Google Cloud Well-Architected Framework

Release Notes

Last reviewed 2024-10-11 UTC

To view all of the content in the Well-Architected Framework on a single page or to to get a PDF output of the content, see View on one page.

The Well-Architected Framework provides recommendations to help architects, developers, administrators, and other cloud practitioners design and operate a cloud topology that's secure, efficient, resilient, high-performing, and cost-effective.

A cross-functional team of experts at Google validates the recommendations in the Well-Architected Framework. The team curates the Well-Architected Framework to reflect the expanding capabilities of Google Cloud, industry best practices, community knowledge, and feedback from you. For a summary of the significant changes to the Well-Architected Framework, see What's new.

The Well-Architected Framework is relevant to applications built for the cloud *and* for workloads migrated from on-premises to Google Cloud, hybrid cloud deployments, and multi-cloud environments.

Well-Architected Framework pillars and perspectives

The Well-Architected Framework is organized into five pillars, as shown in the following diagram. We also provide cross-pillar perspectives that focus on recommendations for selected domains, industries, and technologies like AI and machine learning (ML).

Pillars

construction **Operational excellence**

Efficiently deploy, operate, monitor, and manage your cloud workloads.

security **Security, privacy, and compliance**

Maximize the security of your data and workloads in the cloud, design for privacy, and align with regulatory requirements and standards.

restore **Reliability**

Design and operate resilient and highly available workloads in the cloud.

payment **Cost optimization**

Maximize the business value of your investment in Google Cloud.

speed **Performance optimization**

Design and tune your cloud resources for optimal performance.

Perspectives

saved_search **AI and ML**

A cross-pillar view of recommendations that are specific to AI and ML workloads.

saved_search **Financial services industry (FSI)**

A cross-pillar view of recommendations that are specific to FSI workloads.

Core principles

Before you explore the recommendations in each pillar of the Well-Architected Framework, review the following core principles:

Design for change

No system is static. The needs of its users, the goals of the team that builds the system, and the system itself are constantly changing. With the need for change in mind, build a development and production process that enables teams to regularly deliver small changes and get fast feedback on those changes. Consistently demonstrating the ability to deploy changes helps to build trust with stakeholders, including the teams responsible for the system, and the users of the system. Using DORA's software delivery metrics can help your team monitor the speed, ease, and safety of making changes to the system.

Document your architecture

When you start to move your workloads to the cloud or build your applications, lack of documentation about the system can be a major obstacle. Documentation is especially important for correctly visualizing the architecture of your current deployments.

Quality documentation isn't achieved by producing a specific amount of documentation, but by how clear content is, how useful it is, and how it's maintained as the system changes.

A properly documented cloud architecture establishes a common language and standards, which enable cross-functional teams to communicate and collaborate effectively. The documentation also provides the information that's necessary to identify and guide future design decisions. Documentation should be written with your use cases in mind, to provide context for the design decisions.

Over time, your design decisions will evolve and change. The change history provides the context that your teams require to align initiatives, avoid duplication, and measure performance changes effectively over time. Change logs are particularly valuable when you onboard a new cloud architect who is not yet familiar with your current design, strategy, or history.

Analysis by DORA has found a clear link between documentation quality and organizational performance — the organization's ability to meet their performance and profitability goals.

Simplify your design and use fully managed services

Simplicity is crucial for design. If your architecture is too complex to understand, it will be difficult to implement the design and manage it over time. Where feasible, use fully managed services to minimize the risks, time, and effort associated with managing and maintaining baseline systems.

If you're already running your workloads in production, test with managed services to see how they might help to reduce operational complexities. If you're developing new workloads, then start simple, establish a minimal viable product (MVP), and resist the urge to over-engineer. You can identify exceptional use cases, iterate, and improve your systems incrementally over time.

Decouple your architecture

Research from DORA shows that architecture is an important predictor for achieving continuous delivery. Decoupling is a technique that's used to separate your applications and service components into smaller components that can operate independently. For example, you might separate a monolithic application stack into individual service components. In a loosely coupled architecture, an application can run its functions independently, regardless of the various dependencies.

A decoupled architecture gives you increased flexibility to do the following:

- Apply independent upgrades.
- Enforce specific security controls.
- Establish reliability goals for each subsystem.
- Monitor health.
- Granularly control performance and cost parameters.

You can start the decoupling process early in your design phase or incorporate it as part of your system upgrades as you scale.

Use a stateless architecture

A stateless architecture can increase both the reliability and scalability of your applications.

Stateful applications rely on various dependencies to perform tasks, such as local caching of data. Stateful applications often require additional mechanisms to capture progress and restart gracefully. Stateless applications can perform tasks without significant local dependencies by using shared storage or cached services. A stateless architecture enables your applications to scale up quickly with minimum boot dependencies. The applications can withstand hard restarts, have lower downtime, and provide better performance for end users.

What's new in the Well-Architected Framework

Last reviewed 2025-08-07 UTC

This document lists significant changes to the Google Cloud Well-Architected Framework.

August 7, 2025

- AI and ML perspective: Reliability: Major update to expand the reliability recommendations in the AI and ML perspective.

July 28, 2025

- Added the Financial services industry (FSI) perspective. These documents describe principles and recommendations that are specific to FSI, aligned to each pillar of the Architecture Framework.

May 28, 2025

- AI and ML perspective: Cost optimization: Major update to expand the cost optimization recommendations in the AI and ML perspective.

April 28, 2025

- AI and ML perspective: Operational excellence: Major update to expand the operational excellence recommendations in the AI and ML perspective.

February 5, 2025

- Security, privacy, and compliance pillar: Major update to align the recommendations with core principles of security, privacy, and compliance.

December 30, 2024

- Reliability pillar: Major update to align the recommendations with core principles of reliability.

December 6, 2024

- Performance optimization pillar: Major update to align the recommendations with core principles of performance optimization.

October 31, 2024

- Operational excellence pillar: Major update to align the recommendations with core principles of operational excellence.

October 11, 2024

- Added the AI and ML perspective. These documents describe principles and recommendations that are specific to AI and ML, aligned to each pillar of the Architecture Framework.

September 25, 2024

- Cost optimization pillar: Major update to align the recommendations with core principles of cost optimization.

August 27, 2024

- The System Design category of the Architecture Framework has been removed. The guidance in this category is available in other Google Cloud documentation.

June 4, 2024

- Cost optimization category:
  - Added information about monitoring the cost of Dataflow resources.
  - Add recommendations for optimizing cost by using the at-least-once streaming mode and the dynamic thread scaling feature.

May 31, 2024

- Cost optimization category:
  - Added information about the FinOps hub.

May 30, 2024

- System design category:
  - Added a new design principle, Design for change.

May 23, 2024

- Cost optimization category:
  - Added guidance in Optimize cost: Compute, containers, and serverless about reducing cost by suspending idle Compute Engine VMs .

May 16, 2024

- System design category:

    - Added information about the Batch service in Choose and manage compute.

May 9, 2024

- Cost optimization category:
    - Updated guidance about Dataflow Shuffle in Optimize cost: Databases and smart analytics.

April 22, 2024

- Cost optimization category:
    - Added a recommendation about considering storage cost when configuring the log-retention period in Optimize cost: Cloud operations.

March 29, 2024

- Reliability category:
    - Reorganized the guidance related to SLOs and SLIs to improve usability and alignment with SRE guidance.

January 4, 2024

- Performance optimization category:
    - Updated the guidance for Network Service Tiers.

November 28, 2023

- Reliability category:
    - Reorganized the content to improve readability and consistency:
    - Define SLOs: Moved "Terminology" section to new page: Terminology
    - Adopt SLOs: Moved "SLOs and Alerts" section to new page: SLOs and Alerts

November 9, 2023

- System design category:
    - Added guidance to help cloud architects to choose deployment archetypes for workloads in Google Cloud.

Well-Architected Framework: Operational excellence pillar

Release Notes

Last reviewed 2024-10-31 UTC

To view the content in the operational excellence pillar on a single page or to to get a PDF output of the content, see View on one page.

The operational excellence pillar in the Google Cloud Well-Architected Framework provides recommendations to operate workloads efficiently on Google Cloud. Operational excellence in the cloud involves designing, implementing, and managing cloud solutions that provide value, performance, security, and reliability. The recommendations in this pillar help you to continuously improve and adapt workloads to meet the dynamic and ever-evolving needs in the cloud.

The operational excellence pillar is relevant to the following audiences:

- **Managers and leaders**: A framework to establish and maintain operational excellence in the cloud and to ensure that cloud investments deliver value and support business objectives.
- **Cloud operations teams**: Guidance to manage incidents and problems, plan capacity, optimize performance, and manage change.
- **Site reliability engineers (SREs)**: Best practices that help you to achieve high levels of service reliability, including monitoring, incident response, and automation.
- **Cloud architects and engineers**: Operational requirements and best practices for the design and implementation phases, to help ensure that solutions are designed for operational efficiency and scalability.
- **DevOps teams**: Guidance about automation, CI/CD pipelines, and change management, to help enable faster and more reliable software delivery.

To achieve operational excellence, you should embrace automation, orchestration, and data-driven insights. Automation helps to eliminate toil. It also streamlines and builds guardrails around repetitive tasks. Orchestration helps to coordinate complex processes. Data-driven insights enable evidence-based decision-making. By using these practices, you can optimize cloud operations, reduce costs, improve service availability, and enhance security.

Operational excellence in the cloud goes beyond technical proficiency in cloud operations. It includes a cultural shift that encourages continuous learning and experimentation. Teams must be empowered to innovate, iterate, and adopt a growth mindset. A culture of operational excellence fosters a collaborative environment where individuals are encouraged to share ideas, challenge assumptions, and drive improvement.

For operational excellence principles and recommendations that are specific to AI and ML workloads, see AI and ML perspective: Operational excellence in the Well-Architected Framework.

Core principles

The recommendations in the operational excellence pillar of the Well-Architected Framework are mapped to the following core principles:

- Ensure operational readiness and performance using CloudOps: Ensure that cloud solutions meet operational and performance requirements by defining service level objectives (SLOs) and by performing comprehensive monitoring, performance testing, and capacity planning.
- Manage incidents and problems: Minimize the impact of cloud incidents and prevent recurrence through comprehensive observability, clear incident response procedures, thorough retrospectives, and preventive measures.
- Manage and optimize cloud resources: Optimize and manage cloud resources through strategies like right-sizing, autoscaling, and by using effective cost monitoring tools.
- Automate and manage change: Automate processes, streamline change management, and alleviate the burden of manual labor.
- Continuously improve and innovate: Focus on ongoing enhancements and the introduction of new solutions to stay competitive.

Contributors

Authors:

- Ryan Cox | Principal Architect
- Hadrian Knotz | Enterprise Architect

Other contributors:

- Daniel Lees | Cloud Security Architect
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Gary Harmson | Principal Architect
- Jose Andrade | Customer Engineer, SRE Specialist
- Kumar Dhanagopal | Cross-Product Solution Developer
- Nicolas Pintaux | Customer Engineer, Application Modernization Specialist
- Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework

- Samantha He | Technical Writer
- Zach Seils | Networking Specialist
- Wade Holmes | Global Solutions Director

Ensure operational readiness and performance using CloudOps

Release Notes

Last reviewed 2024-10-31 UTC

This principle in the operational excellence pillar of the Google Cloud Well-Architected Framework helps you to ensure operational readiness and performance of your cloud workloads. It emphasizes establishing clear expectations and commitments for service performance, implementing robust monitoring and alerting, conducting performance testing, and proactively planning for capacity needs.

Principle overview

Different organizations might interpret operational readiness differently. Operational readiness is how *your* organization prepares to successfully operate workloads on Google Cloud. Preparing to operate a complex, multilayered cloud workload requires careful planning for both go-live and day-2 operations. These operations are often called *CloudOps*.

Focus areas of operational readiness

Operational readiness consists of four focus areas. Each focus area consists of a set of activities and components that are necessary to prepare to operate a complex application or environment in Google Cloud. The following table lists the components and activities of each focus area:

Note: The recommendations in the operational excellence pillar of the Well-Architected Framework are relevant to one or more of these operational-readiness focus areas.

Focus area of operational readiness	Activities and components
Workforce	• Defining clear roles and responsibilities for the teams that manage and operate the cloud resources. • Ensuring that team members have appropriate skills. • Developing a learning program.

Focus area of operational readiness	Activities and components
	• Establishing a clear team structure. • Hiring the required talent.
Processes	• Observability. • Managing service disruptions. • Cloud delivery. • Core cloud operations.
Tooling	Tools that are required to support CloudOps processes.
Governance	• Service levels and reporting. • Cloud financials. • Cloud operating model. • Architectural review and governance boards. • Cloud architecture and compliance.

Recommendations

To ensure operational readiness and performance by using CloudOps, consider the recommendations in the following sections. Each recommendation in this document is relevant to one or more of the focus areas of operational readiness.

Define SLOs and SLAs

A core responsibility of the cloud operations team is to define service level objectives (SLOs) and service level agreements (SLAs) for all of the critical workloads. This recommendation is relevant to the governance focus area of operational readiness.

SLOs must be specific, measurable, achievable, relevant, and time-bound (SMART), and they must reflect the level of service and performance that you want.

- **Specific**: Clearly articulates the required level of service and performance.
- **Measurable**: Quantifiable and trackable.
- **Achievable**: Attainable within the limits of your organization's capabilities and resources.

- **Relevant**: Aligned with business goals and priorities.
- **Time-bound**: Has a defined timeframe for measurement and evaluation.

