Préparation de l'environnement de développement.

Jusqu'à présent, nous écrivions du JavaScript directement intégré dans les pages HTML. Avec React, nous allons changer de dimension : nous n'allons plus ajouter des simples scripts, mais construire de véritables applications.

Pour cela, il nous faut un environnement de développement complet, composé de trois outils essentiels :

- Un moteur pour exécuter nos outils
- Un gestionnaire de paquets pour installer des modules
- Un assembleur (bundler) pour optimiser et construire l'application

Dans ce TP, nous allons donc apprendre à transformer votre poste en un environnement moderne et complet pour développer avec React.

1. Prise en main du nuage

Pour travailler dans de bonnes conditions et éviter les problèmes liés aux différences entre les machines personnelles, nous allons utiliser le nuage pédagogique tout au long du module.

La première étape consiste à nous connecter au nuage via le lien suivant :
https://nuage-pedagogique.fr/connexion

Une fois connectés, nous arrivons sur la page d'accueil.
Il suffit d'ouvrir le menu « Nuages » pour afficher la liste des environnements associés à notre compte.

Un nuage nommé « **REACT-XXXX »** devrait apparaître dans votre liste d'environnements.

Précisions : Les « XXXX » correspondent à un identifiant généré automatiquement lors de la création du nuage.

Figure 1 : Exemple d'affichage du nuage React dans la liste sur le Nuage

Si le nuage est éteint (icône rouge), il suffit de le démarrer en cliquant sur le bouton prévu à cet effet.
Le nuage est prêt lorsqu'il passe au vert et que les boutons permettant d'accéder aux différents services deviennent visibles.

Cela devrait ressembler à l'image ci-après :

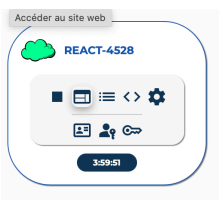


Figure 2 : Exemple d'affichage pour un nuage démarré

Guide des boutons du Nuage Pédagogique	
	Arrêter le nuage. Le nuage se coupe automatiquement toutes les 4h.
	Accéder à la page web. Le nuage React ne diffuse pas de page web par défaut.
	Accéder à PhpMyAdmin. Le panel de gestion des bases de données du nuage.
	Accéder à VSCode. L'éditeur de code par défaut fourni dans le nuage.
	Paramètres du nuage.
	Raccourci du nom d'utilisateur.
	Raccourci du mot de passe utilisateur.
	Raccourci du mot de passe root.

Une fois cette étape validée, nous pouvons ouvrir VS Code depuis le nuage et commencer la configuration de notre environnement de développement.

Précisions :
Il est tout à fait possible d'utiliser votre éditeur de code local en le connectant en SSH au nuage.

2. Mise en place du nuage

Pour pouvoir développer avec React, nous avons besoin d'un environnement JavaScript moderne. La première brique essentielle est Node.js, un moteur capable d'exécuter du JavaScript en dehors du navigateur.

Il s'accompagne de NPM, un gestionnaire de paquets incontournable pour installer les modules nécessaires à un projet moderne.

Nous allons ici installer Node.js à l'aide de NVM, un outil qui permet de gérer facilement plusieurs versions de Node sur la même machine.
Cette approche garantit un environnement propre, stable et adapté aux besoins du module.

2.1. Vérifications préalables.

Certaines machines du nuage sont déjà partiellement configurées. Avant d'installer quoi que ce soit, nous allons effectuer les vérifications nécessaires et apporter quelques modifications.

