**Style guide is now in Confluence: **

https://hashicorp.atlassian.net/wiki/spaces/CCAS/pages/3378151440/Style+guide

last updated December 2024
For any questions, feedback or comments, please reach out to Corporate Communications

Table of contents

IBM acquisition wording update

Company branding and descriptions:

We will continue to use our current company brand wording and descriptions exactly as we have, with one change:

- When writing about the company, on your first reference to HashiCorp, add the descriptor “an IBM company” in text.

Product branding and logos:

- Continue to use our product names and logos as you have in materials that we are creating.

1. Overview

This guide is for anyone who creates or distributes public-facing content on behalf of HashiCorp. This includes the HashiCorp corporate blog (and any blogs affiliated with HashiCorp), corporate social media channels, websites, email copy, newsletters, collateral such as white papers, and external presentations. Care should be taken to ensure voice, language, and content consistency throughout all mediums and channels of communication, while also keeping in mind the appropriate tone for a given channel. Generally, HashiCorp follows the AP Stylebook rules and guidelines, unless HashiCorp specific style is indicated.

- 2020 AP Stylebook

There are several department-specific style guides that provide direction that you should follow *in addition* to what's in this one. If you work in the Events team or write technical content on the Engineering, Product, or Solutions Engineering teams, you'll want to review these as well.

- Tutorial Style Guide - HashiCorp Developer / Education team (ask helpdesk if you need access)
- Engineering Style Guide (requires GitHub access - if needed ask in the #engineering slack channel)
- Events Verbal Identity, Messaging, and Style Guide
- HashiCorp Acronyms (separate list)

2. Voice of HashiCorp

When writing site copy, company blog posts, emails, ads, white papers, and other content not written in a first-person PoV, you're writing as an extension of the company and representing our brand.

Maintaining a consistent brand **voice** and **personality** are important to make all of our interactions feel authentic.

- *Words that describe our voice:*
  - Clear, concise, uncomplicated
  - Professional, but conversational and human
  - Energetic, confident, engaging
  - Concrete, based on real-world examples and offering usable takeaways
  - We talk like the audience we're talking to
- *Words that describe our personality:*
  - Friendly, kind
  - Authentic, trustworthy, sincere

    - Thoughtful, attentive
    - Focused, visionary, ambitious
    - Well-informed, evidence-based claims
    - Helpful, opinionated advisors, teachers, and guides who provide actionable advice
    - Humble, we don’t claim perfection or to be the “best” without evidence
- *Words that **don’t** describe us:*
    - Fluffy, jargon-heavy, salesy, corporate
    - Kitschy, wacky
    - Formal, stuffy, stiff, robotic
    - Know-it-all, arrogant
    - Dull, generic, apathetic
    - Vague
    - Dismissive, condescending (i.e. towards competitors)

Tone sets your writing’s mood and can change depending on the type of content you’re creating. For example, we may use humor in an ad while a white paper might take a more thoughtful tone.

- *Words that describe our tone:*
    - **For long-form content (white papers, blogs, etc.):** Frank, straightforward, matter-of-fact, engaging
    - **For short-form content (web, emails, ads, social, video scripts, etc.):** Energetic, fun, witty, (slightly more) informal

Writing tips for capturing our company voice

- **Get to the point:** Give the bottom line up front. Give the reader a clear idea of what they'll learn in this content in the first few sentences or paragraphs. Then give the reader any necessary context as quickly and succinctly as possible. Then you can dig into the bottom line more deeply.

- **Know your audience**. Use words they would actually use. **Talk like they would**.

- **Put yourself in your reader’s shoes.** Are you addressing what your intended audience wants to know? What level of knowledge will your intended audience have with HashiCorp solutions and terminology? Learn about your audience and research what they like reading about, and what their challenges are.

- **Education before promotion:** Provide original, valuable information to the reader. Promote only when relevant in the content or at the end when you’ve already provided the value.

- **Make your title and content match (pages, blogs):** Deliver what you promise in the title and description. Tell the truth. If the title is too long (90 characters max, but 65 or less preferred to fit in Google results), make it shorter by focusing on the most important element(s). You don't need every detail in the title, but don't leave it cryptic. Start with a title that helps you focus your writing, then go back and **finalize your title *after* you've written the content.**

- **Don't capitalize everything:** It can be tempting, especially when talking about new features or writing marketing copy, to capitalize every feature or important word. Don't. Use lowercase unless a specific term requires capitalization. Read our capitalization guide section and our word list to help get this right.

- **Make it easy (long-form content, blogs):** Guide the reader through a well-organized, scannable article, paper, or post. Avoid presenting a "wall of text". If you have 3+ long paragraphs in a row, reread and scrutinize that area and see if you can:
    - Add subheds
    - Shorten the content
    - Break it up into smaller paragraphs or bullets/numbered lists
    - Bold/italicize key points and/or add links on key words and ideas
    - Do the hard work to make it simple for readers (ideas here)(and here)

- **Don't repeat yourself:** If you've said something once, don't repeat the same information in a different way. This commonly happens in marketing content because there is a reflex to repeat selling points or increase the length of the content. **No fluff, just stuff.**

- **Don't be generic with titles and topics.** Be substantive, specific, and opinionated. Include benefits when possible.

- **Choose clear, everyday wording** over longer, rarely used academic terms.

- **Provide evidence** (you can use links on text) for any non-obvious claims.

- **Use concrete examples:** They are much more powerful than general or abstract claims. (Show, don't just tell.)

- **Avoid generic, obvious, stuffy, fluffy language, marketing speak.** The best way to convey the HashiCorp voice is to be clear, relying on simple language and ideas that are fleshed out and connected.

- **Write the way you talk** (minus the non-grammatical, long-and-convoluted sentences). Think of your reader as someone you're chatting with on a conference floor, describing tools and discussing ideas. Be conversational but direct, to the point, and more concise than typical speech. Use concrete metaphors, similes, or analogies that you might use in a conference conversation or presentation.

- **Don't pack in too many adjectives** (It's tempting to include every possible great thing, but remember - *less is more* and simple is usually better). Adverbs can often be dropped to make writing tighter. Provide examples and/or replace descriptors with action verbs to make your writing more impactful and direct:
  - Generic: "HCP is a powerful, unified platform"
  - Specific: "HCP gives complete visibility over your digital estate, with automated workflows that streamline developer processes"

- **Showing is better than telling**. Lean on concrete customer and industry experiences and real-world examples whenever possible.

- **Don't be afraid to have fun and be energetic** with your writing, particularly in shorter-form content (like organic **social** and **paid ads**), and show some personality. Alliterations, contractions, references to pop culture and recent events, emojis, and *some* wordplay (mainly for ads and emails, and sometimes display text on longer-form content) can make content more relatable and enjoyable. It's a balance though. Be careful about crossing into overly informal language and avoid being obscure, too "hip," or trying too hard to be funny. Consider your audience, their location and native language, and whether they'll understand a particular reference (for example, using programming memes may not work for BDMs).

- **Vary your sentence structure** and pace your writing. Content should follow a rhythm. Don't use too many long sentences in a row, and short sentences are generally preferable to long ones. Occasionally using a short sentence that's just a few words long can add punch to your writing.

- **Use the active voice,** where the subject of the sentence does the action. In passive voice, the subject of the sentence has the action done to it.
  - <u>Passive</u>: It is recommended to configure VCS access when first setting up an organization.
  - <u>Active</u>: We recommend configuring VCS access when first setting up an organization.

  - <u>Passive</u>: To support this behavior, a provider framework has been built into Terraform.
  - <u>Active</u>: Terraform has a provider framework to support this behavior.

  - <u>Passive</u>: Next, the service will be registered
  - <u>Active</u>: Next, Kubernetes will register the service.

- **Address the reader as “you” and don’t walk through steps saying “we.”** Use “I” and “you” *not* “we” and “our” if you're stepping through instructions (*unless* you’re speaking as the company).
    - Right: “In this example, *I’m* going to do this” or “Next, add this code to *your* application.”
    - Wrong: “Now *we’re* going to take *our* code and put it in *our* application. Next, *we* will run this command.”

      In imperative instructions, you can often be more direct by omitting the subject entirely. ("Next, go to the settings page.") It’s okay to address the reader directly. Indirect writing can *feel* more polite as you're writing it, but it's harder to understand and it wastes the reader's time. Good example.

- **Words we don’t want to overuse**
    - Leverage (instead say ‘use’)
    - Utilize (use)
    - Enable (other options: allow, help, let)
    - Challenge
    - Solution
    - Unique (means the only one of its kind, often mistakenly conflated with “uncommon.” Can't be somewhat or very unique, it either is or isn't)
    - Innovative / innovation
    - Disrupt, disruption
    - Seamless (other options: low-friction, low-effort)
    - Robust
    - “We are excited/pleased to announce…”
    - Ensure (be careful that it really does ensure something, it often merely supports or encourages something)
    - Empower
    - Accelerate (speed up, faster)
    - Velocity (fast, speed)
    - Value (define what the value is)
    - Success (define what that success means or is)
    - Achieve

In that same spirit, here’s a perfect example of our voice in action:
https://www.linkedin.com/pulse/platform-teams-phases-cloud-adoption-david-mcjannet/

Identifying the audience

We generally write for three audiences:

- **Business decision makers (BDMs)**

    - Typical positions: CEO, CTO, CISO, CFO, etc.
    - These high-level executives make budget decisions and sign off on large-scale product adoption.
- **Technical decision makers (TDMs)**
    - Typical positions: Platform Team Lead, Head of Cloud, Head of DevOps, VP of Engineering, Security Team, Finance/Procurement, Compliance/Audit (and more, this is a non-exhaustive list)
    - These leaders and managers make decisions about tooling and architecture with the help of practitioners. They often have prior experience with the deeper side of tech, so don't treat them as non-technical.
- **Practitioners**
    - Typical positions: Operations, Application Development, DevOps Engineers, Platform Engineers, Site Reliability Engineers, Cloud Engineers
    - These are the people at the ground level writing code and building and operating systems.

