<https://itnext.io/kubernetes-rancher-2-5-on-your-windows-10-laptop-with-k3d-and-k3s-7404f288342f>

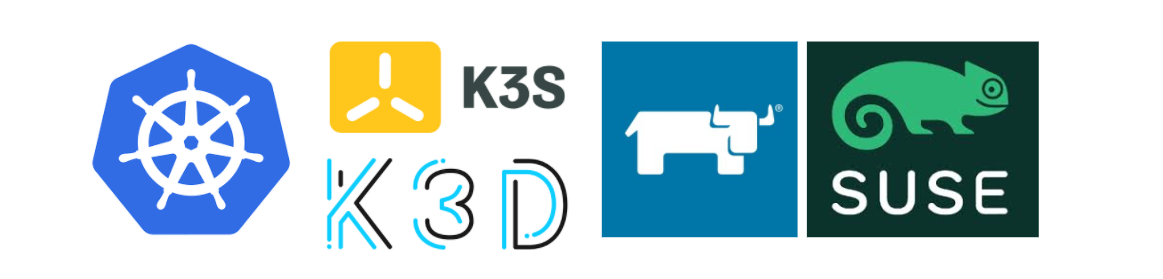
# Rancher 2.5 on your Windows 10 laptop with k3d and k3s and deploy nginx on k8s

[[](https://jyeee.medium.com/?source=post_page-----7404f288342f--------------------------------)](https://jyeee.medium.com/?source=post_page-----7404f288342f--------------------------------)

[Jason Yee](https://jyeee.medium.com/?source=post_page-----7404f288342f--------------------------------)

Follow

[Nov 15](https://itnext.io/kubernetes-rancher-2-5-on-your-windows-10-laptop-with-k3d-and-k3s-7404f288342f?source=post_page-----7404f288342f--------------------------------) · 7 min read



This article shows how to set up a minimal Kubernetes dev env with Rancher Cluster Manager on your Windows 10 laptop using k3d to set up a k3s cluster.

User Admin

Pw reality

<https://k3d.io/usage/guides/exposing_services/>

https://gist.github.com/rafi/d4440661e7de208009701ca3627caa1d

**tl;dr install Docker Desktop & Chocolatey and run the gist at the end**

*This is an update to*[*https://jyeee.medium.com/rancher-2-4-14c31af12b7a*](https://jyeee.medium.com/rancher-2-4-14c31af12b7a)*but using Docker and k3d instead of multipass*

# Prerequisites

* Windows 10 **Pro** with Hyper-V
* Docker Desktop (2.5.x)

Seriously, that’s it! We’ll use chocolatey to help, but otherwise don’t need anything else.

# Overview

1. Enable Hyper-V and check your Docker install
2. Install Chocolatey to install kubectl and helm
3. Install k3d and spin up a single-node k3s Kubernetes cluster
4. Use helm to deploy cert-manager then Rancher to the cluster
5. Deploy apps via Rancher

# 1/5 Enable Hyper-V and check your Docker install

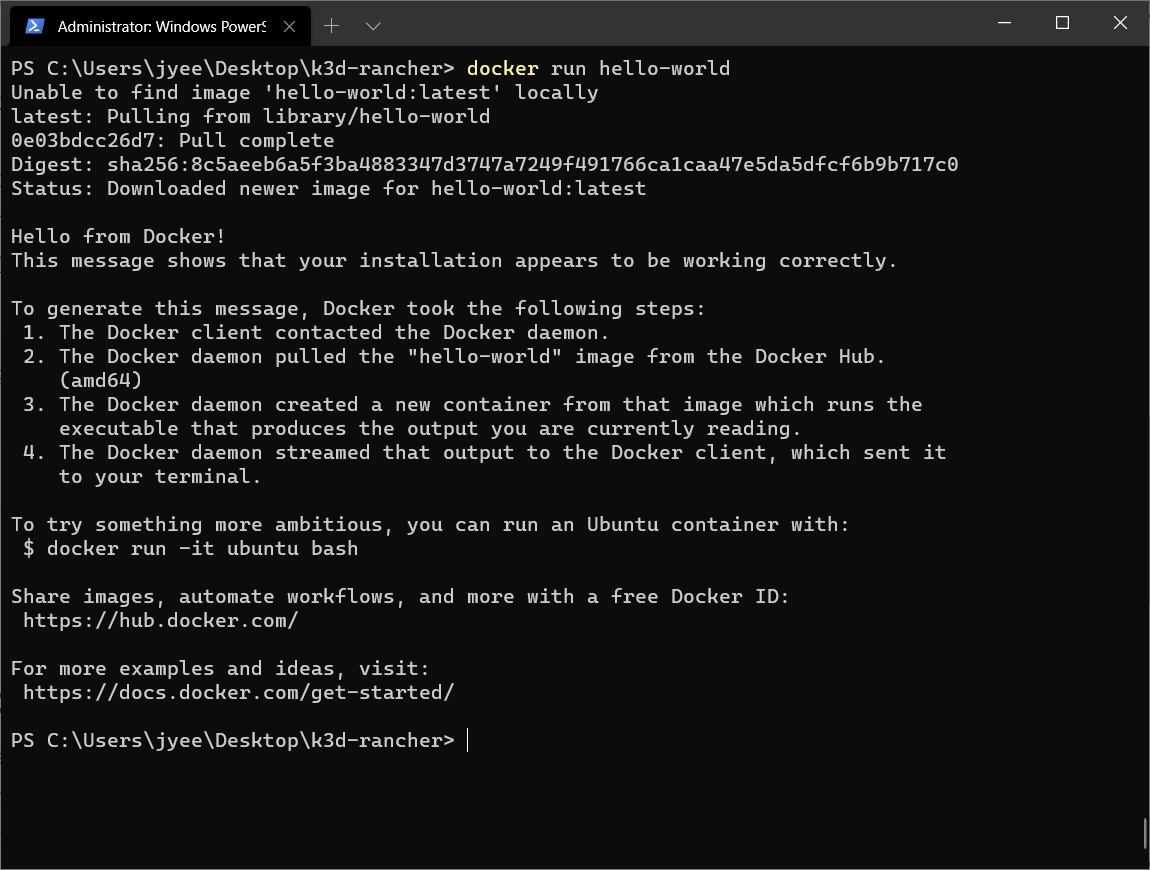
Hyper-V is allows you to quickly run a VM that runs the single-node kubernetes cluster. It comes standard with Windows 10 Pro.

If you haven’t done work with Hyper-V before, you may have to enable it with this command (read more at <https://github.com/kubernetes/minikube/issues/2954>)

**Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All**

This is also a good time to check that your docker install is good

**docker run hello-world**



Hello world!

*Helpful tip: Pin Windows Terminal as the nth item on the task bar. Then hit Win+Ctrl+Shift+n to open it as admin. So if you make it the first one it will be Win+Ctrl+Shift+1*

[*https://stackoverflow.com/questions/62496779/opening-up-windows-terminal-with-elevated-privileges-from-within-windows-termin*](https://stackoverflow.com/questions/62496779/opening-up-windows-terminal-with-elevated-privileges-from-within-windows-termin)

# 2/5 Install Chocolatey to install helm & kubectl

Chocolatey is a command-line package manager for Windows, and it feels like the Homebrew tools in macOS. We’re installing it here because Chocolatey provides easy installation of the Helm & kubectl (Kubernetes CLI) utilities.

Note: I created a working folder k3s-rancher on my desktop to hold the future kubeconfig file.

