# **Stable-Diffusion on Agones 部署**

本指南为Stable-Diffusion-WebUI用户提供了简单的步骤，以使用 GCP GKE 服务启动Stable-Diffusion-WebUI部署，并使用 Filestore 作为模型和输出文件的共享存储。 为了方便多用户的Stable-Diffusion运行时管理，我们使用了 [Agones](https://agones.dev/site/) 作为运行时管理的控制平面，每个独立的Stable-Diffusion-WebUI都托管在一个独立的 POD 中，每个授权用户 将分配一个专用 POD。 用户只需按照步骤运行Stable-Diffusion-WebUI即可。

## **介绍**

本项目使用[Stable-Diffusion-WebUI](https://github.com/AUTOMATIC1111/stable-diffusion-webui)开源作为用户交互前端，客户只需准备Stable-Diffusion模型即可构建/通过容器部署Stable-Diffusion模型。 该项目使用Cloud Build来帮助您使用Stable-Diffusion模型快速构建 docker 镜像，然后您可以基于 docker 镜像进行部署。 为了提供多用户隔离的Stable-Diffusion运行时，使用 [Agones](https://agones.dev/site/) 作为运行时管理的控制平面，Agones 根据用户的资源需求管理和控制Stable-Diffusion运行时的生命周期。

## **新版Agones 部署步骤:**

### **设置环境**

1. 准备一个GCP项目
2. 使用 [cloud shell](https://cloud.google.com/shell/docs/run-gcloud-commands) 启用相关服务API
3. 下载最新版代码

| gcloud services enable compute.googleapis.com artifactregistry.googleapis.com container.googleapis.com file.googleapis.com vpcaccess.googleapis.com redis.googleapis.com cloudscheduler.googleapis.com  git clone https://github.com/nonokangwei/Stable-Diffusion-on-GCP.git -b Stable-Diffusion-on-GCP-X  cd Stable-Diffusion-on-GCP |
| --- |

### **创建GKE集群**

使用下方命令创建GKE集群,在创建一个额外的GPU节点池并开启自动扩展

创建集群时候可以指定集群使用的VPC和子网,后续的资源也将使用该网络

第二条命令--accelerator 可以设置集群的[GPU](https://cloud.google.com/compute/docs/gpus)资源类型和时间分片功能,

| PROJECT\_ID=<replace this with your project id> GKE\_CLUSTER\_NAME=<replace this with your GKE cluster name> REGION=<replace this with your region>  ZONE=<replace this with your zone>  VPC\_NETWORK=<replace this with your vpc network name> VPC\_SUBNETWORK=<replace this with your vpc subnetwork name>  gcloud container clusters create $GKE\_CLUSTER\_NAME \  --project ${PROJECT\_ID} \  --region ${REGION} \  --no-enable-basic-auth \  --release-channel "stable" \  --machine-type "e2-standard-2" \  --image-type "COS\_CONTAINERD" \  --disk-type "pd-balanced" \  --disk-size "100" \  --metadata disable-legacy-endpoints=true \  --scopes "https://www.googleapis.com/auth/cloud-platform" \  --num-nodes "1" \  --logging=SYSTEM,WORKLOAD,API\_SERVER,SCHEDULER,CONTROLLER\_MANAGER \  --monitoring=SYSTEM,API\_SERVER,SCHEDULER,CONTROLLER\_MANAGER \  --enable-ip-alias \  --network "projects/${PROJECT\_ID}/global/networks/${VPC\_NETWORK}" \  --subnetwork "projects/${PROJECT\_ID}/regions/${REGION}/subnetworks/${VPC\_SUBNETWORK}" \  --no-enable-intra-node-visibility \  --default-max-pods-per-node "110" \  --no-enable-master-authorized-networks \  --addons HorizontalPodAutoscaling,HttpLoadBalancing,GcePersistentDiskCsiDriver,GcpFilestoreCsiDriver \  --autoscaling-profile optimize-utilization  gcloud container node-pools create gpu-spot-pool \  --cluster ${GKE\_CLUSTER\_NAME} \  --region ${REGION} \  --node-locations ${ZONE} \  --spot \  --machine-type "g2-standard-12" \  --accelerator "type=nvidia-l4,count=1,gpu-sharing-strategy=time-sharing,max-shared-clients-per-gpu=2" \  --image-type "COS\_CONTAINERD" \  --disk-type "pd-balanced" \  --disk-size "100" \  --metadata disable-legacy-endpoints=true \  --scopes "https://www.googleapis.com/auth/cloud-platform" \  --enable-autoscaling \  --total-min-nodes "0" \  --total-max-nodes "20" \  --location-policy "ANY" \  --enable-autoupgrade |
| --- |

