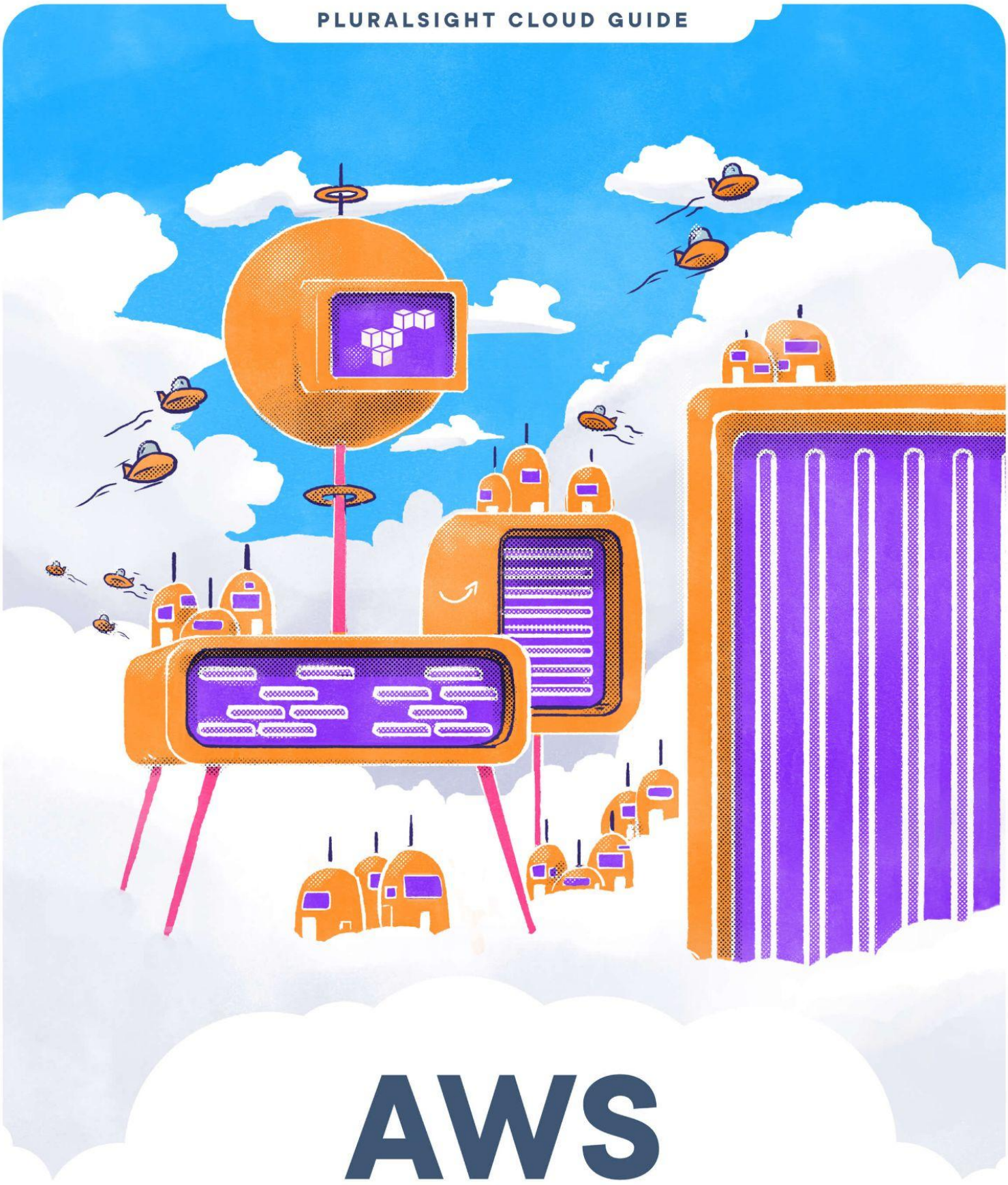
PLURALSIGHT CLOUD GUIDE



AWS
Guidebook

PLURALSIGHT

AWS Guidebook

Last updated: May 2023

Whether you're just diving into the cloud, looking for a refresher, or expanding your knowledge of the different cloud platforms, this guidebook explains the most important terms to help you talk like an AWS local.

