# How to contribute to eShopOnContainers

This repo is a reference and learning resource and everyone is invited to contribute, however not all PRs will be accepted into the main branch (\*\*`dev`\*\*).

There's a general development strategy that's driven by @CESARDELATORRE, who chooses, or defines criteria for choosing, the issues to include in the codebase, given a bunch of constraints and other guidelines.

However you can always get in touch with him, if you want to implement some general-interest feature in your repo and have it referenced from the [documentation](https://docs.microsoft.com/dotnet/standard/microservices-architecture/) or the [Microservices eBook](https://aka.ms/microservicesebook/).

## Coding Standards

There are no explicit coding standards so pay attention to the general coding style, that's (mostly) used everywhere.

However, there's only one \*\*REALLY\*\* important rule: \*\*use spaces for indenting\*\* ?.

## Development Process

In order to help manage community contributions and avoid conflicts, there's a [Development project](https://github.com/dotnet-architecture/eShopOnContainers/projects/3) in this repo, to track assignments to any significant development effort.

Great but... \*\*what's "significant"\*\*?

That's not too easy to define and there are no clear criteria right now but, probably, changing "a couple" lines of code in one file would not qualify while changing "a bunch" of files would.

We'll all be learning in the process so we'll figure it out somehow.

### General Steps

1. Issues are managed as usual with the regular issues list, just like any other repo.

2. Once an issue is marked as a bug, enhancement, new feature or whatever needs development work, it will be labeled as a \*\*backlog-item\*\* and included as the last item in the Backlog project column.

3. Community members can propose themselves to code an issue.

4. @CESARDELATORRE/collaborators will prioritize the backlog items and arrange them in the \*\*Backlog\*\* column, so that the items in the top of the list are implemented first.

5. @CESARDELATORRE/collaborators will review the issues and select the ones approved to begin development with, and move them to the \*\*Approved\*\* column.

6. Issues in the \*\*Approved\*\* column can be assigned to a \*\*collaborator\*\* or to a \*\*community member\*\* who would then begin working on the issue and submit a PR as usual.

## Tests

There's not a tests policy in the project at this moment, but it'll be greatly appreciated if you include them within the [updated test structure](./test/readme.md).

## Forks and Branches

All contributions must be submitted as a [Pull Request (PR)](https://help.github.com/articles/about-pull-requests/) so you need to [fork this repo](https://help.github.com/articles/fork-a-repo/) on your GitHub account.

The main branches are \*\*`dev`\*\* and \*\*`master`\*\*:

- \*\*`dev`\*\*: Contains the latest code \*\*and it is the branch actively developed\*\*.

\*\*All PRs must be against `dev` branch to be considered\*\*. This branch is developed using .NET Core 2.x

- \*\*`master`\*\*: Synced from time to time from \*\*`dev`\*\*. It contains "stable" code.

(\*\*Keep in mind "stable" does not mean PRODUCTION-READY!\*\*)

- Any other branch is considered temporary and could be deleted at any time. Do not submit any PR to them!

## DISCLAIMER - This is not a PRODUCTION-READY TEMPLATE for microservices

eShopOnContainers is a reference application to \*\*showcase architectural patterns\*\* for developing microservices applications on .NET Core. \*\*IT IS NOT A PRODUCTION-READY TEMPLATE\*\* to start real-world application. In fact, the application is in a \*\*permanent beta state\*\*, as it’s also used to test new potentially interesting technologies as they show up.

Since this is a learning resource, some design decisions have favored simplicity to convey a pattern, over production-grade robustness.

## Suggestions

We hope this helps us all to work better and avoid some of the problems/frustrations of working in such a large community.

We'd also appreciate any comments or ideas to improve this.