实验：设置 GitHub 环境



预计所需时间： 15 分钟

您将学习的内容

在本实验中，您将为项目设置一个在线代码库，并学习如何在 **Skills Network Environment** 中创建文件夹和文件。您还将了解如何查看代码的输出。

学习目标

完成本实验后，您将能够：

- 创建一个在线代码库。
- 在 **Skills Network Environment** 中创建文件夹和文件。
- 创建并运行 React 应用程序。
- 执行 Git 操作。

先决条件

- 您必须完成先决课程，特别是 **Git 和 GitHub 入门** 课程。
- 您必须拥有一个 GitHub 账户。如果您想设置 GitHub 账户，请点击 **这里** 查看详细步骤。

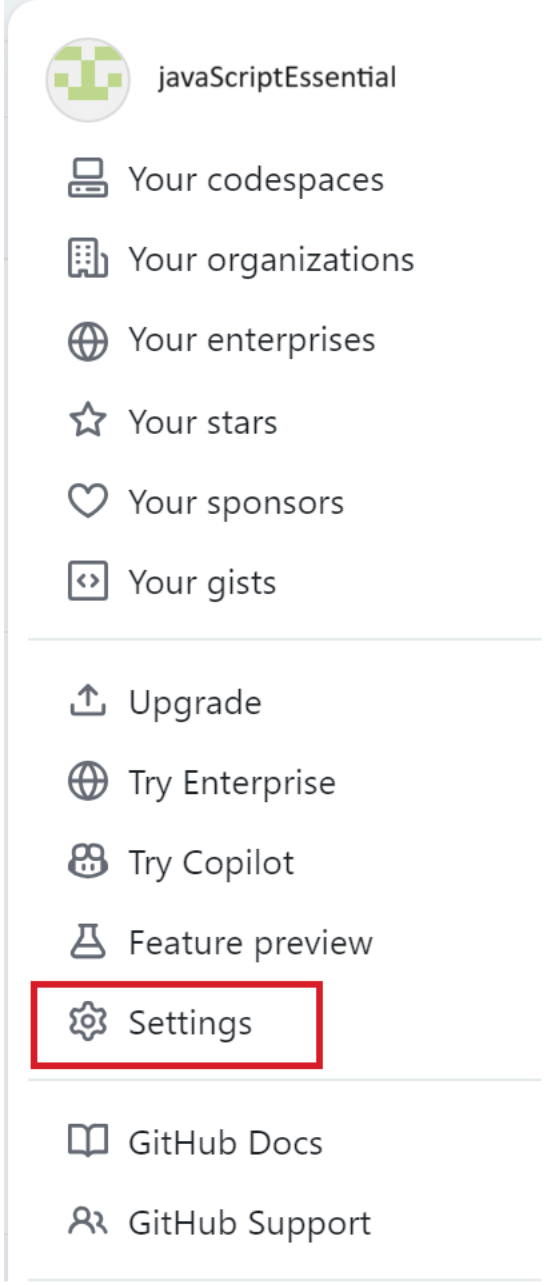
需要执行 Git 命令

遵循一些基本步骤对于确保在 GitHub 仓库中正确管理和持久化您的数据至关重要：

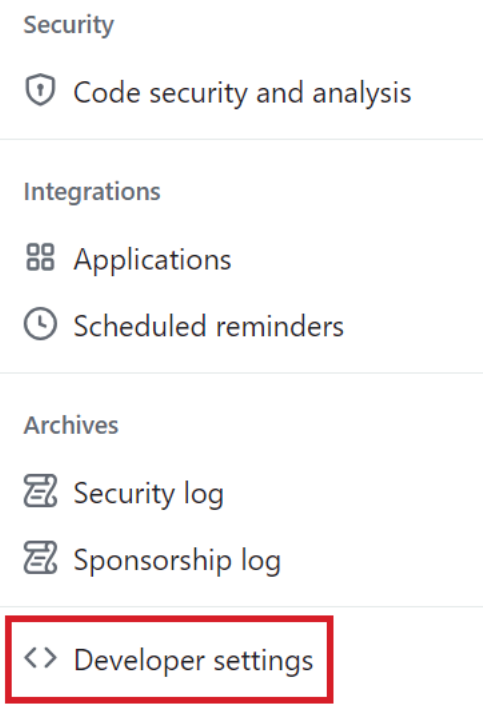
- **定期更新：** 每当您对项目进行更改或添加新组件时，将更新添加、提交并推送到您的 GitHub 仓库是至关重要的。此过程确保您的最新工作安全存储，并可供协作者访问。
- **会话持久性：** 在活动会话期间，您的数据保持可访问。然而，需要注意的是，如果您的会话过期或您注销，您将需要重新克隆仓库才能继续工作。

