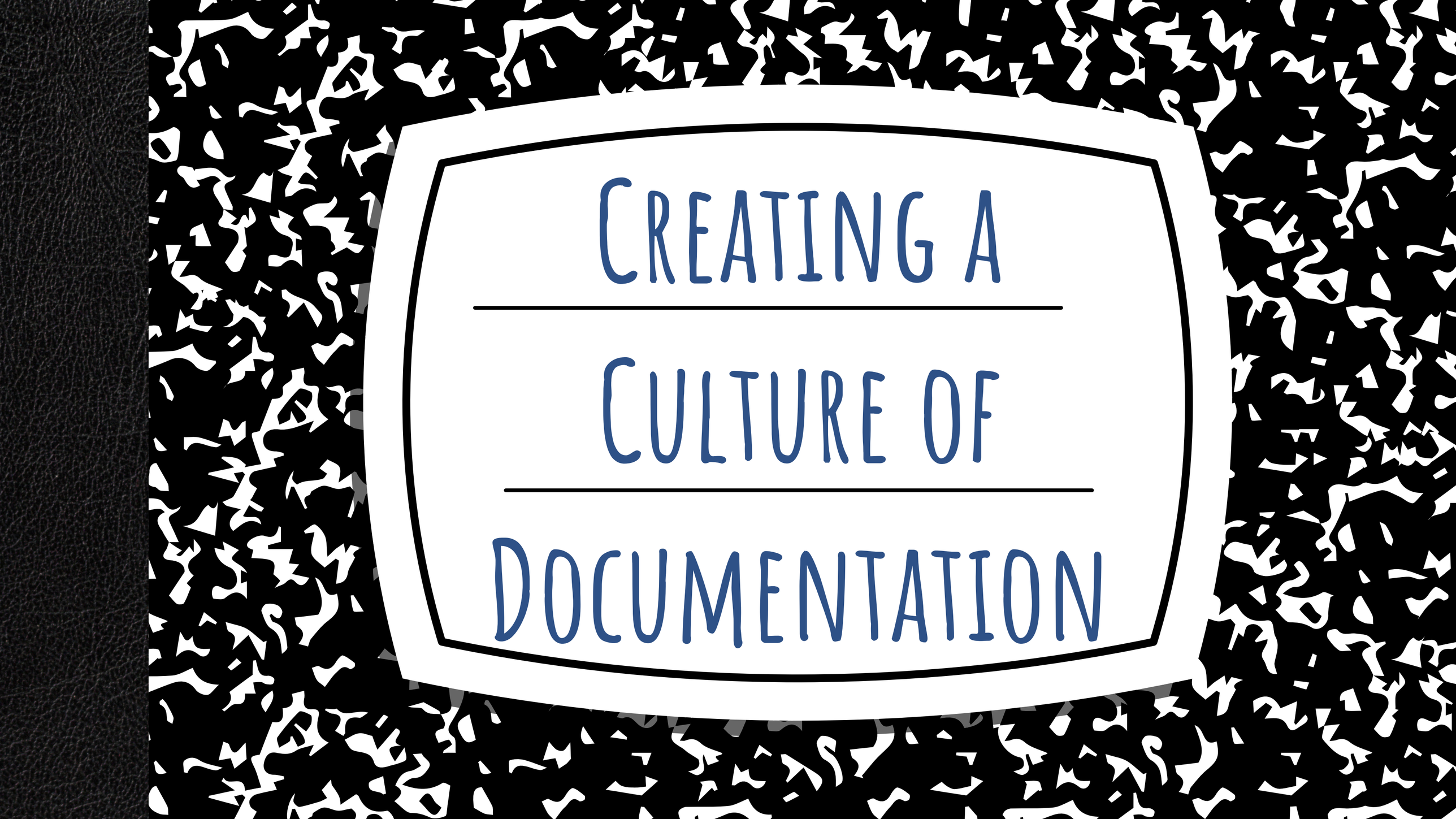

Creating a
Culture of
Documentation

KubeCon EU 2023

Alanna Burke

5:25 PM G106-107

Alanna Burke (she/her)

aburke626

alannaburke@fosstodon.org

alannaburke

Community Manager, Developer Relations, and Documentation

I/O

amazee.io

part of Mirantis

01 What is documentation?

02 Why is it important?

03 What makes good docs?

04 What makes bad docs?

05 What leads to doc failure?

06 What are the consequences of bad docs?

07 Changing your culture

08 What if it doesn't work?

What is Documentation?

