

Gain  
More Freedom  
When  
Migrating  
From Camunda  
Platform  
7 to 8

Stephan Pelikan



Photo by [Alex Radelich](#) on Unsplash

**CAMUNDA**

Platinum  
Partner  
CERTIFIED



Wien

 **Phactum**  
creating solutions



Stephan Pelikan  
senior\_developer@phactum



**7.x**

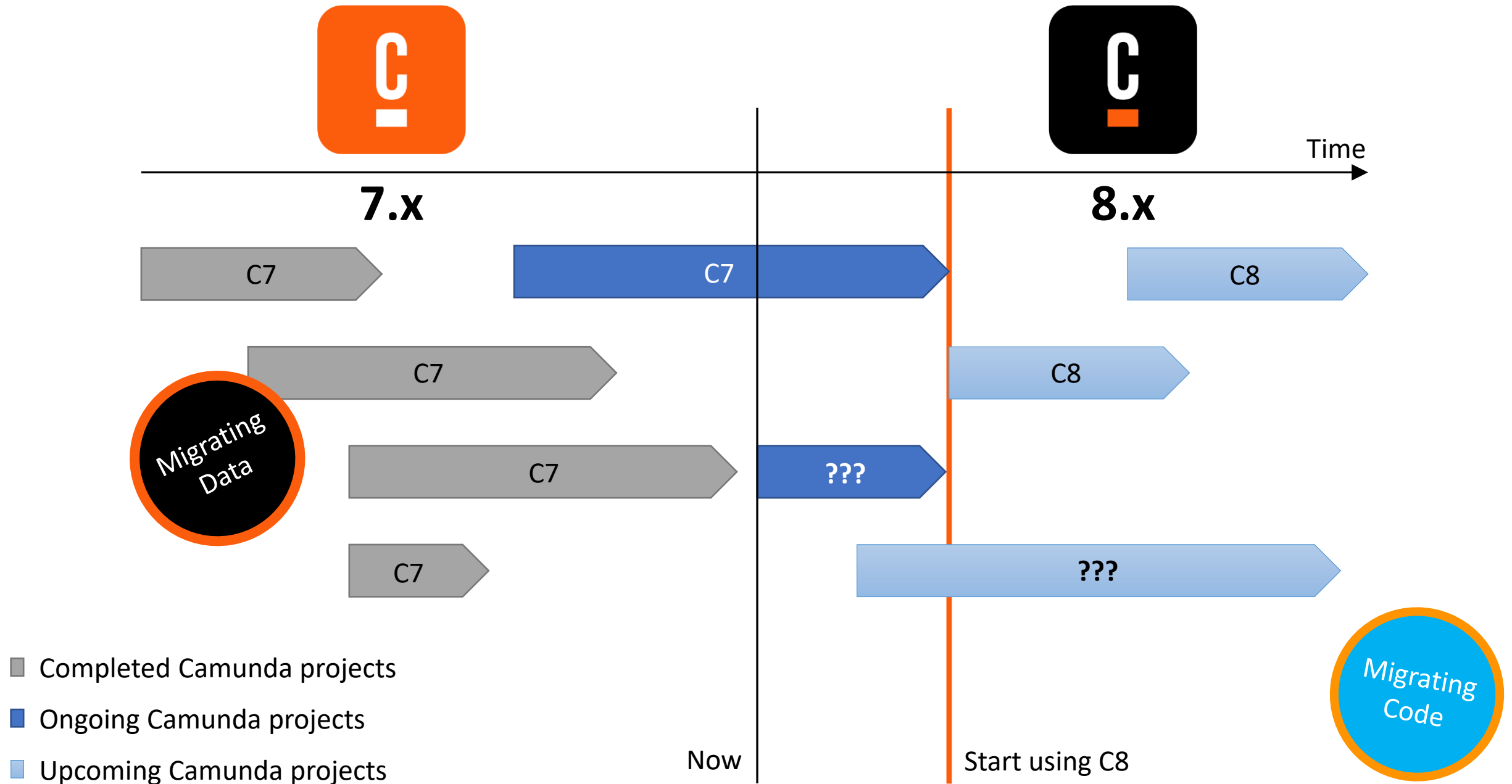


**8.x**



- **Embedded engine**
- **Various APIs to implement tasks**  
Java native, External Task (polling)
- **One TX for business code and Camunda**
- **Process variables**
- **JUEL**
- **etc.**

- **Separate service**
- **One API for tasks**  
Worker (push)
- **Eventual consistency**
- **JSON data object**
- **FEEL**
- **etc.**



- Completed Camunda projects
- Ongoing Camunda projects
- Upcoming Camunda projects

Now

Start using C8

Time

7.x

8.x

C7

C7

C8

C7

C8

C7

???

