AWS Account Setup Guide

Financial and Technical Overview for Full-Stack Web Application Deployment

This guide provides a comprehensive overview of creating and configuring an AWS account to deploy and manage a full-stack web application with enterprise-level scalability and reliability. It includes financial estimates, security configurations, service selection, and technical implementation steps.

1. AWS Account Setup & Initial Configuration

Step 1: Create an AWS Account

1. Visit https://aws.amazon.com and click **"Create an AWS Account."**
2. Enter a valid email address, create a strong password, and provide your desired AWS account name.
3. Submit your billing information (credit card is required, even for Free Tier users).
4. Verify your identity using a phone number (SMS or voice call).
5. Choose a support plan. Select **"Basic"** for a free option.

Step 2: Secure the Root Account & Enable MFA

1. Log in to your AWS Console using root credentials.
2. Enable **Multi-Factor Authentication (MFA)** using Google Authenticator or Authy.

3. Navigate to **IAM (Identity and Access Management)** → **Users** → **Add User**.
4. Create an administrative IAM user with **AdministratorAccess** permissions.
5. Enable MFA for this IAM user to further secure access.

Step 3: Set Up Billing Alerts

1. Go to **Billing Dashboard** → **Budgets** → **Create Budget**.
2. Choose **Cost Budget** and set a monthly spending limit (e.g., $50).
3. Configure notifications to send email alerts when thresholds are crossed.

2. Financial Overview (Cost Estimation)

AWS Free Tier (First 12 Months)

- EC2: 750 hours/month of t2.micro or t3.micro instances
- RDS: 750 hours/month of db.t2.micro
- S3: 5GB of standard storage
- Lambda: 1 million requests/month
- CloudFront: 50GB data transfer/month

Estimated Monthly Costs (Post-Free Tier)

Service	Usage	Estimated Monthly Cost (USD)
EC2	2× t3.medium (for production servers)	~$60

RDS	db.t3.medium with Multi-AZ setup	~$100
S3	50GB storage + 10GB data transfer	~$5
CloudFront	100GB data transfer	~$10
Lambda	5 million requests	~$5
Networking	NAT Gateway + Inter-zone Transfer	~$30
GitHub Copilot	Developer productivity tool	~$10
Total	—	**~$220/month**

Note: Actual costs depend on region, usage, and added services (e.g., Elasticache, EKS, etc.).

3. Technical Overview: AWS Infrastructure Setup

Core AWS Services

Category	AWS Services	Purpose
Compute	EC2, Lambda, ECS, EKS	Host backend APIs and microservices
Database	RDS (PostgreSQL/MySQL), DynamoDB	Store relational and NoSQL data
Storage	S3, EBS, EFS	Store static files, backups, persistent data
Networking	VPC, Route 53, ALB, NAT Gateway	Secure networking, load balancing, DNS
Security	IAM, AWS WAF, Shield, Secrets Manager	Access control, security, DDoS protection
DevOps	CodePipeline, CodeDeploy, CodeBuild	Continuous Integration/Deployment (CI/CD)
Monitoring	CloudWatch, AWS X-Ray	Logs, metrics, traces, alerts

Step-by-Step Deployment Process

1. Set Up VPC and Subnets

- Create a custom **VPC** with **public** and **private** subnets in multiple Availability Zones (e.g., `us-east-1a`, `us-east-1b`).
- Attach **Internet Gateway** to allow public subnet access.
- Configure **NAT Gateway** for internet access from private subnets.

2. Deploy Backend Services

- Use **EC2 instances** for virtual machines running backend servers (Node.js, Python, etc.).
- Or use **ECS/EKS** for managing containerized services via Docker/Kubernetes.
- For managed platform deployment, use **Elastic Beanstalk** for faster provisioning.

3. Configure the Database

- Set up **Amazon RDS** using PostgreSQL/MySQL with Multi-AZ deployment for high availability.
- Use **DynamoDB** for NoSQL storage when high throughput and scalability are needed.

4. Frontend Hosting

- Upload compiled frontend files (e.g., React, Angular) to an **S3 bucket**.
- Enable static website hosting on S3.
- Configure **CloudFront CDN** for content delivery and SSL support.

5. CI/CD Automation

- Connect **AWS CodePipeline** to GitHub or GitLab.
- Use **CodeBuild** to build the application on each commit.

- Deploy changes using **CodeDeploy** (for EC2/ECS targets).

6. Security Configurations

- Use **AWS WAF** to block common web attacks like SQL injection and XSS.
- Store sensitive data (API keys, credentials) in **Secrets Manager**.
- Restrict IAM policies to follow the principle of least privilege.

7. Monitoring and Alerts

- Use **CloudWatch Dashboards** to track CPU, memory, disk, and database metrics.
- Set up **CloudWatch Alarms** to notify via **SNS (Simple Notification Service)**.
- Use **AWS X-Ray** to trace application requests and performance bottlenecks.

4. Best Practices for Cost Optimization

1. Use **Spot Instances** for non-critical or batch workloads.
2. Enable **Auto Scaling** to dynamically adjust resources during traffic changes.
3. Schedule **EC2 and RDS** instances to shut down during non-working hours in development environments.
4. Apply **S3 Lifecycle Policies** to move infrequently accessed files to **Glacier** storage.
5. Monitor **Billing Dashboard** regularly to detect cost anomalies.
6. Use **Trusted Advisor** for cost-saving recommendations and security insights.

Conclusion

Setting up AWS for full-stack applications involves careful planning, cost estimation, and implementation of best practices. Following this guide ensures secure, scalable, and cost-effective infrastructure ready for production workloads. Be proactive about monitoring, scaling, and budgeting to maintain system health and control expenses.