



## OVERVIEW & SETUP

RDS managed cloud service  
# Install CLI  
aws configure / az login / gcloud  
auth login  
# Set region  
aws configure set region us-east-1  
Use IAM roles in production, not  
access keys

## AUTHENTICATION & IAM

IAM Roles, Policies, and Least  
Privilege  
# Assume role  
aws sts assume-role --role-arn  
arn:aws:iam::123:role/MyRole \  
--role-session-name session  
# Attach policy to role  
aws iam attach-role-policy --role-name  
MyRole \  
--policy-arn  
arn:aws:iam::aws:policy/ReadOnlyAccess

## CORE FEATURES

Key capabilities and use cases for  
this service  
# Create resource  
aws <service> create-<resource> --name  
MyResource  
# List resources  
aws <service> list-<resources>  
# Describe resource  
aws <service> describe-<resource> --name  
MyResource

## CONFIGURATION

# AWS CLI config: ~/.aws/credentials  
[default]  
aws\_access\_key\_id = AKIA...  
aws\_secret\_access\_key = ...  
region = us-east-1  
output = json  
Prefer environment variables or  
instance profiles over config files

## MONITORING & ALERTS

Enable CloudWatch / Azure Monitor /  
Cloud Monitoring  
# Create CloudWatch alarm  
aws cloudwatch put-metric-alarm \  
--alarm-name HighCPU \  
--comparison-operator  
GreaterThanOrEqualToThreshold \  
--threshold 80  
Set dashboards for key metrics  
before production launch

## SECURITY BEST PRACTICES

Enable encryption at rest and in  
transit  
# Enable server-side encryption  
--sse-algorithm aws:kms  
--kms-key-id arn:aws:kms:...  
Rotate credentials regularly  
Enable CloudTrail / activity logs  
for audit  
Use VPC for network isolation

## SDK USAGE

# Python (boto3)  
import boto3  
client = boto3.client('s3',  
region\_name='us-east-1')  
client.upload\_file('file.txt',  
'my-bucket', 'key')  
# Node.js  
const AWS =  
require('@aws-sdk/client-s3')  
Use SDK over direct HTTP for  
automatic retry/signing

## CLI COMMANDS

aws <service> help  
aws <service> list-<resources> --query  
'...' --output table  
aws <service> create-<resource>  
--cli-input-json file://input.json  
aws <service> delete-<resource> --name  
...  
aws cloudformation deploy  
--template-file template.yaml

## INTEGRATION PATTERNS

Event-driven: trigger functions on  
service events  
# SNS Lambda SQS pattern  
# API Gateway Lambda DB  
Use managed services over  
self-hosted where possible  
Infrastructure as Code:  
Terraform/CDK/CloudFormation  
Multi-region for disaster recovery

## PRICING & COST OPT

Understand pricing model: per  
request, per GB, per hour  
Use Reserved Instances / Committed  
Use for 30-70% savings  
Enable Cost Explorer / budget alerts  
Right-size resources regularly  
# Tag resources for cost allocation  
aws <service> tag-resource --tags  
Project=myapp Env=prod

## DEPLOYMENT PATTERNS

Blue/Green: shift traffic between  
environments  
Canary: gradual traffic shift with  
rollback  
# Infrastructure as Code  
terraform init && terraform plan &&  
terraform apply  
# Or CloudFormation  
aws cloudformation deploy --stack-name  
myapp \  
--template-file infra.yaml

## COMMON GOTCHAS

Never commit credentials to git use  
secrets manager  
Check service quotas before scaling  
Test in staging before production  
deployments  
Enable MFA for root account and  
admin users  
Review IAM policies regularly  
remove unused permissions