JPL

Common Workflow Service

Standards Based Solution for Managing Operational Processes

Galen Hollins

Jet Propulsion Laboratory,
California Institute of Technology

Overview

What is the Common Workflow Service (CWS)?

A **standards** based Business Process Management (BPM) solution for executing and managing mission operations processes.

Developed under the **M**ultimission **G**round **S**ystem and **S**ervices NASA program. MGSS is the NASA Program responsible for the management, implementation, maintenance and operation of the Advanced Multi-Mission Operations System (AMMOS).

Need for Workflow Solutions at JPL

- Automation and monitoring of operational processes
  - Scientific data pipelines
  - Other automation
- Standardized way to do things
  - Shared, collective benefit of productivity
- Understandability
- Maintainability across developers and projects

Points to Consider

- Distributed processing, performance constraints
- Both short and long-running tasks
- Wide variety of workflow types
- Agile process development/modification

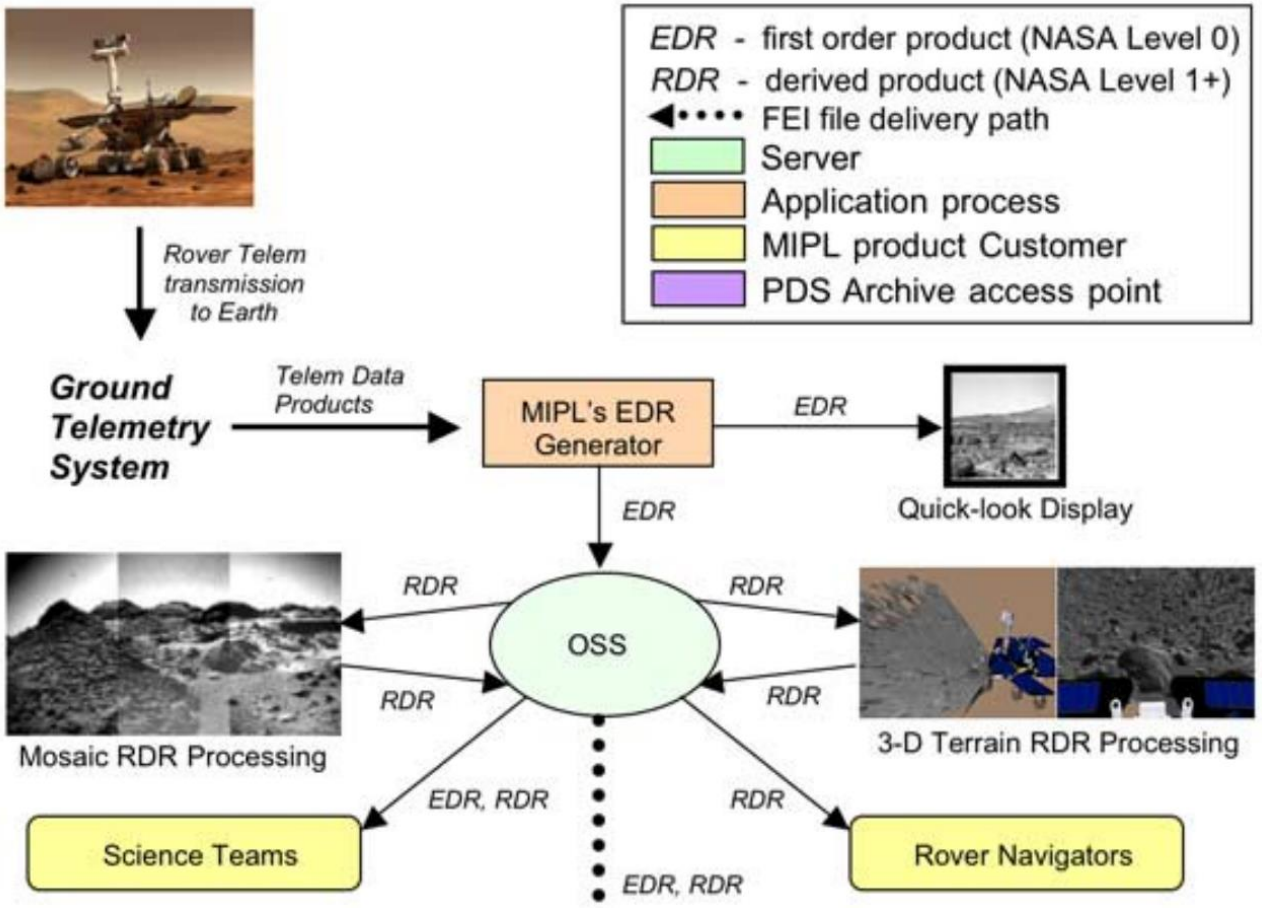
Software Design Approach

1. *Adhere to current BPM industry standards*
   - BPMN 2.0

2. *Leverage existing BPMS solutions*
   - Investigated open-source solutions.
   - Selected Camunda BPM (http://camunda.org)

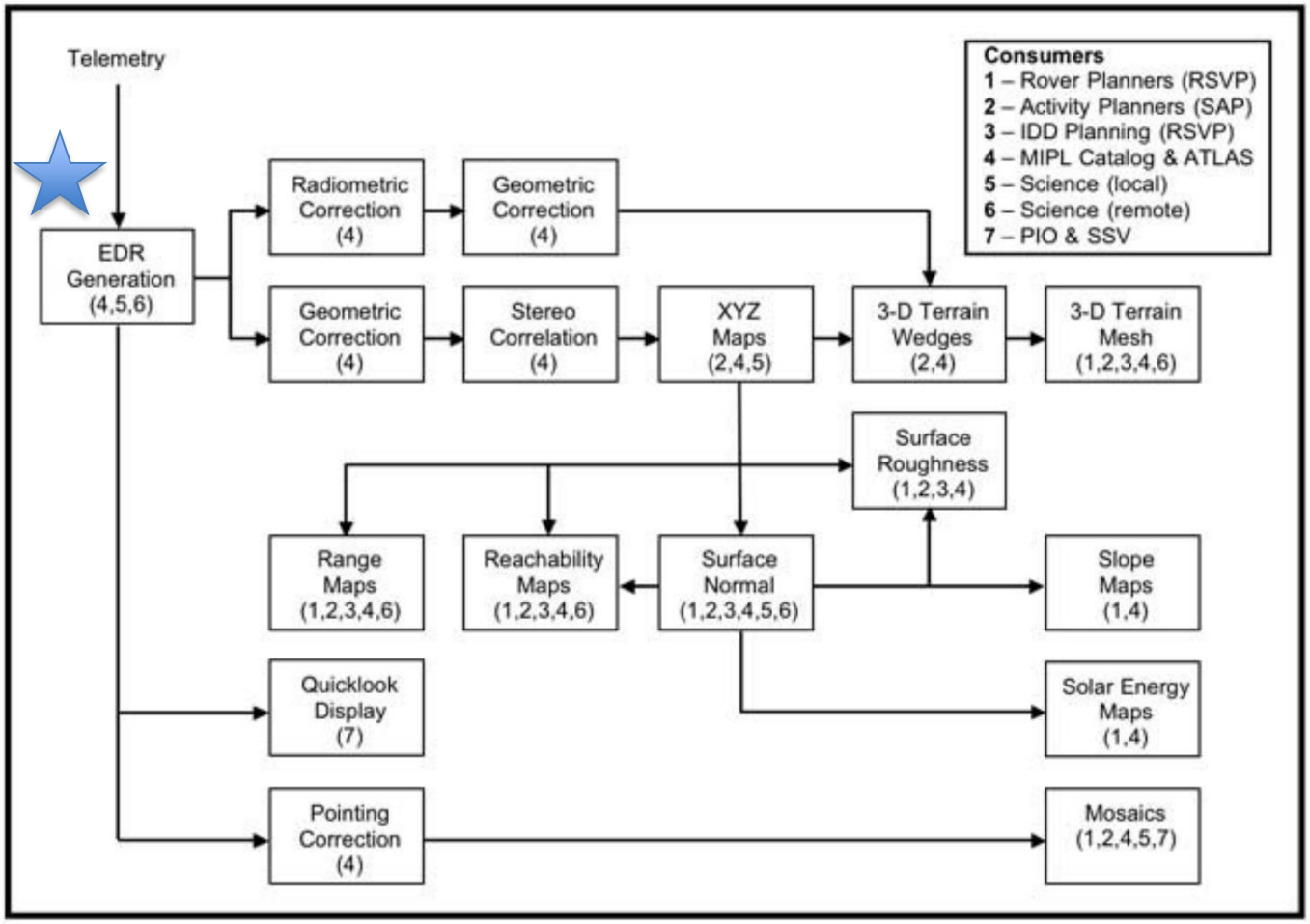
3. *The design must be flexible to support a wide variety of use cases*
   - Scalable architecture to allow for use cases with different performance requirements

Example Mission Data Flow



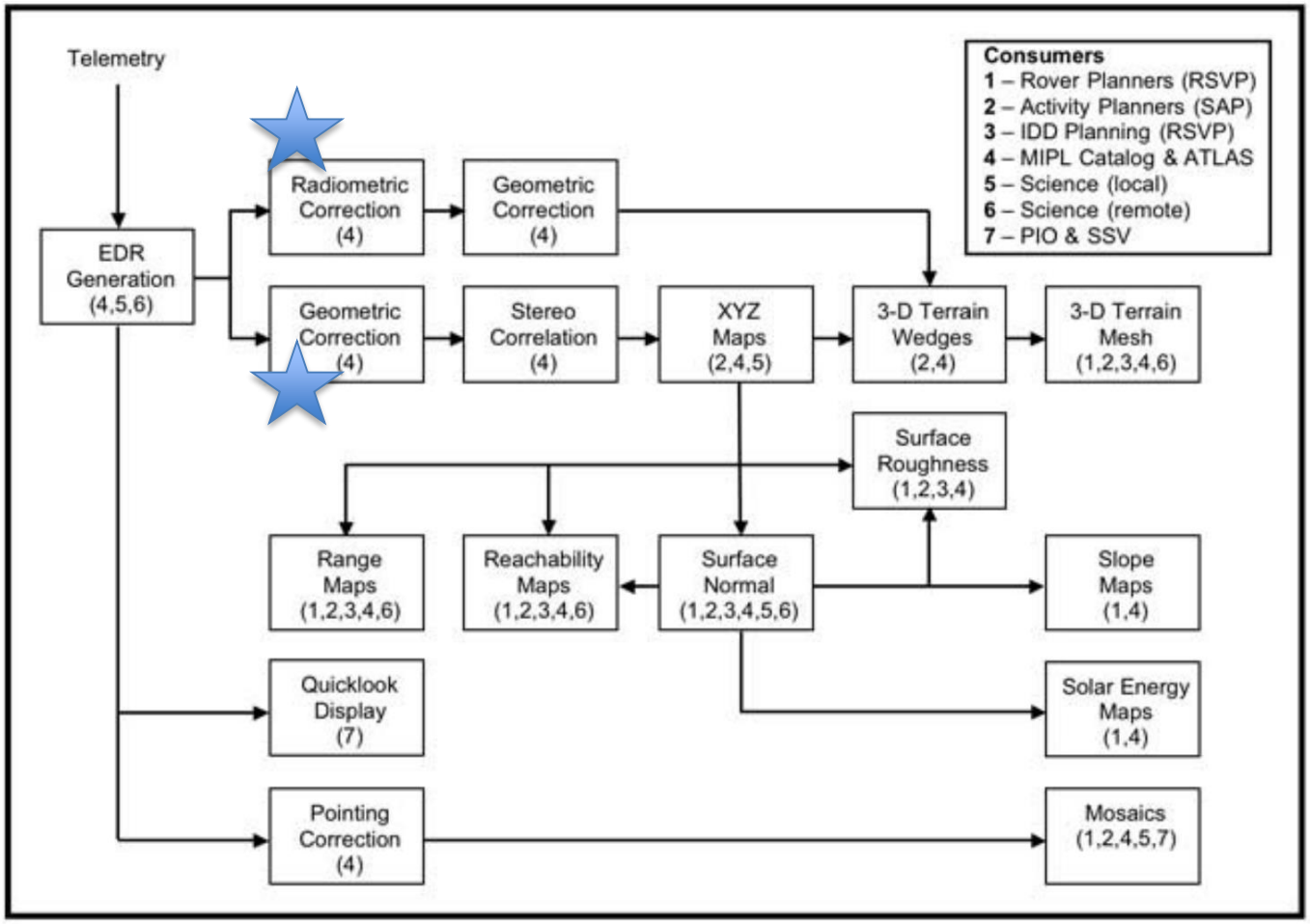
Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Typical Product Generation Pipeline



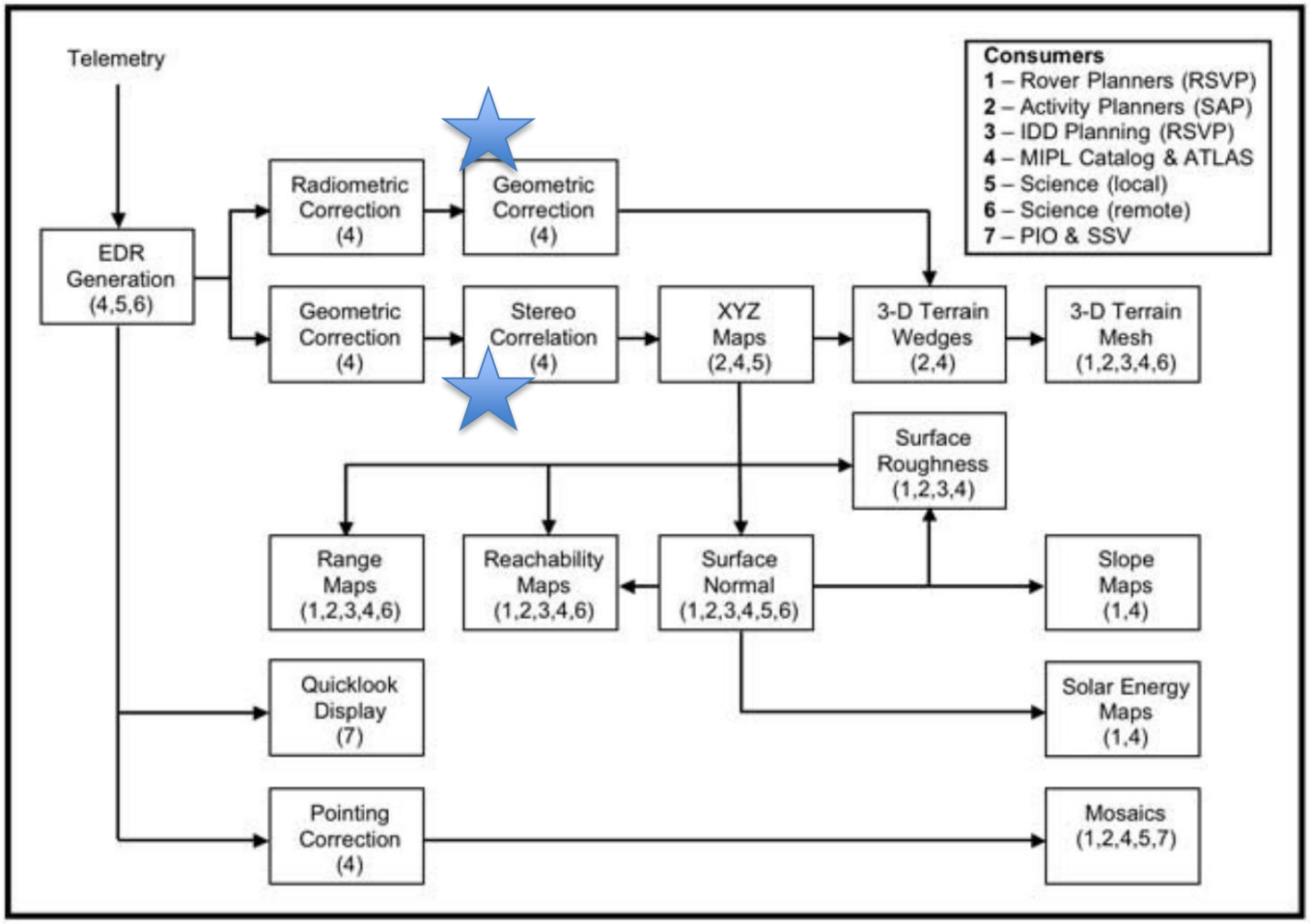
Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Typical Product Generation Pipeline



