

Cloud Computing

Cloud Shell

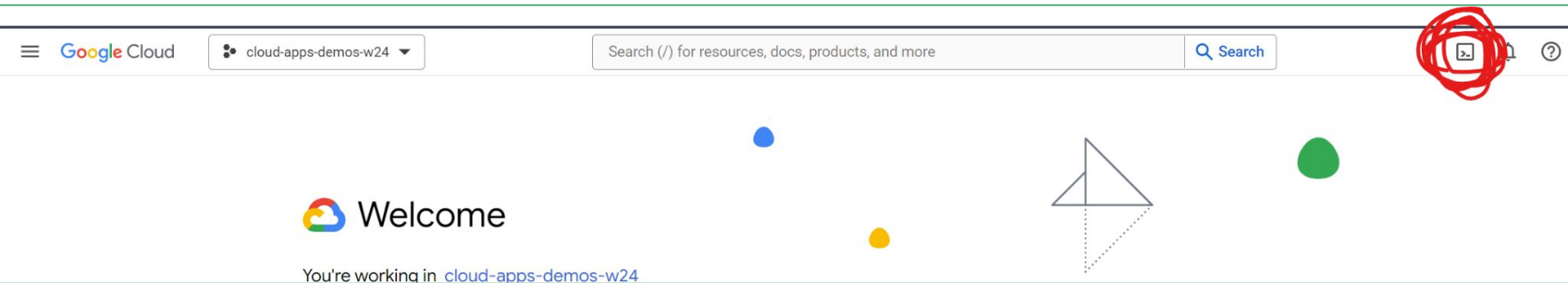
CIS437

Erik Fredericks // frederer@gvsu.edu

Adapted from Google Cloud Computing Foundations, Overview of Cloud Computing (Wufka & Canonico)



The shell



(AWS CloudShell is quite similar)

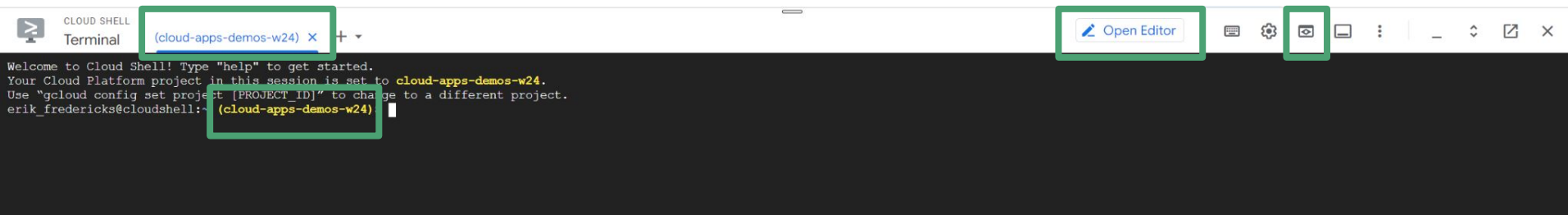
(Microsoft? Also has a shell)

→ Currently PowerShell though

→ Planning on being a sysadmin? LEARN POWERSHELL

→ It's pretty good!

Features: <https://cloud.google.com/shell/>



For Google

Shell is a Linux environment that can control pretty much **all** (Google) services
(Same applies to AWS, but not the Google part)

Also, nicely just highlighting something auto-copies it to your clipboard

One thing to keep in mind

We're going to be playing with a handful of different Google Cloud products here

- Most of which we'll be covering in depth later
- So don't get too flustered if you don't "understand" what it is yet
 - That comes later



There are four ways to interact with Google Cloud



**Google Cloud
Console**

Web user
interface



**Cloud SDK and
Cloud Shell**

Command-line
interface



**REST-based
API**

For custom
applications



**Cloud Console
mobile app**

For iOS and
Android

Example 1: Basic usage

File management (i.e., common commands)