For example, an SLO for a web application might be "99.9% availability" or "average response time less than 200 ms." Such SLOs clearly define the required level of service and performance for the web application, and the SLOs can be measured and tracked over time.

SLAs outline the commitments to customers regarding service availability, performance, and support, including any penalties or remedies for noncompliance. SLAs must include specific details about the services that are provided, the level of service that can be expected, the responsibilities of both the service provider and the customer, and any penalties or remedies for noncompliance. SLAs serve as a contractual agreement between the two parties, ensuring that both have a clear understanding of the expectations and obligations that are associated with the cloud service.

Google Cloud provides tools like Cloud Monitoring and service level indicators (SLIs) to help you define and track SLOs. Cloud Monitoring provides comprehensive monitoring and observability capabilities that enable your organization to collect and analyze metrics that are related to the availability, performance, and latency of cloud-based applications and services. SLIs are specific metrics that you can use to measure and track SLOs over time. By utilizing these tools, you can effectively monitor and manage cloud services, and ensure that they meet the SLOs and SLAs.

Clearly defining and communicating SLOs and SLAs for all of your critical cloud services helps to ensure reliability and performance of your deployed applications and services.

Implement comprehensive observability

To get real-time visibility into the health and performance of your cloud environment, we recommend that you use a combination of Google Cloud Observability tools and third-party solutions. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

Implementing a combination of observability solutions provides you with a comprehensive observability strategy that covers various aspects of your cloud infrastructure and applications. Google Cloud Observability is a unified platform for collecting, analyzing, and visualizing metrics, logs, and traces from various Google Cloud services, applications, and external sources. By using Cloud Monitoring, you can gain insights into resource utilization, performance characteristics, and overall health of your resources.

To ensure comprehensive monitoring, monitor important metrics that align with system health indicators such as CPU utilization, memory usage, network traffic, disk I/O, and application response times. You must also consider business-specific metrics. By tracking these metrics, you can identify potential bottlenecks, performance issues, and resource constraints. Additionally, you can set up alerts to notify relevant teams proactively about potential issues or anomalies.

To enhance your monitoring capabilities further, you can integrate third-party solutions with Google Cloud Observability. These solutions can provide additional functionality, such as advanced analytics, machine learning-powered anomaly detection, and incident management capabilities. This combination of Google Cloud Observability tools and third-party solutions lets you create a robust and customizable monitoring ecosystem that's tailored to your specific needs. By using this combination approach, you can proactively identify and address issues, optimize resource utilization, and ensure the overall reliability and availability of your cloud applications and services.

Implement performance and load testing

Performing regular performance testing helps you to ensure that your cloud-based applications and infrastructure can handle peak loads and maintain optimal performance. Load testing simulates realistic traffic patterns. Stress testing pushes the system to its limits to identify potential bottlenecks and performance limitations. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

Tools like Cloud Load Balancing and load testing services can help you to simulate real-world traffic patterns and stress-test your applications. These tools provide valuable insights into how your system behaves under various load conditions, and can help you to identify areas that require optimization.

Based on the results of performance testing, you can make decisions to optimize your cloud infrastructure and applications for optimal performance and scalability. This optimization might involve adjusting resource allocation, tuning configurations, or implementing caching mechanisms.

For example, if you find that your application is experiencing slowdowns during periods of high traffic, you might need to increase the number of virtual machines or containers that are allocated to the application. Alternatively, you might need to adjust the configuration of your web server or database to improve performance.

By regularly conducting performance testing and implementing the necessary optimizations, you can ensure that your cloud-based applications and infrastructure always run at peak performance, and deliver a seamless and responsive experience for your users. Doing so can help you to maintain a competitive advantage and build trust with your customers.

Plan and manage capacity

Proactively planning for future capacity needs—both organic or inorganic—helps you to ensure the smooth operation and scalability of your cloud-based systems. This recommendation is relevant to the processes focus area of operational readiness.

Planning for future capacity includes understanding and managing quotas for various resources like compute instances, storage, and API requests. By analyzing historical usage patterns, growth projections, and business requirements, you can accurately anticipate future capacity requirements. You can use tools like Cloud Monitoring and BigQuery to collect and analyze usage data, identify trends, and forecast future demand.

Historical usage patterns provide valuable insights into resource utilization over time. By examining metrics like CPU utilization, memory usage, and network traffic, you can identify periods of high demand and potential bottlenecks. Additionally, you can help to estimate future capacity needs by making growth projections based on factors like growth in the user base, new products and features, and marketing campaigns. When you assess capacity needs, you should also consider business requirements like SLAs and performance targets.

When you determine the resource sizing for a workload, consider factors that can affect utilization of resources. Seasonal variations like holiday shopping periods or end-of-quarter sales can lead to temporary spikes in demand. Planned events like product launches or marketing campaigns can also significantly increase traffic. To make sure that your primary and disaster recovery (DR) system can handle unexpected surges in demand, plan for capacity that can support graceful failover during disruptions like natural disasters and cyberattacks.

Autoscaling is an important strategy for dynamically adjusting your cloud resources based on workload fluctuations. By using autoscaling policies, you can automatically scale compute instances, storage, and other resources in response to changing demand. This ensures optimal performance during peak periods while minimizing costs when resource utilization is low. Autoscaling algorithms use metrics like CPU utilization, memory usage, and queue depth to determine when to scale resources.

Continuously monitor and optimize

To manage and optimize cloud workloads, you must establish a process for continuously monitoring and analyzing performance metrics. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

To establish a process for continuous monitoring and analysis, you track, collect, and evaluate data that's related to various aspects of your cloud environment. By using this data, you can proactively identify areas for improvement, optimize resource utilization, and ensure that your cloud infrastructure consistently meets or exceeds your performance expectations.

An important aspect of performance monitoring is regularly reviewing logs and traces. Logs provide valuable insights into system events, errors, and warnings. Traces provide detailed information about the flow of requests through your application. By analyzing logs and traces, you can identify potential issues, identify the root causes of problems, and get a better understanding of how your applications behave under different conditions. Metrics like the round-trip time between services can help you to identify and understand bottlenecks that are in your workloads.

Further, you can use performance-tuning techniques to significantly enhance application response times and overall efficiency. The following are examples of techniques that you can use:

- **Caching**: Store frequently accessed data in memory to reduce the need for repeated database queries or API calls.
- **Database optimization**: Use techniques like indexing and query optimization to improve the performance of database operations.
- **Code profiling**: Identify areas of your code that consume excessive resources or cause performance issues.

By applying these techniques, you can optimize your applications and ensure that they run efficiently in the cloud.

Manage incidents and problems

Release Notes

Last reviewed 2024-10-31 UTC

This principle in the operational excellence pillar of the Google Cloud Well-Architected Framework provides recommendations to help you manage incidents and problems related to your cloud workloads. It involves implementing comprehensive monitoring and observability, establishing clear incident response procedures, conducting thorough root cause analysis, and implementing preventive measures. Many of the topics that are discussed in this principle are covered in detail in the Reliability pillar.

Principle overview

Incident management and problem management are important components of a functional operations environment. How you respond to, categorize, and solve incidents of differing severity can significantly affect your operations. You must also proactively and continuously make adjustments to optimize reliability and performance. An efficient process for incident and problem management relies on the following foundational elements:

- **Continuous monitoring**: Identify and resolve issues quickly.
- **Automation**: Streamline tasks and improve efficiency.
- **Orchestration**: Coordinate and manage cloud resources effectively.
- **Data-driven insights**: Optimize cloud operations and make informed decisions.

These elements help you to build a resilient cloud environment that can handle a wide range of challenges and disruptions. These elements can also help to reduce the risk of costly incidents and downtime, and they can help you to achieve greater business agility and success. These foundational elements are spread across the four focus areas of operational readiness: Workforce, Processes, Tooling, and Governance.

Note: The Google SRE Book defines many of the terms and concepts that are described in this document. We recommend the Google SRE Book as supplemental reading to support the recommendations that are described in this document.

Recommendations

To manage incidents and problems effectively, consider the recommendations in the following sections. Each recommendation in this document is relevant to one or more of the focus areas of operational readiness.

Establish clear incident response procedures

Clear roles and responsibilities are essential to ensure effective and coordinated response to incidents. Additionally, clear communication protocols and escalation paths help to ensure that information is shared promptly and effectively during an incident. This recommendation is relevant to these focus areas of operational readiness: workforce, processes, and tooling.

To establish incident response procedures, you need to define the roles and expectations of each team member, such as incident commanders, investigators, communicators, and technical experts. Establishing communication and escalation paths includes identifying important contacts, setting up communication channels, and defining the process for escalating incidents to higher levels of management when necessary. Regular training and preparation helps to ensure that teams are equipped with the knowledge and skills to respond to incidents effectively.

By documenting incident response procedures in a runbook or playbook, you can provide a standardized reference guide for teams to follow during an incident. The runbook must outline the steps to be taken at each stage of the incident response process, including communication, triage, investigation, and resolution. It must also include information about relevant tools and resources and contact information for important personnel. You must regularly review and update the runbook to ensure that it remains current and effective.

Centralize incident management

For effective tracking and management throughout the incident lifecycle, consider using a centralized incident management system. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

A centralized incident management system provides the following advantages:

- **Improved visibility**: By consolidating all incident-related data in a single location, you eliminate the need for teams to search in various channels or systems for context. This approach saves time and reduces confusion, and it gives stakeholders a comprehensive view of the incident, including its status, impact, and progress.
- **Better coordination and collaboration**: A centralized system provides a unified platform for communication and task management. It promotes seamless collaboration between the different departments and functions that are involved in incident response. This approach ensures that everyone has access to up-to-date information and it reduces the risk of miscommunication and misalignment.
- **Enhanced accountability and ownership**: A centralized incident management system enables your organization to allocate tasks to specific individuals or teams and it ensures that responsibilities are clearly defined and tracked. This approach promotes accountability and encourages proactive problem-solving because team members can easily monitor their progress and contributions.

A centralized incident management system must offer robust features for incident tracking, task assignment, and communication management. These features let you customize workflows, set priorities, and integrate with other systems, such as monitoring tools and ticketing systems.

By implementing a centralized incident management system, you can optimize your organization's incident response processes, improve collaboration, and enhance visibility. Doing so leads to faster incident resolution times, reduced downtime, and improved customer satisfaction. It also helps foster a culture of continuous improvement because you can learn from past incidents and identify areas for improvement.

Conduct thorough post-incident reviews

After an incident occurs, you must conduct a detailed post-incident review (PIR), which is also known as a postmortem, to identify the root cause, contributing factors, and lessons learned. This thorough review helps you to prevent similar incidents in the future. This recommendation is relevant to these focus areas of operational readiness: processes and governance.

The PIR process must involve a multidisciplinary team that has expertise in various aspects of the incident. The team must gather all of the relevant information through interviews, documentation review, and site inspections. A timeline of events must be created to establish the sequence of actions that led up to the incident.

After the team gathers the required information, they must conduct a root cause analysis to determine the factors that led to the incident. This analysis must identify both the immediate cause and the systemic issues that contributed to the incident.

Along with identifying the root cause, the PIR team must identify any other contributing factors that might have caused the incident. These factors could include human error, equipment failure, or organizational factors like communication breakdowns and lack of training.

The PIR report must document the findings of the investigation, including the timeline of events, root cause analysis, and recommended actions. The report is a valuable resource for implementing corrective actions and preventing recurrence. The report must be shared with all of the relevant stakeholders and it must be used to develop safety training and procedures.

To ensure a successful PIR process, your organization must foster a blameless culture that focuses on learning and improvement rather than assigning blame. This culture encourages individuals to report incidents without fear of retribution, and it lets you address systemic issues and make meaningful improvements.

By conducting thorough PIRs and implementing corrective measures based on the findings, you can significantly reduce the risk of similar incidents occurring in the future. This proactive approach to incident investigation and prevention helps to create a safer and more efficient work environment for everyone involved.

Maintain a knowledge base

A knowledge base of known issues, solutions, and troubleshooting guides is essential for incident management and resolution. Team members can use the knowledge base to quickly identify and address common problems. Implementing a knowledge base helps to reduce the need for escalation and it improves overall efficiency. This recommendation is relevant to these focus areas of operational readiness: workforce and processes.

A primary benefit of a knowledge base is that it lets teams learn from past experiences and avoid repeating mistakes. By capturing and sharing solutions to known issues, teams can build a collective understanding of how to resolve common problems and best practices for incident management. Use of a knowledge base saves time and effort, and helps to standardize processes and ensure consistency in incident resolution.

Along with helping to improve incident resolution times, a knowledge base promotes knowledge sharing and collaboration across teams. With a central repository of information, teams can easily access and contribute to the knowledge base, which promotes a culture of continuous learning and improvement. This culture encourages teams to share their expertise and experiences, leading to a more comprehensive and valuable knowledge base.

To create and manage a knowledge base effectively, use appropriate tools and technologies. Collaboration platforms like Google Workspace are well-suited for this purpose because they let you easily create, edit, and share documents collaboratively. These tools also support version control and change tracking, which ensures that the knowledge base remains up-to-date and accurate.

Make the knowledge base easily accessible to all relevant teams. You can achieve this by integrating the knowledge base with existing incident management systems or by providing a dedicated portal or intranet site. A knowledge base that's readily available lets teams quickly access the information that they need to resolve incidents efficiently. This availability helps to reduce downtime and minimize the impact on business operations.

Regularly review and update the knowledge base to ensure that it remains relevant and useful. Monitor incident reports, identify common issues and trends, and incorporate new solutions and troubleshooting guides into the knowledge base. An up-to-date knowledge base helps your teams resolve incidents faster and more effectively.

