All credit for this doc: Victor Yang

Table of Contents

    - route
    - firewall rules
    - layer 4 network lb
    - layer 7 http lb
    - forwarding-rules
    - address
- GCP managed ssl certificate
- StackDriver logging
- Service
    - list service available
    - Enable Service
- Client libraries you can use to connect to Google APIs
- chaining gcloud commands
- one liner to purge GCR images given a date
- GKE
- Cloud Run
- Machine Learning
- Deployment Manager

References

- have fun with them (https://cloudplatform.googleblog.com/2016/06/filtering-and-formatting-fun-with.html)
- projections (https://cloud.google.com/sdk/gcloud/reference/topic/projections)
- filters (https://cloud.google.com/sdk/gcloud/reference/topic/filters)
- resource-keys (https://cloud.google.com/sdk/gcloud/reference/topic/resource-keys)
- scripting-gcloud (https://cloud.google.com/sdk/docs/scripting-gcloud)
- gcloud alpha interactive (http://cloudplatform.googleblog.com/2018/03/introducing-GCPs-new-interactive-CLI.html)
- https://medium.com/@Joachim8675309/getting-started-with-gcloud-sdk-part-1-114924737 (https://medium.com/@Joachim8675309/getting-started-with-gcloud-sdk-part-1-114924737)
- https://medium.com/@Joachim8675309/getting-started-with-gcloud-sdk-part-2-4d049a656f1a (https://medium.com/@Joachim8675309/getting-started-with-gcloud-sdk-part-2-4d049a656f1a)
- https://gist.github.com/bborysenko/97749fe0514b819a5a87611e6aea3db8 (https://gist.github.com/bborysenko/97749fe0514b819a5a87611e6aea3db8)

Other cheatsheets

- https://github.com/dennyzhang/cheatsheet-gcp-A4 (https://github.com/dennyzhang/cheatsheet-gcp-A4)

multiple gcloud config configurations

- https://www.jhanley.com/google-cloud-understanding-gcloud-configurations/ (https://www.jhanley.com/google-cloud-understanding-gcloud-configurations/)
- https://medium.com/infrastructure-adventures/working-with-multiple-environment-in-gcloud-cli-93b2d4e8cf1e (https://medium.com/infrastructure-adventures/working-with-multiple-environment-in-gcloud-cli-93b2d4e8cf1e)

```
gcloud config configurations create pythonrocks
gcloud config configurations list
gcloud config configurations activate pythonrocks
gcloud config set core/account pythonrocks@gmail.com
gcloud auth login
gcloud projects list
gcloud config set project dev-193420
```

switch gcloud context with gcloud config

```
gcloud config list
gcloud config set account pythonrocksk8s201702@gmail.com
gcloud config set project salt-163215
gcloud config set compute/region us-west1
gcloud config set compute/zone us-west1-a
alias demo='gcloud config set account pythonrocksk8s201702@gmail.com && gcloud config set project
 salt-163215 && gcloud config set compute/region us-west1 && gcloud config set compute/zone us-we
st1-a'


cluster=$(gcloud config get-value container/cluster 2> /dev/null)
zone=$(gcloud config get-value compute/zone 2> /dev/null)
project=$(gcloud config get-value core/project 2> /dev/null)

# switch project based on the name
gcloud config set project $(gcloud projects list --filter='name:wordpress-dev' --format='value(p
roject_id)')
```

```
command -v gcloud >/dev/null 2>&1 || { \
 echo >&2 "I require gcloud but it's not installed.  Aborting."; exit 1; }

REGION=$(gcloud config get-value compute/region)
if [[ -z "${REGION}" ]]; then
    echo "https://cloud.google.com/compute/docs/regions-zones/changing-default-zone-region" 1>&2
    echo "gcloud cli must be configured with a default region." 1>&2
    echo "run 'gcloud config set compute/region REGION'." 1>&2
    echo "replace 'REGION' with the region name like us-west1." 1>&2
    exit 1;
fi
```

auth

```
gcloud auth list
gcloud auth login
gcloud auth activate-service-account --key-file=sa_key.json
```

kubectl uses OAuth token generated by

- `gcloud config config-helper --format json`
- `gcloud config config-helper --format='value(credential.access_token)'`
- `gcloud auth print-access-token` generates new token

