GrowthSchool

AWS

COMPLETE NOTES

LEARN - BUILD - OPERATE

Core cloud services every DevOps engineer must know

growthschool.cc | @growthschool.cc

Learning Roadmap

GrowthSchool

Learn AWS for DevOps in order. Every page opens with a plain-English definition, then explains how it works with a simple numbered architecture diagram, real commands, common mistakes, a lab and a revision snapshot. We host an ecommerce "Shop API" on AWS throughout. Free-tier resources are enough to practise most of this.

HOW TO USE THESE NOTES

Create a free-tier account, set a billing alarm FIRST, and never commit access keys.
Every page has a diagram - trace the numbered steps 1 -> 2 -> 3 to see how it works.
Animated versions of the key flows are in output/animations (open the .gif files).

1. Why the Cloud & AWS

GrowthSchool

IN SIMPLE TERMS
AWS (Amazon Web Services) is a cloud provider: you rent computing, storage, and networking over the internet and pay only for what you use, instead of buying your own servers.

* Rent, Do Not Buy

- Instead of buying servers, you rent them by the second and give them back when done.
- You scale up in minutes for a sale and scale down at night - paying only for what you use.

* Shared Responsibility

- AWS secures the cloud itself; you secure what you put in it (data, access, config).
- Most incidents are customer misconfiguration, like an open S3 bucket - not AWS failing.

AWS SECURES (of the cloud)	YOU SECURE (in the cloud)
physical data centres	your data + who can access it
hardware + network	IAM users, roles, policies
the virtualization layer	security groups + encryption
managed service internals	patching what you install

REAL-WORLD EXAMPLE

```
aws sts get-caller-identity   # who am I / which account?
aws ec2 describe-regions --query "Regions[].RegionName"
```

REVISION SNAPSHOT

ASK YOURSELF	Under shared responsibility, who secures the data and access?
DO THIS	Run get-caller-identity and note your account id and identity.

2. Global Infrastructure

GrowthSchool

IN SIMPLE TERMS
AWS runs data centres worldwide grouped into Regions; each Region has several isolated Availability Zones, and edge locations sit close to users.

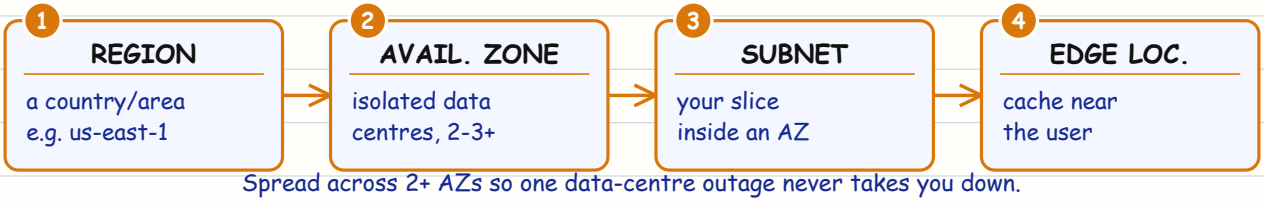
* Regions & AZs

- A Region is a location (like us-east-1); an Availability Zone is an isolated data centre in it.
- Running across 2+ AZs means one data-centre failure does not take your whole app down.

* Pick a Region

- Choose by closeness to users, data-residency laws, which services are offered, and price.

Zoom in: Region -> AZ -> Subnet -> Edge



Spread across 2+ AZs so one data-centre outage never takes you down.

REAL-WORLD EXAMPLE

```
aws ec2 describe-availability-zones --region us-east-1
# design rule: always spread across >= 2 AZs
```

REVISION SNAPSHOT

ASK YOURSELF	Why deploy across two Availability Zones?
DO THIS	List the AZs in your Region and say why you would use two.

3. Accounts, Billing & the CLI

GrowthSchool

IN SIMPLE TERMS
This covers setting up your AWS account safely, watching your bill, and the tools (console, CLI) you use to control everything.

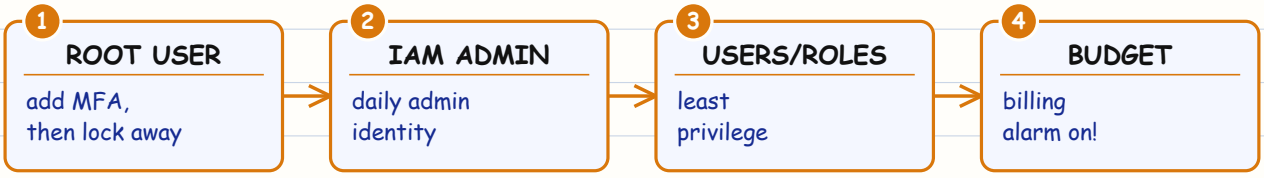
* Protect the Root User

- The root user can do anything - turn on MFA, then never use it for daily work.
- Make a separate IAM admin identity for yourself and lock the root credentials away.

* Control Cost From Day One

- Set a Budget and a billing alarm immediately so a mistake never becomes a shock bill.
- Tag resources (project, env, owner) so you can see and cut cost later.

Set up a new account safely (do this first)



REAL-WORLD EXAMPLE

```
aws configure          # key, secret, region, output
aws budgets describe-budgets --account-id <id>
# prefer SSO / roles over long-lived access keys
```

COMMON MISTAKE
Using the root user day to day. If those keys leak, the whole account is gone. MFA-protect root, make an admin user, and avoid long-lived keys.

REVISION SNAPSHOT

ASK YOURSELF	Why never use the root user for everyday tasks?
DO THIS	Configure the CLI, set a budget, and enable MFA on root.

4. IAM: Identities & Policies

GrowthSchool

IN SIMPLE TERMS
IAM (Identity and Access Management) controls who can do what in your account, using users, groups, roles, and permission policies.

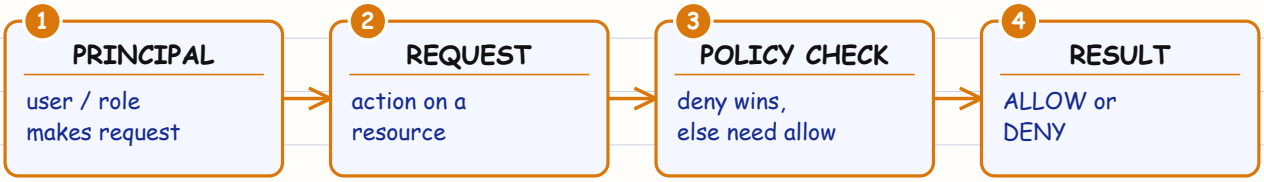
* Who Can Do What

- A user is a person/app, a group bundles users, and a role is temporary access that is assumed.
- A policy is a small JSON rule listing allowed (or denied) actions on which resources.

* How It Decides

- Everything is denied by default. You need an explicit Allow, and any explicit Deny always wins.

How an IAM permission decision is made



REAL-WORLD EXAMPLE

```
{ "Effect": "Allow",
  "Action": ["s3:GetObject"],
  "Resource": "arn:aws:s3:::shop-assets/*" }
```

COMMON MISTAKE
Attaching AdministratorAccess to make something work. Start with the few actions needed; broad permissions are the number-one cause of cloud breaches.

REVISION SNAPSHOT

ASK YOURSELF	If an Allow and a Deny both match, which wins?
DO THIS	Write a policy allowing read-only access to one S3 bucket.

5. IAM Best Practices

GrowthSchool

IN SIMPLE TERMS
IAM best practices are the safe habits - least privilege, roles instead of keys, and MFA - that keep your account from being hacked.

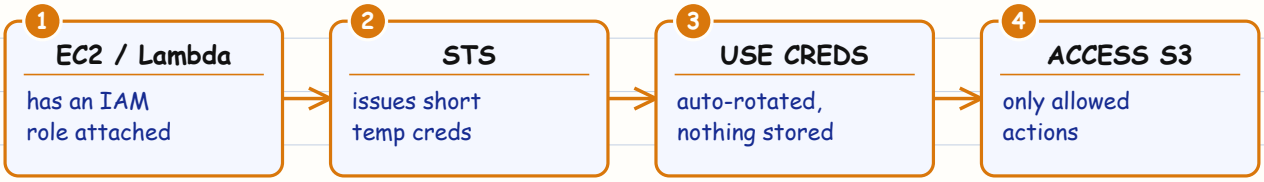
* Prefer Roles Over Keys

- Give EC2/Lambda a role instead of storing keys on the box - it gets short-lived auto-rotated creds.
- There is then nothing to leak, commit to Git, or rotate by hand.

