K8S EXAM

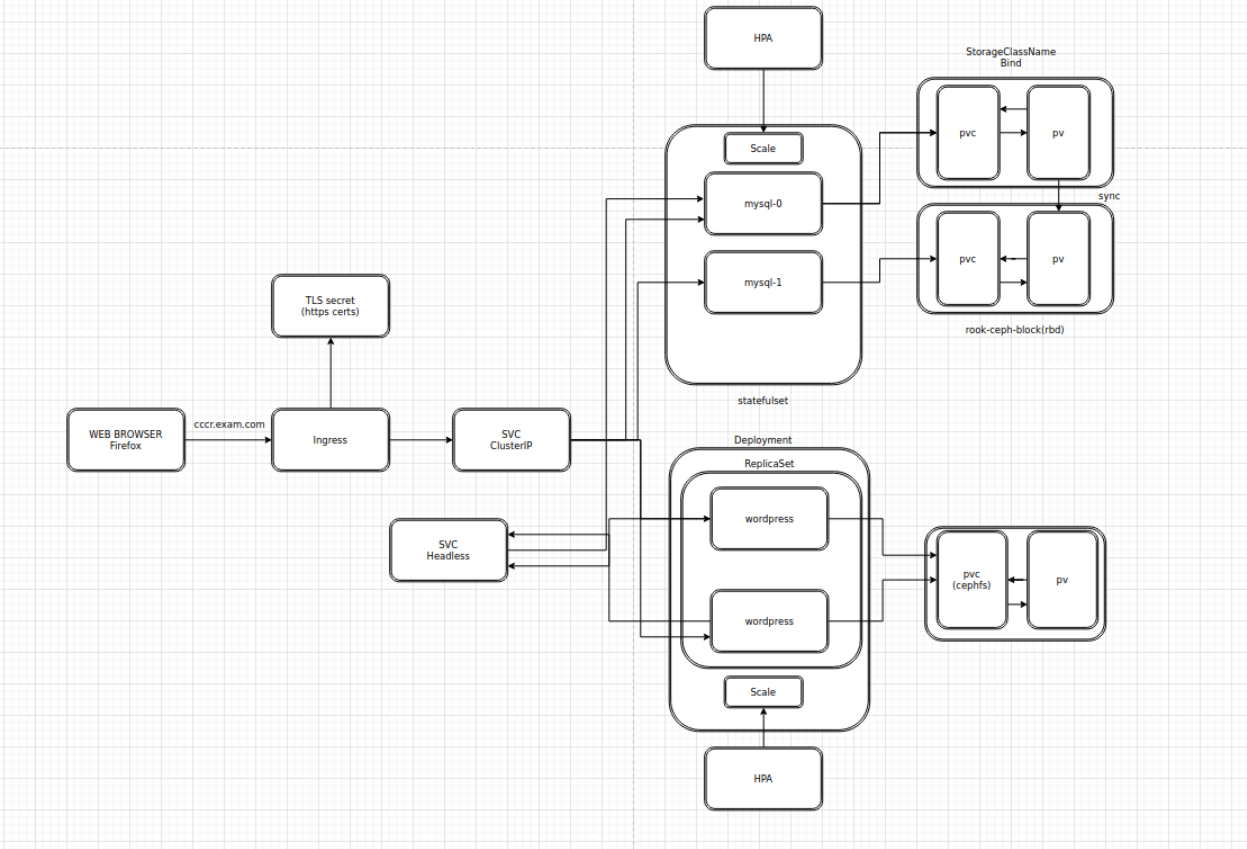
CCCR 구로

20200731 최인혁

Create the Wordpress Application

- 목표 : 현재까지 배워온 쿠버네티스 skill을 종합하여 Wordpress Application을 구축

Architecture



Build Environment

```
master_node:
  cpu: 2
  mem: 3072
  os: centos7

node1:
  cpu: 2
  mem: 3072
  os: centos7

node2:
  cpu: 2
  mem: 3072
  os: centos7

node3:
  cpu: 2
  mem: 3072
  os: centos7

# kubectl version
clientVersion:
  buildDate: "2020-07-15T16:58:53Z"
  compiler: gc
  gitCommit: dff82dc0de47299ab66c83c626e08b245ab19037
  gitTreeState: clean
  gitVersion: v1.18.6
  goVersion: go1.13.9
  major: "1"
  minor: "18"
  platform: linux/amd64
serverVersion:
```

```
  buildDate: "2020-02-11T19:24:46Z"
  compiler: gc
  gitCommit: be3d344ed06bff7a4fc60656200a93c74f31f9a4
  gitTreeState: clean
  gitVersion: v1.16.7
  goVersion: go1.13.6
  major: "1"
  minor: "16"
  platform: linux/amd64
```

Application Building Process

1. Ingress TLS Termination

- Create the openssl private key

```
$ openssl genrsa -out pv.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.............................................+++++
............................+++++
e is 65537 (0x010001)
```

- Create the openssl crt file

```
$ openssl req -new -x509 -key pv.key -out tls.crt -days 3650

You are about to be asked to enter information that will be incorporated
into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]:KR
State or Province Name (full name) [Some-State]:IN
Locality Name (eg, city) []:IN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IN
Organizational Unit Name (eg, section) []:IN
Common Name (e.g. server FQDN or YOUR name) []:cccr.exam.com
Email Address []:cccr.exam.com
```

- Createt the secret

```
$ kubectl create secret tls tls-secret --cert=tls.crt --key=pv.key
secret/tls-secret created
```

- Show the secret information

```
$ kubectl get secrets

NAME                  TYPE                                  DATA   AGE
default-token-zcs4n   kubernetes.io/service-account-token   3      18h
tls-secret            kubernetes.io/tls                     2      5s
```

```
$ kubectl get secret tls-secret -o yaml

apiVersion: v1
data:
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUR6ekNDQXJlZ0F3SUJBZ0lVUzFKdDZROWR0eU5KdzFmeVh5emNMSFFMV2E4d0RRWUpLb1pJaHZjTkFR
  tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcGdJQkFBS0NBUUVBMVFCTjVyeG4wajB0UjFnc3ByVU5acGZJcTE5dlBJVkFCajN1N3oxWXJPe
kind: Secret
metadata:
  creationTimestamp: "2020-07-31T03:19:43Z"
  name: tls-secret
  namespace: default
  resourceVersion: "840751"
  selfLink: /api/v1/namespaces/default/secrets/tls-secret
  uid: 7a559da8-bc65-4909-9dfc-3d66ed78198d
type: kubernetes.io/tls
```

안에 tls.crt 값고 tls.key 값이 들어 있는 것을 확인 할 수 있습니다.