C7

???

Migrating Data

Migrating Code

Getting started

Introduction to Camunda Platform 8

Create an account

Model your first process

Getting started with human task orchestration

Getting started with API endpoints

Getting started with microservice orchestration

Next steps

Update guide

Migrating diagrams from Cawemo

Migrating from Camunda Platform 7

## Prepare for smooth migrations

Sometimes you might not be able to use Camunda 8 right away as described in [What to do When You Can't Quickly Migrate to Camunda 8](#). In this case, you will keep developing Camunda Platform 7 process solutions, but you should establish some practices as quickly as possible to ease migration projects later on.

To implement Camunda Platform 7 process solutions that can be easily migrated, stick to the following rules and development practices:

1. Implement what we call **Clean Delegates** - concentrate on reading and writing process variables, plus business logic delegation. Data transformations will be mostly done as part of your delegate (and especially not as listeners, as mentioned below). Separate your actual business logic from the delegates and all Camunda APIs. Avoid accessing the BPMN model and invoking Camunda APIs within your delegates.
2. Don't use listeners or Spring beans in expressions to do data transformations via Java code.
3. Don't rely on an ACID transaction manager spanning multiple steps or resources.
4. Don't expose Camunda API (REST or Java) to other services or front-end applications.
5. Use primitive variable types or JSON payloads only (no XML or serialized Java objects).
6. Use simple expressions or plug-in FEEL. FEEL is the only supported expression language in Camunda 8. JSONPath is also relatively easy to translate to FEEL. Avoid using special variables in expressions, e.g. `execution` or `task`.
7. Use your own user interface or Camunda Forms; the other form mechanisms are not supported out-of-the-box in Camunda 8.
8. Avoid using any implementation classes from Camunda; generally, those with `*.impl.*` in their package name.
9. Avoid using engine plugins.

We also recommend reviewing [BPMN elements supported in Camunda 8](#), though any feature gap will likely be closed soon.

Camunda Platform 7 vs. Camunda Platform 8

Conceptual differences

Process solutions using Spring Boot

Programming model

Platform deployment

Other process solution architectures

Plugins

Migration overview

When to migrate?

Migration steps

Migration tooling

Adjusting your source code

Client API

Service tasks with attached Java code (Java Delegates, Expressions)