BDMs and TDMs are the default audience for **PMM, Revenue Mktg, and most of marketing**. **Practitioners** are the default audience for **Developer Relations**.

Our principles in our marketing voice and style:

HashiCorp's messaging and voice are based on our core values and should be reflected in our writing:

1. **Integrity** - Be straightforward, accurate, and honest in your writing. Don't be vague or ambiguous to get around an uncomfortable truth. Our content is factual and accurate to the best of our ability. We confirm facts with sources.
2. **Kindness** - Put yourself in your reader's shoes. We have our own goals for content, but they won't matter unless the content aligns with what readers want. Do the hard work to make their experience better. Be an upstanding member of the industry and our community in our interactions.
3. **Pragmatism** - Learn and use the realities of human psychology and the ways people read content (think readability, scannability) Make content easy to understand.
4. **Humility** - Limit the use of terms like "leader"/"de facto"/"lingua franca" to describe our company and products, (use only with Terraform and Vault, where true), and if used, include a link to data that supports any claim. Don't claim to be the best unless you can back it up.
5. **Vision** - Be opinionated, with reasoning and data to support our vision of how organizations should do things. Don't be afraid to be prescriptive, and use phrases like "We strongly believe" "We promote" or that something "represents the best-known way".

6. **Execution** - Make sure content delivers on what the title and introduction promise and imply. Make sure it’s as clean, clear, precise, and accurate as possible.
7. **Communication** - Communication is not just one-way. We need to be aware and not tone-deaf to what customers and the broader community and world are saying..
8. **Beauty works better** - ‘Beautiful’ writing works better, just like well-designed websites or code. No piece of copy is too big, too small, or under the radar to receive a discerning review that might improve the flow and impact of the content.
9. **Reflection** - Testing and reviewing past and current work (and ideas) with a critical eye for improvement and areas for innovation.

3. HashiCorp descriptions and key terms

HashiCorp company internal terms and acronyms

Tagline(s)
Do cloud right

Mission statement
At HashiCorp, we build the infrastructure that enables innovation. Our suite of multi-cloud infrastructure automation products are the underpinnings of the largest enterprises in the world, who rely on our solutions to run, secure, deploy, and connect their mission-critical applications to deliver essential services, communications tools, and entertainment platforms to the world. We’re building a once-in-a-generation infrastructure company with a unique approach: rather than focusing on technologies for their own sake, we build workflows designed to solve real-world problems.

Boilerplate
About HashiCorp
HashiCorp is The Infrastructure Cloud™ company, helping organizations automate multi-cloud and hybrid environments with Infrastructure Lifecycle Management and Security Lifecycle Management. HashiCorp offers The Infrastructure Cloud on the HashiCorp Cloud Platform (HCP) for managed cloud services, as well as self-hosted enterprise offerings and community source-available products. The company is headquartered in San Francisco, California. For more information, visit hashicorp.com.

All product and company names are trademarks or registered trademarks of their respective holders.

1 sentence, 25-word, 50-word company descriptions

[Only use the “™” in very formal contexts]

One-sentence description

HashiCorp helps organizations automate cloud environments with The Infrastructure Cloud™ on the HashiCorp Cloud Platform (HCP).

25-word description
HashiCorp is The Infrastructure Cloud™ Company, helping organizations automate multi-cloud and hybrid environments with Infrastructure Lifecycle Management and Security Lifecycle Management.

50-word description
HashiCorp is The Infrastructure Cloud™ Company, helping organizations automate multi-cloud and hybrid environments with Infrastructure and Security Lifecycle Management. Our suite of products — Terraform, Vault, Consul, Nomad, Boundary, Packer, and Waypoint, along with the HashiCorp Cloud Platform — underpin the most important applications for the largest enterprises in the world.

Our company social media bios
Twitter/X
HashiCorp is The Infrastructure Cloud™ company, helping organizations automate multi-cloud and hybrid environments with Infrastructure Lifecycle Management and Security Lifecycle Management.

LinkedIn
HashiCorp is The Infrastructure Cloud™ company, helping organizations automate multi-cloud and hybrid environments with Infrastructure Lifecycle Management and Security Lifecycle Management. HashiCorp offers The Infrastructure Cloud on the HashiCorp Cloud Platform (HCP) for managed cloud services, as well as self-hosted enterprise offerings and community source-available products.

About Infrastructure Cloud

[New section to go here]

Product descriptions

Terraform - HashiCorp Terraform is the world's most widely used multi-cloud provisioning product. Whether you're deploying to AWS, Azure, Google Cloud, other clouds, or an on-premises datacenter, Terraform can be a single control plane, using infrastructure as code for infrastructure automation, to provision and manage all your infrastructure including servers, databases, load balancers, caches, firewall settings, SSL certificates, queues, monitoring, subnet configurations, routing rules, and almost every other aspect of your infrastructure.

HCP Terraform - HCP Terraform is the fastest way to adopt Terraform, the world's most widely used multi-cloud provisioning product. Offered as a service, HCP Terraform provides everything practitioners, teams, and global businesses need to create and collaborate on infrastructure and manage risks for security, compliance, and operational constraints.

Terraform Enterprise - Terraform Enterprise provides everything to collaborate on infrastructure and manage risks for security, compliance and operational constraints in a self-hosted and managed platform for data localization and privacy, operational policies around data retention, or use cases for a single-tenancy.

Vault - HashiCorp Vault provides the foundation for modern multi-cloud security. It was purpose-built in the cloud era to authenticate and access different clouds, systems, and endpoints, and centrally store, access, and deploy secrets (API keys, credentials, etc.). It also provides a simple workflow to encrypt data in flight and at rest.

Vault Ent, HCP Vault Secrets, HCP Vault Radar

HCP Vault Dedicated - As a fully managed service available for AWS, HCP Vault Dedicated is a way to centrally store, access, and deploy secrets (API keys, credentials, etc.) for workloads across EKS, EC2, AWS Lambda, and other AWS services. HCP Vault also provides a simple workflow to encrypt data in flight and at rest.

Consul - HashiCorp Consul provides a foundation for cloud networking automation by connecting and securing services across any runtime platform and cloud. It provides service discovery and L4/L7 traffic routing for applications, automates manual networking tasks by dynamically reconfiguring network infrastructure as services scale and down, and provides secure service-to-service communication using a service mesh and through identity based security policies and mTLS encryption.

Consul Enterprise,

HCP Consul - As a fully managed service available for AWS, HCP Consul is the easiest way to enable secure service networking and service mesh for workloads across EKS, EC2, AWS Lambda, and other AWS services, and to connect AWS environments to other cloud environments and private datacenters.

Nomad - HashiCorp Nomad is a simple and flexible orchestrator to deploy and manage containers and non-containerized applications across on-premises and cloud environments at scale. Developers can easily define and schedule applications without worrying about underlying infrastructure. Operators can efficiently deploy, scale and expand flexible workloads across any infrastructure without complex reconfiguration.

Boundary - HashiCorp Boundary is a secure remote access solution that provides an easy way to allow access to applications and critical systems with fine-grained authorizations based on trusted identities. Across clouds, local data centers, and low-trust networks, Boundary provides an automated and secure way to protect and safeguard access to infrastructure by trusted identities without exposing the underlying network.

Boundary Ent, HCP Boundary

Waypoint - HashiCorp Waypoint is an application deployment tool that aims to deliver a PaaS-like experience for Kubernetes, ECS, and other platforms. Developers can deploy, manage, and observe their applications with real-time status updates and monitoring through a consistent abstraction of the underlying infrastructure. Teams can extend workflows via built-in plugins and an extensible interface that includes custom builders, deployment platforms, registries, release managers, and more.

HashiCorp Consul Service (HCS) on Azure - HashiCorp Consul Service (HCS) on Azure enables a team to provision HashiCorp-managed Consul clusters directly through the Azure Marketplace and easily leverage Consul's **service discovery** and **service mesh** features within their Azure Kubernetes Service (AKS) or VM-based application environments.

HashiCorp Cloud Platform (HCP) - The HashiCorp Cloud Platform is a fully managed platform offering HashiCorp products as a service to automate infrastructure and security on any cloud. HCP enables teams to focus on building cloud native applications and migrating critical workloads to the cloud faster with fewer resources. *(Note, spell out as the HashiCorp Cloud Platform on first reference, but no need to use "the" when abbreviating as HCP on subsequent reference.)*

Sentinel (is a feature, never use HashiCorp in front of it) : Sentinel is an embeddable policy as code framework to enable fine-grained, logic-based policy decisions that can be extended to source external information to make decisions.

Vagrant: Vagrant is a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.

Packer: Packer is an open source tool for creating identical machine images for multiple platforms from a single source configuration. Packer is lightweight, runs on every major operating system, and is highly performant, creating machine images for multiple platforms in parallel.

How we reference products

In formal or highly visible external communications like the corporate blog, social media posts, white papers, etc. “HashiCorp” should be used in front of the product name in the *first* reference (e.g. “HashiCorp Vault” not just “Vault”). If you’re referencing two products with an “and” only put HashiCorp in front of the first item. (e.g. “HashiCorp Vault and Consul”).

