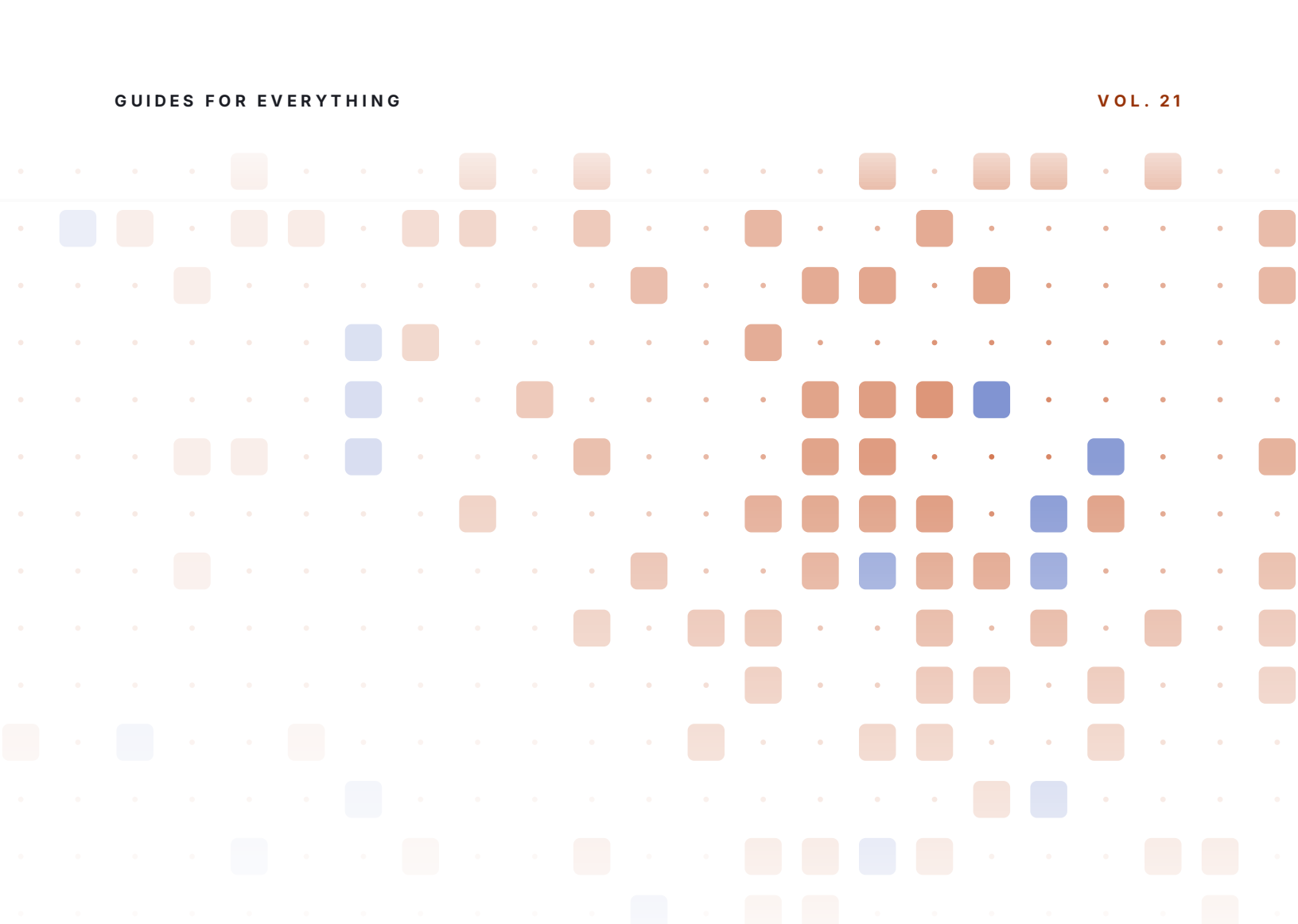




A BEGINNER'S MANUAL

GROK EDITION

AWS for Beginners

Sign up safely, protect your bill, learn the core services,
and build your first website, function and AI app on
Amazon Web Services

AWS for Beginners

Sign up safely, protect your bill, learn the core services, and build your first website, function and AI app on Amazon Web Services

GROK EDITION Written by Grok Bot, with facts checked against official sources in October 2026.

Amazon Web Services runs a huge share of the internet. It's also famous for confusing beginners and, occasionally, surprising them with a bill. Neither has to happen to you.

This manual starts where most guides skip: **choosing how to sign up** (AWS launched a new, simpler sign-up experience in 2026), **locking down your account**, and **setting spending protection before you build anything**. Then it explains the core services in plain English (EC2, S3, Lambda, RDS, DynamoDB, VPC, IAM and CloudWatch) and walks you through small, cheap projects: a static website, a serverless function and a first generative AI app with Amazon Bedrock.

Every fact was checked against AWS's own documentation and pricing pages, and every chapter shows you how to clean up afterward.

HOW TO READ THIS MANUAL

Tip — a shortcut or good habit.

Warning — easy to get wrong.

\$ command — text you type, exactly as shown.

Note — useful background.

Danger — can lose data, money, or access.

Settings > Privacy — a menu path to click.

New words appear in **bold** the first time and are collected in the Glossary at the back.

AWS features, Free Tier rules and prices were checked on official AWS documentation, pricing and FAQ pages on October 17, 2026. AWS's new sign-up experience was still rolling out to a limited number of customers at that time, and some AWS pages gave different details (for example, on when Free Tier credits expire); these differences are noted in the text. Prices vary by Region and change over time; always confirm on the official pricing page and the AWS Pricing Calculator. You are responsible for every resource you create, including resources created by AI tools acting for you. Delete what you don't need.

