Repodex All Shellscripts Combined

build.sh

```
#!/bin/bash

# Exit immediately if a command exits with a non-zero status.
set -e

# Check if cargo is installed
if ! command -v cargo; then # check if cargo is in the path, if not, exit
        echo "cargo could not be found"
        exit 1
fi

FUNCTION_NAME=${1:-repodex} # default to repodex
CURRENT_PATH="$(pwd)"
FINAL_PATH="$CURRENT_PATH/lambdas/repodex/"

echo "Navigating to the lambda directory..."
cd "$FINAL_PATH"

echo "Building lambda function..."
cargo lambda build -r --arm64 -o zip;
```

colors.sh

```
# Colors
RESET='\033[0m'
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[0;33m'
BLUE='\033[0;34m'
MAGENTA='\033[0;35m'
CYAN='\033[0;36m'
WHITE='\033[1;37m'

# Function to print colored text
print_colored() {
  local color="$1"
  local message="$2"

  if [[ -n ${!color} ]]; then
    printf "${!color}%b$RESET" "$message"
  else
    echo "Invalid color: $color"
  fi
}
```

install.sh

```
#!/bin/bash

# Purpose: Installation submenu
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$SCRIPT_DIR/colors.sh"

# Variables
git_label='Git'
git_script="$SCRIPT_DIR/install/install-git.sh"

github_cli_label='Github CLI'
github_cli_script="$SCRIPT_DIR/install/install-github-cli.sh"

# Submenu for the Install option
show_install_menu() {
  echo "Welcome to the installer submenu!"
  echo "1) $git_label"
  echo "2) $github_cli_label"
  echo "3) Go back"
  echo
}
# Function to handle user selection in the install submenu
handle_install_selection() {
  case $1 in
    1)
      print_colored BLUE "==>"
      print_colored WHITE " Running $git_label installation Script...\n"
      "$git_script"
      echo
      ;;
    2)
      print_colored BLUE "==>"
      print_colored WHITE " Running $github_cli_label installation Script...\n"
      "$github_cli_script"
      echo
      ;;
    3)
      print_colored YELLOW "==>"
      print_colored WHITE " Returning to main menu..."
      exit 0
      ;;
    *)
      print_colored RED "==>"
      print_colored WHITE " Invalid option. Please try again.\n"
      ;;
```

```
  esac
}

# Loop for the install submenu
while true; do
  show_install_menu
  read -p "Select software to install: " install_choice
  echo
  handle_install_selection "$install_choice"
done
```

invoke.sh

```
#!/bin/bash

# Exit immediately if a command exits with a non-zero status.
set -e

# Check if cargo is installed
if ! command -v cargo; then # check if cargo is in the path, if not, exit
        echo "cargo could not be found"
        exit 1
fi

FUNCTION_NAME=${1:-repodex} # default to repodex
FUNCTION_PAYLOAD=${2:-'{ "name": "ben" }'} # default to '{ "name": "ben" }'

CURRENT_PATH="$(pwd)"
FINAL_PATH="$CURRENT_PATH/lambdas/repodex/"

echo "Navigating to the lambda directory..."
cd "$FINAL_PATH"

echo "Invoking lambda function..."
cargo lambda invoke "$FUNCTION_NAME" --data-ascii "$FUNCTION_PAYLOAD"
```

main.sh

```
#!/bin/bash

# Purpose: Repodex Entry
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$SCRIPT_DIR/colors.sh"
```

```
# Variables
install_label='Install'
install_script="$SCRIPT_DIR/install.sh"

setup_label='Setup'
setup_script="$SCRIPT_DIR/setup.sh"

watchdog_label='Watchdog'
watchdog_script="$SCRIPT_DIR/watchdog.sh"

# Function to display the menu
show_menu() {
  echo "Welcome to Repodex's Installer!"
  echo "Select an option:"
  echo "1) $install_label"
  echo "2) $setup_label"
  echo "3) $watchdog_label"
  echo "4) Exit"
  echo
}

# Function to run the selected option
run_option() {
  case $1 in
    1)
      print_colored BLUE "==>"
      print_colored WHITE " Running $install_label script...\n"
      # Add the command to run your first script here
      "$install_script"
      echo
      ;;
    2)
      print_colored BLUE "==>"
      print_colored WHITE " Running $setup_label script...\n"
      # Add the command to run your second script here
      "$setup_script"
      echo
      ;;
    3)
      print_colored BLUE "==>"
      print_colored WHITE " Running $watchdog_label script...\n"
      # Add the command to run your third script here
      "$watchdog_script"
      echo
      ;;
    4)
      print_colored RED "==>"
      print_colored WHITE " Exiting...\n"
      exit 0
      ;;
    *)
      print_colored RED "==>"
      print_colored WHITE " Exiting...\n"
      echo
      ;;
```

```
  esac
}

# Main loop
while true; do
  show_menu
  read -p "Enter your choice [1-4]: " choice
  echo
  run_option "$choice"
done
```

