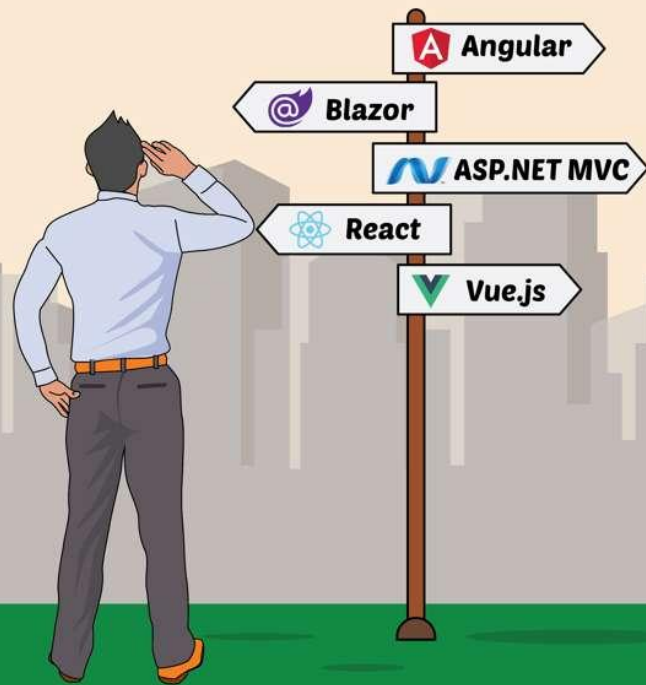


How to choose the stack for your next website



Wednesday, September 21, 2022

Contents

1. Motivation
2. Common types of websites
3. Stacks
4. Example of a static site
5. Example of a blog
6. Example of a client rendering page

1

Motivation

Motivation

Frequently as web developers, we are asked about implementing a new website for a client.

Most of the time, we tend to propose the same solution we have implemented before even if that solution is not the most optimal for the client's needs.

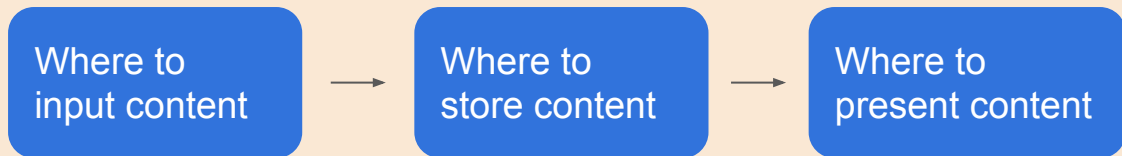
2

Common types of websites

Blogs – news

To educate or inform the visitor about current events or specialized knowledge.

What do we need?



Blogger



Wordpress



Wix



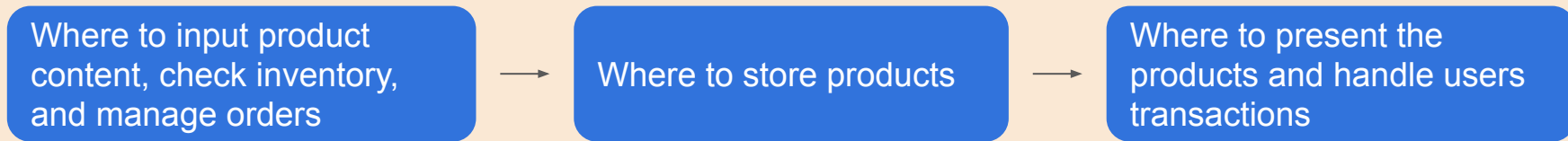
Google Sites

Medium

Medium

Ecommerce

To sell items online with a conventional retail method.