Contents

01 What AWS Is	5
1.1 What you can do with it	5
1.2 The key ideas in one page	6
1.3 The shared responsibility model	6
1.4 How AWS bills you	6
02 Two Ways to Sign Up	7
2.1 Comparing the two paths	8
2.2 The new experience, step by step	8
2.3 The advanced (classic) path, step by step	9
03 The AWS Free Tier	10
3.1 Credits	10
3.2 Free plan or Paid plan?	11
3.3 When do credits expire?	11
3.4 Always Free examples	11
04 Lock Down Your Account First	13
4.1 If you used the new experience	13
4.2 If you used the advanced (classic) path	13
05 Protect Your Bill Before You Build	15
5.1 Option 1: Spend limits (new experience, Paid plan)	15
5.2 Option 2: AWS Budgets (any account)	16
5.3 Option 3: A CloudWatch billing alarm (classic accounts)	16
06 Regions and Availability Zones	17
6.1 Regions in the new experience	18
07 The Console, CloudShell and the CLI	19
7.1 CloudShell	19
7.2 Installing the AWS CLI on your computer	19
7.3 Signing in from the CLI	20
08 IAM: Who Can Do What	21
8.1 The pieces	21
8.2 Reading a policy	22
8.3 Least privilege in practice	22
09 The Core Services, in Plain English	24
9.1 Amazon EC2: virtual computers	24

9.2	Amazon S3: storage for files	25
9.3	AWS Lambda: code without servers	25
9.4	Amazon RDS: managed relational databases	25
9.5	Amazon DynamoDB: fast key-value database	26
9.6	Amazon VPC: your private network	26
9.7	Amazon CloudWatch: logs, metrics and alarms	26
9.8	Choosing between services	26
10	Project 1: A Static Website on S3 and CloudFront	28
10.1	The plan	28
10.2	Step by step	28
10.3	Updating and cleaning up	29
11	Project 2: A Serverless Function with a URL	30
11.1	Create the function	30
11.2	Add a function URL	31
12	Project 3: Host a Web App with Amplify	32
12.1	Steps	32
12.2	Which hosting option?	33
13	Project 4: Your First AI App with Amazon Bedrock	34
13.1	Try it in the console	34
13.2	Call a model from Python	35
14	AWS and AI Coding Agents	36
14.1	Connecting an agent (new experience)	36
14.2	Rules for working with agents on AWS	37
15	Cost Traps and How to Avoid Them	38
15.1	The monthly cleanup routine	39
15.2	Closing an account or project	39
16	Learning Path and Certifications	40
16.1	A four-week starter plan	40
16.2	Beginner certifications	40
17	Troubleshooting	42
	Quick Reference	43
	Glossary	44

CHAPTER 1

What AWS Is

- AWS rents you computing building blocks (servers, storage, databases, AI models) over the internet, billed by usage.
- You pay for what you use, often by the hour, the gigabyte or the request.
- That flexibility is powerful, and it's why cost protection comes first in this manual.

Amazon Web Services (AWS) is a **cloud provider**. Instead of buying a server and putting it in a closet, you rent computing resources from AWS's data centers and pay only while you use them.

1.1 What you can do with it

GOAL	AWS SERVICE YOU'D LIKELY USE
Host a simple website	Amazon S3 with Amazon CloudFront, or AWS Amplify Hosting
Run a small program when something happens	AWS Lambda
Rent a virtual computer	Amazon EC2
Store files, photos and backups	Amazon S3
Run a traditional database	Amazon RDS
Store app data that must scale	Amazon DynamoDB
Build an AI chatbot or summarizer	Amazon Bedrock
Watch logs and set alarms	Amazon CloudWatch

1.2 The key ideas in one page

IDEA	PLAIN ENGLISH
Pay-as-you-go	You're billed for usage, like electricity, rather than a fixed monthly fee
Region	A geographic area with AWS data centers, such as US East (Ohio)
Availability Zone	One or more separate data centers inside a Region
Service	A product, such as S3 or Lambda
Resource	One thing you create in a service, such as a bucket, a function or a server
Console	The website where you click to manage AWS
CLI	The command-line tool for managing AWS by typing commands
IAM	Identity and Access Management: who can do what

1.3 The shared responsibility model

AWS describes security as a shared job. AWS secures the physical data centers, hardware and core infrastructure. **You** are responsible for what you put on it: who has access, whether your data is public, and how your resources are configured.

☆ IMPORTANT

AWS's documentation makes the same point about AI coding tools: when a tool or agent creates resources for you, "you are still responsible for them," including security and charges.

1.4 How AWS bills you

Most services charge by some combination of:

- **Time:** a server or database running per hour or second
- **Storage:** gigabytes stored per month
- **Requests:** number of calls, such as one million Lambda requests
- **Data transfer:** data sent out to the internet
- **Tokens:** for AI models, the text you send and receive

The good news: many small learning projects cost very little, or nothing within the Free Tier. The bad news: a resource you forget about keeps billing. That's why this manual teaches you to **set limits first and clean up always**.

CHAPTER 2

Two Ways to Sign Up

- In 2026 AWS began rolling out a new, simpler sign-up experience built around “projects,” with sign-in through Google, GitHub, Apple or Amazon.
- The classic path, now called “Sign up for AWS (advanced),” gives full control and uses a root user.
- Both include the AWS Free Tier for new customers.

On September 16, 2026, AWS announced a new sign-up experience designed to help builders start faster. Its documentation says it was being released to a limited number of customers at first, so you may see either path.

2.1 Comparing the two paths

FEATURE	SIGN UP FOR AWS (NEW)	SIGN UP FOR AWS (ADVANCED)
Sign-in	Google, GitHub, Apple, Amazon or email, through an AWS Builder ID	Root user email and password
Credit card	Not required for most new customers (a \$1 hold may be used to verify)	Payment method required during sign-up
Organization	Resources grouped in projects , managed in AWS Settings (settings.aws.com)	Standard AWS account
Root user	None; your Builder ID owns the project accounts	Yes, with full access
Region	Assigned from your contact country: US East (Ohio), Europe (Stockholm) or Asia Pacific (Sydney)	You choose any Region
Spend limits	Yes, per project, on the Paid plan	Not available (use AWS Budgets instead)
Permissions	Preconfigured; you and invited team members get administrator access to projects	You design IAM users, roles and policies
Not supported	HIPAA or FedRAMP workloads, AWS Marketplace, Savings Plans, opt-in Regions, choosing a Region	All features available

AWS's guidance is clear: use the **new** experience if you're new and want sensible defaults; use **advanced** if you need complete control, regulated workloads such as HIPAA or FedRAMP, or a specific Region.