Automate incident response

Automation helps to streamline your incident response and remediation processes. It lets you address security breaches and system failures promptly and efficiently. By using Google Cloud products like Cloud Run functions or Cloud Run, you can automate various tasks that are typically manual and time-consuming. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

Automated incident response provides the following benefits:

- **Reduction in incident detection and resolution times**: Automated tools can continuously monitor systems and applications, detect suspicious or anomalous activities in real time, and notify stakeholders or respond without intervention. This automation lets you identify potential threats or issues before they escalate into major incidents. When an incident is detected, automated tools can trigger predefined remediation actions, such as isolating affected systems, quarantining malicious files, or rolling back changes to restore the system to a known good state.

- **Reduced burden on security and operations teams**: Automated incident response lets the security and operations teams focus on more strategic tasks. By automating routine and repetitive tasks, such as collecting diagnostic information or triggering alerts, your organization can free up personnel to handle more complex and critical incidents. This automation can lead to improved overall incident response effectiveness and efficiency.
- **Enhanced consistency and accuracy of the remediation process**: Automated tools can ensure that remediation actions are applied uniformly across all affected systems, minimizing the risk of human error or inconsistency. This standardization of the remediation process helps to minimize the impact of incidents on users and the business.

Manage and optimize cloud resources

Release Notes

Last reviewed 2024-10-31 UTC

This principle in the operational excellence pillar of the Google Cloud Well-Architected Framework provides recommendations to help you manage and optimize the resources that are used by your cloud workloads. It involves right-sizing resources based on actual usage and demand, using autoscaling for dynamic resource allocation, implementing cost optimization strategies, and regularly reviewing resource utilization and costs. Many of the topics that are discussed in this principle are covered in detail in the Cost optimization pillar.

Principle overview

Cloud resource management and optimization play a vital role in optimizing cloud spending, resource usage, and infrastructure efficiency. It includes various strategies and best practices aimed at maximizing the value and return from your cloud spending.

This pillar's focus on optimization extends beyond cost reduction. It emphasizes the following goals:

- **Efficiency**: Using automation and data analytics to achieve peak performance and cost savings.
- **Performance**: Scaling resources effortlessly to meet fluctuating demands and deliver optimal results.

- **Scalability**: Adapting infrastructure and processes to accommodate rapid growth and diverse workloads.

By focusing on these goals, you achieve a balance between cost and functionality. You can make informed decisions regarding resource provisioning, scaling, and migration. Additionally, you gain valuable insights into resource consumption patterns, which lets you proactively identify and address potential issues before they escalate.

Recommendations

To manage and optimize resources, consider the recommendations in the following sections. Each recommendation in this document is relevant to one or more of the focus areas of operational readiness.

Right-size resources

Continuously monitoring resource utilization and adjusting resource allocation to match actual demand are essential for efficient cloud resource management. Over-provisioning resources can lead to unnecessary costs, and under-provisioning can cause performance bottlenecks that affect application performance and user experience. To achieve an optimal balance, you must adopt a proactive approach to right-sizing cloud resources. This recommendation is relevant to the governance focus area of operational readiness.

Cloud Monitoring and Recommender can help you to identify opportunities for right-sizing. Cloud Monitoring provides real-time visibility into resource utilization metrics. This visibility lets you track resource usage patterns and identify potential inefficiencies. Recommender analyzes resource utilization data to make intelligent recommendations for optimizing resource allocation. By using these tools, you can gain insights into resource usage and make informed decisions about right-sizing the resources.

In addition to Cloud Monitoring and Recommender, consider using custom metrics to trigger automated right-sizing actions. Custom metrics let you track specific resource utilization metrics that are relevant to your applications and workloads. You can also configure alerts to notify administrators when predefined thresholds are met. The administrators can then take necessary actions to adjust resource allocation. This proactive approach ensures that resources are scaled in a timely manner, which helps to optimize cloud costs and prevent performance issues.

Use autoscaling

Autoscaling compute and other resources helps to ensure optimal performance and cost efficiency of your cloud-based applications. Autoscaling lets you dynamically adjust the capacity of your resources based on workload fluctuations, so that you have the resources that you need when you need them and you can avoid over-provisioning and unnecessary costs. This recommendation is relevant to the processes focus area of operational readiness.

To meet the diverse needs of different applications and workloads, Google Cloud offers various autoscaling options, including the following:

- Compute Engine managed instance groups (MIGs) are groups of VMs that are managed and scaled as a single entity. With MIGs, you can define autoscaling policies that specify the minimum and maximum number of VMs to maintain in the group, and the conditions that trigger autoscaling. For example, you can configure a policy to add VMs in a MIG when the CPU utilization reaches a certain threshold and to remove VMs when the utilization drops below a different threshold.
- Google Kubernetes Engine (GKE) autoscaling dynamically adjusts your cluster resources to match your application's needs. It offers the following tools:
    - Cluster Autoscaler adds or removes nodes based on Pod resource demands.
    - Horizontal Pod Autoscaler changes the number of Pod replicas based on CPU, memory, or custom metrics.
    - Vertical Pod Autoscaler fine-tunes Pod resource requests and limits based on usage patterns.
    - Node Auto-Provisioning automatically creates optimized node pools for your workloads.

These tools work together to optimize resource utilization, ensure application performance, and simplify cluster management.

- Cloud Run is a serverless platform that lets you run code without having to manage infrastructure. Cloud Run offers built-in autoscaling, which automatically adjusts the number of instances based on the incoming traffic. When the volume of traffic increases, Cloud Run scales up the number of instances to handle the load. When traffic decreases, Cloud Run scales down the number of instances to reduce costs.

By using these autoscaling options, you can ensure that your cloud-based applications have the resources that they need to handle varying workloads, while avoiding overprovisioning and unnecessary costs. Using autoscaling can lead to improved performance, cost savings, and more efficient use of cloud resources.

Leverage cost optimization strategies

Optimizing cloud spending helps you to effectively manage your organization's IT budgets. This recommendation is relevant to the governance focus area of operational readiness.

Google Cloud offers several tools and techniques to help you optimize cloud costs. By using these tools and techniques, you can get the best value from your cloud spending. These tools and techniques help you to identify areas where costs can be reduced, such as identifying underutilized resources or recommending more cost-effective instance types. Google Cloud options to help optimize cloud costs include the following:

- Committed use discounts (CUDs) are discounts for committing to a certain level of usage over a period of time.
- Sustained use discounts in Compute Engine provide discounts for consistent usage of a service.
- Spot VMs provide access to unused VM capacity at a lower cost compared to regular VMs.

Pricing models might change over time, and new features might be introduced that offer better performance or lower cost compared to existing options. Therefore, you should regularly review pricing models and consider alternative features. By staying informed about the latest pricing models and features, you can make informed decisions about your cloud architecture to minimize costs.

Google Cloud's Cost Management tools, such as budgets and alerts, provide valuable insights into cloud spending. Budgets and alerts let users set budgets and receive alerts when the budgets are exceeded. These tools help users track their cloud spending and identify areas where costs can be reduced.

Track resource usage and costs

You can use tagging and labeling to track resource usage and costs. By assigning tags and labels to your cloud resources like projects, departments, or other relevant dimensions, you can categorize and organize the resources. This lets you monitor and analyze spending patterns for specific resources and identify areas of high usage or potential cost savings. This recommendation is relevant to these focus areas of operational readiness: governance and tooling.

Tools like Cloud Billing and Cost Management help you to get a comprehensive understanding of your spending patterns. These tools provide detailed insights into your cloud usage and they let you identify trends, forecast costs, and make informed decisions. By analyzing historical data and current spending patterns, you can identify the focus areas for your cost-optimization efforts.

Custom dashboards and reports help you to visualize cost data and gain deeper insights into spending trends. By customizing dashboards with relevant metrics and dimensions, you can monitor key performance indicators (KPIs) and track progress towards your cost optimization goals. Reports offer deeper analyses of cost data. Reports let you filter the data by specific time periods or resource types to understand the underlying factors that contribute to your cloud spending.

Regularly review and update your tags, labels, and cost analysis tools to ensure that you have the most up-to-date information on your cloud usage and costs. By staying informed and conducting cost postmortems or proactive cost reviews, you can promptly identify any unexpected increases in spending. Doing so lets you make proactive decisions to optimize cloud resources and control costs.

Establish cost allocation and budgeting

Accountability and transparency in cloud cost management are crucial for optimizing resource utilization and ensuring financial control. This recommendation is relevant to the governance focus area of operational readiness.

To ensure accountability and transparency, you need to have clear mechanisms for cost allocation and chargeback. By allocating costs to specific teams, projects, or individuals, your organization can ensure that each of these entities is responsible for its cloud usage. This practice fosters a sense of ownership and encourages responsible resource management. Additionally, chargeback mechanisms enable your organization to recover cloud costs from internal customers, align incentives with performance, and promote fiscal discipline.

Establishing budgets for different teams or projects is another essential aspect of cloud cost management. Budgets enable your organization to define spending limits and track actual expenses against those limits. This approach lets you make proactive decisions to prevent uncontrolled spending. By setting realistic and achievable budgets, you can ensure that cloud resources are used efficiently and aligned with business objectives. Regular monitoring of actual spending against budgets helps you to identify variances and address potential overruns promptly.

To monitor budgets, you can use tools like Cloud Billing budgets and alerts. These tools provide real-time insights into cloud spending and they notify stakeholders of potential overruns. By using these capabilities, you can track cloud costs and take corrective actions before significant deviations occur. This proactive approach helps to prevent financial surprises and ensures that cloud resources are used responsibly.

Automate and manage change

Release Notes

Last reviewed 2024-10-31 UTC

This principle in the operational excellence pillar of the Google Cloud Well-Architected Framework provides recommendations to help you automate and manage change for your cloud workloads. It involves implementing infrastructure as code (IaC), establishing standard operating procedures, implementing a structured change management process, and using automation and orchestration.

Principle overview

Change management and automation play a crucial role in ensuring smooth and controlled transitions within cloud environments. For effective change management, you need to use strategies and best practices that minimize disruptions and ensure that changes are integrated seamlessly with existing systems.

Effective change management and automation include the following foundational elements:

Change governance: Establish clear policies and procedures for change management, including approval processes and communication plans.

Risk assessment: Identify potential risks associated with changes and mitigate them through risk management techniques.

Testing and validation: Thoroughly test changes to ensure that they meet functional and performance requirements and mitigate potential regressions.

Controlled deployment: Implement changes in a controlled manner, ensuring that users are seamlessly transitioned to the new environment, with mechanisms to seamlessly roll back if needed.

These foundational elements help to minimize the impact of changes and ensure that changes have a positive effect on business operations. These elements are represented by the processes, tooling, and governance focus areas of operational readiness.

Recommendations

To automate and manage change, consider the recommendations in the following sections. Each recommendation in this document is relevant to one or more of the focus areas of operational readiness.

Adopt IaC

Infrastructure as code (IaC) is a transformative approach for managing cloud infrastructure. You can define and manage cloud infrastructure declaratively by using tools like Terraform. IaC helps you achieve consistency, repeatability, and simplified change management. It also enables faster and more reliable deployments. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

The following are the main benefits of adopting the IaC approach for your cloud deployments:

Human-readable resource configurations: With the IaC approach, you can declare your cloud infrastructure resources in a human-readable format, like JSON or YAML. Infrastructure administrators and operators can easily understand and modify the infrastructure and collaborate with others.

Consistency and repeatability: IaC enables consistency and repeatability in your infrastructure deployments. You can ensure that your infrastructure is provisioned and configured the same way every time, regardless of who is performing the deployment. This approach helps to reduce errors and ensures that your infrastructure is always in a known state.

Accountability and simplified troubleshooting: The IaC approach helps to improve accountability and makes it easier to troubleshoot issues. By storing your IaC code in a version control system, you can track changes, and identify when changes were made and by whom. If necessary, you can easily roll back to previous versions.

Implement version control

A version control system like Git is a key component of the IaC process. It provides robust change management and risk mitigation capabilities, which is why it's widely adopted, either through in-house development or SaaS solutions. This recommendation is relevant to these focus areas of operational readiness: governance and tooling.

By tracking changes to IaC code and configurations, version control provides visibility into the evolution of the code, making it easier to understand the impact of changes and identify potential issues. This enhanced visibility fosters collaboration among team members who work on the same IaC project.

Most version control systems let you easily roll back changes if needed. This capability helps to mitigate the risk of unintended consequences or errors. By using tools like Git in your IaC workflow, you can significantly improve change management processes, foster collaboration, and mitigate risks, which leads to a more efficient and reliable IaC implementation.

Build CI/CD pipelines

Continuous integration and continuous delivery (CI/CD) pipelines streamline the process of developing and deploying cloud applications. CI/CD pipelines automate the building, testing, and deployment stages, which enables faster and more frequent releases with improved quality control. This recommendation is relevant to the tooling focus area of operational readiness.

CI/CD pipelines ensure that code changes are continuously integrated into a central repository, typically a version control system like Git. Continuous integration facilitates early detection and resolution of issues, and it reduces the likelihood of bugs or compatibility problems.

To create and manage CI/CD pipelines for cloud applications, you can use tools like Cloud Build and Cloud Deploy.

Cloud Build is a fully managed build service that lets developers define and execute build steps in a declarative manner. It integrates seamlessly with popular source-code management platforms and it can be triggered by events like code pushes and pull requests.

Cloud Deploy is a serverless deployment service that automates the process of deploying applications to various environments, such as testing, staging, and production. It provides features like blue-green deployments, traffic splitting, and rollback capabilities, making it easier to manage and monitor application deployments.