When referring to the Enterprise version of any of our core products, capitalize Enterprise. Terraform Enterprise, Vault Enterprise, Consul Enterprise, Nomad Enterprise.

The free, non-commercial, source-available, core versions of our products can often just be referenced by their basic name “HashiCorp Vault”, “Terraform”, etc. If you’re comparing them to the Enterprise or HCP commercial versions, you would refer to them as “Vault Community” (or “Vault Community Edition,” if we need to be extra clear). We no longer refer to them as “Consul open source” or “Nomad OSS”.

For multi-word products (e.g. Vault Enterprise, Terraform Cloud) you should use HashiCorp in front of the first usage as noted above if possible, but in documents where the HashiCorp context is obvious (partway through a HashiCorp newsletter, you can choose to omit “HashiCorp,” even on a first usage). Be sure to put “HashiCorp” in front for very formal documents, like press releases or white papers, especially in places where the name might be quoted out of context. No “HashiCorp” is needed in front of HCP or HCP Vault/TF/etc.

Call them products or offerings. Avoid using the word “tool” or “tooling”

When to use project vs. product
Project: This refers to one of our community offerings (e.g. the Waypoint project, not the Waypoint product).
Product: Use this to refer to one of our offerings with a commercial version (e.g. products like Vault and Terraform).

Don’t use “HashiStack”. Instead use “HashiCorp stack”

Avoid unofficial abbreviations such as TF, TFE, TFC, TFC4B, TFCB, COM, and more. The abbreviation HCP (HashiCorp Cloud Platform) is fine and encouraged after spelling out on first use.

Capitalization guide

Capitalization rules for products and features

- **Capitalize product names** as proper nouns ("Terraform," "Consul").

- **Feature names should not be capitalized** unless it can be used as a stand alone product. When in doubt, look up the term on [HashiCorp Developer]() and see how the term is used in one or two tutorials (sometimes there are differences, so check our [word list]() below too.

  Right: "Run `terraform apply`." (use backticks around `terraform apply` to make our CMS render it as inline code)

  Wrong: "Run Terraform Apply."

***Don't* capitalize generalized features or internal constructs as proper nouns**.

Right: "Terraform Cloud includes a private module registry."

Wrong: "Terraform Cloud includes a Private Module Registry."

Right: "A plan represents the execution plan of a run."

Wrong: "A Plan represents the execution plan of a Run."

Right: "We'll use the Vault namespaces feature."

Wrong: "We'll use the Vault Namespaces feature."

Capitalization rules for UI elements
Typically you will want to capitalize site or UI elements or sections when referring to them in content if they are capitalized in the UI:

Right: "You can set up module sharing from the Site Administration area of a Terraform Enterprise instance. The Registry subsection allows an organization to be chosen to which your current organization's private module registry will be shared."

Exceptions: Some terms may seem like they wouldn't be capitalized, but the Education team has given reasoning for their capitalization:

- Terraform Registry
- Terraform Stacks
- Raft algorithm
- Vault Agent
- Vault Proxy
- Boundary Client Agent / Client Agent (when referring to the Boundary Client Agent)
- Terraform Plugin Framework
- Nomad Autoscaler
- The Infrastructure Cloud
- Infrastructure Lifecycle Management

- Security Lifecycle Management

The general capitalization guidance (feature names should not be capitalized unless it can be used as a stand alone product) applies when writing content about new features as well.

When to use italics or links instead of capitalization

Some product terms will use italics on the first usage because it might be confused with a more general word. A good example: Consul-Terraform-Sync has a primitive called *tasks*. So when you mention this primitive for the first time in a piece of content, you would not capitalize it but you would italicize it. For more information on when to use italics and more detailed info on technical term grammar, skip down to this section of our Tutorial guidelines. Another option is to add a hyperlink to an introduction documentation page for the term upon first usage.

Other topic definitions and usage

In marketing copywriting, you'll often have to describe and understand a number of recurring technical terms relating to our products. Here is a list of many of the most common, and their definitions.

(Don't capitalize these terms when using them in the middle of a sentence in the content you are writing—we don't like to create or over-emphasize buzzwords with capitalization. A frequently used exception: DevOps is always capitalized)

- **certificate authority (CA) -** A certificate authority (CA) is a trusted entity that issues digital certificates, which are data files used to cryptographically link an entity with a public key. Vault can be a CA.

- **CI/CD -** continuous integration/continuous delivery: A label for various degrees and styles of software production pipeline automation. Includes everything in the version control, build management, testing automation, and deployment stages. This is common enough in technical posts that it doesn't need to be spelled out, unless it's unclear. HashiCorp products don't provide CI/CD but they connect with the main tools that do: GitHub Actions, Jenkins, CircleCI, GitLab CI/CD, and many more.

- **cloud operating model (COM)** [*Now an older, less-preferred term for our overarching approach - the main message must revolve around Infrastructure Cloud*] - Our term for the best practices of cloud IT operations—a solid foundation of provisioning, secrets management for security, service networking automation, and then flexible application deployment optimization engines on top of that. See our 4-layer model video and white paper. Describe it as “a” cloud operating model, rather than “the” cloud operating model.

- **CNCF** - The Cloud Native Computing Foundation that manages many open source projects, including Kubernetes, Envoy, Helm, Linkerd, Open Policy Agent, and Prometheus. HashiCorp is a member as of March 2020. See www.cncf.io for details.

- **DevOps -** Using tooling, process (cross-department knowledge), and culture to reduce friction between silos/departments as software moves from design to development, and through testing, security, deployment, and post-production reliability, updates, and maintenance.
    - Note: older versions of our messaging used “DevOps Delivered” and similar phrasing as our tagline and should no longer be used. We want to be described as a “Cloud Infrastructure Automation” or “Multi-cloud Automation” company instead of as a DevOps company.

- **Docker** A very popular, industry-standard open source container format, and, originally, the company that brought this approach to market. HashCorp products work in Docker environments.

- **hybrid cloud -** A state where an organization runs workloads on a third-party provider (public cloud) as well as on their own private, on-premises datacenter (which may or may not be run as a private cloud).

- **immutable infrastructure -** Pieces of infrastructure that don’t change. When a change is needed, any number of pieces are first erased, and then rebooted as fresh infrastructure from a new template. No infrastructure is modified in place, so the likelihood of bugs emerging over time is greatly reduced. Immutable vs. mutable infrastructure. Terraform’s philosophy relies on immutable infrastructure. Configuration management tools like Chef and Puppet, on the other hand, were designed to modify infrastructure and perpetually change it that way.

- **infrastructure as code (IaC) -** Using tools (such as Terraform, Ansible, Puppet, Chef, CloudFormation, etc.) to automate the provisioning and configuration management of virtual and physical infrastructure. This is possible because the infrastructure components are expressed and modified with programming code (No hyphens: No “infrastructure-as-code”).

- **Istio -** A well-known service mesh competitor to Consul. Always refer to the product as Google Istio, since they are still essentially owned by Google. Istio was not donated to the CNCF and instead Google chose to create a new foundation (Open Usage Commons) to own Istio.

- **Kubernetes -** Kubernetes is an open source application platform run by the CNCF that has become a focus for the infrastructure industry. It has some features of Consul (etcd for service discovery, Istio is separate for service mesh) and Nomad all in one solution. While Kubernetes has much more mindshare than Nomad for container orchestration, it is widely considered not as strong in the areas that Vault and Consul handle because of its complexity.

- **microservices -** This term is used to describe an architectural approach in which an application is built in small, often distributed, pieces or components. Previously, applications were built as a single unit (often called a monolith in this context). Consul and Nomad are primarily built with the assumption that microservices or a service-based architecture are being used.

- **multi-cloud -** Usually, when we refer to it, it's a state where an organization has to manage services from different cloud vendors with different processes and syntaxes. See more information in our multi-cloud discussion video. HashiCorp tools integrate with most cloud providers to provide one infrastructure automation workflow for multiple clouds.

- **orchestrator/orchestration -** A broadly used term that most often refers to container orchestration. Containers often work better as infrastructure components in distributed, cloud environments. To manage many of them at scale, a container orchestration engine, often a scheduler like Nomad or Kubernetes, is used.

- **platform team/platform engineering**

- **policy as code -** Also known as *compliance as code*, this is a practice where a framework and language are used to enforce automated gatekeeping for what developers and operators can and cannot submit in their code. Sentinel does this for all HashiCorp products (Not *policy-as-code*).
    - Refer to Sentinel like this: Sentinel, HashiCorp's policy as code framework, …

- **privileged access management (PAM) -** A method for authenticating people across your IT systems. Identity access management (IAM), by contrast, is meant to manage authentication for applications and their services—a much higher scalability solution. PAM (in its traditional incarnation) does not work well for cloud environments. IAM and secrets management are purpose built for the cloud. What's the difference between PAM and IAM? Vault supports IAM. Boundary is a modern PAM solution that does work well for cloud environments.

- **provisioning -** The process of building and deploying any components of a full IT infrastructure via code templates, including databases, load balancers, caches, firewall settings, SSL certificates, queues, monitoring, subnet configurations, routing rules, and almost every other aspect of your infrastructure. Terraform is a provisioning tool.

- **public key infrastructure (PKI) -** A public key infrastructure (PKI) is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption. Vault can provide PKI.