* Least Privilege + MFA

- Grant only what is needed, require MFA for sensitive actions, and review unused permissions.

Roles give temporary keys - nothing to leak



REAL-WORLD EXAMPLE

```
aws iam create-role --role-name shop-api-role \
  --assume-role-policy-document file://trust.json
# app uses the role - no access keys anywhere
```

COMMON MISTAKE
Baking access keys into code or images. They leak via Git and logs. Use roles for compute and a secrets store for the rest.

REVISION SNAPSHOT

ASK YOURSELF	Why are roles safer than access keys for EC2 and Lambda?
DO THIS	Create a narrow role and attach it to a compute resource.

6. VPC Fundamentals

GrowthSchool

IN SIMPLE TERMS
A VPC (Virtual Private Cloud) is your own private network inside AWS where your servers and databases live.

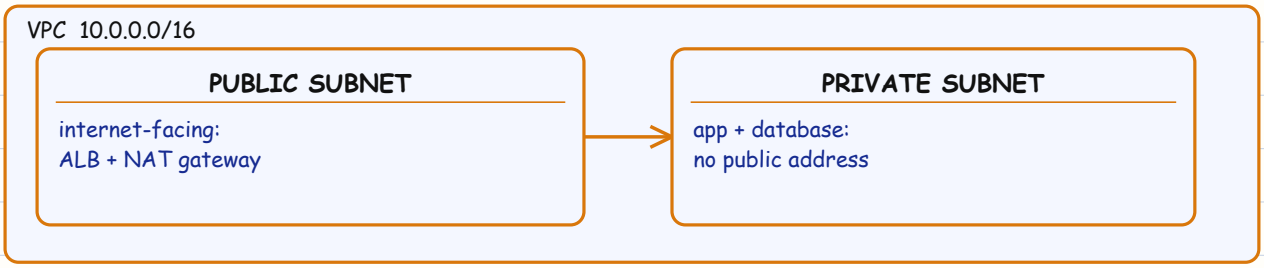
* Your Private Network

- A VPC is an isolated network you define with an address range like 10.0.0.0/16.
- Everything - servers, load balancers, databases - lives inside a VPC in one Region.

* Public vs Private

- Public subnets hold internet-facing bits (load balancer, NAT); private subnets hold app + DB.
- Keep databases in private subnets with no route straight to the internet.

A VPC: your private network, split into public + private subnets



REAL-WORLD EXAMPLE

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16
aws ec2 create-subnet --vpc-id vpc-123 --cidr-block 10.0.1.0/24
```

REVISION SNAPSHOT

ASK YOURSELF What makes a subnet "public" versus "private"?

DO THIS Sketch a VPC with one public and one private subnet across two AZs.

7. Subnets, Routing, IGW & NAT

GrowthSchool

IN SIMPLE TERMS
Subnets split your VPC into sections; route tables, an internet gateway, and a NAT gateway decide what can reach the internet and how.

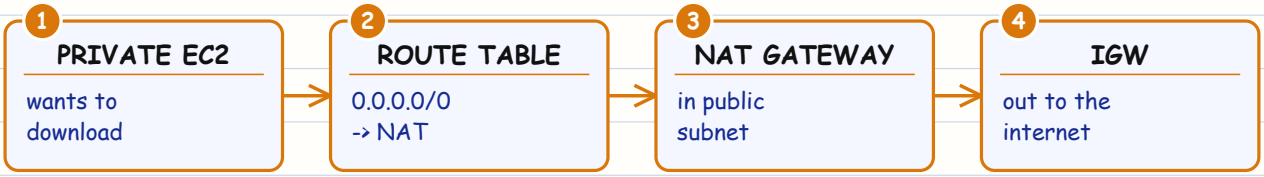
* Route Tables

- Each subnet follows a route table. Public subnets send internet traffic to an Internet Gateway.
- Private subnets send outbound traffic to a NAT gateway so they can reach out but not be reached.

* IGW vs NAT - the key difference

- IGW is TWO-WAY: a public-subnet EC2 with a public IP can be reached FROM the internet (inbound) and reach out.
- NAT is OUTBOUND-ONLY: a private-subnet EC2 can start connections out, but the internet cannot start one IN to it.

Outbound path from a private server (one-way)



INBOUND EXAMPLE: public vs private EC2
Public subnet EC2: a user opens http://<public-ip>; the request enters via the IGW and the web server answers - inbound works (if the security group allows port 80).
Private subnet EC2: that same request never arrives. NAT only passes REPLIES to connections the
instance itself started, so it drops unsolicited inbound. To expose it, put a load balancer in the public subnet and forward traffic to the private instance.

REAL-WORLD EXAMPLE

```
aws ec2 create-internet-gateway            # inbound + outbound
aws ec2 create-nat-gateway --subnet-id subnet-pub \
  --allocation-id eipalloc-1               # outbound only
```

REVISION SNAPSHOT

ASK YOURSELF	Can the internet start an inbound connection to a private-subnet EC2 behind a NAT?
DO THIS	Trace an inbound web request to a public EC2, then explain why it fails for a private one.

8. Security Groups vs NACLs

GrowthSchool

IN SIMPLE TERMS
Security Groups are per-server firewalls and NACLs are per-subnet firewalls; together they control which traffic is allowed.

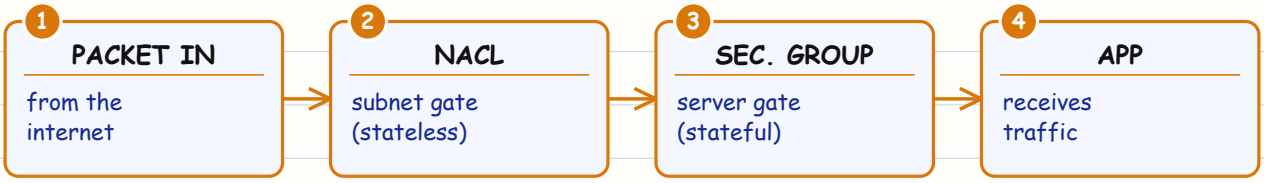
* Security Groups

- A firewall on the server. Stateful: allow traffic in and the reply is automatically allowed back.
- Allow rules only, and they can reference other groups (e.g. "allow from the load balancer").

* Network ACLs

- A firewall on the subnet. Stateless: you must allow both directions. Supports allow and deny.

A packet passes the subnet NACL, then the server SG



REAL-WORLD EXAMPLE

```
aws ec2 authorize-security-group-ingress --group-id sg-api \
  --protocol tcp --port 443 --source-group sg-alb
```

COMMON MISTAKE
Opening SSH (port 22) to 0.0.0.0/0. Bots attack it constantly. Restrict to your IP or use SSM Session Manager instead of open SSH.

REVISION SNAPSHOT

ASK YOURSELF	Why is a Security Group "stateful" but a NACL "stateless"?
DO THIS	Write an SG that only accepts traffic from the load balancer's SG.

9. EC2 Basics

GrowthSchool

IN SIMPLE TERMS
EC2 gives you virtual servers in the cloud that you can start, stop, and resize on demand.

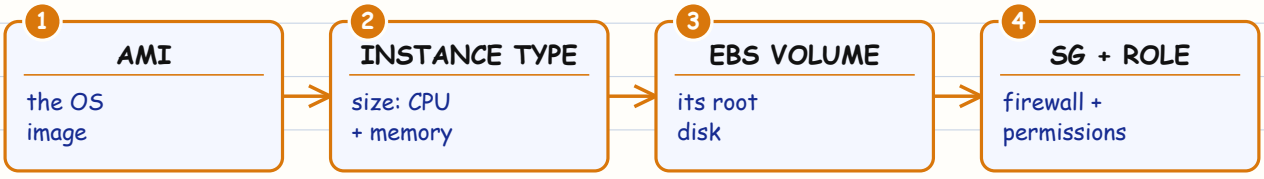
* Virtual Servers

- EC2 gives you virtual machines. You pick an image, a size, a subnet, a firewall and a role.
- It boots in your VPC and you can reach it, resize it, stop it, or throw it away.

* Stop vs Terminate

- Stop pauses compute billing and keeps the disk; terminate deletes the instance for good.

What makes up one EC2 instance



REAL-WORLD EXAMPLE