What is Documentation? (what does it do?)

Guides users

Explains code

Shows thought leadership

Establishes processes

Onboards new customers

Why is Good Documentation Important?

Why is Good Documentation Important?

Better Onboarding

Faster troubleshooting

Retains users/customers

Improves productivity

Saves time

Manage change

Avoid miscommunication

Why are good docs important?

- 61% of professionals in distributed teams say it can be hard to figure out what their colleagues are working on.
- 44% say siloed digital tools make it hard to spot if work is being duplicated.
- 62% report missing opportunities to collaborate and achieve better results.


"Humans are terrible at data storage"
-Sergio Pereira

What Makes Good Docs?

What Makes Good Docs?

Easy to understand

Gets to the point

Good visuals

Vetted for accuracy

Includes author and date

Addresses the right audience

Search

- Home
- Getting started
- Concepts
  - Overview
    - Kubernetes Components
    - The Kubernetes API
  - Working with Kubernetes Objects
    - Understanding Kubernetes Objects
    - **Kubernetes Object Management**
    - Object Names and IDs
    - Labels and Selectors
    - Namespaces
    - Annotations
    - Field Selectors
    - Finalizers
    - Owners and Dependents
    - Recommended Labels
  - Cluster Architecture
  - Containers
  - Windows in Kubernetes
  - Workloads
  - Services, Load Balancing, and Networking
  - Storage
  - Configuration
  - Security
  - Policies
  - Scheduling, Preemption and Eviction
  - Cluster Administration
  - Extending Kubernetes
- Tasks
- Tutorials
- Reference
- Contribute

Kubernetes Documentation / Concepts / Overview / Working with Kubernetes Objects / Kubernetes Object Management

Kubernetes Object Management

The `kubectl` command-line tool supports several different ways to create and manage Kubernetes objects. This document provides an overview of the different approaches. Read the Kubectl book for details of managing objects by Kubectl.

Management techniques

Warning: A Kubernetes object should be managed using only one technique. Mixing and matching techniques for the same object results in undefined behavior.

Management technique	Operates on	Recommended environment	Supported writers	Learning curve
Imperative commands	Live objects	Development projects	1+	Lowest
Imperative object configuration	Individual files	Production projects	1	Moderate
Declarative object configuration	Directories of files	Production projects	1+	Highest

Imperative commands

When using imperative commands, a user operates directly on live objects in a cluster. The user provides operations to the `kubectl` command as arguments or flags.

This is the recommended way to get started or to run a one-off task in a cluster. Because this technique operates directly on live objects, it provides no history of previous configurations.

Examples

Run an instance of the nginx container by creating a Deployment object:

```
kubectl create deployment nginx --image nginx
```

Trade-offs

Advantages compared to object configuration:

- Commands are expressed as a single action word.
- Commands require only a single step to make changes to the cluster.

Disadvantages compared to object configuration:

- Commands do not integrate with change review processes.
- Commands do not provide an audit trail associated with changes.
- Commands do not provide a source of records except for what is live.
- Commands do not provide a template for creating new objects.

Imperative object configuration

In imperative object configuration, the kubectl command specifies the operation (create, replace, etc.), optional flags and at least one file name. The file specified must contain a full definition of the object in YAML or JSON format.

See the API reference for more details on object definitions.

Warning: The imperative `replace` command replaces the existing spec with the newly provided one, dropping all changes to the object missing from the configuration file. This approach should not be used with resource types whose specs are updated independently of the configuration file. Services of type `LoadBalancer`, for example, have their `externalIPs` field updated independently from the configuration by the cluster.

Examples

Create the objects defined in a configuration file:

```
kubectl create -f nginx.yaml
```

Delete the objects defined in two configuration files:

Edit this page
Create child page
Create an issue
Print entire section

Management techniques
Imperative commands
Examples
Trade-offs
Imperative object configuration
Examples
Trade-offs
Declarative object configuration
Examples
Trade-offs
What's next

Search

- Home
- Getting started
- Concepts
- Tasks
- Tutorials
- Reference
  - Glossary
  - API Overview
  - API Access Control
  - Well-Known Labels, Annotations and Taints
  - Kubernetes API
  - Instrumentation
  - Kubernetes Issues and Security
  - Node Reference Information
  - Networking Reference
  - Setup tools
  - Command line tool (kubectl)
  - Component tools
  - Configuration APIs
  - Scheduling
  - **Other Tools**

Other Tools

Kubernetes contains several tools to help you work with the Kubernetes system.

crictl

An API for container runtimes to integrate with kubelet

`crictl` is a command-line interface for inspecting and debugging CRI-compatible container runtimes.

