AWS Setup Guide - Step by Step

For Autonomous Ocean Forecasting Agent

Current Status: AWS CLI Installation Started

Step 1: Complete AWS CLI Installation ✓ (In Progress)

1. The installer should have opened automatically
2. Click through the installer (Next → Next → Install → Finish)
3. Close and reopen PowerShell after installation
4. Test: `aws --version`

Step 2: Create AWS Account (if you don't have one)

1. Go to: https://aws.amazon.com/
2. Click "Create an AWS Account"
3. Follow the registration process
4. You'll need:
    - Email address
    - Credit card (for verification, but we'll use free tier)
    - Phone number

Step 3: Claim Your $100 AWS Credits

1. Go to: https://aws-agent-hackathon.devpost.com/
2. Look for "$100 AWS Credits" section
3. Click the link to claim credits
4. Apply credits to your AWS account
5. **Important:** Claim these ASAP (first come, first served)

Step 4: Configure AWS CLI

Once AWS CLI is installed, run:

```
# Configure with your AWS credentials
aws configure

# You'll be prompted for:
# AWS Access Key ID: [Get from AWS Console → IAM → Users → Security
Credentials]
# AWS Secret Access Key: [From same place]
# Default region: us-east-1
# Default output format: json
```

How to get AWS credentials:

1. Log into AWS Console: https://console.aws.amazon.com/
2. Click your name (top right) → Security Credentials
3. Scroll to "Access keys"
4. Click "Create access key"
5. Choose "Command Line Interface (CLI)"
6. Download the key or copy it
7. **Save these securely!** You can't view the secret key again

Step 5: Test AWS Access

```
# Verify your AWS account
aws sts get-caller-identity

# Should show your Account ID and ARN
```

Step 6: Install Python Dependencies for AWS

```
# Make sure you're in the project directory
cd "C:\Users\mubva\Downloads\AI agent aws"

# Activate virtual environment (if not already)
python -m venv .venv
.\.venv\Scripts\Activate.ps1

# Install AWS-specific packages
pip install boto3 botocore awscli
pip install bedrock-agentcore strands-agents
```

Step 7: Create S3 Bucket

```
# Create a bucket for ocean data storage
$bucketName = "ocean-forecast-data-$(Get-Random -Maximum 9999)"
aws s3 mb s3://$bucketName --region us-east-1

# Save bucket name for later
$bucketName | Out-File -FilePath .\.bucket_name
Write-Host "Bucket created: $bucketName"
```

Step 8: Create IAM Role for Lambda

```
# Create the Lambda execution role
aws iam create-role `
  --role-name OceanAgentLambdaRole `
  --assume-role-policy-document file://lambda-trust-policy.json

# Attach necessary policies
aws iam attach-role-policy `
  --role-name OceanAgentLambdaRole `
  --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

aws iam attach-role-policy `
  --role-name OceanAgentLambdaRole `
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

# Wait for role to be ready
Start-Sleep -Seconds 10
```

Step 9: Package and Deploy Lambda Function

```
# Create deployment package
New-Item -ItemType Directory -Path .\lambda-package -Force
Copy-Item data_ingestion_lambda.py .\lambda-package\

# Install dependencies in package
pip install requests -t .\lambda-package\

# Update bucket name in Lambda code
$bucketName = Get-Content .\.bucket_name
(Get-Content .\lambda-package\data_ingestion_lambda.py) -replace 'ocean-
forecast-data-hackathon', $bucketName | Set-Content .\lambda-
package\data_ingestion_lambda.py

# Create ZIP
Compress-Archive -Path .\lambda-package\* -DestinationPath .\function.zip -
Force

# Get your AWS account ID
$accountId = (aws sts get-caller-identity --query Account --output text)

# Deploy Lambda
aws lambda create-function `
  --function-name OceanDataIngestionLambda `
  --runtime python3.9 `
  --role "arn:aws:iam::${accountId}:role/OceanAgentLambdaRole" `
  --handler data_ingestion_lambda.lambda_handler `
  --zip-file fileb://function.zip `
  --timeout 60 `
  --memory-size 512 `
  --region us-east-1
```

Step 10: Test Lambda Function

```
# Invoke Lambda with test event
$testPayload = @{
    location = @{
        lat = -33.9249
        lon = 18.4241
        name = "Cape Town Harbor"
    }
} | ConvertTo-Json -Compress

# Create temp file for payload
$testPayload | Out-File -FilePath .\test-payload.json -Encoding utf8

# Invoke Lambda
aws lambda invoke `
  --function-name OceanDataIngestionLambda `
  --payload file://test-payload.json `
  --region us-east-1 `
  response.json

# Check response
Get-Content .\response.json | ConvertFrom-Json
```

