Local.yml:

version: "3.9"

services:

api:

build:

context: .

dockerfile: ./docker/local/django/Dockerfile

volumes:

- .:/app:z

- static\_volume:/app/staticfiles

- media\_volume:/app/mediafiles

ports:

- "8000:8000"

env\_file:

- ./.envs/.local/.django

- ./.envs/.local/.postgres

depends\_on:

- postgres

- mailhog

command: /start

networks:

- authors-api

postgres:

build:

context: .

dockerfile: ./docker/local/postgres/Dockerfile

volumes:

- local\_postgres\_data:/var/lib/postgresql/data

- local\_postgres\_data\_backups:/backups

env\_file:

- ./.envs/.local/.postgres

networks:

- authors-api

mailhog:

image: mailhog/mailhog:v1.0.0

container\_name: mailhog

ports:

- "8025:8025"

networks:

- authors-api

networks:

authors-api:

driver: bridge

volumes:

static\_volume:

media\_volume:

local\_postgres\_data: {}

local\_postgres\_data\_backups: {}

Explanation:

1. \*\*`version: "3.9"`\*\*: Specifies the version of the Docker Compose file format you're using. This version is associated with the features and syntax available in the file.

2. \*\*`services:`\*\*: Defines the services that make up your application.

- \*\*`api:`\*\*: Configuration for the Django API service.

- \*\*`build:`\*\*: Configuration for building the Docker image.

- \*\*`context: .`\*\*: The build context, the root directory for building the image. In this case, it's the current directory (`.`).

- \*\*`dockerfile: ./docker/local/django/Dockerfile`\*\*: Path to the Dockerfile for building the image.

- \*\*`volumes:`\*\*: Defines volumes to be mounted for the service.

- \*\*`- .:/app:z`\*\*: Mounts the current directory (`.`) to `/app` inside the container, with SELinux context `z`.

- \*\*`- static\_volume:/app/staticfiles`\*\*: Uses the named volume `static\_volume` for static files.

- \*\*`- media\_volume:/app/mediafiles`\*\*: Uses the named volume `media\_volume` for media files.

- \*\*`ports:`\*\*: Maps ports between the host and the container.

- \*\*`- "8000:8000"`\*\*: Maps port 8000 on the host to port 8000 on the container.

- \*\*`env\_file:`\*\*: Specifies environment files to load.

- \*\*`- ./.envs/.local/.django`\*\*: Path to the Django environment file.

- \*\*`- ./.envs/.local/.postgres`\*\*: Path to the PostgreSQL environment file.

- \*\*`depends\_on:`\*\*: Specifies services that this service depends on.

- \*\*`- postgres`\*\*: Depends on the PostgreSQL service.

- \*\*`- mailhog`\*\*: Depends on the MailHog service.

- \*\*`command: /start`\*\*: Overrides the default command to start the container.

- \*\*`networks:`\*\*: Specifies the networks the service should be connected to.

- \*\*`- authors-api`\*\*: Connects to the `authors-api` network.

- \*\*`postgres:`\*\*: Configuration for the PostgreSQL service.

- \*\*`build:`\*\*: Similar to the `api` service, specifies the build context and Dockerfile.

- \*\*`volumes:`\*\*: Defines volumes for PostgreSQL data and backups.

- \*\*`env\_file:`\*\*: Specifies the PostgreSQL environment file.

- \*\*`networks:`\*\*: Connects to the `authors-api` network.

- \*\*`mailhog:`\*\*: Configuration for the MailHog service.

- \*\*`image: mailhog/mailhog:v1.0.0`\*\*: Uses a pre-built MailHog image.

- \*\*`container\_name: mailhog`\*\*: Sets the container name to "mailhog".

- \*\*`ports:`\*\*: Maps port 8025 on the host to port 8025 on the container.

- \*\*`networks:`\*\*: Connects to the `authors-api` network.

3. \*\*`networks:`\*\*: Defines the networks used by the services.

- \*\*`authors-api:`\*\*: Configuration for the `authors-api` network.

- \*\*`driver: bridge`\*\*: Specifies the network driver as "bridge."

4. \*\*`volumes:`\*\*: Defines named volumes used by the services.

- \*\*`static\_volume:`\*\*, \*\*`media\_volume:`\*\*, \*\*`local\_postgres\_data:`\*\*, \*\*`local\_postgres\_data\_backups:`\*\*: Named volumes.

\*\*Command Explanation:\*\*

The `docker compose -f local.yml config` command is used to display the effective configuration that Docker Compose would use based on the specified `local.yml` file. It validates the file, substitutes variables if present, and prints the final configuration. This can be useful for checking if the configuration is correctly interpreted by Docker Compose before actually running the services.