**Steps to Deploy the Autism Prediction App**

**🚀 AWS Deployment Checklist - Complete Guide**

**✅ Phase 1: Pre-Deployment Setup (30 minutes)**

**Step 1: Organize Your Project Files**

# Create a new directory for deployment

cd "C:\Users\onuba\Documents\ML Projects\AUTISM DIAGNOSIS PREDICTION IN TODDLERS\AUTISM-DIAGNOSIS-PREDICTION-IN-TODDLERS"

# Create necessary directories

mkdir fastapi-deployment

cd fastapi-deployment

**Copy these files to fastapi-deployment/:**

* [ ] main.py (FastAPI app)
* [ ] requirements.txt (FastAPI requirements)
* [ ] Dockerfile
* [ ] saved\_models/ folder (with all .joblib files)

Your structure should look like:

fastapi-deployment/

├── main.py

├── requirements.txt

├── Dockerfile

├── aws-task-definition.json

├── deploy.sh

└── saved\_models/

├── random\_forest\_pipeline.joblib

├── logistic\_regression\_pipeline.joblib

├── xgboost\_pipeline.joblib

├── naive\_bayes\_pipeline.joblib

└── feature\_info.joblib

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

* [ ] Go to https://aws.amazon.com/
* [ ] Click "Create an AWS Account"
* [ ] Follow the signup process (requires credit card)
* [ ] **Important**: Enable billing alerts immediately!

**Step 3: Install AWS CLI**

**Windows:**

# Download and install from:

# https://awscli.amazonaws.com/AWSCLIV2.msi

# Verify installation

aws --version

**Alternative (using Python):**

pip install awscli

aws --version

**Step 4: Install Docker Desktop**

* [ ] Download from: https://www.docker.com/products/docker-desktop
* [ ] Install and restart your computer
* [ ] Open Docker Desktop and ensure it's running
* [ ] Verify: docker --version

**Step 5: Configure AWS Credentials**

# Configure AWS CLI

aws configure

# You'll be prompted for:

# AWS Access Key ID: [Get from AWS Console]

# AWS Secret Access Key: [Get from AWS Console]

# Default region name: us-east-1

# Default output format: json

**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" → Create New Access Key
4. Download and save the credentials securely
5. Use these in aws configure

**✅ Phase 2: Local Testing (15 minutes)**

**Step 6: Test FastAPI Locally**

# Navigate to your project

cd fastapi-deployment

# Install dependencies

pip install -r requirements.txt

# Run the API

uvicorn main:app --reload --host 0.0.0.0 --port 8000

* [ ] Open browser: http://localhost:8000/docs
* [ ] Test /health endpoint
* [ ] Test /models endpoint
* [ ] Test /predict endpoint with sample data

**Sample prediction test:**

{

"A10": 1,

"Sex": "m",

"Ethnicity": "White-European",

"Jaundice": 0,

"Family\_mem\_with\_ASD": 0,

"Who completed the test": "family member",

"Age": 3

}

**Step 7: Test with Docker Locally**

# Build Docker image

docker build -t autism-api-test .

# Run container

docker run -d -p 8000:8000 --name autism-test autism-api-test

# Test it

curl http://localhost:8000/health

# Check logs

docker logs autism-test

# Stop and remove when done

docker stop autism-test

docker rm autism-test

**Checklist:**

* [ ] Docker build succeeds without errors
* [ ] Container runs successfully
* [ ] API responds at http://localhost:8000
* [ ] All endpoints work correctly

**✅ Phase 3: AWS Infrastructure Setup (30 minutes)**

**Step 8: Create VPC and Networking**

# Create infrastructure directory

mkdir infrastructure

cd infrastructure

**Create file: vpc-template.yaml** (copy from artifacts)

# Deploy VPC

aws cloudformation deploy `

--template-file vpc-template.yaml `

--stack-name autism-prediction-vpc `

--region us-east-1

# Verify creation

aws cloudformation describe-stacks `

--stack-name autism-prediction-vpc `

--region us-east-1

**Checklist:**

* [ ] VPC created successfully
* [ ] 2 public subnets created
* [ ] Internet Gateway attached
* [ ] Route tables configured