menu.sh

```
#/bin/bash
      E='echo -e';e='echo -en';trap "R;exit" 2
    ESC=$( $e "\e")
   TPUT(){ $e "\e[${1};${2}H";}
  CLEAR(){ $e "\ec";}
  CIVIS(){ $e "\e[?25l";}
   DRAW(){ $e "\e%@\e(0";}
  WRITE(){ $e "\e(B";}
   MARK(){ $e "\e[7m";}
 UNMARK(){ $e "\e[27m";}
      R(){ CLEAR ;stty sane;$e "\ec\e[37;44m\e[J";};
   HEAD(){ DRAW
           for each in $(seq 1 13);do
           $E "   x                                          x"
           done
           WRITE;MARK;TPUT 1 5
           $E "REPODEX SELECTION MENU                     ";UNMARK;}
           i=0; CLEAR; CIVIS;NULL=/dev/null
   FOOT(){ MARK;TPUT 13 5
           printf "ENTER - SELECT,NEXT                        ";UNMARK;}
  ARROW(){ read -s -n3 key 2>/dev/null >&2
           if [[ $key = $ESC[A ]];then echo up;fi
           if [[ $key = $ESC[B ]];then echo dn;fi;}
     M0(){ TPUT  4 20; $e "Install Watchdog";}
     M1(){ TPUT  5 20; $e "UnInstall Watchdog";}
     M2(){ TPUT  6 20; $e "Run Watcher Locally";}
     M3(){ TPUT  7 20; $e "Lambda Install";}
     M4(){ TPUT  8 20; $e "Date";}
     M5(){ TPUT  9 20; $e "ABOUT  ";}
     M6(){ TPUT 10 20; $e "EXIT   ";}
      LM=6
   MENU(){ for each in $(seq 0 $LM);do M${each};done;}
    POS(){ if [[ $cur == up ]];then ((i--));fi
           if [[ $cur == dn ]];then ((i++));fi
           if [[ $i -lt 0   ]];then i=$LM;fi
           if [[ $i -gt $LM ]];then i=0;fi;}
REFRESH(){ after=$((i+1)); before=$((i-1))
           if [[ $before -lt 0  ]];then before=$LM;fi
           if [[ $after -gt $LM ]];then after=0;fi
           if [[ $j -lt $i      ]];then UNMARK;M$before;else UNMARK;M$after;fi
```

```
          if [[ $after -eq 0 ]] || [ $before -eq $LM ];then
          UNMARK; M$before; M$after;fi;j=$i;UNMARK;M$before;M$after;}
   INIT(){ R;HEAD;FOOT;MENU;}
     SC(){ REFRESH;MARK;$S;$b;cur=`ARROW`;}
     ES(){ MARK;$e "ENTER = main menu ";$b;read;INIT;};INIT
  while [[ "$O" != " " ]]; do case $i in
        0) S=M0;SC;if [[ $cur == "" ]];then R;$e "\n$(w        )\n";ES;fi;;
        1) S=M1;SC;if [[ $cur == "" ]];then R;$e "\n$(ifconfig )\n";ES;fi;;
        2) S=M2;SC;if [[ $cur == "" ]];then R;$e "\n$(df -h    )\n";ES;fi;;
        3) S=M3;SC;if [[ $cur == "" ]];then R;$e "\n$(route -n )\n";ES;fi;;
        4) S=M4;SC;if [[ $cur == "" ]];then R;$e "\n$(date     )\n";ES;fi;;
        5) S=M5;SC;if [[ $cur == "" ]];then R;$e "\n$(xdg-open
https://github.com/ConnorBP/pokemon-git-tracker/blob/main/DESIGN.md)\n";R;exit
0;clear;fi;;
        6) S=M6;SC;if [[ $cur == "" ]];then R;exit 0;fi;;
  esac;POS;done
```

