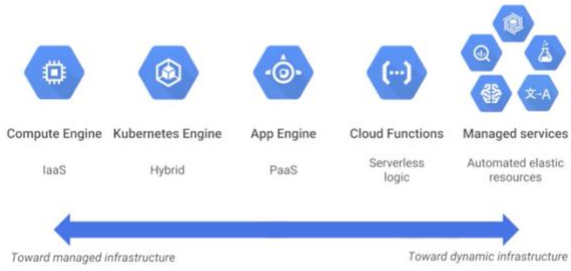
Google Cloud Fundamentals: Core Infrastructure

Introducing Google Cloud Platform

What is cloud computing?

GCP computing architectures meet you where you are



Google Cloud Platform is organized into regions and zones

Google offers customer-friendly pricing

Billing in sub-hour increments	Discounts for sustained use	Discounts for committed use	Discounts for preemptible use	Custom VM instance types
For compute, data processing and other services	Automatically applied to virtual machine use over 25% of a month	Pay less for steady, long-term workloads	Pay less for interruptible workloads	Pay only for the resources you need for your application

Security is designed into Google's technical infrastructure

Layer	Notable security measures (among others)
Operational security	Intrusion detection systems; techniques to reduce insider risk; employee U2F use; software development practices
Internet communication	Google Front End; designed-in Denial of Service protection
Storage services	Encryption at rest
User identity	Central identity service with support for U2F
Service deployment	Encryption of inter-service communication
Hardware infrastructure	Hardware design and provenance; secure boot stack; premises security

Quotas are helpful limits

Rate quota
GKE API: 1,000 requests per 100 seconds

Allocation quota
5 networks per project

Many quotas are changeable

Module introduction

Cloud security requires collaboration

- Google is responsible for managing its infrastructure security.
- You are responsible for securing your data.
- Google helps you with best practices, templates, products, and solutions.

Projects have three identifying attributes

Project ID	Globally unique	Chosen by you	Immutable
Project name	Need not be unique	Chosen by you	Mutable
Project number	Globally unique	Assigned by GCP	Immutable

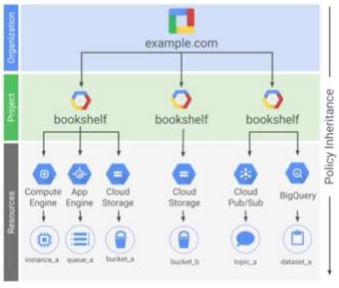
Resource hierarchy levels define trust boundaries

- Group your resources according to your organization structure.
- Levels of the hierarchy provide trust boundaries and resource isolation.

