

2019 Website Redesign Plan

Sean McGinnis
sean.mcginnis@huawei.com

Table of Contents

Overview **3**
Scope 3
Overall Design Goals **4**
Website Audience 4
Top Level Navigation 4
Site Redesign **5**
Default/Home Page 6
Use Case View 6
Displaying Results *6*
Testing Results 7
Filtering 8
Considerations **8**

Overview

OpenLab is a virtual open source testing lab. Its mission is to be a place where developers, testers, and consumers of open source projects go to discover tested combinations of projects that can be used together to address various use cases.

The goal is to make the openlabtesting.org site an easy to use place for these types of users to find the information they need and discover new projects that they can use to meet their needs.

Scope

This is a working design document for the redesign of the openlabtesting.org web site. It is not intended to be an exhaustive and exact specification for that redesign, but rather a tool to use to drive that design to ensure important elements and ideas are taken into consideration as it is done. It should not be expected that this is a complete guideline for a final design implementation.

Overall Design Goals

The OpenLab site should be visually appealing and as simple as possible, while having enough detail and functionality to be useful. Ease of use should be one of the top priorities.

OpenLab is run by a community of open source contributors. As such, making updates to the data displayed by the web site needs to be easy for different people to contribute to. Details on use cases, open source projects, and test results need to be dynamic through a combination of configuration changes made to source controlled repo and from data pulled in from a database of testing results.

Website Audience

There are different target audiences for the OpenLab web site:

1. Project Maintainer
   This user is looking to test their open source project as part of an overall solution stack. They need to be able to find information to easily add their project to the set of projects being tested, view testing results to find out where there are issues, and navigate to test logs to help triage failures and find out more information about test output.

2. Business Analyst/Architect
   This user is looking for information about projects or technology stacks to know what has been tested and help them make technology choices to architect solutions within their company. They need simple views to help them find the relevant projects that can help them meet their needs and discover which projects have been validated to work together.

3. Executives, Journalists, and Industry Experts
   This type of user does not need low level detail, but wants to know about projects or technology stacks being used and their status. High level summaries of projects and their solution space should help guide them to what they need to know.

4. OpenLab Contributors
   This group includes developers and system administrators that would like to participate in running OpenLab by contributing testing and test resource configuration, as well as cloud service providers and others that wish to support the OpenLab mission by providing virtual machines and/or access to hardware on which to run tests.

Top Level Navigation

The navigation of the site needs to be intuitive to allow the different types of audiences to quickly get to the area of interest to them. Those interested in just whether the projects they

are considering will work together should not need to sort through a lot of technical detail and detailed testing output to find what they need. Similarly, project maintainers interested in the status of their tests should not need to wade through high level marketing and generic material before they can drill down into the technical details they need to see.

Site Redesign

Since we have an existing site, it would also be good to keep some of the top level navigation somewhat consistent. The existing top level navigation sections are HOME, DOCS, COMMUNITY, EXPLORE, ABOUT US, and CI DASHBOARD.

The DOCS and CI DASHBOARD sections currently link off to technical details on how to get started testing and monitoring the execution of tests real time, respectively. These are very important things for the project maintainers, but they could be confusing to non-maintainers that just want to find out about OpenLab and tested solutions. They are important and necessary resources, but should be moved to be accessed under a technical focused section of the site.

The COMMUNITY section has some information about the people involved in the OpenLab community with links and details for communication channels: how to sign up for the mailing list, where to find OpenLab on IRC chat, and Twitter account details. These communication details would be good to have on the home page to make it quick and easy for new people coming in to know where to go to get more details and where to get involved.

Giving community members recognition is nice, but we can find other ways to recognize contributions. If things go as we hope, it will also not scale very well to add everyone involved to this page.

EXPLORE is probably the most useful section right now for those interested in what is happening in OpenLab. It currently displays resource usage and gives a very high level list of testing status. It is all status information at this point that requires someone from the team to look at test results and update status.

As we grow, we need the presentation of tested projects and the results to all be dynamically pulled in without the need for user intervention. We will also want to show things in a way that associates the projects with the larger use cases that can be addressed by its usage and have a way to click and drill into more technical details if the user needs to find out more.

It is also not much interest to users of the site how much infrastructure capacity we are using and from which providers. This is really operational details where we need a way to monitor our capacity and make sure we are efficiently and appropriately using infrastructure, but not something a user of the site will need to see.

