UNIVERSITAS MERCU BUANA	**ANSWER SHEET** Semester : Odd / **Even** / Short*) Academic Year : 2020/2021	Q

Student Number	41518010056	Exam No.:	3	Student Signature [illegible]
Name	Yeppy Mangun Puspitajudin			
Faculty / Study Program	Computer Science / Informatics			Lecturer Signature
Course	INTRODUCTION TO CLOUD COMPUTING			
Lecturer	Arif Rifai Dwiyanto, ST, MTI, CISA			Score (00-100)
Time of Exam	Day	Date	Time	Room
	Thursday	08 July 2021	07:30-10:00	I-404

1. Cloud Computing Definition, Benefits, How it Works, and Examples

Definition

Cloud computing is the technological capability to use IT infrastructures and services that are not installed on a local computer or server. Using a network (Internet or Intranet), connections are made to external computers or servers that provide appropriate resources. These services are both provided and used exclusively via technical interfaces, protocols or browsers.

The applications are located outside of the central infrastructure – floating around the company, you could say – hence the cloud metaphor.

Benefits

- Reduced IT costs

Moving to cloud computing may reduce the cost of managing and maintaining your IT systems. Rather than purchasing expensive systems and equipment for your business, you can reduce your costs by using the resources of your cloud computing service provider.

- Scalability

Your business can scale up or scale down your operation and storage needs quickly to suit your situation, allowing flexibility as your needs change. Rather than purchasing and installing expensive upgrades yourself, your cloud computer service provider can handle this for you. Using the cloud frees up your time so you can get on with running your business.

- Business continuity

Protecting your data and systems is an important part of business continuity planning. Whether you experience a natural disaster, power failure or other crisis, having your data stored in the cloud ensures it is backed up and protected in a secure and safe location.

- Collaboration efficiency

Collaboration in a cloud environment gives your business the ability to communicate and share more easily outside of the traditional methods. If you are working on a project across different locations, you could use cloud computing to give employees, contractors and third parties access to the same files.

- Flexibility of work practices

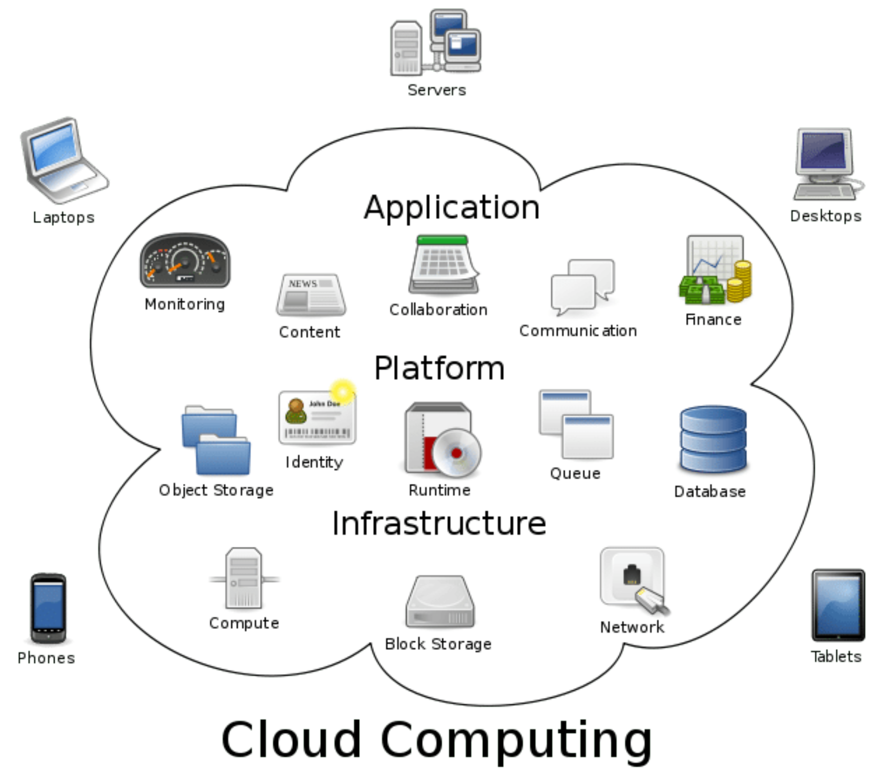
Cloud computing allows employees to be more flexible in their work practices. For example, you have the ability to access data from home, on holiday, or via the commute to and from work (providing you have an internet connection).

- Access to automatic updates

Access to automatic updates for your IT requirements may be included in your service fee. Depending on your cloud computing service provider, your system will regularly be updated with the latest technology.

How it works?

To understand the workings of a cloud system, it is easier to divide it into two sections: the front end and the back end. They are connected to each other through a network, usually the Internet. The front end is the side of the computer user or client. The back end is 'the cloud' section of the system.



The front end consists of the client's computer or computer network. Also the application essential to access the cloud system. It is not necessary that all cloud computing systems have the same user interface. On the back end of the cloud technology system, there are various computers, servers and data storage systems that make up the cloud. A cloud computing system could potentially include any computer program, from data processing to video games. Generally, each application will have its own dedicated server.

Examples

SaaS or Software as a Service. SaaS means instead of installing software on your computer, you access the platform online. Examples would include:

- com, which hosts sales information and databases online
- Square, which processes payments online
- Google Apps such as Google Drive or Calendar
- Slack, which allows collaboration and chat between other users

IaaS or Infrastructure as a Service. IaaS provides infrastructure components such as servers, storage, networking, security, and moreover the cloud. Examples would include:

- Dropbox, a file storage and sharing system
- Microsoft Azure, which offers backup and disaster recovery services, hosting, and more
- Rackspace, which offers data, security, and infrastructure services.

PaaS or Platform as a Service. PaaS provides computing platforms such as operating systems, programming language execution environments, databases, and web servers. Examples would include:

- Google App Engine and Heroku, which allow developers to develop and serve apps

Serverless Computing. Serverless computing (also called simply "Serverless") is simply using a server on the cloud. This offers more elasticity, easier maintenance, and is often more price effective than hosting servers on-site.

2. Problem in companies that can be solved by using Cloud Computing

Problem

I've heard that many start-up company, they faced the problem about maintaining security on their platform especially they had limited technical support (does not have proper IT people to handle).

Solution

So someone proposed a solution to try out about cloud computing. Which analysed can solve:

1. Limited Technical Support

   Outside the cloud, the organization is limited to whoever is working inside the office. In the case of an emergency, they either have to hope their local professionals can get the job done or hire a third-party company to help, which could be costly.

This risk is reduced in the cloud because we'll have the built-in support of experienced professionals, and we won't have to rely on anyone with minimal experience.

Moving to the cloud may seem complicated at first, but the transition can help mitigate a series of long-term problems. The use of public and hybrid cloud services is becoming the new norm.

