Projet English

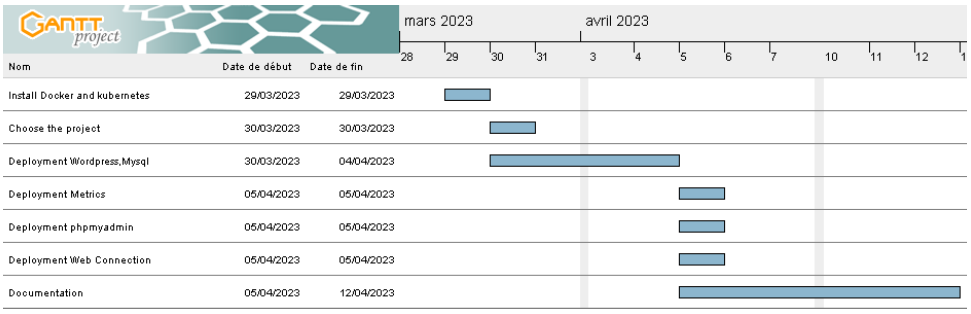
Sommaire

Context

The final mission is developing a project that implements all of the concepts in this course by group of 2 people. We have a specifications to realise the project :

- Use kubernetes
- Containerize an application
- Provide a deployment manual with all necessary files
- Provide a report of the project

With all this information, we had to think about an application that would meet these expectations, so we decided to focus on Wordpress. We wanted to approach an application that could be used in a business context. We have set up a schedule to meet the deadlines by April 13th. Here is the schedule that we have set up:

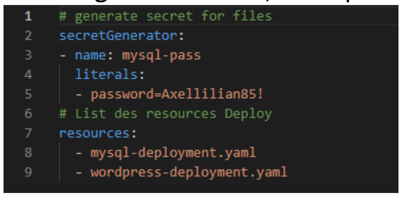


Nom	Date de début	Date de fin
Install Docker and kubernetes	29/03/2023	29/03/2023
Choose the project	30/03/2023	30/03/2023
Deployment Wordpress,Mysql	30/03/2023	04/04/2023
Deployment Metrics	05/04/2023	05/04/2023
Deployment phpmyadmin	05/04/2023	05/04/2023
Deployment Web Connection	05/04/2023	05/04/2023
Documentation	05/04/2023	12/04/2023

To have complete management of Wordpress, we added metrics to the Kubernetes server, installed the PhpMyAdmin application to manage databases. Additionally, we added load management for Wordpress to connection overload.

Wordpress

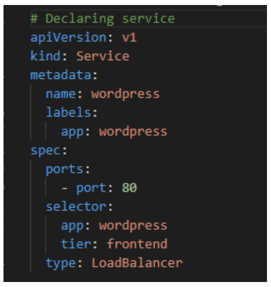
WordPress is a free, open-source content management system used to create websites and blogs. It is one of the most popular platforms in the world and is known for its flexibility, ease of use, and large community of users and developers who contribute to its ongoing development and improvement. With thousands of customizable themes and plugins, WordPress can be used to build a wide range of websites, from personal blogs to complex e-commerce sites.



