**Prometheus Monitoring on Kubernetes**

I assume that you have a kubernetes cluster up and running with kubectl setup on your workstation. If you don’t have a [kubernetes setup](https://devopscube.com/setup-jenkins-on-kubernetes-cluster/), you can set up a cluster on google cloud by [following this article](https://devopscube.com/setup-kubernetes-cluster-google-cloud/).

Latest Prometheus is available as a [docker image](https://hub.docker.com/r/prom/prometheus/" \t "_blank) in its official docker hub account. We will use that image for the setup.

**Connect to the Cluster**

Connect to your Kubernetes cluster and set up the proxy for accessing the Kubernetes dashboard.

**Note:** If you are using GKE, you need to run the following commands as you need privileges to create cluster roles.



|  |  |
| --- | --- |
| 1  2  3  4 | ACCOUNT=$(gcloud info --format='value(config.account)')  kubectl create clusterrolebinding owner-cluster-admin-binding \      --clusterrole cluster-admin \      --user $ACCOUNT |

Let’s get started with the setup.

***Note****: All the configuration files I mentioned in this guide is hosted on [Github](https://github.com/bibinwilson/kubernetes-prometheus" \t "_blank). You can clone the repo using the following command. Thanks to*[*James*](https://github.com/eggsy84)*for contributing to this repo. Please don’t hesitate to contribute to the repo for adding features. You can use the config files from the github repo or create the files on the go as mentioned in the steps.*



|  |  |
| --- | --- |
| 1 | git clone https://github.com/bibinwilson/kubernetes-prometheus |

**Create a Namespace**

First, we will create a Kubernetes namespace for all our monitoring components. Execute the following command to create a new namespace called monitoring.



|  |  |
| --- | --- |
| 1 | kubectl create namespace monitoring |

You need to assign cluster reader permission to this namespace so that Prometheus can fetch the metrics from Kubernetes API’s.

**Step 1:** Create a file named clusterRole.yaml and copy the content of this file –> [ClusterRole Config](https://raw.githubusercontent.com/bibinwilson/kubernetes-prometheus/master/clusterRole.yaml)

**Step 2:** Create the role using the following command.



|  |  |
| --- | --- |
| 1 | kubectl create -f clusterRole.yaml |

**Create a Config Map**

We should create a config map with all the [prometheus scrape config](https://raw.githubusercontent.com/bibinwilson/kubernetes-prometheus/master/config-map.yaml" \t "_blank) and alerting rules, which will be mounted to the Prometheus container in /etc/prometheus as prometheus.yaml and prometheus.rules files.

**Step 1:**Create a file called config-map.yaml and copy the contents of this file –> [Prometheus Config File](https://raw.githubusercontent.com/bibinwilson/kubernetes-prometheus/master/config-map.yaml)

**Step 2:**Execute the following command to create the config map in Kubernetes.



|  |  |
| --- | --- |
| 1 | kubectl create -f config-map.yaml |

The prometheus.yaml contains all the configuration to dynamically discover pods and services running in the Kubernetes cluster. We have the following [scrape jobs](https://prometheus.io/docs/concepts/jobs_instances/) in our Prometheus scrape configuration.

1. kubernetes-apiservers: It gets all the metrics from the API servers.
2. kubernetes-nodes: All Kubernetes node metrics will be collected with this job.
3. kubernetes-pods: All the pod metrics will be discovered if the pod metadata is annotated with prometheus.io/scrape and prometheus.io/port annotations.
4. kubernetes-cadvisor: Collects all cAdvisor metrics.
5. kubernetes-service-endpoints: All the Service endpoints will be scrapped if the service metadata is annotated with prometheus.io/scrape and prometheus.io/port annotations. It will be a blackbox monitoring.

prometheus.rules will contain all the alert rules for sending alerts to alert manager.

**Create a Prometheus Deployment**

**Step 1:** Create a file named prometheus-deployment.yaml and copy the following contents onto the file. In this configuration, we are mounting the Prometheus config map as a file inside /etc/prometheus. It uses the official Prometheus image from docker hub.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | apiVersion: extensions/v1beta1  kind: Deployment  metadata:    name: prometheus-deployment    namespace: monitoring  spec:    replicas: 1    template:      metadata:        labels:          app: prometheus-server      spec:        containers:          - name: prometheus            image: prom/prometheus:v2.12.0            args:              - "--config.file=/etc/prometheus/prometheus.yml"              - "--storage.tsdb.path=/prometheus/"            ports:              - containerPort: 9090            volumeMounts:              - name: prometheus-config-volume                mountPath: /etc/prometheus/              - name: prometheus-storage-volume                mountPath: /prometheus/        volumes:          - name: prometheus-config-volume            configMap:              defaultMode: 420              name: prometheus-server-conf            - name: prometheus-storage-volume            emptyDir: {} |

