

Prerequisites

For installing Kubernetes in your system, here are a few prerequisites that need special attention. The hardware and software requirements are discussed below:

Hardware requirements

- Master node with at least 2 GB memory. (Additional will be great)
- Worker node with 700 MB memory capacity.
- Your Mouse/Keyboard (monitor navigation)

Software requirements

- Hype-V
- Docker Desktop
- Unique MAC address
- Unique product UUID for every node

Ensuring that there is a full range of connectivity between all the machines in the cluster is a must.

Installation Procedure

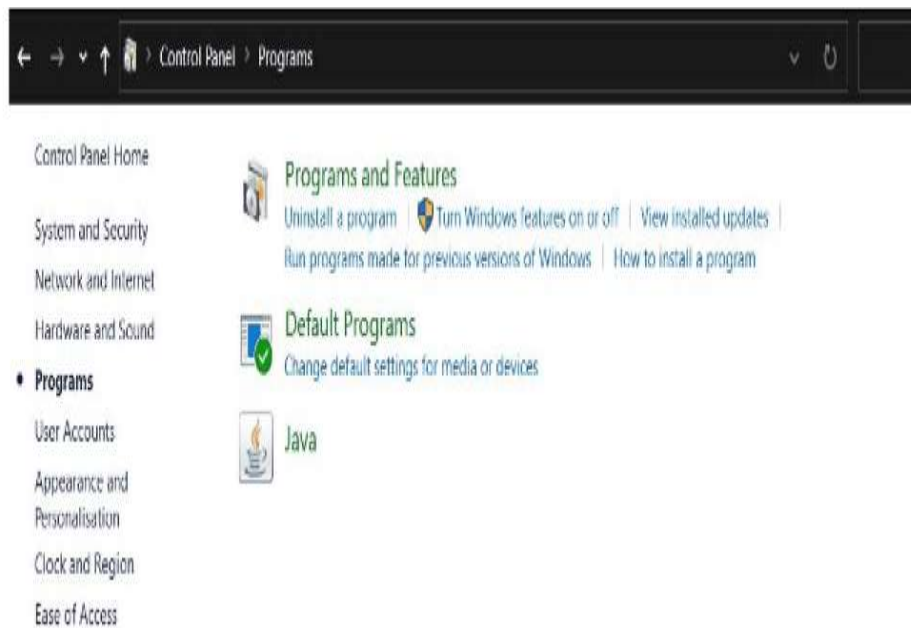
Step 1: Install & Setup Hyper-V

As we all know, Windows has its virtualization software, known as Hyper-V, which is essentially VirtualBox on steroids. Hyper-V allows you to manage your virtual machines (VMs) using either the free Microsoft GUI tool or the command line. It's simple to enable Hyper-V, but first, make sure your PC meets the following requirements:

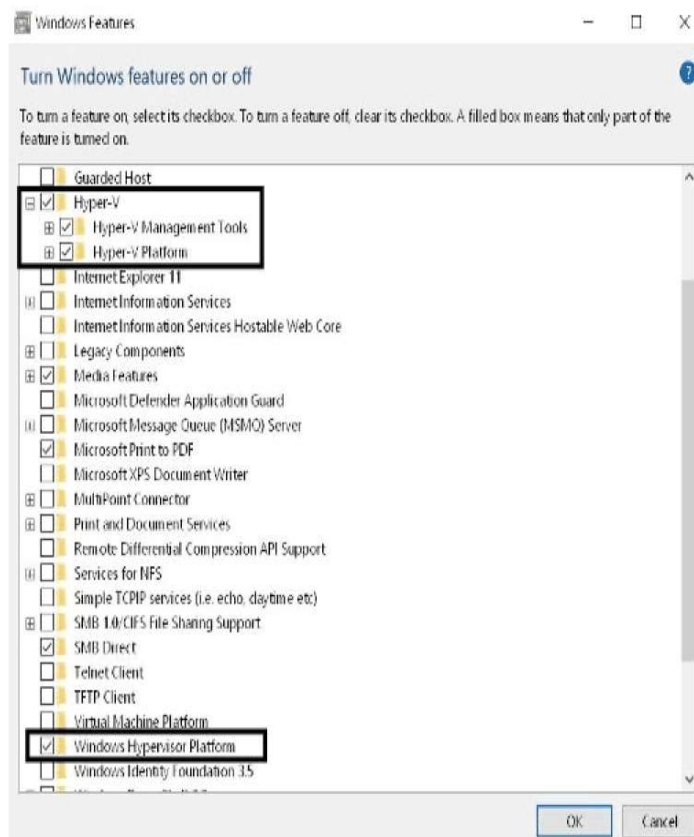
1. Your operating system should be Windows 10 (Enterprise, Pro, or Education), with
2. At least 4GB of RAM and CPU Virtualization support, though you should double-check that it's turned on in your BIOS settings.