setup.sh

```
#!/bin/bash

# Purpose: Setup submenu
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$SCRIPT_DIR/colors.sh"

# Variables
git_label='Git & Github'
git_script="$SCRIPT_DIR/setup/setup-git.sh"

symlink_label='Symbiotic Link'
symlink_script="$SCRIPT_DIR/setup/setup-symlink.sh"

post_commit_label='Post-Commit'
post_commit_script="$SCRIPT_DIR/setup/setup-post-commit.sh"

discord_webhook_label='Discord Webhook'
discord_webhook_script="$SCRIPT_DIR/setup/setup-discord-webhook.sh"

# Submenu for the setup option
show_menu() {
  echo "Welcome to the setup submenu!"
  echo "1) $git_label"
  echo "2) $symlink_label"
  echo "3) $discord_webhook_label"
  echo "4) $post_commit_label"
  echo "5) Go back"
```

```
  echo "6) Exit"
  echo
}
# Function to handle user selection in the setup submenu
handle_selection() {
  case $1 in
    1)
      print_colored BLUE "==>"
      print_colored WHITE " Running $git_label setup script...\n"
      "$git_script"
      echo
      ;;
    2)
      print_colored BLUE "==>"
      print_colored WHITE " Running $symlink_label setup script...\n"
      "$symlink_script"
      echo
      ;;
    3)
      print_colored BLUE "==>"
      print_colored WHITE " Running $discord_webhook_label setup script...\n"
      "$discord_webhook_script"
      echo
      ;;
    4)
      print_colored BLUE "==>"
      print_colored WHITE " Running $post_commit_label setup script...\n"
      "$post_commit_script"
      echo
      ;;
    5)
      print_colored YELLOW "==>"
      print_colored WHITE " Returning to main menu..."
      exit 0
      ;;
    6)
      print_colored RED "==>"
      print_colored WHITE " Exiting...\n"
      exit 1
      ;;
    *)
      print_colored RED "==>"
      print_colored WHITE " Invalid option. Please try again.\n"
      ;;
  esac
}

# Loop for the setup submenu
while true; do
  show_menu
  read -p "Select software to setup: " setup_choice
  echo
  handle_selection "$setup_choice"
done
```

watch.sh

```
#!/bin/bash

# Exit immediately if a command exits with a non-zero status.
set -e

# Check if cargo is installed
if ! command -v cargo; then # check if cargo is in the path, if not, exit
        echo "cargo could not be found"
        exit 1
fi

CURRENT_PATH="$(pwd)"
FINAL_PATH="$CURRENT_PATH/lambdas/repodex/"

echo "Navigating to the lambda directory..."
cd "$FINAL_PATH"

echo "Watching lambda function..."
cargo lambda watch;
```

watchdog.sh

```
#!/bin/bash

# Purpose: Watchdog submenu
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$SCRIPT_DIR/colors.sh"

# Variables
git_label='Git'
git_script="$SCRIPT_DIR/install/install-git.sh"

github_cli_label='Github CLI'
github_cli_script="$SCRIPT_DIR/install/install-github-cli.sh"

# Submenu for the Install option
show_install_menu() {
  echo "Welcome to the installer submenu!"
  echo "1) $git_label"
  echo "2) $github_cli_label"
  echo "3) Go back"
  echo
```

```
}
# Function to handle user selection in the install submenu
handle_install_selection() {
  case $1 in
    1)
      print_colored BLUE "==>"
      print_colored WHITE " Running $git_label installation Script...\n"
      "$git_script"
      echo
      ;;
    2)
      print_colored BLUE "==>"
      print_colored WHITE " Running $github_cli_label installation Script...\n"
      "$github_cli_script"
      echo
      ;;
    3)
      print_colored YELLOW "==>"
      print_colored WHITE " Returning to main menu..."
      exit 0
      ;;
    *)
      print_colored RED "==>"
      print_colored WHITE " Invalid option. Please try again.\n"
      ;;
  esac
}

# Loop for the install submenu
while true; do
  show_install_menu
  read -p "Select software to install: " install_choice
  echo
  handle_install_selection "$install_choice"
done
```

