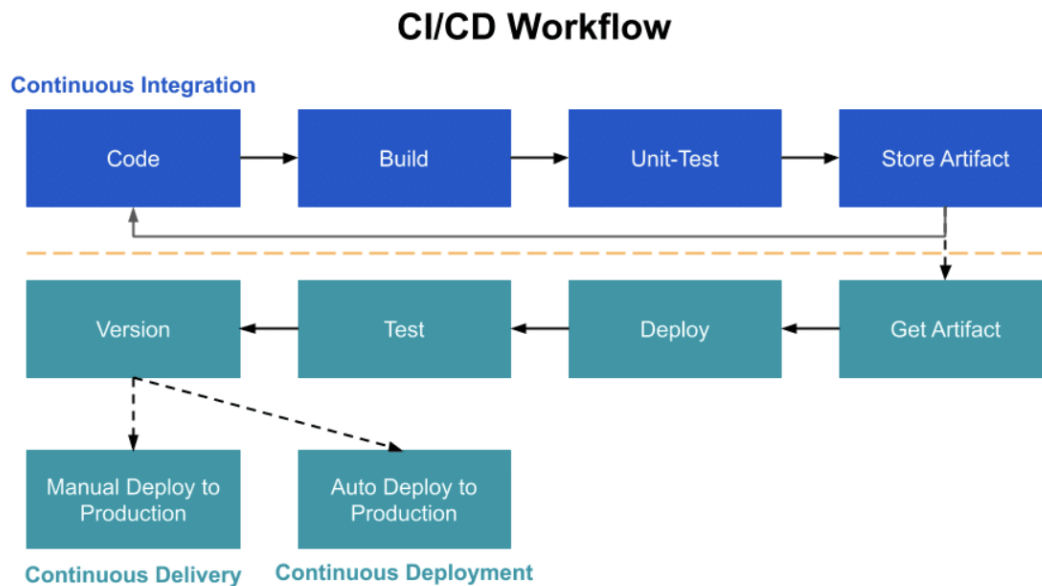


LogDNA and CI/CD



Source: <https://opsani.com/resources/what-is-ci-cd/>

After reading this article, you will be able to answer the following questions:

- What is CI/CD?
- How can centralized logging help me better understand my CI/CD?
- Which monitors and metrics can help me with logging implementation?

Continuous Integration/Continuous Delivery

What Is CI/CD?

As an engineer at any organization, you will encounter CI/CD pipelines. These pipelines enable applications to be updated by deploying the latest code changes which are then manually or automatically deployed to all of your environments. The “CI” in CI/CD stands for Continuous Integration; put simply, this is the practice of automating the integration of code changes from multiple contributors into a single project (which is typically a shared GitHub repository). The “CD” component stands for Continuous Delivery, which equates to high degrees of automation that can help push code into production faster than ever. CI/CD is designed to require less human intervention, since there’s a chance of human error each time humans are involved in a process. By using CI/CD, you can ensure that you have more secure and more efficient pipelines that can be frequently changed or updated on the fly. In order for this to run smoothly and not break everything in sight, testing must be part of your release cycle.

CI/CD Tooling

Even if you are not a “DevOps engineer,” you will still interact with your organization's CI/CD. Fortunately, there are many tools that can

help you. For example, your organization may be using applications like:

- Jenkins <https://www.jenkins.io/>
- Bamboo <https://www.atlassian.com/software/bamboo>
- GitLab <https://about.gitlab.com/>
- CodeShip <https://www.cloudbees.com/codeship/>
- CircleCI <https://circleci.com/>

CI/CD pipeline tools enable you to automate your software delivery process. Without them, you would spend most days running several commands in order to apply your code changes to the applications that your company manages.

When using your CI/CD tool of choice, it's important to remember that these systems are predominantly gated. This means that a feature or release candidate is first released into a lower environment, then run through testing. Finally, a team will either deny or approve the process for deployment into the next environment until it reaches production.

Centralized Logging

Centralized Logging with LogDNA

LogDNA is different from other log management tools in that it aggregates all system and application logs into one centralized system while also using “automatic parsing” and intelligent filters to efficiently query the log lines you need.

