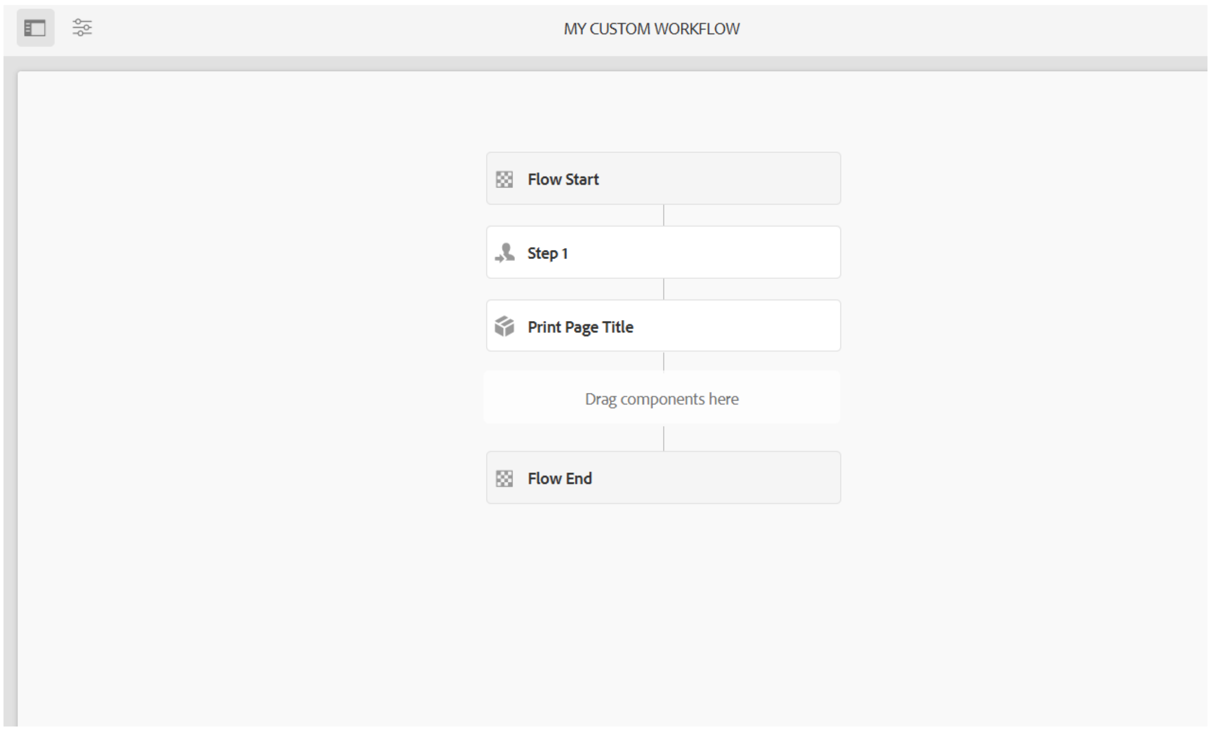
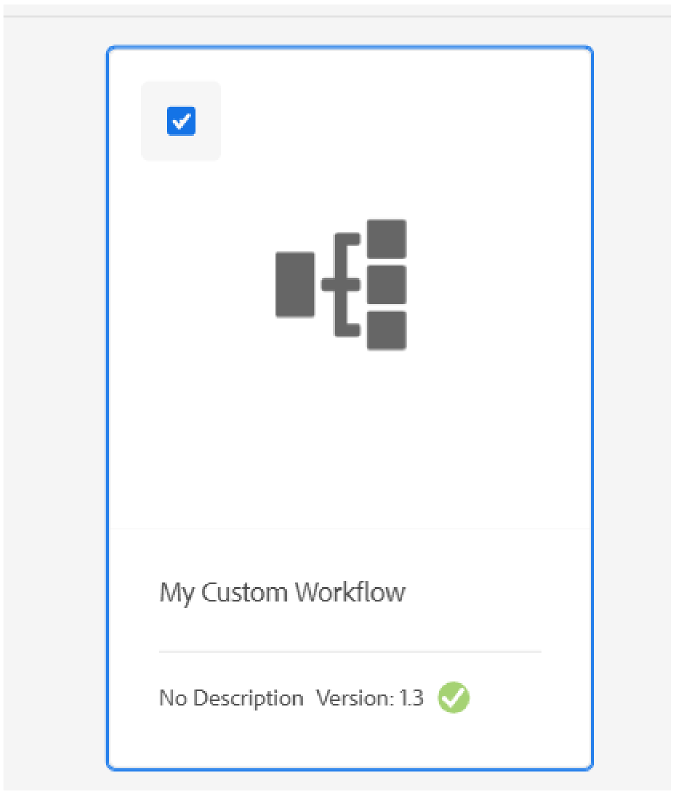
1.Create Custom Workflow (my custom workflow)