- **move file:** `mv file1 file2`
- **copy file:** `cp file1 file2`
- **copy directory:** `cp -r dir1 dest_directory`
- **delete file:** `rm file`
- **delete directory:** `rm -rf dir1`

Git

- **clone:** `git clone <repository url>`

- (If managing git in the Cloud Shell then all your common commands work)
 - (There's also a Cloud Source Repositories that seems to be being actively deprecated)

Effective June 17, 2024, Cloud Source Repositories isn't available to new customers. If your organization hasn't previously used Cloud Source Repositories, you can't enable the API or use Cloud Source Repositories. New projects not connected to an organization can't enable the Cloud Source Repositories API. Organizations that have used Cloud Source Repositories prior to June 17, 2024 are not affected by this change.

Cloud Source Repositories
documentation

[View all product documentation](#)



Common GCP commands

Setting environment variables (not strictly GCP, but you'll do it a lot)

```
$ myvar="HELLO THERE"
```

```
$ echo $myvar
```

Setting region

```
$ gcloud config set compute/region us-east1
```

Setting project IDs

```
$ gcloud config set project PROJECT_ID
```

...essentially everything is wrapped around the gcloud command

- Use `gcloud --help` to learn more
 - e.g., `gcloud config --help` to learn about the config parameter

Example 2: Creating a VM

Many different ways to configure a VM

- Manual, from a template, cloning, etc.

We'll use a publicly available image for the purposes of this demo

- Just know that you can configure everything to your heart's content



<https://cloud.google.com/compute/docs/instances/create-start-instance#gcloud>

```
$ gcloud compute images list
```

```
NAME:  
ubuntu-2204-jammy-v20240801  
PROJECT: ubuntu-os-cloud  
FAMILY: ubuntu-2204-lts  
DEPRECATED:  
STATUS: READY
```

Then, let's look at a bit more

```
$ gcloud compute images describe ubuntu-2204-jammy-v20240801  
--project ubuntu-os-cloud
```

1. **Identify the main components of the system.** The system consists of a **client** and a **server**. The client is responsible for sending requests to the server, and the server is responsible for processing these requests and returning responses.

Also need the zone and machine type

First, what zones are there?

```
$ gcloud compute zones list
```

And what machine types in that zone?

```
$ gcloud compute machine-types list --zones us-east5-a
```

Let's go e2-medium

Now let's create it

```
$ gcloud compute instances create VM_NAME \  
  --zone=ZONE \  
  [--image=IMAGE | --image-family=IMAGE_FAMILY] \  
  --image-project=IMAGE_PROJECT \  
  --machine-type=MACHINE_TYPE
```

i.e.,

```
$ gcloud compute instances create ubuntu-test-vm \  
  --zone=us-east5-a \  
  --image-family=ubuntu-2204-lts \  
  --image-project=ubuntu-os-cloud \  
  --machine-type=e2-medium
```


And...

```
Created [https://www.googleapis.com/compute/v1/projects/cloud-apps-demos-w24/zones/us-east5-a/instances/ubuntu-test-vm] .  
NAME: ubuntu-test-vm  
ZONE: us-east5-a  
MACHINE_TYPE: e2-medium  
PREEMPTIBLE:  
INTERNAL_IP: 10.202.0.2  
EXTERNAL_IP: 34.162.203.115  
STATUS: RUNNING
```

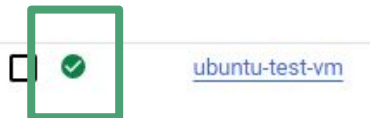
VM instances

 Filter Enter property name or value

<input type="checkbox"/> Status	Name 	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/> 	cis655-microinstance-1	us-east1-b			10.142.0.2 (nic0)		SSH  
<input type="checkbox"/> 	ubuntu-test-vm	us-east5-a			10.202.0.2 (nic0)	34.162.203.115 (nic0)	SSH  

