



FACEBOOK

#BuildforSDG

Setup & Configuration Guide

Charles Odili

Snr. Technical Program Manager

April, 2020

The Goal

Fast track the path to competence, market-readiness & technical leadership for African developers, at scale.

The Strategy

Join a distributed agile team that identifies a relevant high-impact problem and builds an innovative MVP with solid user centric, software engineering, and product delivery standards

AGENDA



Team, Repo & Tools Setup

Product Delivery Setup

Ethics & Collaboration

What Success Looks Like

Team, Repo & Tools Setup

Professionally, software is built and shipped with a collection of tools that play key roles in the continual delivery of quality products that meet stakeholder expectations.

To succeed in this industry and prepare for the market, engineers and team leads need to build mastery for this workflow.

The #BuildForSDG GitHub ORG
is the community where all the
magic is happening

Invites have been going out already. Reach out to
@SAM ESIDEM on Slack, or your **mentor** and **TTL** if
you have not received an invite to the GitHub ORG

Team Setup

Team Setup

1 Mentors should have accepted invites to join the Mentors team and TTLs accepted to join the TTLs team.

2

Check your emails and let us know on Slack if you are yet to be invited to the #BuildforSDG ORG or into your respective teams

3

As a mentor or TTL, if you have not already done so, create a team for each team you manage. Use the right name format, which is **Team-XYZ**. Existing teams that do not follow this format should be renamed or re-created. During team creation, make sure to set **Team visibility** to **Visible**



5

As a mentor or TTL, locate the team(s) you manage and add members of the team to it. You might need to wait for them to first be added to the ORG before you can complete this step

4

During team creation, make sure to not add the team under any parent team, and set **Team visibility** to **Visible**

Repository Setup



Repository Setup

1

As a mentor or TTL, using one of our **language-specific templates**, create a repository for the team(s) you manage.

4

Make sure to set the repo as **Public**, then click the **Create repository** button

- ☒  **Public**
Anyone can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

2

When creating a repo, use one of the JavaScript, PHP, or Python **templates**

Repository template

Start your repository with a template repository's contents.

No template ▾



3

With **BuildForSDG** set as the owner, give the repo a name which should ideally be the name of the solution the team wants to create. If that is not set yet, use the team name, formatted like :

