Commandes Git les plus utilisées

Tableau récapitulatif des commandes git les plus utilisées

Commande Git	Rôle / Explication simple	Métaphore facile à retenir
git init	Démarre Git dans ton projet local.	Tu poses une *boîte vide* dans ta chambre pour y ranger ton projet.
git status	Montre les fichiers nouveaux, modifiés ou supprimés.	Tu regardes ce qu'il y a dans la boîte et ce qui n'est pas encore rangé.
git add .	Prépare tous les fichiers pour le prochain enregistrement (commit).	Tu ranges tous les fichiers dans la boîte avant la photo.
git add fichier.js	Prépare un fichier spécifique.	Tu ranges un seul document dans la boîte.
git commit -m "message"	Enregistre une version complète du projet.	Tu prends une *photo* de ta boîte (un instantané de ton travail).
git log	Liste l'historique de toutes les photos (commits).	Tu regardes ton *album photo* du projet.
git branch	Montre toutes les branches existantes.	Tu regardes toutes les routes de développement ouvertes.
git branch nom-branche	Crée une nouvelle branche (une copie du projet).	Tu ouvres un *nouveau chantier test* à côté.
git checkout nom-branche	Change de branche pour travailler ailleurs.	Tu passes d'un chantier à un autre.
git checkout -b nom-branche	Crée et bascule directement sur la nouvelle branche.	Tu ouvres un nouveau chantier et tu t'y rends tout de suite.
git merge nom-branche	Fusionne une branche dans la branche actuelle.	Tu réunis ton chantier test avec le projet principal.
git branch -d nom-branche	Supprime une branche devenue inutile.	Tu fermes un chantier terminé.
git remote add origin https://...	Lie ton projet local à un dépôt GitHub.	Tu dis à Git où se trouve ton "bureau principal" sur le cloud.
git remote -v	Vérifie la connexion avec GitHub.	Tu vérifies l'adresse où tu envoies ton code.
git push -u origin main	Envoie la branche principale sur GitHub et crée un lien permanent.	Tu envoies ton premier colis et tu colles une *étiquette d'adresse permanente*.
git push	Envoie les modifications suivantes sur GitHub.	Tu postes de nouveaux colis à la même adresse.
git pull	Récupère les changements du dépôt distant.	Tu ouvres la boîte pour prendre les ajouts des autres.
git clone https://...	Copie un dépôt GitHub complet sur ton ordinateur.	Tu télécharges une copie complète du projet pour travailler localement.
git fetch	Vérifie s'il y a de nouveaux commits sur GitHub (sans les fusionner).	Tu regardes les nouveautés sans les prendre.
git diff	Compare les différences entre deux versions.	Tu compares deux photos du projet.
git revert id_commit	Annule un commit spécifique (en créant un nouveau commit correctif).	Tu prends une nouvelle photo qui corrige une erreur ancienne.

git reset --hard id_commit	Revient complètement à une version précédente (destructif).	Tu effaces tout pour revenir à une ancienne photo.
git stash	Sauvegarde temporairement ton travail sans commit.	Tu ranges ton travail inachevé sous ton bureau.
git stash pop	Restaure ton travail mis de côté.	Tu ressors ce que tu avais mis de côté pour continuer.

Exercice pratique corrigé — Simulation projet Tangerois React

TP Git + React complet et professionnel
On va construire pas à pas le projet Tangerois React avec : useContext + useReducer**,**
Bootstrap**,**filtre dynamique par catégorie (via Set) et par prix (entre deux valeurs)**,**
workflow Git complet avec branches, merge, et push vers GitHub.

TP COMPLET — Tangerois React avec Git

Objectif

Créer une mini-boutique en ligne inspirée de tangerois.ma :

- Page d'accueil avec liste de produits sous forme de cards.
- Filtres par catégorie et par prix.
- Gestion globale de l'état avec useContext et useReducer.
- Travail collaboratif avec Git (branches, merge, push/pull).

ÉTAPE 1 — Préparer le projet React

1. Créer le projet

```
npx create-react-app tangerois-react
cd tangerois-react
```

2. Installer Bootstrap

```
npm install bootstrap
```

3. Ajoute dans src/index.js :

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

ÉTAPE 2 — Initialiser Git localement

```
git init                    # Crée le dépôt Git local
git add .                   # Ajoute tous les fichiers
git commit -m "Initialisation du projet React Tangerois"
git branch -M main          # Renomme la branche principale en main
```

ÉTAPE 3 — Créer le dépôt distant GitHub

1. Va sur GitHub → **New Repository**
2. Nomme-le : tangerois-react
3. Copie l'URL du dépôt (exemple : https://github.com/tonCompte/tangerois-react.git)

Ensuite :

```
git remote add origin https://github.com/tonCompte/tangerois-react.git

git push -u origin main
```

Maintenant, ta branche main est liée à GitHub.

