End-to-End AWS EC2 Setup for GPT-2 Prototype

P charan Naga Deep

May 8, 2025

Abstract

This document provides detailed, step-by-step instructions to provision and access an AWS EC2 instance for prototyping a GPT-2 model from scratch. We begin with an overview of AWS global architecture and EC2 internals (with diagram placeholders), then run through all necessary CLI commands for key-pair management, networking, instance launch, and SSH access. We will temporarily work on a `t2.micro` EC2 instance for development and later shift to a GPU Spot instance for full-scale training.

1 AWS Global Architecture

Amazon Web Services (AWS) is organized into *Regions* and *Availability Zones (AZs)*. Each Region (e.g. `us-east-1`, N. Virginia) contains multiple AZs (e.g. `us-east-1a`, `us-east-1b`) to ensure high availability and fault tolerance.

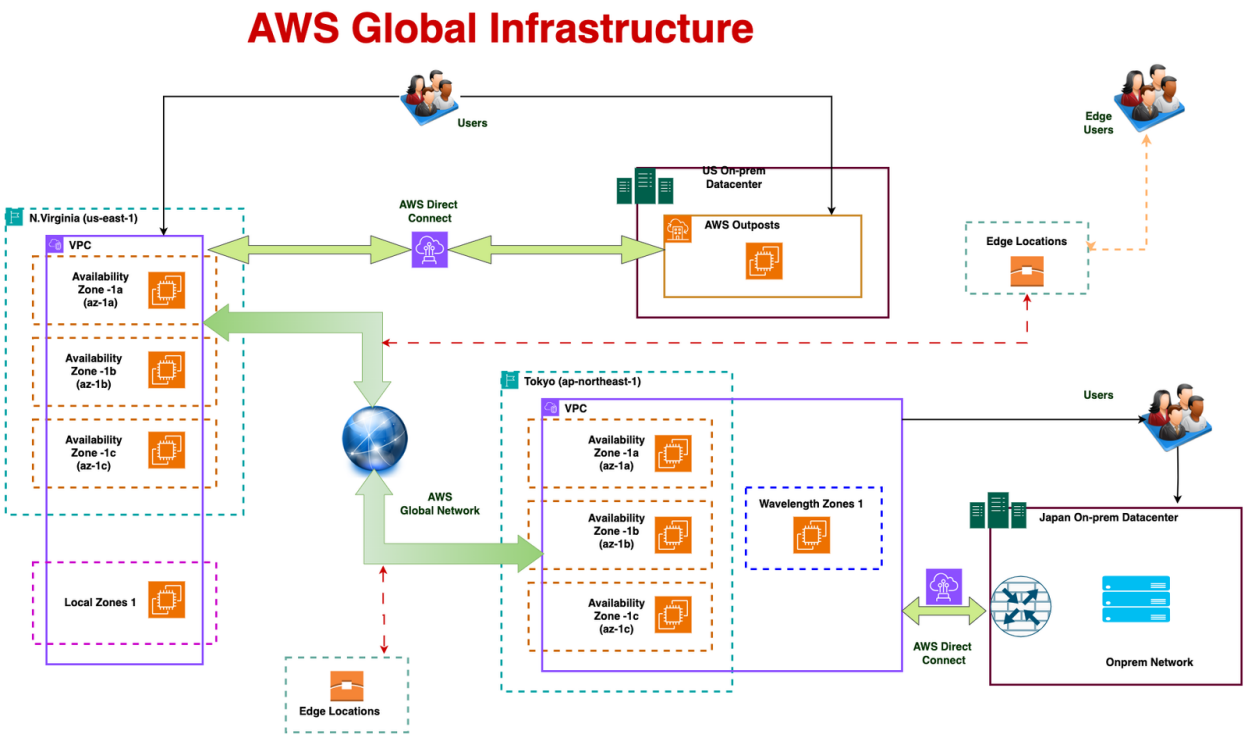

Figure 1: AWS Global Infrastructure Diagram

Core concepts:

- *Region*: Geographical area with multiple data centers.
- *Availability Zone*: Isolated data center within a Region.
- *VPC*: A *Virtual Private Cloud*, i.e. a logically isolated virtual network.
- *IAM*: *Identity and Access Management*, the service for users, groups, and roles.

2 Amazon EC2 Architecture

EC2 (*Elastic Compute Cloud*) provides resizable compute capacity in the AWS cloud. "Cloud computing" is the on-demand delivery of computing resources—like servers, storage, and databases—over the internet. EC2 uses *virtualization*, which is creating virtual (rather than physical) versions of computing resources. This is managed by a *hypervisor*—specifically, the AWS *Nitro Hypervisor*, a lightweight, hardware-accelerated layer that isolates and allocates physical host resources to virtual machines.