Wordpress



shopify

Shopify



Magento®
An Adobe Company

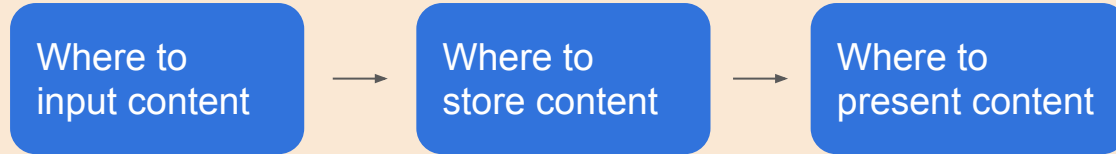
Magento



Woocommerce

Business

To inform prospective clients and consumers about your business and entice them to work with you.



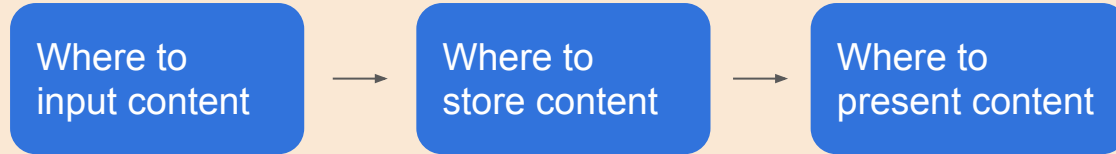
Wordpress



Wix

Portfolio

To display samples of work for certain professionals and attract more clientele.



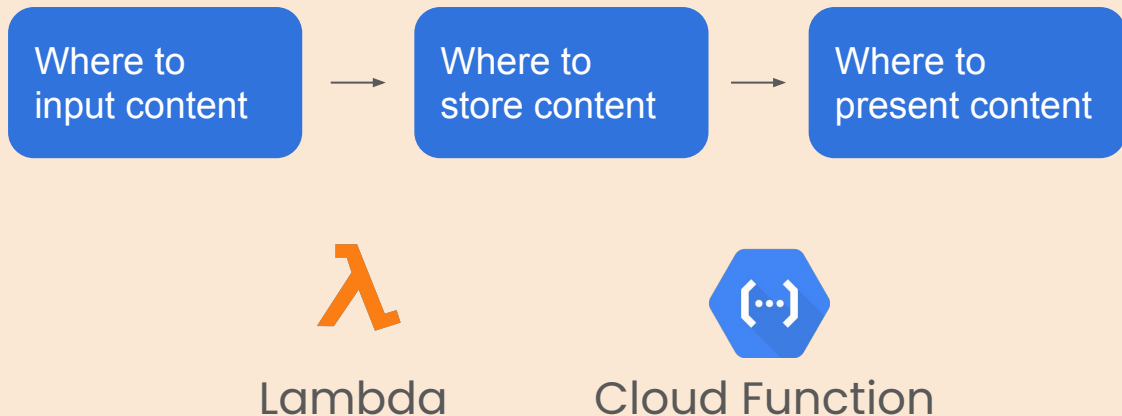
Wordpress



Behance

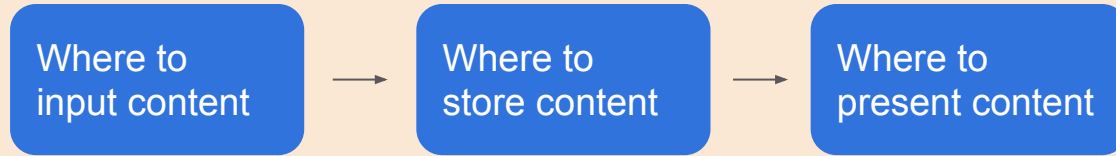
Service provider

To offer a complete online service, such as streaming or online tools like search engines, spell-checkers, photo editors, or translators.



Landing page

To drive customers to a single, specific action, usually as part of a greater marketing campaign.