- Create the wordpress-ingress.yml

```
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: wordpress-ingress
spec:
  # To use tls
  tls:
  - hosts:
    - cccr.exam.com
    # tls-secret has tls key and crt value.
    secretName: tls-secret
  rules:
  - host: cccr.exam.com
    http:
      paths:
      - path: /
        backend:
     # connect the service(wp-clusterip)
          serviceName: wp-clusterip
          servicePort: 80
```

- Create the ingress

```
$ kubectl create -f wordpress-ingress.yml
ingress.networking.k8s.io/wordpress-ingress created
```

- Show the ingress information

```
$ kubectl get ingress

NAME                HOSTS           ADDRESS   PORTS     AGE
wordpress-ingress   cccr.exam.com             80, 443   5m5s


$ kubectl describe ingress

Name:             wordpress-ingress
Namespace:        default
Address:
Default backend:  default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
TLS:
  tls-secret terminates cccr.exam.com
Rules:
  Host           Path  Backends
```

```
  ----           ----  --------
  cccr.exam.com
                 /   wp-clusterip:80 (<error: endpoints "wp-clusterip" not found>)
Annotations:     <none>
Events:
  Type    Reason  Age    From                      Message
  ----    ------  ----   ----                      -------
  Normal  CREATE  5m35s  nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  CREATE  5m35s  nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  CREATE  5m35s  nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  UPDATE  4m50s  nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  UPDATE  4m50s  nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  UPDATE  4m50s  nginx-ingress-controller  Ingress default/wordpress-ingress
```

tls를 사용했기 때문에 80, 443 포트가 열려 있고, 현재 wp-clusterip라는 svc를 만들지 않아 에러가 떠 있는 상태입니다.

Host: cccr.exam.com

2. Service : ClusterIP

- Create the svc(clusterip) → clusterip.yml

```
---
apiVersion: v1
kind: Service
metadata:
  # should be same name (ingress serviceName Part)
  name: wp-clusterip
  labels:
    # To connect with pod
    app: wp
spec:
  selector:
    # To connect with pod
    app: wp
  ports:
  # open the http and https port(80, 443) because we use the tls
  - name: http
    port: 80
  - name: https
    port: 443
```

label로 pod와 연결됩니다. selector로 pod 탐색해서 맞는 부분을 찾습니다.

- Show the svc information

```
$ kubectl get svc
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes     ClusterIP   10.233.0.1     <none>        443/TCP          16h
wp-clusterip   ClusterIP   10.233.25.50   <none>        80/TCP,443/TCP   5s


$ kubectl describe svc wp-clusterip
Name:              wp-clusterip
Namespace:         default
Labels:            app=wp
Annotations:       <none>
Selector:          app=wp
Type:              ClusterIP
IP:                10.233.25.50
Port:              http  80/TCP
TargetPort:        80/TCP
Endpoints:         <none>
Port:              https  443/TCP
TargetPort:        443/TCP
Endpoints:         <none>
Session Affinity:  None
Events:            <none>
```

- Check the ingress

```
$ kubectl describe ingress wordpress-ingress

Name:             wordpress-ingress
Namespace:        default
Address:
Default backend:  default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
TLS:
  tls-secret terminates cccr.exam.com
Rules:
  Host           Path  Backends
  ----           ----  --------
  cccr.exam.com
                 /   wp-clusterip:80 (<none>)
Annotations:     <none>
Events:
  Type    Reason  Age   From                      Message
  ----    ------  ----  ----                      -------
  Normal  CREATE  19m   nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  CREATE  19m   nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  CREATE  19m   nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  UPDATE  18m   nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  UPDATE  18m   nginx-ingress-controller  Ingress default/wordpress-ingress
  Normal  UPDATE  18m   nginx-ingress-controller  Ingress default/wordpress-ingress
```

Rules에 있던 오류가 없어졌습니다.

3. Deployment: Wordpress(Replica:2, Liveness, Readiness)

- Create the wp-deployment.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wp-deployment
spec:
  # Replica
  replicas: 2
  selector:
    matchLabels:
      # should be same label with svc
      app: wp
      # To use Affinity
      tier: wordpress
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  minReadySeconds: 30
  template:
    metadata:
      name: wordpress
      labels:
        app: wp
        tier: wordpress
    spec:
      #affinity
      affinity:
        # pod anti affinity
        # If has tier: wordpress, they are placed on different nodes.
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: "kubernetes.io/hostname"
              labelSelector:
                matchExpressions:
                - key: tier
```

```
                operator: In
                values:
                - wordpress
        # pod affinity
        # To be close with db pod
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: tier
                operator: In
                values:
                  - db
            topologyKey: "kubernetes.io/hostname"
      containers:
        - name: wordpress
          image: wordpress:latest
          # Environment variable
          env:
            - name: WORDPRESS_DB_HOST
              value: "mysql-0.mysql"
            - name: WORDPRESS_DB_USER
              value: "wp-admin"
            - name: WORDPRESS_DB_PASSWORD
              value: "1234"
            - name: WORDPRESS_DB_NAME
              value: "wordpress"
          ports:
            - containerPort: 80
          # liveness Probe
          # http Get Probe
          livenessProbe:
            httpGet:
              port: 80
              path: /
            initialDelaySeconds: 30
            periodSeconds: 10
            timeoutSeconds: 5
          # readiness Probe
          # Check the /var/www/html
          readinessProbe:
            exec:
              command:
              - ls
              - /var/www/html/
            initialDelaySeconds: 5
            periodSeconds: 2
            timeoutSeconds: 1
          volumeMounts:
            - name: wp-vol
              mountPath: /var/www/html
      # Use the ceph storage
      volumes:
        - name: wp-vol
          persistentVolumeClaim:
            claimName: wp-pvc
```

- Create the deployment

```
$ kubectl create -f wp-deployment.yml
deployment.apps/wp-deployment created
```

- Show the pod info

```
$ kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
wp-deployment-56dd744b6f-hggpb   0/1     Pending   0          2s
wp-deployment-56dd744b6f-rd2lb   0/1     Pending   0          2s
```

- Check the pod status