Team-XYZ-Product
Team-XYZ-Frontend
Team-XYZ-Backend

You can create more than one repos if you need to

Owner

 **BuildForSDG** ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? H

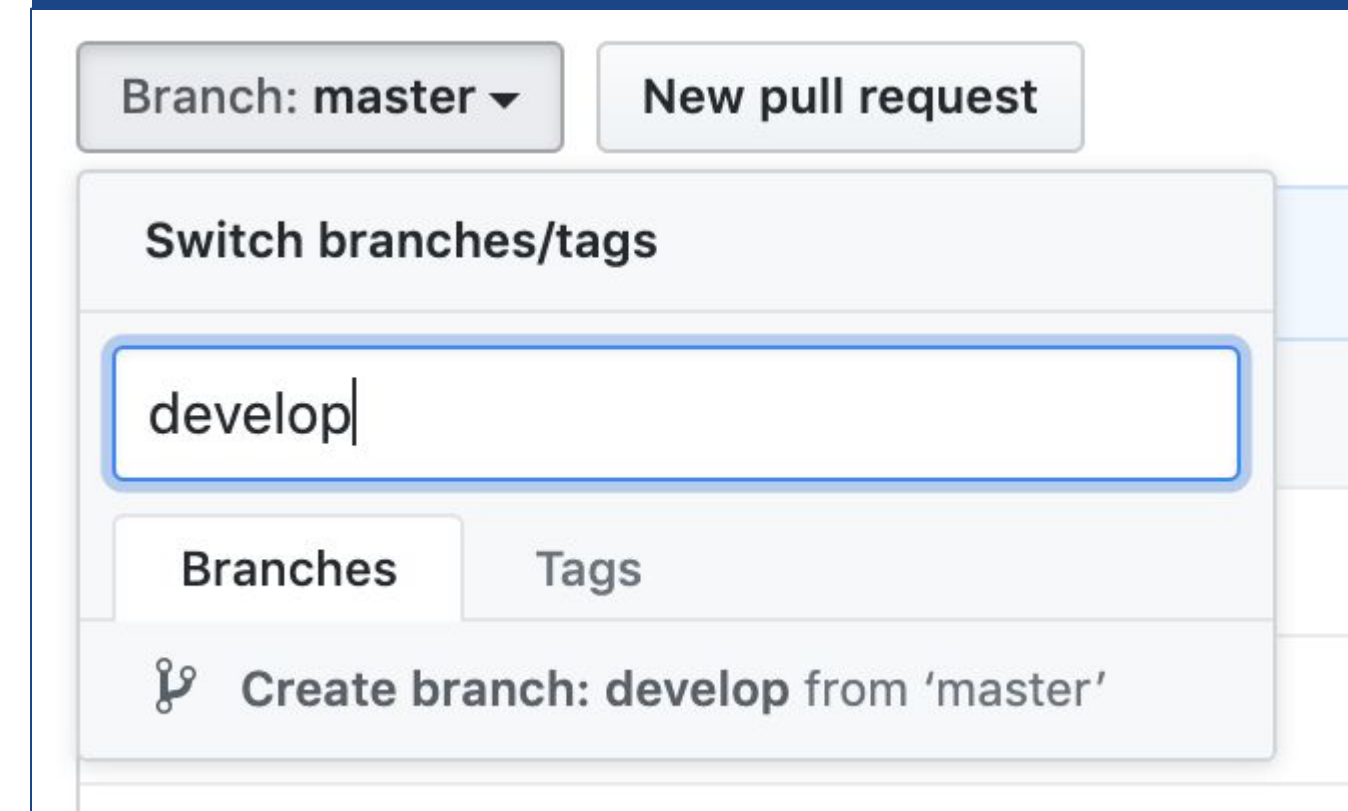
5

As a mentor or TTL, you are to create a new **develop** branch from the existing **master** branch in the repo



6

Click on the **Branch:master** drop down button and type **develop** into the input field. Finally, click on the Create branch:develop from “master” button at the bottom of the popup



Repository Configuration

7 On the repo, click on **Settings**, then click on **Manage access**. To add the team to the repo, click the green **Invite teams or people** button

8 On the popup window, type in the team name to add it. Give the team **Write** access to the repo

9 Still under **Settings**, click on **Branches** and set the default branch to **develop**.



10 Click on **Add rule** under **Branch protection rules** and type **develop** into the **Branch name pattern** field

11

Under **Protect matching branches**, check **Require pull request reviews before merging** and select **3** in the **Require approving reviews** drop down. This ensures 2 team members + the TTL are able to review and approve PRs. Also, check the **Dismiss stale pull request approvals when new commits are pushed** to make sure developers don't by-pass reviews if they update a PR that is already reviewed and approved.

12

Check *Require status checks to pass before merging* and the *Require branches to be up to date before merging* option under it.

Make sure the *Codacy/PR Quality Review* option is **NOT** checked if it is there.

Finally, check *Restrict who can push to matching branches* then add the Github usernames of the TTL and mentor of this team into the input field that shows up

☒ **Require branches to be up to date before merging**

This ensures pull requests targeting a matching branch have been merged. Pull requests will not take effect unless at least one status check is enabled (see [this](#)).

Status checks found in the last week for this repository

☐ Codacy/PR Quality Review



☒ **Restrict who can push to matching branches**

Specify people, teams or apps allowed to push to matching branches. Require these people, teams and apps from merging if the checks fail.

🔍 Search for people, teams or apps

People, teams or apps with push access



Organization administrators, repository administrators, and users with the Maintain role.

Admins can always push. Users with the Maintain role can push when required status checks pass.

13

Repeat steps 10, 11 & 12 for the **master** branch.

In step 11 (for the master branch), there is no need to check the **Require pull request reviews before merging** option since PRs requiring reviews will not be raised to the master branch





13

Since we have disabled automated review and grading of Pull Requests (PRs) from Codacy, we'll need an alternative, likely powered by GitHub Actions.

This slide will be updated with steps to put PR checks in place

14

On Slack, go to the team channel and type the following command. Follow the prompts to complete integrating the team repo with the team channel

Command:

```
/github subscribe owner/repo
```

E.g

```
/github subscribe BuildForSDG/Team-253
```

Codacy Setup

We are leveraging Codacy for intelligent and automated code analysis on the **develop** and **master** branches of product codebases.

This helps to provide industry-grade and researched-backed code quality rating throughout the program, enabling transparency and visibility on the quality of work output

Codacy Setup

1 As the TTL or mentor, go to <https://app.codacy.com> and login with GitHub

2

The welcome screen will ask for a team size figure. Entire an estimate for your team(s) size and click **Lets get started**

3

You should see the **BuildForSDG** organisation listed. Click the **Join** button on it watch and out for the “**You were added to the organization**” notification.

Click on the **Go to add repositories** button, though you might need to refresh the page.

5 On the bottom left of the Codacy dashboard, click on the **Settings** menu item, then click on the **Branches** tab at the top of the page

In the **Select Main Branch** section, select the **develop** branch



4

Click the **Add** button on the repository of your team(s). You might need to first type the repository name into the search field to limit the number of repositories displayed. Click on **Go to repository** after the add process is complete

6

Right on the same page, make sure the **develop** and **master** branches are selected under **ANALYSIS** so they get graded



ANALYSIS GRADE

develop Main branch



A

master



A

7

Click on the Integrations tab and make sure you **un-check** **Pull Request Status**, **Pull Request Comment** and **Pull Request Summary**

 **Github** Integrated by:  Charles Opute Odili

Codacy is now integrated with your GitHub repository. \n integration.