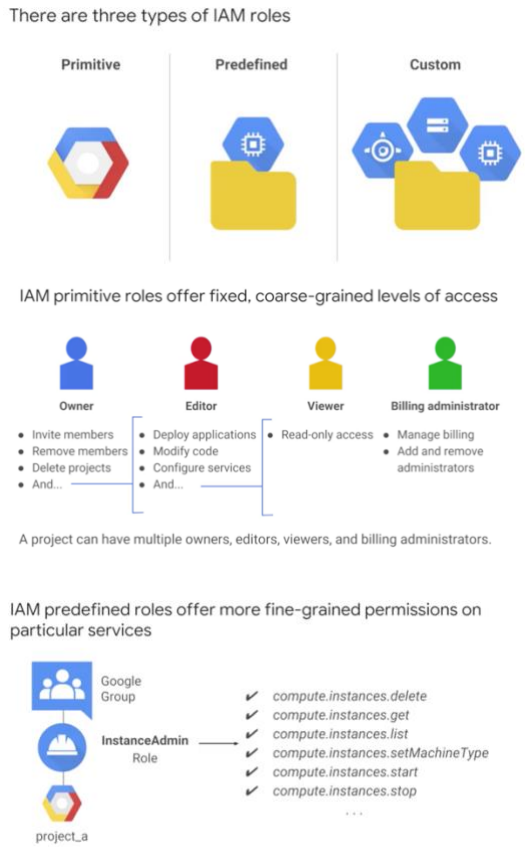
Inherit policy from top-down

An example IAM resource hierarchy

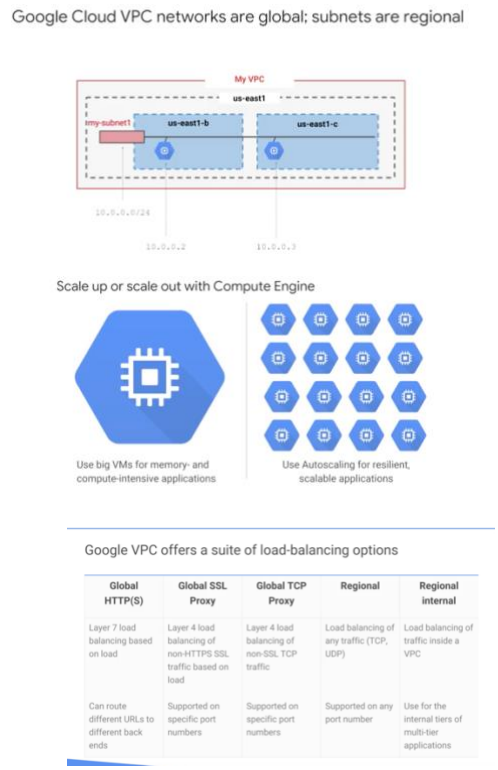
- A policy is set on a resource.
  - Each policy contains a set of roles and role members.
- Resources inherit policies from parent.
  - Resource policies are a union of parent and resource.
- A less restrictive parent policy overrides a more restrictive resource policy.



IAM Roles

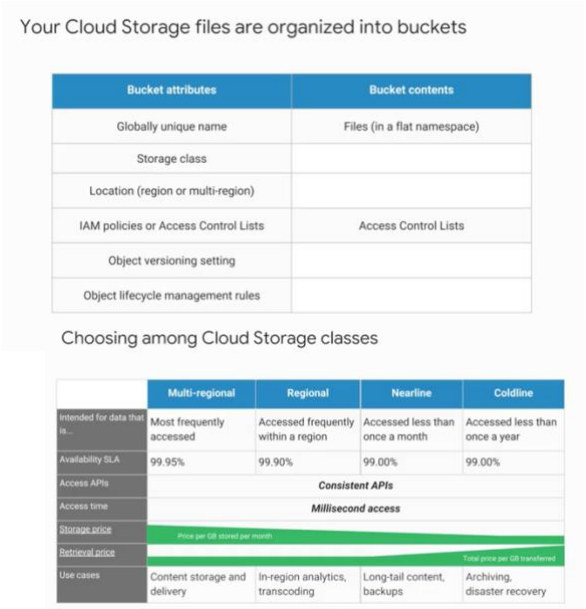


Virtual Private Cloud (VPC) Network & Compute Engine (VM)