2. Reduce Security Risk

   Cloud technology has advanced greatly and now it is actually more secure and reliable than traditional on premise solutions. In fact, 64 percent of enterprises report that the cloud is more secure than their previous legacy systems, and 90 percent of businesses in the USA are currently utilizing a (hybrid) cloud infrastructure.

   Many business owners who are accustomed to using local servers hesitate to transition to the cloud for fear of security risks. They worry that having their information "out there" on the cloud will make it more susceptible to hackers.

   As scary as these fears are, however, they are unlikely to happen. In fact, your data is just as secure in the cloud as it is in bare metal servers. Because cloud hosting has become so popular, it has quickly progressed to the advanced stages of security. In other words, because so many businesses are using cloud hosting in some form, it has been forced to maintain high levels of security to meet all the demand.

3. Benefit of Virtual Private Server (VPS).

1. Increase Website Reliability and Performance

On a shared server, the activities of other customers can have an impact on your website. If another site gets a sudden jump in traffic, for example, our site may experience performance issues.

Slow loading times are a problem, because they drive visitors away and can negatively impact our conversions. What's more, we may also have to contend with security issues, since a hacked website is a risk to all other sites on the same server.

On the other hand, a VPS provides an environment and resources that are specifically for our website and no one else. Traffic from other websites will no longer be your concern, and our site will be safely sectioned off in its own space. These factors can play a huge role in ensuring that our site provides a reliable experience for visitors.

2. Use the Your Resources as We See Fit

Thanks to the dedicated resources a VPS provides, we are free to use them however we need. This is another big distinction between this kind of plan and shared hosting, where we have little control over how resources are allocated.

When we sign up for a VPS, we will know exactly what resources we have, and be able to use them exclusively for content and manage them as needed. This also makes it a lot easier to run more than one website on the same plan.

Another related benefit is that we are not constrained by a single physical server. Therefore, it's easier to expand our resources along with our site's requirements.

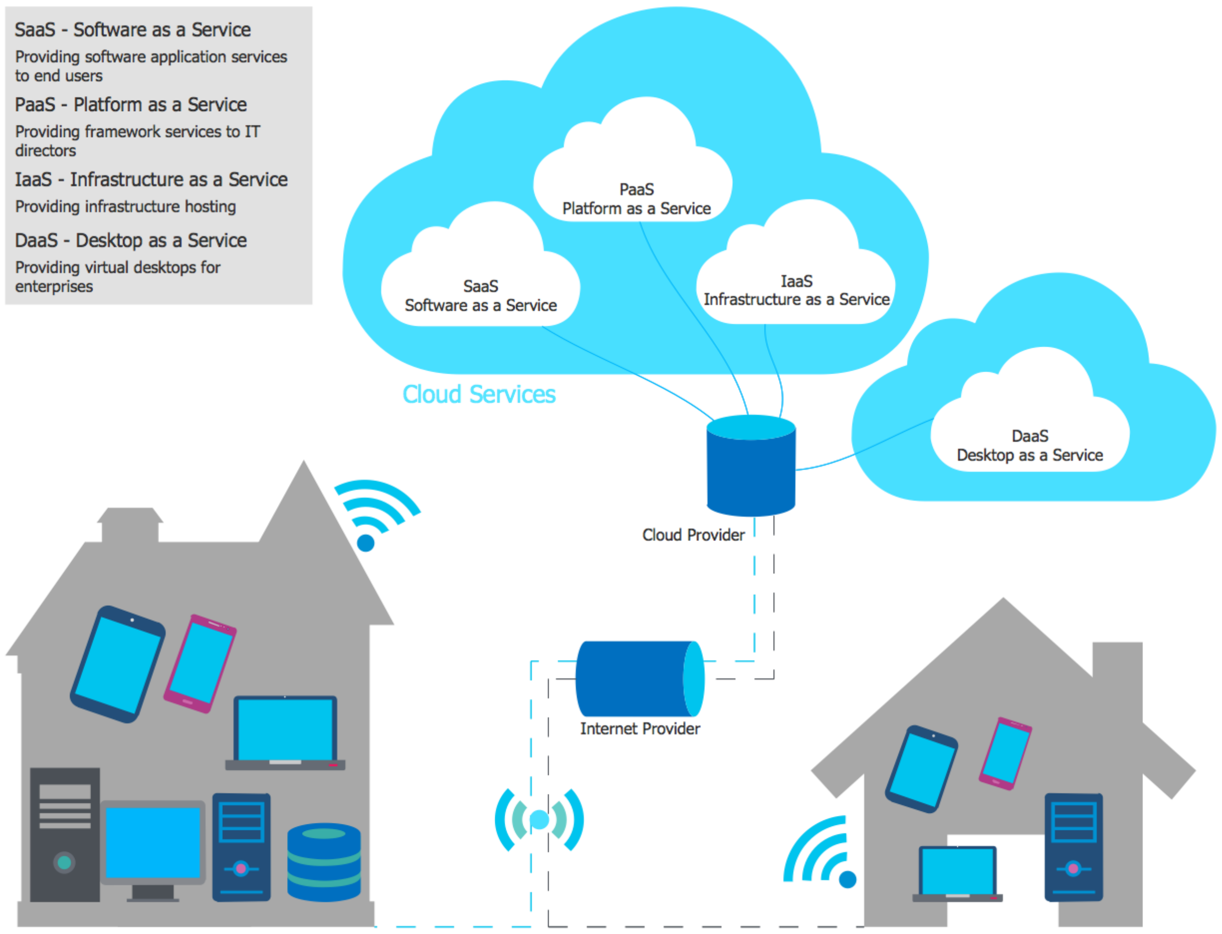
3. Install Only the OS and Software We Are Going to Use

Shared hosting offers very little choice when it comes to configuring the server. Since the web host manages it exclusively, they choose the applications and OS they want to use. In fact, we may not even know how the server is set up on this type of plan.

VPS plans, on the other hand, often provide choices depending on our needs. Many of them will give us various OS options, as well as an auto installer for common development applications. Some unmanaged plans even provide complete freedom over installation and administration.

All in all, a VPS is especially helpful for web developers. For example, we will be able to choose the version of PHP that we prefer, or employ node.js for our web development project. We can also uninstall the applications we don't needed, freeing up additional server resources.

4. Cloud Computing Architecture / Topology


SaaS - Software as a Service
Providing software application services to end users
PaaS - Platform as a Service
Providing framework services to IT directors
IaaS - Infrastructure as a Service
Providing infrastructure hosting
DaaS - Desktop as a Service
Providing virtual desktops for enterprises
PaaS
Platform as a Service
SaaS
Software as a Service
IaaS
Infrastructure as a Service
Cloud Services
DaaS
Desktop as a Service
Cloud Provider
Internet Provider

The diagrams and schemes above visualize the differences architectures types and how Cloud computing works. There is 4 architectures, include SaaS, PaaS, IaaS and DaaS which explained in above image. And each architectures can be used in office as company infrastructure, or in home as private server maybe.