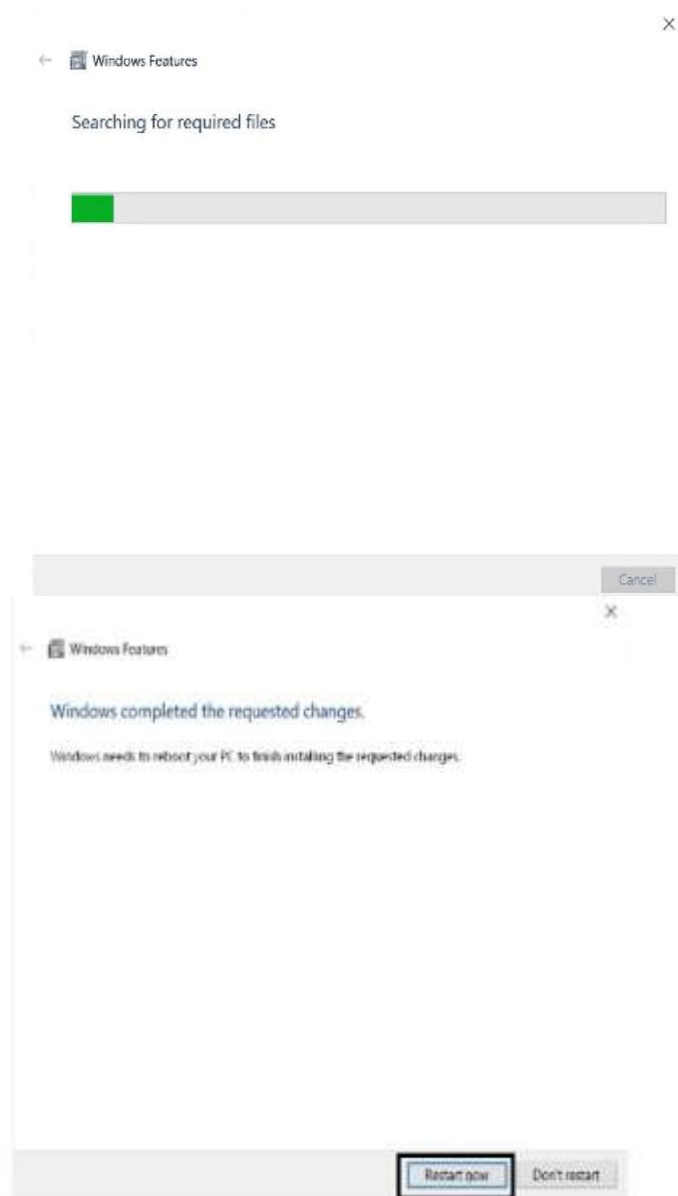
You can disable or enable features like Hyper-V that may not be pre-installed when Windows is installed. Always keep in mind that some of the features require internet access to download additional Windows Update components. To enable Hyper-V on your machine, follow the steps below:

1. Open the Control Panel.
2. Select Programs from the left panel.



3. Next, go to Programs and Features, then Turn Windows Features On or Off.
4. Examine Hyper-V and the Hypervisor Platform for Windows.