任务 1：创建你的代码库并生成个人访问令牌

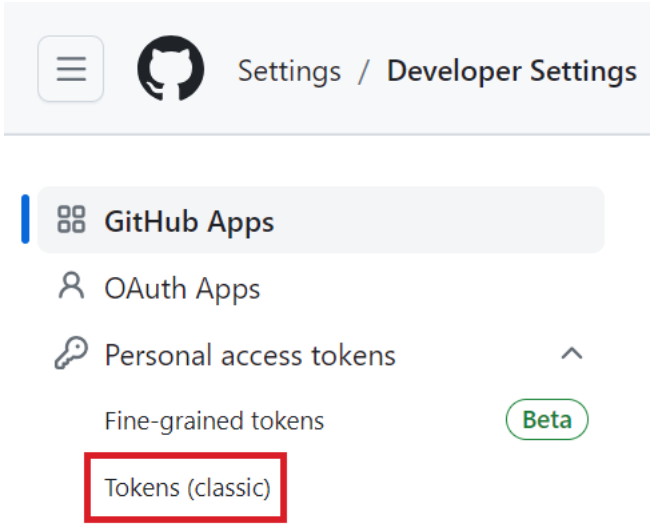
1. 在你的 GitHub 账户中创建一个空白的公共 GitHub 代码库，且不创建任何 README.md 文件。确保将你的代码库设置为公共，并相应地命名。
2. 你可以在 GitHub 注册和创建代码库 实验中找到更多信息。
3. 要将文件推送到 GitHub 代码库，你必须拥有一个 **个人访问令牌** 以确保你的身份验证与 GitHub 账户相关联。
4. 要创建 **个人访问令牌**，请访问你的 GitHub 账户，并点击右上角的个人资料图标。然后点击 **设置**。