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

Java Class: MyCustomWorkflowProcess.java

Location:

core/src/main/java/com/assignment/workflows/MyCustomWorkflowProcess.java

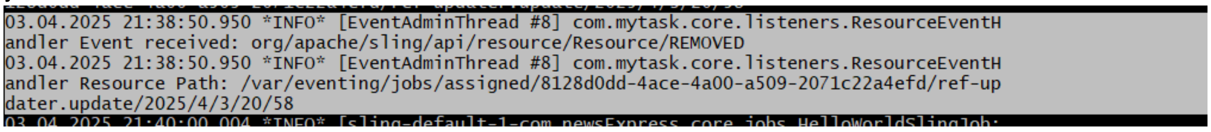
```
package com.assignment.workflows;


import com.adobe.granite.workflow.WorkflowException;

import com.adobe.granite.workflow.exec.WorkItem;

import com.adobe.granite.workflow.exec.WorkflowSession;

import com.adobe.granite.workflow.model.WorkflowNode;

import org.apache.sling.api.resource.ResourceResolver;

import org.apache.sling.api.resource.ResourceResolverFactory;

import org.osgi.service.component.annotations.Component;

import org.osgi.service.component.annotations.Reference;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.adobe.granite.workflow.exec.WorkflowProcess;


import java.util.Collections;


@Component(service = WorkflowProcess.class, property = {"process.label=My Custom
Workflow Process"})

public class MyCustomWorkflowProcess implements WorkflowProcess {


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


   @Reference

   private ResourceResolverFactory resourceResolverFactory;


   @Override

   public void execute(WorkItem workItem, WorkflowSession workflowSession,
WorkflowNode workflowNode) throws WorkflowException {
        String pagePath = workItem.getWorkflowData().getPayload().toString();


        try (ResourceResolver resourceResolver =
resourceResolverFactory.getServiceResourceResolver(Collections.singletonMap(ResourceRes
olverFactory.SUBSERVICE, "workflowUser"))) {
            String pageTitle =
resourceResolver.getResource(pagePath).getValueMap().get("jcr:title", String.class);
            LOG.info("Workflow executed for page: {}, Title: {}", pagePath, pageTitle);
        } catch (Exception e) {
            LOG.error("Error processing workflow: ", e);
        }
    }
}
```

3.Create Event handler in aem and print the resource path in logs.
The event handler will log the **resource path** whenever a new node is created.

Java Class: ResourceEventHandler.java

Location:
core/src/main/java/com/assignment/listeners/ResourceEventHandler.java

```
package com.assignment.listeners;

import org.apache.sling.api.SlingConstants;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventConstants;
import org.osgi.service.event.EventHandler;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@Component(
    service = EventHandler.class,
    immediate = true,
    property = {
        EventConstants.EVENT_TOPIC + "=" + SlingConstants.TOPIC_RESOURCE_ADDED
    }
)
public class ResourceEventHandler implements EventHandler {
    private static final Logger LOG = LoggerFactory.getLogger(ResourceEventHandler.class);

    @Override
    public void handleEvent(Event event) {
        String resourcePath = (String) event.getProperty(SlingConstants.PROPERTY_PATH);
        LOG.info("Resource added at path: {}", resourcePath);
    }
}
```



```
03.04.2025 21:38:50.950 *INFO* [EventAdminThread #8] com.mytask.core.listeners.ResourceEventH
andler Event received: org/apache/sling/api/resource/Resource/REMOVED
03.04.2025 21:38:50.950 *INFO* [EventAdminThread #8] com.mytask.core.listeners.ResourceEventH
andler Resource Path: /var/eventing/jobs/assigned/8128d0dd-4ace-4a00-a509-2071c22a4efd/ref-up
dater.update/2025/4/3/20/58
03.04.2025 21:40:00.004 *INFO* [sling-default-1-com.newsExpress.core.jobs.HelloWorldSlingJob:
```