Neat, HOWEVER

Your VM is now running



MEANING IT IS CONSUMING RESOURCES

- Always shut your (*non free-tier*) VMs off when done, otherwise they'll eat up your available credits

Monthly estimate

\$25.46

That's about \$0.03 hourly

Pay for what you use: no upfront costs and per second billing

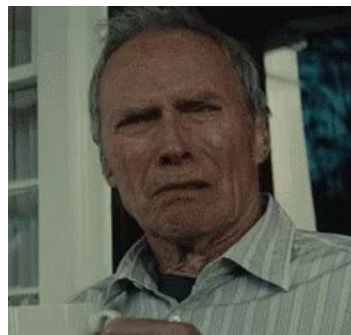
Item	Monthly estimate
2 vCPU + 4 GB memory	\$24.46
10 GB balanced persistent disk	\$1.00
Total	\$25.46

[Compute Engine pricing](#) 

[^ LESS](#)

But wait

We don't need this pesky thing anymore!



We have the *Cloud Shell*

```
$ gcloud compute ssh <vm> --zone=<zone>
```

also! allows for tab completion!

Cloud ... Code Editor

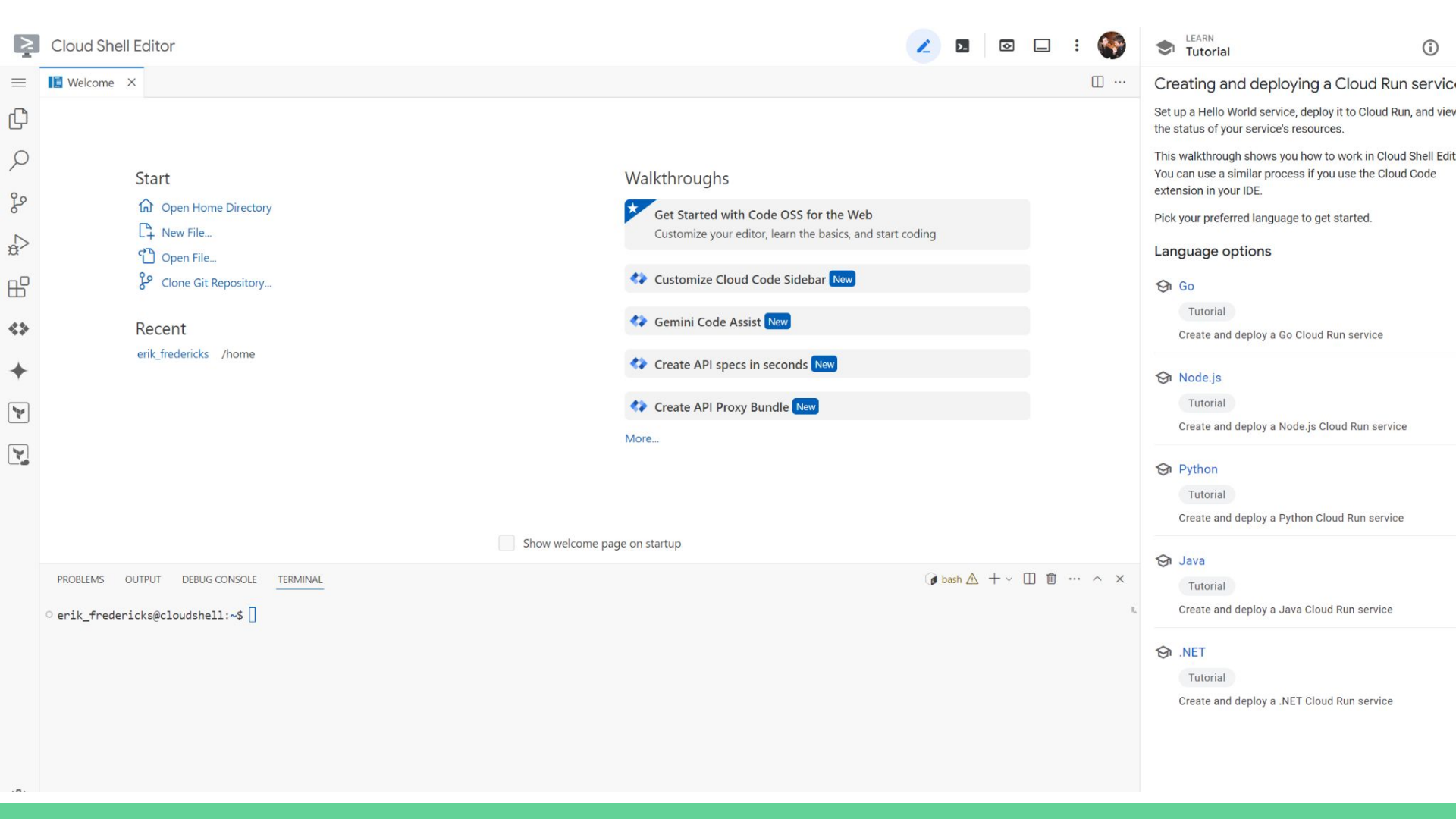
(this is new to me - not sure how long it has been added)