Service tasks as external tasks

Adjusting Your BPMN models

Service tasks

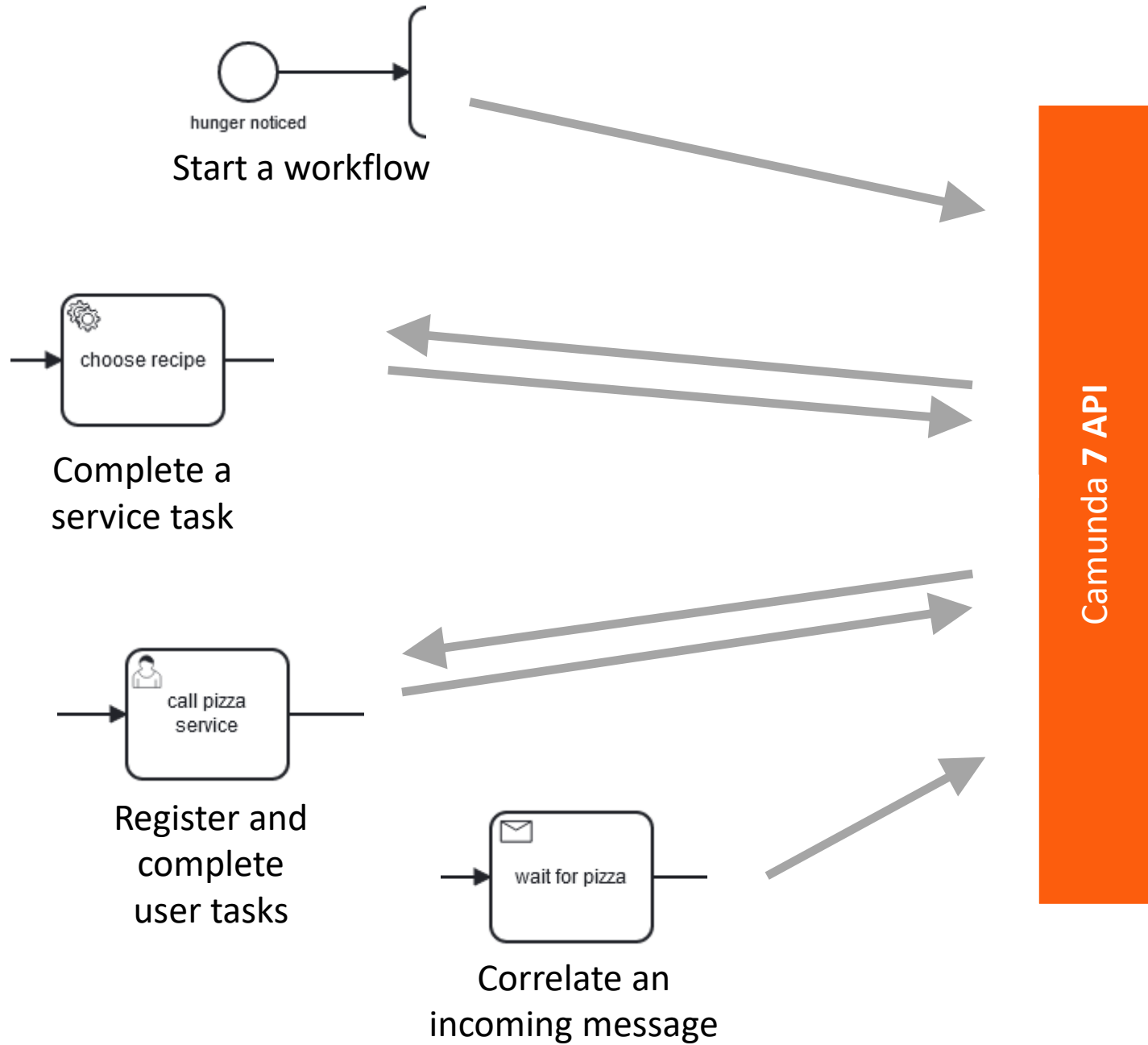
Send tasks

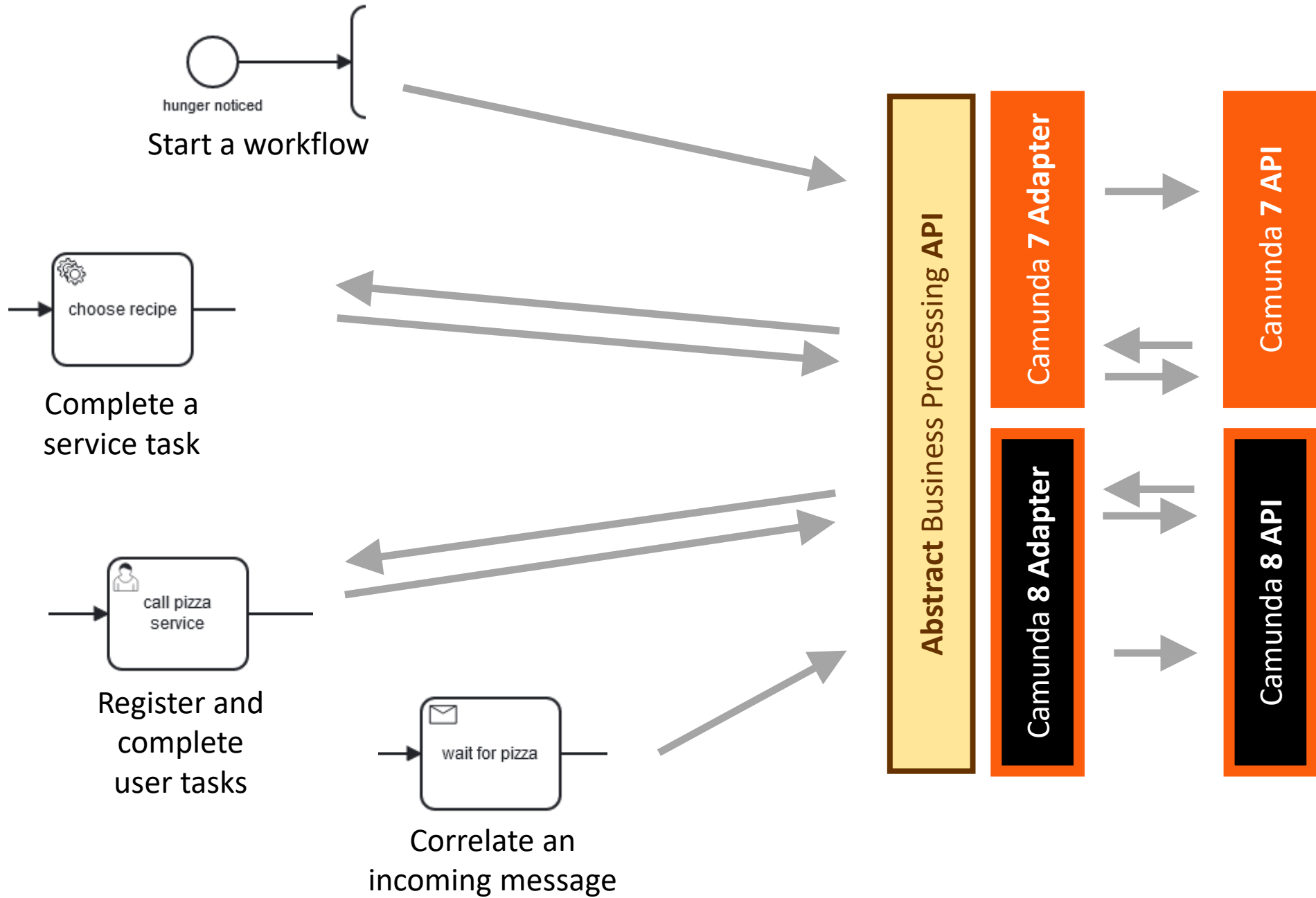
Gateways

Expressions

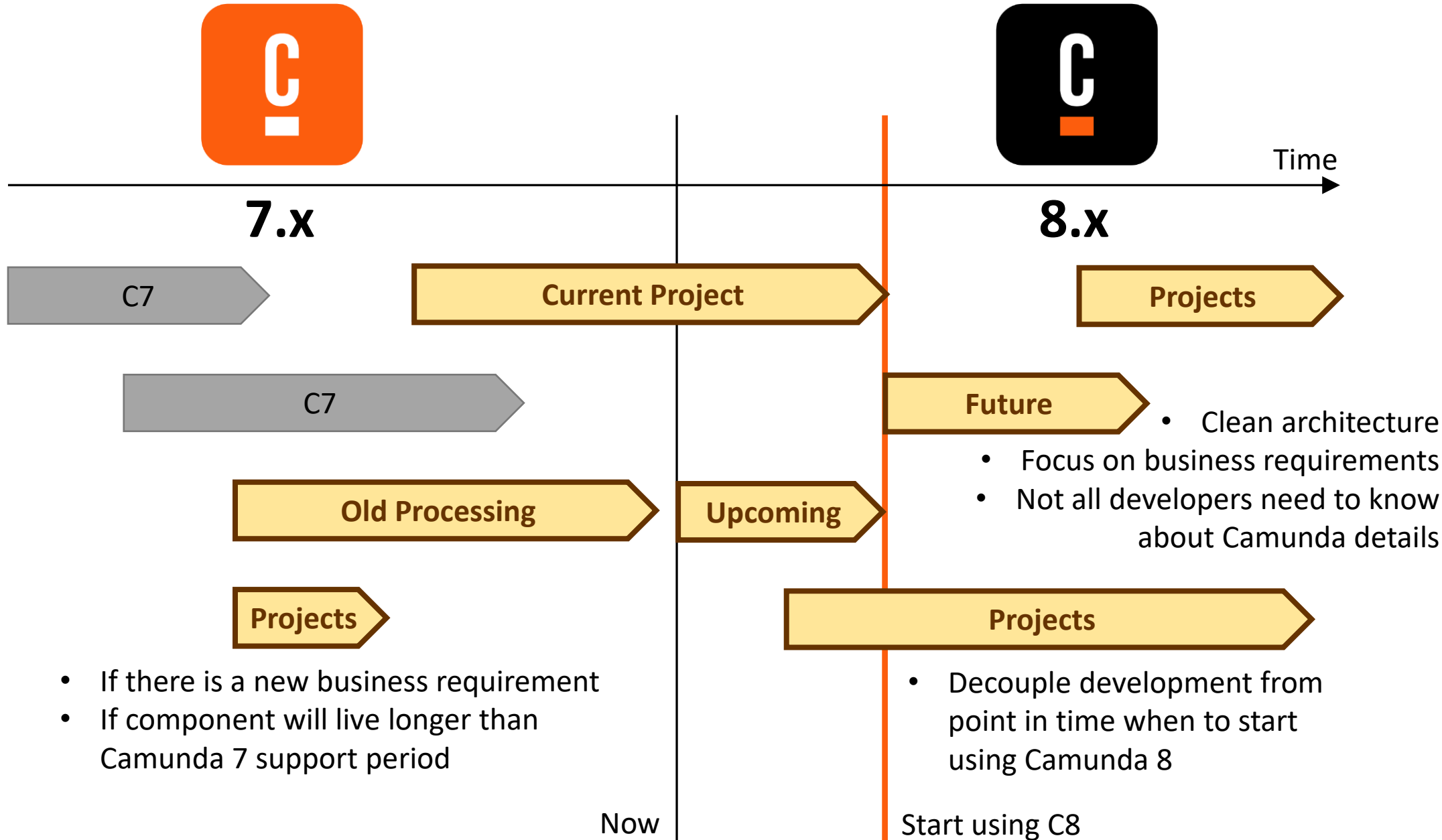
Human tasks

Business rule tasks





Abstract Business Processing API



- If there is a new business requirement
- If component will live longer than Camunda 7 support period

- Decouple development from point in time when to start using Camunda 8



# Cons / Pros

Burden to implement  
(especially for Camunda beginners)

Only features available in C7 & C8

Maintenance of adapters necessary

Not a „no change“ approach