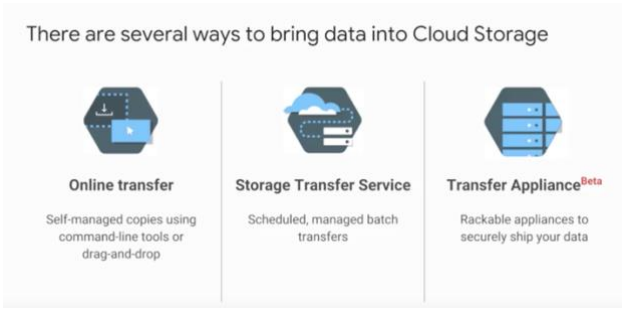


Google Cloud Platform Storage Options

Cloud Storage

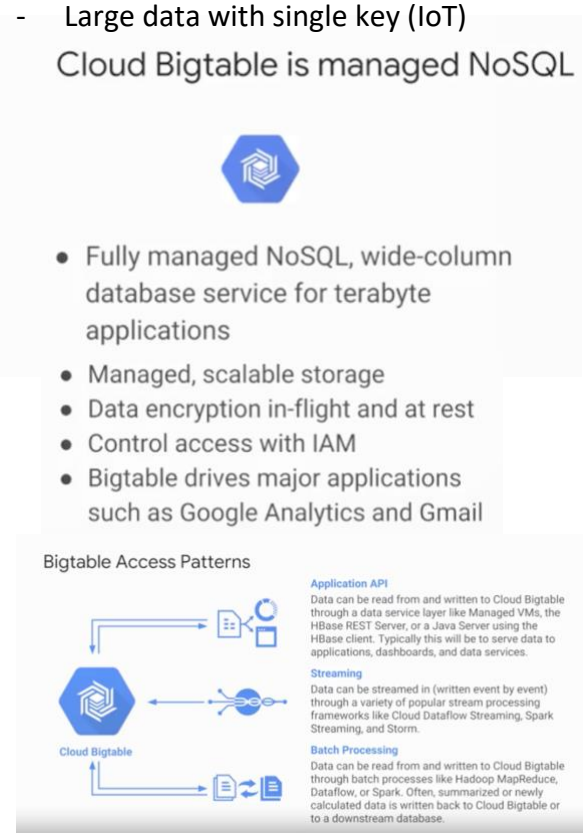


More Freq, High Storage Price → less freq access
Web --→ Kurbernetes

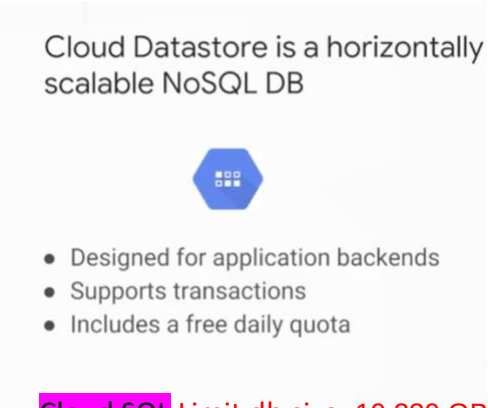


BigTable No-SQL

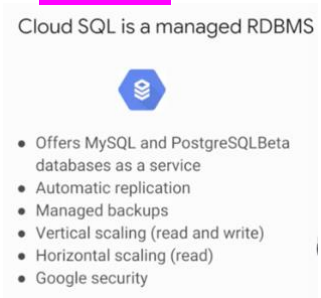
- Large data with single key (IoT)

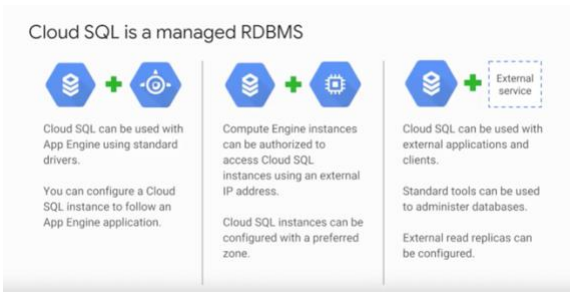


Cloud Datastore (horizonal scalability Nosql) but can do sql-like qu

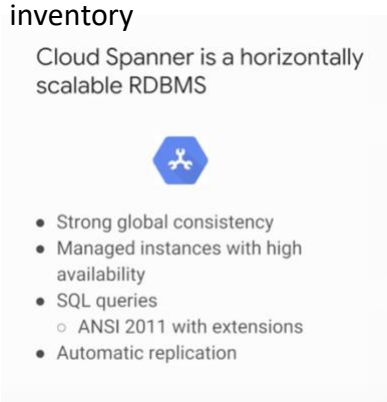


Cloud SQL Limit db size: 10,230 GB





Cloud Spanner (sql, horizonal scalability, global scale) petabyte database sizes
Outgrown db (alternative for cloud sql) ie: inventory



Comparing Storage Options

Comparing storage options: technical details

	Cloud Datastore	Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Transactions	Yes	Single-row	No	Yes	Yes	No
Complex queries	No	No	No	Yes	Yes	Yes
Capacity	Terabytes+	Petabytes+	Petabytes+	Terabytes	Petabytes	Petabytes+
Unit size	1 MB/entity	~10 MB/cell ~100 MB/row	5 TB/object	Determined by DB engine	10,240 MiB/ row	10 MB/row

Comparing storage options: technical details

	Cloud Datastore	Cloud Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Best for	Semi-structured application data, durable key-value data	"Flat" data, Heavy read/write, events, analytical data	Structured and unstructured binary or object data	Web frameworks, existing applications	Large-scale database applications (> ~2 TB)	Interactive querying, offline analytics
Use cases	Getting started, App Engine applications	AdTech, Financial and IoT data	Images, large media files, backups	User credentials, customer orders	Whenever high I/O, global consistency is needed	Data warehousing

Spanner good for E-commerce Stock trader

Check Lab: Google Cloud Fundamentals: Getting Started with Cloud Storage and Cloud SQL

