1. Merging strategy.

For my opinion it should be Git Flow merging strategy.

Reasons:

We have team of 5 developers, and it will be convenient to have one common **development** branch for integration of several **feature**-branches results before moving them to **release** branch. Developers can work independently in parallel.

We have predefined schedule of releases to pre-prod and prod environments, so it makes sense to have **release** branches, which are forked from **develop** branch, where successfully tested features have been already integrated.

**Main** branch is the official story of software code. The code is deployed to pre-prod environment only from this branch.

Example of branches using with sample commits and merge requests is below:

Text

Description automatically generated

The main idea is that the code in **release** branch has already passed integration tests when different **feature** branches were merged in **develop** branch, and single developer does not work with **release** and **main** branches (so we have no risks of release postpone when something go wrong during single feature merging). This approach helps to handle and address the most of bugs earlier (failed features just are declined from merging into develop branch and continue separate “life” until they are completed).

Note: this example was created in https://git.epam.com/vitali\_sazonau/dqe\_mentoring\_program/-/tree/main/Module\_5\_CI%20CD%20Basics repository only for illustrating merge strategy. The project with task files stored in https://github.com/vitalisazonau/ci-cd\_task

1. Docker installation

A picture containing text, clock, green, dark

Description automatically generated

1. Run docker container with Jenkins

Docker file content:

FROM jenkins/jenkins:latest

USER root

RUN mkdir /my\_app

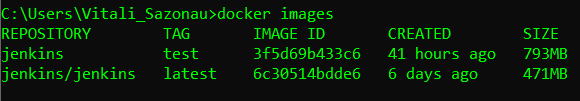
WORKDIR /my\_app

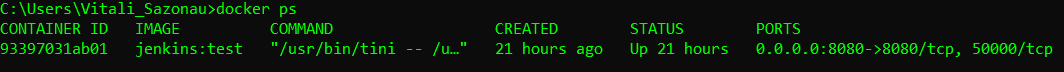
RUN pwd

RUN ls -la

RUN apt-get update

RUN apt-get install -y python3-pip





1. Set up CI Pipeline

Application

Description automatically generated with low confidence

Graphical user interface, text, application, email

Description automatically generated

1. Design Jenkins CI pipeline

Jenkins file for CI pipeline: CI\_pipeline.jenkins

Web hook to trigger CI pipeline:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Design Jenkins CD pipeline

Jenkins file for CD pipeline: CD\_pipeline.jenkins

This pipeline tests code in **release** branch and then merge **release** branch to **main** branch.

Configure personal access token in Github and set it in Jenkins:

Graphical user interface, text, application, email

Description automatically generated