ABOUT US is a very similar subset of the COMMUNITY page. The current governance members are shown as are the logos of the companies that provide infrastructure in order for OpenLab to operate. We want users to be able to find out these things, but they are not important enough to warrant their own top level section of the site.

Infrastructure providers can be recognized on the home page, probably near the bottom so users of the site don't need to scroll through them to get to what they need. The governance members are likely better kept in a governance Git repo and/or on a separate community focused governance web site.

Default/Home Page

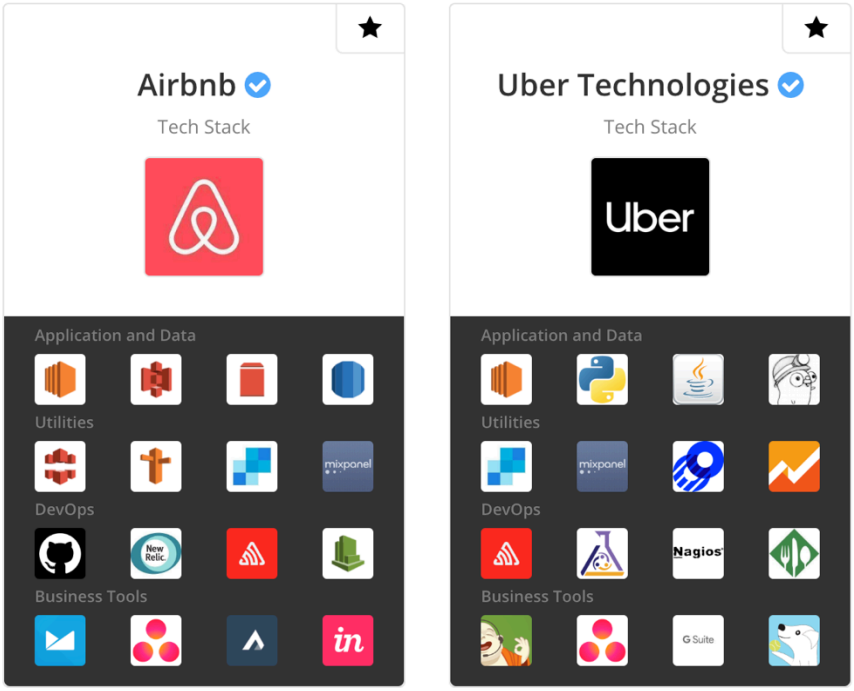
The site's home page should quickly let new visitors learn about what OpenLab is, then provide quick and intuitive routing off to the other areas of the site to let them get to the type of information they are looking for without the need for lengthy exploration of the site. It also needs to be simple and visually appealing while still having content and pointers that are relevant to the new viewer.

Use Case View

We would like the framing of OpenLab activities to be around the use cases they are supporting. Testing of projects should be in support of validating whether projects can be used together to support a higher overall goal.

Displaying Results

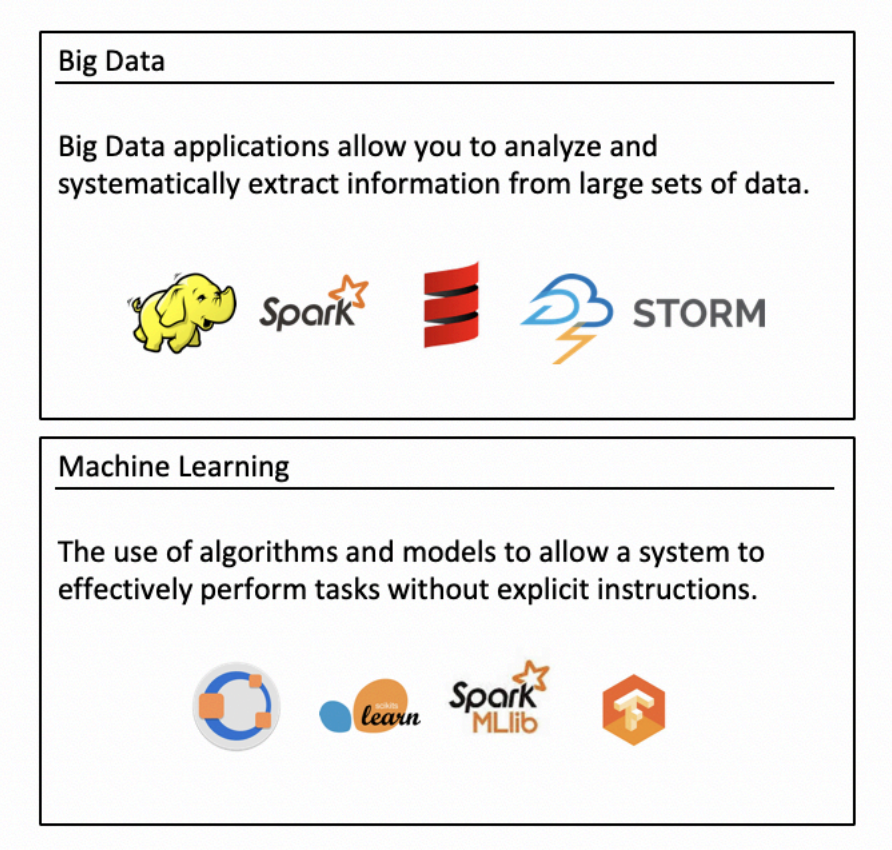
One site similar to what we are trying to do is https://stackshare.io/. StackShare.io has a very nice company-based view to show what projects are in use at them:



This makes it visually easy to see what projects they are using. From there, users are able to click on the company to drill down into more details. The projects are also grouped by their type (Application and Data, Utilities, DevOps, Business Tools, etc).

The OpenLab web site should borrow from this to present different technology stacks from a more use case driven view. So rather than having groupings based on company like StackShare.io, we would have things like “Cloud Provider”, “AI/Machine Learning”, “Big Data”, etc. Users of the OpenLab site would then be able to quickly identify the groups of open source projects that can be used based on the problem they are trying to solve.

We can make this more useful by showing both tested and untested projects. Tested projects can have a badge or some sort of visual marking that is a rollup of the test result status of these projects (for example, a green outline if all tests that include a given project are passing, yellow if there is a certain threshold of failing, red if all tests are failing).



Clicking on a specific panel can then go to a more detailed page for that use case. We could have an expanded description of the general use case, links to other resources, and then a listing of the combinations being tested in OpenLab supporting the different technology stacks that can be used.

Testing Results

Expanding on the use case view above, getting into the details of the use case should let the user see more information about the results of OpenLab testing. Potentially we could have lines linking the projects together that have been specifically tested to validate interaction between them. Another option would be to have multiple rows under the overall use case description that shows each of these tested combinations so they could quickly identify the combinations of interest to them.

Clicking on a project or combination should bring them to a detailed view of the test results. This should be a view that shows summary information, such as the overall pass/fail percentage of tests, and also give some more detail such as versions tested and other relevant details.

Continuing into lower level details, the user should be able to click through on these and get to individual test results. Clicking on individual tests should drill down into the logs and test artifacts from the latest test run. It should be easy for someone to go from a high level summary status view all the way down to looking through test logs to start to identify issues. Some users of the site will only want to see that high level view. Others will be interested in the technical detail and want to drill into the level of detail that best fits their experience and interest.

Filtering

Eventually the site will contain a lot of different use cases and many associated project combinations and test results. The user of the site needs a way to filter down to specific projects that they are considering to help them identify relevant results.

We want to make OpenLab any easy reference resource for someone architecting a new solution. One way to think of this is taking Google as an example. Google does not have the goal to try to get people to spend more time exploring the results of a Google search. That would be frustrating for the end user and make them less likely to use the service. They spend a lot of time and effort trying to make it so a search gets the desired results the first time around and the user goes on to what they were trying to find.

For OpenLab, are goal should not be to try to drive time exploring all aspects of the site. We should make it as easy as possible for these architects to find the relevant data they need so they can then get back to the problem they were trying to solve.

The usefulness of the OpenLab site should be less about the amount of time someone spends there and more about the frequency that they return to find out more information.

Considerations

The use cases, associated projects, and testing status should all be dynamic as much as possible. As OpenLab grows, we will need the ability to easily update details as new test cases are added or removed.

The Use Case and summary views should be driven off of available data. Ideally, an OpenLab community member can propose updates to data files that define these use cases or test case associations in a Git repository. Merging changes to the repo will then cause the OpenLab web site to reflect those updates.

