

Setup your Laptop for Kubernetes and get produktive

*Archy (Ayrat Khayretdinov)
Prune (Sebastien THOMAS)*



BUILDING FOR THE ROAD AHEAD

DETROIT 2022



KubeCon



CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

October 24-28, 2021



Archy
CNCF Ambassador
Google



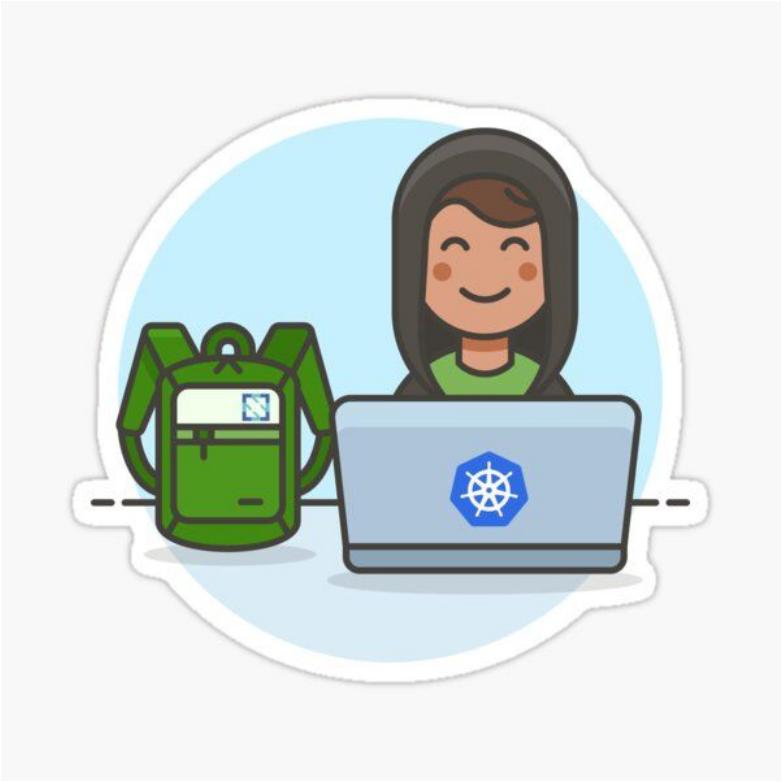
Prune
Software Engineer



Wunderkind











Who I am



Google Cloud



CLOUD NATIVE
EASTERN CANADA



CLOUD NATIVE
COMPUTING FOUNDATION
AMBASSADOR



Archy Khayretdinov
CNCF Ambassador



Who I am



I'm Sébastien / Prune

Lead Software Engineer,
part of the Infrastructure Team at **Wunderkind**
System Engineer, K8s fan, casual Go developer



Co-organiser of *CNCF/Kubernetes Eastern-Canada* meetups

Living in Quebec, CANADA



<https://www.linkedin.com/in/prune/>



<https://github.com/prune998>



CLOUD NATIVE
EASTERN CANADA

Scenarios

- **Scenario 1** - Setup Laptop with Kubectl and Kubernetes
- **Scenario 2** - Authenticating to Kubernetes Cluster (kubeconfig)
- **Scenario 3** - Create K8s resources using kubectl commands ()
- **Scenario 4** - Write Declarative YAML Manifests and kubectl apply
 - Deploy apps to multiple clusters, namespaces
 - Deploy apps to multiple environments (kustomize)
 - Deploy 3rd party apps (helm)
- **Scenario 5**

Agenda

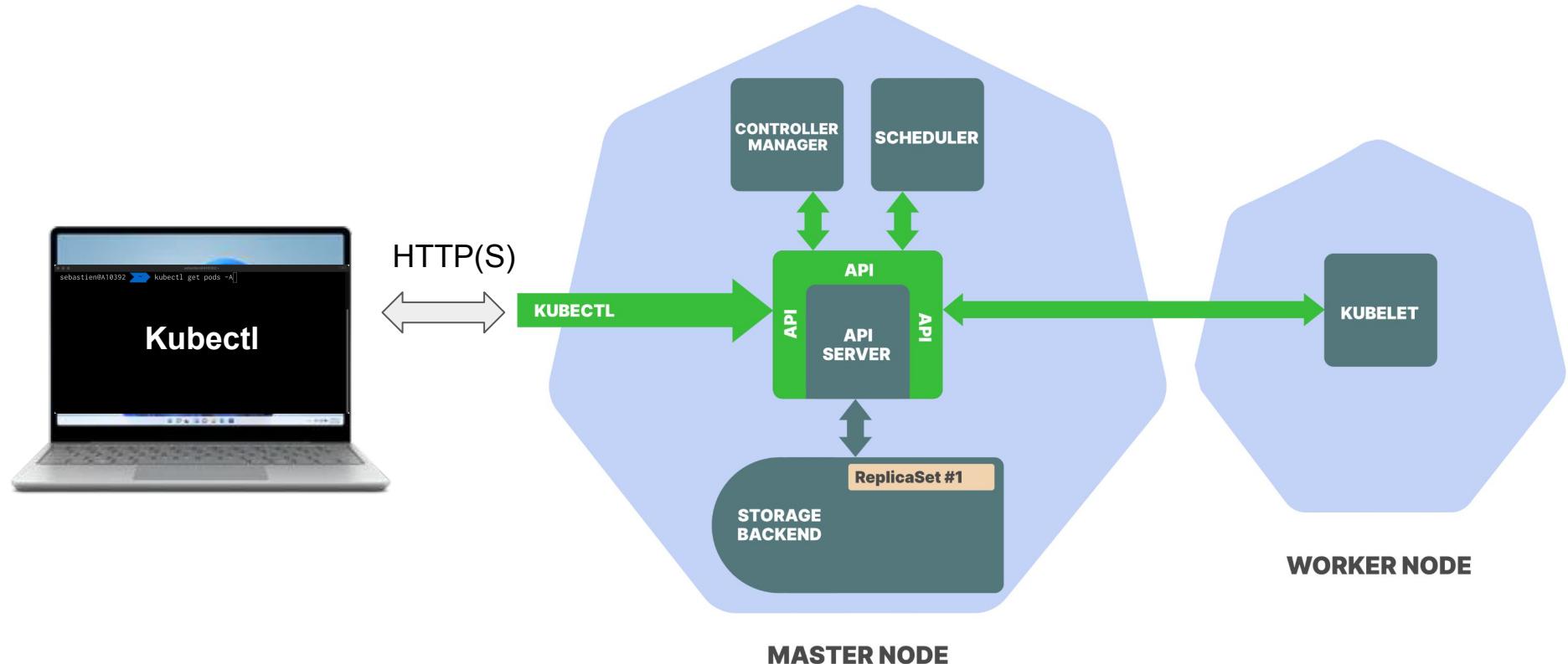
- No more SSH, I need Kubectl
- Replacing Docker-For-Desktop
- Use Containers without Docker
- Create a local K8s cluster
- I'm typing too much kubectl commands
- Kubectl arguments are too long
- My terminal is clogged, I can't read it
- Too many outputs from k8s commands

Agenda

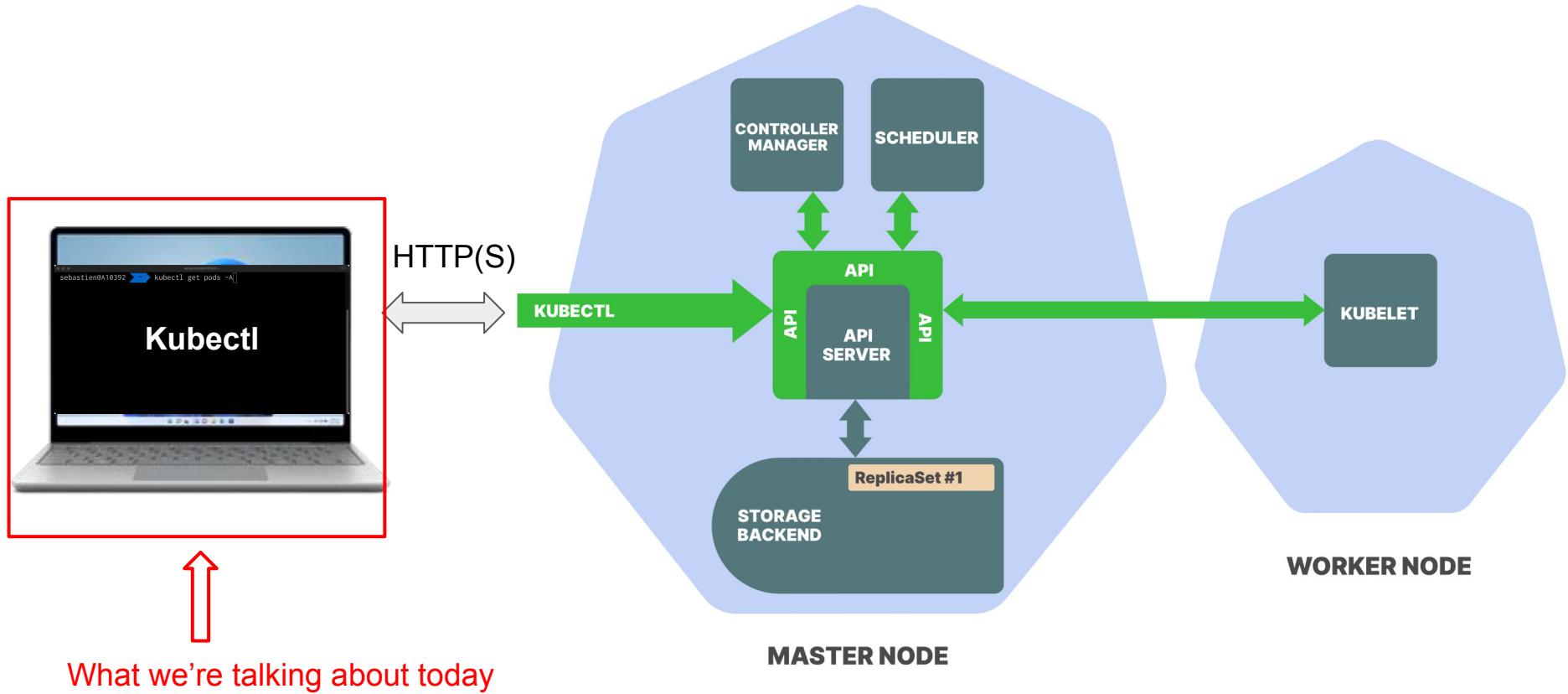
- I want more from kubectl
- Help me switch my k8s context
- Reading logs is a pain
- I'm fed up with the Shell
- YAML is so hard to manage
- I'm too advanced for this talk, bring it on !
- References

What we need for this tutorial?