NOTE

With the new experience you can later **activate advanced features** (multiple Regions, custom governance and more) without migration or downtime. AWS warns that activation can't be reversed, and you must be on the Paid plan first.

2.2 The new experience, step by step

- 1 Open the AWS sign-up page and enter your email, or choose Google, Apple, GitHub or Amazon.
- 2 Enter your name (shared with team members you invite) and verify your email with the code sent to you.
- 3 Create a strong password: 8 to 128 characters, with at least three of uppercase, lowercase, numbers and symbols.

- 4 Enter your contact details. AWS provisions your first project in the Region for your country.
- 5 In the console, choose **Manage projects** to open AWS Settings, where you can invite team members, create projects and manage billing.

2.3 The advanced (classic) path, step by step

- 1 Open the Sign up for AWS page, enter a root user email address and account name, and verify the email.
- 2 Create a strong root password.
- 3 Choose your account plan (Free or Paid; see the next chapter).
- 4 Enter contact information and accept the AWS Customer Agreement.
- 5 Enter billing information and confirm your identity by phone.
- 6 Choose a Support plan (Basic is free).
- 7 Wait for the activation email (usually minutes, sometimes up to 24 hours), then sign in.

TIP

For a business account, AWS recommends a shared, secured company email list (not one employee's address) and a company phone number, so the business keeps access when people change roles.

The AWS Free Tier

- New customers get USD \$100 in credits at sign-up and can earn up to \$100 more by completing activities.
- More than 30 services have **Always Free** monthly allowances, available to every customer.
- The **Free plan** can't be charged, but it ends after six months or when credits run out. The **Paid plan** bills you beyond your credits.

The Free Tier changed significantly in 2025. All details here come from AWS's Free Tier FAQs and Billing documentation, checked October 17, 2026.

3.1 Credits

- **\$100** in credits when you create an account
- **Up to \$100 more** for completing activities listed in the **Explore AWS** widget on Console Home (set the filter to "Earn AWS credits"), within six months of opening the account
- Credits apply across participating services and Regions, and can't be transferred
- Only new customers qualify; if you have or previously had an AWS account, you're not eligible

3.2 Free plan or Paid plan?

	FREE PLAN	PAID PLAN
Charges	None while on the plan	Pay-as-you-go beyond your credits
Length	Up to six months, or until credits are used up, whichever comes first	Ongoing
Services	Most, but not those that could quickly use all your credits (for example Savings Plans, Reserved Instances, some Marketplace offers)	All
Free offers	Always Free allowances	Always Free plus short-term trials
When it ends	Account is suspended; data kept for 90 days, during which you can upgrade to recover it	Not applicable
Downgrade back	Not applicable	Not possible

⚠ WARNING

When the Free plan ends, AWS says you'll lose access to your resources and data, and that AWS permanently deletes the account and its content if you don't upgrade within 90 days. Download anything you want to keep before the six months are up.

3.3 When do credits expire?

AWS's pages were not consistent on this when checked. The Free Tier FAQ said in one answer that credits "expire 6 months from the date you create your AWS account," and in others that they expire 12 months after you create the account (after upgrading, and for activity credits).

Check your own expiration date: in the new experience, look in AWS Settings, Billing; on the classic path, use the Credits page in the Billing and Cost Management console.

3.4 Always Free examples

SERVICE	ALWAYS FREE ALLOWANCE (PER MONTH)
AWS Lambda	1 million requests and 400,000 GB-seconds of compute
Amazon DynamoDB	25 GB of storage, 25 write and 25 read capacity units (provisioned, Standard table class), per Region
Data transfer out to the internet	100 GB, aggregated across services and Regions (except China and GovCloud)

For the full, current list, open the AWS Free Tier page and filter by "Always Free."

Tracking Free Tier usage

- **Console Home, Cost and Usage widget:** credit balance and days remaining on the Free plan
- **Billing and Cost Management, Free Tier page:** usage by service against free limits this month
- **Email alerts:** AWS sends periodic emails about your credit balance and the end of your Free plan

Lock Down Your Account First

- Turn on multi-factor authentication (MFA) before you build anything.
- On the classic path, protect the **root user**, never create root access keys, and use a separate administrative identity for daily work.
- On the new experience, there's no root user; secure your Builder ID login and its provider's two-step verification.

An AWS account with weak security is a gift to criminals, who can run expensive resources on your bill. Ten minutes of setup prevents most problems.

4.1 If you used the new experience

AWS “strongly recommend[s]” MFA and supports **FIDO2 authenticators**: built-in ones like Touch ID or Windows Hello, and physical security keys. Key points from AWS’s documentation:

- ☐ Register **more than one** MFA device. If you lose access to all of them, AWS says you won’t be able to recover your account.
- ☐ If you signed up with Google, GitHub, Apple or Amazon, turn on that provider’s two-step verification.
- ☐ Review your registered devices periodically.
- ☐ Keep a backup security key in a physically secure place.

4.2 If you used the advanced (classic) path

When you create a classic account, you start with the **root user**, which has complete access to everything. AWS’s root user best practices boil down to:

- ☐ Use a strong, unique root password, stored in a password manager.
- ☐ Turn on MFA for the root user immediately.
- ☐ **Don't create access keys for the root user.**
- ☐ Don't use the root user for everyday tasks; sign in as root only for the few tasks that require it.
- ☐ Create an **administrative user** for daily work. AWS recommends IAM Identity Center for this.
- ☐ Prefer temporary credentials (roles and sign-in sessions) over long-term access keys.
- ☐ Protect the root email account itself, because it can be used to reset the root password.

