

TU TECHNOLOGY
UNIVERSITY

CICD WORKSHOP

UPDATED: 5.10.2021

CONTENTS

SECTION 01	
CICD OVERVIEW	5
CONTINUOUS INTEGRATION CONTINUOUS DEPLOYMENT	6
SECTION 02	
YAML	9
"YAML AIN'T MARKUP LANGUAGE"	10
YAML CONFIG FILE	11
YAML LINTER	12
SECTION 03	
CIRCLECI CONFIG	15
SECTION 04	
CREATE PIPELINE	21
ADD APPLICATION IN HAL	22
SETUP DEPLOYMENT TARGET	24
SETUP PROJECT IN CIRCLECI	26
SET SONAR TOKEN IN CIRCLECI	27
SET HAL API TOKEN IN CIRCLECI	30
SET CIRCLCI TOKEN IN HAL	32
REPLACE HAL APP ID AND TARGETS	35
COMMIT CHANGES	37



“Whether you think you can or think
you can't, you're right.”
Henry Ford





SECTION 01

CICD OVERVIEW

CICD

Continuous Integration

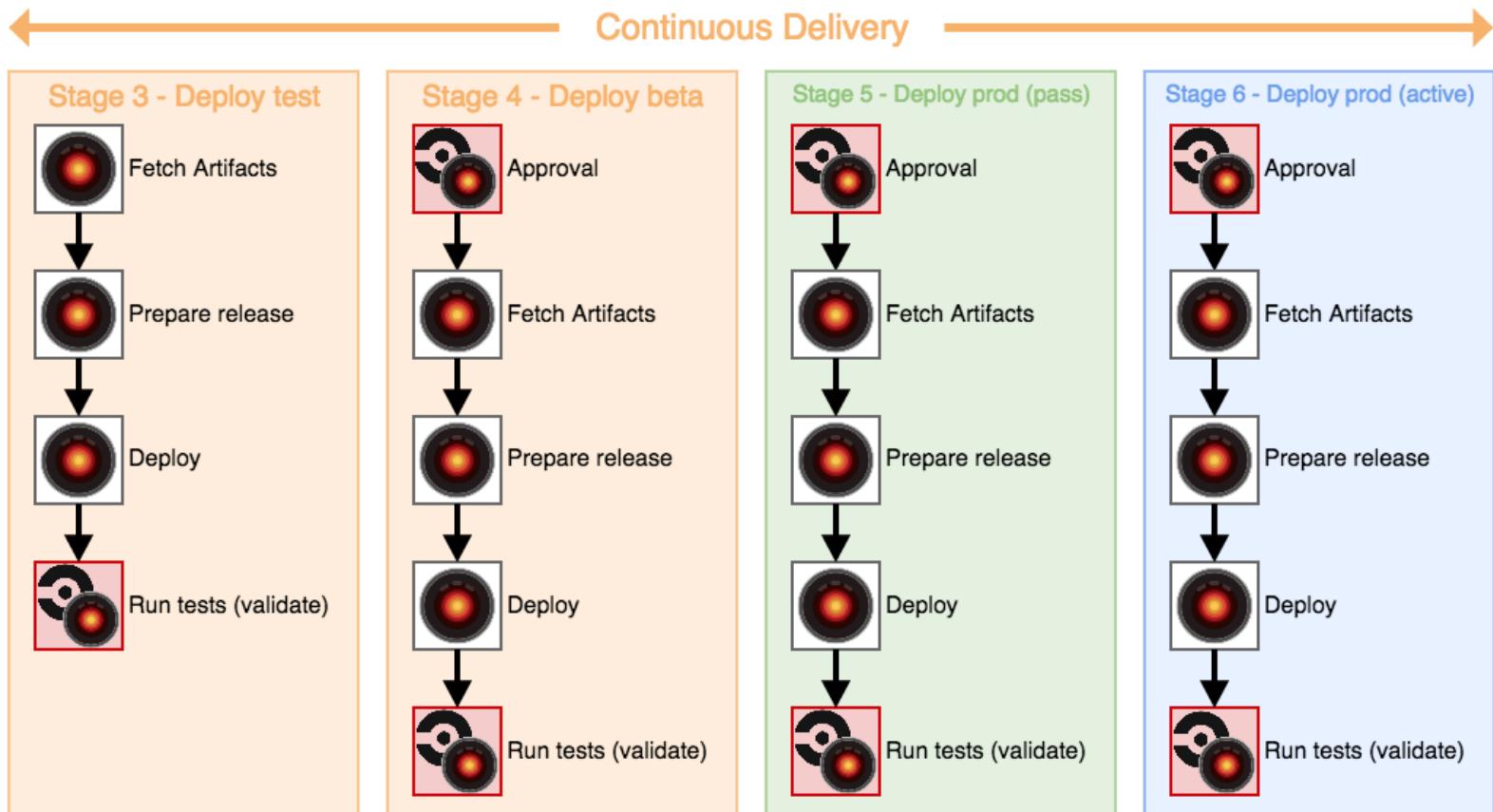
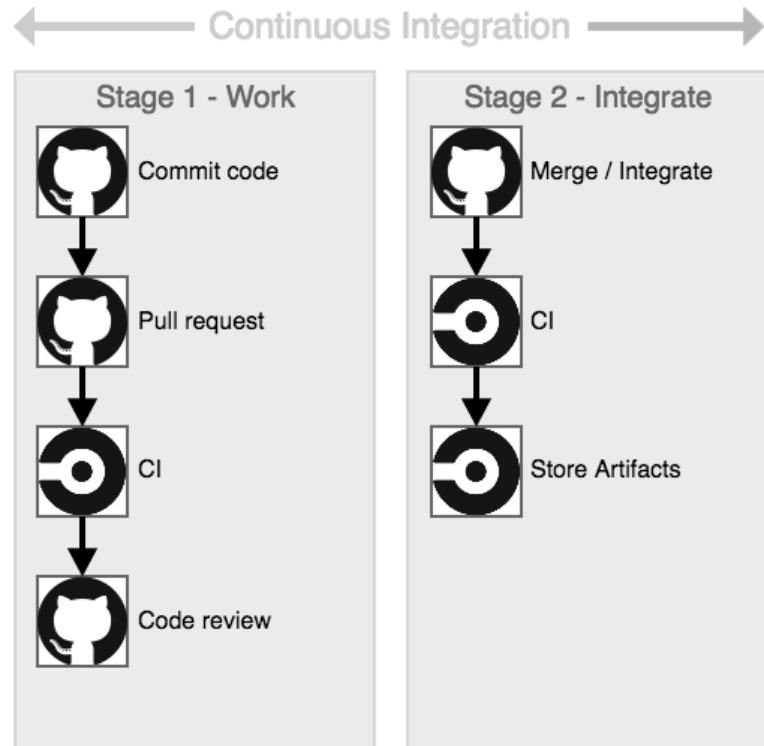
Continuous Integration is a development practice where developers continually run tests and integrate new code into their main branch.

If you merge code frequently, what might you avoid?

Continuous Delivery / Continuous Deployment

Continuous Delivery typically has manual steps (at least to prod), and deployment is fully automated.

Notes



GitHub



Circle CI



Hal



Pipeline system (CircleCI or Hal)

"Don't be afraid to give up the good to
go for the great."
John D. Rockefeller





SECTION 02

YAML

YAML

“YAML Ain’t Markup Language”

YAML was created for data serialization, and is a superset of JSON.

