

Camunda Evolution – Versions 7 vs 8

Overview: Why Two Platforms?

Camunda 7 served as a critical backbone for workflow orchestration for many organizations over the past decade. However, evolving market demands have necessitated a complete reimagining of the platform architecture to address:

- Real-time orchestration requirements
- Hybrid workflow capabilities
- AI-infused business logic
- Cloud-native deployment scenarios
- Horizontal scalability challenges
- Complex microservices orchestration

Camunda 8 represents a fundamental architectural shift addressing these modern challenges while introducing new capabilities such as agentic orchestration, native AI integrations, and flexible connector ecosystems.

Key Architectural Differences

Engine Deployment Model

Camunda 7:

- Embedded engine as a library within your application
- Both engine and application run in the same JVM
- Share thread pools, data sources, and transaction managers
- Tight coupling between engine and application lifecycle
- Traditional monolithic or Spring Boot embedded deployment

- Scaling the application means scaling multiple engine instances

Camunda 8:

- Remote engine architecture (Zeebe as microservice)
- Engine runs as a separate, independently scalable service
- Applications communicate with engine via gRPC or REST APIs
- Stateless worker pattern for job execution
- Cloud-native distributed deployment
- Horizontal scaling of workers and brokers independently

Impact: The transition from embedded to remote represents a fundamental shift from traditional application architecture to cloud-native distributed systems thinking.

Data Storage and Management

Camunda 7:

- Stores diverse data types including serialized Java objects
- Uses Camunda Spin for XML and JSON transformations
- Relational database dependency (H2, PostgreSQL, Oracle, MySQL)
- Transactional consistency between application and engine
- Tight ACID transaction management

Camunda 8:

- Supports only primary data types and JSON as process variables
- No relational database requirement
- Event stream storage with Elasticsearch for secondary storage
- Data mapping handled by application code (Jackson, custom libraries)
- Eventual consistency model

- Distributed event-driven architecture

Impact: Applications must implement their own data transformation logic; serialized Java objects cannot be stored directly in process variables.

Transaction Management

Camunda 7:

- Can share technical ACID transactions between application and workflow engine
- Ensures consistency with single transaction context
- Failure handling through transaction rollback

Camunda 8:

- Cannot share ACID transactions across engine boundary
- Requires achieving consistency without transaction managers
- Compensation logic and idempotent operations essential
- Event-driven consistency patterns

Impact: Process designs must implement explicit compensation logic and idempotent operations; developers must design for eventual consistency.

Comparison Matrix: Camunda 7 vs Camunda 8

Aspect	Camunda 7	Camunda 8
Engine Type	Embedded library	Remote microservice (Zeebe)
Deployment	Spring Boot / App Server	Docker / Kubernetes
Data Types	Serialized Java objects, primitives	Primitives, JSON only
Database	Required (H2/PostgreSQL/Oracle)	Event stream (Elasticsearch)
Monitoring Tool	Cockpit (coupled deployment)	Operate (separate service)
Task List	Built-in Tasklist (coupled)	Separate Tasklist component
Process Definition	XML-based BPMN	BPMN 2.0 (same format)
API Access	Java API, REST API	REST API, gRPC API
Multi-tenancy	Engine-per-tenant pattern	Built-in tenant isolation
Connectors	Camunda Connect (HTTP/SOAP)	Extensible Connectors runtime
Scaling	Horizontal scaling multiplies engines	Independent worker/broker scaling
AI Integration	Limited, custom implementation	Native AI connectors
Cloud Native	Not optimized	Purpose-built
Learning Curve	Moderate	Steeper (distributed systems concepts)
Community	Mature ecosystem	Growing ecosystem
Production Support	Full support available	Support with appropriate licensing

When to Choose Camunda 7 vs Camunda 8

Choose Camunda 7 When:

1. Existing Deployments: Your organization has established Camunda 7 implementations with significant process definitions
2. Monolithic Architecture: Traditional monolithic applications with embedded engine preference
3. Java-Centric Solutions: Heavy reliance on Java-specific features, serialized objects, and embedded transactions
4. Small-Scale Deployments: Limited scale requirements and simple orchestration needs
5. Team Expertise: Your team has deep Camunda 7 knowledge and minimal distributed systems experience
6. Cost Considerations: Open-source self-hosted deployment with free Cockpit/Tasklist in production
7. Stability Requirement: Need battle-tested, mature platform with minimal unknown risks
8. Simple Human Workflows: Basic user task workflows without complex multi-tenant requirements