Create VM → Connect to DB, Load File from GS to Webserber

When cp file from GS, you can modify permission so everyone can see it

gsutil acl ch -u allUsers:R gs://$DEVSHELL_PROJECT_ID/my-excellent-blog.png

Task 2: Deploy a web server VM instance

1. In the GCP Console, on the **Navigation menu** ( ), click **Compute Engine** > **VM instances**.
2. Click **Create Instance**.
3. On the **Create an Instance** page, for **Name**, type `bloghost`
4. For **Region** and **Zone**, select the region and zone assigned by Qwiklabs.
5. For **Machine type**, accept the default.
6. For **Boot disk**, if the **Image** shown is not **Debian GNU/Linux 9 (stretch)**, click **Change** and select **Debian GNU/Linux 9 (stretch)**.
7. Leave the defaults for **Identity and API access** unmodified.
8. For **Firewall**, click **Allow HTTP traffic**.
9. Click **Networking, disks, security, management, sole tenancy** to open that section of the dialog.
10. Click **Management** to open that section of the dialog.
11. Scroll down to the Automation section, and enter the following script as the value for **Startup script**:

```
apt-get update
apt-get install apache2 php php-mysql -y
service apache2 restart
```

> Be sure to supply that script as the value of the **Startup script** field. If you accidentally put it into another field, it won't be executed when the VM instance starts.

11. Leave the remaining settings as their defaults, and click **Create**.

> Instance can take about two minutes to launch and be fully available for use.

12. On the **VM instances** page, copy the **bloghost** VM instance's internal and external IP addresses to a text editor for use later in this lab.

Task 3: Create a Cloud Storage bucket using the gsutil command line

All Cloud Storage bucket names must be globally unique. To ensure that your bucket name is unique, these instructions will guide you to give your bucket the same name as your Cloud Platform project ID, which is also globally unique.

Cloud Storage buckets can be associated with either a region or a multi-region location: **US**, **EU**, or **ASIA**. In this activity, you associate your bucket with the multi-region closest to the region and zone that Qwiklabs or your instructor assigned you to.

1. On the **Google Cloud Platform** menu, click **Activate Cloud Shell** . If a dialog box appears, click **Continue**.
2. For convenience, enter your chosen location into an environment variable called LOCATION. Enter one of these commands:

```
export LOCATION=US
```

3. In Cloud Shell, the DEVSHELL_PROJECT_ID environment variable contains your project ID. Enter this command to make a bucket named after your project ID:

```
gsutil mb -l $LOCATION gs://$DEVSHELL_PROJECT_ID
```

If prompted, click **Authorize** to continue.

4. Retrieve a banner image from a publicly accessible Cloud Storage location:

```
gsutil cp gs://cloud-training/gcpfci/my-excellent-blog.png my-excellent-blog.png
```

5. Copy the banner image to your newly created Cloud Storage bucket:

```
gsutil cp my-excellent-blog.png gs://$DEVSHELL_PROJECT_ID/my-excellent-blog.png
```

6. Modify the Access Control List of the object you just created so that it is readable by everyone:

```
gsutil acl ch -u allUsers:R gs://$DEVSHELL_PROJECT_ID/my-excellent-blog.png
```

Task 4: Create the Cloud SQL instance

1. In the GCP Console, on the **Navigation menu** ( ), click **SQL**.
2. Click **Create instance**.
3. For **Choose a database engine**, select **MySQL**.
4. For **Instance ID**, type **blog-db**, and for **Root password** type a password of your choice.

> Choose a password that you remember. There's no need to obscure the password because you'll use mechanisms to connect that aren't open access to everyone.

5. Select **Single zone** and set the region and zone assigned by Qwiklabs.

> This is the same region and zone into which you launched the **bloghost** instance. The best performance is achieved by placing the client and the database close to each other.

6. Click **Create Instance**.

7. Click on the name of the instance, **blog-db**, to open its details page.
8. From the SQL instances details page, copy the **Public IP address** for your SQL instance to a text editor for use later in this lab.
9. Click on **Users** menu on the left-hand side, and then click **ADD USER ACCOUNT**.
10. For **User name**, type `blogdbuser`
11. For **Password**, type a password of your choice. Make a note of it.
12. Click **ADD** to add the user account in the database.

> Wait for the user to be created.

13. Click the **Connections** tab, and then click **Add network**.

> If you are offered the choice between a **Private IP** connection and a **Public IP** connection, choose **Public IP** for purposes of this lab. The **Private IP** feature is in beta at the time this lab was written.

> The **Add network** button may be grayed out if the user account creation is not yet complete.