Whitespace is important, and used for denoting structure. Remember to never use tabs with YAML. List members are marked by a leading hyphen, and only one member is allowed per line.

YAML has associative arrays that use key value pairs and colon space, ex: `key : value`. An ampersand can be used as an anchor to create a node reference. An asterisk can then be used as an alias to refer back to the anchor.

NOTES

YAML CONFIG FILE

Create Config File

- 1) In GitHub, go to *your fork* of TechU-TestProject-CICD. If you need to fork the repo, you can find it here: <https://git.rockfin.com/techu-training/TechU-TestProject-CICD>
- 2) Click the button to **Create new file**

forked from [techu-training/TechU-TestProject-CICD](#)

Code Pull requests Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

This branch is 1 commit ahead of techu-training:master.

Create new file Upload files Compare

	aduncan1 Update config.yml	684e2ea 8 minutes ago	29 commits
	.circleci	Update config.yml	8 minutes ago
	public	added files	2 years ago
	src	added files	2 years ago

- 3) The new file will be called **.hal.yaml**
The five lines of the file will be as follows:

TechU-TestProject-CICD / .hal.yaml Cancel

Edit new file Preview

```
1 platform: 'linux'
2 image: 'circleci'
3 dist: '.artifact'
4 build:
5   - 'unpack-build-from-circleci ".artifact"'
```

YAML Linter

Before we commit our config file, we need to run it through a linter to verify it is formatted correctly.

- 1) Go to yamllint.com. This is the linter we will use today. ***Remember to never put secrets in an online linting tool***
- 2) Paste your YAML in and click Go. A green banner should pop up verifying you have valid YAML.

YAML Lint

Paste in your YAML and click "Go" - we'll tell you if it's valid or not, and give you a nice clean UTF-8 version of it. Optimized for Ruby.

```
1  ---
2   build:
3     - "unpack-build-from-circleci \".artifact\""
4   dist: .artifact
5   image: circleci
6   platform: linux
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

Go

Valid YAML!

Commit Config File

- 1) In GitHub, scroll down and select the radio button to **Create a new branch for the commit and start a pull request**. You can keep the default name given to the branch.
- 2) Click **Propose new file**.
- 3) For the pull request, you can add a note "Adding Hal config file," then click **Create pull request**.
- 4) On the next page, click **Merge pull request**.
- 5) Finally, click **Attempt merge**. Your PR should now be successfully merged and closed, and the Hal config file should be in your repo.

SECTION 02

YAML

13 of 40

forked from [techu-training/TechU-TestProject-CICD](#)

[Code](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

Test Repo with incomplete YAML [Edit](#)

Manage topics

- 29 commits 2 branches 0 releases 4 contributors

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download ▾](#)

This branch is 3 commits ahead of techu-training:master. [Pull request](#) [Compare](#)

aduncan1 Update config.yml	Latest commit 54984ec 6 hours ago
.circleci	Update config.yml 6 hours ago
public	added files 12 months ago
src	added files 12 months ago
.gitignore	Create .gitignore 12 months ago
.hal.yaml	Create .hal.yaml 7 hours ago
README.md	Update README.md 9 months ago
package.json	added files 12 months ago
sonar-project.properties	added files 12 months ago
sonar-report.sh	added files 12 months ago
yarn.lock	added files 12 months ago

NOTES

"Creativity is intelligence having fun."

Albert Einstein



SECTION 03

CIRCLECI CONFIG

SECTION 03 CIRCLECI CONFIG

16 of 40

Open up the `.circleci` folder to access the `config.yml` file. Here we can see how our build is prepared so it can be deployed by Hal. Lines 1-37 are where many of our defaults are defined.

```
1  attach_workspace: &attach_workspace
2    attach_workspace:
3      at: '.'
4
5  persist_workspace: &persist_workspace
6    persist_to_workspace:
7      root: '.'
8      paths: [ '.' ]
9
10 persist_hal_build_id_to_workspace: &persist_hal_build_id_to_workspace
11   persist_to_workspace:
12     root: '.'
13     paths: [ '.hal_build_id' ]
14
15 persist_coverage_to_workspace: &persist_coverage_to_workspace
16   persist_to_workspace:
17     root: '.'
18     paths: [ 'test-report.xml', 'coverage/' ]
19
20 deploy_nonprod_job: &deploy_nonprod_job
21   working_directory: '~/.project'
22   docker: [ {image: 'qldockerdtr.rockfin.com/circleci/hal-integration:hal-publisher'} ]
23   steps:
24     - *attach_workspace
25     - run: 'deploy-to-hal-passive-only'
26
27 deploy_prod_job: &deploy_prod_job
28   working_directory: '~/.project'
29   docker: [ {image: 'qldockerdtr.rockfin.com/circleci/hal-integration:hal-publisher'} ]
30   steps:
31     - *attach_workspace
32     - run: 'deploy-to-hal'
33
34 deploy_defaults: &deploy_defaults
35   working_directory: '~/.project'
36   docker:
37     - image: 'qldockerdtr.rockfin.com/circleci/hal-integration:hal-publisher'
```

NOTES

Line 39 indicates what version of the config schema we will use.

Line 40 is where we begin our jobs and essentially are telling CircleCI what to do/run.

Line 41-55 is the compile and build job, and is named based on naming conventions at shorty/gatenaming.

Line 57-64 is the unit tests job. This allows us to automate our unit tests so that they can be run any time new code is committed.

```
39  version: 2
40  jobs:
41    compile_and_build:
42      docker:
43        - image: node:12.6
44      steps:
45        - checkout
46        - run:
47          name: Greeting
48          command: echo Hello, world.
49        - run:
50          name: Yarn Install
51          command: yarn install
52        - run:
53          name: Yarn Build
54          command: yarn build
55        - *persist_workspace
56
57    unit_tests:
58      docker:
59        - image: node:12.6
60      steps:
61        - *attach_workspace
62        - run:
63          name: Yarn Test
64          command: yarn test
65
```

NOTES

SECTION 03 CIRCLECI CONFIG

18 of 40

Line 66-79 is the publish build to Hal job. This is how we get our build from CircleCI to Hal for deployment.

Line 81-89 is test deploy. This is what feeds the necessary information to Hal, so Hal can deploy our application to a designated test environment.

Line 91-97 measure code coverage.

```
66    publish_build_to_hal:
67      working_directory: '~/'
68      docker: [ {image: 'qldockerdtr.rockfin.com/circleci/hal-integration:hal-publisher'} ]
69      environment:
70        HAL_APP_ID: '6941'    #Fill in HAL application ID
71      steps:
72        - *attach_workspace
73        - run: mkdir -p "${HOME}/build/"
74        - run: cd build && zip -r "${HOME}/build/build.zip" . && cd ../
75        - run: ls "${HOME}/build/"
76        - store_artifacts:
77          path: "./build/build.zip"
78        - run: 'publish-build-to-hal'
79        - *persist_hal_build_id_to_workspace
80
81    test_deploy:
82      <<: *deploy_defaults
83      environment:
84        - HAL_APP_ID: '6941'    #Fill in HAL application ID
85        - HAL_TARGETS: "39319" #Fill in HAL deployment target ID
86        - HAL_BUILD_FILE: '.hal_build_id'
87      steps:
88        - *attach_workspace
89        - run: 'deploy-to-hal'
90
91    cover:
92      docker:
93        - image: node:12.6
94      steps:
95        - *attach_workspace
96        - run: yarn cover
97        - *persist_coverage_to_workspace
```

NOTES

Line 99-110 is what feeds testing data to SonarQube.

Line 112-129 is where we orchestrate how the jobs above are run using workflows.

Remember, the order designated in your workflows determines the order in which your jobs are run.