Choose Camunda 8 When:

1. Greenfield Projects: New implementations designed with cloud-native architecture
2. Microservices Architecture: Complex microservices orchestration across organizational boundaries
3. High-Volume Processing: Requiring horizontal scaling and high throughput
4. Cloud Deployment: AWS, Azure, GCP, or hybrid cloud infrastructure
5. Real-Time Requirements: Need for low-latency process execution and immediate responses
6. Kubernetes Native: Containerized deployments managed by Kubernetes orchestration
7. AI Integration: Requirement for AI-powered decision making and process intelligence

8. Multi-Tenant SaaS: Applications serving multiple independent customers with strict isolation
9. Event-Driven Architecture: Event streams from Kafka, message queues, or other event sources
10. Independent Scaling: Need to scale different components (workers, brokers, UI) independently

Migration Considerations

Camunda 8 is not a drop-in replacement for Camunda 7. Successful migration requires:

- Complete process definition review and redesign
- Application refactoring for remote engine communication
- Data transformation logic implementation
- Eventual consistency pattern adoption
- Testing and validation of SLAs in new environment
- Process-by-process migration strategy
- Side-by-side operation during transition period

Camunda 7 Setup and Deployment

Environment Requirements

System Requirements:

- Java 17 or higher
- 4GB RAM minimum (8GB recommended)
- 5GB free disk space

Technology Stack:

1. Download Community version of Camunda camunda-bpm-tomcat-7.24.0.zip
2. Download JDK 17 or above.

Download the file either in ZIP or TAR format.

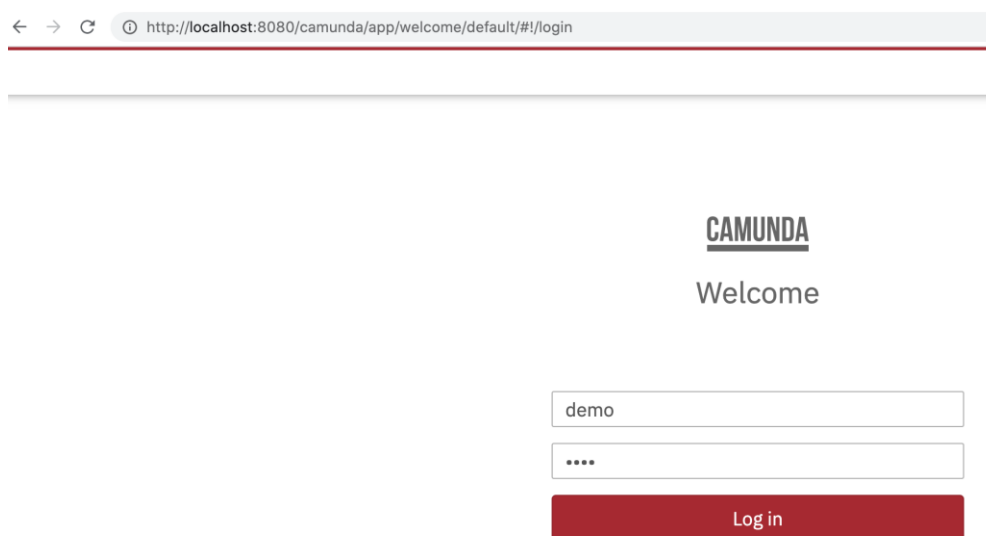
- Extract camunda-bpm-tomcat-7.24.0.zip
- Navigate to C:\CAMUNDA\camunda724 where camunda-bpm-tomcat-7.24.0.zip is extracted
- Start the camunda using command prompt.
- C:\CAMUNDA\camunda724\camunda-bpm-tomcat-7.24.0>start-camunda.bat

This camunda deployer is used to deploy the business processes. By default this package comes with the tomcat server on which the camunda deployer runs. The H2 database is used for the initial startup process. You can change it to different database as per your requirement.