*You Might Like****:***[*Kubernetes Deployment Tutorial For Beginners*](https://devopscube.com/kubernetes-deployment-tutorial/)

**Step 2:** Create a deployment on monitoring namespace using the above file.



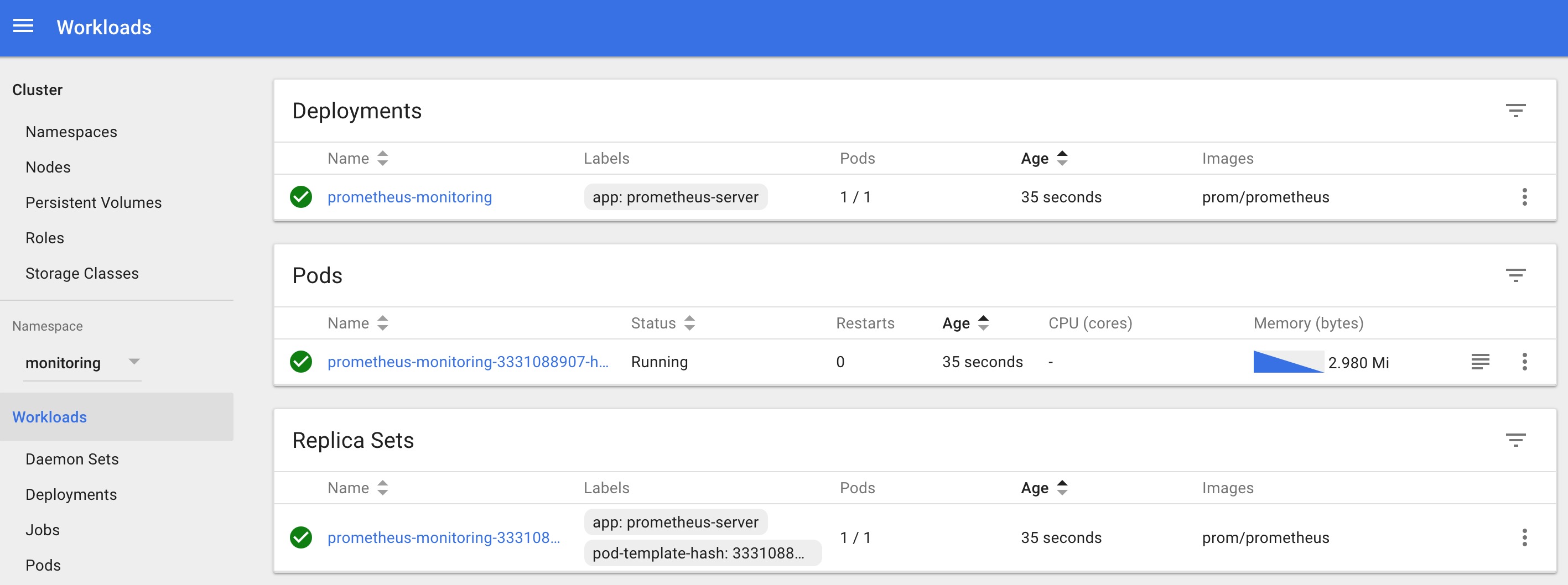
|  |  |
| --- | --- |
| 1 | kubectl create  -f prometheus-deployment.yaml |

**Step 3:** You can check the created deployment using the following command.



|  |  |
| --- | --- |
| 1 | kubectl get deployments --namespace=monitoring |

You can also get details from the kubernetes dashboard like shown below.

[](https://devopscube.com/wp-content/uploads/2017/10/Screen-Shot-2017-10-11-at-11.26.15-AM.jpg)

**Connecting To Prometheus Dashboard**

You can view the deployed Prometheus dashboard in two ways.

1. Using Kubectl port forwarding
2. Exposing the Prometheus deployment as a service with NodePort or a Load Balancer.

We will look at both options.

**Using Kubectl port forwarding**

Using kubectl port forwarding, you can access the pod from your workstation using a selected port on your localhost.

