



BPMN AND CAMUNDA

Alessandro Bocci
name.surname@unipi.it

Advanced Software Engineering (Lab)
19/11/2023

What will you do?

- Use Camunda Modeler to design a business process.
- Implement tasks of the process.
- Deploy and run the process with Camunda.

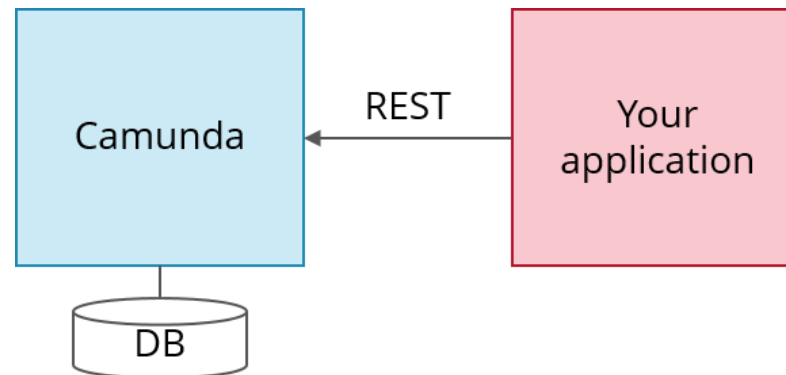


Software Prerequisites

- Camunda Modeler
- Camunda docker image
- Python code (.zip from Moodle)



- Camunda is a framework supporting BPMN for workflow and process automation.
- It provides a RESTful API which allows you to use your language of choice.



- Workflows are defined in BPMN which can be graphically modeled using the Camunda Modeler.

Camunda has got 2 «usage patterns» (A & B later on...)

(A) aka **endpoint-based** integration
(B) aka **queue-based** integration

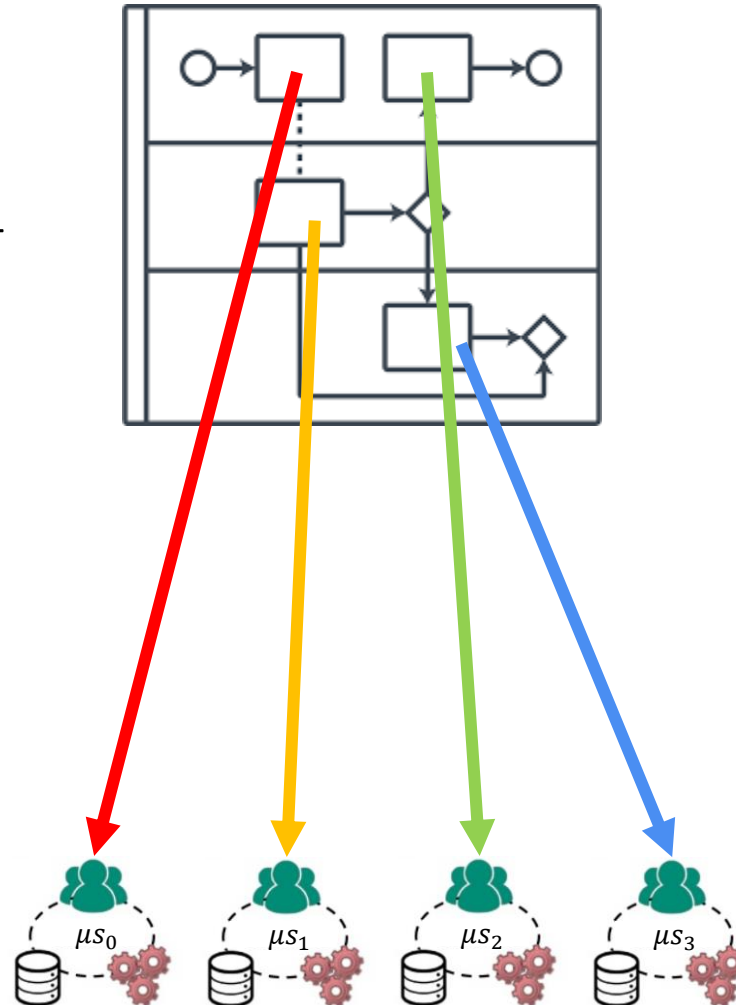
*Be sure you understand these patterns,
ask questions now if you don't.*

It is among the exam questions...

How does it work? (Pattern A)