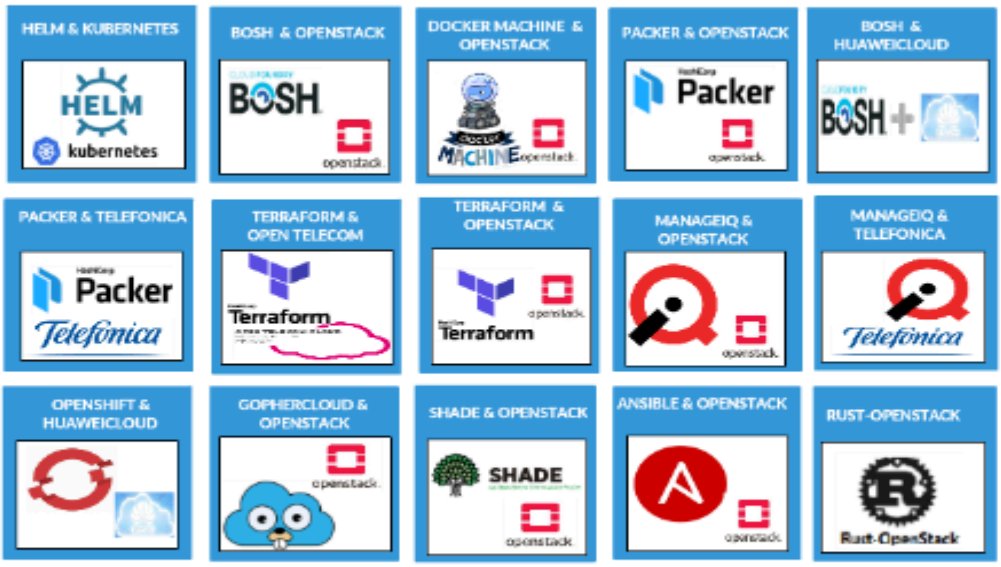
This can be either done by the site pulling in those settings directly from the git repository, or having the merge of a change in the git repository trigger an event that would then push out an update of those changes to a database somewhere that is then used by the site.

Updating dynamic information such as test results and associations between use cases and tested project combinations must not require a manual edit of the web site source to be reflected on the live web site.

Plan B (We decide to use Plan B)
I draw a picture below to group openlab jobs based on app+environment, and I think it's similar thought of this redesin doc. There are three groups: Cloud(and now most of openstack jobs belong to this group), AI and Big Data. Each icon includes one or several jobs, like 'HELM & KUBERNETES', when user click this icon, we will show two jobs information of helm, "helm-integration-test-kubeadm-k8s-v1.12.7" and "helm-integration-test-kubeadm-k8s-v1.14.0", including the helm version, kubernetes version, the build status and other information we want to show.

Cloud

Cloud computing is the delivery of computing services—servers, storage, databases, networking, software, analytics, intelligence and more



Big Data

Big Data applications allow you to analyze and systematically extract information from large sets of data



Artificial Intelligence

Artificial intelligence (AI) is the simulation of human intelligence processes by machines

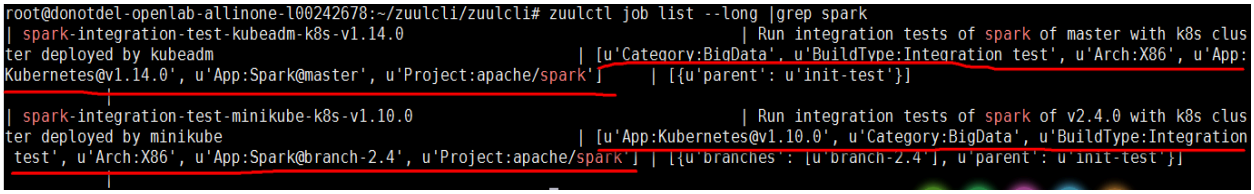


https://docs.google.com/document/d/1BOTSCIS0T3iOwPcjLwSmV-_U9CbXYorS6bnUO2gaouw/edit

1. We set metadata of job in job.tags, and the tags definition is a list of string,see https://zuul-ci.org/docs/zuul/user/config.html#attr-job.tags
2. We will use 'key:value' format for each string to extend the tags attributes of each job, the keys are special, now we have six keys:

```
- job:
    name: spark-integration-test-kubeadm-k8s-v1.14.0
    tags:
      - Category:BigData
      - Project:apache/spark
      - Application:Spark@master
      - Application:Kubernetes@v1.14.0
      - Application:Docker-ce@default
      - Arch:x86_64 or aarch64
      - OS:Ubuntu-xenial
      - BuildType:Integration test
```

