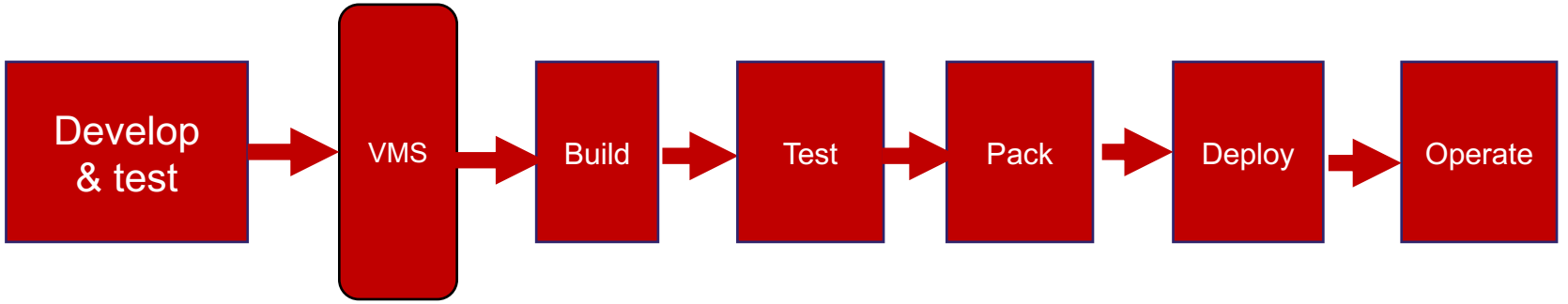
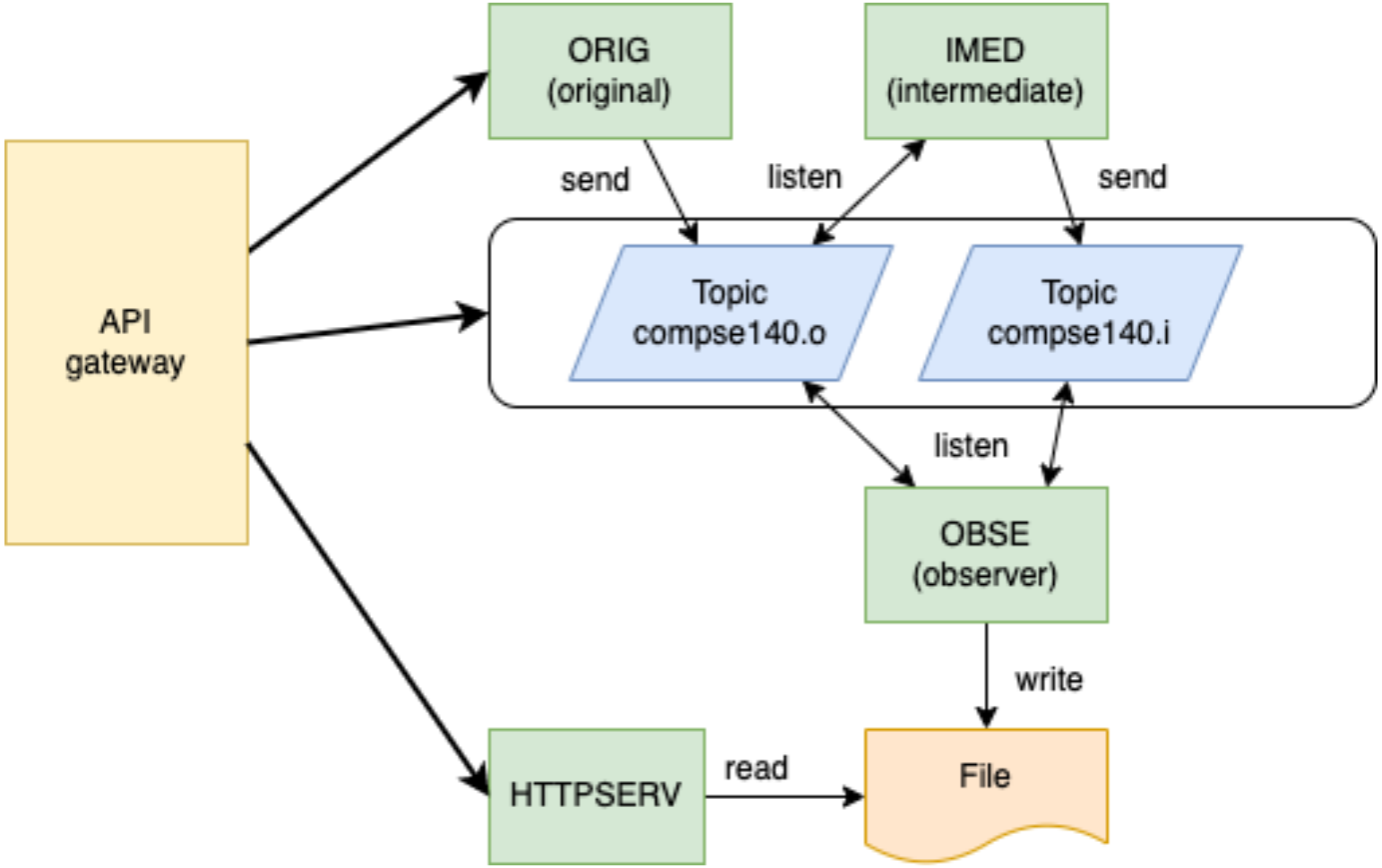
COMP.SE.140 Project 2022

