Creating a Continuous Delivery Pipeline

Getting Started With Google Kubernetes Engine

Version 1.5

Google Cloud

Agenda

Introduction to Jenkins

Provisioning Jenkins

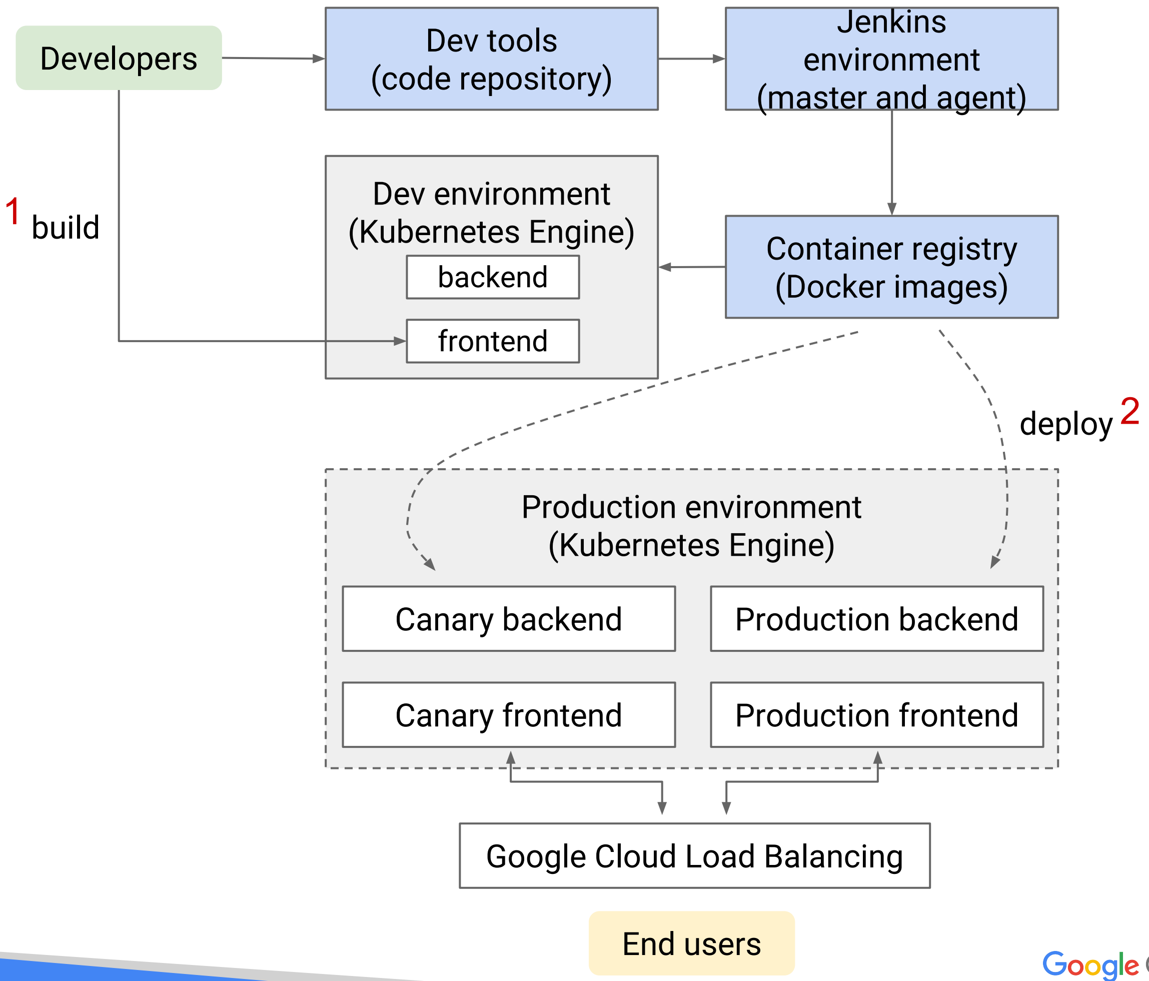
Understanding the application

Creating the Jenkins pipeline

Deploying a Canary release

Google Cloud

Here is the flow that you will go through with Jenkins



Developers
Dev tools (code repository)
Jenkins environment (master and agent)
1 build
Dev environment (Kubernetes Engine)
backend
frontend
Container registry (Docker images)
deploy 2
Production environment (Kubernetes Engine)
Canary backend
Production backend
Canary frontend
Production frontend
Google Cloud Load Balancing
End users