We provide an overview explanation for each term to help you understand the lay of the land. Then we dive into the secrets only the AWS locals know—what to avoid and where to spend the most time. When you want to know more, check out the related courses and hands-on labs.

Ready to explore the world of AWS? Dive right in.

INDEX

Azure Azure Cosmos DB | **Google** Cloud Datastore

Amazon DynamoDB

Overview

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale.

Developers can use DynamoDB to rapidly scale applications up or down depending on traffic. Amazon offers on-demand backups and point-in-time recovery through DynamoDB snapshots, protecting from accidental writes or deletes. Amazon replicates your data across multiple availability zones, ensuring that you don't lose your data.

NoSQL works for specific data models that offer flexible schemas. These databases are well suited to modern applications that require high scalability, such as gaming. They excel at scalability and performance.

Relational databases are optimized for storage, while NoSQL is optimized for compute power.

Off the record

Like any tool, DynamoDB and NoSQL come with tradeoffs. Traditional SQL databases support a wide range of queries, but they can struggle to perform cost effectively at scale. NoSQL databases like DynamoDB can scale as big as you want, but it's up to you to store the data in a way that makes it easy to get it out when you need it.

Beware DynamoDB's simple interface and low barrier to entry. If you approach it with SQL-esque assumptions, you may soon find yourself frustrated by the lack of traditional query operations like JOIN and GROUP BY. But if you take the time to learn how to construct performant NoSQL data models, DynamoDB is an unparalleled way to ensure your service isn't collapsing under the weight of its success.

Want to learn more? Our Amazon DynamoDB Deep Dive course offers 15 hours of course material and hands-on labs with all the information you need to build scalable, high-performance applications.

Amazon Elastic Block Store (Amazon EBS) and snapshots

Overview

We can't talk about Amazon Elastic Block Store (Amazon EBS) without talking about snapshots, so we'll cover both.

Amazon EBS is a virtual storage drive, like a hard disk or SSD, for your Amazon EC2 instances. Think of it as the hard drive connected to your server.

Anything your server writes to a disk ends up stored in Amazon EBS as raw blobs of data called blocks. If you want to make a backup of that EBS volume, take a snapshot of it and store it in S3. Each snapshot serves as an incremental backup for an EBS volume.

You can use the Amazon EBS direct APIs to create Amazon EBS snapshots, write data directly to your snapshots, read data on your snapshots, and identify the differences or changes between two snapshots.

Amazon EBS snapshots are incremental: One snapshot contains the changes from the snapshot before it. Each incremental snapshot represents the changes from one volume to the next. Snapshots can be taken in real time while the volume is attached and in use. However, snapshots capture only data that has been written to your Amazon EBS volume, which might exclude any data locally cached by your application or OS.

Off the record

Like Amazon EC2 instances, Amazon EBS volumes scale vertically. If you outgrow your volume, you have to make it bigger rather than splitting it into lots of little volumes. If you run out of space, it's up to you to provision more storage for the volume. If your disk can't keep up with the fast pace of writes to your busy database, you have to increase the IOPS (input/output operations per second). That requires monitoring and general operational awareness. And it can get expensive.

And then there's the complexity of Amazon EBS snapshots. Snapshots tend to save money as they only incrementally increase the data you have backed up rather than back up the entire environment over and over. If you're regularly backing up your data, it makes more sense to take multiple snapshots rather than repeatedly replicate the entire volume. But many organizations fundamentally misunderstand incremental backups.