**Step 9: Create Application Load Balancer**

**Create file: alb-template.yaml** (copy from artifacts)

# Deploy ALB

aws cloudformation deploy `

--template-file alb-template.yaml `

--stack-name autism-prediction-alb `

--region us-east-1

# Get ALB DNS name (save this!)

aws cloudformation describe-stacks `

--stack-name autism-prediction-alb `

--query "Stacks[0].Outputs[?OutputKey=='LoadBalancerDNS'].OutputValue" `

--output text

**Checklist:**

* [ ] ALB created successfully
* [ ] Target group created
* [ ] Security groups configured
* [ ] DNS name obtained

**Step 10: Create IAM Roles**

# Create ECS Task Execution Role

aws iam create-role `

--role-name ecsTaskExecutionRole `

--assume-role-policy-document '{

\"Version\": \"2012-10-17\",

\"Statement\": [

{

\"Effect\": \"Allow\",

\"Principal\": {

\"Service\": \"ecs-tasks.amazonaws.com\"

},

\"Action\": \"sts:AssumeRole\"

}

]

}'

# Attach policy

aws iam attach-role-policy `

--role-name ecsTaskExecutionRole `

--policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy

# Create ECS Task Role

aws iam create-role `

--role-name ecsTaskRole `

--assume-role-policy-document '{

\"Version\": \"2012-10-17\",

\"Statement\": [

{

\"Effect\": \"Allow\",

\"Principal\": {

\"Service\": \"ecs-tasks.amazonaws.com\"

},

\"Action\": \"sts:AssumeRole\"

}

]

}'

**Checklist:**

* [ ] ecsTaskExecutionRole created
* [ ] ecsTaskRole created
* [ ] Policies attached

**✅ Phase 4: Deploy to AWS (45 minutes)**

**Step 11: Create ECR Repository**

# Get your AWS account ID

$AWS\_ACCOUNT\_ID = aws sts get-caller-identity --query Account --output text

echo "Your AWS Account ID: $AWS\_ACCOUNT\_ID"

# Create ECR repository

aws ecr create-repository `

--repository-name autism-prediction-api `

--region us-east-1

# Get ECR login

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin "$AWS\_ACCOUNT\_ID.dkr.ecr.us-east-1.amazonaws.com"

**Checklist:**

* [ ] ECR repository created
* [ ] Docker logged in to ECR

**Step 12: Build and Push Docker Image**

# Navigate back to project root

cd ..

# Build image

docker build -t autism-prediction-api:latest .

# Tag image for ECR

docker tag autism-prediction-api:latest "$AWS\_ACCOUNT\_ID.dkr.ecr.us-east-1.amazonaws.com/autism-prediction-api:latest"

# Push to ECR

docker push "$AWS\_ACCOUNT\_ID.dkr.ecr.us-east-1.amazonaws.com/autism-prediction-api:latest"

**This will take 5-10 minutes depending on your internet speed.**

**Checklist:**

* [ ] Image built successfully
* [ ] Image tagged correctly
* [ ] Image pushed to ECR

**Step 13: Create ECS Cluster**

# Create ECS cluster

aws ecs create-cluster `

--cluster-name autism-prediction-cluster `

--capacity-providers FARGATE `

--region us-east-1

# Create CloudWatch log group

aws logs create-log-group `

--log-group-name "/ecs/autism-prediction-api" `

--region us-east-1

**Checklist:**

* [ ] ECS cluster created
* [ ] CloudWatch log group created

**Step 14: Update Task Definition**

**Create file: aws-task-definition.json**

# Replace YOUR\_ACCOUNT\_ID with your actual account ID

(Get-Content aws-task-definition.json) -replace 'YOUR\_ACCOUNT\_ID', $AWS\_ACCOUNT\_ID | Set-Content aws-task-definition-updated.json

**Step 15: Register Task Definition**

# Register task definition

aws ecs register-task-definition `

--cli-input-json file://aws-task-definition-updated.json `

--region us-east-1

**Checklist:**

* [ ] Task definition registered successfully

**Step 16: Create Security Group for ECS Tasks**

# Get VPC ID

$VPC\_ID = aws ec2 describe-vpcs `

--filters "Name=tag:Name,Values=autism-prediction-vpc" `

--query "Vpcs[0].VpcId" `

--output text

# Create security group

$ECS\_SG\_ID = aws ec2 create-security-group `

--group-name autism-prediction-ecs-sg `

--description "Security group for ECS tasks" `

--vpc-id $VPC\_ID `

--query "GroupId" `

--output text

echo "ECS Security Group ID: $ECS\_SG\_ID"