Par défaut, le terminal n'est pas correctement configuré. Nous allons ici apporter une modification pour qu'il utilise « BASH » :

File → Preferences → Settings

Dans la page de configuration qui s'ouvre nous devons saisir dans la barre de recherche « terminal default profile » pour trouver l'option suivante :

Terminal › Integrated › Default Profile: Linux

Cette option propose un menu déroulant avec plusieurs choix. Dans notre cas, nous devons sélectionner l'entrée « Bash ».
Comme ceci :

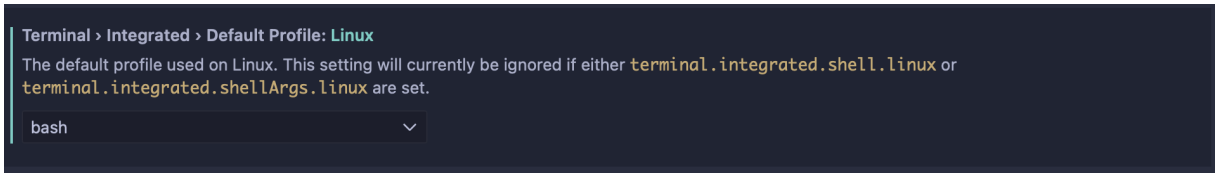


Figure 3 : Capture du fichier de configuration de VS Code avec le paramètre à modifier

Nous pouvons maintenant ouvrir un nouveau terminal depuis le VS Code du Nuage Pédagogique :

Menu → Terminal → New Terminal

Il est aussi possible d'ouvrir un terminal via un raccourci clavier :

```
// Windows

// MacOS
Ctrl + Shift + `
```

Le terminal du nuage utilise par défaut l'utilisateur « root », ce qui est loin d'être sécurisé. Il est donc nécessaire de toujours passer sur l'utilisateur « loginXXXX » attribué au nuage.

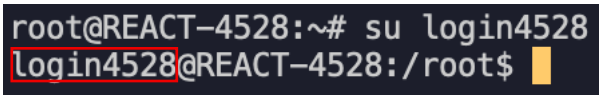
Attention :
L'opération est à reproduire à chaque fois que le nuage est redémarré. Il est très important de bien vérifier que les actions sont effectuées avec l'utilisateur prévu à cet effet.

Pour changer d'utilisateur nous utiliserons la commande suivante :

```
su loginXXXX
```

Remarque : Le « loginXXXX » est à remplacer par celui qui a été créé pour le nuage. Il se trouve dans les options de l'instance.

Le résultat à obtenir :



2.2. Installation de NodeJS.

Par défaut le nuage possède déjà NodeJS mais celle-ci est trop ancienne et ne peut pas être utilisée pour notre cours. Nous pouvons vérifier sa version en utilisant la commande suivante :

```
> node -v
```

La version actuelle de NodeJS (LTS) : v24.11.1
Toutes les dernières versions de NodeJS se trouvent sur ce lien :
https://nodejs.org/en/about/previous-releases

Nous allons donc mettre à jour la version de NodeJS et pour cela nous utiliserons l'utilitaire NVM (Node Version Manager).

Pour installer NVM nous devons suivre la procédure suivante :

```
> curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash

> nano ~/.bashrc
```

Explications :

- La première ligne permet de télécharger l'exécutable NVM et exécute l'installation automatiquement.
- La deuxième ligne permet d'éditer le fichier appelé lorsqu'un terminal est ouvert.

Dans le fichier « ~/.bashrc », nous devons ajouter les lignes suivantes :

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
```

Explications :

- Globalement, ces quelques lignes permettent d'indiquer à bash qu'il faut charger l'exécutable nvm lorsque nous ouvrons un nouveau terminal. Sans ces lignes nous serions obligés d'exécuter cette commande à chaque ouverture de terminal.

Afin de prendre en compte les modifications que nous venons de faire, il faut fermer et rouvrir un nouveau terminal.

Nous devrions maintenant être capable de lancer la commande suivante :

```
> nvm
```

Si ce n'est pas le cas il est parfois utile de fermer le terminal et d'en rouvrir un nouveau. Dans le cas où le problème persiste, il se peut que l'installation ne se soit pas déroulée correctement. De ce fait il suffit de recommencer la procédure.

Puisqu'une version de NodeJS était déjà installée sur notre système nous devons nettoyer ces fichiers pour laisser notre machine le plus propre possible.
Exécutons les commandes suivantes :

