From Corp to Cloud
Google's Virtual Desktops

HOW GOOGLE MOVED ITS VIRTUAL DESKTOPS TO THE CLOUD

MATT FATA
PHILIPPE-JOSEPH ARIDA
PATRICK HAHN
BETSY BEYER

Over one-fourth of Googlers use internal, data-center-hosted virtual desktops. This on-premises offering sits in the corporate network and allows users to develop code, access internal resources, and use GUI tools remotely from anywhere in the world. Among its most notable features, a virtual desktop instance can be sized according to the task at hand, has persistent user storage, and can be moved between corporate data centers to follow traveling Googlers.

Until recently, our virtual desktops were hosted on commercially available hardware on Google's corporate network using a homegrown open-source virtual cluster-management system called Ganeti (http://www.ganeti.org/). Today, this substantial and Google-critical workload runs on GCP (Google Compute Platform). This article discusses the reasons for the move to GCP, and how the migration was accomplished.

WHY MIGRATE TO GCP?

While Ganeti is inexpensive to run, scalable, and easy to integrate with Google's internal systems, running a do-it-yourself full-stack virtual fleet had some notable drawbacks. Because running virtual desktops on Ganeti entailed managing components from the hardware up to the VM manager, the service was characterized by:

- Long lead times to expand fleet capacity.
- Substantial and ongoing maintenance overhead.
- Difficulty in staffing the team, given the required breadth and depth of technologies involved.
- Resource waste of underlying hardware, given working hours for a typical Googler are 8-10 hours a day.
- Duplication of effort in the virtualization space at Google across multiple divisions.

Taken together, these issues reduced the time and resources available to the team tasked with improving the offering. To tackle these problems, the team began migrating Google's top corporate workload to GCP in early 2016.

PLANNING

Planning for the migration consisted of several discrete stages. In aggregate, the planning phases described here took approximately three to four months and involved the participation of roughly 15 subject-matter-expert groups.

Vision

This phase articulated the core business and engineering case for replatforming our virtual desktops to GCP. The top reasons for pursuing virtual desktops on Cloud included:

- Large reduction in engineering toil.
- Improved user experience.
- Reduced total cost of platform ownership.
- Desire to improve GCP.

Customer user journeys

The next step was to study the users and their needs. The primary users—the virtual desktop owners, fleet SREs (Site Reliability Engineers), and the fleet security managers—were identified and surveyed to determine their typical workflows. Using this information, the migration team wrote implementation-agnostic user journeys. To perform effective gap analysis, and to reduce bias during the design phase, the team made a conscious effort to describe user journeys in a purely functional fashion.

Product milestone definition

Based on the survey responses and usage patterns collected, the team grouped customer user journeys (by both technology area and user type) and prioritized them into bands of features labeled alpha, beta, and general availability.

Workstream definition

In parallel to milestone definition, the team grouped requirements into seven streams of related work such as networking and provisioning. Each workstream was assigned a technical lead, a project lead, and a skeleton staff. Each team was virtual, recruited from across reporting lines as needed to address the work domain. The flexibility provided by this form of organization and

associated matrix management turned out to be essential as the project evolved.

Engineering prototyping, gap analysis, and design proposals

Once formed, each workstream examined the critical user journeys in their domains and researched the feasibility of implementing these stories on GCP. To do so, the team performed a gap analysis for each user journey by reading product GCP documentation and running "fail-fast" prototyping sprints. Throughout this process, possible implementations were collected and rated according to complexity, feasibility, and (most importantly) how easily a customer external to Google could implement this solution.

Whenever the migration team arrived at a "Google-only" solution, it filed a feature request to the GCP team requesting a solution that would work for customers outside of Google as well, especially if another enterprise customer would be interested in such functionality. In this way, the team sought to "act like a customer" in an effort to make the platform enterprise-ready. Where the GCP product teams could not deliver a feature in time for a release milestone, they implemented bridging solutions that favored solutions that the public could use (for example, Forseti Security) above Google-only workarounds.