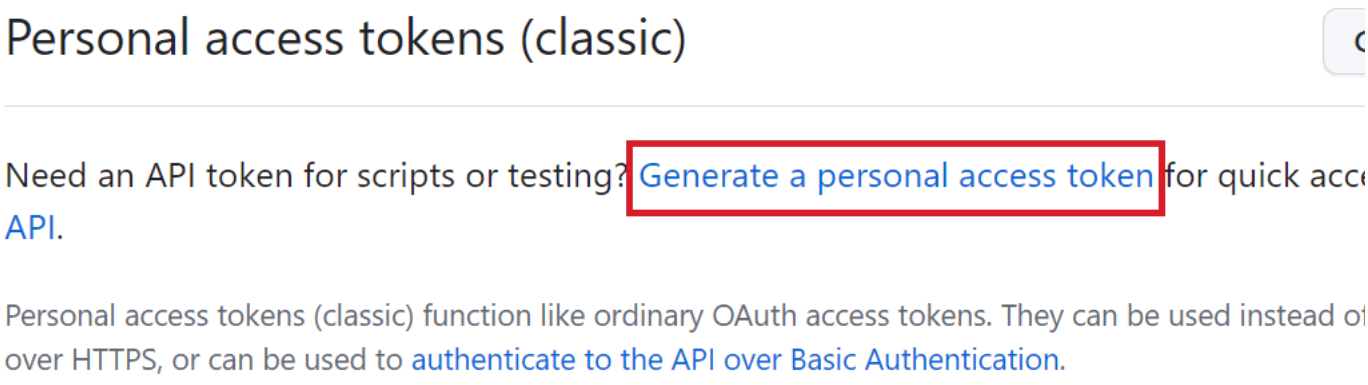
5. 接下来，选择 **开发者设置**。此选项通常位于窗口的底部。



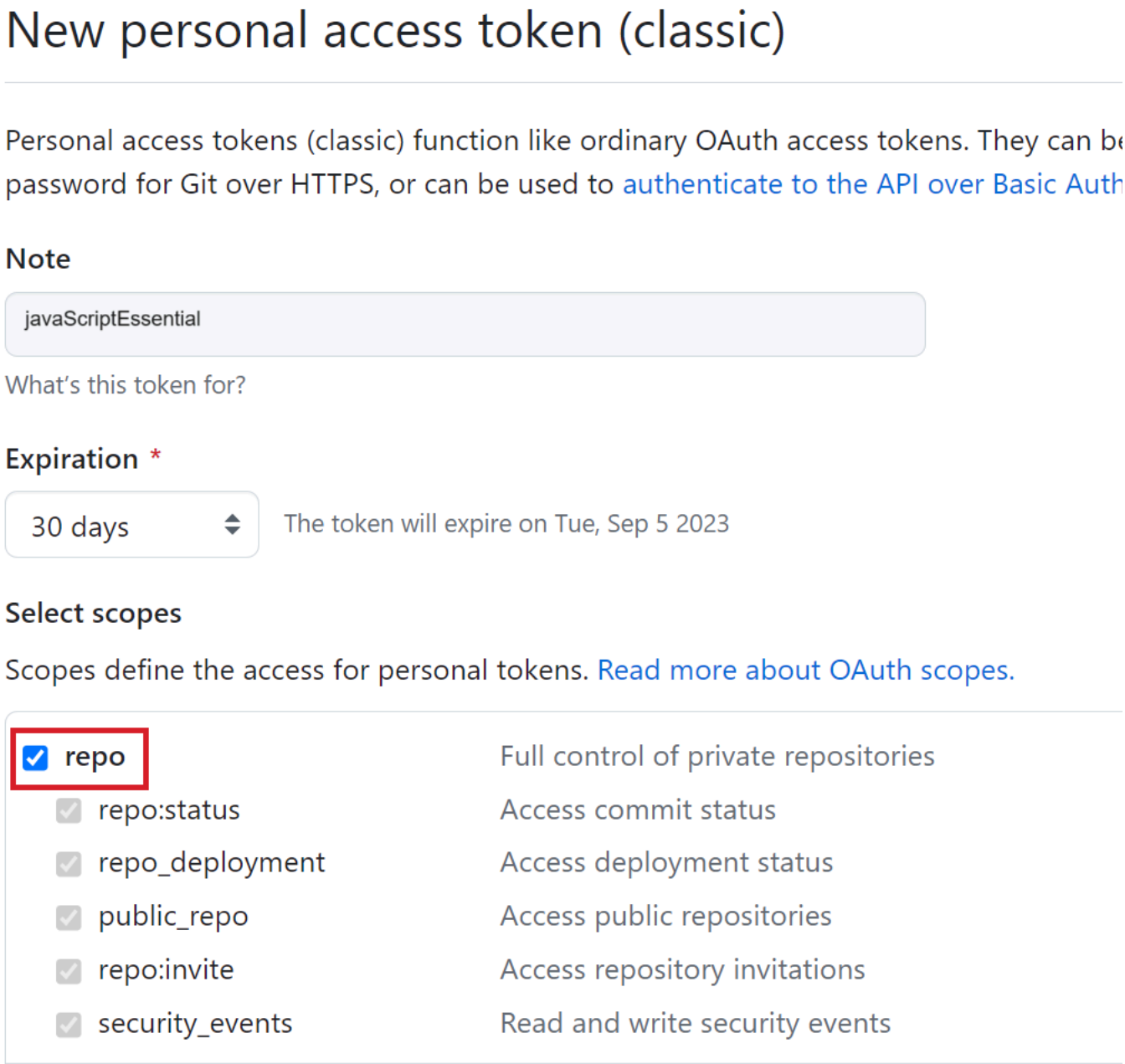
6. 在 **个人访问令牌** 下导航到 **令牌（经典）**。



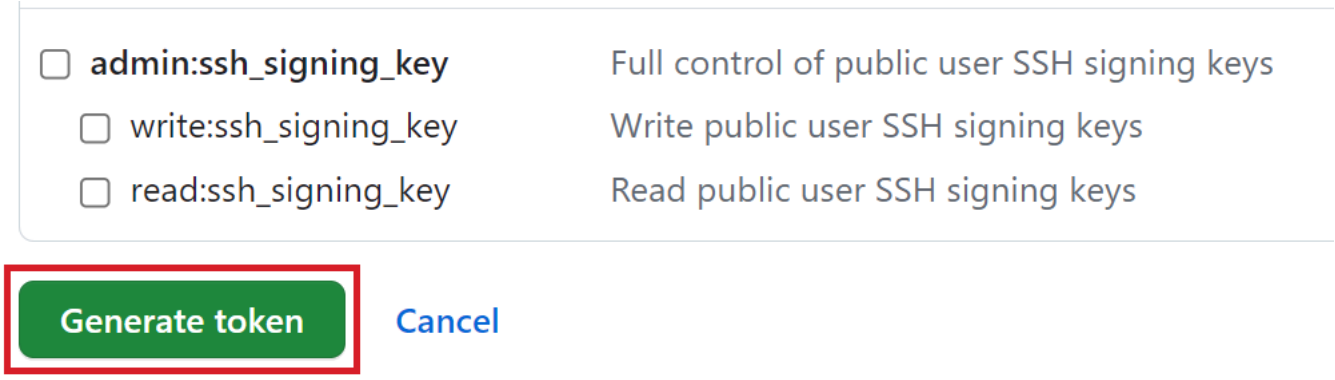
7. 要生成访问令牌，点击 **生成个人访问令牌**。



8. 在 **生成令牌** 页面，填写所需的详细信息，并勾选 **repo** 复选框以启用 `git` 命令的访问。



9. 然后，点击 **生成令牌**。

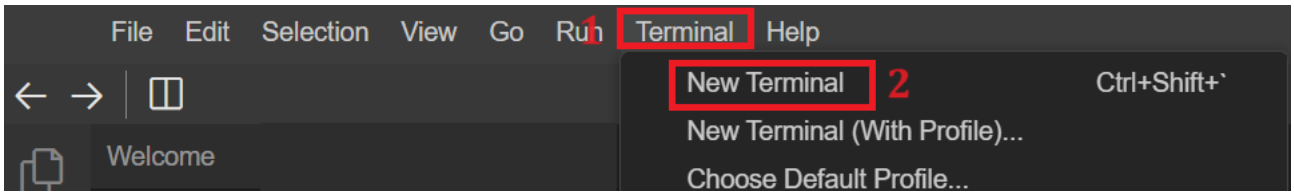


10. 你的个人访问令牌将被生成。该令牌仅在 **30 天** 内有效。当前令牌过期后，你需要生成一个新令牌。

记住： 现在确保复制你的个人访问令牌。你将无法再次看到它！如果你忘记保存或丢失了它，请删除已经创建的令牌并生成新的。

任务 2：在技能网络环境中创建文件

1. 如果技能网络环境中没有打开终端，请在窗口右上角选择“终端”选项卡，然后从下拉菜单中选择“新终端”。这些在给定的截图中显示。



2. 按照给定步骤创建 React 应用程序：

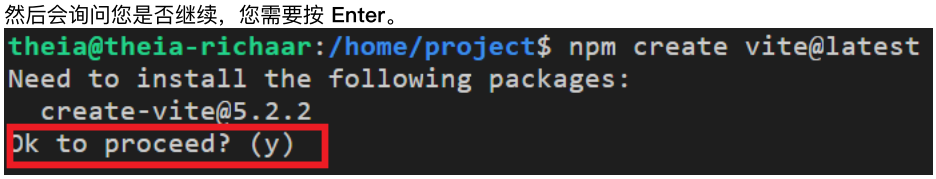
- 在终端中输入以下命令：

```
npm create vite@latest
```

- 终端应类似于给定的截图。