How to Set up Centralized Logging with LogDNA

LogDNA is very simple and fast to set up. You'll be able to gather, monitor, alert, parse, live tail, graph, and analyze your logs in a few simple steps:

1. Set up an account [here](#).
2. Select your preferred [log ingestion method](#).
3. [Search](#) your logs.
4. Create [Views](#) and attach [Alerts](#).
5. Create [Boards and Graphs](#).

For a speedy setup, check out the [Quick Start Guide](#). Below, we'll dive into each of these steps in a bit more detail.

Set up an Account

You can set up an account simply by going to the LogDNA [sign up page](#). You will automatically be enrolled in a 14-day free trial. After you've set up your account, you'll be asked for your organization's name. Then, you will have a few different methods for starting your log ingestion.

Log Ingestion Methods

After you have set up your account, you'll need to decide which log ingestion method will work best for you. You'll be provided with an ingestion key as a way to connect, and you'll have several options:

- Use the provided ingestion key.
- Install the LogDNA Agent.
- Install the integrations for your platform(s).
- Use Syslog.
- Use code libraries.

Choose a way to connect

Now, let's get your logs to your LogDNA organization.
Please choose one of the source types below.

Your Ingestion key is:

.....7583



Via Agent

Install our self-updating collector agent, compatible with a wide variety of operating systems.



Gentoo Linux



Linux Debian-based



Linux RPM-based



macOS Server



Windows Server

Via Platform

We also provide integrations for various platforms.



Akamai



Aptible



Cloud Foundry



CloudWatch



Docker



Elastic Beanstalk



Fluentd



Flynn



Heroku



Kubernetes



NXLog



OpenShift



S3

SaltStack

Via Syslog

Perhaps you wish to use plain old syslog.

rsyslog

syslog-ng

syslog

Via Code

For precise control over how you log, you can use our ingest API or one of our code libraries.

{API}

REST API








Libraries

CI/CD

To show you how it all works, we will use CircleCI as our CI/CD tooling of choice. From the UI, we can see that the build has failed. We don't quite know why it failed, so let's set up logging using LogDNA to dig deeper.

build-and-test Failed

Duration / Finished	Queued	Executor	Branch
 2s / 13s ago	 0s	 Docker Medium 	 circle

STEPS

TESTS ●

ARTIFACTS ●

Setting up LogDNA for CircleCI Tool

You'll need to have version 2.1 of CircleCI, so be sure to check your config file and update if necessary. From here, you can search for the LogDNA Orb. If you'd like more detail before we begin, you can review [this page](#).

What Is an Orb?

An [orb](#) is a sharable package of CircleCI configuration that you can use for your builds. You can choose from a public registry or you can create a private orb that can better suit your needs.

The screenshot shows the CircleCI web interface. At the top, the breadcrumb navigation reads: `Open_Street_Map_Cleaning- > circleci-project-setup > config.yml`. To the right are buttons for `Source`, `Compiled`, and `Save and Run`, along with a close icon. The main area is split into two panels. The left panel displays a `config.yml` file with the following content:

```
1 version: 2.1
2
3 orbs:
4   # The python orb contains a set of prepackaged CircleCI
5   # Orb commands and jobs help you with common scripts
6   # so you don't have to copy and paste it everywhere.
7   # See the orb documentation here: https://circleci.com/docs/2.0/orb-commands/
8   python: circleci/python@1.2
9
10 workflows:
11   sample: # This is the name of the workflow, feel free to change it
12     # Inside the workflow, you define the jobs you want to run
13     # For more details on extending your workflow, see: https://circleci.com/docs/2.0/workflows/
14     jobs:
15       - build-and-test
16
17 jobs:
18   build-and-test: # This is the name of the job, feel free to change it
19     # These next lines define a Docker executor: https://circleci.com/docs/2.0/docker/
20     # You can specify an image from Dockerhub or use a local Docker instance
21     docker:
```

The right panel is an overlay titled "What do you want to do? Use Orbs". It explains that orbs are shareable packages of CircleCI configuration. Below this is a search bar with "logdna" entered and a dropdown menu for "Orb Type" set to "Select...". Under the "ORBS" section, the first result is "logdna/logdna@1.0.0", marked as a "PARTNER". The description for this orb is "Send CircleCI Events to LogDNA upon Build". Additional details include "Org usage last 30 days: 0" and "Last modified: Jul 01, 2020". At the bottom of the interface, there are tabs for "LINTER", "DOCS", and "SAMPLE", with "SAMPLE" currently selected. A status bar at the very bottom shows a red exclamation mark icon and the text "build-and-test ↗".