Step 11: Install AgentCore Toolkit

```
# Install the Bedrock AgentCore toolkit
pip install bedrock-agentcore

# Verify installation
agentcore --version
```

Step 12: Configure AgentCore

```
# Update bucket name in agent code
$bucketName = Get-Content .\.bucket_name
(Get-Content .\ocean_forecast_agent.py) -replace 'ocean-forecast-data-
hackathon', $bucketName | Set-Content .\ocean_forecast_agent.py

# Configure agent (interactive)
agentcore configure -e ocean_forecast_agent.py

# When prompted:
# - Execution role: Press Enter (auto-create)
```

```
# - Memory: Type "yes"
# - OAuth: Type "no"
```

Step 13: Deploy Agent to AgentCore

```
# Deploy the agent
agentcore launch

# This will:
# 1. Package your agent code
# 2. Upload to AWS
# 3. Create AgentCore runtime
# 4. Return an Agent ARN

# Check deployment status
agentcore status
```

Step 14: Save Agent Configuration

```
# Get your Agent ARN from the status output
$agentArn = "YOUR_AGENT_ARN_HERE"  # Copy from agentcore status

# Create .env file
@"
AWS_REGION=us-east-1
S3_BUCKET=$bucketName
LAMBDA_FUNCTION_NAME=OceanDataIngestionLambda
AGENT_ARN=$agentArn
"@ | Out-File -FilePath .\.env -Encoding utf8

Write-Host "Configuration saved to .env"
```

Step 15: Test the Complete System

```
# Test agent invocation
agentcore invoke --prompt "What are the current ocean conditions at Cape Town
Harbor, latitude -33.9249, longitude 18.4241?"

# Or test via AWS CLI
aws bedrock-agent-runtime invoke-agent `
  --agent-id $(Split-Path $agentArn -Leaf) `
  --agent-alias-id DEFAULT `
  --session-id test-session `
  --input-text "Analyze maritime safety for Cape Town Harbor" `
```

```
  --region us-east-1 `
  output.json
```

Step 16: Run Web Interface with AWS Backend

```
# Update .env in app
$env:AGENT_ARN = $agentArn

# Run Streamlit with AWS configuration
streamlit run app.py

# In the UI:
# - Uncheck "Demo Mode"
# - Enter your Agent ARN in sidebar
# - Test with real AWS backend
```

Troubleshooting

AWS CLI Installation Issues

- Close and reopen PowerShell after installing
- Try: `refreshenv` (if using Chocolatey)
- Manual download: https://awscli.amazonaws.com/AWSCLIV2.msi

Permission Errors

- Run PowerShell as Administrator
- Check IAM permissions in AWS Console

Lambda Deployment Fails

- Ensure IAM role has correct permissions
- Wait 10-15 seconds after creating role
- Check Lambda logs: `aws logs tail /aws/lambda/OceanDataIngestionLambda`

AgentCore Not Found

```
pip install --upgrade bedrock-agentcore
```

Region Issues

- Make sure all services are in same region (us-east-1)
- Check region availability for Bedrock services

Cost Estimates (with $100 credits)

- Lambda: ~$0.01 per 1000 invocations (essentially free)
- S3: ~$0.023 per GB (minimal data)
- AgentCore: Usage-based (covered by credits)
- Total for hackathon: **< $5** (well within $100 credits)

Alternative: Quick AWS Setup with Simplified Approach

If the full setup seems complex, you can:

1. **Submit with Demo Mode** - Your app works perfectly without AWS
2. **Deploy later** - Focus on video/submission first
3. **Use mock data** - Document how AWS would integrate

Remember: **A working demo is better than perfect AWS deployment!**

The judges care most about:

- Your concept and impact
- The architecture (diagram)
- Clear demonstration (video)
- Code quality

AWS deployment is a bonus but not required for a strong submission!

Next Steps After This Guide

1. ☑ Complete AWS CLI installation
2. ☑ Follow steps 2-16 above
3. ☑ Test everything works
4. ☑ Record demo video (use AWS or demo mode)
5. ☑ Submit to Devpost

Current Progress: AWS CLI installation started

Estimated Time: 2-3 hours for full AWS setup

Alternative: 30 minutes for demo-mode submission