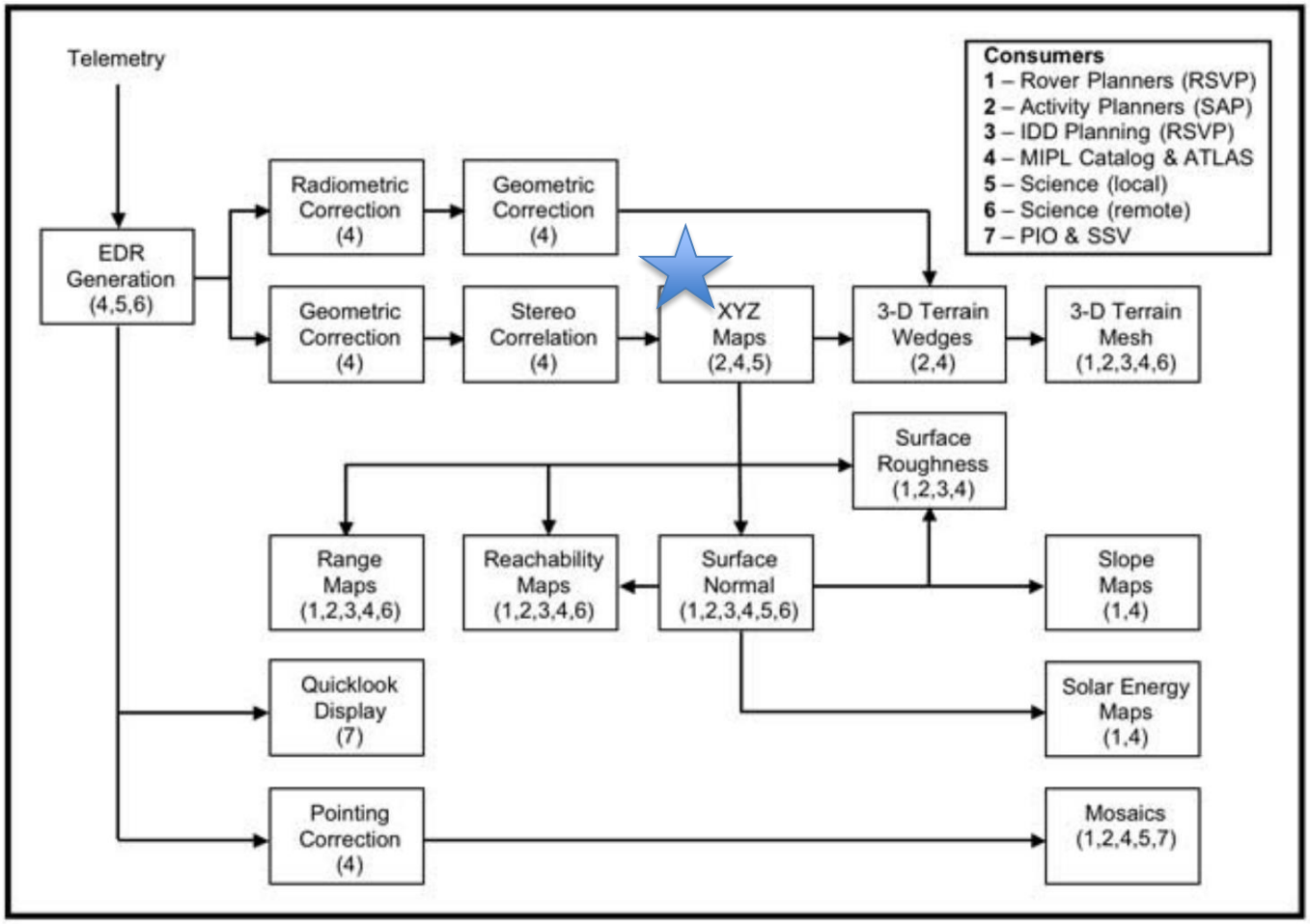
Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Typical Product Generation Pipeline



Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Typical Product Generation Pipeline



Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

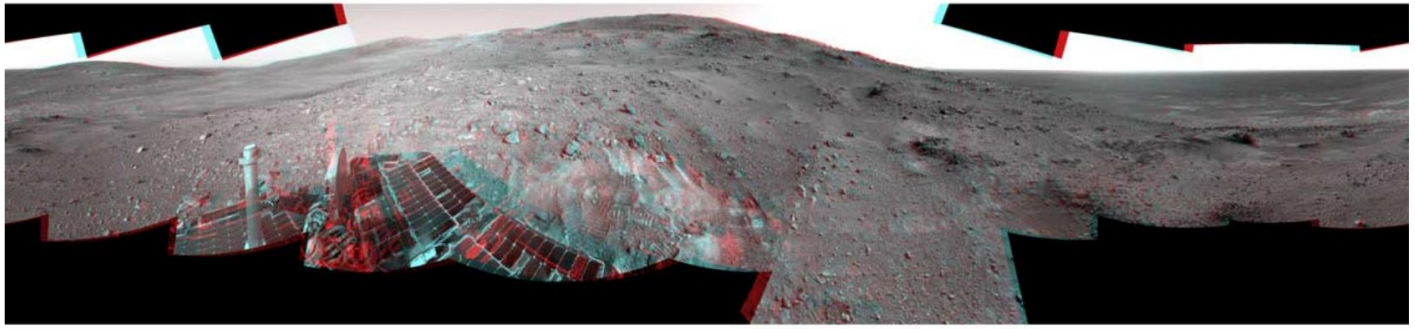
Example Data Products

a

Mosaic

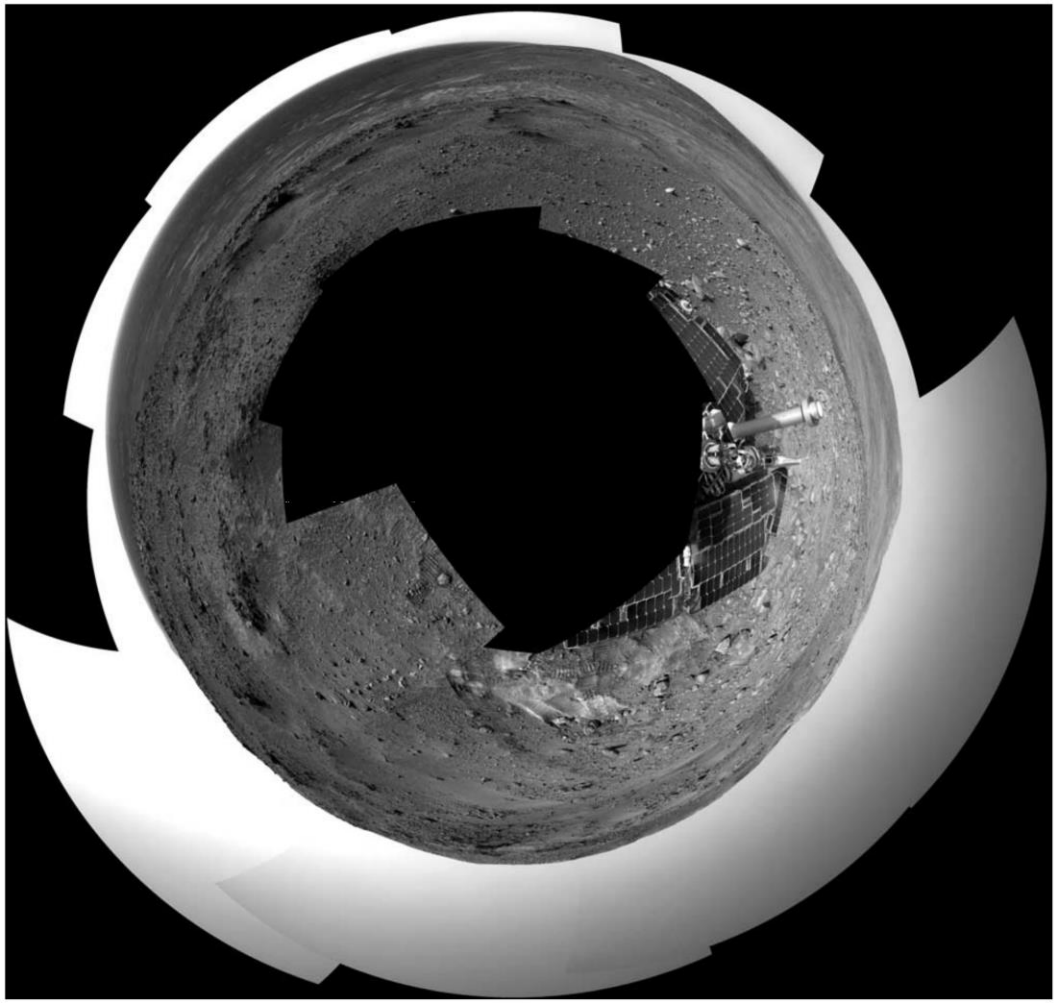
b

Stereo Mosaic



Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Example Data Products

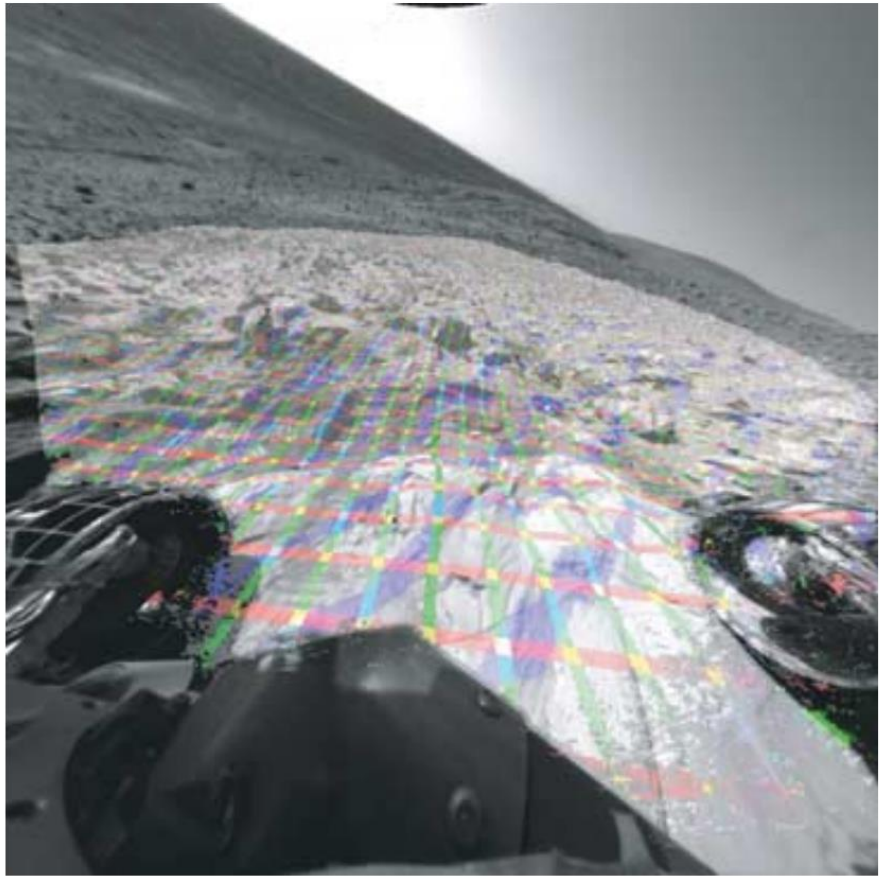


Polar Projection

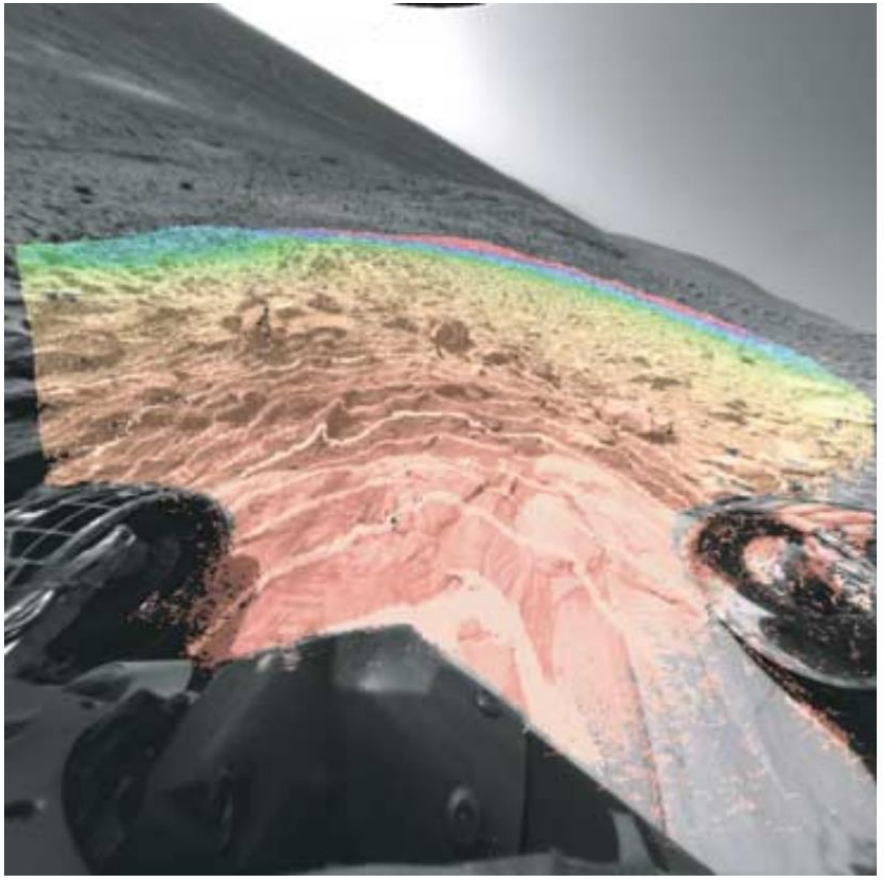
Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Example Data Products

XYZ Image



Range Image



Source: JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 111, E02S02, doi:10.1029/2005JE002462, 2006

Overview

What is the Common Workflow Service (CWS)?

A **standards** based Business Process Management (BPM) solution for executing and managing mission operations processes.