Dashboard

`Dashboard`, the web-based user interface of Kubernetes, allows you to deploy containerized applications to a Kubernetes cluster, troubleshoot them, and manage the cluster and its resources itself.

Helm

> This item links to a third party project or product that is not part of Kubernetes itself. More information

Helm is a tool for managing packages of pre-configured Kubernetes resources. These packages are known as *Helm charts*.

Use Helm to:

- Find and use popular software packaged as Kubernetes charts
- Share your own applications as Kubernetes charts
- Create reproducible builds of your Kubernetes applications
- Intelligently manage your Kubernetes manifest files
- Manage releases of Helm packages

Edit this page
Third party content advice
Create child page
Create an issue
Print entire section

crictl
Dashboard
Helm
Kompose
Kui
Minikube

Search

- Home
- Getting started
- Concepts
  - Overview
    - Kubernetes Components
    - The Kubernetes API
  - Working with Kubernetes Objects
    - **Understanding Kubernetes Objects**
    - Kubernetes Object Management
    - Object Names and IDs
    - Labels and Selectors
    - Namespaces
    - Annotations
    - Field Selectors
    - Finalizers
    - Owners and Dependents
    - Recommended Labels

information to JSON when making the API request.

Here's an example `.yaml` file that shows the required fields and object spec for a Kubernetes Deployment:

application/deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

One way to create a Deployment using a `.yaml` file like the one above is to use the `kubectl apply` command in the `kubectl` command-line interface, passing the `.yaml` file as an argument. Here's an example:

```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml
```

The output is similar to this:

Edit this page
Create child page
Create an issue
Print entire section

Understanding Kubernetes objects
- Object spec and status
- Describing a Kubernetes object
- Required fields

What's next

of an object specification is the `spec` field for the StatefulSet API. For StatefulSet, the `.spec` field specifies the StatefulSet and its desired state. Within the `.spec` of a StatefulSet is a template for Pod objects. That template describes Pods that the StatefulSet controller will create in order to satisfy the StatefulSet specification. Different kinds of object can also have different `.status`; again, the API reference pages detail the structure of that `.status` field, and its content for each different type of object.

What's next

Learn more about the following:

- Pods which are the most important basic Kubernetes objects.
- Deployment objects.
- Controllers in Kubernetes.
- Kubernetes API overview which explains some more API concepts.
- kubectl and kubectl commands.

Feedback

Was this page helpful?

Yes No

Last modified November 13, 2022 at 9:26 PM PST: updated /kubernetes-objects.md (25aa28ff6a)

GETTING STARTED

This section contains the most basic commands for getting a workload running on your cluster.

- `run` will start running 1 or more instances of a container image on your cluster.
- `expose` will load balance traffic across the running instances, and can create a HA proxy for accessing the containers from outside the cluster.

Once your workloads are running, you can use the commands in the WORKING WITH APPS section to inspect them.

create

Create a resource from a file or from stdin.

JSON and YAML formats are accepted.

Usage

```
$ kubectl create -f FILENAME
```

Flags

Name	Shorthand	Default	Usage
allow-missing-template-keys		true	If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.
dry-run		none	Must be "none", "server", or "client". If client strategy, only print the object that would be sent, without

example

Create a pod using the data in pod.json

```
kubectl create -f ./pod.json
```

Create a pod based on the JSON passed into stdin

```
cat pod.json | kubectl create -f -
```

Edit the data in docker-registry.yaml in JSON then create the resource using the edited data

```
kubectl create -f docker-registry.yaml --edit -o json
```

Twilio Flex

Twilio Flex is a digital engagement center for sales and customer support teams that gives companies control over how they communicate with customers and prospects across all channels, and at every stage of the customer journey. Twilio Flex works within existing solutions to drive revenue in three specific ways. First, it provides a customer service contact center with specific channels and integrations orchestrated to remove friction and drive repeat sales. Second, it enables high touch, contextual direct sales interactions. Third, it facilitates relationship management as your in-app digital concierge.

