

**HƯỚNG DẪN DEPLOY ỨNG DỤNG ASP.NET LÊN GOOGLE CLOUD PLATFORM VỚI GOOGLE KUBERNETES ENGINE**

**MỤC LỤC**

[CHƯƠNG 1 : Deploy ASP.NET Core Application lên Google Cloud Platform với Kubernetes Engine. 3](#_Toc77169375)

[1.1 Chuẩn bị cho việc deploy 3](#_Toc77169376)

[*1.1.1* Cài đặt các tài nguyên 3](#_Toc77169377)

[1.2 Các bước thực hiện 3](#_Toc77169378)

[*1.2.1* Khai báo biến môi trường 3](#_Toc77169379)

[ENVIRONMENT VARIABLE 4](#_Toc77169380)

[*1.2.2* Tạo một Repository lên Artifact Registry 4](#_Toc77169381)

[*1.2.3* Build container image cho ứng dụng 5](#_Toc77169382)

[*1.2.4* Push Docker Image của ứng dụng lên Artifact Registry 8](#_Toc77169383)

[*1.2.5* Tạo GKE Cluster 9](#_Toc77169384)

[*1.2.6* Deploy ứng dụng lên GKE & cấu hình auto scale 10](#_Toc77169385)

[*1.2.7* Expose ứng dụng lên Internet. 11](#_Toc77169386)

[*1.2.8* Deploy version mới của ứng dụng 13](#_Toc77169387)

[CHƯƠNG 2 : Deploy ASP.NET Framework Application lên Google Cloud Platform với Kubernetes Engine. 14](#_Toc77169388)

[2.1 Chuẩn bị cho việc deploy 14](#_Toc77169389)

[2.2 Các bước thực hiện 15](#_Toc77169390)

[*2.2.1* Khai báo biến môi trường 15](#_Toc77169391)

[*2.2.2* ENVIRONMENT VARIABLE 15](#_Toc77169392)

[*2.2.3* Set các cấu hình mặc định cho gcloud 16](#_Toc77169393)

[*2.2.4* Build và Run project ở local 16](#_Toc77169394)

[*2.2.1* Tạo GKE Cluster 16](#_Toc77169395)

[*2.2.2* Push hoặc pull image lên Google Container Registry 17](#_Toc77169396)

[*2.2.3* Tạo node pool windows 25](#_Toc77169397)

[*2.2.4* Deploy ứng dụng lên GKE 27](#_Toc77169398)

[*2.2.5* Expose ứng dụng lên Internet 29](#_Toc77169399)

[*2.2.6* Deploy version mới của ứng dụng 31](#_Toc77169400)

[2.3 Kết quả thực hiện 32](#_Toc77169401)

# : Deploy ASP.NET Core Application lên Google Cloud Platform với Kubernetes Engine.

## Chuẩn bị cho việc deploy

### Cài đặt các tài nguyên

Để phục vụ cho việc deploy một ứng dụng ASP.NET lên GCP với KBE ta cần chuẩn bị những thứ như sau:

Nhằm tối giản các bước thực hiện, ở tutorial này chúng ta sẽ sử dụng Cloud Shell của Google Cloud Console. Vì ở Cloud Shell, các bộ command line tools như gcloud, docker, và kubectl đều đã được cài đặt sẵn.

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Ghi chú |
| 1 | Google Cloud Project (Tài khoản enable billing account) |  |
| 2 | Enable các API cần thiết (bao gồm: Artifact Registry API, Kubernetes Engine API) | Để enable các API trên, [nhấp vào đây](https://console.cloud.google.com/flows/enableapi?apiid=artifactregistry.googleapis.com,container.googleapis.com&_ga=2.150453238.1734173899.1626057467-586884957.1611153413&_gac=1.12323782.1625379545.CjwKCAjwlYCHBhAQEiwA4K21m0zCg3oc8ZjtJS6IIe5DcSNnN0zxD1HLgfvvsmkKTx9V2KxXMTJkcRoC1UgQAvD_BwE) |
| 3 | Sample Code mẫu:  ***Mô tả:*** Ứng dụng mẫu mà chúng ta sẽ deploy là một ASP.NET Core API Application, đã được container hoá ứng dụng bằng Docker. | Sample được lưu trữ trên Github tại đây |

## Các bước thực hiện

### Khai báo biến môi trường

Để thuận tiện cho việc thực hiện, chúng ta khai báo các biến môi trường hay được sử dụng để giảm thiểu thời gian thực hiện.