info

```
gcloud info --format flattened
```

```
export PROJECT=$(gcloud info --format='value(config.project)')
```

projects

```
# various way to get project_id
PROJECT_ID=$(gcloud config get-value core/project)
PROJECT_ID=$(gcloud config list project --format='value(core.project)')
PROJECT_ID=$(gcloud info --format='value(config.project)')

# get project_number given project_id or name
gcloud projects list --filter="project_id:${project_id}"  --format='value(project_number)'
gcloud projects list --filter="name:${project_name}"  --format='value(project_number)'
```

zones & regions

To return a list of zones given a region

```
gcloud compute zones list --filter=region:us-central1
```

```
# list regions
gcloud compute regions list
```

billing

```
gcloud beta billing accounts list
gcloud organizations list
```

IAM list permission and roles for a given resource

```
gcloud iam list-testable-permissions <uri>
e.g gcloud iam list-testable-permissions //cloudresourcemanager.googleapis.com/projects/$PROJECT
_ID

gcloud iam list-grantable-roles <uri>
e.g.
gcloud iam list-grantable-roles //cloudresourcemanager.googleapis.com/projects/$PROJECT_ID
gcloud iam list-grantable-roles https://www.googleapis.com/compute/v1/projects/$PROJECT_ID/zones
/us-central1-a/instances/iowa1

# get uri e.g.
gcloud projects list --uri
```

IAM service account

```
export SA_EMAIL=$(gcloud iam service-accounts list \
    --filter="displayName:jenkins" --format='value(email)')
export PROJECT=$(gcloud info --format='value(config.project)')

# creaate and list sa
gcloud iam service-accounts create jenkins --display-name jenkins
gcloud iam service-accounts list
gcloud iam service-accounts list   --filter='email ~ [0-9]*-compute@.*'   --format='table(email)
'

# create & list sa key
gcloud iam service-accounts keys create jenkins-sa.json --iam-account $SA_EMAIL
gcloud iam service-accounts keys list --iam-account=vault-admin@<project_id>.iam.gserviceaccount
.com

# project level: grant roles to sa
gcloud projects get-iam-policy $PROJECT
gcloud projects add-iam-policy-binding $PROJECT  --role roles/storage.admin \
    --member serviceAccount:$SA_EMAIL
gcloud projects add-iam-policy-binding $PROJECT --role roles/compute.instanceAdmin.v1 \
    --member serviceAccount:$SA_EMAIL
gcloud projects add-iam-policy-binding $PROJECT --role roles/compute.networkAdmin \
    --member serviceAccount:$SA_EMAIL
gcloud projects add-iam-policy-binding $PROJECT --role roles/compute.securityAdmin \
    --member serviceAccount:$SA_EMAIL
gcloud projects add-iam-policy-binding $PROJECT --role roles/iam.serviceAccountActor \
    --member serviceAccount:$SA_EMAIL
```

- When granting IAM roles, you can treat a service account either as a resource or as an identity (https://cloud.google.com/iam/docs/granting-roles-to-service-accounts)

```
# service account level: add role to service account
gcloud iam service-accounts get-iam-policy <sa_email>
gcloud iam service-accounts add-iam-policy-binding infrastructure@retviews-154908.iam.gserviceac
count.com --member='serviceAccount:infrastructure@retviews-154908.iam.gserviceaccount.com' --rol
e='roles/iam.serviceAccountActor'
```

- https://cloud.google.com/iam/docs/creating-short-lived-service-account-credentials (https://cloud.google.com/iam/docs/creating-short-lived-service-account-credentials)
- https://medium.com/@tanujbolisetty/gcp-impersonate-service-accounts-36eaa247f87c (https://medium.com/@tanujbolisetty/gcp-impersonate-service-accounts-36eaa247f87c)
- https://medium.com/wescale/how-to-generate-and-use-temporary-credentials-on-google-cloud-platform-b425ef95a00d (https://medium.com/wescale/how-to-generate-and-use-temporary-credentials-on-google-cloud-platform-b425ef95a00d)
- https://cloud.google.com/iam/credentials/reference/rest/v1/projects.serviceAccounts/generateAccessToken (https://cloud.google.com/iam/credentials/reference/rest/v1/projects.serviceAccounts/generateAccessToken) shows the lifetime of the OAuth token of 3600 seconds by default

```
# user:godevopsrocks@gmail.com impersonate as a svc account terraform@${PROJECT_ID}.iam.gservicea
ccount.com
gcloud iam service-accounts add-iam-policy-binding  terraform@${PROJECT_ID}.iam.gserviceaccount.
com --member=user:godevopsrocks@gmail.com --role roles/iam.serviceAccountTokenCreator
gcloud container clusters list --impersonate-service-account=terraform@${PROJECT_ID}.iam.gservic
eaccount.com
```

