

Google Cloud Platform Intro

Why GCP?

- Student-friendly
 - Credits without credit-cards
 - Ability to use pdx.edu accounts for credits
 - Per-second billing
- Supports open-source APIs and tools to avoid vendor lock-in
 - Go
 - Kubernetes
 - TensorFlow*
- Carbon-neutral since 2007
- Abstractions the same across cloud providers

Why GCP?

- Generous free-tier
 - App Engine
 - 28 instance-hours per day
 - Cloud Datastore
 - 1GB storage, 50k reads, 20k writes, 20k deletes
 - VisionAPI
 - 1k units/month
 - Unit == feature (e.g. facial detection)
 - BigQuery
 - Arbitrary loading, copying, exporting
 - First TB of processed data in queries free
 - But, \$0.02 per GB per month storage

Projects

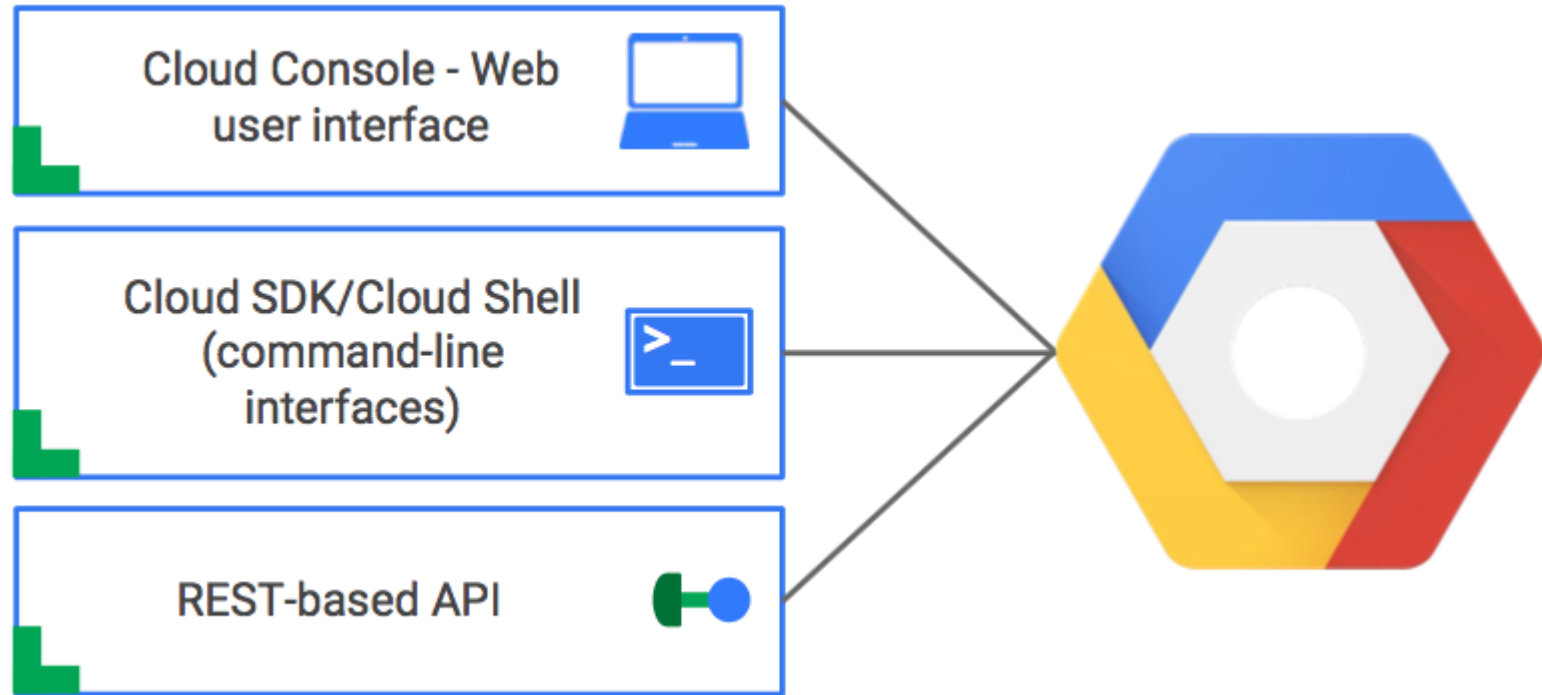
- Many companies with multiple sites
- Each site needs its own
 - Security/access control policies, permissions, and credentials
 - Billing account with separate credit-card/bank accounts
 - Resource and quota tracking
 - Set of enabled services and APIs (most are default OFF and turn on once first used)
- Project abstraction encapsulates this collection
 - Google has 100,000+ projects on GCP to run its sites
 - Contains all resources associated with site and the ability to set permissions on them