```
aws ec2 run-instances --image-id ami-123 --instance-type t3.micro \
  --subnet-id subnet-priv --security-group-ids sg-api --key-name shop
```

COMMON MISTAKE
Terminating when you meant to stop and losing the server. Stop pauses billing; terminate deletes - know which you want.

REVISION SNAPSHOT

ASK YOURSELF	What is the difference between stopping and terminating?
DO THIS	Launch a free-tier t3.micro, stop it, restart it, then terminate it.

10. EC2 Images, Types & User Data

GrowthSchool

IN SIMPLE TERMS
An AMI is the image an EC2 server boots from; instance types set its size, and user-data is a script that runs on first boot.

* AMI & Instance Type

- An AMI is the template a server boots from; bake "golden" AMIs for fast, consistent starts.
- Instance families fit jobs: t (burstable), m (general), c (compute), r (memory), g (GPU).

* User Data

- A first-boot script that installs and starts your app. Prefer baked images for anything complex.

EC2 first-boot sequence

REAL-WORLD EXAMPLE

```
#!/bin/bash   # passed as --user-data
yum install -y docker && systemctl enable --now docker
docker run -d -p 80:8080 <registry>/shop-api:1.0
```

REVISION SNAPSHOT

ASK YOURSELF	What does EC2 user-data do and when does it run?
DO THIS	Launch an instance with user-data that starts a web server.

11. EBS, EFS & Instance Store

GrowthSchool

IN SIMPLE TERMS
EBS is a virtual disk for one server, EFS is shared file storage for many, and instance store is fast but temporary scratch space.

* Three Storage Types

- EBS: a durable disk for one server, survives stop/start. EFS: shared files for many servers.
- Instance store: very fast but wiped when the instance stops - only for scratch data.

* Back It Up

- Snapshot EBS to S3 for backups. Snapshots are incremental, so they are cheap after the first.

Three storage types - pick by the job

REAL-WORLD EXAMPLE

```
aws ec2 create-volume --size 20 --availability-zone us-east-1a --volume-type gp3
aws ec2 create-snapshot --volume-id vol-1 --description "shop backup"
```

COMMON MISTAKE
Putting important data on instance store. It vanishes on stop or hardware failure. Use EBS or EFS for anything that must survive.

REVISION SNAPSHOT

ASK YOURSELF	When do you pick EBS vs EFS vs instance store?
DO THIS	Create a gp3 volume, attach it, and snapshot it.

12. Load Balancing (ELB)

GrowthSchool

IN SIMPLE TERMS
A load balancer spreads incoming traffic across many healthy servers so no single one is overwhelmed.

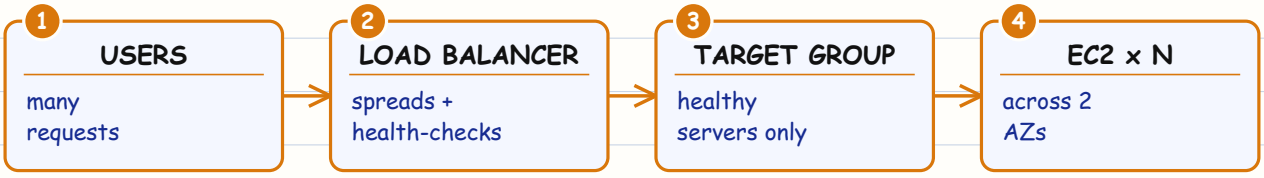
* Spread the Traffic

- A load balancer takes incoming requests and shares them across many healthy servers.
- ALB works at the web layer (routes by host/path); NLB works at the network layer (fast TCP).

* Health Checks

- It probes each target and stops sending traffic to unhealthy ones automatically.

A load balancer sends traffic only to healthy servers



REAL-WORLD EXAMPLE

```
aws elbv2 create-load-balancer --name shop-alb --type application \
  --subnets subnet-a subnet-b
# target group health check path: /health
```

COMMON MISTAKE
A wrong health-check path, so the balancer keeps sending users to broken servers. Point it at a real endpoint like /health.

REVISION SNAPSHOT

ASK YOURSELF	When would you choose an ALB versus an NLB?
DO THIS	Create an ALB with a target group whose health check hits /health.

13. Auto Scaling Groups

GrowthSchool

IN SIMPLE TERMS
An Auto Scaling Group automatically adds or removes servers to match demand and replaces any that become unhealthy.

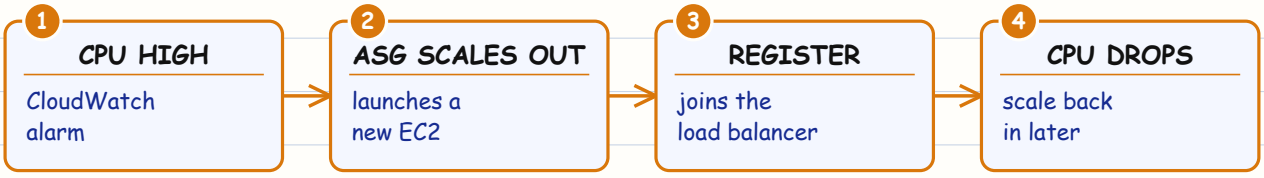
* Elastic Capacity

- An ASG keeps a target number of servers between a min and max, replacing unhealthy ones.
- It uses a launch template (image, size, firewall) and spreads servers across AZs.

* Scaling Policy

- Target tracking is simplest: "keep average CPU near 50%" and AWS adds/removes servers for you.

Auto Scaling reacts to load automatically



REAL-WORLD EXAMPLE

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name shop-asg \
  --launch-template LaunchTemplateName=shop-lt --min-size 2 --max-size 10 \
  --desired-capacity 2 --vpc-zone-identifier "subnet-a,subnet-b"
```

COMMON MISTAKE
Setting min equal to max, so it can never scale. Set a low min for normal load and a higher max for peaks, then let target tracking adjust.

REVISION SNAPSHOT

ASK YOURSELF	What three numbers define an ASG?
DO THIS	Create an ASG across two AZs with a CPU target-tracking policy.

14. S3 Object Storage

GrowthSchool

IN SIMPLE TERMS
S3 is object storage: it holds files (objects) in buckets with almost unlimited scale - great for assets, backups, and logs.

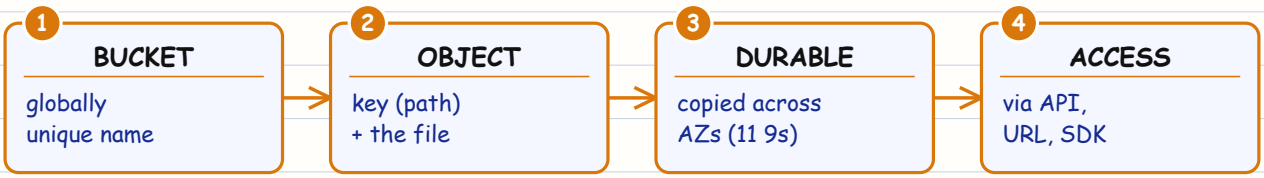
* Buckets & Objects

- S3 stores files (objects) in buckets. A bucket name is unique across all of AWS.
- Each object has a key (its path-like name), the data, and metadata. It is famously durable.

* Best For

- Static assets, backups, logs, and data lakes - not a live filesystem for random edits.

S3: put files (objects) into buckets



REAL-WORLD EXAMPLE

```
aws s3 mb s3://shop-assets-123
aws s3 cp ./logo.png s3://shop-assets-123/img/logo.png
aws s3 sync ./dist s3://shop-assets-123/site
```

COMMON MISTAKE
Trying to use S3 like a mounted disk for constant small writes. Upload/replace whole objects; for a real filesystem use EFS or EBS.

