Infrastructure

Think about how much, what types and what specification of resources you have.[1]

- Consider containers rather than VMs as they're lighter-weight and can share resources more readily.

- Consider Spot VMs ([Azure](#), [GCloud](#), [AWS](#)) for batch tasks to use capacity that would otherwise be wasted (and save money).

- It's better to have one VM running at a higher utilisation than two running at low utilisation rates.

- Can you reduce the specification of your machines?

Considerations for data centres

- Electricity sources for chosen data centres (this may vary over time – see below).

- Other considerations include data centres' use of energy storage and on-site renewables, and approach to cooling and e-waste.

- Read more in this white paper by Schneider Electric: [Guide to Environmental Sustainability Metrics for Data Centers](#).

📢 *Advocate for the environment – ask the cloud service providers you use about their green credentials.*

Finding a good green web host can be hard, but the author of *Designing for Sustainability* recommends [this Green Hosting Directory](#).

CI/CD infrastructure

- Minimise the total number of deployed environments.[2]

- Scale down kubernetes applications when not in use (or try [kube-green](#) to automate it) – can you turn off workloads outside of business hours?

Consider these Sustainable Kubernetes resources from sustainable-computing.io.

- Kepler (Kubernetes-based Efficient Power Level Exporter) uses eBPF to probe energy related system stats and exports as Prometheus metrics.

- PEAKS (Power Efficiency Aware Kubernetes Scheduler) uses metrics exported by Kepler to help Kubernetes schedule to improve energy efficiency by placing Pods on optimal nodes.

- CLEVER (Container Level Energy-efficient VPA Recommender) uses metrics exported by Kepler to recommend Vertical Pod Autoscaler the resource profiles to improve energy efficiency by running workloads.

Think about where your content is

- Use Content Delivery Networks so data is transferred less far.

- CDNs can also help with caching – with something like [Google Hosted Libraries](#), the user might already have the library you're using cached from another site.

- Consider where different aspects of your compute are located to maximise network utilisation within the same cloud and region.

---

[1] https://patterns.greensoftware.foundation/catalog/cloud/match-utilization-requirements-of-vm/
[2] https://patterns.greensoftware.foundation/catalog/cloud/minimizing-deployed-environments

Compute

Run workloads less often

- For example, GitHub's *Dependabot* pauses pull request generation for inactive repositories.

- Give users control over this so they can opt to run things less often – e.g. a "low power mode" which fetches data less often.

Move to asynchronous workloads

- Moving synchronous work to job queues can flatten out resource usage and avoid spikes.

- This makes it easier to adopt demand-shifting (see box).

Reduce the amount of computation needed to deliver a webpage

- Prefer static over dynamic pages where possible.

- Cache dynamic content.

Use data to drive decisions

- Use a tool like Prodfiler, Cloud Carbon Footprint, or the Green Metrics Tool to understand what is running, how much carbon it's using and how that can be reduced.

- Microsoft Azure Emissions Impact Dashboard.

- Client-side, try the CPU timeline (WebKit) or the Performance tab (Chrome). In Firefox, try Power profiling to measure watts!

Seek out more efficient frameworks

- E.g. Quarkus for running Java Virtual Machines.

Embrace demand-shifting[3]

- Demand-shifting is altering when or where computing tasks are undertaken so as to reduce their carbon intensity.

- Choose a greener region in your cloud processing provider.

- Run asynchronous tasks at times of lower carbon intensity.

- Use a service such as the Carbon Aware SDK to retrieve the data you need for this.

Avoid handling requests in certain circumstances

- Implement request throttling for APIs to avoid unnecessary load on systems.

- Use the circuit-breaker pattern to prevent your app making requests which are likely to fail.

- Read more about code-level trade offs between Quality of Service and carbon reductions in this paper: Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation.

Reduce your CI/CD compute

- Run expensive test suites less often.

- Avoid unnecessary constant rebuilds.

