**課題  
Gitの仕組みとGitを⽤いたチーム開発についてドキュメントにまとめること。**

**用語の説明  
リポジトリ：**ファイルを保存するストレージのようなもの  
**リモートリポジトリ：**Webサーバ上に存在するリポジトリ  
**ローカルリポジトリ：**ローカルのPC上に存在するリポジトリ  
**コミット：**作業内容の履歴を保存する作業  
**プル：**他の人が更新した作業の履歴を自分のローカルリポジトリに反映させる作業  
**プッシュ：**自分の更新した履歴をリモートリポジトリに反映させる作業  
**クローン：**すでにGit管理されているところに自分も参加するときに行うダウンロード  
**ブランチ：**ファイルを編集した履歴を分岐させて記録していく機能  
**マスターブランチ：**メインのブランチのこと。現在のバージョンをコピーして  
　　　　　　　　　ほかの作業空間を作り出すことを「ブランチを切る」という。  
**マージ：**切られたブランチで作業を行い、それをほかのブランチに統合する作業  
同じファイルの同じ箇所を編集していた場合は「競合エラー」というのが  
　　　　起こり、どっちの編集を残すかを選んでから、mergeすることができる。

**Gitの基本的な仕組み**Gitとは分散型バージョン管理システム（Distributed Version Control System）であり、ソースコードの変更履歴を管理するツール。​各開発者がローカルリポジトリを持ち、変更を記録（commit）することで、履歴として残す。​その後、リモートリポジトリと同期（push/pull）することで、チーム全体でのコード共有が可能になる

**Gitの基本的な操作**  
git init：新しいGitリポジトリを初期化  
git clone：既存のリポジトリを複製  
git add：変更をステージングエリアに追加  
git commit：ステージングエリアの変更をローカルリポジトリに記録  
git push：ローカルリポジトリの変更をリモートリポジトリに送信  
git pull：リモートリポジトリの変更をローカルリポジトリに取り込み

**チーム開発におけるGitの活用**チーム開発では、複数の開発者が同時に作業を進めるため、Gitのブランチ機能やプルリクエスト（Pull Request）を活用することが重要

**開発の基本的な流れ**リモートリポジトリをクローン（git clone）  
新しいブランチを作成（git branch）し、作業ブランチに切り替え（git checkout）  
コードの変更を行い、ステージング（git add）とコミット（git commit）を実施  
リモートリポジトリにプッシュ（git push）  
プルリクエストを作成し、コードレビューを依頼  
レビュー後、問題がなければメインブランチにマージ（git merge）  
  
このようなフローを採用することで、各開発者が独立して作業を進めつつ、最終的にコードを統合することが可能。  
​また、コードレビューを通じて品質の向上や知識の共有が図れるという利点がある。​

**ブランチ戦略について**チーム開発では適切なブランチ戦略を採用することで作業の分担管理が容易になる。  
以下に​代表的な戦略を示す。  
**Git Flow：​**main（本番用）、develop（開発用）、feature（機能開発用）、release（リリース準備用）、hotfix（緊急修正用）など、複数のブランチを使い分ける​  
**GitHub Flow：​**mainブランチから機能ごとにブランチを作成し、作業後にプルリクエストを通じてレビュー・マージを行う。  
**GitLab Flow：**​Git FlowとGitHub Flowの中間的な戦略で、production（本番用）、staging（テスト用）、main（開発用）などのブランチを使用する。

**プルリクエストとコードレビュー**  
GitHubやGitLabなどのリモートリポジトリを利用することで、プルリクエストを通じてコードの変更を共有し、チームメンバーによるレビューを受けることができる。  
​これにより、コードの品質を保ちながら、効率的な共同作業が可能になる。

**コンフリクトの解消**複数人で同じファイルを編集すると、マージ時にコンフリクトが発生することがありる。​この場合Gitは衝突箇所を明示してくれるため手動で修正して整合性を取ることが可能。​コンフリクトの解消はチームで効果的な開発を行う上で非常に重要である。