And here are the example of GitHub, that use SaaS architecture.

App Server
Mobiles
SaaS
Networks
Database
PCs
Codes
Hebergementwebs.com

GitHub is a web-based version-control and collaboration platform for software developers. GitHub, which is delivered through a software-as-a-service (SaaS) business model, that provides the git version control software, along with space to store code, and social networking functionality.

5. Self-Hosted NextCloud using Kubernetes as SaaS Private Cloud.
   Here is the link of my works:
   https://youtu.be/DXtcSm4ZgYU

   Kubernetes (K8S) is an open-source system for automating deployment, scaling, and management of containerized applications. It is neither IaaS or PaaS, kubernetes is like a container orchestration engine which makes it more like a Container As A Service or CaaS. So, we will create our home-lab cloud in this system.

1. Installation

Before installing the Kubernetes, we have to install Kubectl and Minikube first. Kubectl is command-line tools to control the Kubernetes. It allows us to perform every possible

Kubernetes operation, on the other word, kubectl is a client for the Kubernetes API. Minikube is a utility you can use to run Kubernetes on our local machine. It creates a single node cluster contained in a virtual machine (VM). This cluster lets us do demo Kubernetes operations without requiring the time and resource-consuming installation of full-blown K8s.

a. Install kubectl

In my Mac, I use HomeBrew so the installation is using this command:

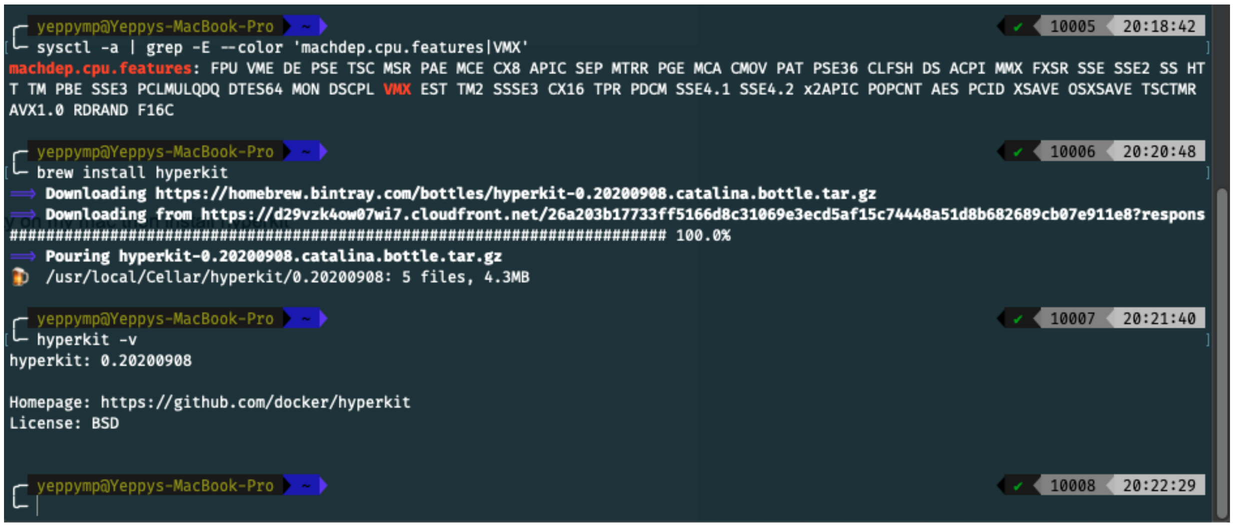
brew install kubectl

After installed, if we want to make sure check kubectl version by this command:

kubectl version --client

b. Install minikube

Each machine has different capabilities, especially virtualization process, to verify virtualization possibility on my mac, I will use hyperkit



```
┌ yeppymp@Yeppys-MacBook-Pro  ~                                              ✓  10005  20:18:42
└ sysctl -a | grep -E --color 'machdep.cpu.features|VMX'
machdep.cpu.features: FPU VME DE PSE TSC MSR PAE MCE CX8 APIC SEP MTRR PGE MCA CMOV PAT PSE36 CLFSH DS ACPI MMX FXSR SSE SSE2 SS HT
T TM PBE SSE3 PCLMULQDQ DTES64 MON DSCPL VMX EST TM2 SSSE3 CX16 TPR PDCM SSE4.1 SSE4.2 x2APIC POPCNT AES PCID XSAVE OSXSAVE TSCTMR
AVX1.0 RDRAND F16C

┌ yeppymp@Yeppys-MacBook-Pro  ~                                              ✓  10006  20:20:48
└ brew install hyperkit
==> Downloading https://homebrew.bintray.com/bottles/hyperkit-0.20200908.catalina.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/26a203b17733ff5166d8c31069e3ecd5af15c74448a51d8b682689cb07e911e8?respons
######################################################################## 100.0%
==> Pouring hyperkit-0.20200908.catalina.bottle.tar.gz
🍺  /usr/local/Cellar/hyperkit/0.20200908: 5 files, 4.3MB

┌ yeppymp@Yeppys-MacBook-Pro  ~                                              ✓  10007  20:21:40
└ hyperkit -v
hyperkit: 0.20200908

Homepage: https://github.com/docker/hyperkit
License: BSD


┌ yeppymp@Yeppys-MacBook-Pro  ~                                              ✓  10008  20:22:29
└
```

Then if it possible, then installing minikube:

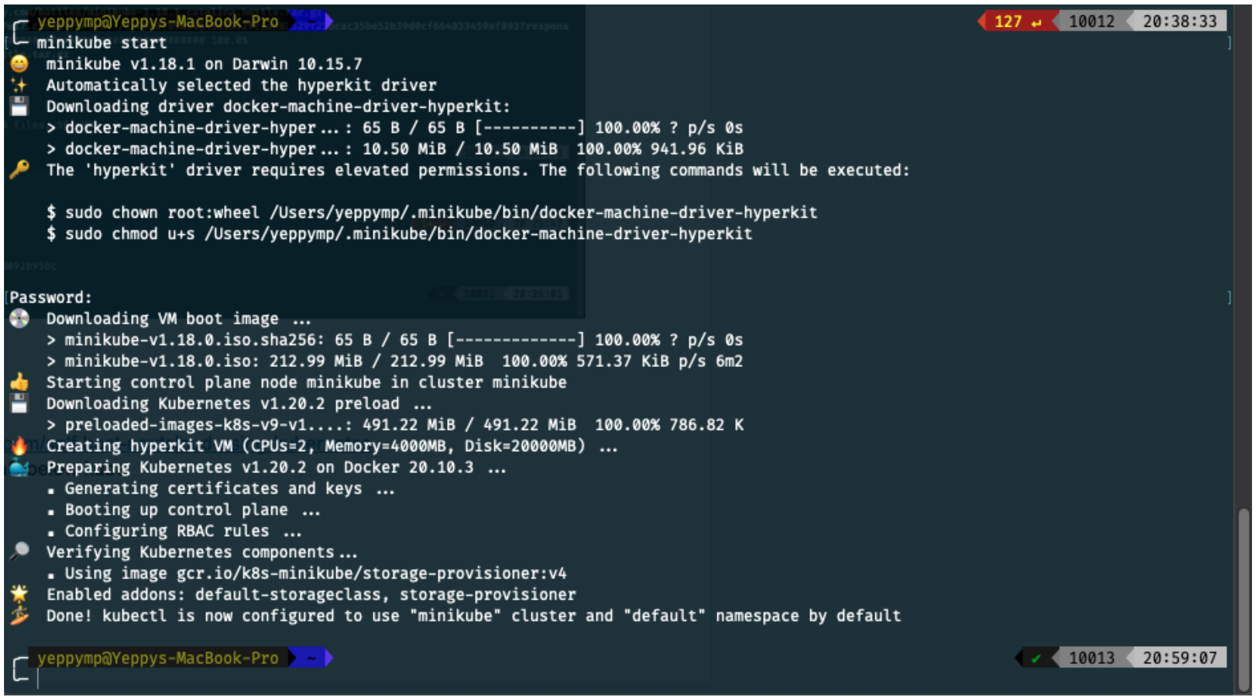
brew install minikube

2. Configuration

The next step is installing NextCloud as SaaS below kube

Starting the Minikube

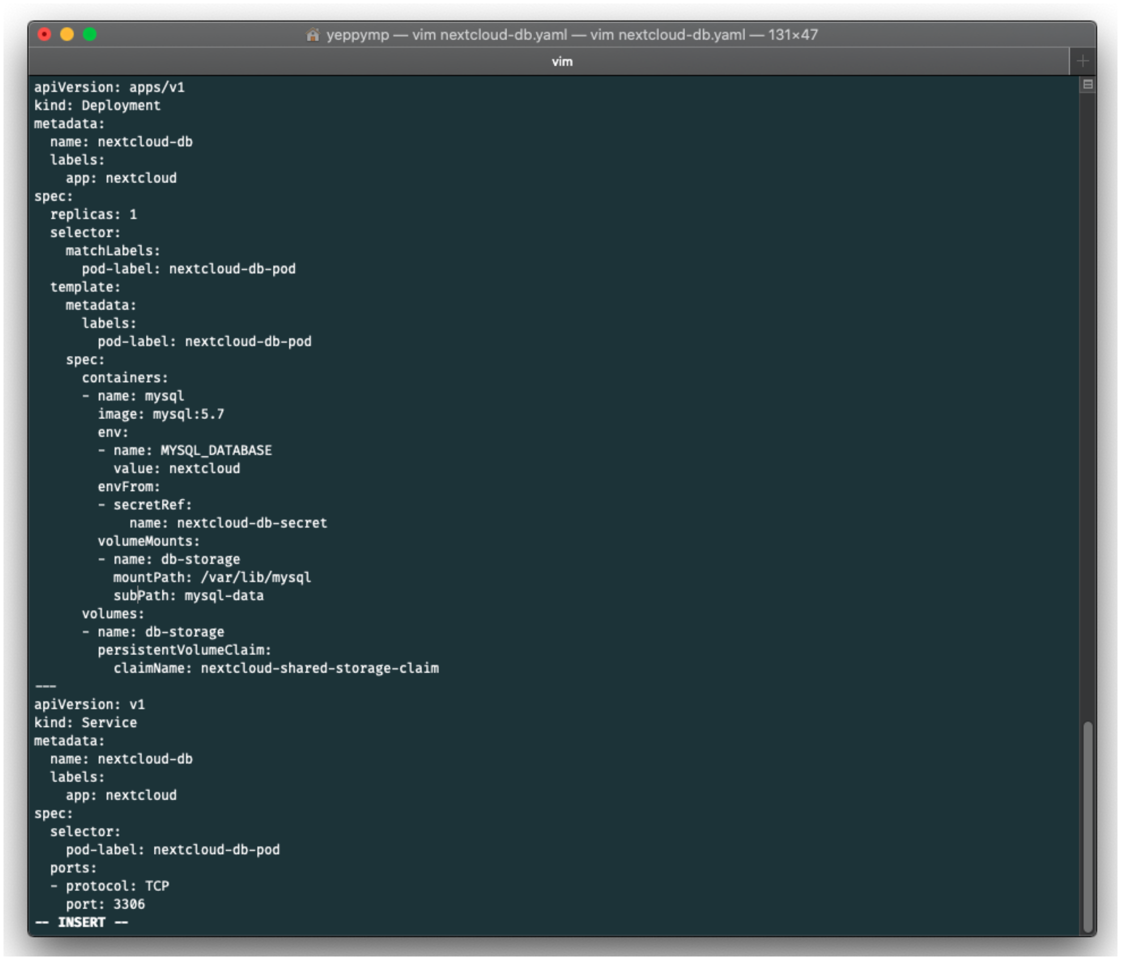
It takes a lot of time, because it has to download the minikube iso, and k8s image that almost takes 1GB.



```
yeppymp@Yeppys-MacBook-Pro  ~                                        127 ↵  10012  20:38:33
minikube start
😄  minikube v1.18.1 on Darwin 10.15.7
✨  Automatically selected the hyperkit driver
💾  Downloading driver docker-machine-driver-hyperkit:
    > docker-machine-driver-hyper...: 65 B / 65 B [----------] 100.00% ? p/s 0s
    > docker-machine-driver-hyper...: 10.50 MiB / 10.50 MiB  100.00% 941.96 KiB
🔑  The 'hyperkit' driver requires elevated permissions. The following commands will be executed:

    $ sudo chown root:wheel /Users/yeppymp/.minikube/bin/docker-machine-driver-hyperkit
    $ sudo chmod u+s /Users/yeppymp/.minikube/bin/docker-machine-driver-hyperkit


Password:
💿  Downloading VM boot image ...
    > minikube-v1.18.0.iso.sha256: 65 B / 65 B [-------------] 100.00% ? p/s 0s
    > minikube-v1.18.0.iso: 212.99 MiB / 212.99 MiB  100.00% 571.37 KiB p/s 6m2
👍  Starting control plane node minikube in cluster minikube
💾  Downloading Kubernetes v1.20.2 preload ...
    > preloaded-images-k8s-v9-v1....: 491.22 MiB / 491.22 MiB  100.00% 786.82 K
🔥  Creating hyperkit VM (CPUs=2, Memory=4000MB, Disk=20000MB) ...
🐳  Preparing Kubernetes v1.20.2 on Docker 20.10.3 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v4
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

yeppymp@Yeppys-MacBook-Pro  ~                                          ✓  10013  20:59:07
```