**设置防火墙规则**

防火墙规则放行agones流量

| gcloud compute firewall-rules create allow-agones \  --network=${VPC\_NETWORK} \  --allow tcp:7000-8000 \  --source-ranges=0.0.0.0/0 |
| --- |

### **获取 GKE 令牌**

| gcloud container clusters get-credentials ${GKE\_CLUSTER\_NAME} --region ${REGION} |
| --- |

### **安装GPU驱动**

| kubectl apply -f https://raw.githubusercontent.com/GoogleCloudPlatform/container-engine-accelerators/master/nvidia-driver-installer/cos/daemonset-preloaded.yaml |
| --- |

**[可选]如果你使用了最新的GPU实例类型.例如 G2/L4,请使用下方新版GPU驱动**

| kubectl apply -f https://raw.githubusercontent.com/GoogleCloudPlatform/container-engine-accelerators/master/nvidia-driver-installer/cos/daemonset-preloaded-latest.yaml |
| --- |

### **创建容器镜像仓库**

| BUILD\_REGIST=<replace this with your preferred Artifacts repo name> gcloud artifacts repositories create ${BUILD\_REGIST} --repository-format=docker --location=${REGION} gcloud auth configure-docker ${REGION}-docker.pkg.dev |
| --- |

### **创建Filestore共享网络存储**

| FILESTORE\_NAME=<replace with filestore instance name> FILESTORE\_ZONE=<replace with filestore instance zone> FILESHARE\_NAME=<replace with file share name>  gcloud filestore instances create ${FILESTORE\_NAME} \  --zone=${FILESTORE\_ZONE} \  --tier=BASIC\_HDD \  --file-share=name=${FILESHARE\_NAME},capacity=1TB \  --network=name=${VPC\_NETWORK} |
| --- |

#### 初始化共享存储目录

创建一个容器或者一台虚拟机,挂载共享存储目录后,在根目录创建 sd-config 和 extensions 两个文件夹,sd-config文件夹存储用户配置,extensions文件夹存储插件,,所有的用户初始环境都会复制该目录到容器来启动webui.

共享存储目录如下:

| | --/ | --/sd-config | --/config.json | --/ui-config.json | --/extensions | -- | -- | --.. | --/<username> | --/config.json | --/ui-config.json |
| --- |

### **设置filestore pv 和 pvc**

替换*<nfs-server-ip>* 为filestore服务IP地址,

创建pv 和 pvc 以便stable diffusion webui 容器使用

| FILESTORE\_SERVER\_IP=<replace with filestore server ip>  sed -i 's/<nfs-server-ip>/$FILESTORE\_SERVER\_IP/g' ./Stable-Diffusion-UI-Agones/agones/nfs\_pv.yaml  kubectl apply -f ./Stable-Diffusion-UI-Agones/agones/nfs\_pv.yaml kubectl apply -f ./Stable-Diffusion-UI-Agones/agones/nfs\_pvc.yaml |
| --- |

### **使用helm安装agones到gke集群t**

安装agones组件到default-pool, 这个节点池会一直运行保证agones组件的可用性

helm已预装到cloudshell,可以使用cloudshell安装agones,

| helm repo add agones https://agones.dev/chart/stable  helm repo update  kubectl create namespace agones-system  helm install sd-agones-release --namespace agones-system -f ./Stable-Diffusion-UI-Agones/agones/values.yaml agones/agones |
| --- |

### **创建redis缓存**

创建resi缓存并记录redis 服务的IP地址,

完成后使用redis 服务地址设置REDIS\_HOST环境变量

| gcloud redis instances create --project=${PROJECT\_ID} sd-agones-cache --tier=standard --size=1 --region=${REGION} --redis-version=redis\_6\_x --network=projects/${PROJECT\_ID}/global/networks/${VPC\_NETWORK} --connect-mode=DIRECT\_PEERING  gcloud redis instances describe sd-agones-cache --region ${REGION} --format=json | jq .host  REDIS\_HOST=<replace with resis server ip> |
| --- |

### **编译基础镜像**

#### WebUI镜像

进入的Dockerfile所在的相对路径./Stable-Diffusion-UI-Agones/sd-webui/

使用docker 或者 cloud build 编译镜像,

本地编译镜像时需要较大的内存,和较好的网络,

如果是cloudshell机器资源有限,可能会失败,推荐使用cloud build 编译或创建一台内存和磁盘较大的机器

编译后把镜像推到容器仓库

| cd ./Stable-Diffusion-UI-Agones/sd-webui  docker build . -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-webui:0.1 docker push ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-webui:0.1  # you can also use cloud build to accelerate the image build sample command refer to follow  gcloud builds submit \  --machine-type=e2-highcpu-32 \  --disk-size=100 \  --region=us-central1 \  -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-webui:0.1 |
| --- |