# Get ALB security group

$ALB\_SG\_ID = aws ec2 describe-security-groups `

--filters "Name=group-name,Values=\*ALBSecurityGroup\*" `

--query "SecurityGroups[0].GroupId" `

--output text

# Allow traffic from ALB to ECS tasks

aws ec2 authorize-security-group-ingress `

--group-id $ECS\_SG\_ID `

--protocol tcp `

--port 8000 `

--source-group $ALB\_SG\_ID

**Checklist:**

* [ ] ECS security group created
* [ ] Ingress rule added

**Step 17: Get Subnet IDs**

# Get subnet IDs

$SUBNET1\_ID = aws ec2 describe-subnets `

--filters "Name=tag:Name,Values=autism-prediction-public-subnet-1" `

--query "Subnets[0].SubnetId" `

--output text

$SUBNET2\_ID = aws ec2 describe-subnets `

--filters "Name=tag:Name,Values=autism-prediction-public-subnet-2" `

--query "Subnets[0].SubnetId" `

--output text

echo "Subnet 1: $SUBNET1\_ID"

echo "Subnet 2: $SUBNET2\_ID"

**Step 18: Get Target Group ARN**

$TARGET\_GROUP\_ARN = aws cloudformation describe-stacks `

--stack-name autism-prediction-alb `

--query "Stacks[0].Outputs[?OutputKey=='TargetGroupArn'].OutputValue" `

--output text

echo "Target Group ARN: $TARGET\_GROUP\_ARN"

**Step 19: Create ECS Service**

# Create ECS service

aws ecs create-service `

--cluster autism-prediction-cluster `

--service-name autism-prediction-service `

--task-definition autism-prediction-api `

--desired-count 2 `

--launch-type FARGATE `

--platform-version LATEST `

--network-configuration "awsvpcConfiguration={subnets=[$SUBNET1\_ID,$SUBNET2\_ID],securityGroups=[$ECS\_SG\_ID],assignPublicIp=ENABLED}" `

--load-balancers "targetGroupArn=$TARGET\_GROUP\_ARN,containerName=autism-prediction-api,containerPort=8000" `

--health-check-grace-period-seconds 300 `

--region us-east-1

**This will take 3-5 minutes to start the tasks.**

**Checklist:**

* [ ] ECS service created
* [ ] Tasks starting
* [ ] Health checks passing

**✅ Phase 5: Testing and Verification (15 minutes)**

**Step 20: Wait for Service to be Healthy**

# Check service status

aws ecs describe-services `

--cluster autism-prediction-cluster `

--services autism-prediction-service `

--region us-east-1 `

--query "services[0].{Status:status,Running:runningCount,Desired:desiredCount}"

# Check task health (repeat until HEALTHY)

aws ecs list-tasks `

--cluster autism-prediction-cluster `

--service-name autism-prediction-service `

--region us-east-1

**Wait until runningCount equals desiredCount (2)**

**Step 21: Get Your API URL**

$ALB\_DNS = aws cloudformation describe-stacks `

--stack-name autism-prediction-alb `

--query "Stacks[0].Outputs[?OutputKey=='LoadBalancerDNS'].OutputValue" `

--output text

echo "🎉 Your API is live at: http://$ALB\_DNS"

echo "📖 API Documentation: http://$ALB\_DNS/docs"

**Step 22: Test Your Live API**

# Test health endpoint

curl "http://$ALB\_DNS/health"

# Test models endpoint

curl "http://$ALB\_DNS/models"

# Test prediction endpoint

curl -X POST "http://$ALB\_DNS/predict" `

-H "Content-Type: application/json" `

-d '{

"A10": 1,

"Sex": "m",

"Ethnicity": "White-European",

"Jaundice": 0,

"Family\_mem\_with\_ASD": 0,

"Who completed the test": "family member",

"Age": 3

}'

**Open in browser:**

http://YOUR\_ALB\_DNS/docs

**Checklist:**

* [ ] Health endpoint responds with "healthy"
* [ ] Models endpoint shows all 4 models
* [ ] Prediction endpoint returns results
* [ ] Interactive docs accessible