Kubectl: Internals



Kubectl: Internals



Requirements for this Tutorial

Mac OS:

- Iterm2 (<https://iterm2.com/>)
- ZSH (default OsX shell for quite some time now)
- Working **Homebrew** installer
- Any **Kubernetes Cluster** (Docker Desktop or K8s on Cloud)

Linux:

- Any terminal
- ZSH preferably or Bash
- Any **Kubernetes Cluster** (Docker Desktop or K8s on Cloud)

Requirements for this Tutorial: K8s Cluster

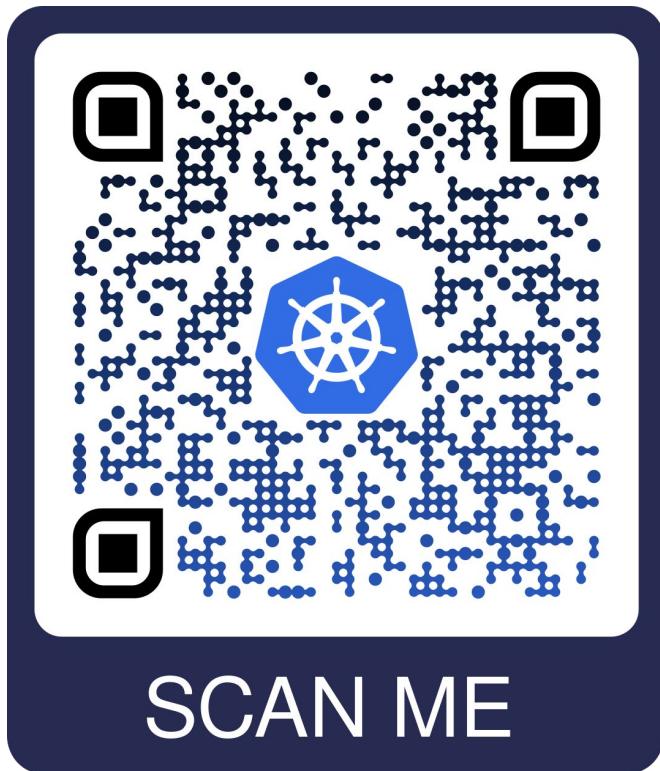
Pick ANY option!



Azure Kubernetes Service (AKS)

Run it yourself !

https://rebrand.ly/k8s_tools



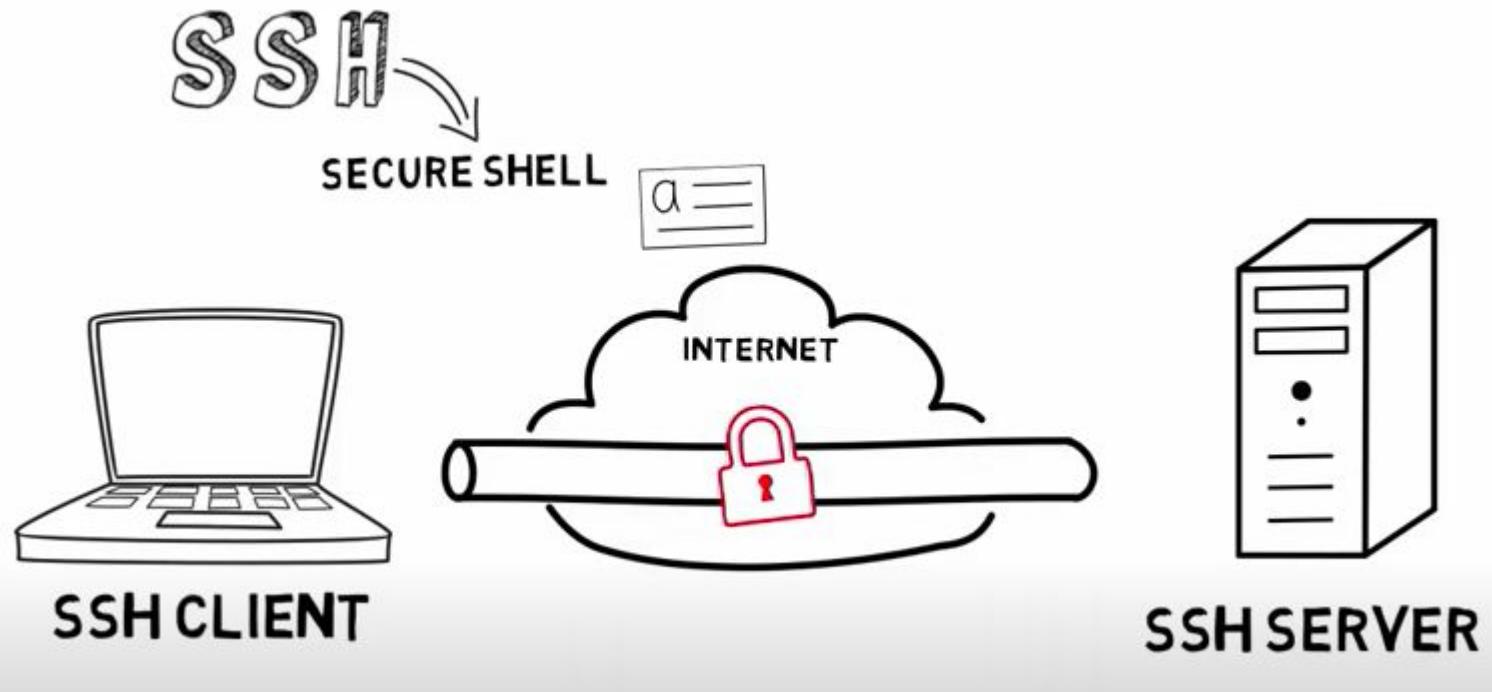


Part 1

New SSH

KubeCTL

Using ssh to connect to VM



```
$ ssh -i ~/.ssh/id_rsa admin@192.168.24.25
```

Kubectl is a new SSH

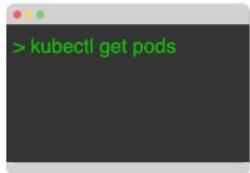
← Tweet



Kelsey Hightower @kelseyhightower

kubectl is the new SSH. I can do this all day.

KUBECTL



KUBERNETES API



KUBERNETES

Demo: Installing Kubectl

Install kubectl

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl"
chmod 755 kubectl
```

(or brew install kubectl or install through gcloud CLI...)

For Linux:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod 755 kubectl
```

Kubectl - Know before it's too late :)

Always try to use the same kubectl version as the server you are targeting

```
kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"22",
GitVersion:"v1.22.2"}
Server Version: version.Info{Major:"1", Minor:"20+",  

GitVersion:"v1.20.9-gke.1001"}
```

```
WARNING: version difference between client (1.22) and server (1.20)
exceeds the supported minor version skew of +/-1
```

Local K8s Dev Environment

Mac/Windows/Linux

What is the requirements?

Must have:

- **Build Docker Images (Docker Engine)**

```
# docker build -t image_name .
```

- **Run Docker Containers**

```
# docker run image_name
```

- **Deploy to Kubernetes**

```
# kubectl create deployment --image=image_name
```

[Optional] Nice to have:

- **Able to run Docker Compose**

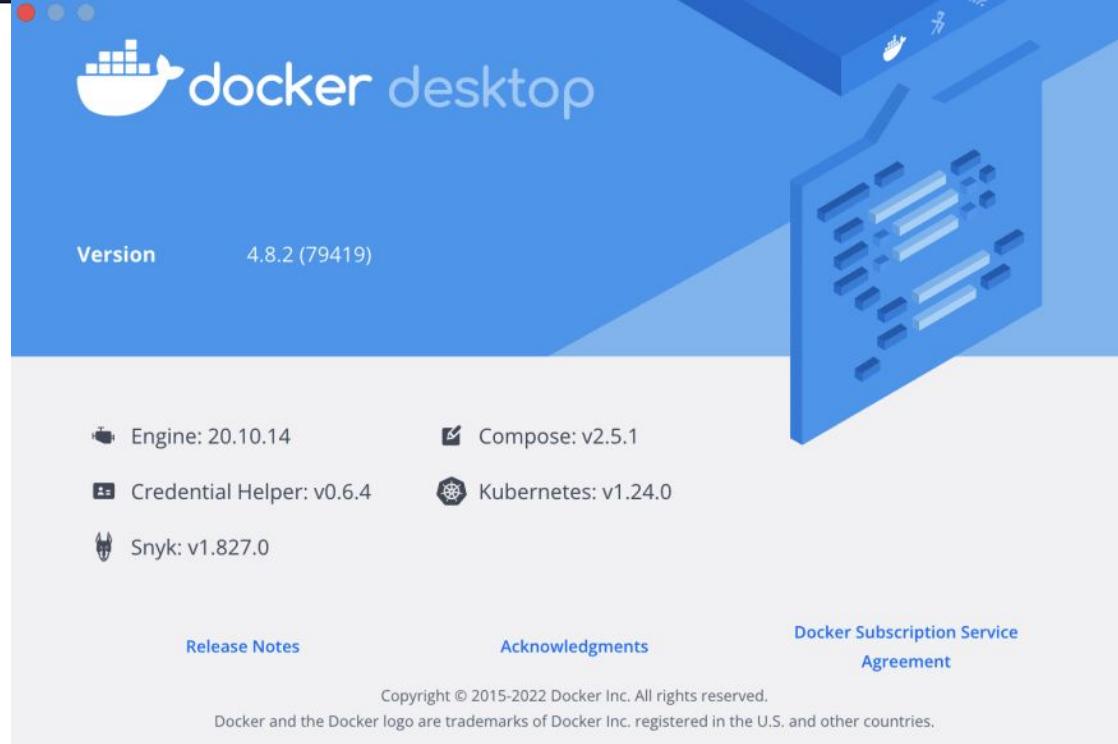
```
# docker-compose up .
```

- **Scan Images**

```
# docker scan image
```

- **Nice UI**

Docker for Desktop



Apple

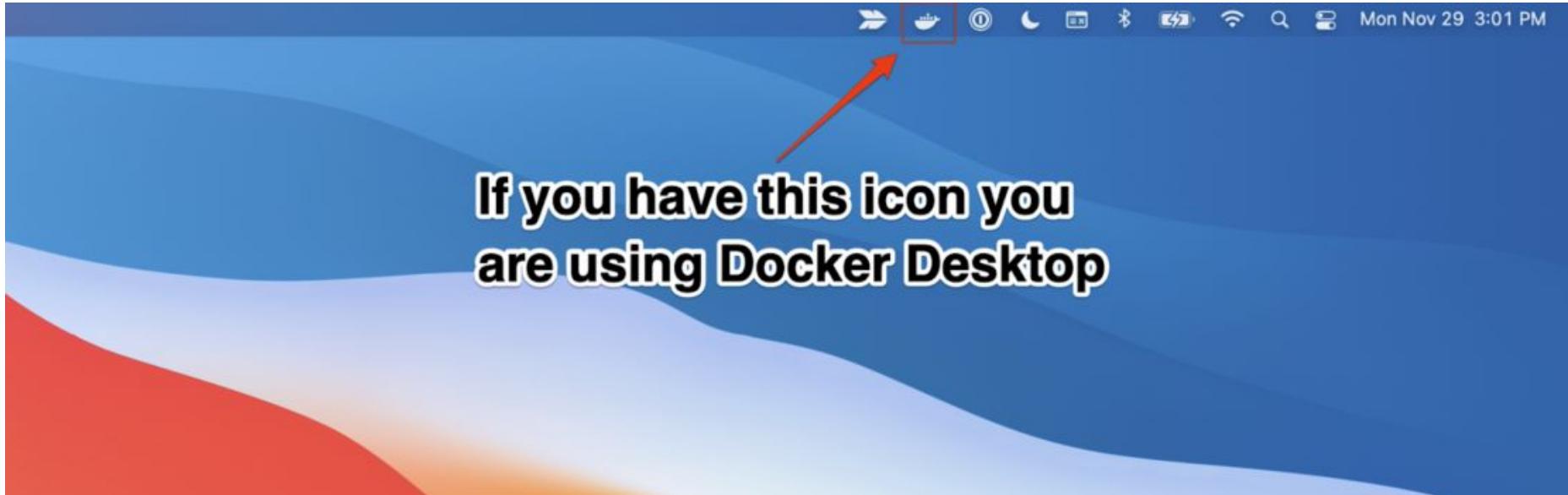


Windows



Linux

[MAC] Docker for Desktop



What's problem with Docker for Desktop?



UNITED STATES ▾

SOFTWARE DEVELOPMENT

CLOUD COMPUTING

MACHINE LEARNING

ANALYTICS

IDG TECH(TALK) COMMUNITY

NEWSLETTER

Home > Containers > Docker

Docker Desktop is no longer free for enterprise users

Docker is changing its pricing plans, ending free Docker Desktop use for larger business customers and replacing its Free plan with a Personal plan.



By **Scott Carey**

Managing Editor, News, InfoWorld | AUG 31, 2021 8:00 AM PDT

Personal



Ideal for individual developers, education, open source communities, and small businesses.

\$0

Includes
Docker Desktop

Pro



Ideal for individual developers

\$5 /month

Includes
Docker Desktop

Team



Ideal for development teams

\$7 /user/month
minimum 5 seats

Includes
Docker Desktop

Business



50+

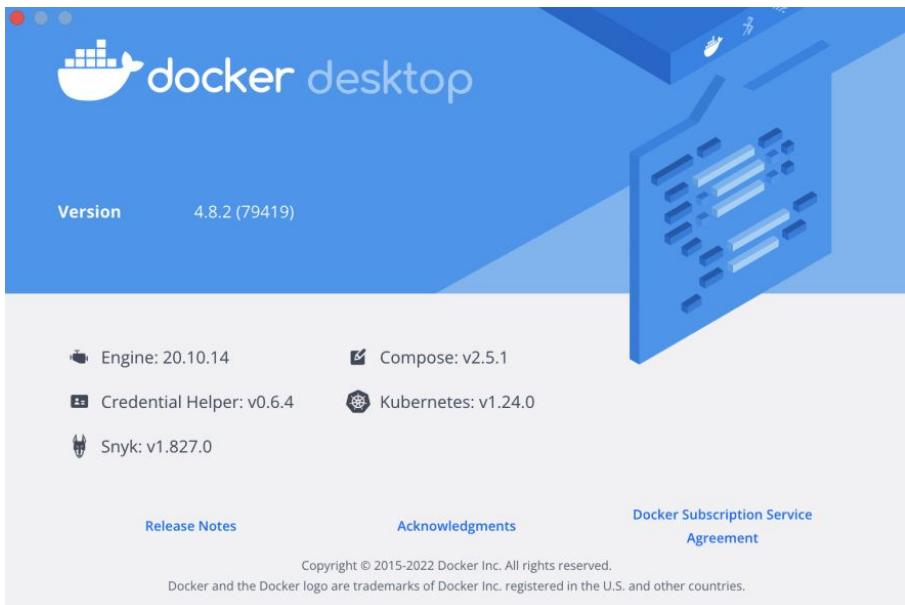
Ideal for medium and large businesses

\$21 /user/month

Includes
Docker Desktop

Docker for Desktop: Pros and Cons

- Multi platform
- UI
- Great Dev Experience
- Paid product for enterprise
- Frequent updates
- High CPU usage
- Little k8s control



Replacing Docker-For-Desktop

For Mac and Windows

Docker for Desktop Alternatives



RANCHER DESKTOP



minikube



REPLACING NEED FOR DOCKER DESKTOP



Using Rancher Desktop as Docker Desktop Replacement



Download Mac (Apple Silicon)



Download Mac (Intel)



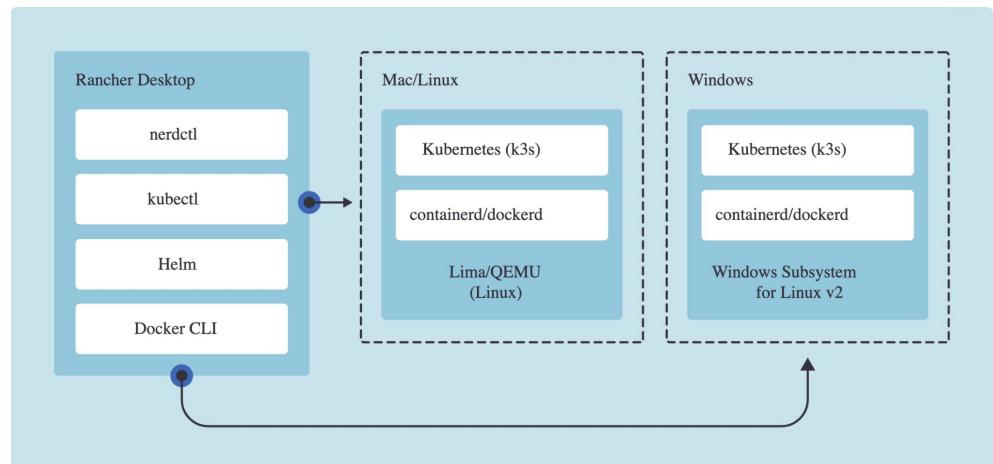
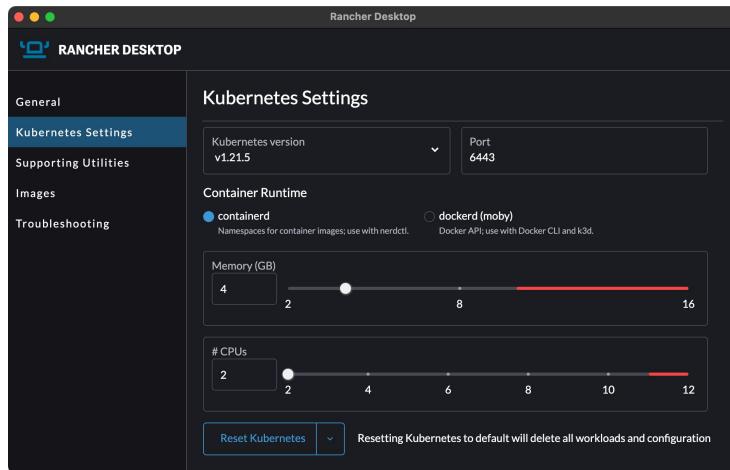
Download Windows



Install on Linux



RANCHER DESKTOP



[Mac] Containers without Docker: Colima



- [Colima](#) Container runtimes on macOS (and Linux) with minimal setup
 - Intel and M1 Macs support
 - Simple CLI interface
 - Docker and **Containerd** support
 - Port Forwarding
 - Volume mounts
 - Kubernetes (hidden K3s)
 - Fully replace Docker-for-Desktop (**but no UI**)

[Mac] Containers without Docker: Colima



```
brew install colima

# optional, not needed for containerd
brew install docker

# default using Docker runtime
colima start

# start kubernetes local cluster
colima start --with-kubernetes

# remove docker completely
colima start --runtime containerd --with-kubernetes
```

Using minikube as Docker Desktop Replacement

- [Minikube](#): local Kubernetes focusing on making it easy to learn and develop for Kubernetes.