Integrating CI/CD pipelines with version control systems and testing frameworks helps to ensure the quality and reliability of your cloud applications. By running automated tests as part of the CI/CD process, development teams can quickly identify and fix any issues before the code is deployed to the production environment. This integration helps to improve the overall stability and performance of your cloud applications.

Use configuration management tools

Tools like Puppet, Chef, Ansible, and VM Manager help you to automate the configuration and management of cloud resources. Using these tools, you can ensure resource consistency and compliance across your cloud environments. This recommendation is relevant to the tooling focus area of operational readiness.

Automating the configuration and management of cloud resources provides the following benefits:

Significant reduction in the risk of manual errors: When manual processes are involved, there is a higher likelihood of mistakes due to human error. Configuration management tools reduce this risk by automating processes, so that configurations are applied consistently and accurately across all cloud resources. This automation can lead to improved reliability and stability of the cloud environment.

Improvement in operational efficiency: By automating repetitive tasks, your organization can free up IT staff to focus on more strategic initiatives. This automation can lead to increased productivity and cost savings and improved responsiveness to changing business needs.

Simplified management of complex cloud infrastructure: As cloud environments grow in size and complexity, managing the resources can become increasingly difficult. Configuration management tools provide a centralized platform for managing cloud resources. The tools make it easier to track configurations, identify issues, and implement changes. Using these tools can lead to improved visibility, control, and security of your cloud environment.

Automate testing

Integrating automated testing into your CI/CD pipelines helps to ensure the quality and reliability of your cloud applications. By validating changes before deployment, you can significantly reduce the risk of errors and regressions, which leads to a more stable and robust software system. This recommendation is relevant to these focus areas of operational readiness: processes and tooling.

The following are the main benefits of incorporating automated testing into your CI/CD pipelines:

Early detection of bugs and defects: Automated testing helps to detect bugs and defects early in the development process, before they can cause major problems in production. This capability saves time and resources by preventing the need for costly rework and bug fixes at later stages in the development process.

High quality and standards-based code: Automated testing can help improve the overall quality of your code by ensuring that the code meets certain standards and best practices. This capability leads to more maintainable and reliable applications that are less prone to errors.

You can use various types of testing techniques in CI/CD pipelines. Each test type serves a specific purpose.

Unit testing focuses on testing individual units of code, such as functions or methods, to ensure that they work as expected.

Integration testing tests the interactions between different components or modules of your application to verify that they work properly together.

End-to-end testing is often used along with unit and integration testing. End-to-end testing simulates real-world scenarios to test the application as a whole, and helps to ensure that the application meets the requirements of your end users.

To effectively integrate automated testing into your CI/CD pipelines, you must choose appropriate testing tools and frameworks. There are many different options, each with its own strengths and weaknesses. You must also establish a clear testing strategy that outlines the types of tests to be performed, the frequency of testing, and the criteria for passing or failing a test. By following these recommendations, you can ensure that your automated testing process is efficient and effective. Such a process provides valuable insights into the quality and reliability of your cloud applications.

Continuously improve and innovate

Release Notes

Last reviewed 2024-10-31 UTC

This principle in the operational excellence pillar of the Google Cloud Well-Architected Framework provides recommendations to help you continuously optimize cloud operations and drive innovation.

Principle overview

To continuously improve and innovate in the cloud, you need to focus on continuous learning, experimentation, and adaptation. This helps you to explore new technologies and optimize existing processes and it promotes a culture of excellence that enables your organization to achieve and maintain industry leadership.

Through continuous improvement and innovation, you can achieve the following goals:

Accelerate innovation: Explore new technologies and services to enhance capabilities and drive differentiation.

Reduce costs: Identify and eliminate inefficiencies through process-improvement initiatives.

Enhance agility: Adapt rapidly to changing market demands and customer needs.

Improve decision making: Gain valuable insights from data and analytics to make data-driven decisions.

Organizations that embrace the continuous improvement and innovation principle can unlock the full potential of the cloud environment and achieve sustainable growth. This principle maps primarily to the Workforce focus area of operational readiness. A culture of innovation lets teams experiment with new tools and technologies to expand capabilities and reduce costs.

Recommendations

To continuously improve and innovate your cloud workloads, consider the recommendations in the following sections. Each recommendation in this document is relevant to one or more of the focus areas of operational readiness.

Foster a culture of learning

Encourage teams to experiment, share knowledge, and learn continuously. Adopt a blameless culture where failures are viewed as opportunities for growth and improvement. This recommendation is relevant to the workforce focus area of operational readiness.

When you foster a culture of learning, teams can learn from mistakes and iterate quickly. This approach encourages team members to take risks, experiment with new ideas, and expand the boundaries of their work. It also creates a psychologically safe environment where individuals feel comfortable sharing failures and learning from them. Sharing in this way leads to a more open and collaborative environment.

To facilitate knowledge sharing and continuous learning, create opportunities for teams to share knowledge and learn from each other. You can do this through informal and formal learning sessions and conferences.

By fostering a culture of experimentation, knowledge sharing, and continuous learning, you can create an environment where teams are empowered to take risks, innovate, and grow. This environment can lead to increased productivity, improved problem-solving, and a more engaged and motivated workforce. Further, by promoting a blameless culture, you can create a safe space for employees to learn from mistakes and contribute to the collective knowledge of the team. This culture ultimately leads to a more resilient and adaptable workforce that is better equipped to handle challenges and drive success in the long run.

Conduct regular retrospectives

Retrospectives give teams an opportunity to reflect on their experiences, identify what went well, and identify what can be improved. By conducting retrospectives after projects or major incidents, teams can learn from successes and failures, and continuously improve their processes and practices. This recommendation is relevant to these focus areas of operational readiness: processes and governance.

An effective way to structure a retrospective is to use the Start-Stop-Continue model:

Start: In the Start phase of the retrospective, team members identify new practices, processes, and behaviors that they believe can enhance their work. They discuss why the changes are needed and how they can be implemented.

Stop: In the Stop phase, team members identify and eliminate practices, processes, and behaviors that are no longer effective or that hinder progress. They discuss why these changes are necessary and how they can be implemented.

Continue: In the Continue phase, team members identify practices, processes, and behaviors that work well and must be continued. They discuss why these elements are important and how they can be reinforced.

By using a structured format like the Start-Stop-Continue model, teams can ensure that retrospectives are productive and focused. This model helps to facilitate discussion, identify the main takeaways, and identify actionable steps for future enhancements.

Stay up-to-date with cloud technologies

To maximize the potential of Google Cloud services, you must keep up with the latest advancements, features, and best practices. This recommendation is relevant to the workforce focus area of operational readiness.

Participating in relevant conferences, webinars, and training sessions is a valuable way to expand your knowledge. These events provide opportunities to learn from Google Cloud experts, understand new capabilities, and engage with industry peers who might face similar challenges. By attending these sessions, you can gain insights into how to use new features effectively, optimize your cloud operations, and drive innovation within your organization.

To ensure that your team members keep up with cloud technologies, encourage them to obtain certifications and attend training courses. Google Cloud offers a wide range of certifications that validate skills and knowledge in specific cloud domains. Earning these certifications demonstrates commitment to excellence and provides tangible evidence of proficiency in cloud technologies. The training courses that are offered by Google Cloud and our partners delve deeper into specific topics. They provide direct experience and practical skills that can be immediately applied to real-world projects. By investing in the professional development of your team, you can foster a culture of continuous learning and ensure that everyone has the necessary skills to succeed in the cloud.

Actively seek and incorporate feedback

Collect feedback from users, stakeholders, and team members. Use the feedback to identify opportunities to improve your cloud solutions. This recommendation is relevant to the workforce focus area of operational readiness.

The feedback that you collect can help you to understand the evolving needs, issues, and expectations of the users of your solutions. This feedback serves as a valuable input to drive improvements and prioritize future enhancements. You can use various mechanisms to collect feedback:

Surveys are an effective way to gather quantitative data from a large number of users and stakeholders.

User interviews provide an opportunity for in-depth qualitative data collection. Interviews let you understand the specific challenges and experiences of individual users.

Feedback forms that are placed within the cloud solutions offer a convenient way for users to provide immediate feedback on their experience.

Regular meetings with team members can facilitate the collection of feedback on technical aspects and implementation challenges.

The feedback that you collect through these mechanisms must be analyzed and synthesized to identify common themes and patterns. This analysis can help you prioritize future enhancements based on the impact and feasibility of the suggested improvements. By addressing the needs and issues that are identified through feedback, you can ensure that your cloud solutions continue to meet the evolving requirements of your users and stakeholders.

Measure and track progress

Key performance indicators (KPIs) and metrics are crucial for tracking progress and measuring the effectiveness of your cloud operations. KPIs are quantifiable measurements that reflect the overall performance. Metrics are specific data points that contribute to the calculation of KPIs. Review the metrics regularly and use them to identify opportunities for improvement and measure progress. Doing so helps you to continuously improve and optimize your cloud environment. This recommendation is relevant to these focus areas of operational readiness: governance and processes.

A primary benefit of using KPIs and metrics is that they enable your organization to adopt a data-driven approach to cloud operations. By tracking and analyzing operational data, you can make informed decisions about how to improve the cloud environment. This data-driven approach helps you to identify trends, patterns, and anomalies that might not be visible without the use of systematic metrics.

To collect and analyze operational data, you can use tools like Cloud Monitoring and BigQuery. Cloud Monitoring enables real-time monitoring of cloud resources and services. BigQuery lets you store and analyze the data that you gather through monitoring. Using these tools together, you can create custom dashboards to visualize important metrics and trends.

Operational dashboards can provide a centralized view of the most important metrics, which lets you quickly identify any areas that need attention. For example, a dashboard might include metrics like CPU utilization, memory usage, network traffic, and latency for a particular application or service. By monitoring these metrics, you can quickly identify any potential issues and take steps to resolve them.

Well-Architected Framework: Security, privacy, and compliance pillar

Last reviewed 2025-02-14 UTC

This page provides a one-page view of all of the pages in the security, privacy, and compliance pillar of the Well-Architected Framework. You can print this page or save it in PDF format by using your browser's print function.

This page doesn't have a table of contents. You can't use the links on this page to navigate within the page.

The Security, Privacy and Compliance pillar in the Google Cloud Well-Architected Framework provides recommendations to help you design, deploy, and operate cloud workloads that meet your requirements for security, privacy, and compliance.

This document is designed to offer valuable insights and meet the needs of a range of security professionals and engineers. The following table describes the intended audiences for this document:

Audience	What this document provides
Chief information security officers (CISOs), business unit leaders, and IT managers	A general framework to establish and maintain security excellence in the cloud and to ensure a comprehensive view of security areas to make informed decisions about security investments.

Security architects and engineers	Key security practices for the design and operational phases to help ensure that solutions are designed for security, efficiency, and scalability.
DevSecOps teams	Guidance to incorporate overarching security controls to plan automation that enables secure and reliable infrastructure.
Compliance officers and risk managers	Key security recommendations to follow a structured approach to risk management with safeguards that help to meet compliance obligations.

To ensure that your Google Cloud workloads meet your security, privacy, and compliance requirements, all of the stakeholders in your organization must adopt a collaborative approach. In addition, you must recognize that cloud security is a shared responsibility between you and Google. For more information, see Shared responsibilities and shared fate on Google Cloud.

The recommendations in this pillar are grouped into core security principles. Each principle-based recommendation is mapped to one or more of the focus areas of cloud security that might be critical to your organization. Each recommendation highlights guidance about the use and configuration of Google Cloud products and capabilities to help improve your organization's security posture.

Core principles

The recommendations in this pillar are grouped within the following core principles of security. Every principle in this pillar is important. Depending on the requirements of your organization and workload, you might choose to prioritize certain principles.

Implement security by design: Integrate cloud security and network security considerations starting from the initial design phase of your applications and infrastructure. Google Cloud provides architecture blueprints and recommendations to help you apply this principle.

Implement zero trust: Use a never trust, always verify approach, where access to resources is granted based on continuous verification of trust. Google Cloud supports this principle through products like Chrome Enterprise Premium and Identity-Aware Proxy (IAP).

Implement shift-left security: Implement security controls early in the software development lifecycle. Avoid security defects before system changes are made. Detect and fix security bugs early, fast, and reliably after the system changes are committed. Google Cloud supports this principle through products like Cloud Build, Binary Authorization, and Artifact Registry.

Implement preemptive cyber defense: Adopt a proactive approach to security by implementing robust fundamental measures like threat intelligence. This approach helps you build a foundation for more effective threat detection and response. Google Cloud's approach to layered security controls aligns with this principle.

Use AI securely and responsibly: Develop and deploy AI systems in a responsible and secure manner. The recommendations for this principle are aligned with guidance in the AI and ML perspective of the Well-Architected Framework and in Google's Secure AI Framework (SAIF).

Use AI for security: Use AI capabilities to improve your existing security systems and processes through Gemini in Security and overall platform-security capabilities. Use AI as a tool to increase the automation of remedial work and ensure security hygiene to make other systems more secure.

Meet regulatory, compliance, and privacy needs: Adhere to industry-specific regulations, compliance standards, and privacy requirements. Google Cloud helps you meet these obligations through products like Assured Workloads, Organization Policy Service, and our compliance resource center.

Organizational security mindset

A security-focused organizational mindset is crucial for successful cloud adoption and operation. This mindset should be deeply ingrained in your organization's culture and reflected in its practices, which are guided by core security principles as described earlier.