|  |  |  |  |
| --- | --- | --- | --- |
| ENVIRONMENT VARIABLE | | | |
| **KEY** | **VALUE (EXAMPLE)** | **DESCRIPTION** | **CLI** |
| ***PROJECT\_ID*** | kubl-deploy-pj | Project trong Google Cloud Console | export PROJECT\_ID=**kubl-deploy-pj**  **--**  **gcloud config set project *%{PROJECT\_ID}*** |
| ***REPOSITORY\_NAME*** | my-api-sample-repo | Tên repository | export REPOSITORY\_NAME=**my-api-sample-repo** |
| ***APP\_NAME*** | api-sample-app | Tên tag tao image trong Docker | export APP\_NAME=**api-sample-app** |
| ***CLUSTER\_NAME*** | aspnet-cluster | Tên cluster Google Kubernetes Engine (GKE) | export CLUSTER\_NAME=**aspnet-cluster** |
| ***GKE\_SERVICE\_NAME*** | api-sample-app-service | Tên service deployment GKE | export GKE\_SERVICE\_NAME=**api-sample-app-service** |
| ***REGION*** | asia-southeast1 | Region sẽ set | export REGION=**asia-southeast1** |
| ***ZONE*** | asia-southeast1-a | Zone sẽ set | export ZONE=**asia-southeast1-a** |

|  |
| --- |
| **Heart with solid fillGhi chú:** Về việc chọn Zone & Region phù hợp với dịch vụ cũng như doanh nghiệp, ta tham khảo các tài nguyên sau:  (1) [Global Locations - Regions & Zones  |  Google Cloud](https://cloud.google.com/about/locations#asia-pacific)  (2) [Regions and zones  |  Compute Engine Documentation  |  Google Cloud](https://cloud.google.com/compute/docs/regions-zones) |

### Tạo một Repository lên Artifact Registry

Chúng ta sẽ store image trên Artifact Registry. Có hai loại dịch vụ registry phổ biến trên GCP bên cạnh Artifact Registry là Container Registry. Song, Artifact Registry có nhiều điểm ưu việt hơn Container Registry, và GCP cũng recommend chúng ta sử dụng nó.

|  |
| --- |
| **Heart with solid fillGhi chú:** So sánh giữa Artifact Registry và Container Registry: [Tham khảo tại đây](https://stackoverflow.com/questions/65725870/google-cloud-artifact-registry-vs-container-registry) |

Các bước thực hiện như sau:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Tạo repository với tên ***REPOSITORY\_NAME*** đã khai báo trước đó | Chạy dòng lệnh sau trên Cloud Shell:  gcloud artifacts repositories create ***${REPOSITORY\_NAME}*** \    --repository-format=docker \    --location=***${REGION}*** \    --description="Docker repository with KBE" |
| Output | | |
|  | | |

### Build container image cho ứng dụng

Trước khi deploy ứng dụng lên GKE, chúng ta phải đóng gói ứng dụng dưới dạng Docker Image. Như trong code sample, chúng ta có thể thấy nó bao gồm source code và Dockerfile khai báo các hướng dẫn về các image cần build.

Các bước thực hiện:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Clone source code từ VCS (Ví dụ ở đây là Github) | Chạy dòng lệnh sau trên Cloud Shell:  git clone [**https://github.com/iammin99/MySampleApp.git**](https://github.com/iammin99/MySampleApp.git) |
| 2 | Trỏ đến thư mục project mà chúng ta muốn deploy | cd **ApiSampleApp** |
| 3 | Build & tag Docker image cho ứng dụng | docker build -t ***${REGION}***-docker.pkg.dev/**${PROJECT\_ID}**/***${REPOSITORY\_NAME}***/**${APP\_NAME}**:v1 .  Dòng command line trên sẽ build các image được khai báo trong Dockerfile, tag name cho nó theo quy tắc của Artifact Registry (AR). Ví dụ ở đây là: asia-southeast1-docker.pkg.dev/kubl-deploy-pj/my-api-sample-repo/api-sample-app:v1. Sau đó sẽ push image này lên AR ở bước sau.  Để kiểm tra image đó đã được build chưa, chúng ta chạy lệnh:  docker images |
| Output | | |
|  | | |

|  |
| --- |
| **Heart with solid fillGhi chú:** Vì project của chúng ta dùng source version là SVN nên chúng ta có thể clone SVN Repo bằng lệnh sau: git svn clone -s SVN\_REPO\_ROOT\_URL |

### Push Docker Image của ứng dụng lên Artifact Registry

Để GKE cluster có thể download và run container image của ứng dụng, chúng ta phải upload container image đó lên registry. Ở đây chúng ta sử dụng Artifact Registry để phục vụ cho việc đó.