Regions and zones in GCP

- Regions: geographic areas where data centers reside
 - us-west, us-east, us-central
 - Consist of collections of zones
- Zones: isolated location within region
 - <https://cloud.google.com/compute/docs/regions-zones/>

us-west1	Oregon	The Dalles, Oregon, USA	us-west1-a	<ul style="list-style-type: none">• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform (default)• 2.0 GHz Intel Xeon (Skylake) platform• Up to 96 vCPU machine types when using the Skylake platform• Local SSDs
			us-west1-b	<ul style="list-style-type: none">• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform (default)• 2.0 GHz Intel Xeon (Skylake) platform• Up to 96 vCPU machine types when using the Skylake platform• Local SSDs• GPUs
			us-west1-c	<ul style="list-style-type: none">• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform (default)• Up to 64 vCPU machine types• Local SSDs

Access to resources



- Also programmatic access in many languages (JavaScript, Python, Go, Java, Ruby)

Command-line GCP

- Install SDK on your local VM (`google-cloud-sdk`) to get commands
 - <https://cloud.google.com/sdk/docs/quickstart-debian-ubuntu>
 - `gcloud`
 - `gsutil` (Cloud Storage)
 - `bq` (Big Query)
- Docker image
 - `docker pull google/cloud-sdk`

Command-line GCP

- Google Cloud Shell
 - Command-line access to cloud resources via web browser
 - Containerized version of Linux with the latest gcloud SDK running on a ComputeEngine instance
 - Has nano, vim, emacs, python2/3, virtualenv, etc.

The image shows two screenshots of the Google Cloud Platform (GCP) dashboard. The top screenshot shows the main dashboard with a blue header bar containing the GCP logo, the project name 'google-faculty-institute', and navigation icons. Below the header, there are tabs for 'DASHBOARD' and 'ACTIVITY', and a button to 'Activate Google Cloud Shell'. The dashboard content includes sections for 'Project info', 'App Engine', and 'Google Cloud Platform status'. The bottom screenshot shows the Google Cloud Shell interface, which is a terminal window with a black background and white text. The terminal displays the message 'Welcome to Cloud Shell! Type "help" to get started.' followed by the prompt 'wuchangfeng@lateral-array-175417:~\$'.

Google Cloud Storage

Google file system (GFS) 2003

- Google search engine
 - Retrieving, storing, and querying of web pages at massive scale
 - Performance requirements
 - Management costs
- File system designed to support Google Search
 - Massive data sets
 - High-throughput, low-latency querying
 - Durability and availability
 - Very little management overhead
 - Dead disks simply replaced and system seamlessly adapts
 - <https://research.google.com/archive/gfs-sosp2003.pdf>
- But, initially proprietary
 - Yahoo! later reverse-engineered GFS
 - Released as Hadoop Distributed File System (HDFS).
 - Open-sourced and distributed by Apache
 - More later...

Google Cloud Storage (gcs)

- Commercial iteration of GFS
 - AWS equivalent is S3
 - Storage done via "buckets"
- Fully-managed, no-ops storage service
 - No administration or capacity management
 - Backed up and versioned automatically
- Replicated and cached over multiple zones/regions
 - Can be fixed to a region based on location of computation
 - Can set multi-region if serving multimedia files to a global population
 - Replicas automatically adapt to load and access patterns to achieve high availability and throughput
- Low latency: 10s of ms on first use, then faster via migration
- Data encrypted at rest when not being used and in flight
 - Key sharding with parts of keys in multiple jurisdictions
 - But, unencrypted when being used
- Massive scale
 - Autism Speaks: 1300 genomes and > 100 TB of data
 - Projected to 10,000 genomes > 1 PB of data

Applications

- Good for large unstructured data that does not need to be queried
 - Images, Video, Zip files
 - Structured data that needs to be queried should use DBs
- Used to feed and store data and logs from all cloud services
 - BigQuery, App Engine, Cloud SQL, ComputeEngine, Dataflow/Dataproc, Etc..
- Access via many methods
 - gcloud SDK, Web interface, REST API
 - Client libraries in Python, Java, PHP, Go

Security, IAM

Cloud security

- In this context, enterprise security
 - Security of the infrastructure running the applications
 - Developers, operations, accounting access to cloud resources
- Securing the applications
 - See CS 495/595: Web Security
 - Some things shared

Traditional enterprise security

- Castle-moat model where trusted access only from within internal networks
 - Firewalls filter external traffic entering enterprise network
 - VPNs for accessing internal services from an external device
 - Implicit trust for machines within internal network
- Issues
 - Enterprise laptops infected on home networks and then moved inside enterprise (WannaCry)
 - Rogue insider with full-access to network and intranet (Edward Snowden)
 - Rogue scripts accessing internal network (DNS rebinding)

Cloud security

- Deperimeterization of network
 - Valid access to cloud resources can come from anywhere
 - Network boundaries that separate “internal” and “external” no longer applicable
- Crux of "zero-trust networks" and Google's BeyondCorp approach <https://www.beyondcorp.com/>
 - Building applications on top of networks you can not trust
 - Reaction against Aurora operation 2009
 - Trust built not from where you connect from (e.g. internal network or VPN), but on strong authentication of user and integrity of the device
 - Restrict kinds of access based on your overall security posture