```
$ kubectl describe pod wp-deployment-56dd744b6f-hggpb
Name:           wp-deployment-56dd744b6f-hggpb
Namespace:      default
Priority:       0
Node:           <none>
Labels:         app=wp
                pod-template-hash=56dd744b6f
                tier=wordpress
Annotations:    <none>
Status:         Pending
IP:
IPs:            <none>
Controlled By:  ReplicaSet/wp-deployment-56dd744b6f
Containers:
  wordpress:
    Image:      wordpress:latest
    Port:       80/TCP
    Host Port:  0/TCP
    Liveness:   http-get http://:80/ delay=30s timeout=5s period=10s #success=1 #failure=3
    Readiness:  exec [ls /var/www/html/] delay=5s timeout=1s period=2s #success=1 #failure=3
    Environment:
      WORDPRESS_DB_HOST:      mysql-0.mysql
      WORDPRESS_DB_USER:      wp-admin
      WORDPRESS_DB_PASSWORD:  1234
      WORDPRESS_DB_NAME:      wordpress
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-zcs4n (ro)
      /var/www/html from wp-vol (rw)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  wp-vol:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  wp-pvc
    ReadOnly:   false
  default-token-zcs4n:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-zcs4n
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason            Age        From               Message
  ----     ------            ----       ----               -------
  Warning  FailedScheduling  <unknown>  default-scheduler  persistentvolumeclaim "wp-pvc" not found
  Warning  FailedScheduling  <unknown>  default-scheduler  persistentvolumeclaim "wp-pvc" not found
```

Events 부분에서 wp-pvc를 찾지 못해서 에러가 났습니다.

4. PVC: StorageClass(cephfs)

- Create the pvc-wp.yml

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pvc
spec:
  accessModes: ["ReadWriteMany"]
  resources:
    requests:
      storage: 1Gi
  # using ceph storage
  # storage class name
  storageClassName: csi-cephfs
```

- Create the pvc

```
$ kubectl create -f pvc-wp.yml
persistentvolumeclaim/wp-pvc created
```

- Show the pv, pvc info

```
$ kubectl get pv,pvc
NAME                                                        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM            STO
persistentvolume/pvc-aef5a062-ebe0-4095-9be4-146b3c1e2328   1Gi        RWX            Delete           Bound    default/wp-pvc   csi

NAME                           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/wp-pvc   Bound    pvc-aef5a062-ebe0-4095-9be4-146b3c1e2328   1Gi        RWX            csi-cephfs     4s
```

storage class name을 이용해 자동으로 pv까지 생성된 것을 볼 수 있다.

- Show the pod info

```
$ kubectl get pod
NAME                             READY   STATUS    RESTARTS   AGE
wp-deployment-56dd744b6f-hggpb   0/1     Pending   0          4m11s
wp-deployment-56dd744b6f-rd2lb   0/1     Pending   0          4m11s
```

- Show the pod status

```
$ kubectl describe pod wp-deployment-56dd744b6f-hggpb

...
Events:
  Type     Reason            Age        From               Message
  ----     ------            ----       ----               -------
  Warning  FailedScheduling  <unknown>  default-scheduler  persistentvolumeclaim "wp-pvc" not found
  Warning  FailedScheduling  <unknown>  default-scheduler  persistentvolumeclaim "wp-pvc" not found
  Warning  FailedScheduling  <unknown>  default-scheduler  pod has unbound immediate PersistentVolumeClaims (repeated 3 times)
  Warning  FailedScheduling  <unknown>  default-scheduler  0/4 nodes are available: 1 node(s) had taints that the pod didn't tolerat
```

DB part와 affinity를 걸어놨기에 tier:db가 없어서 올라가지 않는다.

5. HPA : Deployment

- 파일만 만들어 놓고 DB를 띄운 후 test 실시
- Create the hpa-wp.yml

```
---
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-cpu-wp
spec:
  # Target Reference
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: wp-deployment
```

```
  maxReplicas: 6
  minReplicas: 2
  targetCPUUtilizationPercentage: 80
```

- Create the hpa

```
$ kubectl create -f hpa-wp.yml
horizontalpodautoscaler.autoscaling/hpa-cpu-wp created
```

- Show the hpa info

```
$ kubectl get hpa
NAME         REFERENCE              TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hpa-cpu-wp   Deployment/wp-deployment   1%/80%    2         6         2          44s
```

```
$ kubectl describe hpa hpa-cpu-wp
Name:                                                  hpa-cpu-wp
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Fri, 31 Jul 2020 16:19:57 +0900
Reference:                                             Deployment/wp-deployment
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  1% (6m) / 80%
Min replicas:                                          2
Max replicas:                                          6
Deployment pods:                                       2 current / 2 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ScaleDownStabilized  recent recommendations were higher than current one, applying the highest recent reco
  ScalingActive   True    ValidMetricFound     the HPA was able to successfully calculate a replica count from cpu resource utilizat
  ScalingLimited  False   DesiredWithinRange   the desired count is within the acceptable range
```

6. Service: Headless

- Create the db-svc.yml

```
---
# Headless service for stable DNS entries of StatefulSet members.
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  ports:
  - name: mysql
    port: 3306
  # cluster ip set none
  clusterIP: None
  selector:
    app: mysql
---
# Client service for connecting to any MySQL instance for reads.
# For writes, you must instead connect to the master: mysql-0.mysql.
apiVersion: v1
kind: Service
metadata:
  name: mysql-read
  labels:
    app: mysql
```

```
spec:
  ports:
  - name: mysql
    port: 3306
  selector:
    app: mysql
```

- Create the svc

```
$ kubectl create -f db-svc.yml
service/mysql created
service/mysql-read created
```

- Show the svc info

```
$ kubectl get svc
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
mysql          ClusterIP   None           <none>        3306/TCP         37s
mysql-read     ClusterIP   10.233.3.164   <none>        3306/TCP         37s
...
```

7. Statefulset: Mysql(Replica:2, Liveness, Readiness)

8. PVC: StorageClass(rbd)

- 7~8번 같이
- Create the cm-mysql.yml

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql
  labels:
    app: mysql
data:
  master.cnf: |
    # Apply this config only on the master.
    [mysqld]
    log-bin
  slave.cnf: |
    # Apply this config only on slaves.
    [mysqld]
    super-read-only
```

- Create the configmap

```
$ kubectl create -f cm-mysql.yml
configmap/mysql created
```

- Show the configmap info

```
$ kubectl get configmaps
NAME    DATA   AGE
```

```
mysql   2     3s

$ kubectl describe configmaps
Name:         mysql
Namespace:    default
Labels:       app=mysql
Annotations:  <none>