Các bước thực hiện như sau:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Cấu hình Docker CLI xác thực Artifact Registry API | gcloud auth configure-docker ***${REGION}***-docker.pkg.dev |
| 2 | Push Docker image mà chúng ta vừa build ở bước trước đó lên AR | docker push ***${REGION}***-docker.pkg.dev/**${PROJECT\_ID}**/***${REPOSITORY\_NAME}***/**${APP\_NAME}**:v1 |
| Output | | |
|  | | |

### Tạo GKE Cluster

Sau khi push image lên AR, chúng ta tạo GKE cluster để chạy ứng dụng. GCP cung cấp 2 loại cluster là Standard Cluster và Autopilot Cluster, ở tutorial này chúng ta sẽ demo trên Standard Cluster.

|  |
| --- |
| **Heart with solid fillGhi chú:** Để quyết định chọn Standard Cluster hay Autopilot Cluster của GKE. Chúng ta tham khảo một số nguồn sau:  (1) [Autopilot overview  |  Kubernetes Engine Documentation  |  Google Cloud](https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview?authuser=2&_ga=2.132187658.-586884957.1611153413&_gac=1.116072820.1625379545.CjwKCAjwlYCHBhAQEiwA4K21m0zCg3oc8ZjtJS6IIe5DcSNnN0zxD1HLgfvvsmkKTx9V2KxXMTJkcRoC1UgQAvD_BwE)  (2) [Pricing  |  Google Kubernetes Engine (GKE)  |  Google Cloud](https://cloud.google.com/kubernetes-engine/pricing) |

Các bước thực hiện:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Set zone/region cho cho Compute Region. | **Standard Cluster**  gcloud config set compute/zone ***${ZONE}*** |
| **Autopilot Cluster**  gcloud config set compute/region ***${REGION}*** |
| 2 | Tạo cluster | **Standard Cluster**  gcloud container clusters create ***%{CLUSTER\_NAME}*** |
| **Autopilot Cluster**  gcloud container clusters create-auto ***%{CLUSTER\_NAME}*** |
| Output | | |
|  | | |

### Deploy ứng dụng lên GKE & cấu hình auto scale

Tiếp theo, chúng ta sẽ tạo một Kubernetes Deployment để run application trên cluster vừa tạo. Set số lượng replicas (pods) cho nó. Một pod chỉ chứa một container là docker image ứng dụng của chúng ta.

Sau đó, chúng ta sẽ tạo HorizontalPodAutoscaler (HPA) và cấu hình auto scale nhằm đáp ứng nhu cầu sử dụng.

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Kiểm tra xem có kết nối với GKE Cluster chưa | gcloud container clusters get-credentials ***%{CLUSTER\_NAME}***--zone ***${ZONE}*** |
| 2 | Tạo Kubernetes Deployment cho Docker image của ứng dụng | kubectl create deployment ***%{APP\_NAME}*** –image=***${REGION}***-docker.pkg.dev/**${PROJECT\_ID}**/***${REPOSITORY\_NAME}***/**${APP\_NAME}**:v1 |
| 4 | Tạo HPA & cấu hình auto scale | 1. Scale thay đổi chỉ số replica (số lương Pod) của Deployment. VD: set số lượng pods lên là 2  kubectl scale deployment ***%{APP\_NAME}*** --replicas=**2**  2. Tạo HorizontalPodAutoscaler cho Deployment. Ví dụ, ta thiết lập scale tự động với số lượng POD trong khoảng min = 1, max = 5 và thực hiện scale khi CPU của POD hoạt động ở mức 50% thì thực hiện  kubectl autoscale deployment ***%{APP\_NAME}*** --cpu-percent=**50** --min=**1** --max=**5** |
|  | Xem thông tin và thông tin chi tiết pod vừa tạo. | 1. Xem thông tin pod được tạo  kubectl get pods  2. Xem thông tin chi tiết một pod  kubectl describe pods <ten-pod> |
| Output | | |
|  | | |

### Expose ứng dụng lên Internet.

Service type mặc định của GKE là ClusterIP, nó nhận một địa chỉ IP chỉ co thể liên lạc bên trong cluster. Vì thế, để expose ra ngoài cluster, chúng ta sẽ tạo một Service với type là LoadBalancer. Loại service này sẽ tạo ra một IP cân bằng tải cho các pods, có thể truy cập trên Internet.