IAM (Identity and Access Management)

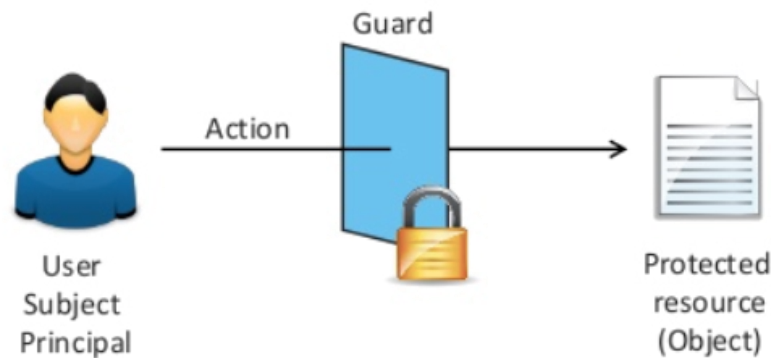
- AWS and GCP approach for implementing cloud security policies
 - Largely similar (i.e. copied)

Identity (Authentication)

- Validating **users** and **applications**
- For users, done via
 - What you know (password)
 - What you have (YubiKey/phone, WebAuthn)
 - Who you are (fingerprint sensor, FaceID)
 - Where you are (network, geographic location)
- For applications (e.g. external web application, internal web application, database)
 - Done via API keys, service-account keys (which must be kept safe!)

Access Management (Authorization)

- Policy to set which users are allowed which actions on which objects
 - Users given roles that grant them specific privileges for access



Types of access management policies

- Discretionary Access Control (object owner decides)
 - Object owner decides
 - Linux model of owner setting coarse permissions on user, group, other
- Mandatory Access Control (system/administrator decides)
 - System or administrator decides
 - Mandated in high-security environments (e.g. government)

Types of access management policies

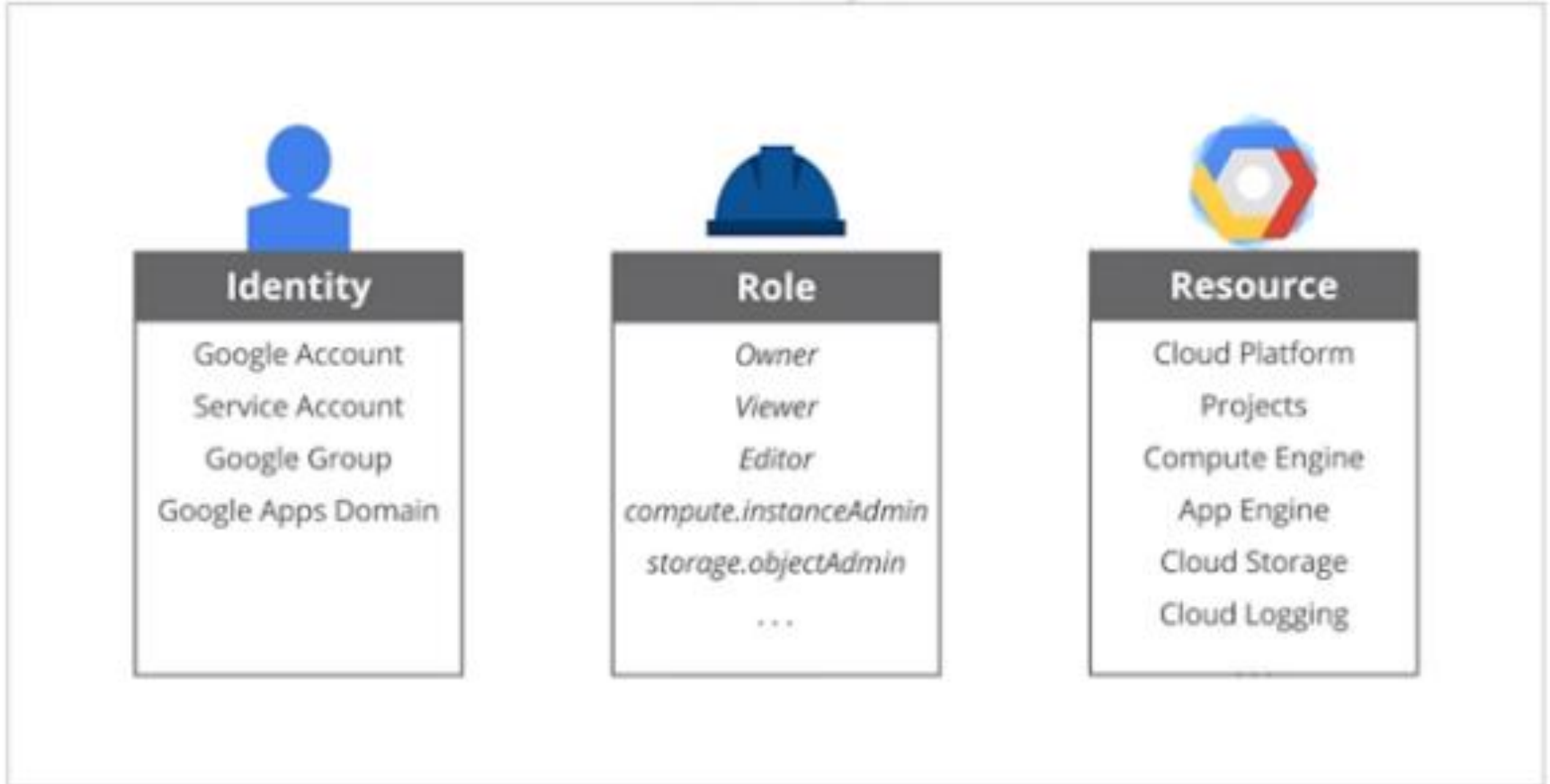
- Role-Based Access Control (system decides based on user role)
 - Role determines privileges afforded
 - Examples
 - IT admin
 - Software developer
 - Billing administrator
 - Third-party integrator
 - Partner users
 - End-users
 - Partner applications
- Principle of least privilege
 - Ensure the minimal level of access that a task or user needs
 - Must apply regardless of the type of policy