GCS bucket level

```
COMPUTE_ENGINE_SA_EMAIL=$(gcloud iam service-accounts list --filter="name:Compute Engine default
 service account" --format "value(email)")
gsutil iam ch serviceAccount:${COMPUTE_ENGINE_SA_EMAIL}:objectViewer gs://bucket-name
```

Custom Roles

```
# list predefined roles
gcloud iam roles list
# list custom roles
gcloud iam roles list --project $PROJECT_ID

# create custom role in the following 2 ways, either on project level (--project [PROJECT_ID]) o
r org level (--organization [ORGANIZATION_ID])
1. gcloud iam roles create editor --project $PROJECT_ID --file role-definition.yaml
2. gcloud iam roles create viewer --project $PROJECT_ID --title "Role Viewer" --description "Cus
tom role description." --permissions compute.instances.get,compu
te.instances.list --stage ALPHA
```

app engine

- https://medium.com/google-cloud/app-engine-project-cleanup-9647296e796a (https://medium.com/google-cloud/app-engine-project-cleanup-9647296e796a)

cloud build

```
# user defined
gcloud builds submit --config=cloudbuild.yaml --substitutions=_BRANCH_NAME=foo,_BUILD_NUMBER=1 .

# override built in TAG_NAME
gcloud builds submit --config=cloudbuild.yaml --substitutions=TAG_NAME=v1.0.1
```

Cloud build trigger GCE rolling replace/start

- https://medium.com/google-cloud/continuous-delivery-in-google-cloud-platform-cloud-build-with-compute-engine-a95bf4fd1821 (https://medium.com/google-cloud/continuous-delivery-in-google-cloud-platform-cloud-build-with-compute-engine-a95bf4fd1821)
- https://cloud.google.com/compute/docs/instance-groups/updating-managed-instance-groups#performing_a_rolling_replace_or_restart (https://cloud.google.com/compute/docs/instance-groups/updating-managed-instance-groups#performing_a_rolling_replace_or_restart)

```
steps:
- name: 'gcr.io/cloud-builders/docker'
  args: [ 'build', '-t', 'gcr.io/$PROJECT_ID/gcp-cloudbuild-gce-angular', '.' ]
- name: 'gcr.io/cloud-builders/gcloud'
  args: [ 'beta', 'compute', 'instance-groups', 'managed', 'rolling-action', 'restart', 'gce-angu
lar-instance-group', '--zone=us-east1-b' ]
images:
- 'gcr.io/$PROJECT_ID/gcp-cloudbuild-gce-angular'
```

kms

- cloud-encrypt-with-kms (https://codelabs.developers.google.com/codelabs/cloud-encrypt-with-kms/#0)
- Integrated with cloud build (https://cloud.google.com/cloud-build/docs/securing-builds/use-encrypted-secrets-credentials)

```
# list all keyrings
gcloud kms keyrings list --location global
# list all keys in my_key_ring
gcloud kms keys list --keyring my_key_ring --location global
```

```
# grant KMS IAM permission to a sv account $USER_EMAIL
gcloud kms keyrings add-iam-policy-binding $KEYRING_NAME \
    --location global \
    --member user:$USER_EMAIL \
    --role roles/cloudkms.admin
gcloud kms keyrings add-iam-policy-binding $KEYRING_NAME \
    --location global \
    --member user:$USER_EMAIL \
    --role roles/cloudkms.cryptoKeyEncrypterDecrypter

# Encrypt and Decrypt in REST API
curl -v "https://cloudkms.googleapis.com/v1/projects/$DEVSHELL_PROJECT_ID/locations/global/keyRin
gs/$KEYRING_NAME/cryptoKeys/$CRYPTOKEY_NAME:encrypt" \
  -d "{\"plaintext\":\"$PLAINTEXT\"}" \
  -H "Authorization:Bearer $(gcloud auth application-default print-access-token)"\
  -H "Content-Type:application/json" \
| jq .ciphertext -r > 1.encrypted

curl -v "https://cloudkms.googleapis.com/v1/projects/$DEVSHELL_PROJECT_ID/locations/global/keyRin
gs/$KEYRING_NAME/cryptoKeys/$CRYPTOKEY_NAME:decrypt" \
  -d "{\"ciphertext\":\"$(cat 1.encrypted)\"}" \
  -H "Authorization:Bearer $(gcloud auth application-default print-access-token)"\
  -H "Content-Type:application/json" \
| jq .plaintext -r | base64 -d
```