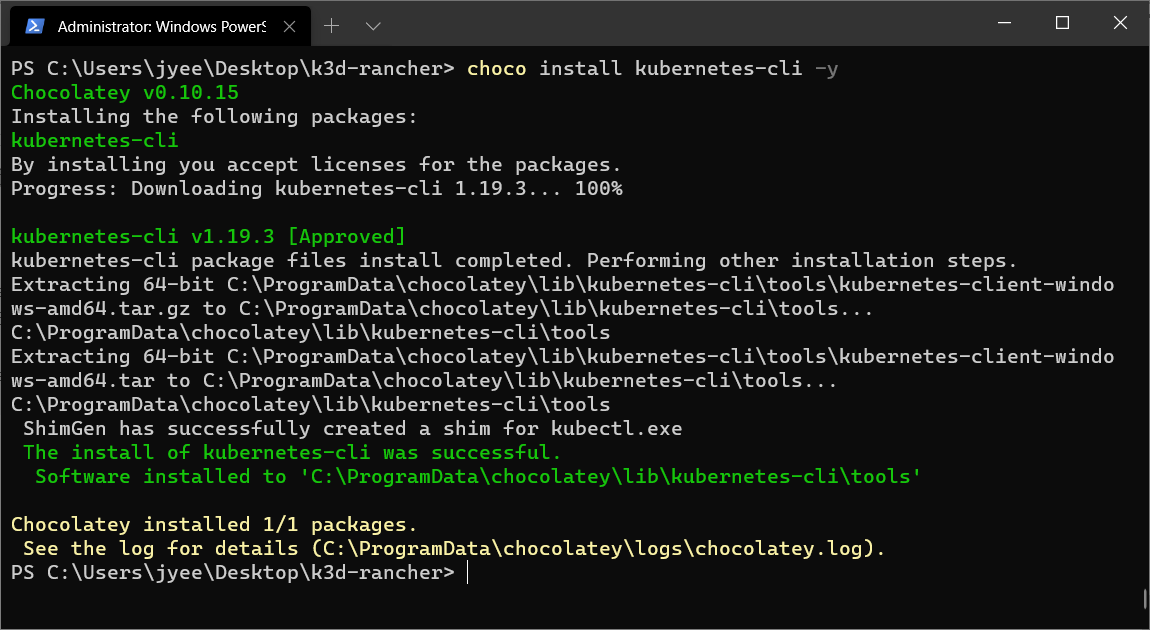
I followed the instructions here <https://chocolatey.org/install#individual>, and use this command to confirm it’s installed.

PS C:\Users\jyee\Desktop\k3s-rancher>

**choco list --local-only**  
Chocolatey v0.10.15  
chocolatey 0.10.15  
1 packages installed.

Then install helm and kubectl with the commands

**choco install kubernetes-cli -y  
choco install kubernetes-helm -y**



The below shows all the Chocolatey packages we should have installed.

PS C:\Users\jyee\Desktop\k3d-rancher>

**choco list --local-only**  
Chocolatey v0.10.15  
chocolatey 0.10.15  
kubernetes-cli 1.19.3  
kubernetes-helm 3.4.0  
3 packages installed.

# 3/5 Install k3d and spin up a single-node k3s Kubernetes cluster

[k3d](https://k3d.io/) is a way to run a [k3s](https://k3s.io/) cluster with only Docker Desktop as a dependency. k3s is a tiny distro of k8s that can run on a Raspberry Pi — but it’s as [certified as any k8s distro and you can run it in production](https://thenewstack.io/how-rancher-labs-k3s-makes-it-easy-to-run-kubernetes-at-the-edge/). Get it by downloading it from GitHub, and to make the instructions work for multiple platforms (hello macOS) I’ve set an alias to k3d for convenience.

PS C:\Users\jyee\Desktop\k3d-rancher>

**wget** [**https://github.com/rancher/k3d/releases/download/v3.2.0/k3d-windows-amd64.exe**](https://github.com/rancher/k3d/releases/download/v3.2.0/k3d-windows-amd64.exe) **-o k3d-windows-amd64-3.2.0.exe**

PS C:\Users\jyee\Desktop\k3d-rancher>

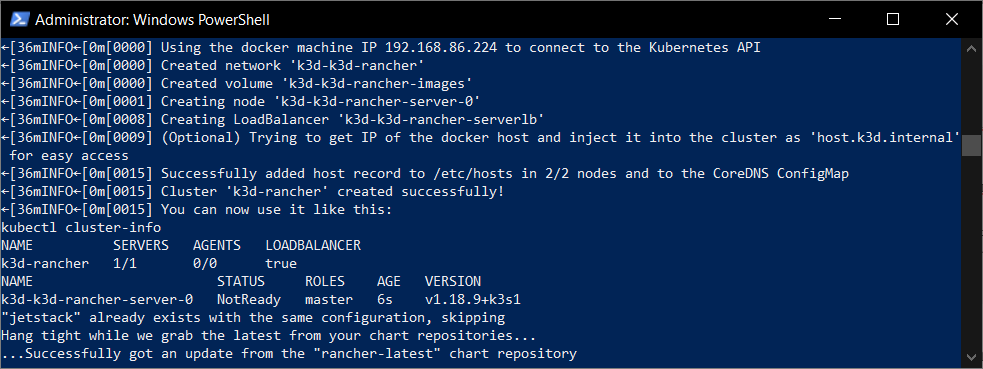
lsDirectory:

C:\Users\jyee\Desktop\k3d-rancherMode LastWriteTime Length Name  
---- ------------- ------ ----  
-a---- 11/11/2020 9:34 PM 22679552 k3d-windows-amd64-3.2.0.exePS

C:\Users\jyee\Desktop\k3d-rancher>

**Set-Alias -Name k3d -Value ~\Desktop\k3d-rancher\k3d-windows-amd64-3.2.0.exe**

Now build the cluster on [https://rancher.localhost](https://rancher.localhost./) where we’ll use the .localhost [DNS magic](https://en.wikipedia.org/wiki/.localhost) to know route any request to the k3s node.



PS C:\Users\jyee\Desktop\k3d-rancher>

**$env:CLUSTER\_NAME="k3d-rancher"**

**Set CLUSTER\_NAME="k3d-rancher"**

PS C:\Users\jyee\Desktop\k3d-rancher>

**$env:RANCHER\_SERVER\_HOSTNAME="rancher.localhost"**

**Set RANCHER\_SERVER\_HOSTNAME="rancher.localhost"**

PS C:\Users\jyee\Desktop\k3d-rancher>

**$env:KUBECONFIG\_FILE="${env:CLUSTER\_NAME}.yaml"**

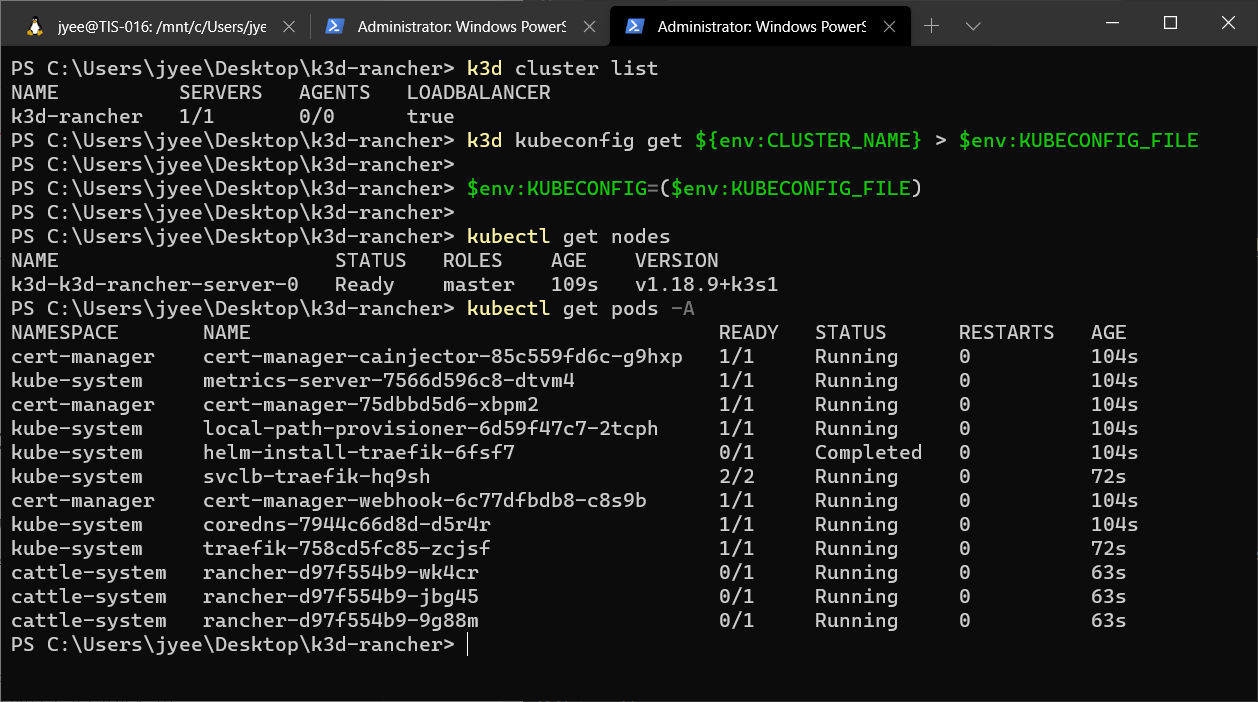
**Set KUBECONFIG\_FILE="${env:CLUSTER\_NAME}.yaml"**

PS C:\Users\jyee\Desktop\k3d-rancher>

**k3d cluster create $env:CLUSTER\_NAME --api-port 6550 --servers 1 --port 443:443 @loadbalancer –wait**

**k3d cluster create k3d-rancher –-api-port 6550 --servers 1 -p** "**443:443@loadbalancer**" **–-wait**

Once you’ve got a cluster, set some env variables, aliases, and your kubectl so you can confirm your cluster is up and running and start running apps on it.



