UMD DATA605 - Big Data Systems

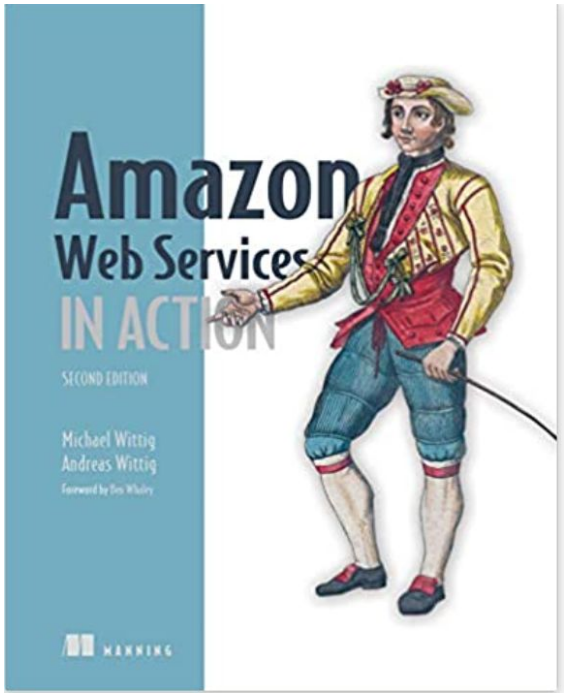
AWS Overview

Dr. GP Saggese

gsaggese@umd.edu

AWS: Resources

- Many tutorials on-line
- Mastery
  - Amazon Web Services in Action 2nd Edition
  - 3rd coming out soon
    - Amazon

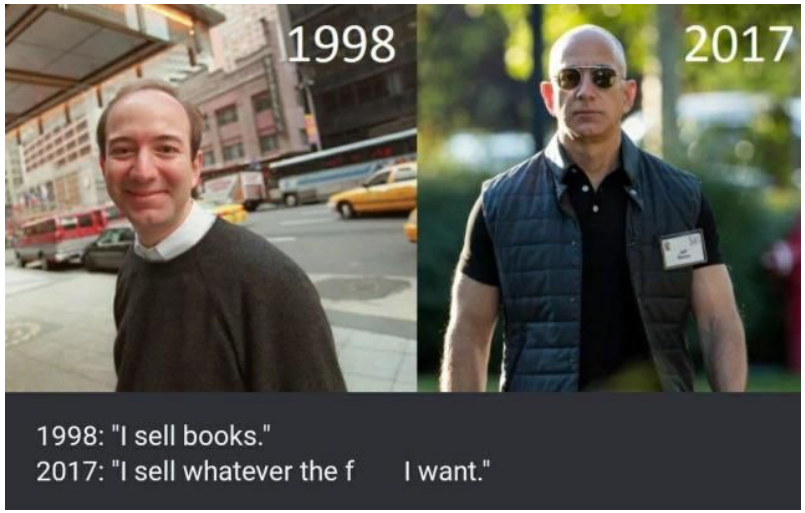


AWS

- AWS is a platform offering solutions for:
  - Computing (e.g., EC2)
  - Storing (e.g., S3)
  - Networking
- It offers different levels of abstractions
  - From virtual hardware to applications, e.g.,
  - Host web sites
  - Run enterprise software
  - Run machine learning applications
- Services can be controlled via:
  - A web interface
  - CLI
  - Programmatically (through language libraries, SDK)

AWS as Business

- Services are charged on a pay-per-use pricing model
- Data centers are distributed globally in US, Europe, Asia, South America
  - 500 new services and features every year
  - $62b dollar per year in revenue
  - It grows 42% year-over-year
  - It controls 30% of cloud business



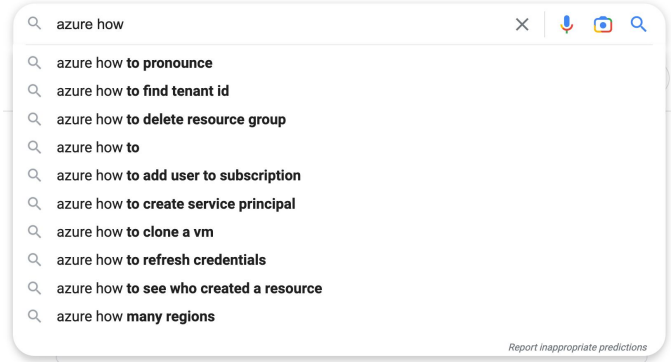
Beef Jezos

Types of Cloud Computing

- Cloud computing enables a shared pool of configurable computing resources
  - E.g., servers, storage, networks, applications, services
- From everywhere
- Convenient
- On-demand
- Clouds can be:
  - Public: open to use by general public (e.g., AWS)
  - Private: to virtualize and share IT infrastructure within a single organization (e.g., government)
  - Hybrid: a mixture of public and private clouds