Agenda

Introduction to Jenkins

Provisioning Jenkins

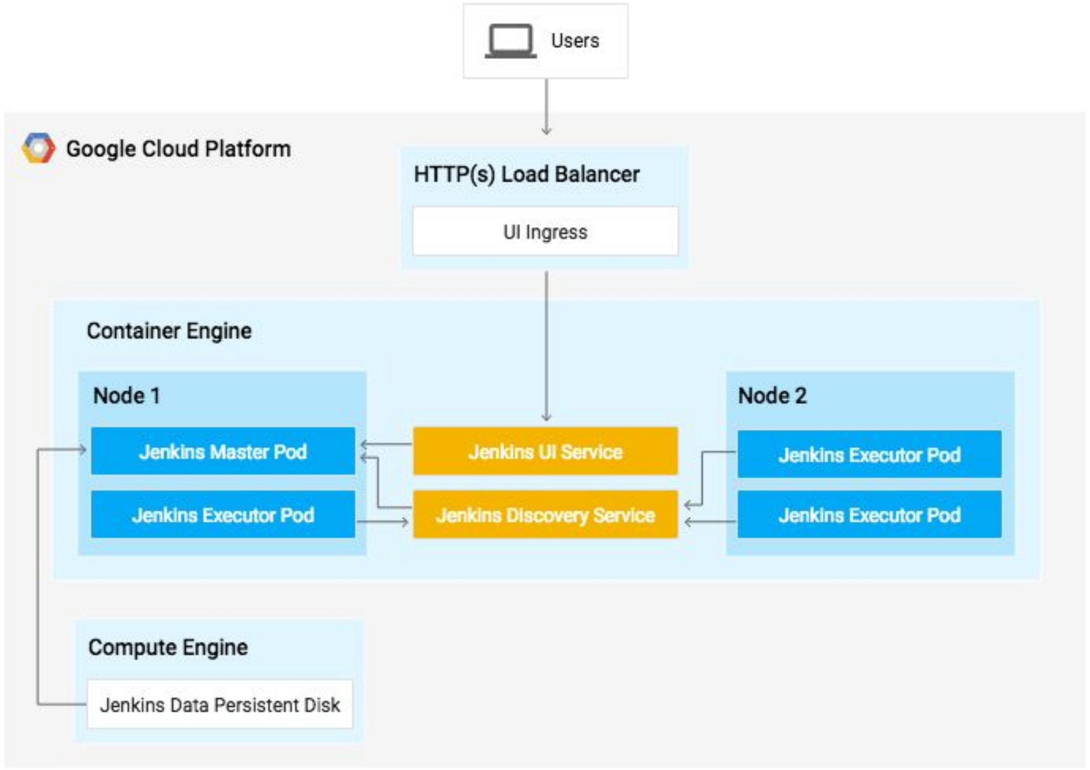
Understanding the application

Creating the Jenkins pipeline

Deploying a Canary release

Google Cloud

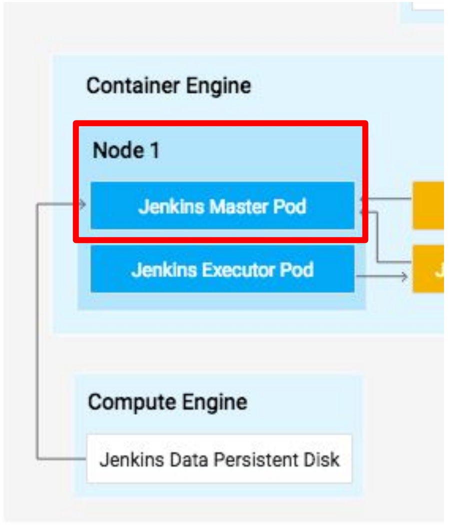
Here is how Jenkins gets deployed to Kubernetes