Let's set this up so that we can be notified of the status of a build event from our CI/CD pipeline. We'll need some information from LogDNA to set this up.

Open_Street_Map_Cleaning- > circleci-project-setup > config.yml

Source

Compiled

Save and Run

X

```
1 version: 2.1
2
3 orbs:
4   # The python orb contains a set of prepackaged CircleCI d
5   # Orb commands and jobs help you with common scripting an
6   # so you dont have to copy and paste it everywhere.
7   # See the orb documentation here: https://circleci.com/de
8   python: circleci/python@1.2
9
10 workflows:
11   sample: # This is the name of the workflow, feel free to
12     # Inside the workflow, you define the jobs you want to
13     # For more details on extending your workflow, see the
14     jobs:
15       - build-and-test
16
17
18 jobs:
19   build-and-test: # This is the name of the job, feel free
20     # These next lines defines a Docker executors: https://
21     # You can specify an image from Dockerhub or use one of
```

LINTER

DOCS

SAMPLE

! build-and-test ↗

← notify COMMAND

Notify LogDNA of the Build Event via cURL [Learn more](#)

COMMAND PARAMETERS

* = required

app

Custom App Name

\$(CIRCLE_JOB) (string)

baseurl

Custom LogDNA Ingestion URL

https://logs.logdna.com/logs/i

hostname

Custom Hostname

\$(CIRCLE_PROJECT_REPONAM

logdna-key

The LogDNA Ingestion Key

\$(LOGDNA_KEY) (string)

tags

Comma-Separated List of Tags

circleci (string)

Preview Snippet

You will need to update your configuration file with the code snippets outlined in the next step.

Open_Street_Map_Cleaning- > circleci-project-setup > config.yml

SourceCompiledSave and RunX

```
1 version: 2.1
2
3 orbs:
4   # The python orb contains a set of prepackaged CircleCI
5   # Orb commands and jobs help you with common scripts
6   # so you don't have to copy and paste it everywhere.
7   # See the orb documentation here: https://circleci.com/docs/2.0/orb-commands/
8   python: circleci/python@1.2
9
10 workflows:
11   sample: # This is the name of the workflow, feel free to change it
12     # Inside the workflow, you define the jobs you want to run
13     # For more details on extending your workflow, see: https://circleci.com/docs/2.0/workflows/
14     jobs:
15       - build-and-test
16
17
18 jobs:
19   build-and-test: # This is the name of the job, feel free to change it
20     # These next lines define a Docker executor: https://circleci.com/docs/2.0/docker/
21     # You can specify an image from Dockerhub or use a local Docker image
22     docker:
```

Instructions: notify COMMAND

A command is a single step that runs within a job. To add the orb command to your config, import the orb and add the orb to an existing or new job. [Learn more](#)

INSTRUCTIONS

Step 1

To import the orb into your config, add the orb under the top-level orbs key:

```
orbs:
  logdna: logdna/logdna@1.0.0
```

Copy code

Step 2

Add this orb command as a step in an existing or new job. Make sure that job is included as part of a workflow. Also check that your job is executing the command with the appropriate image:

```
jobs:
  example-job:
    docker:
      # replace with your preferred image
```

Copy code

Back to Orb Search

build-and-test ↗

Search Your Logs

After you have successfully set up LogDNA for CircleCI, you'll be able to view and search your logs with ease. For example:

Alicia Dale

🔍 Everything ▼
🔍 Filters
🔍 Sources ▼
🔍 Apps ▼
🔍 Levels ▼

👤 USER PREFERENCES

🔑 LOG OUT

🏢 ORGANIZATION ▼

🔔 ALERTS

📁 CATEGORIES

🗄️ ARCHIVING

💰 BILLING ▼

⚙️ INTEGRATIONS ▼

🔍 PARSING

📖 TEMPLATE LIBRARY

👤 TEAM

🔑 USAGE ▼

🔧 SUPPORT

```

ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c.1023 msg=op=destroy kind=server
fp=SHA256:d7:c4:25:72:54:23:20:19:73:bd:e4:52:81:ef:d4:02:57:84:fc:2a:c5:35:65:0d:be:0e:02:4a:3e:f6:5a:0a direction=? spid=14640 suid=0
exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO type=SERVICE_STOP msg=audit(1539953193.955:1156945): pid=1 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:init_t:s0 msg=unit=NetworkManager-dispatcher comm="systemd" exe="/usr/lib/systemd/systemd" hostname=?
addr=? terminal=? res=success'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO rm-dispatcher: req:1 'dhcp4-change' [eth0]: start running ordered scripts...
rm-dispatcher: req:1 'dhcp4-change' [eth0]: new request (4 scripts)

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO systemd: Started Network Manager Script Dispatcher Service.
dbus[462]: [system] Successfully activated service 'org.freedesktop.rm_dispatcher'
dhclient[543]: bound to 10.11.5.50 -- renewal in 1581 seconds.

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO systemd: Starting Network Manager Script Dispatcher Service...
dbus[462]: [system] Activating via systemd: service name='org.freedesktop.rm_dispatcher'
unit='dbus-org.freedesktop.rm_dispatcher.service'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2144] dhcp4 (eth0): state changed bound -> bound
2. compute.internal'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2144] dhcp4 (eth0): domain name 'us-west-
2. compute.internal'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2144] dhcp4 (eth0): nameserver '10.11.0.2'
hostname 'ip-10-11-5.50'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2144] dhcp4 (eth0): lease time 3600