REVISION SNAPSHOT

ASK YOURSELF	Why must a bucket name be globally unique?
DO THIS	Create a bucket and sync a local folder into it.

15. S3 Classes & Lifecycle

GrowthSchool

IN SIMPLE TERMS
S3 storage classes offer cheaper tiers for less-used data; lifecycle rules move or delete objects automatically as they age.

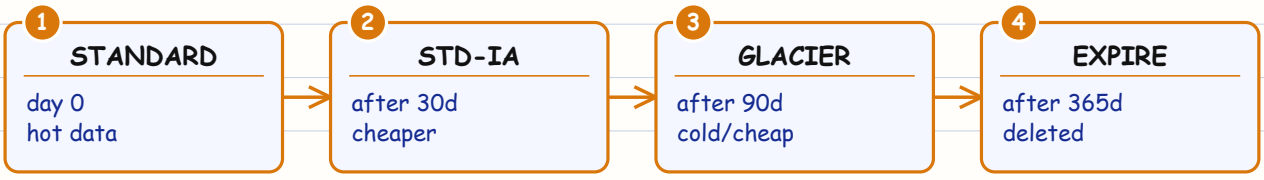
* Storage Classes

- Standard is for hot data; Standard-IA and One Zone-IA are cheaper for rarely used data.
- Glacier tiers are cheapest for cold archives; Intelligent-Tiering moves data for you.

* Lifecycle Rules

- Automatically move objects to cheaper tiers as they age, and delete them after a set time.

Lifecycle: data auto-moves to cheaper tiers over time



REAL-WORLD EXAMPLE

```
aws s3api put-bucket-lifecycle-configuration \
  --bucket shop-assets-123 --lifecycle-configuration file://lifecycle.json
```

COMMON MISTAKE
Keeping everything in Standard forever and overpaying for old logs and backups. Add lifecycle rules to tier or delete old data automatically.

REVISION SNAPSHOT

ASK YOURSELF	How do lifecycle rules cut S3 cost?
DO THIS	Add a rule that moves objects to Glacier after 90 days.

16. S3 Security

GrowthSchool

IN SIMPLE TERMS
S3 security is keeping buckets private by default, controlling access with policies, and encrypting the data.

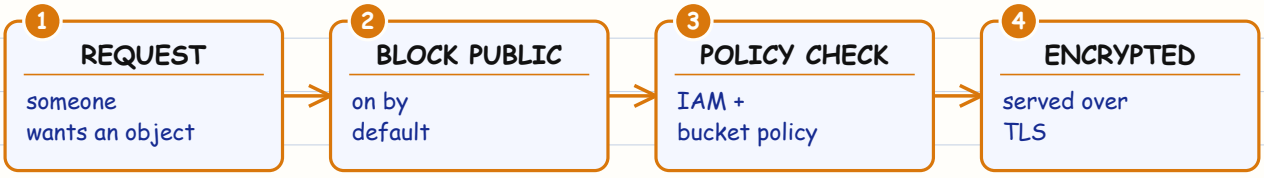
* Private By Default

- Keep Block Public Access ON unless you truly need public files. Default every bucket to private.
- Control access with IAM and bucket policies; share files with pre-signed URLs, not open buckets.

* Always Encrypt

- Turn on default encryption (SSE-S3 or SSE-KMS) and enforce HTTPS-only access.

Every S3 request is gated and encrypted



REAL-WORLD EXAMPLE

```
aws s3api put-public-access-block --bucket shop-assets-123 \
  --public-access-block-configuration BlockPublicAcls=true,\
IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
```

COMMON MISTAKE
Making a bucket public just to share one file. Public buckets are a classic leak. Use a pre-signed URL or CloudFront instead.

REVISION SNAPSHOT

ASK YOURSELF	What is the safest default for Block Public Access?
DO THIS	Enable Block Public Access and default encryption on a bucket.

17. RDS Managed Databases

GrowthSchool

IN SIMPLE TERMS
RDS is managed relational databases (like PostgreSQL or MySQL) where AWS handles backups, patching, and failover for you.

* Managed Relational

- RDS runs engines like PostgreSQL and MySQL for you: patching, backups, and failover are handled.
- Aurora is AWS's faster MySQL/PostgreSQL-compatible engine.

* High Availability

- Multi-AZ keeps a standby copy that takes over automatically; read replicas scale reads out.

RDS Multi-AZ: automatic failover keeps the DB available

REAL-WORLD EXAMPLE

```
aws rds create-db-instance --db-instance-identifier shop-db \
  --engine postgres --db-instance-class db.t3.micro \
  --allocated-storage 20 --multi-az --no-publicly-accessible
```

COMMON MISTAKE
Making the database public or hard-coding its password. Keep it private and put the password in Secrets Manager.

REVISION SNAPSHOT

ASK YOURSELF	What does Multi-AZ give you that a read replica does not?
DO THIS	Launch a private Multi-AZ Postgres instance.

18. DynamoDB (NoSQL)

GrowthSchool

IN SIMPLE TERMS
DynamoDB is a fully managed NoSQL database that is fast and scales automatically, with no servers to run.

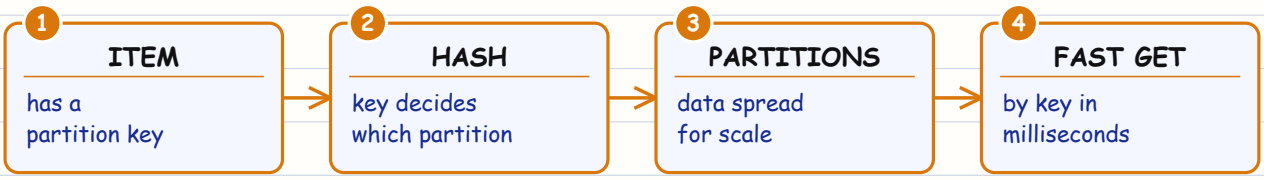
* Serverless Key-Value

- A fully managed NoSQL database with millisecond speed at any scale - no servers to run.
- You store items in tables and design around a partition key (plus an optional sort key).

* Capacity Modes

- On-demand auto-scales and bills per request; provisioned sets fixed capacity for steady load.

DynamoDB spreads items by their partition key



REAL-WORLD EXAMPLE

```
aws dynamodb create-table --table-name Carts \
  --attribute-definitions AttributeName=userId,AttributeType=S \
  --key-schema AttributeName=userId,KeyType=HASH --billing-mode PAY_PER_REQUEST
```

COMMON MISTAKE
Treating DynamoDB like SQL and scanning whole tables. That is slow and costly. Design your access patterns and query by the partition key.

REVISION SNAPSHOT

ASK YOURSELF	What is a partition key and why does it matter?
DO THIS	Create an on-demand table and get an item by its key.

19. Route 53 (DNS)

GrowthSchool

IN SIMPLE TERMS
Route 53 is AWS's DNS service: it turns your domain name into the address of your resources and can route users smartly.

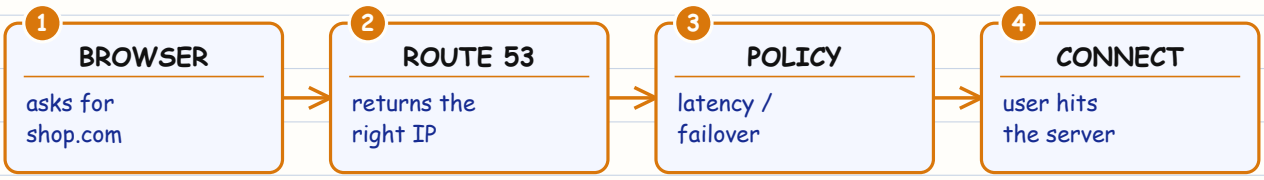
* Managed DNS

- Route 53 hosts your domain and answers "what is the address for shop.example.com?".
- An alias record points a name straight at AWS resources (ALB, CloudFront) for free.

* Smart Routing

- Routing policies can send users to the nearest, or fail over to a healthy backup automatically.