Accessing Camunda

CAMUNDA APPLICATION CAN BE ACCESSED FROM

[HTTP://LOCALHOST:8080/CAMUNDA/](http://localhost:8080/CAMUNDA/)

A screenshot of a web browser showing the Camunda login page. The browser's address bar displays the URL 'http://localhost:8080/camunda/app/welcome/default/#!/login'. The page content includes the Camunda logo (the word 'CAMUNDA' in a bold, sans-serif font with a horizontal line underneath), followed by the word 'Welcome' in a smaller font. Below this, there are two input fields: the first contains the text 'demo' and the second contains four dots '....' representing a password. At the bottom of the form is a red rectangular button with the text 'Log in' in white.

Camunda welcome screen will look as shown in above image. Enter the default credentials as given below.

Username: demo

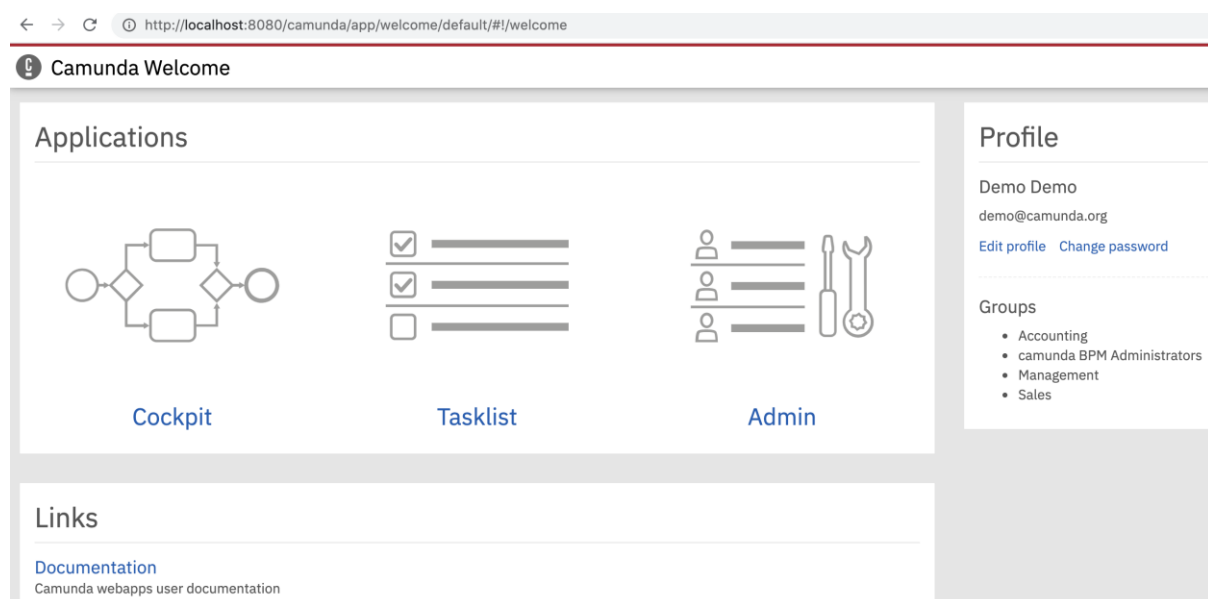
Password: demo

demo user has admin role by default.

Once you sign in to the app, you will see different tabs for performing various tasks to start with the business processes.

- Hit the below url in the browser

<http://localhost:8080/camunda/app/welcome/default/#/welcome>



Camunda is running on web server apache-tomcat-9.0.5.

