Technical Paper

Drone CI

Table of contents

Drone CI

By Cserich Philipp

This technical Paper aims to give a broad introduction into the world of Drone **(by harness)**. It provides one possible setup and several examples how to configure drone for your individual project.

For further information about Drones functionality, extensions and more advanced usage, visit the original documentation.

All sources and many useful guidelines can be found on the last page.

→ Skip content to access links

What is Drone CI?

Drone is a continuous integration and delivery **(CI/CD)** platform that provides a way to automate the building, testing, and deployment of software. It is designed to be lightweight, easy to use, and highly scalable, making it a popular choice for teams looking to streamline[1] their software development and delivery processes. The by far most popular tool operating in that field is called **Jenkins**.

With Drone, developers can define their CI/CD pipelines in a configuration file called .drone.yml that is stored in their code repository[2]. This file specifies the steps that should be taken to build, test, and deploy the code, as well as any dependencies or resources required. When a change is made to the code repository, Drone automatically triggers the pipeline to run, performing the specified steps in order.

Drone offers a wide range of features and integrations, including support for multiple programming languages and platforms, automatic dependency management, and integration with popular version control systems and cloud platforms. It also provides a web-based user interface for monitoring and managing pipelines, as well as APIs[3] for integrating with other tools and systems including GitHub and Co.

Who has published Drone CI

Drone has been created by a company named **harness**.

When should you use Drone CI?

Use Cases

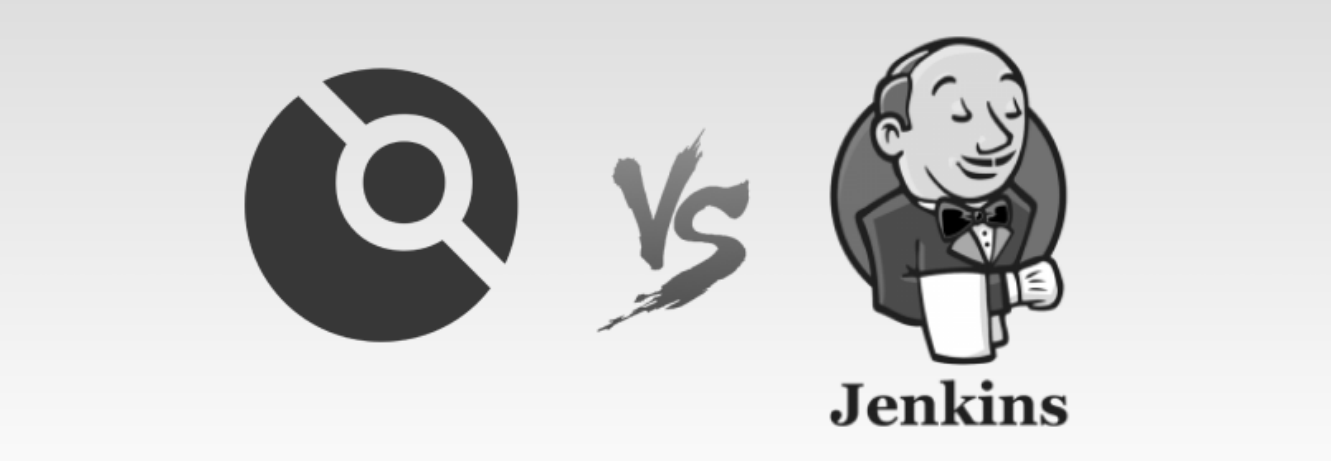
- Your products need to be tested, built and deployed onto a Server
- You do not want to do the deployment process by hand **(extremely important when your production software is updated frequently)**
- You are using a microservices architecture and want to automate the deployment of individual services.
- You want a lightweight and easy-to-use CI/CD platform that can scale to meet the needs of your team.

Overall, Drone can be a good choice if you are looking for a flexible and feature-rich CI/CD platform that is well-suited to a variety of use cases. However, it may not be the best fit for every team or project, and it is worth considering the specific needs and requirements of your team before deciding on a CI/CD solution.

Requirements

- Version Management tool in use **(Github / Gitlab / Bitbucket)**
- Buildable source code
- Docker installed

Comparison



Technology	**Jenkins**	**Drone**
Configuration	UI configured	.drone.yml file
Extension Points	Hilariously many	few verified options
Dependencies & Ressources	Manual	Automatic
Scalability & Performance	Powerful & Complex	lightweight & easy to use

Drone and Jenkins are both capable CI/CD platforms that can be used to automate the software development and delivery process. The right choice for your team will depend on your specific needs and requirements, as well as the size and complexity of your project.