DNS: name in, best address out



REAL-WORLD EXAMPLE

```
aws route53 change-resource-record-sets --hosted-zone-id Z123 \
  --change-batch file://alias.json
dig +short shop.example.com
```

COMMON MISTAKE
Trying to use a CNAME at the domain apex (example.com). DNS forbids it - use a Route 53 alias record for the apex.

REVISION SNAPSHOT

ASK YOURSELF	Why prefer an alias record over a CNAME?
DO THIS	Create an alias record pointing your domain at an ALB.

20. CloudFront (CDN)

GrowthSchool

IN SIMPLE TERMS
CloudFront is a content delivery network (CDN) that caches your content at edge locations close to users for speed.

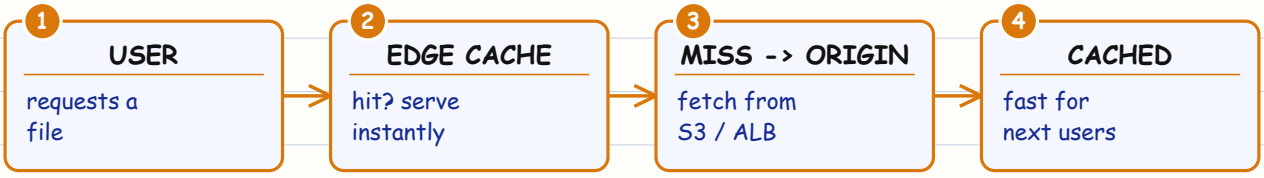
* Content Close to Users

- CloudFront caches your content at edge locations near users, so it loads fast and offloads origin.
- Origins can be S3 (static files) or an ALB/server (dynamic content).

* Secure Delivery

- Terminate HTTPS at the edge and use Origin Access Control so S3 is reachable only via CloudFront.

CloudFront serves from the nearest edge cache



REAL-WORLD EXAMPLE

```
aws cloudfront create-distribution --distribution-config file://dist.json
# after a deploy, refresh the cache:
aws cloudfront create-invalidation --distribution-id E123 --paths "/*"
```

COMMON MISTAKE
Forgetting to invalidate the cache after a release, so users see old files. Invalidate changed paths or use versioned filenames.

REVISION SNAPSHOT

ASK YOURSELF	What does CloudFront do on a cache hit versus a miss?
DO THIS	Serve an S3 site through CloudFront and invalidate after an update.

21. ECR Container Registry

GrowthSchool

IN SIMPLE TERMS
ECR is AWS's private registry for storing your Docker container images securely.

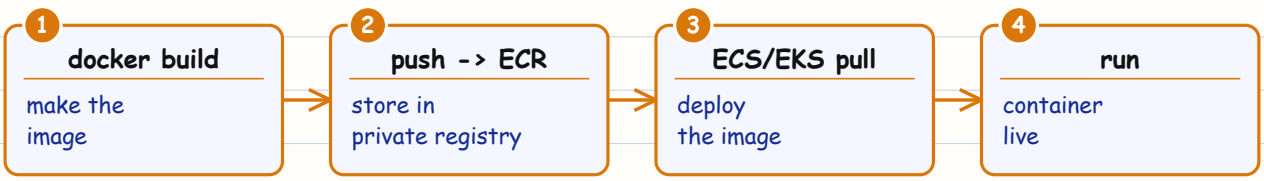
* Private Image Store

- ECR is AWS's private Docker registry, wired into IAM and ECS/EKS.
- You log in with a temporary token, then push and pull images like any registry.

* Keep It Tidy

- Turn on scan-on-push to catch vulnerabilities, and expire old images with a lifecycle policy.

ECR sits between build and deploy



REAL-WORLD EXAMPLE

```
aws ecr create-repository --repository-name shop-api
aws ecr get-login-password | docker login --username AWS \
  --password-stdin <acct>.dkr.ecr.us-east-1.amazonaws.com
```

COMMON MISTAKE
Never expiring old images, so the registry fills with junk layers and cost grows. Add a lifecycle policy to prune them.

REVISION SNAPSHOT

ASK YOURSELF	How do you authenticate Docker to ECR?
DO THIS	Create a repo, log in, and push a SHA-tagged image.

22. ECS & Fargate

GrowthSchool

IN SIMPLE TERMS
ECS runs your containers, and Fargate runs them without you managing any servers - you just say how much CPU and memory to use.

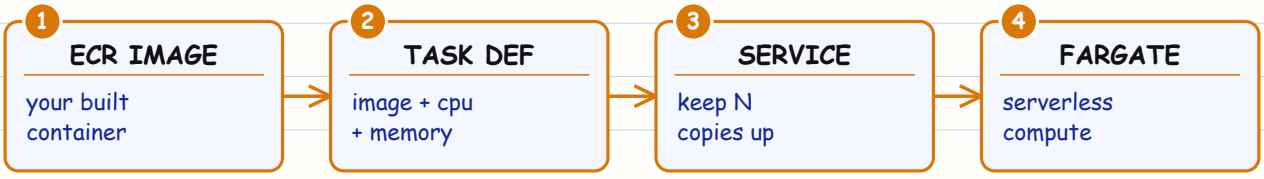
* Run Containers

- ECS runs your containers. A task definition says which image, how much CPU and memory, and ports.
- A service keeps a set number of tasks running behind a load balancer.

* Fargate

- Fargate runs the tasks with no servers to manage - you just pay for the CPU/memory each task uses.

ECS on Fargate: run containers, no servers to manage



REAL-WORLD EXAMPLE

```
aws ecs create-service --cluster shop --service-name shop-api \
  --task-definition shop-api --desired-count 3 --launch-type FARGATE
```

COMMON MISTAKE
Sizing task CPU/memory far above real usage and overpaying. Right-size from CloudWatch metrics.

REVISION SNAPSHOT

ASK YOURSELF	What is the difference between a task definition and a service?
DO THIS	Deploy the Shop API as a 3-task Fargate service.

23. EKS (Kubernetes)

GrowthSchool

IN SIMPLE TERMS
EKS is managed Kubernetes on AWS: AWS runs the Kubernetes control plane and you run your workloads on it.

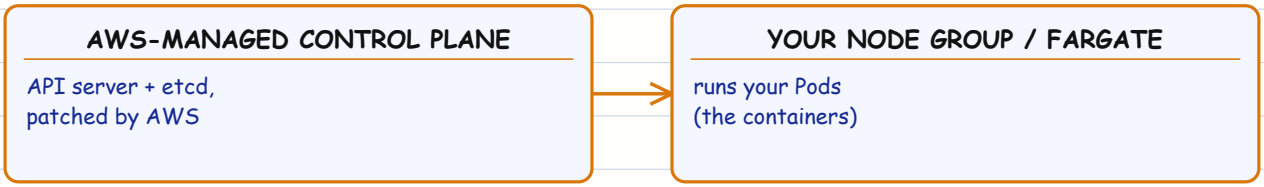
* Managed Kubernetes

- EKS runs the Kubernetes control plane for you; you run workloads on managed nodes or Fargate.
- Choose it when you want the Kubernetes ecosystem and portability rather than ECS's simplicity.

* Wire It Up

- Point kubectl at the cluster with the AWS CLI; use IAM Roles for Service Accounts for pod access.

EKS: AWS runs the control plane, you run the workloads



REAL-WORLD EXAMPLE

```
aws eks update-kubeconfig --name shop-cluster --region us-east-1
kubectl get nodes
kubectl apply -f shop-api.yaml
```

COMMON MISTAKE
Choosing EKS for a small app that ECS/Fargate would run more simply. Kubernetes adds overhead - pick it for a real reason.

REVISION SNAPSHOT

ASK YOURSELF	When would you choose EKS over ECS?
DO THIS	Point kubectl at an EKS cluster and list its nodes.

24. Lambda (Serverless)

GrowthSchool

IN SIMPLE TERMS
Lambda is serverless computing: you upload a function and AWS runs it on demand in response to events, billing per millisecond.