- ☐ Pull Request Status
- ☐ Pull Request Comment
- ☐ Pull Request Summary

Codacy Setup

8 Still under **Settings**, click on the **General** tab at the top and scroll down to the **Codacy Badge** section

9

Click on the **Add badge to repository** button. This will raise a PR on the repository



10

Go to the repository on GitHub, click on the **Pull requests** tab and then click on **Add a Codacy badge to README.md**.

After any available checks pass, notice you still cannot merge the PR? Reviews required. All these work on configuration is paying off alfterall.

Now, click on the **Settings** tab of the repo, then click on **Branches** on the left of the Settings page.

11

Under the **Branch protection rules** section, click the **Edit** button in the option for the **develop** branch.

Under **Require pull request reviews before merging** temporarily set **Require approving reviews** to 1, and also make sure **Codacy/PR Quality Review** under the **Require status checks to pass before merging** section is **NOT checked**.

13

Click on **Add your review**

12

Go back and click on the **Add a Codacy badge to README.md** PR under the **Pull requests** tab of the repo



Review required

At least 1 approving review is required by reviewers with write access. [Learn more.](#)

[Add your review](#)



14

Add a comment and **Approve** the changes in the PR

Changes from all commits ▾ File filter... ▾ Jump to... ▾ ⚙ ▾

0 / 1 files viewed ⓘ

[Review changes ▾](#)

4 ■ ■ ■ README.md

...	...	@@ -1 +1,3 @@
1	-	# dom-utils
	1	+ # dom-utils
	2	+
	3	+ [

15

Merge and accept the PR to complete adding the Codacy badge to the repo. If you used one of our starter repos, you will need to go back and remove the default Codacy badge with a **D rating** - better to create an issue for this and assign it to a team member

Merge pull request

You can also [open this in GitHub Desktop](#) or view

Code Quality D

16

Now go back to the **Settings** tab on the repo, and click on **Branches** on the left of the page.

Under the **Branch protection rules** section, click the **Edit** button in the option for the **develop** branch.

Under **Require pull request reviews before merging** set **Require approving reviews** to 3. This is **super important**.



17

We will be updating here with details on how to setup code coverage reporting for your repository, using Codacy

 coverage 91%



Product Delivery Setup

You are innovating on a real problem & interacting with real stakeholders or their proxies. There has to be a roadmap with actual deadlines as your work [potentially] impacts real users.

To succeed in this industry and prepare for the market, engineers need to be user obsessed, manage stakeholders, and build the muscle for high quality product delivery

**The #BuildForSDG GitHub ORG
is the community where all the
magic is happening**

**We will be centralising the project management and
tracking of all innovations using the in-built Projects
features on GitHub**

1

As a TTL or mentor, on the BuildForSDG ORG, go to the [Projects](#) tab and click on the green **New project** button. You are to create a GitHub project for each of the teams you are supporting

Project template

Save yourself time with a pre-configured project board template.

Template: **Automated kanban with reviews** ▼

Linked repositories

Search BuildForSDG to link repositories to this project for more accurate suggestions

 Search by repository name

 Linked repositories: None yet!

2

The project should be named after the solution or product the team is working on. If this has not been set, then name the project as **Team-XYZ-Project**. In the **Project template** section, **make sure to select the Automated kanban with reviews option**, and set **Visibility** to **Public**.

Finally, in the Linked repositories section, type in and select the name of the team repo(s) to link them to the project being created

3

As a TTL or mentor, on the BuildForSDG ORG, go to the **Issues** tab for a team's repo and click on the **Milestones** link, just before the green **New issue** button.

Click the green **Create a milestone** button to proceed

Labels 9

Milestones 0

New issue

4

Milestones will be used to track work for the three 2-week long sprints in the BuildForSDG program. Name the milestones as **Sprint-1, Sprint-2 and Sprint-3 ONLY**, and set / change their respective due dates as **May 15th, May 29th and June 12th**. Ideally, you should aim and plan to wrap up your project before June 12th

This updated sprint schedule could mean your team's tasks (issues) might need to be re-organised into the right milestones

Sprint - 1

Due by May 03, 2020 Last updated about 12 hours ago

80% complete 1 open 4 closed

Edit Close Delete

5

On the BuildForSDG ORG, click on the Teams tab and search for your team(s). Click on the team link to go to the team's page



#BuildForSDG

 Repositories 434

 Packages

 People 1.7k

 Teams 233



 team-267

☐ Select all

☐ **Team-267**
Build for SDG Team 267



6

On the team's page, click on the **Repositories** tab on the top right of the screen, then use the **Add repository** button to search for, and add the repositories the team is working on. Make sure to give the team **Write** access to the repositories added here

Discussions

Members 10

Teams 0

Repositories 1

Find a repository...

