Handout: Override

Docker Compose allows you to split your configuration into multiple files — typically separating **base**, **development**, and **production** settings. This makes your setup easier to manage and switch between environments.
When you run:

```
docker compose up
```

Docker Compose automatically uses:

- `compose.yaml`
- `compose.override.yaml`

1. The Override File

The override file is **merged** into the base config. You can override or extend any setting, including:

- environment variables
- ports
- volumes
- commands
- labels

Using the following configuration in `compose.yaml`:

```
services:
  app:
  image: myapp
  environment:
    MODE: production
```

and overriding it in `compose.override.yaml`:

```
services:
  app:
  environment:
    MODE: development
```

will result in `MODE=development` — because the override file is applied last and replaces the `MODE` value.

2. Custom File Combinations

You can define your own file combinations with `-f`:

```
docker compose -f compose.yaml -f compose.prod.yaml up --build
```

This tells Docker Compose to use `compose.yaml` as the base, then merge in `compose.prod.yaml`.

When using the `-f` flag, only the files explicitly listed are used. `compose.yaml` or `compose.override.yaml` is not automatically applied in this case.

Always match your cleanup and startup files. If you start your application with custom files using `-f`, you must **also use the same `-f` flags to clean up properly**:

```
docker compose -f compose.yaml -f compose.prod.yaml down
```

If you omit the `-f` flags during shutdown, Compose will fall back to the default files (`compose.yaml` + `compose.override.yaml`), which may lead to stale volumes, orphaned containers, or incomplete teardown of your production setup.

3. Summary

File	Role
`compose.yaml`	Base configuration (shared)
`compose.override.yaml`	Development overrides (auto-applied)
`compose.prod.yaml`	Production overrides (manually used)

- **Keep `compose.yaml` minimal and universal**: Define core settings here — image, volumes, ports, networks, etc.
- **Use `compose.override.yaml` for development**: This file should only contain development-specific changes (e.g., bind mounts, debug settings, dev env vars).
- **Add `compose.prod.yaml` for deployment**: Include production overrides only: stricter env vars, fewer ports, no dev volumes.