**Practical Exercise 23 - Using and Extending Our Composite Custom Action**

**Exercise Description**

**In this practical exercise, our goal is to explore how to use and how to extend our composite custom action.**

In order to achieve that, we will leverage a React application that we will scaffold with the help of the create-react-app utility. Check the tips section for the specific command to scaffold your React application. Here are the instructions for the exercise:

1. Generate a React application:
   1. Create a new folder named 17-custom-actions at the root of the repository.
   2. Using a terminal, cd into this directory and scaffold a React application inside a react-app directory. You can either create the directory yourself or let the create-react-app utility do it for you.
   3. Once the React setup is done, you should see a success message.
   4. Take a few moments to inspect the files and get familiar with the application folder structure.
2. Create a file named 17-1-custom-actions-composite.yaml under the .github/workflows folder at the root of your repository.
3. Name the workflow 17 - 1 - Custom Actions - Composite.
4. Add the following triggers with event filters and activity types to your workflow:
   1. workflow\_dispatch: the workflow\_dispatch should also receive an input named target-env, of type choice and with the following options: dev, prod.
5. Set the run-name option of the workflow to 17 - 1 - Custom Actions - Composite | env - <retrieve target-env input here>.
6. Define a top-level env variable named working-directory and with value 17-custom-actions/react-app.
7. Add a single job named build to the workflow.
   1. It should run on ubuntu-latest.
   2. It should set the default working-directory option for run commands to the value of the env variable working-directory.
   3. It should contain four steps:
      1. The first step should checkout the code.
      2. The second step, named Setup Node and NPM Dependencies, should use the recently created custom action. To reference a custom action created in the same repository as the workflow, you can simply provide the path of the directory where the action.yaml file is located. In this case, this would be ./.github/actions/composite-cache-deps.
         1. Do not forget to provide the necessary inputs via the with key!
      3. The third step, named Test, should run the npm run test command.
      4. The fourth step, named Build, should run the npm run build command.
8. Commit the changes and push the code. Trigger the workflow manually from the UI and take a few moments to inspect the result of the workflow run.
9. Now lets extend our custom action to allow us to omit dev dependencies during the installation process. This can be useful when building production versions of our application, since the bundle size is much smaller.
10. Extend our action.yaml file for the composite action by:
    1. Adding a third input, which should be named target-env, should not be required, have a description of "dev" or "prod". Controls whether dev dependencies are installed, and default to dev.
    2. Changing the Install dependencies step to conditionally determine whether it should run the npm ci or the npm ci --omit=dev variation of the script based on the received target-env value.
    3. Updating the cache key to include the target-env information in order to correctly match dev and prod caches.
11. Extend the 17-1-custom-actions.yaml workflow to pass the target-env input as a parameter to the custom action.
12. Commit the changes and push the code. Trigger the workflow manually from the UI with varying parameters and take a few moments to inspect the result of the workflow run.

**Tips**

**Scaffolding a React application with create-react-app**

To generate a React application with a single command, you can leverage the create-react-app utility. In order to do that, simply run the following inside the 17-custom-actions folder: npx create-react-app --template typescript react-app.