Workstream work breakdown and staffing

With design proposals in place and implementation directions decided, the team created detailed work plans

for each workstream in a central project management tool. The work was organized by customer user journey, and tasks were broken down by objective, key results, and quarter. Drilling down to this level of detail provided enough information to estimate the staffing required for each workstream, to understand interdependencies between streams, and to fine-tune the organization as needed.

TECHNICAL IMPLEMENTATION DETAILS

Once planning was complete, the team was ready to begin implementing the technical details of the migration. This section describes the three main buckets of work.

Background: networking and BeyondCorp

Many of the networking challenges of running a desktop service on GCE (Google Compute Engine) were at least partially solved by the BeyondCorp program (https://cloud.google.com/beyondcorp/). In a BeyondCorp model, access controls are based on known information about a given device and user rather than on the location in a privileged network. When network trust no longer factors into access-control decisions, many services become readily accessible from outside of the corporate network—usually via a corporate laptop, but now also from appropriately managed and inventoried hosts on GCE.

Enterprises that leverage traditional VPNs (virtual private networks) for remote access to applications will have a different networking experience when moving desktops or other services. A typical strategy is to set up a Cloud VPN (https://cloud.google.com/compute/docs/vpn/

overview) in a cloud project and peering with on-premises equipment to bridge the networks together.

Host authentication and authorization

Device authentication is usually performed using client certificates deployed on the host. When a user receives a physical machine (or even a virtual machine on privileged corporate networks), the user can initially log in and request a certificate because the corporate network retains the level of privilege needed to sync login policies. Extending this level of network privilege to public cloud IP ranges is undesirable for security reasons.

To bridge this gap, Google developed a (now-public) API to verify the identity of instances (https://cloud.google.com/compute/docs/instances/verifying-instance-identity). The API uses JWTs (JSON web tokens; JSON is the JavaScript Object Notation data-interchange format) to prove that an instance belongs to a preauthorized Google Cloud project. When an instance first boots, one of the JWTs provided by that API can be exchanged for a client certificate used to prove device identity, unblocking nearly all of the normal communication paths (including syncing login policies for user authorization).

Once the client certificate is in place, Google applications can be accessed via the BeyondCorp/identity-aware proxy as if they were any other Internet-facing service.

Once the client certificate is in place, Google applications can be accessed via the BeyondCorp/identity-aware proxy as if they were any other Internet-facing service. In order for cloud desktops to reach the proxies (and other Internet endpoints), the team set up NAT (network address translation) gateways (https://cloud.google.com/compute/docs/vpc/special-configurations) to forward traffic from the instances to targets outside of

the cloud project. In combination, these approaches allow users to access internal resources and the public Internet seamlessly, without requiring each instance to have a publicly routed IP address.

Provisioning

The first step in designing a provisioning scheme was to map out everything necessary to deliver an end product that met users' needs. Compute Engine instances needed levels of trust, security, manageability, and performance for users to perform their jobs—developing, testing, building, and releasing code—as normal. Working from these requirements, the team used the following specific principles to guide the rest of the design.

Users should interact with a cloud desktop similarly to how they interact with hosts on the corporate network. Users should be able to use their normal authentication mechanisms. They should also be able to use the same tools to check machine statistics or report issues that they would use for physical desktops or Google's legacy virtual desktop platform.

Instances must be securely inventoried. As a first step in the provisioning process, host inventory is bootstrapped. As the hosts are used, further inventory data is collected, both for reliability monitoring and to inform access-control decisions.

Google's corporate network uses multiple inventory systems, cross-referenced to validate such access requests (for more context on BeyondCorp's inventory process, see BeyondCorp: A New Approach to Enterprise Security [https://research.google.com/pubs/pub43231.

html] and BeyondCorp: Design to Deployment at Google [https://research.google.com/pubs/pub44860.html]). Therefore, corporate systems need some metadata during the provisioning process to indicate a privileged virtual desktop creation request. This metadata is then cross-referenced with inventory data pulled from Google Cloud APIs in order to evaluate compliance with security policy and assert trust.