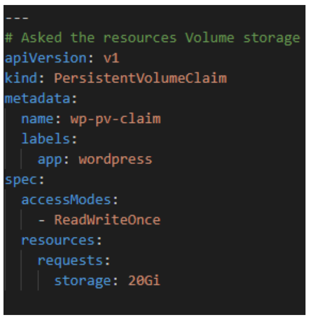
```yaml
# generate secret for files
secretGenerator:
- name: mysql-pass
  literals:
  - password=Axellilian85!
# List des resources Deploy
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

This block is to generate the secret for files and declaring the resources for the file. File kustomization.yaml

This block create the service of wordpress. This file is wordpress-deployment.yaml



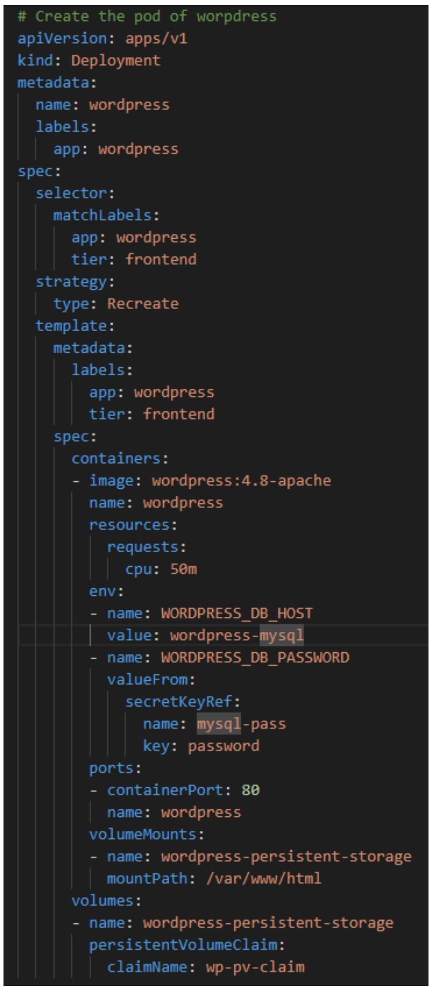
```yaml
# Declaring service
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
```



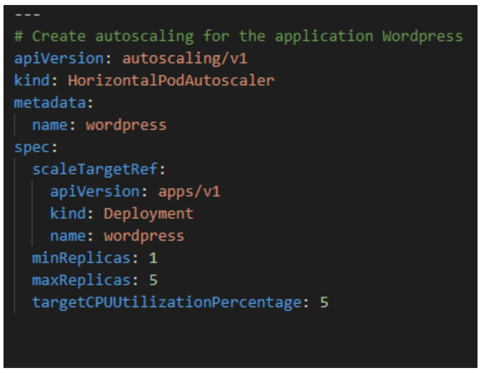
```yaml
---
# Asked the resources Volume storage
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

This block asked the resource the volume storage.

This block creates the pod of WordPress.



```yaml
# Create the pod of worpdress
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.8-apache
        name: wordpress
        resources:
          requests:
            cpu: 50m
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```

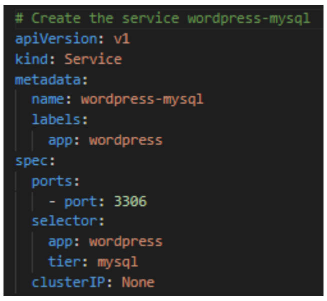


```
---
# Create autoscaling for the application Wordpress
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: wordpress
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: wordpress
  minReplicas: 1
  maxReplicas: 5
  targetCPUUtilizationPercentage: 5
```

Autoscalling allows you to create weights according to the required performance. It is in the file wordpress-deployment.

Mysql

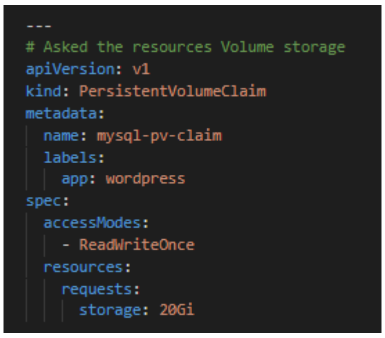
MySQL is a popular open-source relational database management system (RDBMS) that is used to store, manage, and retrieve data. It is commonly used in web applications to store user data, product information, and other types of data.



```
# Create the service wordpress-mysql
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
```

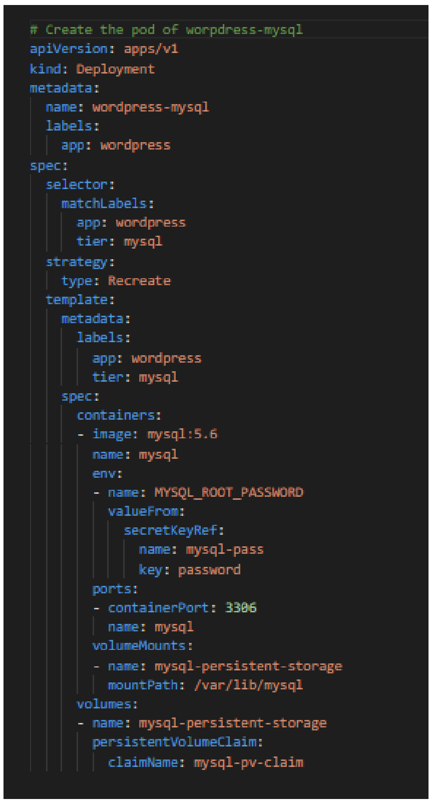
This block creates the service wordpress-mysql.

It is in the file: mysql-deployment.yaml



```
---
# Asked the resources Volume storage
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

This block asked the resource volume storage.