Access management via IAM

- Based on Role-based Access control
- Policy determines **who** can do what **action** to which **resource**
 - Action permissions assigned by role
- Primitive pre-defined roles with permissions
 - Curated roles so you do not need to roll your own
 - Owner (**create, destroy, assign access, read, write, deploy**)
 - Editor (**read, write, deploy**)
 - Reader (**read-only**)
 - Billing administrator (**manage billing**)
- On specified resources that include
 - **Virtual machines, network, database instances**
 - **Cloud storage buckets (gs://...)**
 - **BigQuery stores**
 - **Projects**

GCP example

IAM Policy



<https://cloud.google.com/compute/docs/access/iam>

<https://cloud.google.com/compute/docs/access/iam-permissions>

Example

Who?



-  Google Account (*test@gmail.com*)
-  Service Account (*test@project_id.iam.gserviceaccount.com*)
-  Google Group (*test@googlegroups.com*)
-  Google Apps Domain (*test@example.com*)

What resources?

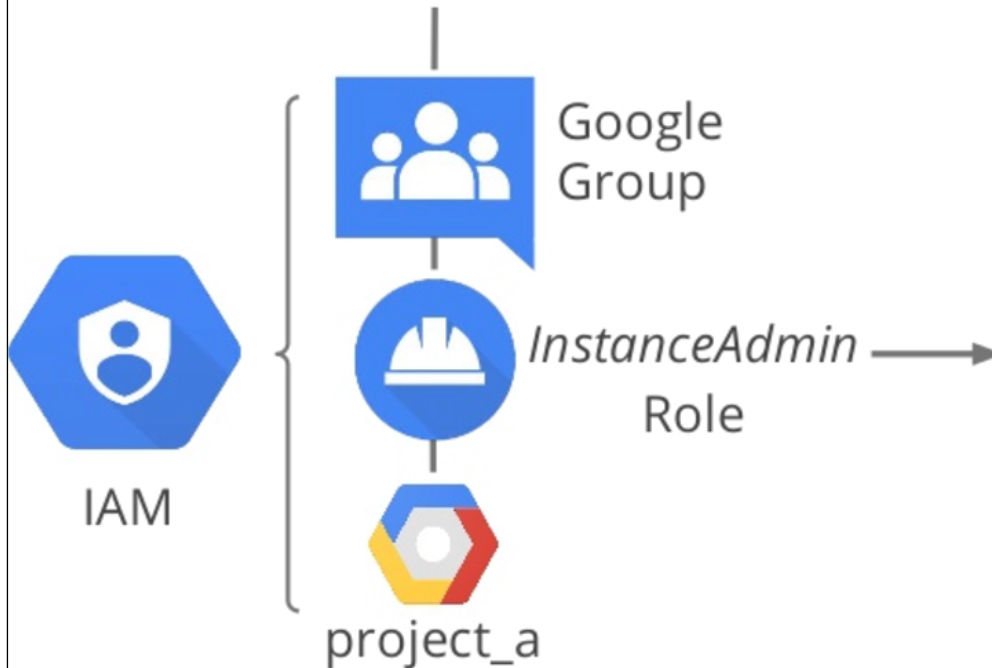
What actions?



List of Permissions

- ✓ *compute.instances.delete*
- ✓ *compute.instances.get*
- ✓ *compute.instances.list*
- ✓ *compute.instances.setMachineType*
- ✓ *compute.instances.start*
- ✓ *compute.instances.stop*

...



