Governing 176 Research Cloud Projects

Budgets that act as allocations, a throttle that stops spending but keeps data, and an audit engine that can only read, with the gaps still open

Charles E. Forsyth III
Research Computing
University of California, Riverside
Riverside, California, USA
charles.forsyth@ucr.edu

Abstract

Ursa Major is UC Riverside's research program on Google Cloud, run by Research Computing with UCR Information Technology Solutions (ITS). The Research Computing knowledge graph listed 176 Google Cloud projects on 5 October 2026, counting Research Computing's own projects; about 90 lab projects used anything in October. Most labs want to start work today without a purchase order. I want to give them a project the day they ask, with a spending limit already on it.

The controls are a budget set by Research Computing, an automatic throttle that stops running virtual machines (VMs) and AI calls at the cap but deletes nothing, caps on the shared cluster, and a read-only audit engine called Skywalker. They have limits. Google's cost data arrives hours after use, so the throttle acts late; it does not stop storage; I have no record of how often it has fired. In a late-October snapshot nearly 150 budgets existed, and some spending projects had none, a gap we are closing. A 2025 quota tool remains a prototype.

CCS Concepts

• **Computer systems organization** → **Cloud computing**; • **Social and professional topics** → **Management of computing and information systems**; • **Security and privacy** → *Access control*.

Keywords

research computing, cloud governance, budgets, cost control, least privilege, audit, Google Cloud, Model Context Protocol

Author note

This Field Note is dated 29 October 2026. The work was directed by me and carried out with AI agents under written briefs, independent checks and explicit approval steps. Figures come from the Skywalker repository and its tests, the Ursa-Major-Quota-Management repository, the Atrium planning documents, the knowledge graph specification and the public Research Computing site. Some counts are rounded and no lab is named; where a count is rounded, the text says "about". Counts and dates are representative values from the stated snapshots, not continuously maintained figures. This note was drafted with AI assistance from those records and reviewed by me. A web edition is at https://charles-forsyth.github.io/papers/.

Cite this note. C. E. Forsyth III. 2026. Governing 176 Research Cloud Projects. Field Notes.
doi:10.5281/zenodo.23752277

1 Context

A lab usually needs cloud resources for one paper, one grant aim or one semester. If a project takes weeks of paperwork, the student puts it on a credit card or gives up. If it takes an hour and nobody watches it, a forgotten graphics processing unit (GPU) machine runs all summer. I have to avoid both. On-demand self-service and metered billing are what define a public cloud [1], and the same elasticity that lets a lab start in an hour lets its bill grow without a limit [2].

The public *KB005: Ursa Major service tiers* article sets the shape. Tier 1 is campus-supported, with no recharge to the lab under current terms: AI model access with a per-lab allowance, specialized hardware such as Tensor Processing Units (TPUs) or Arm processors, and archive storage. Tier 2 is recharge: all other cloud work, billed to a lab funding source through ITS under a memorandum of understanding (MOU). Tier 3 covers very large or multi-year projects that need their own contract. Projects set up before October 2026 may still sit under the older arrangement, which covered a list of general cloud services.

Research Computing is a small team, and labs own the projects. Governance has to work without a person approving every change.

2 Related Work

The closest research peer is CloudBank, the National Science Foundation (NSF)-funded service that gives computer science researchers and educators managed access to several commercial clouds [3]. Its designers had the same problem as ours: grants and campus purchasing were not built for metered, pay-as-you-go spending, so CloudBank handles billing on the researcher's behalf. Today it monitors spending across clouds and sends automated balance alerts [4]. Ursa Major differs in scale and in who holds the money. It serves one campus on one provider, and for most lab projects UCR ITS pays the bill, so we present the budget as an allocation rather than a balance the lab is spending down.

Industry practice is codified in the FinOps Framework, which describes cloud financial management as shared accountability between engineers, finance and the business, with spending made visible to the people who cause it [5]. Our budget meter follows that idea. Where we part ways is the units: a researcher sees allocation, used and left, while dollar figures stay in the staff view.

The provider gives the building blocks. Google Cloud budgets send alerts at thresholds, but Google says a budget is not a hard cap and advises setting it below available funds because of delays in usage reporting [6]. Google publishes a sample that unlinks billing from a project when a budget notification arrives [7], and now offers spend cap budgets that pause eligible services once gross estimated cost passes the budget [8]. Cost anomaly detection flags spikes without a fixed threshold [9]. Organization policy constraints are preventive: they restrict what resources can be created anywhere in the hierarchy, before any cost exists [10]. Cloud Custodian, an open-source rules engine, expresses filters and actions on cloud resources as policy files [11]. Our throttle sits between Google's blunt sample and the native spend cap: it stops running VMs, turns off two named AI services and never touches storage.