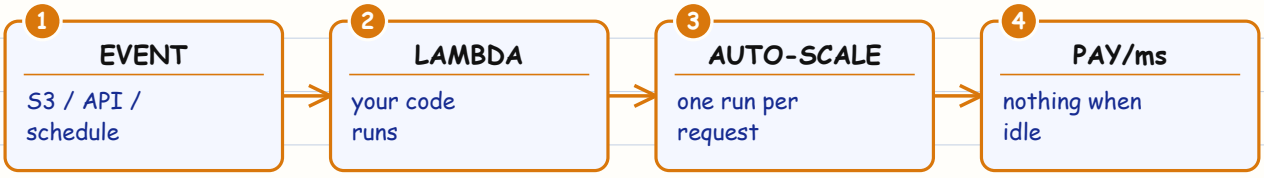
* Functions on Demand

- Upload code and AWS runs it in response to an event - no servers, billed per millisecond.
- It scales automatically per request and costs nothing while idle.

* Know the Limits

- Max 15-minute runtime and a cold-start delay on first call - great for event-driven work, not long jobs.

Serverless: an event triggers your function



REAL-WORLD EXAMPLE

```
aws lambda create-function --function-name resize-img \
  --runtime python3.12 --handler app.handler \
  --role arn:aws:iam::<acct>:role/lambda-role --zip-file fileb://fn.zip
```

COMMON MISTAKE
Using Lambda for an always-on or long job. The 15-minute limit and per-call model do not fit - use ECS/EC2 for sustained work.

REVISION SNAPSHOT

ASK YOURSELF	What work suits Lambda and what is the timeout?
DO THIS	Deploy a function triggered by an S3 upload.

25. API Gateway

GrowthSchool

IN SIMPLE TERMS
API Gateway is a managed front door for APIs that handles routing, security, and rate limiting to your backend.

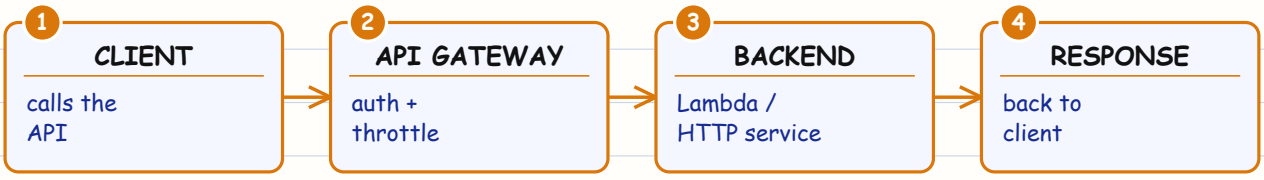
* Front Door for APIs

- API Gateway publishes your API and routes calls to Lambda or a backend service.
- It handles HTTPS, request checks, authorization, and rate limiting for you.

* Protect the Backend

- Add throttling and usage plans so one client cannot overwhelm your backend or your bill.

API Gateway is the guarded front door for APIs



REAL-WORLD EXAMPLE

```
aws apigatewayv2 create-api --name shop-api --protocol-type HTTP \
  --target arn:aws:lambda:us-east-1:<acct>:function:shop-fn
```

COMMON MISTAKE
Exposing a backend with no throttling or auth. One bad client can flood it and spike cost. Add rate limits and authorization.

REVISION SNAPSHOT

ASK YOURSELF	What does API Gateway handle so your backend does not?
DO THIS	Create an HTTP API that invokes a Lambda function.

26. SQS & SNS (Messaging)

GrowthSchool

IN SIMPLE TERMS
SQS is a message queue that buffers work between services; SNS is a publish/subscribe service that fans a message out to many receivers.

* SQS - Queues

- A queue holds messages so a fast producer and a slower consumer are decoupled and buffered.
- Standard is high-throughput; FIFO keeps strict order and exactly-once processing.

* SNS - Pub/Sub

- A topic pushes each message to many subscribers at once - queues, Lambdas, or email (fan-out).

SQS - QUEUE (pull)	SNS - TOPIC (push)
producer -> queue -> consumer	publisher -> topic -> many subs
buffers spikes of work	fan-out to queues, Lambda, email
one message, one worker	each subscriber gets a copy
standard or ordered FIFO	great for notifications

REAL-WORLD EXAMPLE

```
aws sqs create-queue --queue-name shop-orders
aws sns create-topic --name shop-events
aws sns subscribe --topic-arn <arn> --protocol sqs --notification-endpoint <queue-arn>
```

COMMON MISTAKE
Wiring services with direct synchronous calls everywhere, so one slow service stalls the rest. Use a queue to decouple and absorb spikes.

REVISION SNAPSHOT

ASK YOURSELF	What is the core difference between SQS and SNS?
DO THIS	Fan out one SNS message to two SQS queues.

27. CloudWatch (Observability)

GrowthSchool

IN SIMPLE TERMS
CloudWatch is AWS's monitoring service: it collects metrics and logs and raises alarms when something looks wrong.

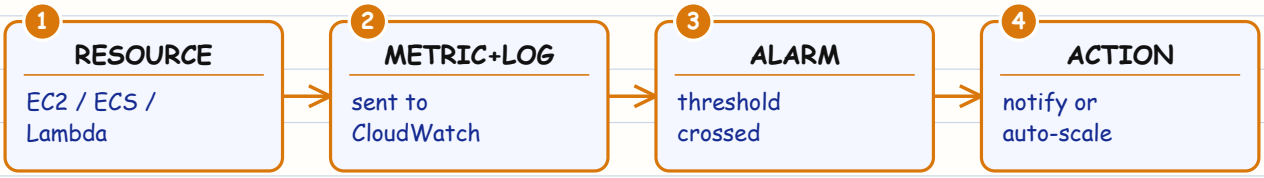
* Metrics & Logs

- CloudWatch collects numbers (CPU, latency) and log lines from AWS services and your apps.
- Dashboards show health at a glance; Logs Insights lets you query logs quickly.

* Alarms React

- An alarm watches a metric and, when a threshold is crossed, notifies you or triggers scaling.

CloudWatch watches, then reacts



REAL-WORLD EXAMPLE

```
aws logs tail /ecs/shop-api --follow
aws cloudwatch put-metric-alarm --alarm-name shop-cpu-high \
  --metric-name CPUUtilization --namespace AWS/ECS \
  --threshold 80 --comparison-operator GreaterThanThreshold \
  --evaluation-periods 5 --period 60
```

COMMON MISTAKE
Shipping with no alarms and hearing about outages from users. Alarm on the few signals that matter: errors, latency, and saturation.

REVISION SNAPSHOT

ASK YOURSELF	What are the three CloudWatch building blocks?
DO THIS	Tail an app log group and create a CPU-high alarm.

28. CloudTrail (Audit)

GrowthSchool

IN SIMPLE TERMS
CloudTrail records who did what in your AWS account, giving you an audit trail for security and troubleshooting.

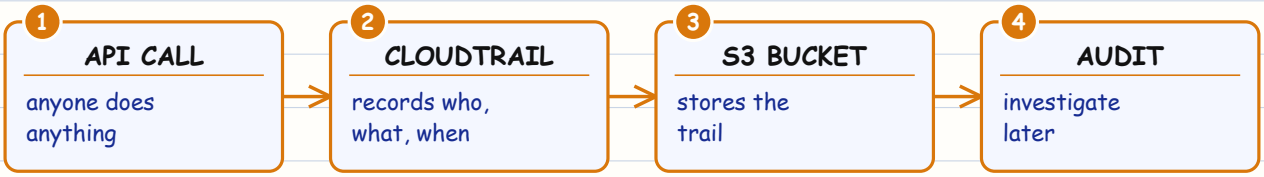
* Who Did What

- CloudTrail records every API action in your account: who, what, when, and from where.
- It is essential for security investigations, compliance, and tracking changes.

* Set It Up Right

- Enable a multi-Region trail that saves to a locked-down S3 bucket, and keep those logs immutable.

CloudTrail = the account's security camera



REAL-WORLD EXAMPLE

```
aws cloudtrail create-trail --name org-trail \
  --s3-bucket-name shop-cloudtrail-logs --is-multi-region-trail
aws cloudtrail lookup-events \
  --lookup-attributes AttributeKey=EventName,AttributeValue=RunInstances
```

COMMON MISTAKE
Turning CloudTrail on only after an incident, leaving no history to investigate. Enable it from day one and protect the log bucket.