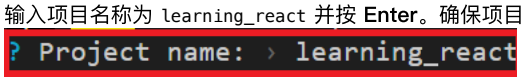
- 然后会询问您是否继续，您需要按 Enter。



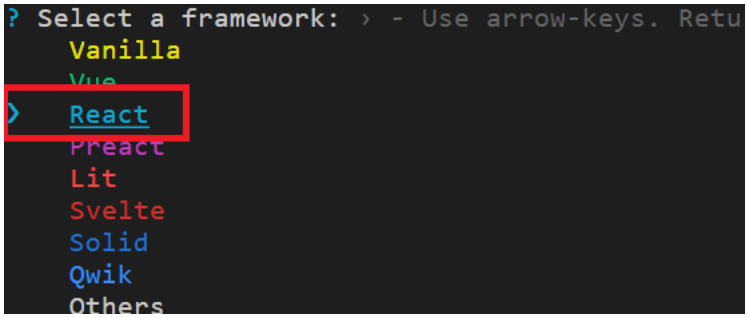
3. 在下一步中，它会要求输入项目名称，如给定的截图所示。



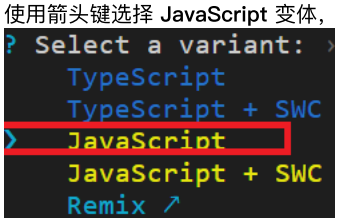
- 输入项目名称为 `learning_react` 并按 **Enter**。确保项目名称为小写。您可以参考给定的截图。



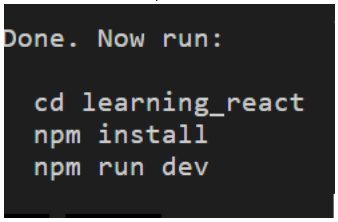
4. 从将显示的列表中选择一个框架。使用箭头键选择 **React** 并按 **Enter**。



5. 使用箭头键选择 **JavaScript** 变体，然后按 **Enter**。



- 完成上述步骤后，它将创建一个名为 `learning_react` 的文件夹。它会要求运行某些命令，这些命令将在终端中显示，如给定的截图所示。



6. 现在您需要进入使用终端创建的应用程序文件夹。为此，请执行以下命令。

```
cd learning_react
```

7. 在终端中输入 *npm install* 命令。它将安装运行 React 应用程序所需的所有文件。

```
npm install
```

8. 现在转到 `learning_react` 文件夹中的 `package.json`。在 **script** 对象下的 **preview** 键中包含给定代码。