If you have three snapshots and roll back to the second one, you're deleting the difference in data between the second and third snapshots. So if you were to say the third snapshot is your backup, that isn't necessarily true. Rolling back to that third snapshot means reverting to the machine state at that time, but the third snapshot isn't a backup of all of your data.

Get hands-on with our Performing a Backup and Restore Using AMI and EBS lab. In just 30 minutes, you'll learn how to backup and restore from Amazon EBS snapshots.

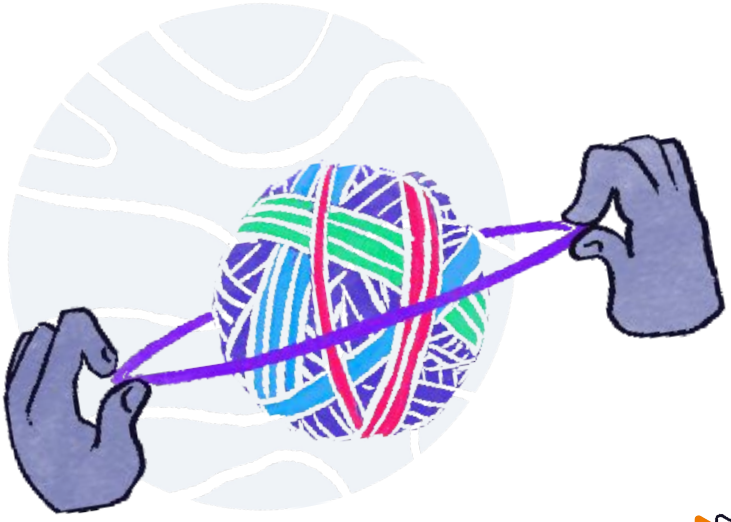
Amazon Elastic Compute Cloud (Amazon EC2)

Overview

The granddaddy of cloud computing services for the better part of two decades, Amazon Elastic Compute Cloud (Amazon EC2) is a hosted version of the virtual machines you might already run in your data center—but backed by the power of AWS. Amazon EC2 reduces the time to obtain and boot new server instances, allowing you to increase or decrease capacity as your needs change. Hence the name "elastic."

Amazon EC2 allows you to set up and configure everything about your instances, from your operating system up to your applications with an Amazon Machine Image (AMI). An AMI is simply a packaged-up environment that includes all the necessary bits to set up and boot your instance.

But there's not just one kind of Amazon EC2 instance. You'll find there's an alphabet soup of instance types ideal for specific scenarios. You can use FPGA hardware for flexible machine learning problems, memory-optimized instances for running databases, and tiny A-series servers for taking your first steps into the cloud.



Off the record

Unlike many cloud services, Amazon EC2 can feel deceptively familiar. It's not that hard to lift and shift workloads from your data center to Amazon EC2 without minimal changes. And therein lies the problem. If you're not careful with spend management, Amazon EC2 can turn out to be less of a bargain than you hoped because you're renting the compute capacity you used to own. This requires cross-functional collaboration as you consider Amazon EC2 pricing options:

- On-demand: On-demand Amazon EC2 uses a fixed rate by the hour (or second), offers flexibility, and helps avoid long-term commitments.
- Reserved instance: Reserve capacity on contracts for one or three years, which can save a lot of money. If you think you can accurately predict how quickly you're going to scale up or down, you can cut costs.
- Savings plans: Closely related to reserved instance, savings plans are a flexible pricing model that offers low prices on Amazon EC2 and Fargate usage in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a one- or three-year term.
- Spot instance: This dirt-cheap server time is based on AWS' excess capacity. But the price will move up and down just like the stock market, and spot instances can be yanked from underneath your application with only a two-minute warning.
- Dedicated hosts: Avoid virtualized time sharing with other AWS customers with physical Amazon EC2 hardware dedicated to you. This option is often necessary if you have specific licensing or regulatory requirements.

Get started with Amazon EC2 with our 30-minute introduction course. Or jump into a hands-on lab to create and interact with an Amazon EC2 instance.