Setting the NextCloud

NextCloud has two main components: a server container and a database container. First, we will start by going over the database:

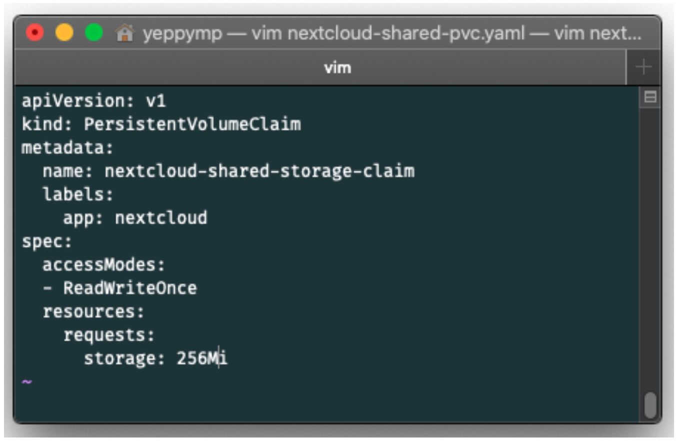


yeppymp — vim nextcloud-db.yaml — vim nextcloud-db.yaml — 131×47

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nextcloud-db
  labels:
    app: nextcloud
spec:
  replicas: 1
  selector:
    matchLabels:
      pod-label: nextcloud-db-pod
  template:
    metadata:
      labels:
        pod-label: nextcloud-db-pod
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        env:
        - name: MYSQL_DATABASE
          value: nextcloud
        envFrom:
        - secretRef:
            name: nextcloud-db-secret
        volumeMounts:
        - name: db-storage
          mountPath: /var/lib/mysql
          subPath: mysql-data
      volumes:
      - name: db-storage
        persistentVolumeClaim:
          claimName: nextcloud-shared-storage-claim
---
apiVersion: v1
kind: Service
metadata:
  name: nextcloud-db
  labels:
    app: nextcloud
spec:
  selector:
    pod-label: nextcloud-db-pod
  ports:
  - protocol: TCP
    port: 3306
-- INSERT --
```

Here, I set some of the values, first is name our Kubernetes deployment, I name it nextcloud-db. The purpose of this is to make sure that the database will run all the time. Next is the replicas, it is used when the node gets some error and cannot be used, the deployment will create the replacement. There is also in spec: containers, it defines the mysql version that I will use.

Kube volumes

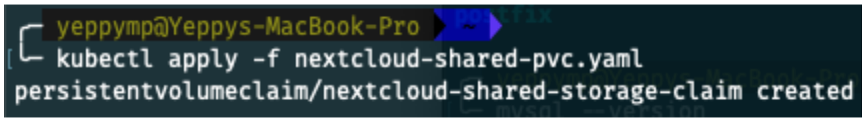


```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nextcloud-shared-storage-claim
  labels:
    app: nextcloud
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 256Mi
~
```

The volumes are used when the container terminates, because when it happens, everything that is stored in the container will be gone. The volumes are designed for durable storage to containers.

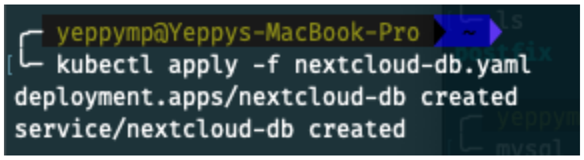
Create MySQL Database Component

First, I have to create/apply the persistent volume claim using this command below:



```
┌ yeppymp@Yeppys-MacBook-Pro  ~
└ kubectl apply -f nextcloud-shared-pvc.yaml
persistentvolumeclaim/nextcloud-shared-storage-claim created
```

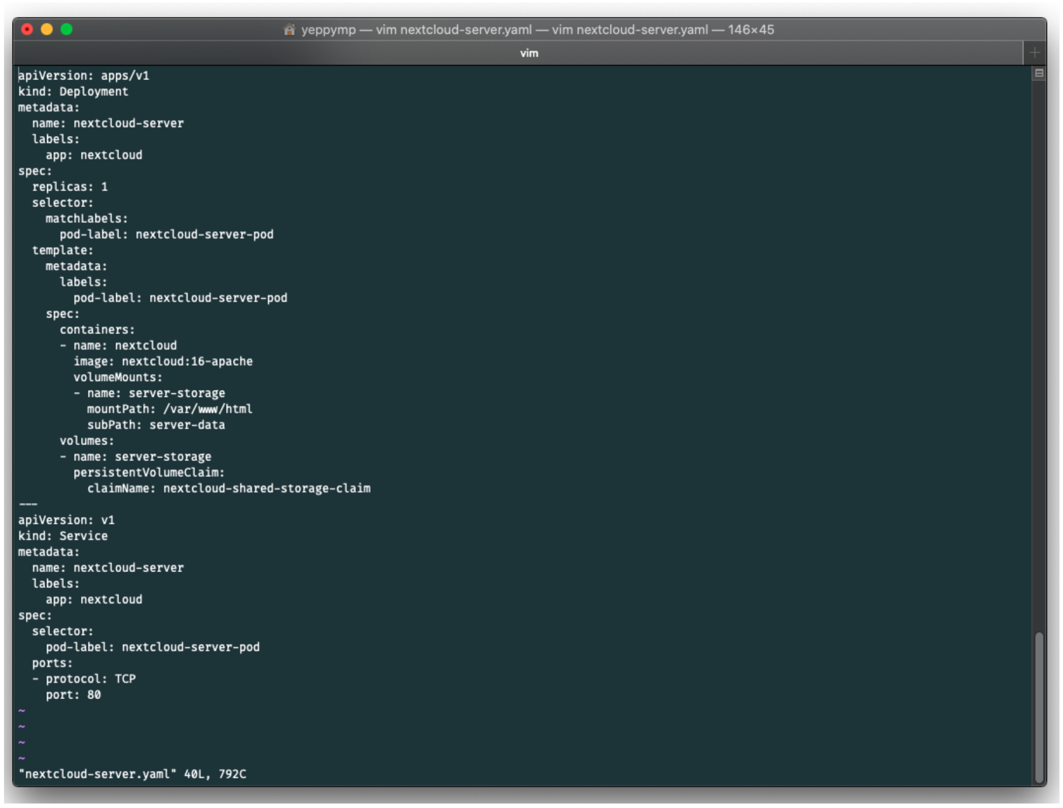
After that, create/apply the Database deployment and service.