An organizational security mindset emphasizes that you think about security during system design, assume zero trust, and integrate security features throughout your development process. In this mindset, you also think proactively about cyber-defense measures, use AI securely and for security, and consider your regulatory, privacy, and compliance requirements. By embracing these principles, your organization can cultivate a security-first culture that proactively addresses threats, protects valuable assets, and helps to ensure responsible technology usage.

Focus areas of cloud security

This section describes the areas for you to focus on when you plan, implement, and manage security for your applications, systems, and data. The recommendations in each principle of this pillar are relevant to one or more of these focus areas. Throughout the rest of this document, the recommendations specify the corresponding security focus areas to provide further clarity and context.

Focus area Activities and components Related Google Cloud products, capabilities, and solutions

Infrastructure security

Secure network infrastructure.

Encrypt data in transit and at rest.

Control traffic flow.

Secure IaaS and PaaS services.

Protect against unauthorized access.

Firewall Policies

VPC Service Controls

Google Cloud Armor

Cloud Next Generation Firewall

Secure Web Proxy

Identity and access management

Use authentication, authorization, and access controls.

Manage cloud identities.

Manage identity and access management policies.

Cloud Identity

Google's Identity and Access Management (IAM) service

Workforce Identity Federation

Workload Identity Federation

Data security

Store data in Google Cloud securely.

Control access to the data.

Discover and classify the data.

Design necessary controls, such as encryption, access controls, and data loss prevention.

Protect data at rest, in transit, and in use.

Google's IAM service

Sensitive Data Protection

VPC Service Controls

Cloud KMS

Confidential Computing

AI and ML security

Apply security controls at different layers of the AI and ML infrastructure and pipeline.

Ensure model safety.

Google's SAIF

Model Armor

Security operations (SecOps)

Adopt a modern SecOps platform and set of practices, for effective incident management, threat detection, and response processes.

Monitor systems and applications continuously for security events.

Google Security Operations

Application security

Secure applications against software vulnerabilities and attacks.

Artifact Registry

Artifact Analysis

Binary Authorization

Assured Open Source Software

Google Cloud Armor

Web Security Scanner

Cloud governance, risk, and compliance

Establish policies, procedures, and controls to manage cloud resources effectively and securely.

Organization Policy Service

Cloud Asset Inventory

Security Command Center Enterprise

Resource Manager

Logging, auditing, and monitoring

Analyze logs to identify potential threats.

Track and record system activities for compliance and security analysis.

Cloud Logging

Cloud Monitoring

Cloud Audit Logs

VPC Flow Logs

Contributors

Authors:

Wade Holmes | Global Solutions Director

Hector Diaz | Cloud Security Architect

Carlos Leonardo Rosario | Google Cloud Security Specialist

John Bacon | Partner Solutions Architect

Sachin Kalra | Global Security Solution Manager

Other contributors:

Anton Chuvakin | Security Advisor, Office of the CISO

Daniel Lees | Cloud Security Architect

Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist

Gary Harmson | Principal Architect

Gino Pelliccia | Principal Architect

Jose Andrade | Customer Engineer, SRE Specialist

Kumar Dhanagopal | Cross-Product Solution Developer

Laura Hyatt | Customer Engineer, FSI

Marwan Al Shawi | Partner Customer Engineer

Nicolas Pintaux | Customer Engineer, Application Modernization Specialist

Noah McDonald | Cloud Security Consultant

Osvaldo Costa | Networking Specialist Customer Engineer

Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework

Samantha He | Technical Writer

Susan Wu | Outbound Product Manager

Implement security by design

This principle in the security pillar of the Google Cloud Well-Architected Framework provides recommendations to incorporate robust security features, controls, and practices into the design of your cloud applications, services, and platforms. From ideation to operations, security is more effective when it's embedded as an integral part of every stage of your design process.

Principle overview

As explained in An Overview of Google's Commitment to Secure by Design, secure by default and secure by design are often used interchangeably, but they represent distinct approaches to building secure systems. Both approaches aim to minimize vulnerabilities and enhance security, but they differ in scope and implementation:

Secure by default: focuses on ensuring that a system's default settings are set to a secure mode, minimizing the need for users or administrators to take actions to secure the system. This approach aims to provide a baseline level of security for all users.

Secure by design: emphasizes proactively incorporating security considerations throughout a system's development lifecycle. This approach is about anticipating potential threats and vulnerabilities early and making design choices that mitigate risks. This approach involves using secure coding practices, conducting security reviews, and embedding security throughout the design process. The secure-by-design approach is an overarching philosophy that guides the development process and helps to ensure that security isn't an afterthought but is an integral part of a system's design.

Recommendations

To implement the secure by design principle for your cloud workloads, consider the recommendations in the following sections:

Choose system components that help to secure your workloads

Build a layered security approach

Use hardened and attested infrastructure and services

Encrypt data at rest and in transit

Choose system components that help to secure your workloads

This recommendation is relevant to all of the focus areas.

A fundamental decision for effective security is the selection of robust system components—including both hardware and software components—that constitute your platform, solution, or service. To reduce the security attack surface and limit potential damage, you must also carefully consider the deployment patterns of these components and their configurations.

In your application code, we recommend that you use straightforward, safe, and reliable libraries, abstractions, and application frameworks in order to eliminate classes of vulnerabilities. To scan for vulnerabilities in software libraries, you can use third-party tools. You can also use Assured Open Source Software, which helps to reduce risks to your software supply chain by using open source software (OSS) packages that Google uses and secures.

Your infrastructure must use networking, storage, and compute options that support safe operation and align with your security requirements and risk acceptance levels. Infrastructure security is important for both internet-facing and internal workloads.

For information about other Google solutions that support this recommendation, see Implement shift-left security.

Build a layered security approach

This recommendation is relevant to the following focus areas:

AI and ML security

Infrastructure security

Identity and access management

Data security

We recommend that you implement security at each layer of your application and infrastructure stack by applying a defense-in-depth approach.

Use the security features in each component of your platform. To limit access and identify the boundaries of the potential impact (that is, the blast radius) in the event of a security incident, do the following:

Simplify your system's design to accommodate flexibility where possible.

Document the security requirements of each component.

Incorporate a robust secured mechanism to address resiliency and recovery requirements.

When you design the security layers, perform a risk assessment to determine the security features that you need in order to meet internal security requirements and external regulatory requirements. We recommend that you use an industry-standard risk assessment framework that applies to cloud environments and that is relevant to your regulatory requirements. For example, the Cloud Security Alliance (CSA) provides the Cloud Controls Matrix (CCM). Your risk assessment provides you with a catalog of risks and corresponding security controls to mitigate them.

When you perform the risk assessment, remember that you have a shared responsibility arrangement with your cloud provider. Therefore, your risks in a cloud environment differ from your risks in an on-premises environment. For example, in an on-premises environment, you need to mitigate vulnerabilities to your hardware stack. In contrast, in a cloud environment, the cloud provider bears these risks. Also, remember that the boundaries of shared responsibilities differ between IaaS, PaaS, and SaaS services for each cloud provider.

After you identify potential risks, you must design and create a mitigation plan that uses technical, administrative, and operational controls, as well as contractual protections and third-party attestations. In addition, a threat modeling method, such as the OWASP application threat modeling method, helps you to identify potential gaps and suggest actions to address the gaps.

Use hardened and attested infrastructure and services

This recommendation is relevant to all of the focus areas.

A mature security program mitigates new vulnerabilities as described in security bulletins. The security program should also provide remediation to fix vulnerabilities in existing deployments and secure your VM and container images. You can use hardening guides that are specific to the OS and application of your images, as well as benchmarks like the one provided by the Center of Internet Security (CIS).

If you use custom images for your Compute Engine VMs, you need to patch the images yourself. Alternatively, you can use Google-provided curated OS images, which are patched regularly. To run containers on Compute Engine VMs, use Google-curated Container-optimized OS images. Google regularly patches and updates these images.

If you use GKE, we recommend that you enable node auto-upgrades so that Google updates your cluster nodes with the latest patches. Google manages GKE control planes, which are automatically updated and patched. To further reduce the attack surface of your containers, you can use distroless images. Distroless images are ideal for security-sensitive applications, microservices, and situations where minimizing the image size and attack surface is paramount.

For sensitive workloads, use Shielded VM, which prevents malicious code from being loaded during the VM boot cycle. Shielded VM instances provide boot security, monitor integrity, and use the Virtual Trusted Platform Module (vTPM).

To help secure SSH access, OS Login lets your employees connect to your VMs by using Identity and Access Management (IAM) permissions as the source of truth instead of relying on SSH keys. Therefore, you don't need to manage SSH keys throughout your organization. OS Login ties an administrator's access to their employee lifecycle, so when employees change roles or leave your organization, their access is revoked with their account. OS Login also supports Google two-factor authentication, which adds an extra layer of security against account takeover attacks.

In GKE, application instances run within Docker containers. To enable a defined risk profile and to restrict employees from making changes to containers, ensure that your containers are stateless and immutable. The immutability principle means that your employees don't modify the container or access it interactively. If the container must be changed, you build a new image and redeploy that image. Enable SSH access to the underlying containers only in specific debugging scenarios.

To help globally secure configurations across your environment, you can use organization policies to set constraints or guardrails on resources that affect the behavior of your cloud assets. For example, you can define the following organization policies and apply them either globally across a Google Cloud organization or selectively at the level of a folder or project:

Disable external IP address allocation to VMs.

Restrict resource creation to specific geographical locations.

Disable the creation of Service Accounts or their keys.

Encrypt data at rest and in transit

This recommendation is relevant to the following focus areas:

Infrastructure security

Data security

Data encryption is a foundational control to protect sensitive information, and it's a key part of data governance. An effective data protection strategy includes access control, data segmentation and geographical residency, auditing, and encryption implementation that's based on a careful assessment of requirements.

By default, Google Cloud encrypts customer data that's stored at rest, with no action required from you. In addition to default encryption, Google Cloud provides options for envelope encryption and encryption key management. You must identify the solutions that best fit your requirements for key generation, storage, and rotation, whether you're choosing the keys for your storage, for compute, or for big data workloads. For example, Customer-managed encryption keys (CMEKs) can be created in Cloud Key Management Service (Cloud KMS). The CMEKs can be either software-based or HSM-protected to meet your regulatory or compliance requirements, such as the need to rotate encryption keys regularly. Cloud KMS Autokey lets you automate the provisioning and assignment of CMEKs. In addition, you can bring your own keys that are sourced from a third-party key management system by using Cloud External Key Manager (Cloud EKM).

We strongly recommend that data be encrypted in-transit. Google encrypts and authenticates data in transit at one or more network layers when data moves outside physical boundaries that aren't controlled by Google or on behalf of Google. All VM-to-VM traffic within a VPC network and between peered VPC networks is encrypted. You can use MACsec for encryption of traffic over Cloud Interconnect connections. IPsec provides encryption for traffic over Cloud VPN connections. You can protect application-to-application traffic in the cloud by using security features like TLS and mTLS configurations in Apigee and Cloud Service Mesh for containerized applications.

By default, Google Cloud encrypts data at rest and data in transit across the network. However, data isn't encrypted by default while it's in use in memory. If your organization handles confidential data, you need to mitigate any threats that undermine the confidentiality and integrity of either the application or the data in system memory. To mitigate these threats, you can use Confidential Computing, which provides a trusted execution environment for your compute workloads. For more information, see Confidential VM overview.

Implement zero trust

This principle in the security pillar of the Google Cloud Well-Architected Framework helps you ensure comprehensive security across your cloud workloads. The principle of zero trust emphasizes the following practices:

Eliminating implicit trust

Applying the principle of least privilege to access control

Enforcing explicit validation of all access requests

Adopting an assume-breach mindset to enable continuous verification and security posture monitoring

Principle overview

The zero-trust model shifts the security focus from perimeter-based security to an approach where no user or device is considered to be inherently trustworthy. Instead, every access request must be verified, regardless of its origin. This approach involves authenticating and authorizing every user and device, validating their context (location and device posture), and granting least privilege access to only the necessary resources.

Implementing the zero-trust model helps your organization enhance its security posture by minimizing the impact of potential breaches and protecting sensitive data and applications against unauthorized access. The zero-trust model helps you ensure confidentiality, integrity, and availability of data and resources in the cloud.

Recommendations

To implement the zero-trust model for your cloud workloads, consider the recommendations in the following sections:

Secure your network

Verify every access attempt explicitly

Monitor and maintain your network

Secure your network

This recommendation is relevant to the following focus area: Infrastructure security.

Transitioning from conventional perimeter-based security to a zero-trust model requires multiple steps. Your organization might have already integrated certain zero-trust controls into its security posture. However, a zero-trust model isn't a singular product or solution. Instead, it's a holistic integration of multiple security layers and best practices. This section describes recommendations and techniques to implement zero trust for network security.

Access control: Enforce access controls based on user identity and context by using solutions like Chrome Enterprise Premium and Identity-Aware Proxy (IAP). By doing this, you shift security from the network perimeter to individual users and devices. This approach enables granular access control and reduces the attack surface.

Network security: Secure network connections between your on-premises, Google Cloud, and multicloud environments.

Use the private connectivity methods from Cloud Interconnect and IPsec VPNs.

To help secure access to Google Cloud services and APIs, use Private Service Connect.

To help secure outbound access from workloads deployed on Google Kubernetes Engine (GKE) and related products, use Cloud Service Mesh egress gateways.

Network design: Prevent potential security risks by deleting default networks in existing projects and disabling the creation of default networks in new projects.

To avoid conflicts, plan your network and IP address allocation carefully.

To enforce effective access control, limit the number of Virtual Private Cloud (VPC) networks per project.