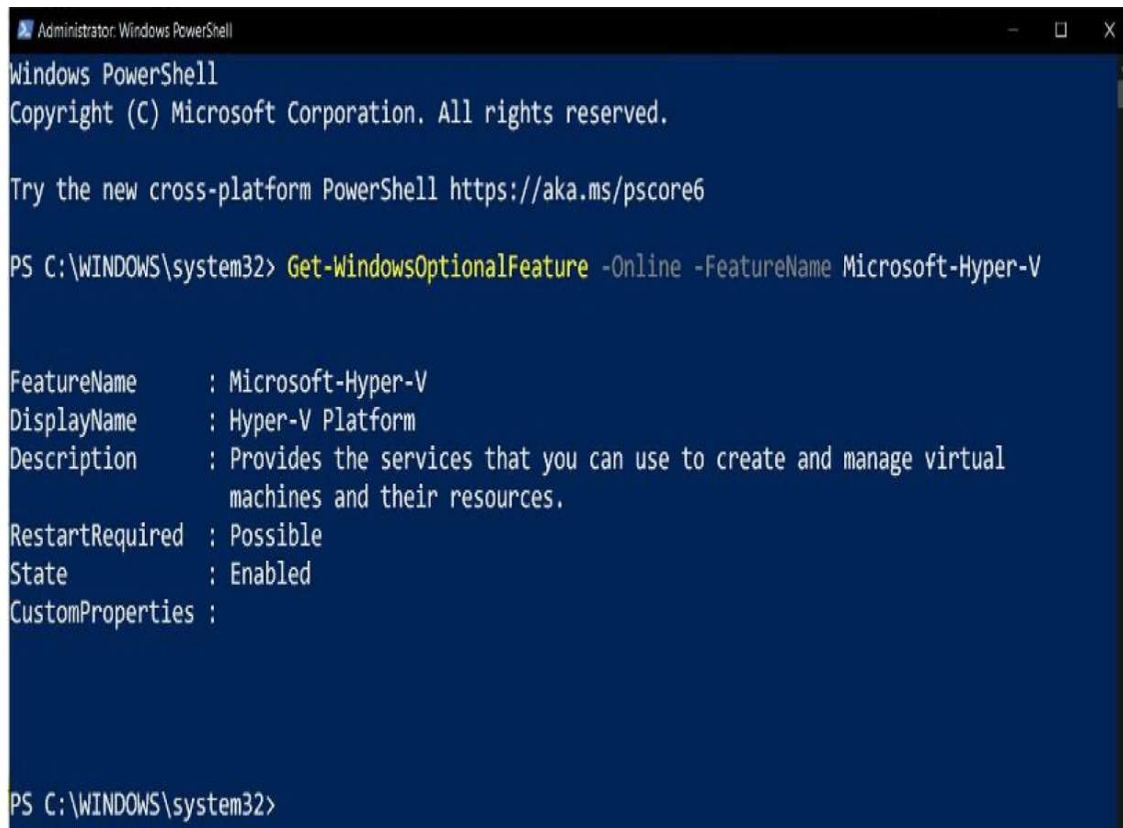


5. Select OK.

Your system will now begin installing Hyper-V in the background; it may be necessary to reboot a few times until everything is properly configured. Don't hold your breath for a notification or anything! Verify that Hyper-V is installed successfully on your machine by running the following command as Administrator in PowerShell:

```
Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V
```

Once the state is shown as Enabled for above command in Power shell, we are good to go.