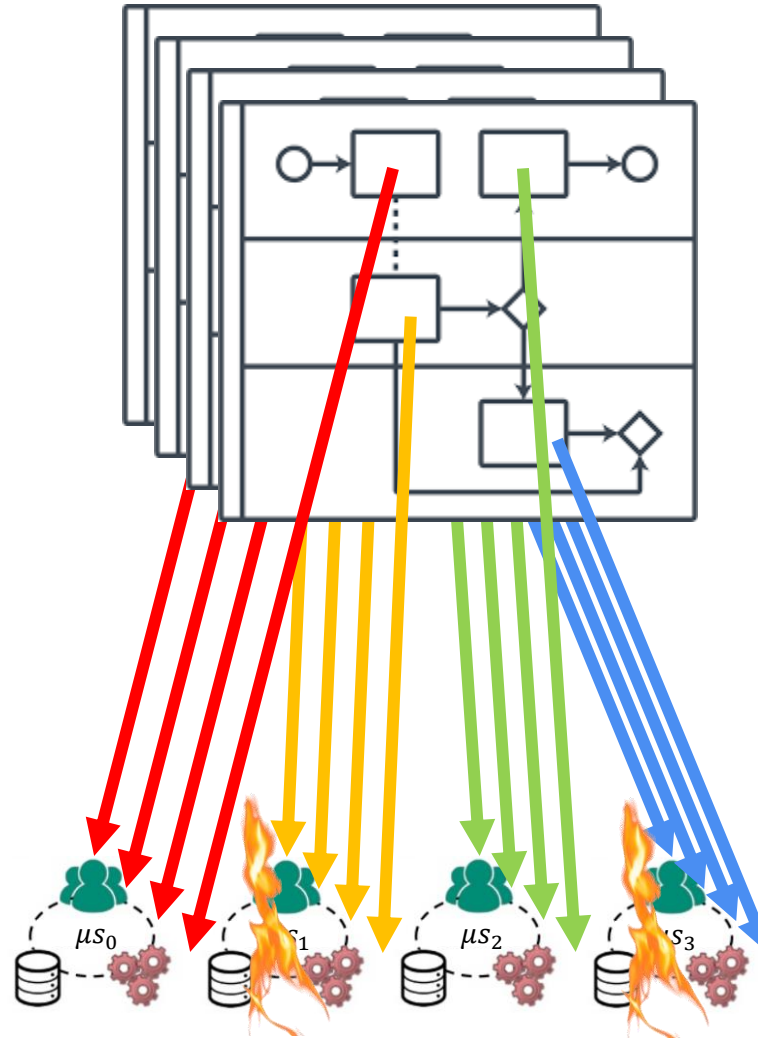
After defining a BPMN process, Camunda can directly call services via built-in *connectors*.

It supports REST



Scaling (Pattern A)

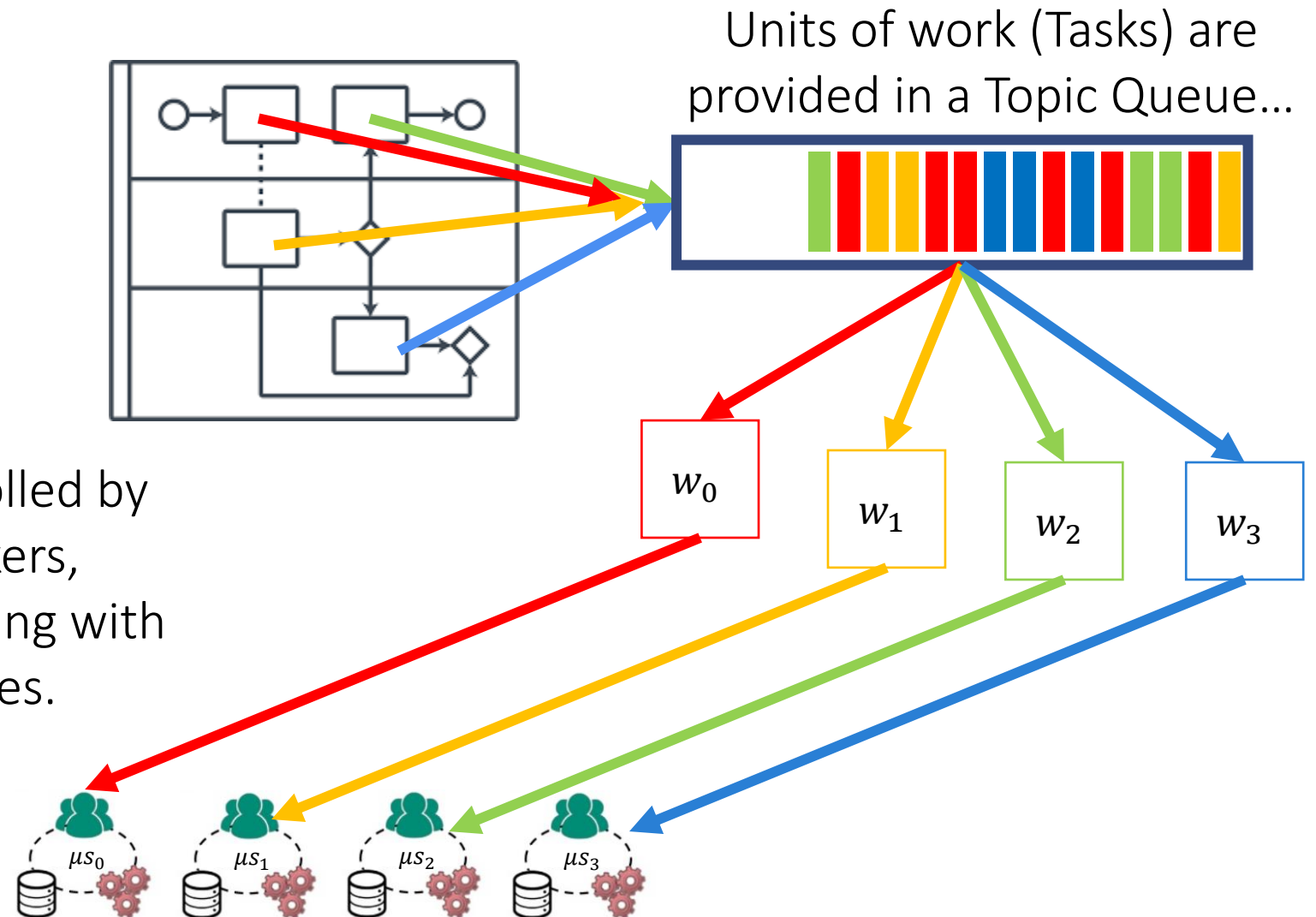
However, it only allows
scaling on process instances,
NOT on microservices.