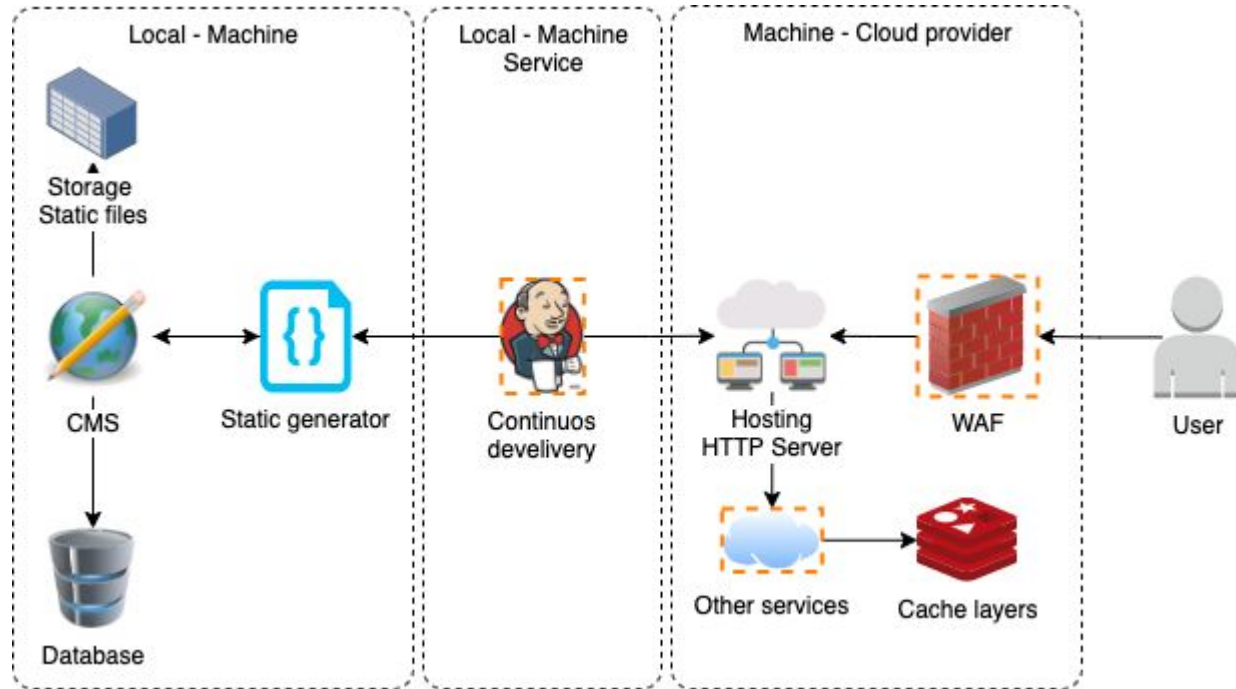
Wordpress



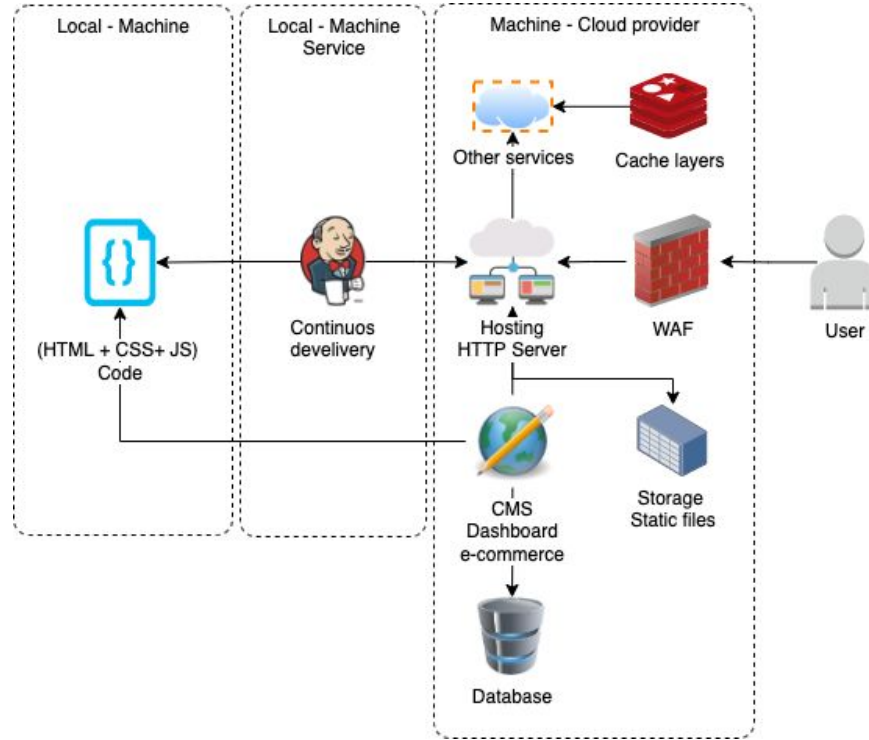
Wix

3

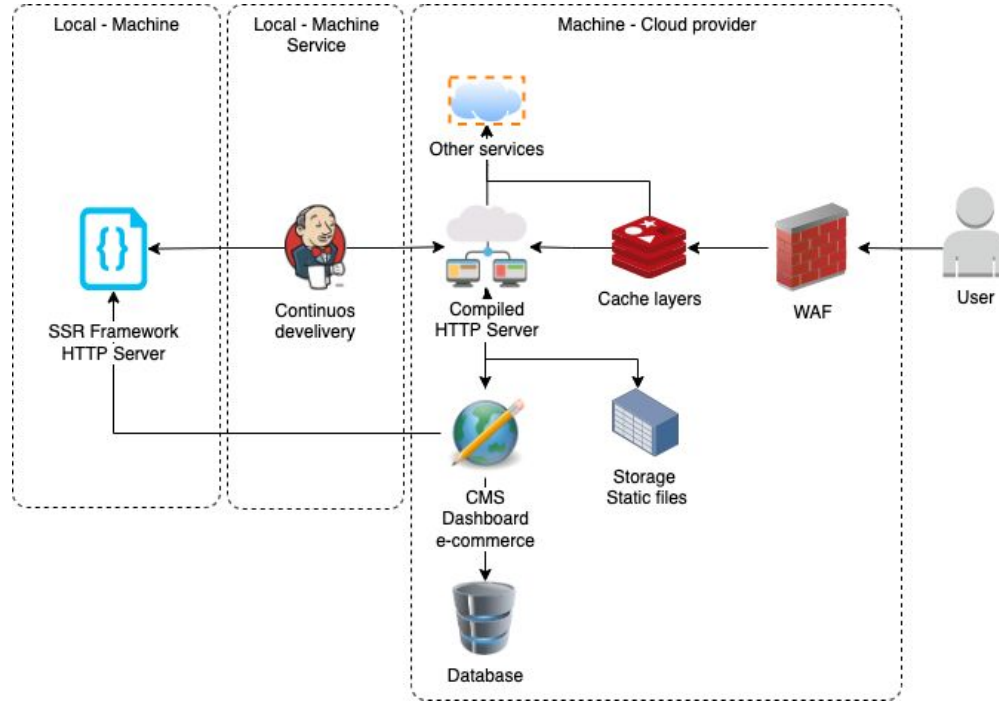
Architectures



Static website



Dynamic websites



Dynamic Content + SEO needs

4

Stacks



Where to input content?

Disk

JSON

Ymal

Po files (translations)

Content

Wordpress

Contentful

Strapi

Prismic

Wagtail

AEM

Cloud database

Firabase

Amplify

...



Database



Other services

Where to store content?

File databases

SQLite

Search engines

Elasticsearch

OpenSearch

Solr

ArangoDB

Database servers

MySQL

Postgres

MongoDb

Cassandra

Cloud databases

Firebase

Dynamodb

Where to input product content, check inventory, and manage orders?

Disk

JSON

Ymal

Ecommerce

Woocommerce

Shopify

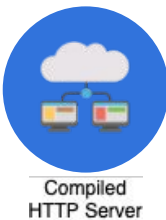
PrestaShop

Magento

Cloud databases

Firebase

Amplify



Where to present content?

Static pages

Pelican

Lektor

MkDocs

Grow SDK

Frozen-Flask

Client side rendering

Vanilla JS

React

Vue

Angular

SolidJS

Svelte

Alpine.js

Lit

Server side rendering

Next

Nuxt

Gatsby

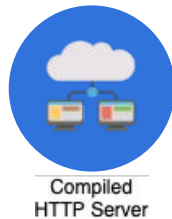
Koa

Express

Django

Flask

Where to present the products and handle users transactions?