- Fail fast when appropriate.

- Offer a "skip CI" option for changes that don't require tests to run (WIP commits, docs, etc).

🔒 *Consider terminating TLS at your border gateway to reduce CPU usage.*

Interested in making Machine Learning more sustainable? See Green Software Patterns for AI.

---

[3] https://hackernoon.com/our-code-is-harming-the-planet-we-need-carbon-aware-design-patterns

Remove unnecessary code and content

Simplify designs

- Involve designers in thinking about sustainability and performance, not just developers.

- Do "Page Briefs" when designing: Why does the page exist?; What is its conversion goal?; What is the minimum amount of content that users need to see to make that conversion?; What is the list of supporting information?[4]

- Remove unnecessary or unused product features.

Do a content audit

- For starters, look for redundant webpages and duplicated files.

- Use analytics to see what content is actually used.

- Conduct experiments to see what content is actually effective – content that doesn't help you achieve your goals isn't needed.

- Consider user data in your audit – get rid of redundant or "orphaned" user content.

Make shorter videos

Skip the long intro and get straight to the point. If you can get your five-minute video down to three minutes, you just did everyone a favour. With video making up so much of online data, this is good for both the viewer and the planet.

Andy Crestodina, quoted in *Designing for Sustainability*.

Wordpress

Developer Danny van Kooten analyzed the carbon emissions of several WordPress plugins he created and found that removing a 20 KB JavaScript dependency from the Mailchimp for WordPress plugin—installed on two million websites—would reduce emissions by 708 tons per year.

Sustainable Web Design, *referencing* CO2 emissions on the web.

- Remove unused WordPress plugins.

- Consider the WP-Optimize Cache plugin to cache your site, clean your database and compress your images.

- Consider disabling pingback and trackback functionality to streamline the amount of work your site does tracking who's mentioned your posts.

- Turn off comments where you don't need them.

See more in Chapter 6 of *Designing for Sustainability* (which also includes tips for Drupal).

Remove unused code…

- unused libraries or framework code.

- unused font file variants.

- unused CSS (try the DevTools coverage tool).

✨ See http://microjs.com for "Fantastic Micro-Frameworks"! ✨

---

[4] *Designing for Sustainability*, ch 5

Store less

Reduce image storage

- Choose appropriate file formats.[5]

- Don't store images at a higher-resolution than necessary.

- Reduce colour variation in images or blur the background to allow for increased compression.

- Don't forget to compress your favicon. It might be small, but it'll be requested a lot!

- If your users can upload images, allow them to crop them as part of the process.

Make use of "flat" assets—images that contain large swaths of flat color rather than gradients, bevels, or other effects—in your design work. File sizes for CSS buttons and vector-based SVG files are smaller than their raster-based counterparts, meaning less data to download. Plus, flat images can be scaled more easily in a responsive design.

Designing for Sustainability, chapter 5.

📺 For more about images, watch *"The Joy of Optimizing Images" by Una Kravets – An Event Apart video*
📄 You might also like *ClimateAction.tech*'s article Create Low-Carbon Images.

Reduce font storage

- WOFF2 files are typically much smaller than TTF (how to convert).

- If embedding fonts, remove unnecessary elements (like dingbats) with a tool like fonttools.

- Consider using system fonts rather than custom ones.

Reduce video storage

- Compress video, and don't serve at higher quality than required.

- If your video is just a talking head, consider mono audio. Your viewers will never notice!

Set storage retention policies

- This includes your log data too![6]

- If you need historic data, consider storing aggregates or averages rather than the full past data set.

Store at the right "temperature"

- Store frequently-accessed data in "hot" (fast access) storage to avoid the processing overhead of retrieval, but store infrequently accessed data in "cold" storage as that has a lower carbon impact for the storage itself.[7]

For more on this topic, see Gerry McGovern's World Wide Waste.

If you're starting a new data project, try the University of Loughborough's Digital Carbon Footprint Toolkit.