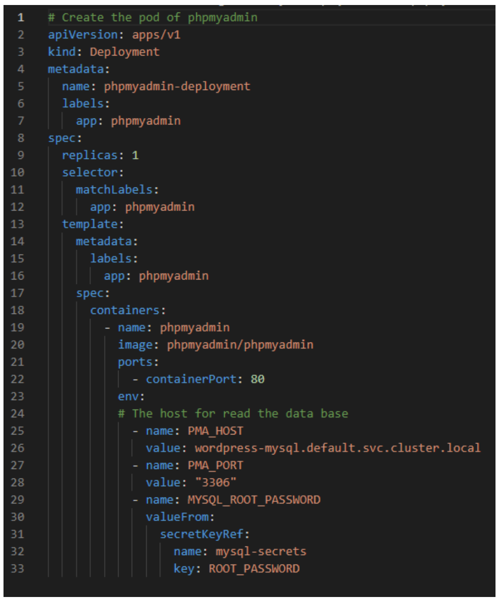


```
# Create the pod of worpdress-mysql
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```

This block creates the pod of wordpress-mysql.

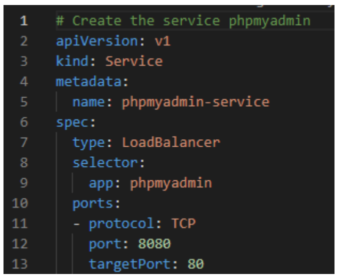
phpMyAdmin

PhpMyAdmin is an open-source web application used for managing MySQL databases. It provides a user-friendly graphical interface that allows users to manage their databases without having to write SQL code. We have added this application because I would access of the database.



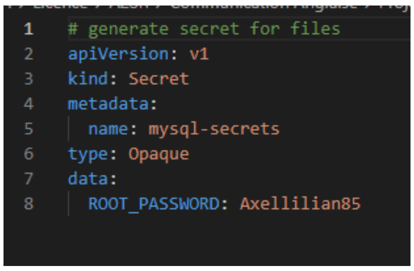
```yaml
# Create the pod of phpmyadmin
apiVersion: apps/v1
kind: Deployment
metadata:
  name: phpmyadmin-deployment
  labels:
    app: phpmyadmin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: phpmyadmin
  template:
    metadata:
      labels:
        app: phpmyadmin
    spec:
      containers:
        - name: phpmyadmin
          image: phpmyadmin/phpmyadmin
          ports:
            - containerPort: 80
          env:
          # The host for read the data base
            - name: PMA_HOST
              value: wordpress-mysql.default.svc.cluster.local
            - name: PMA_PORT
              value: "3306"
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-secrets
                  key: ROOT_PASSWORD
```

This block creates the pod of phpMyAdmin.



```yaml
# Create the service phpmyadmin
apiVersion: v1
kind: Service
metadata:
  name: phpmyadmin-service
spec:
  type: LoadBalancer
  selector:
    app: phpmyadmin
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 80
```

This block creates the service of phpmyadmin.

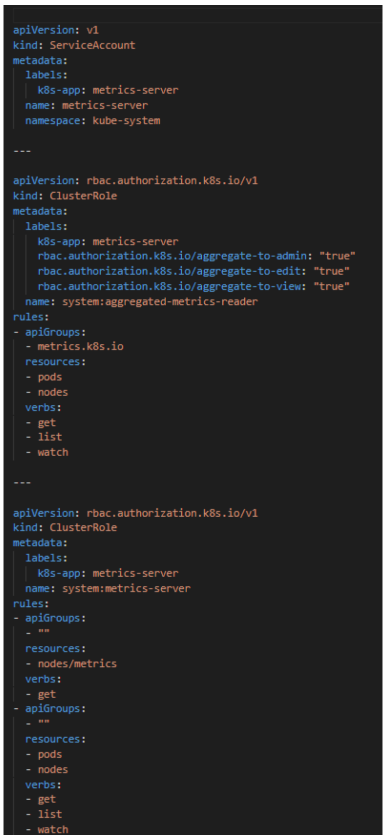