CWS =

BPMN Engine (Camunda BPM)

+

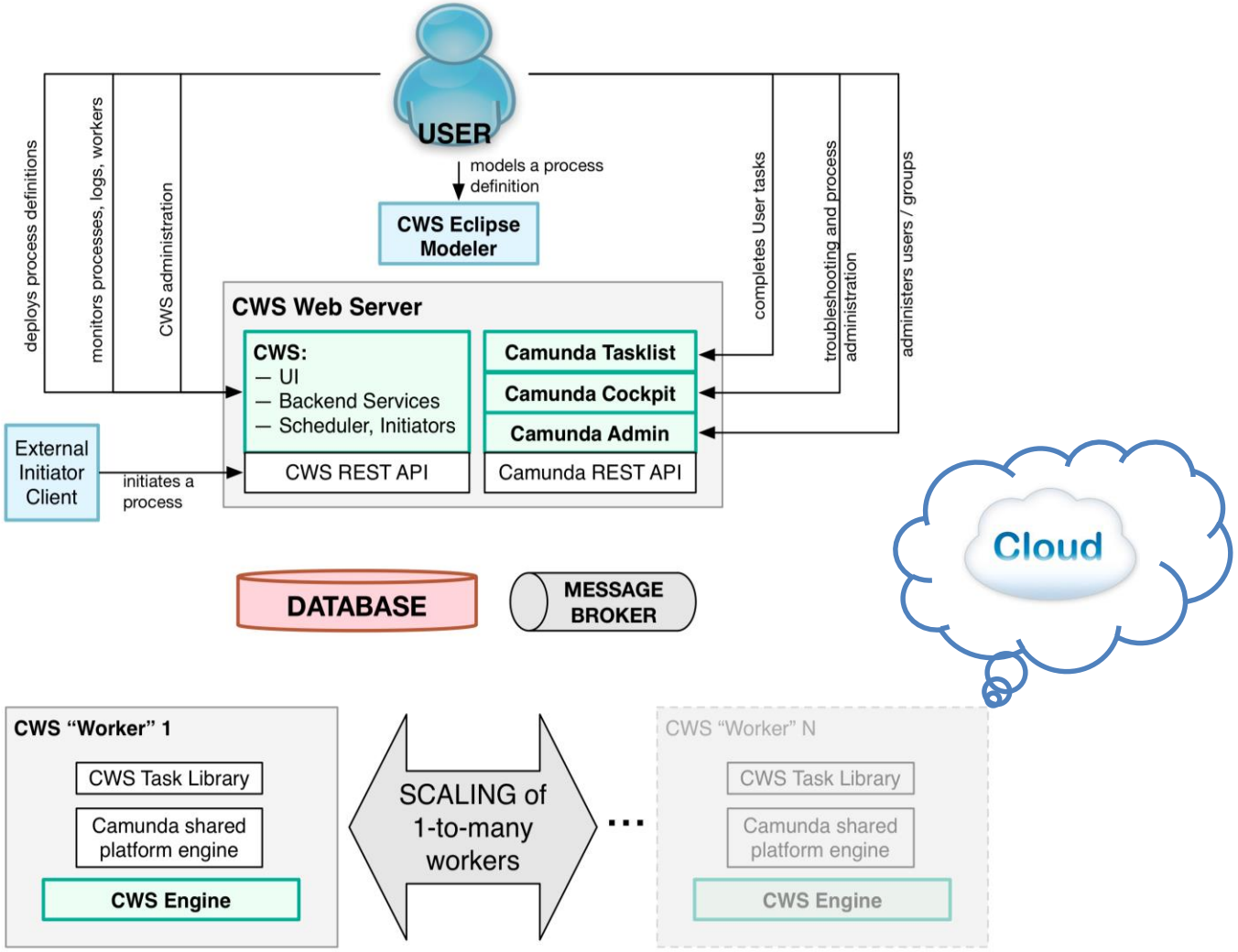
Customized BPMN 2.0 Modeler

+

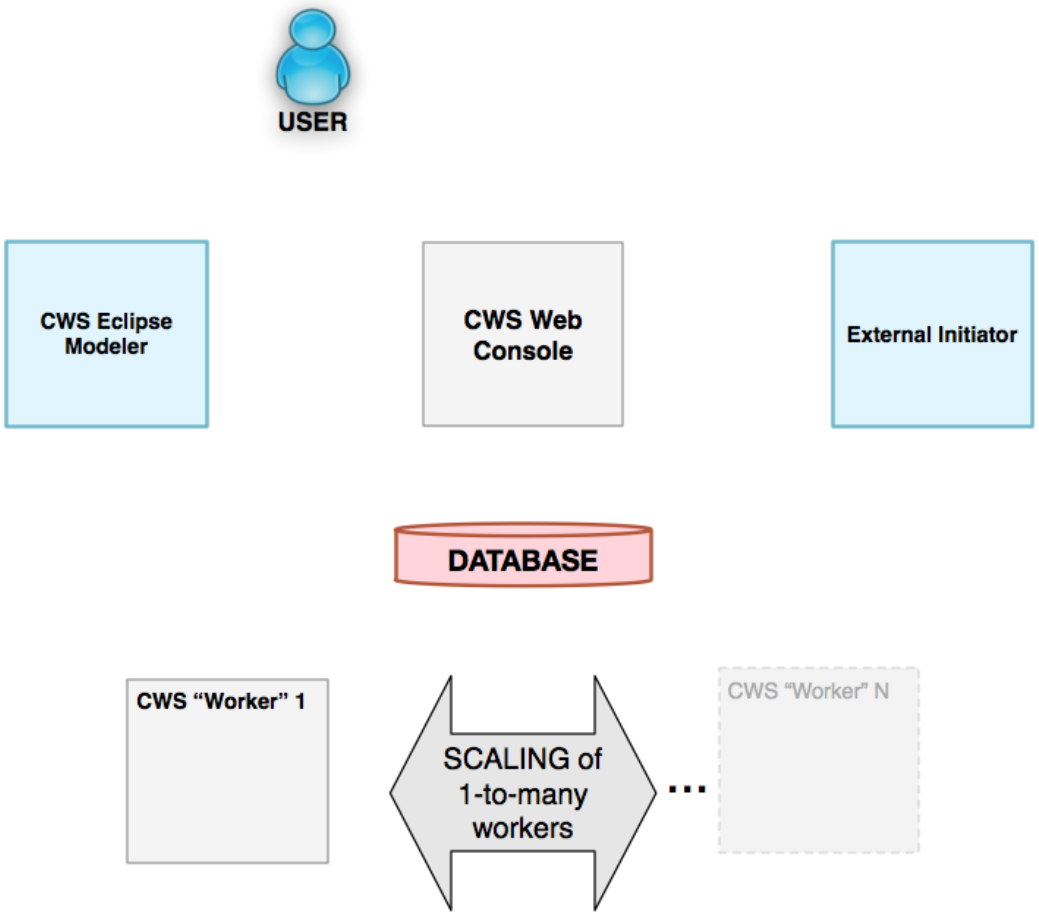
Monitoring and Management Web Applications (CWS + Camunda)

*Applicable to planetary and science SDS processing pipelines, and many other use cases

Architecture



Overview



Overview

CWS Eclipse Modeler

CWS Web Console

External Initiator

DATABASE

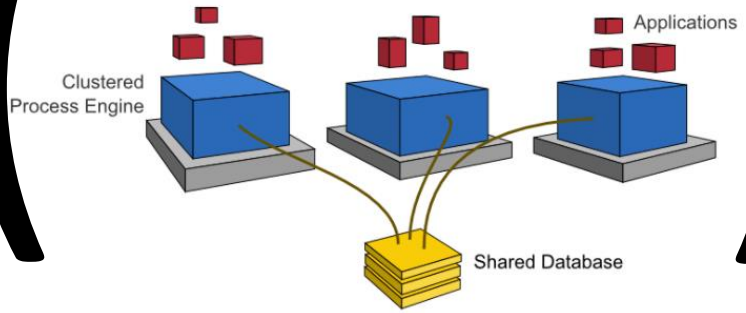
CWS "Worker" 1

SCALING of 1-to-many workers

...

CWS "Worker" N

Camunda "clustered" setup

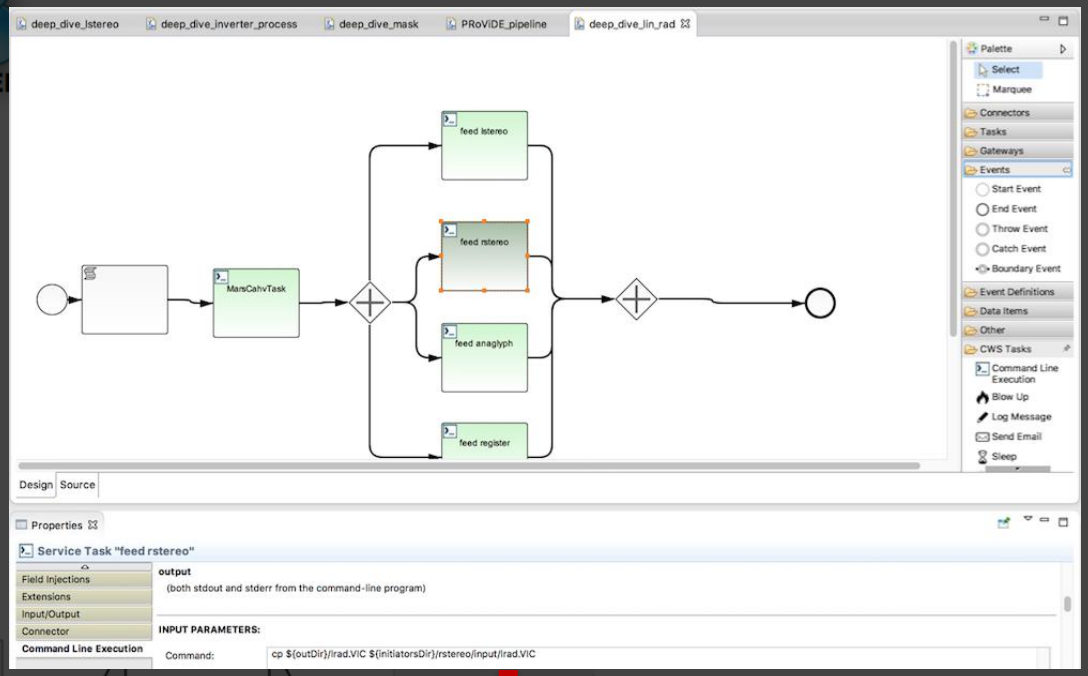


Modeling a Process Definition

Model a process, using CWS modeler application

USER

CWS Eclipse Modeler



CWS "Worker" 1

SCALING of 1-to-many workers

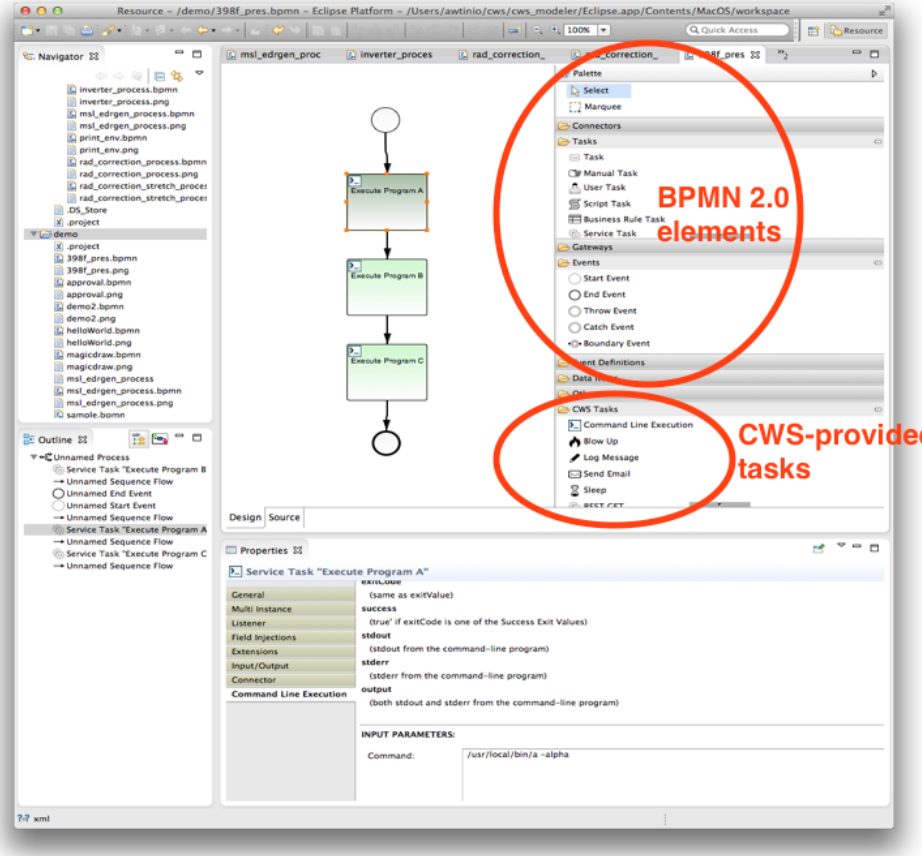
...

produces...