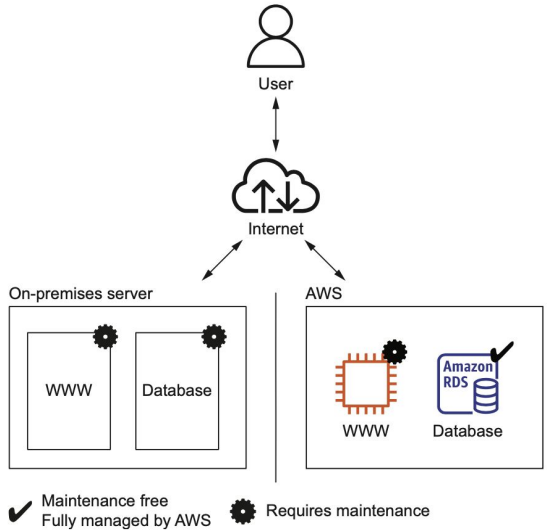
AWS vs Google Cloud vs Microsoft Azure

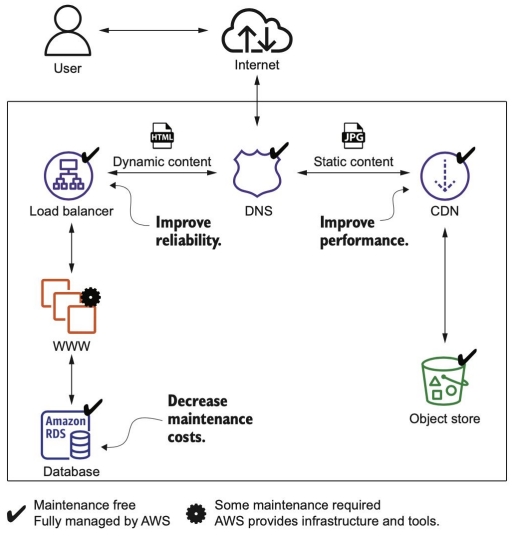
- **Similarities:**
  - Worldwide infrastructure
  - Provide computing, networking, storage as IAAS
    - Amazon EC2, Google Compute Engine, Azure VMs
    - Amazon S3, Google Cloud Storage, Azure Blob storage
  - Pay-as-you-go pricing model
- **Differences:**
  - AWS is market leader, most mature, and powerful
  - AWS (ab)uses a lot of open source technologies
  - Azure provides Microsoft stack in the cloud
  - Google seems more focused on cloud-native applications rather than migrating local applications to the cloud



From On-premise to AWS

- Consider a medium-sized e-commerce site with
  - Web-server: handle requests from customers
  - DB: stores product information and orders
- Web shop consists of:
  - Static content (e.g., `jpg`)
    - Delivered over a content delivery network (CDN)
    - Reduce load on company services
  - Dynamic content (e.g., `html`)
  - E.g., products and prices
  - Delivered by company web server
- AWS solution
  - Database, object store, DNS on AWS
- Potentially managed solutions
  - Use multiple smaller virtual services using load balancer to increase shop reliability



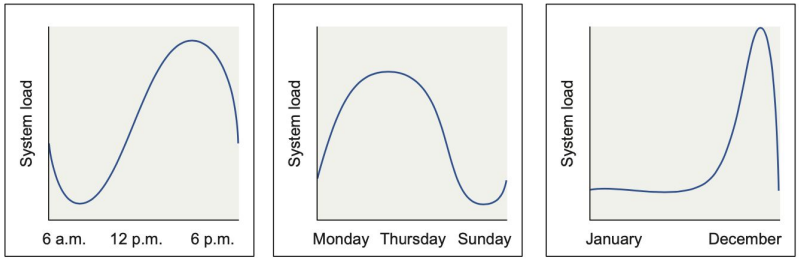


Low-Cost Batch Processing

- Analyze data once a day
- Batch jobs run on a schedule and store results in a DB
  - E.g., tools generate reports from the DB
- AWS bills VMs per minute
  - Instead of paying for machines in traditional data center, pay only when you run your job
- AWS offers spare capacity at a discount
  - It's not crucial to execute the batch jobs at a specific time
  - Run when there is capacity, e.g., saving 50%

Capacity Scaling

- No need to planning since you can schedule capacity on-the-fly
  - You don't have to predict capacity
  - No need to think about rackspace, switches, power supplies
  - If your system grows, you can add more VMs and storage
- You can scale:
  - From 1 to 1000 servers
  - From GB to PB
- You can handle seasonal traffic patterns by scaling up and down the services, e.g.,
  - Day vs night
  - Weekday vs weekend
  - Holiday
  - E.g., you don't need a test system when the team is off (at night and during the weekend)
- Worldwide presence
  - Since AWS has many data centers, you can deploy applications close to your customers
- Service scaling



- Most AWS services are fault-tolerant or highly available by default

Analytics	Application integration	AR and VR
AWS cost management	Blockchain	Business applications
Compute	Containers	Customer enablement
Database	Developer tools	End-user computing
Frontend web and mobile	Game Development	Internet of Things
Machine learning	Management and governance	Media services
Migration and transfer	Networking and content delivery	Quantum technologies
Robotics	Satellite	Security, identity, and compliance
Storage		