Turning on root MFA (classic)

- 1 Sign in as the root user.
- 2 Open the account menu (your account name, top right) and choose **Security credentials**.
- 3 Under **Multi-factor authentication (MFA)**, choose **Assign MFA device**.
- 4 Choose a passkey or security key, or an authenticator app, and follow the prompts.
- 5 Sign out and sign back in to confirm it works.

DANGER

Never paste AWS access keys into chats, code repositories, screenshots or forums. Automated bots scan public code sites for leaked keys within minutes. If a key leaks, deactivate it in IAM immediately and check your bill.

Everyday security habits

- Sign in with your administrative identity, not root.
- Give apps and tools only the permissions they need (more in the IAM chapter).
- Keep S3 buckets private unless you deliberately host public files.
- Watch your email for AWS security and billing notices.

Protect Your Bill Before You Build

- On the new experience's Paid plan, set a **spend limit** on each project.
- On any account, create an **AWS Budget** with email alerts; basic budgets are free.
- Know where to see your costs: Billing and Cost Management, Cost Explorer and the Free Tier page.

5.1 Option 1: Spend limits (new experience, Paid plan)

A spend limit sets “the most you’ll pay per month for a project.” Facts from AWS’s documentation:

DETAIL	WHAT AWS SAYS
Who can use them	Projects on the Paid plan, in the new experience
Minimum	The greater of \$20 or a conservative estimate of your likely spend
What counts	Pre-tax charges; credits are excluded
Alerts	At 50%, 75% and 90% of the limit, and when you’re on track to exceed it within 10 days
At the limit	AWS pauses the project , stopping its resources, until the next month unless you raise the limit
Optional early controls	Stop new resource creation about 7 days before the limit; pause idle EC2, RDS and SageMaker endpoint resources about 5 days before

NOTE

AWS says spend limits are “designed for experimentation, learning, and sandbox workloads.” Pausing can interrupt an app, so production users should plan for it.

5.2 Option 2: AWS Budgets (any account)

AWS Budgets sends alerts when costs exceed, or are forecast to exceed, an amount you set. AWS's pricing page says monitoring and notifications are **free of charge**; the first two budgets with automatic actions are free each month, with later action-enabled budgets costing \$0.10 a day.

- 1 Open **Billing and Cost Management** and choose **Budgets**.
- 2 Choose **Create budget**, then a template such as **Monthly cost budget** (or **Zero spend budget**, which alerts as soon as you spend anything beyond free usage).
- 3 Enter an amount, such as \$10.
- 4 Enter the email addresses for alerts.
- 5 Create the budget.

5.3 Option 3: A CloudWatch billing alarm (classic accounts)

You can also alarm on estimated charges with CloudWatch. AWS's documentation notes two requirements:

- First enable **Receive CloudWatch Billing Alerts** in Billing Preferences.
- Billing data is stored in **US East (N. Virginia)**, so switch the console to that Region before creating the alarm. The data represents worldwide charges.

Where to look at costs

TOOL	USE IT TO...
Billing and Cost Management home	See month-to-date charges and forecasts
Bills	See charges by service and Region
Cost Explorer	Chart costs over time and filter by service
Free Tier page	See usage against free allowances
Credits page (classic) or AWS Settings Billing (new)	See remaining credits and expiration dates

TIP

Make a habit: every Sunday, glance at month-to-date costs. Thirty seconds a week catches forgotten resources before they become expensive.

Regions and Availability Zones

- A **Region** is a geographic area; most resources live in one Region.
- The console shows only the selected Region's resources, which is a classic source of "where did my server go?"
- New-experience projects are fixed to one assigned Region.

AWS runs data centers around the world, grouped into Regions such as US East (N. Virginia) `us-east-1`, US East (Ohio) `us-east-2` and Europe (Stockholm) `eu-north-1`. Each Region contains multiple **Availability Zones**, which are separate data centers that let you build apps that survive one location failing.

Why Regions matter to beginners

ISSUE	WHAT TO KNOW
Finding resources	The console shows one Region at a time. If something "disappeared," check the Region selector at the top right.
Prices	Prices differ by Region.
Service availability	Not every service or AI model is in every Region.
Latency	Choose a Region near your users.
Data location	Your data stays in the Region you choose unless you move it.
Global services	Some services, such as IAM and CloudFront, are global.

6.1 Regions in the new experience

AWS assigns each new-experience project to one Region based on the country in your contact address: **US East (Ohio)**, **Europe (Stockholm)** or **Asia Pacific (Sydney)**. You can't change it; to use other Regions you'd activate advanced features. AWS recommends confirming your project's Region before you create resources: open AWS Settings, choose **Project**, then **Additional Info**.

The Console, CloudShell and the CLI

- The **console** is the point-and-click website, the best place to start.
- **CloudShell** is a browser-based terminal with the AWS CLI already installed.
- The **AWS CLI** lets you manage AWS from your own computer; `aws login` signs you in with short-lived credentials.

Finding your way around the console

- **Search bar** (top): type a service name, such as “S3” or “Lambda.”
- **Region selector** (top right): the Region you’re working in.
- **Account menu** (top right): security credentials, billing and sign-out.
- **CloudShell icon** (top bar): opens a terminal in your browser.
- **Favorites**: star the services you use most.

7.1 CloudShell

CloudShell gives you a Linux terminal in the browser, already signed in as you and with the AWS CLI installed. It’s ideal for trying commands without installing anything.

7.2 Installing the AWS CLI on your computer

AWS’s documentation says the `aws login` command requires AWS CLI version 2.32.0 or later. Install the latest version 2 from AWS’s “Installing the AWS CLI” page:

SYSTEM	HOW
Windows	Download and run the MSI installer from AWS
macOS	Download and run the PKG installer from AWS
Linux	Download the zip, unzip it, run the install script (commands below)

INSTALL AWS CLI V2 ON LINUX (X86_64)

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

7.3 Signing in from the CLI

SIGN IN AND CHECK WHO YOU ARE

```
aws login
aws sts get-caller-identity
```

According to AWS, `aws login` opens your browser to sign in, then **rotates credentials automatically every 15 minutes**, keeping the session valid for up to 12 hours. `aws sts get-caller-identity` prints the account and identity you’re using: a good first check whenever something seems wrong.