Get started >_

1 TWILIO SERVERS

2 YOUR APP

```
import React from 'react';
import { withTaskContext } from '@twilio/flex-ui';

const TaskSIDText = {
  color: "#FFF"
};

class CustomCRM extends React.Component {
  render() {
    const { task } = this.props;
    return <div style={TaskSIDText}>
      <p>Access agent task data</p>
      <p>Task SID: <span style={{ fontWeight: 'bold'
    </div>;
  }
}
```

3 Easily deploy and manage your new contact center!

START BUILDING YOUR CONTACT CENTER

DEVELOPER DOCUMENTATION ADMINISTRATOR GUIDE END USER GUIDES

What Makes Bad Docs?

What Makes Bad Docs?

Fragmented across platforms

Incomplete

Out of date

Makes assumptions

Uses jargon

Not tailored to the right audience

Creating and configuring Media Types

Last updated on 13 November 2020

Basic concepts

If you have worked with **nodes** before, you will find some similarities in the basic concepts when dealing with **media entities**. For example, **Media Types** are the equivalent to **media entities** of what **Content Types** are for **nodes**. In other words, the `media` entity (just like any other content entity in Drupal 8) has **bundles**, which for the end user are exposed using the terminology of **Media Types**.

As soon as you enable the **Media** module, some Media Types will be created automatically for you:

- Document
- Image
- Audio file (local)
- Video file (local)
- Remote video

(Note that these are available when using the Standard profile. If you are using another distribution or install profile, other types may be available instead of these ones, or you may need to create your own.)

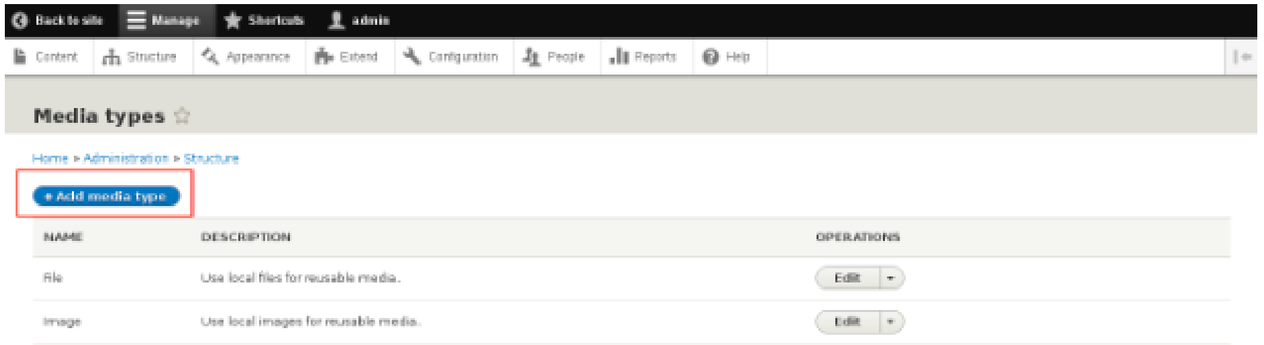
They are ready to use, and if you want to directly create a Media entity you can go to `/media/add/{type_name}` and see how the media entity form looks like.

One important difference of media entities when compared to nodes is that *not all media types are equal*. They are specialized so that they are able to know how to manage the media resource they are dealing with in the best way. For example, media types that deal with *images* will be able to validate file extensions, as well as know how to map width/height image information to Drupal fields. On the other hand, this knowledge won't be necessary or useful to *youtube* media entities, which in turn will probably need to know how to retrieve a thumbnail or the video author information from the remote video server.