```
--  
99    sonar:  
100      working_directory: '~/project'  
101      docker:  
102        - image: qldockerdtr.rockfin.com/qapow/sonarscanner  
103      steps:  
104        - *attach_workspace  
105        - run: ls coverage  
106        - run:  
107          name: Replace Circle Hash  
108          command: sed -i -e "s~\${CIRCLE_SHA}~\$CIRCLE_SHA1~;" sonar-project.properties  
109        - run: cat sonar-project.properties  
110        - run: ./sonar-report.sh  
111  
112      workflows:  
113        version: 2  
114        workflow:  
115          jobs:  
116            - compile_and_build  
117            - unit_tests:  
118              requires: [ compile_and_build ]  
119            - cover:  
120              requires: [ unit_tests ]  
121            - sonar:  
122              requires: [ cover ]  
123            - publish_build_to_hal:  
124              requires: [ unit_tests ]  
125              filters:  
126                branches:  
127                  only: main  
128            - test_deploy:  
129              requires: [ publish_build_to_hal ]
```

NOTES

"The only limit to our realization of
tomorrow will be our doubts of today."
Franklin D. Roosevelt





SECTION 04

CREATE PIPELINE

SECTION 04 CREATE PIPELINE

22 of 40

ADD APPLICATION IN HAL

- 1) In Chrome, go to hal9000/ and sign into Hal. (Use a Dash account for more security)
- 2) Once signed-in, click on Apps.

The screenshot shows the Hal application interface. At the top, there's a dark header bar with the Hal logo, version 3.1.0-ea692e1, and a navigation menu with 'Dashboard', 'Apps' (which is highlighted with a red underline), 'Job History', and 'Docs'. A user profile icon for 'aduncan1' is on the right. Below the header, the main area has a red background. On the left, there's a large circular icon with a red and yellow center, next to the word 'Hal'. In the middle-left, the text 'I AM A HAL 9000 COMPUTER' is displayed. To the right, a section titled 'Deploy your App, When and How you want.' lists several features with checkmarks: Native support for AWS, Run Infrastructure as Code with Terraform or CloudFormation, Flexible YAML-based Configuration as Code, Supports any language in flexible docker environments, Run jobs on Linux or Windows, Deploy containers to ECS, Kubernetes, or Docker Enterprise, and Write custom scripted deployments to run any process. At the bottom right of this section is a 'View your Dashboard' button.

- 3) Click on Add Application.

The screenshot shows the 'Your Applications' section of the Hal interface. The top navigation bar is identical to the previous screenshot. Below it, a breadcrumb navigation shows 'Streams' → 'Organizations' → 'All Applications' → 'Your Applications'. The main title 'Your Applications' is centered above a horizontal line. Below the line are two buttons: 'Add Application' (highlighted with a red circle and a hand cursor icon) and 'Add Organization'. Further down is a search bar with a magnifying glass icon and the placeholder text 'Search all applications...'. The background of this section is white.

- 4) Give the new application for your pipeline a name, ex. Tech U CICD Workshop.
- 5) Set the organization to **Tech U**.
*Make sure you select Tech U with a space. If you select TechU without the space your pipeline will NOT work.
- 6) For **GitHub Enterprise Repository**, enter in the URL for your fork.
- 7) Click **Add Application**. (It is not necessary to fill out Additional Details or Advanced options for this workshop.)

Add Application

Select an organization and enter your GitHub Repository below. If you are not sure which Organization to use, you can search for your Team or Release Train in [Organizations](#).

Your environments and deployment settings are configured in the next step, and everything you set here can be changed later.

Required Information

Name **4)**

Title for this application such as [My Secret Project](#).

Organization **5)**

 x ▾

GitHub Enterprise Repository **6)**

Enter the full URL or organization/repository such as <https://git.rockfin.com/team-a/project-b> or [team-a/project-b](#).

[Show advanced options](#)

7)

Additional Details

App Identifier

Unique identifier such as [my-project-1234](#), or [App ID](#) from AppHub.

Provider

 ▼

Platform

 ▼

Language

 ▼

SECTION 04 CREATE PIPELINE

24 of 40

SETUP DEPLOYMENT TARGET

- 1) Scroll down under the blue header labeled Add deployment configuration for any environment, and click on Manually configure your environments.

BUILD AND DEPLOY

Start New Build This application has no deployment targets configured for **test** and cannot be deployed yet.

Manage Application

[View Job History](#)

CONFIGURATION

Deployment Settings

Encrypted Configuration

Enable Runtime Configuration

CI / CD

[CircleCI Guides](#)

Set up your application

AWS Elastic Beanstalk
Set up your environments and Blue/Green deployment configuration.

Mulesoft Anypoint
Add environment configuration for API Gateway or ESB.

Terraform
Configure Hal to use your terraform modules.

Configure **Configure** **Configure**

Add deployment configuration for any environment.

You can also manually configure your environments if we do not have a wizard for your application type such as [On-premises Linux](#), [AWS S3](#), [AWS CodeDeploy](#), or [Custom Scripts](#). [Manually configure your environments](#)