PS C:\Users\jyee\Desktop\k3d-rancher>

**k3d cluster list**  
NAME SERVERS AGENTS LOADBALANCER  
k3d-rancher 1/1 0/0 truePS C:\Users\jyee\Desktop\k3d-rancher>

**k3d kubeconfig get ${env:CLUSTER\_NAME} > $env:KUBECONFIG\_FILE**

PS C:\Users\jyee\Desktop\k3d-rancher> **$env:KUBECONFIG=($env:KUBECONFIG\_FILE)**

PS C:\Users\jyee\Desktop\k3d-rancher>

**kubectl get nodes**  
NAME STATUS ROLES AGE VERSION  
k3d-k3d-rancher-server-0 Ready master 109s v1.18.9+k3s1PS C:\Users\jyee\Desktop\k3d-rancher>

**kubectl get pods -A**  
NAMESPACE NAME READY STATUS RESTARTS AGE  
cert-manager cert-manager-cainjector-85c559fd6c-g9hxp 1/1 Running 0 104s  
...  
cattle-system rancher-d97f554b9-9g88m 0/1 Running 0 63s

# 4/5 Use helm to deploy cert-manager then Rancher to the cluster

These commands install Rancher and generally follow these instructions <https://rancher.com/docs/rancher/v2.x/en/installation/k8s-install/helm-rancher/>

# Install cert-manager with helm  
**helm repo add jetstack** [**https://charts.jetstack.io**](https://charts.jetstack.io/) **helm repo update  
kubectl create namespace cert-manager  
helm install cert-manager jetstack/cert-manager --namespace cert-manager --version v1.0.4 --set installCRDs=true --wait  
kubectl -n cert-manager rollout status deploy/cert-manager  
date**

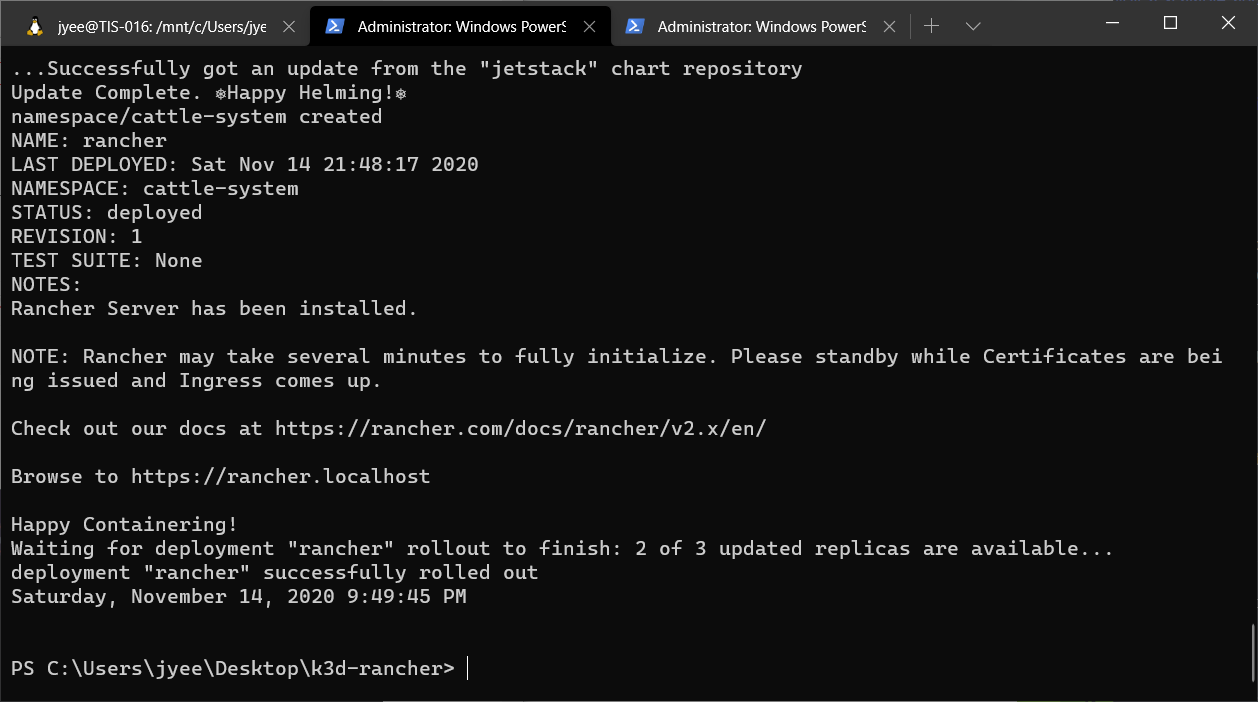
# Install Rancher  
**helm repo add rancher-latest** [**https://releases.rancher.com/server-charts/latest**](https://releases.rancher.com/server-charts/latest) **helm repo update  
kubectl create namespace cattle-system**

**helm install rancher rancher-latest/rancher --namespace cattle-system --set hostname=${env:RANCHER\_SERVER\_HOSTNAME} --wait**

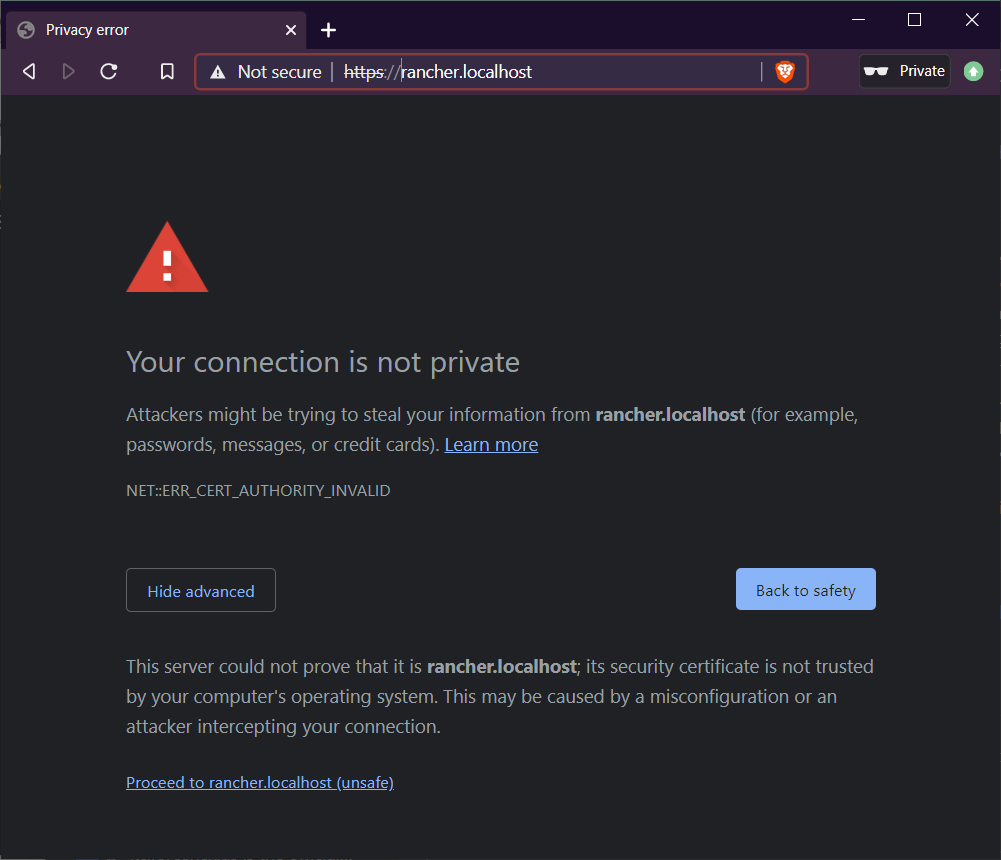
**helm install rancher rancher-latest/rancher --namespace cattle-system --set hostname=rancher.localhost --wait**