Skywalker's design draws on old security principles. Least privilege and separation of privilege go back to Saltzer and Schroeder [12] and appear as controls in NIST SP 800-53 [13]; Google's own guidance warns against broad basic roles for the same reason [14]. Skywalker applies them twice, through the caller's Identity and Access Management (IAM) roles and through a check in code, and it keeps the reading path apart from the one automatic actor. Its interface follows the Model Context Protocol (MCP) specification, which defines how a client such as an AI assistant discovers and calls tools on a server [15].

On the compute side, the shared cluster runs Slurm [16], and labs whose work fits a Kubernetes cluster [17] can use the National Research Platform (NRP), which grew out of the Pacific Research Platform [18] and offers shared GPUs at no charge to researchers [19]. The NIH STRIDES Initiative is a second program peer: it partners with commercial cloud providers so NIH-funded researchers get discounted cloud services, with monthly spending reports, account monitoring and overage alerts [20]. I found no published account of a campus governing a fleet of this size on a single provider with a non-destructive throttle; that is a result of a limited search, not a novelty claim.

3 The Rules I Govern By

Four rules sit behind every control below.

(1) **Throttle the one thing.** Google Cloud's blunt tool is to unlink the project from billing; Google publishes a sample that does exactly that from a budget notification [7]. It stops the spending, and it stops everything else too, puts data at risk, and is a pain to undo. I would rather stop the running VMs or turn off one expensive application programming interface (API). Usually it is one service running away.
(2) **Then stop for real, after data is safe.** A cap that only sends email does not stop spending. When a project hits its limit, work stops, but only in ways that cannot lose what a student or a course needs. The throttle never deletes anything, and that is why I am comfortable automating it.
(3) **Treat a budget as an allocation.** For most lab projects UCR ITS pays the bill. Researchers should see how much of their allocation is left, in the units they think in.
(4) **Plan on how fast a project could spend.** Inside Research Computing I look at daily rate and the week-over-week trend more than at last month's total.

4 The Controls, Layer by Layer

Table 1 lists the controls in order of visibility.

4.1 Budgets and the throttle

The public article on project budgets and spending alerts states the policy. Every Ursa Major project can have a budget, Tier 1 and Tier 2. Research Computing sets budgets up; project owners do not. On a Tier 2 project the budget matches the lab's funding on request. On a Tier 1 project the budget is the project's Tier 1 limit.

The check runs each time Google Cloud updates the project's costs, several times an hour. When reported spending for the period reaches the budget, running VMs are stopped and Vertex AI and the Gemini API are turned off for the project. Disks, Cloud Storage buckets and the project itself are left alone.

Stopping a VM costs time and nothing else, though anything held only in memory is lost. Turning off the AI APIs makes calls fail, which a researcher notices in minutes, but their code and data stay put. I picked those two because, in my experience, that is where runaway bills come from: a VM nobody shut down, or a script calling a model in a loop. I have not measured that against the billing export.

Raising the budget does not turn anything back on. The lab tells us what it needs running, and we restart it, so nothing comes back on by accident through the budget path.

A Google budget is not a hard cap. Google's budget documentation says there is a delay between using a resource and its cost reaching Cloud Billing, and recommends setting a budget below the funds you actually have [6]. The throttle can only act on cost Google has reported, so a GPU machine or a model loop keeps running through that delay. I do not yet have a measured figure for how far past the cap a project has gone before the throttle acted.

4.2 What the throttle has to reach, and what can undo it

Stopping a VM is not always the end of it. A managed instance group (MIG) recreates or restarts instances to hold its target size [21]. A Google Kubernetes Engine (GKE) node pool is a MIG underneath, and its autoscaler can add nodes back [22]. A Slurm cluster that uses power saving, as the shared cluster's Tier 1 partitions do, starts nodes on demand when jobs arrive [23]. In each case a stopped VM can come back without anyone raising a budget. The real fix is to scale the group, pool or partition to zero, or to block new instances, not to stop the instances one by one. The public article describes only the VM stop, and I have not yet verified how the throttle behaves against each of these.

The throttle also runs as a service account that needs the right to stop instances and to turn services off in every budgeted project. It holds broad privileges. Its permissions should be limited to those two actions, granted only on Ursa Major projects, and its use logged [14]; who can act as that account matters as much as what it can do. Writing that review down is listed under What's Next.

