

# Challenge2 Step By Step

Alex Fletcher edited this page 5 days ago · 1 revision

## Challenge 2 - Step - by - Step

#### Introduction

As a development team at Globoticket, it's crucial to ensure a consistent and efficient development environment for all team members. Setting up a standardized codespace helps achieve this by providing a cloud-based development environment preconfigured with all necessary tools and extensions. This guide will walk you through the steps to create, configure, and maintain a codespace that can run C# code, ensuring every developer has the same setup. This minimizes setup time, reduces configuration errors, and allows the team to focus on building and improving the application.

#### Prerequisites

Before you start, make sure you have the following prerequisites:

Finish Challenge 1 - Ensure you have completed the steps in Challenge 1 to set up a codespace with the necessary
extensions.

#### **Release Pipeline Configuration**

Ensure the workflows target the production environment:

- 1. Edit the .github/workflows/build-deploy-catalog.yml and make sure you add environment: production to the deploy inh
- 2. Edit the .github/workflows/build-deploy-frontend.yml and make sure you add environment: production to the deploy job.
- 3. Edit the .github/workflows/build-deploy-ordering.yml and make sure you add environment: production to the deploy job.

Create/update the production environment:

- 1. Navigate to the Settings tab of your repository.
- 2. Open the Environments section.
- If an environment named production does not exist, click New environment and create a new environment named production, otherwise open the production environment.
- 4. Set the following options:
  - o Check Required Reviewers and add your team as reviewer
  - o Check Prevent self-review.
  - o Make sure you click Save protection rules.
- 5. Under Deployment branches and tags select Protected Branched Only.

### **Branch Protection Rules**

- 1. Navigate to the **Settings** tab of your repository.
- 2. Open the Rules section
- 3. Click on the Rulesets.
- 4. Click on the  ${\bf New\ Ruleset}$  button and select the  ${\bf New\ Branch\ ruleset}$  option.
- 5. Set the following options:
  - o Ruleset name: Protect main branch
  - o Enforcement status: Active
  - Under Target branches, click Add Target and select Include default branch.
  - o Under Rules, check the following rules:
    - Restrict deletions
    - Require linear history
    - Require a pull request before merging
      - Require Approvals: 1
      - Dismiss stale pull request approvals when new commits are pushed
      - Require review from Code Owners
      - Require approval of the most recent reviewable push
    - Require conversation resolution before merging
    - Block force pushes

### **Codeowners Setup**

- 1. Add a new file to the repository in the .github folder with the following name CODEOWNERS
- 2. For each folder and file you want to protect add a line with the folder or file name and the GitHub team that should review any changes. For example:

/catalog/ @globaldevopsexperience/your-team /frontend/ @globaldevopsexperience/your-team /ordering/ @globaldevopsexperience/your-team

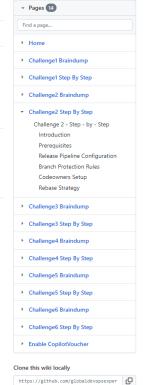
4. Commit the file to the repository. If needed merge your changes to the main branch.

## Rebase Strategy

1. Navigate to the 🕲 Settings tab of your repository.

Also add a line to peotect the CODEOWNERS file itself:
 /.github/CODEOWNERS @globaldevopsexperience/your-team

- 2. Click on the General section.
- 3. Scroll down to the Pull Requests section.
- 4. Make sure Allow merge commits is unchecked.



© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information