

Software Engineering Freelancer Training

Collaborative Software Development

Google Drive Link

Introduction

We want to enable cross-functional collaboration, encourage software engineers to adopt multiple roles and facilitate lateral movement in engineering while enforcing the same common industry best practices across the whole engineering organization.



Getting started

For each customer

- Review customer presentation deck
- Meet with the customer's team
- Get rate card approval
- Get added to the WeChat group
- Get added to the <customer>@coderbunker.com
 Google Groups



Roles

Commonalities to all roles (<u>moxre information here</u>)

Engineers are engineers first, their title second.

Responsibilities

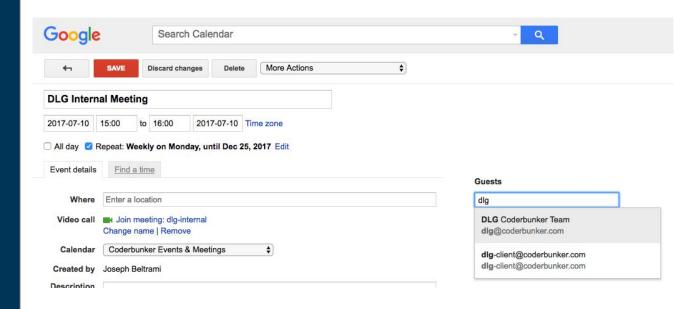
The first responsibility of engineers is to solve problems that are of value to the business.

All engineers are expected to apply industry best practices to their work, such as (but not limited to):

- elaborating designs and efficient algorithms
- adhering to good, internally defined code writing practices
- documenting processes and procedures
- asking for or performing peer reviews
- continually testing their work
- monitoring their systems
- managing sources and artifacts
- track issues
- understand the system as a whole



Calendar



Used shared calendar <u>Coderbunker Events & Meetings</u>



Generalized Setup

- Create parent project folder with name of customer in Opportunities
- Checkout repositories from Github
- Follow README.md instructions for each repositories
- Run system as-is (database -> backend -> frontend)



Issue Management

- Use single <u>Github Project Boards</u> per customer (organization)
- Issues are created per repository
- One issue should be assigned to a single person
- Don't close issue until it's been deployed and verified on production!



Development

- Assign issues to yourself only when you starting working on it
- Update description with <u>tasks list</u>
- Create branches per feature prefixed with issueXXX
- . Refer to issue # in every commit
- Refer to issue # in every pull request
- Rebase interactively if necessary



- Javascript
 - Airbnb style: https://github.com/airbnb/javascript

Code Style



Project Quality

Make sure every project has:

- Quality checks enforced by pre-commit hooks:
 - Automated linting
 - Javascript: eslint
 - Automated tests
 - Automated dependency checks
- Proper packages management
- Proper naming
- Continuous build/testing solution (Codeship)



Commits

 Small focused commits; don't put too many things in the same commit



Bug Reporting

- ACTUAL vs EXPECTED results
- test environment and software versions used (if relevant)
- steps to reproduce (which user, how to do it again?)
- business impact (how much money are we losing? are we losing customers?)
- Instructions, screenshots or link video to reproduce
- . do we have a workaround?

Writing issues

- One problem per issue
- Describe versions used and steps
- Don't hesitate to initial comment with "Edit comment"
- Use <u>markdown block of code</u>
 <u>backticks</u> (```code```) to preserve
 console output or code formatting
- For very long error output, create and link to a gist (https://gist.github.com)

#CoderBunker

(https://gist.github.com)

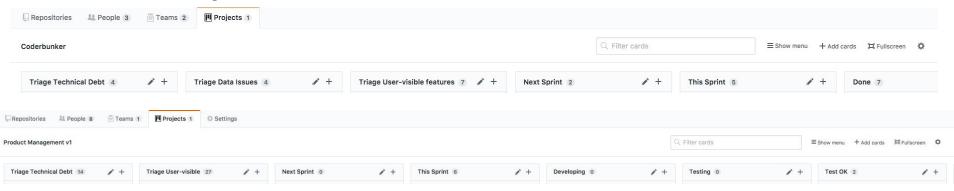
Issues Organization

- Break down new features into more achievable and focused tasks so you can move them to done
- Be specific with clear completion criteria
- . Show progress, by moving issues to Done quickly in every sprint

#CoderBunker

 Postpone longer term work in future issues

Github Projects



- Project board, usually Organization-wide and named Coderbunker
- Adapt the board as needed, but typically new issues should come in the Triage.
- Product Manager should sort priorities in each column from most important to least important Mix of technical debt and user-visible features in every sprint
- Sprint should target a week (this may take longer for first few sprints)
- There may be a testing stage

Time tracking

- Every work should be associated to an issue
- Fill in your timesheet every 2-3 hours of work at least with reference of issue worked on



Management

Establish "OKRs"Objectives and Key Results

Key Results should be MEASURABLE

Should be discussed with customer



Git usage and configuration ~/.gitconfig

```
[user]
                                      Setup your name and email!
      name = Ricky Ng-Adam
                                      Should match one of your email in
      email = rngadam@gmail.com
                                      Github
[alias]
      st = status
      ci = commit
      co = checkout
                                      Standard aliases
      br = branch
      ss = status -sb
      II = log --oneline --decorate
      nr = name-rev --name-only
[color]
                                      Coloring
      branch = auto
      diff = auto
      interactive = auto
      status = auto
[merge]
      tool = /dev/null
                                      rebase feature branches always
[branch]
      autosetuprebase = always
                                      git config --global
[push]
                                      branch.autosetuprebase always
      default = tracking
```

#CoderBunker

Pull Requests

Code Review process

- Create local branches with your modifications
 - Lowercase naming
 - Prefix with related issue
 - issue3-poc-bug-fixing
- Create draft pull requests
 - Assign to peer developer on project as reviewer
 - Make sure description contains [WIP] DO NOT MERGE
 - Reference related issue to benefit from auto-linking
 - issue #3
 - Address reviewers comments
- Create final pull requests
 - Address reviewers comments or postpone to related issues for bigger changes
 - Rebase and cleanup commits, reference issue in commit
 - Make sure modifications pass all tests locally
 - Push code to Github
 - Reviewer is the one that should merge



Deployment To production

Deployments in production for our customers should:

- have a documented lists of steps in a Github issues
- have an event on the Coderbunker calendar
 - Leads and devops as guests
 - Calendar event should have link to the issue.
- Use <u>twelve-factor</u> approach to configuration (.env file)
 - NodeJS: https://github.com/motdotla/dotenv



DevOps Toolset

Name	Description
DockerHub	Continuous Delivery pipeline
Docker	Environments, development tools
Github	Repo service provider
Icinga	Server monitoring