Add repository

☐ Select all

☐ BuildForSDG/farmrail updated 2 hours ago

Write

7

On the team's page, click on the **Projects** tab on the top right of the screen, then use the **Add project** button to search for, and add the project the team's project, making sure to grant **Write** access to the project

The screenshot displays the 'Delivery Management' interface. At the top, there is a navigation bar with several tabs: 'Discussions', 'Members 10', 'Teams 0', 'Repositories 1', 'Projects 1', and 'Settings'. The 'Projects 1' tab is currently selected and highlighted with an orange underline. Below the navigation bar, there is a green button labeled 'Add project'. Below this button, there is a light blue box containing the text '1 project in the Team-267 team'. At the bottom of the interface, there is a card for a project named 'Farmrail', which was 'updated 2 hours ago'. This card has a dropdown menu set to 'Write' and a red trash icon.

Discussions Members 10 Teams 0 Repositories 1 **Projects 1** Settings

Add project

1 project in the Team-267 team

Farmrail
updated 2 hours ago

Write

When all of this configuration is done, and it is time for the team to start innovating on their solution, follow the steps below to manage what needs to be done and assign tasks across the team. Always optimize to have no member of the team idle during a sprint.

Conducting online research for a feature or improving documentation **is as good a task as writing code**

1.

The team, supported by the TTL and mentor, needs to create a list of features for the solution being built

2.

Break down a given feature into tasks that can be done by one developer in a few days. Arrange the tasks into 3 groups in a logical sequential order

3.

Create tasks as issues on the team's repo and assigned to a milestone (Sprint-1, Sprint-2 or Sprint-3) which is a way to group them into a sequential execution flow

The key to nailing managing the project management setup is to first label your issues (tasks), then make sure they are attached to a milestone in the repo and to your project.

[REFACTOR] Add linting to produce API #43

[Edit](#)[New issue](#)[Open](#)

kmwamasali opened this issue 2 days ago · 0 comments · May be fixed by [#46](#)



kmwamasali commented 2 days ago

Member



Is your feature request related to a problem? Please describe.

Add linting and make necessary file adjustments

Describe the solution you'd like

Use PyLint

Additional context

branch: rf-produce-api-lint

Assignees



LadPaule

Labels



enhancement

Projects



Farmrail



In progress ▾

Milestone



Sprint-1



kmwamasali added the **enhancement** label 2 days ago



kmwamasali added this to the **Sprint-1** milestone 2 days ago

Create a **app.properties** file in the **root of all project repos** and add / replace its contents the default template contents. Edit the file and provide the required details for each entry. Each entry either has a single value (e.g **name** and **team.name**) or a comma separated list of values (e.g **code.languages** and **repos** - if there are multiple repos for the project)

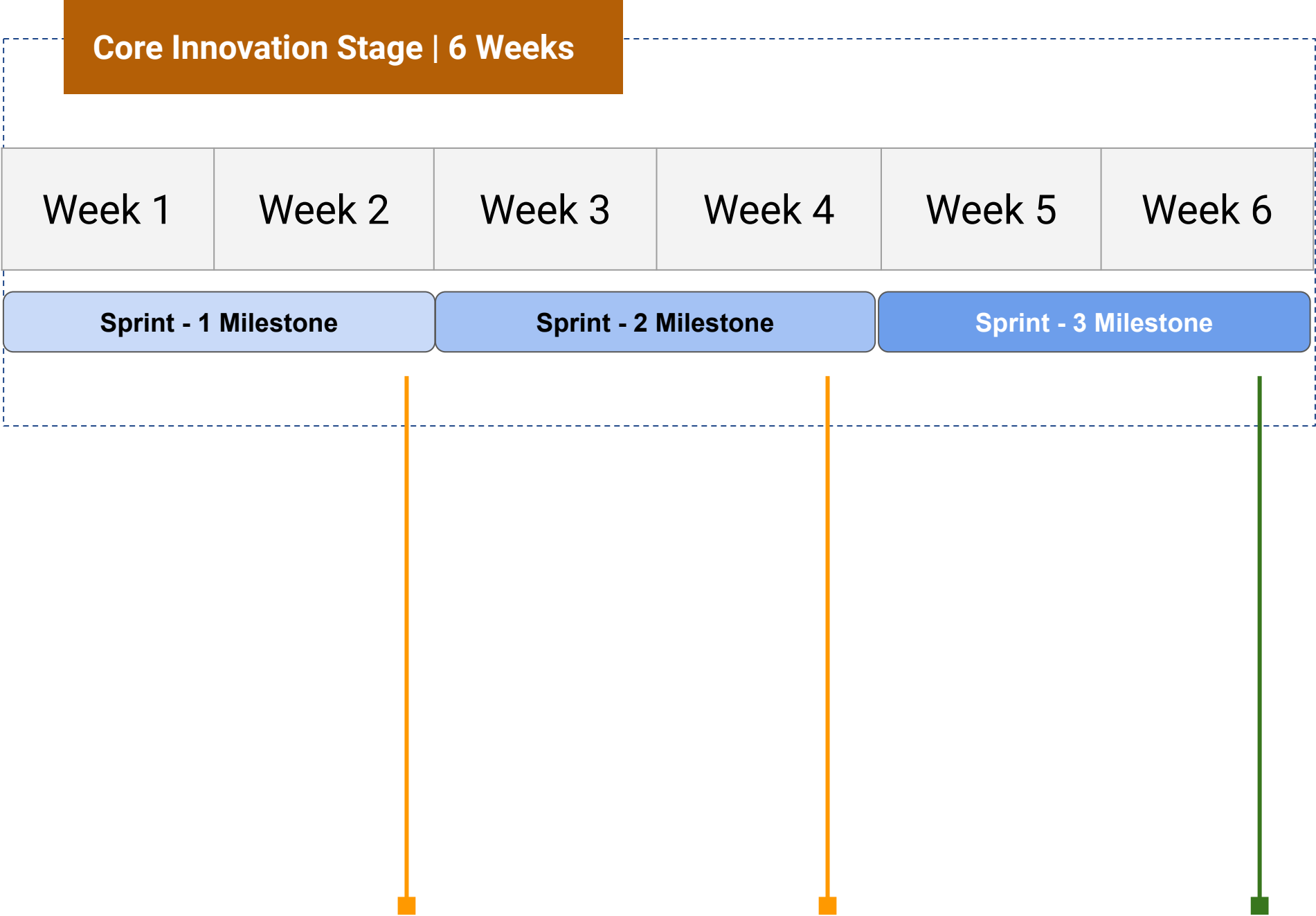
Use the **code.opensource.facebook** entry to provide a comma separated list of any Facebook open-source or platform API / tools the repo is using.