4.3 Caps on the shared cluster

Labs that need TPUs, Arm or other specialized hardware are meant to move onto the shared Ursa Major Slurm cluster, which Research Computing runs in Google Cloud. Its Tier 1 `arm`, `tpu` and `tpuspot`

Table 1: The controls from what a lab sees to what only staff see. None requires a lab to ask before using its project; they act on limits set up front. The quota row is a prototype, not a control in use.

Layer	What it does	When it acts	Status
Tier and terms	Decides who pays for what; sets Tier 1 limits per project	At setup	In use (KB005)
Budget and alerts	A cap per project, set by Research Computing; alerts at thresholds the lab chooses	As reported cost nears the cap	In use where a budget exists
Throttle	Stops running VMs, turns off Vertex AI and the Gemini API; keeps disks, buckets and the project	When reported cost reaches the cap, hours after use	In use where the budget is wired to it
Job caps	Limit nodes, hours and spend per job and per day on the shared cluster	Before a job reaches the scheduler	In use, light traffic so far
Audit	Reads spend, budgets, exposure, access and idle resources across the fleet	On demand, read-only	In use (Skywalker)
Quotas	Would narrow what a project can create, service by service	Set by staff	2025 prototype, not in use

partitions, built in October 2026, power down to zero when idle, and the project budget covers their Compute Engine spend. Researchers reach the cluster through ursa-bifrost, our job tool and MCP server, a standard interface that lets AI assistants call tools [15]. Its caps limit nodes, hours and spend per job and per day, and a job over a cap is refused before it reaches the scheduler, the same idea as the throttle applied to one job.

5 Skywalker: Seeing the Fleet

Skywalker is how I check whether the other controls are doing their job; Figure 1 shows how it and the throttle connect. It began on 18 December 2025 as a command-line audit tool and became an MCP server on 4 October 2026 (`v0.34.0`), with per-person project scope in `v0.35.0` the same day.

5.1 Read-only, enforced in code

The MCP server acts as the person asking. Each person signs in with Google, and every tool call runs on their own access token, so they see what their account can already see and Google's audit logs name them. The server's own service account has no Google Cloud data roles.

The broad `cloud-platform` scope is needed because Google's read-only scopes do not cover budgets, asset search, IAM, Recommender, quotas or Security Command Center. Because the scope is broad, I added a second limit in code. `check_read_only` in `src/skywalker/intel/gcp.py` refuses any request that is not a GET or one of three read calls (`getIamPolicy`, Logging `entries:list`, and a single BigQuery `SELECT`) before it leaves the process. There are no write tools.

That check is defense in depth. The real limit is the caller's own IAM roles, and a pattern check on SQL is not a parser. Read-only is also not free: a BigQuery `SELECT` against the billing export is billed by the bytes it scans [24]. The repository holds 134 test functions, and a mutation check script lists 37 deliberate code changes, each of which must make at least one test fail.

5.2 What it answers

The server has 21 tools. Seventeen are open to anyone on the user list, limited to their own projects; four are staff-only because they read the whole billing account or every project. The one I use most is `skywalker_overview`: findings first, then spend with a 7-day

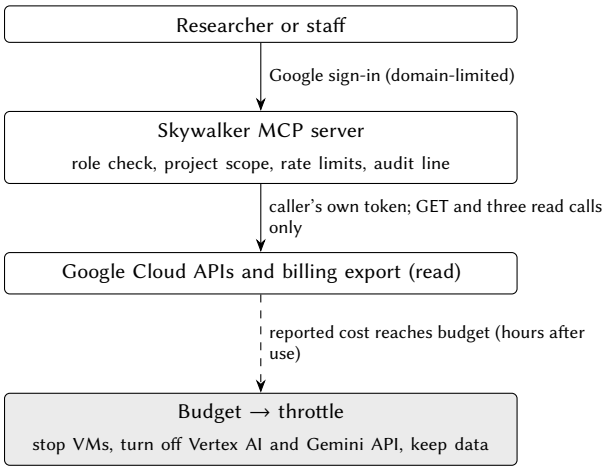


Figure 1: Reading and acting are separate paths. Skywalker only reads, as the person asking; the throttle (shaded) is the only automatic actor, it stops services without deleting anything, and it acts on reported cost, which trails real use.

trend, budget, VMs and GPUs, idle disks and addresses, risky IAM grants, public exposure and recent admin changes. The staff tool `skywalker_budgets` marks every budget as over (100% or more), warning (75% to 100%) or unbudgeted, and counts budgets with no throttle topic attached. That count is the most useful thing it reports, because a budget with no throttle topic only sends email and looks the same as any other budget on a list.