ÉTAPE 4 — Créer la base du projet React

Structure initiale :

```
src/
 ├── components/
 │   ├── ProductCard.js
 │   ├── Filter.js
 │   └── Home.js
 ├── context/
 │   ├── ProductContext.js
 │   └── productReducer.js
 ├── data/
 │   └── products.js
 └── App.js
```

1. **productReducer.js**

```
export const initialState = {
  ……
};

export const productReducer = (state, action) => {
  switch (action.type) {
    case  "……":
                ……..
                ……..
    case ''……'' :
                ……..
                ……
    case ''……..'':
                ……..
                ……
    default:
      return state;
  }
};
```

2. **ProductContext.js**

```
import React, { createContext, useReducer } from 'react';
import { productReducer, initialState } from './productReducer';
import { productsData } from '../data/products';

export const ProductContext = createContext();

export const ProductProvider = ({ children }) => {
  const [state, dispatch] = useReducer(productReducer, initialState);
```

```
  React.useEffect(() => {
            …….
  }, []);

  return (
   <ProductContext.Provider value={{ state, dispatch }}>
    {children}
   </ProductContext.Provider>
  );
};
```

3. products.js (exemple de données)

```
export const productsData = [
  { id: 1, name: "TV Samsung 50\"", category: "Télévision", price: 6000, image: "tv.jpg" },
  { id: 2, name: "Lave-linge LG", category: "Électroménager", price: 3500, image: "lave-linge.jpg" },
  { id: 3, name: "Four électrique", category: "Cuisine", price: 1200, image: "four.jpg" },
  { id: 4, name: "Casque JBL", category: "Audio", price: 800, image: "casque.jpg" },
  { id: 5, name: "Climatiseur Samsung", category: "Climatisation", price: 9000, image:
"climatiseur.jpg" }
];
```

4. Filter.js

```
import React, { useContext, useState } from 'react';
import { ProductContext } from '../context/ProductContext';

const Filter = () => {
  const { state, dispatch } = useContext(ProductContext);
  const [priceMin, setPriceMin] = useState(state.priceRange[0]);
  const [priceMax, setPriceMax] = useState(state.priceRange[1]);

  const handlePriceChange = () => {
   dispatch({ type: "FILTER_BY_PRICE", payload: { min: priceMin, max: priceMax } });
  };

  return (
   <div className="p-3 border rounded mb-4 bg-light">
    <h5>Catégories</h5>
    {state.categories.map(cat => (
     <div key={cat}>
      <input type="radio" name="category" onChange={() => dispatch({ type:
"FILTER_BY_CATEGORY", payload: cat })} />
      <label className="ms-2">{cat}</label>
     </div>
    ))}

    <h5 className="mt-3">Prix (MAD)</h5>
    <input type="number" value={priceMin} onChange={e => setPriceMin(Number(e.target.value))}
/> -
    <input type="number" value={priceMax} onChange={e =>
setPriceMax(Number(e.target.value))} />
```

```
    <button className="btn btn-sm btn-primary ms-2"
onClick={handlePriceChange}>Filtrer</button>
   </div>
  );
};

export default Filter;
```

5. Home.js

```
import React, { useContext } from 'react';
import { ProductContext } from '../context/ProductContext';
import ProductCard from './ProductCard';
import Filter from './Filter';

const Home = () => {
  const { state } = useContext(ProductContext);

  return (
   <div className="container mt-4">
    <div className="row">
     <div className="col-md-3">
      <Filter />
     </div>
     <div className="col-md-9">
      <div className="row">
       {state.filteredProducts.map(product => (
        <div key={product.id} className="col-md-4 mb-3">
         <ProductCard product={product} />
        </div>
       ))}
      </div>
     </div>
    </div>
   </div>
  );
};

export default Home;
```

6. ProductCard.js

```
 import React from 'react';
import { Card, Button } from 'react-bootstrap';

const ProductCard = ({ product }) => {
  return (
   <Card>
    <Card.Img variant="top" src={product.image} alt={product.name} />
    <Card.Body>
     <Card.Title>{product.name}</Card.Title>
     <Card.Text>{product.price} MAD</Card.Text>
```

```
      <Button variant="primary">Afficher</Button>
    </Card.Body>
  </Card>
 );
};

export default ProductCard;
```

7. **App.js**

```
import React from 'react';
import { ProductProvider } from './context/ProductContext';
import Home from './components/Home';

const App = () => (
 <ProductProvider>
  <Home />
 </ProductProvider>
);

export default App;
```

ÉTAPE 5 — Workflow Git (branches & fonctionnalités)

Étape	Commandes Git	Objectif
Créer une branche pour le filtre prix	git checkout -b filtre-prix	Travailler sans toucher au main
Commit après ajout du filtre	git add . → git commit -m "Ajout du filtre prix"	Sauvegarder la nouvelle fonctionnalité
Revenir sur la branche principale	git checkout main	Préparer la fusion
Fusionner les changements	git merge filtre-prix	Intégrer la nouvelle fonctionnalité
Supprimer la branche	git branch -d filtre-prix	Nettoyer les branches terminées
Pousser sur GitHub	git push	Mettre à jour le dépôt distant