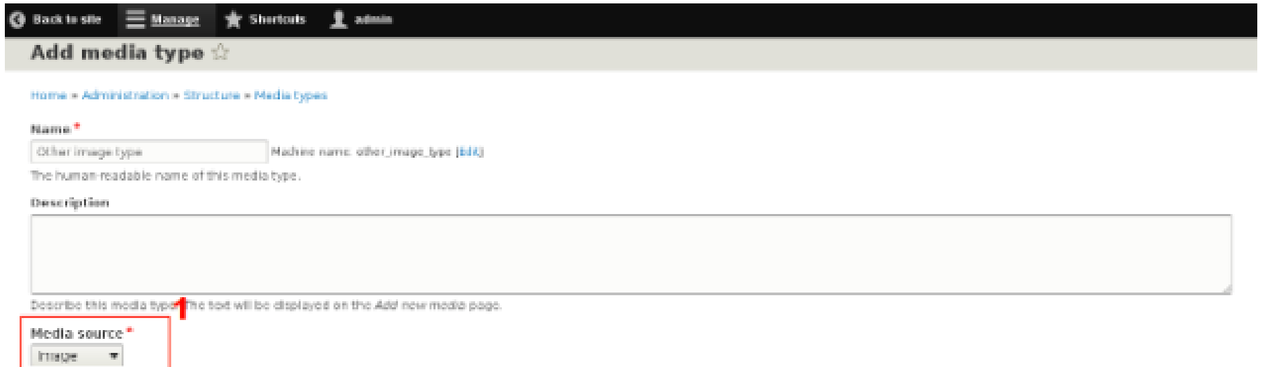
Help improve this

Creating and configuring a Media Type

Just like with content types, you will probably want your site to have additional Media Types, or you may want to edit the existing types Drupal core creates for you. In order to do so, navigate to **Structure -> Media Types**, or visit `/admin/structure/media`:



You will need then to provide the basic information for your media type:



Media module

Media is a drop-in replacement for the Drupal core upload field with a unified User Interface where editors and administrators can upload, manage, and reuse files and multimedia assets.

This guide is about the D8 Media core module. If you are looking for information about the contributed D7 Media module, click here.

Media module overview

A general overview of what the Media module is and its purpose

Creating and configuring Media Types

Basic information about installation and defining Media Types (bundles)

Setting up private access to Media items

Recommendations about handling access restrictions that affect media items

Official Ubuntu Documentation

Documentation developed and maintained by the Ubuntu Documentation Project.

Release	Desktop	Server	Installation Guide
Ubuntu 22.10 (Kinetic Kudu)	HTML	*unpublished*	*unpublished*
Ubuntu 22.04 LTS (Jammy Jellyfish)	HTML	HTML and PDF	*unpublished*
Ubuntu 20.04 LTS (Focal Fossa)	HTML	HTML and PDF	per architecture
Ubuntu 18.04 LTS (Bionic Beaver)	HTML	HTML and PDF	per architecture
Ubuntu 23.04 (Lunar Lobster) Preliminary	HTML	*unpublished*	*unpublished*

Notes:

- As of the Ubuntu LTS release in 2020, the server documentation has moved to a different site, and will automatically update when changes are made to the discourse source code pages.
- The Desktop links above are available in many different languages. They will be displayed in the preferred language specified by your browser. If the preferred language is not available they will be displayed in English. The Serverguide and the Installation Guide are available solely in English.
- Unsupported releases are not shown here. See instructions for upgrading to a supported version.
- Useful information: LTS details and release & end-of-life dates (for all versions).

More Ubuntu documentation (IAAS, Juju, Cloud-init, ...)

What Leads to Doc Failure?

What leads to Doc Failure?

Docs are often everyone's problem and no one's job

No time

No incentive

No experience

Red tape

Information hoarding

Hard to keep up to date

What leads to Doc Failure?

Excuses

"The code is self-documenting"

"It's too hard to keep it up to date"

"It adds too much time to my development"

What Are the Consequences of Bad Docs?

Case Study - Google (2014)

- 48% of Google engineers cited bad docs as #1 productivity issue
- 50%+ of SRE issues cited problems with docs
- Docs were considered everybody's problem, but nobody's job

What Are the Consequences of Bad Docs?

Having to redo work because requirements unclear

Missing info on areas of codebase

Docs are hard to find and waste time

What Are the Consequences of Bad Docs?

If IT folks spend around 20% of their time just looking for information...

- Average salary – $60,000/year
- Average hourly labor rate (fully burdened) – $43/hour
- Time wasted due to bad documentation – 20%
- Annual cost of time waste – $13,760 per employee
- Annual opportunity cost of time waste – $34,400 per employee

BAD DOCS ARE WORSE THAN NO DOCS!