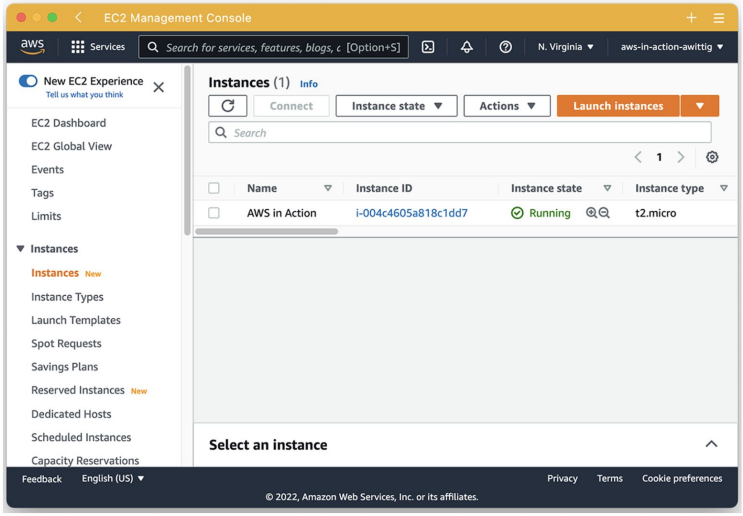
Pay-per-use

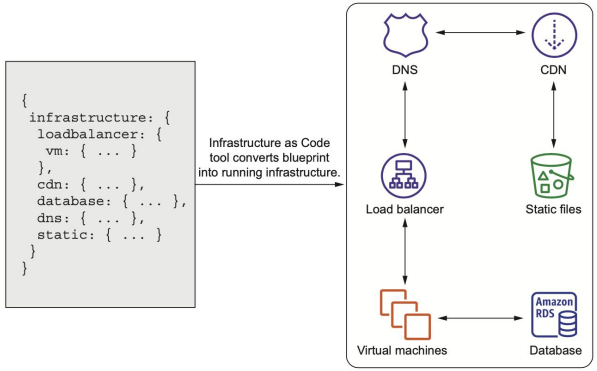
- Free tier
  - Use some AWS services for free for 12 months after signing up
  - Get experience using services (e.g., EC2, S3)
  - If you exceed the limits, you start paying (without further notice)
    - Need to set an alarm
- AWS bill
  - Similar to an electric bill
  - Services are billed based on usage (linear relationship), e.g.,
    - Hours of virtual server (typically rounded up to hours)
    - Used storage in GB (allocated or real capacity)
    - Data traffic in GB or number of requests
- Advantages
  - You don't need to make upfront investments or commitment in infrastructure
  - The price to start a project is lowered
  - One big server or two smaller ones with the same capacity have the same price
  - Easier to divide system into smaller parts, because the cost is the same
  - Make fault tolerance / high performance affordable

Service	January usage	February usage	February charge
Visits to website	100,000	500,000	
CDN	25 M requests + 25 GB traffic	125 M requests + 125 GB traffic	$115.00
Static files	50 GB used storage	50 GB used storage	$1.15
Load balancer	748 hours + 50 GB traffic	748 hours + 250 GB traffic	$19.07
Web servers	1 virtual machine = 748 hours	4 virtual machines = 2,992 hours	$200.46
Database (748 hours)	Small virtual machine + 20 GB storage	Large virtual machine + 20 GB storage	$133.20
DNS	2 M requests	10 M requests	$4.00
Total cost			***$472.88***

Interacting with AWS

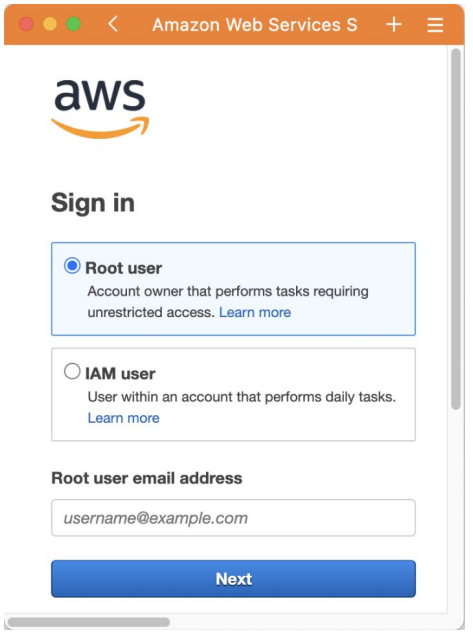
- AWS API allows to interact with services
- GUI (Management Console)
  - Easy to start interacting with services
  - Set up a cloud infrastructure for development and testing
- Command-line tool (CLI)
  - Manage and access AWS services
  - Automate recurring tools
- SDKs
  - Use library in almost any language to interact with AWS
  - Integrate applications with AWS
  - E.g., **`boto3`** for Python
- Blueprints
  - Description of your system containing all services and dependencies
  - It describes the system, it does not explain how to achieve it





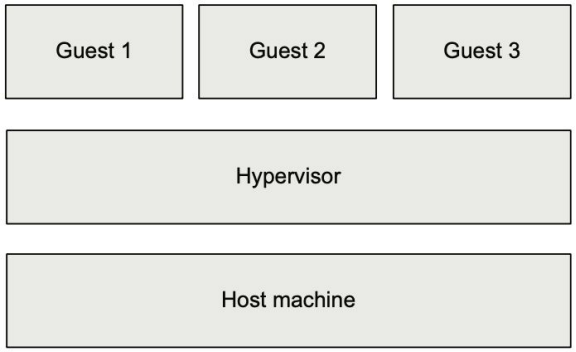
Accounts and Users

- Users
  - By default there is one root AWS account
  - You can attach multiple users to an account
  - Never use root account to develop!
  - Always use 2FA!
  - Multiple users can access an account with different privileges
  - You can create multiple accounts to isolate different workloads
  - You need a phone number and a credit card to sign up
- Key pair
  - To access a virtual server you need to create a key pair
  - Public key (in AWS and on virtual servers)
  - Private key is your secret (don't lose it, you can't retrieve it)