14. For **Name**, type `web front end`
15. For **Network**, type the external IP address of your **bloghost** VM instance, followed by `/32`

    The result will look like this:

    ```
    35.192.208.2/32
    ```

> Be sure to use the external IP address of your VM instance followed by /32. Do not use the VM instance's internal IP address. Do not use the sample IP address shown here.

16. Click **Done** to finish defining the authorized network.
17. Click **Save** to save the configuration change.

> **Note**: If the message appears like **Another operation is in progress**, wait for few minutes until you see the green check for **blog-db** to save the configuration.

Task 5: Configure an application in a Compute Engine instance to use Cloud SQL

1. On the **Navigation menu** (☰), click **Compute Engine** > **VM instances**.
2. In the VM instances list, click **SSH** in the row for your VM instance **bloghost**.
3. In your ssh session on **bloghost**, change your working directory to the document root of the web server:

   ```
   cd /var/www/html
   ```

4. Use the **nano** text editor to edit a file called **index.php**:

   ```
   sudo nano index.php
   ```

5. Paste the content below into the file:

```
<html>
<head><title>Welcome to my excellent blog</title></head>
<body>
<h1>Welcome to my excellent blog</h1>
<?php
 $dbserver = "CLOUDSQLIP";
$dbuser = "blogdbuser";
$dbpassword = "DBPASSWORD";
// In a production blog, we would not store the MySQL
// password in the document root. Instead, we would store it in
a
// configuration file elsewhere on the web server VM instance.
$conn = new mysqli($dbserver, $dbuser, $dbpassword);
if (mysqli_connect_error()) {
        echo ("Database connection failed: " .
mysqli_connect_error());
} else {
        echo ("Database connection succeeded.");
}
?>
</body></html>
```

10. Return to your ssh session on **bloghost**. Use the **nano** text editor to edit **index.php** again.

    ```
    sudo nano index.php
    ```

11. In the **nano** text editor, replace `CLOUDSQLIP` with the Cloud SQL instance Public IP address that you noted above. Leave the quotation marks around the value in place.
12. In the **nano** text editor, replace `DBPASSWORD` with the Cloud SQL database password that you defined above. Leave the quotation marks around the value in place.
13. Press **Ctrl+O**, and then press **Enter** to save your edited file.
14. Press **Ctrl+X** to exit the nano text editor.
15. Restart the web server:

    ```
    sudo service apache2 restart
    ```

16. Return to the web browser tab in which you opened your **bloghost** VM instance's external IP address. When you load the page, the following message appears:

    ```
    Database connection succeeded.
    ```

Task 6: Configure an application in a Compute Engine instance to use a Cloud Storage object

1. In the GCP Console, click **Cloud Storage > Browser**.
2. Click on the bucket that is named after your GCP project.
3. In this bucket, there is an object called **my-excellent-blog.png**. Copy the URL behind the link icon that appears in that object's **Public access** column, or behind the words "Public link" if shown.

> If you see neither a link icon nor a "Public link", try refreshing the browser. If you still do not see a link icon, return to Cloud Shell and confirm that your attempt to change the object's Access Control list with the **gsutil acl ch** command was successful.

4. Return to your ssh session on your **bloghost** VM instance.
5. Enter this command to set your working directory to the document root of the web server:

   ```
   cd /var/www/html
   ```

6. Use the **nano** text editor to edit **index.php**:

```
sudo nano index.php
```

7. Use the arrow keys to move the cursor to the line that contains the **h1** element. Press **Enter** to open up a new, blank screen line, and then paste the URL you copied earlier into the line.

8. Paste this HTML markup immediately before the URL:

```
<img src='
```

9. Place a closing single quotation mark and a closing angle bracket at the end of the URL:

```
'>
```

The resulting line will look like this:

```
<img src='https://storage.googleapis.com/qwiklabs-gcp-0005e186fa559a09/my-excellent-blog.png'>
```

The effect of these steps is to place the line containing `<img src='...'>` immediately before the line containing `<h1>...</h1>`

> Do not copy the URL shown here. Instead, copy the URL shown by the Storage browser in your own Cloud Platform project.