Các bước thực hiện:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Tạo Kubernetes Service cho Deployment | kubectl expose deployment ***${APP\_NAME}*** --name= ***${GKE\_SERVICE\_NAME}*** --type=**LoadBalancer** --port **80** --target-port **8080**  ***Chú thích:***  --port 80: số port cấu hình trên Load Balancer  --target-port 8080: số port là container của ứng dụng lắng nghe. |
| 2 | Xem thông tin & thông tin chi tiết Service vừa tạo | 1. Xem thông tin service được tạo  kubectl get service  2. Xem thông tin chi tiết service vừa tạo  kubectl describe service ***${GKE\_SERVICE\_NAME}*** |
| Output | | |
|  | | |

|  |
| --- |
| **Heart with solid fillGhi chú:** Tham khảo về việc expose external IP: [Exposing an External IP Address to Access an Application in a Cluster | Kubernetes](https://kubernetes.io/docs/tutorials/stateless-application/expose-external-ip-address/) |

### Deploy version mới của ứng dụng

Phần này sẽ được áp dụng khi một version mới của ứng dụng được cập nhật, chúng ta sẽ build và deploy Docker Image mới của ứng dụng lên GKE Cluster trước đó.

GKE cung cấp tính năng rolling update để chúng ta có thể update Deployment mà không xảy ra “thời gian chết”. Khi một Deployment được cập nhật, thì Deployment dừng lại các Pod, scale lại số lượng Pod về 0, sau đó sử dụng template mới của Pod để tạo lại Pod, Pod cũ không xóa hẳn cho đến khi Pod mới đang chạy, quá trình này diễn ra đến đâu có thể xem bằng lệnh kubectl describe deploy/<deploy-name>. Cập nhật như vậy nó đảm bảo luôn có Pod đang chạy khi đang cập nhật.

Các bước thực hiện như sau:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Update repository khi có thay đổi | Ví dụ ở đây là dùng Github repository, vì vậy, chúng ta cd đến vị trí của repository đã push trước đó trên Artifact Registry, rồi pull code mới về  cd ApiSampleApp  git pull |
| 2 | Build và tag Docker image của ứng dụng với version mới | docker build -t ***${REGION}***-docker.pkg.dev/**${PROJECT\_ID}**/***${REPOSITORY\_NAME}***/**${APP\_NAME}**:v2 . |
| 3 | Push docker image mới lên AR | docker push ***${REGION}***-docker.pkg.dev/**${PROJECT\_ID}**/***${REPOSITORY\_NAME}***/**${APP\_NAME}**:v2 |
| 4 | Áp các cập nhật lên Deployment tồn tại trước đó | kubectl set image deployment/**${APP\_NAME}** **${APP\_NAME}**=***${REGION}***-docker.pkg.dev/**${PROJECT\_ID}**/***${REPOSITORY\_NAME}***/**${APP\_NAME}**:v2  Để xem tiến trình run pod như thế nào:  watch kubectl get pods |
| Output | | |
|  | | |

## Kết quả thực hiện

Graphical user interface, text, application

Description automatically generated

# : Deploy ASP.NET Framework Application lên Google Cloud Platform với Kubernetes Engine.

## Chuẩn bị cho việc deploy

Để phục vụ cho việc deploy một ứng dụng .NET Framework lên GCP với KBE ta cần chuẩn bị những thứ như sau:

Vì .NET Framework chỉ chạy trên môi trường Windows, vì vậy chúng ta sẽ thực hiện deploy trên CMD local và chuẩn bị các thứ như sau:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Ghi chú |
| 1 | Google Cloud Project (Tài khoản enable billing account) |  |
| 2 | Enable các API cần thiết (bao gồm: Artifact Registry API, Kubernetes Engine API) | Để enable các API trên, [nhấp vào đây](https://console.cloud.google.com/flows/enableapi?apiid=artifactregistry.googleapis.com,container.googleapis.com&_ga=2.150453238.1734173899.1626057467-586884957.1611153413&_gac=1.12323782.1625379545.CjwKCAjwlYCHBhAQEiwA4K21m0zCg3oc8ZjtJS6IIe5DcSNnN0zxD1HLgfvvsmkKTx9V2KxXMTJkcRoC1UgQAvD_BwE) |
| 3 | Sample Code mẫu:  ***Mô tả:*** Ứng dụng mẫu mà chúng ta sẽ deploy là một ASP.NET Framework 4.5 API Application, đã được container hoá ứng dụng bằng Docker. | Sample được lưu trữ trên Github [tại đây](https://github.com/iammin99/docker-aspnet.git) |
| 4 | Cài Docker Desktop |  |
| 5 | Cài Cloud SDK | Install CloudSDK [tại đây](https://cloud.google.com/sdk/docs/quickstarts) |
| 6 | Cài Kubernetes command-line tool | Sau khi cài ClouSDK, chúng ta install Kubernetes CLI bằng dòng lệnh:  gcloud components install kubectl |

