GOS2022
4/22/2024: Contributor: Shinika Balasundar

GitHub Process Manual

A DevOps Guide

Table Of Contents:

Introduction

This document will cover general GitHub processes and a strategy for using various GitHub functionalities, with provided demos on how the process will entail.

GitHub Pages for Website Release

GitHub Pages provides a seamless way to host static websites directly from your GitHub repository.

Steps To Create Website:

Navigate to your repository on GitHub.

Go to the "Settings" tab.

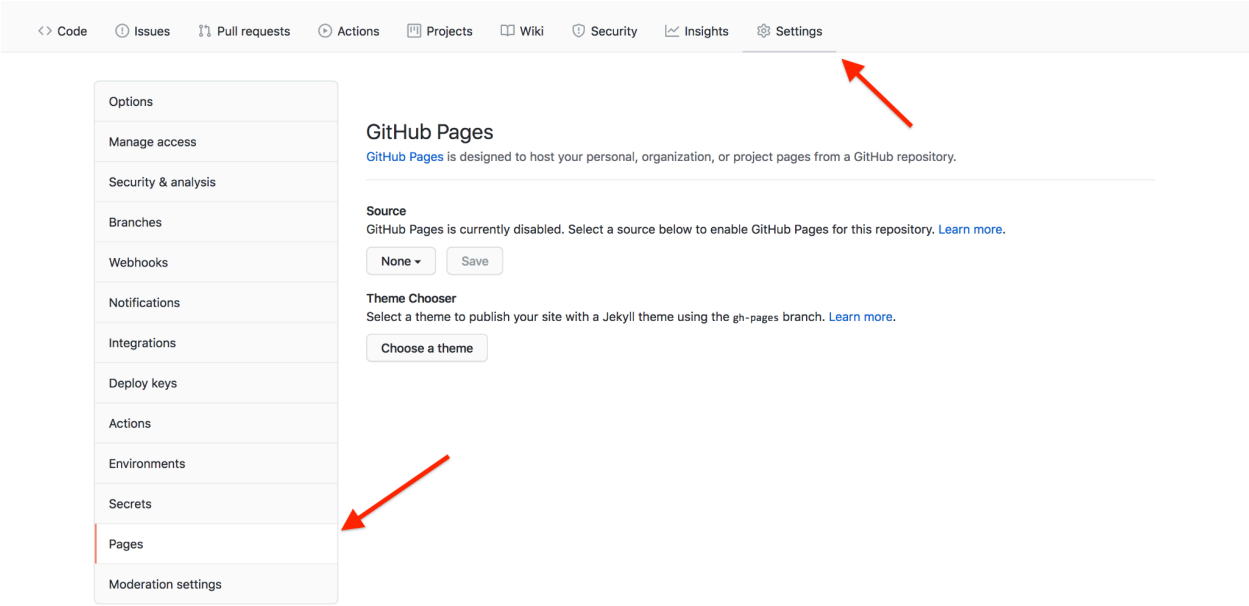
Scroll down to the "GitHub Pages" section.

Select a source branch (e.g., main or master) and a folder for your website content (e.g., /docs or /docs/build).

Customize your website using Markdown, HTML, and CSS.

Commit and push changes to your repository to publish the website.

There are opportunities to expand this website stack to more UI-emphasized design or other application stacks such as MERN.



Example Online

Utilizing Issues Page for Task Organization

The Issues page on GitHub serves as a centralized hub for tracking tasks, bugs, and feature requests.

Steps:

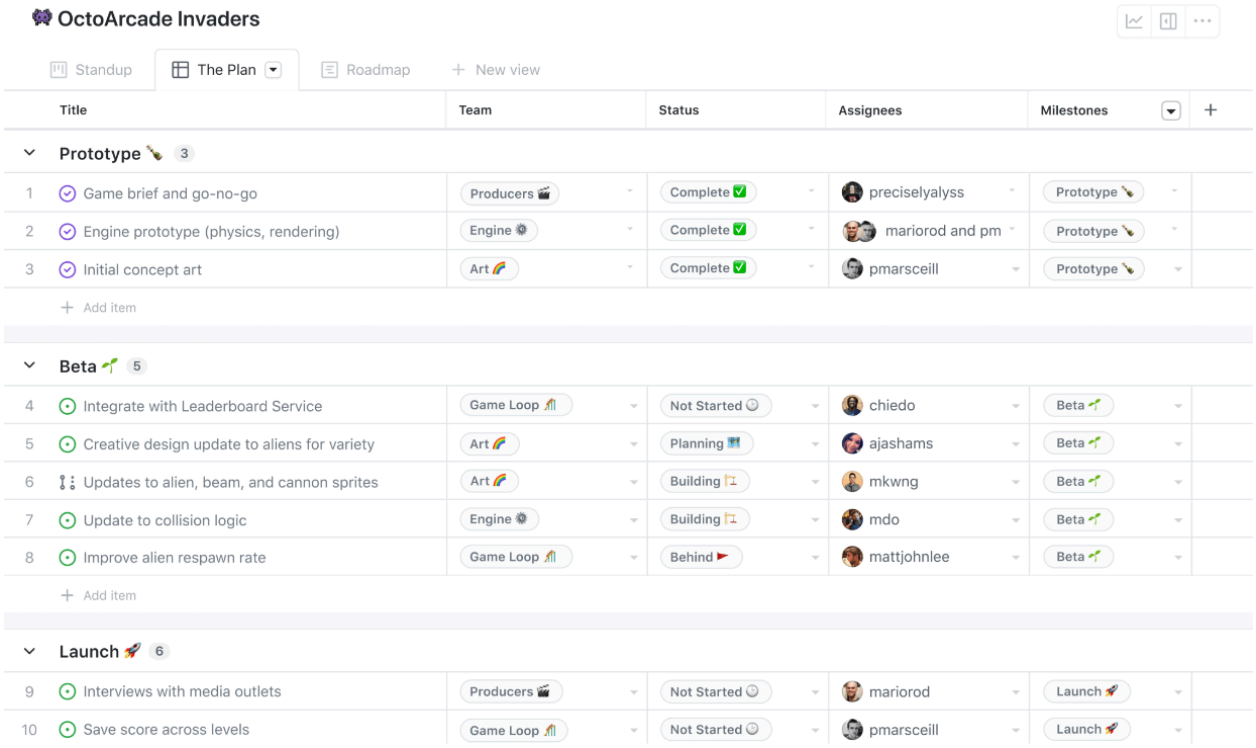
Navigate to the "Issues" tab in your repository.

Click on "New Issue" to create a new task.

Use labels to categorize issues (e.g., bug, enhancement, documentation).

Assign issues to team members and set milestones for project goals.

You can use the project board feature to visualize and prioritize tasks across columns (e.g., To Do, In Progress, Done).



OctoArcade Invaders

Standup | The Plan | Roadmap | New view

	Title	Team	Status	Assignees	Milestones
	Prototype 3				
1	Game brief and go-no-go	Producers	Complete	preciselyalyss	Prototype
2	Engine prototype (physics, rendering)	Engine	Complete	mariorod and pm	Prototype
3	Initial concept art	Art	Complete	pmarsceill	Prototype
	Add item				
	Beta 5				
4	Integrate with Leaderboard Service	Game Loop	Not Started	chiedo	Beta
5	Creative design update to aliens for variety	Art	Planning	ajashams	Beta
6	Updates to alien, beam, and cannon sprites	Art	Building	mkwng	Beta
7	Update to collision logic	Engine	Building	mdo	Beta
8	Improve alien respawn rate	Game Loop	Behind	mattjohnlee	Beta
	Add item				
	Launch 6				
9	Interviews with media outlets	Producers	Not Started	mariorod	Launch
10	Save score across levels	Game Loop	Not Started	pmarsceill	Launch

Example from GitHub

GitHub Actions for Code Quality Checks

Implement GitHub Actions to automate code quality checks for programming.

Steps:

Create a .github/workflows directory in your repository.

Add a YAML file (e.g., ci.yml) to define your workflow.

Configure the workflow to include steps such as compiling code, running tests, and checking code style.

Utilize GitHub's marketplace for pre-built actions or create custom actions as needed.

Define triggers for the workflow (e.g., push to specific branches, pull requests).

Test and validate the workflow by pushing code changes and observing the actions' output.

```yaml
name: Run Cppcheck

on:
  pull_request:
    branches:
      - main

jobs:
  cppcheckfinal:
    name: cppcheckfinal
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Install cppcheck
        run: sudo apt-get install -y cppcheck  # Install cppcheck on the Linux runner

      - name: Run Cppcheck
        run: cppcheck --enable=all 'unit_test/main'  # Run cppcheck on the main directory

      - name: Check Cppcheck results
        run: |
          cppcheck --enable=all --xml 'unit_test/main' 2> cppcheck_results.xml
          if grep -q "<error " cppcheck_results.xml; then
            echo "Cppcheck found errors."
            exit 1
          else
            echo "Cppcheck passed without errors."
          fi
```

Example File: Checks within unit_test/main to find CPPcheck failures/warnings

Project Page for Agile Sprint Board

Create a Project Page on GitHub to manage tasks using an Agile sprint board. This can be done for any process and not just Agile.