Changing Your Culture


"Culture is not immutable"
-Riona MacNamara

Changing your Culture

Start from the top

Choose someone to drive it

Make it easy

Start with standardization

Empower contributors

Start from the Top

Establish that clear, concise, writing is the expectation from everyone

Higher ups and stakeholders lead by example with quality writing

Choose someone to drive it

Make someone responsible for managing documentation.

This person also creates templates, trainings, and any other support needed to create docs.

Make it easy

Choose tools that integrate into existing workflows

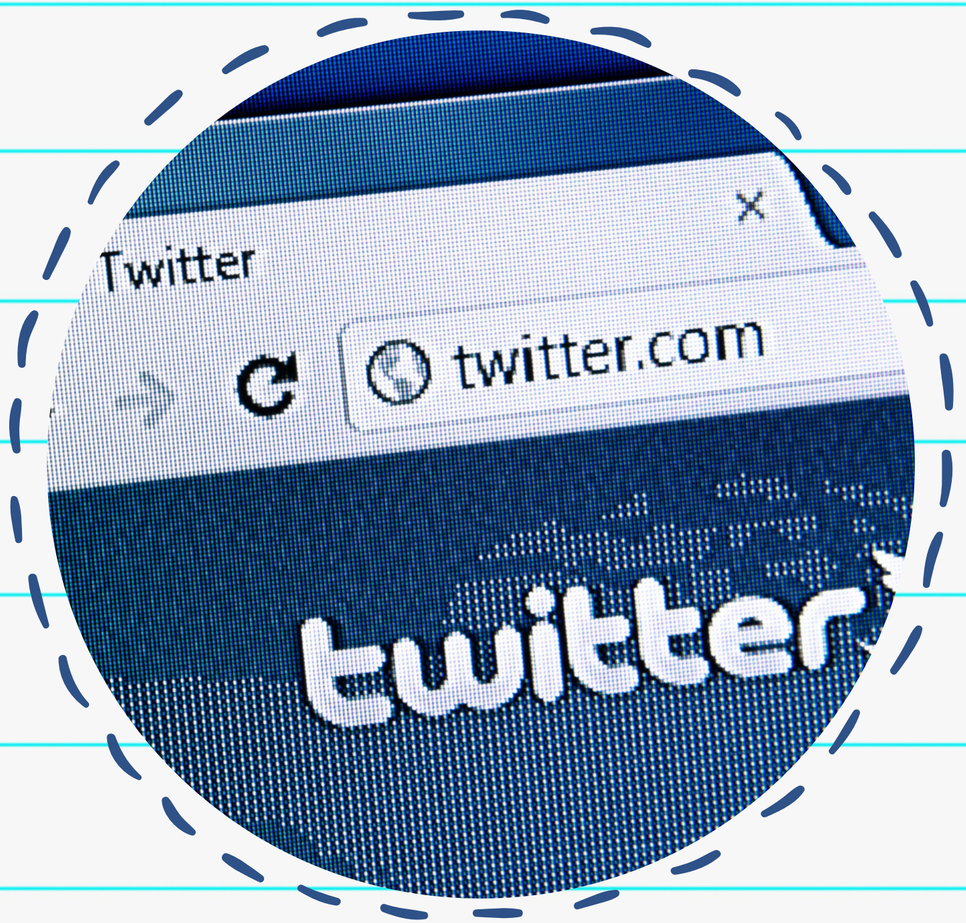
Let devs stay in their editor

Use Git

Ensure everyone has access

Case Study - Twitter (2014)

- Back in 2014, Twitter had 3 technical doc writers.
- Encourage documentation via education and "DocDays"
- Build a new docs-as-code stack
- Create documentation templates



Start with Standardization

- Establish a tool set
- Create a style guide
- Have an approval process
- Create templates
- Use spell checkers and linters
- Establish a maintenance schedule

Documentation Tools

Markdown

ReadtheDocs

mkdocs

GitBook

Confluence

Sphinx

Case Study - Fortune 500 Company

- Helped the client reduce content overload and increased content reuse by 20% YoY
- Helped reduce support calls by 35%
- Guidelines designed for the client were also implemented as the standards for future releases

Empower contributors

Use a system that allows for contribution from everyone.

Create a positive feedback loop.

Give trainings on how to write docs.

Allow everyone to have a sense of ownership over docs.

What if it doesn't work?

What if it doesn't work?

Don't accept merge requests that don't include needed documentation updates.