An EC2 *instance* is a virtual server: it behaves like a physical machine but runs as a software guest on shared hardware. Each instance lives in a *VPC*, attaches *EBS (Elastic Block Store) volumes* for persistent block storage, and is secured by *Security Groups* and accessed via *Key Pairs*. Traffic routing and internet connectivity within a VPC is handled by *Subnets*, *Route Tables*, and an *Internet Gateway*.

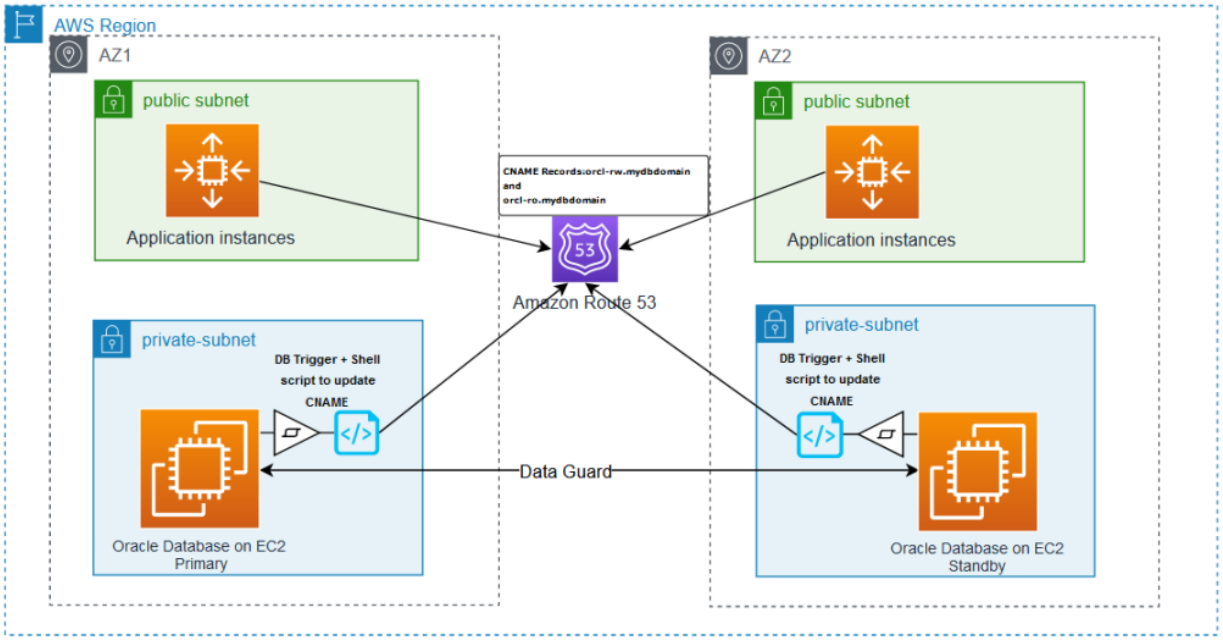

Figure 2: EC2 High-Level Architecture

Key Terminologies

Elastic Compute Cloud (EC2)

The AWS service for launching and managing virtual servers.

Cloud Computing
Delivery of compute and storage resources over the internet on a pay-as-you-go basis.

Virtualization
Creating virtual versions of hardware resources to run multiple isolated systems on one physical host.

Hypervisor
Software layer that creates and manages virtual machines; the *Nitro Hypervisor* is AWS's custom implementation.

Instance
A virtual server launched on EC2; essentially a virtual machine.

VPC (Virtual Private Cloud)
A logically isolated network in AWS, containing *Subnets* (sub-divisions of IP address ranges), *Route Tables* (rules for traffic flow), and an *Internet Gateway* (connects the VPC to the internet).

EBS (Elastic Block Store) Volume
Network-attached block storage that persists independently of the instance lifecycle.

Block Storage
Storage that provides raw volumes (like a physical disk) for data read/write operations.

Security Group
A virtual firewall attached to instances; it is *stateful*, meaning return traffic for allowed requests is automatically permitted.

Key Pair
An SSH public/private key pair for secure login. Implements *public-key cryptography*, a system using a public key (shared) and a private key (kept secret).

Instance State
The lifecycle status of an instance: **pending** → **running** → **stopping** → **stopped** → **terminated**.

3 Step-by-Step Setup (with Explanations)

Execute the following commands sequentially in a Linux shell (WSL Ubuntu, EC2, or local Linux).