Clean architecture

Decouple development

Centralized maintenance for  
multiple projects

Simplify your software  
& scale your teams easier

# Cons / Pros

Burden to implement  
(especially for Camunda beginners)

Only features available in C7 & C8

Maintenance of adapters necessary

Not a „no change“ approach

Clean architecture

Decouple development

Centralized maintenance for  
multiple projects

Simplify your software  
& scale your teams easier

we did a little for you

README.md

Community Extension An open source community maintained project Compatible with Camunda Platform

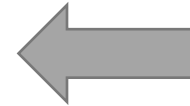
maven central unknown Lifecycle Incubating license Apache V.2

 **VanillaBP**

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills

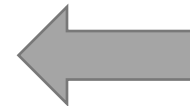
## pom.xml

```
<dependency>  
  <groupId>io.vanillabp</groupId>  
  <artifactId>spi-for-java</artifactId>  
</dependency>
```



Interfaces, Enums &  
Annotations

```
<dependency>  
  <groupId>org.camunda.community.vanillabp</groupId>  
  <artifactId>camunda7-spring-boot-adapter</artifactId>  
</dependency>
```



SPI Binding,  
BPMN deployment

```
<dependency>  
  <groupId>org.camunda.community.vanillabp</groupId>  
  <artifactId>camunda8-spring-boot-adapter</artifactId>  
</dependency>
```