REVISION SNAPSHOT

ASK YOURSELF	What question does CloudTrail answer that CloudWatch does not?
DO THIS	Enable a multi-Region trail and look up a recent API event.

29. SSM & Secrets Manager

GrowthSchool

IN SIMPLE TERMS
Systems Manager stores configuration and gives safe server access; Secrets Manager stores passwords and keys and can rotate them.

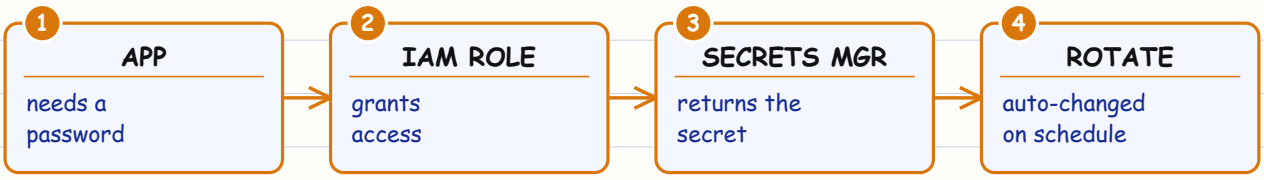
* Systems Manager

- Parameter Store keeps configuration (and secure strings); Session Manager gives shell access
- with no open SSH port and a full audit trail.

* Secrets Manager

- Stores passwords and keys, controls who can read them, and can rotate them automatically.

Apps fetch secrets at runtime - never bake them in



REAL-WORLD EXAMPLE

```
aws ssm put-parameter --name /shop/db-host --type String --value db.internal
aws ssm start-session --target i-123     # shell, no SSH port
aws secretsmanager get-secret-value --secret-id shop/db-password
```

COMMON MISTAKE
Passing secrets as plain env vars committed to a repo. Use Secrets Manager or secure parameters and fetch them at runtime with a role.

REVISION SNAPSHOT

ASK YOURSELF	When do you use Parameter Store vs Secrets Manager?
DO THIS	Store a config value and a secret, then read them via the CLI.

30. IaC: CloudFormation & Terraform

GrowthSchool

IN SIMPLE TERMS
This covers Infrastructure as Code on AWS - defining resources in CloudFormation (AWS-native) or Terraform instead of clicking.

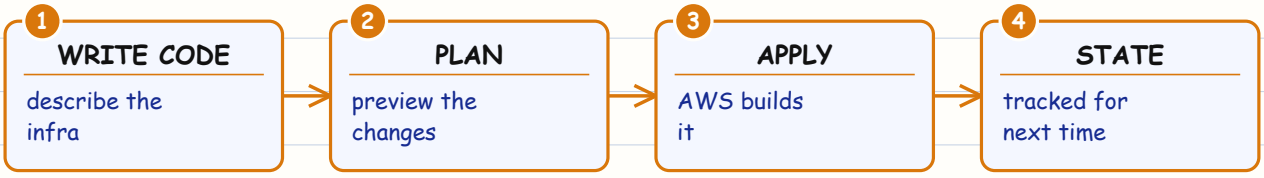
* Why IaC on AWS

- Describe infrastructure in files so environments are repeatable, reviewed, and version-controlled.
- CloudFormation is AWS-native; Terraform is multi-cloud (see the Terraform notes).

* Stacks & State

- Both track what they created (a stack or state file) and converge reality to your description.

Infrastructure as Code: describe, preview, apply



REAL-WORLD EXAMPLE

```
aws cloudformation deploy --template-file shop.yaml --stack-name shop-prod
aws cloudformation describe-stacks --stack-name shop-prod
# Terraform: terraform init && terraform apply
```

COMMON MISTAKE
Creating resources by hand next to IaC-managed ones. They drift and cause confusing changes. Manage each resource in code, in one place.

REVISION SNAPSHOT

ASK YOURSELF	What problem do CloudFormation and Terraform both solve?
DO THIS	Deploy a small stack and inspect its resources.

31. CI/CD on AWS

GrowthSchool

IN SIMPLE TERMS
CI/CD on AWS uses services like CodeBuild and CodePipeline to build, test, and deploy your app automatically.

* The Pipeline

- Source -> build+test (CodeBuild) -> push image (ECR) -> deploy (CodeDeploy/ECS), tied by CodePipeline.
- Each commit flows automatically toward production with checks along the way.

* Ship Safely

- Build one image tagged by commit and promote it everywhere; roll back automatically on failed health.

AWS CI/CD pipeline for the Shop API

REAL-WORLD EXAMPLE

```
# buildspec.yml (CodeBuild), simplified
phases:
  build:
    commands:
      - docker build -t $REPO:$CODEBUILD_RESOLVED_SOURCE_VERSION .
      - docker push $REPO:$CODEBUILD_RESOLVED_SOURCE_VERSION
```

REVISION SNAPSHOT

ASK YOURSELF	Why build one image and promote it, instead of rebuilding per environment?
DO THIS	Sketch a pipeline from commit to an ECS deployment.

32. Well-Architected Framework

GrowthSchool

IN SIMPLE TERMS
The Well-Architected Framework is AWS's checklist of six pillars for building secure, reliable, and cost-effective systems.

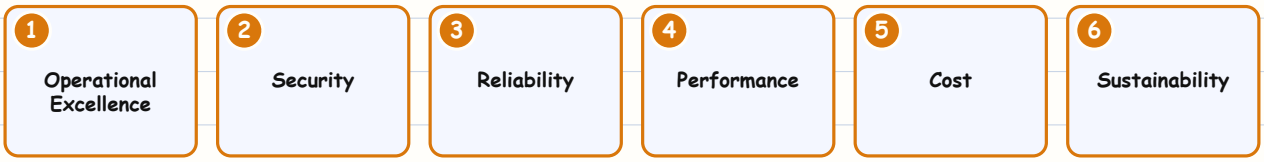
* Six Pillars

- Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization, Sustainability.
- They are a lens for reviewing any workload for gaps and trade-offs.

* Apply It Often

- Automate ops, secure least-privilege, design for failure, measure performance, and right-size for cost.

The 6 pillars of the Well-Architected Framework



REAL-WORLD EXAMPLE

```
# quick self-review
# Multi-AZ? least-privilege IAM? encrypted? alarms + backups?
# tested restore? right-sized and tagged for cost?
```

COMMON MISTAKE
Treating Well-Architected as a one-time checkbox. Workloads change, so review the pillars regularly.

REVISION SNAPSHOT

ASK YOURSELF	Name the six pillars.
DO THIS	Review the Shop API against each pillar and note one gap.

33. Cost Optimization

GrowthSchool

IN SIMPLE TERMS
Cost optimization is choosing the right pricing models and turning off waste so you pay as little as possible.

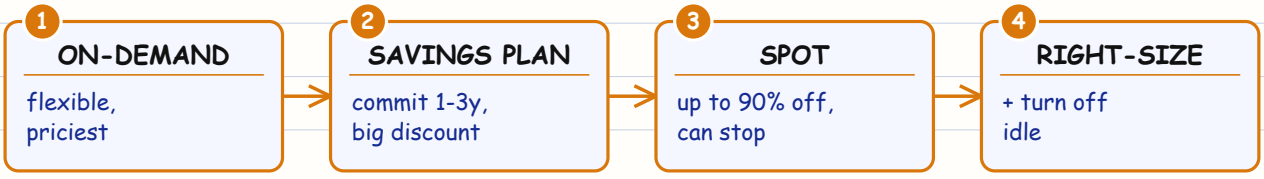
* Pricing Models

- On-Demand is flexible but priciest; Savings Plans/Reserved trade a commitment for big discounts;
- Spot is up to ~90% cheaper for work that can be interrupted.

* Cut Waste

- Right-size from metrics, switch off idle resources, use S3 lifecycle, and tag everything for visibility.

Pick the cheapest model that fits the workload



REAL-WORLD EXAMPLE

```
aws ce get-cost-and-usage --time-period Start=2025-01-01,End=2025-02-01 \
  --granularity MONTHLY --metrics UnblendedCost
```

COMMON MISTAKE
Leaving dev resources running 24/7. Schedule them off, right-size, and use Spot for interruptible work.