How does it work? (Pattern B)

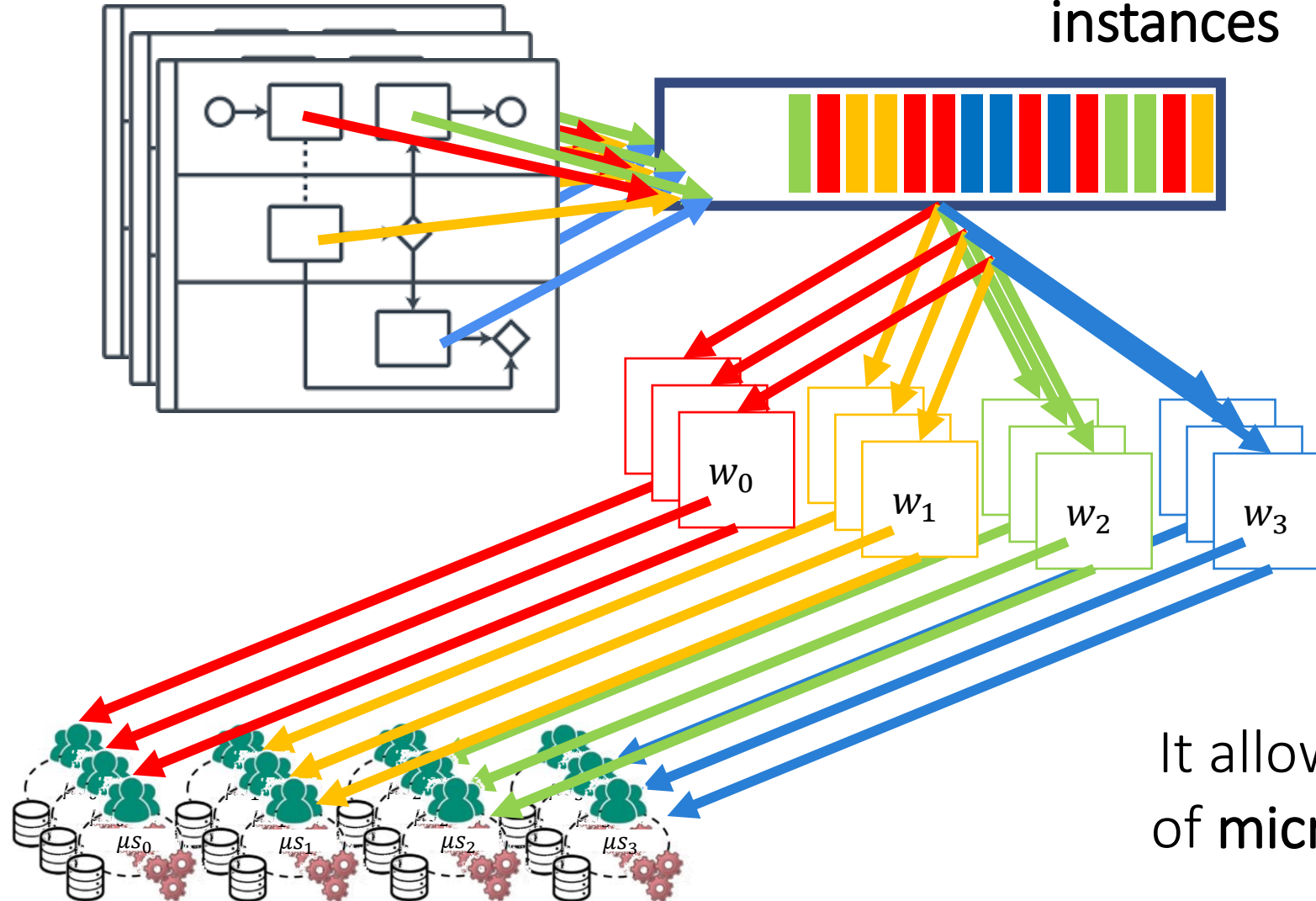
A more interesting pattern is known as *External Task*.

...that can be polled by RESTful workers, possibly interacting with microservices.