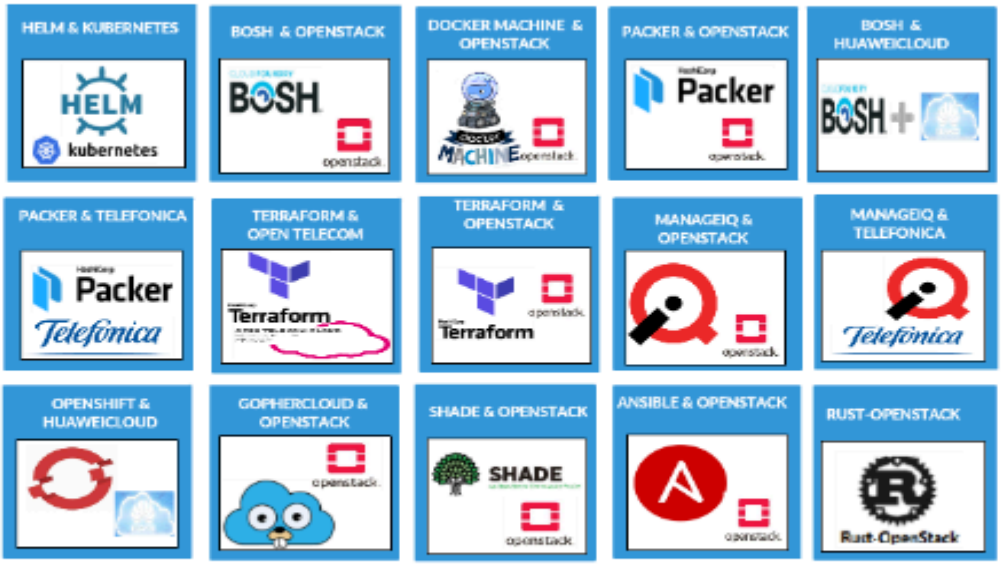
   And more details:
   - Category: Now it includes three values: Cloud, BigData and AI, and the jobs are grouped based on this attribute
   - Project: It indicates the name of the project which to run the job
   - Application: It indicates all software/runtime that the job using. The format of 'Application' value should be 'AppName@AppVersion', and maybe there are several softwares for one job, this key can be extended if needed. The 'AppName' could be OpenStack, Kubernetes, Spark, Helm, Terraform, Python, Java, Ruby and so on. The 'AppVersion' can be master, branch-2.4, branch-2.7, v1.14.0, stable/stein, stable/rocky and so on
   - Arch: the architecture which the job running on, we have two types of this attribute now: x86_64 and aarch64(and they come from 'uname -i')
   - OS: the operating system which the job running on, it cloud be ubuntu-xenial or ubuntu-trusty
   - BuildType: the build type of the job, Integration test, Unit test and Compile
3. Calling Zuul API 'job list' to get all jobs of OpenLab, and just care about the jobs which has tags(tags is [] if we don't set the tags for job). Group these jobs by 'Category' first, then group the jobs to software stack by project based on the tag item 'Project' for each category, we can use detail API of project to get all of the jobs in the project, Here we take spark jobs as example, according to the tags of the jobs, the two jobs of spark are grouped into 'BigData' category, and the project is 'apache/spark', see all the jobs of the project apache/spark: http://status.openlabtesting.org/project/apache/spark



See below diagram, there is a stack ‘SPARK & KUBERNETES’ in group ‘BigData’, and the logo of all softwares(all ‘AppName’ tags for a project jobs) should be shown:

Cloud

Cloud computing is the delivery of computing services—servers, storage, databases, networking, software, analytics, intelligence and more



Big Data

Big Data applications allow you to analyze and systematically extract information from large sets of data