**Step 1:** First, get the Prometheus pod name.



|  |  |
| --- | --- |
| 1 | kubectl get pods --namespace=monitoring |

The output will look like the following.



|  |  |
| --- | --- |
| 1  2  3 | ➜  kubectl get pods --namespace=monitoring  NAME                                     READY     STATUS    RESTARTS   AGE  prometheus-monitoring-3331088907-hm5n1   1/1       Running   0          5m |

**Step 2:** Execute the following command with your pod name to access Prometheus from localhost port 8080.

***Note:****Replace prometheus-monitoring-3331088907-hm5n1 with your pod name.*



|  |  |
| --- | --- |
| 1 | kubectl port-forward prometheus-monitoring-3331088907-hm5n1 8080:9090 -n monitoring |

**Step 3:** Now, if you access http://localhost:8080 on your browser, you will get the Prometheus home page.

**Exposing Prometheus as a Service**

To access the Prometheus dashboard over a IP or a DNS name, you need to expose it as Kubernetes service.

**Step 1:** Create a file named prometheus-service.yaml and copy the following contents. We will expose Prometheus on all kubernetes node IP’s on port 30000.

***Note:****If you are on AWS or Google Cloud, You can use Loadbalancer type, which will create a load balancer and points it to the service.*



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | apiVersion: v1  kind: Service  metadata:    name: prometheus-service    namespace: monitoring    annotations:        prometheus.io/scrape: 'true'        prometheus.io/port:   '9090'  spec:    selector:      app: prometheus-server    type: NodePort    ports:      - port: 8080        targetPort: 9090        nodePort: 30000 |

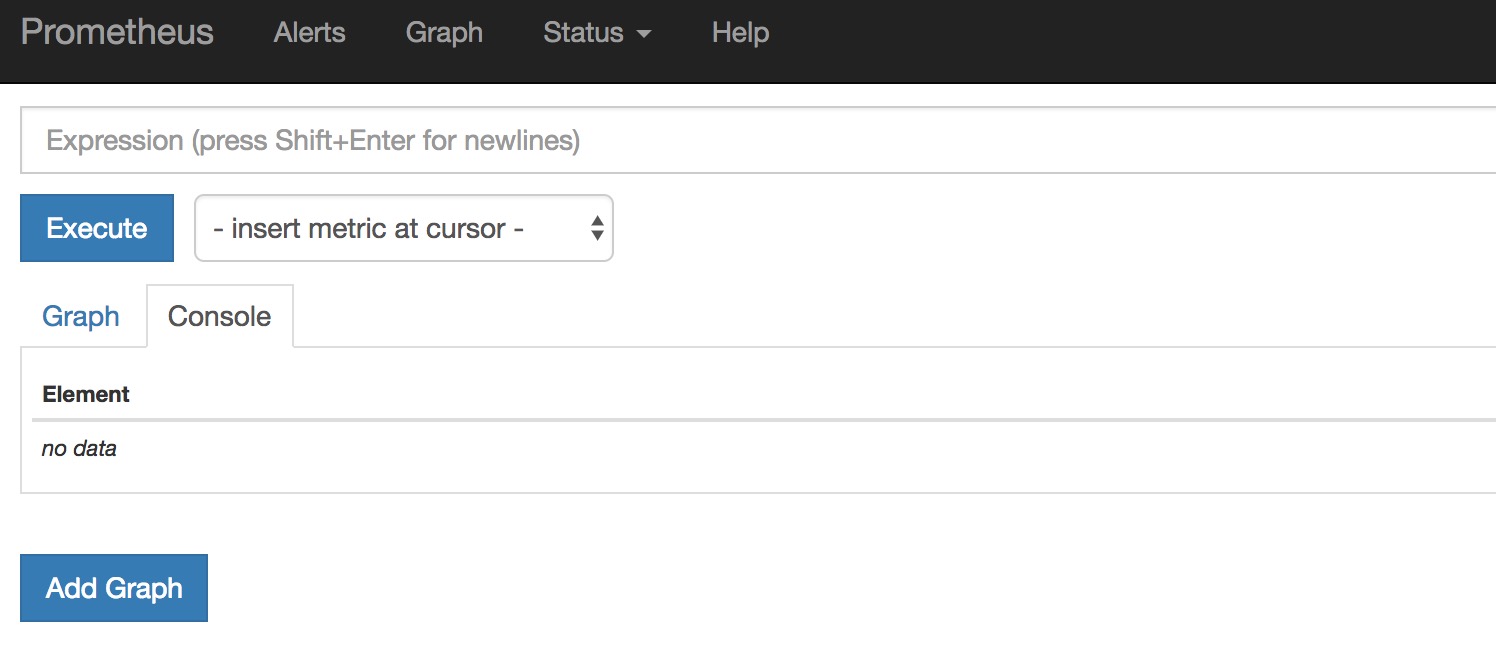
The annotations in the above service YAML makes sure that the service endpoint is scrapped by Prometheus. The prometheus.io/port should always be the target port mentioned in service YAML