- Official Kubernetes tool maintained by K8s SIGs
- Supports the latest Kubernetes release
- Cross-platform (Linux, macOS, Windows)
- Multiple container runtimes (CRI-O, containerd, docker)
- Can use minikube as a Docker Desktop Replacement



```
# Install Docker CLI (https://download.docker.com/ or `brew install docker`)  
# Install Minikube (https://minikube.sigs.k8s.io/docs/start/)  
# Install HyperKit (https://minikube.sigs.k8s.io/docs/drivers/hyperkit/)  
minikube start --driver hyperkit # will be auto-detected  
Or  
minikube start --container-runtime=docker --no-kubernetes  
minikube docker-env  
eval $(minikube docker-env)
```

Local Kubernetes

Kind

Local Cluster: Kind

Kind stands for Kubernetes in Docker.

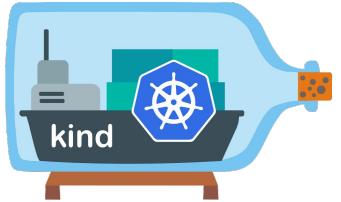
It's a tool to create **lightweight** Kubernetes clusters using **Docker** container "nodes".



- Official Kubernetes tool maintained by K8s SIGs
- It can be used to deploy Local K8s cluster or for CI
- Support ingress / LB (with some tuning)
- Support deployment of multiple clusters / versions
- Supports deployment of single or multi node clusters

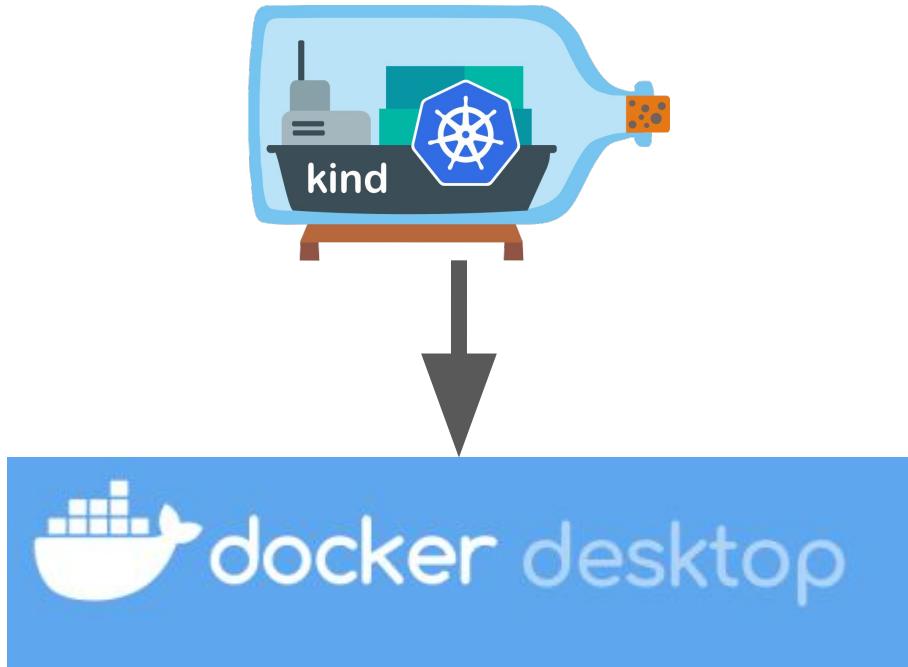
<https://kind.sigs.k8s.io/>

Local Cluster: Kind

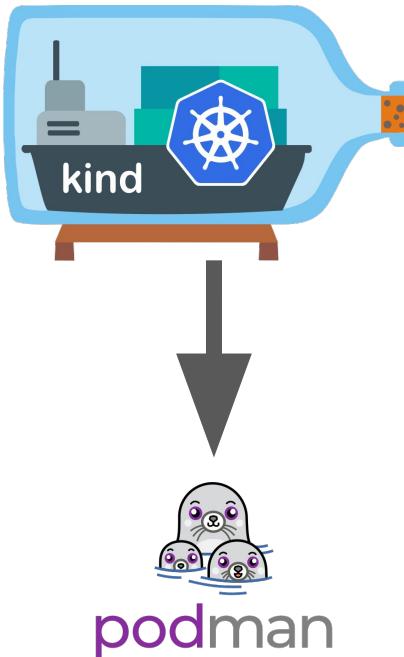


```
$ brew install kind  
  
$ kind create cluster  
  
$ kind get clusters
```

Local Cluster: Kind



Local Cluster: Kind



Replacing Docker Engine

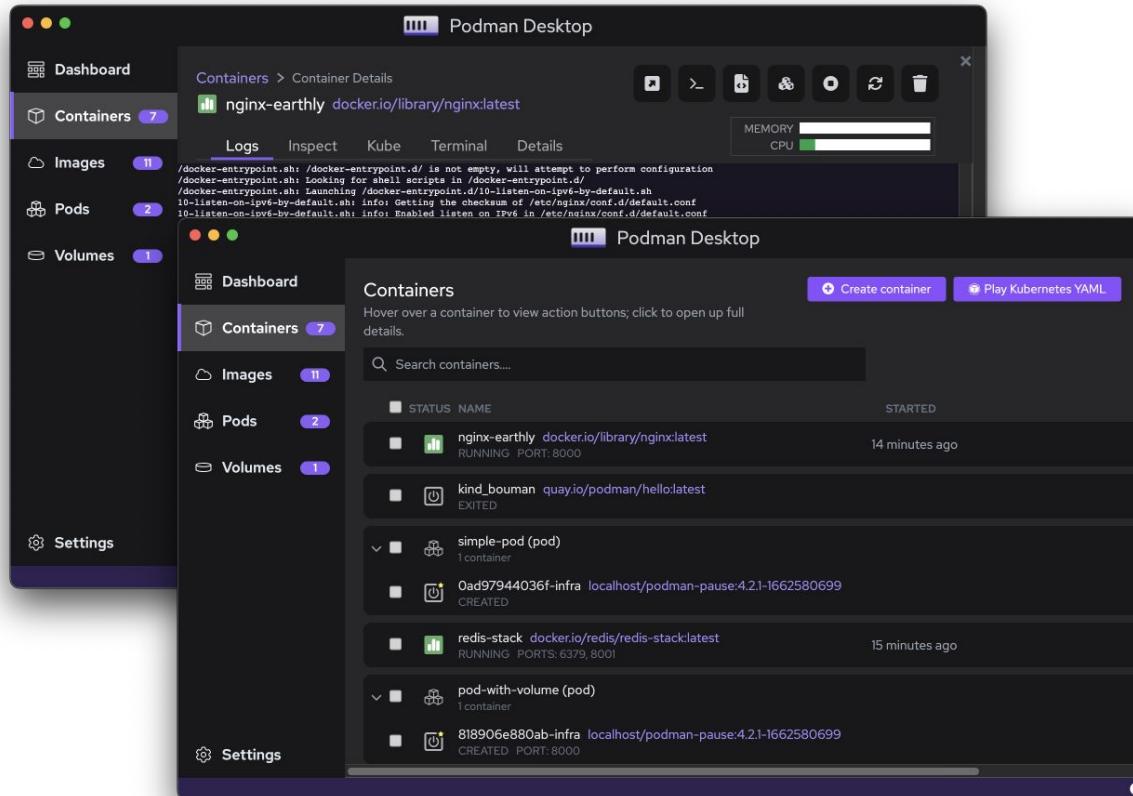
podman=rootless containers

Containers without Docker Engine: PodMan



- Podman is the container Swiss-Army knife maintained by RedHat
 - Replaces **Docker-For-Desktop, Docker Engine, Docker CLI**
 - Works for Windows, Mac, Linux
 - Podman version 3.4+ now support **M1 Apple Macs**
 - **Rootless** mode provide more security
 - Multiple image formats including the OCI and Docker image
 - Container image management
 - (managing image layers, overlay filesystems, etc)
 - **However doesn't provide Kubernetes cluster!!! So use Kind**

(Optional) Podman Desktop



Demo: Installing Podman and Kind

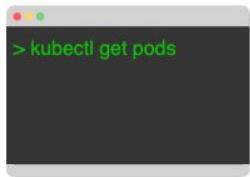


Authenticating to Kubernetes

kubeconfig

Authenticating to Kubernetes Cluster

KUBECTL

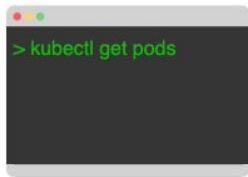


KUBERNETES



KUBECONFIG

KUBECTL



KUBERNETES API

KUBERNETES



KUBECONFIG

Demo: Authenticating to Kubernetes



KUBECONFIG

Deploying application on k8s

```
# kubectl create  
# kubectl apply
```

Creating kubernetes resources

- Creating k8s resources with command line

```
# kubectl create deployment webserver --image=nginx --port=80
# kubectl create service clusterip webserver --tcp 80:80
# kubectl create configmap webserver-config --from-file web.config
```