- double note - you'll see this quite often as all cloud providers like to change their interface at what feels like a daily rate

Essentially, VSCode in the cloud

Can hook nicely into your cloud apps if you'd like

<https://cloud.google.com/shell/docs/launching-cloud-shell-editor>



Local tools (!!)

Can install in a local environment as well

<https://cloud.google.com/shell/docs/using-cloud-shell-with-gcloud-cli>

(Note - comes pre-installed in Compute Engine instances)

(Installation instructions directly)

<https://cloud.google.com/sdk/docs/install#deb>

Authorization flows (outside of Cloud Shell)

This leads us to the topic of authorization

- You don't want just anybody able to remote in and run commands, right?

When running `$ gcloud init`:

```
erik@hometop:~$ gcloud init
Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

You must sign in to continue. Would you like to sign in (Y/n)?
```



Google Cloud SDK wants to access your Google Account



erik.fredericks@gmail.com

This will allow **Google Cloud SDK** to:

- See, edit, configure, and delete your Google Cloud data and see the email address for your Google Account. ⓘ
- View and sign in to your Google Cloud SQL instances ⓘ
- View and manage your Google Compute Engine resources ⓘ
- View and manage your applications deployed on Google App Engine ⓘ

Make sure you trust Google Cloud SDK

You may be sharing sensitive info with this site or app. Learn about how Google Cloud SDK will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your [Google Account](#).

[Learn about the risks](#)