Instances must be securely managed. A host must be able to securely download the information needed to construct its authorization policies such that only permitted users have access. Hosts must also be able to update packages and configurations driven by the operating system installation and must be able to send logs to a central location so irregular behavior or installation-related issues can be detected.

Instances must be created to user specifications, with constraints. Instances should be provisioned with enough resources for users to do their work effectively but should also have caps to prevent users from gratuitously creating high-specification/high-cost devices. Users can choose where to build their instances (typically, in regions close to their physical location; disaster-recovery reasons might dictate a different location).

Because of the need to manage and inventory the devices in corporate systems, there were two options for creating new instances once Desktop was migrated to Cloud:

- Creating cloud desktop instances on behalf of the requester.
- Allowing users to interface natively with Cloud to create

and manipulate their instances. In this scenario, it would be necessary to observe these changes and make corresponding updates in the corporate tools.

The team decided on the first approach and integrated existing virtual-machine management tools with Compute Engine. As a result, they could enforce more complex business logic in line with the user's request.

The workflows built around the Compute Engine provisioning process focus mostly on translating data provided by the requesting user, plus data known about the user (group membership, job role), into a request that can be passed to Compute Engine. The instance can then be tracked through the creation process, as well as during any first-boot operating-system updates and configuration, to make sure a usable machine is delivered to the user.

Operating system

Google uses an internally managed Linux distribution based on Debian, called gLinux, for corporate hosts. On its corporate network, gLinux is installed by loading a bootstrap environment on first boot. Large parts of the root file system are unpacked from a tarball, and the Debian installer then performs the actual installation.

On Cloud, this process starts with an image created by gLinux release tooling, which is uploaded to Cloud storage and imported into GCE (https://cloud.google.com/compute/docs/images/import-existing-image). Creating a disk from this image results in a fully runnable and bootable file system. You can boot directly from the imaged disk and only need to perform some small modifications on first boot before it's fully usable: the file system grows to fill

the full span of the disk, the hostname updates, and a few other GCE-specific modifications are performed.

To avoid the burden of maintaining and testing separate behavior on different platforms, the team needed to minimize specific customizations to gLinux on Compute Engine. Fortunately, this effort required very few modifications, most of which focused on DNS (Domain Name System) name resolution. For example, corporate DNS zones, which many applications running on Google's corporate network require, are not available off-network. To address this need, the team introduced a DNS resolver that runs inside each instance to proxy requests for internal DNS zones back to the corporate servers over HTTP through a BeyondCorp proxy.

ALPHA AND BETA ROLLOUTS

Before beginning the migration to GCP in earnest, the team conducted alpha and beta rollouts as initial sets of features were ready. The alpha release targeted roughly 100 users, while the beta release targeted roughly 1,000 users.

To evaluate the success of both releases, the following metrics were tracked and compared with Google's existing corporate fleet statistics:

➡ *Pager load.* There was a 95 percent drop in pager load once we migrated virtual desktops to Cloud, in large part because of the platform abstraction provided by GCP. While the team maintained a large fleet of physical servers, storage units, network equipment, and support software to run virtual desktops on the corporate network, GCE removed all these concerns.

➡ *Interrupts load.* Initially, the migration to Cloud led to an increase in interrupts as a result of the novelty of the system (which also resulted in a corresponding increase in product bug reports). After this initial surge, interrupts load dropped to just 20 percent of the volume experienced when virtual desktops were hosted on the corporate network.

➡ *Login rates.* Seven- and 30-day login rates before and after the migration were comparable.

To inform the roadmap for future milestones, the team also collected data during alpha and beta releases via two main avenues:

➡ *User-reported feedback.* User feedback in the form of tickets, bug reports, emails, and word of mouth provided a list of items to fix. The team filed bugs to track each list item, prioritized by severity, number of users impacted, and aggregated customer preference. The last metric was made measurable by offering power users 100 "feature points" to apportion across features as they wished. These metrics could then be used to inform development priorities.