```
> nvm use system
> npm uninstall -g a_module
```

Explications :

- La première ligne indique que nous souhaitons utiliser la version de NodeJS actuellement installée sur le nuage.
- La deuxième ligne permet de désinstaller tous les modules NodeJS actuellement installés sur le nuage.

Nous pouvons maintenant supprimer l'ancienne version de NodeJS. Pour cela nous devons d'abord passer sous root :

```
exit
```

Puis lançons la commande suivante :

```
> apt remove nodejs -y
```

Nous pouvons revenir à l'utilisateur :

```
> su loginXXXX
```

Nous pouvons enfin installer la dernière version stable de NodeJS à l'aide de NVM grâce à la commande suivante :

```
> nvm install --lts
```

Explication :
Le paramètre « --lts » indique à nvm qu'il doit aller chercher la dernière version stable de NodeJS.

Quelques commandes utiles : NVM	
nvm install <version>	Installer une version spécifique de NodeJS
nvm use <version>	Utiliser une version spécifique de NodeJS
nvm list	Liste les versions de NodeJS actuellement installées
nvm uninstall <version>	Désinstaller une version spécifique de NodeJS
nvm --version	Afficher la version de nvm

Pour vérifier que NodeJS et NPM soient correctement installés, exécutons les commandes suivantes :

```
> node -v
> npm -v
```

Explications :
Ces commandes permettent d'afficher la version de NodeJS et de NPM installé sur le nuage.

Nous disposons maintenant d'une installation propre et contrôlée de Node.js et NPM. Cette base est indispensable pour la suite du module, notamment pour gérer les dépendances, utiliser des outils modernes et préparer la création de notre premier projet React.

3. Paramétrages de VSCode

Maintenant que Node.js est installé, nous allons nous consacrer à Visual Studio Code afin d'obtenir un environnement de développement confortable pour la suite du module. VS Code n'est pas seulement un éditeur de code, il fournit aussi de nombreux outils qui facilitent le développement d'application.

Nous allons, de ce fait, l'enrichir de plusieurs extensions essentielles et ajuster quelques réglages pour rendre l'écriture du code plus fluide, plus propre et plus lisible.

3.1. Installation des extensions

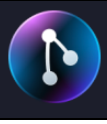
Pour travailler efficacement, nous allons installer plusieurs extensions qui améliorent l'expérience de développement et plus particulièrement le développement autour de React.

Commençons par ouvrir le menu « Extension » dans la barre latérale à gauche. Il est également possible d'utiliser un raccourci :

```
// MacOS
Cmd + Shift + X

// Windows
```

Nous allons pouvoir rechercher les extensions utiles grâce à la barre de recherche. Voici la liste des extensions à installer :

Liste des extensions à installer	
	**Auto Rename Tag** Modifie automatiquement la balise fermante lorsque la balise ouvrante est modifiée.
	**Error Lens** Met les erreurs directement dans le code pour une meilleure visibilité.
	**ES7+ React/Redux/React-Native snippets** Propose des raccourcis utiles pour les syntaxes modernes.
	**ESLint** Facilite l'écriture de code propre en signalant les erreurs et incohérences.
	**Path Intellisense** Autocomplète les chemins de fichiers.
	**Prettier – Code formatter** Formate automatiquement le code selon des règles définies.
	**GitLens — Git supercharged** Permet de mieux visualiser l'historique Git et les modifications.

Information :
Cette liste n'a aucune prétention à être la meilleure. Il est tout à fait possible de la personnaliser avec d'autres extensions. Cependant, le cours s'appuiera bien évidemment sur la liste proposée.

3.2. Formater automatiquement le code

Nous allons maintenant ajuster quelques paramètres de VS Code.
L'objectif est que le code soit automatiquement formaté et corrigé lorsqu'un fichier est enregistré.

Ouvrons la palette de commandes :

