

# Git 講座

# はじめに

- Git は奥が深い
- 私も4割しか理解していない
- 今日は基礎的なことしかやりません
- やること: コミット・プッシュ  
チェックアウト (ブランチの概念), 歴史など

# 用語

- ディレクトリ: フォルダのこと.

# Git とは

- 2005年、リーナス・トーバルズというプログラマが作った  
**分散型バージョン管理システム**
  - リーナス・トーバルズ: Linux という OS を作ったおじさん
  - 今は濱野純（はまのじゅん）という Google のエンジニアが管理しています



# Git の核となる概念

# リポジトリ

- プロジェクトの全履歴を保存する「倉庫」
- `git init` → 新規のリポジトリを作成
- `git clone` → 既存のリポジトリを複製

# 4つの領域

- 作業ディレクトリ→ステージング  
→ローカルリポジトリ →リモートリポジトリ
- 作業ディレクトリ: 実際にファイルを編集する場所
- ステージング: 次のコミットに含める準備を行う場所
- ローカルリポジトリ: 自分のPC内の履歴のデータベース
- リモートリポジトリ: GitHub などのサーバ上にある  
共有データベース

# コミット

- ある時点の「スナップショット」を示すもの
- 各コミットには
  - 変更内容
  - 作者と日時データ
  - コミットメッセージ
  - 一意のハッシュ値

The screenshot displays a GitHub commit page for the repository 'rune-of-mer / RuneCore'. The commit ID is 'cc37697', made by user 'm1sk9' 3 days ago. The commit message is 'feat: Add admin command and service'. The diff view shows changes to the file 'src/main/kotlin/org/lyralis/runeCore/RuneCore.kt'. The diff highlights the addition of two new imports: 'RuneGachaAdminCommand' and 'RuneGachaCommand' from the package 'org.lyralis.runeCore.command.impl.gacha'. The left sidebar shows the file tree structure, and the right sidebar shows the search results within the code.

```
@@ -15,13 +15,13 @@ import org.lyralis.runeCore.command.impl.RuneSettingsCommand
15 import org.lyralis.runeCore.command.impl.RuneShopCommand
16 import org.lyralis.runeCore.command.impl.RuneTrashCommand
17 import
   org.lyralis.runeCore.command.impl.experience.RuneExperienceCom
   mand
18 import
   org.lyralis.runeCore.command.impl.level.RuneLevelCommand
19 import
   org.lyralis.runeCore.command.impl.money.RuneMoneyCommand
20 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpaCommand
21 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
22 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
23 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
24 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
25 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
26 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
27 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
28 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
29 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
30 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
31 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
32 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
33 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
34 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
35 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
36 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
37 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
38 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
39 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
40 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
41 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
42 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
43 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
44 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
45 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
46 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
47 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
48 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
49 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
50 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
51 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
52 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
53 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
54 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
55 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
56 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
57 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
58 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
59 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
60 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
61 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
62 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
63 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
64 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
65 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
66 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
67 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
68 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
69 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
70 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
71 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
72 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
73 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
74 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
75 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
76 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
77 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
78 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
79 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
80 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
81 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
82 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
83 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
84 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
85 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
86 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
87 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
88 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
89 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
90 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
91 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
92 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
93 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
94 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
95 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
96 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
97 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
98 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
99 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
100 import
   org.lyralis.runeCore.command.impl.teleport.RuneTpcCommand
```