➡ *Surveys.* Surveys measured subjective impressions from the user base with the goal of improving marketing. For example, subjective feedback on the relative performance of virtual desktops on Cloud versus the corporate network was split (when, in fact, performance on Cloud was superior). In response, the team emphasized and more heavily promoted benchmarking results to customers, with the hope of prompting more objective valuations. Surveys also helped quantify fears about the transition, most notably around performance, user-data

migration, and the ability to roll back to the corporate network-hosted offering when a given user workflow was not supported.

Based on survey feedback, the team emphasized the following aspects to make the migration to Cloud attractive to users:

➡ *Improved VM specs.* Users were offered large increases in standard CPU, RAM, and disk specs.

➡ *One-click personal data migration.* The migration process was easy to begin with, and it automated the most time-consuming part of users' workflow: copying over personal data.

➡ *Easy rollback.* The migration process allowed users to roll back to their Corp-hosted instances, which were simply shut down before the migration to Cloud. Unused Corp-hosted instances were deleted after a grace period of 90 days.

➡ *Impact on the company.* Clearly articulating the cost reduction to Google helped reassure users they were doing the right thing for the company.

MIGRATING USERS: TECHNICAL AND PROCESS DETAILS

After collecting data and feedback from alpha and beta rollouts, the team was ready to proceed with the general migration to Cloud. This section details the main features of the migration process.

Trade-in

Once the provisioning system was ready for new users, approximately 20,000 virtual desktop users had to be moved to the new product. The team briefly considered

a naive strategy of simply moving each user's disks into a new Cloud instance, but experience in managing the existing platform pointed in a slightly different direction.

Occasionally, the old platform experienced a fault that required significant work to return a user's instance to full functionality. This fault became more common as incremental, automatic modifications to an instance accrued. To spare unnecessary toil, the team's default first response in these scenarios was to ask users if they needed the data on the disk in question. Much of the time, the answer was no, as users didn't have any important data on their disks. Following this strong (but anecdotal) signal from corporate network-hosted virtual desktops, the team based the move to Cloud around the concept of user-involved *trade-ins*, as opposed to a traditional behind-the-scenes *migration*.

To carry out such a trade-in program, the team crafted two workflow pipelines: one to handle cases where users explicitly indicated that they did not need their data moved (or could move it themselves); and one for outliers who needed all of their data moved. The former (and by far, most common) case required simply performing two straightforward tasks: powering off the original instance and creating a new instance with a similar name. This approach required minimal engineering effort; used minimal compute, bandwidth, and storage resources; and provided a strong signal for how much benefit users gathered from traditional stateful disks.

Exceptional cases

Google employs a great many engineers; if you give an

engineer a Linux machine, there's a good chance that the machine will be customized within an inch of its life. For these cases, the team wanted to provide a path to use the new platform that did not force all users to migrate their own data. However, the team was wary of setting a "magical migration" expectation for edge cases that could not possibly be fulfilled.

For users who requested data migration, the best option was to move their home directories in-place to the new cloud instances and provide an on-disk backup of their entire operating system. While this strategy duplicated a significant amount of operating-system data already present on the gLinux system, it meant the team could proceed without worrying that important files were not transferred, only to be noticed months later.

The team was wary of setting a "magical migration" expectation for edge cases that could not possibly be fulfilled.

Moving bits

The actual uploading proceeded in a straightforward manner. For each user request, a job execution system crafted a signed Google Cloud Storage URL entitling the bearer to perform an HTTP PUT request to a bucket for a period of time. To ensure that long-running upload jobs were not interrupted as the workflow system deployed, an Upstart script on the old virtualization platform processed the upload. Upon upload completion, a cloud worker instance fired up to create a new disk image by merging the user's data onto a copy of the golden master image.