- **secrets management -** A method for managing digital authentication credentials (secrets), including passwords, keys, APIs, and tokens for use in applications, services, privileged accounts, and other sensitive parts of an IT ecosystem. Sometimes called “credential management”. Vault does this. Do **not** use: secret management (we always use “secrets” with an ‘s’).

- **service discovery -** A method, enabled by tooling, that tracks, monitors, and automatically manages applications components called services. It tracks them in a central service registry and reroutes traffic if a service is unhealthy or if other parameters are put in place. Consul does this.

- **service mesh -** A more complete self-service service networking automation state with service discovery as a foundation, and building on top of it with secured service-to-service communication and fine-grained traffic management and observability at multiple networking layers (L4 and L7). Consul does this.

- **zero trust (networking/security/architecture) -** Zero trust is a security perspective where you do n’t assume the security methods at your IT environment’s perimeter will not stop all attackers, so you also treat all traffic and individuals inside your environment as potential attackers and always require credentials (which could be managed via Vault) and authenticated service-to-service communication (which could be managed via Consul service mesh). Boundary secures the human-to-machine connections aspect of zero trust security.

4. Writing principles

Content planning and messaging

Note: While this section gives some general tips about our audience, it's important to do personal research to understand them. ***Read the newsletters, forums, and blogs that they’re reading.*** *Learn to talk like they talk, and understand the topics they care about.*

It takes research to know which topics your audience cares about. You need to understand your target audience’s likes and dislikes, along with what information would be new and useful to them vs. what they already know. You don’t want to waste their time or come off as uninformed.

Think about these questions before you write or outline any piece of content:

- What is the goal of the content? Articulate as precisely and narrowly as possible. (awareness, leads, etc.)
- What audience do you need to reach to achieve that goal?
- What does that audience want/need?
- Where will this content be placed? Will our audience easily find it in the places they typically look for information?
- What do they already know? Are they a current user or not yet familiar with us?
- How can you give your audience what they want while also meeting HashiCorp's goals?

Writing rules

In general, for the HashiCorp blog and marketing materials, we use AP Style, American spellings, and American English. If you are writing specifically for a different locale than the US, then use the spellings and language of that locale. There are a few differences (for example, the Oxford comma—we use it). We outline the most important rules below. In high tech, there are often reasons to break or change traditional rules, or even create some new ones. In general, we are going for clarity and consistency. Use these specifics along with that philosophy as your guide.

For blogs - **Say exactly what the post will encompass quickly:** 99% of blog posts should begin with a brief introduction (try to keep it under 130 words) that ends with one or two "nutshell" sentences saying directly what your blog will be about/what the reader will learn/takeaways/topics covered in the entirety of the blog.

For blogs - **Break up the post into sections:** Your post must also be broken up into sections with H2 headings.

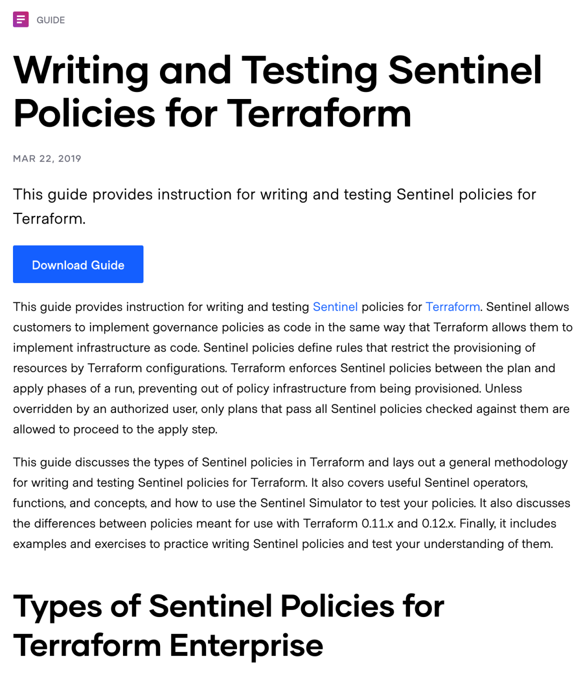
Use a single space after a sentence.

Titles and subheadings

We use sentence case for titles and subheadings. That includes web pages, Google/PPT slides from our template, blogs, resource posts, developer.hashicorp.com tutorials, white papers, webinars, videos, HashiConf/Days/Talk session titles, diagram text, and headings+text in tables/spreadsheets.

Sentence case should also be used for summaries, blurbs, CTAs, buttons, etc. Punctuation should not be used in titles or for display copy except in situations using multiple sentences that need to be separated.

Wrong (for both title, subtitle & the button)



GUIDE

Writing and Testing Sentinel Policies for Terraform

MAR 22, 2019

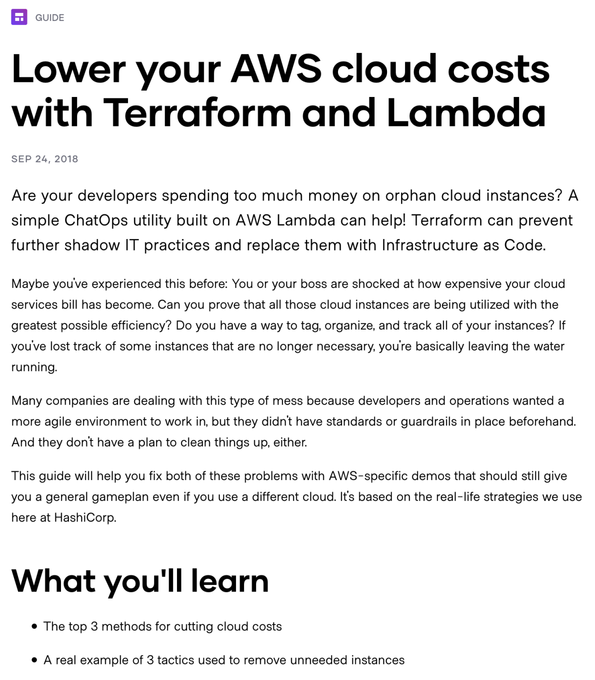
This guide provides instruction for writing and testing Sentinel policies for Terraform.

This guide provides instruction for writing and testing Sentinel policies for Terraform. Sentinel allows customers to implement governance policies as code in the same way that Terraform allows them to implement infrastructure as code. Sentinel policies define rules that restrict the provisioning of resources by Terraform configurations. Terraform enforces Sentinel policies between the plan and apply phases of a run, preventing out of policy infrastructure from being provisioned. Unless overridden by an authorized user, only plans that pass all Sentinel policies checked against them are allowed to proceed to the apply step.

This guide discusses the types of Sentinel policies in Terraform and lays out a general methodology for writing and testing Sentinel policies for Terraform. It also covers useful Sentinel operators, functions, and concepts, and how to use the Sentinel Simulator to test your policies. It also discusses the differences between policies meant for use with Terraform 0.11.x and 0.12.x. Finally, it includes examples and exercises to practice writing Sentinel policies and test your understanding of them.

Types of Sentinel Policies for Terraform Enterprise

Correct (for both title and subtitle)



GUIDE

Lower your AWS cloud costs with Terraform and Lambda

SEP 24, 2018

Are your developers spending too much money on orphan cloud instances? A simple ChatOps utility built on AWS Lambda can help! Terraform can prevent further shadow IT practices and replace them with Infrastructure as Code.

Maybe you've experienced this before: You or your boss are shocked at how expensive your cloud services bill has become. Can you prove that all those cloud instances are being utilized with the greatest possible efficiency? Do you have a way to tag, organize, and track all of your instances? If you've lost track of some instances that are no longer necessary, you're basically leaving the water running.

Many companies are dealing with this type of mess because developers and operations wanted a more agile environment to work in, but they didn't have standards or guardrails in place beforehand. And they don't have a plan to clean things up, either.

This guide will help you fix both of these problems with AWS-specific demos that should still give you a general gameplan even if you use a different cloud. It's based on the real-life strategies we use here at HashiCorp.

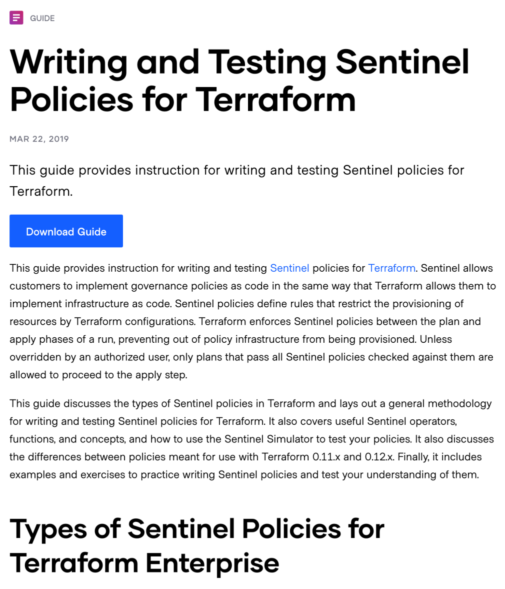
What you'll learn

- The top 3 methods for cutting cloud costs
- A real example of 3 tactics used to remove unneeded instances

Use the Oxford comma/serial comma (but not with ampersands)
Correct Example: I'm going to the store to get eggs, bread, and milk. The title of my book is "Good, Bad & Ugly." In general, avoid ampersands except where needed to conserve space.