Steps:

Go to the "Projects" tab in your repository.

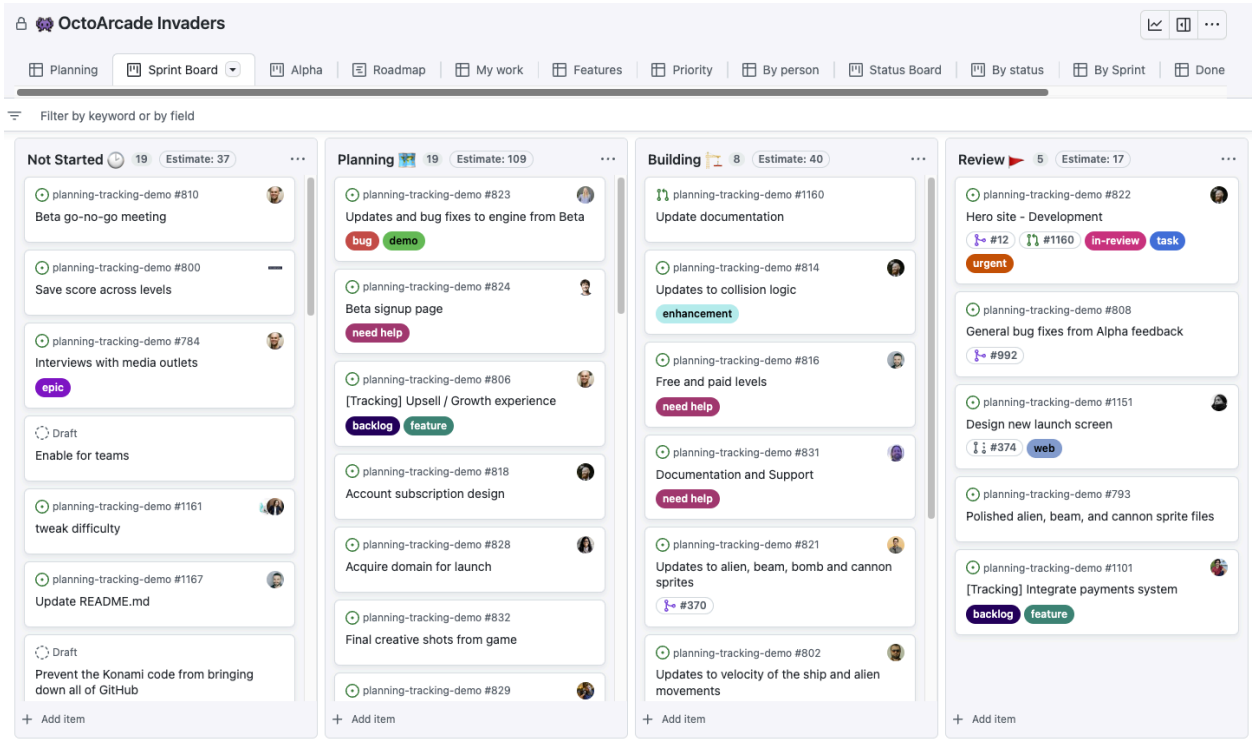
Click on "New project" to create a new project.

Choose a template (e.g., Automated Kanban, Basic Kanban) or create a custom template.

Customize columns based on your workflow (e.g., To Do, In Progress, Review, Done).

Add and prioritize tasks/cards within the project board.

Custom features can be added in regards to start/end-date connecting with issues to close and open. Also, it can be integrated together with various releases.