Setup

Step 1 : Preparation

Create an OAuth App

(An OAuth app is used to centralize logins and password handling - One centered Authentication without handing your password to every third party Application)

1. Go to the GitHub developer settings page by clicking on your profile picture in the top right corner of the page and selecting "Settings" from the drop-down menu.
2. In the left-hand menu, click on "OAuth Apps" under the "Developer Settings" heading.
3. Click on the "New OAuth App" button.
4. In the "Application name" field, enter a name for your OAuth app.
5. In the "Homepage URL" field, enter the URL of your app's homepage.
6. In the "Authorization callback URL" field, enter the URL that GitHub should redirect to after the user has authorized your app.
7. Click the "Create OAuth App" button.

GitHub will then create your OAuth app and display a page with your client ID and client secret. These values will be used to authenticate your app when it makes API requests to GitHub.

You will also need to configure the permissions that your OAuth app requires. This can be done by clicking on the "Edit" button next to the "Scopes" heading on the OAuth app's settings page. GitHub will then display a list of available permissions that you can select from.

Once you have created and configured your OAuth app, you can use it to authenticate API requests to GitHub on behalf of your users. This can be useful for integrating your app with GitHub or for building integrations or tools that interact with GitHub data and functionality.

Shared Secret[4]

```
$ openssl rand -hex 16
```

Is used for communication between drone server and runners

Step 2 : Installation

Pull Drone Image[5]

```
$ docker pull drone/drone:[latest version]
```

[latest version] → 2

Step 3 : NGROK & DRONE Server Setup

Register to NGrok

(Ngrok is just used for the OAuth input → creating a valid globally reachable host adress)

→ Get your Ngrok Authtoken

1. start ngrok via docker image

```
docker run -it -e NGROK_AUTHTOKEN=<token> ngrok/ngrok http 80
```

2. run Drone docker image with following variables :

Volume

/var/lib/drone /data

DRONE_GITHUB_CLIENT_ID

c049c6f2caf1c9e8626e **(From your GitHub OAuth App)**

DRONE_GITHUB_CLIENT_SECRET

202bc9e4a409fcfd25d78c128d43d723664f05ad **(From your GitHub OAuth App)**

DRONE_RPC_SECRET

d80949b9fc2717d4949e078cdac552b1 **(previously generated Shared Secret)**

DRONE_SERVER_HOST

https://ece8-84-115-228-194.eu.ngrok.io **(ngrok forwarding domain)**

DRONE_SERVER_PROTO

https **(depending on ngrok)**

→ For more config. options click here

Step 4 : Drone Runner Setup

What is a runner?

A Drone runner is a program or service that executes the steps of a CI/CD pipeline[6][7][8].

In Drone, a pipeline is defined in a configuration file called .drone.yml that is stored in the code repository. This file specifies the steps that should be taken to build, test, and deploy the code, as well as any dependencies or resources required. When a change is made to the code repository, Drone automatically triggers the pipeline to run, and a runner is responsible for executing the steps in the pipeline.

Drone runners can be installed on your own infrastructure or used as a managed service, such as the Drone Cloud. They can be run in a variety of environments, including local development environments, virtual machines, and container[9] orchestration platforms.

Drone runners can be configured to run pipelines in parallel, allowing you to scale your CI/CD process to meet the needs of your team.

Run the drone runner container :

```
$ docker pull drone/drone-runner-docker:1
```

Necessary configurations :

Volume

/var/run/docker.sock : /var/run/docker.sock

DRONE_RPC_PROTO

http

DRONE_RPC_HOST

DRONE_RPC_SECRET

d80949b9fc2717d4949e078cdac552b1 **(previously generated Shared Secret)**

DRONE_RUNNER_NAME

name the runner as you want

→ For more config. options click here

Step 5 : Verification

You can verify the state for each docker container by using the

```
$ docker logs runner
```

command or by clicking on the ngrok link

If everything has been configured correctly, you should be able to see the register screen when accessing the link.

Warning !

This type of configuration is only one of the many options that Drone provides. The instructions for use with other languages and tools can be found here

Usage - After Server Setup

Connect Drone to your Github account

Configure Pipelines

Pipelines are used to automate your software delivery process. They are configured via the .drone.yml file, which needs to be placed in the root directory of the project.

You can setup pipelines for any preference. With Docker, Kubernetes , SSH / Exec Commands ...

→ Pipeline types are specified via the **type** tag **(as highlighted below)**

Java Example :

```
kind: pipeline
**type: Docker**
name: java example              #pipelinename
steps:
- name: test                    #name of this step
  image: [Preffered Maven Image]
  commands:
  - mvn install
  - mvn test                    #runs the tests
- name: build                   #name of this step
  image: [Preffered Maven Image]
  commands:
  - mvn install
  - mvn build                   #builds the software
```

Python Example :

```
kind: pipeline
**type : docker**
name: default
steps:
- name: test
image: python
commands:
- pip install pytest
- pytest
```

Seperating multiple pipelines in this document, requires one line of --- before and after every single pipeline

Node Example :

```
---
kind: pipeline
name: node6
steps:
- name: test
image: node:6
commands:
- npm install
- npm test
---
kind: pipeline
name: node8
steps:
- name: test
image: node:8
commands:
- npm install
- npm test
...
```

Explanation : *This file is testing the software in multiple node versions*

`.drone.yml` files are always configured in the same way. At first you need to define that it is a pipeline and what name it has, followed by the deployment steps.