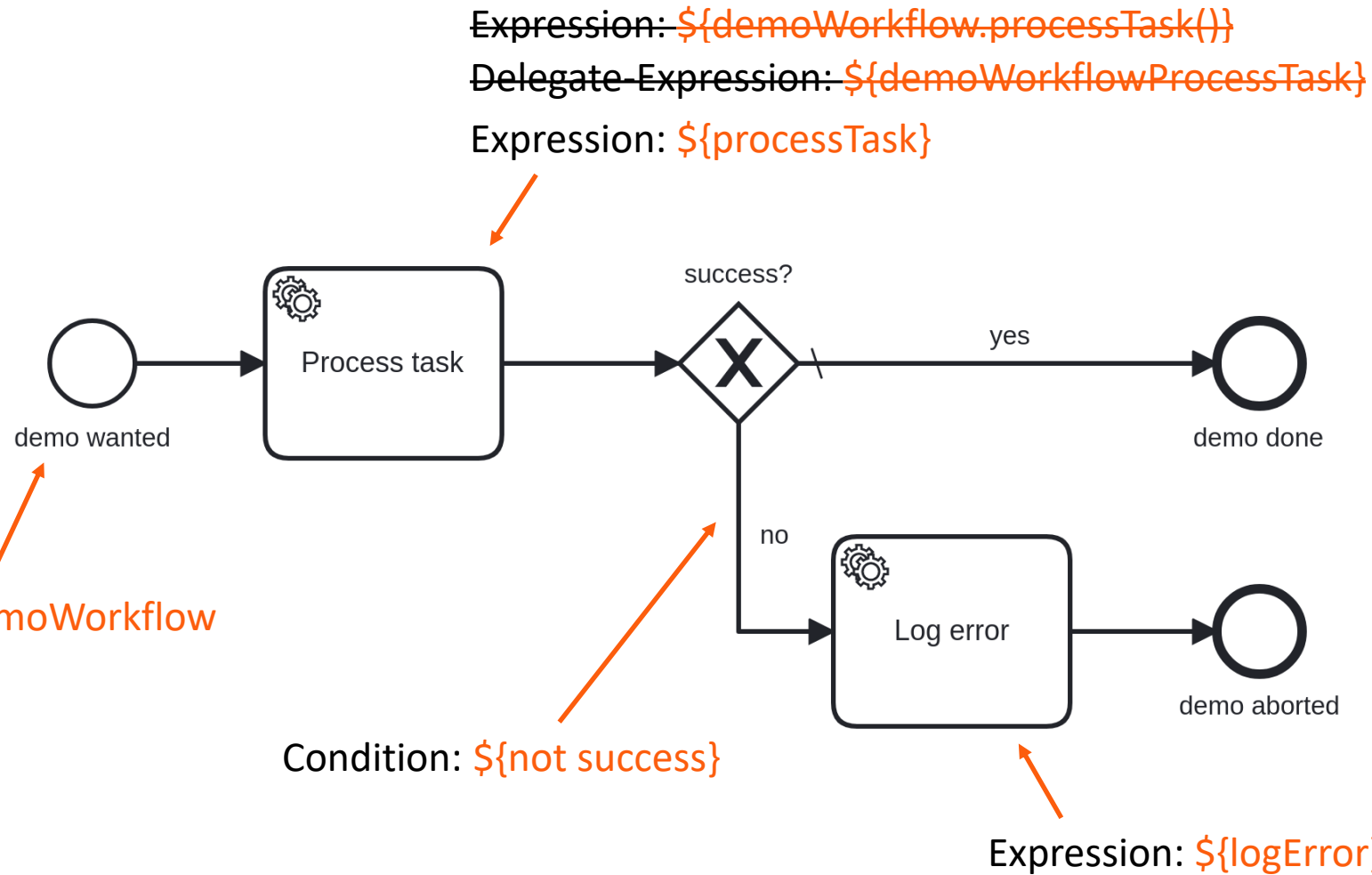


## VanillaBP

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills

c7/demo.bpmn

BPMN process ID: DemoWorkflow



**VanillaBP**

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills

ID: DemoWorkflow

start a workflow

Expression: \${processTask}

Expression: \${logError}

Condition: \${not success}

```
1 @workflowService(workflowAggregateClass = DemoAggregate.class)
2 @Service
3 public class DemoWorkflow {
4     @Autowired
5     private ProcessService<DemoAggregate> processService;
6
7     public void startDemo(String id) throws Exception {
8         var demo = new DemoAggregate(id, false);
9         processService.startWorkflow(demo);
10    }
11
12    @workflowTask
13    public void processTask(DemoAggregate demo) {
14        demo.setSuccess(true);
15    }
16
17    @workflowTask(taskDefinition = "logError")
18    public void logErrorOnFailure() {
19        logger.info("error");
20    }
21 }
22 }
```

```
1 @Entity @Table(name = "DEMO")
2 @Getter @Setter @AllArgsConstructor
3 public class DemoAggregate {
4     @Id @Column(name = "ID")
5     private String id;
6     @Column(name = "SUCCESS")
7     private boolean success;
8 }
```



**VanillaBP**

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills

# Validation on booting the application

Caused by: java.lang.IllegalStateException: **No Spring Data repository defined for demo.DemoAggregate**  
at io.vanillabp.springboot.utils.JpaSpringDataUtil.getRepository(JpaSpringDataUtil.java:62)

Caused by: java.lang.RuntimeException: **No public method annotated with @WorkflowTask is matching task having task-definition 'procesTask' of process 'DemoWorkflow'. Tested for:**  
at io.vanillabp.springboot.adapter.TaskWiringBase.wireTask(TaskWiringBase.java:199)