```
┌ yeppymp@Yeppys-MacBook-Pro  ~
└ kubectl apply -f nextcloud-db.yaml
deployment.apps/nextcloud-db created
service/nextcloud-db created
```

NextCloud Server Deployment

After the database is running, the next step is to create the Kubernetes config files for the nextcloud server components.

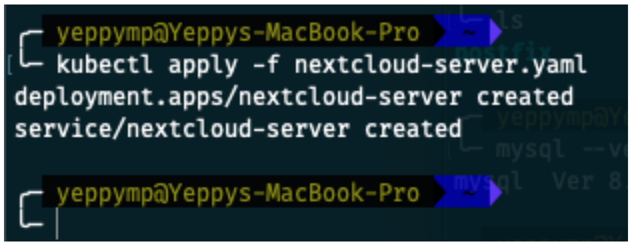


yeppymp — vim nextcloud-server.yaml — vim nextcloud-server.yaml — 146×45
vim

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nextcloud-server
  labels:
    app: nextcloud
spec:
  replicas: 1
  selector:
    matchLabels:
      pod-label: nextcloud-server-pod
  template:
    metadata:
      labels:
        pod-label: nextcloud-server-pod
    spec:
      containers:
      - name: nextcloud
        image: nextcloud:16-apache
        volumeMounts:
        - name: server-storage
          mountPath: /var/www/html
          subPath: server-data
      volumes:
      - name: server-storage
        persistentVolumeClaim:
          claimName: nextcloud-shared-storage-claim
---
apiVersion: v1
kind: Service
metadata:
  name: nextcloud-server
  labels:
    app: nextcloud
spec:
  selector:
    pod-label: nextcloud-server-pod
  ports:
  - protocol: TCP
    port: 80
~
~
~
~
"nextcloud-server.yaml" 40L, 792C
```

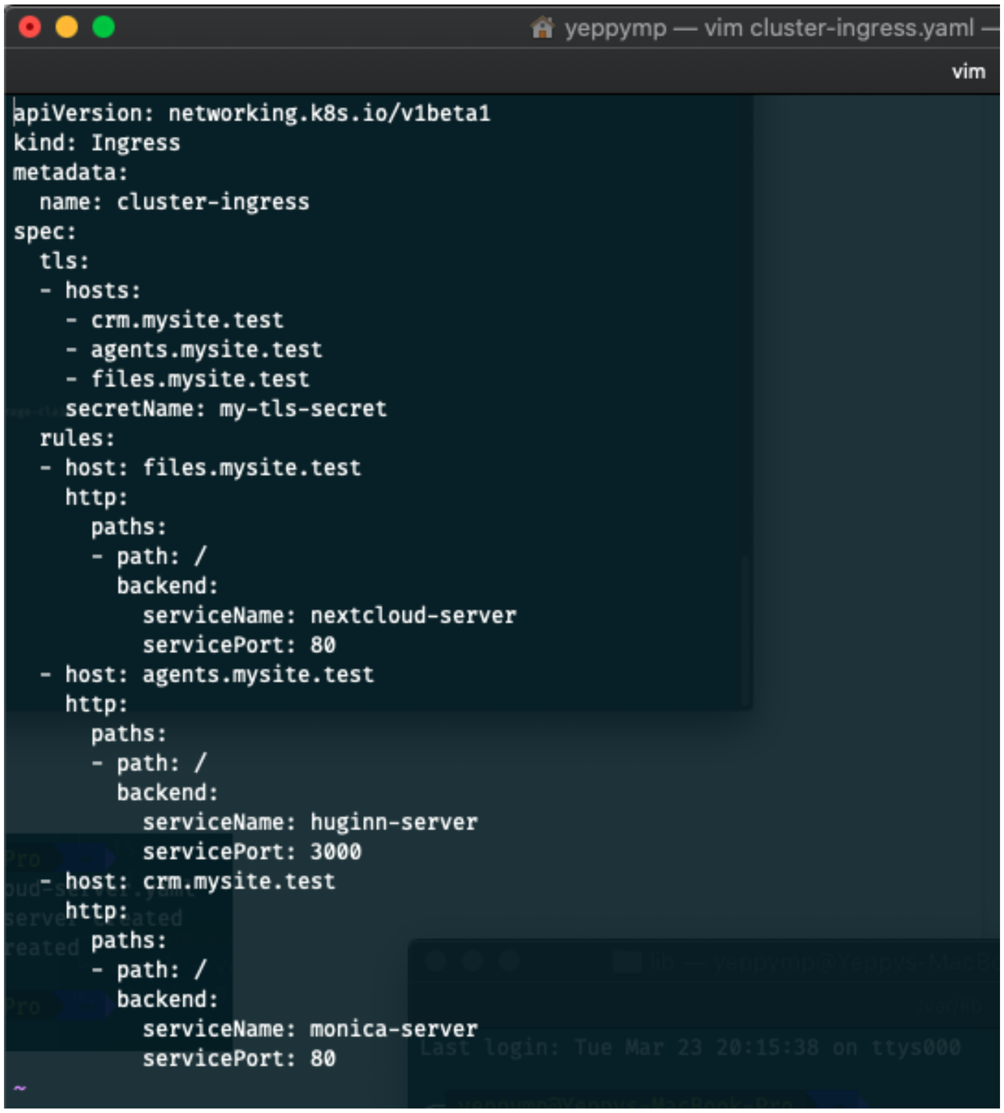
Create/Apply NextCloud Server

To create the NextCloud server development and service, run this command in the same directory as before.



```
┌ yeppymp@Yeppys-MacBook-Pro  ~
└ kubectl apply -f nextcloud-server.yaml
deployment.apps/nextcloud-server created
service/nextcloud-server created

┌ yeppymp@Yeppys-MacBook-Pro  ~
└
```