- 2) For the datacenter provider and environment, select dev-aws.

Select a datacenter provider and environment

On-premises / Kubernetes / Flyway

sandbox	dev	test
beta	uat	staging
pre-prod	prod	

AWS

sandbox-aws	dev-aws	test-aws
beta-aws	uat-aws	prod-aws

3) For the Deployment Type, select AWS S3.

Deployment Type

- General:** Script RSync Terraform Kubernetes Flyway
- AWS:** AWS S3 AWS Elastic Beanstalk AWS CodeDeploy AWS Lambda
- Blue/Green:** AWS Elastic Beanstalk Blue/Green AWS CodeDeploy Blue/Green

- 4) Under Configure Deployment, give your deployment a name, ex. Tech U Workshop.
- 5) For Authentication, select Tech U - Sandbox - Deployment (AWS Role). *If this option is not available, ensure you selected Tech U as your organization when adding the app.
- 6) For Region, select us-east-2.
- 7) For Bucket Name, type in pjsawicki-ql-training.
- 8) Click Add Deployment Setting.

Configure Deployment

AWS S3

Name **4)**

Tech U Workshop

Optionally, give a deployment a unique name for easy reference or to avoid confusion with multiple deployments of the same type.

URL

Optionally, Add a URL to help users find where the code is deployed.

Script Context

Optionally, pass extra context text or data to your scripts and commands in your `.hal.yaml` configuration (defined as `HAL_CONTEXT`). **Do not use this for secrets!**

Authentication (optional) **5)**

Tech U - Sandbox - Deployment (AWS Role)

AWS Role credentials will be injected into the default AWS profile. API Token credentials will be injected into the `AUTH_API_KEY` and `AUTH_API_SECRET` environment variables.

RSync authentication is handled globally and cannot currently use auth credentials.

AWS Account

qltraining (825301540467)

This AWS account is connected to your organization for this environment.

Region **6)**

us-east-2

Contact an administrator if you need to add additional regions.

Bucket Name **7)**

pjsawicki-ql-training

Hal automatically creates buckets in every account / region that can be used for deployments. To use this bucket for this deployment, enter `hal-drop-825301540467-$REGION`.

See "Granting Hal Access to Deploy to your Bucket" for more information.

S3 File object or path

Available placeholders:

`$APPID $APPKEY $ENV $BUILDID $PUSHID $DATE $TIME`

Leave blank to use `$PUSHID.tar.gz`. Prefix with `sync=` to use file syncing. Learn more in Hal Documentation.

Custom Parameters

Optionally, define additional variables that will be available to your scripts when your application is deployed with this configuration. **Do not use this for secrets!**

8)

Add Deployment Setting

Cancel

SECTION 04 CREATE PIPELINE

26 of 40

SETUP PROJECT IN CIRCLECI

- 1) Go to `circleci.foc.zone` and sign into CircleCI.
- 2) In CircleCI, you will click **Add Project** on the left to view your projects.
- 3) Click **Set Up Project** for your project named `TechU-TestProject-CICD`

The screenshot shows the CircleCI web interface. The top navigation bar includes a user icon, the username 'aduncan1', a dropdown menu, and links for 'Updates' and 'Support'. The sidebar on the left features icons for 'JOBS', 'WORKFLOWS', 'INSIGHTS', 'ADD PROJECTS' (which is highlighted with a red circle containing the number 2), 'TEAM', and 'SETTINGS'. The main content area is titled 'Add Projects' and contains instructions: 'CircleCI helps you ship better code, faster. Let's add some projects on CircleCI.' It says 'To kick things off, you'll need to choose a project to build. We'll start a new build for you each time someone pushes a new commit.' Below this, a list of projects is shown: 'Linux / Windows' and 'macOS'. Under 'Linux / Windows', there is a 'Filter projects...' input field and a 'Show Forks' checkbox. The project 'TechU-TestProject-CICD' is listed with a dropdown arrow. The project 'acd_git_lab' is also listed. On the far right, there are three buttons: 'Set Up Project' (highlighted with a red circle and number 3), 'Set Up Project', and 'Get Help'.

- 4) Linux and Node may already be selected for your Operating System and Language, respectively, but if not, go ahead and select those settings, then click Start Build.

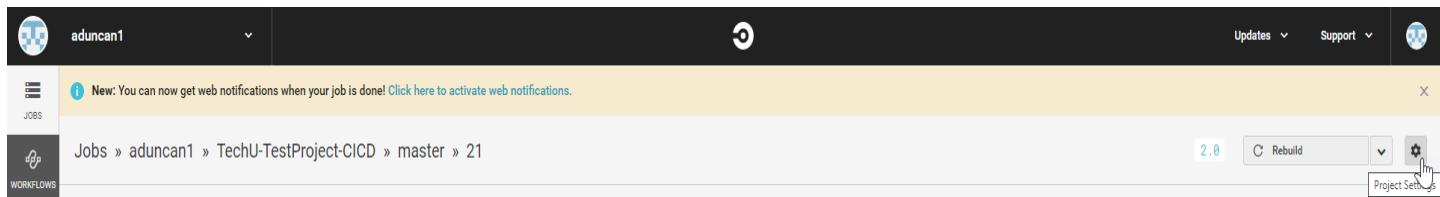
The screenshot shows the 'Set Up Project' screen for the 'TechU-TestProject-CICD' project. The top navigation bar includes a user icon, the username 'aduncan1', a dropdown menu, and a 'Updates' link. The sidebar on the left has 'ADD PROJECTS' selected. The main content area is titled 'Projects > Add Projects > aduncan1/TechU-TestProject-CICD' and is titled 'Set Up Project'. It contains instructions: 'CircleCI helps you ship better code, faster. To kick things off, you'll need to add a `config.yml` file to your project, and start building. After that, we'll start a new build for you each time someone pushes a new commit.' It says 'Select from the following options to generate a sample `.yml` for your project.' Below this, the 'Operating System' section shows 'Linux' selected. The 'Language' section shows 'Node' selected. The 'Next Steps' section lists five steps:

1. Create a folder named `.circleci` and add a file `config.yml` (so that the filepath be in `.circleci/config.yml`).
2. Populate the `config.yml` with the contents of the sample `.yml` (shown below).
3. Update the sample `.yml` to reflect your project's configuration.
4. Push this change up to GitHub.
5. Start building! This will launch your project on CircleCI and make our webhooks listen for updates to your work.