**kubectl -n cattle-system rollout status deploy/rancher  
date**

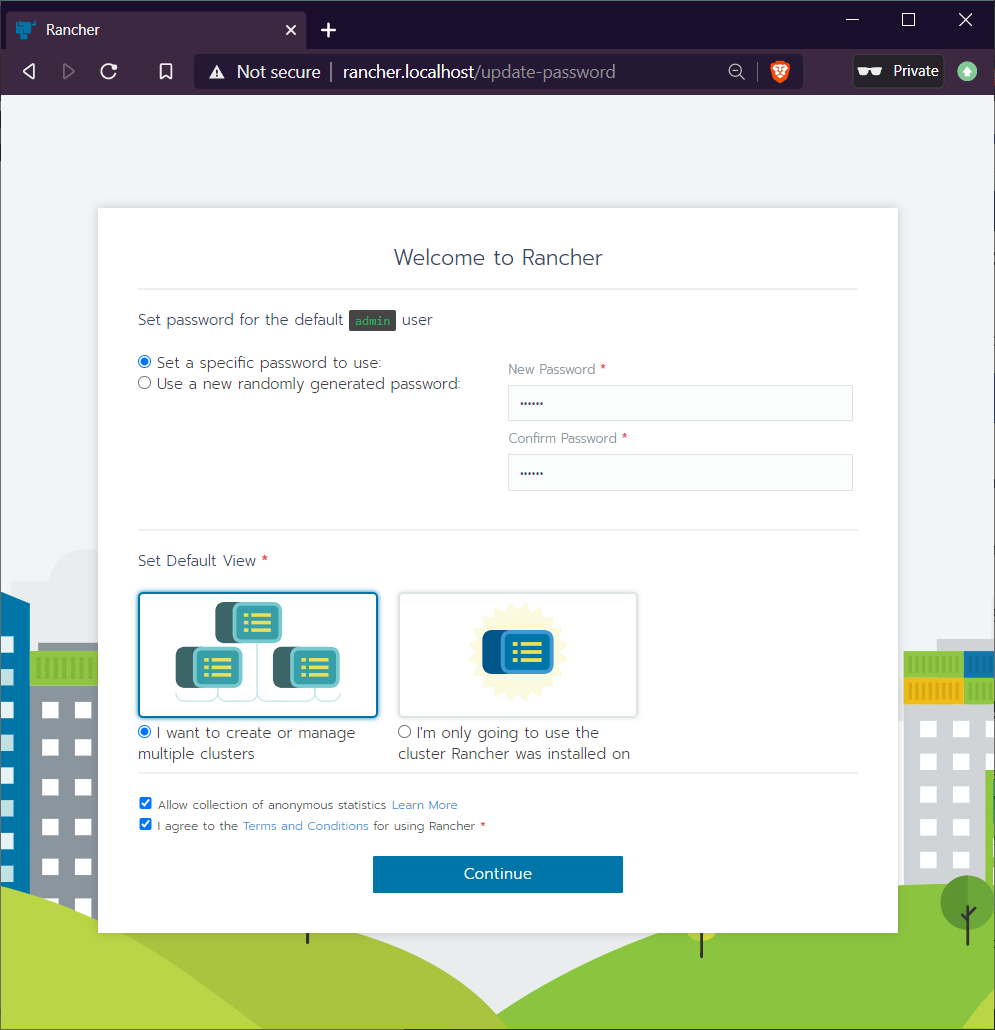
After a successful install, you’ll see these logs



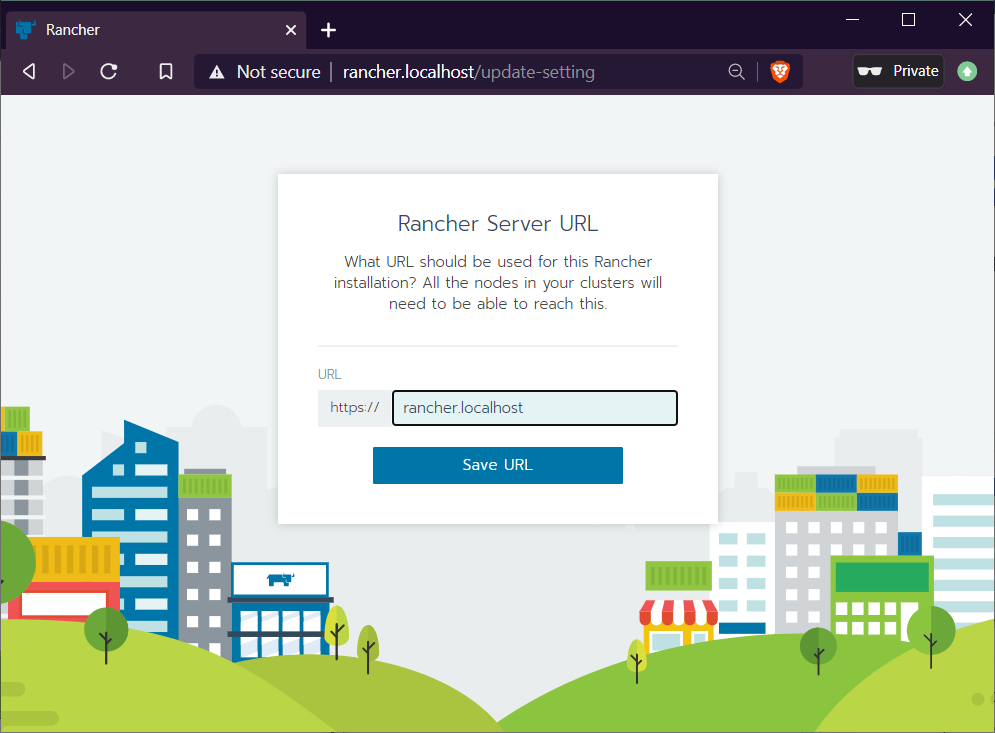
After you see the line deployment "rancher" successfully rolled out browse to [https://rancher.localhost](https://rancher.localhost./) where we’ll use the .localhost [DNS magic](https://en.wikipedia.org/wiki/.localhost) to know route any request to the k3s node. If you don’t see this screen, you may have to use the [thisisunsafe](https://dev.to/brettimus/this-is-unsafe-and-a-bad-idea-5ej4)chrome trick

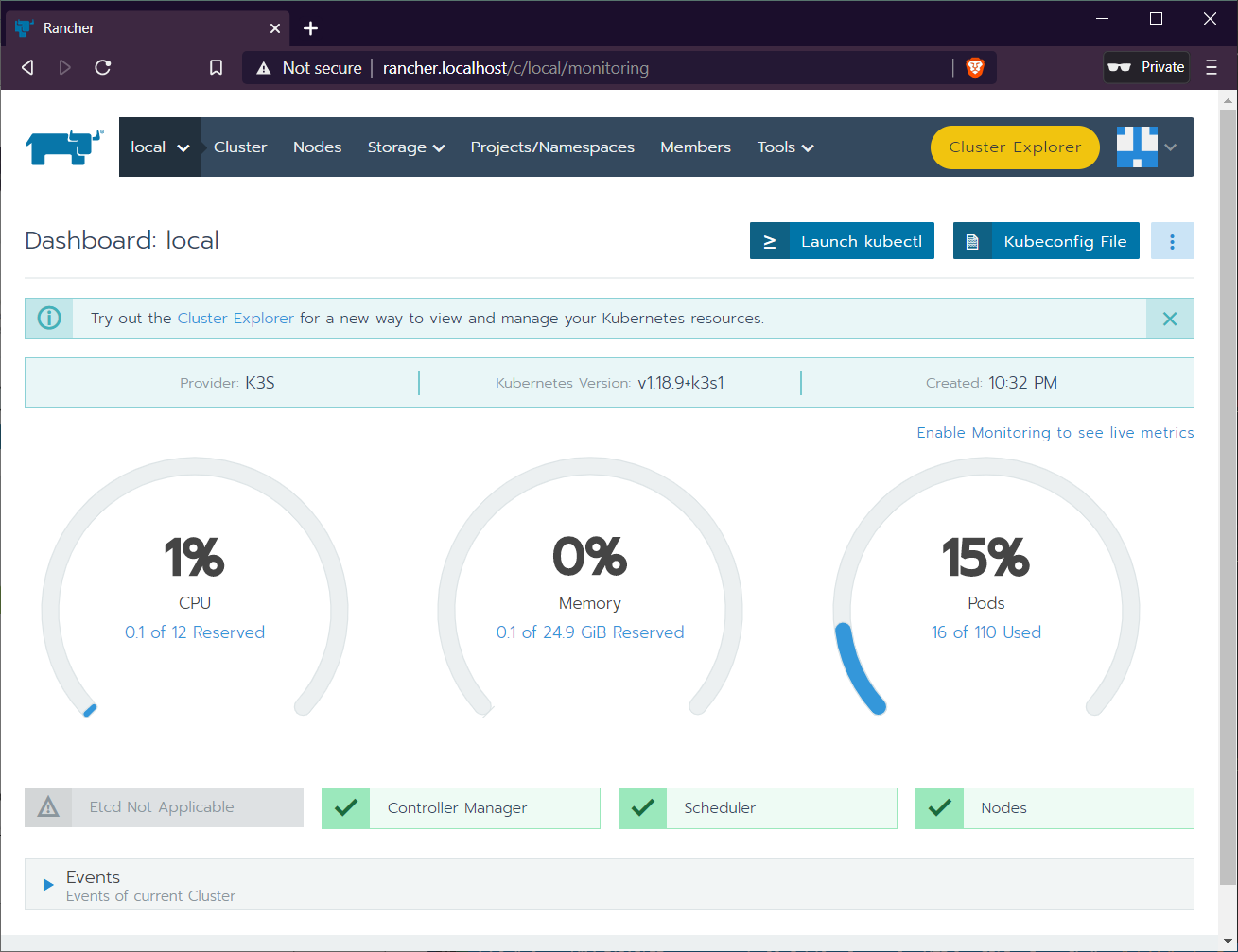


I left the defaults and created a new password for the admin user. If I have to log in again, I can use admin and this password