Azure Azure DNS | **Google** Cloud DNS

Amazon Route 53

Overview

Amazon Route 53 is Amazon's DNS web service for AWS. It enables developers to help their users find the quickest route from the end user's computer to the server running an application.

Route 53 is particularly essential for failover or switching to a redundant live system to make sure your user can still use your app. You can use Route 53's health checks to route traffic so the user always arrives at healthy servers.

Users may access your app from all over the world. Route 53 helps you figure out how to build the metaphorical freeway from their computer or phone (or fridge) across the internet to your service.

Off the record

When someone on your team brings up anything related to Route 53, your ears should perk up because it's usually related to your services' overall health. Route 53 routes traffic with different routing types, like latency or geographical closeness, to your server.

Route 53 finds alternate freeways if a sinkhole destroys part of your primary freeway—and the end user shouldn't notice a difference.

Get hands-on with our Configuring Amazon S3 Buckets to Host a Static Website with a Custom Domain lab. In 30 minutes, create a cost-efficient website hosting for sites with files like HTML, CSS, Javascript, fonts, and images.

Azure Azure Virtual Network (VNet) | **Google** Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud (Amazon VPC)

Overview

Think of a VPC as a virtual data center in the cloud. Your data center may have public web servers that are directly accessible to the internet and a more private set of servers that can be accessed only by direct connection or over a VPN. If you're dealing with sensitive or regulated customer information, you'd want to isolate that data even more.

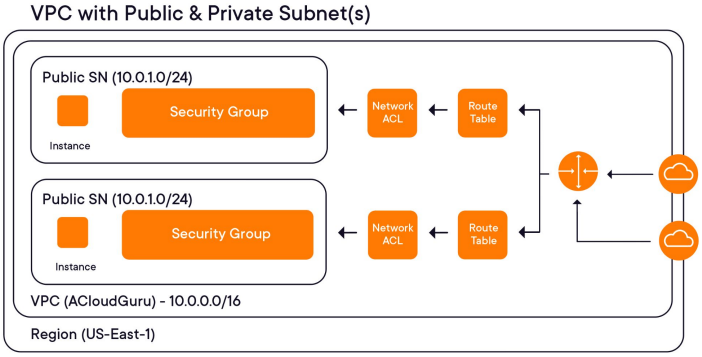
VPCs allow you to provision an isolated section of AWS where you can launch resources in a defined spot. Think of it as a private sandbox for your resources and data without confidential information touching the internet. VPCs are the place to put your database, application servers, back-end reporting processes, and anything you don't want directly exposed to an internet connection.

VPCs allow you to set your IP range, create subnets, and configure root tables and network gateways. You could create a public-facing subnet connected to the web while your backend systems are isolated and not connected to the internet. Multiple layers of security can help you control access to each subnet—what's known as defense-in-depth.

Every AWS account gives you a default VPC out of the box, but you'll likely want to set up your own for improved security and customization.

Off the record

You're trying to keep some data safe and off the internet while connecting internet-facing services to that data and the web simultaneously. That's complex.



All these elements are there to defend your information. But there are many points where that structure could break down. The whole process gets particularly hairy when you're configuring a custom VPC and don't have a ton of expertise in networking.

One example: There's no transitive peering between VPCs. You can have one VPC talk to another VPC, but if that VPC talks to a third one, the first can't speak to the third. If VPC A can talk to VPC B, and B can talk to C, A can't speak to C.

Also, many people are increasingly questioning their need for a VPC. After all, the rise of Zero Trust networking and serverless architectures has moved more workloads out of private networks and into reliance on good authorization strategies like AWS Identity and Access Management (IAM).

But the reality is that anyone trying to connect legacy services to the cloud will probably need to implement a VPC. After all, it's the only way to talk to the private network in your old data center.

Get hands-on with our Configuring a Basic VPC in AWS lab. In less than two hours, create a VPC, subnets across multiple availability zones, routes, and an internet gateway.