At the bottom, there is a 'Copy to Clipboard' button and a 'Start building' button (highlighted with a red circle and number 4).

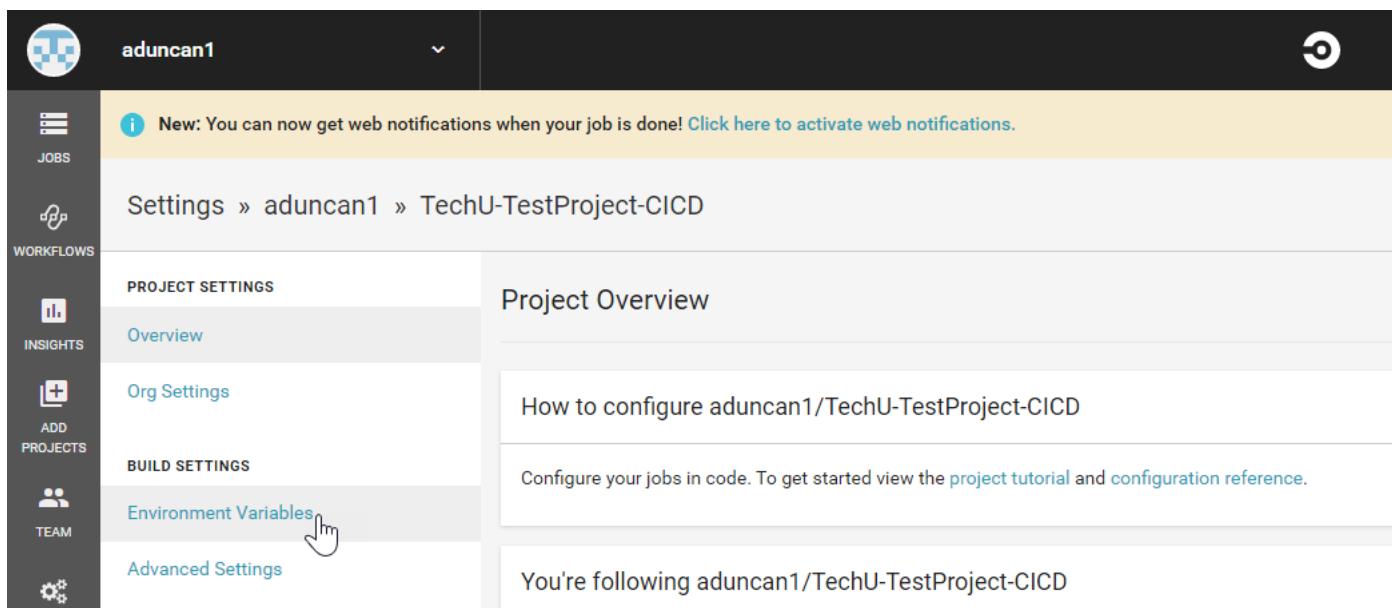
SET SONAR TOKEN IN CIRCLECI

1) Once you are taken back to your project page in CircleCI, click the gear in the upper right-hand corner to get to your Project Settings.



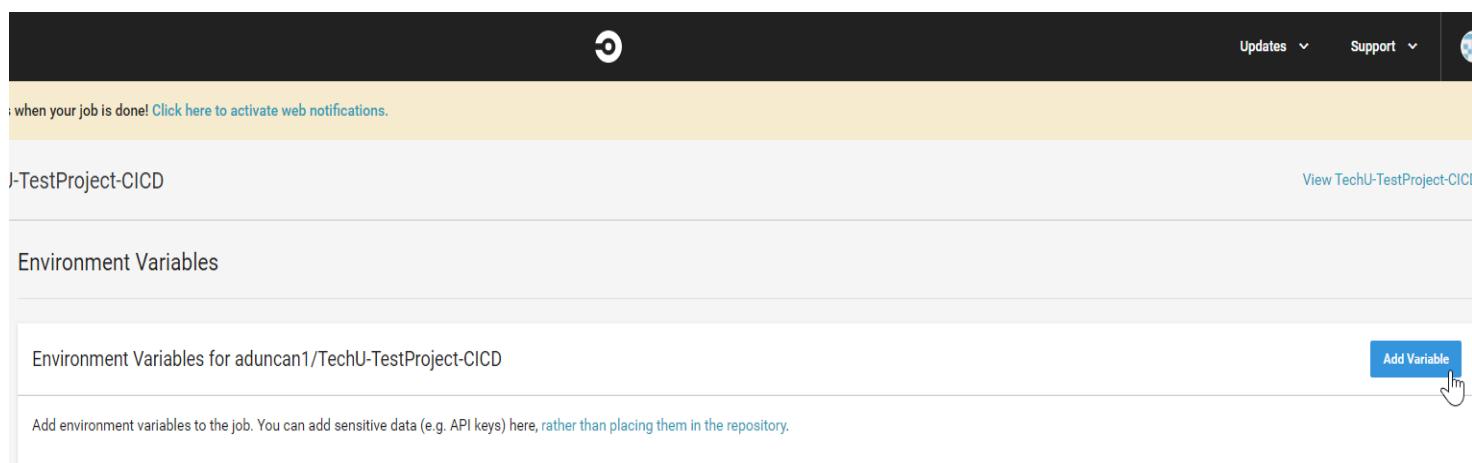
A screenshot of the CircleCI web interface. The top navigation bar shows the user's name 'aduncan1' and various links like 'Updates', 'Support', and a profile icon. Below the navigation is a yellow banner with a message about web notifications. The main content area shows the project path 'Jobs » aduncan1 » TechU-TestProject-CICD » master » 21'. On the far right, there is a 'Rebuild' button and a 'Project Settings' gear icon, which is highlighted with a mouse cursor. The left sidebar has sections for 'JOBS', 'WORKFLOWS', 'INSIGHTS', 'ADD PROJECTS', 'TEAM', and 'PROJECTS'.

2) On the left, under Build Settings, click Environment Variables.



A screenshot of the CircleCI 'Project Settings' page for 'aduncan1'. The left sidebar includes 'PROJECT SETTINGS' (with 'Overview' selected), 'BUILD SETTINGS' (with 'Environment Variables' highlighted and a mouse cursor over it), and 'Advanced Settings'. The main content area shows 'Project Overview' and 'How to configure aduncan1/TechU-TestProject-CICD'. A note says 'Configure your jobs in code. To get started view the [project tutorial](#) and [configuration reference](#)'. At the bottom, it says 'You're following aduncan1/TechU-TestProject-CICD'.

3) Inside Environment Variables, click Add Variable.



A screenshot of the CircleCI 'Environment Variables' page for 'aduncan1/TechU-TestProject-CICD'. The top navigation bar and sidebar are similar to the previous screenshot. The main content area shows the heading 'Environment Variables' and a sub-section 'Environment Variables for aduncan1/TechU-TestProject-CICD'. A blue 'Add Variable' button is visible on the right side, with a mouse cursor hovering over it. A note at the bottom says 'Add environment variables to the job. You can add sensitive data (e.g. API keys) here, rather than placing them in the repository.'

SECTION 04 CREATE PIPELINE

28 of 40

- 4) Go to Sonar/ and log into SonarQube (you can login with GitHub).
- 5) Once logged in, click Add project and then select Manually from the drop-down menu.

The screenshot shows the SonarQube dashboard. At the top, there are navigation links: sonarqube, Projects, Portfolios, Issues, Rules, Quality Profiles, Quality Gates, a search bar "Search for projects...", and a "DA" button. On the left, there's a sidebar with "My Favorites" (All), "Filters", and sections for "Quality Gate" (Passed: 0, Failed: 0), "Reliability" (A-E: 0), and "Security" (A-B: 0). The main area displays "2 projects": "dolphin" and "violin_lego". Each project card shows a star icon, the project name, a message about the main branch not being analyzed yet, and a "Configure analysis" button. A dropdown menu on the right is open, showing options: GitHub (selected), Manually, and a hand cursor icon pointing at the Manually option.

- 6) For your **Project key**, you can name it anything, the name is not important here. The **Display name** will be autogenerated.
- 7) Click **Set Up**.

The screenshot shows the "Create new project / Create manually" page. At the top, there are navigation links: sonarqube, Projects, Portfolios, Issues, Rules, Quality Profiles, Quality Gates, a search bar "Search for projects and files...", and a "DA" button. Below the header, there are fields for "Project key*" (nameAnything) and "Display name*" (nameAnything). Both fields have a green border and a checkmark icon. Below each field is a small explanatory text: "Up to 400 characters. All letters, digits, dash, underscore, period or colon." and "Up to 255 characters" respectively. At the bottom left, there is a "Set Up" button with a hand cursor icon pointing at it.

SECTION 04 CREATE PIPELINE

29 of 40

- 8) To generate a token, it can be named anything, the name is also not important at this step.
- 9) Click **Generate**.

The screenshot shows the SonarQube interface for a project named 'nameAnything'. The top navigation bar includes links for Projects, Portfolios, Issues, Rules, Quality Profiles, Quality Gates, a search bar, and a 'DA' button. Below the navigation is a breadcrumb trail: 'nameAnything' > 'master' > 'Project Settings' > 'Project information'. The main content area is titled 'Analyze your project' with the sub-instruction 'We initialized your project on SonarQube, now it's up to you to launch analyses!'. A numbered step 1 is displayed, titled 'Provide a token', with two options: 'Generate a token' (selected) and 'Use existing token'. A text input field contains 'nameAnything' and a 'Generate' button is visible. A note below states: 'The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#)'. A hand cursor icon is shown pointing at the 'Generate' button.

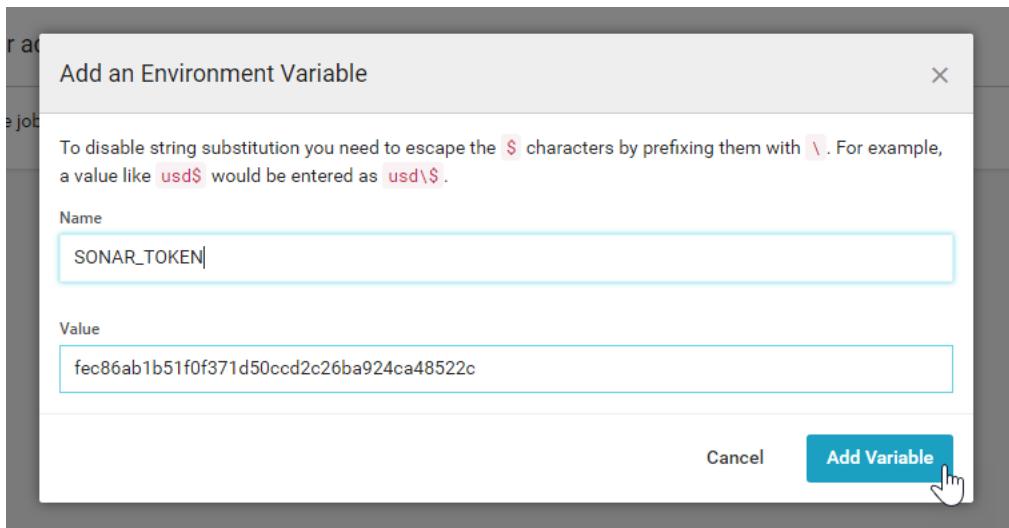
- 10) Highlight the token value that is generated and copy it. You can exit SonarQube.

The screenshot shows the same SonarQube interface for the 'nameAnything' project. The 'Provide a token' step is completed, displaying the generated token value: 'nameAnything: fec86ab1b51f0f371d50ccd2c26ba924ca48522c'. A trash bin icon is next to the token value. A note below reiterates: 'The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#)'. A 'Continue' button is visible at the bottom left of the step panel.