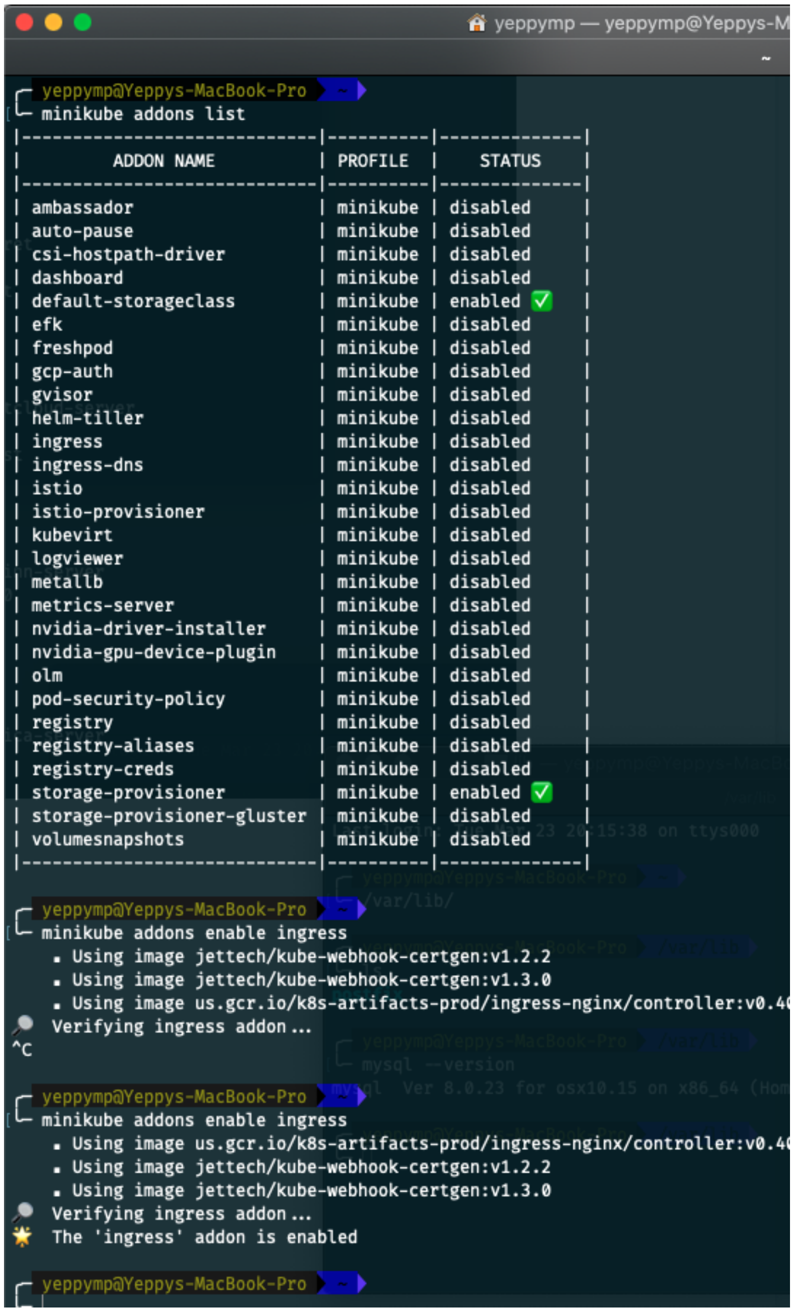
Kubernetes ingress



```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: cluster-ingress
spec:
  tls:
  - hosts:
    - crm.mysite.test
    - agents.mysite.test
    - files.mysite.test
    secretName: my-tls-secret
  rules:
  - host: files.mysite.test
    http:
      paths:
      - path: /
        backend:
          serviceName: nextcloud-server
          servicePort: 80
  - host: agents.mysite.test
    http:
      paths:
      - path: /
        backend:
          serviceName: huginn-server
          servicePort: 3000
  - host: crm.mysite.test
    http:
      paths:
      - path: /
        backend:
          serviceName: monica-server
          servicePort: 80
~
```

What I am going to do right now is configure the database-related environment variables on the NextCloud web page. First thing that we have to do is make the NextCloud server browser-accessible. By default, clients outside of the cluster cannot communicate with pods inside of the cluster. The way that we are going to solve this is by creating a Kubernetes Ingress object. It will allow external clients to connect with the NextCloud server in our cluster.

Enable Ingress Addon

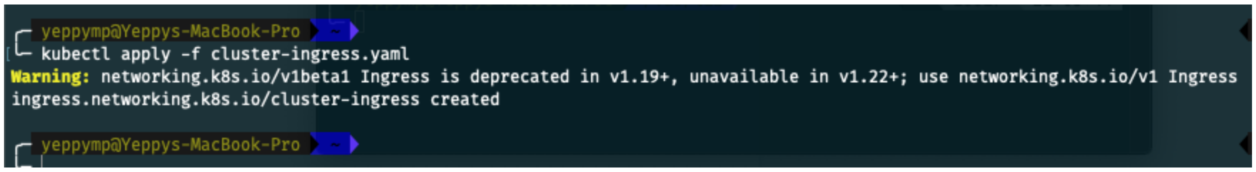


```
yeppymp@Yeppys-MacBook-Pro
└─ minikube addons list
|-----------------------------|----------|--------------|
|         ADDON NAME          | PROFILE  |    STATUS    |
|-----------------------------|----------|--------------|
| ambassador                  | minikube | disabled     |
| auto-pause                  | minikube | disabled     |
| csi-hostpath-driver         | minikube | disabled     |
| dashboard                   | minikube | disabled     |
| default-storageclass        | minikube | enabled ✅   |
| efk                         | minikube | disabled     |
| freshpod                    | minikube | disabled     |
| gcp-auth                    | minikube | disabled     |
| gvisor                      | minikube | disabled     |
| helm-tiller                 | minikube | disabled     |
| ingress                     | minikube | disabled     |
| ingress-dns                 | minikube | disabled     |
| istio                       | minikube | disabled     |
| istio-provisioner           | minikube | disabled     |
| kubevirt                    | minikube | disabled     |
| logviewer                   | minikube | disabled     |
| metallb                     | minikube | disabled     |
| metrics-server              | minikube | disabled     |
| nvidia-driver-installer     | minikube | disabled     |
| nvidia-gpu-device-plugin    | minikube | disabled     |
| olm                         | minikube | disabled     |
| pod-security-policy         | minikube | disabled     |
| registry                    | minikube | disabled     |
| registry-aliases            | minikube | disabled     |
| registry-creds              | minikube | disabled     |
| storage-provisioner         | minikube | enabled ✅   |
| storage-provisioner-gluster | minikube | disabled     |
| volumesnapshots             | minikube | disabled     |
|-----------------------------|----------|--------------|

yeppymp@Yeppys-MacBook-Pro
└─ minikube addons enable ingress
    ▪ Using image jettech/kube-webhook-certgen:v1.2.2
    ▪ Using image jettech/kube-webhook-certgen:v1.3.0
    ▪ Using image us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller:v0.4
🔎  Verifying ingress addon ...
^C

yeppymp@Yeppys-MacBook-Pro
└─ minikube addons enable ingress
    ▪ Using image us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller:v0.4
    ▪ Using image jettech/kube-webhook-certgen:v1.2.2
    ▪ Using image jettech/kube-webhook-certgen:v1.3.0
🔎  Verifying ingress addon ...
🌟  The 'ingress' addon is enabled

yeppymp@Yeppys-MacBook-Pro
```

For creating the ingress object, I have to enable the ingress on the cluster, because the ingress is disable by default. For doing it, I use the commands above.