XML BPMN 2.0 Process Definition File

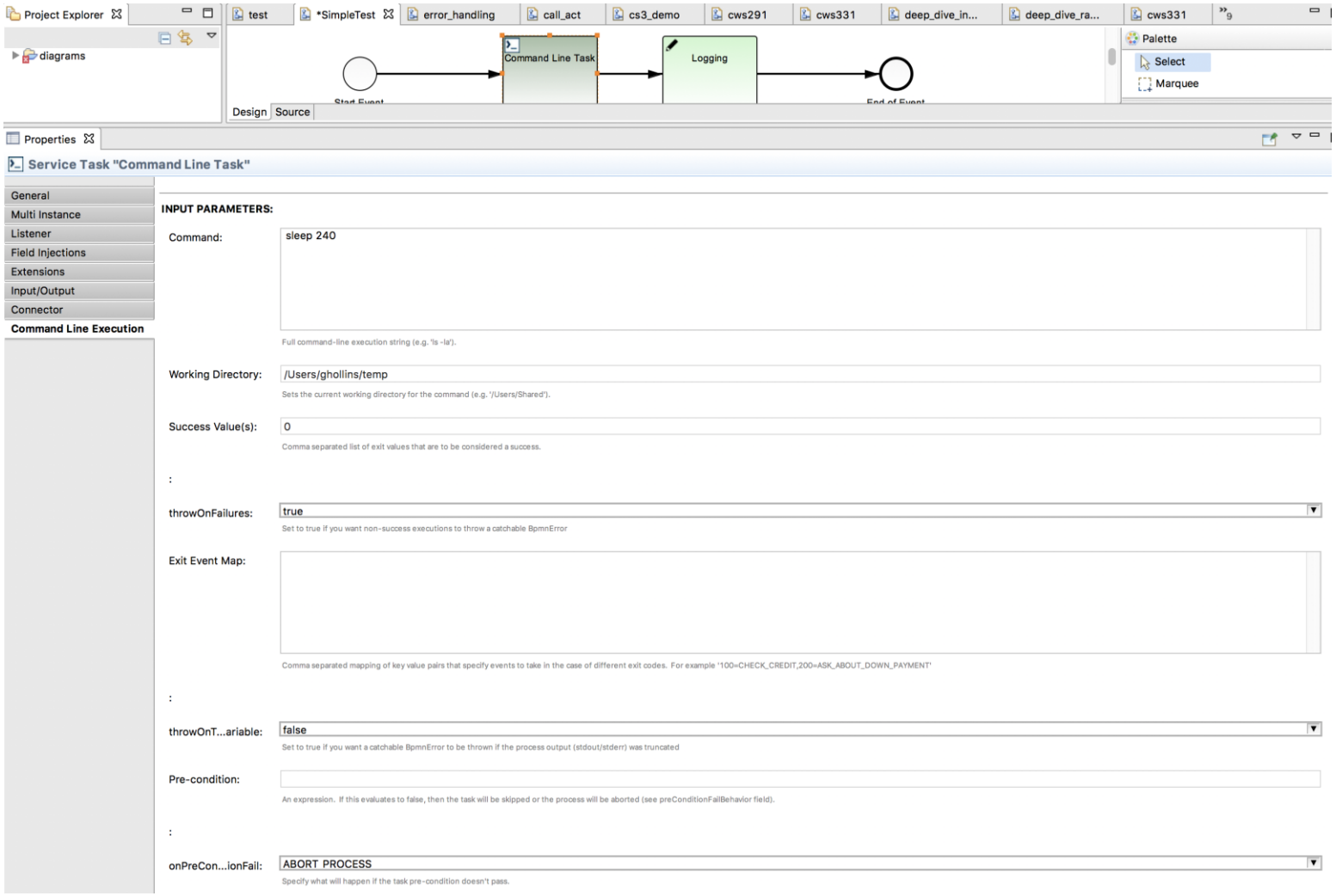
Creating a Process Definition

Process Modeler GUI – drag and drop process modeling



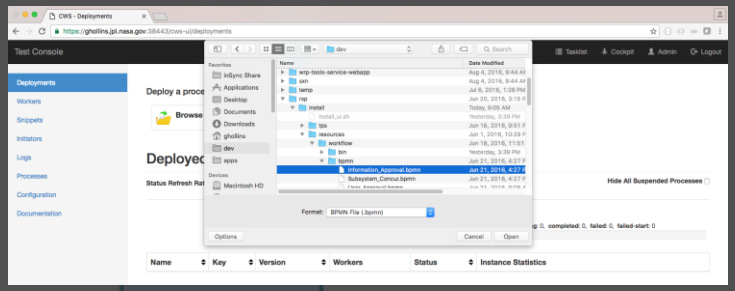

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/201
  <bpmn2:process id="Process_1" isExecutable="false">
    <bpmn2:serviceTask id="ServiceTask_2" camunda:class="jpl.cws.task.CmdLineExecTask" name="Execute Program B">
      <bpmn2:extensionElements>
        <camunda:field expression="true" name="throwOnFailures"/>
        <camunda:field expression="false" name="throwOnTruncatedVariable"/>
        <camunda:field expression="ABORT_PROCESS" name="onPreConditionFail"/>
        <camunda:field name="cmdLine">
          <camunda:expression>/usr/local/bin/b -beta</camunda:expression>
        </camunda:field>
      </bpmn2:extensionElements>
      <bpmn2:incoming>SequenceFlow_2</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_3</bpmn2:outgoing>
    </bpmn2:serviceTask>
    <bpmn2:sequenceFlow id="SequenceFlow_3" name="" sourceRef="ServiceTask_2" targetRef="ServiceTask_3"/>
    <bpmn2:endEvent id="EndEvent_1">
      <bpmn2:incoming>SequenceFlow_4</bpmn2:incoming>
    </bpmn2:endEvent>
    <bpmn2:startEvent id="StartEvent_1">
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" name="" sourceRef="StartEvent_1" targetRef="ServiceTask_1"/>
    <bpmn2:serviceTask id="ServiceTask_1" camunda:class="jpl.cws.task.CmdLineExecTask" name="Execute Program A">
      <bpmn2:extensionElements>
        <camunda:field expression="true" name="throwOnFailures"/>
        <camunda:field expression="false" name="throwOnTruncatedVariable"/>
        <camunda:field expression="ABORT_PROCESS" name="onPreConditionFail"/>
        <camunda:field name="cmdLine">
