dlab

Release 1.0.0

Aug 14, 2019

Contents

CHAPTER 1

What is DLAB?

1.1 How to contribute

There are lots of ways to contribute to the project. People with different expertise can help to improve the web content, the documentation, the code, and the tests. Even this README file is a result of collaboration of multiple people.

infrastructure-provisioning : Much of the logic of the project is in this module. Here's your chance to get elbow-deep into machine learning, Tensorflow, R, Jupiter, Apache Zeppelin, containers, and more.

integration-tests : A software project is as good as its tests are. Following this simple idea, we are trying to cover the integration capabilities with automated tests. If you're passionate about the quality - please consider chiming in.

maven : The build system - the heart of any project. If you think Maven is too old-fashioned, create a PR and move us to Gradle!

doc : Have you seen a project that doesn't need to improve its documentation?!

website : We are using GH pages to build and manage the site's content. If you're interested in making it better, check-out *gh-pages* branch and dig in. If you are not familiar with the Github Pages - check it out, it's pretty simple yet powerful!

giving feedback : Tell us how you use DLab, what was great and what was not so much. Also, what are you expecting from it and what would you like to see in the future? Opening an issue will grab our attention. Seriously, this is the great way to contribute!

1.1.1 Roles

Much like projects in ASF, DLab recognizes a few roles. Unlike ASF's projects, our structure is a way simpler. There are only two types:

A Contributor is a user who contributes to a project in the form of code or documentation. Developers take extra steps to participate in a project,are active on the developer forums, participate in discussions, provide PRs (patches), documentation, suggestions, and criticism. Contributors are also known as developers.

A Committer is a developer that was given write access to the code repository. Not needing to depend on other people to commit their patches, they are actually making short-term decisions for the project. By submitting your code or other content to the project via PR or a patch, a Committer agrees to transfer the contribution rights to the Project. From time to time, the project's committership will go through the list of contributions and make a decision to

invite new developers to become a project committer.

1.1.2 RTC model

DLab supports Review-Then-Commit model of development. The following rules are used in the RTC process:

- a developer should seek peer-review and/or feedback from other developers through the PR mechanism (aka code review).
- a developer should make a reasonable effort to test the changes before submitting a PR for review.
- any non-document PR is required to be opened for at least 24 hours for community feedback before it gets committed unless it has an explicit +1 from a committer
- any non-document PR needs to address all the comment and reach consensus before it gets committed without a +1 from other committers
- a committer can commit documentation patches without explicit review process. However, we encourage you to seek the feedback.

1.1.3 Branch Naming Conventions

Example

Available packages:

Name	Description
aws	dlab_aws package
azure	dlab_azure package
ci	ci/cd related files
core	dlab_core package
doc	documentation related files
gcp	dlab_gcp package

Available sections:

Name	Description
controller	controller changes
license	license information
logger	logger implementation
readme	readme information
setup	setup helper change
travis	travis related changes
usecase	use case changes

1.1.4 PR Naming Conventions

Example

The description should contain the following headings and the related content:

NOTE: To close an issue numbered 123, you must use the phrase "Closes #123" or "Closes: #123" in your pull request description or commit message. Once the branch is merged into the default branch, the issue will close.

NOTE: We expect to have one commit per pull request.

NOTE: After creation PR needs to be labeled regards branch namespace.

1.2 Plugins

1.2.1 What for

1.2.2 Example

`dlab_deployment.registry`

Plugin public name.

Module Contents

`PLUGIN_PREFIX = deployment`

`bootstrap``()`
    Bootstrap Deployment Plugin

`dlab_aws.registry`

Plugin public name.

Module Contents

`PLUGIN_PREFIX = aws`

`bootstrap``()`
    Bootstrap AWS Plugin

`dlab_gcp.registry`

Plugin public name.

Module Contents

`PLUGIN_PREFIX = gcp`

`bootstrap``()`
    Bootstrap GCP Plugin

dlab_azure.registry

Plugin public name.

Module Contents

PLUGIN_PREFIX = azure

bootstrap()
    Bootstrap Azure Plugin

1.3 Quick Start

1.4 Installation

1.5 ommand Line Interface

command line interface that allows many types of operation on a DAG, starting services, and supporting development and testing.

Important: Should be updated with adding Argparse

1.6 Changelog

Python Module Index

Index