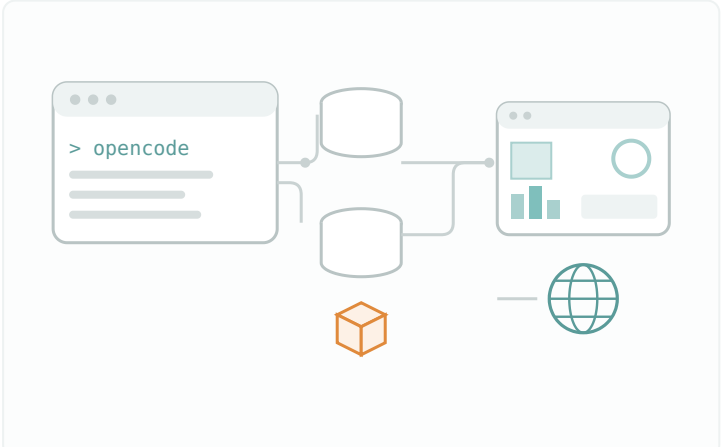
手順書

研修Day2
OpenCodeハンズオンマニュアル

人流データとオープンデータを、OpenCode で触る

ブラウザ可視化を Quick Tunnel で共有する



このハンズオンのゴール

人流データとオープンデータを OpenCode で触り、可視化を共有する

仙台研修 Day1（Tableau）の「人流がつながって見える」体験を、この環境向けに置き換えた内容です。 OpenCode（AI コーディング環境）に日本語で指示するだけで、CSV の結合・集計からブラウザ用ダッシュボードの作成、 そして一時 URL での共有までを体験します。

① 対話でさわる

OpenCode にプロンプトを貼り、リポジトリや MCP、人流 CSV を調べる。

② Viz を作る

2 年分の人流データを結合・集計し、年・月・地点・地図で可視化する。

③ 共有する

ローカル表示を確認し、Quick Tunnel の一時 URL で隣の人に見せる。

Day1（Tableau）との対応

Day1（Tableau）	Day2（OpenCode）
2025＋2026 のユニオン＋地点結合	同じデータをアプリ側で結合・集計
年比較・月別推移・地点別	ブラウザのチャートで同じ見方
地図＋日付操作	簡易マップ（難しければ一覧＋棒）
Tableau Public の共有 URL	`cloudflared` の一時 URL

この資料の位置づけ

本書は**準備編（GitHub Codespaces の作成）**から始めます。OpenCode の API キー設定や認証など詳細なセットアップは**別手順書**も参照してください。

概要

所要時間	約 45〜60 分（途中区切り可）
前提	GitHub アカウントがあり、対象リポジトリ（`kzkski/cloudflare-dev-container`）にアクセスできること
起動手順	**準備編（Step 0）**で GitHub Codespaces を作成します（API キー等の詳細は別手順書も参照）
後日回し	Skills の深掘り、Hono スキャフォールド練習、FIWARE Orion（moc-skill）

当日の流れ（目安）

時刻	Step	内容
0–2 分	0	動作確認
2–7 分	1	基本対話
7–17 分	2	MCP（オープンデータ検索）
17–24 分	3	人流ウォームアップ
24–50 分	4	Viz 作成（本丸）
50–55 分	4-6	Quick Tunnel 公開
55–60 分	—	振り返り

30 分で区切る場合

Step 3 まで完了 → あとから Step 4。時間切れでも、止まった Step のプロンプトから再開できます。なお、Codespace の作成（準備編）は上記の所要時間には含みません。

進捗チェックリスト

- [ ] Step 0　動作確認
- [ ] Step 1　基本対話（README 要約）
- [ ] Step 2　ckan-open-data MCP
- [ ] Step 3　人流 CSV ウォームアップ
- [ ] Step 4-1〜4-3　Viz（年・月・地点）
- [ ] Step 4-4　地図（またはフォールバック）
- [ ] Step 4-5　ローカル表示
- [ ] Step 4-6　Quick Tunnel 公開