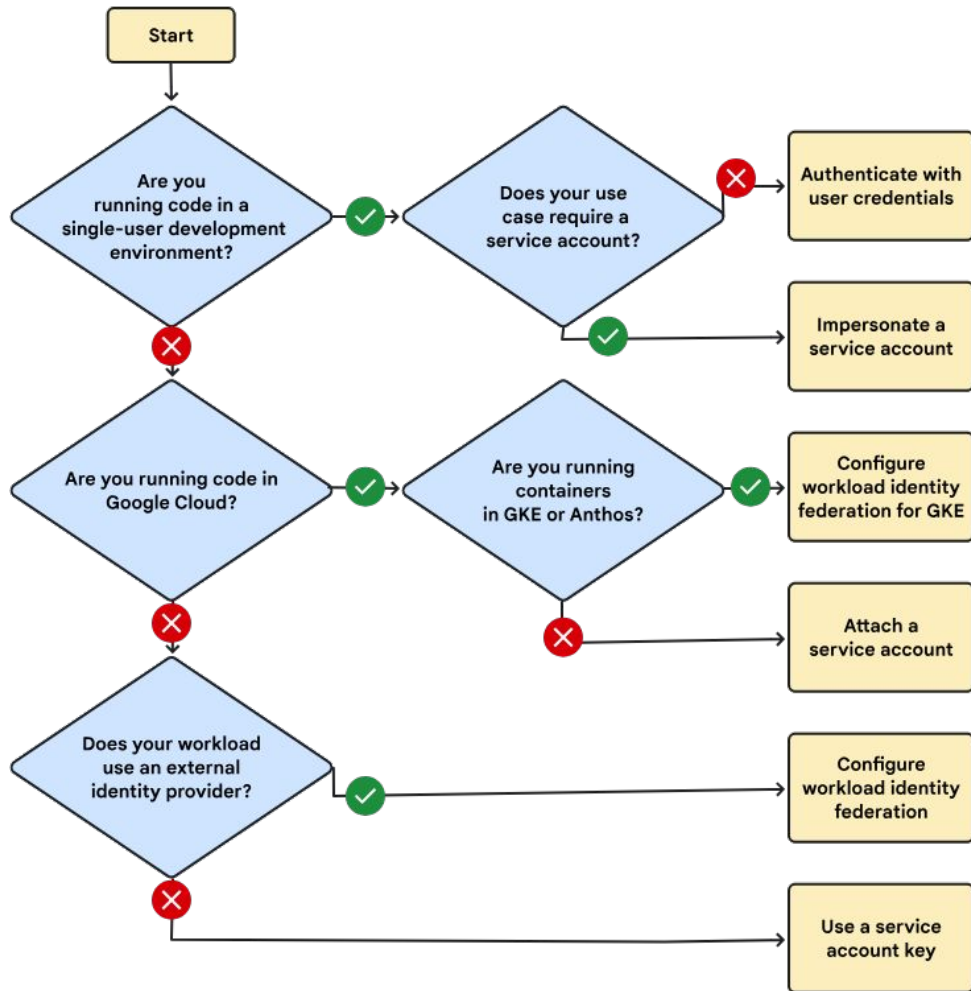
Cancel

Allow

Let's try that last one again, but *locally*

```
$ gcloud compute instances create ubuntu-test-vm-local \  
  --zone=us-east5-a \  
  --image-family=ubuntu-2204-lts \  
  --image-project=ubuntu-os-cloud \  
  --machine-type=e2-medium
```

You may need other libraries to get going (e.g., AppEngine specific)



VED



Example 3: Creating and filling a Bucket

What is a bucket?

- Place to store files!

Naturally could do this through the Console, but let's try in the shell

```
$ gcloud storage buckets create gs://BUCKET_NAME  
--location=BUCKET_LOCATION
```

so...

```
gcloud storage buckets create gs://fredericks-test-bucket  
--location=us-east1
```

*Bucket name must be **globally** unique*

Adding to the bucket

Let's grab a file first

Naturally...

```
$ wget
```

```
https://gifdb.com/images/high/obi-wan-kenobi-well-hello-there-hzgui7yr5ketac2c.gif
```

Then to make life easier

```
$ mv obi-wan-kenobi-well-hello-there-hzgui7yr5ketac2c.gif  
hello.gif
```


Uploading

```
$ gcloud storage cp OBJECT_LOCATION gs://DESTINATION_BUCKET_NAME
```

ergo

```
$ gcloud storage cp hello.gif gs://fredericks-test-bucket
```

Example 4: RESCUING AN SSH SESSION

One of **the most common** problems you'll have with VMs (i.e., Compute Engine)