Make doc writing an official part of everyone's job description

Develop intrinsic value

Questions?

Let me know if you have any feedback for this talk!

Alanna Burke (she/her)

aburke626

alannaburke@fosstodon.org

alannaburke

Community Manager, Developer Relations, and Documentation

amazee.io

part of Mirantis

References

Andre, P. (2023, February 13). Motivating Developers to Care About Documentation. DX. https://getdx.com/best-practices/documentation-culture-engineering

Barney, D, & Oakes, J. (2018, April 26). Building a Documentation Culture. Kaseya. https://www.kaseya.com/blog/2018/04/26/building-a-documentation-culture/

Begly, N. (2021, September 7). How Google, Twitter, and Spotify built a culture of documentation. Doctave. https://www.doctave.com/blog/2021/09/07/how-google-twitter-and-spotify-build-culture-of-documentation.html

Donohue, W. (2018, October 15). 3 Ways to Create a Culture of Documentation. I Done This Blog. http://blog.idonethis.com/3-ways-create-culture-documentation/

Franklin, S, & Gargenta, M. (2014, May 5). TechDocs at Twitter: Creating the Culture of Documentation [Conference session]. Read the Docs, Portland, Oregon, United States. https://www.youtube.com/watch?v=6y4eQ6gYwdU

References

Hinson, B. (2019, August 28). How to Build a Culture of Documentation. Hickam's Dictum. https://hickamsdictum.com/how-to-build-a-culture-of-documentation-534bd75817d6

ITGlue. (n.d) How to Build a Productive Documentation Culture Inside Your MSP. https://f.hubspotusercontent40.net/hubfs/445399/Crewhu-ITGlue-HowToBuildAProductiveDocCulture_Guide.pdf

ITGlue. (2015, April 29). The True Cost of Documentation. https://www.itglue.com/blog/true-cost-bad-documentation/

Kutt, M. (2022, October 28). What is a documentation culture? (And how to create one). Qatalog. https://qatalog.com/blog/post/what-is-a-documentation-culture/

Lau, Y. (2021, June 23). Remote Work: Creating A Documentation-First Culture. Forbes. https://www.forbes.com/sites/forbeshumanresourcescouncil/2021/06/23/remote-work-creating-a-documentation-first-culture/?sh=40bab8295859

MacNamara, R. (2016, July 11-13). The Knowledge: Towards a Culture of Engineering Documentation [Conference session]. SREcon16, Dublin, Ireland. https://www.usenix.org/conference/srecon16europe/program/presentation/macnamara

References

Pereira, S. (2023, January 23). How To Nurture a Culture of Documentation in Remote Teams. Remote Work Academy. https://www.remote-work.io/how-to-nurture-a-culture-of-documentation-in-remote-teams/

Projectindiana. (2021, March 8). Building a Documentation Culture – How to Get Your Staff To Start Documenting. ITGlue. https://www.itglue.com/blog/building-a-documentation-culture/

Qatalog & Cornell University. (2021). Workgeist Report. https://assets.qatalog.com/language.work/qatalog-2021-workgeist-report.pdf

Rabaino, L. (2016, July 29). On building a culture of documentation. Vox Media Storytelling Studio. https://storytelling.voxmedia.com/2016/7/29/12299564/on-building-a-culture-of-documentation

Scott, J. (2019, August 30). How to build a documentation culture. The Startup. https://medium.com/swlh/how-to-build-a-documentation-culture-ffc6a1efa271

References

Thackston, A. (n.d.). Creating a Culture of Documentation. Allison Thackston. https://www.allisonthackston.com/articles/culture-of-documentation.html

Victorino, R. (2020, September 2). How Stripe Built a Writing Culture. Slab. https://slab.com/blog/stripe-writing-culture/

Vieira, T. (2020, August 26). How to build a design documentation culture from scratch. UX Collective. https://uxdesign.cc/how-to-build-up-the-design-documentation-culture-a425ef5ecbf7

Wipro. (n.d.) Reimagining Tech Documentation With a Holistic Approach. https://www.wipro.com/business-process/reimagining-tech-documentation-with-a-holistic-approach/

More talks I've given on docs

Writing Inclusive Documentation

Documentation is like a plant - you need to tend to it!

Documenting an Agile Team - how to keep up

Documentation as a deliverable: the business case for documentation