0

準備編｜GitHub Codespaces を起動する

ハンズオンを始める前に、GitHub Codespaces でこのリポジトリの開発環境を立ち上げます。
ボタン操作だけで、opencode / wrangler / cloudflared が入った環境ができあがります。
（リポジトリ：kzkski/cloudflare-dev-container）

0　準備編｜GitHub Codespaces を起動する

GitHub 上のボタン操作だけで、`opencode` / `wrangler` / `cloudflared` などが揃った開発環境（Codespace）が立ち上がります。

任意の事前準備（OpenCode の有料 API を使う場合のみ）

リポジトリの **Settings → Secrets and variables → Codespaces** を開き、**New repository secret** をクリック。Name に `OPENCODE_API_KEY`、Value に API キーを入力して保存します。詳細は**別手順書**を参照。

1　GitHub にログインしてリポジトリを開く

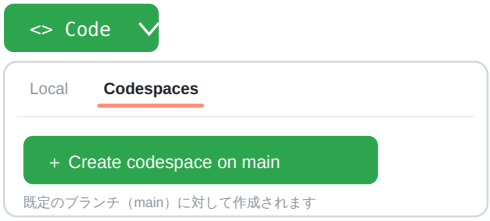
まず GitHub に**ログインした状態**で、ブラウザから次のリポジトリを開きます。

> **リポジトリ URL**
>
> **`https://github.com/kzkski/cloudflare-dev-container`**

※ ログインしていないと、次の `Codespaces` 作成メニューが表示されません。

2　Codespace を作成する

開いたページで `Code` → `Codespaces` → `Create codespace on main` を選択します。



3　ビルドとセットアップの完了を待つ

コンテナのビルドが進みます（作成ログで進捗を確認できます）。完了すると Codespaces の画面が開き、ポート `8787`（Wrangler Dev Server）の転送が設定されます。

続いて `.devcontainer/setup.sh` が自動実行され、OpenCode・Skills・Wrangler などがインストールされます。**初回は 3〜5 分程度**かかることがあります。完了まで気長に待ちます。

4　新しいターミナルを開く

セットアップ完了後、**新しいターミナルを開いて**から `opencode` や `wrangler` を使い始めます（PATH の反映に新しいセッションが必要です）。

✓ 完了条件

- Codespaces の画面が開き、ポート 8787 が転送されている
- 新しいターミナルで `opencode` / `wrangler` が使える

1

OpenCode をさわってみる

まずは OpenCode が使えることを確認し、リポジトリの文脈を読ませて
対話の基本を掴みます。
（Step 0・Step 1）

0 動作確認（2 分）

目的：OpenCode が使えることを確認する。ターミナルで次を実行します。

terminal

```
$ opencode auth list
$ opencode
```

起動後、次を貼ります。

OpenCode に貼るプロンプト

こんにちは。短い返事で自己紹介して。

✓ 完了条件

- 対話でき、返答がある

⚠ うまくいかないとき

コマンドが見つからない → ターミナルを開き直す。それでもダメなら別手順書のセットアップ確認。
認証エラー → 別手順書の `OPENCODE_API_KEY` 等を確認。

1 基本対話とファイル参照（5 分）

目的：リポジトリの文脈を読ませて、対話の基本を掴む。

OpenCode に貼るプロンプト

このリポジトリの `README.md` を読んで要約して。
どんなツールが入っていて、何ができる環境か、箇条書きで短く教えて。

✓ 完了条件

- OpenCode / Skills / MCP / Wrangler / cloudflared / 人流サンプル などに触れた要約が出る

任意の深掘り

`data/people-flow/README.md` も読んで、CSV の役割を3行で説明して。

2

オープンデータを検索する

事前登録済みのリモート MCP「ckan-open-data」を使い、仙台市オープンデータを検索します。

（Step 2）

2 ckan-open-data MCP（8〜10 分）

目的：事前登録済みのリモート MCP を使い、仙台市オープンデータを検索する。

操作：① OpenCode 内の MCP 一覧で `ckan-open-data` が有効か確認する　② 次のプロンプトを貼る。

OpenCode に貼るプロンプト（必須）

```
ckan-open-data の MCP ツール search_datasets を使って、
portal=sendai、keyword=人口、limit=5 でデータセットを検索し、
タイトルと概要を日本語で要約して。
ツールが使えなければ、その旨と設定の確認ポイントを教えて。
```

