Absolutely, Dave. Based on your setup, here’s a comprehensive, contributor-friendly documentation draft that walks through configuration, environment setup, and deployment for the Lookout App. It’s tailored for both local development and Codespaces, with secure secret handling and reproducibility baked in.

🚀 Lookout App – Configuration & Deployment Guide

📦 Overview

The Lookout App is a Flask-based API that connects to a SQL Server backend. It supports hybrid development across local Docker and GitHub Codespaces, with secure secret injection and dynamic environment configuration.

🧰 Prerequisites

Local Development

* Docker & Docker Compose
* Bash shell (for start-dev.sh)
* Python 3.11+
* ODBC Driver 18 for SQL Server

Codespaces

* GitHub account with Codespaces enabled
* Secrets configured in repository settings

🧪 Environment Setup

1. **Secrets Configuration (Codespaces)**

Go to your GitHub repo → Settings → Codespaces → Secrets and add:

| **Secret Name** | **Description** |
| --- | --- |
| SQL\_SERVER\_USER\_CODESPACES | SQL Server username |
| SQL\_SERVER\_PASSWORD\_CODESPACES | SQL Server password |
| SQL\_SERVER\_CONTAINER\_SERVICE\_CODESPACES | Hostname or container name for SQL Server |

These will be injected into the dev container and used to generate .env.

2. **Local .env Setup**

Copy the template and fill in your local values:

cp .env.template .env

Edit .env with:

ENV=local SQL\_SERVER\_USER=your\_local\_user SQL\_SERVER\_PASSWORD=your\_local\_password SQL\_SERVER\_CONTAINER\_SERVICE=sqlserver

3. **Dev Container Initialization (Codespaces)**

On Codespaces startup:

* setup-env.sh runs via postCreateCommand
* It generates .env from GitHub secrets
* It creates .env.ready to signal readiness

You’ll see logs like:

🔧 Generating .env from Codespaces secrets... ✅ .env file created. ✅ .env.ready flag set.

🐳 Docker Compose Services

flask-app

* Flask API server
* Reads config from .env
* Waits for .env.ready before launching

sqlserver

* SQL Server 2022 container
* Password injected via .env
* Healthcheck ensures readiness before Flask connects

🚀 Starting the App

Local

docker compose up -d ./start-dev.sh

Codespaces

Auto-starts via postCreateCommand. If needed:

chmod +x start-dev.sh ./start-dev.sh

🔍 Diagnostic Endpoints

| **Route** | **Description** |
| --- | --- |
| /ping | Lists available databases |
| /db-status | Runs SELECT 1 to confirm connection |
| /config | Shows current config (masked) |
| /home | Basic welcome route |

🧪 Testing SQL Connectivity

curl http://localhost:5000/db-status

Expected response:

{ "status": "connected", "result": 1 }

🧼 Teardown

docker compose down rm .env .env.ready

🧭 Troubleshooting

* **App hangs on startup**  
  Check if .env.ready exists. If not, re-run setup-env.sh.
* **SQL Server connection fails**  
  Validate SQL\_SERVER\_CONTAINER\_SERVICE matches the Compose service name (sqlserver).
* **Missing secrets in Codespaces**  
  Confirm GitHub secrets are correctly named and injected.

Let me know if you want this split into multiple files (README.md, docs/setup.md, etc.) or want diagrams to visualize the flow. You’ve built a seriously elegant hybrid setup—this doc will make onboarding a breeze.

Great question, Dave — and you're asking it at exactly the right moment in your workflow evolution.

Using a GitHub Action to deploy or validate your Codespaces environment offers **automation, reproducibility, and team scalability** that go beyond what .devcontainer alone can do. Here's how it stacks up:

🧠 Why Use a GitHub Action?

✅ 1. **Automated Validation of Your Dev Environment**

Instead of manually launching Codespaces and hoping everything works, a GitHub Action can:

* Build your dev container from .devcontainer/Dockerfile
* Run setup-env.sh to generate .env dynamically
* Confirm that secrets are injected and .env.ready is created
* Launch your Flask app and hit /db-status to verify SQL connectivity

This acts like a CI-backed smoke test for your entire setup.

🚀 2. **Catch Config Drift Early**

Your .devcontainer config might work today, but what if:

* Someone changes the Dockerfile?
* A secret name is mistyped?
* A new contributor forgets to copy .env.template?

A GitHub Action can catch these issues automatically on every push or PR — before they break Codespaces or local dev.

👥 3. **Onboarding Confidence for Teams**

Imagine a new contributor joins and pushes a change. With a GitHub Action:

* Their setup is validated in CI
* You get a green checkmark confirming .env was generated, Flask booted, and SQL connected
* No more “it works on my machine” surprises

🔐 4. **Secure Secret Injection in CI**

You can inject secrets via GitHub Actions (separate from Codespaces secrets), allowing you to:

* Simulate production or staging environments
* Run integration tests with real credentials
* Avoid hardcoding sensitive values in .env

🧪 5. **Test Your Setup Outside Codespaces**

Not everyone uses Codespaces. A GitHub Action lets you:

* Build and test your container on GitHub’s runners
* Validate that your setup works in a clean, reproducible environment
* Catch platform-specific bugs (e.g., file permissions, race conditions)

🛠️ What You’d Need to Add

To make it truly useful, you’d want to:

* Inject secrets via env: or with: blocks
* Run setup-env.sh inside the container
* Validate .env.ready before launching Flask
* Hit /ping or /db-status to confirm SQL connectivity

Want me to refactor your current deploy-dev.yml into a modular, environment-aware validator that mirrors your Codespaces flow? I’ll make it elegant, secure, and CI-friendly.