```
// MacOS
Cmd + Shift + P

// Windows
```

Ctrl + Shift + P

Un champ texte apparait en haut de l'écran et demande de saisir nos mots clés. Nous allons ici rechercher l'option suivante :

Preferences: Open User Settings (JSON)

Un nouveau fichier s'ouvre, il doit normalement contenir une ligne avec notre paramètre « bash » que nous avons fait en début de ce TP. Ce fichier est en réalité la version « JSON » de la configuration de VSCode. Puisque nous allons ajouter plusieurs paramètres, il est beaucoup plus simple d'éditer la configuration de cette manière.

Dans ce fichier, ajoutons les instructions nécessaire pour obtenir ce résultat :

```
{
   "terminal.integrated.defaultProfile.linux": "bash",
   "files.autoSave": "onFocusChange",
   "editor.codeActionsOnSave": {
     "source.fixAll.eslint": true
   },
   "prettier.trailingComma": "es5",
   "eslint.validate": ["javascript", "javascriptreact"],
   "editor.formatOnSave": true,
   "editor.defaultFormatter": "esbenp.prettier-vscode",
   "prettier.singleQuote": true
}
```

Explications :

- « files.autoSave » ce paramètre permet d'indiquer à quel moment il faut sauvegarder le fichier. Dans notre cas nous choisissons la valeur « onFocusChange » pour sauvegarder dès que nous n'utilisons plus le fichier.
- « editor.codeActionOnSave » permet d'ajouter différentes actions provenant des autres extensions lorsqu'un fichier est sauvegardé. Dans notre cas nous souhaitons que l'extension eslint puisse être actionnée pour réorganiser notre code.
- « prettier.trailingComma » cette option permet d'ajouter automatiquement des virgules et des points virgules dans le code, là où il en faut. La valeur « es5 » permet de limiter l'ajout des virgule et des points virgules en fin de ligne en suivant les recommandations en vigueur du javascript.
- « eslint.validate » permet d'activer l'action d'eslint sur les fichiers JavaScript et les fichier JavaScript écrits avec la syntaxe de React.
- « editor.formatOnSave » permet d'indiquer à l'éditeur de code que dès qu'un fichier est sauvegardé, le code doit être formaté. Par exemple le code va être automatiquement indenté en fonction des normes et des bonnes pratiques. Très utile pour garder des fichiers propres.

- « editor.defaultFormatter » permet choisir l'extension qui s'assure du bon formatage du code.
- « prettier.singleQuote » se charge de transformer les guillemets en apostrophes. C'est aujourd'hui une bonne pratique en termes de programmation pour certains langages comme JavaScript.

Ces paramètres garantissent un formatage homogène et automatique à chaque enregistrement.
Cela contribue à un code plus lisible et plus simple à maintenir.

3.3. Test de la configuration

Afin de valider la configuration, nous allons créer un petit fichier de test.

Travail à réaliser :

- Créer un fichier test.js dans un dossier temporaire.
- Placer volontairement le code ci-après, mal formaté.
- Enregistrer le fichier

```
const hello = (name) =>{console.log("Hello " + name)}
```

Si tout est bien configuré, VS Code devrait automatiquement reformater le code :

```
const hello = (name) => {
  console.log("Hello " + name);
};
```

VS Code est maintenant configuré !

Nous disposons d'un éditeur propre, efficace, qui applique automatiquement les bonnes pratiques et facilite l'écriture du code moderne.

Dans la prochaine partie, nous allons configurer Git afin de mettre en place un workflow propre et adapté à notre module.

4. Installation et configuration de Git

Dans cette partie, l'objectif est de préparer un environnement Git fonctionnel, associé à notre compte GitHub, afin de pouvoir récupérer et déposer les projets.

Information :
Git n'est plus à présenter. A ce stade nous considérerons que les compétences en Git sont acquises.

4.1. Vérifier l'installation de Git

La plupart des machines du nuage pédagogique incluent déjà Git.
Nous commençons donc par vérifier sa présence :