余裕があれば（任意）

```
続けて list_organizations で portal=sendai の組織を最大10件、名前だけ一覧して。

さっき見つかったデータセットのうち1件について、get_dataset で詳細を取り、
どんなリソース（CSV等）があるか短く教えて。
```

✓ 完了条件

- MCP 経由の検索結果（または明確なエラー理由）が返る

うまくいかないとき

症状	対処
MCP が一覧にない	`~/.config/opencode/opencode.json` の URL が `/mcp` で終わるか確認し、`opencode` を再起動
接続できない	`curl -sI "${CKAN_MCP_BASE_URL}/mcp"` で応答確認（別ターミナル）
HTTP 429	レート制限。少し待って再試行

詳細はリポジトリ README の「ckan-open-data MCP のトラブルシュート」を参照。

3

人流データの下調べ

Day1 のデータ準備に相当する「結合できるか」を先に確認します。可視化は Step 4 で行います。

（Step 3）

3 人流 CSV ウォームアップ（5〜7 分）

目的：Day1 のデータ準備に相当する「結合できるか」を先に確認する。可視化は Step 4。

使うファイル

ファイル	役割
`data/people-flow/people-flow-2025.csv`	2025 年の人流
`data/people-flow/people-flow-2026.csv`	2026 年の人流
`data/people-flow/people-flow-sensor.csv`	地点マスタ（緯度経度・地点名）

結合キーは `identifcation`（元データのスペルのまま）。

OpenCode に貼るプロンプト

```
data/people-flow/ の人流データを調べて。

やってほしいこと:
1. people-flow-2025.csv と people-flow-2026.csv を縦に結合できるか確認
2. people-flow-sensor.csv と identifcation で結合できるか確認
3. locationName に「定禅寺」を含む地点について、holidayFlg 別の平均 peopleCount を比較

結果は表か箇条書きで短く。まだダッシュボードは作らなくてよい。
```

✓ 完了条件

- 結合の可否が分かる
- 定禅寺通付近の休日／平日の差が言葉で分かる

任意の深掘り

センサー地点は何箇所？ `locationName` を一覧して。

4

人流 Viz を作って公開する

2 年分の人流データをアプリ側で結合・集計し、ブラウザで可視化。最後に Quick Tunnel の一時 URL で共有します。

（Step 4-1 〜 4-6・本丸）

完成イメージ（チェック項目）

必須

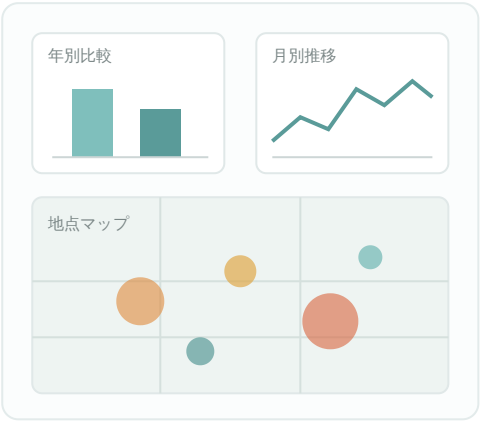
- [ ] 年別の peopleCount 比較
- [ ] 月別の推移
- [ ] 地点（locationName）別の比較
- [ ] ローカルでブラウザ表示
- [ ] Quick Tunnel で URL 共有