compute engine

gcloud command for creating an instance?

from web console

```
gcloud compute instances create [INSTANCE_NAME] \
  --image-family [IMAGE_FAMILY] \
  --image-project [IMAGE_PROJECT] \
  --create-disk image=[DISK_IMAGE],image-project=[DISK_IMAGE_PROJECT],size=[SIZE_GB],type=[DISK_
TYPE]

gcloud compute instances create micro1 --zone=us-west1-a --machine-type=f1-micro --subnet=defa
ult --network-tier=PREMIUM --maintenance-policy=MIGRATE --service-account=398028291895-compute
@developer.gserviceaccount.com --scopes=https://www.googleapis.com/auth/devstorage.read_only,htt
ps://www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https
://www.googleapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.read
only,https://www.googleapis.com/auth/trace.append --min-cpu-platform=Automatic --image=debian-9-s
tretch-v20180510 --image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard
--boot-disk-device-name=micro1
```

list compute images

```
gcloud compute images list --filter=name:debian --uri
https://www.googleapis.com/compute/v1/projects/debian-cloud/global/images/debian-8-jessie-v201801
09
https://www.googleapis.com/compute/v1/projects/debian-cloud/global/images/debian-9-stretch-v20180
105

# Use the following command to see available non-Shielded VM Windows Server images
```

```
gcloud compute images list --project windows-cloud --no-standard-images
# Use the following command to see a list of available Shielded VM images, including Windows imag
es
gcloud compute images list --project gce-uefi-images --no-standard-images
```

list an instance

- filters (https://cloud.google.com/sdk/gcloud/reference/topic/filters)
- resource-keys (https://cloud.google.com/sdk/gcloud/reference/topic/resource-keys)

```
gcloud compute instances list --filter="zone:us-central1-a"
gcloud compute instances list --project=dev --filter="name~^es"
gcloud compute instances list --project=dev --filter=name:kafka --format="value(name,INTERNAL_I
P)"
gcloud compute instances list --filter=tags:kafka-node
gcloud compute instances list --filter='machineType:g1-small'
```

move instance

`gcloud compute instances move <instance_wanna_move> --destination-zone=us-central1-a --zone=us-central1-c`

ssh & scp

```
#--verbosity=debug is great for debugging, showing the SSH command
# the following is a real word example for running a bastion server that talks to a GKE cluster
(master authorized network)
gcloud compute ssh --verbosity=debug <instance_name> --command "kubectl get nodes"

gcloud compute scp  --recurse ../manifest <instance_name>:
```

SSH via IAP

- https://cloud.google.com/iap/docs/using-tcp-forwarding (https://cloud.google.com/iap/docs/using-tcp-forwarding)

```
# find out access-config-name's name
gcloud compute instances describe oregon1
# remove the external IP
gcloud compute instances delete-access-config  oregon1 --access-config-name "External NAT"
# connect via IAP, assuming the IAP is granted to the account used for login.
gcloud beta compute ssh oregon1 --tunnel-through-iap
```

ssh port forwarding for elasticsearch

```
gcloud compute --project "foo" ssh --zone "us-central1-c" "elasticsearch-1"  --ssh-flag="-L loca
lhost:9200:localhost:9200"
```