10. Press **Ctrl+O**, and then press **Enter** to save your edited file.

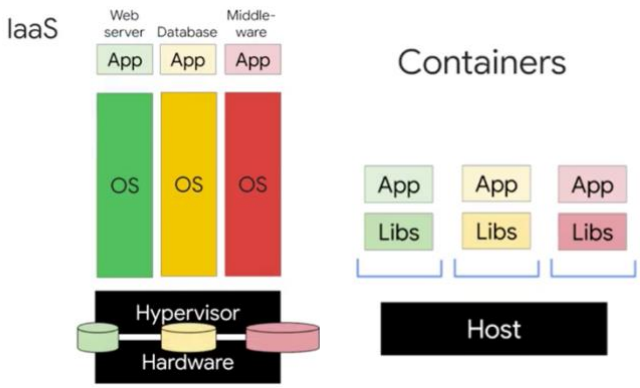
11. Press **Ctrl+X** to exit the nano text editor.

12. Restart the web server:

```
sudo service apache2 restart
```

13. Return to the web browser tab in which you opened your **bloghost** VM instance's external IP address. When you load the page, its content now includes a banner image.

Containers, Kubernetes, and Kubernetes Engine



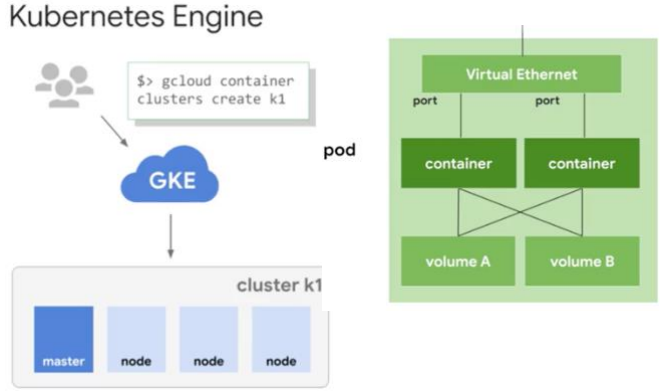
Dockerfile

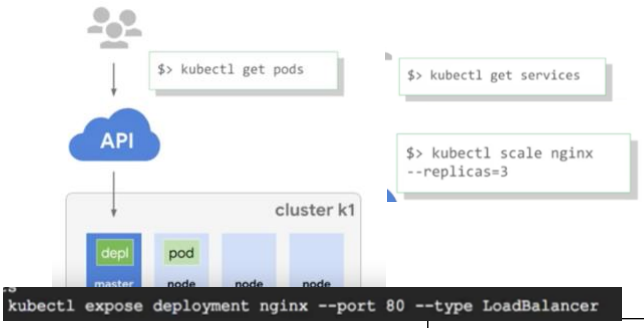
```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENTRYPOINT ["python3", "app.py"]
```

Build and run

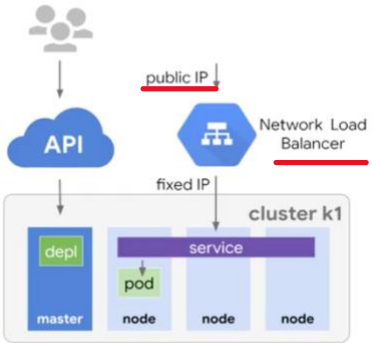
```
$> docker build -t py-server .
$> docker run -d py-server
```

Kubernetes Engine






```
kubectl expose deployment nginx --port 80 --type LoadBalancer
```



So, other containers can get access to the service via public IP

Pod IP address not fixed like service

```
$> kubectl autoscale nginx --min=10 --max=15 --cpu=80
```

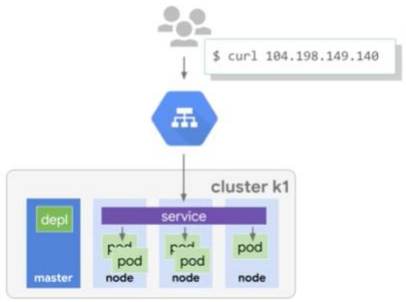
Change replica pods from 3 to 5 to scale out the load balancer but share the same service

1. Deployment config file

Kubernetes

```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.7
        ports:
        - containerPort: 80
```

Kubernetes Engine



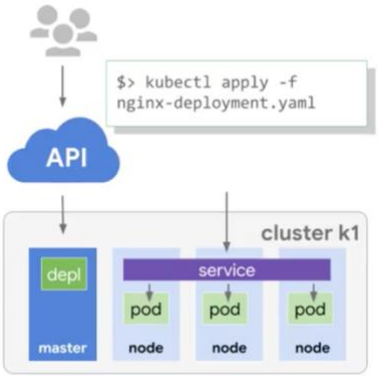
6. check the service if it is still unaffected

```
$> kubectl get services
```

```
NAME    TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)  AGE
nginx   LoadBalancer   10.0.65.118   104.198.149.140  80/TCP   5m
```