A screenshot of a Windows PowerShell terminal window. The title bar reads "Administrator: Windows PowerShell". The terminal text includes the standard PowerShell copyright notice, a link to the new cross-platform PowerShell, and the execution of the command `Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V`. The output shows that the feature is named "Microsoft-Hyper-V", displayed as "Hyper-V Platform", provides services for creating and managing virtual machines, requires a restart, and is currently enabled.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V

FeatureName      : Microsoft-Hyper-V
DisplayName       : Hyper-V Platform
Description       : Provides the services that you can use to create and manage virtual
                  machines and their resources.
RestartRequired  : Possible
State             : Enabled
CustomProperties  :

PS C:\WINDOWS\system32>
```

Step 2: Download Docker for Windows and install it.

Kubernetes is a container orchestration system built on top of Docker. It is essentially just a tool for communicating with Docker containers and managing everything at an enterprise level. Simply go to install Docker and [click](#) to Get Docker Desktop for Windows (stable).

Windows users can use Docker Desktop.

Docker Desktop for Windows is a version of Docker optimized for Windows 10. It's a native Windows application that makes developing, shipping, and running dockerized apps simple. Docker Desktop for Windows is the fastest and most reliable way to develop Docker apps on Windows, as it uses Windows-native Hyper-V virtualization and networking. Docker Desktop for Windows can run Docker containers on both Linux and Windows.

Installation of Docker Desktop

Let us take a look on the different steps involved in installing docker desktop.

1. Double-click Docker for Windows Installer to run the installer.
2. Docker starts automatically once the installation is complete. Docker is running and accessible from a terminal, as indicated by the whale in the notification area.

3. Run .
4. Try out some Docker commands in a command-line terminal like PowerShell!
5. Run the Docker version to check the version.
6. Run Docker run hello-world to verify that Docker can pull and run images.
7. Boom!