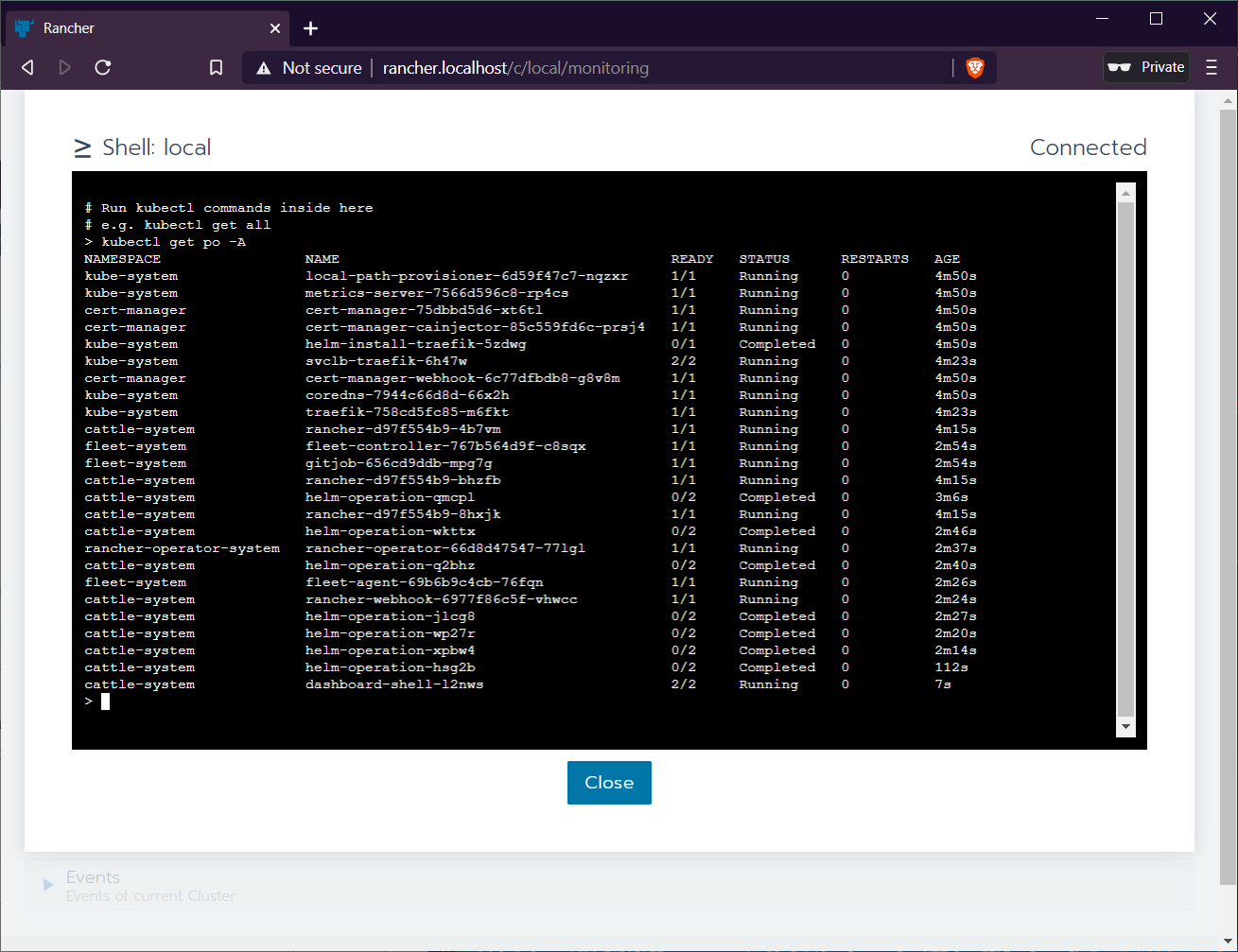


Use rancher.localhost as your rancher install to utilize the .localhost magic. In production, this magic would be a load balancer and wildcard cert.





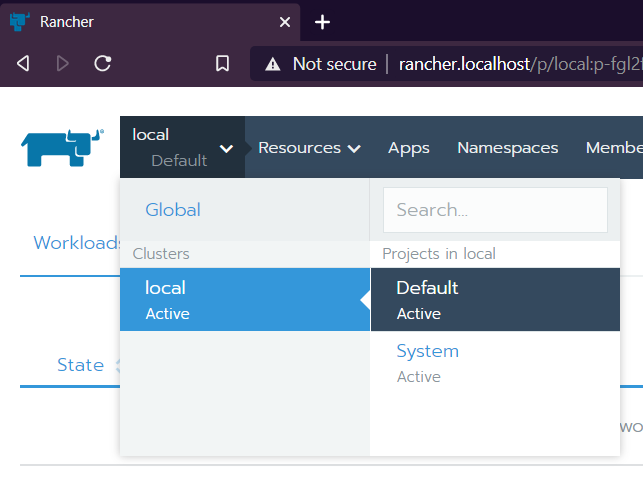
One of the nicest things about Rancher is that you can launch and run kubectl in your browser by clicking the **Launch kubectl** button!



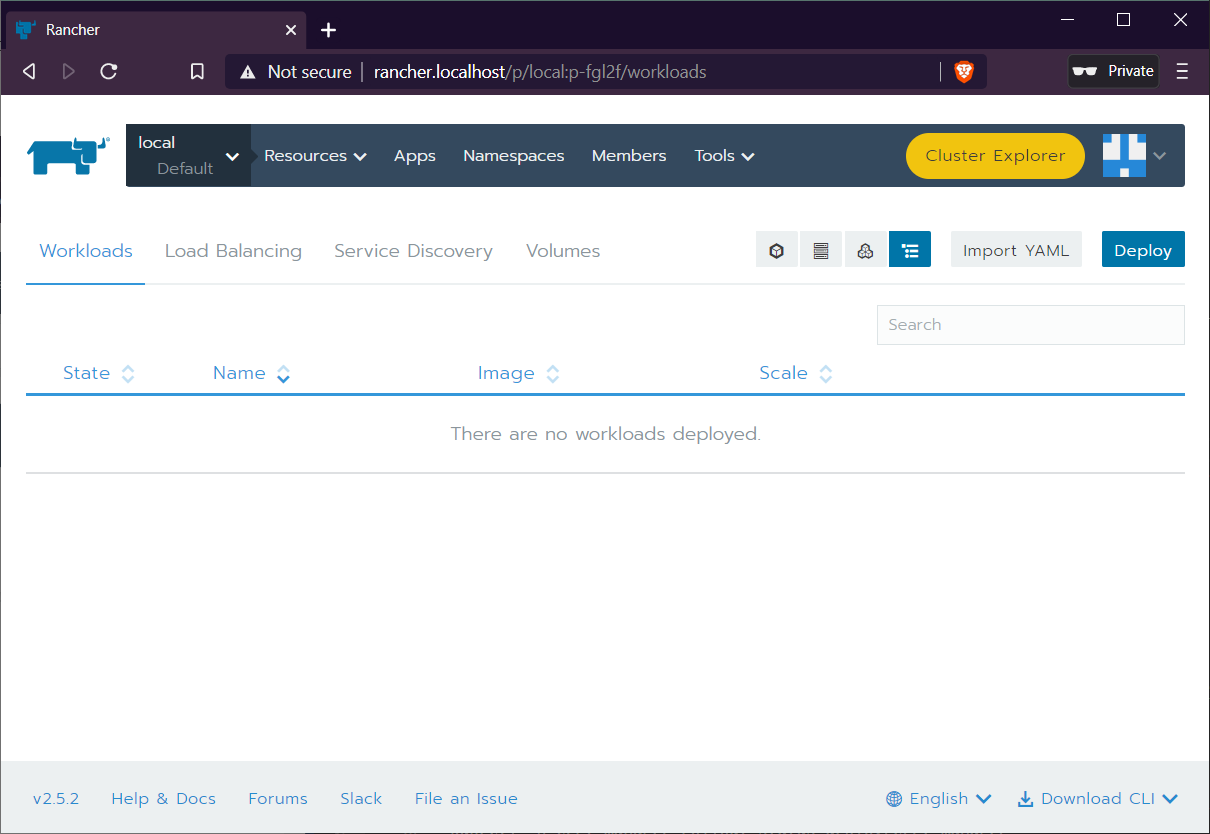
# 5/5 Deploy apps via Rancher

We’ll use the [same procedure shown here](https://jyeee.medium.com/rancher-2-4-14c31af12b7a) to deploy an nginx container and expose it by using the Rancher Cluster Manager UI.

