

CSCI 466: Networks

Cloud Computing, Deploying using AWS

Reese Pearsall
Fall 2024

Announcements

Workday on 11/22 and 12/2 (no class)

PA4 due 11/24

Flask App

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def example():
    return "Hello World!"

@app.route('/reese')
def reese():
    return "I am reese!"

@app.route('/raven')
def raven():
    name = request.headers['name']

    file = open("raven.txt", "r", encoding="utf8")
    filetext = file.read()

    return "Hello there, " + name + "\n \n" + filetext

if __name__ == '__main__':
    app.run()
```

Sending HTTP requests to API

```
C:\Users\reese>curl --header "name: reese" 127.0.0.1:5000
Hello, reese
```

```
Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
    While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
‘Tis some visitor,” I muttered, “tapping at my chamber door—
```

We have built our API, and we have tested it locally. We are ready to deploy it to the cloud!

<https://aws.amazon.com/console/>

First, we will need to make an account, and then log into the AWS console

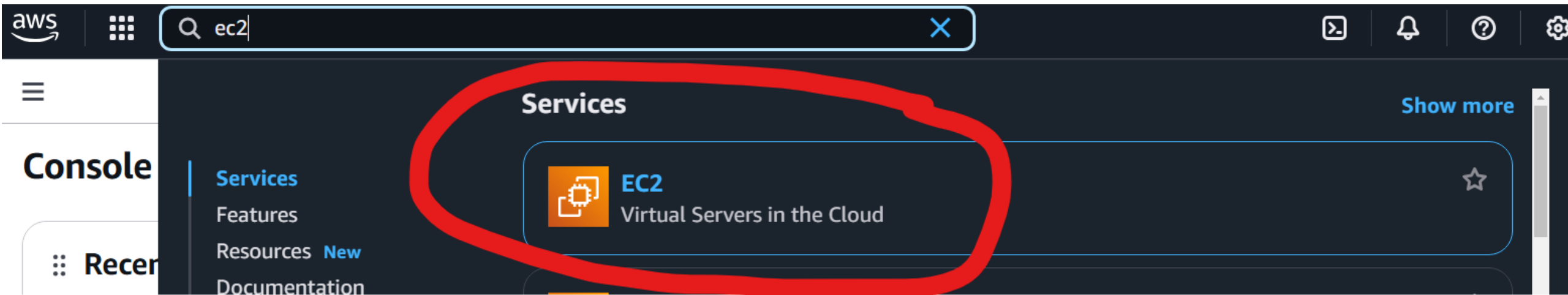
AWS Management Console

Everything you need to access and manage the AWS Cloud — in one web interface

Sign in

Amazon AWS

We will put our application on an **EC2** instance (a virtual server in the cloud)



Deploying with AWS

The screenshot displays the AWS Management Console interface. On the left, a navigation sidebar lists various services: Dashboard, EC2 Global View, Events, Instances (expanded), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (expanded), AMIs, AMI Catalog, Elastic Block Store (expanded), Volumes, Snapshots, and Lifecycle Manager. The main content area is divided into several panels. The top panel, 'Resources', shows a summary of EC2 resources in the US East (N. Virginia) Region, including 1 running instance, 0 auto scaling groups, 0 capacity reservations, 0 dedicated hosts, 6 key pairs, 0 elastic IPs, 0 load balancers, 7 security groups, 0 snapshots, 1 instance, 0 placement groups, and 1 volume. Below this is the 'Launch instance' panel, which is highlighted with a red circle and a red arrow pointing to the 'Launch instance' button. The 'Launch instance' panel includes a description, the 'Launch instance' button, a 'Migrate a server' button, and a note about the region. To the right of the 'Launch instance' panel is the 'Service health' panel, which shows the status of the EC2 service as 'operating normally'. On the far right, there are panels for 'Account attributes' and 'Explore AWS'.

Resources

EC2 Global View

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	1	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	1
Key pairs	6	Load balancers	0	Placement groups	0
Security groups	7	Snapshots	0	Volumes	1

Launch instance

To get started, launch an Amazon EC2 Instance, which is a virtual server in the cloud.

Launch instance ▼

Migrate a server

Note: Your instances will launch in the US East (N. Virginia) Region

Service health

AWS Health Dashboard

Region
US East (N. Virginia)

Status
✔ This service is operating normally.

Zones

Account attributes

Default VPC

vpc-0da6ffda4683eb8db

Settings

Data protection and security

Zones

EC2 Serial Console

Default credit specification

EC2 console preferences

Explore AWS

10 Things You Can Do Today to Reduce AWS Costs

Explore how to effectively manage your AWS costs without compromising on performance or capacity. [Learn more](#)

Save up to 90% on EC2 with Spot Instances

Optimize price-performance by combining EC2 purchase options in a single EC2 ASG. [Learn more](#)

Deploying with AWS

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

[Add additional tags](#)

Give your EC2 a name

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Our cloud machine will be an Ubuntu 22.04 web server

ubuntu

Ubuntu

Free tier eligible

Verified provider