3.1 Install AWS CLI

```
sudo apt update
sudo apt install -y unzip curl
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o awscliv2.zip
unzip awscliv2.zip
sudo ./aws/install
```

Explanation:

- `sudo apt update`: Refreshes the local package index so you get the latest versions.
- `sudo apt install -y unzip curl`: Installs `unzip` (to extract zip archives) and `curl` (to download files from the web). The `-y` flag auto-confirms prompts.
- `curl ... -o awscliv2.zip`: Downloads the AWS CLI v2 installer archive to `awscliv2.zip`.
- `unzip awscliv2.zip`: Extracts the installer files.
- `sudo ./aws/install`: Runs the extracted installer, placing the `aws` binary in your `/usr/local/bin` so you can run `aws` commands globally.

3.2 Configure AWS CLI

```
aws configure
# Enter your AWS Access Key ID
# Enter your AWS Secret Access Key
# Default region name: us-east-1
# Default output format: json
```

Explanation:

- `aws configure` prompts you for your IAM credentials and default settings.
- It writes your Access Key, Secret Key, region, and output format into `$HOME/.aws/credentials` and `$HOME/.aws/config`.
- `us-east-1` sets the default AWS Region to North Virginia.

3.3 Generate an SSH Key Pair

```
ssh-keygen -t rsa -b 4096 -m PEM -f ~/gpt2-new-key.pem -N ""
aws ec2 import-key-pair \
  --key-name gpt2-new-key \
  --public-key-material fileb://~/gpt2-new-key.pem.pub
```

Explanation:

- `ssh-keygen -t rsa -b 4096 -m PEM -f ... -N ""` creates a new 4096-bit RSA private key in PEM format, saving to `~/gpt2-new-key.pem` with no passphrase, and a matching `.pub` file.
- `aws ec2 import-key-pair` uploads the public key to AWS as `gpt2-new-key` so EC2 instances can inject it into `~/.ssh/authorized_keys`.

3.4 Create a Security Group

```
MY_IP=$(curl -s https://checkip.amazonaws.com)/32
SG_ID=$(aws ec2 create-security-group \
  --group-name gpt2-prototype-sg \
  --description "SSH only from my IP" \
  --query GroupId --output text)
aws ec2 authorize-security-group-ingress \
  --group-id $SG_ID --protocol tcp --port 22 --cidr $MY_IP
```

Explanation:

- `MY_IP=$(curl ...)/32` fetches your public IP and appends `/32` to form a CIDR block (single-host).
- `aws ec2 create-security-group` makes a new firewall rule set named `gpt2-prototype-sg` and returns its ID.
- `authorize-security-group-ingress` adds an inbound rule allowing TCP port 22 (SSH) only from your IP.

3.5 Find Latest Ubuntu 22.04 AMI

```
AMI_ID=$(aws ec2 describe-images \
  --owners 099720109477 \
  --filters \
    "Name=name,Values=ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*" \
    "Name=state,Values=available" \
  --query 'sort_by(Images,&CreationDate)[-1].ImageId' \
  --output text)
echo "Using AMI: $AMI_ID"
```

Explanation:

- `describe-images` lists AMIs owned by Canonical (099720109477) matching the Ubuntu 22.04 pattern.
- The filters restrict to available (‘state=available‘) images.
- `sort_by(...)[-1]` picks the newest one by creation date.
- `AMI_ID` holds the resulting AMI identifier for launching.

3.6 Launch the EC2 Instance

```
aws ec2 run-instances \
  --image-id $AMI_ID \
  --instance-type t2.micro \
```

```
  --key-name gpt2-new-key \
  --security-group-ids $SG_ID \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=gpt2-prototype
    }]'
```

Explanation:

- `run-instances` requests a new EC2 VM using the specified AMI and hardware profile (`t2.micro`).
- `-key-name` tells EC2 to install your SSH public key.
- `-security-group-ids` applies your firewall rules.
- `-tag-specifications` labels the instance with `Name=gpt2-prototype` for easy identification.

3.7 Obtain the Public DNS

```
PUBLIC_DNS=$(aws ec2 describe-instances \
  --filters \
    "Name=tag:Name,Values=gpt2-prototype" \
    "Name=instance-state-name,Values=running" \
  --query 'Reservations[0].Instances[0].PublicDnsName' \
  --output text)
echo $PUBLIC_DNS
```

Explanation:

- `describe-instances` filters to your running `gpt2-prototype` instance.
- The `PublicDnsName` field contains the hostname you'll SSH into.
- Storing it in `PUBLIC_DNS` and echoing makes the next step easy.

3.8 SSH into the Instance

```
chmod 400 ~/gpt2-new-key.pem
ssh -i ~/gpt2-new-key.pem ubuntu@$PUBLIC_DNS
```

Explanation:

- `chmod 400` ensures only you can read the private key (SSH requires strict permissions).
- `ssh -i ... ubuntu@$PUBLIC_DNS` starts a secure shell session as the `ubuntu` user on the EC2 VM, using your private key for authentication.