推奨

- [ ] 緯度経度の地図（円の大きさ ≒ 通行量）

任意（時間があれば）

- [ ] 日付スライダー
- [ ] 時間単位の再生



成果物の置き場（推奨）

リポジトリ直下に `handson-viz/` を作る想定です。OpenCode への指示にも含めます。 Codespace の `wrangler dev`（ポート 8787）で動く構成にします。

Step 4 のミニ目次

Step	内容
4-1	要件を一度に渡す（3 分）
4-2	データ準備の確認（5 分）
4-3	チャート 3 種（10〜12 分）
4-4	地図（またはフォールバック）（8〜10 分）
4-5	ローカルで表示確認（3 分）
4-6	Quick Tunnel で共有（5 分）

4　人流 Viz を作って公開する

4-1 要件を一度に渡す（3 分）

OpenCode に貼るプロンプト

仙台市の人流センサー CSV で、ブラウザ閲覧用の簡易ダッシュボードを作って。

【データ】
- data/people-flow/people-flow-2025.csv
- data/people-flow/people-flow-2026.csv
- data/people-flow/people-flow-sensor.csv
- 結合キーは identifcation（スペル注意）

【要件】
1. 2025 と 2026 を縦結合し、sensor と結合する
2. 年別の peopleCount 合計（または平均）の比較チャート
3. 月別の peopleCount 推移チャート
4. 地点（locationName）別の比較（セレクト or 色分け）
5. 緯度経度の地図。難しければ地点一覧 + peopleCount の棒で代替可
6. できれば日付で絞り込める UI
7. 配置先は handson-viz/ 。Codespace で wrangler dev（ポート 8787）で動く構成にして
8. 全 CSV をブラウザに載せると重いので、必要な集計を JSON 等に落としてフロントは軽くして

まずは最小で動くところまで作って。説明は短く、コマンドも教えて。

✓ 完了条件

- 方針と作成ファイルの説明が出て、実装が進む（または完了する）

うまく進まないとき

続きを進めて。まだ動く画面が無いなら、年別と月別の2チャートだけ先に表示できるようにして。

4-2 データ準備の確認（5 分）

Day1 の「データソース準備」に相当。

OpenCode に貼るプロンプト

handson-viz のデータ準備を確認して。

確認してほしいこと:
- 2025 と 2026 の行がまとまっているか
- identifcation で locationName / latitude / longitude が付いているか
- dateObservedFrom を日時として扱えているか
- 地点数はおおよそ28か

問題があれば修正して。結果は短く報告して。

✓ 完了条件

- 結合と日時・地点数が妥当

4-3 チャート 3 種（10〜12 分）

Day1 の棒・月別・地点色分けに相当。

まだ足りないとき用プロンプト

ダッシュボードに次の3つが揃っているか確認し、無ければ追加して。

A. 年別（2025 vs 2026）の peopleCount 比較
B. 月別の peopleCount 推移（折れ線でよい）
C. 地点（locationName）を選べる、または複数地点が比較できる表示

見た目より「同じ問いが読めること」を優先。CDN の Chart.js 等を使ってよい。

✓ 完了条件

- A〜C が表示される（この時点で `wrangler dev` できていればなお良い）

4-4 地図（またはフォールバック）（8〜10 分）

推奨プロンプト

地図を追加して。

- 各地点を latitude / longitude でプロット（Leaflet + CDN でよい）
- 円の大きさは peopleCount（集計済み）に比例
- ホバーまたはクリックで locationName を表示
- 地図が難しければ、同じデータを地点一覧 + 棒グラフにフォールバックしてよい