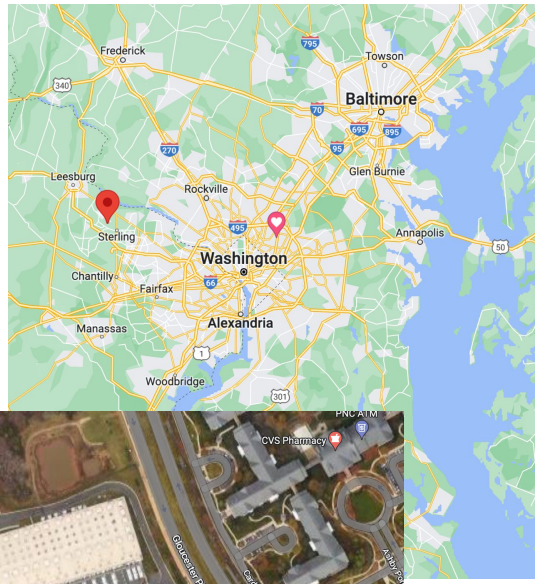
AWS VMs

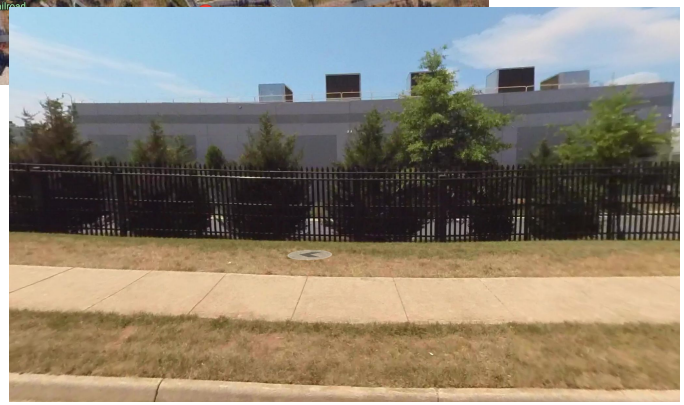
- Multiple VMs can run on the same hardware
- Virtual servers (aka "guests") are isolated from each other on the same
  - Physical server (aka "host machine", "bare metal")
  - Each physical server consists of CPUs, memory, networking interfaces, and storage
- Hypervisor (software + CPU hardware)
  - isolates guests
  - schedules requests to the hardware
- VMs can be started and stopped on-demand
- AWS used to use Xen (an open-source hypervisor)
- Today AWS uses Nitro (hardware assisted virtualization)
  - Performance close to bare metal



Starting an EC2 Instance

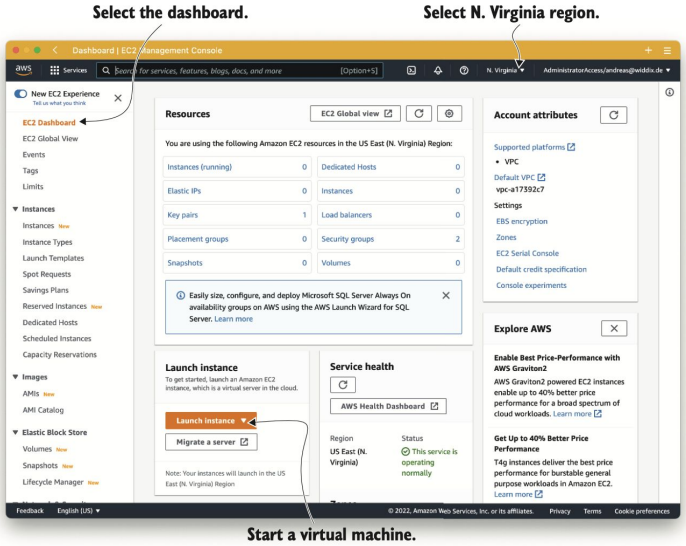
- Select region
  - **us-east-1**
  - 21155 Smith Switch Road, Ashburn, VA, USA

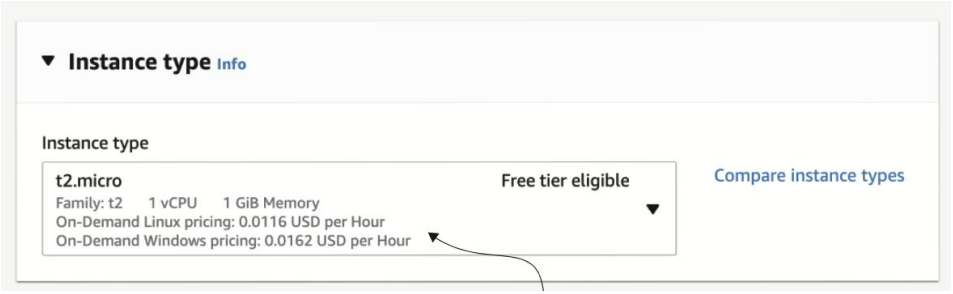


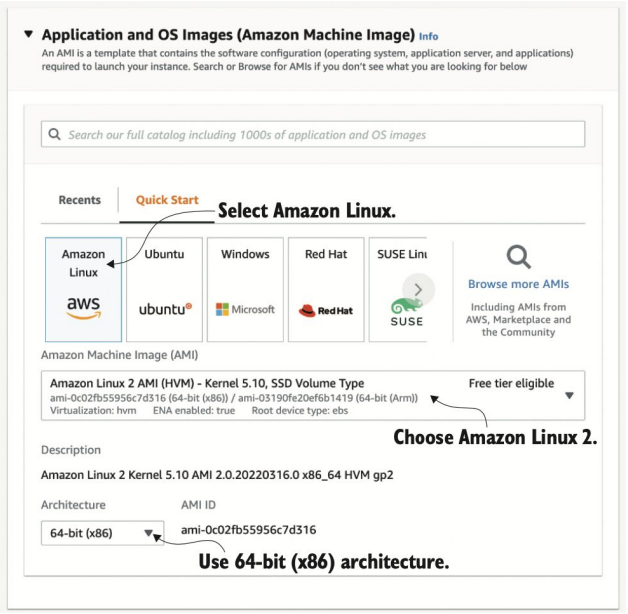


Starting an EC2 Instance

- Select OS
  - Amazon Machine Image (AMI)
  - Contain OS + pre-installed software
  - No need to spend (dev and machine) time installing packages
- Choose instance parameters
  - Later
- Configure instance
  - Network, shutdown behavior, termination protection, monitoring