Data
====
master.cnf:
----
# Apply this config only on the master.
[mysqld]
log-bin

slave.cnf:
----
# Apply this config only on slaves.
[mysqld]
super-read-only

Events:  <none>
```

app=mysql 라벨을 가진 pod에 설정해주는 master.cnf, slave.cnf 2개 Data part가 생겼다.

master와 slave를 나누는 설정 값이 들어 있고, master는 rw, slave는 ro다.

- Create the wp-db.yml

```
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
      tier: db
  serviceName: mysql
  # Replica
  replicas: 2
  template:
    metadata:
      labels:
        # labels, using configmaps, svc
        app: mysql
        # using affinity
        tier: db
    spec:
      initContainers:
      - name: init-mysql
        image: mysql:5.7
        # if ordinal == 0, master, else slave
        command:
        - bash
        - "-c"
        - |
          set -ex
          # Generate mysql server-id from pod ordinal index.
          [[ `hostname` =~ -([0-9]+)$ ]] || exit 1
          ordinal=${BASH_REMATCH[1]}
          echo [mysqld] > /mnt/conf.d/server-id.cnf
          # Add an offset to avoid reserved server-id=0 value.
          echo server-id=$((100 + $ordinal)) >> /mnt/conf.d/server-id.cnf
          # Copy appropriate conf.d files from config-map to emptyDir.
          if [[ $ordinal -eq 0 ]]; then
            cp /mnt/config-map/master.cnf /mnt/conf.d/
          else
            cp /mnt/config-map/slave.cnf /mnt/conf.d/
          fi
        volumeMounts:
        - name: conf
          mountPath: /mnt/conf.d
        - name: config-map
```

```
          mountPath: /mnt/config-map
      - name: clone-mysql
        image: gcr.io/google-samples/xtrabackup:1.0
        command:
       ...
      # affinity
      affinity:
        # pod anti affinity
        # If has tier: db, they are placed on different nodes.
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: "kubernetes.io/hostname"
              labelSelector:
                matchExpressions:
                - key: tier
                  operator: In
                  values:
                  - db
      ...
        # liveness Probe
        # EXEC Probe
        livenessProbe:
          exec:
            command: ["mysqladmin", "ping"]
          initialDelaySeconds: 30
          periodSeconds: 10
          timeoutSeconds: 5
        # readiness Probe
        readinessProbe:
          exec:
            # Check we can execute queries over TCP (skip-networking is off).
            command: ["mysql", "-h", "127.0.0.1", "-e", "SELECT 1"]
          initialDelaySeconds: 5
          periodSeconds: 2
          timeoutSeconds: 1
      - name: xtrabackup
        image: gcr.io/google-samples/xtrabackup:1.0
        ports:
        ...
        volumeMounts:
        - name: data
          mountPath: /var/lib/mysql
          subPath: mysql
        - name: conf
          mountPath: /etc/mysql/conf.d
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
          limits:
            cpu: 200m
            memory: 400Mi
      volumes:
      - name: conf
        emptyDir: {}
      - name: config-map
        configMap:
          name: mysql
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      accessModes: ["ReadWriteOnce"]
      resources:
        requests:
          storage: 10Gi
      # using storage class Name
      # ceph storage
      storageClassName: rook-ceph-block
```

- Create the DB

```
$ kubectl get pod
statefulset.apps/mysql created
```

- Show the DB info

```
$ kubectl get pod
NAME                            READY   STATUS     RESTARTS   AGE
mysql-0                         2/2     Running    0          47s
mysql-1                         0/2     Init:0/2   0          3s
wp-deployment-56dd744b6f-hggpb   0/1     Pending    0          36m
wp-deployment-56dd744b6f-rd2lb   1/1     Running    0          36m
```

```
$ kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
mysql-0                         2/2     Running   0          91s
mysql-1                         2/2     Running   0          47s
wp-deployment-56dd744b6f-hggpb   1/1     Running   0          37m
wp-deployment-56dd744b6f-rd2lb   1/1     Running   0          37m
```

안올라오던 wp-delpoyment pod도 모두 다 올라온 것을 확인 할 수 있습니다.

- Show the pv, pvc

```
$ kubectl get pv,pvc
NAME                                                        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
persistentvolume/pvc-3fe404b0-dbf7-42e8-9acf-8c7eef336e96   10Gi       RWO            Delete           Bound    default/data-mysql-0
persistentvolume/pvc-c2491d32-ea7e-4342-a6db-03c00a6561d9   1Gi        RWX            Delete           Bound    default/wp-pvc
persistentvolume/pvc-fca1db11-3a71-4ccf-9494-31c42c60efcd   10Gi       RWO            Delete           Bound    default/data-mysql-1

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
persistentvolumeclaim/data-mysql-0    Bound    pvc-3fe404b0-dbf7-42e8-9acf-8c7eef336e96   10Gi       RWO            rook-ceph-block
persistentvolumeclaim/data-mysql-1    Bound    pvc-fca1db11-3a71-4ccf-9494-31c42c60efcd   10Gi       RWO            rook-ceph-block
persistentvolumeclaim/wp-pvc          Bound    pvc-c2491d32-ea7e-4342-a6db-03c00a6561d9   1Gi        RWX            csi-cephfs
```

storage class Name 으로 인해 동적으로 pvc,pv가 생성되었습니다.

mysql-0에는 rw, mysql-1은 ro로 rook-ceph-block이 올라온 것을 확인 할 수 있습니다.

9. HPA: Statefulset

- Create the hpa-db.yml

```
---
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-cpu-db
spec:
  # Target Reference
  scaleTargetRef:
    apiVersion: apps/v1
    kind: StatefulSet
    name: mysql
  maxReplicas: 6
  minReplicas: 2
  targetCPUUtilizationPercentage: 80
```

- Create the hpa

```
$ kubectl create -f hpa-db.yml
horizontalpodautoscaler.autoscaling/hpa-cpu-wp created
```

- Show the hpa info

```
$ kubectl get hpa
NAME         REFERENCE                  TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hpa-cpu-db   StatefulSet/mysql          1%/70%          2         5         2          8s
...