#### Agones sidecar镜像

进入的Dockerfile所在的相对路径 ./Stable-Diffusion-UI-Agones/agones-sidecar,

使用docker 或者 cloud build 编译镜像,编译后把镜像推到容器仓库

| cd ./Stable-Diffusion-UI-Agones/agones-sidecar  # 本地编译镜像 docker build . -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-gs:0.1 docker push ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-gs:0.1  # cloud build编译镜像  gcloud builds submit \  --machine-type=e2-highcpu-32 \  --disk-size=100 \  --region=us-central1 \  -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-gs:0.1 |
| --- |

### **创建VPC 私有连接**

使用下方命令创建名称为 sd-agones-connector 的vpc 私有连接,名称可以修改,如需修改做好记录,后续部署cloud function 会使用该连接

| gcloud compute networks vpc-access connectors create sd-agones-connector --network ${VPC\_NETWORK} --region ${REGION} --range 192.168.240.16/28 |
| --- |

### **用户会话管理程序**

#### 创建Cloud Function

该程序会定期查看用户会话信息,清空限制的会话,回收资源,释放节点

用户会话过期时间默认为15分钟,可以设置TIME\_INTERVAL 环境变量指定用户会话过期时间

注意设置REDIS\_HOST环境变量,否则程序无法查询redis中的用户会话信息.

| cd ./Stable-Diffusion-UI-Agones/cloud-function  gcloud functions deploy redis\_http --runtime python310 --trigger-http --allow-unauthenticated --region=${REGION} --vpc-connector=sd-agones-connector --egress-settings=private-ranges-only --set-env-vars=REDIS\_HOST=${REDIS\_HOST}  # note the function trigger url  gcloud functions describe redis\_http --region us-central1 --format=json | jq .httpsTrigger.url |
| --- |

#### 创建程序调用定时器.

注意替换 ${FUNCTION\_URL} 内容为上一步创建cloud function trigger URL.

| gcloud scheduler jobs create http sd-agones-cruiser \  --location=${REGION} \  --schedule="\*/5 \* \* \* \*" \  --uri=${FUNCTION\_URL} |
| --- |

### **GS Controller 模块**

这个cloud function 用于处理stable diffusion 容器的创建请求,收到用户访问请求后会通过该模块完成stable diffusion webui 容器的创建和销毁.

#### 设置环境变量

通过命令找到GKE API server 公网IP并设置为环境变量 K8S\_ENDPOINT

| gcloud container clusters describe ${GKE\_CLUSTER\_NAME} --region us-central1 --format=json | jq .privateClusterConfig.publicEndpoint  K8S\_ENDPOINT=IP\_ADDRESS\_FROM\_OUTPUT |
| --- |

#### 更新/gs.yaml文件内容

文件路径: ./Stable-Diffusion-UI-Agones/cloud-function-gs-controller/localpackage/gs.yaml

替换文件中的镜像URL

33行 <Simple-Game-Server-Image>

42行 <Stable-Diffusion-WebUI-Image>

#### 创建 gs controller 函数

| cd ./Stable-Diffusion-UI-Agones/cloud-function-gs-controller/  gcloud functions deploy agones\_gs\_backend --runtime python310 --trigger-http --allow-unauthenticated --region=${REGION} --set-env-vars=k8s\_endpoint=${K8S\_ENDPOINT} |
| --- |

### **State Controller模块**

这个cloud function用于管理stable diffusion 容器的状态,收集agones-relay-http 容器发来的状态事件,并更新redis存储的容器状态值

.

| cd ./Stable-Diffusion-UI-Agones/cloud-function-state-controller gcloud functions deploy agones\_listener\_http --runtime python310 --trigger-http --allow-unauthenticated --region=${REGION} --vpc-connector=sd-agones-connector --egress-settings=private-ranges-only --set-env-vars=REDIS\_HOST=${REDIS\_HOST} |
| --- |

### **Agones管理的容器状态上报**

用于收集agones容器事件,上报状态更新到State Controller函数

#### 编译agones-relay-http 镜像

| cd ./Stable-Diffusion-on-GCP/Stable-Diffusion-UI-Agones/agones-relay-http docker build . -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/agones-relay-http:0.1 docker push ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/agones-relay-http:0.1  gcloud builds submit \  --machine-type=e2-highcpu-32 \  --disk-size=100 \  --region=us-central1 \  -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/agones-relay-http:0.1 |
| --- |

#### 获取函数State Controller的调用URL并用于部署文件

| gcloud functions describe agones\_listener\_http --region us-central1 --format=json | jq .httpsTrigger.url |
| --- |

#### 更新 yaml 文件后部署

使用之前命令输出的内容替换文件 install.yaml 中 *<agones-relay-http-image>* 和 *<cloud-function-state-controller-url>*