Caused by: java.lang.RuntimeException: **You need to autowire 'io.vanillabp.spi.process.ProcessService<demo.DemoAggregate>' in your code to be able to start workflows!**  
at io.vanillabp.camunda7.wiring.Camunda7TaskWiring.lambda\$1(Camunda7TaskWiring.java:81)

## DemoAggregateRepository.java

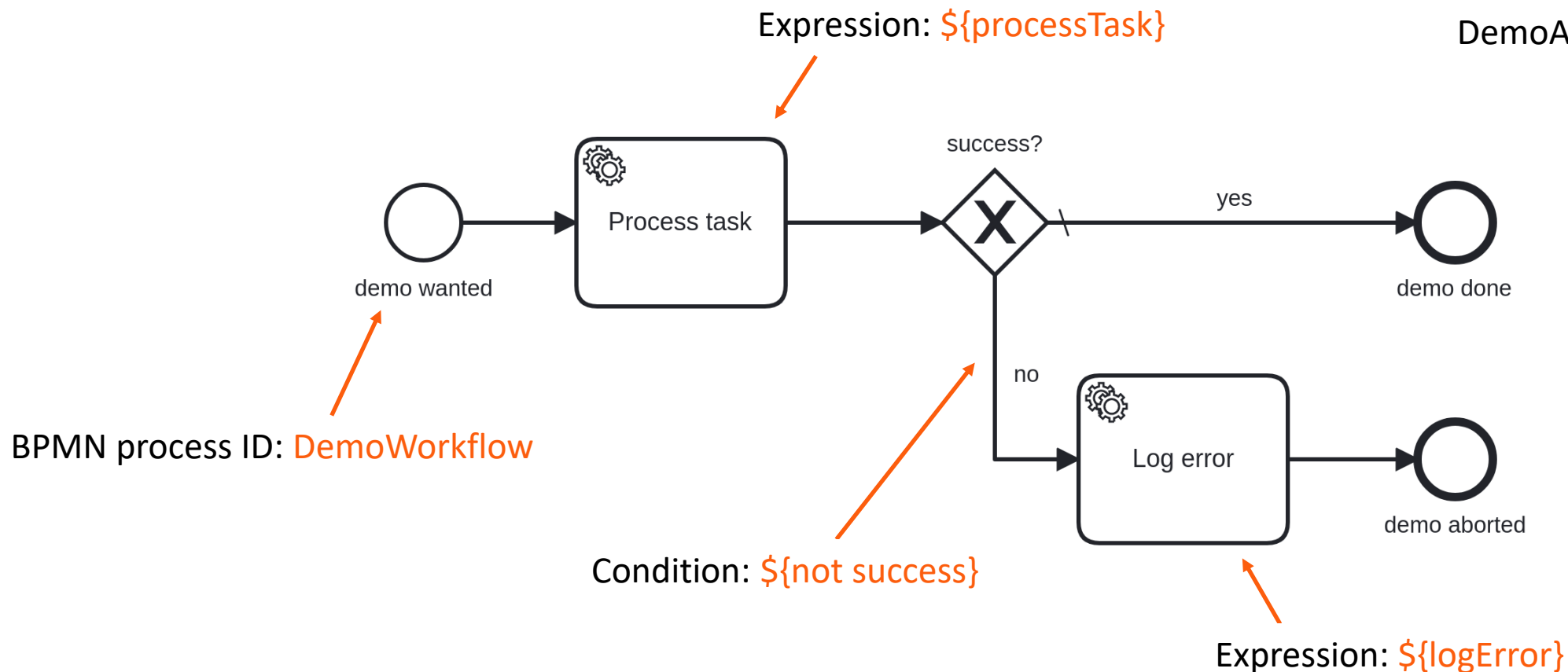
```
1 @Repository
2 public interface DemoAggregateRepository
3     extends JpaRepository<DemoAggregate, String> {
4
5 }
```