Jenkins is run through a Kubernetes deployment

For the master, you define:

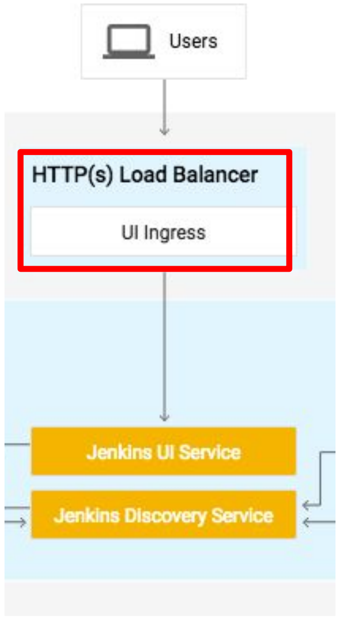
- 1 replica
- Image
- Ports
- Mount volume and path



```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: jenkins
  namespace: jenkins
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: master
    spec:
      containers:
      - name: master
        image: jenkins:1.642.4
        ports:
        - containerPort: 8080
        - containerPort: 50000
        env:
        volumeMounts:
        - mountPath: /var/jenkins_home
          name: jenkins-home
      volumes:
      - name: jenkins-home
        gcePersistentDisk:
          pdName: jenkins-home
          fsType: ext4
          partition: 1
```