Two older bugs shaped how it scopes queries. In June 2026 one fix stopped metric queries under a shared monitoring scope from counting one project's usage against another, and another made the fleet-wide model, API key and idle-resource audits refuse to run without an explicit scope. Both errors would have produced numbers that looked right and were not.

6 What the Fleet Looks Like

Table 2 summarizes the fleet from those scans. The counts come from different tools and do not reconcile into one ledger. The knowledge graph includes Research Computing's own projects and some records for projects that no longer run; budgets can sit on idle or

Table 2: Shape of the fleet from read-only scans in late October 2026. Counts other than the first cover lab projects only, with Research Computing's infrastructure projects excluded; counts are rounded and no lab is named.

Measure	Count
Google Cloud projects in the knowledge graph, all owners (5 October)	176
Lab projects with any use in October	about 90
Budgets on the billing account	149
Lab projects with VMs running at scan time	about 30
Lab projects holding compute resources	about 45

older projects, so 149 budgets and about 90 active projects do not contradict each other.

Spending is concentrated. The top 10 projects made up 53% of October's gross use, the top 20 made up 76% and the top 30 made up 89%. About 20 to 30 labs matter for any decision about money; the rest is a long tail of small or idle projects.

Of the projects holding compute, a handful run their own cluster, a similar number use GPUs or TPUs without one, and most run only central processing unit (CPU) VMs. Most own-cluster labs run older per-lab Slurm deployments with an always-on controller and login node. They are the natural first movers onto the shared cluster, where that always-on pair goes away. Some GPU work may fit elsewhere, such as the free NRP Nautilus cluster [19] or the campus AI gateway for model calls.

7 Budgets as Allocations

The control researchers actually see is the budget meter, and it changed in October 2026. Atrium is the researcher portal Research Computing is building over its MCP servers. In late October a read-only review found that it put cost at the centre of almost every page: on the home page, three of the five headline numbers were money. That is the wrong message for a lab whose budget UCR ITS pays. On 27 October I recorded four decisions in the Atrium specification:

- **Work first, money second** (D17). Researcher views lead with jobs, keys and what is running. Money is shown against an allocation, in dollars.
- **Budgets are allocations** (D18). The words are "allocation", "used" and "left". Secure projects for sensitive data and recharge services are real money and stay in plain dollars.
- **Most projects move** (D19). AI work moves to the AI gateway and general compute elsewhere; only the specialized-hardware cases stay, through the shared cluster.
- **Never invent an allocation** (D20). Allocations on the shared cluster are per lab and per person, warn-only for now, while the job caps still refuse. Projects without a budget show usage with no meter until one is set.

Interrupts still happen, at thresholds: Atrium's "Needs you" list fires at 80% of a budget. Research Computing's own costs, such as the cluster's hourly rate and fleet totals, are visible only in the staff view, and every dollar figure stays one click away.

8 Other Ways to Do This

Table 3 lists the alternatives I weighed.

The biggest lesson from the comparison is that most of what we do is reactive. Organization policies and budgets created with the project stop problems before any cost is reported, which matters given the billing delay. Google now offers spend cap budgets that pause eligible services natively, and cost anomaly detection that watches rate rather than total. Cloud Custodian would put the throttle's logic in policy files a reviewer can read. CloudBank monitors spending across clouds and sends automated balance alerts. I built Skywalker because I wanted one read-only view, acting as the person asking, that labs and staff could both use through an AI assistant; Custodian, Recommender and Security Command Center each cover part of that.