SECTION 04 CREATE PIPELINE

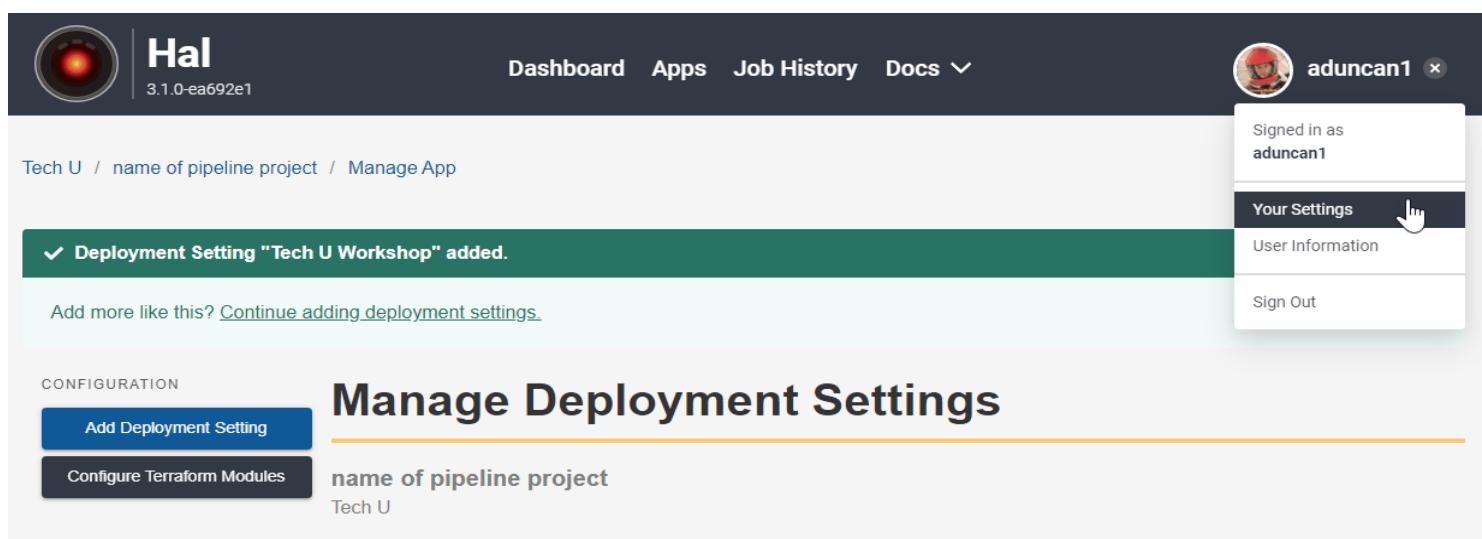
30 of 40

- 11) Back in CircleCI, in the Add an Environment Variable window, paste in the value from SonarQube.
- 12) For the Name, type SONAR_TOKEN. This name needs to be exact or CircleCI will not know what it is.
- 13) Click Add Variable.



SET HAL API TOKEN IN CIRCLECI

- 1) In CircleCI, inside Environment Variables, click Add Variable.
- 2) Go to Hal. Once there, click your user name in the top-right corner, then click Your Settings from the drop-down menu that appears.



The screenshot shows the CircleCI Hal interface. At the top, there's a navigation bar with 'Hal' and version '3.1.0-ea692e1'. Below it, a green banner says 'Deployment Setting "Tech U Workshop" added.' A link to 'Continue adding deployment settings.' is shown. On the left, under 'CONFIGURATION', there are buttons for 'Add Deployment Setting' and 'Configure Terraform Modules'. The main area is titled 'Manage Deployment Settings' and shows a single entry for 'name of pipeline project' with 'Tech U'. On the right, a user profile for 'aduncan1' is shown with options to 'Sign in as aduncan1', 'Your Settings' (which has a hand cursor pointing at it), 'User Information', and 'Sign Out'.

- 3) On the Your Settings page, scroll down to Add new personal access token.
- 4) For the Token Name, you can name it anything.
- 5) For the Token Scope, select Builds and Non-prod deployments from the drop-down.
- 6) Click Generate Token.

Personal Access Tokens

Personal Access Tokens can be used to access the Hal API.

You have no personal access tokens saved.

Add new personal access token

You must still have access to specific organizations or applications to be able to deploy them through the API. Scopes are only used to *limit* the situations a token can be used in, and **do not grant any special access your user account does not already have**. For example, creating a **prod** token when you do not have access access to prod will have no effect.

Token Name **4**
niceDay

Token Scope

- Read-only
- Read-only
- Builds and Non-prod deployments **5****
- Prod deployments
- Infrastructure deployments (Non-prod and prod)

Generate Token **6**

- 7) Now that your token has been generated, click **Copy** to copy the value of the token.

Personal Access Tokens

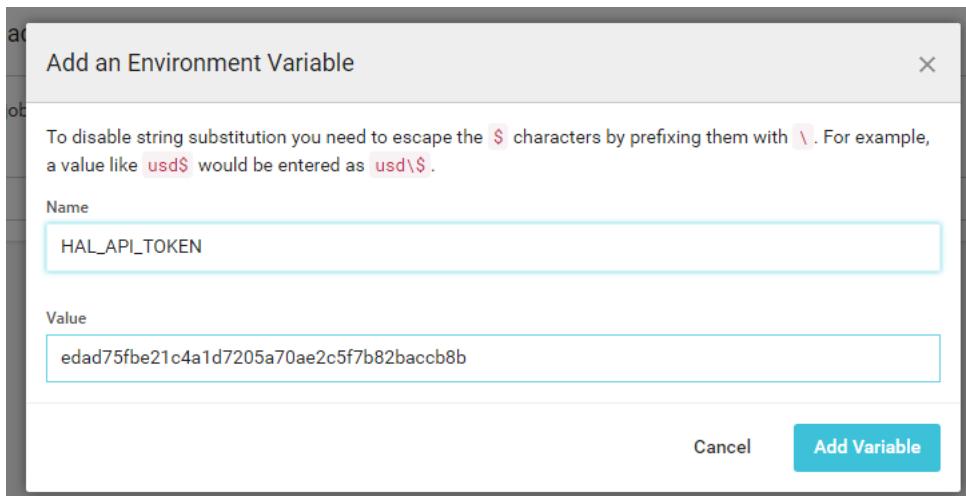
Personal Access Tokens can be used to access the Hal API.

Name	Scope	Token Secret	
niceDay	Builds and Non-prod deployments	Copy Show Secret ...cb8b	Revoke Token

SECTION 04 CREATE PIPELINE

32 of 40

- 8) Back in CircleCI, in the Add an Environment Variable window, paste in the value from Hal.
- 9) For the Name, type HAL_API_TOKEN. This name needs to be exact or CircleCI will not know what it is.
- 10) Click Add Variable.



SET CIRCLECI TOKEN IN HAL

- 1) In the menu on the left side, scroll down under PERMISSIONS and click API Permissions.

The screenshot shows the CircleCI settings interface. The sidebar has sections for JOBS, WORKFLOWS, INSIGHTS, ADD PROJECTS, TEAM, and SETTINGS. Under SETTINGS, the "API Permissions" section is highlighted with a cursor icon. The main panel shows PROJECT SETTINGS with Environment Variables selected. On the right, a sidebar shows Environment Variables for the project, with fields for Name (containing HAL_API_TOKEN and SONAR_TOKEN) and a note to add environment variables to the project.

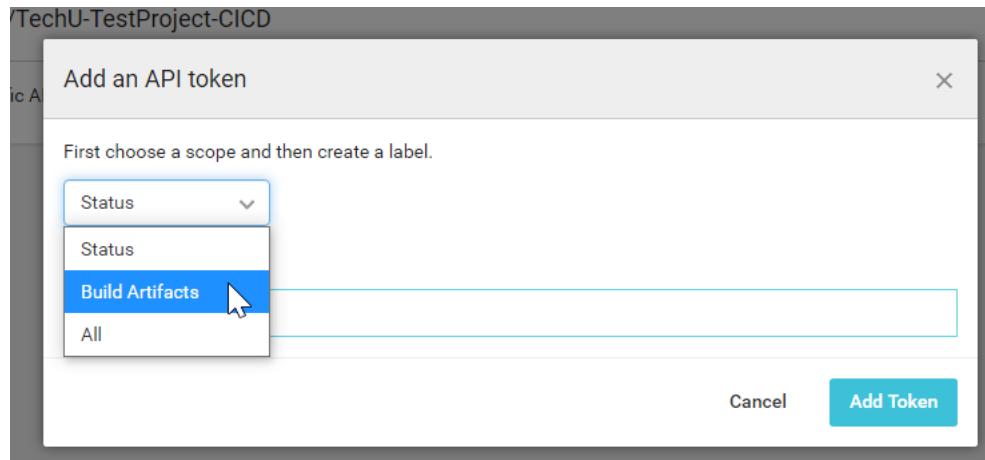
SECTION 04 CREATE PIPELINE

33 of 40

- 2) On the API Permissions page, click Create Token.

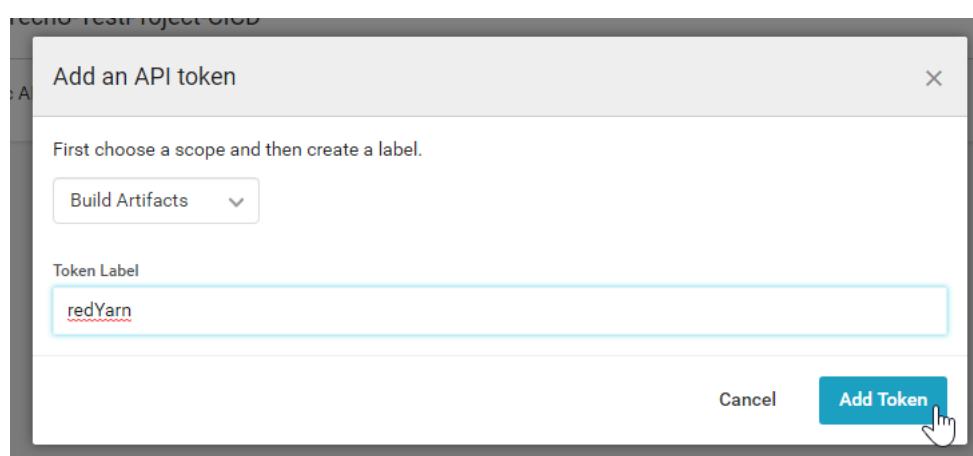
The screenshot shows a web interface for managing API permissions. At the top, there are links for 'Updates' and 'Support'. Below that, a message says 'when your job is done! Click here to activate web notifications.' A project name '-TestProject-CICD' is shown with a link 'View TechU-TestProject-CICD ». On the left, there's a section titled 'API Permissions'. Underneath it, a list of 'API tokens for aduncan1/TechU-TestProject-CICD' is shown. A blue button labeled 'Create Token' is visible on the right side of the token list, with a mouse cursor hovering over it.

- 3) In the Add an API token window that pops-up, click the drop-down for Status, and select Build Artifacts.



- 4) For the Token Label, you can call it anything you want.

- 5) Click Add Token.



SECTION 04 CREATE PIPELINE

34 of 40

6) In Hal, under CONFIGURATION, go to Secret Variables.

The screenshot shows the Hal application interface. At the top, there is a navigation bar with a logo, the word "Hal", the version "3.2.0-c3c2d75", and links for "Dashboard", "Apps", "Job History", "Docs", and a user profile "aduncan1". Below the navigation bar, the page title is "Tech U CICD Workshop 5_10_21". On the left, there are two main sections: "BUILD AND DEPLOY" and "CONFIGURATION". Under "BUILD AND DEPLOY", there are buttons for "Start New Build", "Manage Application", and "View Job History". Under "CONFIGURATION", there are buttons for "Deployment Settings", "Secret Variables" (which has a hand cursor icon over it), and "Manage Runtime Secrets". The "Secret Variables" button is highlighted with a blue background. In the center, the "Deployment Settings" section is displayed, showing a yellow box labeled "Tech U Workshop 5_10_2021" and a note below it stating "This configuration has never been deployed." At the bottom right of this section, there is an "Info" link.