- Creating k8s resources **Declaratively** with **YAML**:

```
# kubectl apply -f nginx_pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Creating Pods and Deployments

Kubectl can be used to generate YAML manifests and help you build your own:

- `kubectl create <resource>` to create a resource (kubectl run for pods)
- `-output yaml (-o yaml)` to dump the yaml
- `--dry-run=client` to simulate the resource
- `kubectl explain <resource>` to get explanation of the resource

Demo: Deploying applications on K8s with YAML

kubectl apply



Part 2

I'm Typing Too Much Kubectl

Aliasing

I'm Typing Too Much Kubectl

Just create some Alias: use **k** instead of **kubectl**

```
alias k=kubectl
```

You can add this command to your *.zshrc* so it is added to all your (new) shells

Spoiler alert: it's the same for Bash (in your *.bashrc*)

BONUS

- Use singular name of the resource
- Use short name of the resource

pods -> pod -> po

```
k api-resources
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
pod	po	v1	true	Pod
deployment	deploy	apps/v1	true	Deployment
ingress	ing	networking.k8s.io/v1	true	Ingress
service	svc	v1	true	Service
namespace	ns	v1	false	Namespace

Kubectl arguments too long

Shell Completion

Shell Setup: Completion

Completions allows to use <tab> to list options or some list of resources

```
source <(kubectl completion zsh)
```

For Bash:

```
source <(kubectl completion bash)
complete -o default -F __start_kubectl k
```

Shell Setup: BASH

Alias, Completion... in your .zshrc: use **k** instead of **kubectl**

```
alias k=kubectl
source <(kubectl completion bash)
complete -o default -F __start_kubectl k
ulimit -n 2048          # kubectl opens one cnx (file) per resource
```

Shell Setup: Side note

Kubectl, or the Kube GO client, sometimes open multiple connexions to the server, sometimes too much. This is much experiences when using `helm diff` on a large setup

Change your Open-File limit (file descriptors):

```
ulimit -n 2048    # kubectl opens one cnx (file) per resource
```

Terminal for Produktivity

Oh My ZSH !

Shell Setup: ZSH

ZSH, OhMyZSH and themes:

- [Oh-My-ZSH](#) : lots of features in your shell
 - Use plugins

```
plugins=(brew kubectl git python tmux vault terraform)
```

- Themes
 - [Agnoster ZSH theme](#): better prompt using Powerline Fonts
 - [PowerLevel10k](#): emphasizes speed, flexibility and out-of-the-box experience
- Fonts
 - Powerline Font: recommend [NerdFonts](#) Inconsolata or FiraCode

Shell Setup: ZSH

Alias k instead of kubectl is not needed anymore

```
# alias k=kubectl # replaced by OhMyZSH kubectl plugin
source <(kubectl completion zsh)
ulimit -n 2048          # kubectl opens one cnx (file) per resource
```

Kubernetes Shell Tooling



There's too many output from kubectl, I can't read it

Kubecolor

Colorize the output of kubectl command, highlight important features



```
brew install kubecolor/tap/kubecolor
```

Or

```
go install -ldflags="-X main.Version=latest"  
github.com/kubecolor/kubecolor/cmd/kubecolor@latest
```

```
# in your .zshrc or .bashrc  
  
alias kubectl=kubecolor  
  
compdef kubecolor=kubectl # only needed for zsh
```

Kubecolor

(<https://github.com/kubecolor/kubecolor>)

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-6d4b75cb6d-j2774	1/1	Running	0	20h
kube-system	coredns-6d4b75cb6d-jvmg9	1/1	Running	0	20h
kube-system	etcd-demo-control-plane	1/1	Running	0	20h
kube-system	kindnet-wx2sx	1/1	Running	0	20h
kube-system	kindnet-wx9jg	1/1	Running	0	20h
kube-system	kube-apiserver-demo-control-plane	1/1	Running	0	20h
kube-system	kube-controller-manager-demo-control-plane	1/1	Running	43 (3h11m ago)	20h
kube-system	kube-proxy-6wh5v	1/1	Running	0	20h
kube-system	kube-proxy-nknzg	1/1	Running	0	20h
kube-system	kube-scheduler-demo-control-plane	1/1	Running	43 (3h11m ago)	20h
local-path-storage	local-path-provisioner-6b84c5c67f-6fhrr	1/1	Running	0	20h

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-6d4b75cb6d-j2774	1/1	Running	0	20h
kube-system	coredns-6d4b75cb6d-jvmg9	1/1	Running	0	20h
kube-system	etcd-demo-control-plane	1/1	Running	0	20h
kube-system	kindnet-wx2sx	1/1	Running	0	20h
kube-system	kindnet-wx9jg	1/1	Running	0	20h
kube-system	kube-apiserver-demo-control-plane	1/1	Running	0	20h
kube-system	kube-controller-manager-demo-control-plane	1/1	Running	43 (3h19m ago)	20h
kube-system	kube-proxy-6wh5v	1/1	Running	0	20h
kube-system	kube-proxy-nknzg	1/1	Running	0	20h
kube-system	kube-scheduler-demo-control-plane	1/1	Running	43 (3h19m ago)	20h
local-path-storage	local-path-provisioner-6b84c5c67f-6fhrr	1/1	Running	0	20h

Kubecolor

(<https://github.com/kubecolor/kubecolor>)

```
prune@arcadia:~
> export KUBECOLOR_OBJ_FRESH=12h # highlight resources newer than 12h
k get pods -A
k run another-test-pod --image=alpine:latest sleep 30
k get pods
sleep 10
k get pods
NAMESPACE          NAME                READY   STATUS        RESTARTS   AGE
default            test-pod           0/1     CrashLoopBackOff  5 (18s ago)  6m24s
kube-system        coredns-6d4b75cb6d-j2774  1/1     Running      0          20h
kube-system        coredns-6d4b75cb6d-jvmg9   1/1     Running      0          20h
kube-system        etcd-demo-control-plane  1/1     Running      0          20h
kube-system        kindnet-wx2sx         1/1     Running      0          20h
kube-system        kindnet-wx9jg         1/1     Running      0          20h
kube-system        kube-apiserver-demo-control-plane  1/1     Running      0          20h
kube-system        kube-controller-manager-demo-control-plane  1/1     Running      43 (3h58m ago)  20h
kube-system        kube-proxy-6wh5v       1/1     Running      0          20h
kube-system        kube-proxy-nknzg       1/1     Running      0          20h
kube-system        kube-scheduler-demo-control-plane  1/1     Running      43 (3h57m ago)  20h
local-path-storage local-path-provisioner-6b84c5c67f-6fhrr  1/1     Running      0          20h
pod/another-test-pod created
NAME              READY   STATUS        RESTARTS   AGE
another-test-pod  0/1     ContainerCreating  0          0s
test-pod          0/1     CrashLoopBackOff  5 (18s ago)  6m24s
NAME              READY   STATUS        RESTARTS   AGE
another-test-pod  1/1     Running      0          11s
test-pod          0/1     CrashLoopBackOff  5 (29s ago)  6m35s
  ✓  10s ✘  11:41:11 ⏱
```

Shell Setup: ZSH

Alias, Completion... in your .zshrc: revisited

```
export PATH="${KREW_ROOT:-$HOME/.krew}/bin:$PATH"
alias kubectl=kubecolor
source <(kubectl completion zsh)
complete -F __start_kubectl k
compdef kubecolor=kubectl
source <(stern --completion=zsh)
ulimit -n 2048          # kubectl opens one cnx (file) per resource
```

I need to read my logs

stern

Stern allows you to **tail multiple pods** on Kubernetes and **multiple containers** within the pod. Each result is **color** coded for quicker debugging.

Install:

```
brew install stern
```

```
stern -n dv-oma dv-oma

+ dv-oma-dev-77df6c779f-6cvzh >api
+ dv-oma-dev-77df6c779f-6cvzh >cloud-sql-proxy
+ dv-oma-dev-77df6c779f-6cvzh >frontend
+ dv-oma-dev-77df6c779f-krng1 >cloud-sql-proxy
+ dv-oma-dev-77df6c779f-krng1 >frontend
+ dv-oma-dev-77df6c779f-krng1 >api
dv-oma-dev-77df6c779f-6cvzh cloud-sql-proxy 2021/10/06 21:13:31 ephemeral certificate will expire soon,
refreshing now.
dv-oma-dev-77df6c779f-6cvzh cloud-sql-proxy 2021/10/06 22:32:41 Instance closed connection
dv-oma-dev-77df6c779f-krng1 cloud-sql-proxy 2021/10/07 21:13:01 ephemeral certificate will expire soon,
refreshing now.
dv-oma-dev-77df6c779f-6cvzh api 10.220.4.26 - - [08/Oct/2021:14:58:48 +0000] "GET /api/metrics HTTP/1.1" 403 26
"" "Datadog Agent/7.31.1"
```

Containers inside the pod

Pod name (each pod a different color)

Real log from the container

```
stern -n dv-oma dv-oma
+ dv-oma-dev-77df6c779f-6cvzh >api
+ dv-oma-dev-77df6c779f-6cvzh >cloud-sql-proxy
+ dv-oma-dev-77df6c779f-6cvzh >frontend
+ dv-oma-dev-77df6c779f-krngl >cloud-sql-proxy
+ dv-oma-dev-77df6c779f-krngl >frontend
+ dv-oma-dev-77df6c779f-krngl >api
dv-oma-dev-77df6c779f-6cvzh cloud-sql-proxy 2021/10/06 21:13:31 ephemeral certificate will expire soon,
refreshing now.
dv-oma-dev-77df6c779f-6cvzh cloud-sql-proxy 2021/10/06 22:32:41 Instance closed connection
dv-oma-dev-77df6c779f-krngl cloud-sql-proxy 2021/10/07 21:13:01 ephemeral certificate will expire soon,
refreshing now.
dv-oma-dev-77df6c779f-6cvzh api 10.220.4.26 - - [08/Oct/2021:14:58:48 +0000] "GET /api/metrics HTTP/1.1" 403 26
"" "Datadog Agent/7.31.1"
```

Logs in JSON ?

```
--output json | jq '.'
```

Tail only the last 10 logs ?

```
--tail 10
```

Search pods in all namespaces ?

```
--all-namespaces
```

Search pods per labels (selectors) ?

```
-l app=prometheus
```

New: Logs in JSON with namespace/pod/container ?

```
--all-namespaces --output extjson | jq `.'
```

New: Logs in JSON with pretty-print with colors ?

```
--output ppextjson
```

I want to extend Kubectl with new commands

Kubectl Plugins

Kubectl Plugin Manager

kubectl krew list

Krew - kubectl plugin manager

Krew is kubectl plugin manager.