A few safe first commands

LOOK AROUND (READ-ONLY)

```
aws configure list
aws s3 ls
aws ec2 describe-regions --output table
aws lambda list-functions
```

TIP

Add `--region us-east-2` (or your Region) to a command if results seem empty. Many “nothing found” surprises are Region mix-ups.

CHAPTER 8

IAM: Who Can Do What

- **IAM** controls access: identities (users, roles) and the **policies** that grant them permissions.
- Follow **least privilege**: give only the permissions needed, and no more.
- Prefer **roles** with temporary credentials over long-term access keys.

IAM stands for Identity and Access Management. Every request to AWS is checked against IAM: *who* is asking, and *are they allowed* to do this?

8.1 The pieces

TERM	PLAIN ENGLISH
IAM user	A long-term identity with a password or access keys. AWS now recommends avoiding these where possible.
IAM role	An identity that people or services assume temporarily. No long-term password or keys.
Policy	A JSON document listing allowed or denied actions on resources
Permission	One allowed action, such as <code>s3:GetObject</code>
IAM Identity Center	AWS's recommended way to give people sign-in access across accounts
Principal	Whoever is making a request: a user, role or service

NOTE

In AWS's new sign-up experience, AWS preconfigures access for you: project members get administrator access, and AWS says IAM users and roles you create there are for programmatic access only, not console sign-in. Its **IAM role manager** also lets many resources in a project reach each other automatically.

8.2 Reading a policy

Policies look intimidating but follow one pattern: **Effect, Action, Resource**.

READ-ONLY ACCESS TO ONE S3 BUCKET

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject", "s3:ListBucket"],
      "Resource": [
        "arn:aws:s3::my-example-bucket",
        "arn:aws:s3::my-example-bucket/*"
      ]
    }
  ]
}
```

PART	MEANING
"Version": "2012-10-17"	The policy language version (always this date; it isn't when you wrote it)
"Effect": "Allow"	This statement grants permission
"Action"	What's allowed: list the bucket and read objects
"Resource"	Which things: this bucket, and every object in it (/*)

An **ARN** (Amazon Resource Name) is a unique ID for a resource, such as `arn:aws:s3::my-example-bucket`.

8.3 Least privilege in practice

- ☐ Start with AWS **managed policies** for learning, then narrow them as you understand your needs.
- ☐ Avoid `"Action": "*" and "Resource": "*"` together; that's full access.
- ☐ Give each app or function its own role with only what it needs.
- ☐ Remove permissions you no longer use.

Ask an AI to explain a policy

PROMPT

Explain this AWS IAM policy in plain English for a beginner. Say exactly what it allows and denies, on which resources, and point out anything risky or broader than necessary. Then suggest a tighter version if one is possible.

[paste policy JSON]

⚠ DANGER

Never paste real access keys or secrets into an AI chat. A policy document itself contains no secrets, but your account ID and resource names are details you may prefer to replace with placeholders.

The Core Services, in Plain English

- You can do a lot with eight services: EC2, S3, Lambda, RDS, DynamoDB, VPC, CloudWatch and IAM.
- Each has a common beginner trap; knowing it in advance saves money and frustration.

9.1 Amazon EC2: virtual computers

EC2 (Elastic Compute Cloud) rents you virtual servers called **instances**. You choose an **instance type** (size), an operating system image (an **AMI**), storage and network settings.

CONCEPT	PLAIN ENGLISH
Instance	A virtual computer
Instance type	Its size: CPUs and memory (for example, the “t” family is for small, burstable workloads)
AMI	The starting disk image, such as Amazon Linux or Ubuntu
EBS volume	The instance’s virtual hard drive
Security group	A firewall that controls which traffic reaches the instance
Key pair	The key you use to sign in over SSH

⚠ WARNING

Stopping an EC2 instance stops compute charges, but its EBS storage still bills. **Terminating** deletes it (and usually its main volume). Public IPv4 addresses are also billed by the hour; see the VPC pricing page.

9.2 Amazon S3: storage for files

S3 (Simple Storage Service) stores **objects** (files) in **buckets**. It's used for backups, images, data and static websites.

- Bucket names are globally unique across all of AWS.
- New buckets block public access by default; keep it that way unless you deliberately host public content.
- You pay for storage, requests and data transferred out.
- Storage classes trade price for retrieval speed; Standard is the default.

BASIC S3 COMMANDS