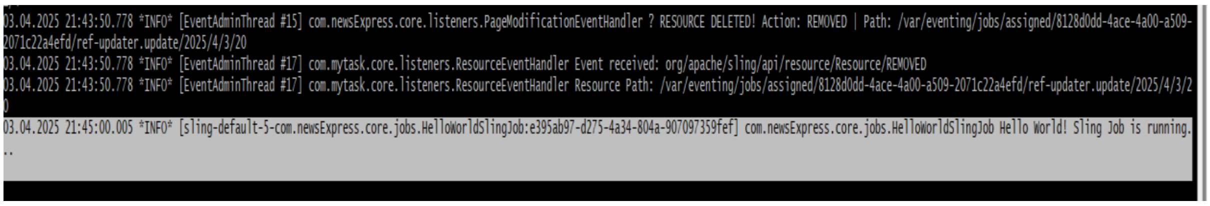
4. create sling job to print hello world message in logs

This job will run asynchronously and print "Hello World" in logs.

Java Class: HelloWorldSlingJob.java

Location: core/src/main/java/com/assignment/jobs/HelloWorldSlingJob.java

```
package com.assignment.jobs;

import org.apache.sling.event.jobs.consumer.JobConsumer;
import org.osgi.service.component.annotations.Component;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Map;

@Component(
        service = JobConsumer.class,
        property = {
                JobConsumer.PROPERTY_TOPICS + "=assignment/helloworldjob"
        }
)
public class HelloWorldSlingJob implements JobConsumer {

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

    @Override
    public JobResult process(Job job) {
        LOG.info("Hello World from Sling Job!");
        return JobResult.OK;
    }
}
```



```
03.04.2025 21:43:50.778 *INFO* [EventAdminThread #15] com.newsExpress.core.listeners.PageModificationEventHandler ? RESOURCE DELETED! Action: REMOVED | Path: /var/eventing/jobs/assigned/8128d0dd-4ace-4a00-a509-
2071c22a4efd/ref-updater.update/2025/4/3/20
03.04.2025 21:43:50.778 *INFO* [EventAdminThread #17] com.mytask.core.listeners.ResourceEventHandler Event received: org/apache/sling/api/resource/Resource/REMOVED
03.04.2025 21:43:50.778 *INFO* [EventAdminThread #17] com.mytask.core.listeners.ResourceEventHandler Resource Path: /var/eventing/jobs/assigned/8128d0dd-4ace-4a00-a509-2071c22a4efd/ref-updater.update/2025/4/3/2
0
03.04.2025 21:45:00.005 *INFO* [sling-default-5-com.newsExpress.core.jobs.HelloWorldSlingJob:e395ab97-d275-4a34-804a-907097359fef] com.newsExpress.core.jobs.HelloWorldSlingJob Hello World! Sling Job is running.
..
```

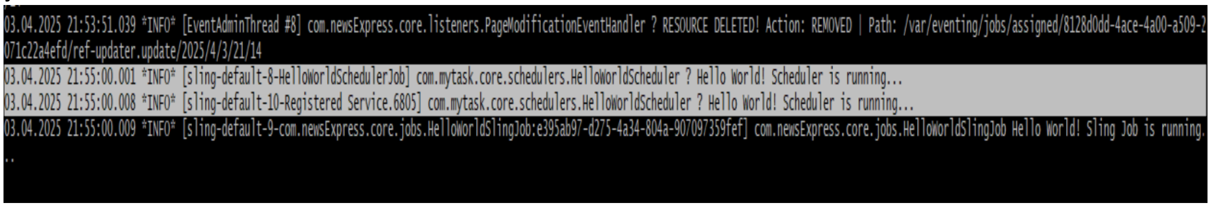
5. Create one schedular to print the yellow world in logs in every 5 mins through custom configuration using cron expression.

The scheduler prints "Yellow World" every 5 minutes using a **cron expression**.

