectl get secret -n monitoring prometheus-grafana -o jsonpath="{.data.admin-password}" | base64 --decode

Access Grafana by navigating to http://<NodeIP>:<NodePort> in your browser.

* **Username**: admin
* **Password**: Output from the command above[Stack Overflow+2Platform9+2GitHub+2](https://platform9.com/kb/kubernetes/how-to-install-upgrade-prometheus-stack-using-helm-chart?utm_source=chatgpt.com)

**📊 Step 7: Import Kubernetes Dashboards into Grafana**

Grafana comes pre-configured with several dashboards. To import additional dashboards:[kubernetestraining.io](https://kubernetestraining.io/blog/deploying-the-kube-prometheus-stack-a-comprehensive-guide-to-kubernetes-monitoring?utm_source=chatgpt.com)

1. In Grafana, go to **Dashboards** > **Import**.
2. Enter the dashboard ID (e.g., 15759 for Kubernetes Node View).
3. Click **Load** and then **Import**.

You can find more dashboards at:

* [Grafana Dashboards for Kubernetes](https://github.com/dotdc/grafana-dashboards-kubernetes)
* [Modern Grafana Dashboards for Kubernetes](https://0xdc.me/blog/a-set-of-modern-grafana-dashboards-for-kubernetes/)

**📈 Step 8: Create Alerts in Grafana**

1. Navigate to the desired dashboard (e.g., **Kubernetes / Views / Nodes**).
2. Click on the panel you want to set an alert for and select **Edit**.
3. Go to the **Alert** tab and click **Create Alert Rule**.
4. Set the evaluation interval (e.g., every 1 minute).
5. In the **Query** section, use a PromQL expression to define the alert condition. For example, to monitor CPU usage:

promql

CopyEdit

100 \* (1 - avg(rate(node\_cpu\_seconds\_total{mode="idle", instance="172.31.37.208:9100"}[5m])))

1. Define the alert condition (e.g., when the query result is above 80%).
2. Set the **For** duration (e.g., 1 minute).
3. Save the alert rule.

**📣 Step 9: Configure Microsoft Teams Integration for Alerts**

**🔧 Create Incoming Webhook in Microsoft Teams**

1. In Microsoft Teams, go to the desired channel.
2. Click on the ellipsis (...) next to the channel name and select **Connectors**.
3. Find and add the **Incoming Webhook** connector.
4. Provide a name and upload an image (optional), then click **Create**.
5. Copy the generated webhook URL.[YouTube](https://www.youtube.com/watch?v=U6sUFiBkdgk&utm_source=chatgpt.com)[UMH Blog+1Medium+1](https://learn.umh.app/course/how-to-install-grafana-alerts-in-microsoft-teams/?utm_source=chatgpt.com)

**🔧 Add Teams Webhook to Grafana**

1. In Grafana, navigate to **Alerting** > **Contact points**.
2. Click **New contact point**.
3. Set a name (e.g., Teams).
4. Select **Microsoft Teams** as the contact point type.
5. Paste the webhook URL obtained from Teams.
6. Click **Save contact point**.[UMH Blog](https://learn.umh.app/course/how-to-install-grafana-alerts-in-microsoft-teams/?utm_source=chatgpt.com)[Grafana Labs Community Forums+3Reddit+3Medium+3](https://www.reddit.com/r/grafana/comments/15n5wik/use_adaptive_card_templating_in_microsoft_teams/?utm_source=chatgpt.com)[Medium+2Grafana Labs Community Forums+2Reddit+2](https://community.grafana.com/t/grafana-alerting-with-ms-office-teams/147159?utm_source=chatgpt.com)

**🔧 Configure Notification Policies**

1. Go to **Alerting** > **Notification policies**.
2. Click **New policy**.
3. Set the **Matching labels** (e.g., teams=kube).
4. Under **Contact points**, select the Teams contact point created earlier.
5. Click **Save policy**.[Medium+1UMH Blog+1](https://medium.com/towardsdev/sending-grafana-notifications-on-teams-channel-cefd88749420?source=user_profile---------13----------------------------&utm_source=chatgpt.com)

**🔧 Assign Labels to Alert Rules**

When creating or editing an alert rule, ensure that the **Labels** section includes the label used in the notification policy (e.g., teams=kube). This ensures that alerts are routed to the correct contact point.

**✅ Step 10: Test the Setup**

To test the alerting setup:

1. Trigger an alert condition (e.g., simulate high CPU usage).
2. Verify that the alert appears in Grafana.
3. Check the Microsoft Teams channel for the alert notification.

**📚 Additional Resources**

* **kube-prometheus-stack Helm Chart**: [Artifact Hub](https://artifacthub.io/packages/helm/prometheus-community/kube-prometheus-stack/45.7.1)
* **Grafana Microsoft Teams Integration**: [Grafana Documentation](https://grafana.com/docs/grafana/latest/alerting/notifications/microsoft-teams/)
* **Prometheus Community Helm Charts**: [GitHub Repository](https://github.com/prometheus-community/helm-charts)[Artifact Hub+1Artifact Hub+1](https://artifacthub.io/packages/helm/prometheus-community/kube-prometheus-stack/45.7.1?utm_source=chatgpt.com)

This lab guide provides a step-by-step approach to setting up a robust monitoring and alerting system using Prometheus, Grafana, and Microsoft Teams. Ensure to tailor the configurations to fit your specific cluster and organizational requirements.[Grafana Labs Community Forums+1YouTube+1](https://community.grafana.com/t/grafana-alerting-with-ms-office-teams/147159?utm_source=chatgpt.com)