For the ingress you define

- TLS cert secret
- Service name
- Service port



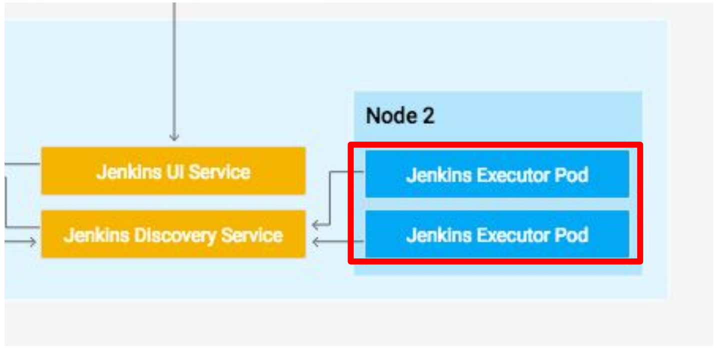
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: jenkins
  namespace: jenkins
spec:
  tls:
  - secretName: tls
  backend:
    serviceName: jenkins-ui
    servicePort: 8080
```

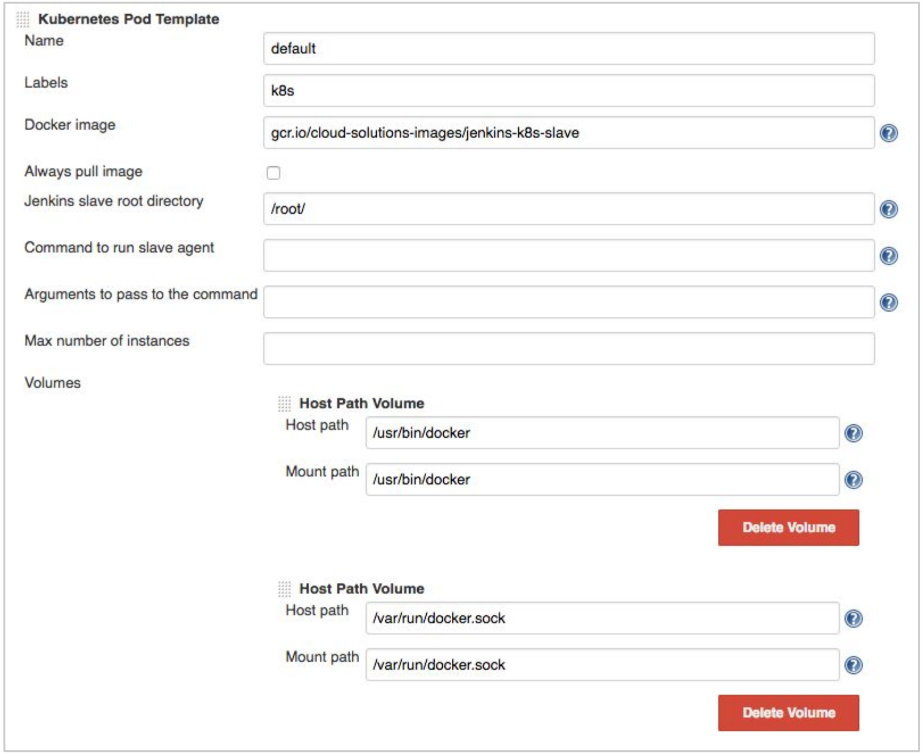
Google Cloud

The Jenkins executors (agents) are defined inside Jenkins

You define

- Your Docker image to run
- Docker binary/socket





Kubernetes Pod Template

Name	default
Labels	k8s
Docker image	gcr.io/cloud-solutions-images/jenkins-k8s-slave
Always pull image	☐
Jenkins slave root directory	/root/
Command to run slave agent	
Arguments to pass to the command	
Max number of instances	