Install: <https://krew.sigs.k8s.io/docs/user-guide/setup/install>

Adding Plugins: <https://github.com/kubernetes-sigs/krew-index>



```
kubectl krew list
```

PLUGIN	VERSION
ctx	v0.9.4
krew	v0.4.1
ns	v0.9.4
whoami	v0.0.36

```
kubectl krew search
```

NAME	DESCRIPTION	
INSTALLED		
access-matrix	Show an RBAC access matrix for server resources	no
blame	Show who edited resource fields.	no
cert-manager	Manage cert-manager resources inside your cluster	no
ctx	Switch between contexts in your kubeconfig	yes
...		

How to create YAML manifest for my deployment?

kubectl **neat**

Switching Namespaces is a pain

kubectl ns

Switching namespaces with Kubeconfig context

List / change Namespaces (ns)

```
# List all namespaces, current NS in yellow
kubectl ns

datadog-agents
default
kube-public
kube-system

# Set default NS by hand
kubectl config set-context --current --namespace=kube-system

# Set default Namespace
kubectl ns kube-system
```

Switching Cluster Context is a pain

kubectl **ctx**

Switching clusters with Kubeconfig context

List / change Cluster (context)

```
# list all the existing context, current one in yellow
kubectl ctx

arn:aws:eks:us-east-1:111111111111:cluster/eks-example
gke-dv-st-cluster-1
gke-dev_us-central1_test-gke-cluster

# change context "manually"
kubectl config use-context gke-dev_us-central1_test-gke-cluster

# change the context using ctx
k ctx gke-dev_us-central1_test-gke-cluster
```

Local Cluster: Bonus

Export the current Kube Config into another file:

```
kubectl config view --minify --raw > ~/.kube/demo.config
```

```
export KUBECONFIG=~/.kube/demo.config
```

```
k ctx
```

```
k get pods -A
```

Our favorite kubectl plugins

- whoami: who the cluster thinks you are from your authentication
<https://github.com/rajatjindal/kubectl-whoami>
- who-can: RBAC rules introspection
<https://github.com/aquasecurity/kubectl-who-can>
- View-secret: directly view secret content without having to decode

```
kubectl krew install ctx ns stern whoami who-can view-secret
```

I want multiple context at
the same time

kubie

Kubie offers **context** switching, **namespace** switching and **prompt** modification in a way that makes each shell independent from others.



```
# install
brew install kubie

# display a selectable menu of contexts
kubie ctx

# display a selectable menu of namespaces
kubie ns

# execute a command in all contexts matched by the wildcard and in the
given namespace
kubie exec 'kind-*' kube-system kubectl get pods
```

Other cool plugins recommended:

- <https://twitter.com/ahmetb/status/1563263534397419522>
- get-all, images, access-matrix, resource-capacity, rbac-tool,
rbac-view, rbac-lookupm, view-allocations, view-secret,
view-utilization, whoami, who-can, eksporter/neat,

Use `kubectl krew search` **to list them all !**

I have too many context !

prompt

PowerLevel10k Prompt customization

Shell customization examples:

```
AWS: awsma-shared-dev ➜ sebastien@A10392 ➜ ~/Documents/dev/github.com/_/_/_/prune/tenant-at-the-end
```

AWS current account
Local user @ computer name
Current path
Git Branch name
(green = no changes to commit)

```
❯ ➜ ~/Doc/d/gith/p/devops ➜ master ?2 ➜ kubectl get ...
```

✓ kind-demo/kube-system * 09:02:34

K8s cluster / Namespace

What if I'm using Cloud K8s

Cloud CLIs

Cloud Provider: Google

Install / Setup:

```
brew install --cask google-cloud-sdk
gcloud components install kubectl # Optional

gcloud init
gcloud auth login
gcloud components install gke-gcloud-auth-plugin

gcloud config set compute/region us-east1
```

Configure GKE cluster in kube config:

```
gcloud container clusters get-credentials <cluster> --project <project>
```

Cloud Provider: AWS

```
curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"  
sudo installer -pkg AWSCLIV2.pkg -target /
```

Setup SSO for AWS:

```
export AWS_DEFAULT_REGION=us-east-1  
export AWS_PAGER="" # prevent aws cli to auto-page = display inline  
export BROWSER#echo # Do not open a browser, let you choose which browser to open  
complete -C '/usr/local/bin/aws_completer' aws # add that to .zshrc for  
completion  
unset AWS_ACCESS_KEY_ID AWS_SECRET_ACCESS_KEY  
aws configure sso  
aws sso login --profile profile_xxxxxxx  
export AWS_PROFILE=profile_xxxxxxx
```

Cloud Provider: AWS

Configure EKS cluster in kube config:

```
aws eks update-kubeconfig \
--region us-east-1 \
--name <cluster_name> \
--alias <friendly_name>
```

Cloud Provider: completion

Add completion for Google and AWS CLIs

```
# gcloud
source
"/usr/local/Caskroom/google-cloud-sdk/latest/google-cloud-sd
k/path.zsh.inc"
source
"/usr/local/Caskroom/google-cloud-sdk/latest/google-cloud-sd
k/completion.zsh.inc"

# AWS
complete -C '/usr/local/bin/aws_completer' aws
```



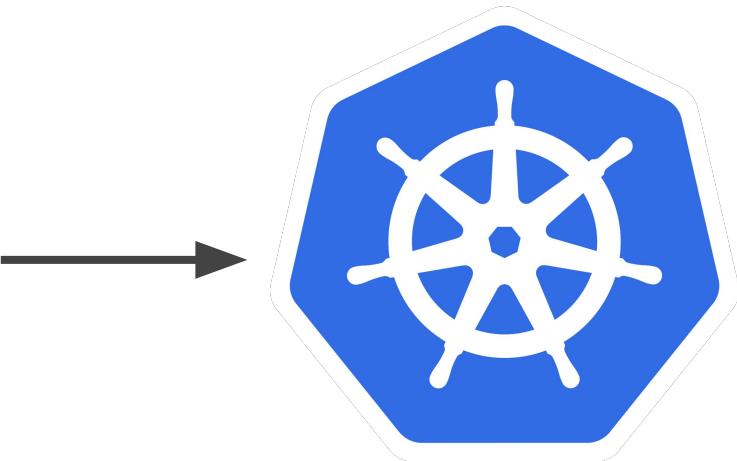
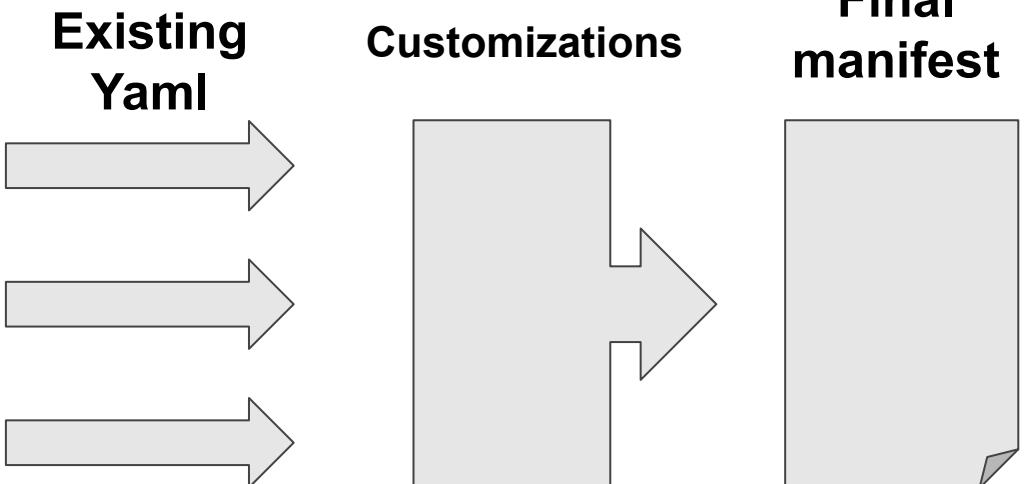
Part 3

How can I deploy my app to Dev/Stg/Prod?

Kustomize

Kustomize

Kustomize is a Kubernetes native, **template-free** way to customize application configuration



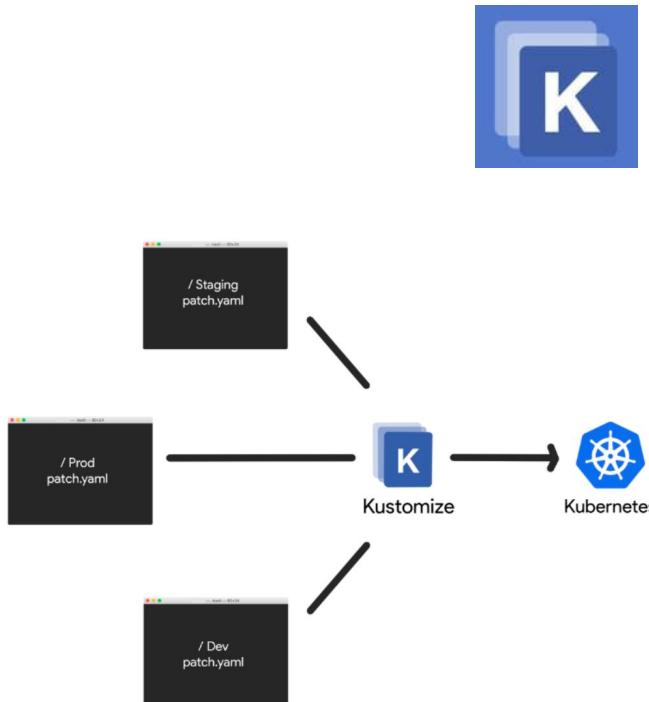
kustomize

kustomize build .

kubectl apply

How it works?