(WARNING : The configuration depends on the pipeline type that is used and will always look different)

Deployment steps always include image and build commands. They can be used in many different ways, for example testing, building, deploying ...

See pipeline reference

Generic .drone.yml files configured for an individual programming language can be found right here

Signing Config Files

Configure Webhooks

The deployment **Trigger** needs to be set up before actually using Drone CI. In our case, Github Webhooks do the job perfectly. I would recommend configuring the Webhooks so that each main commit will be deployed onto Drone.

If everything happens to be setup in the correct way, Drone should execute all steps mentioned in the **.drone.yml** file

Extensions / Plugins

Drone CI has a number of extensions, or plugins, that can be used to add additional functionality and integrations to the platform. Here are some of the most useful extensions for Drone:

1. **Version control system integrations:** Drone provides integrations with popular version control systems, such as GitHub, GitLab, and Bitbucket, allowing you to trigger pipelines and access repository information directly from the platform. **Examples** : Github / Bitbucket / GitLab
2. **Cloud platform integrations:** Drone has integrations with various cloud platforms, such as AWS, Google Cloud, and Azure, allowing you to automate the deployment and management of cloud resources as part of your CI/CD pipeline. **Examples** : AWS(ECS instances/ S3 buckets) / Azure
3. **Notification integrations:** Drone provides integrations with various notification services, such as Slack, email, and SMS, allowing you to receive updates and alerts about the status of your pipelines. **Examples** : Slack- / Discord - notifications
4. **Dependency management:** Drone has built-in support for managing dependencies and resources, making it easy to set up and run pipelines without having to manually configure dependencies and resources.
5. **Code analysis and testing tools:** Drone integrates with a wide range of code analysis and testing tools, such as code coverage[10] tools, static analysis tools, and unit testing[11] frameworks[12], allowing you to automatically run these tools as part of your pipeline.
6. **Custom plugins:** Drone has a plugin system that allows users to create and share custom plugins to add additional functionality to the platform. This can be useful for extending Drone to meet the specific needs of your team or project. **Find more**

Abstract

- [1] Streamline

To streamline means to simplify or improve the efficiency of a process or system by removing unnecessary steps or elements.

- [2] Repository

A repository is a place where data, information, or resources are stored and organized, often in a systematic way.

- [3] API

An API, or Application Programming Interface, is a set of protocols and tools for building software applications. It specifies how software components should interact and provides a way for different systems to communicate with each other. APIs are often used to allow different systems to access the functionality of a software library or web service.

- [4] Shared Secret

A shared secret is a piece of information that is known only to a limited number of people and is used to authenticate the identity of a person or device.

- [5] Image

In the context of Docker, an image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the application code, libraries, dependencies, and runtime.

- [6] CI (Continous Integration)

Continuous integration (CI) is a software development practice in which code changes are automatically built, tested, and integrated into a shared repository on a frequent basis.

- [7] CD (Continous Delivery)

Continuous delivery (CD) is a software development practice in which code changes are automatically built, tested, and made available for deployment on a frequent basis. The goal of CD is to make it possible to deploy code changes to production environments at any time, with minimal manual effort.

- [8] Pipeline

A pipeline is a series of processes or stages through which something passes, typically involving the sequential execution of tasks or the transportation of goods.

- [9] Container

In the context of Docker, a container is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the application code, libraries, dependencies, and runtime. Containers are built from Docker images and provide a portable, isolated environment for running applications.

- [10] Code coverage

Code coverage is a measure of how much of the source code of a software application has been tested.

- [11] Unit testing

Unit testing is a software testing technique in which individual units or components of a software application are tested in isolation from the rest of the application.

- [12] Framework

A framework is a set of libraries, tools, and conventions for building and organizing code in a specific programming language or environment.

-

Sources

Official Documentation

- https://docs.drone.io/

Blogs

- https://kari-marttila.medium.com/using-drone-ci-56383b978490
- https://dev.to/alex_barashkov/getting-started-with-open-source-drone-ci-4pgc
- https://samuelsson.dev/how-to-install-and-configure-drone-ci-on-a-self-hosted-server/
- https://www.docker.com/blog/bring-continuous-integration-to-your-laptop-with-the-drone-ci-docker-extension/

Docker Hub Resources

- https://hub.docker.com/r/cimg/openjdk
- https://hub.docker.com/r/circleci/python

NGrok Installation

- https://ngrok.com/download
- https://dashboard.ngrok.com/get-started/your-authtoken

Official Harness Quickstart Guide

- https://www.youtube.com/watch?v=Qf8EHRzAgHQ