Kari Systä

15.11.2022

Project


Develop & test
VMS
Build
Test
Pack
Deploy
Operate


ORIG (original)
IMED (intermediate)
send
listen
send
API gateway
Topic compse140.o
Topic compse140.i
listen
OBSE (observer)
write
HTTPSERV
read
File

Tampereen yliopisto
Tampere University

Schedule

- The instructions disclosed: 14.11.2022
  - Students can start by installing the gitlab-ci
  - New versions to resolve ambiguous parts may be published later.
- Discussions in the lecture: 15.11.2022
  - Students are asked to give clarification questions
- Latest submission if you want course to graded in 2022: 07.12.2022
- Latest submission to pass the course: 31.01.2023

Project includes

1. Install the pipeline infrastructure using gitlab-ci. This means that you should:
   - install gitlab and runners on their own machine. A fresh virtual machine is recommended. Instructions to help in this process are below in section gitlab-ci.
   - Define the pipeline using `.gitlab-ci.yml` for the application you implemented for the message-queue exercise. The result of the pipeline should be a running system, so the containers should be started automatically. (In other words: “git push => the system is up and running)
   - Test the pipeline with the current version of the application.
2. Create, setup and test an automatic testing framework
   - First, you need to select the testing tools. We do not require any specific tool, even your own test scripts can be used.
   - Create test to the existing functionality of the application (see “Application and its new features” below)
3. Implement the changes and additional functionalities to the RabbitMQ exercise

API gateway

GET /messages
Returns all message registered with OBSE-service

PUT /state (payload “INIT”, “PAUSED”, “RUNNING”, “SHUTDOWN”)

PAUSED = ORIG service is not sending messages

RUNNING = ORIG service sends messages

If the new state is equal to previous nothing happens.

There are two special cases:

INIT = everything is in the initial state and ORIG starts sending again, state is set to RUNNING
SHUTDOWN = all containers are stopped

GET /state
get the value of state

GET /run-log
Get information about state changes

Example output:
2020-11-01T06:35:01.373Z: INIT
2020-11-01T06:40:01.373Z: PAUSED
2020-11-01T06:40:01.373Z: RUNNING

GET /message-log
Forward the request to HTTPSERV and return the result

GET /node-statistic (optional)

Return core statistics (the five (5) most important in your mind) of the RabbitMQ. (For getting the information see https://www.rabbitmq.com/monitoring.html)
Output should syntactically correct and intuitive JSON. E.g:
{ “fd_used”: 5, …}

GET /queue-statistic (optional)

Return a JSON array per your queue. For each queue return “message delivery rate”, “messages publishing rate”, “messages delivered recently”, “message published lately”. (For getting the information see https://www.rabbitmq.com/monitoring.html)

Tampereen yliopisto
Tampere University

End report

1. **Instructions for the teaching assistant**

   Implemented optional features

   List of optional features implemented.

   Instructions for examiner to test the system.

   Pay attention to optional features.

2. **Description of the CI/CD pipeline**

   Briefly document all steps:

   Version management; use of branches etc

   Building tools

   Testing; tools and test cases

   Packing

   Deployment

   Operating; monitoring

3. **Example runs of the pipeline**

   Include some kind of log of both failing test and passing.

4. **Reflections**

   Main learnings and worst difficulties

   Especially, if you think that something should have been done differently, describe it here.

