12

CHAPTER

運用と トラブルシューティング

公開して終わりじゃない — 動かし続けるための知識

本書の最終章 — おつかれさま

AGENDA

この章で学ぶこと

01 ログの確認
Gunicorn / Nginx / system ログを読む

02 よくあるエラーと対処
502 / 500 / メモリ不足

03 パフォーマンス監視
htop / df / iftop で見る

04 バックアップ戦略
手動 + cron で自動化

05 セキュリティ強化
SSH ポート変更・ Fail2ban ・ UFW

06 実験終了後はサーバー停止
課金事故を防ぐ最重要事項

LOGS

ログ確認 — トラブルシュートの第一歩

Gunicorn

Django アプリ

```
journalctl -u
adventure-game -f
```

Python エラーや
リクエスト処理を確認

Nginx

アクセス / エラー

```
tail -f /var/log/
nginx/error.log
```

404 / 502 / SSL 問題
を確認

System

全体ログ

```
tail -f
/var/log/syslog
```

メモリ不足や
ハードウェア問題

journalctl の主要オプション

`-u <name>`
サービス指定

`-f`
リアルタイム追跡

`-n 100`
最新 100 行

`--since`
時刻指定

PART 1

よくあるエラー対処

502 / 500 / メモリ不足を捌く

Bad Gateway — Gunicorn が応答していない

ERROR CODE

502 Bad Gateway

Nginx が Gunicorn (= 中継先) に
接続できなかった

考えられる原因

- Gunicorn が起動していない
- ソケットファイル接続失敗
- Gunicorn がクラッシュ

→ ログに ModuleNotFoundError が出ることが多い

対処の 3 ステップ

01 状態確認

```
$ sudo systemctl status
  adventure-game
```

02 再起動

```
$ sudo systemctl restart
  adventure-game
```

03 ログ確認

```
$ sudo journalctl -u
  adventure-game -n 50
```

500

Internal Server Error — Django 側のエラー

ERROR CODE

500 Internal Error

Django アプリ側で例外が発生

考えられる原因

- settings.py の誤り
- DB の破損 / 権限不足
- Python のバグ（NameError 等）
- 環境変数（.env）の誤り

→ Traceback で具体的な行と原因がわかる

デバッグ 3 ステップ

01 Nginx エラーログ

```
$ sudo tail -f \
  /var/log/nginx/error.log
```

02 Gunicorn ログ

```
$ sudo journalctl -u \
  adventure-game -n 100
```

03 DEBUG=True
(SSH トンネル経由)

```
$ ssh -L 8001:localhost:8001 \
  ubuntu@$IP
# 別ターミナル: runserver
```

⚠ 本番で DEBUG=True にしっぱなしは厳禁

MEMORY

メモリ不足 — t2.micro の 1GB 問題

症状の確認

free -h で見る

```
               total used  free avail
Mem:           976Mi 850Mi  50Mi  95Mi
Swap:             0B    0B    0B   0B
```

判断基準

100MB 以下	注意
50MB 以下	対処必須
Swap 0B	緩衝なし

Swap ファイルで緩衝

メモリ不足時にディスクを仮想メモリ化

```
# 1GB の Swap ファイル作成
$ sudo fallocate -l 1G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile

# 永続化（再起動後も有効）
$ echo '/swapfile swap swap defaults 0 0' \
    | sudo tee -a /etc/fstab

# 確認
$ free -h
```

推奨：RAM の 1〜2 倍のサイズ

PART 2

監視 & バックアップ

サーバーの健康診断 + データ保全

パフォーマンス監視 — 3 つの基本ツール

htop

CPU & メモリ

```
$ sudo apt install
  htop
$ htop
```

プロセス一覧
+ リアルタイム

df -h

ディスク容量

```
$ df -h
```

Filesystem
使用 / 全体 / Use%

iftop

ネットワーク

```
$ sudo apt install iftop
$ sudo iftop -i eth0
```

送受信トラフィック
通信先 IP

BACKUP

DB バックアップ + cron で自動化

MANUAL

手動バックアップ + 復元

```
# バックアップディレクトリ作成
$ mkdir -p /home/ubuntu/backups

# 日付付きでコピー
$ cp /home/ubuntu/adventure-game/db.sqlite3 \
    /home/ubuntu/backups/\
    db_$(date +%Y%m%d_%H%M%S).sqlite3

# 復元
$ ls -lht /home/ubuntu/backups/
$ cp /home/ubuntu/backups/db_xxx.sqlite3 \
    /home/ubuntu/adventure-game/db.sqlite3
$ sudo systemctl restart adventure-game
```

AUTO (cron)

毎日 3 時にバックアップ

```
$ crontab -e

# 毎日 3:00
0 3 * * * cp ... \
  db_$(date +\%Y\%m\%d).db

# 30 日前を削除
0 4 * * * find ... \
  -mtime +30 -delete
```

cron 構文

分 時 日 月 曜

cron では % は \% でエスケープ必須

PART 3

セキュリティ強化

SSH ポート変更 + Fail2ban + UFW で守る

SSH ポート変更 (22 → 54321) 5 ステップ

01

sshd_config

Port 54321 に変更

```
/etc/ssh/sshd_config
sudo systemctl
       restart sshd
```

02

SG に新ポート追加

AWS CLI で 54321 開放

```
authorize-security-
group-ingress \
      --port 54321
```