gateway 10.11.5.1
plan 24 (255.255.255.0)
address 10.11.5.50

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO dhclient[543]: DHCPACK from 10.11.5.1 (xid=0x567f66eb)
dhclient[543]: DHCPREQUEST on eth0 to 10.11.5.1 port 67 (xid=0x567f66eb)

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.161:1156942): pid=14611 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c.1023 msg=op=destroy kind=server
fp=SHA256:a9:c3:b3:8a:9c:30:ce:da:80:95:98:27:39:62:5d:9a:f3:f1:c1:67:21:77:20:0d:f7:8c:04:a2:70:55:8c:48 direction=? spid=14611 suid=0
exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.161:1156941): pid=14611 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c.1023 msg=op=destroy kind=server
fp=SHA256:e8:92:46:d3:6b:17:eb:b3:a2:a7:0e:43:0d:e9:8c:d3:6c:b2:c3:65:9d:0d:66:22:f9:ed:f2:3e:c0:f0:3b:c3 direction=? spid=14611 suid=0
exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.161:1156940): pid=14611 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c.1023 msg=op=destroy kind=server
fp=SHA256:d7:c4:25:72:54:23:20:19:73:bd:e4:52:81:ef:d4:02:57:84:fc:2a:c5:35:65:0d:be:0e:02:4a:3e:f6:5a:0a direction=? spid=14611 suid=0
exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'

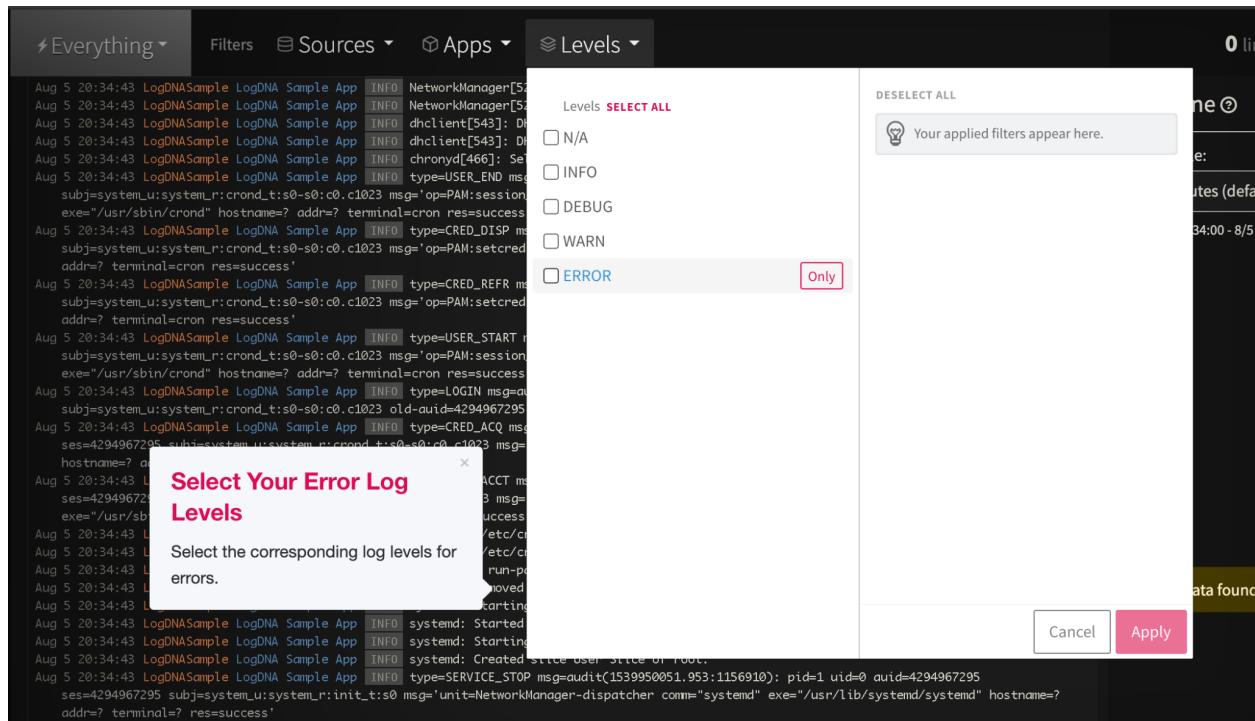
Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO type=USER_ERR msg=audit(1539953067.161:1156939): pid=14611 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c.1023 msg=op=PAM:bad_init grantor=? acct=??? exe="/usr/sbin/sshd"
hostname=ds7978.dedicated.turbodns.co.uk addr=109.104.88.43 terminal=ssh res=failed'

Jul 26 22:08:11 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.158:1156938): pid=14611 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c.1023 msg=op=destroy kind=session fp= direction=both spid=14612 suid=74 rport=40868
laddr=10.11.5.50 lport=22 exe="/usr/sbin/sshd" hostname=? addr=109.104.88.43 terminal=? res=success'

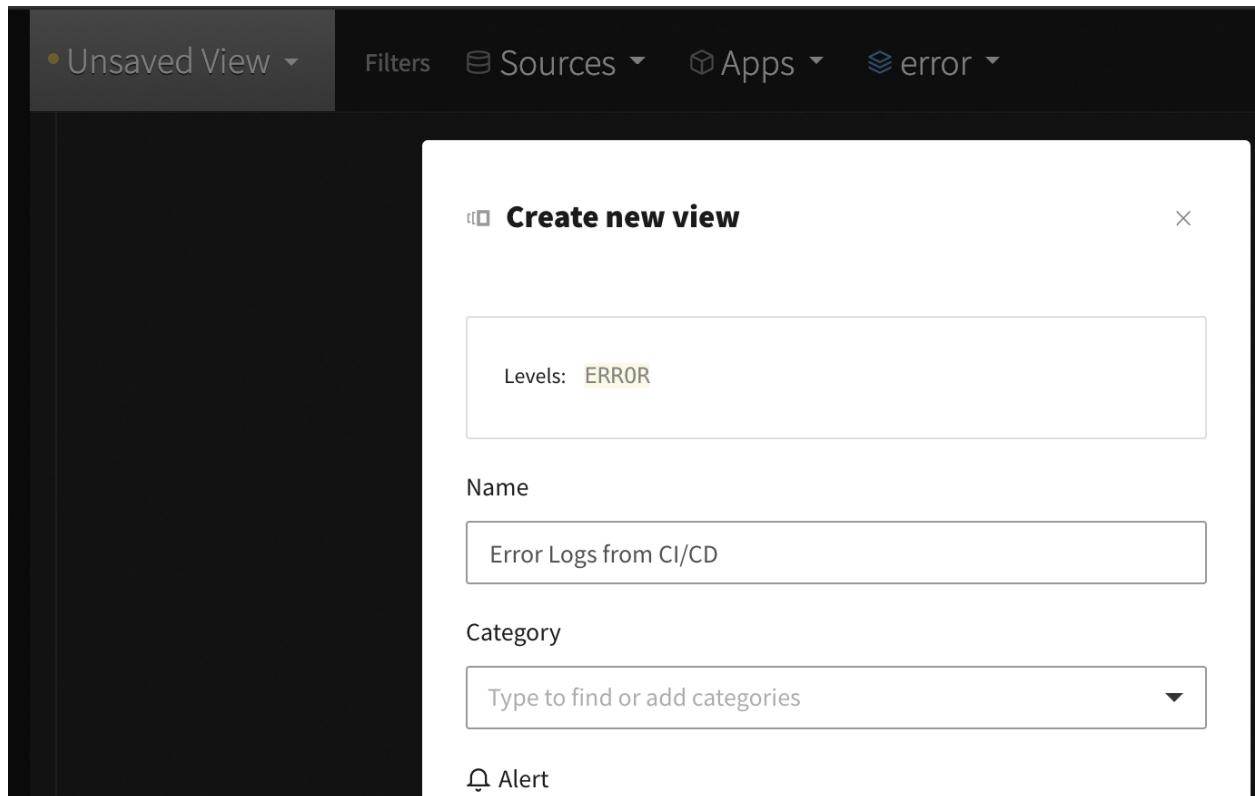
```