Volumes	

Host Path Volume

Host path	/usr/bin/docker
Mount path	/usr/bin/docker

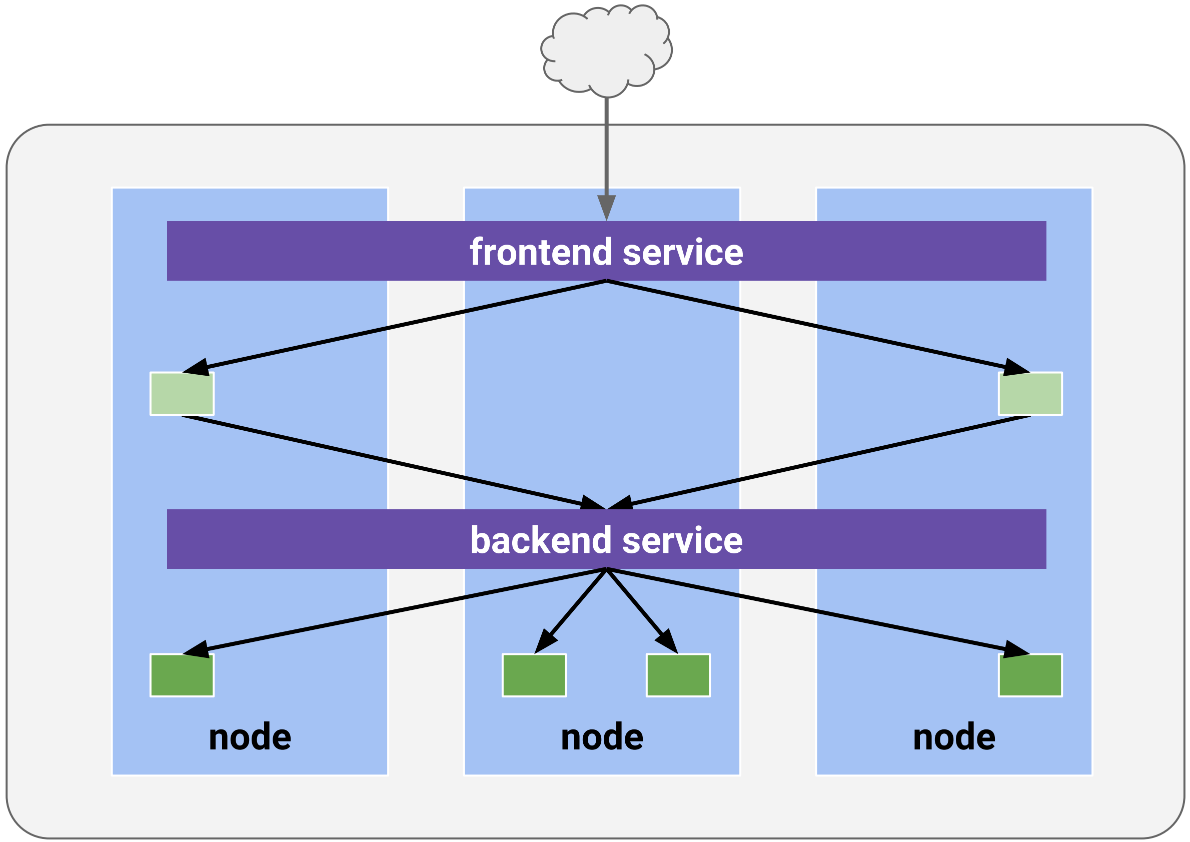
Host Path Volume

Host path	/var/run/docker.sock
Mount path	/var/run/docker.sock

Agenda

Google Cloud

The application has a frontend and backend (frontend exposed to the Internet)



Agenda

Introduction to Jenkins

Provisioning Jenkins

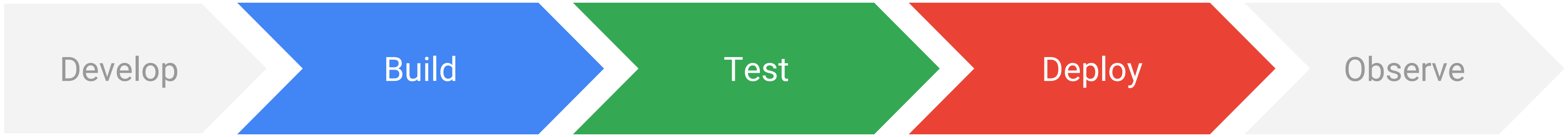
Understanding the application

Creating the Jenkins pipeline

Deploying a Canary release

Google Cloud

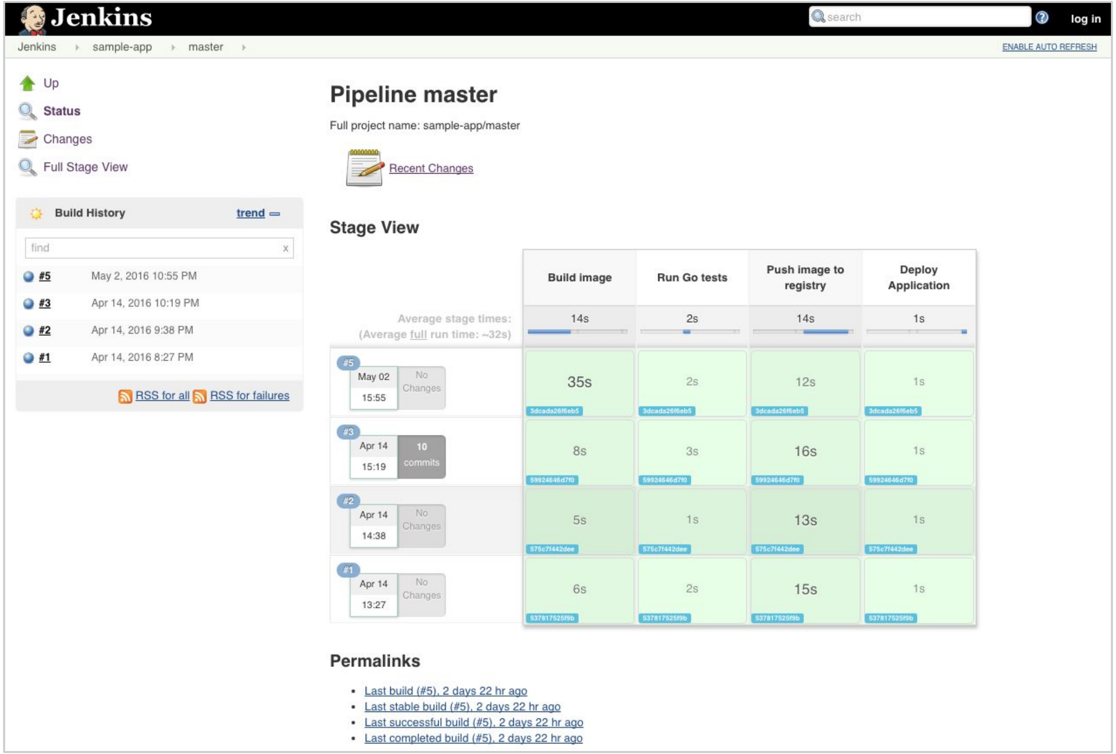
You build the Jenkins pipeline that defines how your build, test, and deploy cycle managed



Example Jenkins pipeline file with checkout, build, test, push, and deployment