03

新ポートで接続テスト

既存セッション維持 別ターミナルで

```
ssh -i ... \
  -p 54321 \
        ubuntu@$IP
```

04

22 番を閉じる

新ポート確認後のみ

```
revoke-security-
group-ingress \
       --port 22
```

05

Fail2ban / UFW 更新

新ポート番号で設定し直す

```
port = 54321
  ufw allow 54321/tcp
```

⚠ STEP 04 (22 番閉じる) は STEP 03 で新ポート接続を必ず確認してから！

ログイン失敗を自動ブロック

/etc/fail2ban/jail.local

```
[DEFAULT]
bantime  = 3600      # 1 時間ブロック
findtime = 600       # 10 分以内に
maxretry = 5         # 5 回失敗で

[sshd]
enabled = true

port = 54321         #  変更後ポート
```

効果

10 分で 5 回失敗

↓

その IP を
1 時間自動ブロック

起動コマンド

```
$ sudo systemctl enable \
    fail2ban
$ sudo systemctl start fail2ban
$ sudo fail2ban-client \
    status sshd
```

Uncomplicated Firewall — シンプルな FW

PRINCIPLE

デフォルトすべて拒否 + 必要なポートだけ許可

セットアップコマンド

```
$ sudo apt install ufw -y

# デフォルトポリシー
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing

# 許可ポート
$ sudo ufw allow 54321/tcp    # SSH
$ sudo ufw allow 80/tcp       # HTTP
$ sudo ufw allow 443/tcp      # HTTPS

# 有効化
$ sudo ufw enable
```

開放するポート

54321	SSH	管理用
80	HTTP	リダイレクト
443	HTTPS	本番アクセス

SCALING

スケーリングの考え方 — 縦と横

VERTICAL

垂直（スケールアップ）

インスタンスを大きく

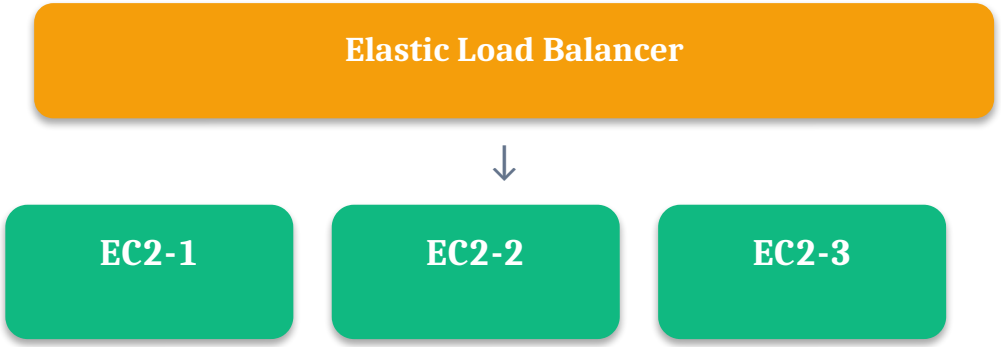
`t2.micro`	1 vCPU / 1GB	$9/ 月
`t2.small`	1 vCPU / 2GB	$19/ 月
`t2.medium`	2 vCPU / 4GB	$37/ 月
`t3.small`	2 vCPU / 2GB	$16/ 月

→ stop-instances → modify → start で切替

HORIZONTAL

水平（スケールアウト）

インスタンスを増やす



→ Auto Scaling で負荷に応じ自動増減

（本書の範囲外。将来的に検討）

STOP / TERMINATE

実験終わったらサーバー停止 — 課金事故防止

WARNING

起動したまま放置 → 不正アクセスで **Gemini API** に課金集中の恐れ

STOP 停止

後でまた使う

```
$ aws ec2 stop-instances \
  --instance-ids $ID
```

EC2 料金	停止 (¥0)
EBS 料金	継続課金
再開	可能
データ	残る

TERMINATE 削除

もう使わない！

```
$ aws ec2 terminate-instances \
  --instance-ids $ID
```

EC2 料金	完全停止
EBS 料金	完全停止
再開	不可（新規作成）
データ	完全に消える

WHAT YOU LEARNED

本書を通じて身につけたこと

全 12 章の旅 — おつかれさまでした

01 LLM 活用

Cursor / プロンプト設計
エラー解決

02 Django 開発

MVT / migrate / ORM

03 Gemini API

AI 連携 / プロンプト

04 htmx UI

JS 最小で動的 UI

05 AWS デプロイ

EC2 / Nginx / Gunicorn

06 HTTPS 化

Route 53 + Let's Encrypt

07 運用スキル

ログ / バックアップ / 監視

08 セキュリティ

Fail2ban / UFW / IAM

FIN

おつかれさま。これがあなたのスタート地点。

この章で学んだこと

- ✓ ログの見方（Gunicorn / Nginx / system）
- ✓ 502 / 500 / メモリ不足の対処
- ✓ htop / df / iftop で監視
- ✓ DB バックアップ + cron 自動化
- ✓ SSH ポート変更 + Fail2ban + UFW
- ✓ スケーリング（垂直 / 水平）
- ✓ EC2 停止 / 削除 で課金事故防止

NEXT STEP

あなた次第

- ▸ 新機能を追加する
- ▸ 別のアイデアで作り直す
- ▸ 友人に URL を共有
- ▸ GitHub でオープンソース化
- ▸ ポートフォリオに加える

→ 旅は終わらない、ここから始まる

実験が終わったら必ず stop または terminate。LLM で分からないことは、即聞く。