**✅ Phase 6: Monitoring and Maintenance (Ongoing)**

**Step 23: View Logs**

# View real-time logs

aws logs tail "/ecs/autism-prediction-api" --follow --region us-east-1

# View last 10 minutes

aws logs tail "/ecs/autism-prediction-api" --since 10m --region us-east-1

**Step 24: Set Up Cost Alerts**

# Create billing alarm (requires SNS topic)

aws sns create-topic --name billing-alerts --region us-east-1

# Subscribe to email notifications

aws sns subscribe `

--topic-arn arn:aws:sns:us-east-1:$AWS\_ACCOUNT\_ID:billing-alerts `

--protocol email `

--notification-endpoint your-email@example.com `

--region us-east-1

# Create billing alarm ($50 threshold)

aws cloudwatch put-metric-alarm `

--alarm-name "High-AWS-Bill" `

--alarm-description "Alert when AWS bill exceeds $50" `

--metric-name EstimatedCharges `

--namespace AWS/Billing `

--statistic Maximum `

--period 21600 `

--evaluation-periods 1 `

--threshold 50 `

--comparison-operator GreaterThanThreshold `

--alarm-actions arn:aws:sns:us-east-1:$AWS\_ACCOUNT\_ID:billing-alerts `

--region us-east-1

**Step 25: Configure Auto-Scaling**

# Register scalable target

aws application-autoscaling register-scalable-target `

--service-namespace ecs `

--scalable-dimension ecs:service:DesiredCount `

--resource-id service/autism-prediction-cluster/autism-prediction-service `

--min-capacity 2 `

--max-capacity 10 `

--region us-east-1

# Create scaling policy

aws application-autoscaling put-scaling-policy `

--policy-name autism-api-cpu-scaling `

--service-namespace ecs `

--scalable-dimension ecs:service:DesiredCount `

--resource-id service/autism-prediction-cluster/autism-prediction-service `

--policy-type TargetTrackingScaling `

--target-tracking-scaling-policy-configuration '{

\"TargetValue\": 70.0,

\"PredefinedMetricSpecification\": {

\"PredefinedMetricType\": \"ECSServiceAverageCPUUtilization\"

}

}' `

--region us-east-1

**📊 Success Metrics**

After deployment, you should see:

* ✅ 2 ECS tasks running
* ✅ ALB health checks passing
* ✅ API responding in < 2 seconds
* ✅ CloudWatch logs showing requests
* ✅ $0 charges for first month (Free Tier)

**🎉 Congratulations!**

Your autism prediction API is now **LIVE and accessible worldwide**!

**Your API Endpoints:**

* 🌐 **Base URL**: http://your-alb-dns.amazonaws.com
* 📖 **Documentation**: http://your-alb-dns.amazonaws.com/docs
* 🏥 **Health Check**: http://your-alb-dns.amazonaws.com/health
* 🤖 **Models**: http://your-alb-dns.amazonaws.com/models
* 🔮 **Predict**: http://your-alb-dns.amazonaws.com/predict

**Share Your API:**

Check out my autism prediction API!

Documentation: http://your-alb-dns.amazonaws.com/docs

**🆘 Troubleshooting Common Issues**

**Issue 1: Tasks not starting**

# Check task logs

aws ecs describe-tasks `

--cluster autism-prediction-cluster `

--tasks $(aws ecs list-tasks --cluster autism-prediction-cluster --query "taskArns[0]" --output text) `

--region us-east-1

**Issue 2: Health checks failing**

* Check CloudWatch logs for errors
* Verify security groups allow port 8000
* Ensure models folder is in the Docker image

**Issue 3: Cannot access API**

* Wait 5-10 minutes for ECS tasks to fully start
* Check ALB target group has healthy targets
* Verify security group rules

**Issue 4: High costs**

* Reduce desired count from 2 to 1
* Use t3.micro for development
* Delete unused resources

**📞 Next Steps**

1. **Week 1**: Monitor costs and usage
2. **Week 2**: Add custom domain (yourapi.com)
3. **Week 3**: Implement API authentication
4. **Month 2**: Build frontend application
5. **Month 3**: Launch publicly!

**Need help?** Check AWS CloudWatch logs or reach out!