任意（時間がある人）

日付スライダーを付けて、選んだ日の地点別 peopleCount が地図（または一覧）に反映されるようにして。
時間再生までは不要。まずは日付1日単位でよい。

✓ 完了条件

- 地点と通行量の関係が画面で分かる（地図 or 一覧）

4-5 ローカルで表示確認（3 分）

OpenCode が案内したディレクトリ・コマンドに従います。典型例：

terminal

```
$ cd handson-viz
$ npm install   # 必要な場合のみ
$ wrangler dev
```

Codespaces の **Ports** で 8787（または表示ポート）を開き、チャートを操作します。

動かないとき OpenCode へ

```
wrangler dev でローカル表示できない。エラーログは次の通り。
（ここにエラーを貼る）

修正して、起動コマンドを1つだけ教えて。
```

✓ 完了条件

- 自分のブラウザでダッシュボードが動く

4-6 Quick Tunnel で共有（5 分）

Day1 の Tableau Public URL に相当。**別ターミナル**を開きます（`wrangler dev` は動かしたまま）。

terminal（別ターミナル）

```
$ cloudflared tunnel --url http://localhost:8787
```

① 表示された `https://xxxx.trycloudflare.com` をコピー　② 自分のブラウザで開く　③ 可能なら隣の人と URL を交換して見る。

✓ 完了条件

- 一時 URL でダッシュボードが見える

⚠ 注意

URL は一時的です。プロセス終了や時間経過で消えます（本番向けの恒常 Tunnel ではありません）。

完成です！お疲れ様でした！

Step 4 振り返りチェック

- [ ] 2 年分をまとめて見られた
- [ ] 地点マスタとつながった
- [ ] 月・地点の違いが画面で分かった
- [ ] （推奨）地図で空間が分かった
- [ ] URL で他者に見せられた

時間切れ・あとから再開

止まった場所	再開方法
Step 2 まで	Step 3 のプロンプトから
Step 3 まで	Step 4-1 の長いプロンプトから
チャート途中	Step 4-3 の「A〜C 追加」プロンプト
地図で詰まった	Step 4-4 のフォールバック指示
ローカルは動く	Step 4-6 の `cloudflared` だけ
Tunnel まで完了	日付スライダーを宿題に（任意プロンプト）

⚠ Tunnel だけ失敗したとき

> `cloudflared tunnel` で公開できない。実行したコマンドと出力は次の通り。
> （ここに貼る）
>
> `localhost` のポート番号の確認方法と、正しい `cloudflared` コマンドを教えて。

付録｜プロンプト早見表（全文・コピペ用）①

各 Step で OpenCode に貼るプロンプトを省略せずに掲載しています。そのままコピーして使えます。

Step 0｜疎通（動作確認）

```
こんにちは。短い返事で自己紹介して。
```

Step 1｜環境把握（README 要約）

```
このリポジトリの README.md を読んで要約して。
どんなツールが入っていて、何ができる環境か、箇条書きで短く教えて。
```

Step 1｜任意の深掘り

```
data/people-flow/README.md も読んで、CSV の役割を3行で説明して。
```

Step 2｜オープンデータ検索（必須）

```
ckan-open-data の MCP ツール search_datasets を使って、
portal=sendai、keyword=人口、limit=5 でデータセットを検索し、
タイトルと概要を日本語で要約して。
ツールが使えなければ、その旨と設定の確認ポイントを教えて。
```

Step 2｜余裕があれば（任意 1）

```
続けて list_organizations で portal=sendai の組織を最大10件、名前だけ一覧して。
```

Step 2｜余裕があれば（任意 2）

```
さっき見つかったデータセットのうち1件について、get_dataset で詳細を取り、
どんなリソース（CSV等）があるか短く教えて。
```

Step 3｜人流の下調べ

```
data/people-flow/ の人流データを調べて。

やってほしいこと:
1. people-flow-2025.csv と people-flow-2026.csv を縦に結合できるか確認
2. people-flow-sensor.csv と identifcation で結合できるか確認
3. locationName に「定禅寺」を含む地点について、holidayFlg 別の平均 peopleCount を比較

結果は表か箇条書きで短く。まだダッシュボードは作らなくてよい。
```