```yaml
# generate secret for files
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secrets
type: Opaque
data:
  ROOT_PASSWORD: Axellilian85
```

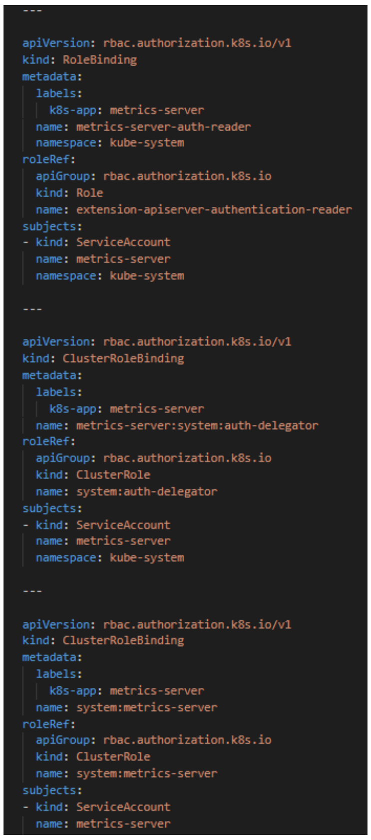
This file creates the secret for the configuration of phpMyAdmin.

Metric

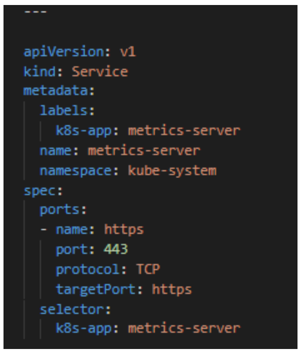
Metrics Server is a Kubernetes component that collects and stores performance metrics from the nodes and pods of a Kubernetes cluster. It is used to monitor resource usage in the cluster and to help administrators make informed decisions about the capacity and efficiency of the cluster.



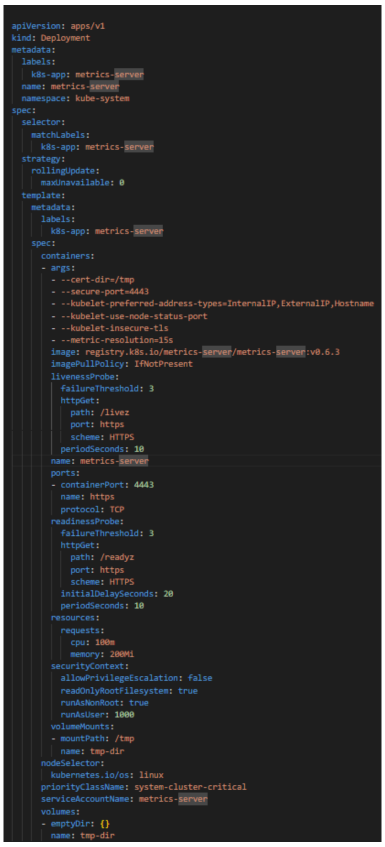
```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server
  namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    k8s-app: metrics-server
    rbac.authorization.k8s.io/aggregate-to-admin: "true"
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
    rbac.authorization.k8s.io/aggregate-to-view: "true"
  name: system:aggregated-metrics-reader
rules:
- apiGroups:
  - metrics.k8s.io
  resources:
  - pods
  - nodes
  verbs:
  - get
  - list
  - watch

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    k8s-app: metrics-server
  name: system:metrics-server
rules:
- apiGroups:
  - ""
  resources:
  - nodes/metrics
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
  verbs:
  - get
  - list
  - watch
```



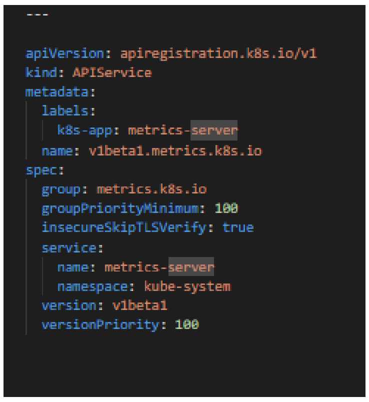
```yaml
---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server-auth-reader
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: extension-apiserver-authentication-reader
subjects:
- kind: ServiceAccount
  name: metrics-server
  namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server:system:auth-delegator
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
- kind: ServiceAccount
  name: metrics-server
  namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    k8s-app: metrics-server
  name: system:metrics-server
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:metrics-server
subjects:
- kind: ServiceAccount
  name: metrics-server
```