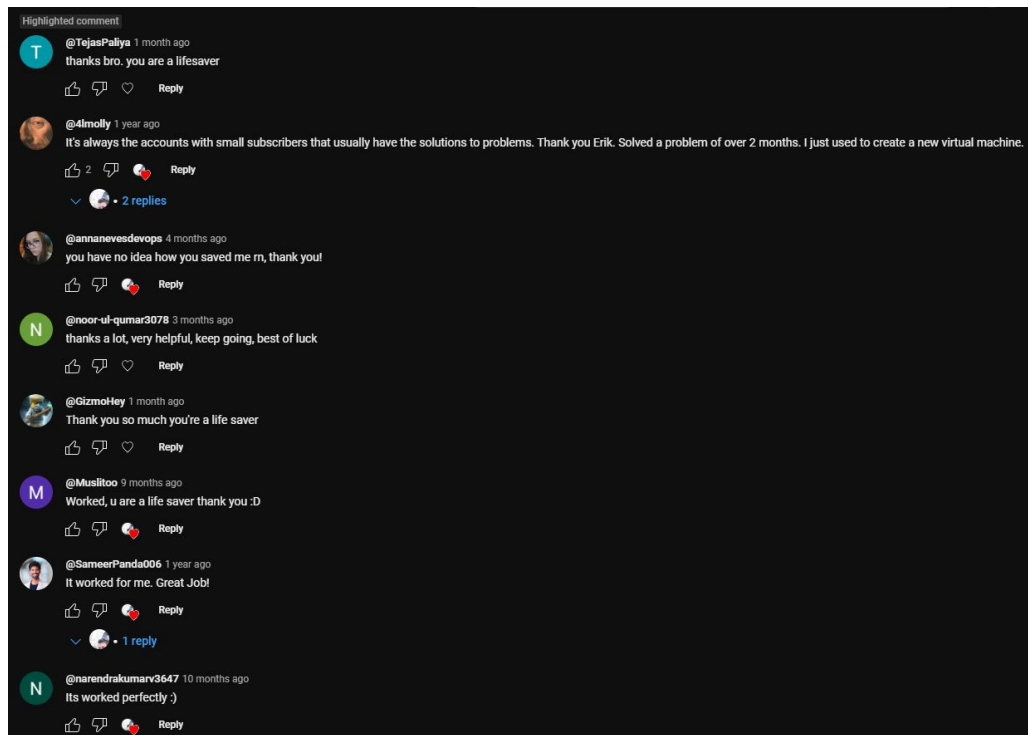
How does this happen?

- 1) You create a VM
- 2) You connect to the VM and spend a *lot* of time getting it just so
- 3) You enable the firewall
- 4) You close the SSH session
- 5) ...
- 6) You re-open the SSH session?
 - a) You re-open the SSH session?



Weirdly, the video I've posted for CIS655 that has the most comments from people around the world

*Not trying to flex, just weird
getting YT notifications for class
material*



Anyway, what we need to do is enable the serial console

(This is assuming you made a Debian-flavored server and can use the `ufw` command - you'd have to use whichever firewall you enabled for other server OSs)

1. Shut the machine off
2. Edit it and click the checkbox for remote access via the serial console
3. Add the bash script mentioned in the link above, replacing `USERNAME` with your Google Cloud username and password with a temporary one
4. Save it, turn on your machine, connect via serial console, login with the username/pw you just added
5. `sudo ufw allow 22`
6. Shut off, edit machine again, disable serial and remove bash script, done

What just happened?

We connected via serial

- Pretend we plugged in a keyboard instead of remoted in

We used our local account with a temporary password to access the VM

We opened the SSH port (22) so that it wasn't blocked

Then, we cleaned up after ourselves so that there aren't any dangling login methods others can exploit