**Step 2:** Create the service using the following command.

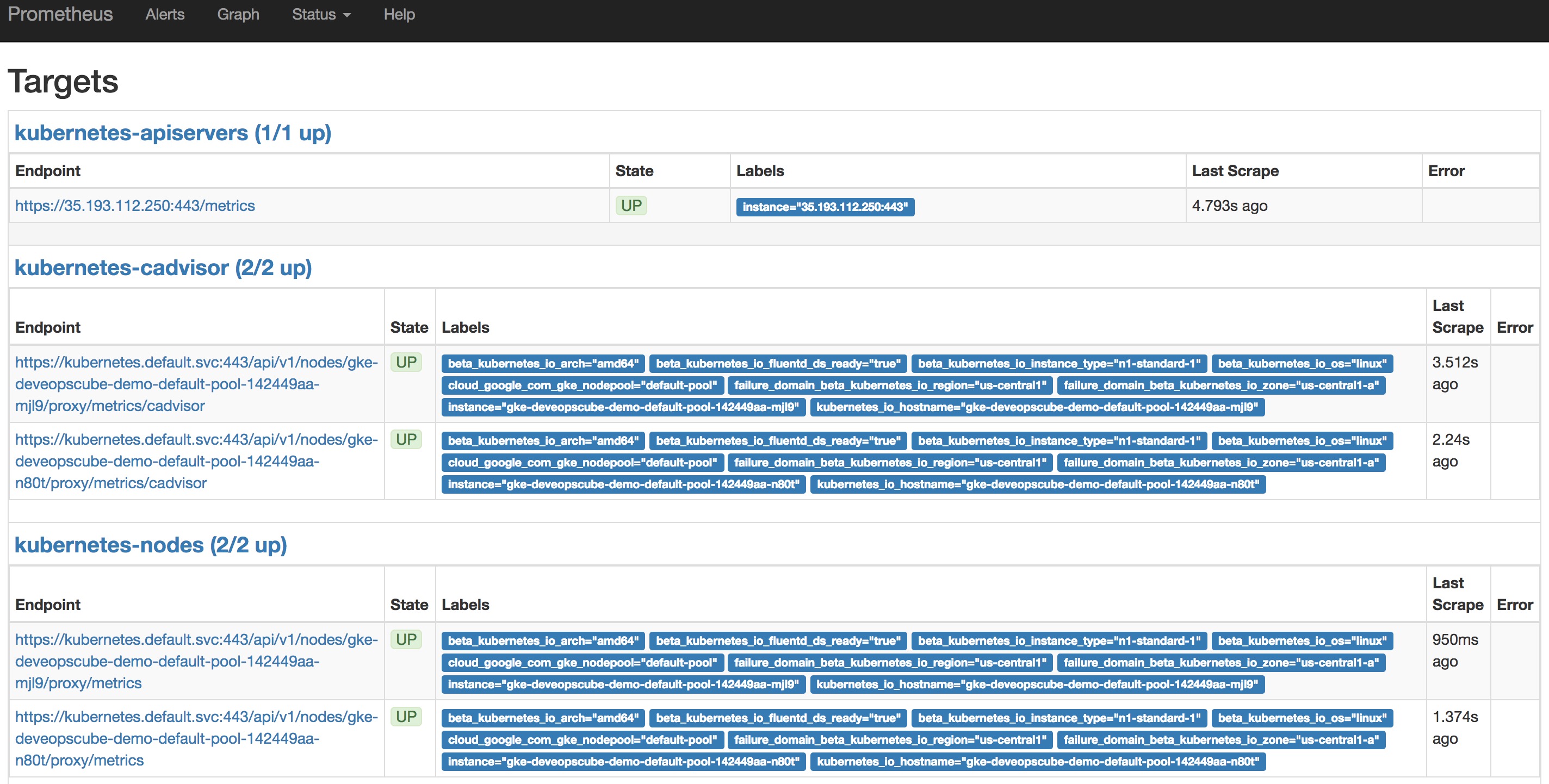


|  |  |
| --- | --- |
| 1 | kubectl create -f prometheus-service.yaml --namespace=monitoring |