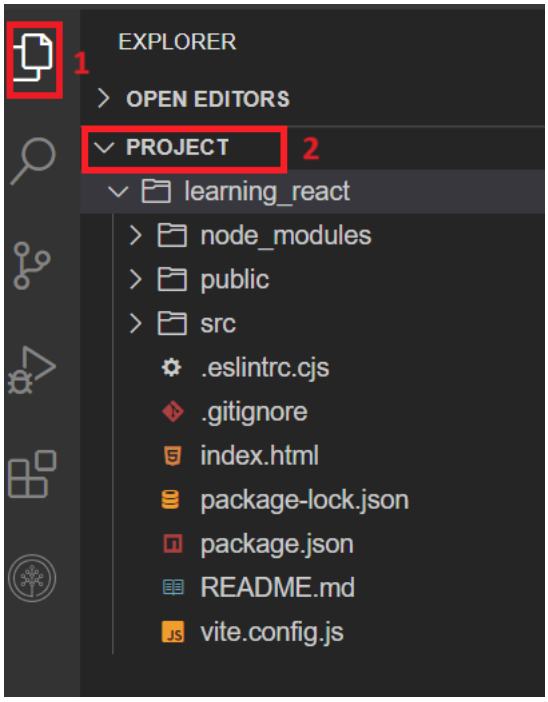
```
"preview": "vite build; vite preview --host"
```