install-git.sh

```
#!/bin/bash

# Purpose: Install git
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$(dirname "$SCRIPT_DIR")/colors.sh"

# Check if connected to the internet
check_connection() {
```

```
  print_colored YELLOW "==>"
  print_colored WHITE " Checking Internet Connection...\n"
  if ping -q -c 1 -W 1 google.com &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Internet Connection Detected.\n"
  else
    print_colored RED "==>"
    print_colored WHITE " No Internet Connection. Please connect to the internet
and try again.\n"
    exit 1
  fi
}

# Function to install Git via APT
install_git_apt() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected Debian/Ubuntu.\n"
  #@info: Update the package repository
  sudo apt-get update
  #@info: Check if Git is installed
  if command -v git &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Git is already installed. Proceed to the next step.\n"
    exit 0
  fi
  print_colored GREEN "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "git...\n"
  sudo apt-get install git -y
}

# Function to install Git via YUM
install_git_yum() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected CentOS/RHEL.\n"
  #@info: Update the package repository
  sudo yum update
  #@info: Check if Git is installed
  if command -v git &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Git is already installed. Proceed to the next step.\n"
    exit 0
  fi
  print_colored GREEN "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "git...\n"
  sudo yum install -y git
}

# Function to install Git via DNF
install_git_dnf() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected Fedora.\n"
  #@info: Update the package repository
  sudo dnf update
  #@info: Check if Git is installed
```

```
  if command -v git &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Git is already installed. Proceed to the next step.\n"
    exit 0
  fi
  print_colored GREEN "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "git...\n"
  sudo dnf install -y git
}

# Function to install Git on macOS
install_git_brew() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected MacOS.\n"
  #@info: Check if Homebrew is installed
  if ! command -v brew &>/dev/null; then
    print_colored YELLOW "Homebrew is not installed.\n"
    print_colored BLUE "==>"
    print_colored WHITE " Installing "
    print_colored GREEN "Homebrew...\n"
    /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
  fi

  #@info: Update the package repository
  brew update
  brew upgrade

  #@info: Check if Git is installed
  if command -v git &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Git is already installed. Proceed to the next step.\n"
    exit 0
  fi
  print_colored GREEN "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "git...\n"

  brew install git
}

# Main function
check_connection
if [[ "$OSTYPE" == "linux-gnu"* ]]; then
  if command -v apt &>/dev/null; then
    install_git_apt
  elif command -v yum &>/dev/null; then
    install_git_yum
  elif command -v dnf &>/dev/null; then
    install_git_dnf
  else
    print_colored RED "==>"
    print_colored WHITE " Unsupported package manager. Please install Git
manually.\n"
  fi
```

```
elif [[ "$OSTYPE" == "darwin"* ]]; then
  install_git_brew
else
  print_colored RED "==>"
  print_colored WHITE " Unsupported operating system. Please install Git
manually.\n"
fi
```