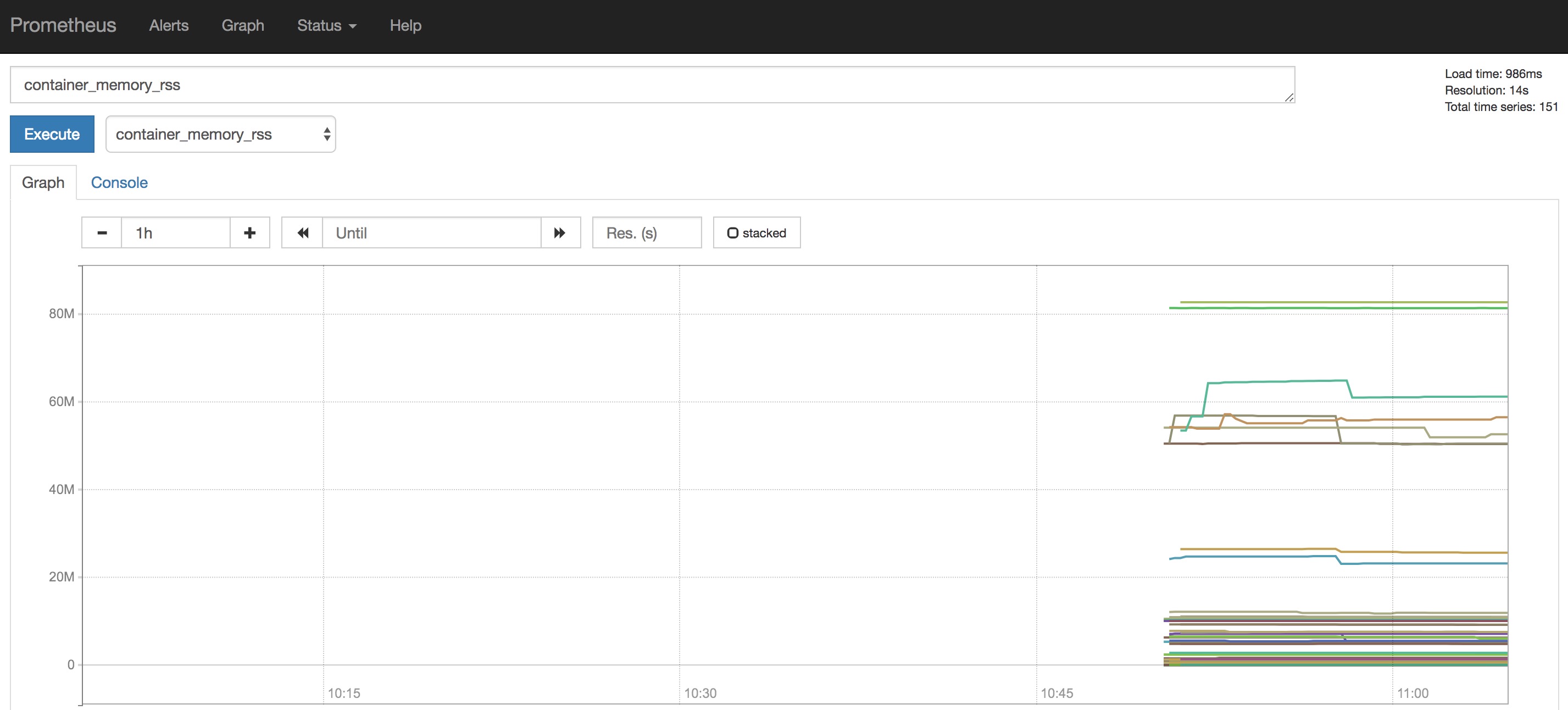
**Step 3:** Once created, you can access the Prometheus dashboard using any Kubernetes node IP on port 30000. If you are on the cloud, make sure you have the right firewall rules for accessing the apps.

[](https://devopscube.com/wp-content/uploads/2017/10/Screen-Shot-2017-10-11-at-12.15.57-PM.jpg)

**Step 4:** Now if you browse to status --> Targets, you will see all the Kubernetes endpoints connected to Prometheus automatically using service discovery as shown below. So you will get all kubernetes container and node metrics in Prometheus.

[](https://devopscube.com/wp-content/uploads/2017/10/Screen-Shot-2017-10-20-at-4.30.36-PM.jpg)

**Step 5:** You can head over the homepage and select the metrics you need from the drop-down and get the graph for the time range you mention. An example graph for container memory utilization is shown below.

[](https://devopscube.com/wp-content/uploads/2017/10/Screen-Shot-2017-10-20-at-4.34.58-PM.jpg)

**Setting Up Alert Manager**

We have covered the Alert Manager setup in a separate article. Please follo