- 确保脚本对象应如给定截图所示。

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnin
  "preview": "vite build; vite preview --host"
},
```

9. 接下来的步骤将允许您运行您的 React 应用程序。

   现在点击给定截图中编号为 1 的 **Explorer** 图标，然后通过点击扭动箭头展开 **project** 文件夹。您将看到 React 应用程序的整个文件夹结构。

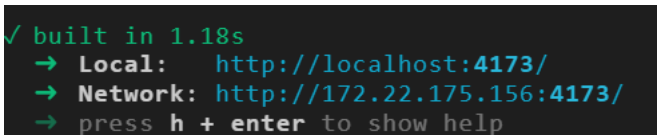


10. 之后，在终端中输入 `npm run preview` 命令以运行应用程序。

    ```
    npm run preview
    ```

    您的终端将显示以 `Local` 开头的一行，后面跟着本地主机路径和端口号。

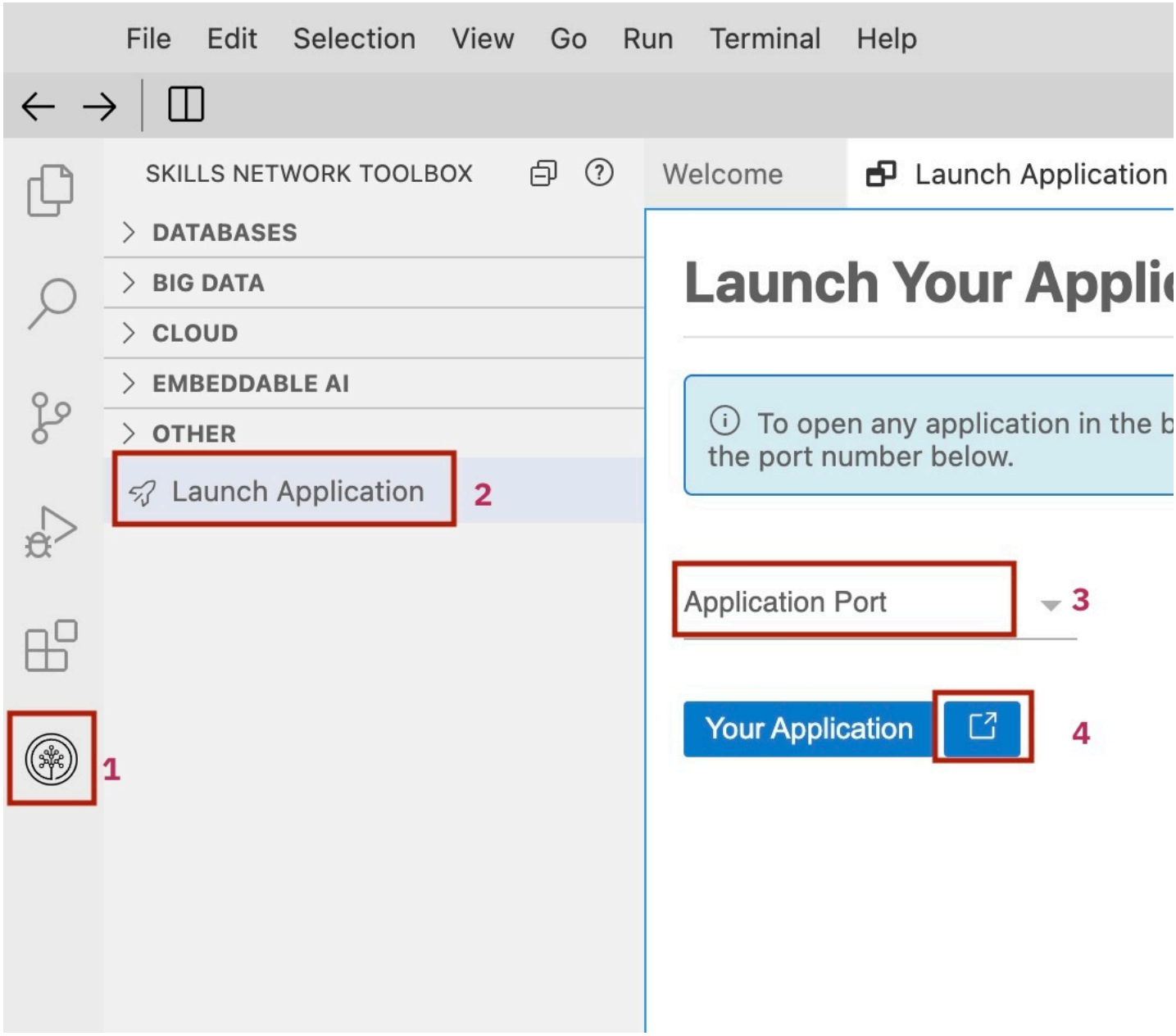
    您可以看到下面的示例屏幕，其中端口号为 4173。此端口将运行您的 React 应用程序。



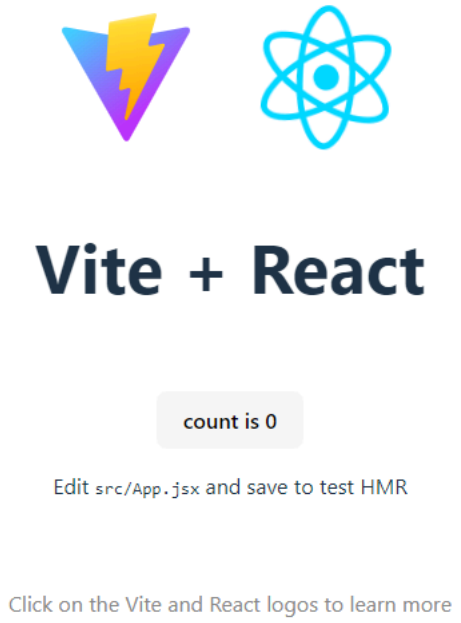
任务 3：检查您的输出

接下来，您将采取必要步骤在浏览器中查看您的输出。

1. 在 IDE 环境的左下角选择 **Skills Network** 图标，如截图中的编号一所示。这将打开“Skills Network Toolbox”。
2. 此操作将打开 **Skills Network Toolbox**。接下来，点击 **Launch Application**（参见编号二）。
3. 在 **Application Port** 中输入端口号 **4173**（参见编号三），然后点击 。

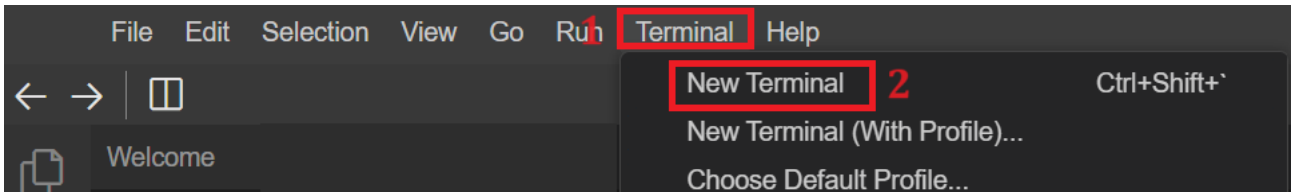

File Edit Selection View Go Run Terminal Help
SKILLS NETWORK TOOLBOX
DATABASES
BIG DATA
CLOUD
EMBEDDABLE AI
OTHER
Launch Application
2
1
Welcome
Launch Application
Launch Your Appli
To open any application in the b
the port number below.
Application Port
3
Your Application
4

4. React 应用程序将打开您的默认浏览器，您将在其中看到如下输出。



任务 4：在终端中执行 Git 命令

1. 现在，选择窗口右上角的 "Terminal" 标签，然后从下拉菜单中选择 "New Terminal"，如截图所示。



确保你的终端路径显示为 `/home/project/ cd <learning_react>`，其中 `<learning_react>` 应替换为你自己的项目名称。

2. 接下来，初始化终端以创建一个 Git 仓库，这样你就可以在这个终端中执行所有 Git 命令。使用提供的命令并按 **Enter**。

