

**HRS Cloud Workflow Engine Recommendation**

Given that your onshore HRS uses **Camunda 7.0** for workflow state management, I have several strategic recommendations for the cloud version, each with distinct advantages depending on your migration strategy and requirements.

**Primary Recommendation: Camunda 8**

**Why Camunda 8 is the Strategic Choice**

**Cloud-Native Architecture**: Camunda 8 uses the **Zeebe engine**, which is purpose-built for cloud environments with horizontal scalability and no external database dependencies. This aligns perfectly with your AWS deployment goals.[[1]](#fn1)[[2]](#fn2)

**Enterprise Continuity**: Since Camunda 7's final feature release is **October 2025**, migrating to Camunda 8 ensures long-term support and continued innovation. The platform offers **native AI integrations** and **agentic orchestration** - valuable for intelligent alert remediation.[[1]](#fn1)

**High Availability**: Zeebe provides **fault tolerance** with pre-configured replication and can recover from failures with **no data loss and minimal downtime** - critical for BCP scenarios.[[3]](#fn3)

**Migration Considerations**

**Architectural Differences**: Camunda 8 has significant changes from version 7:[[4]](#fn4)[[2]](#fn2)

* Different engine (Zeebe vs. Activiti-based)
* No embedded engine option (cloud-native only)
* Modified multi-tenancy approach
* Different API patterns

**Migration Strategy**:

* **Parallel Development**: Build new cloud workflows in Camunda 8 while keeping existing Camunda 7 workflows onshore
* **Gradual Transition**: Start with AWS-native workflows before migrating complex hybrid scenarios
* **No Direct Migration**: Workflows will need to be **redesigned rather than migrated** due to architectural differences

**Alternative Options**

**Option 2: Temporal**

**Best for Mission-Critical Workflows**: Temporal excels at **fault-tolerant and stateful execution** with distributed architecture. Ideal if your HRS workflows involve long-running processes or complex state management.[[5]](#fn5)

**Advantages**:

* Multi-language support for diverse integration requirements
* Excellent for handling network partitions and service failures
* Strong consistency guarantees for critical BCP scenarios

**Option 3: Apache Airflow**

**Best for Data-Heavy Operations**: If your HRS workflows involve significant **data pipeline orchestration**, Airflow provides robust scheduling and monitoring with Python-based workflow authoring.[[5]](#fn5)

**Advantages**:

* Extensive connector ecosystem
* Strong community support
* Excellent for complex dependency management

**Option 4: Argo Workflows**

**Best for Kubernetes-Native Deployment**: If you're deploying on **Amazon EKS**, Argo provides **container-native execution** with deep Kubernetes integration.[[5]](#fn5)

**Advantages**:

* Perfect for containerized workloads
* Excellent parallelism and dependency handling
* GitOps-friendly workflow management

**Recommended Architecture for HRS Cloud**

**Hybrid Approach with Camunda 8**

┌─────────────────┐ ┌─────────────────┐  
│ Onshore HRS │ │ Cloud HRS │  
│ (Camunda 7) │◄──►│ (Camunda 8) │  
├─────────────────┤ ├─────────────────┤  
│ • Legacy alerts │ │ • AWS alerts │  
│ • Local systems │ │ • Hybrid alerts │  
│ • BCP failback │ │ • BCP primary │  
└─────────────────┘ └─────────────────┘

**Implementation Strategy**

**Phase 1**: Deploy Camunda 8 on **Amazon EKS** with:

* **Zeebe cluster** for workflow execution
* **Operate** for monitoring and troubleshooting
* **Tasklist** for manual intervention workflows
* **Modeler** (web-based) for collaborative workflow development

**Phase 2**: Implement **connector ecosystem** for:[[3]](#fn3)

* AWS service integrations (EC2, RDS, Lambda, CloudWatch)
* PagerDuty webhook processing
* Secure communication with onshore systems
* Custom connectors for legacy system integration

**Phase 3**: Build **BCP-specific workflows** leveraging:

* Multi-region deployment capabilities
* Real-time process monitoring
* Automated rollback mechanisms
* Cross-environment state synchronization

**Key Advantages for HRS Use Case**

**Scalability**: Zeebe can **scale throughput linearly** by adding cluster nodes, essential for handling alert bursts during incidents.[[3]](#fn3)

**Observability**: Built-in **audit trail** with append-only logging provides complete visibility for compliance and troubleshooting.[[3]](#fn3)

**Integration Flexibility**: **Language-agnostic client model** allows integration with existing Java, Python, or other language-based services.[[3]](#fn3)

**Operational Excellence**: **Reactive publish-subscribe model** enables microservices to maintain autonomy while participating in workflows - perfect for distributed HRS architecture.[[3]](#fn3)

**Final Recommendation**

**Choose Camunda 8** for your HRS cloud implementation. While it requires rebuilding workflows rather than direct migration, it provides the best long-term strategic value with cloud-native architecture, enterprise support, and alignment with your existing Camunda expertise. The investment in redesigning workflows will pay dividends in scalability, reliability, and future-proofing your HRS platform.

Start with a **proof of concept** focusing on simple AWS-native alert remediation workflows to validate the architecture before committing to the full migration strategy.

⁂

1. <https://camunda.com/platform-7/>

1. <https://blog.viadee.de/en/camunda-8-camunda-7-differences>

1. <https://docs.camunda.io/docs/components/>

1. <https://docs.camunda.io/docs/guides/migrating-from-camunda-7/conceptual-differences/>

1. <https://merlio.app/blog/open-source-workflow-engines-2024>

1. <https://www.nected.ai/blog/top-workflow-orchestration-tools>

1. <https://docs.camunda.org/manual/latest/>

1. <https://pretius.com/blog/camunda-7-vs-camunda-8>

1. <https://en.wikipedia.org/wiki/Camunda>

1. <https://www.rst.software/blog/camunda-7-vs-camunda-8---key-differences-and-considerations-before-migration>