文件路径: ./Stable-Diffusion-UI-Agones/agones-relay-http/deploy/install.yaml

文件中需要替换的部署:

* 55行 替换 <agones-relay-http-image> 为上一步编译的镜像URL
* 63行 替换 <cloud-function-state-controller-url> 为函数State Controller调用的URL

使用kubectl 部署资源到GKE

| kubectl apply -f ./Stable-Diffusion-UI-Agones/agones-relay-http/deploy/install.yaml |
| --- |

### **Nginx proxy模块**

#### 编译nginx镜像

进入的Dockerfile所在的相对路径 ./Stable-Diffusion-on-GCP/Stable-Diffusion-UI-Agones/nginx

使用docker 或者 cloud build 编译镜像,

注意替换 sd.lua 文件中21 *${REDIS\_HOST}* 和53 *${agones\_gs\_backend}* 内容为redis 服务地址和函数GS Controller调用URL.

| gcloud functions describe agones\_gs\_backend --region us-central1 --format=json | jq .httpsTrigger.url |
| --- |

| cd ./Stable-Diffusion-UI-Agones/nginx  docker build . -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-nginx:0.1 docker push ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-nginx:0.1  gcloud builds submit \  --machine-type=e2-highcpu-32 \  --disk-size=100 \  --region=us-central1 \  -t ${REGION}-docker.pkg.dev/${PROJECT\_ID}/${BUILD\_REGIST}/sd-nginx:0.1 |
| --- |

#### 部署到GKE

please replace the image URL in the deployment.yaml line 19

| kubectl apply -f ./Stable-Diffusion-UI-Agones/nginx/deployment.yaml |
| --- |

### 

### **部署负载均衡和IAP**

为了用户可以通过负载均衡访问在GKE上部署的stable diffusion 容器,并且可以授权指定用户访问.使用IAP作为授权管理的组件

#### 配置OAuth consent screen 和 OAuth 凭据,

参考[OAuth配置](https://cloud.google.com/iap/docs/enabling-kubernetes-howto#oauth-configure).完成凭据配置

#### 为负载均衡预留固定IP.

***在你自己拥有的域名解析服务中创建一条A记录指向这个公网IP地址***

| gcloud compute addresses create sd-agones --global gcloud compute addresses describe sd-agones --global --format=json | jq .address |
| --- |

#### 创建Kubernetes资源.

* 创建kubernetes 密钥,注意使用你刚创建的凭据替换 *client\_id* 和 *client\_secret*.
* 修改 managed-cert.yaml 文件,替换 *DOMAIN\_NAME1* 为 ***上一步A记录使用的子域名***
* 然后部署下列kubernetes资源(managed-cert , backendconfig , service , ingress)

| kubectl create secret generic iap-secret --from-literal=client\_id=client\_id\_key \  --from-literal=client\_secret=client\_secret\_key  kubectl apply -f ./ingress-iap/managed-cert.yaml kubectl apply -f ./ingress-iap/backendconfig.yaml kubectl apply -f ./ingress-iap/service.yaml kubectl apply -f ./ingress-iap/ingress.yaml |
| --- |

创建完毕后根据[文档](https://cloud.google.com/iap/docs/enabling-kubernetes-howto#iap-access)说明授于 ***IAP-secured Web App User*** 角色后,用户即可访问stable diffusion webui

## FAQ:

#### 共享存储目录结构

初始化之后的目录结构示例,启动容器之前

├── accounts.google.com.xxxxxxxx

│ ├── extensions

│ ├── outputs

│ └── sd-config

│ ├── config.json

│ └── ui-config.json

├── extensions

│ ├── put extensions here.txt

│ └── sd\_dreambooth\_extension

├── models

│ ├── Stable-diffusion

│ │ ├── v1-5-pruned-emaonly.civitai.info

│ │ ├── v1-5-pruned-emaonly.preview.png

│ │ └── v1-5-pruned-emaonly.safetensors

└── sd-config

├── config.json

└── ui-config.json

#### 查看容器是否就绪

kubectl get po

kubectl get gs



#### 设置容器的资源分配策略

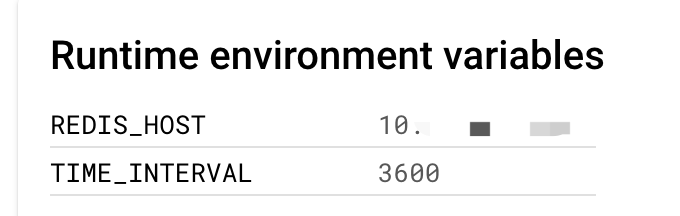
kubectl get po

kubectl get gs



#### 设置用户会话超时时间

设置redis\_http cloud function 环境变量 新加一条 TIME\_INTERVAL 数值单位为秒



1. 下