```
node {
  def project = 'vic-goog'
  def appName = 'gceme'
  def feSvcName = "${appName}-frontend"
  def imageTag =
"gcr.io/${project}/${appName}:${env.BUILD_NUMBER}"

  checkout scm

  stage 'Build image'
  sh("docker build -t ${imageTag} .")

  stage 'Run Go tests'
  sh("docker run ${imageTag} go test")

  stage 'Push image to registry'
  sh("gcloud docker push ${imageTag}")

  stage "Deploy Application"
  sh("sed -i.bak 's#IMAGE_NAME#${imageTag}#' ./k8s/*.yaml")
  sh("kubectl --namespace=production apply -f k8s/")
}
```

A configured pipeline has run a few times with different stages, times, status, and logs



Jenkins › sample-app › master ›

ENABLE AUTO REFRESH

- Up
- Status
- Changes
- Full Stage View

Build History trend

#5	May 2, 2016 10:55 PM
#3	Apr 14, 2016 10:19 PM
#2	Apr 14, 2016 9:38 PM
#1	Apr 14, 2016 8:27 PM

RSS for all RSS for failures

Pipeline master

Full project name: sample-app/master

Recent Changes

Stage View

	Build image	Run Go tests	Push image to registry	Deploy Application
Average stage times: (Average full run time: ~32s)	14s	2s	14s	1s
#5 May 02 15:55 No Changes	35s 3dcada26f6eb5	2s 3dcada26f6eb5	12s 3dcada26f6eb5	1s 3dcada26f6eb5
#3 Apr 14 15:19 10 commits	8s 59924646d7f0	3s 59924646d7f0	16s 59924646d7f0	1s 59924646d7f0
#2 Apr 14 14:38 No Changes	5s 575c7f442dee	1s 575c7f442dee	13s 575c7f442dee	1s 575c7f442dee
#1 Apr 14 13:27 No Changes	6s 537817525f9b	2s 537817525f9b	15s 537817525f9b	1s 537817525f9b

Permalinks

- Last build (#5), 2 days 22 hr ago
- Last stable build (#5), 2 days 22 hr ago
- Last successful build (#5), 2 days 22 hr ago
- Last completed build (#5), 2 days 22 hr ago

Google Cloud

Agenda

Introduction to Jenkins

Provisioning Jenkins

Understanding the application

Creating the Jenkins pipeline

Deploying a Canary release

Google Cloud

With Canary, you have the same labels across deployments

```
kind: Service
apiVersion: v1
metadata:
  name: frontend
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: awesome-stuff
    role: frontend
```

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: frontend-prod
spec:
  replicas: 90
  template:
    metadata:
      name: frontend
      labels:
        app: awesome-stuff
        role: frontend
        env: prod
    spec:
      containers:
      - name: frontend
        image: my-img:v1
        ports:
        - name: ui
          containerPort: 80
```

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: frontend-staging
spec:
  replicas: 10
  template:
    metadata:
      name: frontend
      labels:
        app: awesome-stuff
        role: frontend
        env: staging
    spec:
      containers:
      - name: frontend
        image:my-img:v2
        ports:
        - name: ui
          containerPort: 80
```

But you have another label to distinguish production from staging

```yaml
kind: Service
apiVersion: v1
metadata:
  name: frontend
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: awesome-stuff
    role: frontend
```

```yaml
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: frontend-prod
spec:
  replicas: 90
  template:
    metadata:
      name: frontend
      labels:
        app: awesome-stuff
        role: frontend
        env: prod
    spec:
      containers:
      - name: frontend
        image: my-img:v1
        ports:
        - name: ui
          containerPort: 80
```

```yaml
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: frontend-staging
spec:
  replicas: 10
  template:
    metadata:
      name: frontend
      labels:
        app: awesome-stuff
        role: frontend
        env: staging
    spec:
      containers:
      - name: frontend
        image:my-img:v2
        ports:
        - name: ui
          containerPort: 80
```

Google Cloud

Lab