## Các bước thực hiện

### Khai báo biến môi trường

|  |  |  |  |
| --- | --- | --- | --- |
| ENVIRONMENT VARIABLE | | | |
| **KEY** | **VALUE (EXAMPLE)** | **DESCRIPTION** | **CLI** |
| ***PROJECT\_ID*** | kubl-deploy-pj | Project trong Google Cloud Console | set PROJECT\_ID=**kubl-deploy-pj**  **------**  **gcloud config set project *%PROJECT\_ID%*** |
| ***REPOSITORY\_NAME*** | my-api-sample-repo | Tên repository | setREPOSITORY\_NAME=**my-api-sample-repo** |
| ***APP\_NAME*** | api-sample-app | Tên tag tao image trong Docker | set APP\_NAME=**api-sample-app** |
| ***CLUSTER\_NAME*** | aspnet-cluster | Tên cluster Google Kubernetes Engine (GKE) | set CLUSTER\_NAME=**aspnet-cluster** |
| ***GKE\_SERVICE\_NAME*** | api-sample-app-service | Tên service deployment GKE | set GKE\_SERVICE\_NAME=**api-sample-app-service** |
| ***REGION*** | asia-southeast1 | Region sẽ set | set REGION=**asia-southeast1** |
| ***ZONE*** | asia-southeast1-a | Zone sẽ set | set ZONE=**asia-southeast1-a** |

### Set các cấu hình mặc định cho gcloud

|  |  |
| --- | --- |
| Set default project | gcloud config set project **%PROJECT\_ID%** |
| Set default zone | gcloud config set compute/zone **%ZONE%** |
| Set default region | gcloud config set compute/region **%REGION%** |

### Build và Run project ở local

Đầu tiên, chúng ta sẽ build ứng dụng và thử run bằng Docker Desktop ở local:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Build container image cho ứng dụng | docker build -t ***%REPOSITORY\_NAME%***/**%APP\_NAME%**:v1 . |
| 2 | Run container image của ứng dụng | docker run -d -p 8080:80 ***%REPOSITORY\_NAME%***/**%APP\_NAME%**:v1 |
| Output | | |
|  | | |

### Tạo GKE Cluster

Tiếp theo, chúng ta tạo GKE cluster để chạy ứng dụng. GCP cung cấp 2 loại cluster là Standard Cluster và Autopilot Cluster, ở tutorial này chúng ta sẽ demo trên Standard Cluster.

|  |
| --- |
| **Heart with solid fillGhi chú:** Để quyết định chọn Standard Cluster hay Autopilot Cluster của GKE. Chúng ta tham khảo một số nguồn sau:  (1) [Autopilot overview  |  Kubernetes Engine Documentation  |  Google Cloud](https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview?authuser=2&_ga=2.132187658.-586884957.1611153413&_gac=1.116072820.1625379545.CjwKCAjwlYCHBhAQEiwA4K21m0zCg3oc8ZjtJS6IIe5DcSNnN0zxD1HLgfvvsmkKTx9V2KxXMTJkcRoC1UgQAvD_BwE)  (2) [Pricing  |  Google Kubernetes Engine (GKE)  |  Google Cloud](https://cloud.google.com/kubernetes-engine/pricing) |

Các bước thực hiện:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Tạo cluster | **Standard Cluster**  gcloud container clusters create ***%CLUSTER\_NAME%*** --cluster-version **1.20.7** --enable-ip-alias --num-nodes **1** --zone ***%ZONE%***--service-account **"chumin99@kubl-deploy-pj.iam.gserviceaccount.com"** |
| **Autopilot Cluster**  gcloud container clusters create-auto ***%CLUSTER\_NAME%*** --cluster-version **1.20.7** --enable-ip-alias --num-nodes **1** --zone ***%ZONE%***--service-account **"chumin99@kubl-deploy-pj.iam.gserviceaccount.com"** |
| Output | | |
| A picture containing text  Description automatically generated | | |

### Push hoặc pull image lên Google Container Registry

Ở tutorial này, vì chúng ta sẽ deploy trên môi trường local, chúng ta sẽ sử dụng Container Registry (CR) để store Docker Image thay vì Artifact Registry, vì điều đó là không cần thiết.

Các bước thực hiện có thể tham khảo như sau:

Để push hoặc pull image từ local lên GCR chúng ta sẽ dùng Docker hoặc third-party tool nào đó, cần phải tag nó với tên registry rồi push image đó lên.