Scaling (Pattern B)

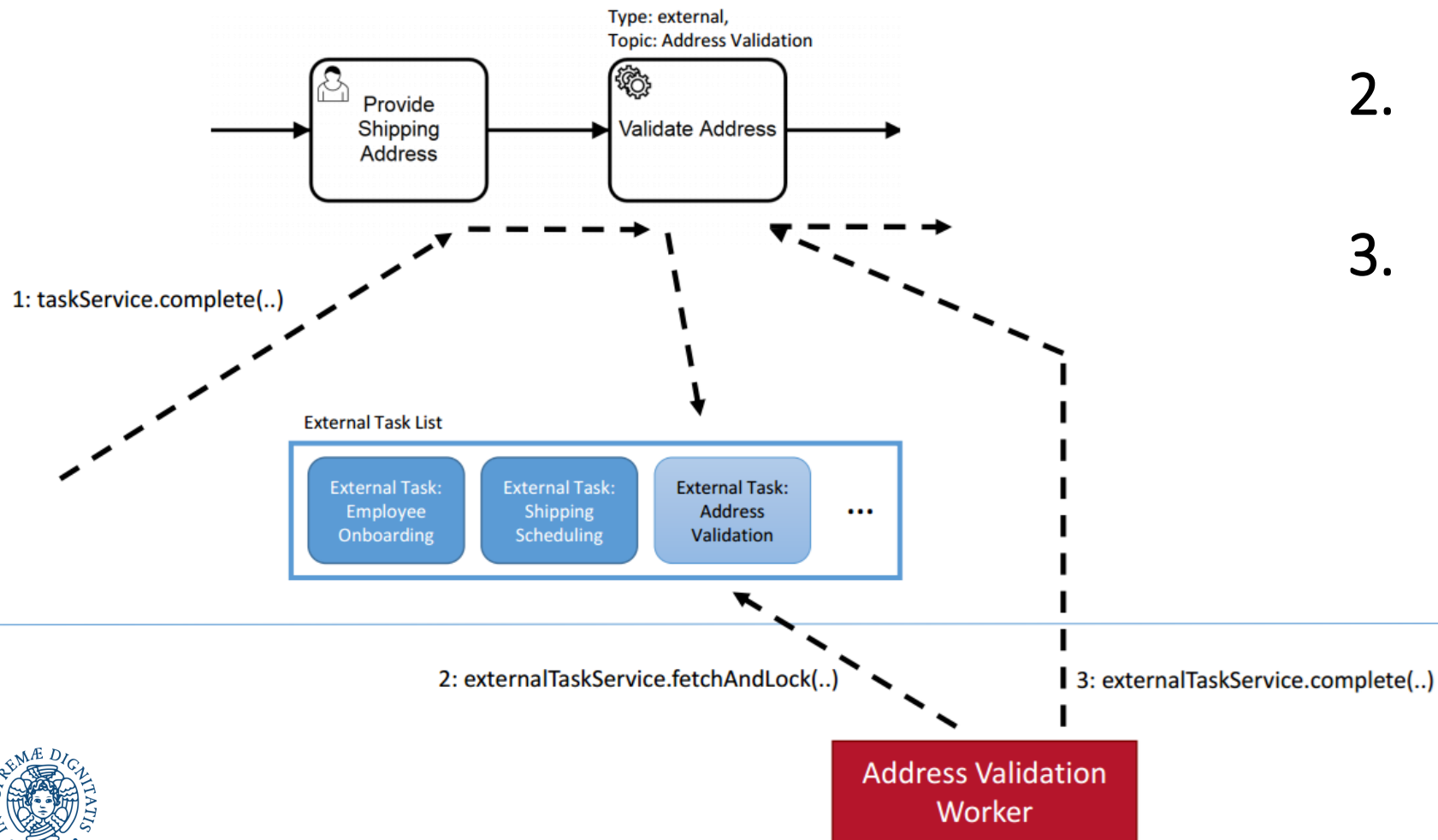
It allows scaling
of **process**
instances