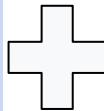
- [base] folder represents **existing Kubernetes YAML**
- [overlay] folder [dev/stg/prod]
 - **kustomization.yaml** defines files that needs to be customized/patched
 - Customizations files that needs to be overlaid on top of existing YAML files
- The output can be applied directly to k8s with kubectl



How it works?

base.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: myapp
      image: busybox
      command: ['app']
```



patch.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: myapp
      command: ['app', '--dev']
```

output.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: myapp
      image: busybox
      command: ['app', '--dev']
```

kustomize build .

dev/kustomization.yaml

```
kustomize build dev/ | kubectl apply -f -
```

```
~/myapp
└── base/
    └── deployment.yaml
    └── service.yaml
    └── kustomization.yaml
└── dev/
    └── deployment.yaml
    └── kustomization.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
```

```
kind: Kustomization
```

```
resources
```

```
- ../base/
```

```
namespace: frontend
```

```
namePrefix: dev-
```

```
patchesStrategicMerge:
```

```
- deployment.yaml
```

Kustomize Overview

System for managing and applying patches on top of existing Kubernetes YAML

Additional Features

- Add a prefix to all resource names
- Apply common:
 - Labels
 - Annotations
- Easily manage ConfigMaps and Secrets
- Apply patches (new resources)

Kustomize is a Kubernetes native, **template-free** way to customize application configuration



- Bundled with kubectl, but not all the features are available
- Better install the full version if you use specific features
- Only output rendered YAML, you have to apply manually

```
kubectl kustomize --enable-alpha-plugins /path/to/kustomize/folder
```

```
kustomize build --enable-alpha-plugins /path/to/kustomize/folder
```

Apply the resulting yaml

```
kustomize build --enable-alpha-plugins \  
/path/to/kustomize/folder | kubectl apply -f -
```

Kustomize

Kustomize requires a `kustomization.yaml` file in the target folder

```
cat /path/to/kustomize/folder/kustomization.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
```

```
resources:
  - my_resources_file.yaml
  - ../base
```

```
patches:
  - ./manifests/my_patch.yaml
```

```
generators:
  - my_generator.yaml
```

Can be used to quickly template app deployment in multiple environments

```
cat /path/to/kustomize/folder/kustomization.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
```

```
resources:
- ../../..../base
```

```
namespace: dev
namePrefix: dev-
```

How can I deploy 3rd party applications?

Helm

Artifact Hub - is a new home for Helm Charts

Helm is the package manager for Kubernetes.



Artifact Hub - is a new home for Helm Charts

<https://artifacthub.io/>

The screenshot shows the Artifact Hub homepage with a teal background. At the top left is the logo "Artifact HUB BETA". Top right are "SIGN UP" and "SIGN IN" buttons, and a gear icon. The main title "Find, install and publish Kubernetes packages" is centered. Below it is a search bar with placeholder "Search packages" and a help icon. A tip below the search bar says "Tip: Use or to combine multiple searches. Example: postgresql or mysql". Below the search bar, there's a section for sample queries with links like "browse all packages", "Helm Charts in the storage category", and "OLM operators for databases". Two large numbers are displayed: "2086 PACKAGES" and "38119 RELEASES". At the bottom, it says "Artifact Hub is an Open Source project" and has links for GitHub, Slack, and Twitter.

SIGN UP SIGN IN

Find, install and publish
Kubernetes packages

Search packages

Tip: Use **or** to combine multiple searches. Example: [postgresql](#) [or](#) [mysql](#)

You can also [browse all packages](#) - or - try one of the sample queries:

[Packages from verified publishers](#) [Prometheus packages in official repositories](#) [Helm Charts provided by Bitnami](#)

[Helm Charts in the storage category](#) [OLM operators for databases](#)

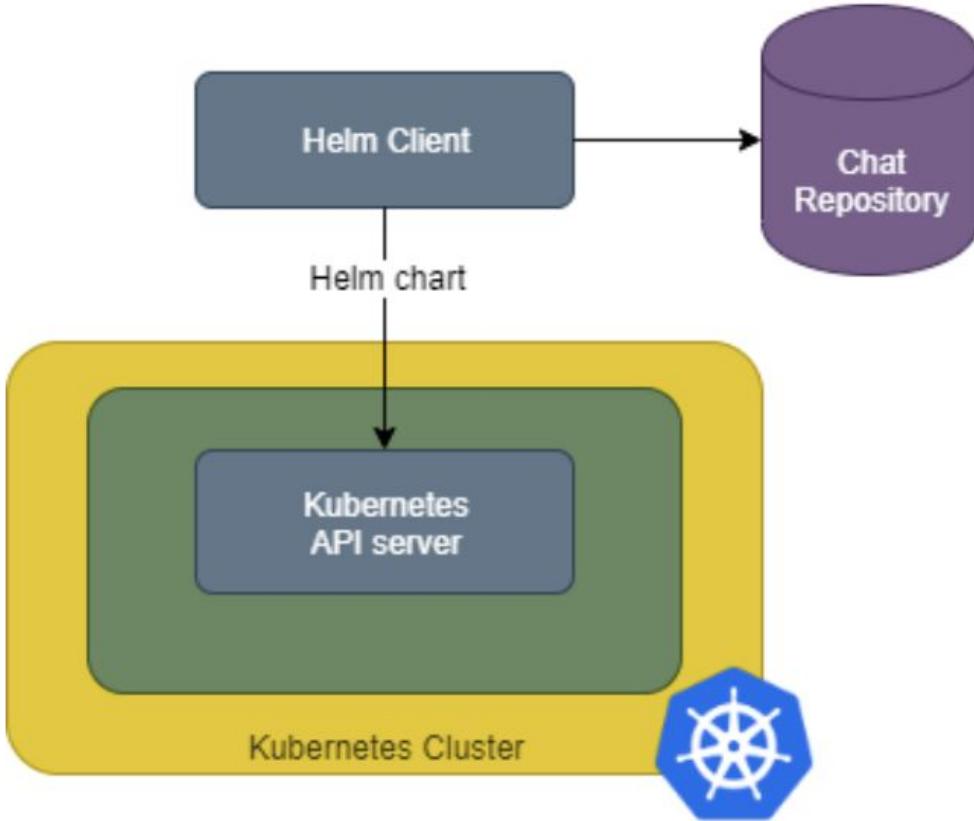
2086 | 38119

PACKAGES RELEASES

Artifact Hub is an **Open Source** project

GitHub Slack Twitter

How it works?



```
brew install helm
```

Then you can install a chart or manage remote repositories

```
helm repo add mysql-operator https://mysql.github.io/mysql-operator/  
helm repo update  
helm install my-mysql-operator mysql-operator/mysql-operator \  
--namespace mysql-operator --create-namespace
```

Helm 3 uses many strategies to manage deployments and rollbacks



Part 4 UIs

How can I observe what's deployed in my cluster?

K9s

Interfaces: K9s

- K9s - Kubernetes CLI To Manage Your Clusters In Style!
- Open-Source
- In your terminal, like `top`
- CRUD operations on K8s resources



<https://github.com/derailed/k9s>