Camunda 7 Components Overview

Cockpit (Process Monitoring)

Cockpit provides operational visibility into running processes:

Key Features:

- View active process instances
- Monitor execution progress
- Inspect process variables
- Access execution logs
- Incident management
- Batch operations on instances

Navigation Path: Cockpit → Processes → HelloWorldProcess

Information Available:

- Instance count (running, completed, failed)
- Activity history and timing
- Variable state at each step
- Error/incident details
- Task assignments

Tasklist (Human Tasks)

Tasklist is the interface for completing human-oriented work:

Key Features:

- Display assigned tasks
- Filter and search tasks
- Task forms and variables
- Task delegation
- Claiming and completion
- Priority and due date management

Accessing User Tasks:

- Requires User Task element in BPMN
- Task assignment via assignee attribute
- Forms via task forms
- Completion transitions process

Admin Console (Administration)

Admin console manages system configuration:

Key Features:

- User and group management
- Authorization configuration
- Application access control
- Tenant management
- License information

Authorization Control:

- Resource-based access control
- Role assignment (camunda-admin, camunda-user, etc.)
- Permission granularity (CREATE, READ, UPDATE, DELETE, ALL)

Camunda 8 Setup and Deployment

Architecture Components

Camunda 8 consists of several specialized components working together:

Core Components:

1. Zeebe: Process automation engine
2. Operate: Process monitoring and troubleshooting
3. Tasklist: Human task management
4. Optimize: Process analytics (enterprise)
5. Console: Cluster configuration (cloud/enterprise)
6. Web Modeler: Browser-based process design
7. Connectors: Integration with external systems
8. Identity: Authentication and authorization

Infrastructure Components:

1. Elasticsearch: Secondary storage and analytics
2. PostgreSQL: Identity and Web Modeler data (full config)
3. Keycloak: OIDC authentication provider (full config)

Environment Requirements

System Requirements:

- Docker 20.10.16 or later
- Docker Compose 1.27.0 or later
- 8GB RAM minimum (16GB recommended for full configuration)
- 10GB free disk space

- Internet connection for initial image downloads

Network Ports Required:

- 8080: Main web interface (Operate, Tasklist)
- 8087: Console (full config)
- 8083: Optimize (full config)
- 8070: Web Modeler (full config)
- 26500: gRPC endpoint for clients
- 9200: Elasticsearch
- 5432: PostgreSQL (full config)
- 18080: Keycloak (full config)

Quick Start: Docker Compose Setup

Step 1: Download Camunda 8 Distribution

```
```bash
```

```
Navigate to working directory
```

```
mkdir -p ~/camunda8-projects
```

```
cd ~/camunda8-projects
```

```
Download Camunda 8 Docker Compose
```

```
Visit: https://github.com/camunda/camunda-platform/releases
```

```
Download: camunda-8-{version}-docker-compose.tar.gz
```

```
tar -xzf camunda-8-*-docker-compose.tar.gz
```

```
cd camunda-8-docker-compose
```

```
```
```

Step 2: Directory Structure

...

camunda-8-docker-compose/

├─ docker-compose.yaml # Lightweight config (recommended)

├─ docker-compose-full.yaml # Full stack config

├─ docker-compose-web-modeler.yaml # Web Modeler only

├─ .env # Environment variables

├─ .env.cloud # Cloud configuration

└─ connector-secrets.txt # Connector credentials

...