9 What Went Wrong

- **Projects with no budget.** The throttle only acts where a budget exists. The late-October snapshot found roughly 45 projects with spend in October and no budget, a different set from the about 45 holding compute. We are adding budgets, largest spenders first. Until each has one, D20 shows its usage without a meter, which keeps the screen honest and does not cap the spending.
- **The cap trails the spending.** Budgets act on reported cost, which arrives hours after use [6]. The cutoff acts hours after the spending, and I have no measured overshoot figure or count of throttle events to publish.
- **Credits are a budget setting we chose.** Google budgets can count cost before or after credits. Ours count gross cost, before credits, as the public article says, so a project can reach its cap even when credits would cover most of the bill. That keeps the cap tied to real usage; it also means the number on the budget and the net bill differ, and we have to explain both.
- **The throttle does not stop storage.** Because nothing is deleted, persistent disks and stored data keep billing after the cap, so the cap is not a ceiling. I accept that as the price of never losing data.
- **Stopped machines can come back.** Managed instance groups, GKE autoscaling and Slurm power saving can restart what the throttle stopped. I have not verified the throttle's handling of each.
- **A quota tool that stayed a prototype.** `qm.py` was written over 2 days in May 2025 and nobody has touched it since. About 12 of its 30 metric names still carry comments like "needs verification" or "inferred". It runs one project at a time, through `gcloud`, by hand, and it is not part of how we govern projects today.
- **The shared cluster is not yet the default.** Use through bifrost in the 30 days before the scan was minimal, mostly staff testing. The migration in D19 is a plan, not a result.

10 Lessons

(1) **Stop the service, keep the data.** Labs accept an automatic cutoff when it cannot lose data.

Table 3: Approaches I compared against, and where each would change what we do.

Approach	What it gives	Where it fits here
Budget created with every project	No project exists without a cap	The structural fix for the unbudgeted projects; budgets are a separate step today
Organization policy constraints [10]	Preventive limits on regions, resource types and service account keys, with no billing delay	Not used for cost yet; the best candidate to replace the quota prototype
Google spend cap budgets [8]	Native pause of eligible services at the cap, on gross estimated cost	Worth testing against our own throttle for the services it covers
Google cost anomaly detection [9]	Flags unusual spend without a fixed threshold	Matches the burn-rate rule; complements Skywalker's 7-day trend
Cloud Custodian [11]	Open-source policy as code: find, stop or tag resources by rule	Could run the throttle and the audit rules as reviewed policy files
CloudBank [3]	NSF-funded cloud access with monitored spending and balance alerts	The closest research peer; its alerts model is close to ours

(2) **Use the narrowest control that stops the harm.** Unlinking billing stops everything, so it gets used late or never.
(3) **Create the budget with the project.** These projects slipped through because the budget was a separate step from the project.
(4) **Check that every budget is wired to a throttle.** A budget with no throttle topic only sends email, and it looks the same on a list.
(5) **Treat a budget cap as late.** Billing data trails use by hours, so set the cap below the money you have and add preventive limits.
(6) **Enforce read-only in code as well as in IAM.** Cloud scopes are coarse, and a check before the request leaves the process is one you can test.

11 What's Next

- Put a budget with a throttle on each unbudgeted project, largest first, and make budget creation part of project creation.
- Measure the gap between crossing a cap and the throttle acting, and start counting throttle events; a quarter's event count and median overshoot past the cap would settle whether the throttle works.
- Check the throttle against instance groups, GKE and Slurm power saving, and write down its service account's permissions and who can use it.
- Test organization policy constraints and Google's spend cap budgets as preventive layers, in place of the quota prototype.
- Move the older per-lab Slurm clusters onto the shared cluster, where job caps and allocations already apply.

Sources

Public repositories and pages: the Skywalker repository (https:/ / github . com / charles-forsyth / Skywalker: README, git history, tags, tests and the mutation check script) and its showcase page; the ursa-bifrost repository (https://github.com/UCR-Research-Computing/ursa-bifrost); the Ursa Major service page (https:/ / ucr-research-computing . github . io / services / ursa-major/); KB005, Ursa Major service tiers (https:/ / ucr-research-computing . github . io / kb / kb005-ursa-major-service-tiers/); and Ursa Major project budgets and spending alerts (https:/ / ucr-research-computing . github.io/kb/ursa-major-project-budget/).

An earlier Field Note, *Directing a Fleet of AI Builders* [25], describes how I direct AI agents.

Private records, by name: the Ursa-Major-Quota-Management repository (`qm.py` and its README); the ucr-slurm-production repository, the shared cluster's configuration of record, and its Tier 1 Arm and TPU plan; the Atrium specification (decisions D17 to D20), its allocation framing plan and its 27 October 2026 compute scan; and the knowledge graph specification's live-state counts.