```
aws s3 mb s3://my-unique-bucket-name-2026
aws s3 cp notes.txt s3://my-unique-bucket-name-2026/
aws s3 ls s3://my-unique-bucket-name-2026/
aws s3 rm s3://my-unique-bucket-name-2026/notes.txt
aws s3 rb s3://my-unique-bucket-name-2026
```

9.3 AWS Lambda: code without servers

Lambda runs your code when something happens (an upload, a web request, a schedule) and charges for requests and running time. You don't manage servers.

- Always Free: 1 million requests and 400,000 GB-seconds a month, according to the Lambda pricing page.
- Memory can be set from 128 MB to 10,240 MB; more memory also means more CPU.
- Languages include Python, Node.js, Java and others.

9.4 Amazon RDS: managed relational databases

RDS runs databases such as PostgreSQL and MySQL for you, handling backups, patching and failover options.

- Good for traditional apps that need SQL.
- Charged by the hour while running, plus storage.
- **Trap:** a forgotten database keeps billing. Stop or delete test databases when done. (AWS notes stopped RDS instances restart automatically after seven days.)

9.5 Amazon DynamoDB: fast key-value database

DynamoDB is a serverless NoSQL database. You store items by key and pay for reads, writes and storage, or provision capacity.

- Always Free per Region: 25 GB of storage plus 25 read and 25 write capacity units (provisioned mode, Standard table class). AWS estimates that's enough for about 200 million requests a month, depending on item size.
- Great for small apps, prototypes and learning.

9.6 Amazon VPC: your private network

A VPC (Virtual Private Cloud) is your own isolated network in AWS, with **subnets**, **route tables** and **gateways**. Every account starts with a default VPC in each classic Region, which is fine for learning.

⚠ WARNING

NAT gateways are a famous beginner surprise. AWS's VPC pricing example for US East (Ohio) shows \$0.045 per hour while provisioned, plus \$0.045 per GB processed, and a partial hour is billed as a full hour. That's over \$30 a month for one idle NAT gateway. Delete NAT gateways you don't need.

9.7 Amazon CloudWatch: logs, metrics and alarms

CloudWatch collects **logs** (like Lambda output), **metrics** (like CPU use) and lets you create **alarms** that email you.

- Lambda writes logs to CloudWatch Logs automatically.
- Set log retention so old logs don't pile up forever.
- Use alarms for billing (see "Protect Your Bill") and for app health.

9.8 Choosing between services

IF YOU NEED...	START WITH...	BECAUSE...
A website with only HTML, CSS and JavaScript	S3 and CloudFront, or Amplify Hosting	No servers to manage
A small API or automation	Lambda	Pay per request; Always Free allowance
A full Linux server you control	EC2	Maximum flexibility
A SQL database	RDS	Managed backups and patching

A simple, scalable data store	DynamoDB	Serverless; Always Free allowance
An AI feature	Bedrock	Many models through one API

 TIP

You don't need to learn every service. Most beginner projects use three or four of these. Learn each one when a project needs it, and keep the trap for each in mind.

Project 1: A Static Website on S3 and CloudFront

- Store your website files in a private S3 bucket.
- Serve them worldwide through Amazon CloudFront, which adds HTTPS and caching.
- Cost for a small personal site is usually very low; delete it when you're done experimenting.

A **static** website is plain HTML, CSS, JavaScript and images, with no server code. It's the perfect first AWS project.

10.1 The plan

PIECE	JOB
S3 bucket	Holds your files (kept private)
CloudFront distribution	Delivers them quickly over HTTPS
Origin access control (OAC)	Lets only CloudFront read the private bucket

10.2 Step by step

- 1 Make a tiny site.**
Create `index.html` on your computer with a heading and a paragraph.
- 2 Create a bucket.**
In the S3 console, choose **Create bucket**, give it a unique name, and leave **Block all public access** turned on.
- 3 Upload** `index.html` to the bucket.

4 Create a CloudFront distribution.

In the CloudFront console, choose your S3 bucket as the origin and choose **origin access control** when asked how CloudFront should access the bucket. Set the **default root object** to `index.html`.

5 Update the bucket policy.

CloudFront offers a policy to copy; paste it into the bucket's **Permissions, Bucket policy** so CloudFront can read the files.

6 Wait for the distribution to deploy, then open its domain name (ending in `cloudfront.net`).**NOTE**

AWS's documentation for the new sign-up experience says some CloudFront features that create resources outside your project's Region aren't available, including AWS WAF with CloudFront. A basic distribution like this one is a common first project; if the console blocks an option, follow its guidance.

Ask AI to help with the site itself**PROMPT**

Write a single-file personal homepage (index.html) with embedded CSS. Sections: a short intro, three projects with one-line descriptions, and contact links. Clean, accessible, mobile-friendly, no external libraries. Use placeholder text I can replace.

10.3 Updating and cleaning up

- To update: upload the new file, then create a CloudFront **invalidation** for `/*` (or `/index.html`) so visitors see the change sooner.
- To clean up: disable and then delete the distribution, empty the bucket, and delete the bucket.

Project 2: A Serverless Function with a URL

- Write a tiny Python function in Lambda.
- Give it a **function URL** so you can call it from a browser.
- Watch its logs in CloudWatch.

11.1 Create the function

- 1 Open the **Lambda** console and choose **Create function**.
- 2 Choose **Author from scratch**, name it `hello-api`, and pick a recent Python runtime.
- 3 Choose **Create function**.
- 4 Replace the code with the example below, then choose **Deploy**.
- 5 Choose **Test**, create a test event (the default is fine), and run it.

LAMBDA_FUNCTION.PY

```
import json

def lambda_handler(event, context):
    params = event.get("queryStringParameters") or {}
    name = params.get("name", "world")
    return {
        "statusCode": 200,
        "headers": {"Content-Type": "application/json"},
        "body": json.dumps({"message": f"Hello, {name}!"})
    }
```

11.2 Add a function URL

- 1 In the function, open **Configuration**, then **Function URL**, and choose **Create function URL**.
- 2 For a quick, public test, choose auth type **NONE**. (For anything real, choose **AWS_IAM** so only signed requests work.)
- 3 Save, then open the URL in your browser, adding `?name=Sam` to the end.

⚠ DANGER

A function URL with auth type NONE is open to anyone on the internet who finds it. AWS's documentation reminds you to make sure the function has the correct permissions. Delete public test URLs when you're done.

See the logs

Open the **Monitor** tab and choose **View CloudWatch logs**. Each run writes a log stream. Add `print()` statements to your code to see values while debugging.

Costs

Lambda's Always Free allowance (1 million requests and 400,000 GB-seconds each month) covers far more than a test function like this would use.

Project 3: Host a Web App with Amplify

- **AWS Amplify Hosting** deploys websites and front-end apps straight from a Git repository.
- Every push to your branch can rebuild and redeploy automatically.
- Good for React, Vue, Next.js or plain static sites.

12.1 Steps

- 1 Put your site in a GitHub (or other supported Git provider) repository.
- 2 Open the **AWS Amplify** console and choose to deploy an app.
- 3 Connect your Git provider and choose the repository and branch.
- 4 Review the detected build settings (Amplify recognizes many frameworks) and deploy.
- 5 When it finishes, open the provided URL. Push a change to see it redeploy.

TIP

Amplify Hosting charges for build minutes, storage and data served. Check the Amplify pricing page and keep an eye on Billing while experimenting. Delete the app when you're done.

12.2 Which hosting option?

OPTION	BEST FOR	EFFORT
S3 and CloudFront	Learning how AWS pieces fit together	Medium
Amplify Hosting	Sites in Git that you update often	Low
EC2	Apps that need a full server you control	High

Project 4: Your First AI App with Amazon Bedrock

- **Amazon Bedrock** gives you access to many foundation models (from Amazon and other companies) through one AWS API.
- Try models in the console **playground** before writing code.
- You pay per use (typically per token); check Bedrock pricing for your model and Region.

13.1 Try it in the console

- 1 Open the **Amazon Bedrock** console in a supported Region.
- 2 Open the **playground** (chat or text) and choose a model.
- 3 Depending on the model, you may need to request or accept access first; follow the console's prompts.
- 4 Type a prompt and compare answers from two different models.

13.2 Call a model from Python

The Bedrock **Converse API** gives one consistent way to chat with different models. This example uses the AWS SDK for Python (boto3) and assumes you've signed in with `aws login`.

BEDROCK_HELLO.PY