Push vs. pull

The beta phase revealed a high demand for virtual desktops hosted on Cloud, so the team wanted to make

sure that the general launch adequately anticipated demand. To avoid overwhelming the network links on the old virtualization platform and the team's capacity for toil in handling a surge of requests, users were not allowed to request trade-ins themselves. Instead, trade-ins were first offered to a population of users who were most likely to need them: those with low disk usage who likely wouldn't need to move their entire disk, and users whose instances were hosted on old hardware. In this way, the team could both balance capacity limits and precisely target user populations whose machine locations would buy the most reduction in toil. Once users actually started using the platform, high demand for cloud desktops meant that users wanted to opt in earlier. Attempts to create new cloud desktop instances without an explicit invitation to do so were a signal to send those users a trade-in invitation.

EARLY KNOWN ISSUES/LIMITATIONS

While initial reception of the new service was positive, a few barriers caused mild inconvenience to users and some completely broken use cases.

Most of these pain points affected the provisioning process and were largely caused by misunderstanding the various SLOs (service-level objectives) and delays within Google's inventory and trust pipeline. The team required an independent signal of user intent in order to grant trusted access to a new desktop instance, and once a user provided this signal by "enrolling" the instance (post-creation), the user still needed to wait several hours before the instance became fully trusted by all systems. Provisioning and testing an instance therefore took three user interactions

over a period of three to five hours. Once the team became aware of this issue, they made changes to the trust pipeline and folded the enrollment step into the initial user request, thereby eliminating the hours of waiting. If the team had considered the timing of provisioning and other operations as building requirements, they could have made these improvements earlier.

Broken use cases ended up flushing out many bad assumptions that various applications and groups of users made about network access. For example, instead of using a library to check if a user is on the corporate network or the production network, some suboptimal implementations instead depended on hostname or IP space. These issues were typically addressed by updating the code of individual applications to remove the bad assumptions.

There were also issues with some workflows that technically violated security policy but had been granted exceptions. Cloud desktop enforces these policies by default; it encourages users to fix their workflows rather than carry forward bad practices. For example, on Google's corporate network, users can get an exception to connect directly to some application databases. This is practical because the database server and the user are on the same network. These sorts of use cases should be steered toward BeyondCorp gateways.

LESSONS LEARNED

As with any complicated launch, the development team learned a number of lessons along the way, both technical and nontechnical.

Moving disks into the cloud is convenient for users but much more time-consuming (and costly) than the alternative.

Technical

Being able to control the flow of traffic to your system makes operating it infinitely simpler. Find a bug? Stop sending invites. Everything humming along? Turn up the volume. Even if integrating this functionality is difficult, it's worth implementing from square one if possible.

If you offer two options, and one seems like less work, everyone will choose that easier option. To offset this impulse, if one option is much more costly, expose that cost at the user decision point. For example, moving disks into the cloud is convenient for users but much more time-consuming (and costly) than the alternative. The team exposed the cost of moving disks as a 24-hour duration, which was much less convenient than the one-hour duration for a simple exchange of a corporate network-hosted instance for a Cloud-hosted instance. Simply exposing this information when users had to choose between the two options saved an estimated 1.8 petabytes of data moves.

Before the migration, the team didn't know what proportion of users depended heavily on the contents of their local disks. It turns out that only about 50 percent of users cared enough about preserving their disks to wait 24 hours for the move to complete. That's a valuable data point for future service expansions or migrations.

Your future self will be thankful if you take the opportunity to homogenize when making lasting changes. Previous generations of the corporate network-hosted virtual desktop system had a slightly different on-disk layout than the current models used for testing. Not only was this an unpleasant surprise in production, but it was

also almost impossible to test since no existing tools would create the old disk type. Fortunately, during the design phase the team had resisted the urge to "simplify" the data-copying phase by putting user data on a second GCE disk—doing so would have made these instances special snowflakes for the lifetime of the Cloud-hosted platform.

Nontechnical

Organizing the team into virtual workstreams has multiple benefits. This strategy allowed the team to quickly gather expertise across reporting chains, expand and contract teams throughout the project, reduce communication overhead between teams, assign singular deliverable objectives to work groups, and reduce territoriality across teams.