Finally, use **code.opensource.contributions** entry to provide a comma separated list of repos containing code your team has **created** and **open-sourced** as a result of innovating in the BuildforSDG program.

Create an issue for this on each reapo and assign it to someone

26 lines (23 sloc) | 401 Bytes

```
1  # the product
2  name:
3  sdgs:
4  repos:
5  app.urls:
6  codacy.urls:
7  app.forCOVID19: NO
8
9  # the codebase
10 code.languages:
11 code.frameworks:
12 code.opensource.facebook:
13 code.opensource.contributions:
14
15 # the team
16 team.name:
17 team.ttls:
18 team.mentors:
19 team.slackchannel:
20
21 # == DO NOT EDIT ==
22 # about the program
23 # =====
24 program.cohort:1
25 program.name:BuildforSDG
26 program.community:https://github.com/BuildFo
```



TTL + ~2 team members deploy **work-in-progress** features in the **develop branch** to a staging environment / server for demos, stakeholder feedback, and the rest of the world to see

TTL + ~2 team members deploy **bug-free** features in the **master branch** to a production environment / server for the final demo, stakeholder feedback, and the rest of the world to see

Though not mandatory, we strongly recommend that teams plan and ship their solution at least twice during the program. **The first release** can be a work-in-progress version from the **develop** branch. **The second** should be a **bug-free** production release

What Success Looks Like

Let's make it super clear what standards must be upheld and some metrics for measuring your success as an individual engineer, a team, and the product being built

You can't improve what is not measured nor can you measure what is not defined

⚙️ Performance & Evaluation

SCOPE

Evaluation is done at the **engineer** and **team** level.

Team level performance is a **computed weighted average** of relevant performance metrics across the team.

Impact is only measured at the team level and the team is solely responsible for defining and quantifying it

CATEGORIES

Execution How well is the team or engineer executing against the product's roadmap

Quality What is the quality of work output delivered by engineers in the team

Collaboration What the is level and quality of team work across the team

Impact A quantifiable measure of the positive effects this product can have on humanity.

Individual Engineer Metrics

Execution

1. **Contributions**

- (a) How many PRs has this engineer raised.
- (a) How much of this engineer's commit and PR messages communicate clearly and adhere to the [conventional commits](#) spec

Quality

1. **Code Quality Rating**

On a scale of **F (very poor)** to **A (very good)**, what is the Codacy quality certification of the code changes proposed by this engineer via PRs

Collaboration

1. **PRs Reviewed**

How many merged PRs did this engineer contribute to by providing feedback and reviews



Team Metrics

Execution

1. **Burndown** **>= 75%**

How likely is the team to deliver the product roadmap and meet the deadline

2. **Time To Merge**

As engineers work on tasks, what is the AVG time from their first commit till when the eventual PR is merged

Quality

1. **Code Quality Rating** **>= B**

On a scale of F to A, what is the code quality certification of the team repo.

2. **Test Coverage** **>= 70%**

How much aspects of the codebase is covered by unit tests that are passing

Collaboration

1. **Work Spread** **>= 65%**

What % of the team is actually doing the work

2. **PR Lead Time**

After engineers raise a PR, how long does it take to be reviewed, approved and merged

Appendices

Still In Search For Solutions / Product Ideas Or Need To Firm-up What You Have ?