```
import boto3

client = boto3.client("bedrock-runtime", region_name="us-east-2")

MODEL_ID = "PASTE-A-MODEL-ID-FROM-THE-BEDROCK-CONSOLE"

response = client.converse(
    modelId=MODEL_ID,
    messages=[
        {
            "role": "user",
            "content": [{"text": "Explain what AWS Lambda is in two sentences."}]
        },
    ],
    inferenceConfig={"maxTokens": 300},
)

print(response["output"]["message"]["content"][0]["text"])
print(response["usage"])
```

NOTE

Copy the exact model ID (or inference profile ID) from the Bedrock console for your Region; IDs change as models are released. The `usage` printout shows input and output tokens, which is what you're billed for.

Keeping AI costs small

- ☐ Set `maxTokens` so a runaway answer can't get long.
- ☐ Use smaller, cheaper models for simple tasks.
- ☐ Test in the playground before running code in loops.
- ☐ Keep your Budget or spend limit in place.

AWS and AI Coding Agents

- AWS now offers the **AWS MCP Server** and the **Agent Toolkit for AWS**, so AI coding tools can work with your account.
- In the new experience, AWS Settings can give a coding agent credentials for one project.
- You remain responsible for everything an agent creates and every charge it causes.

AWS's September 2026 announcement says that after sign-up “you can connect your coding agent to AWS with a single prompt that automatically installs the AWS CLI and the Agent Toolkit for AWS.” The AWS MCP Server lets agents search AWS documentation without authentication and, with your IAM credentials, run AWS API calls and follow curated skills.

14.1 Connecting an agent (new experience)

- 1 Open AWS Settings at settings.aws.com and choose the project.
- 2 Choose **Create and manage cloud infrastructure**, then **Use your AWS credentials with AI coding tools**.
- 3 Choose **Copy agent prompt** and paste it into your coding agent; choose a profile name when asked.
- 4 Renew access when needed with `aws login --profile your-profile-name`. AWS says access lasts 12 hours per login and can be renewed for 90 days before you must grant access again in the browser.

14.2 Rules for working with agents on AWS

- ☐ Keep a **spend limit** or **Budget** in place before letting an agent create resources.
- ☐ Use a separate project or account for experiments.
- ☐ Ask the agent to **list what it will create** before it creates it.
- ☐ Ask it to produce a **cleanup script** for everything it made.
- ☐ Review IAM permissions it requests; refuse “full access” shortcuts.
- ☐ Check the console afterward; AWS says to view an agent’s resources in each service’s console.

☆ IMPORTANT

AWS’s documentation states: “You are responsible for any actions that your AI coding tool takes on your behalf in your projects.” Volume 5 of this series explains MCP, and Volume 13 covers AI agents.

Cost Traps and How to Avoid Them

- Most surprise bills come from a short list of forgotten or misunderstood resources.
- Check every Region you've used, not just your favorite.
- Delete, don't just stop, what you no longer need.

TRAP	WHY IT COSTS MONEY	HOW TO AVOID IT
Forgotten EC2 instances	Billed while running	Terminate test instances; check every Region
Stopped instances' EBS volumes and snapshots	Storage bills even when the instance is stopped	Delete unneeded volumes and snapshots
Public IPv4 addresses and unattached Elastic IPs	Billed by the hour	Release addresses you don't use
NAT gateways	Hourly plus per-GB charge, even when idle	Delete after experiments; use free S3 gateway endpoints where possible
RDS databases left running	Billed per hour plus storage	Delete test databases; stopped ones restart after seven days
Load balancers	Hourly charge	Delete with the app they served
Big data transfers out	Charged per GB beyond free allowances	Use CloudFront and caching; watch downloads
Logs kept forever	CloudWatch Logs storage grows	Set retention periods
AI model calls in loops	Per-token charges add up	Set max tokens; test small; keep budgets
Leaked access keys	Attackers launch expensive resources	Never share keys; use <code>aws login</code> ; enable MFA

15.1 The monthly cleanup routine

- 1 Open **Billing and Cost Management** and look at charges by service.
- 2 For each service with charges, open its console and **check every Region** you've used.
- 3 Delete what you don't need: instances, volumes, snapshots, IP addresses, NAT gateways, databases, load balancers.
- 4 Empty and delete test S3 buckets.
- 5 Delete old Lambda function URLs and test functions.
- 6 Recheck the bill a day later.

TIP

Ask your AI assistant to keep a running “resources I created” list as you work through a tutorial, with the console location of each. Cleanup takes minutes when you have that list.

15.2 Closing an account or project

- **New experience:** close a project, or close your whole account, in AWS Settings.
- **Classic:** close the account from the account page while signed in as the root user.
- Before closing, download anything you want to keep, and delete resources so nothing lingers on a final bill.

Learning Path and Certifications

- Learn by building small projects, then cleaning them up.
- AWS Skill Builder offers official training, including free content.
- Two beginner-friendly certifications: **Cloud Practitioner** and **AI Practitioner**.

16.1 A four-week starter plan

WEEK	FOCUS	PROJECT
One	Account setup, security, budgets, Regions	Sign up, enable MFA, create a budget
Two	S3, CloudFront, IAM basics	Static website (Project 1)