Client side rendering

Vanilla JS

React

Vue

Angular

SolidJS

Svelte

Alpine.js

Lit

Server side

Django

Flask

Express

NestJS

5

Examples

Setup



Wagtail

+



Nuxt

Wagtail

1. Create env:

```
python3 -m virtualenv venv
```

2. Install Django and Wagtail

```
pip install Django==3.2.12 wagtail==2.15.5
```

3. Create a wagtail project

```
wagtail start project
```

4. Continue steps in:

<https://wagtail.org/developers/>

Set Wagtail API

1. Add App:

```
INSTALLED_APPS = [  
    # API  
    'rest_framework',  
    'wagtail.api.v2',  
]
```

2. Add api.py config and api path is urls.py:

https://docs.wagtail.org/en/stable/advanced_topics/api/v2/configuration.html

Set Django Storages

1. Install package for GCP:

```
pip install "django-storages[google]"
```

2. Set config in base.py:

```
GS_CREDENTIALS = service_account.Credentials.from_service_account_file("credentials.json")
```

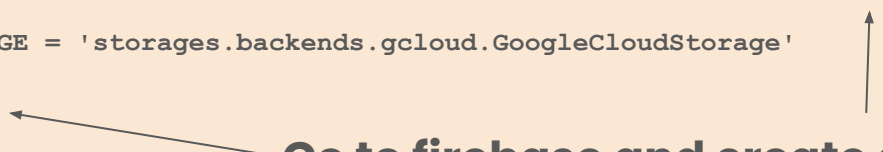
```
DEFAULT_FILE_STORAGE = 'storages.backends.gcloud.GoogleCloudStorage'
```

```
GS_BUCKET_NAME =
```

```
GS_QUERYSTRING_AUTH = False
```

```
GS_DEFAULT_ACL = 'publicRead'
```

Go to firebase and create a new project, a bucket in the storage section and download a service account key from project configurations.



Wagtail API

Custom Pages Api

OPTIONS

GET

GET /api/v2/pages/3/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "id": 3,
  "meta": {
    "type": "core.HomePage",
    "detail_url": "/api/v2/pages/3/",
    "html_url": "http://localhost:8000/",
    "slug": "python-medellin",
    "show_in_menus": false,
    "seo_title": "",
    "search_description": "",
    "first_published_at": "2022-09-21T20:07:36.795872Z",
    "parent": null,
    "relative_url": "/"
  },
  "title": "Python Medellin",
  "main_header": "Meetup Python Medellin",
  "hero_image": {
    "xs": {
      "url": "https://storage.googleapis.com/python-meetup-363200.appspot.com/images/clean_505480862.width-375.jpg",
      "width": 375,
      "height": 210,
      "alt": "Python Medellin"
    },
    "sm": {
      "url": "https://storage.googleapis.com/python-meetup-363200.appspot.com/images/clean_505480862.width-600.jpg",
      "width": 600,
      "height": 338,
      "alt": "Python Medellin"
    },
    "lg": {
      "url": "https://storage.googleapis.com/python-meetup-363200.appspot.com/images/clean_505480862.width-1024.jpg",
      "width": 1024,
      "height": 580,
      "alt": "Python Medellin"
    }
  }
}
```

Nuxt

1. Use nvm or something to lock Node version:

```
lts/gallium
```

2. Create a Nuxt App:

```
yarn create nuxt-app frontend
```

Recommended options:

JavaScript, Yarn, UI framework: None, Axios, All Linting tools, Jest.

Rendering mode: Universal

Deployment target: Static (static, dynamic) / Server(SSR)

Continuous integration: None

Git

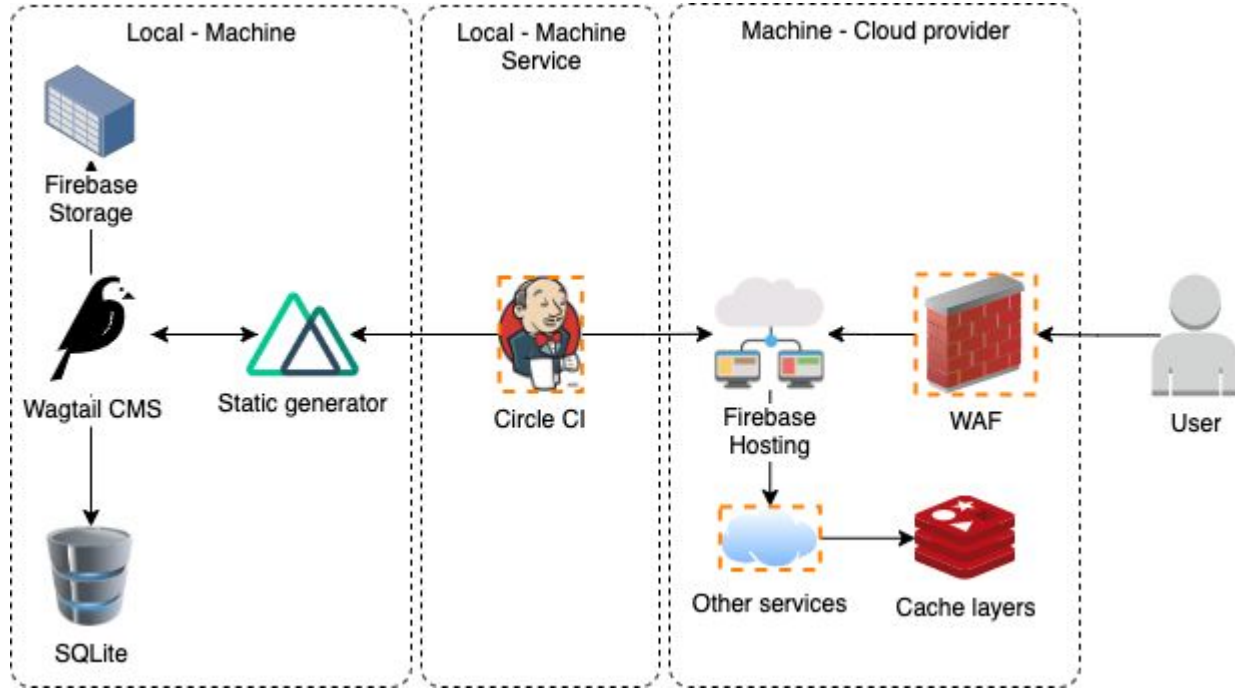
Static Website

Settings in nuxt.config.js:

```
target: 'static',
```

Repo:

[Talks - 22-09-21 - Static Website](#)



Static website

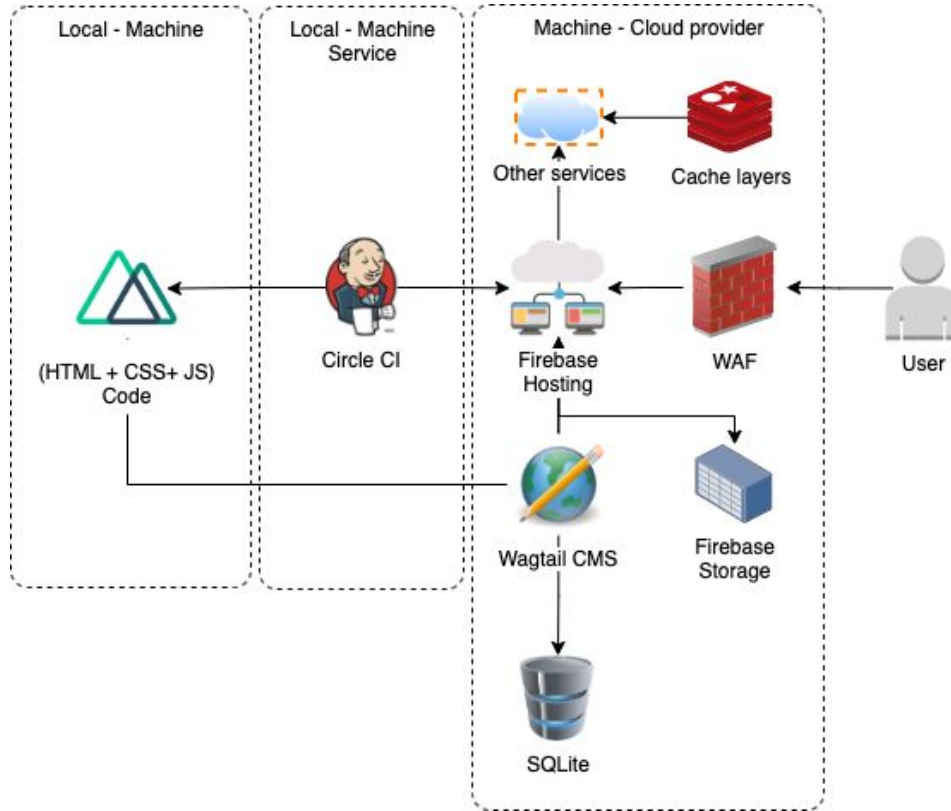
Dynamic Website

Settings in nuxt.config.js:

```
ssr: false,  
target: 'static',
```

Repo:

[Talks - 22-09-21 - Dynamic Website](#)



Dynamic websites

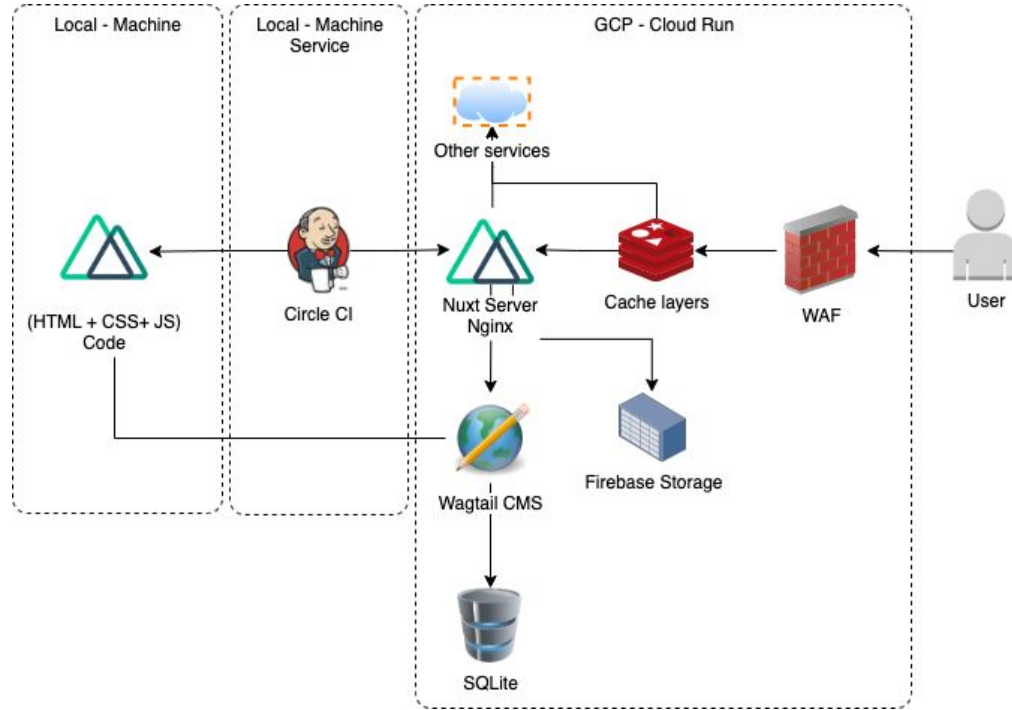
SSR Website

Settings in nuxt.config.js:

```
ssr: true,  
target: 'server',
```

Repo:

[Talks - 22-09-21 - Server Side Website](#)



Dynamic Content + SEO needs