| **Memo** | Classification: Internal Use Only |
| --- | --- |
| **To** | [Krishna](mailto:murali.krishna@winmore.app)[Hari Charan](mailto:hari.charan@winmore.app)[Manu Kodiyan](mailto:manu.kodiyan@winmore.app)[Manoj K](mailto:manoj.k@winmore.app)[Ushree Mazumdar](mailto:ushree.mazumdar@winmore.app)[Akash Bhandari](mailto:akash.bhandari@winmore.app)[Tushar Rasal](mailto:tushar.rasal@winmore.app)[Bhavesh Rote](mailto:bhavesh.rote@winmore.app)[Kushal Gowda](mailto:kushal.gowda@winmore.app) |
| **Title** | **AI dev Infra Ops** |
| **Author** | [Sriram Kalyanaraman](mailto:sriram.kalyanaraman@winmore.app) |

## **1. Executive Summary**

This memo provides step-by-step instructions to set up a backend development environment for the Winmore AI Platform. The environment will support our AI team's development activities across four core products: Rate AI, Contract AI, Allocation AI, and Gen BI.

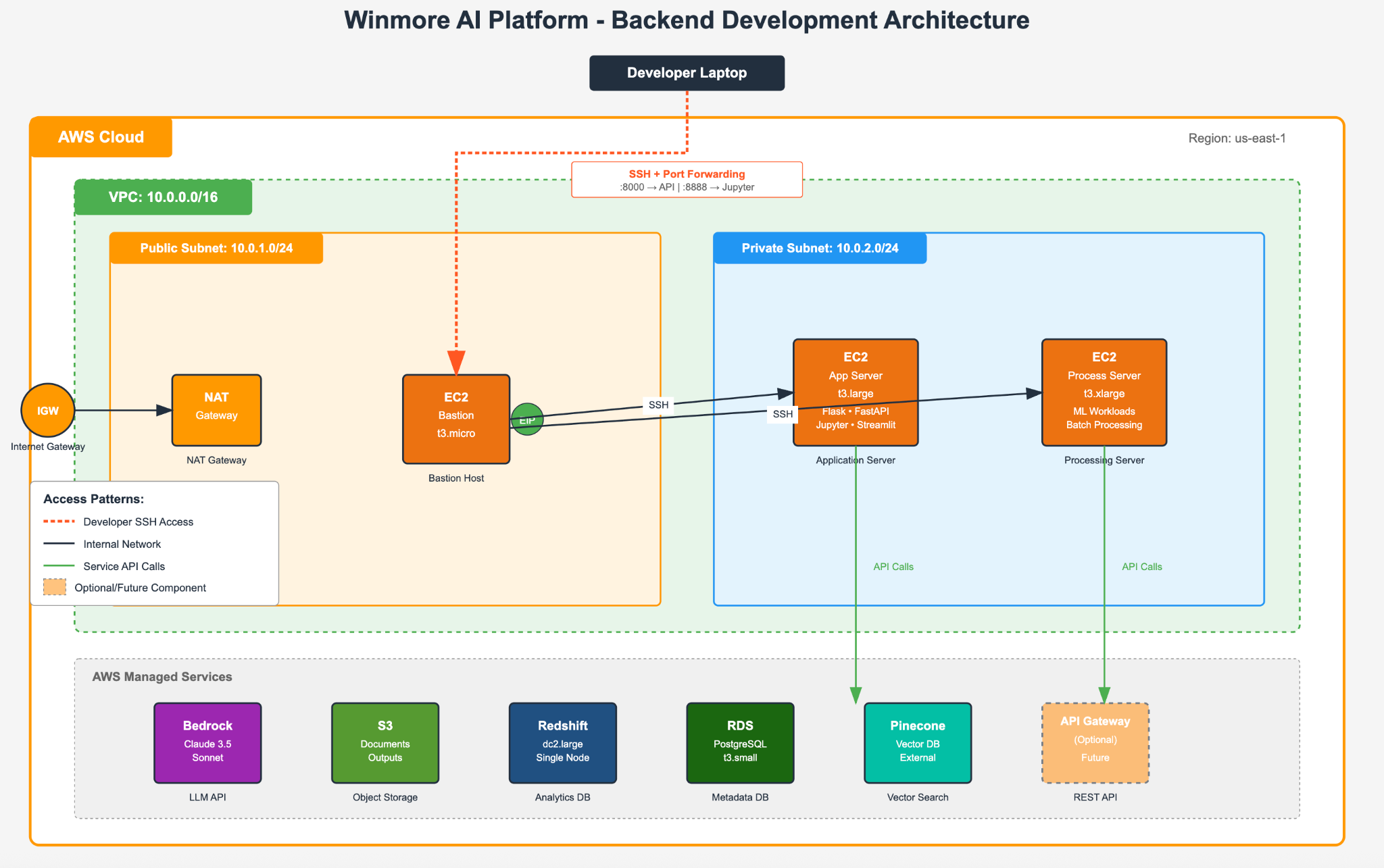
**Scope:** Backend development environment (no frontend/ALB required at this stage)

