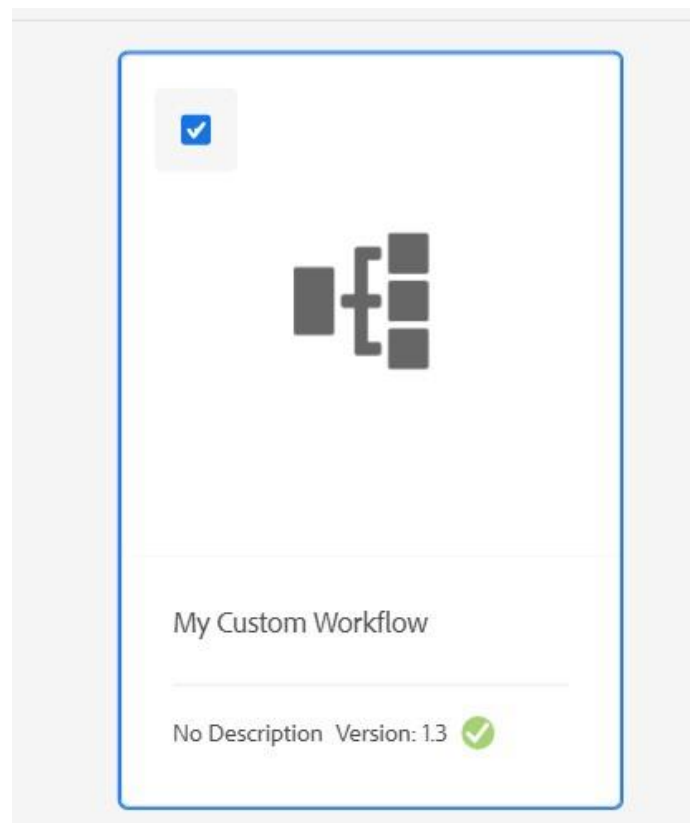
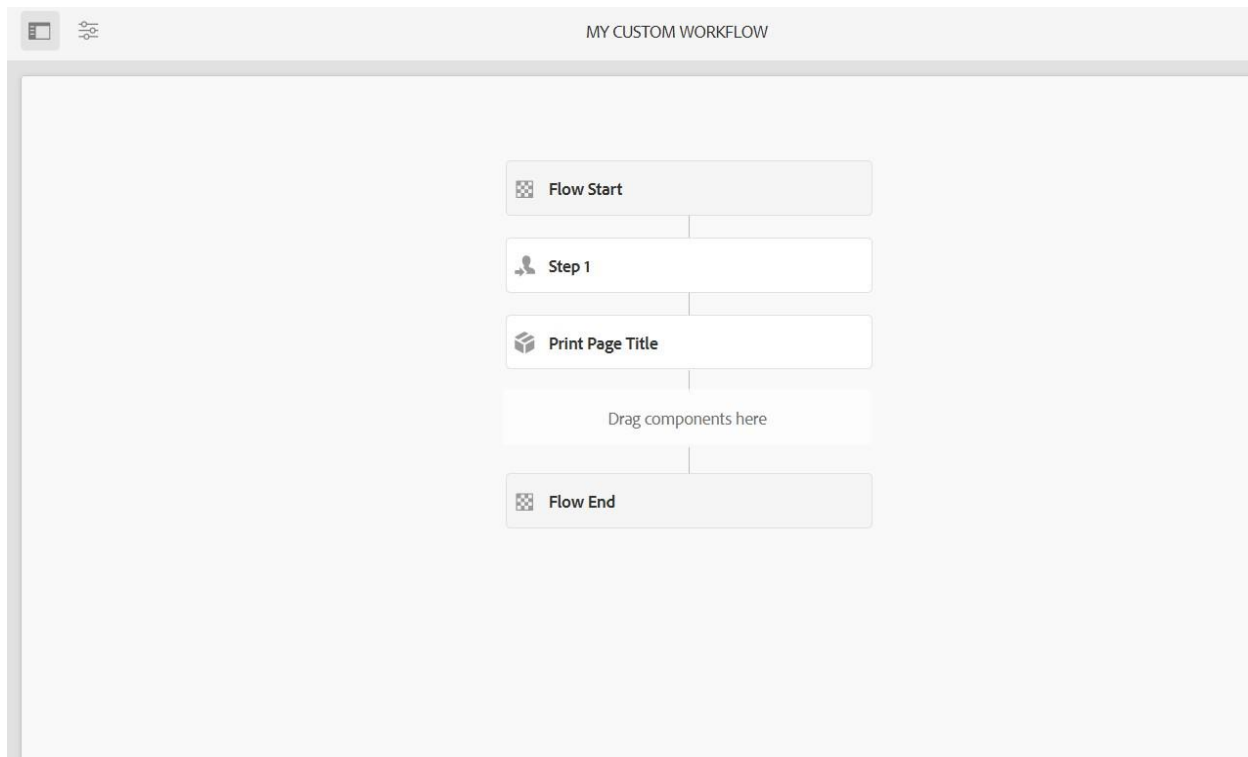