See

<https://www.hackthegoals.be/>
<https://isbhacks2019.devpost.com/>

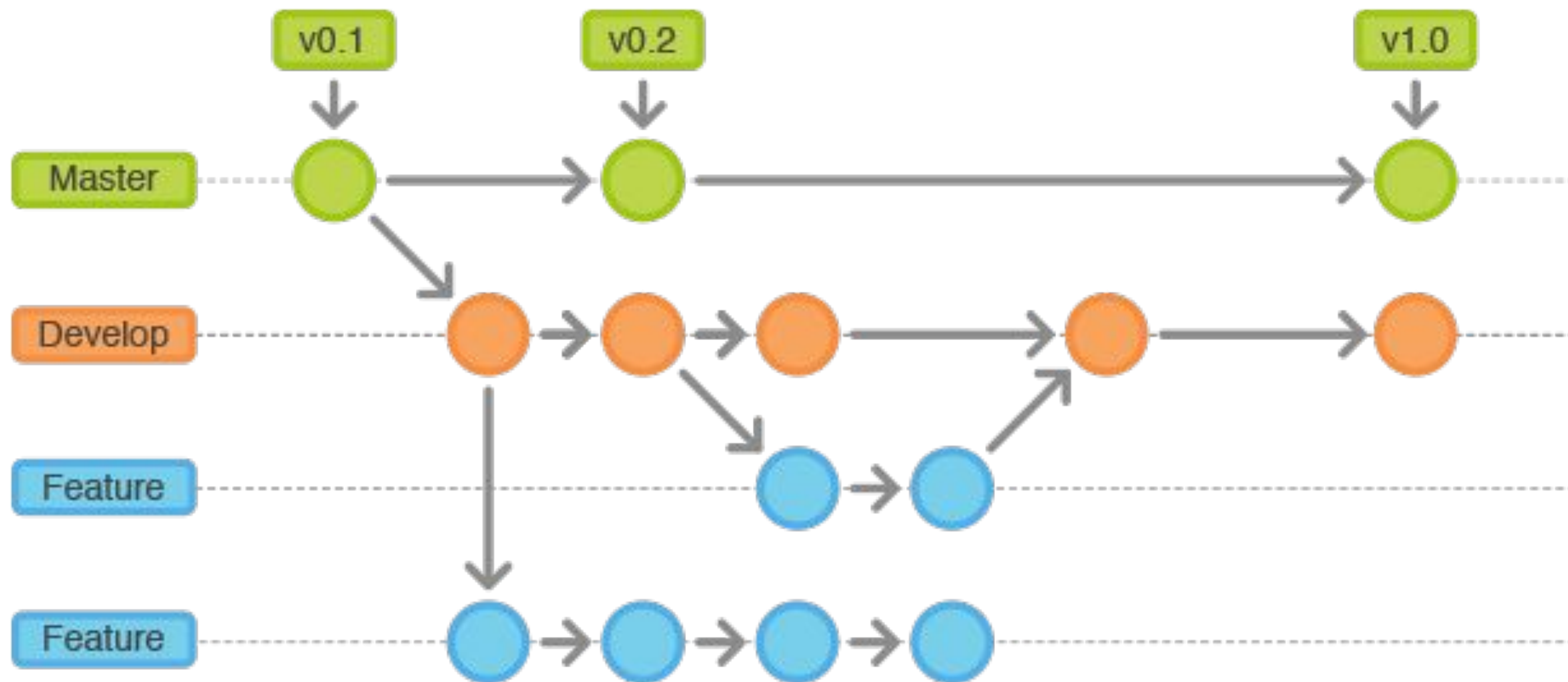
Standards To Uphold

Professionally, software is built and shipped with a collection of tools that play key roles in the continual delivery of quality products that meet stakeholder expectations.

To prepare for the market and succeed in this industry, engineers and team leads need to build mastery for this workflow.

Git Flow Workflow

Gitflow Workflow



Gitflow Workflow

1 Using Git & Github, work is done on **feature branches** which are created from the **develop** branch, **not** the **master** branch