```
git init
```

3. 然后您需要配置终端，以便在 **技能网络环境** 中执行必要的命令。

```
git config --global --add safe.directory /home/project
```

按 **Enter**

重要

`git config` 命令将帮助您在项目文件夹环境中使用 git 命令。

4. 接下来，您需要用您的电子邮件地址配置 git。将 *you@example.com* 替换为您的电子邮件地址。确保您的电子邮件地址用双引号括起来。

```
git config --global user.email "you@example.com"
```

按 **Enter**。

现在你应该使用 `config` 命令来设置你的用户名。将 *Your Name* 替换为你的名字，并确保使用双引号，如下所示。

```
git config --global user.name "Your Name"
```

5. 接下来，执行 `git add` 和 `git commit` 以保存对 GitHub 仓库的更改。依次运行以下命令：

```
git add --a
git commit -m "initial commit"
```

6. 执行 `git push` 命令将文件推送到您的 GitHub 存储库。您需要在您的 GitHub 存储库中设置主分支。

```
git branch -M main
```

7. 在 origin 变量中添加你的 GitHub 仓库 URL。还要将 `<git-repo-url>` 替换为你的 GitHub 仓库 URL。
   例如：
   ```
   git remote add origin https://github.com/<youraccountname>/<yourrepositoryname>
   ```
   ```
   git remote add origin <git-repo-url>
   ```

按 **Enter**

8. 然后，将文件的内容推送到你的 GitHub 仓库：
   ```
   git push -u origin main
   ```

按 **Enter**

重要

*上述命令将您的第一次推送命令设置为直接将 **所有** 更改上传到主分支。您只需执行一次。之后 `git push` 也会执行相同的操作。*

9. 在使用 `git push` 命令将文件推送到 GitHub 时，它会要求您在终端中输入您的 GitHub 账户的用户名和密码。

- 输入您的用户名，然后按回车。
- 对于您的密码，您需要粘贴在步骤 1 中生成的 **个人访问令牌**。

重要

将您的 **个人访问令牌** 粘贴到终端后，由于安全原因不会显示，但它确实存在。只需按回车，它将把您的文件和文件夹推送到 GitHub 仓库。

10. 您的所有文件已被推送到您的 GitHub 目录。

注意： 您还可以参考 Hands-on Lab: Getting Started with Branches using Git Commands 以获取有关 git 命令的更多详细信息。

任务 5：使用 GitHub Pages 部署

1. 要在 GitHub 上部署你的 React 应用程序，你需要安装 `gh-pages`。这使你可以将其作为将项目部署到 GitHub Pages 的工具。在终端中执行以下命令
   ```
   npm install gh-pages --save-dev
   ```
2. 在 `package.json` 文件中 `"build": "vite build"` 之前添加以下行。
   ```
   "predeploy": "npm run build",
   "deploy": "gh-pages -d dist",
   ```

```
"scripts": {
    "dev": "vite",
    "predeploy": "npm run build",
    "deploy": "gh-pages -d dist",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warn
```

3. 然后在 vite.config.js 文件中，在 plugins 之前添加这一行：[react()]。
   ```
   base: "/YOUR_REPOSITORY_NAME",
   ```

```
export default defineConfig({
    base: "/YOUR_REPOSITORY_NAME",
  plugins: [react()],
})
```

注意：将 <YOUR_REPOSITORY_NAME> 替换为你自己的仓库名称，例如如果你的 GitHub 仓库名称是 learning_react，则应写为 `base: "/learning_react"`

4. 现在在终端中执行部署命令，以执行在 package.json 文件中定义的 "deploy" 脚本，将项目部署到 GitHub Pages，使用 gh-pages 工具。
   ```
   npm run deploy
   ```

```
theia@theia-ritikaj:/home/project/react$ npm run deploy

> react@0.0.0 predeploy
> npm run build


> react@0.0.0 build
> vite build

vite v5.3.1 building for production...
✓ 34 modules transformed.
dist/index.html                   0.49 kB │ gzip:  0.30 kB
dist/assets/react-CHdo91hT.svg    4.13 kB │ gzip:  2.14 kB
dist/assets/index-DiwrgTda.css    1.39 kB │ gzip:  0.72 kB
dist/assets/index-DQ9y_QkT.js   143.38 kB │ gzip: 46.06 kB
✓ built in 1.20s