Three	Lambda, CloudWatch, DynamoDB	Serverless function (Project 2); store data in DynamoDB
Four	Bedrock and cost review	First AI app (Project 4); full cleanup

16.2 Beginner certifications

CERTIFICATION	FORMAT (FROM AWS'S PAGES)	GOOD FOR
AWS Certified Cloud Practitioner	90 minutes, 65 questions (multiple choice or multiple response), \$100	Anyone who wants a broad understanding of AWS cloud concepts, services, security and billing
AWS Certified AI Practitioner	90 minutes, 65 questions, \$100	People who want to understand AI, machine learning and generative AI concepts and AWS services

Both can be taken at a Pearson VUE testing center or online with a proctor. AWS offers an Exam Prep Plan on **AWS Skill Builder**, an official practice question set and an official practice exam.

Study with AI

PROMPT

I'm studying for the AWS Certified Cloud Practitioner exam. Quiz me with 10 scenario-based multiple-choice questions on [topic, e.g. billing and pricing]. After each answer I give, tell me if I'm right, explain why in two sentences, and name the AWS service or concept to review. Keep a score.

NOTE

Always check practice answers against the official exam guide and AWS documentation. AI can be wrong about details, especially recent changes like the 2025 Free Tier and the 2026 sign-up experience.

Troubleshooting

PROBLEM	LIKELY CAUSE	WHAT TO DO
"My resources disappeared"	Wrong Region selected	Check the Region selector; in the new experience, check your project's Region in AWS Settings
<code>AccessDenied</code> errors	Missing IAM permissions	Run <code>aws sts get-caller-identity</code> ; check the role or user's policies
CLI says it can't find credentials	Not signed in, or the session expired	Run <code>aws login</code> (CLI 2.32.0 or later)
<code>aws login</code> not recognized	Old CLI version	Update to the latest AWS CLI version 2
S3 website shows "Access Denied"	Bucket policy or origin access not set	Add the CloudFront-provided bucket policy; check the default root object
CloudFront still shows the old page	Caching	Create an invalidation
Lambda function URL returns an error	Code error	Check the CloudWatch logs from the Monitor tab
Bedrock says the model isn't available	Region or model access	Choose a supported Region; complete any access steps in the console
Unexpected charges	Forgotten resources	Billing by service, then check each Region; follow the cleanup routine
Project paused	Spend limit reached	Raise the limit in AWS Settings, or wait for next month
Can't use a service in the new experience	Not supported there	Check AWS's supported-services list; consider activating advanced features

Quick Reference

- **Sign up:** new experience (projects, Builder ID) or advanced (root user, full control)
- **Free Tier:** \$100 credits plus up to \$100 more; Free plan up to six months; 30+ Always Free services
- **First hour:** MFA, budget or spend limit, check your Region
- **Classic root user:** strong password, MFA, no access keys, don't use daily
- **Costs:** Billing home, Bills, Cost Explorer, Free Tier page

- **CLI:** `aws login`, then `aws sts get-caller-identity`
- **Lambda Always Free:** 1M requests and 400,000 GB-seconds a month
- **DynamoDB Always Free:** 25 GB, 25 RCU, 25 WCU per Region
- **Cost traps:** EC2, EBS, IPv4, NAT gateways, RDS, load balancers, logs
- **Clean up:** every Region; delete, don't just stop
- **Agents:** spend limit first; you're responsible for what they create

Sources (checked October 17, 2026)

- Free Tier: [AWS Free Tier FAQs](#); [Explore AWS services with AWS Free Tier](#); [Choosing a plan](#)
- New experience: [What's New, September 16, 2026](#); [Compare sign-up options](#); [Sign up for AWS \(new\)](#); [AWS Regions for your projects](#); [Create a spend limit](#); [MFA in AWS Settings](#); [Connect an AI coding tool](#); [Activate advanced features](#)
- Classic sign-up and security: [Sign up for AWS \(advanced\)](#); [Root user best practices](#)
- Costs: [AWS Budgets pricing](#); [CloudWatch billing alarms](#); [VPC pricing](#); [Lambda pricing](#); [DynamoDB pricing](#)
- Certifications: [Cloud Practitioner](#); [AI Practitioner](#)

Glossary

- **AMI:** Amazon Machine Image; the starting disk image for an EC2 instance.

- **ARN:** Amazon Resource Name; a unique ID for an AWS resource.

- **Availability Zone:** one or more separate data centers within a Region.

- **AWS Builder ID:** a personal AWS profile; in the new experience it owns your project accounts.

- **AWS Settings:** the site (settings.aws.com) for managing projects, billing and team members in the new experience.

- **Bucket:** a container for objects in S3.

- **CloudShell:** a browser-based terminal with the AWS CLI preinstalled.

- **EBS:** Elastic Block Store; virtual disks for EC2.

- **Function URL:** a direct HTTPS address for a Lambda function.

- **IAM role:** an identity assumed temporarily, with no long-term credentials.

- **Instance:** a virtual server in EC2.

- **Least privilege:** granting only the permissions needed.

- **NAT gateway:** lets private resources reach the internet; billed hourly and per GB.

- **Policy:** a JSON document that allows or denies actions on resources.

- **Project:** in the new experience, an AWS account plus sharing settings and an optional spend limit.

- **Region:** a geographic area containing AWS data centers.

- **Root user:** the all-powerful identity of a classic AWS account.

- **Spend limit:** a monthly cap on a project's pre-tax costs; the project pauses when reached.

- **VPC:** Virtual Private Cloud; your isolated network in AWS.