   Amount effort (hours) used

   Give your estimate

Grading

As already been communicated this project affects 40% of in the evaluation of the overall course. For that 40% we use the following table

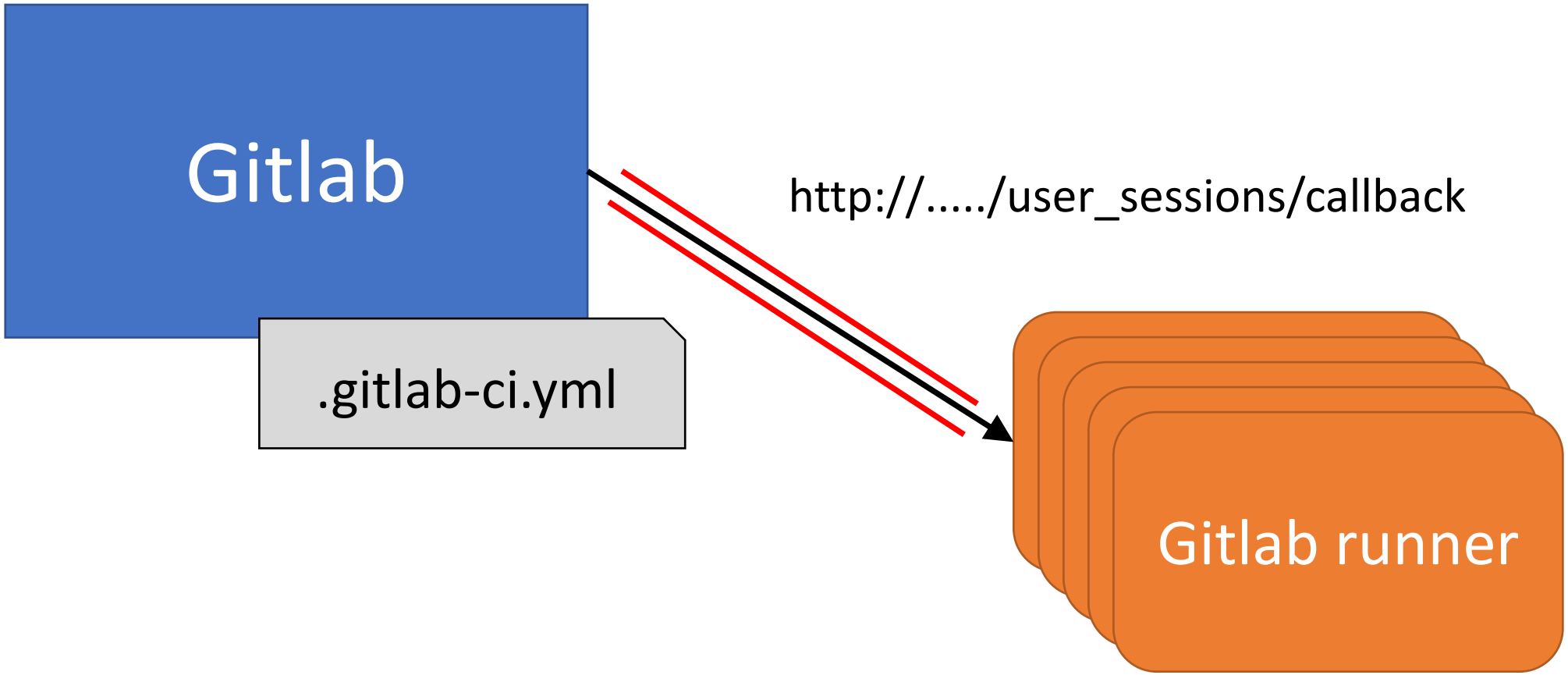
Compulsory parts work according to requirements	0..20 %
Implementation of optional features (each optional feature is worth of 5%)	0..30 %
Overall quality (clean code, good comments, ….)	0..5%
Quality of the end report	0..5% (+ up to 5% compensation of a good analysis of your solution and description of a better way to implement.)

Note: optional points can compensate problems elsewhere, but the total sum is capped at 50%. That means that max 10% can be used to compensate lost points in exercises and exam.

Tampereen yliopisto
Tampere University

Gitlab CI

https://docs.gitlab.com/ee/ci/



Tampereen yliopisto
Tampere University

Types of runners

Shared Runners

- These runners are useful for jobs multiple projects which have similar requirements. Instead of using multiple runners for many projects, you can use a single or a small number of Runners to handle multiple projects which will be easy to maintain and update.

Specific Runners

- These runners are useful to deploy a certain project, if jobs have certain requirements or specific demand for the projects. Specific runners use *FIFO* (First In First Out) process for organizing the data with first-come first-served basis.

Tampereen yliopisto
Tampere University

How to install .gitlab-ci.yml?

```
git add .gitlab-ci.yml
git commit -m "Add .gitlab-ci.yml”
git push origin master
```

passed	#2913		master 43dda676  more tests		00:00:36 1 month ago
passed	#2912		master 32e0f29b  more tests		00:00:36 1 month ago
failed	#2911		master 8bf6c037  more tests		00:00:16 1 month ago

```
Sphinx error:
Missing config path exercises/hello__hello/config.yaml
make: *** [html] Error 1
Makefile:60: recipe for target 'html' failed

*** ERROR in compile-rst

▼
▼
ERROR: Job failed: exit code 1
```