#### Xác thực đến Google Container Registry

Để push hoặc pull image từ private registry, yêu cầu chúng ta phải xác thực đến GCR với các permission tối thiểu bao gồm:

* ***Storage Object Viewer*** để pull image
* ***Storage Legacy Bucket Writer*** để pull và push image

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Tạo một GCP IAM Service Account (Khuyến khích) | **C1: Chạy bằng Command Line** |
| **# Tạo GCP Service Account; format của account là một địa chỉ email**  SA\_EMAIL=$(gcloud iam service-accounts --format='value(email)' create k8s-gcr-auth-ro)  **# Tạo json key file và liên kết với service account vừa tạo**  gcloud iam service-accounts keys create k8s-gcr-auth-ro.json --iam-account=$SA\_EMAIL  **# Lấy Project ID**  PROJECT=$(gcloud config list core/project --format='value(core.project)')  **# Bổ sung các ràng buộc IAM Policy cho service account cũng như project đã được xác định trước đó**  gcloud projects add-iam-policy-binding $PROJECT --member serviceAccount:$SA\_EMAIL --role **roles/storage.objectAdmin** |
| **C2: Thực hiện ở Google Cloud Console** |
| **B1:** Ở trang GCP, vào trang Service Accounts của dịch vụ IAM & Admin. [Nhấn vào đây](https://console.cloud.google.com/iam-admin/serviceaccounts?_ga=2.158711802.1734173899.1626057467-586884957.1611153413&_gac=1.213585382.1625379545.CjwKCAjwlYCHBhAQEiwA4K21m0zCg3oc8ZjtJS6IIe5DcSNnN0zxD1HLgfvvsmkKTx9V2KxXMTJkcRoC1UgQAvD_BwE)  **B2:** Chọn Google Cloud Project đang thực hiện, ở đây chúng ta chọn project kubl-deploy-project  **B3:** Nhấn vào nút Create Service Account  **B4:** Thêm quyền cho service account vừa tạo. Các role cần thêm bao gồm:  1. Storage Object Admin  2. Kubernetes Engine Admin (roles/container.admin)  **B5:** Nhấn Done là hoàn thành việc tạo Service Account.  **B6:** Nhấn vào thông tin Service Account vừa tạo. Vào tab Keys và nhấn Create New Key, chọn dạng file .json và download về local. |
| 2 | Cấu hình Docker để có thể access vào Container Registry [Dùng gcloud credential helper] | 1. **Đăng nhập vào gcloud bằng user mà chúng ta sẽ chạy các Docker command** (Ở đây chúng ta sử dụng service account vừa tạo ở trên)   gcloud auth activate-service-account **ACCOUNT** --key-file=**KEY-FILE**  **Trong đó:**   * **ACCOUNT** là service account mình vừa tạo.   **chumin99@kubl-deploy-pj.iam.gserviceaccount.com**   * **KEY-FILE** là service account key file vừa download xuống local ở bước trước   **kubl-deploy-pj-iam-key.json**   1. **Cấu hình Docker với dòng lệnh sau:**   gcloud auth configure-docker  Credentials sẽ được lưu ở %USERPROFILE%/.docker/config.json đối với Windows. |
| Output | | |
| Text  Description automatically generated | | |

#### Thêm Kubernetes Secret vào Kubernetes Cluster

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Thêm Kubernetes Secret vào Kubernetes Cluster | set ***SECRET\_NAME***= *gcr-json-key*  *kubectl create secret docker-registry* ***%SECRET\_NAME%*** *--docker-server=****https://asia.gcr.io*** *--docker-username=* ***\_json\_key*** *--docker-email=****<email-address>*** *--docker-password="$(cat* **kubl-deploy-pj-iam-key.json**  *)"*  Trong đó:   |  |  | | --- | --- | | SECRET\_NAME | Tên Secret sẽ tạo | | docker-server | Nên set là “[https://asia.gcr.io](https://gcr.io/)”  Tham khảo: <https://cloud.google.com/container-registry/docs/pushing-and-pulling> | | docker-username | Phải set là \_json\_key | | docker-email | Email address nào cũng được, miễn hợp lệ, tuy không dùng đến nhưng bắt buộc | | docker-password | Là content bên trong file json key mà chúng ta vừa tạo ở bước trước. | |
| 2 | Để xem content bên trong Secret vừa tạo | Ta chạy lệnh sau:  kubectl get secret ***%SECRET\_NAME%*** --output=yaml |
| Output | | |
| Text  Description automatically generated | | |

#### Sử dụng Secret cho Deployment

Có 2 cách để sử dụng Secret này, bao gồm:

1. Add Secret vào ImagePullSecrets trong service account mặc định của Kubernetes namespace. Với phương thức này, mỗi một pod được deploy sẽ sử dụng Secret khi pull image lên.
2. Một cách khác nữa là, mình sẽ add Secret riêng trực tiếp vào file cấu hình deployment cho từng pod nào mà cần đến nó.