Welcome to App Engine

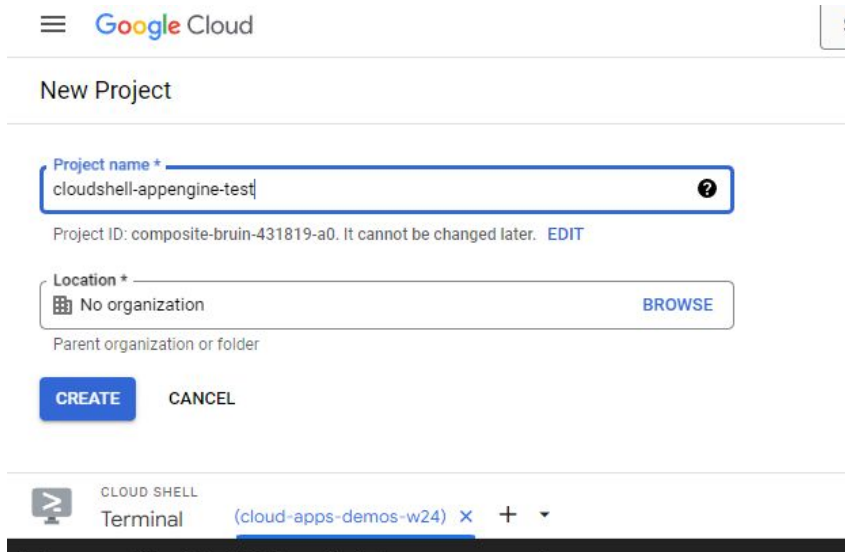
Build scalable apps in any language on Google's infrastructure

Deploying an App Engine app

ONE THING TO KEEP IN MIND THAT *CAN* BE THE MOST FRUSTRATING THING IN THE WORLD WITH APP ENGINE

- Once you enable App Engine on a project, it **cannot** be deleted **or** disabled
 - (Unless if this has been changed, but for the past 5 years you can't)
- Meaning, you have to **delete the project** if you don't want it active anymore

So first, let's try to be smart about this since we know we'll be deleting it anyway



Google Cloud

New Project

Project name *
cloudshell-appengine-test

Project ID: composite-bruin-431819-a0. It cannot be changed later. [EDIT](#)

Location *
No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

CLOUD SHELL
Terminal (cloud-apps-demos-w24) x + ▾

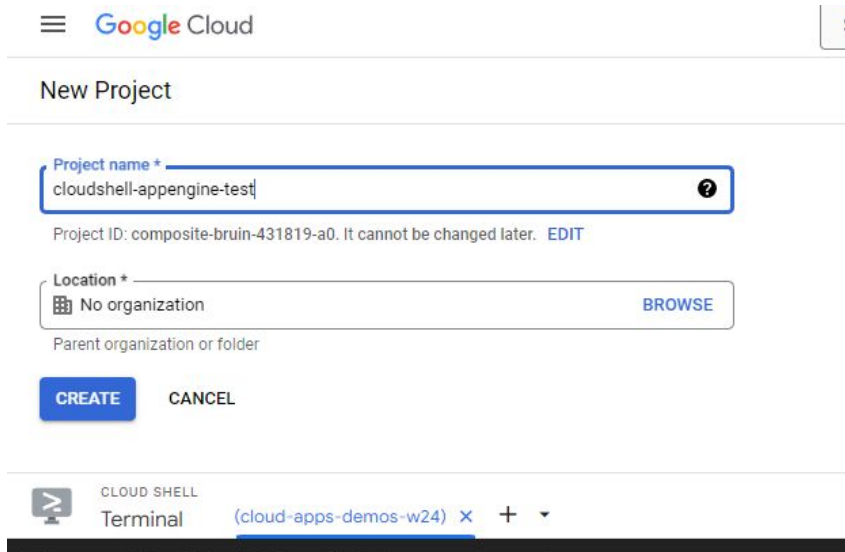
But we still have to deal with this!



```
erik_fredericks@cloudshell:~ (cloud-apps-demos-w24) $
```

How?

So first, let's try to be smart about this since we know we'll be deleting it anyway



The screenshot shows the Google Cloud 'New Project' form. At the top is the Google Cloud logo. Below it is the title 'New Project'. There are two main input fields: 'Project name *' and 'Location *'. The 'Project name' field contains the text 'cloudshell-appengine-test' and has a help icon. Below it, the 'Project ID' is shown as 'composite-bruin-431819-a0' with a note that it cannot be changed later and an 'EDIT' link. The 'Location' field shows 'No organization' with a 'BROWSE' button. Below these fields are 'CREATE' and 'CANCEL' buttons. At the bottom, there is a 'CLOUD SHELL' section with a 'Terminal' tab and a dropdown menu showing '(cloud-apps-demos-w24)'.