REVISION SNAPSHOT

ASK YOURSELF	When is Spot appropriate and when is it not?
DO THIS	Find your biggest service in Cost Explorer and one saving.

34. Shop API Reference Architecture

GrowthSchool

IN SIMPLE TERMS
This shows how all the services fit together into one complete, production-style architecture for the Shop API.

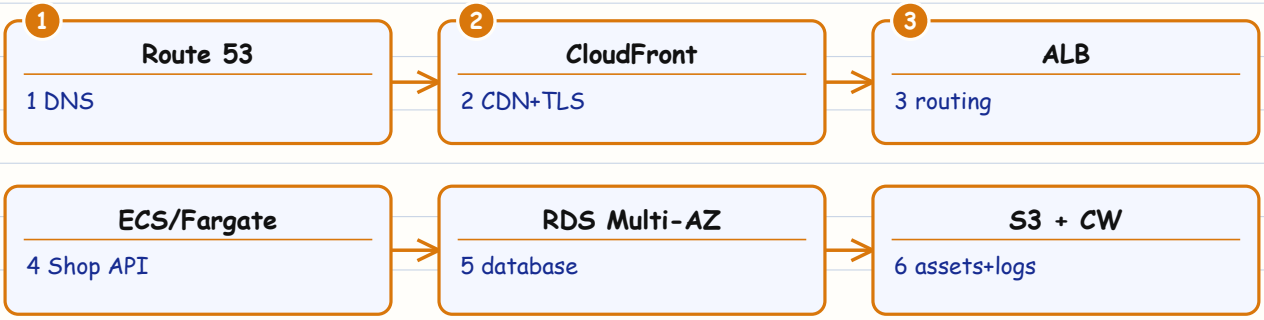
* End to End

- Route 53 -> CloudFront -> ALB -> ECS Fargate in private subnets across two AZs.
- RDS Postgres (Multi-AZ) for data, S3 for assets, Secrets Manager for the DB password.

* Operated Properly

- CloudWatch logs/alarms, CloudTrail audit, IAM roles (no keys), and CI/CD pushing tagged images.

Shop API on AWS - follow the request 1 -> 6



REAL-WORLD EXAMPLE

```
user -> Route53 -> CloudFront -> ALB -> Fargate -> RDS
assets: CloudFront -> S3 (OAC)   secrets: role -> Secrets Manager
```

REVISION SNAPSHOT

ASK YOURSELF	Trace a Shop API request from the user to the database.
DO THIS	Draw the full architecture and label each service's job.

35. Troubleshooting Playbook

GrowthSchool

IN SIMPLE TERMS
Troubleshooting is the calm, step-by-step way to fix common AWS problems like access denied or unreachable resources.

* Access & Network

- AccessDenied: a policy is missing an action or has an explicit deny - read the exact message.
- Cannot reach a server: check the SG, NACL, route table, and whether it is in a public subnet.

* Compute & Data

- No internet from a private server: missing NAT route. 5xx behind an ALB: unhealthy targets.
- RDS refused: security group, subnet routing, or wrong endpoint/credentials.

REAL-WORLD EXAMPLE

```
aws sts get-caller-identity
aws elbv2 describe-target-health --target-group-arn <arn>
aws logs tail /ecs/shop-api --follow
```

COMMON MISTAKE
Blaming "the network" before checking IAM and Security Groups - the usual culprits. Check permissions, SG rules, and routes first.

REVISION SNAPSHOT

ASK YOURSELF	First checks for AccessDenied vs a connectivity error?
DO THIS	Diagnose an unhealthy ALB target using target-health + logs.

36. AWS CLI Cheat Sheet

GrowthSchool

IN SIMPLE TERMS
This is a quick-reference list of the AWS CLI commands you will use every day.

* Daily Drivers

- sts, ec2, s3/s3api, iam, elbv2, ecs, rds, logs, cloudwatch, ssm, secretsmanager.
- Always confirm your --region and profile before deciding something "does not exist".

TRIAGE LOOP
get-caller-identity -> describe the resource -> check SG/IAM/route -> read logs.
This orients you on almost any AWS issue in under a minute.

ONE-PAGE COMMAND REFERENCE

```
aws sts get-caller-identity          aws s3 ls / cp / sync
aws ec2 describe-instances           aws ec2 run/stop/terminate-instances
aws elbv2 describe-target-health     aws ecs update-service --desired-count N
aws rds describe-db-instances        aws logs tail <group> --follow
aws cloudwatch put-metric-alarm      aws ssm start-session --target i-x
aws secretsmanager get-secret-value --secret-id <id>
```

REVISION SNAPSHOT

ASK YOURSELF	Which command confirms your identity, account, and Region?
DO THIS	Recall ten AWS CLI commands grouped by service.

37. Capstone Challenge

GrowthSchool

IN SIMPLE TERMS
This capstone ties everything together by building and operating the full Shop API stack on AWS.

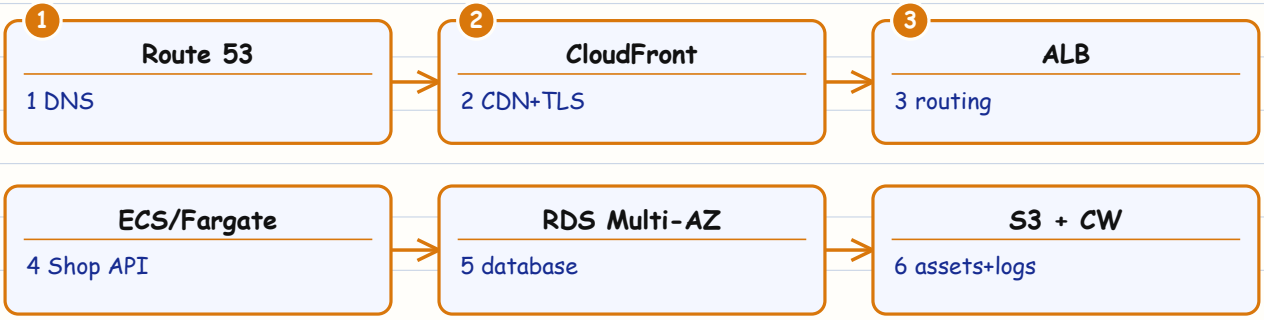
* Build the Shop API on AWS

- VPC with public+private subnets over 2 AZs, an ALB, and ECS Fargate running the Shop API.
- RDS Postgres (Multi-AZ, private) and S3 for assets, all defined in CloudFormation or Terraform.

* Operate It

- IAM roles (no keys), Secrets Manager for the DB password, CloudWatch alarms, CloudTrail on,
- and a pipeline that builds a tagged image to ECR and deploys it.

Shop API on AWS - follow the request 1 -> 6



STUDENT LAB
Deliverables: a reachable HTTPS endpoint, a private DB, least-privilege roles, alarms on high CPU/5xx, a billing budget, and infra fully in code. Tear it all down cleanly afterwards.

REVISION SNAPSHOT

ASK YOURSELF	Is your stack Multi-AZ, least-privilege, observable, and in code?
DO THIS	Complete every deliverable, then destroy the stack to avoid cost.

38. Interview & Revision Notes

GrowthSchool

IN SIMPLE TERMS
This page gathers the key ideas and common interview questions so you can revise quickly.

* Explain Clearly

- Region vs AZ; IAM user vs role; SG (stateful) vs NACL (stateless); public vs private subnet.
- S3 vs EBS vs EFS; RDS Multi-AZ vs read replica; ECS/Fargate vs EKS vs Lambda; SQS vs SNS.

* Common Questions

- How do you give an app permissions without keys? How do you make it highly available? How cut cost?
- Where do secrets live? What does the shared responsibility model mean in practice?

REAL-WORLD EXAMPLE

```
# 30-second self test
aws sts get-caller-identity   # identity + account
# SG vs NACL? role vs key? Multi-AZ vs read replica?
# S3 vs EBS vs EFS? ECS vs EKS vs Lambda? SQS vs SNS?
```

REVISION SNAPSHOT

ASK YOURSELF	Can you explain IAM roles vs access keys in 30 seconds?
DO THIS	Answer every "Common Question" above out loud without notes.