The migration to Cloud allowed the team to reconsider certain implementations that had ossified over time within the team and organization.

APPLYING THIS EXPERIENCE TO OTHERS

Since a cloud desktop is composed of a Compute Engine instance running a custom image (production of which is fairly cheap and well documented [https://cloud.google.com/compute/docs/images#custom_images]), the infrastructure scales extraordinarily well. Very little changed when piloting with a dozen instances versus running with thousands, and what Google has implemented here should be directly applicable to other, smaller companies without requiring much specialization to the plan detailed in this article.

FUTURE PLANS

While the migration of virtual desktops to Cloud wasn't painless, it has been a solid success and a foundation for further work. Looking to the future, the Google Corporate Cloud Migrations team is engaged in two primary streams of work: improving the virtual desktop experience and enabling Google corporate server workloads to run on Cloud.

In the desktop space, the team plans to improve the service management experience by developing various tools that supplement the Google Cloud platform to help manage the fleet of cloud desktops. These add-ons include a disk-inspection tool and a fleet-management command-line tool that integrates and orchestrates actions between Cloud and other corporate systems.

There are several possibilities for improving fleet cost effectiveness. On the simple end of the spectrum, cloud desktop could automatically request that owners of idle machines delete instances they don't actually need.

Finally, the end-user experience could be improved by implementing a self-serve VM cold migration between data centers, allowing traveling users to relocate their instances to a nearby data center to reduce latency to their VMs. Note that these plans are scoped to cloud desktop as part of the customer/application-specific logic, as opposed to features Google as a company is planning for Compute Engine in general.

As for server workloads, the team is building on lessons learned from cloud desktop to provide a migration path. The main technical challenges in this space include:

- Cataloging and characterizing the corporate fleet.
- Creating scalable and auditable service and VM lifecycle

management frameworks.
- Maintaining multiple flavors of managed operating systems.
- Extending BeyondCorp semantics to protocols that are hard to proxy.
- Tackling a new set of security and compliance requirements.
- Creating performant-shared storage solutions for services requiring databases.
- Creating migration tools to automate toilsome operations.
- Implementing a number of service-specific requirements.

Migrating server workloads also has the added organizational complexity of a heterogeneous group of service owners, each with varying priorities and requirements from the departments and business functions they support.

Matt Fata *is a Site Reliability Manager at Google, where he works on corporate virtualization solutions. He has previously worked as a network engineer and as an IT support desk manager.*

Related articles

Titus: Introducing Containers to the Netflix Cloud
Andrew Leung, Andrew Spyker, and Tim Bozarth
Approaching container adoption in an already cloud-native infrastructure
https://queue.acm.org/detail.cfm?id=3158370

Reliable Cron across the Planet
Štepán Davidovic, Kavita Guliani...or how I stopped worrying and learned to love time
https://queue.acm.org/detail.cfm?id=2745840

Virtualization: Blessing or Curse?
Evangelos Kotsovinos
Managing virtualization at a large scale is fraught with hidden challenges.
https://queue.acm.org/detail.cfm?id=1889916

Philippe-Joseph Arida *is a Technical Program Manager at Google, where he is currently working on making GCP the best platform for enterprise workloads. He has previously worked at Microsoft and held internships at Oracle, Matrox, and Tampere University of Technology. Philippe is a graduate of Computer Engineering from McGill University.*

Patrick Hahn *is a Site Reliability Engineer at Google and the Technical Lead of the cloud desktop project. He has previously worked as a sysadmin in the web development, managed IT, and quantitative finance industries.*

Betsy Beyer *is a Technical Writer for Google Site Reliability Engineering in NYC, and the editor of* Site Reliability Engineering: How Google Runs Production Systems *and the forthcoming* Site Reliability Workbook. *She has previously written documentation for Google datacenters and hardware operations teams. She holds degrees from Stanford and Tulane.*

CONTENTS