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-005fc0f236362e99f (64-bit (x86)) / ami-07ee04759daf109de (64-bit (Arm))

Ubuntu Server 22.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Platform: ubuntu

Root device type: ebs

Virtualization: hvm

Select

☒ 64-bit (x86)

☐ 64-bit (Arm)

ENA enabled: Yes

▼ Instance type [Info](#) [Get advice](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

☐ All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

Use the t2.micro instance type

Deploying with AWS

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

[Add additional tags](#)

Give your EC2 a name

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Our cloud machine will be an Ubuntu 22.04 web server

ubuntu

Ubuntu

Free tier eligible

Verified provider

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-005fc0f236362e99f (64-bit (x86)) / ami-07ee04759daf109de (64-bit (Arm))

Ubuntu Server 22.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Platform: ubuntu

Root device type: ebs

Virtualization: hvm

Select

☒ 64-bit (x86)

☐ 64-bit (Arm)

ENA enabled: Yes

▼ Instance type [Info](#) [Get advice](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

☐ All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

Use the t2.micro instance type

Deploying with AWS

You will need to generate a key pair (this gives you a way to log into your cloud machine remotely)

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select



Create new key pair

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

csci466-pa5

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ RSA

RSA encrypted private and public key pair

☐ ED25519

ED25519 encrypted private and public key pair

Private key file format

☒ .pem

For use with OpenSSH

☐ .ppk

For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

Cancel

Create key pair

▼ Network settings Info

Edit

Network Info

vpc-0da6ffda4683eb8db

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

We'll create a new security group called 'launch-wizard-6' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0

☒ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Enable port 80 and 443 to allow for HTTP requests

Deploying with AWS

✓ Success

Successfully initiated launch of instance (i-0226d22bb4c6ff2ad)

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance ID = i-0226d22bb4c6ff2ad

All states

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
✓	csci-466-nov20	i-0226d22bb4c6ff2ad	Running	t2.micro	Initializing	View alarms +	us-east-1d	ec2-54-159-45-62.com...	54.159.45.62	-

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

i-0226d22bb4c6ff2ad (csci-466-nov20)

Connection Type

Connect using EC2 Instance Connect

Public IPv4 address

54.159.45.62

Username

ubuntu

Connect using EC2 Instance Connect Endpoint

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel

Connect

If you want to connect to your machine with SSH, you can follow the instructions there

Setting up our server

Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1015-aws x86_64)

* Documentation: <https://help.ubuntu.com>
* Management: <https://landscape.canonical.com>
* Support: <https://ubuntu.com/pro>

System information as of Wed Nov 20 09:17:04 UTC 2024

System load:	0.21	Processes:	108
Usage of /:	21.1% of 7.57GB	Users logged in:	0
Memory usage:	22%	IPv4 address for eth0:	172.31.23.249
Swap usage:	0%		

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: `sudo pro status`

The list of available updates is more than a week old.
To check for new updates run: `sudo apt update`

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in `/usr/share/doc/*/copyright`.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "`sudo <command>`".
See "`man sudo_root`" for details.

ubuntu@ip-172-31-23-249:~\$

We are now connected to
our cloud server !

Setting up our server

We need to run some commands to configure our machine:

// get python set up