Right

Wrong (for both title and subtitle)


GUIDE

Writing and Testing Sentinel Policies for Terraform

MAR 22, 2019

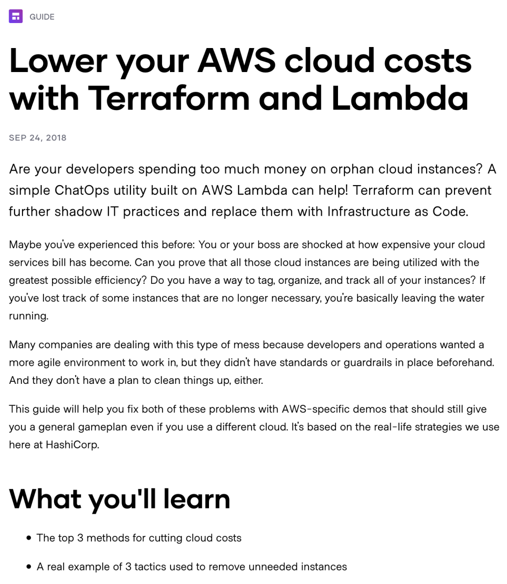
This guide provides instruction for writing and testing Sentinel policies for Terraform.

Download Guide

This guide provides instruction for writing and testing Sentinel policies for Terraform. Sentinel allows customers to implement governance policies as code in the same way that Terraform allows them to implement infrastructure as code. Sentinel policies define rules that restrict the provisioning of resources by Terraform configurations. Terraform enforces Sentinel policies between the plan and apply phases of a run, preventing out of policy infrastructure from being provisioned. Unless overridden by an authorized user, only plans that pass all Sentinel policies checked against them are allowed to proceed to the apply step.

This guide discusses the types of Sentinel policies in Terraform and lays out a general methodology for writing and testing Sentinel policies for Terraform. It also covers useful Sentinel operators, functions, and concepts, and how to use the Sentinel Simulator to test your policies. It also discusses the differences between policies meant for use with Terraform 0.11.x and 0.12.x. Finally, it includes examples and exercises to practice writing Sentinel policies and test your understanding of them.

Types of Sentinel Policies for Terraform Enterprise


GUIDE

Lower your AWS cloud costs with Terraform and Lambda

SEP 24, 2018

Are your developers spending too much money on orphan cloud instances? A simple ChatOps utility built on AWS Lambda can help! Terraform can prevent further shadow IT practices and replace them with Infrastructure as Code.

Maybe you've experienced this before: You or your boss are shocked at how expensive your cloud services bill has become. Can you prove that all those cloud instances are being utilized with the greatest possible efficiency? Do you have a way to tag, organize, and track all of your instances? If you've lost track of some instances that are no longer necessary, you're basically leaving the water running.

Many companies are dealing with this type of mess because developers and operations wanted a more agile environment to work in, but they didn't have standards or guardrails in place beforehand. And they don't have a plan to clean things up, either.

This guide will help you fix both of these problems with AWS-specific demos that should still give you a general gameplan even if you use a different cloud. It's based on the real-life strategies we use here at HashiCorp.

What you'll learn

- The top 3 methods for cutting cloud costs
- A real example of 3 tactics used to remove unneeded instances

Also: Don't put a section heading right after the title. Titles and subsections should never be right on top of each other. Always try to have text before a new heading.

Prefer the active voice over the passive voice
In active voice, the subject of the sentence does the action. In passive voice, the subject of the sentence has the action done to it.

> Active: We recommend configuring VCS access when first setting up an organization.
>
> Passive: It is recommended to configure VCS access when first setting up an organization.
>
> Active: Terraform has a provider framework to leverage this behavior.
>
> Passive: To hook into this behavior, a provider framework has been built into Terraform.
>
> Active: Next, Kubernetes will register the service.
>
> Passive: Next, the service will be registered.

Address the reader as "you" and don't walk through steps saying "we":
Use "I" and "you" *not* "we" and "our" if you're stepping through instructions (*unless* you're speaking as the company).

> Right: "In this example, *I'm* going to do this" or "Next, add this code to *your* application."
>
> Wrong: "now *we're* going to take *our* code and put it in *our* application. Next, *we* will run this command."

In imperative instructions, you can often be more direct by omitting the subject entirely, which is also great. ("Next, go to the settings page.") In general, don't avoid addressing the reader. Indirect writing can *feel* more polite as you're writing it, but it's harder to understand and it wastes the reader's time.
Good example

Pronouns to use for an organization:

Spell out unfamiliar acronyms/initialisms/abbreviations for the first use (*unless* they are broadly known in the software community or the audience of this article, e.g. Don't bother writing out the full acronym for SQL or REST)

First use: test-driven development (TDD)

Second use: TDD

Team or department names: When referring to one particular department at a real company or a term that is always and only associated with a department at companies (e.g. Human Resources), capitalize that department name.

When referring to a general department name that exists in in many companies but may not be the exact name of a company's team or department, and it's a word that can mean other things besides the team name, don't capitalize that department or team name (e.g. security team, development team, platform team).

Right: "Many companies have Human Resources departments, but not everyone has security teams or platform teams. They might have a DevOps Engineering or Cloud Engineering team with those precise names, but we only capitalize precise names."

Wrong: "Some companies will put their Platform Team, their Security Team, and Application Development in the same room."

Slashes (/)
In sentences, don't use it like a comma. Don't put spaces on either side of it. Here are its uses.

Quotes and punctuation
If they apply to the quoted material, they go within the quotation marks. If they apply to the whole sentence, they go outside it:

Correct: Sandy asked them, "Why do you log out after work?"

Also Correct: Did the bug occur every time you pressed "Log Out"?

How to format titles for books, songs, movies, chapters, etc.
Use the table further down on this article. Note that we generally use AP Style.

When listing partners/companies

If you're ever listing companies or partners in an article (this happens a lot in new integration blogs) always *alphabetize* the list to remove any semblance of favoritism.

Hyperlinks in text
Avoid using raw URLs as hyperlinks in content. Link on existing text instead.

Right: Read Taking Vault to Another Level.

Wrong: Read the full blog post: http://hashicorp.com/blog/Ishouldntdothis

For Accessibility: Links should provide information on the associated action or destination. Avoid saying "click here" or "learn more." However, in social media posts, this is OK to do. Examples can be found in this article: Designing Better Links for Websites and Emails: Guideline.

The same approach is beneficial for SEO: use the name or other identifying words for the what you're linking to.

Right: Read Taking Vault to Another Level.

Wrong: Read the full blog post here.

Inline code formatting:

Whenever you're mentioning a code command, method, function, variable, etc in a sentence, format it as code with the backticks (`words in code`) if you're writing in markdown. Use the `Consolas` or `courier` font in Google Docs so that the markdown converter plugin will pick it up.

Always provide code as text, rarely as screenshots

This is so practitioners can copy/paste our code samples to try them out. It's also better for SEO. Screenshots are only useful if you're using special markups or boxes around code that's purely illustrative and wouldn't be useful to copy.

Exclamation points (!)

Avoid using exclamation points. Only use it if you're connecting with understood, strong reader emotion.

Right: Achieving this centralization is a huge improvement in security posture, but it's not the end of the journey. This is because applications don't keep secrets! It turns out, most applications do a worse job keeping secrets than our close friends.

Wrong: We are excited to announce Vault 1.3!

Em Dashes (—)
To type an em dash: (Mac: Option + Shift + - (minus)) (Windows: Alt + Ctr l+ - (minus))

When using long dashes (known as em dashes), don't use a double dash, and use them properly, as an aside statement that is like a parentheses statement, but directly relevant to the sentence. Proper usage examples include:

"You've got hundreds of people across the company, all wanting to make use of Vault across thousands of systems — many of which are production."

"My team looks after several things — but pertinent to this talk — we look after our Vault clusters."

As you see from the examples, we use the AP style of em dash with spaces before and after.

Possessives
If you have a hard time figuring out where to put the apostrophe in a possessive word (boy's toys, the banners' colors), read the rules on possessives.

File format name usage:

Alone in text: GIF, JPG, JPGs (plural)

In an actual file name: ben-twitter-profile.jpg

Numbers

For more formal, non-technical writing, write out "one" through "nine" and write 10 and above as numerals.

For more technical writing in sentences with technical terms, write most numbers as numerals for ease of reading. Spell out "one" if it's not used in tandem with another number (like in a range 1 to 4 for example) in the sentence. If you've used "one" or another written number nearby, do the same for other nearby numbers for consistency.

- One new employee started on Monday, and 12 start next week.
- I have between 1 and 4 friends.
- We have one shot to get this right.
- I ate three donuts at Coffee Hour.

    - We started with 1 load balancer. Then we had 2 load balancers.
    - There are four main reasons why you should migrate to the cloud
    - One of my friends came to chat. Then two others showed up.

Numbers over 3 digits get commas:

- 999
- 1,000
- 150,000

Write out big numbers in full if you want to advertise their gravity. Abbreviate them if there are space restraints, as in a title, tweet, or a chart: 1K, 150K, 1M, $3B, $5.1B. More technical content can also use abbreviations more frequently since that audience favors brevity.

Dates

Generally, spell out the day of the week and the month (can abbreviate the longer months if needed for space. Do not add st, th, etc. after the numeral(s).

- Saturday, January 24
- Sat., Jan. 24

Use the date format of the country your content is targeting. For example, if you're writing an email to Europe, the dates should be [Day][Month][Year]. Importantly, when giving the date for an event, include the day of the week whenever relevant. The Events team style guide for dates is a good reference and does not abbreviate months or days.

Times

11 a.m. PT; 11:30 p.m. PT   No need to specify standard or daylight time (for example: PDT or PST). No need to include :00 for events that begin on the hour. Use the time zone for the location of the event. For virtual events not tied to a specific location, use the time zone(s) where most of the audience is.