AWS Auto Scaling

Overview

AWS Auto Scaling provides a simple, powerful user interface that lets you build scaling plans for resources, including Amazon EC2 instances and Spot Fleets, Amazon Elastic Container Service (Amazon ECS) tasks, Amazon DynamoDB tables and indexes, and Amazon Aurora Replicas. There are three auto-scaling options on AWS right now:

- Amazon EC2 Scaling focuses on Amazon EC2 and nothing else.
- Application auto-scaling controls scaling for stuff that isn't Amazon EC2. You can use this when you're working with DynamoDB, elastic containers, or Apache tools.
- AWS Auto Scaling provides a scalable predictive feature and a central way to manage scalability.

When you're configuring an auto-scaling group, answer these four questions:

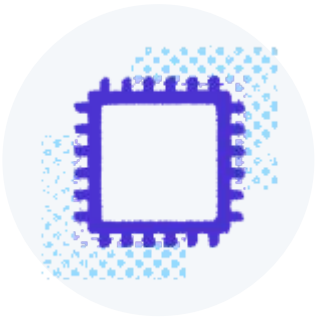
- How many servers do you want to maintain uptime?
- Do you want to adjust your server count manually?
- Do you want to schedule when to scale up or down?
- Would you like to base it on conditions with your product performance?

Off the record

Let's say you've just launched a new product, and suddenly you have a swarm of users frantically trying to pre-order through your app or website. If you haven't configured your auto scaling correctly—for example, you expected them to show at a different time—you're about to collapse under the weight of your success.

Not only do you have to make the right call on which auto-scaling option to use, you also have to pick the right route for auto scaling. The wrong configuration or selections can be costly and time consuming.

Get hands-on with our Launching an Autoscaling Group in AWS lab. In 30 minutes, set up an automation that allows a web server to scale out based on traffic needs.



Availability Zones

Overview

Each AWS region has multiple Availability Zones, which can contain several subnets. You can—and should—utilize multiple Availability Zones to create redundancy. An Availability Zone consists of one or more AWS data centers. When you create a VPC, it spans all Availability Zones across a region, and you can add subnets to each Availability Zone.

Availability Zones exist so AWS can ensure customer applications are consistently accessible despite any issues with the servers, such as component failure or a backhoe cutting the network link to a data center. Availability Zones are situated geographically far enough apart that a localized problem shouldn't affect more than one at a time.

Off the record

Availability Zones create high availability and fault tolerance for your applications. High availability means you're minimizing downtime, while fault tolerance means continuously offering services even if things break down. By launching instances in multiple Availability Zones, you can protect your service from going down.

Say we have two Availability Zones with instances in public subnets in each availability zone. You can configure your Amazon EC2 instance in one Availability Zone to automatically replicate over to another Amazon EC2 instance in another Availability Zone in the case of a failover scenario. But you still have to ensure you set up those backups first.

AWS Lambda

Overview

AWS Lambda lets you run code on an execution environment, which is even more abstracted than an Amazon EC2 instance. That's why Lambda-based systems are often called serverless. There are still servers somewhere; you just don't have to know or care about them.

For example, when you talk to Alexa, your question might not query a specific relational database to capture a response. Instead, it may pass a request through AWS API Gateway to a Lambda function. And those Lambda functions can trigger another Lambda function. In classic AWS fashion, it's Lambdas all the way down.

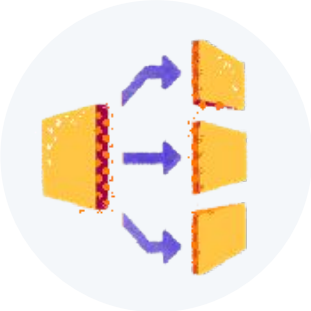
To be clear, this kind of scaling is different from auto scaling if you have users spread across multiple servers. A million users hitting your site may trigger a million Lambda events. The difference between 1 user and 1 million users calling that action is a function of cost, not speed. You pay only whenever your application executes code.