Segmentation: Isolate workloads but maintain centralized network management.

To segment your network, use Shared VPC.

Define firewall policies and rules at the organization, folder, and VPC network levels.

To prevent data exfiltration, establish secure perimeters around sensitive data and services by using VPC Service Controls.

Perimeter security: Protect against DDoS attacks and web application threats.

To protect against threats, use Google Cloud Armor.

Configure security policies to allow, deny, or redirect traffic at the Google Cloud edge.

Automation: Automate infrastructure provisioning by embracing infrastructure as code (IaC) principles and by using tools like Terraform, Jenkins, and Cloud Build. IaC helps to ensure consistent security configurations, simplified deployments, and rapid rollbacks in case of issues.

Secure foundation: Establish a secure application environment by using the Enterprise foundations blueprint. This blueprint provides prescriptive guidance and automation scripts to help you implement security best practices and configure your Google Cloud resources securely.

Verify every access attempt explicitly

This recommendation is relevant to the following focus areas:

Identity and access management

Security operations (SecOps)

Logging, auditing, and monitoring

Implement strong authentication and authorization mechanisms for any user, device, or service that attempts to access your cloud resources. Don't rely on location or network perimeter as a security control. Don't automatically trust any user, device, or service, even if they are already inside the network. Instead, every attempt to access resources must be rigorously authenticated and authorized. You must implement strong identity verification measures, such as multi-factor authentication (MFA). You must also ensure that access decisions are based on granular policies that consider various contextual factors like user role, device posture, and location.

To implement this recommendation, use the following methods, tools, and technologies:

Unified identity management: Ensure consistent identity management across your organization by using a single identity provider (IdP).

Google Cloud supports federation with most IdPs, including on-premises Active Directory. Federation lets you extend your existing identity management infrastructure to Google Cloud and enable single sign-on (SSO) for users.

If you don't have an existing IdP, consider using Cloud Identity Premium or Google Workspace.

Limited service account permissions: Use service accounts carefully, and adhere to the principle of least privilege.

Grant only the necessary permissions required for each service account to perform its designated tasks.

Use Workload Identity Federation for applications that run on Google Kubernetes Engine (GKE) or run outside Google Cloud to access resources securely.

Robust processes: Update your identity processes to align with cloud security best practices.

To help ensure compliance with regulatory requirements, implement identity governance to track access, risks, and policy violations.

Review and update your existing processes for granting and auditing access-control roles and permissions.

Strong authentication: Implement SSO for user authentication and implement MFA for privileged accounts.

Google Cloud supports various MFA methods, including Titan Security Keys, for enhanced security.

For workload authentication, use OAuth 2.0 or signed JSON Web Tokens (JWTs).

Least privilege: Minimize the risk of unauthorized access and data breaches by enforcing the principles of least privilege and separation of duties.

Avoid overprovisioning user access.

Consider implementing just-in-time privileged access for sensitive operations.

Logging: Enable audit logging for administrator and data access activities.

For analysis and threat detection, scan the logs by using Security Command Center Enterprise or Google Security Operations.

Configure appropriate log retention policies to balance security needs with storage costs.

Monitor and maintain your network

This recommendation is relevant to the following focus areas:

Logging, auditing, and monitoring

Application security

Security operations (SecOps)

Infrastructure security

When you plan and implement security measures, assume that an attacker is already inside your environment. This proactive approach involves using the following multiple tools and techniques to provide visibility into your network:

Centralized logging and monitoring: Collect and analyze security logs from all of your cloud resources through centralized logging and monitoring.

Establish baselines for normal network behavior, detect anomalies, and identify potential threats.

Continuously analyze network traffic flows to identify suspicious patterns and potential attacks.

Insights into network performance and security: Use tools like Network Analyzer. Monitor traffic for unusual protocols, unexpected connections, or sudden spikes in data transfer, which could indicate malicious activity.

Vulnerability scanning and remediation: Regularly scan your network and applications for vulnerabilities.

Use Web Security Scanner, which can automatically identify vulnerabilities in your Compute Engine instances, containers, and GKE clusters.

Prioritize remediation based on the severity of vulnerabilities and their potential impact on your systems.

Intrusion detection: Monitor network traffic for malicious activity and automatically block or get alerts for suspicious events by using Cloud IDS and Cloud NGFW intrusion prevention service.

Security analysis: Consider implementing Google SecOps to correlate security events from various sources, provide real-time analysis of security alerts, and facilitate incident response.

Consistent configurations: Ensure that you have consistent security configurations across your network by using configuration management tools.

Implement shift-left security

This principle in the security pillar of the Google Cloud Well-Architected Framework helps you identify practical controls that you can implement early in the software development lifecycle to improve your security posture. It provides recommendations that help you implement preventive security guardrails and post-deployment security controls.

Principle overview

Shift-left security means adopting security practices early in the software development lifecycle. This principle has the following goals:

Avoid security defects before system changes are made. Implement preventive security guardrails and adopt practices such as infrastructure as code (IaC), policy as code, and security checks in the CI/CD pipeline. You can also use other platform-specific capabilities like Organization Policy Service and hardened GKE clusters in Google Cloud.

Detect and fix security bugs early, fast, and reliably after any system changes are committed. Adopt practices like code reviews, post-deployment vulnerability scanning, and security testing.

The Implement security by design and shift-left security principles are related but they differ in scope. The security-by-design principle helps you to avoid fundamental design flaws that would require re-architecting the entire system. For example, a threat-modeling exercise reveals that the current design doesn't include an authorization policy, and all users would have the same level of access without it. Shift-left security helps you to avoid implementation defects (bugs and misconfigurations) before changes are applied, and it enables fast, reliable fixes after deployment.

Recommendations

To implement the shift-left security principle for your cloud workloads, consider the recommendations in the following sections:

Adopt preventive security controls

Automate provisioning and management of cloud resources

Automate secure application releases

Ensure that application deployments follow approved processes

Scan for known vulnerabilities before application deployment

Monitor your application code for known vulnerabilities

Adopt preventive security controls

This recommendation is relevant to the following focus areas:

Identity and access management

Cloud governance, risk, and compliance

Preventive security controls are crucial for maintaining a strong security posture in the cloud. These controls help you proactively mitigate risks. You can prevent misconfigurations and unauthorized access to resources, enable developers to work efficiently, and help ensure compliance with industry standards and internal policies.

Preventive security controls are more effective when they're implemented by using infrastructure as code (IaC). With IaC, preventive security controls can include more customized checks on the infrastructure code before changes are deployed. When combined with automation, preventive security controls can run as part of your CI/CD pipeline's automatic checks.

The following products and Google Cloud capabilities can help you implement preventive controls in your environment:

Organization Policy Service constraints: configure predefined and custom constraints with centralized control.

VPC Service Controls: create perimeters around your Google Cloud services.

Identity and Access Management (IAM), Privileged Access Manager, and principal access boundary policies: restrict access to resources.

Policy Controller and Open Policy Agent (OPA): enforce IaC constraints in your CI/CD pipeline and avoid cloud misconfigurations.

IAM lets you authorize who can act on specific resources based on permissions. For more information, see Access control for organization resources with IAM.

Organization Policy Service lets you set restrictions on resources to specify how they can be configured. For example, you can use an organization policy to do the following:

Limit resource sharing based on domain.

Limit the use of service accounts.

Restrict the physical location of newly created resources.

In addition to using organizational policies, you can restrict access to resources by using the following methods:

Tags with IAM: assign a tag to a set of resources and then set the access definition for the tag itself, rather than defining the access permissions on each resource.

IAM Conditions: define conditional, attribute-based access control for resources.

Defense in depth: use VPC Service Controls to further restrict access to resources.

For more information about resource management, see Decide a resource hierarchy for your Google Cloud landing zone.

Automate provisioning and management of cloud resources

This recommendation is relevant to the following focus areas:

Application security

Cloud governance, risk, and compliance

Automating the provisioning and management of cloud resources and workloads is more effective when you also adopt declarative IaC, as opposed to imperative scripting. IaC isn't a security tool or practice on its own, but it helps you to improve the security of your platform. Adopting IaC lets you create repeatable infrastructure and provides your operations team with a known good state. IaC also improves the efficiency of rollbacks, audit changes, and troubleshooting.

When combined with CI/CD pipelines and automation, IaC also gives you the ability to adopt practices such as policy as code with tools like OPA. You can audit infrastructure changes over time and run automatic checks on the infrastructure code before changes are deployed.

To automate the infrastructure deployment, you can use tools like Config Controller, Terraform, Jenkins, and Cloud Build. To help you build a secure application environment using IaC and automation, Google Cloud provides the enterprise foundations blueprint. This blueprint is Google's opinionated design that follows all of our recommended practices and configurations. The blueprint provides step-by-step instructions to configure and deploy your Google Cloud topology by using Terraform and Cloud Build.

You can modify the scripts of the enterprise foundations blueprint to configure an environment that follows Google recommendations and meets your own security requirements. You can further build on the blueprint with additional blueprints or design your own automation. The Google Cloud Architecture Center provides other blueprints that can be implemented on top of the enterprise foundations blueprint. The following are a few examples of these blueprints:

Deploy an enterprise developer platform on Google Cloud

Deploy a secured serverless architecture using Cloud Run

Build and deploy generative AI and machine learning models in an enterprise

Import data from Google Cloud into a secured BigQuery data warehouse

Automate secure application releases

This recommendation is relevant to the following focus area: Application security.

Without automated tools, it can be difficult to deploy, update, and patch complex application environments to meet consistent security requirements. We recommend that you build automated CI/CD pipelines for your software development lifecycle (SDLC). Automated CI/CD pipelines help you to remove manual errors, provide standardized development feedback loops, and enable efficient product iterations. Continuous delivery is one of the best practices that the DORA framework recommends.

Automating application releases by using CI/CD pipelines helps to improve your ability to detect and fix security bugs early, fast, and reliably. For example, you can scan for security vulnerabilities automatically when artifacts are created, narrow the scope of security reviews, and roll back to a known and safe version. You can also define policies for different environments (such as development, test, or production environments) so that only verified artifacts are deployed.

To help you automate application releases and embed security checks in your CI/CD pipeline, Google Cloud provides multiple tools including Cloud Build, Cloud Deploy, Web Security Scanner, and Binary Authorization.

To establish a process that verifies multiple security requirements in your SDLC, use the Supply-chain Levels for Software Artifacts (SLSA) framework, which has been defined by Google. SLSA requires security checks for source code, build process, and code provenance. Many of these requirements can be included in an automated CI/CD pipeline. To understand how Google applies these practices internally, see Google Cloud's approach to change.

Ensure that application deployments follow approved processes

This recommendation is relevant to the following focus area: Application security.

If an attacker compromises your CI/CD pipeline, your entire application stack can be affected. To help secure the pipeline, you should enforce an established approval process before you deploy the code into production.

If you use Google Kubernetes Engine (GKE) or Cloud Run, you can establish an approval process by using Binary Authorization. Binary Authorization attaches configurable signatures to container images. These signatures (also called attestations) help to validate the image. At deployment time, Binary Authorization uses these attestations to determine whether a process was completed. For example, you can use Binary Authorization to do the following:

Verify that a specific build system or CI pipeline created a container image.

Validate that a container image is compliant with a vulnerability signing policy.

Verify that a container image passes the criteria for promotion to the next deployment environment, such as from development to QA.

By using Binary Authorization, you can enforce that only trusted code runs on your target platforms.

Scan for known vulnerabilities before application deployment

This recommendation is relevant to the following focus area: Application security.

We recommend that you use automated tools that can continuously perform vulnerability scans on application artifacts before they're deployed to production.

For containerized applications, use Artifact Analysis to automatically run vulnerability scans for container images. Artifact Analysis scans new images when they're uploaded to Artifact Registry. The scan extracts information about the system packages in the container. After the initial scan, Artifact Analysis continuously monitors the metadata of scanned images in Artifact Registry for new vulnerabilities. When Artifact Analysis receives new and updated vulnerability information from vulnerability sources, it does the following:

Updates the metadata of the scanned images to keep them up to date.

Creates new vulnerability occurrences for new notes.

Deletes vulnerability occurrences that are no longer valid.

Monitor your application code for known vulnerabilities

This recommendation is relevant to the following focus area: Application security.

Use automated tools to constantly monitor your application code for known vulnerabilities such as the OWASP Top 10. For more information about Google Cloud products and features that support OWASP Top 10 mitigation techniques, see OWASP Top 10 mitigation options on Google Cloud.

Use Web Security Scanner to help identify security vulnerabilities in your App Engine, Compute Engine, and GKE web applications. The scanner crawls your application, follows all of the links within the scope of your starting URLs, and attempts to exercise as many user inputs and event handlers as possible. It can automatically scan for and detect common vulnerabilities, including cross-site scripting, code injection, mixed content, and outdated or insecure libraries. Web Security Scanner provides early identification of these types of vulnerabilities without distracting you with false positives.

In addition, if you use GKE to manage fleets of Kubernetes clusters, the security posture dashboard shows opinionated, actionable recommendations to help improve your fleet's security posture.

Implement preemptive cyber defense

This principle in the security pillar of the Google Cloud Well-Architected Framework provides recommendations to build robust cyber-defense programs as part of your overall security strategy.

This principle emphasizes the use of threat intelligence to proactively guide your efforts across the core cyber-defense functions, as defined in The Defender's Advantage: A guide to activating cyber defense.