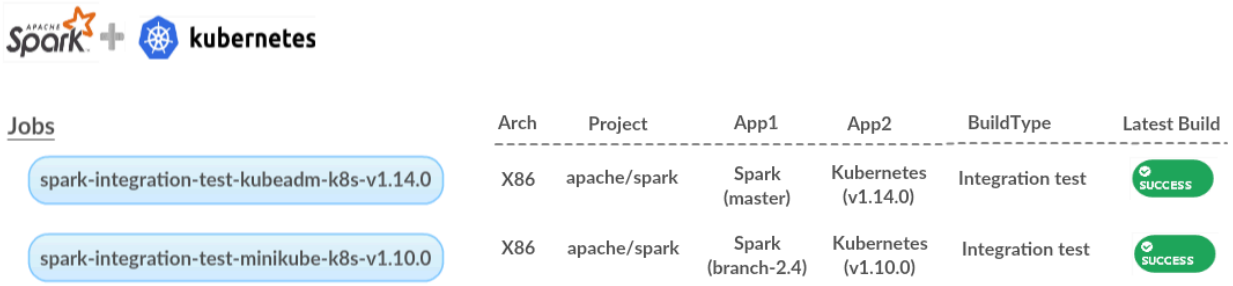
Artificial Intelligence

Artificial intelligence (AI) is the simulation of human intelligence processes by machines



4. Double click the icon, we will enter a new page to show all jobs of the project. And the page includes the jobs' name, arch, build type, project and all apps info(including name and version). Also the page will show the latest build result of the job, the status should be one of : https://github.com/openstack-infra/zuul/blob/master/doc/source/user/jobs.rst#build-status

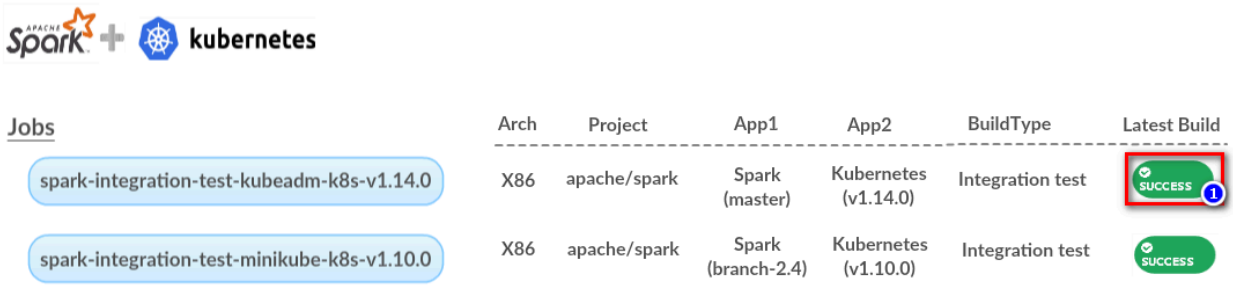
   Notes: all of these info comes from the job tags.