The 2nd `localhost` is relative to elasticsearch-1`

ssh reverse port forwarding

for example, how to connect to home server's flask server (tcp port 5000) for a demo or a local game server in development

```
GOOGLE_CLOUD_PROJECT=$(gcloud config get-value project)
gcloud compute --project "${GOOGLE_CLOUD_PROJECT}" ssh --zone "us-west1-c" --ssh-flag="-v -N -R
:5000:localhost:5000" "google_cloud_bastion_server"
```

generate ssh config

```
gcloud compute config-ssh
```

debugging

gcloud debugging: `gcloud compute instances list --log-http` serial port debug
(https://cloud.google.com/compute/docs/instances/interacting-with-serial-console)

instance level metadata

```
curl -s "http://metadata.google.internal/computeMetadata/v1/instance/?recursive=true&alt=text" -
H "Metadata-Flavor: Google"
leader=$(curl -s "http://metadata.google.internal/computeMetadata/v1/instance/attributes/leader"
 -H "Metadata-Flavor: Google")
```

project level metadata

```
gcloud compute project-info describe
gcloud compute project-info describe --flatten="commonInstanceMetadata[]"
```

instances, template, target-pool and instance group

```
cat << EOF > startup.sh
#! /bin/bash
apt-get update
apt-get install -y nginx
service nginx start
sed -i -- 's/nginx/Google Cloud Platform - '"\$HOSTNAME"'/' /var/www/html/index.nginx-debian.htm
l
EOF

gcloud compute instance-templates create nginx-template  --metadata-from-file startup-script=sta
rtup.sh
gcloud compute target-pools create nginx-pool
gcloud compute instance-groups managed create nginx-group \
         --base-instance-name nginx \
         --size 2 \
         --template nginx-template \
         --target-pool nginx-pool
```

MIG with startup and shutdown scripts

https://cloud.google.com/vpc/docs/special-configurations#multiple-natgateways
(https://cloud.google.com/vpc/docs/special-configurations#multiple-natgateways)

```
gsutil cp gs://nat-gw-template/startup.sh .

gcloud compute instance-templates create nat-1 \
    --machine-type n1-standard-2 --can-ip-forward --tags natgw \
    --metadata-from-file=startup-script=startup.sh --address $nat_1_ip

gcloud compute instance-templates create nat-2 \
    --machine-type n1-standard-2 --can-ip-forward --tags natgw \
    --metadata-from-file=startup-script=startup.sh  --address $nat_2_ip
```

disk snapshot

```
gcloud compute disks snapshot kafka-data1-1 --async --snapshot-names=kafka-data-1 --project pro
ject_a --zone us-west1-a
Use [gcloud compute operations describe URI] command to check the status of the operation(s).
```

regional disk

```
 gcloud beta compute instance attach-disk micro1 --disk pd-west1 --disk-scope regional
```

Networking

network and subnets

```
 gcloud compute networks create privatenet --subnet-mode=custom
 gcloud compute networks subnets create privatesubnet-us --network=privatenet --region=us-centra
l1 --range=172.16.0.0/24
 gcloud compute networks subnets create privatesubnet-eu --network=privatenet --region=europe-we
st1 --range=172.20.0.0/20
 gcloud compute networks subnets list --sort-by=NETWORK
```

route

tag the instances with `no-ips`

```
gcloud compute instances add-tags existing-instance --tags no-ip
gcloud compute routes create no-ip-internet-route \
    --network custom-network1 \
    --destination-range 0.0.0.0/0 \
    --next-hop-instance nat-gateway \
    --next-hop-instance-zone us-central1-a \
    --tags no-ip --priority 800
```

firewall rules

- https://medium.com/@swongra/protect-your-google-cloud-instances-with-firewall-rules-69cce960fba (https://medium.com/@swongra/protect-your-google-cloud-instances-with-firewall-rules-69cce960fba)

```
# allow SSH, RDP and ICMP for the given network
gcloud compute firewall-rules create managementnet-allow-icmp-ssh-rdp --direction=INGRESS --pri
ority=1000 --network=managementnet --action=ALLOW --rules=tcp:22,3389,icmp --source-ranges=0.0.
0.0/0
# allow internal from given source range
gcloud compute firewall-rules create mynetwork-allow-internal --network \
mynetwork --action ALLOW --direction INGRESS --rules all \
--source-ranges 10.128.0.0/9
gcloud compute firewall-rules list --filter="network:mynetwork"
```

```
## DENY
gcloud compute firewall-rules create mynetwork-deny-icmp \
--network mynetwork --action DENY --direction EGRESS --rules icmp \
--destination-ranges 10.132.0.2 --priority 500
gcloud compute firewall-rules list \
--filter="network:mynetwork AND name=mynetwork-deny-icmp"

# sort-by
gcloud compute firewall-rules list --sort-by=NETWORK
```