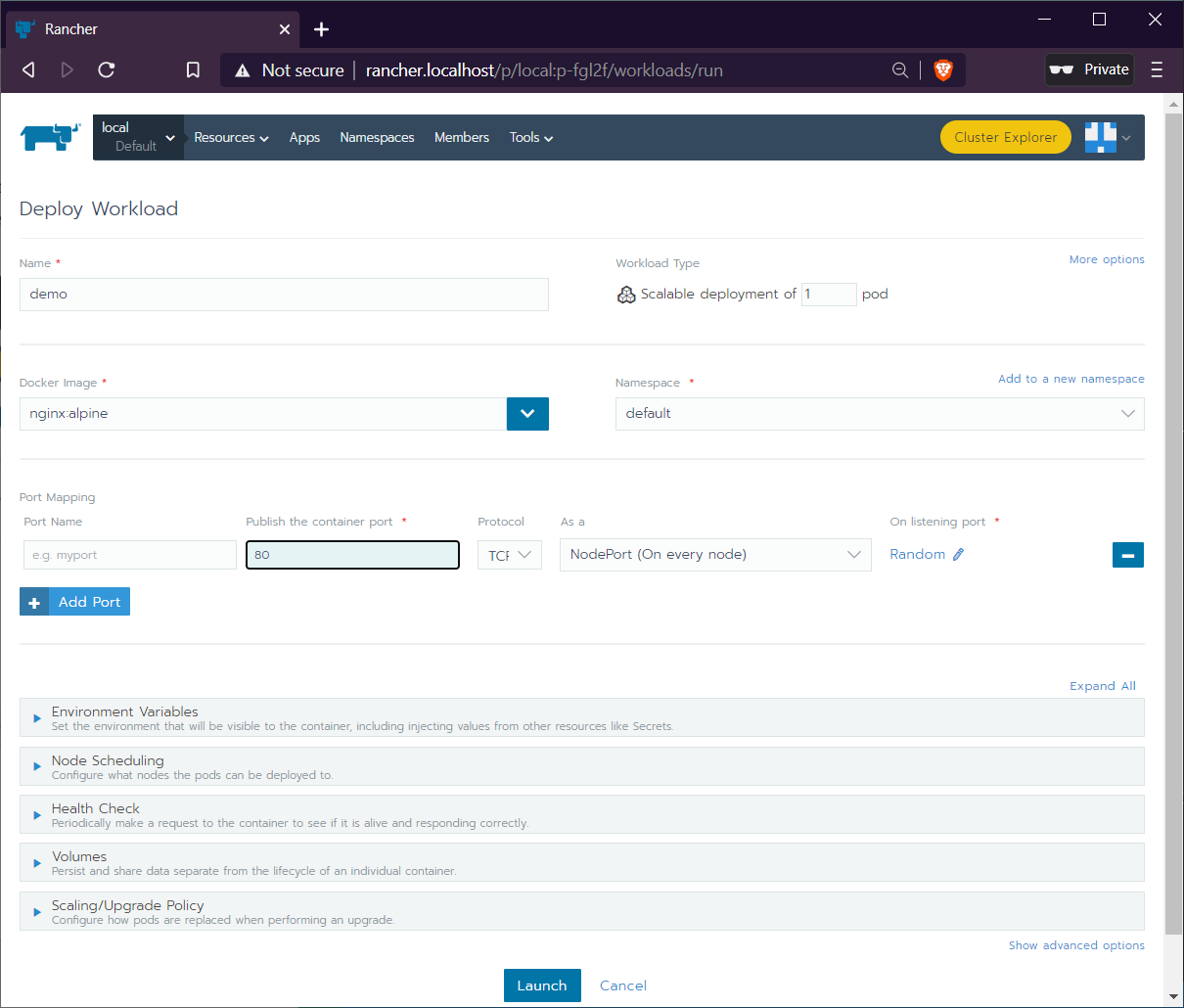
Mouse over the local cluster in the top left and go to the **Default**Rancher Project. Rancher Projects provide a layer of management around namespaces and allow admins of a cluster to easily implement RBAC and resource limits.



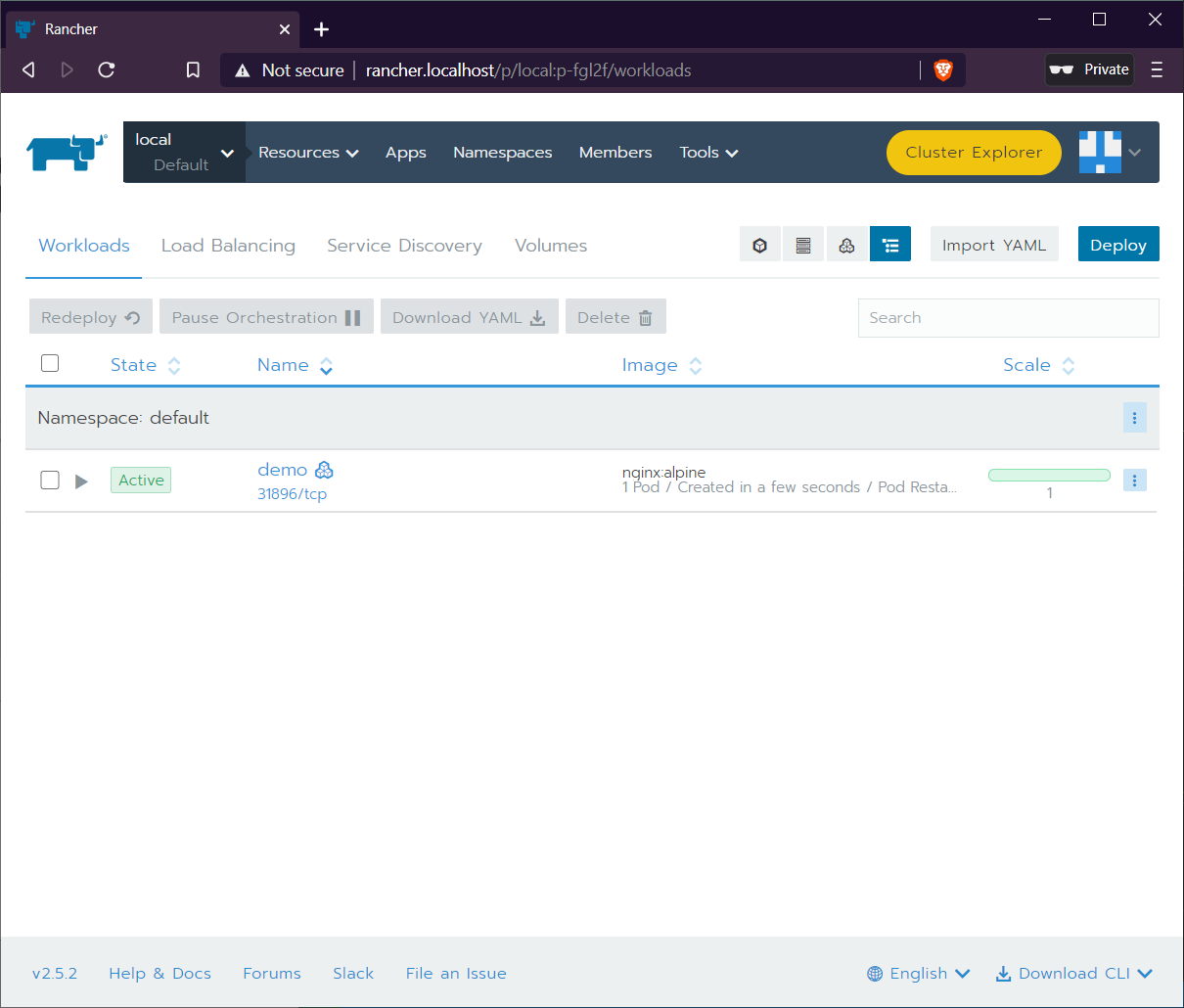
In the **Default**Project page, you can see that no workloads are deployed. Let’s deploy a workload by clicking the blue **Deploy**button in the upper right