Percentages

Use the % symbol instead of spelling out "percent." **Correct:** 50%

Ranges and spans

Use a hyphen (-) to indicate a range or span of numbers.

- It takes 20-30 days.

Colons

Use colons, not dashes, to set off the header from the body:
"GitOps: Projects can be configured" not "GitOps - Projects can be configured"
Capitalize after a colon if it is in a bullet, if the next word is a proper noun, or if the clause after could be a complete sentence.

Ordered and unordered (Bulleted) lists

Capitalize the first word in a bullet. • Like this
Use numbers rather than bullets only if:

- A need to refer to the elements by number may arise;
- The sequence of the items is critical; or
- The numbering has some independent meaning, for example in a listing of musical tracks.

Use punctuation after a bullet if its a complete sentence. If not, don't.
Bullets should have similar structure and phrasing, so you should be consistent with punctuation or no punctuation after them. Most of the time you should use a colon to end the statement just before a list, but in some cases a period or ellipses (...) could make sense.

Footnotes

Avoid, especially in blogs. OK if necessary for references in white papers and research documents

Job titles

Capitalize job titles. e.g. Founder and CTO Armon Dadgar. Jill Samson, the VP of Finance.
Lowercase formal titles that appear on their own. e.g. The governor likes ramen.

Academic degrees

Use bachelor's and master's not B.A., M.S., or M.A. It's fine to use doctoral abbreviations such as LL.D. and Ph.D.

Cardinal directions

AP Style calls for lowercase north, south, east, west, northeast and so on when you're talking about compass directions. "Drive south for 3 miles." But when you're talking about a region, use a capital letter. "Rain is expected in the Northeast."

Tutorial and code block best practices:

These are best practices pulled from our HashiCorp Learn Education team. Another style document that's required reading for tutorials is Google's Code in Text style guide - In general try to follow their advice for how code, commands, and other elements should be written and formatted in a post. One good piece of advice from this guide (Grammatical treatment of code elements):

In general, don't use code elements such as keywords and filenames as if they were English verbs or nouns. Don't inflect the name of a code element, to make it plural or possessive. Instead, include a noun after the name of the code element, and inflect that noun.

Give instructions. No need for “next, you do this” “next, we do this”

Preferred	**Not preferred**
Select the Azure policy	We are going to select the Azure policy
Next, configure the server in order to…	Next, you need to configure the server…

Plain verbs for titles and section titles

Use plain verbs rather than present participle in tutorial and section titles. This makes titles shorter and makes titles sound actionable.

Preferred	**Not preferred**
Configure the server	Configuring the server
Join the agents	Joining the agents

Use the grammatical person "you" for the practitioner

Preferred	**Not preferred**
In this tutorial, you learned how ...	In this tutorial, we learned how ...
Terraform Cloud's API lets you create workspaces without a VCS connection.	Terraform Cloud's API allows us to create workspaces without a VCS connection.

Only use "we" to describe a recommendation by HashiCorp

Preferred	**Not Preferred**
We recommend configuring VCS access when first setting up an organization.	It is recommended to configure VCS access when first setting up an organization.

Represent confusable vocabulary words in italics *on first usage*

Use italics for product-specific vocabulary terms that can be confused with a more general usage of a word. These are not feature names but specific things in a feature. Only italicize the first usage of the vocab word, and don't italicize after unless you're using a feature vocab word alongside the same word in a general context. When in doubt, eliminate potential confusion. Another option is to add a hyperlink to an introduction documentation page for the term upon first usage.

Preferred	**Not preferred**
This web service is called a *counting service.*	This web service is called a Counting Service.
The central functions for CTS are called *tasks*. When you run a task…	Terraform Enterprise *workspaces* can be organized in many ways. [workspaces are not a general, common word, so italics are not needed, and workspaces are not capitalized as we don't like to capitalize primitives]
The key first step in Consul is setting up intentions.	The key first step in Consul is setting up intentions.

Use colons if your sentence is leading into a code snippet (or bulleted list)

Ex: Start the proxy with the default configuration:

```
Code  snippet code
```

But **use periods if you're making a statement right before a code snippet.**

Ex: Run the apply command.

```
terraform apply
```

Use a descriptive, imperative sentence before a code block

Preferred	Not Preferred
Start the proxy with the default configuration:	Next, let’s run our default configuration. Run the following:

Represent user shell prompt with a `$`

Preferred

```
$ curl localhost:9090
```

Not Preferred

```
> curl localhost:9090
```

Commands that exceed 100 characters overflow the rendered code block. **Use the shell's line continuation character to continue commands across multiple lines.**

```
$ docker run \
      --name postgres \
      --env POSTGRES_USER=root \
      --env POSTGRES_PASSWORD=rootpassword \
      --detach  \
      --publish 5432:5432 \
      postgres
```

Remove timestamps in log output

Preferred

```
[INFO] serf: EventMemberJoin:
server1.dc1 10.20.10.11
```

Not Preferred

```
2017/08/30 19:05:13 [INFO] serf:
EventMemberJoin: server1.dc1
10.20.10.11
```

Use only one command per code block

There should be one prompt and one output. Additional commands in the same block potentially obfuscate all the commands required to perform an operation from the practitioner.

Recap + more code block rules:

- Use “shell-session” syntax for shell commands, CLI, text
- Represent **user shell prompt** with a $
- Represent root / privileged shell prompt with a #
- Represent **comments** with a ##
- Use long command flags for a HashiCorp CLI
- Use short command flags for a non-Hashicorp CLI when it is common usage
- Continue **longer commands** across multiple lines
- Include command output unless it’s not relevant to the illustration
- Remove timestamps in log output
- Use only **one command per code block**
- Use present tense to describe output.

Inclusive Language

Our goal is to use language and style that are clear and consistent. However, we should work hard to make sure we are not being exclusionary or insensitive. Below are a few areas to pay attention to, but this list isn’t intended to be comprehensive. Be sensitive to what your audience might not appreciate or would find inappropriate. It’s easy to find other ways to phrase something. Choose clearer terms instead of terms that may or may not have deeper, charged, or unintended meaning. For a comprehensive guide to inclusive language, this is a good resource: An Incomplete Guide to Inclusive Language for Startups and Tech

Non-Violent Communication

In most cases using non-violent language will help you avoid hyperbole, as well as idioms and figures of speech that can confuse non-native English speakers. Additionally, it feels more friendly and inclusive. Be wary of phrases that originally came from military metaphors and avoid if you can (“rolling thunder,” “shock and awe,” etc.).

Preferred	**Not preferred**

Press the Escape key	Hit Escape
Type your password	Punch in your password

Sometimes technical language is itself violent. In these cases be as specific as possible, to make it clear that you're referring to a specific technical concept.

Preferred	**Not preferred**
Use the kill command	Kill the service
primary/secondary	master/slave

(In parallel with our efforts to remove wording like this from our products — which started in 2019 — we want to make sure our writing follows the same approach.)

Ableism

Avoid assuming that your audience is interacting with a guide or the product in a specific way. Instead of stating what the user will experience, instead say what the technology will do. This has the additional advantage of clarifying for the user how the technology works, and will help our guides make sense when translated to other formats like speech.

Good	**Bad**
Consul prints the list of services	Look at the list of services
The list includes the counting service	You can see the counting service

Remember that mental and emotional ability is also included in ableism. Avoiding ableist language here can also help you avoid hyperbole.

Preferred	**Not preferred**
Reasonable defaults	Sane defaults

This is useful because...	This is crazy useful
Basic functionality check	Sanity check

Use the most accurate language you can. Don't avoid using terms that reference ability if they are accurate. For example, if a button is "disabled" unless you complete a confirmation step, there's no need to skirt the term. If you can avoid charged words by replacing them with an equally good substitute feel free; if there is no good substitute, optimize for clarity.

Pronouns

If you are using the grammatical person "you" for the user you probably won't use very many pronouns in your writing. If you do find yourself using pronouns, make sure they are necessary, and if so, use the gender neutral "they/them" for both singular and plural.

Good	**Bad**
Once you give the developer the token they can...	Once you give the developer the token he/she can...

Posting guide

Read the docs for how to author drafts in google docs properly, handle files (under 1MB please) and post on our website.

SEO best practices

Old seo best practices doc:
https://docs.google.com/document/d/114TwYqYeNATgO72Vt8XhCvi_qEnJjU_63Hms0ZGVNAY/edit#heading=h.b9ys96mgkxn2

5. Word list - Definitive ways to write technical software terms and other words

The previous section's words are repeated here for the definitive way to write them. For software terms, the best way to know how to spell or capitalize a project name is to look on the project's primary webpage or GitHub repo. Spell and capitalize the project how the creators spell and capitalize it (examples: Neo4j, jQuery, MongoDB. The same guidance holds for all commercial, company, or other product names: check on the owner's website to see how they refer to them.

24/7 not 24-7

ACL, TLS, URL, ... (*Capitalize* every acronym and abbreviation, don't be lazy and write "url" unless it's in a code snippet or a code formatted word)

access control lists (ACLs)

agile development avoid saying "Agile" since there's no definitive definition so it doesn't merit having proper noun status.

AI and artificial intelligence either is OK, interchangeable

America/American: Avoid using these US-based terms. Spell out *United States* in text on first reference when it's used as a noun and use *US* as an adjective.