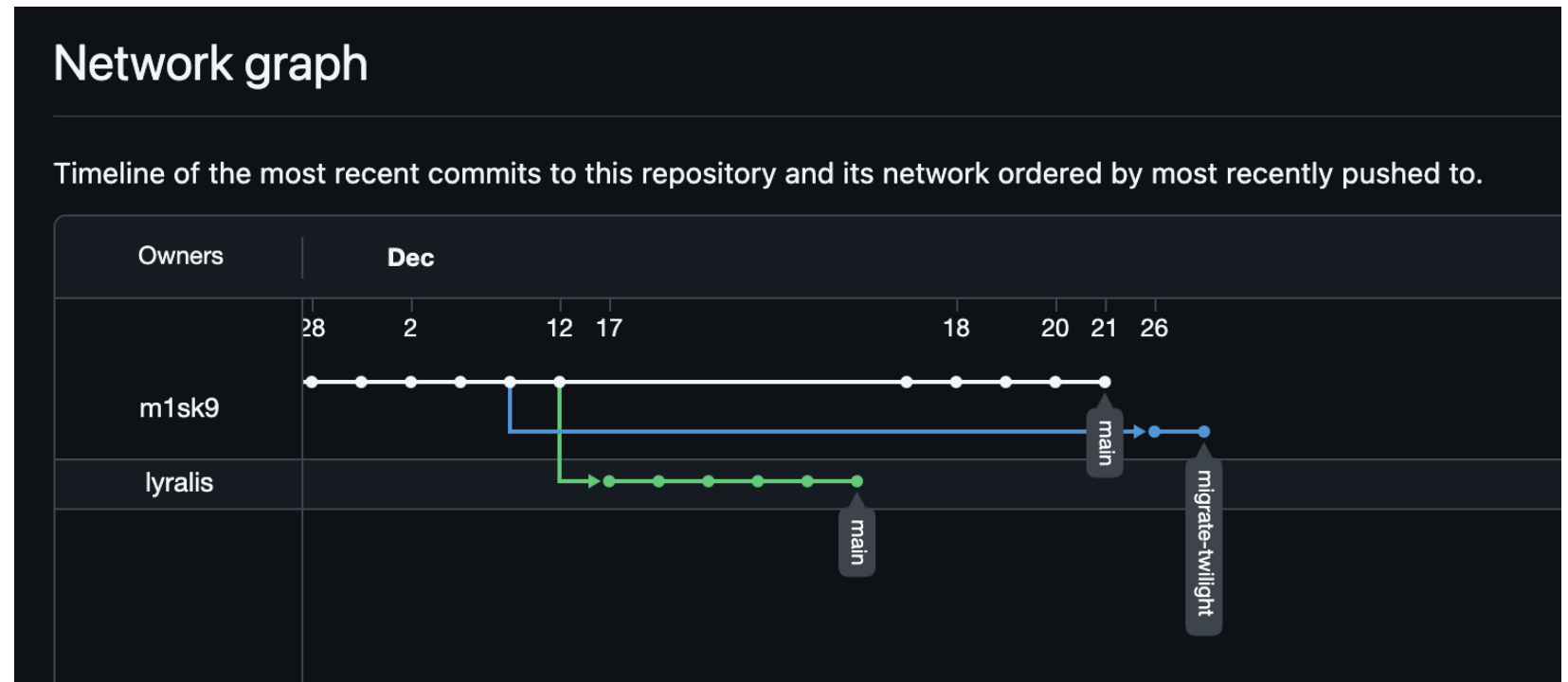


# コミット/プッシュ

- GitHub に変更をアップロードするには コミット・プッシュが必要
- `git add <ファイル名>`, `git add .` → 変更をステージングへ
- `git commit -m <メッセージ>` → コミットする
- `git push origin main` → リモートリポジトリにプッシュする

# ブランチの考え方

- Git には「コミット履歴の分岐点」である「ブランチ」という考え方がある



# ブランチ

- “main” ブランチ→メインブランチであり, 「動く状態」であることが求められる
- Git はこのブランチを無限に生やすことができる
- この “main” を維持したまま 「新機能の実装」や「バグ修正」などができ, うまくいかなければいつでも削除ができる

# 並行作業もできる

- 無限に作成できるので複数の作業を同時に進行できる

```
m1sk9@dev-m1sk9-lap1 ~/R/g/r/RuneCore (feat/Add-ability-weapon)> git branch
chore/add-claude-file
chore/update-docs-and-debug-server
claude/add-notification-queue-TmaMf
claude/implement-gacha-system-2NgLn
claude/player-settings-system-3Gc2B
docs/add-database
docs/deploy-javadoc
* feat/Add-ability-weapon
feat/add-command-system
feat/add-more-money-system
feat/custom-item
feat/debug-mode
feat/gui
feat/level-system
feat/menu
feat/money
feat/setup-db
feat/shop
feat/teleport
feat/world
fix/database-uuid-type
fix/deploy-javadoc-workflow
main
perf/debug-server
rename-package
renovate/mariadb-12.x
m1sk9@dev-m1sk9-lap1 ~/R/g/r/RuneCore (feat/Add-ability-weapon)>
```

# マージ

- ブランチを「統合」する
- リモートリポジトリの変更を統合する場合はプルリクエストというものを作成する
  - プルリクエストは GitHub での名称であって、それ以外のサービス (Bitbucket や GitLab) などではまた別の名称で呼ばれてる

# コンフリクト

- 全ての状態で完璧に「統合」するのは難しく，変更箇所が重なっていると「コンフリクト」という状態に陥る
- このコンフリクトの解消が Git を使う上で特に難しい難関とも言える場所でもある

# 基本はコマンドライン

- Git はコマンドラインからの操作が基本となる
- でもコマンドが複数あって覚えるのが難しい

```
m1sk9@dev-m1sk9-lap1 ~-> git --help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
[--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
[--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  backfill   Download missing objects in a partial clone
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
m1sk9@dev-m1sk9-lap1 ~->
```

# GUI 環境(画面上) からでの Git 操作

- JetBrains IDE (IDEA) や VSCode は Git 対応が進んでる
- 画面上からのクリックだけで簡単にコミットやプッシュができる
- ただ、この操作に慣れてしまってコマンドラインでの操作ができないのは本末転倒で、エンジニアを目指しているのなら尚更コマンドラインに慣れておかないといけない



# Progate というサービス

- Progate というプログラミング学習サービスに Git コースがあります
- 私は使ったことがないのでわかりやすいかどうかはわかりませんが、このパワーポイントよりは流石にわかりやすいと思います

<https://prog-8.com/courses/git>

# とりあえず

- コミット, プッシュ, チェックアウト, マージ, プルくらいは覚えておきましょう

# 使うな

- Git を使う上で使わないほうがいいコマンドを教えます

別にコマンド自体は使うし，便利です

何が起きるか十二分に理解していない状態で使わないほうがいいです

- `git push -f` , `git push --force`
- `git reset --hard`
- `git rebase` (これを僕たちは歴史改変と呼んでいます)

# Git を使う上で守っておいてほしいこと

1. コミットメッセージはわかりやすいものにしましょう
  - “直した”とか“新しい機能”とかふざけたメッセージは避けます
  - 使ったらしばく
2. わからなければ聞くこと
  - Git は難しく，操作をミスると履歴が破壊されたり，修復困難になりがちです（私も何回も履歴を破壊して他人を困らせました）
  - わからなければ同じチームの人などに聞きましょう