install-github-cli.sh

```
#!/bin/bash

# Purpose: Install Github CLI
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$(dirname "$SCRIPT_DIR")/colors.sh"

# Check if connected to the internet
check_connection() {
  print_colored YELLOW "==>"
  print_colored WHITE " Checking Internet Connection...\n"
  if ping -q -c 1 -W 1 google.com &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Internet Connection Detected.\n"
  else
    print_colored RED "==>"
    print_colored WHITE " No Internet Connection. Please connect to the internet
and try again.\n"
    exit 1
  fi
}

# Function to install Git via APT
install_gh_apt() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected Debian/Ubuntu.\n"

  #@info: Update the package repository
  sudo apt-get update

  #@info: Check if GitHub CLI is installed
  if command -v gh &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " GitHub CLI is already installed. Proceed to the next
step.\n"
    exit 0
  fi
```

```
  print_colored BLUE "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "GitHub CLI...\n"
  sudo apt-get install gh -y
}

# Function to install Git via YUM
install_gh_yum() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected CentOS/RHEL.\n"

  #@info: Update the package repository
  sudo yum update

  #@info: Check if GitHub CLI is installed
  if command -v gh &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " GitHub CLI is already installed. Proceed to the next
step.\n"
    exit 0
  fi
  print_colored BLUE "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "GitHub CLI...\n"
  sudo yum install -y gh
}

# Function to install Git via DNF
install_gh_dnf() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected Fedora.\n"

  #@info: Update the package repository
  sudo dnf update

  #@info: Check if GitHub CLI is installed
  if command -v gh &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " GitHub CLI is already installed. Proceed to the next
step.\n"
    exit 0
  fi
  print_colored BLUE "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "GitHub CLI...\n"
  sudo dnf install -y gh
}

# Function to install Git on macOS
install_gh_brew() {
  print_colored GREEN "==>"
  print_colored WHITE " Detected MacOS.\n"

  #@info: Check if Homebrew is installed
  if ! command -v brew &>/dev/null; then
    print_colored YELLOW "Homebrew is not installed.\n"
```

```
    print_colored BLUE "==>"
    print_colored WHITE " Installing "
    print_colored GREEN "Homebrew...\n"
    /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
  fi

  #@info: Update Homebrew and installed formulas
  brew update
  brew upgrade

  #@info: Check if GitHub CLI is installed
  if command -v gh &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " GitHub CLI is already installed. Proceed to the next
step.\n"
    exit 0
  fi

  print_colored BLUE "==>"
  print_colored WHITE " Installing "
  print_colored GREEN "GitHub CLI...\n"
  brew install gh
}

# Main function
check_connection
if [[ "$OSTYPE" == "linux-gnu"* ]]; then
  if command -v apt &>/dev/null; then
    install_gh_apt
  elif command -v yum &>/dev/null; then
    install_gh_yum
  elif command -v dnf &>/dev/null; then
    install_gh_dnf
  else
    print_colored RED "==>"
    print_colored WHITE " Unsupported package manager. Please install Git
manually.\n"
  fi
elif [[ "$OSTYPE" == "darwin"* ]]; then
  install_gh_brew
else
  print_colored RED "==>"
  print_colored WHITE " Unsupported operating system. Please install Git
manually.\n"
fi
```

setup-discord-webhook.sh

```
#!/bin/bash

# Script to setup Discord webhook values in the current Git repository
```

```
# Define the config file name
config_filename=".discord_webhook_config"

# Check if we're in a Git repository
if git rev-parse --git-dir >/dev/null 2>&1; then
  config_file_path="$(git rev-parse --show-toplevel)/$config_filename"

  # Prompt for Discord Webhook ID
  read -p "Enter your Discord Webhook ID: " webhook_id

  # Prompt for Discord Webhook Token (input not shown)
  read -sp "Enter your Discord Webhook Token: " webhook_token
  echo # Move to a new line after input

  # Confirm before saving
  echo "You're about to save this information. Continue? (y/n)"
  read -n 1 -r
  echo # Move to a new line for output

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    # Save the values to the config file
    echo "webhook_id=$webhook_id" >"$config_file_path"
    echo "webhook_token=$webhook_token" >>"$config_file_path"

    # Add the config file to .gitignore to ensure it's not tracked
    gitignore_path="$(git rev-parse --show-toplevel)/.gitignore"

    if ! grep -q "$config_filename" "$gitignore_path"; then
      echo "$config_filename" >>"$gitignore_path"
      echo "$config_filename has been added to .gitignore."
    fi

    echo "Webhook details saved to $config_filename."
  else
    echo "Setup canceled by user."
  fi
else
  echo "This directory is not a Git repository."
fi
```

setup-git.sh

```
#!/bin/bash

# Purpose: Setting up Git and GitHub CLI
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
```

```
source "$(dirname "$SCRIPT_DIR")/colors.sh"

# Function to check for required commands
check_command() {
  if ! command -v "$1" &>/dev/null; then
    print_colored RED "==>"
    print_colored WHITE " Required command not found: $1\n Please install it and
try again.\n"
    exit 1
  fi
}

# Check for required commands
check_command ping
check_command gh
check_command git

# Check if connected to the internet
check_connection() {
  print_colored YELLOW "==>"
  print_colored WHITE " Checking Internet Connection...\n"
  if ping -q -c 1 -W 1 google.com &>/dev/null; then
    print_colored GREEN "==>"
    print_colored WHITE " Internet Connection Detected.\n"
  else
    print_colored RED "==>"
    print_colored WHITE " No Internet Connection. Please connect to the internet
and try again.\n"
    exit 1
  fi
}

# Main function
check_connection
gh auth login

# Check if GitHub CLI login was successful
if gh auth status; then
  print_colored GREEN "==>"
  print_colored WHITE " GitHub CLI authentication successful. Continue to use the
watchdog.\n"
else
  print_colored RED "==>"
  print_colored WHITE " GitHub CLI authentication failed. Please try again or
check your authentication method.\n"
  exit 1
fi
```