Service accounts

- Provides identity for software/applications
 - Allows authenticated access based on a shared secret key
 - e.g. A Slack bot authenticating itself to Slack
 - Service account identified via e-mail address that includes Project number or ID
 - Example

The screenshot shows the Google Cloud Platform IAM & Admin console. The left sidebar contains navigation options: IAM & Admin, IAM, Identity, Quotas, Service accounts (highlighted), Labels, GCP Privacy & Security, Settings, Encryption keys, Identity-Aware Proxy, and Roles. The main content area is titled 'Service Accounts' and includes a '+ CREATE SERVICE ACCOUNT' button and a 'DELETE' button. Below this, there is a search bar and a table of service accounts for the project 'google-faculty-institute'. The table has columns for 'Service account name', 'Service account ID', and 'Key ID'. The 'My Codelab Service Account' row is circled in red, showing its email address and key ID.

<input type="checkbox"/>	Service account name ^	Service account ID	Key ID
<input type="checkbox"/>	App Engine app default service account	lateral-array-175417@appspot.gserviceaccount.com	No keys
<input type="checkbox"/>	Compute Engine default service account	597544055645-compute@developer.gserviceaccount.com	No keys
<input type="checkbox"/>	My Codelab Service Account	codelab@lateral-array-175417.iam.gserviceaccount.com	3eb690179c248fc...

Service accounts

- Google manages keys for certain services automatically (AppEngine, ComputeEngine)
- Must restrict permissions per-key
 - Prevent service account compromise from compromising entire project (least privilege)

IAM policies

- Massive number of resources
- Each resource must have highly granular control over access to properly secure resources (e.g. many permissions)
- Primitive roles (owner, editor, reader) with fixed permissions not enough

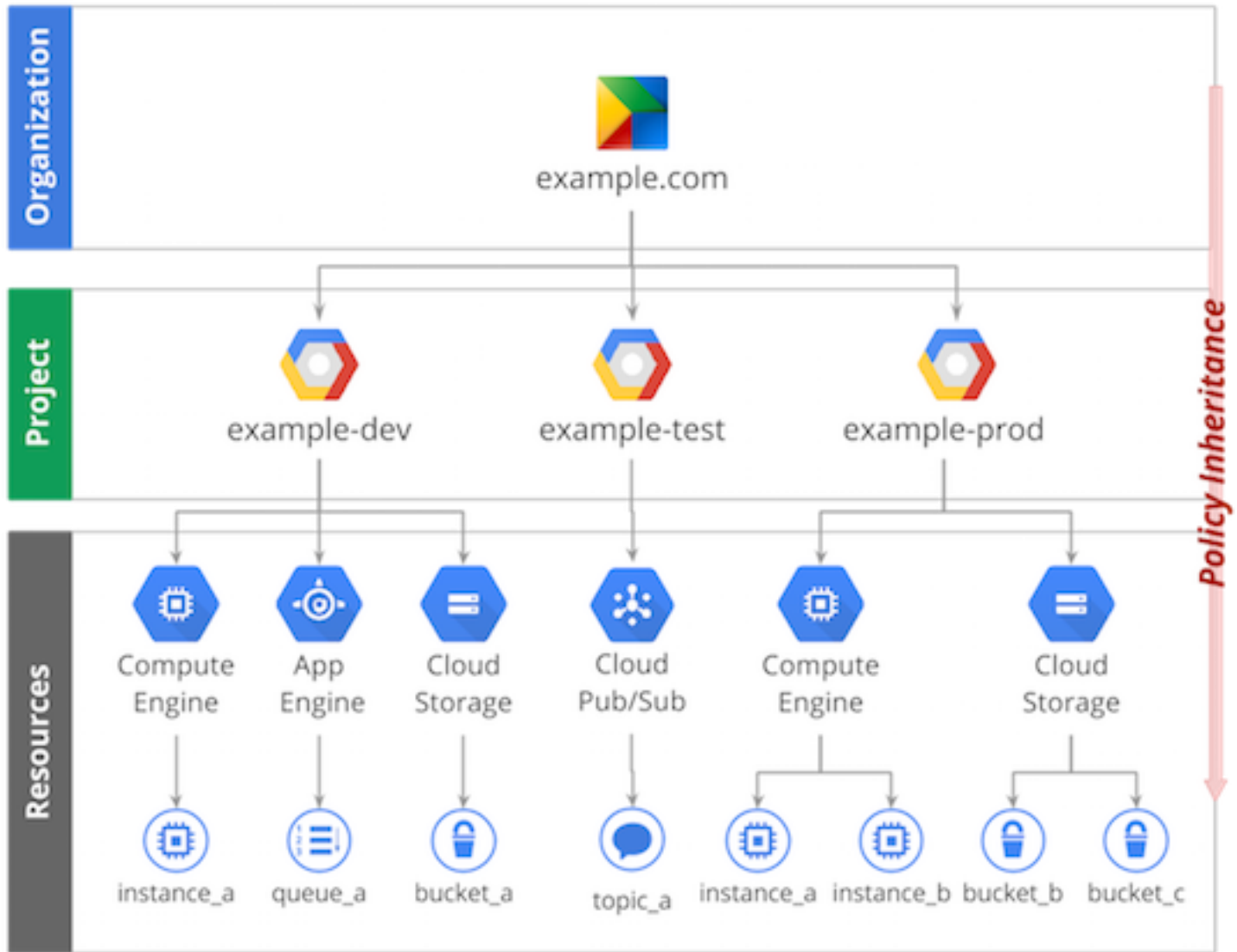
Examples

- e-Commerce site with a crashing bug
 - Developer wants to access logs is given reader access to instance
 - Can read logs to do job
 - But can also access all personally identifiable information of the site's users!
- Continuous integration tool used in DevOps is given editor access to deploy updates
 - Can update code, but also modify storage buckets, compute instances, and network configuration!