It allows scaling
of **workers**

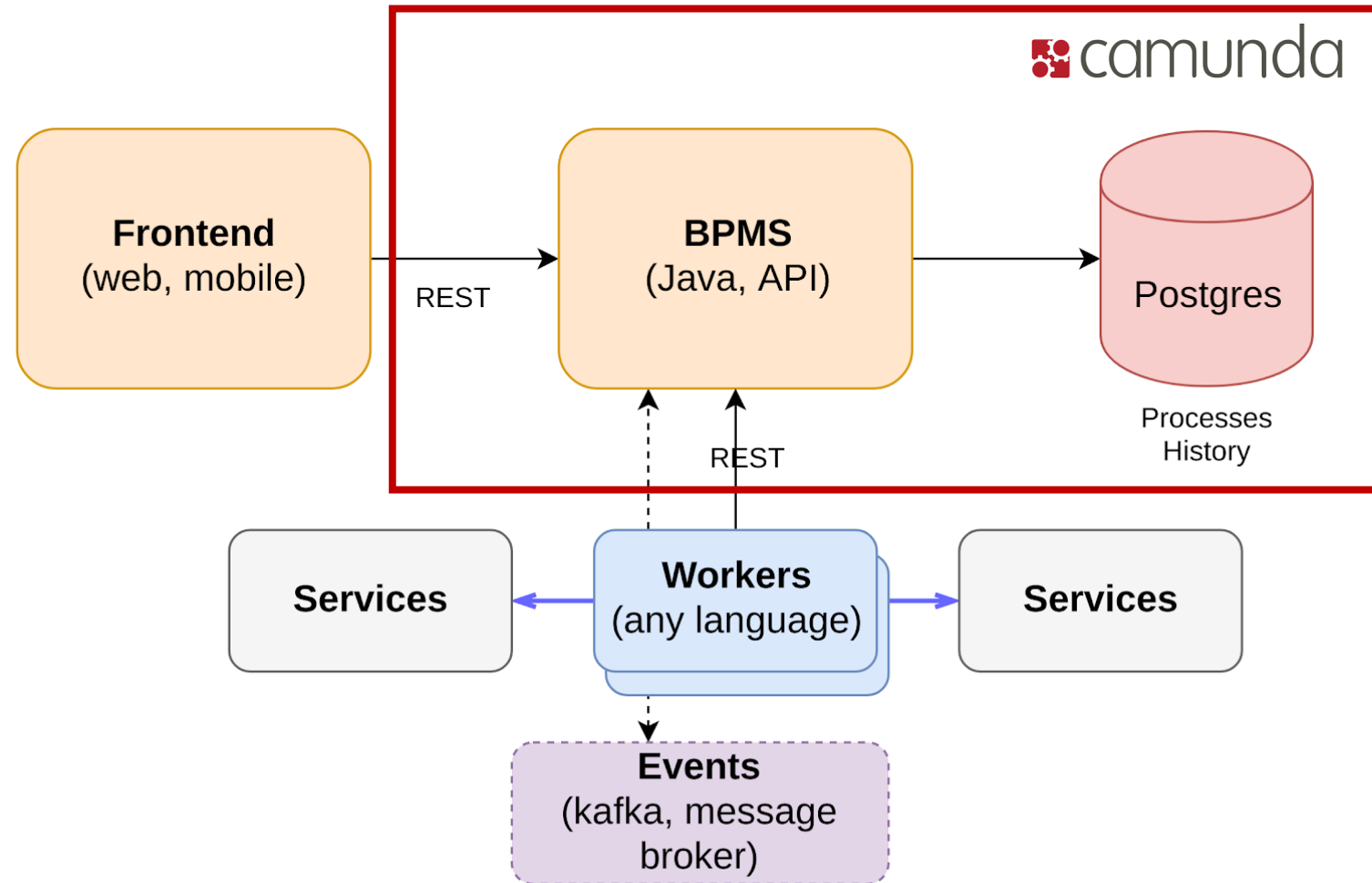
It allows scaling
of **microservices**

Step-by-step



1. **Process Engine:**
Creation of an external task instance
2. **External Worker:** Fetch and lock external tasks
3. **External Worker & Process Engine:**
Complete external task instance

Software with Camunda



camunda in Docker

- We can use Docker to run **Camunda BPM Platform**:

First time:

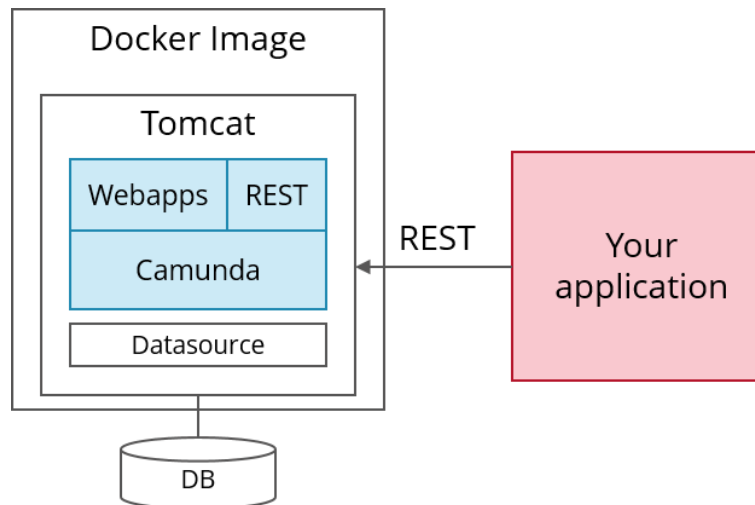
```
docker run -d --name camunda -p 8080:8080 camunda/camunda-bpm-platform:latest
```

From the second time:

```
docker start camunda
```



- Browse **127.0.0.1:8080/camunda** and enter credentials **demo demo**.



Three Menu Entry

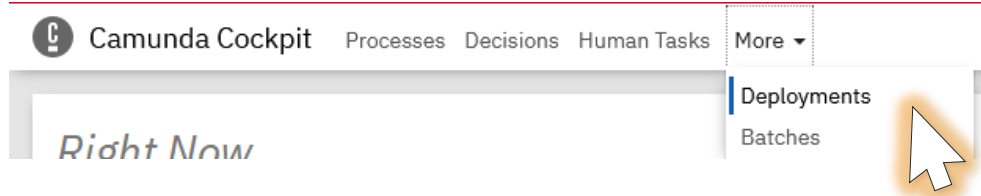
- Cockpit: to check process, instances and depolyment
- Tasklist: to check list of manual tasks
- Admin: admin stuff, we don not need it