setup-post-commit.sh

```
#!/bin/bash

# Purpose: Setup symlink submenu
```

```
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$(dirname "$SCRIPT_DIR")/colors.sh"

# Write or append to the post-commit hook
cat <<EOF >>.git/hooks/post-commit
#!/bin/bash

# Find the root directory of the Git repository
REPO_ROOT=\$(git rev-parse --show-toplevel)

# Construct the path to the configuration file
CONFIG_FILE="\$REPO_ROOT/.discord_webhook_config"

# Check if the configuration file exists
if [ -f "\$CONFIG_FILE" ]; then
    # Source the configuration file
    source "\$CONFIG_FILE"

    # Now you can use variables defined in .discord_webhook_config
    echo "Webhook ID: \$webhook_id"
    echo "Webhook Token: \$webhook_token"
else
    echo "Configuration file not found: \$CONFIG_FILE"
fi

api_url="https://ad2bgbgkd2e3kaqhydkos7dsme0crshb.lambda-url.us-east-
2.on.aws/action"
commit_hash=\$(git rev-parse HEAD)
commit_author=\$(git log -1 --pretty=format:'%an')
read added removed modified <<<\$(git diff --numstat HEAD~1 HEAD | awk
'{adds+=\$1; dels+=\$2; mods+=\$3} END {print adds, dels, mods}')
repo_url=\$(git config --get remote.origin.url)
repo_name=\$(echo \$repo_url | sed -e 's/.*\/\([^\.]*\)\.git/\1/')
commit_message=\$(git log -1 --pretty=format:'%B')

json_body=\$(cat <<EOF2
{
    "repo": "\$repo_name",
    "branch": "\$(git rev-parse --abbrev-ref HEAD)",
    "date": "\$(date +'%b %d, %Y')",
    "commit": {
        "hash": "\$commit_hash",
        "url": "https://github.com/ConnorBP/pokemon-git-
tracker/commit/\$commit_hash",
        "author": "\$commit_author",
        "message": "\$commit_message"
    },
    "changes": {
        "added": \$added,
```

```
        "removed": \$removed,
        "modified": \$modified
    },
    "webhook": {
        "id": "\$webhook_id",
        "token": "\$webhook_token"
    }
}
EOF2
)

curl -X POST -H "Content-Type: application/json" -d "\$json_body" \$api_url
EOF

chmod +x .git/hooks/post-commit

echo "Post-commit hook setup complete. It will now run after every commit."
```

setup-symlink.sh

```
#!/bin/bash

# Purpose: Setup symlink submenu
# Author: The bois

# Attempt to follow the symlink with readlink -f (Linux)
SCRIPT_DIR=$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}" 2>/dev/null ||
realpath "${BASH_SOURCE[0]}" 2>/dev/null || echo "${BASH_SOURCE[0]}")")"
&>/dev/null && pwd)

# Import colors
source "$(dirname "$SCRIPT_DIR")/colors.sh"

# Check if the symlink destination directory is in PATH
if [[ ":$PATH:" != *":/usr/local/bin:"* ]]; then
  print_colored YELLOW "==>"
  print_colored WHITE " Your /usr/local/bin directory isn't in the PATH.
Adjusting...\n"
  export PATH="$PATH":/usr/local/bin
fi

# Create a symbolic link for main.sh
sudo ln -s "$PWD/scripts/main.sh" ~/bin/repodex

# Check if Repodex symlink was successful
if command -v "repodex" &>/dev/null; then
  print_colored GREEN "==>"
  print_colored WHITE " Repodex has been set up. You can now run 'repodex' from
anywhere."
else
  print_colored RED "==>"
  print_colored WHITE " Repodex setup failed. The 'repodex' command is not
available.\n"
```

```
  exit 1
fi
```