Java Class: YellowWorldScheduler.java
Location:
core/src/main/java/com/assignment/schedulers/YellowWorldScheduler.java

```
package com.assignment.schedulers;

import org.apache.sling.commons.scheduler.Scheduler;
import org.osgi.service.component.annotations.Activate;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.Reference;
import org.osgi.service.metatype.annotations.Designate;
import org.slf4j.Logger;
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!");
```

```
   }
}
```



```
03.04.2025 21:53:51.039 *INFO* [EventAdminThread #8] com.newsExpress.core.listeners.PageModificationEventHandler ? RESOURCE DELETED! Action: REMOVED | Path: /var/eventing/jobs/assigned/8128d0dd-4ace-4a00-a509-2
071c22a4efd/ref-updater.update/2025/4/3/21/14
03.04.2025 21:55:00.001 *INFO* [sling-default-8-HelloWorldSchedulerJob] com.mytask.core.schedulers.HelloWorldScheduler ? Hello World! Scheduler is running...
03.04.2025 21:55:00.008 *INFO* [sling-default-10-Registered Service.6805] com.mytask.core.schedulers.HelloWorldScheduler ? Hello World! Scheduler is running...
03.04.2025 21:55:00.009 *INFO* [sling-default-9-com.newsExpress.core.jobs.HelloWorldSlingJob:e395ab97-d275-4a34-804a-907097359fef] com.newsExpress.core.jobs.HelloWorldSlingJob Hello World! Sling Job is running.
..
```

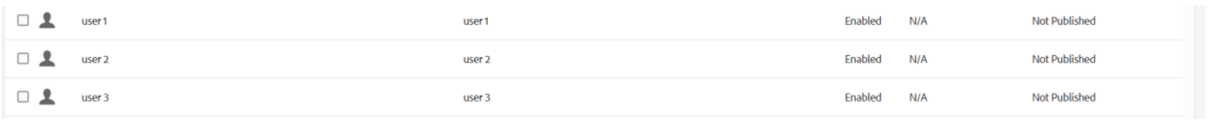
6. Create 3 users and add them in a group(Dev author create this new group) and give permission to read only for /content and /dam folder only and they should have replication access as well.

Go to http://localhost:4502/useradmin

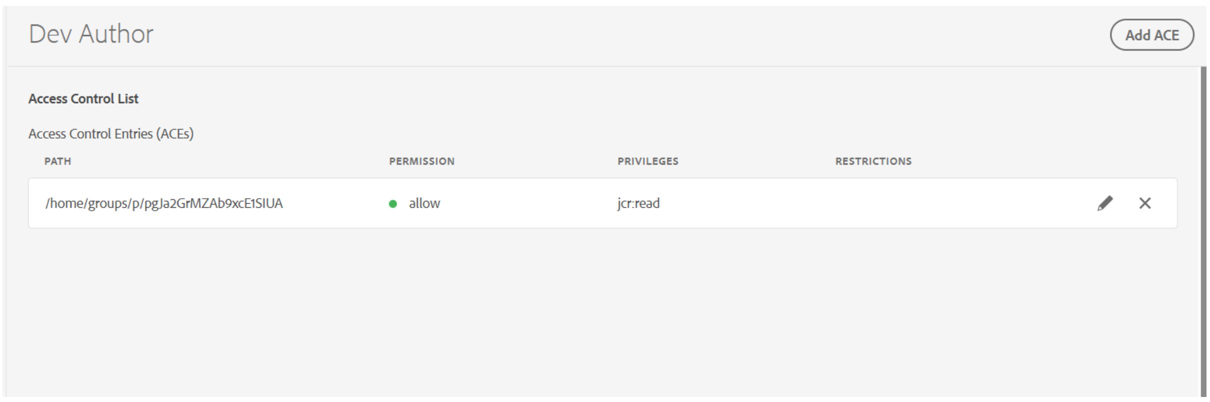
1. **Click "Create User"**
   - User 1: user1
   - User 2: user2
   - User 3: user3
2. **Click "Create Group" → Name: "Dev Author"**
3. **Add user1, user2, and user3 to "Dev Author" group**

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**



user 1	user 1	Enabled	N/A	Not Published
user 2	user 2	Enabled	N/A	Not Published
user 3	user 3	Enabled	N/A	Not Published



Access Control List

Access Control Entries (ACEs)

PATH	PERMISSION	PRIVILEGES	RESTRICTIONS
/home/groups/p/pgJa2GrMZAb9xcE1SIUA	● allow	jcr:read	