Off the record

It's a whole different paradigm. Lambda functions have short runtime limits, rely on event-driven programming, and don't play well with some legacy technologies.

And yet many people refer to serverless architecture as the future of applications—for good reason. But from a business perspective (especially if you're the one with the team corporate card), the primary reason is the cost savings of serverless architectures. The free tier provides one million requests per month and only 20 cents per million requests after that. And that's on top of all the sysadmin time you're freeing up.

Learn to utilize AWS Lambda to transform data and build basic ETL pipelines in our Processing Serverless Data Using AWS Lambda course.



Elastic Load Balancing (ELB)

Overview

It's exactly what it sounds like: a physical or virtual device designed to help you balance the network load across multiple servers. You have a variety of ways to plan out your auto-scaling with three options:

- **Application load balancers (ALBs)** can see inside your application and the requests it's making. Application load balancers are a good fit for a scaling engine that makes intelligent decisions for you.
- **Network load balancers (NLBs)** work best when you need to focus on performance. These load balancers operate at the connection level.
- **Classic load balancers** are legacy elastic load balancers. This is a cheap option if you don't care how AWS routes your traffic. (Amazon has largely deprecated these.)

When in doubt, use the ALB, which comes with more features and is easier to configure. Use an NLB when you need to handle millions of requests per second.

Off the record

First, you have to select the best load balancer. And second, you have a lot of specific features you can enable to make the load balancer more efficient. As always, there are tradeoffs. The challenge is maximizing the value you get out of your cloud configurations.

Application load balancers are best-suited for HTTP and HTTPS traffic operating at Layer 7. Applications use network load balancers to balance TCP traffic, working on the connection level (or Layer 4). Legacy balancers can use Layer-7-specific features for HTTP/HTTPS applications, but it isn't application aware.

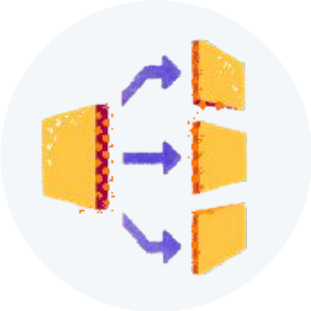
Once you've selected a load balancer, you'll need to decide how you'd like to route your traffic to your various Amazon EC2 instances. These instances may be across multiple availability zones and tied to different load balancers. But it's possible to intelligently route traffic to optimize your performance for each Amazon EC2 instance. Here are a few top-level configurations you'll have to address:

Sticky sessions ensure that only one user is tied to one Amazon EC2 instance. But you may also end up with unused Amazon EC2 instances by using sticky sessions.

Cross-zone load balancing automatically balances your incoming traffic across all Amazon EC2 instances regardless of the availability zone you're currently using. You may sacrifice some performance by sending your traffic into different geographic regions if you use cross-zone load balancing.

Path patterns route traffic intelligently based on URL paths, such as myurl.com/images. This may not account for surges in traffic to specific pages—like a celebrity posting an image on your images page.

Get hands-on with our Creating a Load Balancer lab. In 90 minutes, install and configure a load balancer as the front end for two pre-built Apache nodes.



Identity and Access Management (IAM)

Overview

AWS Identity and Access Management (IAM) allows you to manage users and their level of access to an AWS console. It enables you to set up users, groups, permissions, and roles and grant access to different parts of the AWS platform. You can also set up granular permissions down to an individual user getting access to one service and not another.

IAM is core to any cloud usage, from giving developers access to resources so they can push updates to production or giving auditors access to inspect your work. And IAM is a global service—IAM resources are managed at the AWS account level, not just specific regions or availability zones.

Ultimately, IAM is how AWS resources speak to each other, how you audit them, and how you control access for your developers to update them. That means getting IAM right has implications beyond the security team.