AWS Instance Type

- https://aws.amazon.com/ec2/instance-types
- An "instance type" describes the amount of computing power available
- E.g., **`t2.micro`**
  - `t`: instance family (small, cheap)
  - `2`: generation (second)
  - `micro`: size (1 vCPU, 1GB memory)
  - 0.013 USD / hr
- E.g., **`m4.large`**
  - 2 CPUs
  - 8GB memory
  - 0.14 USD / hr
- Instance family
  - T: cheap, baseline
  - M: general purpose
  - C: compute optimized
  - R: memory optimized
  - D: storage optimized for HDD
  - I: storage optimized for SSD
  - F: with FPGAs
  - P, G, CG: with GPUs

M7g | Mac | M6g | M6i | M6in | M6a | M5 | M5n | M5zn | M5a | M4 | A1 | T4g | T3 | T3a | T2

Instance	vCPU*	CPU Credits / hour	Mem (GiB)	Storage	Network Performance
t2.nano	1	3	0.5	EBS-Only	Low
t2.micro	1	6	1	EBS-Only	Low to Moderate
t2.small	1	12	2	EBS-Only	Low to Moderate
t2.medium	2	24	4	EBS-Only	Low to Moderate
t2.large	2	36	8	EBS-Only	Low to Moderate
t2.xlarge	4	54	16	EBS-Only	Moderate
t2.2xlarge	8	81	32	EBS-Only	Moderate

AWS Instance Type

- Price on AWS website is somehow unclear
  - Burst mode
  - vCPUs
  - Multi-tenancy
  - On-demand vs spot vs reserved vs pre-paid
- https://instances.vantage.sh/
- 642 types of machines
- Min: $37 / yr
- Max: $1.91M / yr (500 CPUs and 24TB of mem)

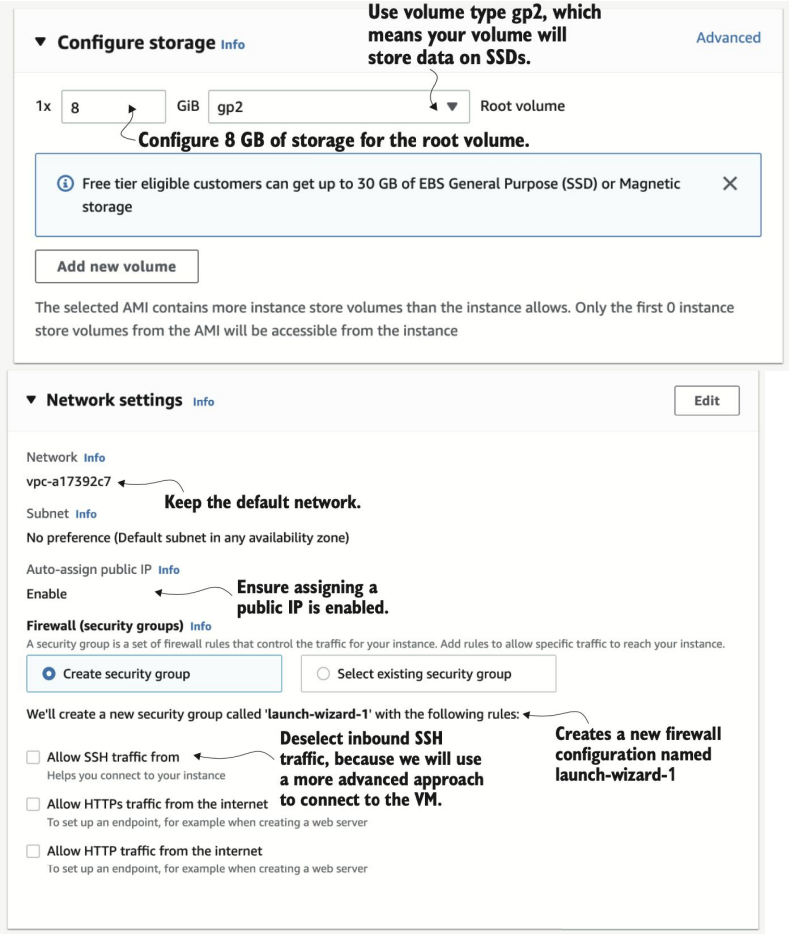
Compute	Value
vCPUs	448
Memory (GiB)	24576.0

Region: US East (N. Virginia) | Pricing Unit: Instance | Cost: Hourly | Reserved: 1-year - No Upfront | Columns | Compare Selected | Clear Filters | Export | Search...