IAM complexity

- Granular access control leads to hundreds of thousands of permissions and complex policies
- Organized as a hierarchy to ease management burden
 - Set permissions across all projects at once
 - Set permissions of resources (i.e. 1000s of VMs/buckets in project) at once
 - Command-line scripting, configuration management via commercial tools
 - Implement inheritance of permissions where higher-level permissions trump lower ones

Hierarchical management



IAM complexity

- But,
 - AWS => 3000+ types of permissions/resources available
 - Motivates approaches like RepoKid from Netflix to automatically revoke unused permissions via ML

Labs

Cloud Storage Lab #1

- Interact with Cloud Storage (USGS data)
 - Data processing with Python, Matlab+Basemap
- In Cloud Shell

```
git clone https://github.com/GoogleCloudPlatform/training-data-analyst
cd training-data-analyst/CPB100/lab2b
```

- `ingest.sh`

```
#!/bin/bash
# remove older copy of file, if it exists
rm -f earthquakes.csv
# download latest data from USGS
wget
http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_week.csv
v -O earthquakes.csv
```

- Perform a head on `earthquakes.csv` to ensure it has been pulled down properly

- `install_missing.sh` gets basemap, numpy, matlab packages for Python

```
sudo apt-get update
sudo apt-get --fix-missing install python-mpltoolkits.basemap
python-numpy python-matplotlib
```

- Processing script `transform.py` to generate plots of earthquakes
 - Import packages

```
import csv
import urllib2
import cStringIO
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
```

- Earthquake class definition
 - Each line of CSV is an earthquake instance ingested and parsed into a list that the class creates instances out of

```
class Earthquake:
    def __init__(self, row):
        # Parse earthquake data from USGS
        self.timestamp = row[0]
        self.lat = float(row[1])
        self.lon = float(row[2])
        try:
            self.magnitude = float(row[4])
        except ValueError:
            self.magnitude = 0
```

- Ingest data via URL (can also use local file:///)

```
def get_earthquake_data(url):
    # Read CSV earthquake data from USGS
    response = urllib2.urlopen(url)
    csvio = cStringIO.StringIO(response.read())
    reader = csv.reader(csvio)
    header = next(reader)
    quakes = [Earthquake(row) for row in reader]
    quakes = [q for q in quakes if q.magnitude > 0]
    return quakes
```

```
quakes = get_earthquake_data('http://earthquake.usgs.gov/earthquakes/feed/v1.0/su
```

- Create Basemap, setup markers based on earthquake magnitude

```
# Set up Basemap
mpl.rcParams['figure.figsize'] = '16, 12'
m = Basemap(projection='kav7', lon_0=-90, resolution = '1', area_thresh = 1000.0)
m.drawcoastlines()
m.drawcountries()
m.drawmapboundary(fill_color='0.3')
m.drawparallels(np.arange(-90.,99.,30.))
junk = m.drawmeridians(np.arange(-180.,180.,60.))

# control marker color and size based on magnitude
def get_marker(magnitude):
    markersize = magnitude * 2.5;
    if magnitude < 1.0:
        return ('bo'), markersize
    if magnitude < 3.0:
        return ('go'), markersize
    elif magnitude < 5.0:
        return ('yo'), markersize
    else:
        return ('ro'), markersize
```

- Plot quakes onto map m
 - Grab x,y coordinates on plot based on longitude and latitude
 - Get color and size
 - Add marker to plot

```
# add earthquake info to the plot
for q in quakes:
    x,y = m(q.lon, q.lat)
    mcolor, msize = get_marker(q.magnitude)
    m.plot(x, y, mcolor, markersize=msize)
```

- Emit image




```
# add a title
plt.title("Earthquakes {0} to {1}".format(start_day, end_day))
plt.savefig('earthquakes.png')
```

- Create storage bucket (see Database Lab #2 or via console web UI) and copy output files to it