Ansible

autoscaling

AWS: Amazon Web Services spell out on first reference (note that many AWS products have specific names, some are branded Amazon and some AWS… check to be sure)

AWS re:Invent Amazon's cloud conference, note unusual capitalization and punctuation use whole thing on first reference

AZ availability zone, spell out on first reference

add-on (noun, adjective), add on (verb)

auth method

autoscaling (general) different from proper nouns such as Nomad Autoscaler, AWS Auto Scaling groups.

backend (not back-end or back end. Examples: On our blog, in our job postings, and in our Terraform and Vault docs)

bandwidth one word

Bash

Beta

Boundary Client Agent / Client Agent (when referring to the Boundary Client Agent) because it's a separate download.

catalog not catalogue

CDN acronym for Content Delivery Network, OK to use without spelling out

certificate authority (CA)

certificate signing request

chaos engineering

checkbox

Chrome capitalize for Google Chrome browser

CI/CD, continuous integration / continuous delivery (no need to spell out unless it's unclear in context)

CLI command-line interface (spell out on first reference except in technical material for practitioners)

client-server

cloud native (adj) (In "cloud native technology" "cloud native" is so integral to its meaning that the entire phrase serves as a noun. The same is said for an "open source project." Following this logic, "cloud native" and "open source" are not compound adjectives that need to be hyphenated.) In general, avoid using to reduce confusion with the more specific Cloud Native Computing Foundation.

cloud operating model lower-case, spell out, do not abbreviate as COM

CNCF Cloud Native Computing Foundation

code base (noun) (not "code-base" or "codebase")

command line (noun), **command-line** (adj)

Common Vulnerability and Exposure (CVE)

Consul API gateway or **a Consul API gateway**

Consul dataplane

Consul-Terraform-Sync (not branded with “HashiCorp” in front of it). Write the full name followed by “(CTS)” the first time it appears in a document. Use “CTS” in subsequent references.

COVID-19 on first reference, Covid is OK after that

coworker

CRM customer relationship management, spell out on first reference

cyberattack one word

cybersecurity one word

data use as singular. **Example**: *Significant data is available to support the report.*

data lake *not datalake*

database

datacenter (mainly because it looks better when you write “multi-datacenter” rather than “multi-data center”)

Day 0, Day 1, Day 2, Day N (not Day Zero)

DBMS use sparingly, spell out on first reference

Default Project (It’s capitalized as a UI element: See “**Capitalization rules for UI elements**” section above) This is the name that HCP Terraform gives to the project that it automatically creates for each organization. It is a proper noun.

deprovision not de-provision

DevEx stands for “developer experience”. It's quickly becoming a mainstream IT term.

DevOps (not devops --and it’s Dev and Ops when used separately. Also, there is no such thing as “a DevOps”—call them “DevOps engineers” or “DevOps practitioners”)

disc optical media such as CDs or DVDs

disk magnetic media such as disk drives or floppy disks

Docker

double-click

drift detection

drop-down (noun, adj), **drop down** (verb)

east-west (lower case when discussing datacenter traffic - similarly north-south

e-book

e-commerce (the industry) (not ecommerce or eCommerce, *capitalize the E at the beginning of a sentence)*

emoji (singular and plural)

encryption as service can be EaaS on second reference

endpoint (not end-point)

Ethernet capitalize

EU European Union, no need to spell out

fail-safe (hyphenated)

financial services spell out — do not abbreviate as finserv, finserve, or FinServ

fintech try to avoid but if necessary should be "fintech" per AP style. Don't need to spell it out, but explain if not clear from context

five-nines (not 5-9s, or five-9s)

Flash memory (not FLASH or flash)

frontend (not front-end or front end. proof: On our blog, in our job postings, and in our Terraform and Vault docs)

function as a service (FaaS) spell out on first reference

geolocation

GB abbreviation for gigabytes, caps, no space between number and abbreviation. **Example:** Informix's new database needs 2GB of free system memory.

GB/s abbreviation for gigabytes per second for computer throughput speeds, no need to spell out on first use, caps as shown. For consistency across our sites, don't use Gbps even though that's technically correct too.

GDPR The EU General Data Protection Regulation. No need to spell out, but may want to link if in an unfamiliar context (https://eugdpr.org).

GitHub (GitLab too**)**

Go programming language, not Golang unless needed for clarity in context

Google Cloud Platform (GCP) This term has been deprecated by Google. Service is now called **Google Cloud**. Do not use Google Cloud Platform or GCP on new content, but no need to change existing content.

Google Kubernetes Engine (GKE)

governance, risk, and compliance (GRC) lower case, spell out on first reference

GUI graphical user interface, no need to spell out. OK to use UI as well.

hard code, hard coded, hard coding (verb), **hard-coded** (adj)

HashiCorp (always spelled out completely, never abbreviate to only "Hashi," capitalize the "C" always, never shorten to HC)

HashiCorp Cloud Platform (HCP) use "the" when the term is spelled out ("HashiCorp today announced the newest release of the HashiCorp Cloud Platform") but don't use "the" when abbreviated ("HCP enables teams to focus on building applications faster and with fewer resources.")

HashiCorp Developer. You can call articles there HashiCorp guides or HashiCorp tutorials on our HashiCorp Developer platform.

HashiCorp stack not HashiStack

hashtag

HCL, HCL2 (use "HashiCorp Configuration Language (HCL)" on the first instance, use HCL2 when appropriate, not HCLv2)

HCP Vault Radar you can use just "Vault Radar" on subsequent mentions. First mention should include HCP.

healthcare one word

he/she avoid, use "they" when possible or when preferred pronouns are unknown

homepage

honeypot when referring to these cybersecurity traps for threat actors, honeypot is condensed to one word

hybrid cloud

IAM identity access management, spell out on first reference

IdP (not IDP) identity provider (spell out on first reference)(small "d" because the d is not its own word)

immutable infrastructure

InfoSec, AppSec, SecOps, NetOps, DevOps, etc

infrastructure as a service (IaaS)

infrastructure as code (no initial caps, no hyphens, and we tend *not* to use the acronym IaC. Perfectly normal to use the term "infrastructure code" to refer to the TF configuration code.)

The Infrastructure Cloud, or just **Infrastructure Cloud** capitalize the "The". The Infrastructure Cloud can be referred to as an approach, blueprint. From the announcement: "HashiCorp created The Infrastructure Cloud to provide a unified approach to managing the entire lifecycle of your cloud infrastructure and security resources"

Infrastructure Lifecycle Management (ILM)

integrate

integrated storage a Vault feature some old docs have capitalized. Just link on first mention and lower case to show that it's a feature and not the general words integrated or storage.

internet (never capitalize unless it begins a sentence)

internet of things

I/O input/output, not IO. no need to spell out

Istio

Java capitalize

JavaScript intercap

JSON acronym for JavaScript Object Notation, no need to spell out

just in time (hyphenate when used as a compound modifier: just-in-time delivery)

KB abbreviation for kilobytes, caps, no space between number and abbreviation: 64KB

Key Management Interoperability Protocol (KMIP)

key-value (as in "key-value store". Not key/value, though many publications use **KV store**)

Kubernetes (use K8s abbreviation only after first reference and in very informal contexts)

KV v1 secrets engine

KV v2 secrets engine

least-privilege access (hyphenated)

lifecycle as in software delivery lifecycle

Like (the social media activity, with capitalization)

live stream (two words, lower case)

localhost

login *n., adj.;* **log in** *verb.;* e.g. I'm going to log in to my computer.

log on *verb.*

lunch & learn

machine learning (hyphenate when used as a compound modifier) OK to use ML without spelling it out in appropriate contexts

MB abbreviation for megabytes, caps, no space between number and abbreviation: 64MB

MB/s Abbreviation for modem and network speed in megabits per second. No need to spell out, caps as shown. For consistency across our sites, don't use Mbps even though that's technically correct too.

MHz abbreviation for megahertz.no need to spell out, 64MHz

microservices

microservice architectures, microservices architecture

Microsoft Azure, *shorthand*: **Azure** on second reference

millisecond 1,000th of a second. Abbreviated as ms (not msec) used closed up: 100ms no need to spell out.

multi-cloud

multi-factor authentication (MFA) lower case, spell out on first reference

.NET

NGINX in more formal places, Ngnix is fine in informal writing

no-code provisioning (hyphenated)

Node.js, *shorthand*: **Node** (careful with this one, "node" can mean a lot of things)

Nomad Autoscaler the generalized term is lower case “I’m now running an autoscaler on this cluster”

north-south (lower case when discussing datacenter traffic - similarly east-west

OK

OS (operating system and plural: operating systems. Don’t use “OSes” or “OS” for plural. No need to spell this out the first time as it is commonly known for all our audiences. But always spell out when used in plural form)

online (never capitalize unless it begins a sentence)

on-premises (**on-prem** is OK for later casual mentions in informal writing. Avoid "on-premise")

open source (noun, verb, adj) In “open source technology” “open source” is so integral to its meaning that the entire phrase serves as a noun. The same is said for an “open source project.” Following this logic, “cloud native” and “open source” are not compound adjectives that need to be hyphenated.