Name	API Name	Instance Memory	vCPUs	Instance Storage	Network Performance	Linux On Demand cost
Filter...	Filter...	Min Mem: 0	Min vCPUs: 0	Min Storage: 0	Filter...	Filter...
T4G Nano	t4g.nano	0.5 GiB	2 vCPUs for a 1h 12m burst	EBS only	Up to 5 Gigabit	$0.0042 hourly
T3A Nano	t3a.nano	0.5 GiB	2 vCPUs for a 1h 12m burst	EBS only	Up to 5 Gigabit	$0.0047 hourly
T3 Nano	t3.nano	0.5 GiB	2 vCPUs for a 1h 12m burst	EBS only	Up to 5 Gigabit	$0.0052 hourly
T2 Nano	t2.nano	0.5 GiB	1 vCPUs for a 1h 12m burst	EBS only	Low to Moderate	$0.0058 hourly
T4G Micro	t4g.micro	1.0 GiB	2 vCPUs for a 2h 24m burst	EBS only	Up to 5 Gigabit	$0.0084 hourly
U-24TB1 112xlarge	u-24tb1.112xlarge	24576.0 GiB	448 vCPUs	EBS only	100 Gigabit	$218.4000 hourly
U-18TB1 112xlarge	u-18tb1.112xlarge	18432.0 GiB	448 vCPUs	EBS only	100 Gigabit	$163.8000 hourly
U-12TB1 112xlarge	u-12tb1.112xlarge	12288.0 GiB	448 vCPUs	EBS only	100 Gigabit	$109.2000 hourly

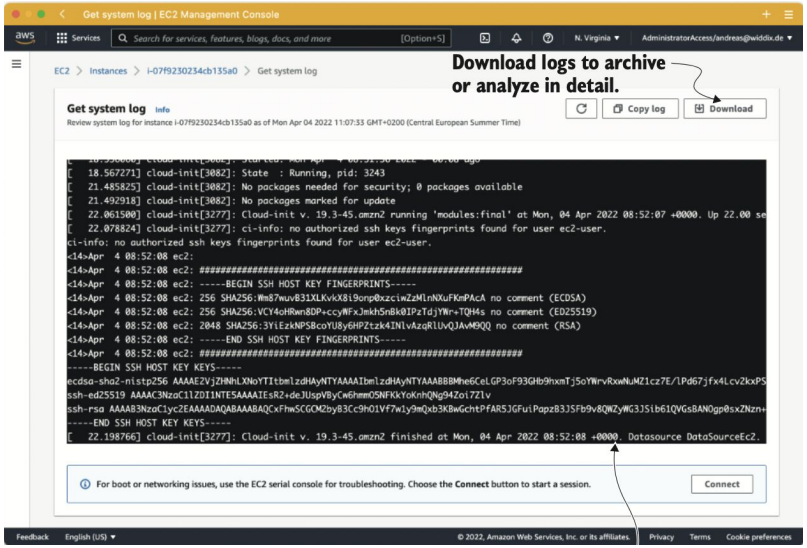
Starting an EC2 Instance

- Add storage
  - Size
  - Volume type (SSD or magnetic HDDs)
- Tag
- Configure a firewall
  - How to access using SSH
  - Select key-pair



Starting an EC2 Instance

- How to monitor the instance
- Start instance
- Find public IP
- Connect
- > ssh -i $PATH/mykey.pem ubuntu@$PUBLIC_UP
- > cat /proc/cpuinfo
- > free -m
- > sudo apt-get update
- > sudo apt-get install ...



States of a VM

- Start
  - You can start a stopped VM
- Stop
  - A stopped VM is not billed
  - Attached resources (e.g., HDD) will still incur in charges
  - Your data storage (local disk) will persist
  - The VM can be restarted later, but on a different host (with different IP)
- Reboot
  - Your data storage will persist
  - All software will still be installed after a reboot
  - The VM will restart but on a different host
- Terminate
  - It means delete: you cannot restart
  - Data storage is wiped out
  - Attached volumes persist

Moving / Upgrading EC2 instances

- Scale up / down
  - If you need more computing power you can increase the size of the VM
  - Stop the VM
  - Change instance type (e.g., `m3.large`)
  - Start the VM
  - Public and private IP addresses change
- Moving across AWS regions
  - Different regions have different distance from your users
  - Compliance
    - Are you allowed to store and process data in that country?
  - Service availability
    - Some AWS services are not available in certain regions
  - Redundancy
  - Costs
    - Service costs vary by region
  - Each region is a collection of data centers located in the same area
  - Regions are independent from each other
  - Data is not transferred between regions
  - Some AWS services (e.g., IAM, CDN, DNS) act globally
- Often you want an Elastic IP address to get a fixed public IP

Optimizing Costs

- On-demand instances
  - Maximum flexibility, no restrictions
  - You start and stop VMs when you want
  - Pay by hour
- EC2 / Compute saving plans
  - Term: 1 yr vs 3 yrs
  - Commit to a certain number of hours
  - Payment options: all, partial, no upfront
  - Discount: up to 3x cheaper than on-demand price
  - Useful for dev servers
- Capacity reservation
  - Get machines even in peak hours
- Spot instances
  - Bid for unused capacity
  - Price based on supply / demand
  - Discount: up to 10x cheaper than on-demand price
  - Useful for running asynchronous tasks

Programming th Infrastructure

- On AWS *everything* can be controlled via an API over HTTPS, e.g.,
  - Start a VM
  - Create 1TB of storage
  - Start Hadoop cluster
- Jeff Bezos 2002 API mandate
  - HackerNews
  - An eye-witness from Google about it
- You can use:
  - HTTP requests to the API
  - CLI (Command Line Interface): call AWS API from your terminal
  - SDK (Software Development Kit): call AWS API from your code
  - CloudFormation: templates that describe the state of the infrastructure and are translated into API calls

1. All teams will henceforth expose their data and functionality through service interfaces.
2. Teams must communicate with each other through these interfaces.
3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols – doesn't matter.
5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
6. Anyone who doesn't do this will be fired.
7. Thank you; have a nice day!