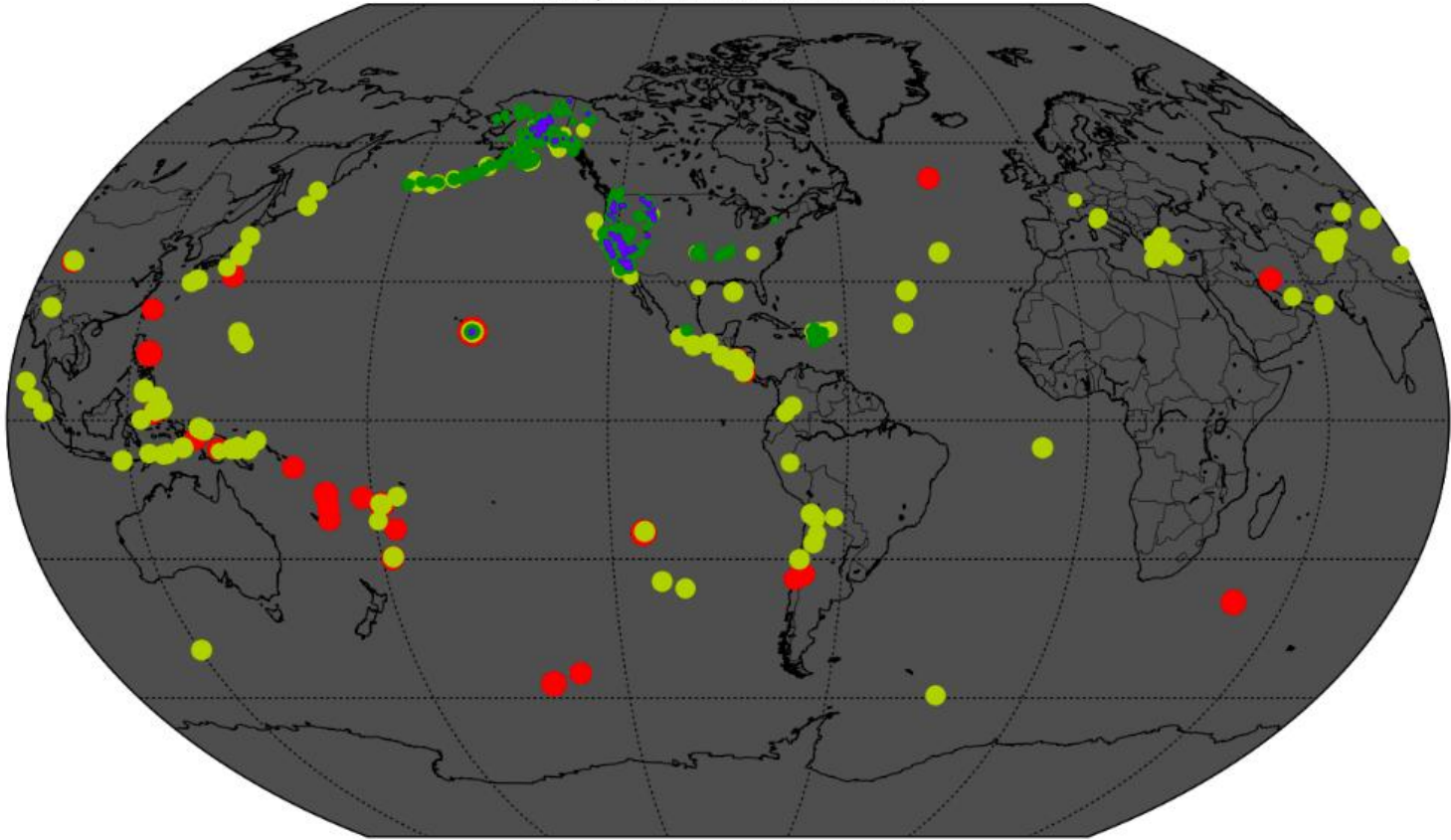
```
gsutil cp earthquakes.* gs://<YOUR-BUCKET>/
```

- Then make files in bucket public (to create links)

[Buckets](#) / earthquake-wuchang

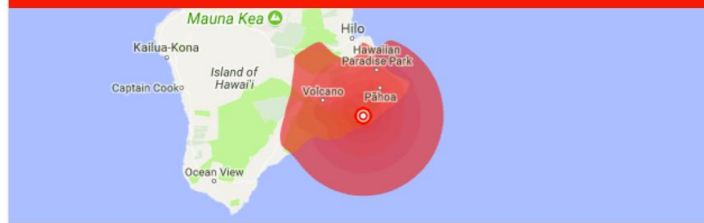
<input type="checkbox"/>	Name	Size	Type	Storage class	Last modified	Share publicly
<input type="checkbox"/>	 earthquakes.csv	523.87 KB	text/csv	Regional	5/6/18, 12:00 PM	<input checked="" type="checkbox"/> Public link
<input type="checkbox"/>	 earthquakes.htm	751 B	text/html	Regional	5/6/18, 12:00 PM	<input checked="" type="checkbox"/> Public link
<input type="checkbox"/>	 earthquakes.png	311.22 KB	image/png	Regional	5/6/18, 12:00 PM	<input checked="" type="checkbox"/> Public link