References

[1] Peter M. Mell and Timothy Grance. 2011. *The NIST Definition of Cloud Computing*. Special Publication 800-145. National Institute of Standards and Technology. doi:10.6028/NIST.SP.800-145

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A View of Cloud Computing. *Commun. ACM* 53, 4 (2010), 50–58. doi:10.1145/1721654.1721672

[3] Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, Katherine Yelick, Eric Van Dusen, and James Mitchell. 2021. CloudBank: Managed Services to Simplify Cloud Access for Computer Science Research and Education. In *Practice and Experience in Advanced Research Computing (PEARC '21)*. ACM, New York, NY, USA, 1–4. doi:10.1145/3437359.3465586

[4] CloudBank. 2026. CloudBank: Cloud Access for Research and Education. Retrieved October 28, 2026 from https://www.cloudbank.org/

[5] FinOps Foundation. 2026. FinOps Framework. Retrieved October 28, 2026 from https://www.finops.org/framework/

[6] Google Cloud. 2026. Create, Edit, or Delete Budgets and Budget Alerts. Cloud Billing documentation. Retrieved October 28, 2026 from https://cloud.google.com/billing/docs/how-to/budgets

[7] Google Cloud. 2026. Disable Billing Usage with Notifications. Cloud Billing documentation. Retrieved October 28, 2026 from https://cloud.google.com/billing/docs/how-to/disable-billing-with-notifications

[8] Google Cloud. 2026. Spend Cap Budgets. Cloud Billing documentation. Retrieved October 28, 2026 from https://cloud.google.com/billing/docs/how-to/budgets-spend-caps

[9] Google Cloud. 2026. View and Manage Cost Anomalies. Cloud Billing documentation. Retrieved October 28, 2026 from https://cloud.google.com/billing/docs/how-to/manage-anomalies

[10] Google Cloud. 2026. Introduction to the Organization Policy Service. Resource Manager documentation. Retrieved October 28, 2026 from https://cloud.google.com/resource-manager/docs/organization-policy/overview

[11] Cloud Custodian Project. 2026. Cloud Custodian. Retrieved October 28, 2026 from https://cloudcustodian.io/

[12] J. H. Saltzer and M. D. Schroeder. 1975. The Protection of Information in Computer Systems. *Proc. IEEE* 63, 9 (1975), 1278–1308. doi:10.1109/PROC.1975.9939

[13] Joint Task Force Interagency Working Group. 2020. *Security and Privacy Controls for Information Systems and Organizations.* Special Publication 800-53 Rev. 5. National Institute of Standards and Technology. doi:10.6028/NIST.SP.800-53r5

[14] Google Cloud. 2026. Use IAM Securely. Identity and Access Management documentation. Retrieved October 28, 2026 from https://cloud.google.com/iam/docs/using-iam-securely

[15] Model Context Protocol. 2025. Model Context Protocol Specification, Version 2025-06-18. Retrieved October 28, 2026 from https://modelcontextprotocol.io/specification/2025-06-18

[16] Andy B. Yoo, Morris A. Jette, and Mark Grondona. 2003. SLURM: Simple Linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing (Lecture Notes in Computer Science).* Springer, Berlin, Heidelberg, 44–60. doi:10.1007/10968987_3

[17] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, Omega, and Kubernetes. *Queue* 14, 1 (2016), 70–93. doi:10.1145/2898442.2898444

[18] Larry Smarr, Camille Crittenden, Thomas DeFanti, John Graham, Dmitry Mishin, Richard Moore, Philip Papadopoulos, and Frank Würthwein. 2018. The Pacific Research Platform. In *Practice and Experience on Advanced Research Computing (PEARC '18).* ACM, New York, NY, USA, 1–8. doi:10.1145/3219104.3219108

[19] National Research Platform. 2026. National Research Platform. Retrieved October 28, 2026 from https://nationalresearchplatform.org/

[20] National Institutes of Health, Office of Data Science Strategy. 2026. STRIDES Initiative. Retrieved October 28, 2026 from https://datascience.nih.gov/strides

[21] Google Cloud. 2026. Instance Groups. Compute Engine documentation. Retrieved October 28, 2026 from https://cloud.google.com/compute/docs/instance-groups

[22] Google Cloud. 2026. About GKE Cluster Autoscaling. Google Kubernetes Engine documentation. Retrieved October 28, 2026 from https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-autoscaler

[23] SchedMD. 2026. Slurm Power Saving Guide. Retrieved October 28, 2026 from https://slurm.schedmd.com/power_save.html

[24] Google Cloud. 2026. BigQuery Pricing. Retrieved October 28, 2026 from https://cloud.google.com/bigquery/pricing

[25] Charles E. Forsyth III. 2026. Directing a Fleet of AI Builders. Field Note. Retrieved October 28, 2026 from https://charles-forsyth.github.io/papers/2026-10-27-directing-a-fleet-of-ai-builders.html