```
brew install k9s

k9s -n <namespace>          # To run K9s in a given namespace

k9s --context <context> # Start K9s in an existing KubeConfig context

k9s --readonly # Start K9s in readonly mode - no editing
```

Interfaces: K9s

Context: kind-kind <0> all <a> Attach <l> Logs <y> YAML
Cluster: kind-kind <1> kube-system <ctrl-d> Delete <p> Logs Previous
User: kind-kind <2> default <d> Describe <shift-f> Port-Forward
K9s Rev: v0.26.3 <e> Edit <s> Shell
K8s Rev: v1.23.4 <?> Help <n> Show Node
CPU: n/a <ctrl-k> Kill <f> Show PortForward



Pods(all)[49]							
NAMESPACE↑	NAME	PF	READY	RESTARTS	STATUS	IP	NODE
argocd	argocd-application-controller-0	●	1/1	2	Running	10.244.1.6	kind-worker3
argocd	argocd-applicationset-controller-79f97597cb-9jdqw	●	1/1	1	Running	10.244.2.8	kind-worker
argocd	argocd-dex-server-6fd8b59f5b-gjxsj	●	1/1	1	Running	10.244.3.3	kind-worker2
argocd	argocd-notifications-controller-5549f47758-lr4zz	●	1/1	1	Running	10.244.3.2	kind-worker2
argocd	argocd-redis-79bdbdf78f-qbmss	●	1/1	1	Running	10.244.2.7	kind-worker
argocd	argocd-repo-server-5569c7b657-46fjd	●	1/1	1	Running	10.244.1.2	kind-worker3
argocd	argocd-server-664b7c6878-vccmq	●	1/1	1	Running	10.244.3.7	kind-worker2
crossplane	after-provider63943de-sync-1653010786-hrsv6	●	0/1	0	Completed	10.244.3.13	kind-worker2
crossplane	after-providera3a5d18-sync-1653923874-xvb7w	●	0/1	0	Completed	10.244.3.15	kind-worker2
crossplane	after-providerf6bc9f7-sync-1653011163-xb5zm	●	0/1	0	Completed	10.244.1.9	kind-worker3
crossplane	crossplane-6fdc95d9c4-mqn6d	●	1/1	1	Running	10.244.2.4	kind-worker
crossplane	crossplane-provider-aws-866abfb37fc-57b649b489-v2mx9	●	1/1	1	Running	10.244.3.6	kind-worker2
crossplane	crossplane-rbac-manager-6454b958cc-92z4k	●	1/1	1	Running	10.244.1.3	kind-worker3
default	echo-1	●	1/1	1	Running	10.244.3.8	kind-worker2
default	echo-2	●	1/1	1	Running	10.244.1.7	kind-worker3
default	test	●	1/1	3146	Running	10.244.2.6	kind-worker

I love UIs

Interfaces/UI/IDE

Kubernetes IDE: Lens



- Lens is the IDE for Kubernetes clusters
- Lets you inspect Kubernetes cluster and its applications.
- Standalone desktop app; no in-cluster components required
- Supports:
 - Connection to Remote clusters
 - Port-Forward
- Manages:
 - CustomResourceDefinitions (CRDs)
 - Helm chart deployment inspection

The screenshot shows the Lens application running on a laptop. On the left, a sidebar menu lists various Kubernetes resources: Workloads (Overview, Pods, Deployments, DaemonSets, StatefulSets, ReplicaSets, Jobs, CronJobs), Configuration, Network, Storage, Namespaces, Events, and Access Control. The main area displays a list of pods under the 'Pods' tab. A specific pod, 'calico-node-rwf4g', is selected, and its details are shown on the right side of the screen. The pod's status is 'Running' on node 'ip-192-168-81-7.eu-north'. The right panel also shows CPU and Memory usage over time, and a terminal window at the bottom.

Name	Namespace	Container
aws-node-hpz2f	kube-system	
aws-node-rgs	kube-system	
calico-node-rwf4g	kube-system	
calico-node-x8lr	kube-system	
calico-typha-78bb876fc47-sw5...	kube-system	
calico-typha-horizontal-autos...	kube-system	
cert-manager-789598c8d7-r14...	kube-system	
coredns-5b7d965bf9-8rq5q	kube-system	
dashboard-758bd48745-qpsft	kontena-lens	
helm-api-0	kontena-lens	
k8s-resource-applier-6d58457...	kontena-lens	
kube-proxy-v9x5g	kube-system	
kube-state-metrics-76ffbc879...	kontena-stats	
license-enforcer-5799c9c4c...	kontena-lens	
mariadb-1568285718-master-0	jakoleinm	
mariadb-1568285718-slave-0	jakoleinm	
metrics-server-6bd546f4cd-n...	kube-system	

<https://k8slens.dev/>

Interfaces: Octant

- Do-it-all UI
- Local “server” app that expose a local webserver for your browser
- Includes plugins to extend capabilities
- Not under active development anymore...



<https://reference.octant.dev/>

```
brew install octant
mkdir -p $HOME/.config/octant/plugins
```

```
octant
```

Interfaces: Octant

The screenshot shows the Octant interface with the following details:

Sidebar:

- Octant logo
- Filter by labels dropdown
- Applications (selected)
- Namespace Overview
- Workloads (selected)
- Discovery and Load Balancing
- Config and Storage
- Custom Resources
- RBAC
- Events
- Cluster Overview
- Kind
- Plugins

Overview Section:

Overview

Deployments

Name	Labels	Status	Age	Containers	Selector
multi-deployment	app:multi-deployment	1/1	14h	first second	app:multi-deployment
simple-deployment	app:simple-deployment	0/2	19h	alpine	app:simple-deployment

Items per page: 100 | 1 - 2 of 2 items

Pods

Name	Labels	Ready	Phase	Status	Restarts	Node	Age
multi-deployment-6f9fb8c49d-l4tzx	app:multi-deployment	2/2	Running	Running	0	demo-worker	14h
simple-deployment-5cff76596f-clgl7	app:simple-deployment	0/1	Running	CrashLoopBackOff	94	demo-worker	19h

Interfaces: KubeNav

- [Iphone](#) and [Android](#) app
- [Desktop app](#)



The screenshot shows the KubeNav application window. On the left is a sidebar menu with the following items:

- Overview
- Bookmarks
- Workloads
 - Cron Jobs
 - Daemon Sets
 - Deployments
 - Jobs
 - Pods
 - Replica Sets
 - Replication Controllers
 - Stateful Sets
- Discovery and Load Balancing
- Edgegate

The main area is titled "Overview" and contains three circular performance metrics:

- CPU:** Capacity: 468000m, Usage: 37545m, Request: 475733m, Limit: 942798m
- Memory:** Capacity: 1715006Mi, Usage: 291082Mi, Request: 644698Mi, Limit: 1260515Mi
- Pods:** Capacity: 14520, Usage: 2158

Below these metrics is a section titled "Warnings" which lists events from the Kubernetes event log:

NAME	NAMESPACE	LAST SEEN	REASON	MESSAGE
analytics-api-6bb6888b8c-7xtsh	analytics-api	1m	FailedScheduling	(combined from similar events): 0/134 nodes
analytics-api-6bb6888b8c-qwknl	analytics-api	2m	FailedScheduling	(combined from similar events): 0/134 nodes
analytics-api	analytics-api	2m	FailedGetResourceMetric	no recommendation

Working with YAML is so hard

VsCode Extensions

VsCode extensions

- [Kubernetes](#): Develop, deploy and debug Kubernetes applications
- [YAML](#): Language Support, with built-in Kubernetes syntax support
- [Indent-Rainbow](#): helper to better see Yaml indentations

```
---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:  
  name: external-dns  
rules:  
  - apiGroups:  
    - ""  
      resources:  
      - endpoints  
      - pods  
      - services  
      verbs:  
      - get  
      - watch  
      - list  
  - apiGroups:  
    - extensions
```

Day 2 Ops

Working with K8s Resources

- [KubePug](#) to ensure your cluster is not using deprecated resources

Verifies the current kubernetes cluster or input files, checking whether exists objects in this deprecated API Versions, allowing the user to check before migrating



\

I need for tooling for my advanced usage

Other tooling

Working with K8s Resources

- [Dive](#) to inspect the Docker Images
Ensure your Docker (container) images are not too big and does not contain unnecessary data
- [Dasel](#) to query and modify data structures using selector strings
Comparable to [jq](#) / [yq](#), but supports JSON, YAML, TOML, XML and CSV with zero runtime dependencies

Dasel to replace JQ

```
# Pretty Print JSON
echo '[ {"name": "Tom"}, {"name": "Paul"} ]' | dasel -p json
[
  {
    "name": "Tom"
  },
  {
    "name": "Paul"
  }
]
```

```
# JSON to YAML
echo '[ {"name": "Tom"}, {"name": "Paul"} ]' | dasel -r json -w yaml
- name: Tom
- name: Paul
```

Dasel for Kubernetes

```
# Select the image for a container named auth
dasel -p yaml select -f deployment.yaml -s
"spec.template.spec.containers.(name=manager).image"
alpine:latest

# Change the image for a container named auth
dasel put string -f deployment.yaml -s
"spec.template.spec.containers.(name=manager).image" "ubuntu:latest"

# Update replicas to 3
dasel put int -f deployment.yaml -s "spec.replicas" 3
```

Dasel for Kubernetes

```
# Add a new env var
dasel put object -f deployment.yaml -s
"spec.template.spec.containers.(name=manager).env.[]" -t string -t
string name=MY_NEW_ENV_VAR value=MY_NEW_VALUE

# Update an existing env var
dasel put string -f deployment.yaml -s
"spec.template.spec.containers.(name=manager).env.(name=MY_NEW_ENV_VAR)
.value" NEW_VALUE
```

References

- Install Kubectl: <https://kubernetes.io/docs/tasks/tools/>
- **K8s Shell**
 - Kubectl tuning: <https://suraj.io/post/being-productive-with-kubectl/>
 - Kubie: switch k8s context per shell <https://github.com/sbstp/kubie>
 - Krew: <https://github.com/kubernetes-sigs/krew>
 - Kubecolor: <https://github.com/kubecolor/kubecolor>
 - Stern: <https://github.com/stern/stern>
 - NerdFonts: <https://www.nerdfonts.com/>
 - OhMyZsh: <https://ohmyz.sh/>
 - Agnoster: <https://github.com/agnoster/agnoster-zsh-theme>
 - PowerLevel10k: <https://github.com/romkatv/powerlevel10k>
 - Node-shell: <https://github.com/kvaps/kubectl-node-shell>
- **K8s app Configuration Editing**
 - Kustomize: <https://kustomize.io/>
 - Kpt: <https://kpt.dev/>

References

- **Local K8s Clusters:**
 - Kind: <https://kind.sigs.k8s.io>
 - K3s: <https://k3s.io/>
 - Minikube: <https://minikube.sigs.k8s.io/docs/start/>
- **Kubernetes UIs:**
 - K9s: <https://github.com/derailed/k9s>
 - Mirantis Lens: <https://k8slens.dev/>
 - Infra App: <https://infra.app/>
 - Kubenav: <https://kubenav.io/>
 - Octant: <https://github.com/vmware-tanzu/octant>
- **Cool tools:**
 - KubePug: <https://github.com/rikatz/kubepug>
 - Dive: <https://github.com/wagoodman/dive>
 - Dasel: <https://github.com/TomWright/dasel>
- **Containerd ecosystem (Build, Run, Scan):**
 - Colima: <https://github.com/abiosoft/colima>
 - Podman: <https://podman.io/>



Epilogue

BUILDING FOR THE ROAD AHEAD

DETROIT 2022



Please scan the QR Code above
to leave feedback on this session



BUILDING FOR THE ROAD AHEAD

DETROIT 2022