$ kubectl describe hpa hpa-cpu-db
Name:                                                  hpa-cpu-db
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Fri, 31 Jul 2020 16:13:39 +0900
Reference:                                             StatefulSet/mysql
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  3% (22m) / 80%
Min replicas:                                          2
Max replicas:                                          6
StatefulSet pods:                                      2 current / 2 desired
Conditions:
  Type            Status  Reason               Message
  ----            ------  ------               -------
  AbleToScale     True    ScaleDownStabilized  recent recommendations were higher than current one, applying the highest recent reco
  ScalingActive   True    ValidMetricFound     the HPA was able to successfully calculate a replica count from cpu resource utilizat
  ScalingLimited  False   DesiredWithinRange   the desired count is within the acceptable range
Events:           <none>
```

- Load the db-pod

```
$ kubectl exec mysql-1 -- sha1sum /dev/zero &
[1] 32331
Defaulting container name to mysql.
Use 'kubectl describe pod/mysql-1 -n default' to see all of the containers in this pod.


$ kubectl exec mysql-1 -- sha1sum /dev/zero &
[2] 32424
Defaulting container name to mysql.
Use 'kubectl describe pod/mysql-1 -n default' to see all of the containers in this pod.


$ kubectl exec mysql-1 -- sha1sum /dev/zero &
[3] 32534
Defaulting container name to mysql.
Use 'kubectl describe pod/mysql-1 -n default' to see all of the containers in this pod.

$ kubectl  exec mysql-0 -- sha1sum /dev/zero &
[8] 6310
Defaulting container name to mysql.
Use 'kubectl describe pod/mysql-0 -n default' to see all of the containers in this pod.

$ kubectl  exec mysql-0 -- sha1sum /dev/zero &
[9] 6442
Defaulting container name to mysql.
Use 'kubectl describe pod/mysql-0 -n default' to see all of the containers in this pod.


$ kubectl  exec mysql-0 -- sha1sum /dev/zero &
[10] 6536
Defaulting container name to mysql.
Use 'kubectl describe pod/mysql-0 -n default' to see all of the containers in this pod.
```

- Check

```
$ kubectl get hpa
NAME         REFERENCE                  TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
```

```
hpa-cpu-db   StatefulSet/mysql          171%/80%   2         6         5         22m

$ kubettl top pod
NAME                            CPU(cores)   MEMORY(bytes)
mysql-0                         1002m        272Mi
mysql-1                         1001m        205Mi
mysql-2                         17m          224Mi


$ kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
mysql-0                         2/2     Running   0          70m
mysql-1                         2/2     Running   0          69m
mysql-2                         2/2     Running   0          97s
...
```

```
NAME                            CPU(cores)   MEMORY(bytes)
mysql-0                         1000m        272Mi
mysql-1                         1055m        206Mi
wp-deployment-7b8b89f547-qjw5f  9m           51Mi
wp-deployment-7b8b89f547-rqwsg  7m           61Mi
```

```
student@student-Aspire-E5-576 ~/exam kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
mysql-0                         2/2     Running   0          70m
mysql-1                         2/2     Running   0          69m
mysql-2                         2/2     Running   0          97s
mysql-3                         0/2     Pending   0          51s
wp-deployment-7b8b89f547-qjw5f  1/1     Running   0          39m
wp-deployment-7b8b89f547-rqwsg  1/1     Running   0          39m
```

- Test wp-pod

```
student@student-Aspire-E5-576 ~/exam kubectl get hpa                    8 3372 17:37:01
NAME         REFERENCE                   TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-cpu-db   StatefulSet/mysql           3%/80%     2         6         2          83m
hpa-cpu-wp   Deployment/wp-deployment     200%/80%   2         6         5          77m
student@student-Aspire-E5-576 ~/exam kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
mysql-0                         2/2     Running   0          13m
mysql-1                         2/2     Running   0          12m
wp-deployment-7b8b89f547-4dntk  0/1     Pending   0          38s
wp-deployment-7b8b89f547-l89hx  0/1     Pending   0          38s
wp-deployment-7b8b89f547-qjw5f  1/1     Running   3          100m
wp-deployment-7b8b89f547-rqwsg  1/1     Running   3          100m
wp-deployment-7b8b89f547-vztcn  0/1     Pending   0          23s
student@student-Aspire-E5-576 ~/exam
```

10. PodAffinity 및 PodAntiAffinity (wp/db <-> wp/db)

- Add podAntiAffinity part in wp-db.yml

```
#wp-db.yml

---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
...
      # affinity
      affinity:
        # pod anti affinity
        # If has tier: db, they are placed on different nodes.
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: "kubernetes.io/hostname"
```

```
              labelSelector:
                matchExpressions:
                - key: tier
                  operator: In
                  values:
                  - db
...
```

- Add podAntiAffinity & podAffinity part in wp-deployment.yml

```
#wp-deployment.yml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wp-deployment
...
    spec:
      #affinity
      affinity:
        # pod anti affinity
        # If has tier: wordpress, they are placed on different nodes.
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - topologyKey: "kubernetes.io/hostname"
              labelSelector:
                matchExpressions:
                - key: tier
                  operator: In
                  values:
                  - wordpress
        # pod affinity
        # To be close with db pod
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: tier
                operator: In
                values:
                  - db
            topologyKey: "kubernetes.io/hostname"
...
```

- Check the pod