> react@0.0.0 deploy
> gh-pages -d dist

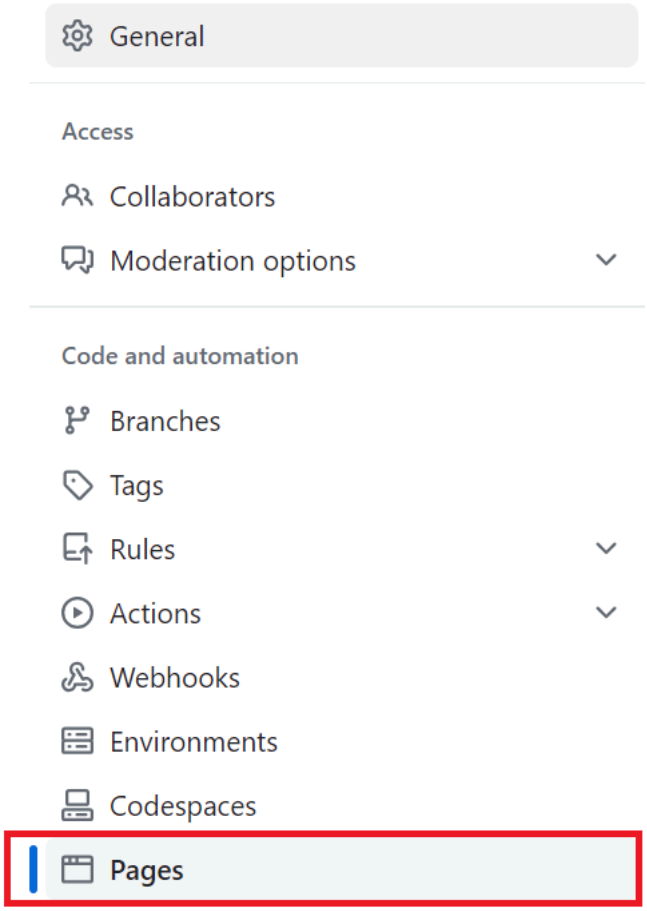
Username for 'https://github.com':
Password for 'https://[redacted]@github.com':
Published
theia@theia-ritikaj:/home/project/react$
```

注意：每当你对代码进行任何更改时，都需要保存所有文件并执行 git 命令。

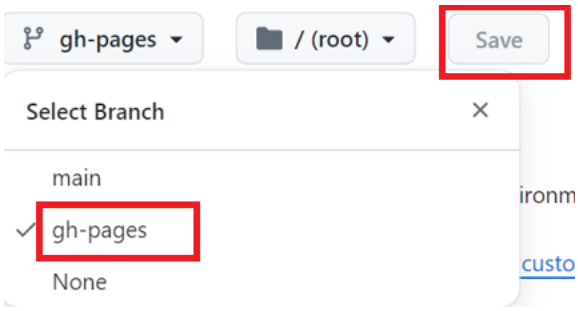
5. 执行 `git add` 和 `git commit` 命令以更新代码中的更改。然后执行 `git push` 命令以更新你的 GitHub 仓库，以便进行适当的代码管理。
6. 转到你的 GitHub 仓库。然后，导航到你创建的网站的仓库。
7. 在你的仓库名称下，点击 **Settings**。



8. 导航到左侧导航栏。在侧边栏的 **Code and Automation** 部分，点击 **Pages**。



9. 你将看到如下页面。点击下拉菜单中看到的 **None**，然后点击 **gh-pages**，再点击 **Save** 按钮。

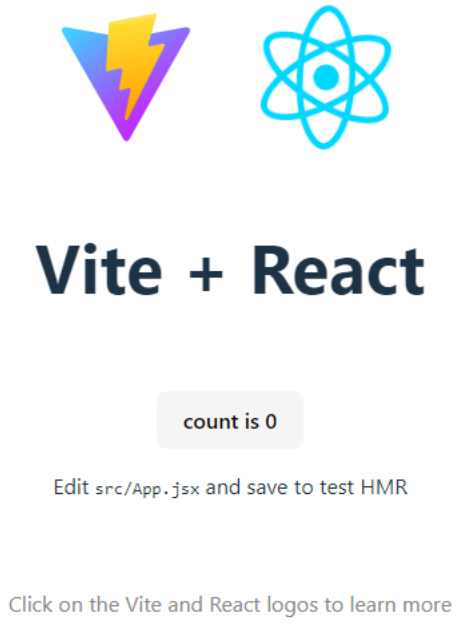


10. 刷新你的页面，你将看到如下链接。替换 **shoppingreact**，你将看到你的 GitHub 仓库名称。



注意： 如果你无法看到链接，请等待 (1–2) 分钟，然后再次刷新页面。

11. 点击上述生成的链接以查看你的在线网站。



注意：
1. 如果你没有对默认的 React 应用程序进行任何更改，你应该会看到与任务 3，第 6 步类似的输出，如下所示。
2. 在 GitHub Pages 上部署后，可能需要一些时间才能正确显示所有内容和图像。请再等几分钟以完全加载应用程序。

通过遵循这些指南，你可以维护一个组织良好且高效的 GitHub 仓库，确保你的工作安全存储并易于你和你的合作者访问。

Author(s)

Richa Arora

© IBM Corporation. 保留所有权利。