**Estimated Monthly Cost:** $700-1,000

**Timeline:** Please provide your estimate after reviewing the requirements

**Team Size:** 7 developers requiring access

## **2. Architecture Overview**



Developer Access Pattern:

1. Developer SSH → Bastion Host → App/Processing Servers

2. API Testing via SSH Tunnel: localhost:8000 → Bastion → App Server:8000

3. Jupyter Notebook: localhost:8888 → Bastion → App Server:8888

4. File Upload: SCP through Bastion → Processing Server

## **3. Pre-Setup Requirements**

### **3.1 AWS Account Prerequisites**

* AWS Account with appropriate permissions (AdministratorAccess or custom policy)
* AWS CLI installed and configured
* Region Selection: **us-east-1** (recommended for Bedrock availability)

**AWS Documentation:**

* [AWS CLI Installation Guide](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)
* [AWS CLI Configuration](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html)

## **4. Step-by-Step Setup Instructions**

### **Phase 1: Foundation Setup**

#### ***Step 1: VPC and Network Configuration***

**Objective:** Create isolated network for AI platform backend development

**Actions:**

1. Create VPC with CIDR: 10.0.0.0/16
2. Create Public Subnet: 10.0.1.0/24 (for NAT Gateway, Bastion)
3. Create Private Subnet: 10.0.2.0/24 (for EC2 instances)
4. Create Internet Gateway and attach to VPC
5. Create NAT Gateway in public subnet
6. Configure route tables appropriately

**AWS Console Path:** VPC → Your VPCs → Create VPC

**CLI Commands:**

# Create VPC

aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=winmore-ai-dev-vpc}]'

# Create Subnets

aws ec2 create-subnet --vpc-id vpc-xxxxx --cidr-block 10.0.1.0/24 --availability-zone us-east-1a

aws ec2 create-subnet --vpc-id vpc-xxxxx --cidr-block 10.0.2.0/24 --availability-zone us-east-1a

**AWS Documentation:**

* [VPC Creation Guide](https://docs.aws.amazon.com/vpc/latest/userguide/working-with-vpcs.html)
* [NAT Gateway Setup](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html)

#### ***Step 2: Security Groups Configuration***

**Create the following Security Groups:**

| **Security Group Name** | **Purpose** | **Inbound Rules** | **Outbound Rules** |
| --- | --- | --- | --- |
| winmore-bastion-sg | Bastion Host Access | SSH (22) from Office IPs | All traffic |
| winmore-backend-sg | Backend Servers | SSH (22) from Bastion  Custom TCP (8000-9000) from Bastion  All traffic from within VPC | All traffic |
| winmore-rds-sg | Database Access | PostgreSQL (5432) from backend-sg  Redshift (5439) from backend-sg | All traffic |

**AWS Documentation:** [Security Groups Guide](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html)

### **Phase 2: Compute Infrastructure**

#### ***Step 3: EC2 Instance Setup***

**3.1 Bastion Host**

* Instance Type: t3.micro
* AMI: Amazon Linux 2023
* Subnet: Public subnet
* Security Group: winmore-bastion-sg
* Elastic IP: Allocate and associate

**3.2 Application/API Server**

* Instance Type: t3.large
* AMI: Ubuntu 22.04 LTS
* Subnet: Private subnet
* Security Group: winmore-backend-sg
* Storage: 100GB EBS (gp3)

**User Data Script for App Server:**

#!/bin/bash

apt-get update

apt-get install -y python3.9 python3-pip git docker.io nginx

# Install Python packages

pip3 install boto3 pandas numpy scikit-learn flask fastapi uvicorn

pip3 install streamlit jupyter jupyterlab matplotlib seaborn

pip3 install python-multipart openpyxl PyPDF2

# Create app directories

mkdir -p /opt/winmore-ai/{uploads,outputs,logs}

# Configure Jupyter

jupyter notebook --generate-config

echo "c.NotebookApp.ip = '0.0.0.0'" >> ~/.jupyter/jupyter\_notebook\_config.py

echo "c.NotebookApp.port = 8888" >> ~/.jupyter/jupyter\_notebook\_config.py

# Start services

systemctl start docker

systemctl enable docker

**3.3 Processing Server**

* Instance Type: t3.xlarge
* AMI: Ubuntu 22.04 LTS
* Subnet: Private subnet
* Security Group: winmore-backend-sg
* Storage: 200GB EBS (gp3)