2

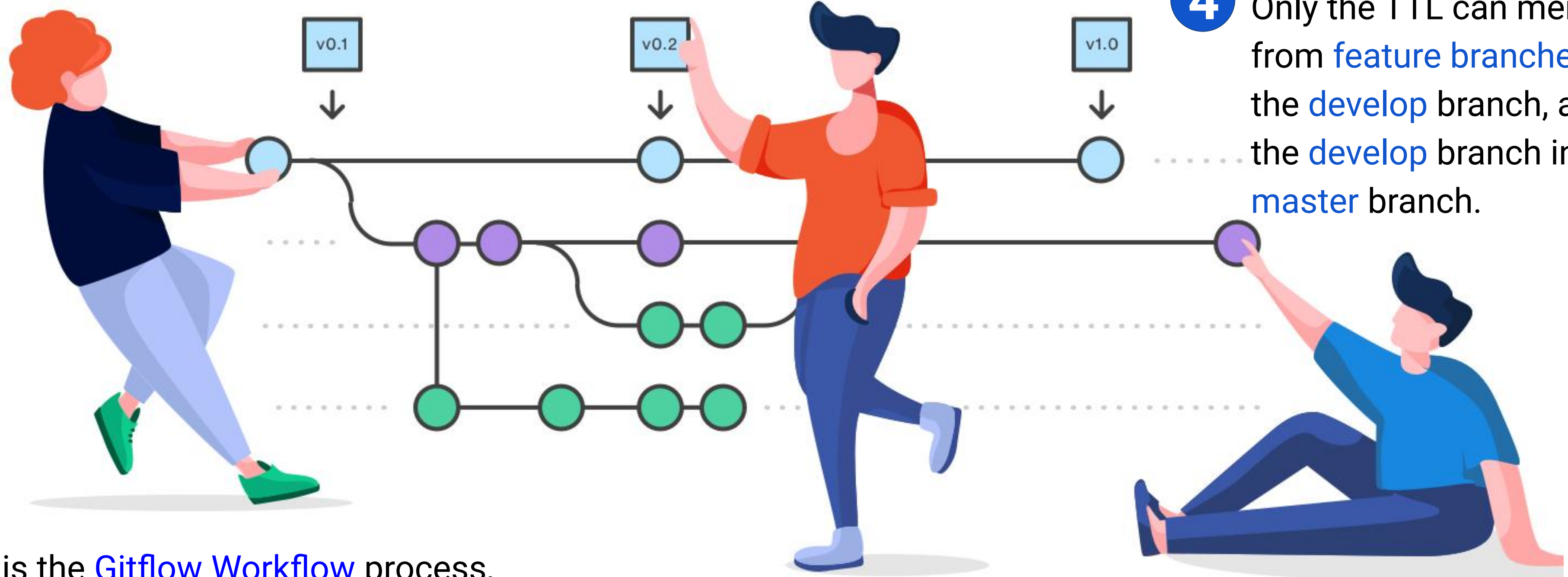
Commit messages should be semantic and adhere to the [conventional commits](#) spec

3

A minimum of 3 reviewers (2 team members + the TTL) are required to review & approve a PR before it can be merged back into the **develop** branch by the TTL

4

Only the TTL can merge code from **feature branches** into the **develop** branch, and from the **develop** branch into the **master** branch.