Step 3: Start Camunda 8 (Lightweight Configuration)

```
```bash
```

```
Start all services (default lightweight config)
```

```
docker-compose up -d
```

```
Monitor startup progress (especially Keycloak if using full config)
```

```
docker-compose logs -f
```

```
Wait for services to initialize (typically 2-5 minutes)
```

```
```
```

Step 4: Verify Deployment

```
```bash
```

# Check container status

docker-compose ps

# Expected output shows healthy status for all containers

# Check specific service logs

docker-compose logs zeebe

docker-compose logs operate

docker-compose logs tasklist

...

## Step 5: Access Camunda 8 Components

Lightweight Configuration (Default):

- Operate: <http://localhost:8088/operate>
- Tasklist: <http://localhost:8088/tasklist>
- gRPC Endpoint: localhost:26500
- REST API: <http://localhost:8088/v2>

Full Configuration:

- Operate: <http://localhost:8088/operate>
- Tasklist: <http://localhost:8088/tasklist>
- Console: <http://localhost:8087>
- Optimize: <http://localhost:8083>
- Web Modeler: <http://localhost:8070>
- Keycloak: <http://localhost:18080/auth/>
- gRPC Endpoint: localhost:26500
- REST API: <http://localhost:8088/v2>

Default Credentials:

- Username: `demo`
- Password: `demo`

## Zeebe Architecture Overview

Zeebe is the cloud-native process engine at the core of Camunda 8.

Why Zeebe?

Performance Characteristics:

- No central database bottleneck
- High throughput via distributed processing
- Consistent low latency regardless of volume
- Fault tolerance with automatic recovery
- No data loss on broker failure

Key Design Principles:

- Event-sourced architecture
- Partitioned data distribution
- Replication across brokers
- Automatic failover
- Distributed consensus (Raft)

## Zeebe Architecture Components

### 1. Clients

- Applications and workers

- Java, Go, Node.js, C#, Python SDKs
- Communication via gRPC (high-performance)
- REST API (alternative, lower performance)

## 2. Gateways

- Request routing and load balancing
- Cluster communication gateway
- Client connection management
- Request/response handling

## 3. Brokers

- Core processing units
- Partition leadership
- Command and event processing
- State machine execution

## 4. Partitions

- Distributed data units
- Event stream persistence
- Parallel processing capability
- Replication for fault tolerance

## 5. Exporters

- Event stream consumers
- Export to external systems
- Operate/Tasklist data feeding
- Custom analytics pipelines

## Partition Strategy

...

Broker 1	Broker 2	Broker 3
[Partition 0]	[Partition 1]	[Partition 2]
[Replica 1]	[Replica 2]	[Replica 0]
[Replica 2]	[Replica 0]	[Replica 1]

...

Processes distributed across partitions for:

- Parallel processing
- Load distribution
- Replication redundancy
- Failover resilience

## Zeebe Job Workers

Job workers execute service tasks in Camunda 8 processes.

Worker Responsibilities:

1. Subscribe to specific job types
2. Fetch available jobs from broker
3. Execute business logic
4. Report completion/failure
5. Update process variables
6. Handle errors and retries

- Current activity location
- Execution history
- Variable state
- Incident information
- Audit trail

#### Operations Available:

- Suspend/resume instances
- Cancel instances
- Modify execution (skip activities, repeat steps)
- Update variables
- Resolve incidents

#### Incident Management

Incidents represent process errors or issues:

#### Common Incidents:

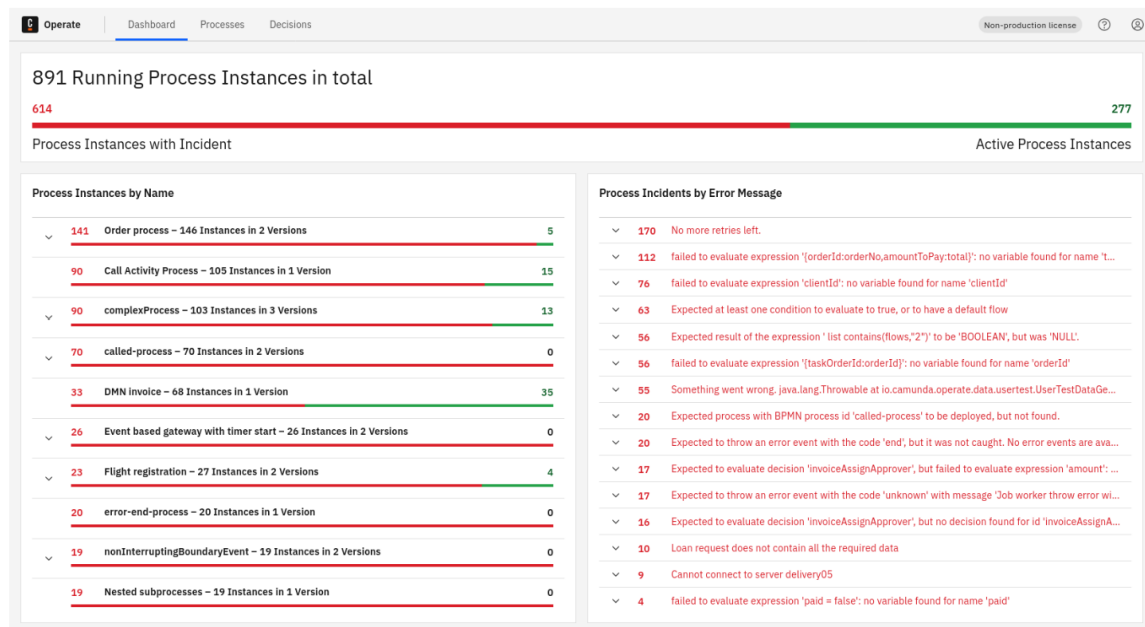
- Job execution failures
- Unhandled exceptions
- Missing external tasks
- Timer configuration errors
- Task assignment failures

#### Resolution Steps:

1. Click incident in Cockpit