Step 3｜任意の深掘り

```
センサー地点は何箇所？ locationName を一覧して。
```

Step 4-1｜Viz 一式作成

```
仙台市の人流センサー CSV で、ブラウザ閲覧用の簡易ダッシュボードを作って。

【データ】
- data/people-flow/people-flow-2025.csv
- data/people-flow/people-flow-2026.csv
- data/people-flow/people-flow-sensor.csv
- 結合キーは identifcation（スペル注意）

【要件】
1. 2025 と 2026 を縦結合し、sensor と結合する
2. 年別の peopleCount 合計（または平均）の比較チャート
3. 月別の peopleCount 推移チャート
4. 地点（locationName）別の比較（セレクト or 色分け）
5. 緯度経度の地図。難しければ地点一覧 + peopleCount の棒で代替可
6. できれば日付で絞り込める UI
7. 配置先は handson-viz/ 。Codespace で wrangler dev（ポート 8787）で動く構成にして
8. 全 CSV をブラウザに載せると重いので、必要な集計を JSON 等に落としてフロントは軽くして

まずは最小で動くところまで作って。説明は短く、コマンドも教えて。
```

Step 4-1｜うまく進まないとき

```
続きを進めて。まだ動く画面が無いなら、年別と月別の2チャートだけ先に表示できるようにして。
```

Step 4-2｜データ準備の確認

```
handson-viz のデータ準備を確認して。

確認してほしいこと:
- 2025 と 2026 の行がまとまっているか
- identifcation で locationName / latitude / longitude が付いているか
- dateObservedFrom を日時として扱えているか
- 地点数はおおよそ28か

問題があれば修正して。結果は短く報告して。
```

Step 4-3｜チャート 3 種

```
ダッシュボードに次の3つが揃っているか確認し、無ければ追加して。

A. 年別（2025 vs 2026）の peopleCount 比較
B. 月別の peopleCount 推移（折れ線でよい）
C. 地点（locationName）を選べる、または複数地点が比較できる表示

見た目より「同じ問いが読めること」を優先。CDN の Chart.js 等を使ってよい。
```

Step 4-4｜地図（推奨）

```
地図を追加して。

- 各地点を latitude / longitude でプロット（Leaflet + CDN でよい）
- 円の大きさは peopleCount（集計済み）に比例
- ホバーまたはクリックで locationName を表示
- 地図が難しければ、同じデータを地点一覧 + 棒グラフにフォールバックしてよい
```

Step 4-4｜任意（日付スライダー）

```
日付スライダーを付けて、選んだ日の地点別 peopleCount が地図（または一覧）に反映されるようにして。
時間再生までは不要。まずは日付1日単位でよい。
```

Step 4-5｜ローカルで動かないとき

```
wrangler dev でローカル表示できない。エラーログは次の通り。
（ここにエラーを貼る）

修正して、起動コマンドを1つだけ教えて。
```

Step 4-6｜Quick Tunnel で公開できないとき

```
cloudflared tunnel で公開できない。実行したコマンドと出力は次の通り。
（ここに貼る）

localhost のポート番号の確認方法と、正しい cloudflared コマンドを教えて。
```

ターミナルで使う主なコマンド

```
opencode auth list          # 認証確認
opencode                    # 起動
wrangler dev                # ローカル起動（handson-viz 内・ポート8787）
cloudflared tunnel --url http://localhost:8787   # 一時URLで共有（別ターミナル）
```

参考リンク

- ✓ リポジトリ README（ツール概要・MCP トラブルシュート）
- ✓ 人流データ説明：`data/people-flow/README.md`
- ✓ Day1 Tableau 完成イメージ（研修資料）：人流センサーの年次データ可視化・地図・日付操作

後日メニュー（この資料では扱わない）

- ✓ OpenCode Skills（Hono / Cloudflare / Wrangler など）の意図的な呼び出し
- ✓ `npm create cloudflare@latest` からの Hono プロジェクト作成練習
- ✓ moc-skill（FIWARE Orion）の読み取りクエリ