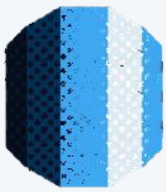
Off the record

AWS IAM is a bespoke system, built from the ground up to handle the authentication and authorization challenges of massively distributed cloud applications. So even though it's fundamental to the success of your AWS deployment, developers often have to learn it from scratch. It's not like anything you had in your data center.

You should always set up multi-factor authentication (MFA) on root AWS accounts and customize password rotations. If you're a little too cavalier when using identity federation, one compromised account could end up leading to a breach across your entire AWS footprint. And long-lived access and secret keys are easy to provision for users but could let an outside hacker assume the rights of your employees if those keys leak.

Ultimately, there's no substitute for a careful plan when it comes to laying out your AWS IAM strategy. Infrastructure as code, federated identities, and properly restrictive policies aren't simple defaults to implement, but they go a long way toward ensuring that your systems have exactly the access they need.

Azure Azure SQL Database | **Google** Cloud SQL



Amazon Relational Database Service (Amazon RDS)

Overview

Amazon Relational Database Service (Amazon RDS) is a managed database that manages everything. Amazon RDS abstracts the whole process of running and maintaining a database so you're engaging it only when you need to read or write data. CPU, memory, storage, and IOPS are split. You can also scale them independently, bringing each one up or down.

The aim is to lower the overall cost of ownership and strip away routine tasks. That can include provisioning, backup, recovery, and other core requirements for any system. As a result, you can focus on higher-level tasks.

Off the record

Amazon RDS gives you a lot of database engines to choose from. There are commercial databases, like Oracle and Microsoft SQL Server, and open-source engines, like MariaDB, Postgres, and MySQL. And then there are cloud-native solutions, like Amazon Aurora. Amazon RDS isn't a magic bullet to getting everything else off your plate. You'll still have a lot of things to consider, such as:

- What database engine you're using
- How you're handling backups and monitoring
- The size of your underlying storage and compute

And then there's a lot to configure. And a lot that can go wrong, especially when you're planning for a set-it-and-forget-it tool, specifically as you scale up. When things start to go south, you'll have to dig back through and reconfigure some of your setups (and hope you aren't trying to repair something while it's already in action).

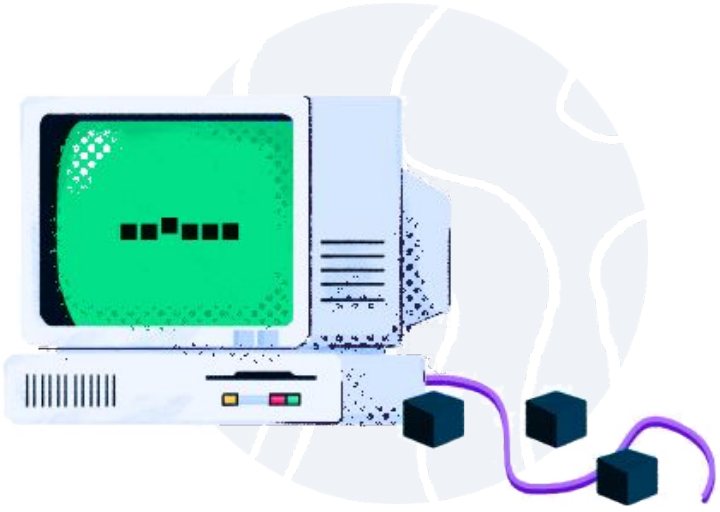
Build your knowledge of relational databases with our Introduction to Amazon RDS course. Learn to leverage the power of Amazon RDS in your own projects, then set up a WordPress site using Amazon EC2 and Amazon RDS in this hands-on lab.

Amazon Simple Queue Service (Amazon SQS)

Overview

Amazon Simple Queue Service (Amazon SQS), first announced in 2004, is one of AWS's oldest services. It provides access to a message queue to store messages while waiting for a computer to process them. Amazon SQS enables you to decouple components of an application and let them communicate asynchronously on their own schedules. Each message is sent to a service like a Lambda function or Amazon EC2 instance.