---

[5] https://patterns.greensoftware.foundation/catalog/web/deprecate-gifs, https://patterns.greensoftware.foundation/catalog/web/serve-images-in-modern-formats
[6] More about logging and sustainability: https://learn.microsoft.com/en-gb/azure/architecture/framework/sustainability/sustainability-security#security-monitoring. And on AWS, switching your compression algorithm to ZSTD could reduce storage by 30%.
[7] Source for AWS, source for Azure.

Transfer less

As a rule of thumb, the more data transferred, the more energy used in the data center, telecoms networks, and end user devices. ([Sustainable Web Design](#))

Avoid transferring content at all

- Don't poll for updates; use [WebSockets](#) or [Fetch](#) with a persistent connection instead.
- Implement appropriate security measures to eliminate unnecessary network traffic.[8]
- [Avoid serving unnecessary content to search engines](#)
- [Cache content on users' devices](#).
- Use cookie-free (sub)domains for static components, such as images, that don't require cookie requests.
- Minimise HTTP headers – only include those you need.
- Reduce DNS lookups by increasing DNS TTL.[9]

Don't serve content until it's needed

- Don't auto-play videos, and [import on interaction](#).
- Defer offscreen images.
- [Transfer appropriately-small images for a user's device](#).

Consider compression and serialisation

- Compress what you transfer.
- Minify Javascript and other assets.
- Consider appropriate serialisation formats for API messages.[10]

Reduce the length of user journeys on your site

- Visiting fewer pages should mean lower carbon impact as less data is transferred and less work done by your servers and the user's device.
- Findability of content (on search engines and internal search) also reduces the carbon impact of users bouncing around to different pages or sites looking for what they need.
- Think about this in your Content Strategy.
- Design your APIs carefully to reduce the number of requests for a given operation.

Here are some questions to answer [about your internal search]: Can the search field be used effectively on mobile devices? Does it properly interpret search queries that contain nonstandard characters? Do search queries include content that exists in third-party systems, like shopping carts, CRMs, content management systems plug-ins, or donation engines? If your onsite search engine can't perform these tasks, you're frustrating users and wasting time and energy.

Designing for Sustainability, ch 4.

😎 [CO2.js](#) *is a JavaScript library that allows developers to estimate the emissions associated with their apps, websites and software. At its core, CO2.js takes an input of data, in bytes, and returns an estimate of the carbon emissions produced to move that data over the internet.*

See also Google's [PageSpeed Insights](#) tool.

---

[8] [Related Green Software pattern](#); [Bot control in AWS](#); [Bot control in Azure](#)

[9] [https://kinsta.com/blog/reduce-dns-lookups/#tip-2--change-ttl-values-to-take-advantage-of-dns-cache](#)

[10] [https://learn.microsoft.com/en-gb/azure/architecture/framework/sustainability/sustainability-application-design#improve-api-efficiency](#)

Help your users make a difference

Be an influence for good on your users' real-world behaviour

Think about behavioural nudges related to the software's touch-points with "real world" behaviour. For example, a route-planning app encouraging low-carbon travel modes.

Highlight sustainable choices

Help users manage their impact on your software's computation and storage load. For example:

- Not running unnecessary computational tasks.

- [Turning off camera](#) to avoid data processing.

- Deleting their unnecessary data.

Reduce the energy impact of users' devices

- Ensure that your app works on old devices so that it's not an additional pressure on users to incur the carbon costs of getting a new device.

- Don't play animations when they're not visible to users.[11] This happens automatically with declarative animations (CSS Animations and Transitions),

- Timers (e.g from `setTimeout` & `setInterval`) can wake the CPU to run. Try to combine into fewer, infrequent timers.

- Use the [Page Visibility API](#) or [blur events](#) to prevent inactive pages using CPU for things like timers and UI updates)

- Check out this [Catalog of Energy Patterns for Mobile Applications](#) and [Green Software Lab's tools for Android developers](#).