Earthquakes 2018-04-29 to 2018-05-06



Magnitude 6.9 earthquake

11 miles from Ainaloa, HI · May 4, 3:32 PM

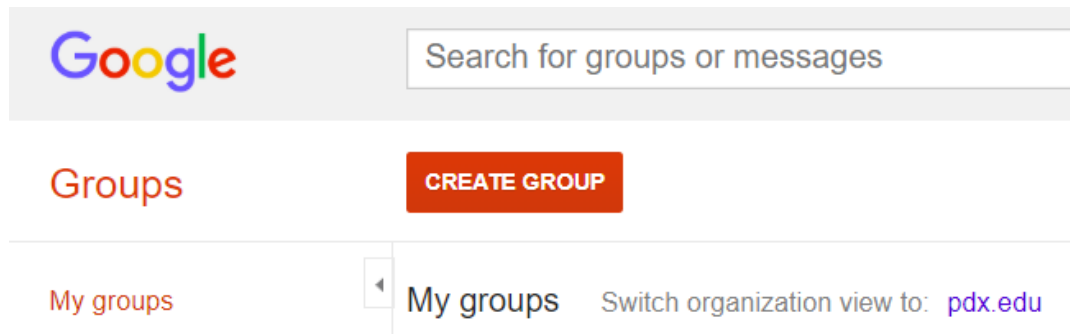


Cloud Storage Lab #1

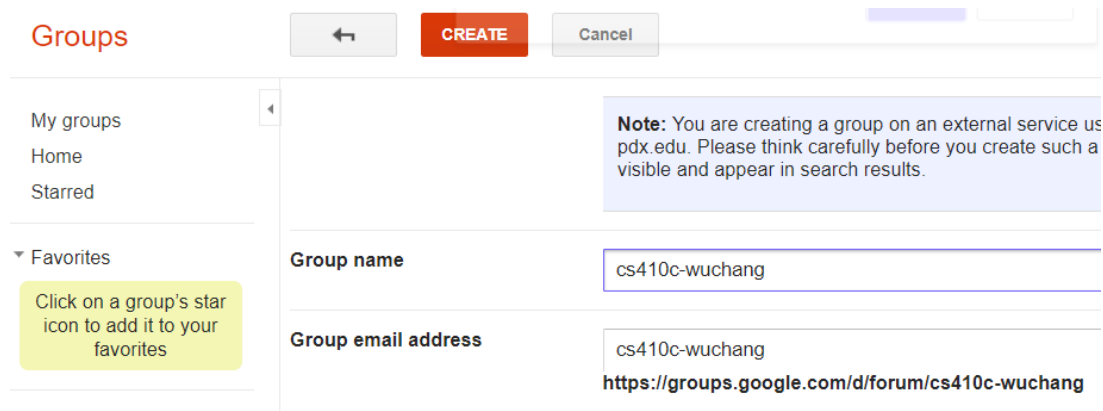
- <https://codelabs.developers.google.com/codelabs/cpb100-cloud-storage> (15 min)

IAM Lab #1

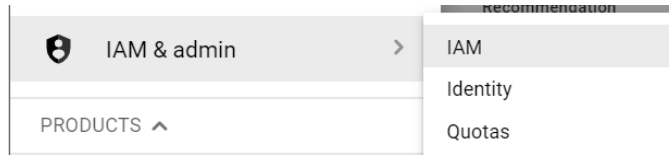
- Create a Google group at <https://groups.google.com> called cs410-OdinID



- Add yourself, me (wuchang@pdx.edu), and your partner (if working in a group)



- In IAM, add the group to project permissions (Project=>Viewer)

A screenshot of the Google Cloud Platform IAM & Admin interface. The page title is 'IAM & Admin' and the project is 'google-faculty-institute'. The main heading is 'IAM' with '+ ADD' and '- REMOVE' buttons. The left sidebar shows a navigation menu with 'IAM' selected. The main content area is titled 'Permissions for project "google-faculty-institute"' and contains explanatory text and a table of members.

Permissions for project "google-faculty-institute"

These permissions affect the entire "google-faculty-institute" project and all of its resources. To grant permissions, add a member and then select a role for them. Members can be people, domains, groups, or service accounts.

Some roles are in beta development and might be changed or deprecated in the future. [Learn more](#) ↗.

Filter by name or role View by: Roles ▾

Type	Members ^	Role(s)	
<input type="checkbox"/>	My Codelab Service Account codelab@lateral-array-175417.iam.gserviceaccount.com	Owner ▾	
<input type="checkbox"/>	Wu-chang Feng wuchangfeng@gmail.com	Owner ▾	

New members

cs410c-wuchang@googlegroups.com ✕

Select a role

🔍 Type to filter

Project

App Engine

Viewer

Read access to all resources.

Browser

Editor

Owner

Viewer

IAM Lab #1

- Test with your partner or with me (if you do not have a partner)
- For help only
 - <https://cloud.google.com/iam/docs/quickstart>

Extra

Google Cloud Storage Lab #2

- Hosting a static web-site using gcs
 - <https://cloud.google.com/storage/docs/hosting-static-website>

Managing credentials

- GCP credentials and keys should be protected at all times
 - Audit Github, Bitbucket, Dockerhub, web
 - Crawlers continuously looking for credentials on public repositories
 - Immediately regenerate keys if exposed
 - Instagram AWS credentials on snapshot
 - Canary API tokens