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | **Option 1:** Add Secret vào ImagePullSecrets trong service account mặc định | Chạy dòng lệnh sau:  *kubectl patch serviceaccount default -p "{\"imagePullSecrets\": [{\"name\":"****%SECRET\_NAME%****\"}]}"* |
| 2 | **Option 2:** Add Secret vào file cấu hình Deployment của từng Pod, bằng cách khai báo value cho property ***imagePullSecrets*** | Ta tham khảo edit cấu hình file .yaml như sau:   |  | | --- | | apiVersion: v1  kind: Pod  metadata:    name: <pod\_name>  spec:    containers:      - name: <container\_name>        image: gcr.io/<registry\_name>/<image\_name>:<tagname>    imagePullSecrets:        - name: your\_secret\_name |   Ta có sample\_deployment.yaml như sau:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | |  | apiVersion: v1 | |  | kind: Pod | |  | metadata: | |  | name: aspnetmvc | |  | spec: | |  | containers: | |  | - name: aspnetmvc | |  | image: asia.gcr.io/kubl-deploy-project/aspnetmvc:v1 | |  | imagePullSecrets: | |  | - name: gcr-json-key | |  |  | | |
| Output | | |
|  | | |

|  |
| --- |
| **Heart with solid fillGhi chú**: Ở đây chúng ta sẽ chọn thực hiện theo cách 1. Vì image của chúng ta sẽ lưu trữ trên default registry của GCR. Khi nào chúng ta cần triển khai mỗi pod là một image từ registry khác nhau thì sẽ cân nhắc cách thứ 2. |

#### Push hoặc pull image lên Container Registry

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Tag local image với registry name | docker tag ***%REPOSITORY\_NAME%***/**%APP\_NAME%**:v1 asia.gcr.io/***%PROJECT\_ID%***/**%APP\_NAME%**:v1 |
| 2 | Push image đã được tag lên Container Registry | docker push asia.gcr.io/***%PROJECT\_ID%***/**%APP\_NAME%**:v1 |
| 3 | (Optional)  Pull image từ Container Registry về local | docker pull asia.gcr.io/***%PROJECT\_ID%***/**%APP\_NAME%**:v1 |
| Output | | |
|  | | |

### Tạo node pool windows

Để chạy container Windows Server, cluster của chúng ta phải có ít nhất một node pool Windows và một node pool Linux. Không thể tạo một cluster mà chỉ dùng một Windows Server node pool. Linux node pool được yêu cầu ở đây dùng để run các cluster add-on quan trọng.

Vì tính quan trọng của nó, chúng ta không được resize Linux node pool xuống 0 và đảm bảo nó có đủ dung lượng để chạy các cluster add-on.

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Kiểm tra xem đã kết nối với GKE Cluster chưa | gcloud container clusters get-credentials ***%CLUSTER\_NAME%*** --zone ***%ZONE%*** |
| 1 | Tạo node pool windows | gcloud container node-pools create **NODE\_POOL\_NAME**  --cluster=**CLUSTER\_NAME** --image-type=**IMAGE\_NAME** --no-enable-autoupgrade  --machine-type=**MACHINE\_TYPE\_NAME**  **Trong đó:**   |  |  | | --- | --- | | **NODE\_POOL\_NAME** | Tên của Windows Server node pool sẽ tạo | | **CLUSTER\_NAME** | Tên của cluster mà chúng ta đã tạo ở bước 2.1.1 | | **IMAGE\_NAME** | Chúng ta xác định một trong các giá trị sau:   * **WINDOWS\_LTSC\_CONTAINERD**: Windows Server LTSC with Containerd * **WINDOWS\_SAC\_CONTAINERD**: Windows Server SAC with Containerd * **WINDOWS\_LTSC:** Windows Server LTSC with Docker * **WINDOWS\_SAC:** Windows Server SAC with Docker | | **MACHINE\_TYPE\_NAME** | Định nghĩa machine type sử dụng. n1-standard-2 là machine type nhỏ nhất được recommend sử dụng như Windows Server node. Machine type như f1-micro và g1-small không support trong trường hợp này. Để thêm thông tin, tham khảo [tại đây](https://cloud.google.com/compute/pricing#standard_machine_types) |   Theo tutorial này, chạy dòng lệnh sau để tiếp tục:  gcloud container node-pools create **windows-pool** --cluster=***%CLUSTER\_NAME%*** --image-type=**WINDOWS\_LTSC** --no-enable-autoupgrade --machine-type=**n1-standard-2** |
| Output | | |
|  | | |

|  |
| --- |
| **Heart with solid fillGhi chú:** Để hiểu thêm về việc quyết định chọn Window Server Node Image nào có thể tham khảo tại đây: [Creating a cluster using Windows Server node pools (google.com)](https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-cluster-windows#choose_your_windows_server_node_image) |

|  |
| --- |
| **Heart with solid fillGhi chú:** Nên chọn image type **WINDOWS\_LTSC** vì nó sẽ giúp chúng ta tránh tình trạng version không tương thích khi upgrade node pool. Còn **WINDOWS\_SAC** không ổn định, phải cập nhật thường xuyên, nếu không sẽ không tương thích (**WINDOWS\_SAC** chỉ tương thích 18 tháng sau version đó phát hành) |

### Deploy ứng dụng lên GKE

Tiếp theo, chúng ta sẽ tạo một Kubernetes Deployment để run application trên cluster vừa tạo. Set số lượng replicas (pods) cho nó. Một pod chỉ chứa một container là docker image ứng dụng của chúng ta.