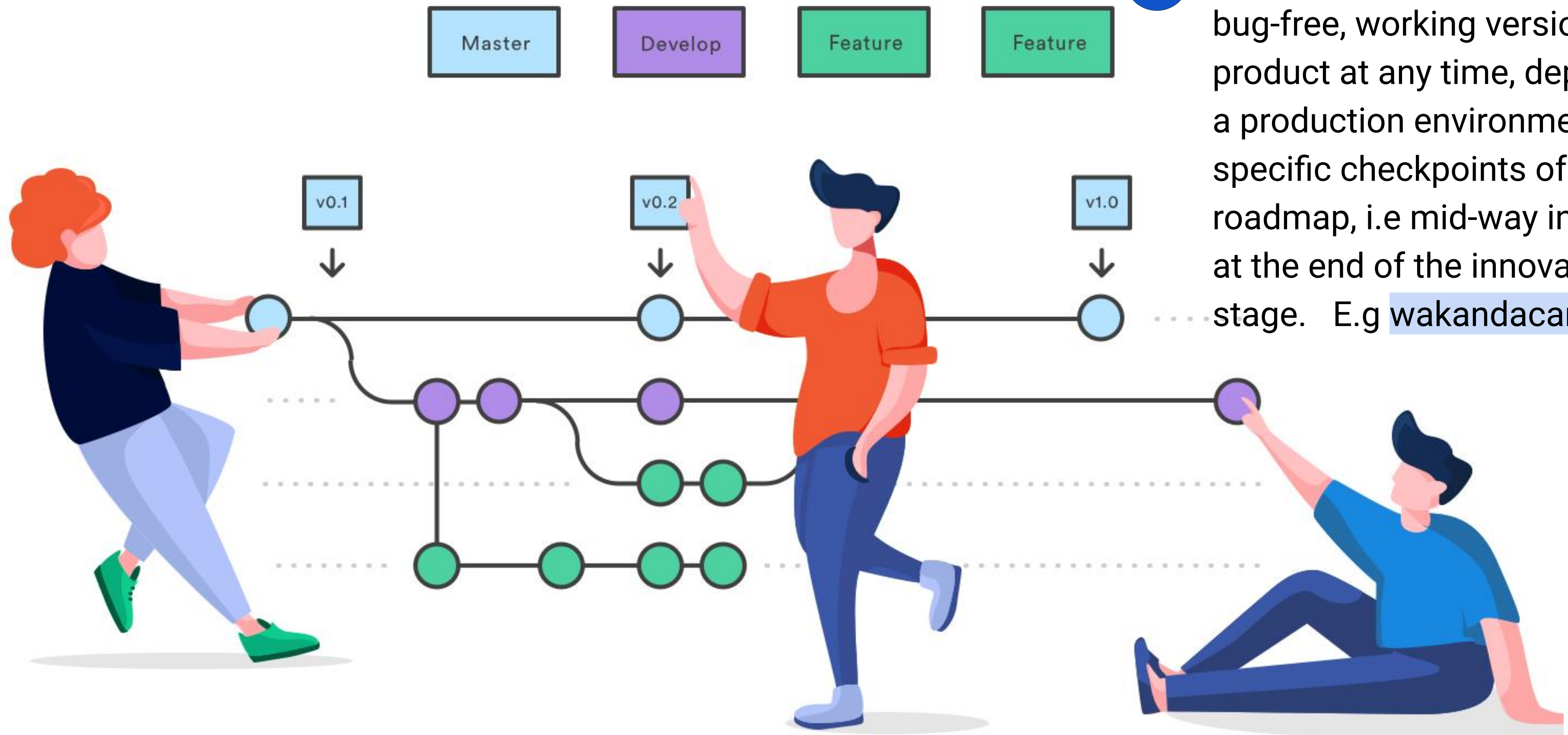
5

This is the [Gitflow Workflow](#) process, and forms the base of streamlined collaborative delivery

Gitflow Workflow

6 The `develop` branch represents a work-in-progress view of the product and is deployed to a staging server for end-of-sprint demos and stakeholder engagement. E.g `wakandacart.staging.app`

7 The `master` branch represents a bug-free, working version of the product at any time, deployed to a production environment at specific checkpoints of the roadmap, i.e mid-way into and at the end of the innovation stage. E.g `wakandacart.app`



Conventions

Commit Messages

The Conventional Commits specification is a lightweight convention for commit messages. It provides an easy set of rules for creating an explicit commit history and dovetails with SemVer, by describing the features, fixes, and breaking changes made in commit messages.

Format

```
<type>[optional scope]: <description>  
  
[optional body]  
  
[optional footer(s)]
```

Ideally, the commit message heading which contains the description, should not be more than 50 characters

Example

```
feat: allow provided user preferences to override default preferences  
  
This allows the preferences associated with a user account to override  
and customise the default app preference like theme, timezone e.t.c  
  
Fixes #025
```

Feature Branch Names

Branches created should be named using the following format

Format

```
{issue or story type prefix}-{dash separated word summary}
```

Issue or story type prefixes are one of the following:

ft => Feature

bg => Bug

ch => Chore

rf => Refactor

Examples

```
ft-facebook-auth
```

```
bg-facebook-auth-timeouts
```

Projects Management On GitHub

Project Management

Improve DX for RequestIdleCallback call #6

Open chalu opened this issue 22 hours ago · 0 comments

Issues are created for each feature, bug fix or chore needed to deliver the product roadmap. Open issues move into the TODO

Team projects are managed with a roadmap on GitHub Projects which automates workflow and moves assigned tasks from the project's TODO list all the way to DONE as the team works. DONE tasks are features, bug fixes or chores that have been tested, reviewed ready for production

Wakanda DOM
Updated 22 seconds ago

2

1 To do

Improve DX for RequestIdleCallback call
#6 opened by chalu
enhancement
Sprint - 1

0 In progress

3

0 Review in progress

0 Reviewer approved

4

4 Done

Improve DX for RequestAnimationFrame API calls
#3 opened by chalu
enhancement
Sprint - 1
2 linked pull requests

Improve Selector API DX
#1 opened by chalu
enhancement
Sprint - 1
1 linked pull request

feat: add adapter for querySelector function
#2 opened by chalu
enhancement
Sprint - 1

5

Sprint - 1

Due by May 01, 2020 80% complete

1 Open 4 Closed

Improve DX for RequestIdleCallback call enhancement
#6 opened 1 minute ago by chalu

Open Source Integrations

Open Source Integrations

Special mention and post-program support will be given to projects that integrate one or more Facebook open source or product platforms

This includes getting spotlighted by Facebook Developer Circles and getting your project featured on their official global GitHub Organisation

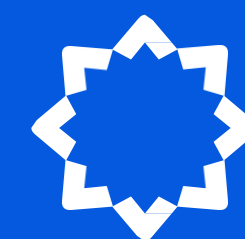
Easy Picks

Jest, React, GraphQL, Flow, Facebook Login

Criteria for projects to be spotlighted on the Facebook Developer Circles Github ORG

- Join a Facebook Developer Circle
- The project is maintained by more than one person.
- The project MUST follow the [Contributor Covenant](#) guidelines and the code of conduct stated in the document document
- The project implemented at least one Facebook open source project or Platform API
- Maintainers or the project have signed the [Facebook open source CLA](#)
- The project brings tangible impact to the local community

THANK YOU



Andela

Copyright 2020
Andela