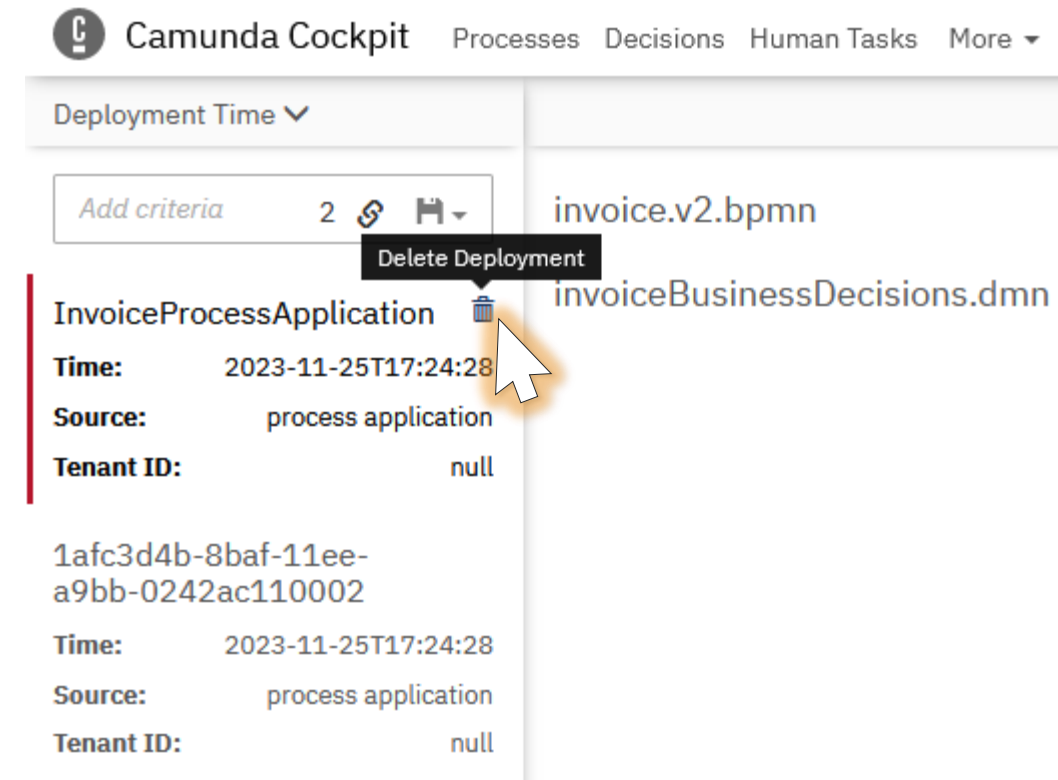
Click on the home icon top-right to switch between them
(only when you are not on the main menu)

Erase default deployments and process

1. From the Cockpit, open the deployments



2. Delete all the the deployments



3. Tick on 'cascade' and confirm

Today's Lab



Today's Lab

Download the .zip file from the Moodle

LAB TODO

1. Create the BPMN process for a legion campaign against an enemy capital.
2. Complete **fight.py**, **capital.py** and **result.py** workers.
3. Deploy and run the process with Camunda.



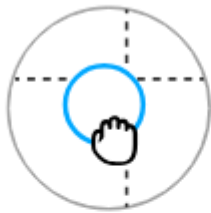
Today's Lab



Camunda Modeler

- Install the Camunda Modeler from <https://camunda.com/download/modeler/>

- To draw the process, choose Camunda 7



Model

Create BPMN workflow diagrams and DMN decision tables in an editor that both business users and developers love to use.



Execute

Execute your workflows and decisions in powerful engines that are paired with essential applications for process automation projects.



Enjoy

Never fear Business Process Management again as you will love Camunda. If you find that hard to believe, you should just give it a try.

Our business process

1. Setup information.
 2. Battle outside the capital.
- If the legion wins the battle:
3. Attack the capital!
 4. In the end: see the result of the campaign.

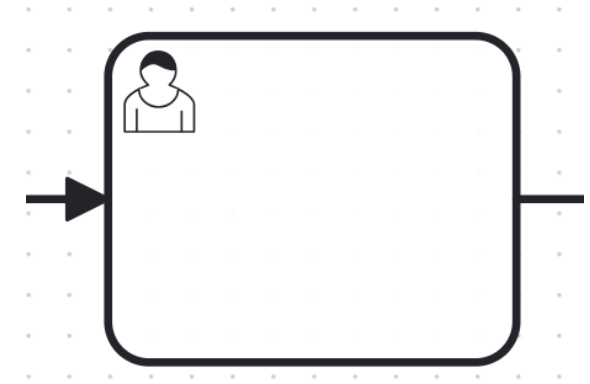


Setup information

Create a User task with a form asking for:

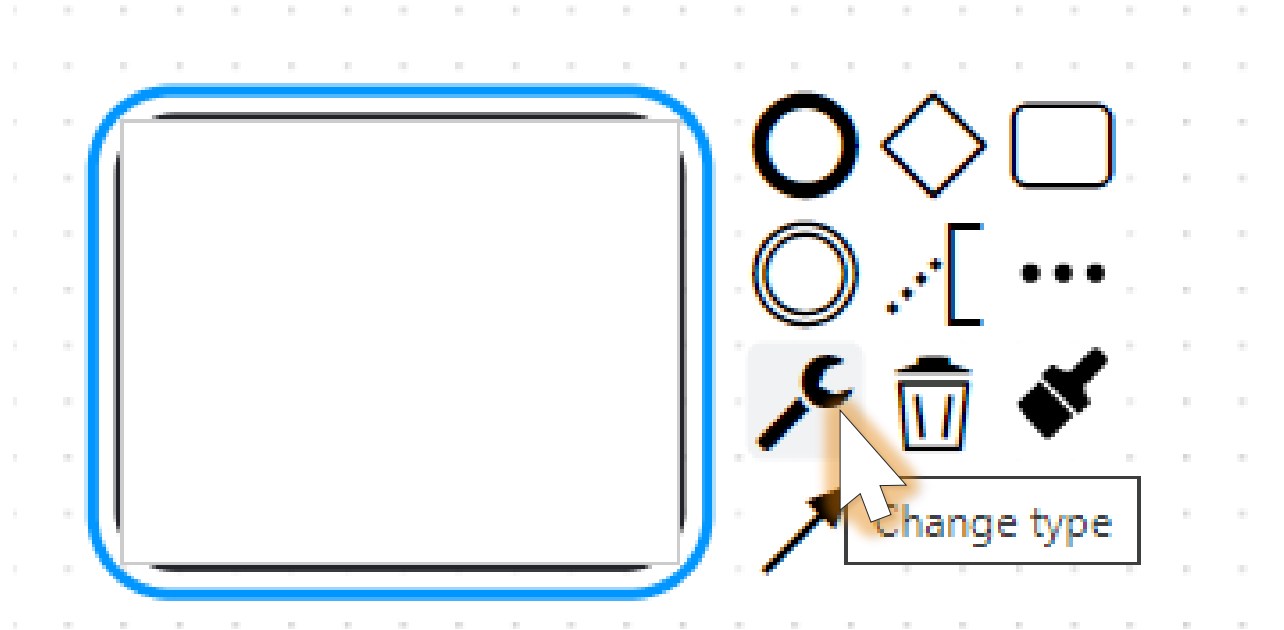
- The number of men of the legion
- Its strength value (number between 0-100)
- The number of defendants outside the capital
- Their strength value (number between 0-100)
- The number of defendants in the capital
- Their strength value (number between 0-100)

Choose meaningful variable names.



Change element's type

- After drawing an element



- Click it and using the wrench icon you can change its type

Forms

In the panel on the right:

- Click on + and add a variable with
 - ID - `legionSize`
 - Label - How many men compose the legion?
 - Type - `long`
 - Default Value - `100`

Forms

Form fields + 6

▼ `legionSize`

ID

`legionSize`

Refers to the process variable name

Label

How many men compose the legion?

Type

`long` ▼

Default value

100

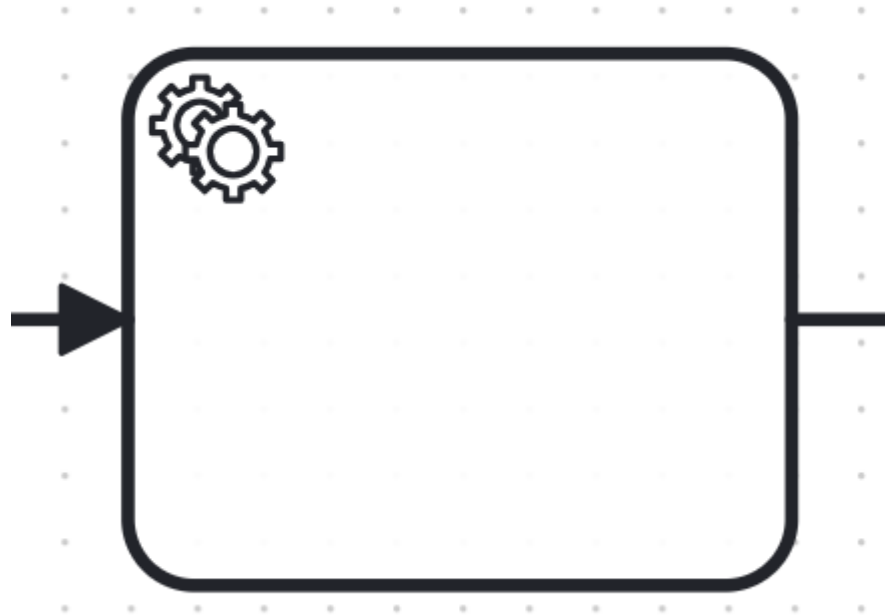
● Constraints +

● Properties +

Battle outside the capital

Create a Service task

- Type: External
- Topic: choose a meaningful name

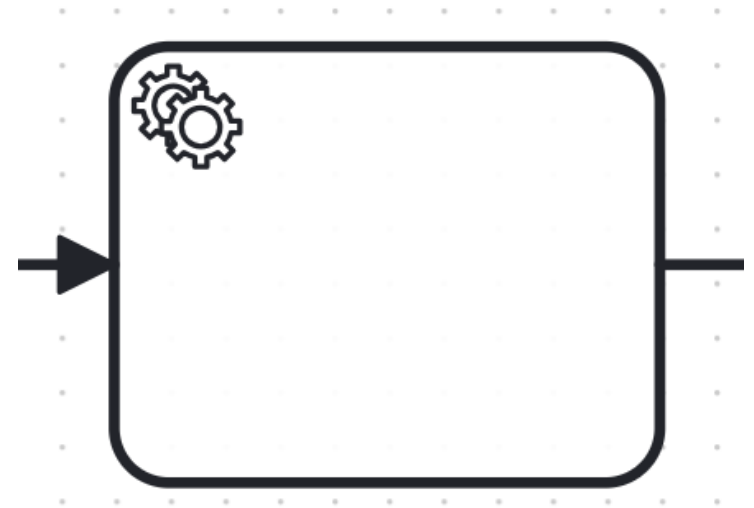


Attack the capital

If the legion survivors are more than half of the capital defendants, the legion attacks the capital.