As long as the Docker Desktop for Windows app is running, Docker is accessible from any terminal. The Docker whale in the taskbar has a setting button that can be accessed from the UI.

For a detailed step by step installation guide with screenshot, visit the blog - [How to Install Docker on Windows, Mac, & Linux: A Step-By-Step Guide](#)

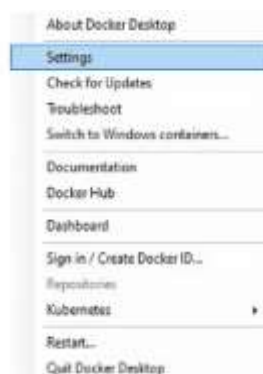
WARNING: FOLLOW THE INSTRUCTIONS BELOW! If Docker was successfully installed but you can't find its tray icon, you'll need to restart your computer. Check the official troubleshooting [guide here](#) if the issue persists.

Step 3: Install Kubernetes on Windows 10

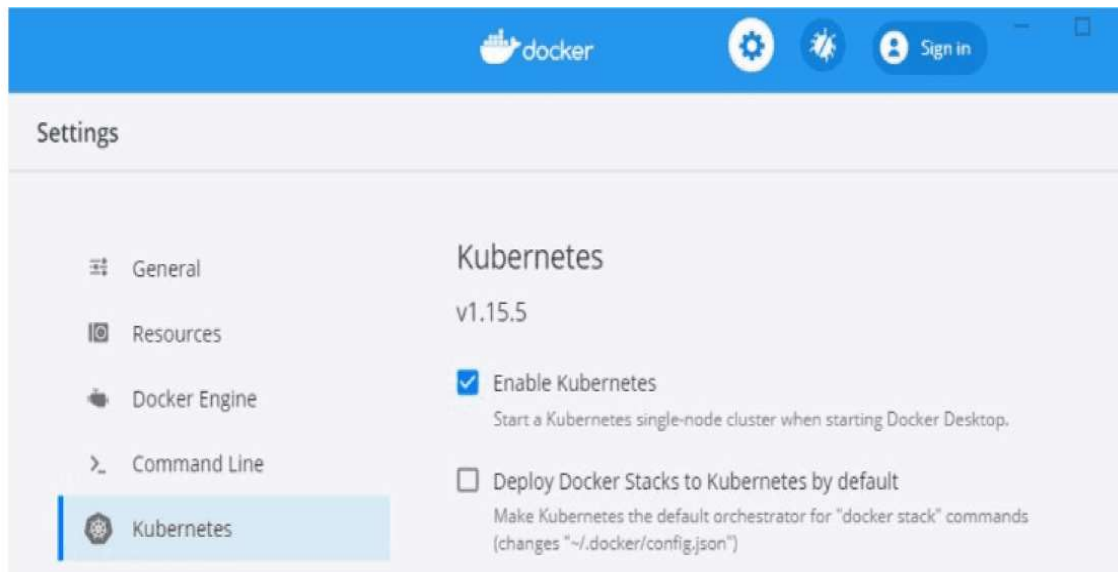
Docker includes a graphical user interface (GUI) tool that allows you to change some settings or install and enable Kubernetes.

To install Kubernetes, simply follow the on-screen instructions on the screen:

1. Right-click the Docker tray icon and select Properties.
2. Select "Settings" from the drop-down menu.

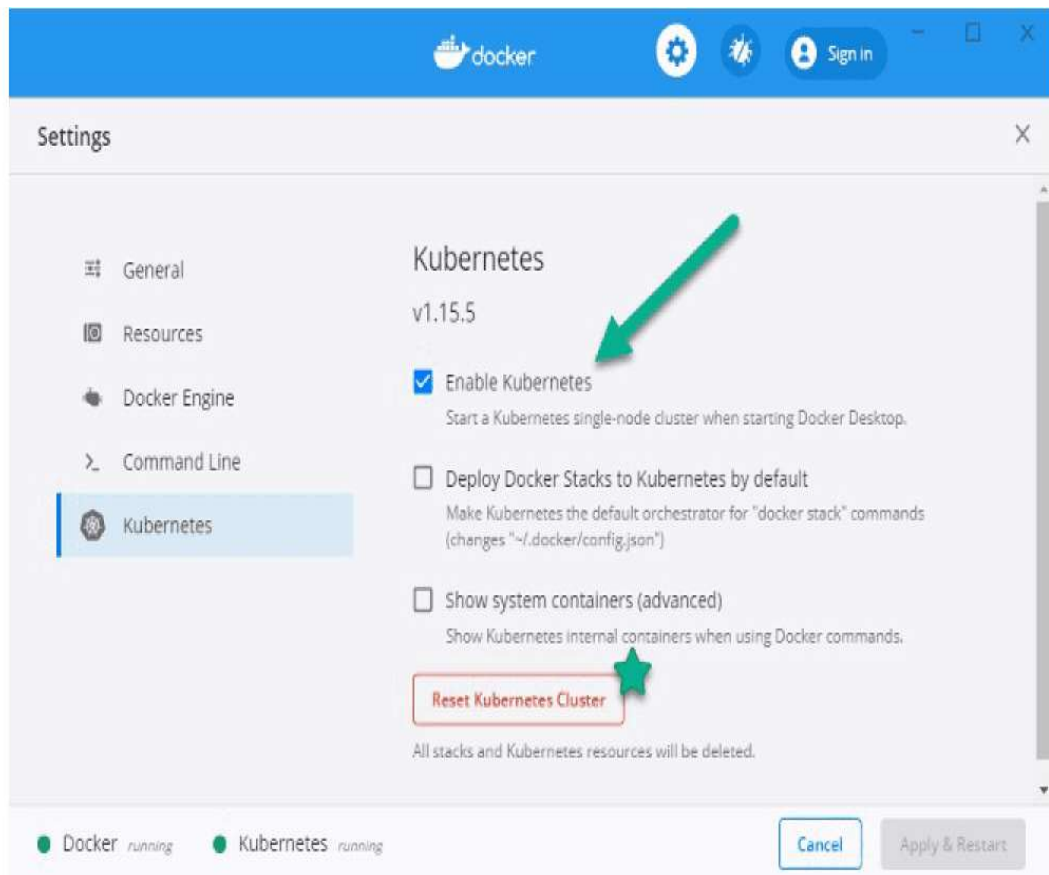


3. Select "Kubernetes" from the left panel.
4. Check Enable Kubernetes and click "Apply"



Docker will install additional packages and dependencies during the installation process. It may take between 5 and 10 minutes to install, depending on your Internet speed and PC performance. Wait until the message 'Installation complete!' appears on the screen. The Docker app can be used after Kubernetes has been installed to ensure that everything is working properly. Both icons at the bottom left will turn green if both services (Docker and Kubernetes) are running successfully and without errors.

Example.



Step 4: Install Kubernetes Dashboard

The official web-based UI for [managing Kubernetes resources](#) is [Kubernetes Dashboard](#). It isn't set up by default. Kubernetes applications can be easily deployed using the cli tool `kubectl`, which allows you to interact with your cloud and manage your [Pods](#), Nodes, and Clusters. You can easily create or update [Kubernetes resources](#) by passing the `apply` argument followed by your YAML configuration file.

Use the following commands to deploy and enable the Kubernetes Dashboard.

1. Get the yaml configuration file from [here](#).
2. Use this to deploy it.