          <camunda:expression>/usr/local/bin/a -alpha</camunda:expression>
        </camunda:field>
      </bpmn2:extensionElements>
      <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_2</bpmn2:outgoing>
    </bpmn2:serviceTask>
    <bpmn2:sequenceFlow id="SequenceFlow_2" name="" sourceRef="ServiceTask_1" targetRef="ServiceTask_2"/>
    <bpmn2:serviceTask id="ServiceTask_3" camunda:class="jpl.cws.task.CmdLineExecTask" name="Execute Program C">
      <bpmn2:extensionElements>
        <camunda:field expression="true" name="throwOnFailures"/>
        <camunda:field expression="false" name="throwOnTruncatedVariable"/>
        <camunda:field expression="ABORT_PROCESS" name="onPreConditionFail"/>
```

CWS Task Properties



Deploying a Process Definition

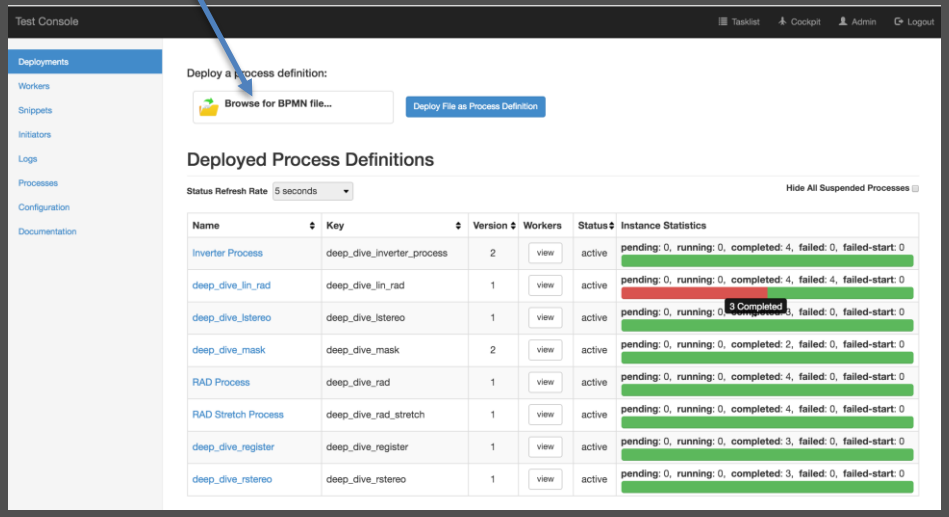
Choose BPMN model file from filesystem



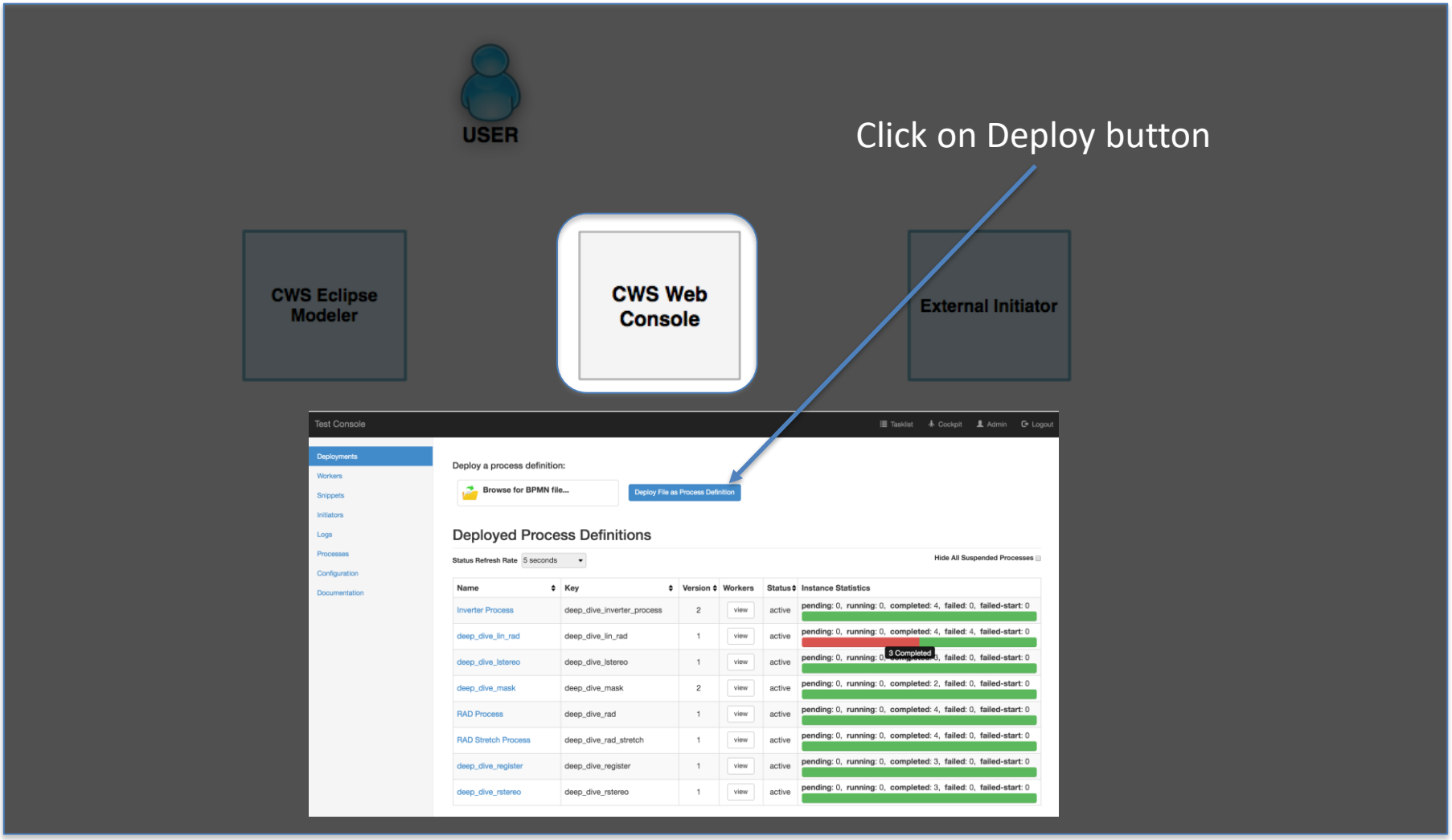
CWS Eclipse Modeler

CWS Web Console

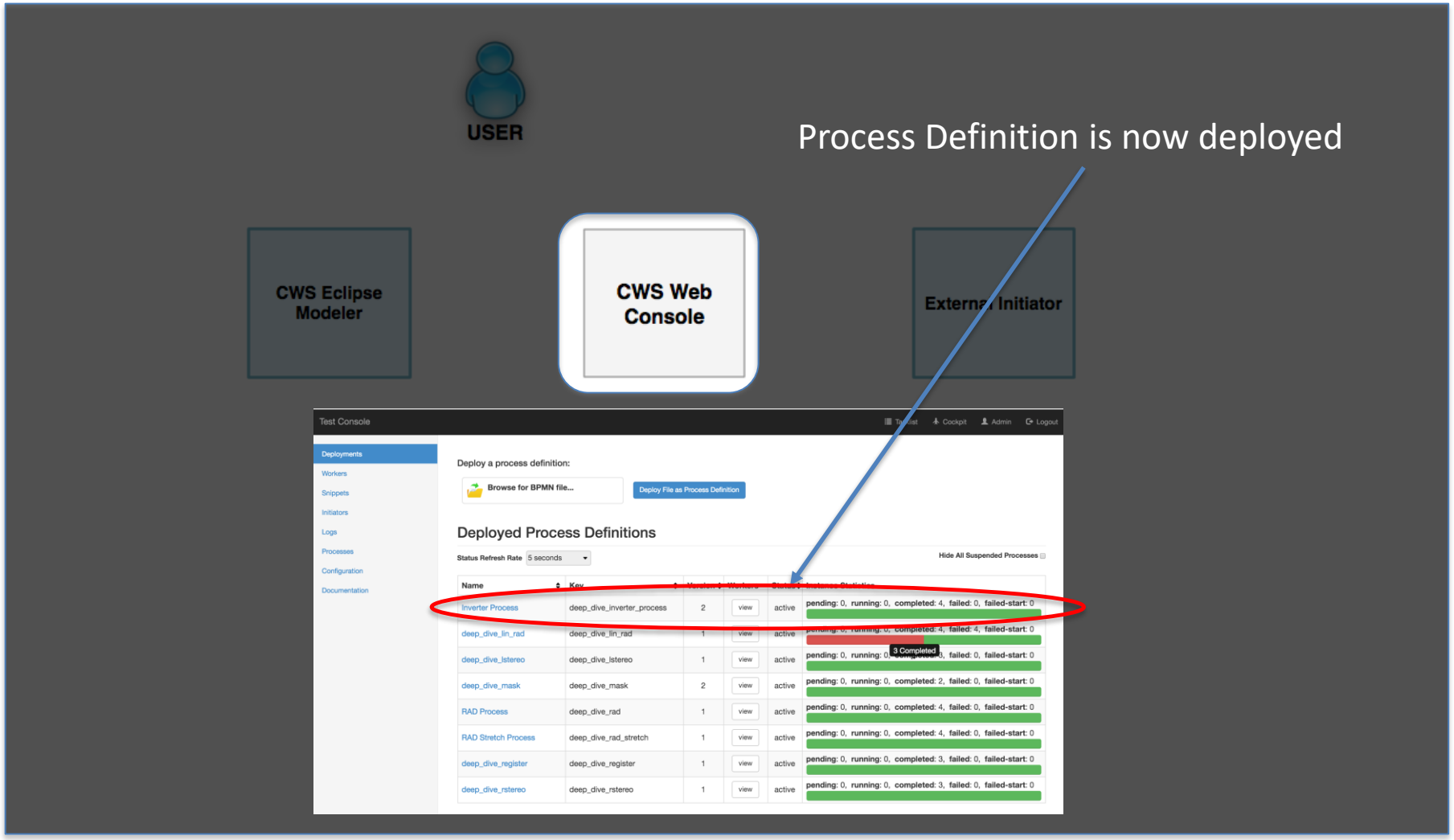
External Initiator



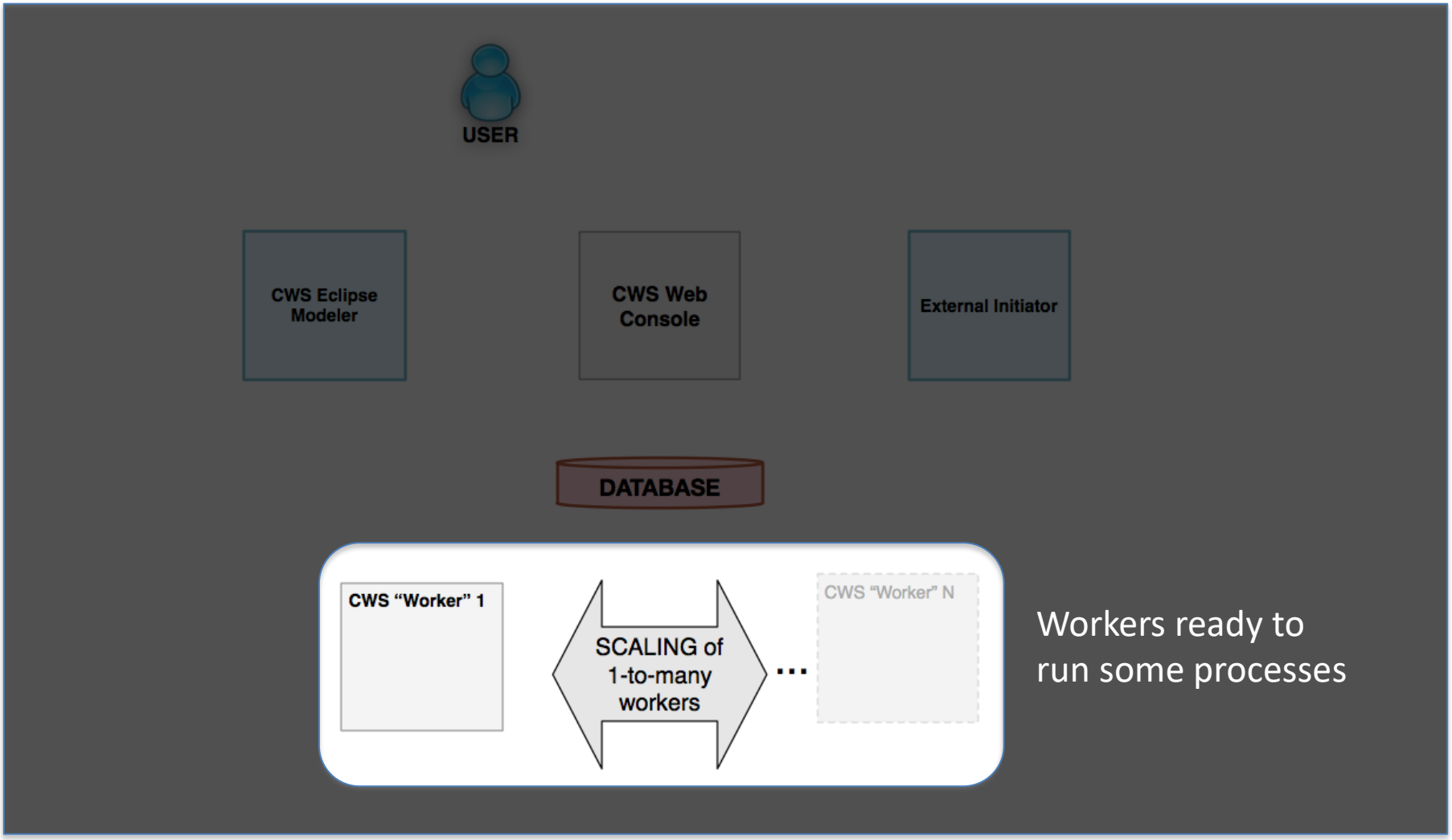
Deploying a Process Definition



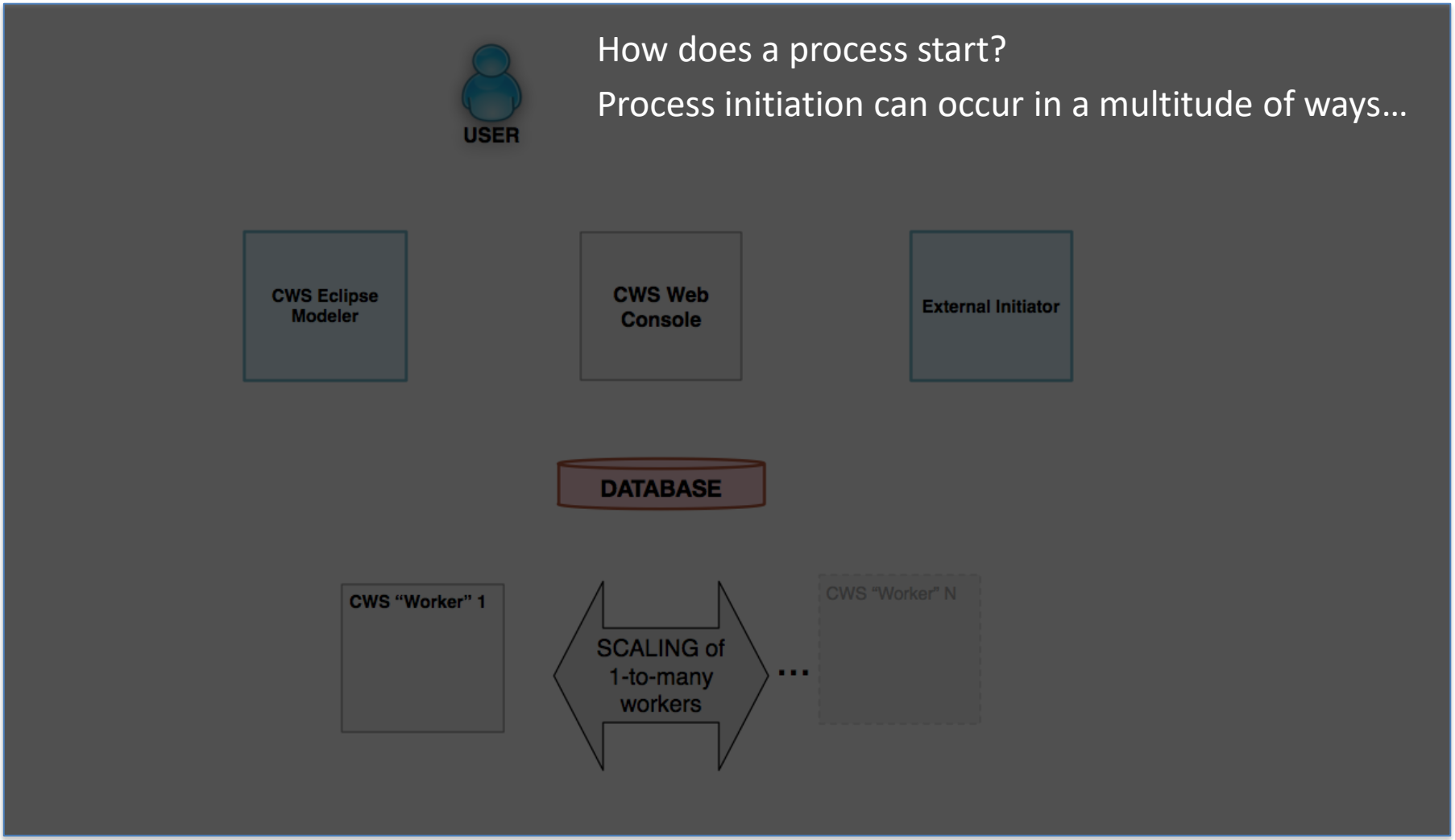
Deploying a Process Definition



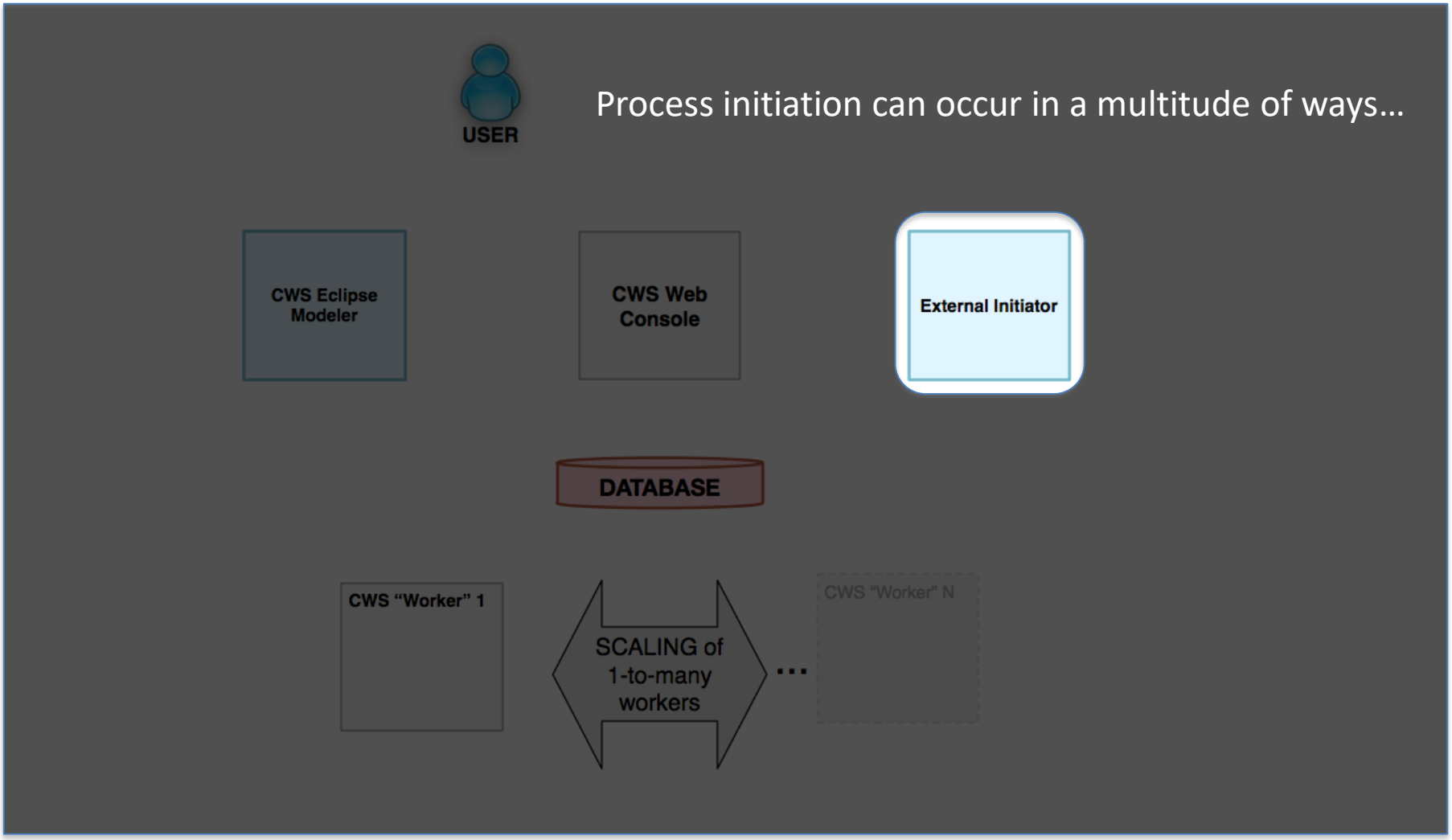
Workers



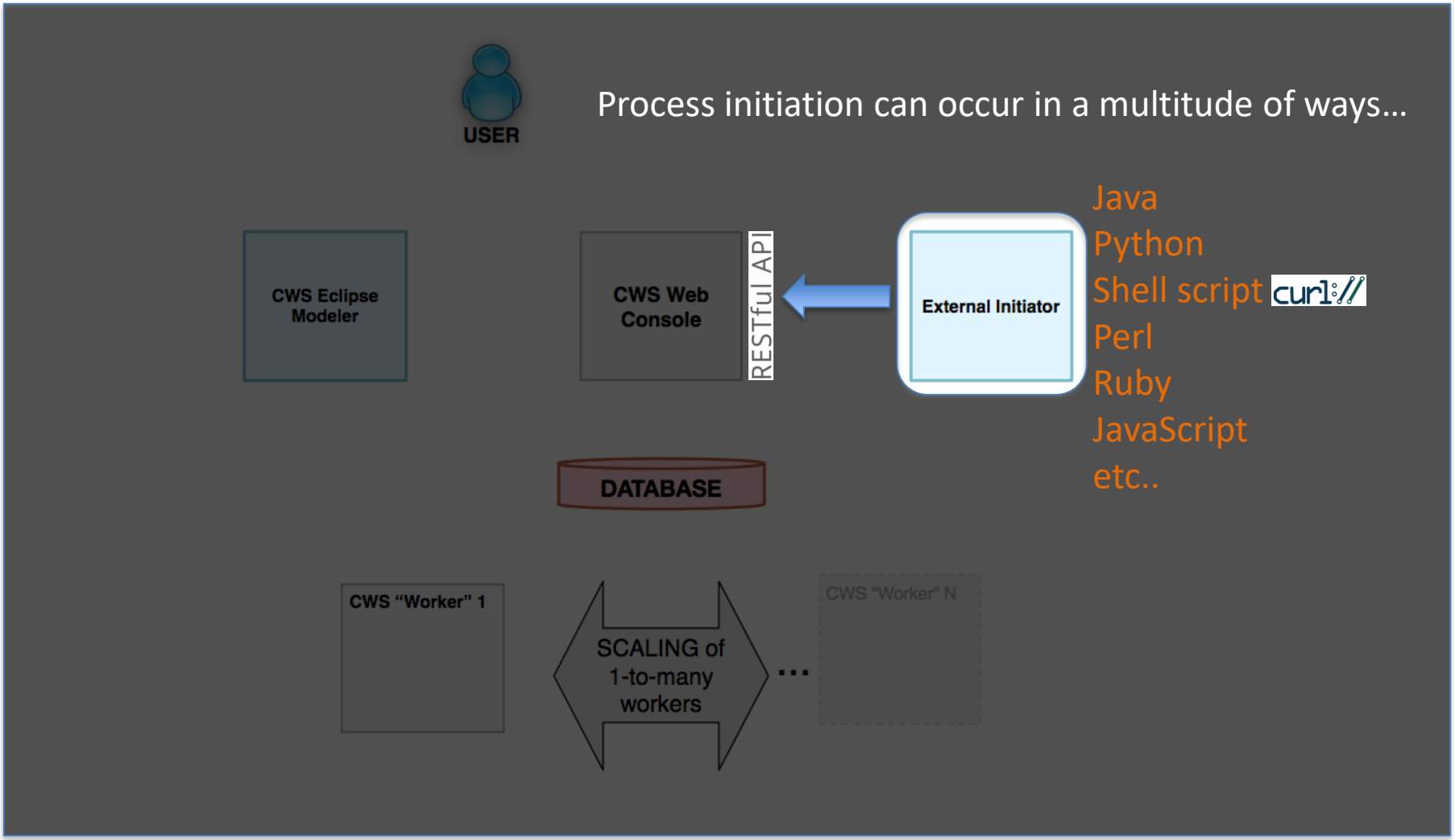
Process Initiation



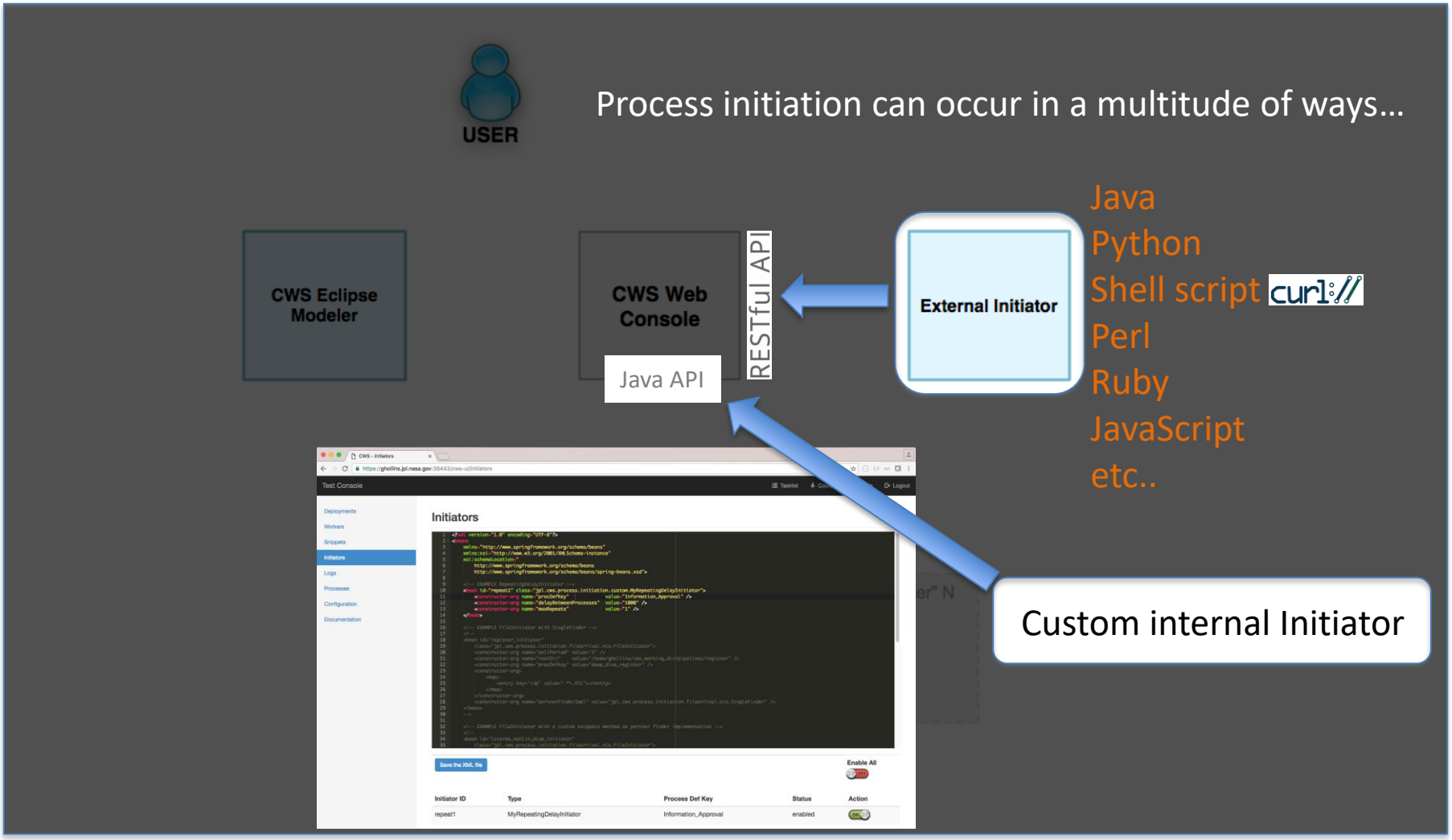
Process Initiation



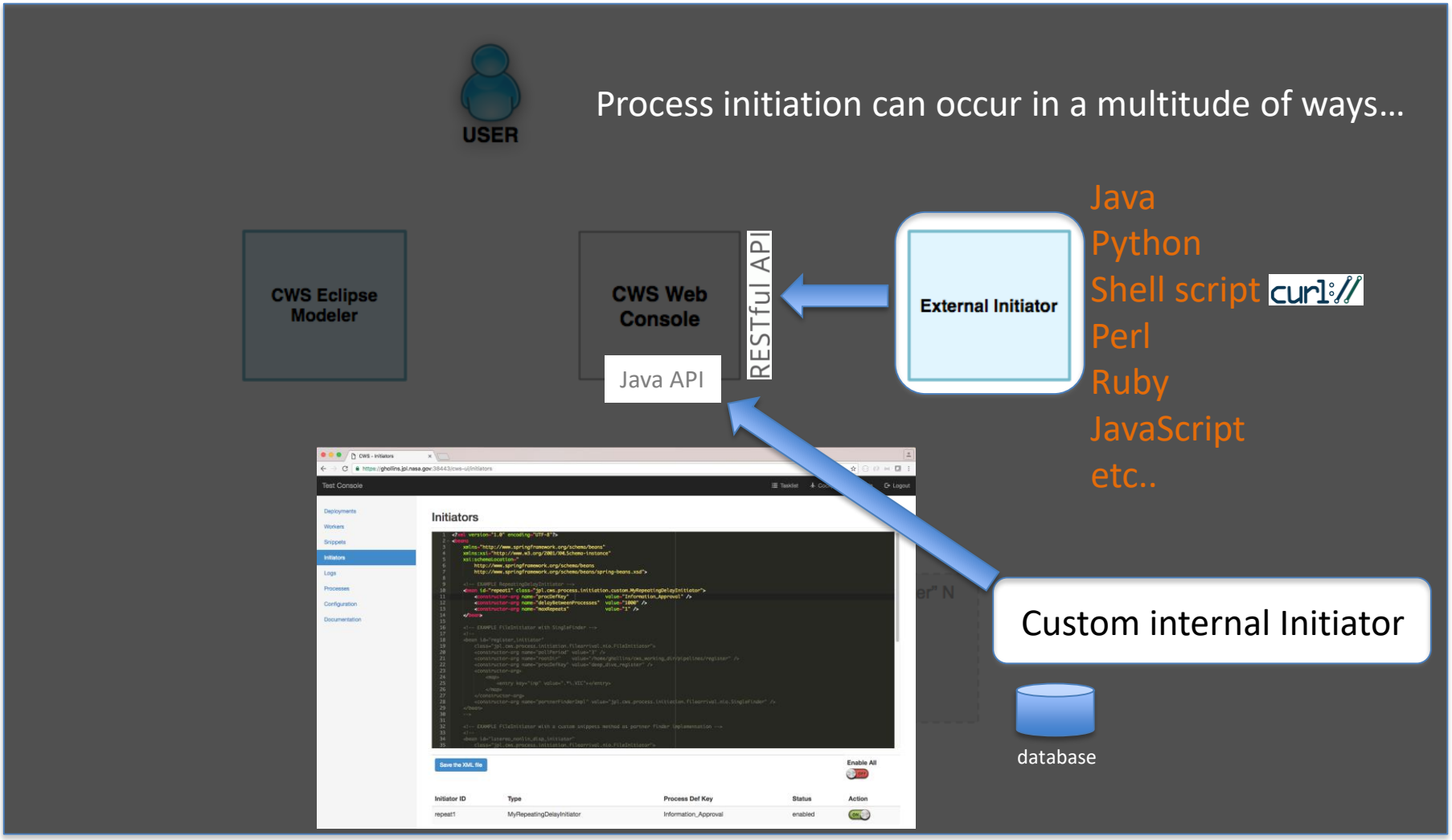
Process Initiation



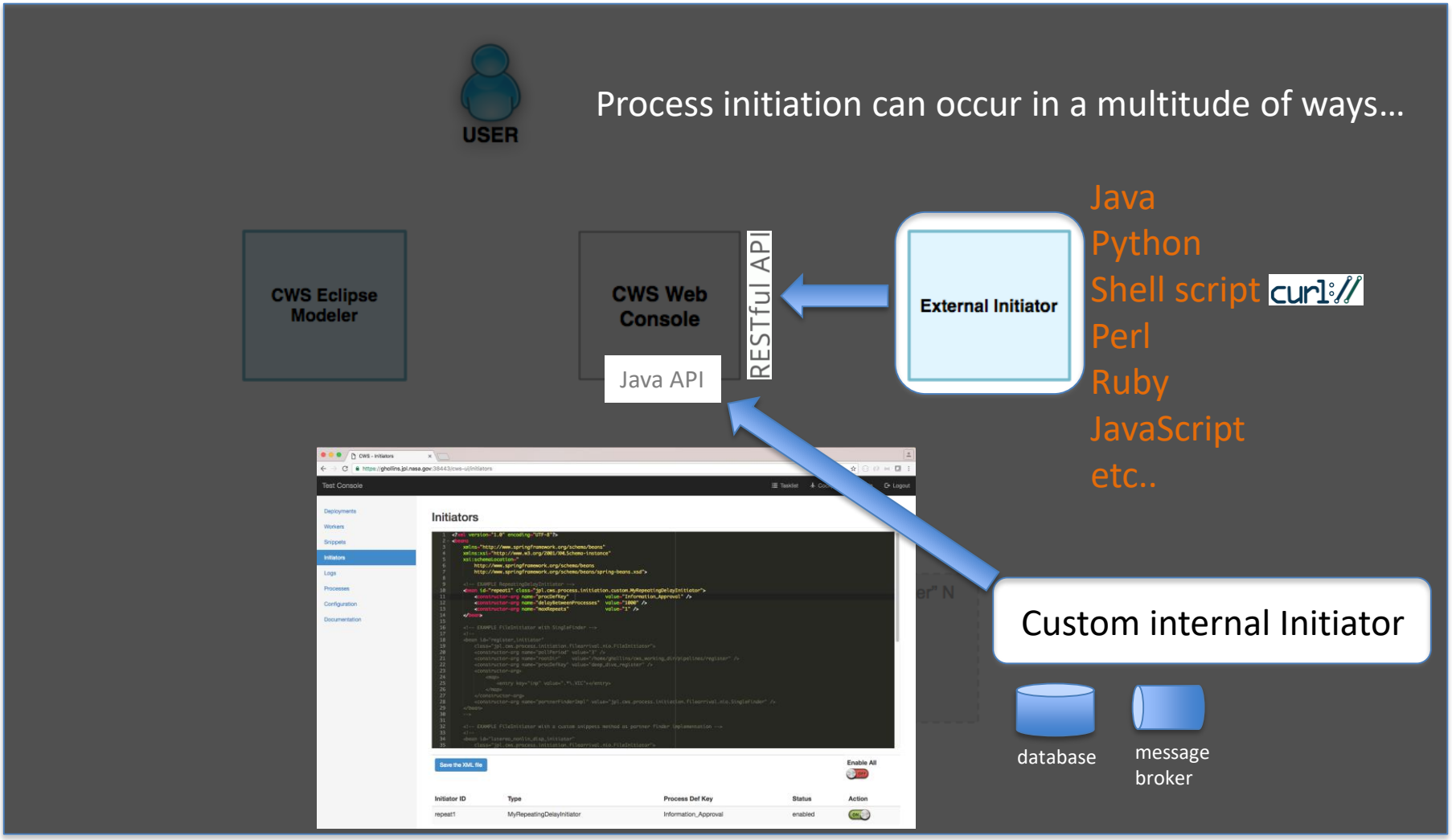
Process Initiation



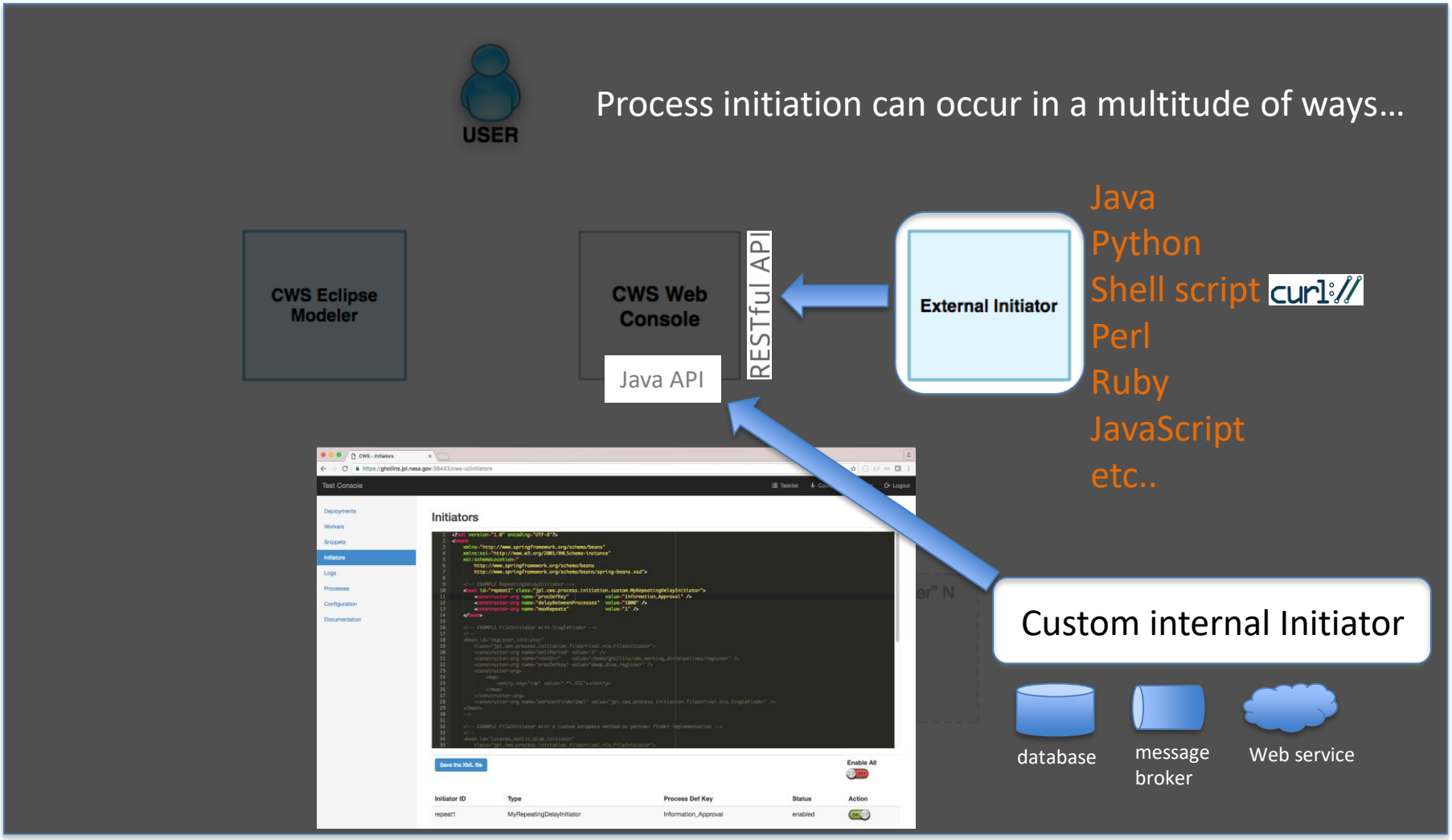
Process Initiation



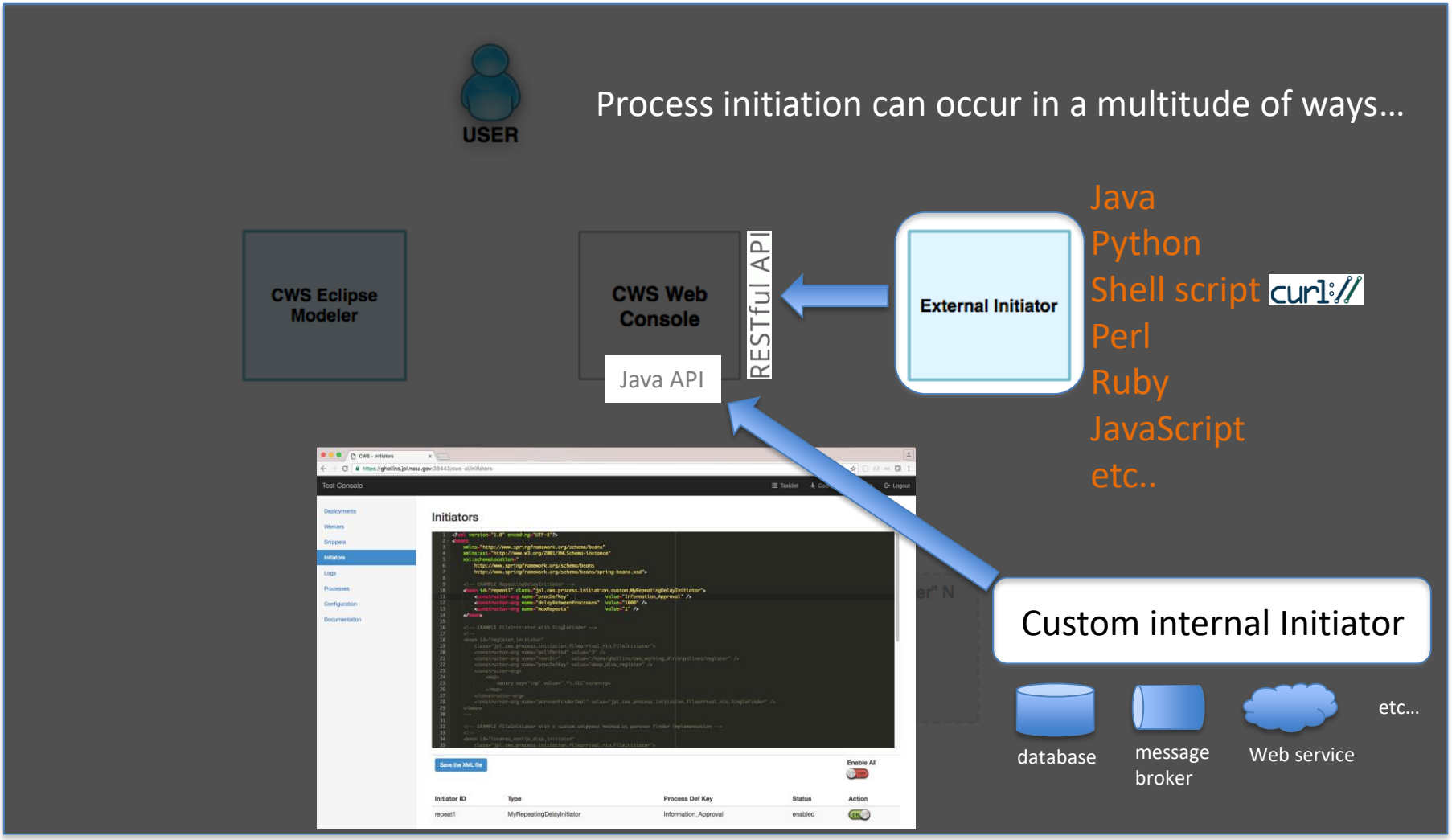
Process Initiation



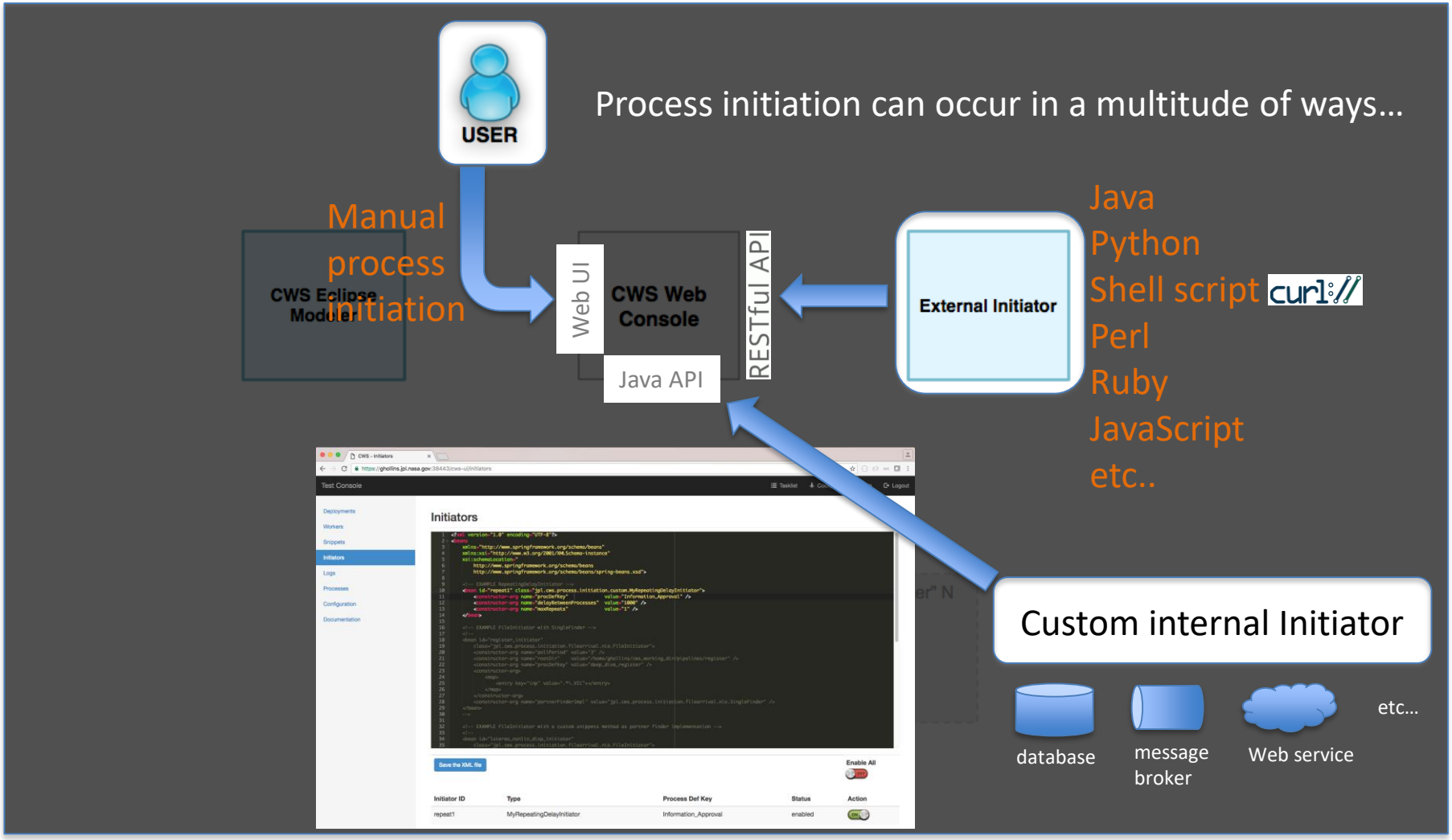
Process Initiation



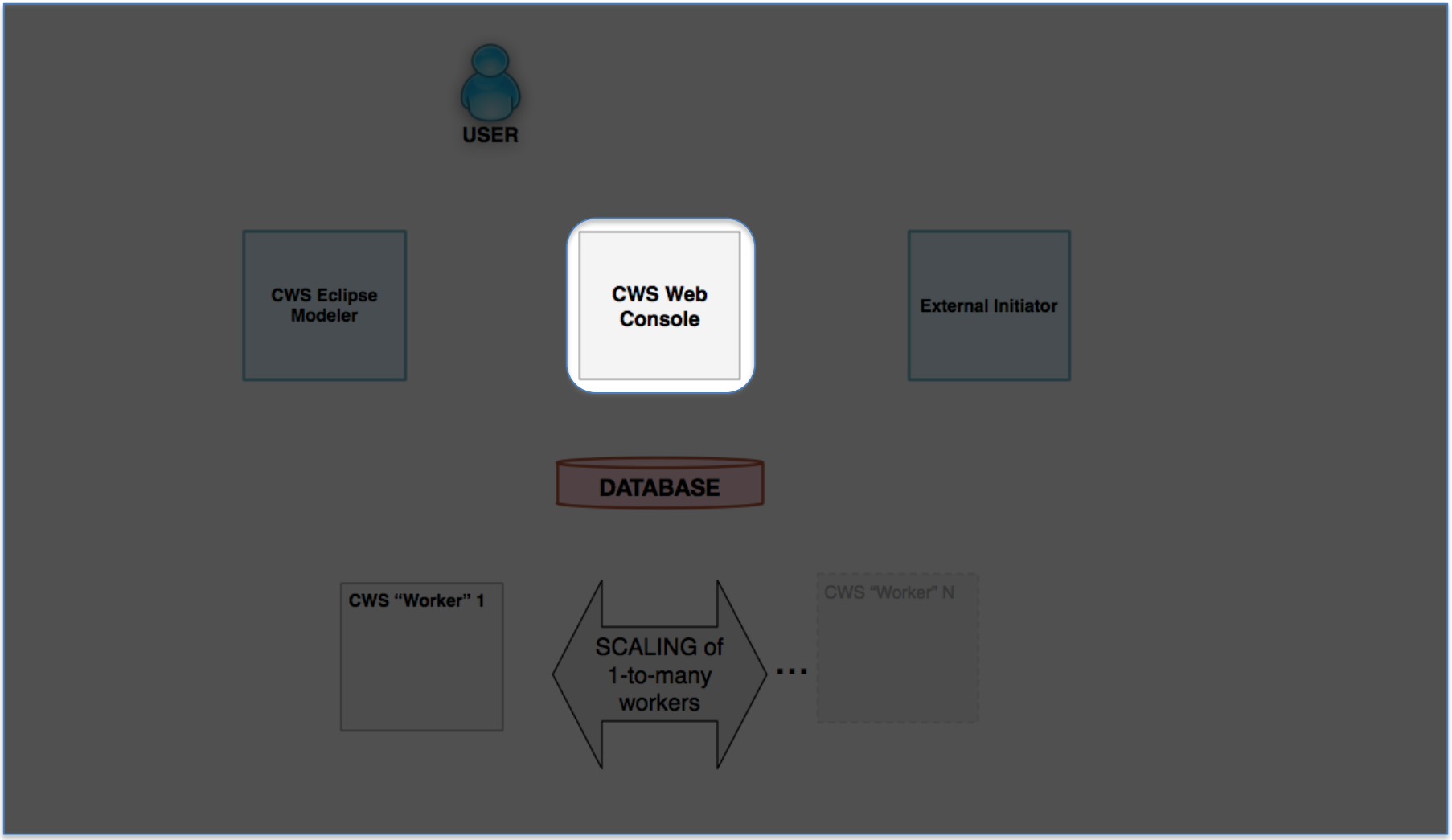
Process Initiation



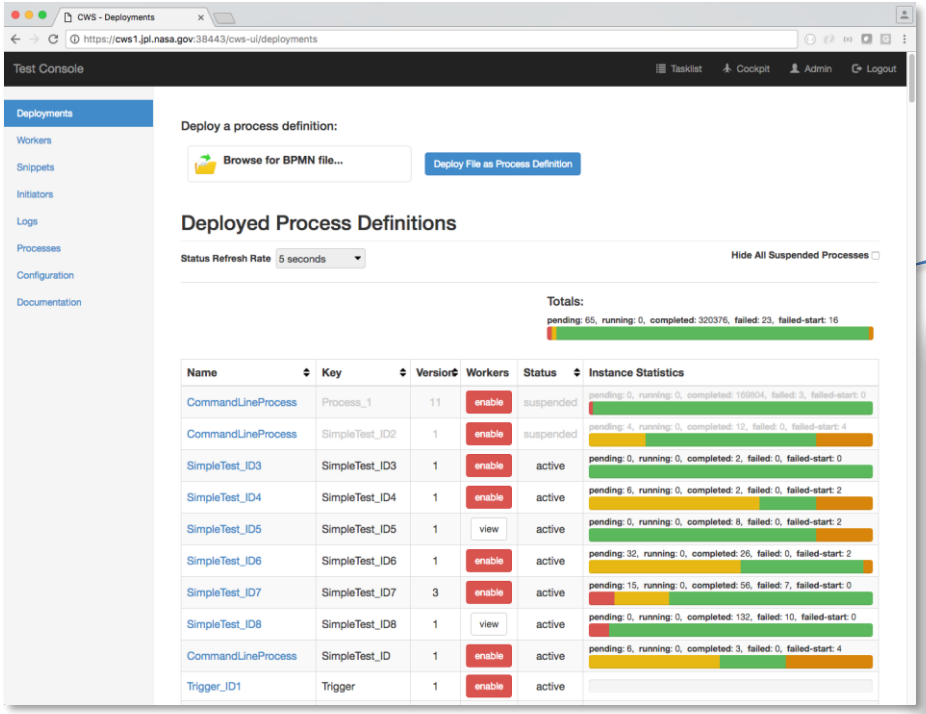
Process Initiation