See also [How Web Content Can Affect Power Usage](#) on the WebKit blog, from which some of the above is taken.

Try this – *Ecograder*

Automated analysis can only get you so far, but the most thorough sustainability checker for websites I've found so far is https://ecograder.com/

This document was last updated in September 2024.

---

[11] https://webkit.org/blog/8582/intersectionobserver-in-webkit/
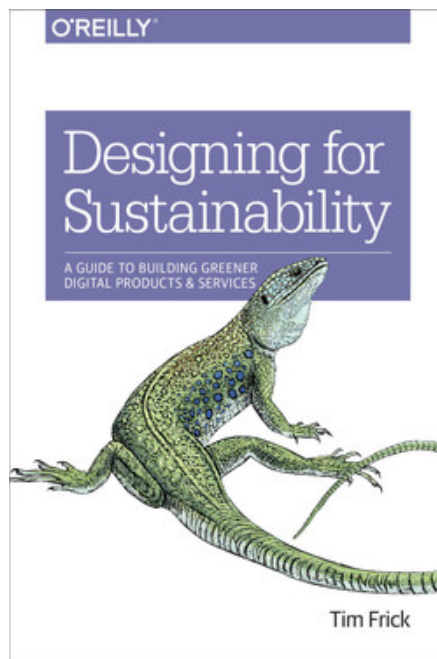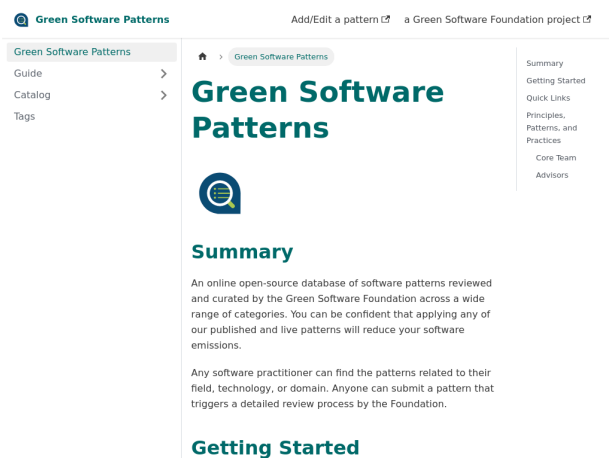
Resources

This document is a summary of the following resources, with subsequent additional research:
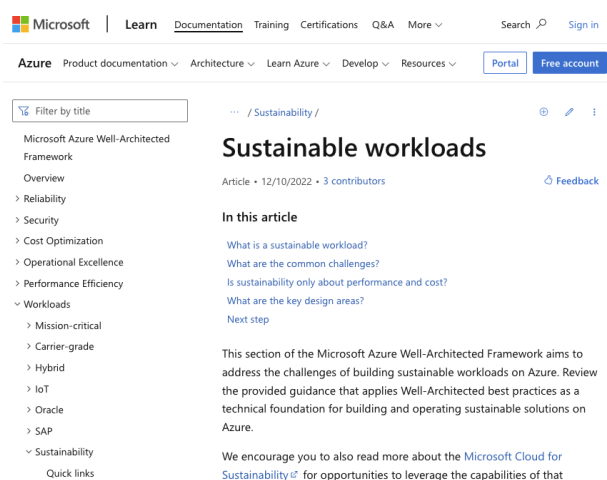

[Sustainable Web Design](#) by Tom Greenwood.

[Designing for Sustainability](#) by Tim Frick.

[Green Software Patterns](#) by the Green Software Foundation.

Azure [Sustainable workloads](#) documentation.[12]

See also

- [SustainableUX](#) online event and newsletter.

- [20 ways to make your website more energy efficient](#).

- Greenpeace's [Clicking Clean: Who is winning the race to build a green internet?](#)

---

[12] The advice for AWS is at:
https://docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/sustainability-pillar.html ([linter here](#)).