```
. kubectl apply -f .\recommended.yaml
```

Copy Code

3. Run the following command to see if it's up and running.:

```
kubectl.exe get -f .\recommended.yaml.txt
```

Copy Code

```
Administrator: Windows PowerShell

PS C:\WINDOWS\system32> kubectl.exe get -f .\recommended.yaml.txt

NAME                                STATUS    AGE
namespace/kubernetes-dashboard    Active    2m10s

NAME                                SECRETS    AGE
serviceaccount/kubernetes-dashboard    1          2m10s

NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP
PORT(S)    AGE
service/kubernetes-dashboard    ClusterIP    10.97.3.127    <none>
443/TCP    2m9s
```

Step 5: Access the dashboard

The dashboard can be accessed with tokens in two ways: the first is by using the default token created during Kubernetes installation, and the second (more secure) method is by creating users, giving them permissions, and then receiving the generated token. We'll go with the first option for the sake of simplicity.

1. Run the following command PowerShell (not cmd)

```
((kubectl -n kube-system describe secret default | Select-String "token:") -split " ")[1]
```

Copy Code

2. Copy the generated token
3. Run

```
kubectl proxy.
```

Copy Code

4. Open the following link on your browser:


```
http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/
```

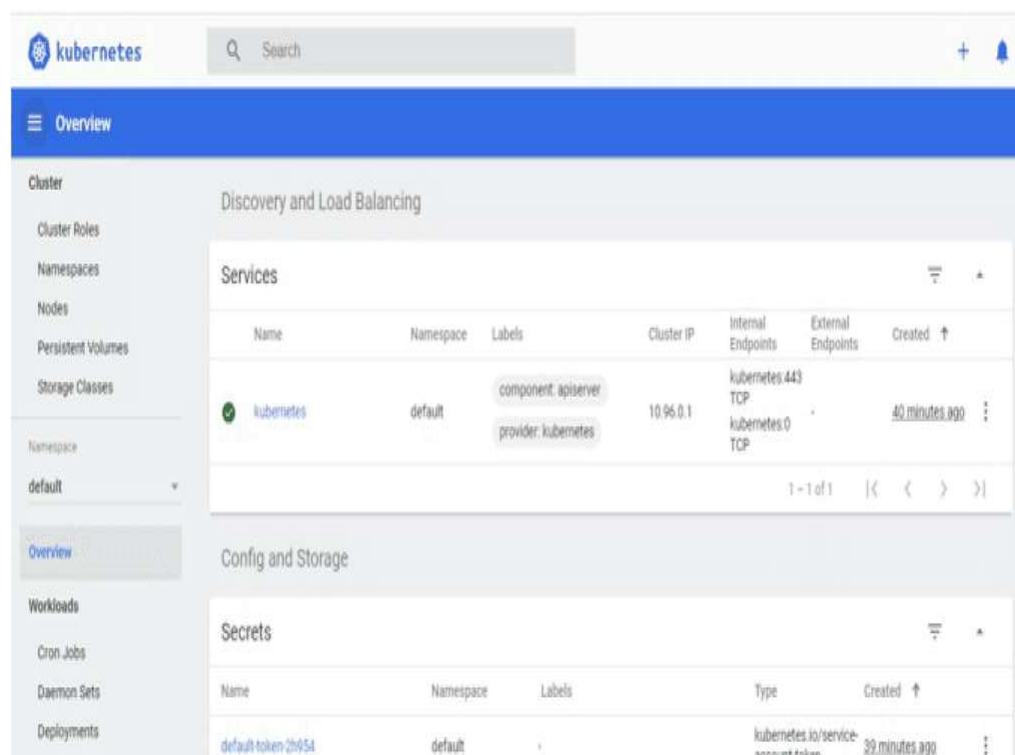
Copy Code

5. Select

Token & paste the generated token

6. Sign In

Finally



You'll be able to see the dashboard and your cloud resources if everything is set up correctly. You can then do almost all of the "hard" work without having to deal with the CLI every time. You may occasionally get your hands dirty with the command line, but if you don't understand Docker and Kubernetes or don't have the time to manage your own cloud, it's better to stick with some PaaS providers that can be quite expensive.

Kubernetes Uninstallation Process

The procedures for uninstalling cert-manager on Kubernetes are outlined below.

Depending on which method you used to install cert-manager - static manifests or helm - you have two options.

Warning: To uninstall cert-manager, follow the same steps as you did to install it, but in reverse. Whether cert-manager was installed from static manifests or helm, deviating from

