AWS Hands-on Detailed Guides

This document contains six step-by-step AWS implementation guides with exact console/CLI locations.

# 1. Hosting Static Websites on AWS S3 and EC2

1. A) Hosting on Amazon S3
2. 1. Go to AWS Console → S3 → Create bucket → Enter a unique bucket name.
3. 2. Region: Choose the nearest region (e.g., ap-south-1).
4. 3. Permissions: Uncheck “Block all public access” (only if you want public access).
5. 4. After bucket creation: Go to “Properties” tab → Enable “Static website hosting”.
6. 5. Specify index.html and error.html as documents.
7. 6. Upload files: Go to “Objects” → Upload → Add files.
8. 7. Permissions: Go to “Permissions” → Bucket Policy → Add public-read policy if needed.
9. 8. Access the site via “Website endpoint” provided under Properties.

For HTTPS + Custom Domain:

* - Use Route 53 → Create hosted zone → Add A record alias to CloudFront.
* - Create a CloudFront distribution with your S3 bucket as origin.
* - Attach an SSL certificate from ACM in region us-east-1.

1. B) Hosting on Amazon EC2
2. 1. AWS Console → EC2 → Launch instance → Choose Amazon Linux 2023 AMI.
3. 2. Instance type: t2.micro or t3.micro.
4. 3. Security group: Allow inbound 22 (SSH), 80 (HTTP), 443 (HTTPS).
5. 4. Connect to instance via SSH: ssh -i key.pem ec2-user@public-ip
6. 5. Install web server: sudo dnf install -y nginx
7. 6. Deploy files: sudo mv index.html /usr/share/nginx/html/
8. 7. Access site at http://EC2\_PUBLIC\_IP
9. 8. To bind domain: Route 53 → A record → Point to Elastic IP of EC2.
10. 9. For HTTPS: Use Certbot on EC2 or create an ALB with ACM certificate.

# 2. EC2 Setup and MySQL Database Management

1. 1. AWS Console → EC2 → Launch instance → Select Amazon Linux 2023.
2. 2. Security group: allow 22 (SSH), 3306 (MySQL) (restrict to your IP).
3. 3. Connect: ssh -i key.pem ec2-user@public-ip
4. 4. Install MySQL: sudo dnf install -y mysql-server
5. 5. Start service: sudo systemctl start mysqld
6. 6. Run: sudo mysql\_secure\_installation → set root password, remove test DB.
7. 7. Connect to MySQL: mysql -u root -p
8. 8. Create DB: CREATE DATABASE company\_db;
9. 9. Create user: CREATE USER 'appuser'@'%' IDENTIFIED BY 'StrongPass123!';
10. 10. Grant access: GRANT ALL PRIVILEGES ON company\_db.\* TO 'appuser'@'%';
11. 11. Create table: CREATE TABLE employees(...);
12. 12. Create trigger: Use CREATE TRIGGER before\_salary\_update BEFORE UPDATE...
13. 13. Create stored procedure: DELIMITER // CREATE PROCEDURE GetEmployeesByDept(...);
14. 14. Backup: mysqldump -u root -p company\_db > backup.sql
15. 15. Restore: mysql -u root -p company\_db < backup.sql

# 3. Web Application Deployment using AWS Elastic Beanstalk

1. 1. Install AWS CLI and EB CLI locally.
2. 2. Prepare app: Flask app with application.py entry point.
3. 3. Run: eb init -p python-3.11 my-app → Select region → IAM role.
4. 4. Run: eb create my-app-env → creates environment with EC2 + ALB.
5. 5. Deploy: eb deploy
6. 6. To add database: During eb create, choose “Configure more options” → Add RDS.
7. 7. To set env vars: eb setenv DB\_HOST=... DB\_USER=...
8. 8. Access app at URL: http://my-app-env.region.elasticbeanstalk.com
9. 9. For custom domain: Route 53 → Alias record → Point to EB environment CNAME.
10. 10. For HTTPS: Go to EB Console → Configuration → Load Balancer → Add HTTPS listener → Attach ACM cert.
11. 11. Logs: eb logs ; Health: eb health

# 4. Serverless Computing – S3 and Lambda Integration

1. 1. Create bucket: AWS Console → S3 → Create bucket.
2. 2. Upload sample file to test trigger.
3. 3. Create Lambda: Console → Lambda → Create function → Author from scratch.
4. 4. Runtime: Python 3.11; Execution role: Create new role with basic Lambda + S3 access.
5. 5. Code: Add handler code to process S3 object and write to output bucket.
6. 6. Add Trigger: In Lambda → Add trigger → Choose S3 → Event type “All object create events”.
7. 7. Specify prefix (e.g., incoming/) and suffix (.txt).
8. 8. Save → Now uploads trigger Lambda automatically.
9. 9. Test: aws s3 cp file.txt s3://bucket/incoming/
10. 10. Logs: CloudWatch → Log groups → /aws/lambda/function-name.
11. 11. For presigned URL: Use boto3 generate\_presigned\_url API.
12. 12. For failures: Configure DLQ in Lambda configuration → Async invocation settings.

# 5. EC2 Auto Scaling using Launch Templates and Scaling Policies

1. 1. Create Launch Template: EC2 → Launch Templates → Create → Choose AMI, instance type, SG, IAM role, add User Data script.
2. 2. Create Target Group: EC2 → Target Groups → Create target group → Instances, HTTP:80.
3. 3. Create Load Balancer: EC2 → Load Balancers → Create ALB → Attach SG → Add listener HTTP:80 → Forward to TG.
4. 4. Create Auto Scaling Group: EC2 → Auto Scaling Groups → Create → Choose Launch Template → Select subnets across 2 AZs → Attach ALB Target Group.
5. 5. Set min=2, max=6, desired=2.
6. 6. Scaling Policies:
7. a) Target Tracking: Add scaling policy → Choose CPUUtilization → Target 50%.
8. b) Step Scaling: Create CloudWatch alarms → Attach to scale-in/out policies.

c) Scheduled Scaling: Set fixed capacity for time schedules.

1. 7. Updates: Use Instance Refresh in ASG console to roll out AMI updates.
2. 8. Advanced: Configure Warm Pools and Lifecycle Hooks for graceful termination.

# 6. S3 Bucket File Management and Public Access Configuration

1. 1. Create bucket: S3 → Create bucket → Enter name → Region → Keep “Block Public Access ON” for private use.
2. 2. Upload: S3 → Objects → Upload → Add files.
3. 3. CLI upload: aws s3 cp file.txt s3://my-bucket/
4. 4. Organize with prefixes (folders): e.g., incoming/, processed/.
5. 5. Access:

* - Private: Use IAM policies for app roles.
* - Public: Disable Block Public Access → Bucket policy with s3:GetObject for "\*".
* - Specific files: Upload with --acl public-read.
* - Temporary access: Use aws s3 presign for presigned URLs.

1. 6. Enable Versioning: S3 → Properties → Bucket Versioning → Enable.
2. 7. Add Lifecycle Rules: Management tab → Lifecycle rules → Transition to Glacier after X days.
3. 8. Encryption: Properties → Default encryption → SSE-S3 or SSE-KMS.
4. 9. Monitoring: Enable CloudTrail data events for S3; Enable Access Logs to another bucket.
5. 10. Static Website Hosting: Properties → Enable website hosting → Set index.html and error.html → Access via endpoint.