```
$ kubectl get pod -o wide
NAME                            READY   STATUS    RESTARTS   AGE   IP              NODE    NOMINATED NODE   READINESS GATES
mysql-0                         2/2     Running   0          10m   10.233.96.135   node2   <none>           <none>
mysql-1                         2/2     Running   0          10m   10.233.90.210   node1   <none>           <none>
wp-deployment-56dd744b6f-hggpb   1/1     Running   0          46m   10.233.90.211   node1   <none>           <none>
wp-deployment-56dd744b6f-rd2lb   1/1     Running   0          46m   10.233.96.134   node2   <none>           <none>
```

- pod anti affinity 때문에 mysql pod 끼리는 다른 node에, wp-deployment pod 끼리는 다른 node에 배치되었습니다.
- 하지만 다른 노드라고 따로따로 배치 된 것이 아니라 pod affinity 때문에 mysql pod가 존재하는 파드에는 wp-deployment pod도 같이 배치 된 것을 확인할 수 있습니다.

11. ConfigMap, Secret

- 7번에 master, slave 설정하는 cnf파일이 configMap으로 사용되었습니다.
- 1번에 tls 정보를 넣는데 Secret이 사용되었습니다.

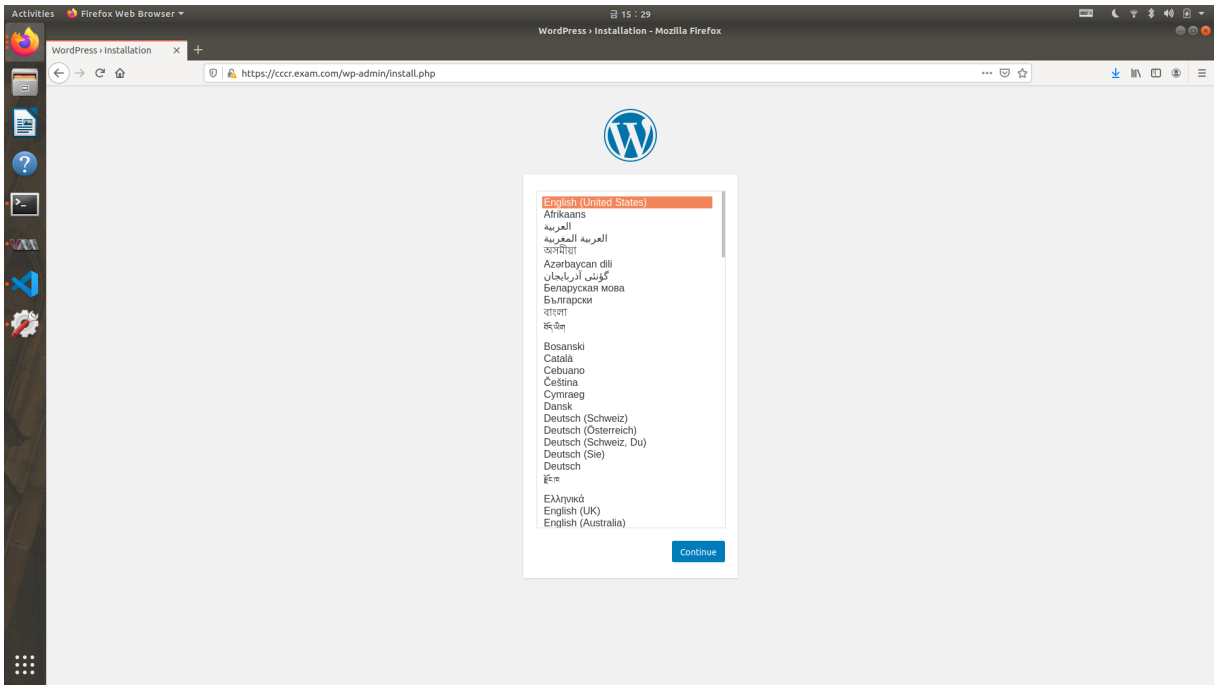
Result

cluster ip(내부용)를 사용하기 때문에 결과를 보기 위해서는 /etc/hosts 파일을 수정해주어야 합니다.

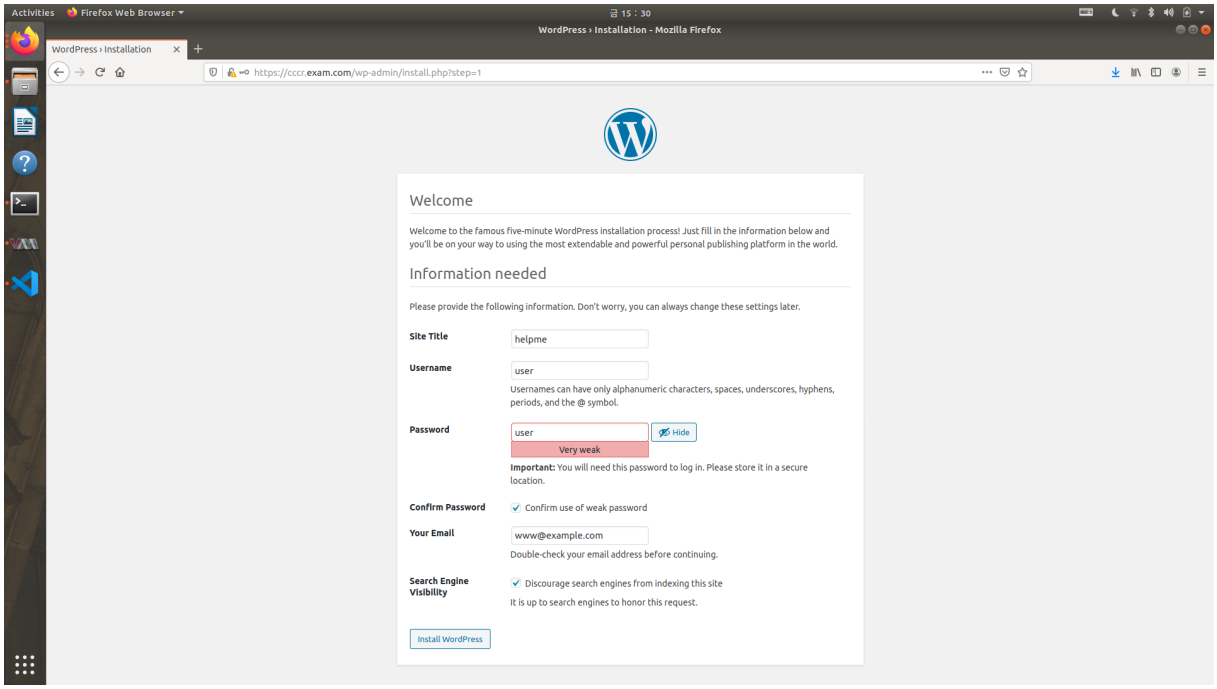
