**Practical Exercise 15 - Using Caching to Speed Up Installing Dependencies**

**Exercise Description**

**In this practical exercise, our goal is to explore how we can use caching to temporarily store and retrieve the contents of folders between within and between workflow runs.**

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 13-caching 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 the first version of the workflow:
   1. Create a file named 13-caching.yaml under the .github/workflows folder at the root of your repository.
   2. Name the workflow 13 - Using Caching.
   3. Add the following triggers to your workflow:
      1. workflow\_dispatch: the workflow\_dispatch trigger should accept an input named use-cache, of type boolean, default value of true, and description Whether to execute cache step
   4. Add a single job named build to the workflow:
      1. The job should run on ubuntu-latest
      2. The job should set 13-caching/react-app as the default working-directory option for run commands.
      3. The job should contain six steps:
         1. The first one, named Checkout code, should checkout the repository code into the current working directory.
         2. The second one, named Setup Node, should setup Node with the version 20.x.
         3. The third one, named Install dependencies, should execute the npm ci command.
         4. The fourth one, named Testing, should execute the npm run test command.
         5. The fifth one, named Building, should execute the npm run build command.
         6. The sixth one, named Deploying to nonprod, should print the following message: "Deploying to nonprod".
   5. Commit the changes and push the code. Trigger the workflow a few times and inspect the running times of each step. How much time does it take to install the dependencies? How much time would it take if the workflow ran 1000 times?
3. Extend the workflow with both an input for node-version and the possibility of using caching. In order to do that:
   1. Add a new input named node-version, of type choice, options 18.x, 20.x, and 21.x, default value of 20.x, and description Node version.
   2. Update the Setup Node step to use the value provided as an input instead of the hard-coded value 20.x.
   3. Add a new step after the Setup Node step:
      1. It should be named Download cached dependencies.
      2. It should have an id of cache.
      3. It should run only if the input use-cache is set to true.
      4. It should use the actions/cache@v3 third-party action.
      5. It should pass the following inputs to the action:
         1. The key should be deps-node-modules-${{ hashFiles('13-caching/react-app/package-lock.json') }}
            1. It's important to provide a key that remains stable as long as the list of dependencies does not change, and that changes as soon as the list of dependencies changes. This is easily achieved by hashing the package-lock.json file, since this file contains a list of all the dependencies, both direct and indirect, of our application.
         2. The path where to restore the cache should be 13-caching/react-app/node\_modules. It's important to provide the entire path, since the working-directory option set as default at the job level applies only to run commands.
   4. Update the Install dependencies step to run if and only if no cache is found for the provided key. This can be done by leveraging the outputs of our cache step. If there is a cache, the cache-hit output will be set to 'true' (a string, not a boolean).
   5. Commit the changes and push the code. Trigger the workflow a few times and inspect the running times of each step. How much time does it take to install the dependencies? How much time would it take if the workflow ran 1000 times? How big is the difference when compared to the version without caching?

**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 13-caching folder: npx create-react-app --template typescript react-app.