layer 4 network lb

```
gcloud compute firewall-rules create www-firewall --allow tcp:80
gcloud compute forwarding-rules create nginx-lb \
         --region us-central1 \
         --ports=80 \
         --target-pool nginx-pool

gcloud compute firewall-rules list --sort-by=NETWORK
```

layer 7 http lb

- https://cloud.google.com/solutions/scalable-and-resilient-apps (https://cloud.google.com/solutions/scalable-and-resilient-apps)

```
gcloud compute http-health-checks create http-basic-check
gcloud compute instance-groups managed \
       set-named-ports nginx-group \
       --named-ports http:80

gcloud compute backend-services create nginx-backend \
      --protocol HTTP --http-health-checks http-basic-check --global

gcloud compute backend-services add-backend nginx-backend \
    --instance-group nginx-group \
    --instance-group-zone us-central1-a \
    --global

gcloud compute url-maps create web-map \
    --default-service nginx-backend

gcloud compute target-http-proxies create http-lb-proxy \
    --url-map web-map

gcloud compute forwarding-rules create http-content-rule \
        --global \
        --target-http-proxy http-lb-proxy \
        --ports 80
gcloud compute forwarding-rules list
```

forwarding-rules

```
gcloud compute forwarding-rules list --filter=$(dig +short <dns_name>)
gcloud compute forwarding-rules describe my-forwardingrule --region us-central1
gcloud compute forwarding-rules describe my-http-forwardingrule --global
```

address

```
# get the external IP address of the instance
gcloud compute instances describe single-node \
     --format='value(networkInterfaces.accessConfigs[0].natIP)

gcloud compute addresses describe https-lb --global --format json

# list all IP addresses
gcloud projects list --format='value(project_id)' | xargs -I {} gcloud compute addresses list --f
ormat='value(address)' --project {}  2>/dev/null | sort | uniq -c
```

GCP managed ssl certificate

```
gcloud beta compute ssl-certificates create example-mydomain --domains example.mydomain.com
gcloud beta compute ssl-certificates list
gcloud beta compute ssl-certificates describe example-mydomain
# It takes 30mins+ to provision the TLS, one of conditions is the target-https-proxies needs to
be associated with the cert.
gcloud beta compute target-https-proxies list
```

StackDriver logging

```
gcloud logging read "timestamp >= \"2018-04-19T00:30:00Z\"  and logName=projects/${project_id}/lo
gs/requests and resource.type=http_load_balancer" --format="csv(httpRequest.remoteIp,httpRequest
.requestUrl,timestamp)" --project=${project_id}
```

Service

list service available

`gcloud services list --available`

Enable Service

```
# chain
gcloud services enable cloudapis.googleapis.com && \
cloudresourcemanager.googleapis.com && \
compute.googleapis.com

# or not chain
gcloud services enable container.googleapis.com
gcloud services enable containerregistry.googleapis.com
gcloud services enable cloudbuild.googleapis.com
gcloud services enable iam.googleapis.com
gcloud services enable logging.googleapis.com
gcloud services enable monitoring.googleapis.com
gcloud services enable storage-api.googleapis.com
gcloud services enable storage-component.googleapis.com
gcloud services enable sourcerepo.googleapis.com
```

```
function enable-service() {
  SERVICE=$1
  if [[ $(gcloud services list --format="value(serviceConfig.name)" \
                              --filter="serviceConfig.name:$SERVICE" 2>&1) != \
                              "$SERVICE" ]]; then
    echo "Enabling $SERVICE"
    gcloud services enable $SERVICE
  else
    echo "$SERVICE is already enabled"
  fi
}

enable-service container.googleapis.com
```