```
$ cat /etc/hosts
...
192.168.122.21 cccr.exam.com
192.168.122.22 cccr.exam.com
192.168.122.23 cccr.exam.com
```

ubuntu에 설치되어 있는 firefox를 통해 접근합니다.

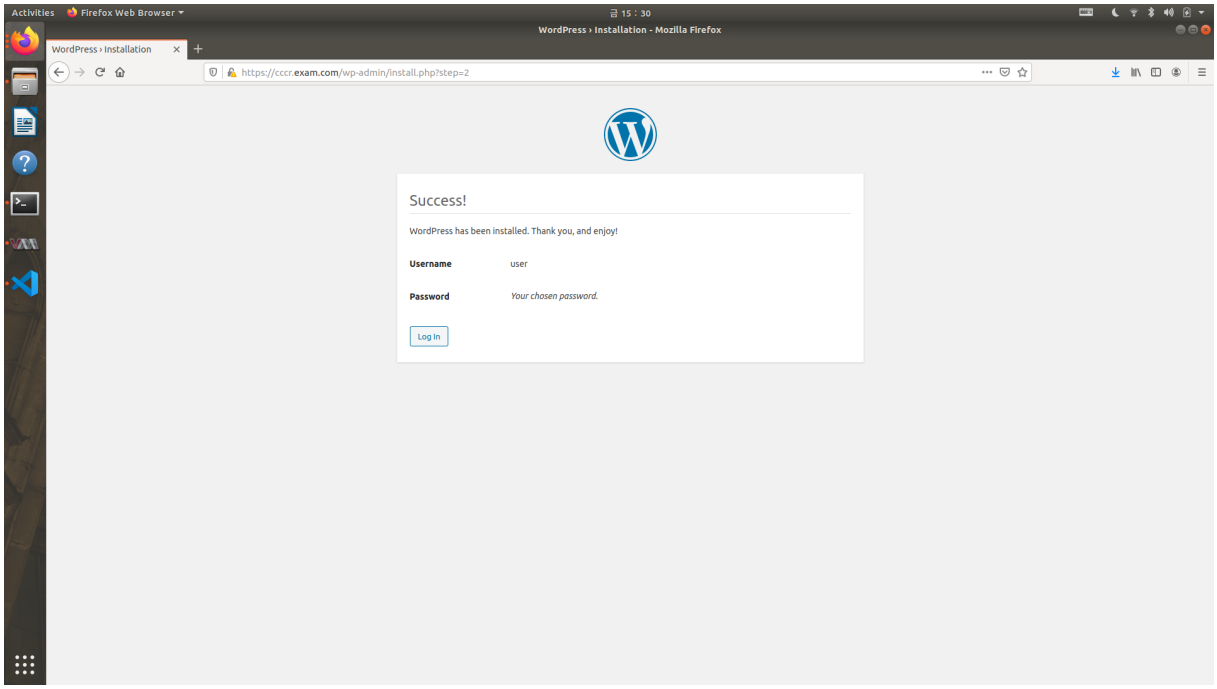
1. Install Page



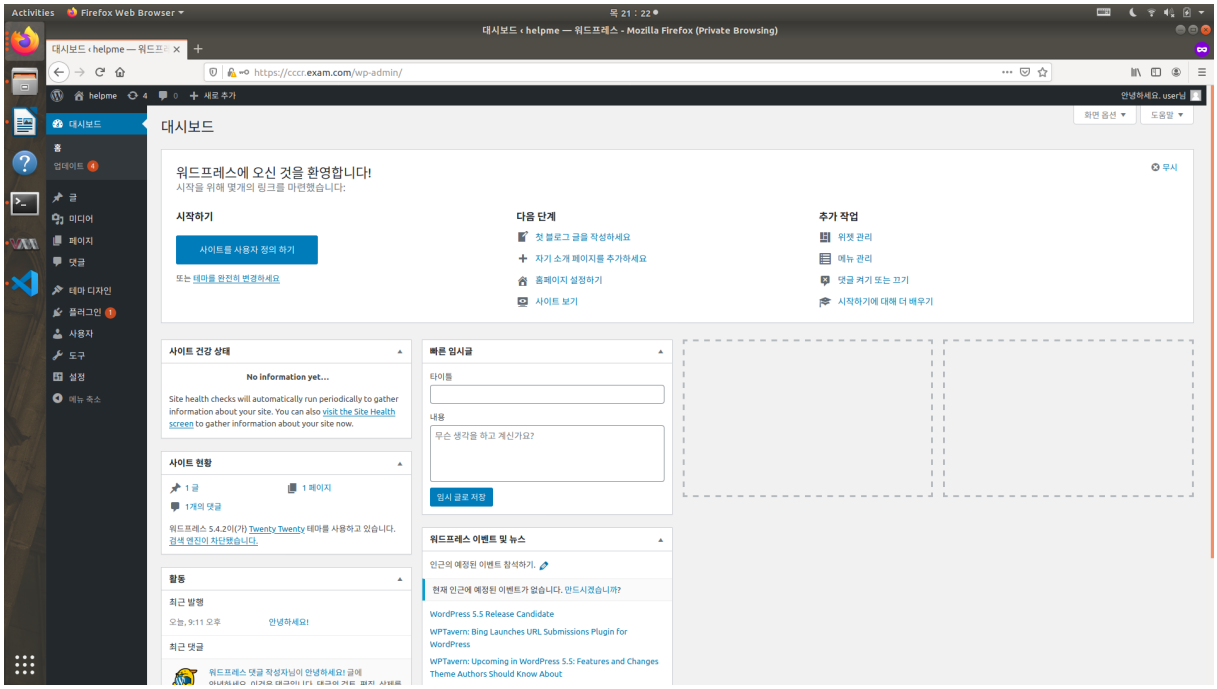
2. Sign up Page



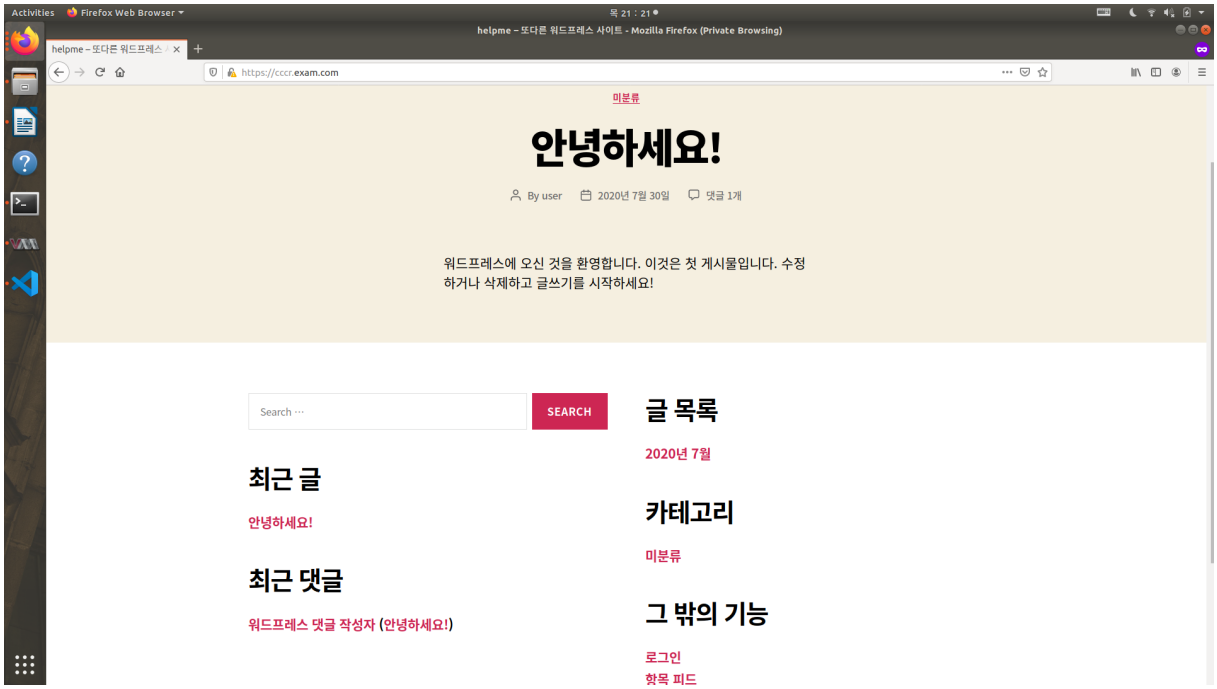
3. Sign up Success(Write DB)



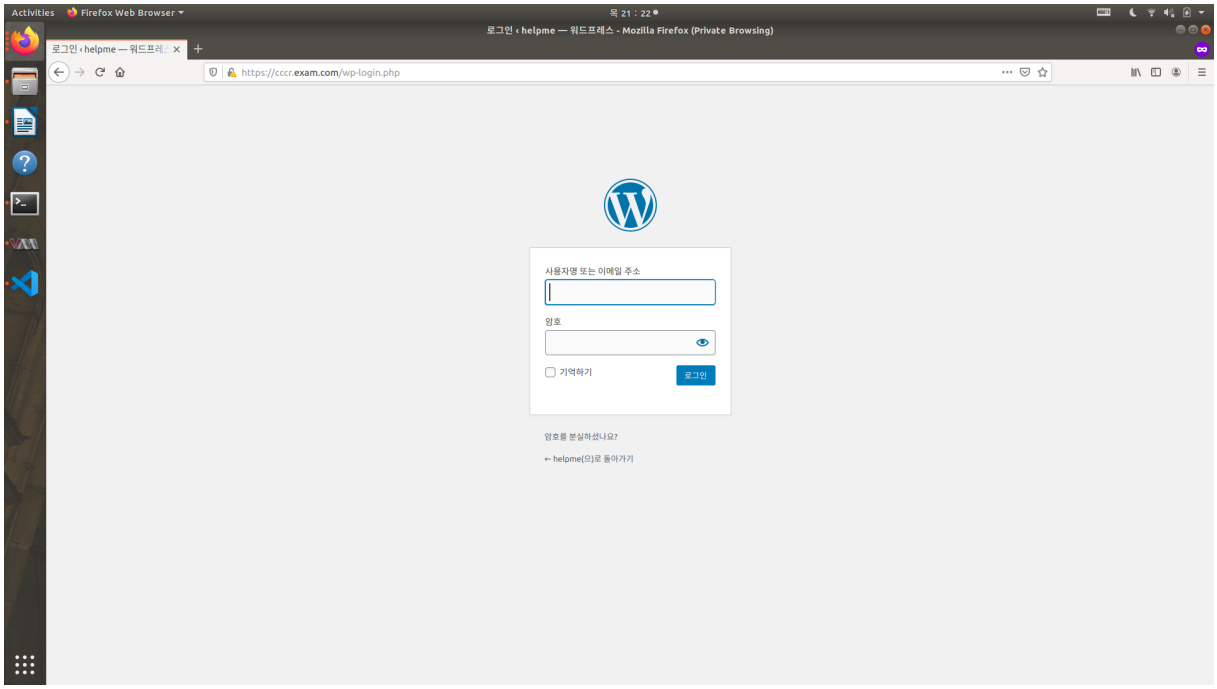
4. DashBoard



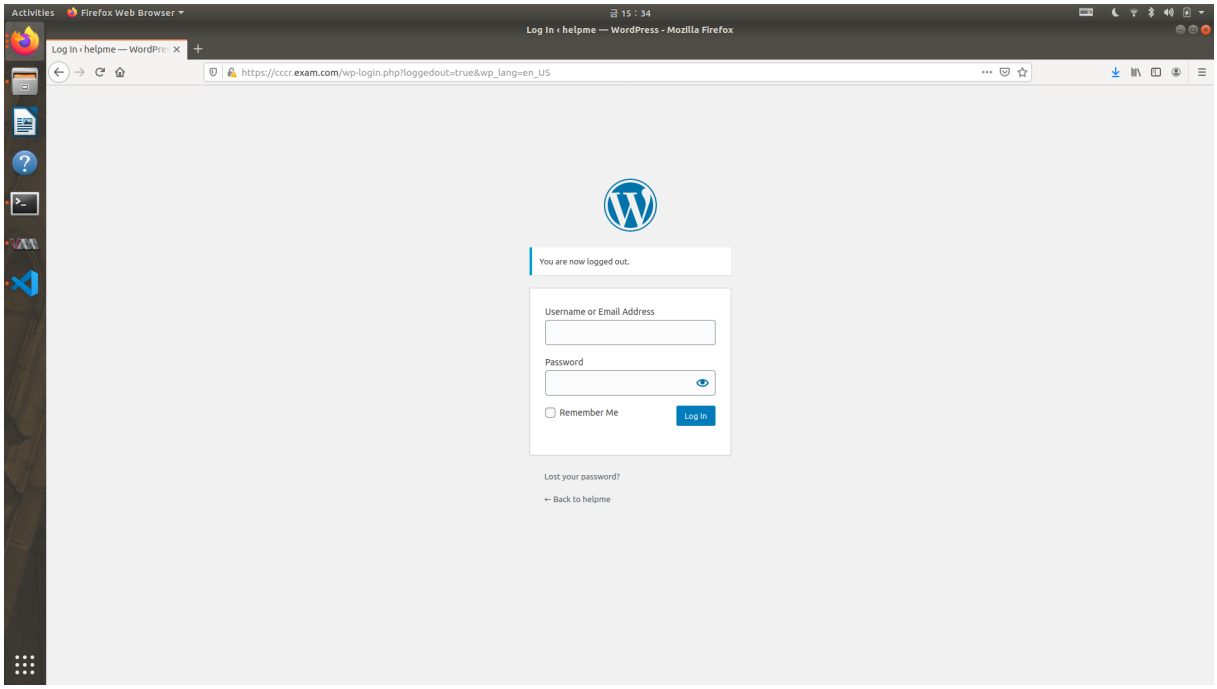
5. Homepage



6. Login Page



7. Log out Page



- 목,금으로 테스트 하고 만들어서 한글 버전과 영어버전 wordpress 2장입니다.