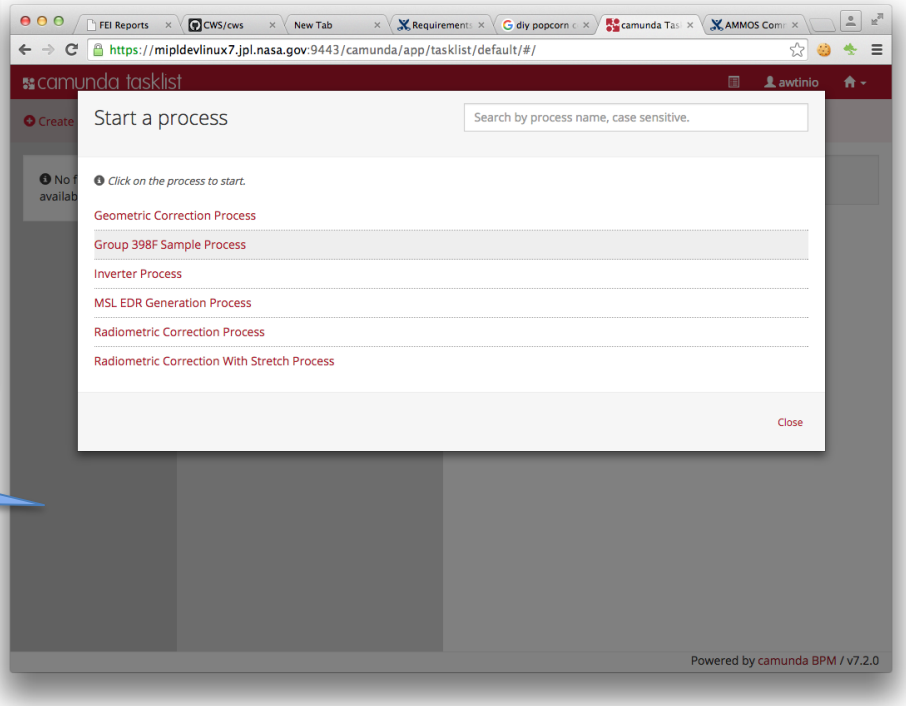
Process Monitoring and Management



Deploying and Executing Process



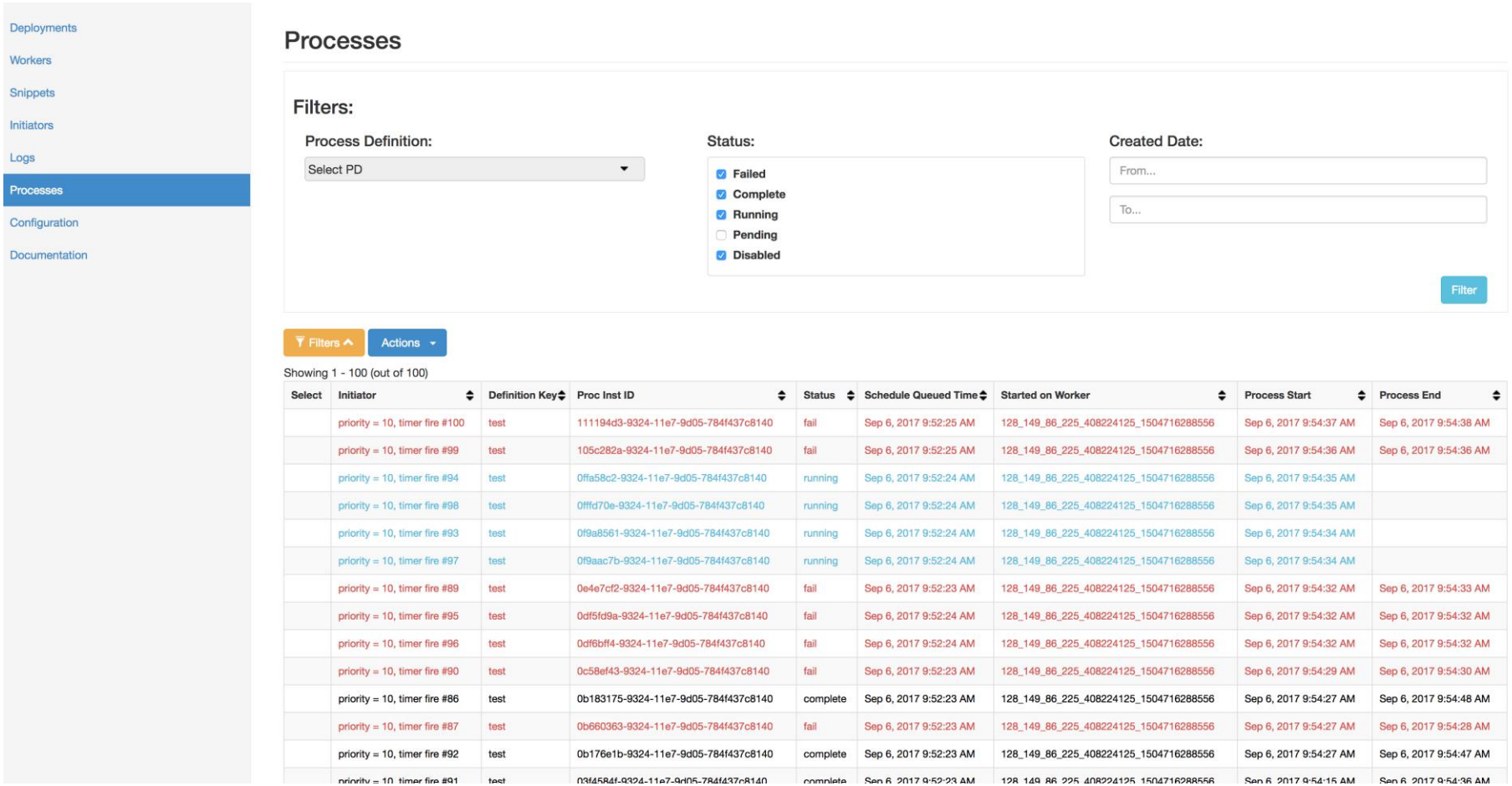
Process definitions deployed to the CWS via the web based console



Manual execution of processes can be done through the web console as well**

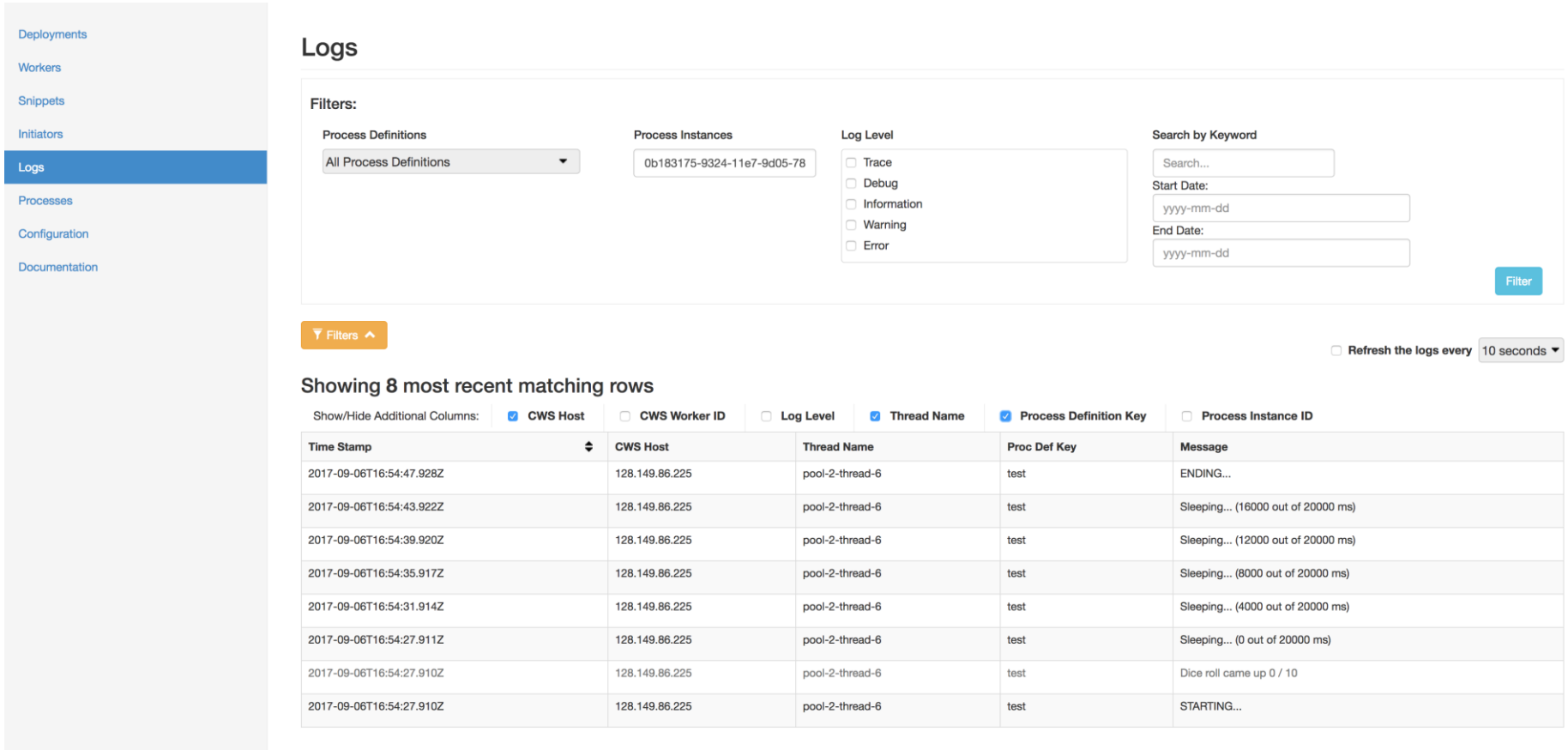
** Typical process executions are triggered automatically via event listeners

Monitoring and Management

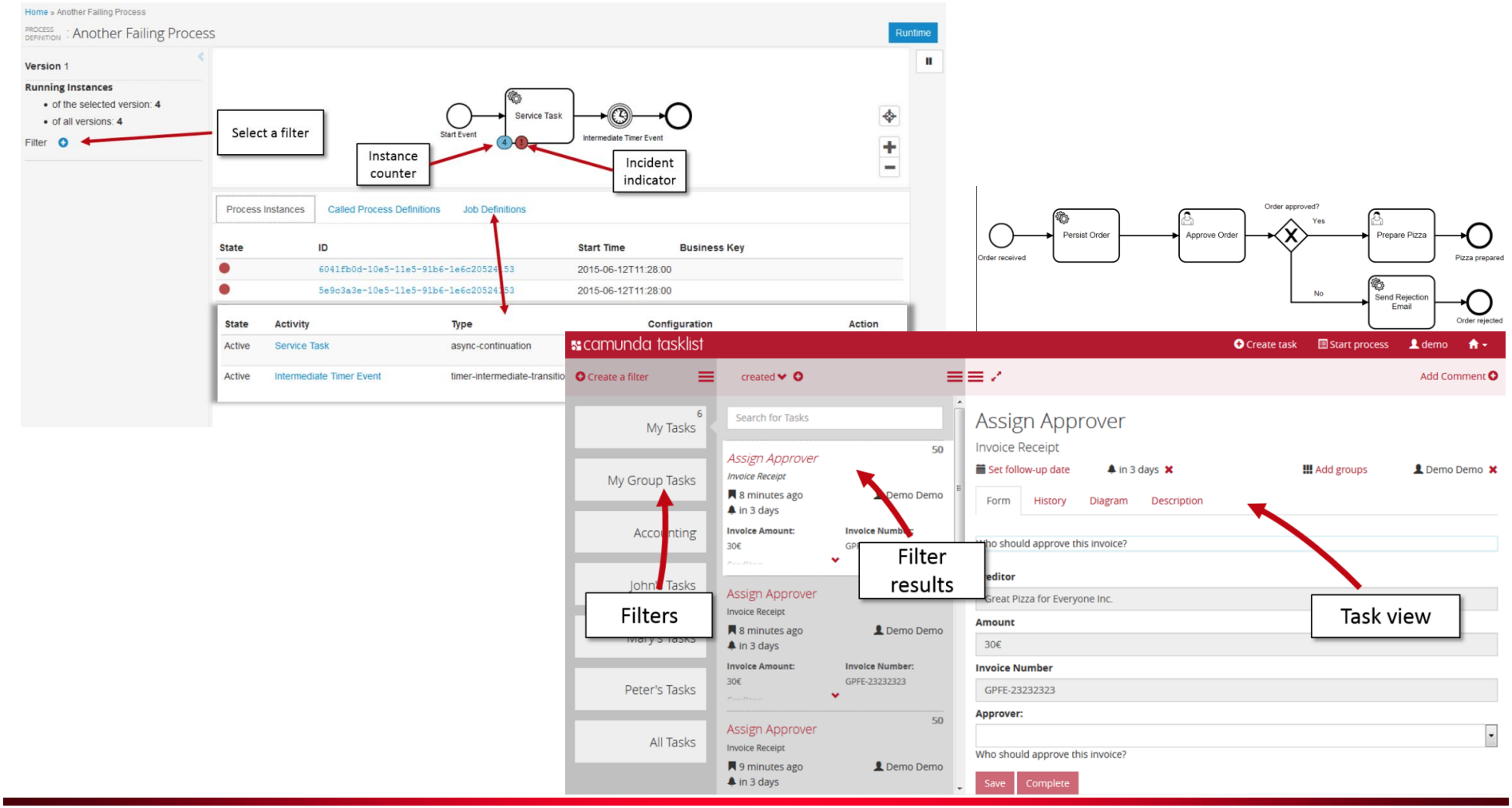


Select	Initiator	Definition Key	Proc Inst ID	Status	Schedule Queued Time	Started on Worker	Process Start	Process End
	priority = 10, timer fire #100	test	111194d3-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:25 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:37 AM	Sep 6, 2017 9:54:38 AM
	priority = 10, timer fire #99	test	105c282a-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:25 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:36 AM	Sep 6, 2017 9:54:36 AM
	priority = 10, timer fire #94	test	0ffa58c2-9324-11e7-9d05-784f437c8140	running	Sep 6, 2017 9:52:24 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:35 AM	
	priority = 10, timer fire #98	test	0fffd70e-9324-11e7-9d05-784f437c8140	running	Sep 6, 2017 9:52:24 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:35 AM	
	priority = 10, timer fire #93	test	0f9a8561-9324-11e7-9d05-784f437c8140	running	Sep 6, 2017 9:52:24 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:34 AM	
	priority = 10, timer fire #97	test	0f9aac7b-9324-11e7-9d05-784f437c8140	running	Sep 6, 2017 9:52:24 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:34 AM	
	priority = 10, timer fire #89	test	0e4e7cf2-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:23 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:32 AM	Sep 6, 2017 9:54:33 AM