```
> git --version
```

Si une version apparaît, nous pouvons passer à la configuration.
Dans le cas contraire, nous devons l'installer via le gestionnaire de paquets du système :

```
> exit
> apt install git -y
> su loginXXXX
```

4.2. Configurer Git pour la première fois

Nous allons maintenant déclarer notre identité.
Ces informations seront associées à chaque commit que nous effectuerons :

```
> git config --global user.name "Prénom Nom"
> git config --global user.email "mon-email@example.com"
```

Pour ne pas à avoir à définir la branche par défaut de chaque projet, nous pouvons l'insérer dans la configuration :

```
> git config --global init.defaultBranch main
```

Cette configuration est à réaliser qu'une seule fois pour ce nuage.

4.3. Connecter le nuage avec GitHub

Nous allons maintenant connecter notre nuage avec notre compte GitHub.

Pour éviter de saisir un mot de passe à chaque opération Git, nous allons utiliser une clé SSH.
Cela nous permettra de communiquer avec GitHub de manière sécurisée.

Générons la clé grâce à la commande suivante :

```
> ssh-keygen -t rsa -b 4096
```

Nous devons valider chaque étape en appuyant sur Entrée, sans définir de mot de passe pour la clé.

Une fois la clé générée, il nous faut récupérer la clé publique :

```
> cat /var/www/html/.ssh/id_rsa.pub
```

La clé va s'afficher dans la console, il suffit de la copier pour pouvoir l'enregistrer sur GitHub.

Après s'être connecté sur notre compte GitHub, nous devons accéder aux paramètres suivants :

```
Settings → SSH and GPG Keys → New SSH Key
```

La page doit ressembler à cela :

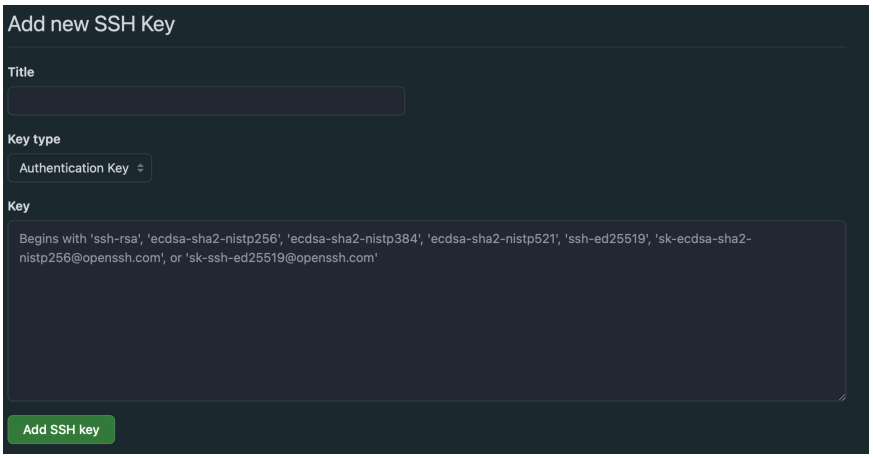


Figure 4 : Capture du formulaire de génération d'une clé SSH sur GitHub

Pour le titre, il est fortement recommandé d'en choisir un facilement identifiable. Nous pourrions par exemple utiliser le nom du nuage, comme ceci :

```
Nuage Pédagogique – REACT-XXXX
```

Dans la zone « key » il suffit de coller la clé publique précédemment copiée.

Pour vérifier le fonctionnement de la clé, nous utilisons la commande suivante :

```
> ssh -T git@github.com
```

Si tout est correctement configuré, GitHub nous accueille par un message de confirmation.

Nous disposons maintenant d'une installation Git complète, associée à notre compte GitHub et prête à être utilisée.

Dans la prochaine partie, nous allons passer à l'étape suivante :
« Initialiser notre premier projet React. »

Ce sera l'occasion de découvrir les outils qui accompagnent React, de comprendre comment un projet moderne est structuré et de commencer à travailler sur les premières fonctionnalités de React.