AWS Hosting: Step-by-Step Guide

This guide walks you through hosting a simple server on Amazon Web Services (AWS). We cover creating an AWS account, launching an EC2 instance (virtual server), assigning an Elastic IP, configuring security (firewall) rules, and connecting via SSH. Each section includes best practices and tips to help beginners avoid common pitfalls and control costs.

1. Create an AWS Account

- **Sign up and verify.** Go to aws.amazon.com and click **Create an AWS Account**. Enter your email address (this becomes the "Root" user), choose an account name, and set a strong password [1] [2]. AWS will email you a 6-digit verification code – enter it to verify your email address [1]. You may need to solve a CAPTCHA at this step.
- **Enter your details.** Provide your contact information (name, address, etc.) and accept the AWS Customer Agreement. Choose *Personal* for a learning/test account unless you are creating a business account.
- **Add payment info.** AWS requires a credit or debit card to prevent fraud, even for Free Tier users [2]. On the billing page, enter your card details. AWS will place a temporary $1 authorization on your card to verify it (this is refunded)[3]. Make sure the card has at least $1 available for this hold [1]. No charges will occur as long as you stay within Free Tier limits.
- **Verify your phone.** AWS will ask you to verify your phone number by sending a text or voice call with a code. Enter the code to confirm.
- **Set up support plan.** Choose the **Basic (Free)** support plan, which is sufficient for new users.

Once your account is active, you'll land in the AWS Management Console. You can now access AWS services (watch out for region selection in the top-right – choose a region close to you).

Best practice: Consider enabling multi-factor authentication (MFA) on your root account and creating an IAM user for daily use. This improves security by reducing reliance on the root user. Also, monitor billing alerts to avoid unexpected charges [2].

2. Launch an EC2 Instance

Now create a virtual machine (EC2 instance) on AWS:

- **Open EC2 service.** In the AWS Console, search for and select **EC2** under Compute services.
- **Click *Launch Instance*.** On the EC2 Dashboard, click **Launch Instance** [2]. This starts the setup wizard.
- **Name and OS image.** Give the instance a friendly name (for example, "MyUbuntuServer"). Choose an Amazon Machine Image (AMI) that contains the OS; for example, select **Ubuntu Server 22.04 LTS (HVM), SSD Volume Type**.

- **Choose instance type.** For the Free Tier, pick **t2.micro** (or **t3.micro**) – these small instance types are free for up to 750 hours per month for the first year[6] [7]. Using a larger instance will incur charges.
- **Configure key pair (SSH key).** In **Key pair (login)** settings, select **Create a new key pair**. Name the key (e.g. "my-aws-key") and download the resulting `.pem` file [8]. This file is your SSH private key—store it safely, and do **not** share it. You'll need it to log in later [8].
- **Configure network/security.** When prompted to configure the security group (firewall), add a rule for SSH: allow **TCP port 22** from your own IP address (choose "My IP"[9]). This ensures only you can SSH into the server. If you plan to run a web server, also add rules for HTTP (TCP port 80) and HTTPS (TCP port 443) from anywhere (0.0.0.0/0).
- **Review and launch.** Accept defaults for storage (usually 8 GB) and other settings. Finally, click **Launch Instance** (confirm the key pair selection when prompted).

AWS will initialize the instance. In the EC2 console under **Instances**, you'll see your new instance with status "running" once it's ready. It will have a public IP assigned (next we'll make that IP static).

Best practice: Use the Free Tier instance type (t2.micro/t3.micro) to avoid costs [6] [7]. Tag your instance with a Name so you can easily identify it later. Do **not** disable or lose the `.pem` key file—without it, you cannot SSH in.

3. Allocate an Elastic IP Address

By default, the public IP of an EC2 instance can change when you stop/start it. To have a fixed address:

- **Go to Elastic IPs.** In the EC2 console sidebar under **Network & Security**, click **Elastic IPs**.
- **Allocate a new address.** Click **Allocate Elastic IP address**, then **Allocate**. AWS will provide a public IPv4 address (an "Elastic IP").
- **Associate with your instance.** Select the new Elastic IP, click **Actions → Associate Elastic IP address**. Choose your running instance from the list and confirm. The Elastic IP is now linked to your server's network interface.

You can now use this static IP to connect to your instance.

Tip: AWS now charges **$0.005 per hour** for each public IPv4 address in use (effective Feb 2024)[10]. Under the Free Tier, you get 750 hours of public IPv4 usage per month for 12 months[11], which covers one running instance. To avoid extra fees, release any Elastic IPs you no longer need, and don't allocate more than necessary.