**AWS Documentation:**

* [EC2 Getting Started](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html)
* [EC2 User Data Guide](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html)

### **Phase 3: AWS Bedrock Configuration**

#### ***Step 4: Enable AWS Bedrock***

**Important:** Bedrock access requires manual approval from AWS. This can take 1-2 business days.

**Steps:**

1. Navigate to AWS Bedrock Console
2. Click "Model access" in left navigation
3. Request access for: **Claude 3.5 Sonnet**
4. Fill out use case form (mention: document processing, data extraction)
5. Wait for approval email from AWS

**IAM Policy for Bedrock Access:**

{

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

"Statement": [

{

"Effect": "Allow",

"Action": [

"bedrock:InvokeModel",

"bedrock:InvokeModelWithResponseStream"

],

"Resource": "arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-3-5-sonnet-20240620-v1:0"

}

]

}

**Test Bedrock Connection:**

# Test script to verify Bedrock access

import boto3

import json

bedrock = boto3.client('bedrock-runtime', region\_name='us-east-1')

response = bedrock.invoke\_model(

modelId='anthropic.claude-3-5-sonnet-20240620-v1:0',

body=json.dumps({

"prompt": "Test connection: Reply with 'Connection successful'",

"max\_tokens": 50,

"temperature": 0

})

)

print(json.loads(response['body'].read()))

**AWS Documentation:**

* [Bedrock Model Access Guide](https://docs.aws.amazon.com/bedrock/latest/userguide/model-access.html)
* [Bedrock IAM Permissions](https://docs.aws.amazon.com/bedrock/latest/userguide/security-iam.html)

### **Phase 4: Data Services Setup**

#### ***Step 5: Amazon Redshift Setup***

**Configuration:**

* Cluster Type: Single Node (dev environment)
* Node Type: dc2.large
* Database Name: winmore\_analytics
* Master Username: admin
* VPC: winmore-ai-dev-vpc
* Subnet Group: Create new in private subnet
* Publicly Accessible: No

**Post-Setup SQL:**

-- Create schemas for each AI component

CREATE SCHEMA rate\_ai;

CREATE SCHEMA contract\_ai;

CREATE SCHEMA allocation\_ai;

CREATE SCHEMA gen\_bi;

-- Create dev user

CREATE USER dev\_team PASSWORD 'SecurePassword123!';

GRANT ALL ON SCHEMA rate\_ai, contract\_ai, allocation\_ai, gen\_bi TO dev\_team;

**AWS Documentation:** [Redshift Getting Started](https://docs.aws.amazon.com/redshift/latest/gsg/rs-gsg-launch-sample-cluster.html)

#### ***Step 6: RDS PostgreSQL Setup***

**Configuration:**

* Engine: PostgreSQL 15.x
* Instance Class: db.t3.small
* Storage: 50GB gp3
* Database Name: winmore\_metadata
* VPC: winmore-ai-dev-vpc
* Security Group: winmore-rds-sg
* Publicly Accessible: No

**AWS Documentation:** [RDS PostgreSQL Setup](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.PostgreSQL.html)

#### ***Step 7: S3 Bucket Configuration***

**Create the following buckets:**

| **Bucket Name** | **Purpose** | **Configuration** |
| --- | --- | --- |
| winmore-ai-dev-documents | Input document storage | Versioning: Enabled  Encryption: SSE-S3 |
| winmore-ai-dev-outputs | Processed results | Lifecycle: 30-day transition to IA |
| winmore-ai-dev-models | Model artifacts | Access: Private only |

**Bucket Policy Example:**

{

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

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::YOUR-ACCOUNT-ID:role/winmore-ai-dev-role"

},

"Action": [

"s3:GetObject",

"s3:PutObject",

"s3:DeleteObject"

],

"Resource": "arn:aws:s3:::winmore-ai-dev-documents/\*"

}

]