Put the new version of app using: rolling updates
With no downtime

2. Used the Updated config file



Kubernetes

```yaml
spec:
  # ...
  replicas: 5
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  # ...
```

3. view your replicates and states

```
$> kubectl get replicasets
```

```
NAME                DESIRED   CURRENT   READY   AGE
nginx-2035384211    5         5         5       18s
```

4. View all 5 pod replicas

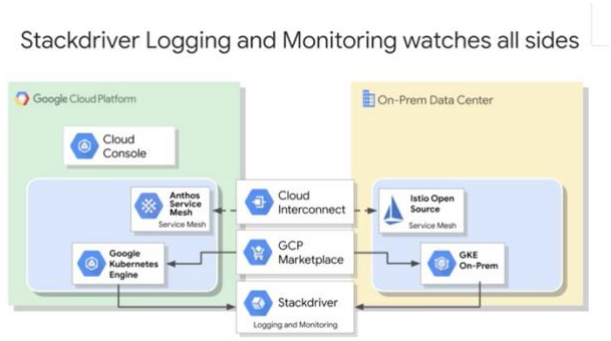
```
$> kubectl get pods
```

```
NAME                     READY   STATUS    RESTARTS   AGE
nginx-2035384211-7ci7o   1/1     Running   0          18s
nginx-2035384211-kzszj   1/1     Running   0          18s
nginx-2035384211-qqcnn   1/1     Running   0          18s
nginx-2035384211-aabbc   1/1     Running   0          18s
nginx-2035384211-knlen   1/1     Running   0          18s
```

5. Check if the replicas are running

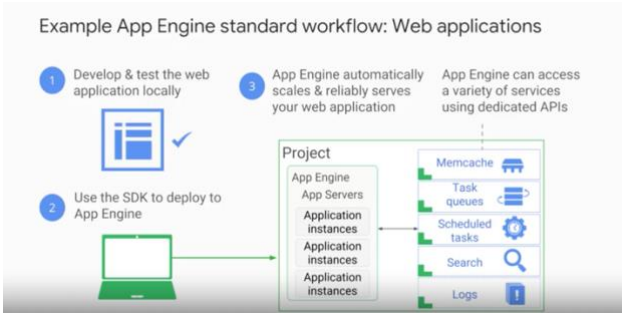
```
$> kubectl get deployments
```

Introduction to Hybrid and Multi-Cloud Computing (Anthos)



Introduction to App Engine

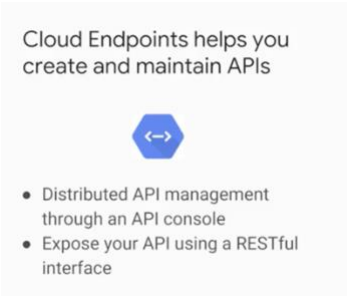
Good for focusing only codes not infrastructure (PaaS) → Use Sandbox



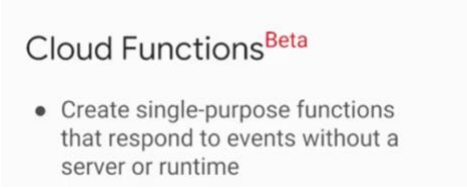
Comparing the App Engine environments

	Standard Environment	Flexible Environment
Instance startup	Milliseconds	Minutes
SSH access	No	Yes (although not by default)
Write to local disk	No	Yes (but writes are ephemeral)
Support for 3rd-party binaries	No	Yes
Network access	Via App Engine services	Yes
Pricing model	After free daily use, pay per instance class, with automatic shutdown	Pay for resource allocation per hour; no automatic shutdown