4. Configure Security Group (Inbound Rules)

Security Groups are AWS's virtual firewalls for your instance[12]. You must allow incoming traffic on needed ports:

- **Edit inbound rules.** Select your instance, go to the **Security** tab, and click the security group link. In the security group details, click **Edit inbound rules**.
- **Allow SSH (port 22).** Ensure there is a rule: Type **SSH**, Protocol **TCP**, Port Range **22**, Source set to your IP (or a limited range) [9]. This lets you SSH in. If this rule is missing, add it.

- **Allow web (ports 80/443) [optional].** If you're serving a website or API, add rules for **HTTP (TCP 80)** and **HTTPS (TCP 443)**. For HTTP(S), Source can be **Anywhere (0.0.0.0/0)**, but only do this if your server should be public.
- **Save rules.** Click **Save rules** after adding the necessary entries.

Security tip: Do **not** open more ports than needed, and restrict sources whenever possible. For example, if only you should SSH in, don't set SSH source to "Anywhere." By default, AWS security groups are stateful: once outbound traffic is allowed (usually all outbound), the return traffic is automatically allowed[1][2] . But inbound must be explicitly opened.

5. Connect to Your Server via SSH

Now SSH into your Ubuntu server from your local machine:

- **Set key permissions (Linux/macOS).** In your terminal, navigate to where you saved the key (e.g. `~/Downloads`) and run:

  ```
  chmod 400 my-aws-key.pem
  ```

  This restricts the key file's permissions as required by SSH.
- **SSH command.** Connect using the `ssh` command. For Ubuntu instances, the default user is `ubuntu`. Use the Elastic IP address from step 3:

  ```
  ssh -i "/path/to/my-aws-key.pem" ubuntu@<your-elastic-ip>
  ```

  Replace `"/path/to/my-aws-key.pem"` with the actual path to your `.pem` file and `<your elastic-ip>` with the IP (e.g. `54.123.45.67`).
- **First-time login.** On first connection, you may see a warning like "The authenticity of host cannot be established." Type `yes` to continue. You should then be logged into the server's shell.
- **Windows note:** On Windows, you can use PowerShell or the Windows Subsystem for Linux (WSL) with the same `ssh` command. If using PuTTY, you must convert the `.pem` to a PuTTY `.ppk` key using PuTTYgen.

Reminder: Keep your `.pem` file safe. Anyone who has it can access your server. Never share it or upload it to public code repositories.

Optional: Best Practices & Cleanup

- **Terminate unused resources.** AWS charges for running instances and other resources (storage, Elastic IPs, etc.). If you stop using the server, **terminate** the instance in the console to avoid charges. Also **release** any unused Elastic IPs (free for one attached address, charged otherwise[2]).
- **Monitor costs.** Use the AWS Billing dashboard or AWS Budgets to set alarms and watch your usage [2] . This helps ensure you stay within the Free Tier limits or your desired budget.

- **Use IAM users.** Instead of always using the root account, create an IAM user with limited privileges for daily operations. This improves account security.
- **Network security.** In production, consider using SSH key passphrases and/or tools like Amazon EC2 Instance Connect or Session Manager (which allow SSH without exposing port 22). AWS security is a shared responsibility [12], so apply standard hardening (update software, use firewalls, etc.).
- **Data backup.** If this is for a real project, back up any important data (e.g., using EBS snapshots or S3).

By following these steps with care, beginners can launch and manage a basic AWS-hosted server. AWS provides documentation and free-tier benefits to help you learn cloud hosting without upfront costs [6] [2].

References: Official AWS guides and documentation were used to ensure accurate, up-to-date information [6] [8] [10] [3].

---

[1] [3] [4] How to Create an AWS Free Tier Account – A Step-by-Step Guide
https://www.freecodecamp.org/news/how-to-create-an-aws-free-tier-account/

[2] AWS Free Tier FAQs – How to use AWS Cloud Services for Free
https://aws.amazon.com/free/registration-faqs/

[5] [6] Tutorial 1: Launch my very first Amazon EC2 instance - Amazon Elastic Compute Cloud
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/tutorial-launch-my-first-ec2-instance.html

[7] [8] [9] How to Launch an Ubuntu EC2 Instance in AWS: A Step-by-Step Guide - DEV Community
https://dev.to/anil_kumar_/how-to-launch-an-ubuntu-ec2-instance-in-aws-a-step-by-step-guide-d5e

[10] [11] [13] New – AWS Public IPv4 Address Charge + Public IP Insights | AWS News Blog
https://aws.amazon.com/blogs/aws/new-aws-public-ipv4-address-charge-public-ip-insights/

[12] Amazon EC2 security groups for your EC2 instances - Amazon Elastic Compute Cloud
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html

By: Mohammed Galal, IT-L3