Jeff Bezos 2002 API mandate

Infrastructure-As-Code

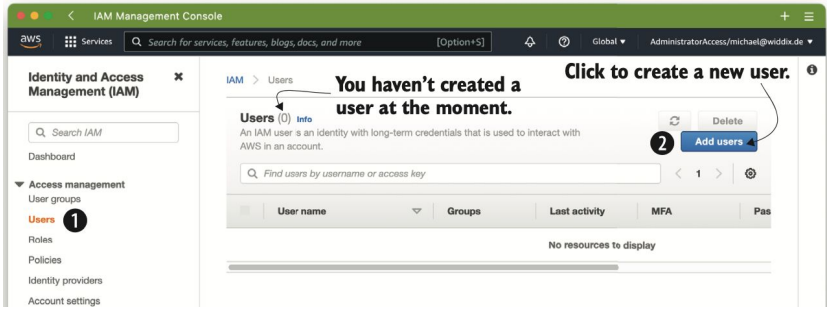
- Use high-level programming language to control IT systems
- You can apply software development to hardware:
  - Code repository
  - Automated tests
  - Continuous integration
- DevOps
  - Use software to bring development and operations closer together
  - Mix devs and ops in the same team
  - Devs are responsible for operational tasks (e.g., being on-call)
  - Ops are involved in software development, making system easier to operate
  - New role SRE in Google parlance
  - Foster communication and collaboration

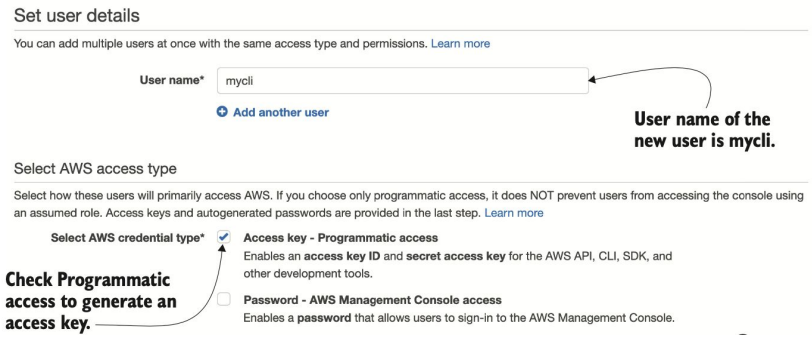
Infrastructure-as-Code: Advantages

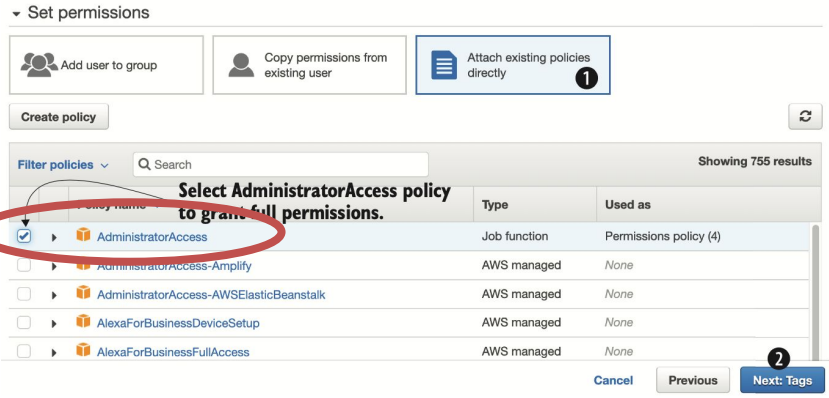
- Save time by:
  - Reusing scripts or ready-to-use blueprints
  - Automating tasks done regularly
- Fewer mistakes
- Consistency of actions
- Deployment pipeline
  - Multiple deploys per day
  - Commit to the repo
  - Source code is built
  - Automatic tests (e.g., integration tests)
  - Build testing environment
  - Run acceptance tests in isolation
  - Changes are propagated to production
  - Monitor production
- The script / blueprint is a detailed documentation

Create User Account

- Do not to use AWS root account for everything
- Create a new user
- IAM = Identity and Access Management
- Access key ID + secret access key
- Access control
  - Enable programmatic access
  - Enable console access
  - Limit what an user can do through policies







Bad idea!

AWS Command Line Interface (CLI)

- Provide unified interface to all AWS services
- Output is in JSON format

```
> apt-get install awscli
```

- Authenticate

```
> aws configure
```

  - AWS access key ID
    - E.g., `AKIAIRUR3YLPOSVD7ZCA`
  - AWS secret access key
    - E.g., `SSKIng7jkAKERpcT3YphX4cD87sBYgWVw2enqBj7`
  - Default region name
    - E.g., `us-east-1`

- Execute command

```
> aws <service> <action> --key value
```

```
$ aws ec2 describe-regions
{
  "Regions": [
    {
      "Endpoint": "ec2.eu-north-1.amazonaws.com",
      "RegionName": "eu-north-1",
      "OptInStatus": "opt-in-not-required"
    },
    [...]
    {
      "Endpoint": "ec2.us-west-2.amazonaws.com",
      "RegionName": "us-west-2",
      "OptInStatus": "opt-in-not-required"
    }
  ]
}
```

Software Development Kit (SDK)