Cloud Endpoints → APIs & Apache Edges



Development in the Cloud

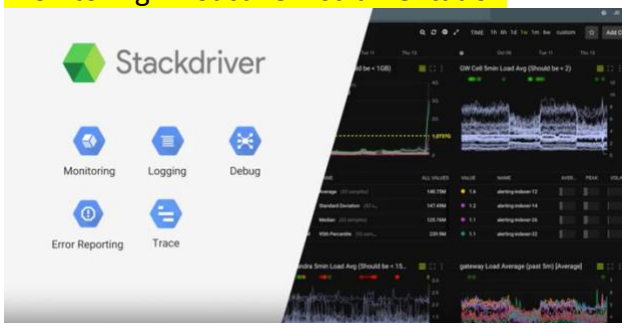


Trigger when event happens like new data comes to GS

Deployment: Infrastructure as code

Use yaml templates to declare conf and dependency

Monitoring: Proactive instrumentation



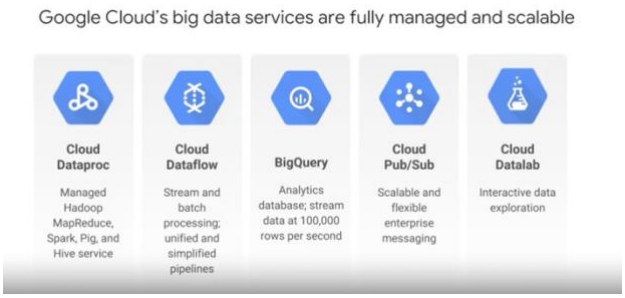
Lab: use .yaml file to deploy the VM machine

Just like you did on the console

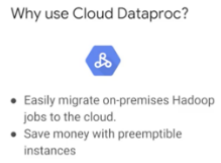
```
# To install the Stackdriver monitoring agent:
$ curl -sSO https://repo.stackdriver.com/stack-install.sh
$ sudo bash stack-install.sh --write-gcm

# To install the Stackdriver logging agent:
$ curl -sSO https://dl.google.com/cloudagents/install-logging-agent.sh
$ sudo bash install-logging-agent.sh
```

Google Cloud Big Data Platform

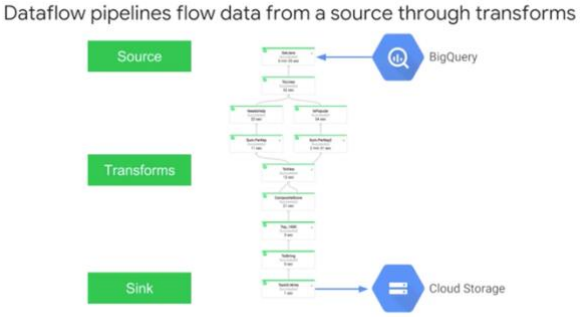


Cloud DataProc -> good when you know the data size
→ you need to launch and manage the cluster



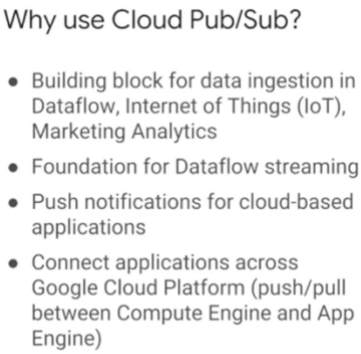
Build cluster→ do job (spark) → delete cluster
Cost ab 1 min billing → preemptible is 80% cheaper

Cloud DataFlows -> good when you dunno know the data size,
→ automated-cluster, real-time streaming

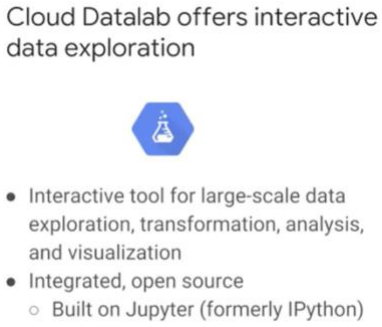


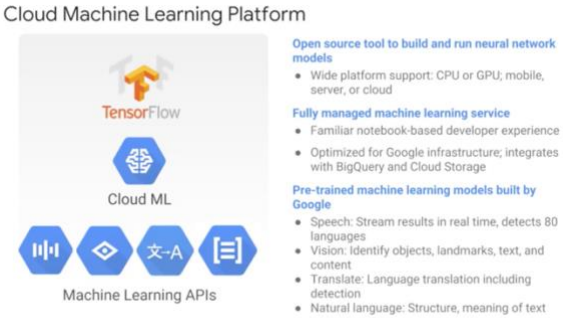
BigQuery

Cloud Pub/Sub
For streaming messages



Cloud Datalab





Lab Instructions can be found on:
https://github.com/ayoub-berdeddouch/gcp-practiceproject