Client libraries you can use to connect to Google APIs

- https://medium.com/google-cloud/simple-google-api-auth-samples-for-service-accounts-installed-application-and-appengine-da30ee4648 (https://medium.com/google-cloud/simple-google-api-auth-samples-for-service-accounts-installed-application-and-appengine-da30ee4648)

chaining gcloud commands

```
gcloud compute forwarding-rules list --format 'value(NAME)' \
| xargs -I {}  gcloud compute forwarding-rules delete {}  --region us-west1 -q

gcloud projects list --format='value(project_id)' \
| xargs -I {} gcloud compute addresses list --format='value(address)' --project {}  2>/dev/null
| sort | uniq -c

gcloud compute instances list --filter=elasticsearch --format='value(NAME)' \
| xargs -I {} -p gcloud compute instances stop {}
gcloud compute instances list --filter=elasticsearch --format='value(INTERNAL_IP)' \
| xargs -I {} ssh {} "sudo chef-client"

# delete non default routes
gcloud compute routes list --filter="NOT network=default" --format='value(NAME)' \
| xargs -I {} gcloud compute routes delete -q {}
```

one liner to purge GCR images given a date

```
DATE=2018-10-01
IMAGE=<project_id>/<image_name>
gcloud container images list-tags gcr.io/$IMAGE --limit=unlimited --sort-by=TIMESTAMP   \
--filter="NOT tags:* AND timestamp.datetime < '${DATE}'" --format='get(digest)' | \
while read digest;do gcloud container images delete -q --force-delete-tags gcr.io/$IMAGE@$dig
est ;done
```

GKE

```
# create a private cluster
gcloud beta container clusters create private-cluster \
    --private-cluster \
    --master-ipv4-cidr 172.16.0.16/28 \
    --enable-ip-alias \
    --create-subnetwork ""
```

```
gcloud compute networks subnets create my-subnet \
    --network default \
    --range 10.0.4.0/22 \
    --enable-private-ip-google-access \
    --region us-central1 \
    --secondary-range my-svc-range=10.0.32.0/20,my-pod-range=10.4.0.0/14

gcloud beta container clusters create private-cluster2 \
    --private-cluster \
    --enable-ip-alias \
    --master-ipv4-cidr 172.16.0.32/28 \
    --subnetwork my-subnet \
    --services-secondary-range-name my-svc-range \
    --cluster-secondary-range-name my-pod-range

 gcloud container clusters update private-cluster2 \
    --enable-master-authorized-networks \
    --master-authorized-networks <external_ip_of_kubectl_instance>
```

```
# create a GKE cluster with CloudRun,Istio, HPA enabled
gcloud beta container clusters create run-gke \
  --addons HorizontalPodAutoscaling,HttpLoadBalancing,Istio,CloudRun \
  --scopes cloud-platform \
  --zone us-central1-a \
  --machine-type n1-standard-4 \
  --enable-stackdriver-kubernetes \
  --no-enable-ip-alias
```

```
# create a VPC native cluster
gcloud container clusters create k1 \
--network custom-ip-vpc --subnetwork subnet-alias \
--enable-ip-alias --cluster-ipv4-cidr=/16   --services-ipv4-cidr=/22
```

```
# get the GKE endpoint
gcloud container clusters describe mycluster --format='get(endpoint)'
```

```
# generate a ~/.kube/config for private cluster with private endpoint
gcloud container clusters get-credentials private-cluster --zone us-central1-a --internal-ip
```

Cloud Run

```
# deploy a service on Cloud Run in us-central1 and allow unauthenticated user
gcloud beta run deploy --image gcr.io/${PROJECT-ID}/helloworld --platform managed --region us-c
entral1 --allow-unauthenticated

# list services
gcloud beta run services list
# get endpoint url for a service
gcloud beta run services describe <service_name> --format="get(status.url)"
```

Machine Learning

```
brew install bat
gcloud ml language analyze-entities --content="Michelangelo Caravaggio, Italian painter, is know
n for 'The Calling of Saint Matthew'." | bat  -l json
```

Deployment Manager

- https://cloud.google.com/sdk/gcloud/reference/deployment-manager/deployments/ (https://cloud.google.com/sdk/gcloud/reference/deployment-manager/deployments/) Play with the commands for preview and cancel-preview.