- SDK is a library that call to AWS API from your favorite programming language
- E.g., Python, Go, Ruby, C++, JavaScript
- Handles
  - Authentication
  - Retry on error
  - HTTPs communication
  - XML / JSON de-/serialization
- Cons
  - Imperative approach
  - Need to deal with dependencies

```python
import boto3

# Get the service resource.
dynamodb = boto3.resource('dynamodb')

# Create the DynamoDB table.
table = dynamodb.create_table(
    TableName='users',
    KeySchema=[
        {
            'AttributeName': 'username',
            'KeyType': 'HASH'
        },
        {
            'AttributeName': 'last_name',
            'KeyType': 'RANGE'
        }
    ],
    AttributeDefinitions=[
        {
            'AttributeName': 'username',
            'AttributeType': 'S'
        },
        {
            'AttributeName': 'last_name',
            'AttributeType': 'S'
        },
    ],
    ProvisionedThroughput={
        'ReadCapacityUnits': 5,
        'WriteCapacityUnits': 5
    }
)

# Wait until the table exists.
table.wait_until_exists()

# Print out some data about the table.
print(table.item_count)
```

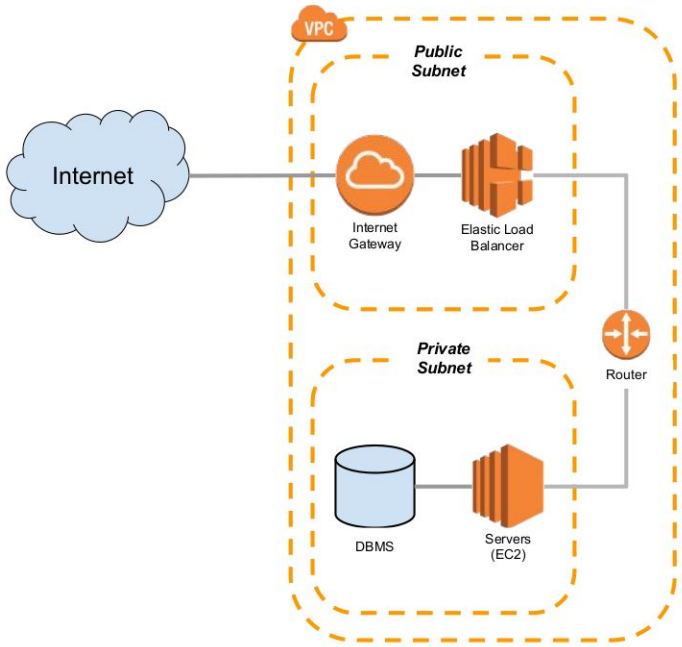
AWS CloudFormation

- AWS CloudFormation uses templates to describe infrastructure
  - E.g., JSON, YAML
- Declarative vs imperative approach
  - Declare the system, rather than listing the necessary steps to build the system
- Pros
  - Consistent way to describe the infrastructure
    - People implement the same specs in a different way
  - Handle dependencies
  - Replicable
  - Customizable
    - Inject custom parameters to customize templates
  - Testable
    - Create infrastructure from a template, tests, and shut it down
  - Updatable
    - If you update the template, CloudFormation automatically applies changes to the infrastructure
  - It is documentation
    - JSON or YAML document
    - Use it as code in a VCS
- AWS Stacks processes CloudFormation templates

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Metadata:
  License: Apache-2.0
Description: 'AWS CloudFormation Sample Template DynamoDB_Table: This template demonstrates
  the creation of a DynamoDB table.  **WARNING** This template creates an Amazon DynamoDB
  table. You will be billed for the AWS resources used if you create a stack from
  this template.'
Parameters:
  HashKeyElementName:
    Description: HashType PrimaryKey Name
    Type: String
    AllowedPattern: '[a-zA-Z0-9]*'
    MinLength: '1'
    MaxLength: '2048'
    ConstraintDescription: must contain only alphanumberic characters
  HashKeyElementType:
    Description: HashType PrimaryKey Type
    Type: String
    Default: S
    AllowedPattern: '[S|N]'
    MinLength: '1'
    MaxLength: '1'
    ConstraintDescription: must be either S or N
  ReadCapacityUnits:
    Description: Provisioned read throughput
    Type: Number
    Default: '5'
    MinValue: '5'
    MaxValue: '10000'
    ConstraintDescription: must be between 5 and 10000
Resources:
  myDynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
      - AttributeName: !Ref 'HashKeyElementName'
        AttributeType: !Ref 'HashKeyElementType'
      KeySchema:
      - AttributeName: !Ref 'HashKeyElementName'
        KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: !Ref 'ReadCapacityUnits'
        WriteCapacityUnits: !Ref 'WriteCapacityUnits'
Outputs:
  TableName:
    Value: !Ref 'myDynamoDBTable'
    Description: Table name of the newly created DynamoDB table
```

Securing Your System

- Install software updates
  - New security vulnerabilities are found and fixed every day
- Restrict access to your AWS account
  - Multiple persons and scripts are accessing an AWS account
  - Principle of "least privilege": grant only permissions needed to perform a job
- Control network traffic
  - Leave open only the ports that you strictly need
  - Close everything else
    - E.g., 80 for HTTP traffic and 443 for HTTPS
  - Encrypt traffic (and data)
- Create a private network in AWS
  - Subnets that are not reachable from the Internet



AWS Shared-responsibility Principle

- AWS is responsible for:
  - Protect network through monitoring Internet access (e.g., prevent DDoS attacks)
  - Ensuring physical security of data centers
  - Decommissioning storage devices after end of life
- You are responsible for:
  - Restrict access using IAM
  - Encrypting network traffic (e.g., HTTPS)
  - Configuring firewall for VPN
  - Encrypting data
  - Updating OS (kernel, libs) and software
- More info here