}

**AWS Documentation:** [S3 Bucket Creation Guide](https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket.html)

### **Phase 5: External Services**

#### ***Step 8: Pinecone Vector Database Setup***

**Actions Required:**

1. Create Pinecone account at [pinecone.io](https://www.pinecone.io/)
2. Create new project: "winmore-ai-dev"
3. Create index with following specs:
   * Index name: winmore-dev-index
   * Dimensions: 1536 (for OpenAI embeddings compatibility)
   * Metric: cosine
   * Pod type: s1.x1 (starter)
   * Replicas: 1
4. Generate API key and store in AWS Secrets Manager

**Store API Key in Secrets Manager:**

aws secretsmanager create-secret \

--name winmore/pinecone/api-key \

--description "Pinecone API key for dev environment" \

--secret-string '{"api\_key":"YOUR-PINECONE-API-KEY","environment":"us-east-1"}'

**Documentation:** [Pinecone Quick Start Guide](https://docs.pinecone.io/docs/quickstart)

#### ***Step 9: API Gateway Setup (Optional - For Future Frontend)***

**Note:** API Gateway is optional for backend development. It can be added later when frontend is ready. For now, developers will use SSH tunneling to access APIs.

**If setting up now for future use:**

* Type: REST API
* Name: winmore-ai-dev-api
* Endpoint Type: Regional

**Create placeholder resources:**

/rate-ai

POST /process

GET /status/{job-id}

/contract-ai

POST /extract

GET /results/{document-id}

/allocation-ai

POST /parse

GET /booking/{booking-id}

/gen-bi

POST /query

GET /visualization/{query-id}

**AWS Documentation:** [API Gateway Developer Guide](https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html)

### **Phase 6: IAM and Access Management**

#### ***Step 10: IAM Roles and Users Setup***

**10.1 Create IAM Role for EC2 Instances:**

Role Name: winmore-ai-dev-ec2-role

Attached Policies:

- AmazonS3FullAccess (scope down in production)

- AmazonBedrockFullAccess

- SecretsManagerReadWrite

- CloudWatchLogsFullAccess

**10.2 Create IAM Users for Development Team:**

| **User Name** | **Groups** | **Access Type** |
| --- | --- | --- |
| sriram-dev | winmore-ai-developers | Programmatic & Console |
| akash-dev | winmore-ai-developers | Programmatic & Console |
| hari-dev | winmore-ai-developers | Programmatic & Console |
| ushree-dev | winmore-ai-developers | Programmatic & Console |
| tushar-dev | winmore-ai-developers | Programmatic & Console |
| bhavesh-dev | winmore-ai-developers | Programmatic & Console |
| kushal-dev | winmore-ai-developers | Programmatic & Console |

**Group Policy for winmore-ai-developers:**

{

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

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:DescribeInstances",

"s3:\*",

"bedrock:\*",

"secretsmanager:GetSecretValue",

"redshift:DescribeClusters",

"rds:DescribeDBInstances"

],

"Resource": "\*",

"Condition": {

"StringEquals": {

"aws:RequestedRegion": "us-east-1"

}

}

}

]

}

**AWS Documentation:** [IAM Getting Started](https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started.html)

## **5. Developer Testing Workflows**

### **Backend API Testing Options**

**Option 1: SSH Tunneling for API Access**

# Connect to bastion and forward ports

ssh -i winmore-dev.pem -L 8000:10.0.2.10:8000 -L 8888:10.0.2.10:8888 ec2-user@bastion-ip

# Now access from local machine:

# API: http://localhost:8000

# Jupyter: http://localhost:8888

**Option 2: Direct SSH Development**

# SSH to bastion

ssh -i winmore-dev.pem ec2-user@bastion-ip

# Then SSH to app server

ssh -i winmore-dev.pem ubuntu@10.0.2.10

# Run scripts directly

python3 rate\_ai\_processor.py --file test\_rate.xlsx

**Option 3: File Upload and Testing**

# Upload file through bastion

scp -i winmore-dev.pem -o ProxyJump=ec2-user@bastion-ip \

test\_file.xlsx ubuntu@10.0.2.10:/opt/winmore-ai/uploads/

# Process via API

curl -X POST http://localhost:8000/rate-ai/process \

-F "file=@test\_rate.xlsx" \

-H "Content-Type: multipart/form-data"

**Sample Backend API Code:**

# app.py - Simple Flask API for testing

from flask import Flask, request, jsonify

import boto3

import pandas as pd

app = Flask(\_\_name\_\_)

bedrock = boto3.client('bedrock-runtime', region\_name='us-east-1')

@app.route('/rate-ai/process', methods=['POST'])

def process\_rate():

file = request.files['file']

df = pd.read\_excel(file)