Principle overview

When you defend your system against cyber attacks, you have a significant, underutilized advantage against attackers. As the founder of Mandiant states, "You should know more about your business, your systems, your topology, your infrastructure than any attacker does. This is an incredible advantage." To help you use this inherent advantage, this document provides recommendations about proactive and strategic cyber-defense practices that are mapped to the Defender's Advantage framework.

Recommendations

To implement preemptive cyber defense for your cloud workloads, consider the recommendations in the following sections:

Integrate the functions of cyber defense

Use the Intelligence function in all aspects of cyber defense

Understand and capitalize on your defender's advantage

Validate and improve your defenses continuously

Manage and coordinate cyber-defense efforts

Integrate the functions of cyber defense

This recommendation is relevant to all of the focus areas.

The Defender's Advantage framework identifies six critical functions of cyber defense: Intelligence, Detect, Respond, Validate, Hunt, and Mission Control. Each function focuses on a unique part of the cyber-defense mission, but these functions must be well-coordinated and work together to provide an effective defense. Focus on building a robust and integrated system where each function supports the others. If you need a phased approach for adoption, consider the following suggested order. Depending on your current cloud maturity, resource topology, and specific threat landscape, you might want to prioritize certain functions.

Intelligence: The Intelligence function guides all the other functions. Understanding the threat landscape—including the most likely attackers, their tactics, techniques, and procedures (TTPs), and the potential impact—is critical to prioritizing actions across the entire program. The Intelligence function is responsible for stakeholder identification, definition of intelligence requirements, data collection, analysis and dissemination, automation, and the creation of a cyber threat profile.

Detect and Respond: These functions make up the core of active defense, which involves identifying and addressing malicious activity. These functions are necessary to act on the intelligence that's gathered by the intelligence function. The Detect function requires a methodical approach that aligns detections to attacker TTPs and ensures robust logging. The Respond function must focus on initial triage, data collection, and incident remediation.

Validate: The Validate function is a continuous process that provides assurance that your security control ecosystem is up-to-date and operating as designed. This function ensures that your organization understands the attack surface, knows where vulnerabilities exist, and measures the effectiveness of controls. Security validation is also an important component of the detection engineering lifecycle and must be used to identify detection gaps and create new detections.

Hunt: The Hunt function involves proactively searching for active threats within an environment. This function must be implemented when your organization has a baseline level of maturity in the Detect and Respond functions. The Hunt function expands the detection capabilities and helps to identify gaps and weaknesses in controls. The Hunt function must be based on specific threats. This advanced function benefits from a foundation of robust intelligence, detection, and response capabilities.

Mission Control: The Mission Control function acts as the central hub that connects all of the other functions. This function is responsible for strategy, communication, and decisive action across your cyber-defense program. It ensures that all of the functions are working together and that they're aligned with your organization's business goals. You must focus on establishing a clear understanding of the purpose of the Mission Control function before you use it to connect the other functions.

Use the Intelligence function in all aspects of cyber defense

This recommendation is relevant to all of the focus areas.

This recommendation highlights the Intelligence function as a core part of a strong cyber-defense program. Threat intelligence provides knowledge about threat actors, their TTPs, and indicators of compromise (IOCs). This knowledge should inform and prioritize actions across all cyber-defense functions. An intelligence-driven approach helps you align defenses to meet the threats that are most likely to affect your organization. This approach also helps with efficient allocation and prioritization of resources.

The following Google Cloud products and features help you take advantage of threat intelligence to guide your security operations. Use these features to identify and prioritize potential threats, vulnerabilities, and risks, and then plan and implement appropriate actions.

Google Security Operations (Google SecOps) helps you store and analyze security data centrally. Use Google SecOps to map logs into a common model, enrich the logs, and link the logs to timelines for a comprehensive view of attacks. You can also create detection rules, set up IoC matching, and perform threat-hunting activities. The platform also provides curated detections, which are predefined and managed rules to help identify threats. Google SecOps can also integrate with Mandiant frontline intelligence. Google SecOps uniquely integrates industry-leading AI, along with threat intelligence from Mandiant and Google VirusTotal. This integration is critical for threat evaluation and understanding who is targeting your organization and the potential impact.

Security Command Center Enterprise, which is powered by Google AI, enables security professionals to efficiently assess, investigate, and respond to security issues across multiple cloud environments. The security professionals who can benefit from Security

Command Center include security operations center (SOC) analysts, vulnerability and posture analysts, and compliance managers. Security Command Center Enterprise enriches security data, assesses risk, and prioritizes vulnerabilities. This solution provides teams with the information that they need to address high-risk vulnerabilities and to remediate active threats.

Chrome Enterprise Premium offers threat and data protection, which helps to protect users from exfiltration risks and prevents malware from getting onto enterprise-managed devices. Chrome Enterprise Premium also provides visibility into unsafe or potentially unsafe activity that can happen within the browser.

Network monitoring, through tools like Network Intelligence Center, provides visibility into network performance. Network monitoring can also help you detect unusual traffic patterns or detect data transfer amounts that might indicate an attack or data exfiltration attempt.

Understand and capitalize on your defender's advantage

This recommendation is relevant to all of the focus areas.

As mentioned earlier, you have an advantage over attackers when you have a thorough understanding of your business, systems, topology, and infrastructure. To capitalize on this knowledge advantage, utilize this data about your environments during cyberdefense planning.

Google Cloud provides the following features to help you proactively gain visibility to identify threats, understand risks, and respond in a timely manner to mitigate potential damage:

Chrome Enterprise Premium helps you enhance security for enterprise devices by protecting users from exfiltration risks. It extends Sensitive Data Protection services into the browser, and prevents malware. It also offers features like protection against malware and phishing to help prevent exposure to unsafe content. In addition, it gives you control over the installation of extensions to help prevent unsafe or unvetted extensions. These capabilities help you establish a secure foundation for your operations.

Security Command Center Enterprise provides a continuous risk engine that offers comprehensive and ongoing risk analysis and management. The risk engine feature enriches security data, assesses risk, and prioritizes vulnerabilities to help fix issues quickly. Security Command Center enables your organization to proactively identify weaknesses and implement mitigations.

Google SecOps centralizes security data and provides enriched logs with timelines. This enables defenders to proactively identify active compromises and adapt defenses based on attackers' behavior.

Network monitoring helps identify irregular network activity that might indicate an attack and it provides early indicators that you can use to take action. To help proactively protect your data from theft, continuously monitor for data exfiltration and use the provided tools.

Validate and improve your defenses continuously

This recommendation is relevant to all of the focus areas.

This recommendation emphasizes the importance of targeted testing and continuous validation of controls to understand strengths and weaknesses across the entire attack surface. This includes validating the effectiveness of controls, operations, and staff through methods like the following:

Penetration tests

Red-blue team and purple team exercises

Tabletop exercises

You must also actively search for threats and use the results to improve detection and visibility. Use the following tools to continuously test and validate your defenses against real-world threats:

Security Command Center Enterprise provides a continuous risk engine to evaluate vulnerabilities and prioritize remediation, which enables ongoing evaluation of your overall security posture. By prioritizing issues, Security Command Center Enterprise helps you to ensure that resources are used effectively.

Google SecOps offers threat-hunting and curated detections that let you proactively identify weaknesses in your controls. This capability enables continuous testing and improvement of your ability to detect threats.

Chrome Enterprise Premium provides threat and data protection features that can help you to address new and evolving threats, and continuously update your defenses against exfiltration risks and malware.

Cloud Next Generation Firewall (Cloud NGFW) provides network monitoring and data-exfiltration monitoring. These capabilities can help you to validate the effectiveness of your current security posture and identify potential weaknesses. Data-exfiltration monitoring helps you to validate the strength of your organization's data protection mechanisms and make proactive adjustments where necessary. When you integrate threat findings from Cloud NGFW with Security Command Center and Google SecOps, you can optimize network-based threat detection, optimize threat response, and automate playbooks. For more information about this integration, see Unifying Your Cloud Defenses: Security Command Center & Cloud NGFW Enterprise.

Manage and coordinate cyber-defense efforts

This recommendation is relevant to all of the focus areas.

As described earlier in Integrate the functions of cyber defense, the Mission Control function interconnects the other functions of the cyber-defense program. This function enables coordination and unified management across the program. It also helps you coordinate with other teams that don't work on cybersecurity. The Mission Control function promotes empowerment and accountability, facilitates agility and expertise, and drives responsibility and transparency.

The following products and features can help you implement the Mission Control function:

Security Command Center Enterprise acts as a central hub for coordinating and managing your cyber-defense operations. It brings tools, teams, and data together, along with the built-in Google SecOps response capabilities. Security Command Center provides clear visibility into your organization's security state and enables the identification of security misconfigurations across different resources.

Google SecOps provides a platform for teams to respond to threats by mapping logs and creating timelines. You can also define detection rules and search for threats.

Google Workspace and Chrome Enterprise Premium help you to manage and control end-user access to sensitive resources. You can define granular access controls based on user identity and the context of a request.

Network monitoring provides insights into the performance of network resources. You can import network monitoring insights into Security Command Center and Google SecOps for centralized monitoring and correlation against other timeline based data points. This integration helps you to detect and respond to potential network usage changes caused by nefarious activity.

Data-exfiltration monitoring helps to identify possible data loss incidents. With this feature, you can efficiently mobilize an incident response team, assess damages, and limit further data exfiltration. You can also improve current policies and controls to ensure data protection.

Product summary

The following table lists the products and features that are described in this document and maps them to the associated recommendations and security capabilities.

Google Cloud product	Applicable recommendations
Google SecOps	Use the Intelligence function in all aspects of cyber defense: Enables threat hunting and IoC matching, and integrates with Mandiant for comprehensive threat evaluation. Understand and capitalize on your defender's advantage: Provides curated detections and centralizes security data for proactive compromise identification. Validate and improve your defenses continuously: Enables continuous testing and improvement of threat detection capabilities. Manage and coordinate cyber-defense efforts through Mission Control: Provides a platform for threat response, log analysis, and timeline creation.

Security Command Center Enterprise	Use the Intelligence function in all aspects of cyber defense: Uses AI to assess risk, prioritize vulnerabilities, and provide actionable insights for remediation.

Understand and capitalize on your defender's advantage: Offers comprehensive risk analysis, vulnerability prioritization, and proactive identification of weaknesses.

Validate and improve your defenses continuously: Provides ongoing security posture evaluation and resource prioritization.

Manage and coordinate cyber-defense efforts through Mission Control: Acts as a central hub for managing and coordinating cyber-defense operations.

Chrome Enterprise Premium	Use the Intelligence function in all aspects of cyber defense: Protects users from exfiltration risks, prevents malware, and provides visibility into unsafe browser activity.

Understand and capitalize on your defender's advantage: Enhances security for enterprise devices through data protection, malware prevention, and control over extensions.

Validate and improve your defenses continuously: Addresses new and evolving threats through continuous updates to defenses against exfiltration risks and malware.

Manage and coordinate cyber-defense efforts through Mission Control: Manage and control end-user access to sensitive resources, including granular access controls.

Google Workspace	Manage and coordinate cyber-defense efforts through Mission Control: Manage and control end-user access to sensitive resources, including granular access controls.

Network Intelligence Center	Use the Intelligence function in all aspects of cyber defense: Provides visibility into network performance and detects unusual traffic patterns or data transfers.

Cloud NGFW	Validate and improve your defenses continuously: Optimizes network-based threat detection and response through integration with Security Command Center and Google SecOps.

Use AI securely and responsibly

This principle in the security pillar of the Google Cloud Well-Architected Framework provides recommendations to help you secure your AI systems. These recommendations are aligned with Google's Secure AI Framework (SAIF), which provides a practical approach to address the security and risk concerns of AI systems. SAIF is a conceptual framework that aims to provide industry-wide standards for building and deploying AI responsibly.

Principle overview

To help ensure that your AI systems meet your security, privacy, and compliance requirements, you must adopt a holistic strategy that starts with the initial design and extends to deployment and operations. You can implement this holistic strategy by applying the six core elements of SAIF.

Google uses AI to enhance security measures, such as identifying threats, automating security tasks, and improving detection capabilities, while keeping humans in the loop for critical decisions.

Google emphasizes a collaborative approach to advancing AI security. This approach involves partnering with customers, industries, and governments to enhance the SAIF guidelines and offer practical, actionable resources.

The recommendations to implement this principle are grouped within the following sections:

Recommendations to use AI securely

Recommendations for AI governance

Recommendations to use AI securely

To use AI securely, you need both foundational security controls and AI-specific security controls. This section provides an overview of recommendations to ensure that your AI and ML deployments meet the security, privacy, and compliance requirements of your organization. For an overview of architectural principles and recommendations that are specific to AI and ML workloads in Google Cloud, see the AI and ML perspective in the Well-Architected Framework.

Define clear goals and requirements for AI usage

This recommendation is relevant to the following focus areas:

Cloud governance, risk, and compliance

AI and ML security

This recommendation aligns with the SAIF element about contextualizing AI system risks in the surrounding business processes. When you design and evolve AI systems, it's important to understand your specific business goals, risks, and compliance requirements.

Keep data secure and prevent loss or mishandling

This recommendation is relevant to the following focus areas:

Infrastructure security

Identity and access management

Data security

Application security

AI and ML security

This recommendation aligns with the following SAIF elements:

Expand strong security foundations to the AI ecosystem. This element includes data collection, storage, access control, and protection against data poisoning.

Contextualize AI system risks. Emphasize data security to support business objectives and compliance.

Keep AI pipelines secure and robust against tampering

This recommendation is relevant to the following focus areas:

Infrastructure security

Identity and access management

Data security

Application security

AI and ML security

This recommendation aligns with the following SAIF elements:

Expand strong security foundations to the AI ecosystem. As a key element of establishing a secure AI system, secure your code and model artifacts.