```
sudo apt-get update
```

```
sudo apt-get install python3-venv
```

//create folder for our files and activate python

```
mkdir pa5
```

```
cd pa5
```

```
python3 -m venv venv
```

```
source venv/bin/activate
```

//install flask

```
pip install Flask
```

Setting up our server

Now, lets copy and paste over out `app.py` and our `raven.txt`

```
(venv) ubuntu@ip-172-31-41-227:~/pa5$ ls  
__pycache__  app.py  raven.txt  venv
```

Gunicorn

We will use gunicorn to manage our Flask application and forward requests to it

```
pip install gunicorn
```

We want to make sure gunicorn is always running, so we will make a background process for it

```
sudo nano /etc/systemd/system/pa5.service
```

```
[Unit]
Description=Gunicorn instance for a simple api
After=network.target
[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/pa5
ExecStart=/home/ubuntu/pa5/venv/bin/gunicorn -b localhost:8000 app:app
Restart=always
[Install]
WantedBy=multi-user.target
```

This file will make sure that on startup, it run the command

```
gunicorn localhost:8000 app:app
```

```
sudo systemctl daemon-reload
sudo systemctl start pa5
sudo systemctl enable pa5
```

Start the process

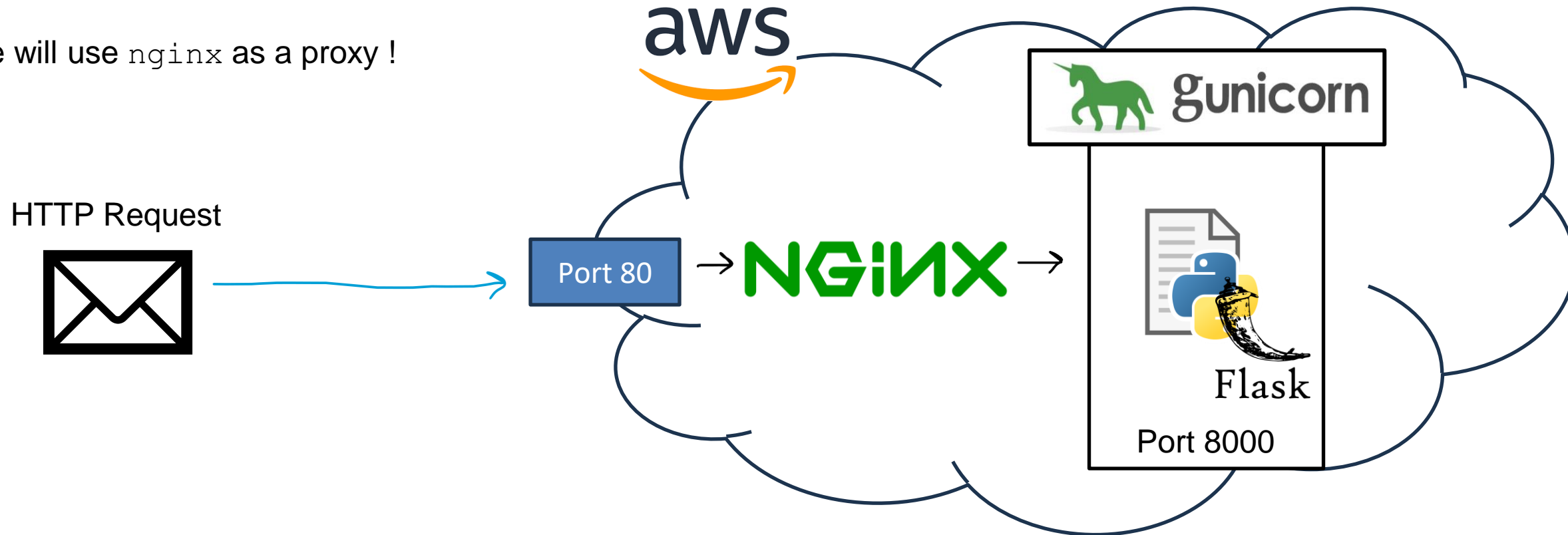
Our Flask API is running on port 8000 !

Nginx

Our app is listening on port 8000.

We want to take requests on port 80 (port for HTTP requests) and redirect them to gunicorn, which will redirect them to our Flask app!

We will use `nginx` as a proxy !



Nginx

Our app is listening on port 8000.

We want to take requests on port 80 (port for HTTP requests) and redirect them to gunicorn, which will redirect them to our Flask app!

We will use `nginx` as a proxy !

```
sudo apt-get install nginx  
sudo systemctl enable nginx
```

Then we need to configure nginx to reroute traffic

```
sudo nano /etc/nginx/sites-available/default
```


Nginx

```
sudo nano /etc/nginx/sites-available/default
```

```
#  
# In most cases, administrators will remove this file from sites-enabled/ and  
# leave it as reference inside of sites-available where it will continue to be  
# updated by the nginx packaging team.  
#  
# This file will automatically load configuration files provided by other  
# applications, such as Drupal or Wordpress. These applications will be made  
# available underneath a path with that package name, such as /drupal8.  
#  
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.  
##
```

```
# Default server configuration
```

```
upstream flaskapp {  
    server 127.0.0.1:8000;  
}
```

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;
```

```
# SSL configuration
```

```
# Add index.php to the list if you are using PHP  
index index.html index.htm index.nginx-debian.html;
```

```
server_name _;
```

```
location / {
```

```
    # First attempt to serve request as file, then  
    # as directory, then fall back to displaying a 404.  
    proxy_pass http://flaskapp;  
}
```

Listen on port 80

Route requests to this function

Forward to port 8000

```
sudo systemctl restart nginx
```

Testing

i-09878ba140f93f07c (csci466-pa5)

PublicIPs: 34.238.139.130 PrivateIPs: 172.31.41.227

We can now send curl requests to the public API of our cloud machine!

```
reese@Reese-PC MINGW64 ~  
$ curl 34.238.139.130  
Hello world!  
reese@Reese-PC MINGW64 ~  
$ curl --header "name: reese" 34.238.139.130/raven  
Hello there, reese  
  
Once upon a midnight dreary, while I pondered, weak and weary,  
Over many a quaint and curious volume of forgotten lore—  
While I nodded, nearly napping, suddenly there came a
```

Our cloud application is now online 24/7, and can accept requests!

PA5