```yaml
---

apiVersion: v1
kind: Service
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server
  namespace: kube-system
spec:
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: https
  selector:
    k8s-app: metrics-server
```



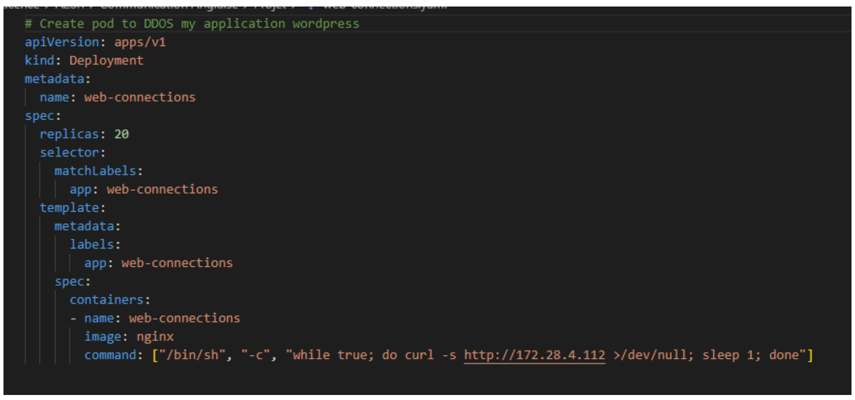
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server
  namespace: kube-system
spec:
  selector:
    matchLabels:
      k8s-app: metrics-server
  strategy:
    rollingUpdate:
      maxUnavailable: 0
  template:
    metadata:
      labels:
        k8s-app: metrics-server
    spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --kubelet-use-node-status-port
        - --kubelet-insecure-tls
        - --metric-resolution=15s
        image: registry.k8s.io/metrics-server/metrics-server:v0.6.3
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /livez
            port: https
            scheme: HTTPS
          periodSeconds: 10
        name: metrics-server
        ports:
        - containerPort: 4443
          name: https
          protocol: TCP
        readinessProbe:
          failureThreshold: 3
          httpGet:
            path: /readyz
            port: https
            scheme: HTTPS
          initialDelaySeconds: 20
          periodSeconds: 10
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
        securityContext:
          allowPrivilegeEscalation: false
          readOnlyRootFilesystem: true
          runAsNonRoot: true
          runAsUser: 1000
        volumeMounts:
        - mountPath: /tmp
          name: tmp-dir
      nodeSelector:
        kubernetes.io/os: linux
      priorityClassName: system-cluster-critical
      serviceAccountName: metrics-server
      volumes:
      - emptyDir: {}
        name: tmp-dir
```



```yaml
---

apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    k8s-app: metrics-server
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: metrics-server
    namespace: kube-system
  version: v1beta1
  versionPriority: 100
```

DDOS

It is to test the autoscaling of the WordPress. To confirm the solution of autoscaling.

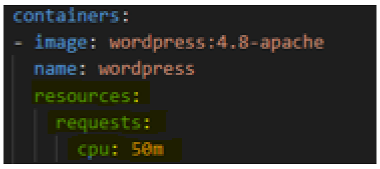


```yaml
# Create pod to DDOS my application wordpress
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-connections
spec:
  replicas: 20
  selector:
    matchLabels:
      app: web-connections
  template:
    metadata:
      labels:
        app: web-connections
    spec:
      containers:
      - name: web-connections
        image: nginx
        command: ["/bin/sh", "-c", "while true; do curl -s http://172.28.4.112 >/dev/null; sleep 1; done"]
```

ISSUES

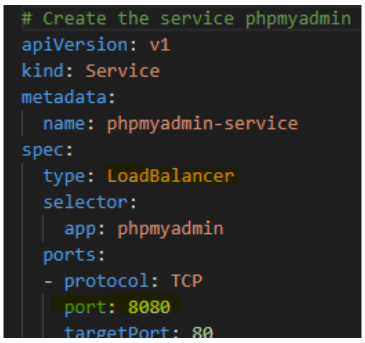
I have issues like:

- To add the autoscaling, I have an error code to the deployment because we don't have the request resource so I add three line to declared the request resource:



```yaml
containers:
- image: wordpress:4.8-apache
  name: wordpress
  resources:
    requests:
      cpu: 50m
```

- To add the metric server, I have added one line: --kubelet-insecure-tls- Do not verify the CA of serving certificates presented by Kubelets.
- For phpMyAdmin, I have modified two lines because we don't acces in website so I change the port and the access