Example GitHub Project Page

GitHub Discussions for Community Engagement

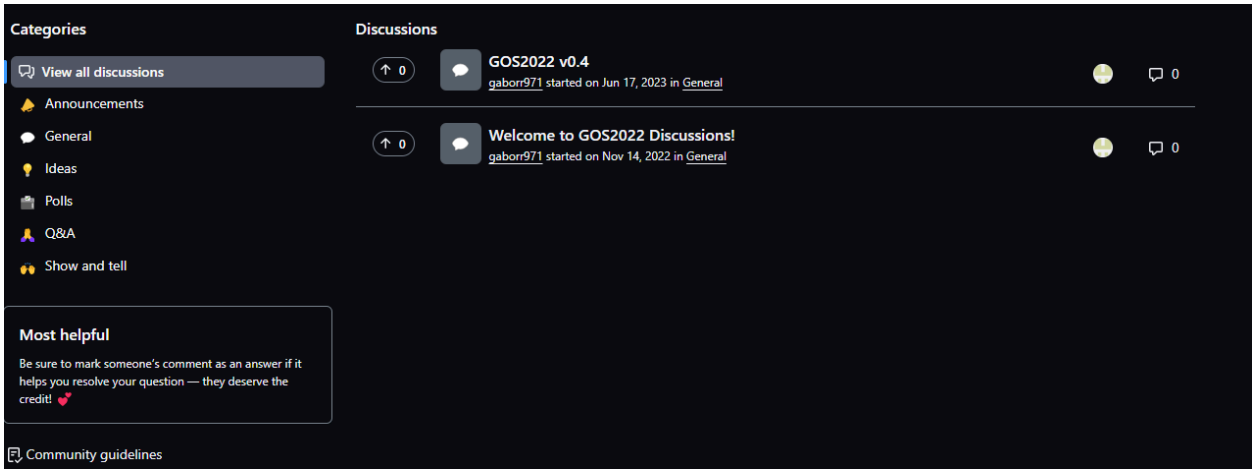
GitHub Discussions offers a platform for engaging with the community, gathering feedback, and fostering collaboration.

Steps:

Go to the "Discussions" tab in your repository.

Click on "New discussion" to start a new thread or category.

Define discussion categories (e.g., General, Q&A, Feature Requests) to organize topics.



GOS Discussions Page as of: 4/22/2024

Key Components of DevOps Implementation

Effective DevOps implementation involves several key components to ensure seamless collaboration and continuous improvement:

Version Control: **Utilize Git for version control** to track changes and manage codebase collaboratively.

Continuous Integration (CI): **Implement CI pipelines to automate code integration**, testing, and deployment.

Continuous Delivery (CD): Set up CD pipelines to automate software delivery and **deployment to production environments.**

Monitoring and Logging: **Implement monitoring tools to track system performance**, detect issues, and analyze logs for troubleshooting.

Collaboration and Communication: Utilize collaboration platforms such as Slack, Microsoft Teams, or Discord for **real-time communication** and collaboration among team members.

Security: Integrate **security practices into the development and deployment pipelines** to ensure code and infrastructure are secure.

Branch Protection and Pull Requests

Branch protection and pull request (PR) management are crucial for maintaining code quality and security in a collaborative environment.

Branch Protection:

Navigate to your repository settings.

Go to the "Branches" tab.

Choose the branch (e.g., main, master) you want to protect.

Enable branch protection rules such as requiring pull request reviews, status checks, and branch administrators' approval.

Optionally, enforce other restrictions like requiring signed commits or preventing force pushes.

Pull Requests (PRs):

When creating a new feature or fixing a bug, create a new branch from the main or development branch.

Make changes and commit them to your branch.

Push your branch to the repository and create a pull request.

Assign reviewers, add labels, and provide a clear description of the changes in the pull request.

Once the PR is approved and all checks pass (e.g., code review, CI/CD checks), merge the PR into the main branch.

By implementing branch protection and PR management, you ensure that changes to the codebase are reviewed, tested, and approved before merging, maintaining code integrity and project stability.

Enforce maintainers of certain features and have them be automatic reviewers of code PR requests.

Example: Kernel Maintainer: John, GOS Tool Maintainer: Alice

Onboarding Process for New Collaborator

Branch protection and pull request (PR) management are crucial for maintaining code quality and security in a collaborative environment.

1. **Introduction to Project**
    a. The new collaborator is introduced to the open-source project's objectives, scope, and mission **through the GitHub pages website (GOS.com)**
2. GitHub Account Setup
    a. The collaborator creates a GitHub account if they don't have one already.
3. Access Request
    a. The collaborator requests access to the project repository by contacting the project administrator or maintainer.
4. Familiarization with GitHub
    a. The new collaborator familiarizes themselves with GitHub's interface, version control concepts, and basic Git commands. GOS website demonstrates processes.
    b. Review of collaboration guidelines, including coding standards, commit message conventions, and branch naming conventions.
        i. **GOS standardization: user/name/feature, user/name/issue#**
5. Code Contribution
    a. Guidance on how to fork the repository, create feature branches, make changes, commit code, and push changes to GitHub.
6. Pull Requests
    a. Overview of the pull request (PR) process, including creating PRs, assigning reviewers, addressing feedback, and merging changes.
7. Code Review
    a. Understanding the importance of code review in maintaining code quality and receiving constructive feedback from team members.
8. Community Engagement
    a. Encouragement to participate in GitHub Discussions, engage with the community, and contribute ideas and feedback.