CUSTOMER USAGE INSTRUCTIONS

SigNoz Observability & AI Command Center

Version 2.0 | Ubuntu 26.04 LTS | SigNoz v0.150.0 | AWS Marketplace AMI

A self-hosted observability platform for **application performance monitoring (APM), distributed traces, metrics, logs, dashboards and alerting**. The OpenTelemetry collector, SigNoz application, database and HTTPS gateway run as a managed Docker stack on one EC2 instance.

01 Launch the EC2 instance

1. Launch the product from AWS Marketplace on an **x86_64** EC2 instance. For a comfortable start, use **8 GB+ RAM** and **40 GB+ EBS** storage.
2. Configure the EC2 Security Group: allow **TCP 22** from administrator IPs and **TCP 443** from intended dashboard users. Do not publicly expose database, SigNoz backend or telemetry ports.
3. Wait for the EC2 instance to be running and pass its status checks. SSH in as the **ubuntu** user with your AWS EC2 key pair (PuTTY with a **.ppk** key also works).

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

4. Let the one-time setup finish. SigNoz containers, the first administrator account and the instance HTTPS certificate are created automatically.

```
sudo cloud-init status --wait
```

02 Get credentials and open the dashboard

Read the new, instance-specific administrator login using either command:

```
sudo cc-signoz credentials
cat ~/FIRST_LOGIN.txt
```

Open **https://YOUR_PUBLIC_IP/**, then sign in with the generated email and password. The administrator is **pre-created**: **do not use the old “create an admin user” signup instructions.**

Code Creator welcome and setup page: **https://YOUR_PUBLIC_IP/codecreator/**

> **HTTPS certificate:** a unique self-signed certificate is generated per instance, so browsers show a certificate trust warning. Verify the EC2 address belongs to you before continuing. For routine production access, install a trusted certificate for your own DNS name.

CONNECT APPLICATIONS & OPERATE

Send telemetry. Manage the stack.

The dashboard starts automatically, but observability data appears only after your applications send it.

03 OpenTelemetry (OTLP) ingestion

The included collector supports **OTLP/gRPC (TCP 4317)** and **OTLP/HTTP (TCP 4318)**. Both bind to **localhost by default**; remote applications cannot reach them until private ingestion is deliberately enabled.

1. In your AWS Security Group, allow TCP 4317/4318 only from the **trusted application servers' Security Group** inside the VPC.
2. Enable OTLP on the instance's **private IPv4** address:

```
sudo env ACK_PRIVATE_SG=YES cc-signoz otlp-private on
```

3. Configure instrumented apps to send data to the SigNoz instance's **private IP**:

```
OTLP/gRPC: YOUR_SIGNOZ_PRIVATE_IP:4317
OTLP/HTTP: http://YOUR_SIGNOZ_PRIVATE_IP:4318
```

Use SigNoz in-app instrumentation guidance. To return to localhost-only ingestion:

```
sudo cc-signoz otlp-private off
```

> **Important:** OTLP has **no native TLS or authentication**. Keep ports 4317/4318 private; **never** allow 0.0.0.0/0. Use a secured gateway for internet transport.

04 Code Creator command center

COMMAND	PURPOSE
`sudo cc-signoz help`	List available administration commands
`sudo cc-signoz status`	Check services, container status and HTTPS readiness
`sudo cc-signoz credentials`	Show the generated dashboard login
`sudo cc-signoz logs`	Show recent application and collector logs
`sudo cc-signoz restart`	Restart the application stack and HTTPS gateway
`sudo cc-signoz acceptance`	Run the quick built-in health checks
`sudo cc-signoz rotate-password`	Generate a new root administrator password
`sudo cc-signoz cve`	Show targeted host Brotli and container inventory

Customer safety: Never run `cc-signoz seal-ami` on a customer server. It is a destructive, builder-only packaging command.

SECURITY, STORAGE & SUPPORT

Keep your observability server safe.

Use your AWS Security Group and standard EC2 operations to control access, data retention and backups.

05 Security and data protection

AREA	CUSTOMER GUIDANCE
`Browser access`	Expose TCP 443 only to intended users; TCP 22 only to trusted administrators. HTTPS uses a self-signed certificate until you replace it.
`Telemetry ingestion`	Ports 4317/4318 are localhost-only by default. If enabled, allow only trusted private application sources; transport is unencrypted by default.
`Internal services`	Do not expose SigNoz port 8080, PostgreSQL, ClickHouse or Keeper ports directly to the internet.
`Login and SSH keys`	App credentials are generated at first boot and displayed via cc-signoz credentials. EC2 authorized_keys holds SSH access keys, not the SigNoz dashboard password.
`Data and backups`	Encrypt attached EBS volumes, use EBS snapshots, monitor free disk space, and review retention settings in the SigNoz interface.

The AMI does not independently encrypt application data stored inside the instance. **EBS encryption at rest is controlled by AWS volume configuration.** Browser HTTPS encrypts traffic but is not publicly trusted until you install a CA-issued certificate.

06 Troubleshooting and maintenance

If the dashboard is unavailable immediately after launch, allow time for database initialization, then check:

```
sudo cloud-init status --wait
sudo cc-signoz status
sudo cc-signoz logs
```

If a restart is necessary, use **sudo cc-signoz restart**. A temporary HTTP 502 during startup can indicate that the backend has not become ready. Check logs if it persists.

For Ubuntu package maintenance, use `sudo apt update && sudo apt upgrade -y` during an appropriate maintenance window. Do not manually replace the Foundry-generated SigNoz container images without a tested upgrade procedure.

07 Optional: use a stable public IP

An EC2 auto-assigned public IPv4 address may change after **stop/start**. To keep a stable endpoint, allocate and associate an **Elastic IP** with your instance in the EC2 Console, then point your DNS name to it. Check AWS pricing and update the trusted HTTPS certificate to match your domain.

AWS documentation: Elastic IP addresses

Support: info@codecreator.com | SigNoz on AWS Marketplace