Like before, create a Service task

- Type: External
- Topic: choose a meaningful name

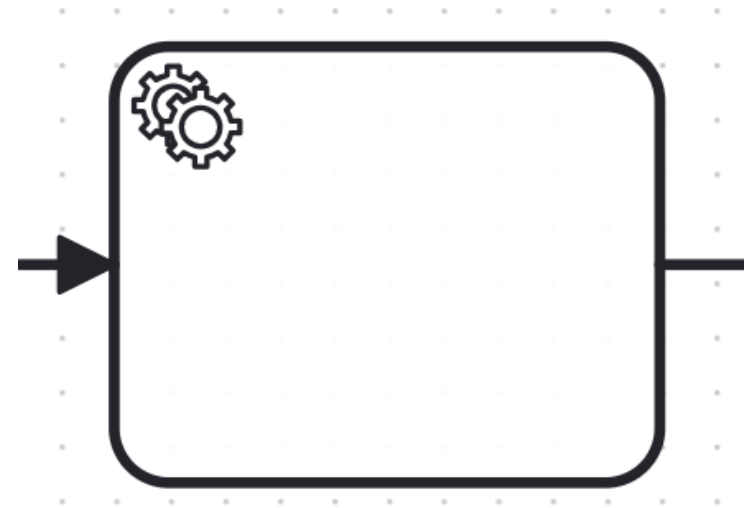


Result

Either the legion lost the first battle or attacked the capital, we ended up collecting the result.

Like before, create a Service task

- Type: External
- Topic: choose a meaningful name



Deploy

- Give the whole process a name by clicking in any empty part of the window and completing the Properties of the process.

General



Id
RomanLegions ✕
This maps to the process definition key.

Name
Roman Legion ✎

Version Tag

☒ Executable

- **Save** and click on **Deploy Current Diagram**



...

Deploy diagram


Deployment name
romanEmpire

Tenant ID
Optional

REST endpoint
http://localhost:8080/engine-rest

Include additional files +

Deploy

XML Camunda 7.20 

External Tasks

- They exploit the REST API of Camunda
[<https://docs.camunda.org/manual/7.9/reference/rest/external-task/>]
- Particularly:

Fetch and Lock

`POST /external-task/fetchAndLock`

Ask for a task to compute (and lock it to prevent other worker from doing the job)

Complete

`POST /external-task/{id}/complete`

Tell that a task has been completed to Camunda 😊

Service task implementation

Implement the service tasks with python programs.

- Battle outside the capital: **battle.py**
- Capital attack: **capital.py**
- Final result: **result.py**

With the library **pycamunda** REST calls are transparent.

Service task implementation

We supply also the Worker class to improve the worker's life cycle:

1. Worker object creation
2. Worker subscribe to the topic(s)
3. Worker run

Service task implementation

You have to:

- Complete the code by matching topics and variable names with the BPMN diagram (substitute the comments).
- Implement the fight logic of the two battles.
- Return the correct variables to the process.

You can add prints to understand what is happening (you will see them in the terminal executing each service task)

Fight logic

$$\text{Sur} = \frac{L \cdot Ls \cdot Rs - D \cdot Ds \cdot Rd}{Ls}$$

In each battle, you have:

- Legion number (L) and strength (Ls)
 - Defendants number (Ds) and strength (Ds)
1. Multiply each side number and strength.
 2. Multiply also for a random value between 0 and 1 (Rs and Rd)
 3. Do: legion value minus defendants value
 4. Divide by the legion strength to obtain the legion survivors (Sur)

First battle

The legion wins the battle if the survivors are more than half the capital defendants.

If the legion won the battle, add a third of the survivors to the legion as reinforcement.

The Python function handler should return a dictionary (string, integer) with

- the final value of the survivors and
- the result of the battle (0 if lost and 1 if won). The result value should be used in the BPMN decision to attack the capital.

Second battle

The legion attacks with the survivors (and reinforcements) of the previous one.

The capital is conquered if the legion has survivors.

The Python function handler should return a dictionary with the type as before having only the survivor number.

Run the business process


- Launch the python programs in three different terminals

```
python3 <file_name>.py
```

- Click on the 'play' symbol in Camunda Modeler and start the service instance.
- Go to the Tasklist page of Camunda and under 'All tasks' you should find your form.
- Claim the task, fill out the form to launch the battle and go to the terminal of the result to see how the campaign went.

Some adviced values for the form

 Set follow-up date

 Set due date

 Add groups

 Claim

Form

History

Diagram

Description

How big is the legion?

100

How big is the defendant army?

80

How many defenders in the capital?

50

How much is strong the legion?

100

How much are strong the defendants?

40

How much is strong the capital defence?

60

Save

Complete



Lab take away

- ☐ Learn the relation between business processes and microservices.
- ☐ Draw a non-trivial BPMN diagram.
- ☐ Understand and implement external services.
- ☐ Deploy and run a business process with Camunda.