**VanillaBP**

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills

c8/demo.bpmn



Add directory „c8/“ for new BPMNs and define in Spring config

Switch the Maven dependency to „camunda8-spring-boot-adapter“



**VanillaBP**

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills

# Live demo

<https://github.com/vanillabp/simple-vanillabp-demo>

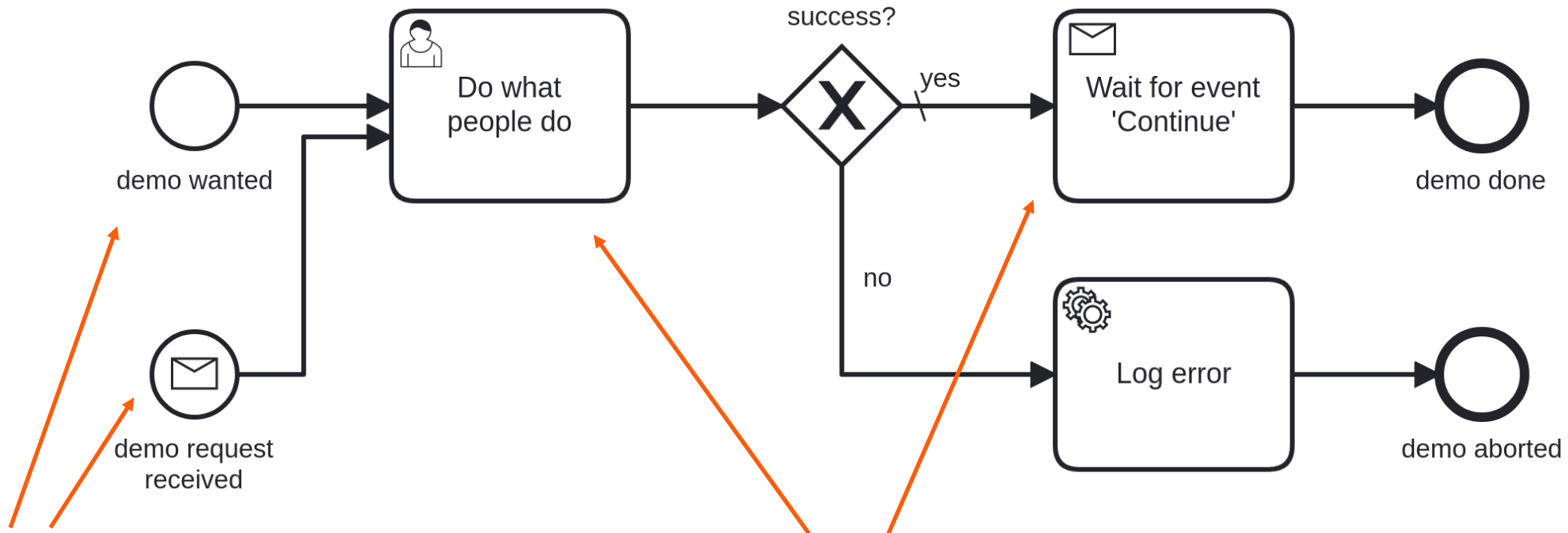


# Live demo

<https://github.com/vanillabp/simple-vanillabp-demo>

## Migration scenario showcase


- Start **new workflow** instances using **C8**
- **Keep C7 workflow instances** until completed by running into an end-event
- No code-change, only by changing configuration



Start workflows:  
Using the configured target

Complete task /Correlate message:  
Dynamically depending on the target  
platform of the workflow instance

# VanillaBP Abstraction

Wiring of BPMN and services  

Service-like tasks  

Asynchronous tasks  

Multi-instance tasks  

Deployment of BPMN  

  Process variables

  Receive-like tasks

  User tasks

  BPMN errors

  BPMN process versions

run C7 processes or C8 processes  
using the same code ✓

run C7 processes next to C8 processes ✓

Support async tasks for C7  
based on external tasks

run C7 processes instances next to C8 processes  
instances of the same workflow ✓

BPMN version-specific tasks

JakartaEE 

## VanillaBP simplifies

- business processing code
- migration from C7 to C8
- BPMN (C7)

**Focus on your business**

„The taste of BPMN should be as subtle as Vanilla and not detract from the flavor of your business.“





# VanillaBP

Business processing at its best taste  
[va·nil·la] ... simple, plain, no frills



<https://www.vanillabp.io>

<https://github.com/vanillabp/spi-for-java>

comprehensive  
documentation

<https://github.com/camunda-community-hub/vanillabp-camunda7-adapter>



<https://github.com/camunda-community-hub/vanillabp-camunda8-adapter>



<https://github.com/vanillabp/simple-vanillabp-demo>