# Call Bedrock

prompt = f"Extract rates from: {df.head().to\_string()}"

response = bedrock.invoke\_model(

modelId='anthropic.claude-3-5-sonnet-20240620-v1:0',

body=json.dumps({"prompt": prompt, "max\_tokens": 1000})

)

return jsonify({"status": "processed", "result": response})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8000)

## **6. Post-Setup Validation**

### **Validation Checklist**

**Network Connectivity:**

* ☐ SSH to Bastion Host successful
* ☐ SSH from Bastion to private EC2 instances successful
* ☐ EC2 instances can reach internet via NAT Gateway
* ☐ Port forwarding working (8000, 8888)

**Service Access:**

* ☐ Bedrock API accessible from EC2 instances
* ☐ S3 buckets accessible
* ☐ Redshift cluster connectable
* ☐ RDS instance connectable
* ☐ Pinecone API returns successful health check

**Development Tools:**

* ☐ Python environment configured
* ☐ Jupyter notebook accessible via tunnel
* ☐ Flask/FastAPI server running
* ☐ boto3 can call Bedrock successfully

**Validation Script:**

#!/bin/bash

# validate\_backend\_setup.sh - Run on app server

echo "=== Winmore AI Backend Dev Environment Validation ==="

# Test Bedrock

echo -n "Testing Bedrock access... "

python3 -c "import boto3; print('✓' if boto3.client('bedrock-runtime', region\_name='us-east-1').list\_foundation\_models() else '✗')"

# Test S3

echo -n "Testing S3 access... "

aws s3 ls s3://winmore-ai-dev-documents &> /dev/null

if [ $? -eq 0 ]; then echo "✓ SUCCESS"; else echo "✗ FAILED"; fi

# Test Python packages

echo -n "Testing Python environment... "

python3 -c "import flask, fastapi, boto3, pandas, streamlit; print('✓ SUCCESS')" 2>/dev/null || echo "✗ FAILED"

# Test API server

echo -n "Testing Flask API... "

python3 -c "from flask import Flask; app = Flask(\_\_name\_\_); print('✓ SUCCESS')" || echo "✗ FAILED"

echo "=== Validation Complete ==="

## **7. Handover Information**

### **Deliverables for Development Team**

Upon completion, please provide the following to the AI development team:

| **Item** | **Details** | **Delivery Method** |
| --- | --- | --- |
| Bastion Host IP | Elastic IP address | Secure email/Slack |
| SSH Keys | .pem file for EC2 access | Encrypted file transfer |
| IAM Credentials | Access keys for each developer | Individual secure delivery |
| Internal IPs | App server: 10.0.2.x  Process server: 10.0.2.y | Team documentation |
| Database Endpoints | Redshift & RDS connection strings | AWS Secrets Manager |
| SSH Tunnel Commands | Pre-configured commands for each service | Team README |

**Sample README for Developers:**

# Winmore AI Dev Environment Access

## Quick Start

1. Add SSH key to your SSH agent: `ssh-add winmore-dev.pem`

2. Connect with port forwarding:

`ssh -L 8000:10.0.2.10:8000 -L 8888:10.0.2.10:8888 ec2-user@[BASTION-IP]`

3. Access services:

- API: http://localhost:8000

- Jupyter: http://localhost:8888

- Streamlit: http://localhost:8501

## Direct Server Access

- App Server: `ssh -J ec2-user@[BASTION-IP] ubuntu@10.0.2.10`

- Process Server: `ssh -J ec2-user@[BASTION-IP] ubuntu@10.0.2.11`

## File Upload

`scp -o ProxyJump=ec2-user@[BASTION-IP] myfile.xlsx ubuntu@10.0.2.10:/opt/winmore-ai/uploads/`

## **8. Cost Optimization Notes**

### **Dev Environment Cost Controls**

* **Auto-shutdown:** Configure EC2 instances to stop after business hours (save ~50%)
* **Redshift pause:** Pause cluster when not in use (weekends)
* **S3 lifecycle:** Delete test outputs after 7 days
* **Right-sizing:** Start with t3.large, upgrade only if needed

## **9. Future Considerations (Phase 2)**

The following items are noted for future implementation:

* Application Load Balancer (when frontend is ready)
* API Gateway with custom domain
* CloudWatch detailed monitoring and alarms
* AWS Backup for automated backups
* VPN connection for office network
* CI/CD pipeline with CodePipeline

**Timeline:** Please review this document and provide your estimated timeline for completion. We are ready to support with any clarifications needed.