2. Review error details
3. Investigate root cause
4. Update variables if needed
5. Retry or compensate
6. Resolve incident

## Camunda 8: Operate Operations

### Process Instance Monitoring



#### Dashboard Overview:

- Running processes statistics
- Incident count
- Process definition list
- Recent process instances

#### Instance View:

- Timeline visualization
- Execution flow with timing
- Variable state at each step
- Parent/child process relationships
- Retry information

### Filtering and Search:

- Filter by process definition
- Search by instance ID
- Filter by state
- Filter by incident presence
- Custom variable filters

### Incident Management in Operate

#### Incident Types:

- Incident: General execution error
- Error event caught: Modeled error handling
- Task failed: Service task execution failure
- Variable mapping error: Variable transformation failure

#### Operations:

- Update variable to fix data issues
- Retry failed task
- Migrate to different process version
- Batch operations on multiple instances
- Delete completed instances

### Process Instance Modification

Operate allows modifying running instances:

#### Capabilities:

- Add tokens to activities

- Cancel activity executions
- Move tokens between activities
- Modify variables before/after activities

Use Cases:

- Skip failed activities
- Repeat validation steps
- Resume after manual interventions
- Implement process improvements mid-execution

Tasklist: Human Task Management

Task Completion Workflow

1. Task Assignment: Assign to user or group
2. Claim: User claims unassigned task
3. Complete: User fills form and completes
4. Delegation: Optional reassignment
5. Escalation: Unassigned after timeout

Task Filters and Views

Camunda 7 Tasklist:

- My Tasks: Tasks assigned to current user
- Group Tasks: Tasks assigned to user's groups
- All Tasks: All accessible tasks
- Custom filters by variables, dates, priorities

Camunda Tasklist

Keyboard Shortcuts Create task Start process Amy Johnston

Create a filter + Created + Add Comment +

My Tasks

My Group Tasks

Accounting

John's Tasks

Mary's Tasks

Peter's Tasks

All Tasks (5)

Review Invoice  
Created 35 minutes ago  
Invoice A... 10.99  
Invoice Nu... PSAC5342

Assign Reviewer

Review Invoice  
Created 35 minutes ago  
Invoice A... 10.99  
Invoice Nu... PSAC5342

Prepare Bank Transfer  
Invoice Receipt  
Due in 7 days, Created 35 minutes ago  
Invoice A... 900  
Invoice Nu... BOS43934

Approve Invoice  
Invoice Receipt  
Due in 7 days, Created 35 minutes ago  
Invoice A... 30  
Invoice Nu... GPFF3232323

Prepare Bank Transfer

Invoice Receipt (v. V1.0)

Set follow... in 7 days Accounting Claim

Form History Diagram Description

Please prepare the bank transfer for the following invoice

Invoice Docu... invoice.pdf

Creditor Bobby's Office Supplies

Amount 900

Invoice Number BOS-43934

Appro... by demo

Save Complete

Date and Time displayed in local timezone: Europe/Amsterdam

Powered by Camunda Platform / v7.15.0

## Camunda 8 Tasklist:

- My Tasks: Personal task assignments
- All Tasks: Tenant-accessible tasks
- Custom views via variable filters
- Task sort by priority, due date, creation

Tasklist Tasks Processes

Filters

Tasks queue

Selected task details

Company registration  
Business unit management

Unassigned Assign to me

Details

Creation date  
20 May 2024, 15:23

Candidates  
No candidates

Priority  
High

Due date  
01 June 2024

Follow up date  
No follow up date

Task summary

Form

Company details

Legal company name

Name

Work phone 012345 E-mail 012345

Company Address

Address

City enter city ZIP Code 012345

Income details

Yearly income

2023 2022 2021

Amount