OpenShift (from IBM’s RedHat, not “Openshift”)

Operator when referring to a Kubernetes Operator (i.e. Terraform Kubernetes Operator) it is capitalized.

opt-in (noun, adj), **opt in** (verb)

orchestrator / orchestration

OSS (open source software, spell out on first reference - note that we don’t call our base free versions OSS versions anymore. They are the “Community” versions)

pentest - short for penetration test.

platform as a service (PaaS)

platform team/engineer

plaintext / plain text: Use plaintext only to refer to non-encrypted or decrypted text in content about encryption. Use plain text to refer to text where no style information (bold, italics) is included, as opposed to formatted/rich text.

pod as in Kubernetes pod

policy as code (not hyphenated)

pop-up (noun, adj), **pop up** (verb)

PostgreSQL or **Postgres** - Either one works, both are official names

principle of least privilege (PoLP OK on subsequent references)

privileged access management (PAM)

producer-consumer model

provider(s) (as in, “Terraform provider” or “Terraform providers.” Not “Terraform Provider(s).”)

pseudocode one word

public key infrastructure (PKI) lower case, spell out on first reference

QA acronym for quality assurance or testing, no need to spell out

Q&A question and answer (not Q & A or Qs & As) no need to spell out

QoS abbreviation for quality of service; spell out on first mention

radix tree

Raft when referring to the algorithm

RAM random access memory, not DRAM. no need to spell out

README

read/write

resilience not “resiliency”

REST, RESTful (no need to spell out this abbreviation)

ROI (no need to spell out)

rollout, rollback

Ruby on Rails, *shorthand*: **Rails**

run tasks

scalable

seasons do not capitalize: spring, fall, winter, summer

A note about using “secret” vs “secrets” in feature names: In general, there are only rare cases where "secrets" would be preferred. Unless you really are talking about a collection of secrets, there's no real reason to use the pluralization. For example, you wouldn’t say ‘resources rotation’ you would say ‘resource rotation’. The same logic applies to secrets. If it’s being mistaken for the general adjective “secret” then consider reworking the sentence construction or context before.

secret auto-rotation

secret detection

secret rotation

secret scanning

secret sprawl (singular on the ‘secret’ here)

secret transformation

secrets engine - a major named Vault component type

secrets lifecycle management

secrets management/manager (not “secret management”)

secrets sync - A major Vault feature

Security Lifecycle Management (SLM)

serverless (*not* "server-less")

service discovery

service mesh

service principle

setup n., adj.; **set up** v.

shadow IT (not capitalized)

signup n., adj.; **sign up** v.

siloed (not silo'd or silo-ed)

single sign-on (SSO)

SLA service-level agreement (spell out on first reference)

SOC 2 (note the space before the 2 https://www.aicpa-cima.com/topic/audit-assurance/audit-and-assurance-greater-than-soc-2)

software as a service (SaaS)

SQL Structured Query Language. No need to spell out. Can be pronounced sequel or "Es-Cue-El," but treat as if it begins with a consonant. Example**:** *a SQL query, not "an SQL query."*

standby when referring to Vault performance standby nodes. No space or hyphen.

start up v., **startup** n./adj.

state file

sync

system integrators (*not* "systems," though "systems integration" can be used). On second reference, can call them SIs (*not* SI's)

tensor (lowercase when generic)

Tensorflow (capitalized)

Terraform Core - a component of Terraform

`terraform plan` and **`terraform apply`** - Often authors will write these as Terraform Plan and Apply. While Terraform plan and apply are alright in some instances, most of the time you should revert the usage of this to being the same as code in text usage. Never capitalize plan and apply, and typically you'll want to write the whole command as shown in bold above, or write "In Terraform, run `plan` and then `apply`" formatting the words plan and apply as code. If it gets too clunky, "Terraform plan and apply" are permissible sometimes. Don't pluralize ("Run your applies and destroys next") or treat plan and apply as regular nouns or verbs. This is explained in Google's Code in Text guide.

Terraform Plugin Framework capitalized and no "the"

Terraform module(s) (that are found in the Terraform Registry. Not "Terraform Modules.")
Terraform provider(s)

Terraform Registry (not "Terraform Module Registry," not "Terraform registry")

[*Don't* use a shorthand for **Terraform** or Terraform Enterprise unless it's a verbatim line of code.

> ***Good***: Terraform, Terraform Enterprise, Terraform Cloud, Terraform Cloud Business Tier
>
> ***Bad***: TF, TFE, TFC, TFC4B]

Terraform Stacks / Stacks / Stack

touchpoint one word

t-shirt

tweet, retweet

UK not U.K.

Unix

US not U.S.

use case not use-case

username

Vault Agent

Vault Proxy

VMware

VPN virtual private network, no need to spell out

Web

webhook not capitalized

webpage

website

WebSocket

white paper (not whitepaper)

whiteboard (one word)

Wi-Fi

10x or 100x: if abbreviation is needed you can use this but we prefer you say something like "10 times faster." Don't use x10 or x100

zero trust (zero trust networking, zero trust architecture, and zero trust security - do not hyphenate)

6. LEGAL: TRADEMARK AND REGISTERED MARKS

We do not use a legal notice at the end of formal documentation. Trademark and Registered marks should be used in press releases and other formal documentations (e.g. white papers, solution briefs), but do not need to be used in less formal documents like blogs.

Registered Marks is this $^{®}$ and should be used in first reference, in documents such as a press release or white paper, following the headline. Use this in superscript style (this $^{®}$, not this ®).

What gets a Registered Mark in the US?: HashiCorp$^{®}$, Terraform$^{®}$, Consul$^{®}$

Trademark is this ™ and should also be used in first reference, in documents such as a press release or white paper, following the headline.

Which brands need this in the U.S.?: Vagrant™, Packer™, Vault™, Nomad™ and Waypoint™

International Use: please refer to this list for Registered and Trademark usage per region. Updated as of April 2021]

Country	**Trademark Name**	**Status**
Australia	TERRAFORM	Registered
Australia	TERRAFORM Design	Registered

Australia	WAYPOINT	Pending
Canada	CONSUL	Pending
Canada	CONSUL Design	Pending
Canada	HASHICORP	Pending
Canada	HASHICORP Design	Pending
Canada	NOMAD	Pending
Canada	NOMAD Design	Pending
Canada	PACKER	Pending
Canada	PACKER Design	Pending
Canada	TERRAFORM	Pending
Canada	TERRAFORM Design	Pending
Canada	VAGRANT	Pending
Canada	VAGRANT Design	Pending
Canada	VAULT	Pending
Canada	VAULT Design	Pending
Canada	WAYPOINT	Pending
EUTM	CONSUL	Registered
EUTM	CONSUL Design	Registered
EUTM	HASHICORP	Registered
EUTM	HASHICORP Design	Registered
EUTM	NOMAD	Withdrawn
EUTM	NOMAD Design	Registered
EUTM	PACKER	Registered
EUTM	PACKER Design	Registered
EUTM	TERRAFORM	Registered
EUTM	TERRAFORM Design	Registered
EUTM	VAGRANT	Registered
EUTM	VAGRANT Design	Registered
EUTM	VAULT	Pending
EUTM	VAULT Design	Registered
EUTM	WAYPOINT	Pending
Japan	CONSUL	Pending
Japan	CONSUL Design	Pending
Japan	HASHICORP	Pending
Japan	HASHICORP Design	Pending
Japan	NOMAD	Pending
Japan	NOMAD Design	Pending
Japan	TERRAFORM	Pending
Japan	TERRAFORM Design	Pending
Japan	VAULT	Pending
Japan	VAULT Design	Pending
Japan	WAYPOINT	Pending
Republic of Korea (South)	TERRAFORM	Registered

Republic of Korea (South)	TERRAFORM Design	Registered
Republic of Korea (South)	WAYPOINT	Pending
Singapore	TERRAFORM	Registered
Singapore	TERRAFORM Design	Registered
Singapore	WAYPOINT	Pending
United States of America	CONSUL	Registered
United States of America	CONSUL Design	Registered
United States of America	HASHICORP	Registered
United States of America	HASHICORP Design	Registered
United States of America	NOMAD Design	Registered
United States of America	PACKER	Abandoned
United States of America	PACKER Design	Registered
United States of America	TERRAFORM	Registered
United States of America	TERRAFORM Design	Pending
United States of America	VAGRANT	Not Yet Filed
United States of America	VAGRANT Design	Not Yet Filed
United States of America	VAULT	Pending
United States of America	VAULT Design	Registered
United States of America	WAYPOINT	Pending
WIPO	TERRAFORM	Not yet filed
WIPO	TERRAFORM Design	Not yet filed

7. Useful Resources and Writing Guides

- HashiCorp Writing Best Practices Talk
- Infrastructure Cloud design style guide
- Content Types & Channel Menu
- HashiCorp Social Media Guidelines & Best Practices
- Events Verbal Messaging and Style Guide
- HashiCorp numbers for public use
- Stats for content marketing points
- Miriam-Webster Dictionary
- 2020 AP Stylebook
- Apple Style Guide
- Github style guide
- Google Developer Documentation Style Guide
- An Incomplete Guide to Inclusive Language for Startups and Tech
- Technical Writing Workshop by HashiCorp Education Architect Kaitlin Carter
- Engineering Style Guide (requires GitHub access - if needed ask in the #engineering slack channel)
- SEO Best Practices

- Grammarly – use the site, not the plug-in
- How HashiCorp Works - Writing Practices
- Writing an Effective Request for Change (RFC)
- How To Hook Readers With Your Blog Introduction
- How to be concise
- 10 Content Writing Tips
- Becoming a Better Writer as a Software Engineer (but useful for everyone!)
- Blog writing for developers (also useful for all)
- How People Read Online: New and Old Findings
- HashiCorp Acronym List
- Smart Brevity Checklist (Axios)
- Smart Brevity Workbook Bundle (Axios)
- A few examples of communications in Smart Brevity (Axios)
- Infographic Ideas

Books

- *The Elements of Style*
- *Don't Make Me Think*