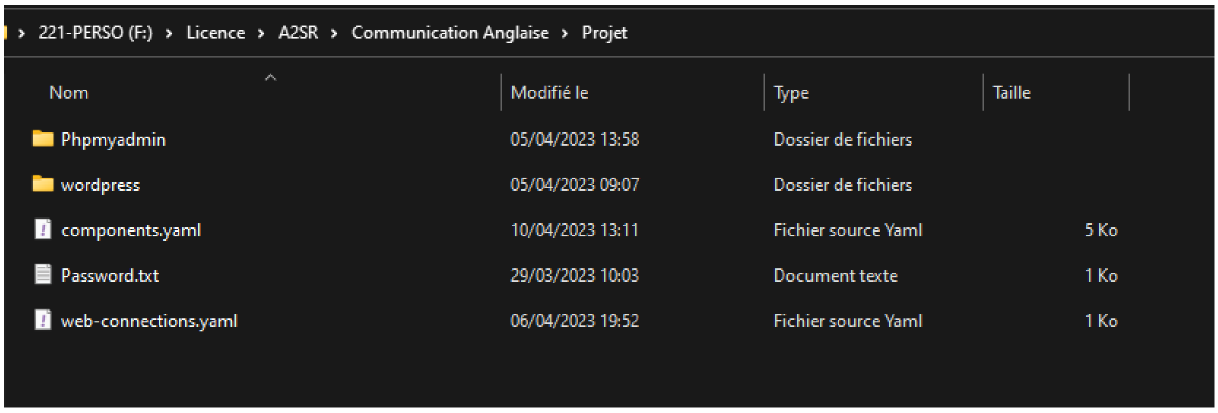


```yaml
# Create the service phpmyadmin
apiVersion: v1
kind: Service
metadata:
  name: phpmyadmin-service
spec:
  type: LoadBalancer
  selector:
    app: phpmyadmin
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 80
```

- The autoscaling don't work because the resource was good so I create a service DDOS to test the autoscaling

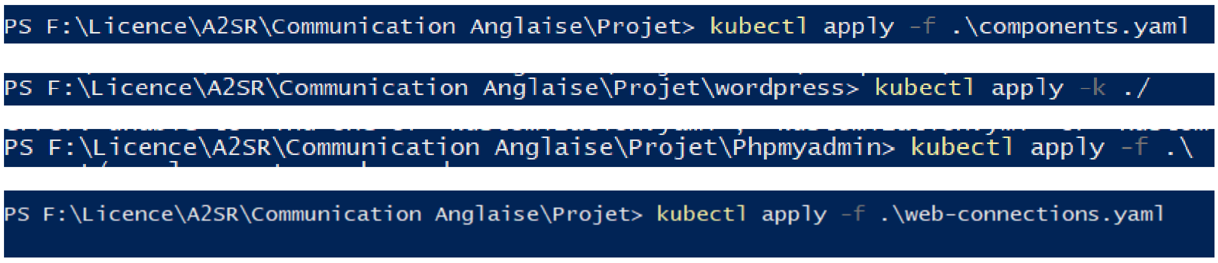
Deployment the file yaml

Download file in GitHub:



› 221-PERSO (F:) › Licence › A2SR › Communication Anglaise › Projet

Nom	Modifié le	Type	Taille
Phpmyadmin	05/04/2023 13:58	Dossier de fichiers	
wordpress	05/04/2023 09:07	Dossier de fichiers	
components.yaml	10/04/2023 13:11	Fichier source Yaml	5 Ko
Password.txt	29/03/2023 10:03	Document texte	1 Ko
web-connections.yaml	06/04/2023 19:52	Fichier source Yaml	1 Ko



```
PS F:\Licence\A2SR\Communication Anglaise\Projet> kubectl apply -f .\components.yaml
```

```
PS F:\Licence\A2SR\Communication Anglaise\Projet\wordpress> kubectl apply -k ./
```

```
PS F:\Licence\A2SR\Communication Anglaise\Projet\Phpmyadmin> kubectl apply -f .\
```

```
PS F:\Licence\A2SR\Communication Anglaise\Projet> kubectl apply -f .\web-connections.yaml
```

Glossary

apiVersion: the version of the Kubernetes API used for this service.

kind: the type of Kubernetes resource.

metadata: additional metadata for the service, such as the name of the service.

spec: the specification

selector: the labels to use for selecting the pods to expose as a service.

ports: the list of ports to expose on the service.

name: the name of the port.

port: the port number to use for the service.

targetPort: the port number to use for the selected pods.

type: the type of service to create. If chosen LoadBalancer, which will create an external IP address for accessing in web server.

containers: the list of containers to run in the pod, which includes:

name: the name of the container.

image: the container image to use for running the server.

command: the command to run to start the server and deploy.

ports: the list of ports the container should expose, which includes:

name: the name of the port.

containerPort: the port number on which the container is listening for incoming connections.

env: the environment variables to use for the container.