- 7) For the Variable Name, type CIRCLECI_TOKEN. This name needs to be exact or Hal will not recognize it.
- 8) For the Value, paste in the token value you copied from CircleCI.
- 9) For the Environment, select Global (Available in all environments) from the drop-down menu.
- 10) Click Add Variable.

The screenshot shows a modal dialog titled "Add Encrypted Configuration". It contains three input fields: "Variable Name" with the value "CIRCLECI_TOKEN", "Value" with the value "433e9d89a4a39ad551735e6a500092978a860d65", and "Environment" with the value "Global (Available in all environments)". A green "Add Variable" button is at the bottom right, with a hand cursor icon over it. Numbered callouts (7, 8, 9, 10) point to each of these four fields respectively.

REPLACE HAL APP ID AND TARGETS

1) In Hal, on the main page for your app, scroll down to **Deployment Settings**, and click on the name of your project. Be sure to click the white text. Clicking the yellow box won't work.

The screenshot shows the Hal app interface. At the top, there's a navigation bar with a user icon, the text "aduncan1", and a dropdown arrow. Below the navigation bar, the URL "Tech U / name of pipeline project / Manage App" is visible. On the left, there's a sidebar with "BUILD AND DEPLOY" and "CONFIGURATION" sections. Under "BUILD AND DEPLOY", there are buttons for "Start New Build" (blue), "Manage Application" (black), and "View Job History". Under "CONFIGURATION", there are buttons for "Deployment Settings" (blue) and "Encrypted Configuration" (black). The main content area has a title "★ name of pipeline project". Below it, there's a section titled "Deployment Settings" with a sub-section for "Tech U Workshop". A tooltip says "This configuration has never been deployed." A cursor is hovering over the "Tech U Workshop" text. The entire screenshot is framed by a light gray border.

2) From this page, you can find your **Hal App ID** and **Hal deployment target ID** in the URL. The Hal App ID will be the 4-digit number, and the Hal deployment target ID will be the 5-digit number. We will be copying and pasting these into our CircleCI YAML file in our Git repo.

The screenshot shows the Hal app interface. At the top, there's a browser-like header with tabs: "I-TestProject-CIC", "Deployment Setting - name of pi", "Project settings - aduncan1/Tech", and a "+" button. Below the header, the URL "hal.zone/applications/6941/deployments/39319" is shown, with the numbers "6941" and "39319" highlighted with red boxes. The main content area has a title "Deployment Setting". Below it, there's a "JOBS" section with a button "Start Build for Deployment". The entire screenshot is framed by a light gray border.

SECTION 04 CREATE PIPELINE

36 of 40

3) In GitHub, in your fork, open the folder called `.circleci`.

This screenshot shows a GitHub repository interface. At the top, there are navigation links: Code (selected), Pull requests 0, Projects 0, Wiki, Insights, and Settings. Below the header, it says "Test Repo with incomplete YAML". On the right, there's an "Edit" button. Under the repository stats (28 commits, 2 branches, 0 releases, 3 contributors), there are buttons for Branch: master, New pull request, Create new file, Upload files, Find file, and Clone or download. A note says "This branch is 2 commits ahead of techu-training:master." Below this, a list of commits shows: "aduncan1 Merge pull request #1 from aduncan1/aduncan1-patch-1" (Latest commit 9dde4d8 1 hour ago), ".circleci" (Update config.yml 5 months ago), "public" (added files 12 months ago), "src" (added files 12 months ago), and ".gitignore" (Create .gitignore 12 months ago). A cursor points at the ".circleci" commit.

4) Open the file called `config.yml`.

This screenshot shows a GitHub repository interface. At the top, there are navigation links: Code (selected), Pull requests 0, Projects 0, Wiki, Insights, and Settings. Below the header, it says "Branch: master" followed by the repository name "TechU-TestProject-CICD / .circleci /". Underneath, it says "This branch is 2 commits ahead of techu-training:master." On the right, there are buttons for Create new file, Upload files, Find file, and History. A note says "ksezniaak Update config.yml" (Latest commit 58e77eb on Jun 10). Below this, a list of commits shows: "config.yml" (Update config.yml 5 months ago). A cursor points at the "config.yml" commit.

5) Click the pencil icon on the right to Edit this file.

This screenshot shows a GitHub file editor for the "config.yml" file. At the top, there are navigation links: Code (selected), Pull requests 0, Projects 0, Wiki, Insights, and Settings. Below the header, it says "Branch: master" followed by the repository name "TechU-TestProject-CICD / .circleci / config.yml". On the right, there are buttons for Find file and Copy path. A note says "ksezniaak Update config.yml" (58e77eb on Jun 10). Below this, it says "2 contributors". At the bottom, there's a summary: "129 lines (116 sloc) | 3.36 KB". On the far right, there are buttons for Raw, Blame, History, and a pencil icon with a tooltip "Edit this file". A cursor points at the pencil icon. The code editor shows the following content:

```
1 attach_workspace: &attach_workspace
2   attach_workspace:
3     at: '..'
```

SECTION 04 CREATE PIPELINE

37 of 40

6) Scroll down to lines 70 though 85. You will be pasting in the Hal App ID on line 70 and line 84. You will paste in the Hal deployment target ID on line 85.

```
working_directory: ~/
docker: [ {image: 'qldockerdtr.rockfin.com/circleci/hal-integration:hal-publisher'} ]
environment:
  HAL_APP_ID: '####'    #Fill in HAL application ID
steps:
  - *attach_workspace
  - run: mkdir -p "${HOME}/build/"
  - run: cd build && zip -r "${HOME}/build/build.zip" . && cd ../
  - run: ls "${HOME}/build/"
  - store_artifacts:
    path: "./build/build.zip"
  - run: 'publish-build-to-hal'
  - *persist_hal_build_id_to_workspace

test_deploy:
  <<: *deploy_defaults
environment:
  - HAL_APP_ID: '####'    #Fill in HAL application ID
  - HAL_TARGETS: "#####"  #Fill in HAL deployment target ID
  - HAL_BUILD_IDFILE: 'hal_build_id'
steps:
  - *attach_workspace
  - run: 'deploy-to-hal'

cover:
  docker:
    - image: node:12.6
  steps:
```

COMMIT CHANGES

1) Click Commit changes.

Commit changes

Update config.yml

Add an optional extended description...

Commit directly to the `master` branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes **Cancel**

SECTION 04 CREATE PIPELINE

38 of 40

Once those changes are committed, that kicks off the pipeline to begin running again in CircleCI. You can watch it run in CircleCI, in **Workflows**. It can take a few minutes to run.

The screenshot shows the CircleCI Workflows interface for a project named 'aduncan1'. On the left is a sidebar with icons for JOBS, WORKFLOWS (selected), INSIGHTS, ADD PROJECTS, TEAM, and SETTINGS. The main area displays a workflow titled 'master / workflow' triggered by 'Update config.yml'. The status is 'RUNNING' with a timestamp of '1 min ago' and a commit hash of '54984ec'. Below the status, it says '6 jobs in this workflow'. A horizontal flowchart shows the sequence of jobs: 'compile_and...' (green box, 01:10), 'unit_tests' (green box, 00:09), 'publish_build...' (blue box, 00:04), 'sonar' (purple box, 00:04), 'cover' (blue box, 00:04), and 'test_deploy' (purple box, 00:04). All jobs are marked with green checkmarks.

If the pipeline fails, you can see at what point it failed, and click on that job for more details.

The screenshot shows the CircleCI Workflows interface for the same project 'aduncan1'. The sidebar and workflow title are identical. However, the status is now 'FAILED' with a timestamp of '19 hrs ago' and a commit hash of '3c77d71'. The flowchart shows the same six jobs as before, but the 'publish_build...' job is highlighted in red with a red exclamation mark, indicating it failed. The other five jobs are green with checkmarks.

Once the problem is resolved, instead of completely rerunning the build, you can **Rerun from failed**.

The screenshot shows the CircleCI Workflows interface for the failed pipeline. The sidebar and workflow title are the same. The 'Rerun' button is highlighted with a red border. A dropdown menu is open, showing two options: 'Rerun from failed' and 'Rerun from beginning'. The 'Rerun from failed' option is selected, indicated by a blue arrow pointing to it.

NOTES