Create ingress object

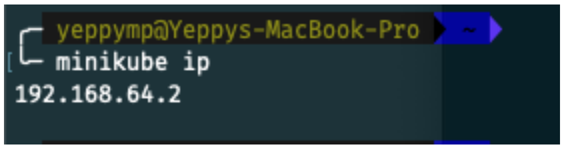
To create/apply the ingress object, use same command as before



```
yeppymp@Yeppys-MacBook-Pro
└─ kubectl apply -f cluster-ingress.yaml
Warning: networking.k8s.io/v1beta1 Ingress is deprecated in v1.19+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
ingress.networking.k8s.io/cluster-ingress created

yeppymp@Yeppys-MacBook-Pro
```

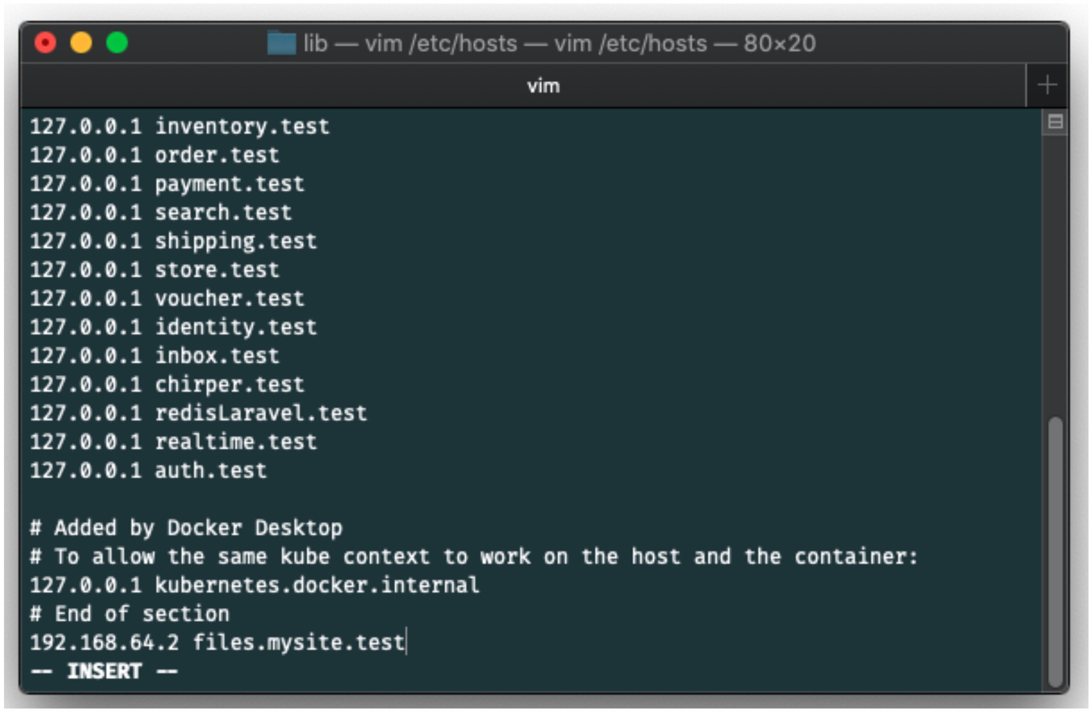
Update host file

I have to map the IP address to my minikube node. Because when I did not do that, it would show a 404 error because there is nothing there.



```
┌ yeppymp@Yeppys-MacBook-Pro  ~
└ minikube ip
192.168.64.2
```

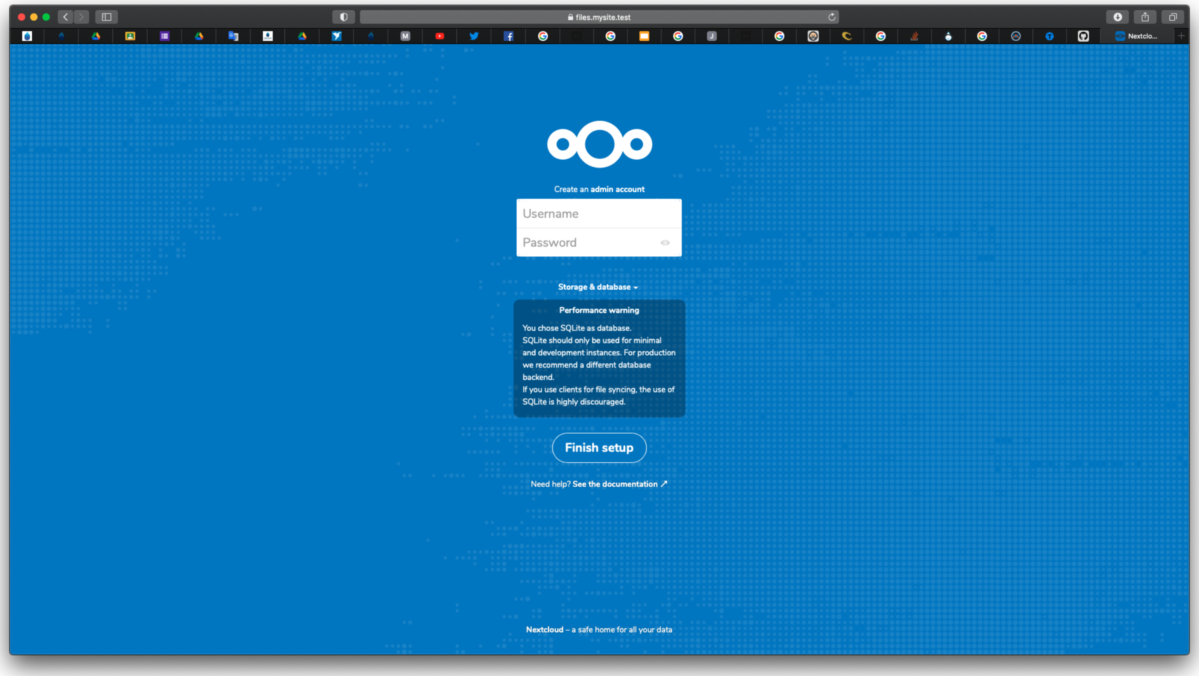
Editing host and add kube ip to host files.mysite.test



lib — vim /etc/hosts — vim /etc/hosts — 80×20

vim

```
127.0.0.1 inventory.test
127.0.0.1 order.test
127.0.0.1 payment.test
127.0.0.1 search.test
127.0.0.1 shipping.test
127.0.0.1 store.test
127.0.0.1 voucher.test
127.0.0.1 identity.test
127.0.0.1 inbox.test
127.0.0.1 chirper.test
127.0.0.1 redisLaravel.test
127.0.0.1 realtime.test
127.0.0.1 auth.test

# Added by Docker Desktop
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
192.168.64.2 files.mysite.test
-- INSERT --
```



Performance warning

You chose SQLite as database.
SQLite should only be used for minimal and development instances. For production we recommend a different database backend.
If you use clients for file syncing, the use of SQLite is highly discouraged.

Now nextcloud can be accessed on my local connection (Private) using files.mysite.test. Above is when I access it on my laptop.

3. Conclusion

So that is how to setup self-hosted NextCloud under kubernetes, nextcloud as SaaS, and kubernetes either as PaaS or LaaS. Well now I have a private cloud on my laptop, this report just until kubernetes & nextcloud are ready to use, for application deployment we need some more steps, which configuring database, storage, domains etc. It is like some basic setups in hosting while we are going to deploy our application.