Create Views for Your Error Logs

Setting up views with your error logs is super easy! Under the “Levels” tab, you can sort by the log level. If you only select the “Error” logs, you’ll create a view that will only display your error logs.



After selecting the log level error, you can create a new view so that you only see the error logs.

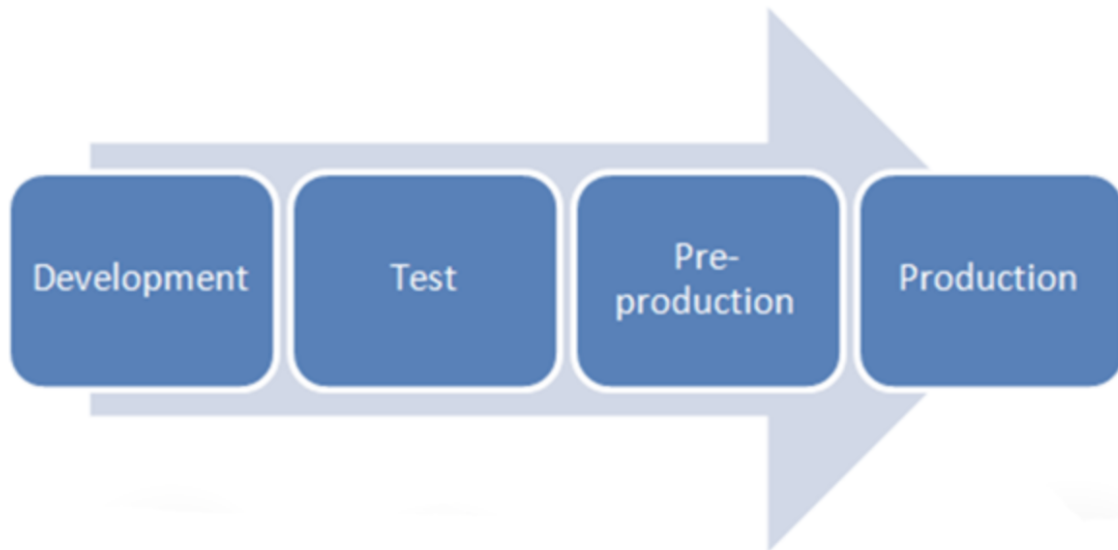


Using Logging for Your CI/CD

Managing and Measuring Your CI/CD

Now that you've (probably) interacted with a CI/CD tool, let's see how we can make it better. More specifically, let's see how we can better measure what success looks like from the CI/CD pipeline.

One of the most important things that centralized logging can do for your pipeline is to give you the ability to watch changes being applied to data as it flows in each environment, from development, to non-production, to production environments.



If you don't have logging set up for your CI/CD pipeline, you won't be able to see how changes affect the environment, and it will be extremely difficult to know how to move forward on the path to a successful production release. Without logs, you won't be able to understand the effects of your code as it moves through the pipeline, which will prevent your team from being able to confirm that your changes are acceptable to move to the next environment in your stack.

Conclusion

You probably don't have time to watch a deployment move through your CI/CD pipeline. A logging system will allow you to see what happened at a specific time that triggered a specific alert on lower environments ("lower" meaning earlier in the pipeline). This level of insight can be a tremendous help when it comes to understanding your workflow and finding the bottlenecks in your pipeline process.