Apache Pinot Real Time Analytics Server with Docker by Optick

Customer Instructions

This document explains how to access, validate, and use your Apache Pinot real time analytics server after launching the AMI from AWS Marketplace.

1. Product Overview

Apache Pinot Real Time Analytics Server with Docker by Optick is a self managed real time analytics starter server for low latency dashboards, streaming event analytics, customer facing analytics, and AI era data applications. The AMI includes a working Kafka to Pinot demo dataset so you can run sample queries shortly after first launch.

Item	Value
Operating system	Ubuntu 24.04 LTS
Default OS user	ubuntu
Application path	/opt/optick-pinot
Main landing page	http://YOUR_PUBLIC_IP/
Apache Pinot Controller	http://YOUR_PUBLIC_IP:9000
Demo table	optick_events
Deployment method	Docker Compose with systemd autostart

2. First Access

Connect to your instance using your Amazon private key and the **ubuntu** user. Replace YOUR_KEY.pem and YOUR_PUBLIC_IP with your own EC2 key file and instance public IP address.

```
ssh -i YOUR_KEY.pem ubuntu@YOUR_PUBLIC_IP
```

After login, read the first login notes:

```
cat /home/ubuntu/FIRST_LOGIN.txt
```

Show the main product URLs:

```
optick-pinot-url
```

3. Required Security Group Ports

Port	Purpose	Recommendation
TCP 22	SSH access	Restrict to your trusted IP address when possible
TCP 80	Optick landing page	Open from your browser
TCP 9000	Apache Pinot Controller	Open from your browser or trusted IP range

For production style deployments, restrict public access to trusted IP ranges or place access behind your own network controls.

4. Web Interfaces

Open the Optick landing page:

http://YOUR_PUBLIC_IP/

Open the Apache Pinot Controller:

http://YOUR_PUBLIC_IP:9000

After first launch or reboot, allow about five minutes for Pinot, Kafka, ZooKeeper, and the demo initialization service to finish starting before running demo queries.

5. Included Services

Service	Purpose
ZooKeeper	Cluster coordination for Pinot
Apache Kafka	Streaming event broker for the demo pipeline
Pinot Controller	Cluster management, schemas, tables, tenants, and UI
Pinot Broker	SQL query routing
Pinot Server	Segment storage and query execution
Pinot Minion	Background Pinot maintenance jobs
Nginx	Public landing page on port 80

6. Helper Commands

The AMI includes helper commands for common administration tasks. Commands that you enter are highlighted in red throughout this document.

Command	Purpose
optick-pinot-url	Show public landing page and Pinot Controller URLs
optick-pinot-status	Show service URLs, Docker status, and health checks
optick-pinot-logs	Show recent Docker Compose logs
optick-pinot-load-demo	Load additional sample events into Kafka
optick-pinot-query-demo	Run sample Pinot SQL queries from SSH

Check status:

optick-pinot-status

View recent logs:

optick-pinot-logs

Load more demo events:

optick-pinot-load-demo

Run demo queries:

optick-pinot-query-demo

7. Working Demo Dataset

The demo table is named optick_events. It simulates application and business event traffic across regions, applications, device types, routes, latency, errors, and revenue.

Run this query in the Apache Pinot Query Console:

```
SELECT event_type, count(*) AS events
FROM optick_events
GROUP BY event_type
ORDER BY events DESC
LIMIT 10
```

Compact count query from SSH:

```
curl -sS -X POST "http://localhost:8099/query/sql" \
  -H "Content-Type: application/json" \
  -d '{"sql":"SELECT count(*) AS event_count FROM optick_events"}' | jq .
```

8. Health Checks

Controller health:

```
curl http://localhost:9000/health
```

Broker health:

```
curl http://localhost:8099/health
```

Server health:

```
curl http://localhost:8097/health
```

Docker Compose status:

```
cd /opt/optick-pinot
sudo docker compose ps
```

9. First Launch Automation

The AMI includes first launch automation so every new EC2 instance regenerates public IP based instructions and starts from clean Pinot, Kafka, and ZooKeeper runtime state. This prevents old build instance segment metadata from being reused on customer launches.

Service	Purpose
optick-pinot-firstboot.service	Regenerates FIRST_LOGIN.txt and landing page with the current public IP
optick-pinot-runtime-reset.service	Resets nonportable runtime state on first AMI launch
optick-pinot-stack.service	Starts the Docker Compose Pinot stack
optick-pinot-demo-init.service	Recreates the Kafka topic, Pinot schema, realtime table, and demo events

Check service status:

```
systemctl status optick-pinot-firstboot.service --no-pager
systemctl status optick-pinot-runtime-reset.service --no-pager
systemctl status optick-pinot-stack.service --no-pager
systemctl status optick-pinot-demo-init.service --no-pager
```

10. Troubleshooting

Issue	Suggested action
Landing page opens but Pinot query fails right after launch	Wait about five minutes, then rerun optick-pinot-query-demo. Pinot needs time to rebuild routing and initialize the demo.
Browser cannot open port 9000	Confirm the EC2 security group allows TCP 9000 from your IP address.
No demo rows returned	Run optick-pinot-load-demo, wait one minute, then run optick-pinot-query-demo.
Need logs	Run optick-pinot-logs or sudo docker compose logs --tail=200 from /opt/optick-pinot.
Need to restart stack	Run sudo systemctl restart optick-pinot-stack.service, wait five minutes, then check status.

Restart stack:

```
cd /opt/optick-pinot
sudo systemctl restart optick-pinot-stack.service
sleep 300
optick-pinot-status
```

11. Recommended Instance Sizes

Use case	Recommended size
Development and evaluation	t3.large or m6i.large
Heavier demos and ingestion tests	m6i.xlarge or m7i.xlarge
Production style evaluation	m6i.2xlarge or larger, depending on ingestion volume and query concurrency

This AMI is a self managed starter server. For large scale production Pinot, review storage layout, cluster sizing, data retention, access controls, backups, and ingestion architecture before going live.