But we still have to deal with this!



The terminal screenshot shows a command prompt with the user 'erik_fredericks' at 'cloudshell'. The prompt is '~ (cloud-apps-demos-w24) \$'. The text '(cloud-apps-demos-w24)' is highlighted in red.

```
gcloud config set project <project id>
```

Anyway, here's the deployment

~~<https://cloud.google.com/shell/docs/deploy-app-engine-app>~~ BROKEN!

<https://codelabs.developers.google.com/codelabs/cloud-app-engine-python3>

We'll step through it together...

(After we get to the code, move on to the next slide because we have **EXTRA CONTENT TO SEE**)

Testing!

Prior to deploying to App Engine, we can test it locally

First, install the Python libraries

- A better Python program would do this in a virtual environment...
- `python3 -m pip install -r requirements.txt`

And then run the program

- `python3 main.py`

And then open the proxied debug view



Permissions error

If we see this:

ERROR: (gcloud.app.deploy) Permissions error fetching application [apps/cloudshell-appengine-test]. Please make sure that you have permission to view applications on the project and that <email> has the App Engine Deployer (roles/appengine.deployer) role

Then go to Cloud Build and enable it



Cloud Build API

[Google Enterprise API](#)

Continuously build, test, and deploy.

ENABLE

TRY THIS API [↗](#)

Permissions error

If we see this:

ERROR: (gcloud.app.deploy) Permissions error fetching application [apps/cloudshell-appengine-test]. Please make sure that you have permission to view applications on the project and that <email> has the App Engine Deployer (roles/appengine.deployer) role

Then go to IAM and give yourself the App Engine Deployer role

- If that doesn't work, try App Engine Admin

Edit access to "cloudshell-appengine-test"

Principal ?

erik.fredericks@gmail.com

Project

cloudshell-appengine-test

Assign roles

Roles are composed of sets of permissions and determine what the principal can do with this resource. [Learn more](#)

Role

Owner

IAM condition (optional) ?

+ ADD IAM CONDITION

Full access to most Google Cloud resources. See the list of included permissions.

Select a role

IAM condition (optional) ?

Filter app engine d

App Engine Deployer

Necessary permissions to deploy new code to App Engine, and remove old versions.

App Engine Memcache Data Admin

Can get, set, delete, and flush App Engine Memcache items.

App Engine Admin

Full management of App Engine apps (but not storage).

App Engine Service Admin

Can view and change traffic splits, scaling settings, and delete old versions; can't create new versions.

[MANAGE ROLES](#)

End result



So what do we have here?

- A Flask application running on the cloud
 - Could become anything! But, you now have a backend server that can do all kinds of incredible things
- Anything supported by App Engine can be hosted here!
 - Per usual though, check the \$\$ guide

Delete the project when done! You don't want bots hitting that URL!

...why?

Updating!

We have a neat thing in a bucket somewhere...

Update the HTML page:

- Grab the URL to the GIF and paste into the HTML

```

```

Redeploy app

```
$ gcloud app deploy --quiet
```

But wait...

Didn't work?

What's wrong?

Update URL and redeploy!

Permissions!

Need to be **very careful with this**

- Public access means anybody/thing can access
- What happens if somebody bots your site?
 - \$\$\$

But, assume we have the cash to support it, let's make it public

- Alternatively, you can grant fine-grained access to whatever cloud product needs the file

<https://cloud.google.com/storage/docs/access-control/making-data-public>

```
$ gcloud storage objects update gs://BUCKET_NAME/OBJECT_NAME  
--add-acl-grant=entity=AllUsers,role=READER
```

What you need to do today!

TBD