1. Create Custom Workflow (my custom workflow) and



2. Create custom workflow process and print the page title in logs and run this workflow in page so that it can give some metadata in logs

```
README.md  <> newsroom.html  MyCustomWorkflowProcess.java  </> _cq_dialog\content.xml  ⌵  ⋮

10  import org.osgi.service.component.annotations.Reference;
11  import org.slf4j.Logger;
12  import org.slf4j.LoggerFactory;
13  import com.adobe.granite.workflow.exec.WorkflowProcess;
14
15  import java.util.Collections;
16
17  @Component(service = WorkflowProcess.class, property = {"process.label=My CustomWorkflow Process"}) 1 usage new *
18  public class MyCustomWorkflowProcess implements WorkflowProcess {
19
20      private static final Logger LOG = 2 usages
21          LoggerFactory.getLogger(MyCustomWorkflowProcess.class);
22
23      @Reference 1 usage
24      private ResourceResolverFactory resourceResolverFactory;
25
26      @Override no usages new *
27      @
28      public void execute(WorkItem workItem, WorkflowSession workflowSession,
29                          WorkflowNode workflowNode) throws WorkflowException {
30          String pagePath = workItem.getWorkflowData().getPayload().toString();
31
32          try (ResourceResolver resourceResolver =
33              resourceResolverFactory.getServiceResourceResolver(Collections.singletonMap(ResourceRes_olverFacto
34              String pageTitle =
35                  resourceResolver.getResource(pagePath).getValueMap().get("s: jcr:title", String.class);
36              LOG.info("Workflow executed for page: {}, Title: {}", pagePath, pageTitle);
37          } catch (Exception e) {
38              LOG.error("Error processing workflow: ", e);
39          }
40      }
41  }
```

3. Create Event handler in aem and print the resource path in logs.

```
2
3  import org.apache.sling.api.SlingConstants;
4  import org.osgi.service.component.annotations.Component;
5  import org.osgi.service.event.Event; import org.osgi.service.event.EventConstants; imp
6  import org.slf4j.LoggerFactory;
7
8  @Component( 1 usage new *
9      service = EventHandler.class,          immediate = true,          property = {
10      EventConstants.EVENT_TOPIC + "=" + SlingConstants.TOPIC_RESOURCE_ADDED
11  }
12  )
13  public class ResourceEventHandler implements EventHandler {
14
15      private static final Logger LOG = LoggerFactory.getLogger(ResourceEventHandler.class);
16
17      @Override new *
18      @
19      public void handleEvent(Event event) {
20          String resourcePath = (String) event.getProperty(SlingConstants.PROPERTY_PATH);
21      }
22  }
```

4. create sling job to print hello world message in logs

```
MyCustomWorkflowProcess.java  ResourceEventHandler.java  HelloWorldSlingJob.java x  v  :  @
m

4      import org.apache.sling.event.jobs.consumer.JobConsumer; import org.osgi.framework.BundleContext;
5      import org.slf4j.LoggerFactory;
6
7      import java.util.Map;
8
9      @Component( 1usage new *
10          service = JobConsumer.class,
11          property = {
12              JobConsumer.PROPERTY_TOPICS + "=assignment/helloworldjob"
13          }
14      )
15      public class HelloWorldSlingJob implements JobConsumer {
16
17          private static final Logger LOG = LoggerFactory.getLogger(HelloWorldSlingJob.class);
18
19          @Override new *
20          public JobResult process(Job job) {          LOG.info("Hello World from Sling Job!");
21          }
22
23          @Override new *
24          public JobResult process(Job job) {
25              return null;
26          }
27      }
```

5. Create one scheduler to print the yellow world in logs in every 5 mins through custom configuration using cron expression.

```
import org.apache.sling.commons.scheduler.Scheduler; import org.osgi.service.component.annotations.Component;
import org.slf4j.LoggerFactory;

@Component(service = Runnable.class, immediate = true, property = {
    "scheduler.expression=0 */5 * * * ?",
    "scheduler.concurrent=false"
})
@Designate(ocd = YellowWorldScheduler.Config.class) public class YellowWorldScheduler implements Runnable {

    private static final Logger LOG = LoggerFactory.getLogger(YellowWorldScheduler.class);

    @Reference
    private Scheduler scheduler;

    @Activate
    protected void activate() {
        LOG.info("Yellow World Scheduler Activated");
    }

    @Override public void run() {
        LOG.info("Yellow World from Scheduler!");
    }
}
```

6. Create 3 users and add them in a group(Dev author create this new group) and give permission to read only for /content and /dam folder only and they should have replication access as well.

Go to <http://localhost:4502/useradmin>

- 1. Click "Create User" ○ User 1: user1 ○ User 2: user2 ○ User 3: user3**
- 2. Click "Create Group" → Name: "Dev Author"**
- 3. Add user1, user2, and user3 to "Dev Author" group**

Assign Permissions

- 1. Select "Dev Author" Group**
- 2. Go to "Permissions" Tab**
- 3. Set Read-Only Access to /content and /dam ○ Path: /content
→ Permission: Read ○ Path: /dam → Permission: Read**
- 4. Enable Replication Access ○ Path: /etc/replication → Allow Replication**