	priority = 10, timer fire #95	test	0df5fd9a-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:24 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:32 AM	Sep 6, 2017 9:54:32 AM
	priority = 10, timer fire #96	test	0df6bff4-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:24 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:32 AM	Sep 6, 2017 9:54:32 AM
	priority = 10, timer fire #90	test	0c58ef43-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:23 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:29 AM	Sep 6, 2017 9:54:30 AM
	priority = 10, timer fire #86	test	0b183175-9324-11e7-9d05-784f437c8140	complete	Sep 6, 2017 9:52:23 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:27 AM	Sep 6, 2017 9:54:48 AM
	priority = 10, timer fire #87	test	0b660363-9324-11e7-9d05-784f437c8140	fail	Sep 6, 2017 9:52:23 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:27 AM	Sep 6, 2017 9:54:28 AM
	priority = 10, timer fire #92	test	0b176e1b-9324-11e7-9d05-784f437c8140	complete	Sep 6, 2017 9:52:23 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:27 AM	Sep 6, 2017 9:54:47 AM
	priority = 10, timer fire #91	test	03f4584f-9324-11e7-9d05-784f437c8140	complete	Sep 6, 2017 9:52:23 AM	128_149_86_225_408224125_1504716288556	Sep 6, 2017 9:54:15 AM	Sep 6, 2017 9:54:36 AM

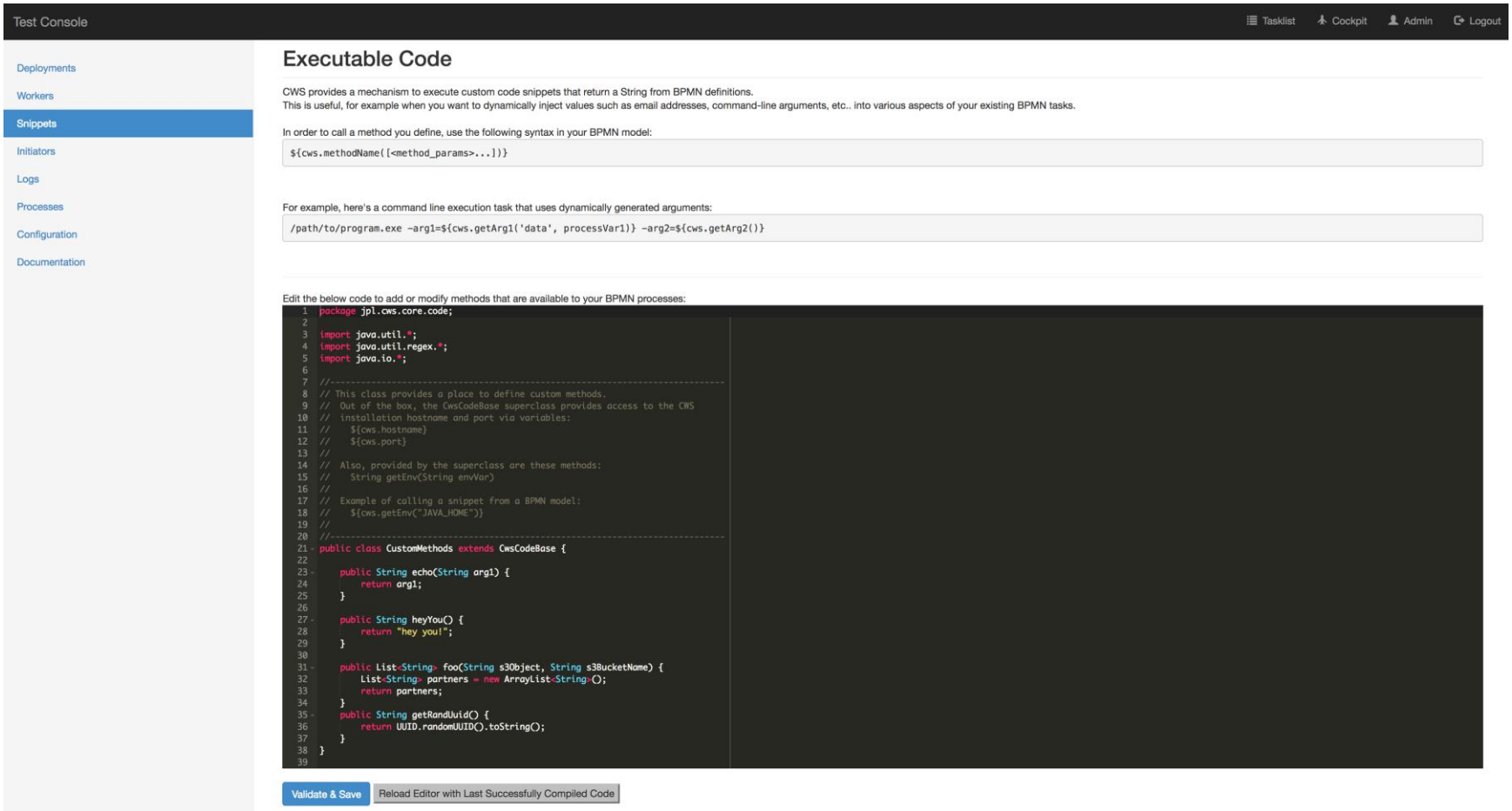
Monitoring and Management



Monitoring and Management



Code Snippets



Code Snippets Example Usage

In order to call a method you define, use the following syntax in your BPMN model:

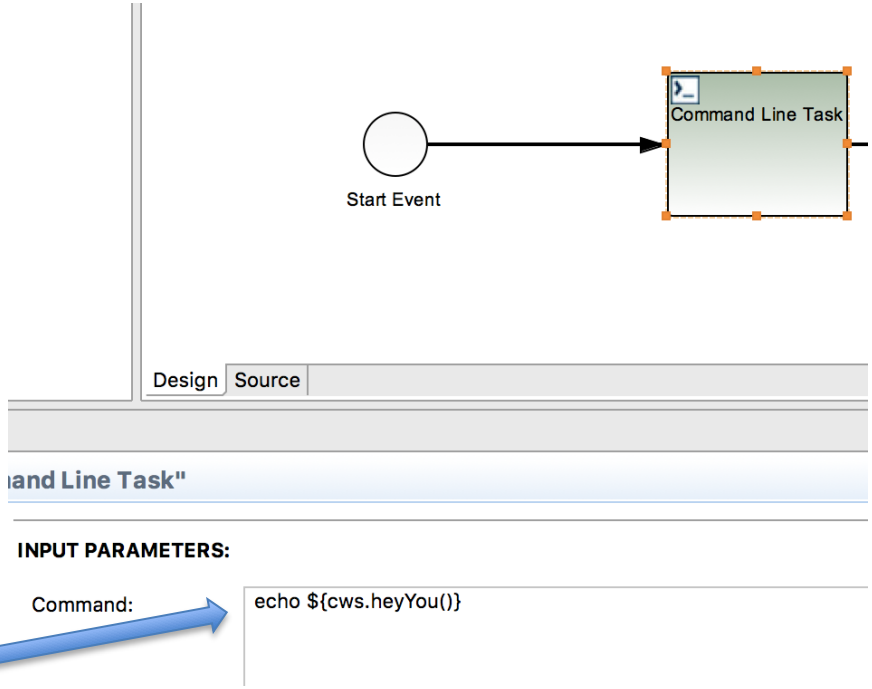
```
${cws.methodName([<method_params>...])}
```

For example, here's a command line execution task that uses dynamically generated arguments:

```
/path/to/program.exe -arg1=${cws.getArg1('data', processVar1)} -arg2=${cws.getArg2()}
```

Edit the below code to add or modify methods that are available to your BPMN processes:

```
1  package jpl.cws.core.code;
2
3  import java.util.*;
4  import java.util.regex.*;
5  import java.io.*;
6
7  //------------------------------------------------------------------------
8  // This class provides a place to define custom methods.
9  //  Out of the box, the CwsCodeBase superclass provides access to the CWS
10 //  installation hostname and port via variables:
11 //    ${cws.hostname}
12 //    ${cws.port}
13 //
14 //  Also, provided by the superclass are these methods:
15 //    String getEnv(String envVar)
16 //
17 //  Example of calling a snippet from a BPMN model:
18 //    ${cws.getEnv("JAVA_HOME")}
19 //
20 //------------------------------------------------------------------------
21 public class CustomMethods extends CwsCodeBase {
22
23     public String echo(String arg1) {
24         return arg1;
25     }
26
27     public String heyYou() {
28         return "hey you!";
29     }
30
31     public List<String> foo(String s3Object, String s3BucketName) {
32         List<String> partners = new ArrayList<String>();
33         return partners;
34     }
35     public String getRandUuid() {
36         return UUID.randomUUID().toString();
37     }
```



CWS Features Summary

- *Process Execution Engine*
  - BPMN 2.0 execution engine
  - Orchestration of **automated** tasks AND **user** tasks
- *Drag and Drop Process Modeler*
  - *Library of Common Tasks*
- *Human/User Task Management*
  - Handles life cycle of human tasks (assigning, claiming, completing, etc…)
- *Distributed and Load Balanced*
  - Execution of processes can be distributed across computing nodes/workers
- *Event-based Process Initiation*
  - *Plug-in architecture where custom event listeners can be implemented and plugged-in.*
- *Web-based Control and Management Interface*
- *Security Features*
  - Pluggable authentication model
  - Secured connections

CWS Future Work

- *Move away from Eclipse modeler, and go with Camunda / Web modeler*

- *Out-of-the box components and installation for external tasks*

JPL

Demo

Contact Info

Galen Hollins (CWS Task Lead / Software Engineer)

Jet Propulsion Laboratory
California Institute of Technology Pasadena CA

Galen.A.Hollins@jpl.nasa.gov

Jim Wood (CWS Software Engineer)

Jet Propulsion Laboratory
California Institute of Technology Pasadena CA

James.F.Wood@jpl.nasa.gov

Adrian Tinio (MGSS IDS System Engineer)

Jet Propulsion Laboratory
California Institute of Technology Pasadena CA

Adrian.W.Tinio@jpl.nasa.gov

JPL

Questions/Comments