Those decoupled components can store messages in a fail-safe queue that waits for computation. This can be really useful if, for example, you lose an availability zone. The message becomes available in the queue if an Amazon EC2 instance goes down, enabling other Amazon EC2 instances to pick up the slack.



Off the record

If you're building on Amazon SQS, you are, by definition, creating a distributed application that works across a network via decoupled pieces. And that brings with it a whole new layer of design and operational considerations.

Choose whether you want to create a standard queue or a FIFO (first in, first out) queue. Both have advantages:

- **Standard queues** ensure that your message is delivered at least once and make a best effort to arrange them in the same order as received. But that's a best effort, and you shouldn't assume order.
- **FIFO queues** deliver messages in the exact order they were received. The message is delivered and remains available until a consumer processes it and deletes it. Duplicates are not allowed into the queue in this case.

Amazon SQS also implements visibility timeout—how long the queue waits for one consumer before making it available to the next one. For example, a message is made available to one Amazon EC2 instance and isn't executed in some target time. That message will then be available for another Amazon EC2 instance to grab it and run it. But this also means Amazon SQS may deliver your message more than once.

Get hands-on with our Working with AWS SQS Standard Queues lab. Send messages to an Amazon SQS queue that you create and learn how to use multiple Amazon SQS consumers to process queue data at the same time.

Azure Azure Virtual Network | **Google** VPC Virtual Network

Subnets

Overview

Shorthand for “subnetwork,” a subnet is a subsection of a network. When you create a VPC, you’ll have a series of subnets associated with all the applications within a specific availability zone. You can provision resources, like an Amazon EC2 instance or an Amazon RDS database, within particular subnets that can be public or private.

Public subnets have a route to the internet that’s associated with an internet gateway. Public subnets can also talk to other public subnets. Private subnets do not have a path to the internet, but they can connect to public subnets within your VPC.

Off the record

You can’t have one subnet across multiple availability zones. You’ll probably hear something along the lines of “one subnet equals one availability zone.” Let’s say you’ve decided to launch a VPC within a particular region, and AWS offers a set of availability zones within that region. If you'd like to keep some information private, like a set of customer information in an Amazon RDS database, you’d launch a private subnet within one availability zone.

However, suppose you want to launch a subnet within a different availability zone. That subnet can't talk to your private subnet in a different availability zone—your private subnet won’t span multiple availability zones. This is an important consideration when figuring out how to handle disaster recovery.

Get started with subnets in our DevOps Subnetting Fundamentals course. Learn the basics of subnetting and the strategies to use in your projects.

About Pluralsight

Pluralsight helps organizations around the globe advance their technology workforce. Because the hardest part of building a business isn't building software and technology. It's building up the people who grow your business. That's why everyone from CIOs to developers trust Pluralsight—the only partner that helps leaders build better teams and better products, all at the same time.

Our software and solutions are purpose-built to address your top challenges and outcomes:

- Onboard new engineers faster
- Build products faster and improve the developer experience
- Develop internal cloud talent and enable cloud transformation
- Improve retention and cut hiring costs
- Improve cycle times and reduce burnout for remote teams
- Develop teams that deliver on key tech initiatives
- Increase delivery speed and overcome Agile roadblocks
- Hire job-ready, diverse talent
- Build fluency and collaboration organization-wide

Our cloud transformation solutions help you create the cloud talent you need, when you need it, to deliver on your biggest, boldest vision. Pluralsight Skills delivers expert-authored courses in the latest cloud technologies, paired with unlimited access to hands-on labs, sandboxes, and certification prep. Upskilling your teams with Skills equips your teams to execute on strategic cloud investments that ultimately drive innovation, automation, and efficiency.

PLURALSIGHT