Travis CI

You’ve got 99 problems, and testing is one of ’em!

Prepared by:
Altynbek Orumbayev
1. Introduction to Continuous Integration
2. Travis CI concepts
3. Quick Setup Guide
4. Why Travis?
5. Conclusion and overview
Continuous Integration (CI) is a software development practice that is based on a frequent integration of the code into a shared repository. Each check-in is then verified by an automated build.
Introduction to Continuous Integration

- **VCS**: trigger CI build
- **CI SERVER**: run tests → success → build
- **APP SERVER**: deploy
Travis CI Concepts

An automated process that clones your repository into a virtual environment and then carries out a series of phases.

A group of jobs. For example, a build might have two jobs, each of which tests a project with a different version of a programming language.

The sequential steps of a job. For example, the install phase, comes before the script phase, which comes before the optional deploy phase.

A group of jobs that run in parallel as part of sequential build process composed of multiple stages.
The build is considered *broken* when one or more of its jobs completes with a state that is not *passed*.

Travis CI offers a few different infrastructure environments, so you can select the setup that suits your project best:

- **Errored**
- **Failed**
- **Canceled**
- **Container-based**
- **Sudo-enabled**
- **OS X**
1. Signup using github into Travis on travis-ci.org
2. Enable the repository in which you want to integrate C
3. Add .travis.yml to your repository
4. Trigger your first build with `git push` :-}
Travis CI Concepts

.yaml sample for Java

```yaml
.before_cache:
  - rm -f $HOME/.gradle/caches/modules-2/modules-2.lock
  - rm -fr $HOME/.gradle/caches/*/plugin-resolution/
.cache:
  directories:
    - $HOME/.gradle/caches/
    - $HOME/.gradle/wrapper/
```

.yaml sample for Swift

```yaml
language: objective-c
xcode_project: MyNewProject.xcodeproj # path to your xcodeproj folder
xcode_scheme: MyNewProjectTests
```
Why Travis?
Why Travis?

Pros:
- Price (it’s free)
- Customization
- Plugins system
- Full control of the system

Cons:
- Dedicated server (or several servers) are required. That results in additional expenses. For the server itself, DevOps, etc…
- Time needed for configuration / customization

Pros:
- Build matrix out of the box
- Fast start
- Lightweight YAML config
- Free plan for open-sourced projects
- No dedicated server required

Cons:
- Price is higher compared to CircleCI, no free enterprise plan
- Customization (for some stuff you’ll need 3rd parties)
Conclusion and overview

Cloud-based

Docker support

Lighweight yaml and easy setup

Open source

Build matrix support
THANK YOU FOR YOUR ATTENTION!