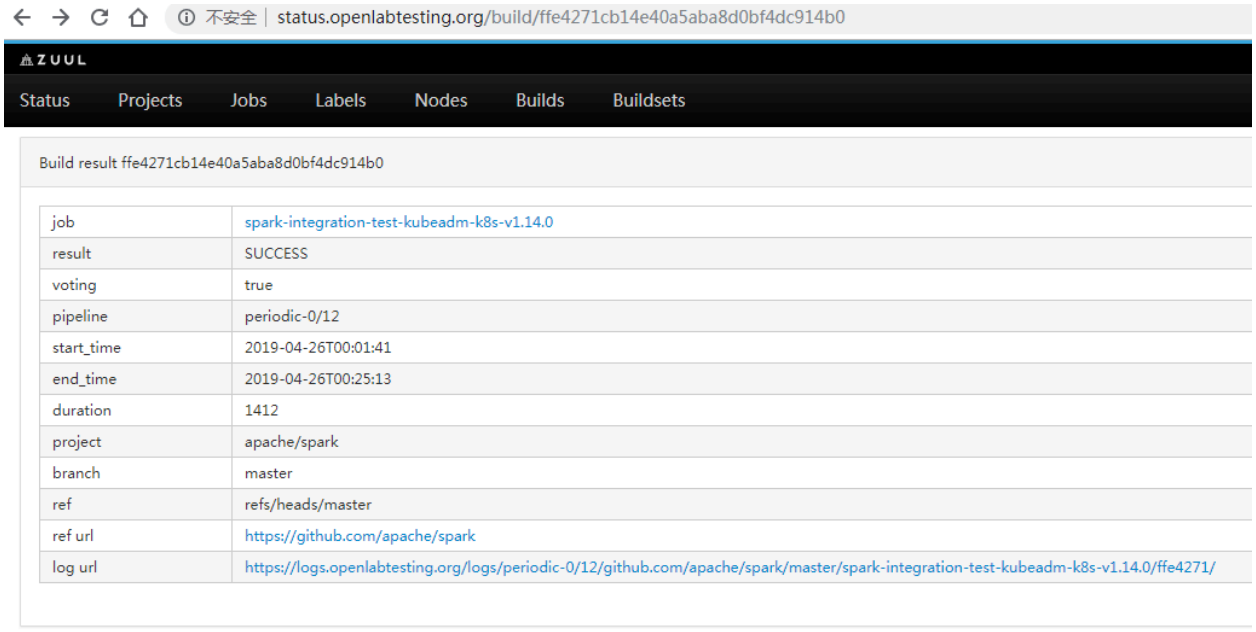
Notes: You can get the latest build result by calling 'build list --job-name' API, and get the latest one item, because the results are ordered by start time, like:

```
root@donotdel-openlab-allinone-l00242678:~/zuulcli/zuulcli# zuulctl build list --job-name spark-integration-test-kubeadm-k8s-v1.14.0
+----------------------------------+-------------+-----------------------------------------+-------------+---------------------+
| UUID                             | Project     | Job Name                                | Result      | Start Time          |
+----------------------------------+-------------+-----------------------------------------+-------------+---------------------+
| 13c7dbcffa374152b5ce0eeb059f8b50 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | RETRY_LIMIT | 2019-04-29T00:11:19 |
| 62436a0f6fef4abf8913527b5b982242 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | RETRY_LIMIT | 2019-04-28T12:12:50 |
| b4546db4454945b88189ae3b676195cd | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | FAILURE     | 2019-04-28T00:01:57 |
| b9d031b5da2442c3bdb1dc2c20a97f09 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | POST_FAILURE | 2019-04-27T12:09:53 |
| e9256e26efe346d693438252e5fa9506 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | RETRY_LIMIT | 2019-04-27T00:10:24 |
| 1414d4df2fbc4ddc9ece0edcb7a93543 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | RETRY_LIMIT | 2019-04-26T12:11:53 |
| ffe4271cb14e40a5aba8d0bf4dc914b0 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | SUCCESS     | 2019-04-26T00:01:41 |
| ab3bd2a238fb468ebd665b0299ef9594 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | RETRY_LIMIT | 2019-04-25T12:09:51 |
| 2c47e6ff2f664019a9eee916a59dbcd6 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | POST_FAILURE | 2019-04-25T06:31:58 |
| e88424eccb0c46028 1508a5abf0aad1e | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | FAILURE     | 2019-04-25T00:02:57 |
| ed54e248583b412bb97003759e44d043 | apache/spark | spark-integration-test-kubeadm-k8s-v1.14.0 | FAILURE     | 2019-04-24T12:02:32 |
```

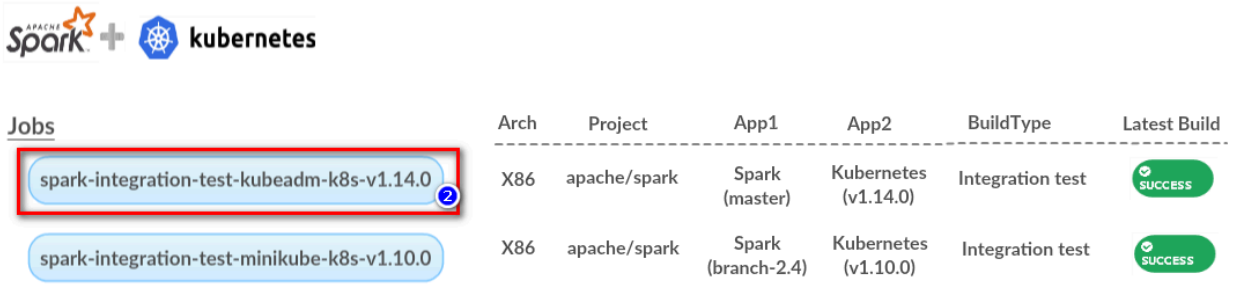
1) Click the build status icon will enter the latest build page of the job:



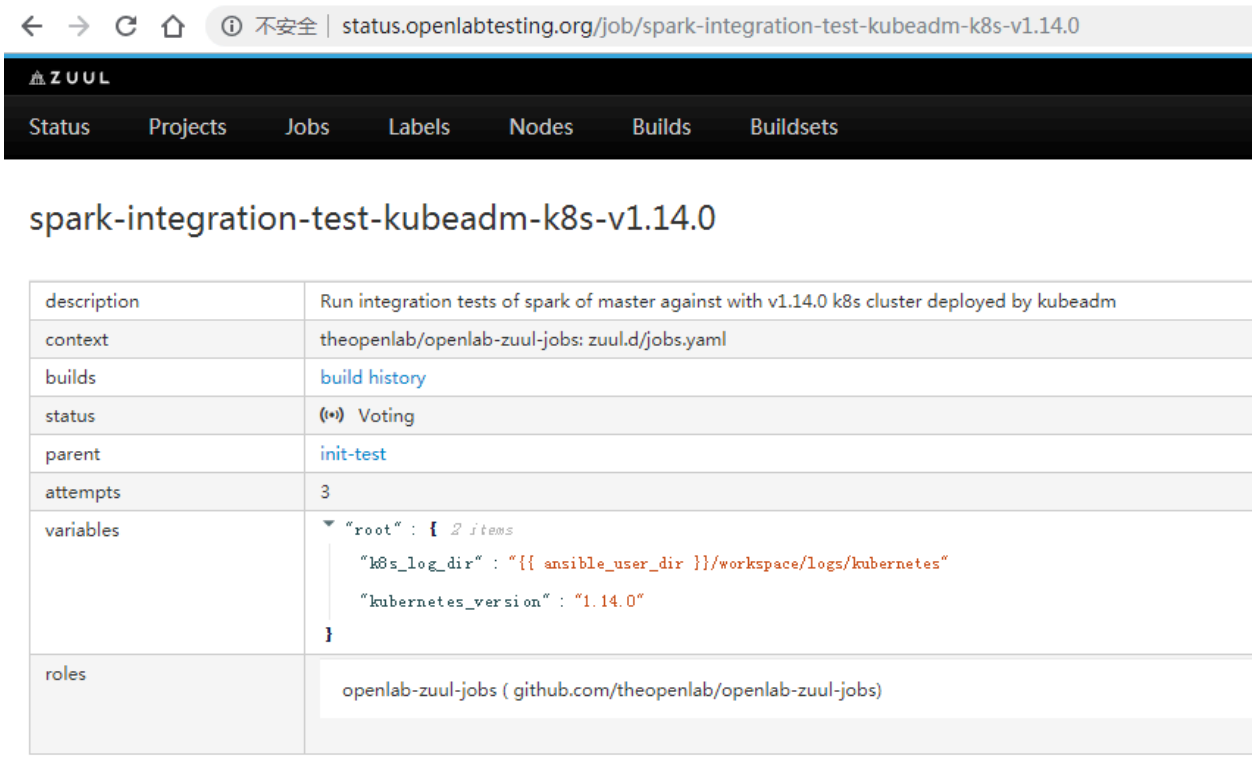
Jump to the page of the latest build of the job:



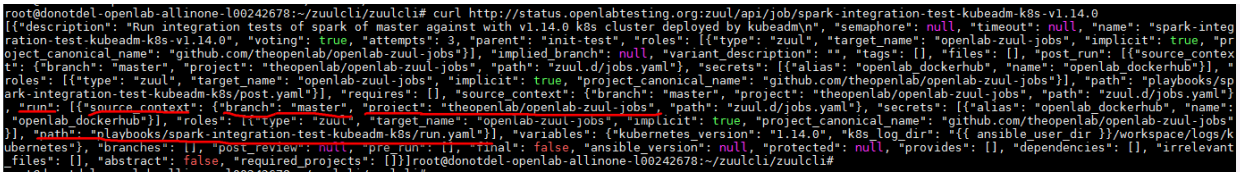
2) Click the job name will enter the job details page:



Jump to the page of job details:



And it will be good to show the url of run.yaml of the job on this page too. You can call 'job show' api to collect the url of run.yaml:



like the url of run.yaml for job 'spark-integration-test-kubeadm-k8s-v1.14.0' is ***github.com/theopenlab/openlab-zuul-jobs/tree/master/playbooks/spark-integration-test-kubeadm-k8s/run.yaml***

Joined by: github.com/*run(or parent.run).source_context.project*/tree/*master*/run(or parent.run).path