



# #CoderBunker

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.**



#CoderBunker

# 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



#CoderBunker

# Roles

## Commonalities to all roles ([moxre information here](#))

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



#CoderBunker

# Calendar

Google Search Calendar

← SAVE Discard changes Delete More Actions

**DLG Internal Meeting**

2017-07-10 15:00 to 16:00 2017-07-10 [Time zone](#)

☐ All day ☒ Repeat: **Weekly on Monday, until Dec 25, 2017** [Edit](#)

Event details [Find a time](#)

Where

Video call [Join meeting: dlg-internal](#)  
[Change name](#) | [Remove](#)

Calendar

Created by Joseph Beltrami

Description

**Guests**

DLG Coderbunker Team  
dlg@coderbunker.com

dlg-client@coderbunker.com  
dlg-client@coderbunker.com

Used shared calendar [Coderbunker Events & Meetings](#)

# 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)**



#CoderBunker

# Issue Management

- Use single [Github Project Boards](#) 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 [tasks list](#)**
- **Create branches per feature prefixed with issueXXX**
- **Refer to issue # in every commit**
- **Refer to issue # in every pull request**
- **Rebase interactively if necessary**





# Code Style

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



#CoderBunker

# 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)



#CoderBunker

# Commits

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



#CoderBunker

# 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?**



#CoderBunker

## Writing issues

- One problem per issue
- Describe versions used and steps
- Don't hesitate to initial comment with "Edit comment"
- Use markdown block of code backticks (````code````) to preserve console output or code formatting
- For very long error output, create and link to a gist (<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**
- **Postpone longer term work in future issues**



# Github Projects

The image displays two examples of GitHub Project boards. The top board, titled 'Coderbunker', has columns for 'Triage Technical Debt' (4 items), 'Triage Data Issues' (4 items), 'Triage User-visible features' (7 items), 'Next Sprint' (2 items), 'This Sprint' (5 items), and 'Done' (7 items). The bottom board, titled 'Product Management v1', has columns for 'Triage Technical Debt' (14 items), 'Triage User-visible' (27 items), 'Next Sprint' (0 items), 'This Sprint' (6 items), 'Developing' (0 items), 'Testing' (0 items), and 'Test OK' (2 items). Both boards show navigation tabs (Repositories, People, Teams, Projects, Settings), a search bar, and utility icons (Show menu, Add cards, Fullscreen, Settings).

- 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**



#CoderBunker



# Management

- **Establish "OKRs"**  
**Objectives and Key Results**

**Key Results should be  
MEASURABLE**

**Should be discussed with  
customer**



**#CoderBunker**

# Git usage and configuration

## ~/.gitconfig

```
[user]
  name = Ricky Ng-Adam
  email = rngadam@gmail.com
```

```
[alias]
  st = status
  ci = commit
  co = checkout
  br = branch
  ss = status -sb
  ll = log --oneline --decorate
  nr = name-rev --name-only
```

```
[color]
  branch = auto
  diff = auto
  interactive = auto
  status = auto
```

```
[merge]
  tool = /dev/null
```

```
[branch]
  autosetuprebase = always
```

```
[push]
  default = tracking
```

Setup your name and email!  
Should match one of your email in Github

Standard aliases

Coloring

rebase feature branches always

```
git config --global
branch.autosetuprebase always
```



#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



#CoderBunker

# 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 [twelve-factor](#) approach to configuration (.env file)
  - NodeJS: <https://github.com/motdotla/dotenv>



#CoderBunker

# DevOps Toolset

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



#CoderBunker