the following process can result in issues and potentially broken states. To avoid this, make sure you follow the steps outlined below when uninstalling.

Step 1: Before continuing, make sure that all user-created cert-manager resources have been deleted. You can check for any existing resources with the following command:

```
$ kubectl get Issuers,ClusterIssuers,Certificates,CertificateRequests,Orders,Challenges --all-namespaces
```

Copy Code

After you've deleted all of these resources, you can uninstall cert-manager by following the steps outlined in the installation guide.

Step 2: Using regular manifests to uninstall.

1. Uninstalling from a regular manifest installation is as simple as reversing the installation process and using the delete command.

```
kubectl.
```

Copy Code

2. Delete the installation manifests using a link to your currently running version vX.Y.Z like so:

```
$ kubectl delete -f https://github.com/jetstack/cert-manager/releases/download/vX.Y.Z/cert-manager.yaml
```

Copy Code

Step 3: Uninstalling with Helm.

1. Uninstalling cert-manager from a Helm installation is as simple as reversing the installation process and using the delete command on both the server and the client. kubectl and helm.

```
$ helm --namespace cert-manager delete cert-manager
```

Copy Code

2. Next, delete the cert-manager namespace:

```
$ kubectl delete namespace cert-manager
```

Copy Code

3. Finally, delete the cert-manger CustomResourceDefinitions using the link to the version vX.Y.Z you installed:

```
$ kubectl delete -f https://github.com/jetstack/cert-manager/releases/download/vX.Y.Z/cert-manager.crds.yaml
```

Copy Code

The namespace is in the process of being terminated.

The namespace may become stuck in a terminating state if it is marked for deletion without first deleting the cert-manager installation. This is usually because the APIService resource is still present, but the webhook is no longer active and thus no longer reachable.

4. To fix this, make sure you ran the above commands correctly, and if you're still having problems, run:

```
$ kubectl delete apiservice v1beta1.
```