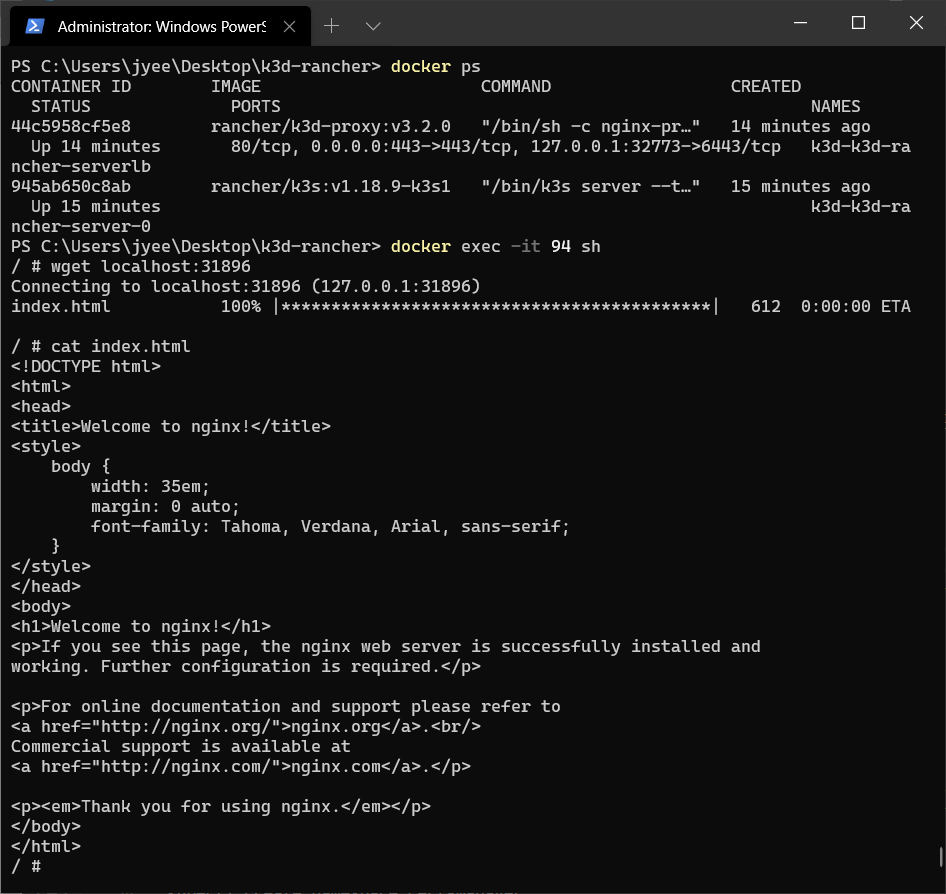
Use the nginx:alpine image and expose port 80 as per this screenshot and click **Launch**



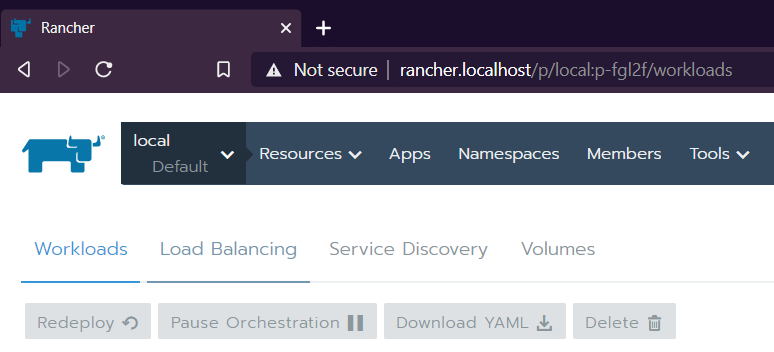
You should see that a “pod” was created and has a “nodeport” that exposes this workload on the nodes in the cluster on port 31896 (imo there is a LOT of terminology/complexity with k8s and Rancher helps you visualize a lot of that).



In other words, if you ssh into one of the nodes, you could access this on port 31896 as shown below with docker exec

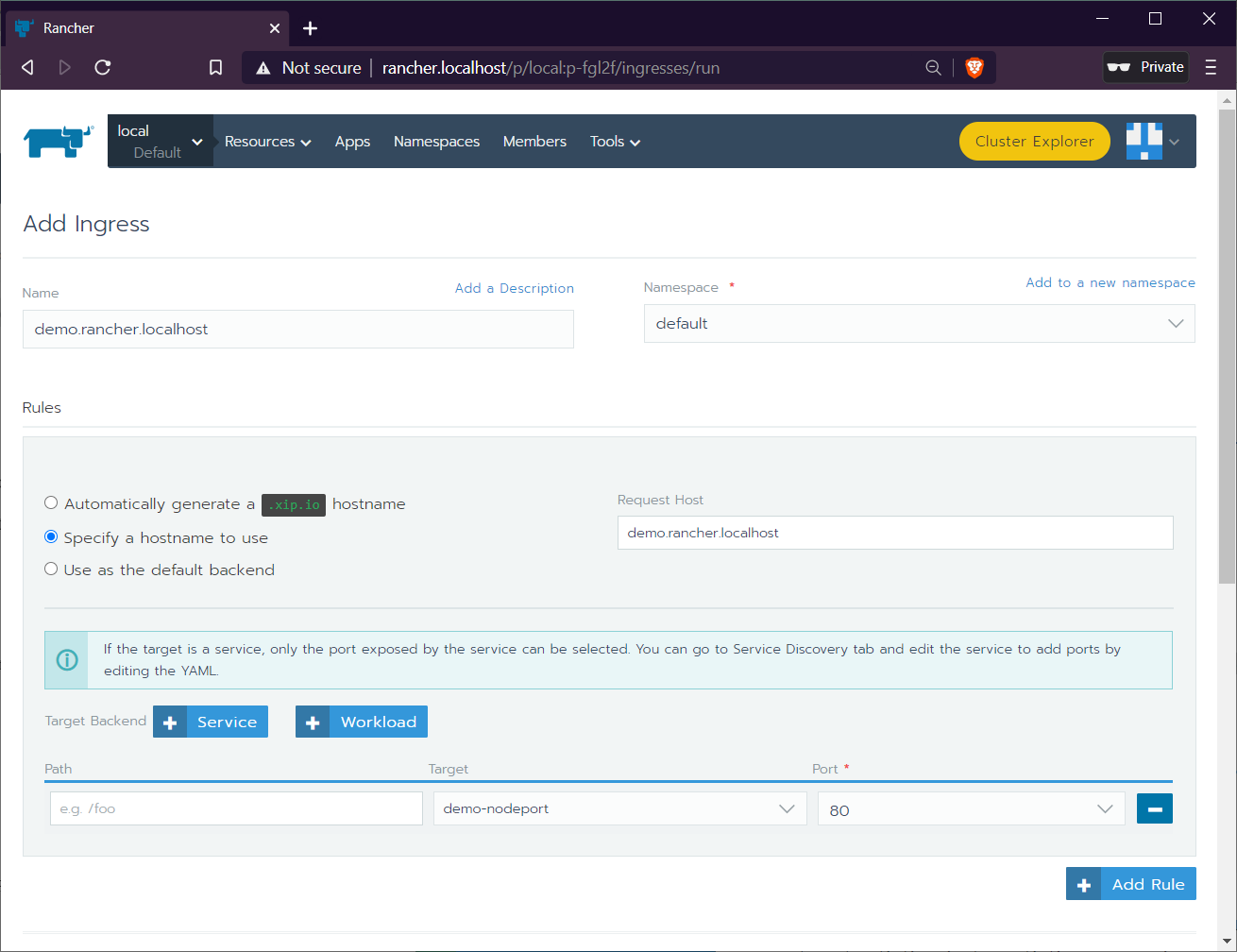


To share this service OUTSIDE of the cluster without exposing any of the nodes, use the very convenient Rancher GUI again to set up an ingress. Click on the **Load Balancing** tab to the right of Workloads and click **Add Ingress**