Sau đó, chúng ta sẽ tạo HorizontalPodAutoscaler (HPA) và cấu hình auto scale nhằm đáp ứng nhu cầu sử dụng.

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Kiểm tra xem có kết nối với GKE Cluster chưa | gcloud container clusters get-credentials ***%CLUSTER\_NAME%*** --zone ***%ZONE%*** |
| 2 | Viết file cấu hình deployment .yaml | Ta viết file cấu hình .yaml tham khảo như sau:  [Link file](https://github.com/iammin99/docker-aspnet/blob/394b67b77baf0cd8efe81f2fe77f76d5a28b3b1f/kube-deployment.yml): kube-deployment.yaml |
| 3 | Apply file cấu hình cho Kubernetes | * Trỏ đến thư mục chứa file .yaml (nếu chưa trỏ đến – dùng lệnh cd) * Apply file .yaml   kubectl apply -f **kube\_deployment.yaml** |
| Xem thêm | Tạo HPA & cấu hình auto scale | 1. Scale thay đổi chỉ số replica (số lương Pod) của Deployment. VD: set số lượng pods lên là 2  kubectl scale deployment ***%{APP\_NAME}*** --replicas=**2**  2. Tạo HorizontalPodAutoscaler cho Deployment. Ví dụ, ta thiết lập scale tự động với số lượng POD trong khoảng min = 1, max = 5 và thực hiện scale khi CPU của POD hoạt động ở mức 50% thì thực hiện  kubectl autoscale deployment ***%{APP\_NAME}*** --cpu-percent=**50** --min=**1** --max=**5** |
| Xem thêm | Xem thông tin và thông tin chi tiết pod vừa tạo. | 1. Xem thông tin pod được tạo  kubectl get pods  2. Xem thông tin chi tiết một pod  kubectl describe pods <ten-pod> |
| Output | | |
| Text  Description automatically generated | | |

### Expose ứng dụng lên Internet

Service type mặc định của GKE là ClusterIP, nó nhận một địa chỉ IP chỉ co thể liên lạc bên trong cluster. Vì thế, để expose ra ngoài cluster, chúng ta sẽ tạo một Service với type là LoadBalancer. Loại service này sẽ tạo ra một IP cân bằng tải cho các pods, có thể truy cập trên Internet.

Các bước thực hiện:

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Tạo Kubernetes Service cho Deployment | kubectl expose deployment ***${APP\_NAME}*** --name= ***${GKE\_SERVICE\_NAME}*** --type=**LoadBalancer** --port **80** --target-port **8080**  ***Chú thích:***  --port 80: số port cấu hình trên Load Balancer  --target-port 8080: số port là container của ứng dụng lắng nghe. |
| 2 | Xem thông tin & thông tin chi tiết Service vừa tạo | 1. Xem thông tin service được tạo  kubectl get service  2. Xem thông tin chi tiết service vừa tạo  kubectl describe service ***${GKE\_SERVICE\_NAME}*** |
| Output | | |
|  | | |

### Deploy version mới của ứng dụng

|  |  |  |
| --- | --- | --- |
| STT | Công việc | Thực hiện |
| 1 | Update repository khi có thay đổi | Pull code mới về ở local  cd ApiSampleApp  git pull |
| 2 | Build và tag Docker image của ứng dụng với version mới | docker build -t ***%REPOSITORY\_NAME%***/**%APP\_NAME%**:v2 . |
| 3 | Push docker image mới lên CR | docker push asia.gcr.io/***%PROJECT\_ID%***/**%APP\_NAME%**:v2 |
| 4 | Áp các cập nhật lên Deployment tồn tại trước đó | kubectl set image deployment/**${APP\_NAME}** **${APP\_NAME}**= asia.gcr.io/***%PROJECT\_ID%***/**%APP\_NAME%**:v2  ***Cách khác:*** Cập nhật image trong file .yaml sau đó apply  Để xem tiến trình run pod như thế nào:  watch kubectl get pods |
| Output | | |
|  | | |

## Kết quả thực hiện

Sau khi deploy thành công, ứng dụng sẽ được expose qua đường dẫn và số port mà chúng ta đã config ở bước [2.2.5](#_Expose_ứng_dụng)

**Link truy cập:** <http://34.126.181.225:8080>

Graphical user interface, text

Description automatically generated