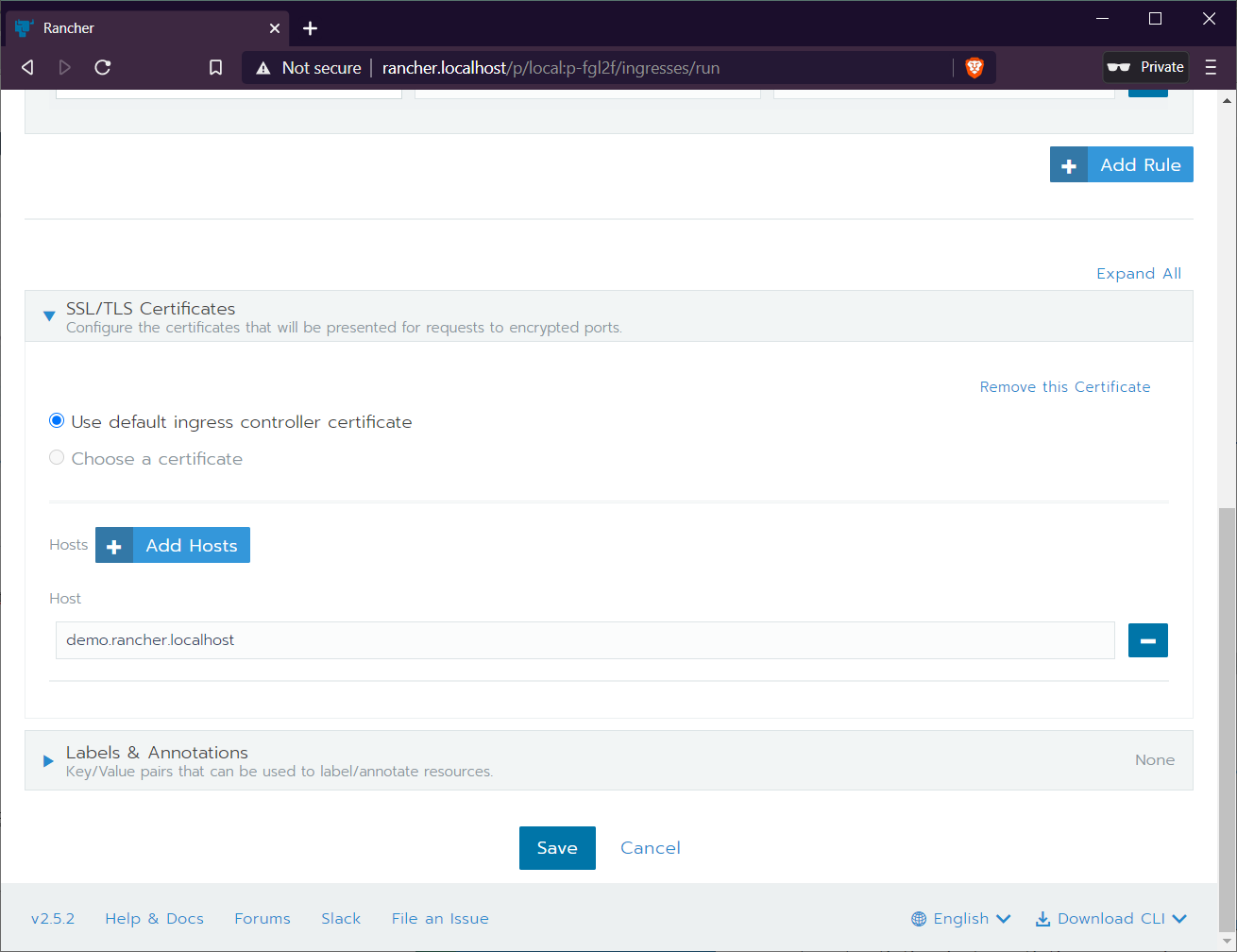


I configured the ingress with (note TWO screenshots)

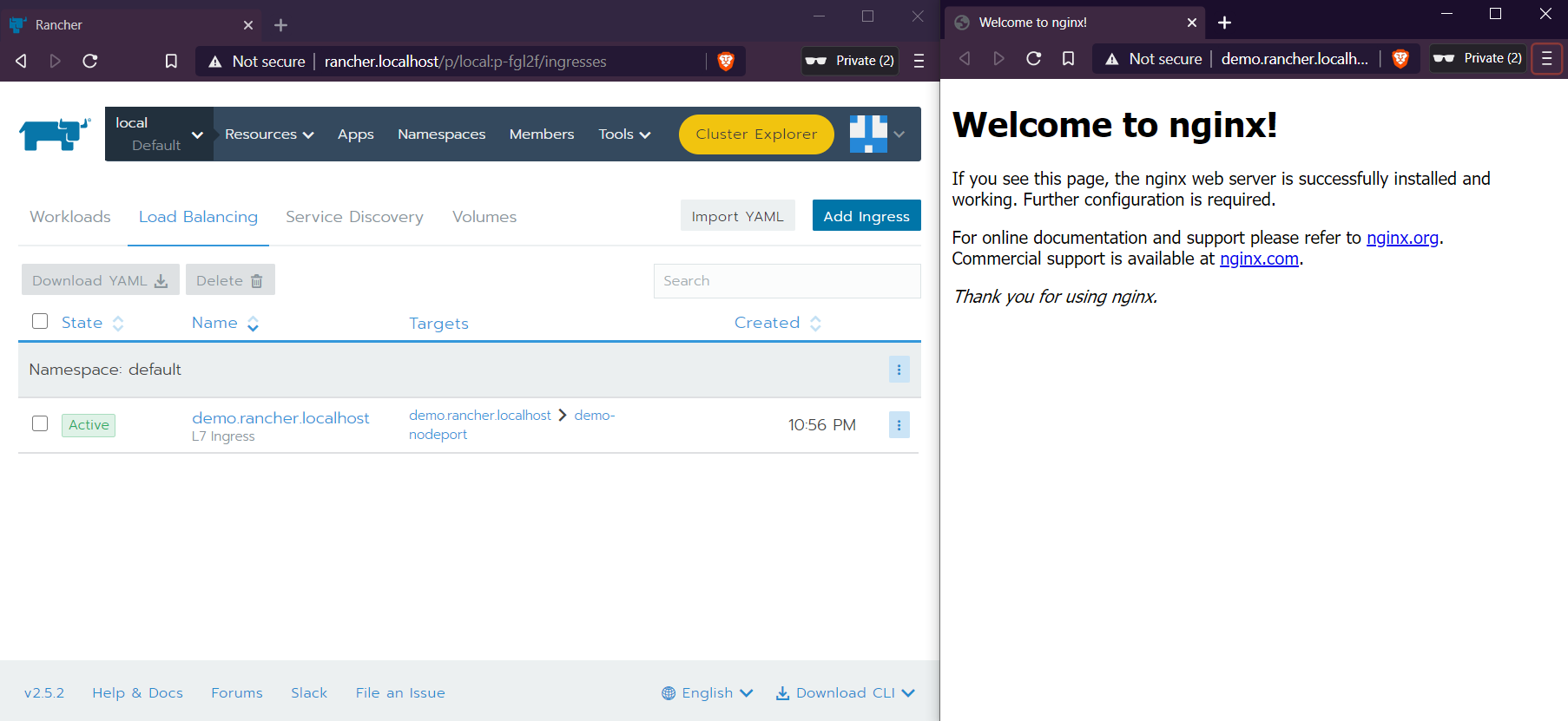
* name: **demo.rancher.localhost**
* Specify a hostname to use (set Request Host): **demo.rancher.localhost**
* Click the minus sign to remove the default workload and click the + to add a **Service**as the target backend
* Change the Target field to **demo-nodeport** and it will automatically select the correcdt port



* Add a certificate to the same host **demo.rancher.localhost**



After clicking save, you’ll see that there’s a new ingress that takes you where you expect to go thanks to some Rancher and DNS magic!



In production, you’ll want to write some deployment yamls and deploy them through a CI/CD pipeline. But for now, enjoy having a dev lab on your laptop to explore k8s and Rancher to manage it!

Thank you to [Brandon Gulla](https://medium.com/u/ce902113ac3a?source=post_page-----7404f288342f--------------------------------) for introducing me to k3d and [Thorsten K](https://medium.com/u/f4cb33e4a07f?source=post_page-----7404f288342f--------------------------------) for the k3d demo that continues to inspire me and hopefully many other devs!!

Here’s a gist of a consolidated PS1 script I used to spin up the cluster and install Rancher. The prerequisites are to install Docker and Chocolatey.

## [ITNEXT](https://itnext.io/?source=post_sidebar--------------------------post_sidebar-----------)

#### ITNEXT is a platform for IT developers & software engineers…

Follow

#### 23

* [Kubernetes](https://itnext.io/tagged/kubernetes)
* [K3d](https://itnext.io/tagged/k3d)
* [K3s](https://itnext.io/tagged/k3s)
* [Rancher](https://itnext.io/tagged/rancher)
* [Nginx](https://itnext.io/tagged/nginx)