Adapt controls for faster feedback loops. Because it's important for mitigation and incident response, track your assets and pipeline runs.

Deploy apps on secure systems using secure tools and artifacts

This recommendation is relevant to the following focus areas:

Infrastructure security

Identity and access management

Data security

Application security

AI and ML security

Using secure systems and validated tools and artifacts in AI-based applications aligns with the SAIF element about expanding strong security foundations to the AI ecosystem and supply chain. This recommendation can be addressed through the following steps:

Implement a secure environment for ML training and deployment

Use validated container images

Apply Supply-chain Levels for Software Artifacts (SLSA) guidelines

Protect and monitor inputs

This recommendation is relevant to the following focus areas:

Logging, auditing, and monitoring

Security operations

AI and ML security

This recommendation aligns with the SAIF element about extending detection and response to bring AI into an organization's threat universe. To prevent issues, it's critical to manage prompts for generative AI systems, monitor inputs, and control user access.

Recommendations for AI governance

All of the recommendations in this section are relevant to the following focus area: Cloud governance, risk, and compliance.

Google Cloud offers a robust set of tools and services that you can use to build responsible and ethical AI systems. We also offer a framework of policies, procedures, and ethical considerations that can guide the development, deployment, and use of AI systems.

As reflected in our recommendations, Google's approach for AI governance is guided by the following principles:

Fairness

Transparency

Accountability

Privacy

Security

Use fairness indicators

Vertex AI can detect bias during the data collection or post-training evaluation process. Vertex AI provides model evaluation metrics like data bias and model bias to help you evaluate your model for bias.

These metrics are related to fairness across different categories like race, gender, and class. However, interpreting statistical deviations isn't a straightforward exercise, because differences across categories might not be a result of bias or a signal of harm.

Use Vertex Explainable AI

To understand how the AI models make decisions, use Vertex Explainable AI. This feature helps you to identify potential biases that might be hidden in the model's logic.

This explainability feature is integrated with BigQuery ML and Vertex AI, which provide feature-based explanations. You can either perform explainability in BigQuery ML or register your model in Vertex AI and perform explainability in Vertex AI.

Track data lineage

Track the origin and transformation of data that's used in your AI systems. This tracking helps you understand the data's journey and identify potential sources of bias or error.

Data lineage is a Dataplex Universal Catalog feature that lets you track how data moves through your systems: where it comes from, where it's passed to, and what transformations are applied to it.

Establish accountability

Establish clear responsibility for the development, deployment, and outcomes of your AI systems.

Use Cloud Logging to log key events and decisions made by your AI systems. The logs provide an audit trail to help you understand how the system is performing and identify areas for improvement.

Use Error Reporting to systematically analyze errors made by the AI systems. This analysis can reveal patterns that point to underlying biases or areas where the model needs further refinement.

Implement differential privacy

During model training, add noise to the data in order to make it difficult to identify individual data points but still enable the model to learn effectively. With SQL in BigQuery, you can transform the results of a query with differentially private aggregations.

Use AI for security

This principle in the security pillar of the Google Cloud Well-Architected Framework provides recommendations to use AI to help you improve the security of your cloud workloads.

Because of the increasing number and sophistication of cyber attacks, it's important to take advantage of AI's potential to help improve security. AI can help to reduce the number of threats, reduce the manual effort required by security professionals, and help compensate for the scarcity of experts in the cyber-security domain.

Principle overview

Use AI capabilities to improve your existing security systems and processes. You can use Gemini in Security as well as the intrinsic AI capabilities that are built into Google Cloud services.

These AI capabilities can transform security by providing assistance across every stage of the security lifecycle. For example, you can use AI to do the following:

Analyze and explain potentially malicious code without reverse engineering.

Reduce repetitive work for cyber-security practitioners.

Use natural language to generate queries and interact with security event data.

Surface contextual information.

Offer recommendations for quick responses.

Aid in the remediation of events.

Summarize high-priority alerts for misconfigurations and vulnerabilities, highlight potential impacts, and recommend mitigations.

Levels of security autonomy

AI and automation can help you achieve better security outcomes when you're dealing with ever-evolving cyber-security threats. By using AI for security, you can achieve greater levels of autonomy to detect and prevent threats and improve your overall security posture. Google defines four levels of autonomy when you use AI for security, and they outline the increasing role of AI in assisting and eventually leading security tasks:

Manual: Humans run all of the security tasks (prevent, detect, prioritize, and respond) across the entire security lifecycle.

Assisted: AI tools, like Gemini, boost human productivity by summarizing information, generating insights, and making recommendations.

Semi-autonomous: AI takes primary responsibility for many security tasks and delegates to humans only when required.

Autonomous: AI acts as a trusted assistant that drives the security lifecycle based on your organization's goals and preferences, with minimal human intervention.

Recommendations

The following sections describe the recommendations for using AI for security. The sections also indicate how the recommendations align with Google's Secure AI Framework (SAIF) core elements and how they're relevant to the levels of security autonomy.

Enhance threat detection and response with AI

Simplify security for experts and non-experts

Automate time-consuming security tasks with AI

Incorporate AI into risk management and governance processes

Implement secure development practices for AI systems

Note: For more information about Google Cloud's overall vision for using Gemini across our products to accelerate AI for security, see the whitepaper Google Cloud's Product Vision for AI-Powered Security.

Enhance threat detection and response with AI

This recommendation is relevant to the following focus areas:

Security operations (SecOps)

Logging, auditing, and monitoring

AI can analyze large volumes of security data, offer insights into threat actor behavior, and automate the analysis of potentially malicious code. This recommendation is aligned with the following SAIF elements:

Extend detection and response to bring AI into your organization's threat universe.

Automate defenses to keep pace with existing and new threats.

Depending on your implementation, this recommendation can be relevant to the following levels of autonomy:

Assisted: AI helps with threat analysis and detection.

Semi-autonomous: AI takes on more responsibility for the security task.

Google Threat Intelligence, which uses AI to analyze threat actor behavior and malicious code, can help you implement this recommendation.

Simplify security for experts and non-experts

This recommendation is relevant to the following focus areas:

Security operations (SecOps)

Cloud governance, risk, and compliance

AI-powered tools can summarize alerts and recommend mitigations, and these capabilities can make security more accessible to a wider range of personnel. This recommendation is aligned with the following SAIF elements:

Automate defenses to keep pace with existing and new threats.

Harmonize platform-level controls to ensure consistent security across the organization.

Depending on your implementation, this recommendation can be relevant to the following levels of autonomy:

Assisted: AI helps you to improve the accessibility of security information.

Semi-autonomous: AI helps to make security practices more effective for all users.

Gemini in Security Command Center can provide summaries of alerts for misconfigurations and vulnerabilities.

Automate time-consuming security tasks with AI

This recommendation is relevant to the following focus areas:

Infrastructure security

Security operations (SecOps)

Application security

AI can automate tasks such as analyzing malware, generating security rules, and identifying misconfigurations. These capabilities can help to reduce the workload on security teams and accelerate response times. This recommendation is aligned with the SAIF element about automating defenses to keep pace with existing and new threats.

Depending on your implementation, this recommendation can be relevant to the following levels of autonomy:

Assisted: AI helps you to automate tasks.

Semi-autonomous: AI takes primary responsibility for security tasks, and only requests human assistance when needed.

Gemini in Google SecOps can help to automate high-toil tasks by assisting analysts, retrieving relevant context, and making recommendations for next steps.

Incorporate AI into risk management and governance processes

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

You can use AI to build a model inventory and risk profiles. You can also use AI to implement policies for data privacy, cyber risk, and third-party risk. This recommendation is aligned with the SAIF element about contextualizing AI system risks in surrounding business processes.

Depending on your implementation, this recommendation can be relevant to the semi-autonomous level of autonomy. At this level, AI can orchestrate security agents that run processes to achieve your custom security goals.

Implement secure development practices for AI systems

This recommendation is relevant to the following focus areas:

Application security

AI and ML security

You can use AI for secure coding, cleaning training data, and validating tools and artifacts. This recommendation is aligned with the SAIF element about expanding strong security foundations to the AI ecosystem.

This recommendation can be relevant to all levels of security autonomy, because a secure AI system needs to be in place before AI can be used effectively for security. The recommendation is most relevant to the assisted level, where security practices are augmented by AI.

To implement this recommendation, follow the Supply-chain Levels for Software Artifacts (SLSA) guidelines for AI artifacts and use validated container images.

Meet regulatory, compliance, and privacy needs

This principle in the security pillar of the Google Cloud Well-Architected Framework helps you identify and meet regulatory, compliance, and privacy requirements for cloud deployments. These requirements influence many of the decisions that you need to make about the security controls that must be used for your workloads in Google Cloud.

Principle overview

Meeting regulatory, compliance, and privacy needs is an unavoidable challenge for all businesses. Cloud regulatory requirements depend on several factors, including the following:

The laws and regulations that apply to your organization's physical locations

The laws and regulations that apply to your customers' physical locations

Your industry's regulatory requirements

Privacy regulations define how you can obtain, process, store, and manage your users' data. You own your own data, including the data that you receive from your users. Therefore, many privacy controls are your responsibility, including controls for cookies, session management, and obtaining user permission.

The recommendations to implement this principle are grouped within the following sections:

Recommendations to address organizational risks

Recommendations to address regulatory and compliance obligations

Recommendations to manage your data sovereignty

Recommendations to address privacy requirements

Recommendations to address organizational risks

This section provides recommendations to help you identify and address risks to your organization.

Identify risks to your organization

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

Before you create and deploy resources on Google Cloud, complete a risk assessment. This assessment should determine the security features that you need to meet your internal security requirements and external regulatory requirements.

Your risk assessment provides you with a catalog of organization-specific risks, and informs you about your organization's capability to detect and counteract security threats. You must perform a risk analysis immediately after deployment and whenever there are changes in your business needs, regulatory requirements, or threats to your organization.

As mentioned in the Implement security by design principle, your security risks in a cloud environment differ from on-premises risks. This difference is due to the shared responsibility model in the cloud, which varies by service (IaaS, PaaS, or SaaS) and your usage. Use a cloud-specific risk assessment framework like the Cloud Controls Matrix (CCM). Use threat modeling, like OWASP application threat modeling, to identify and address vulnerabilities. For expert help with risk assessments, contact your Google account representative or consult Google Cloud's partner directory.

After you catalog your risks, you must determine how to address them—that is, whether you want to accept, avoid, transfer, or mitigate the risks. For mitigation controls that you can implement, see the next section about mitigating your risks.

Mitigate your risks

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

When you adopt new public cloud services, you can mitigate risks by using technical controls, contractual protections, and third-party verifications or attestations.

Technical controls are features and technologies that you use to protect your environment. These include built-in cloud security controls like firewalls and logging. Technical controls can also include using third-party tools to reinforce or support your security strategy. There are two categories of technical controls:

You can implement Google Cloud's security controls to help you mitigate the risks that apply to your environment. For example, you can secure the connection between your on-premises networks and your cloud networks by using Cloud VPN and Cloud Interconnect.

Google has robust internal controls and auditing to protect against insider access to customer data. Our audit logs provide you with near real-time logs of Google administrator access on Google Cloud.

Contractual protections refer to the legal commitments made by us regarding Google Cloud services. Google is committed to maintaining and expanding our compliance portfolio. The Cloud Data Processing Addendum (CDPA) describes our commitments with regard to the processing and security of your data. The CDPA also outlines the access controls that limit Google support engineers' access to customers' environments, and it describes our rigorous logging and approval process. We recommend that you review Google Cloud's contractual controls with your legal and regulatory experts, and verify that they meet your requirements. If you need more information, contact your technical account representative.

Third-party verifications or attestations refer to having a third-party vendor audit the cloud provider to ensure that the provider meets compliance requirements. For example, to learn about Google Cloud attestations with regard to the ISO/IEC 27017 guidelines, see ISO/IEC 27017 - Compliance. To view the current Google Cloud certifications and letters of attestation, see Compliance resource center.

Recommendations to address regulatory and compliance obligations

A typical compliance journey has three stages: assessment, gap remediation, and continual monitoring. This section provides recommendations that you can use during each of these stages.

Assess your compliance needs

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

Compliance assessment starts with a thorough review of all of your regulatory obligations and how your business is implementing them. To help you with your assessment of Google Cloud services, use the Compliance resource center. This site provides information about the following:

Service support for various regulations

Google Cloud certifications and attestations

To better understand the compliance lifecycle at Google and how your requirements can be met, you can contact sales to request help from a Google compliance specialist. Or, you can contact your Google Cloud account manager to request a compliance workshop.

For more information about tools and resources that you can use to manage security and compliance for Google Cloud workloads, see Assuring Compliance in the Cloud.

Automate implementation of compliance requirements

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

To help you stay in compliance with changing regulations, determine whether you can automate how you implement compliance requirements. You can use both compliance-focused capabilities that Google Cloud provides and blueprints that use recommended configurations for a particular compliance regime.

Assured Workloads builds on the controls within Google Cloud to help you meet your compliance obligations. Assured Workloads lets you do the following:

Select your compliance regime. Then, the tool automatically sets the baseline personnel access controls for the selected regime.

Set the location for your data by using organization policies so that your data at rest and your resources remain only in that region.

Select the key-management option (such as the key rotation period) that best meets your security and compliance requirements.

Select the access criteria for Google support personnel to meet certain regulatory requirements such as FedRAMP Moderate. For example, you can select whether Google support personnel have completed the appropriate background checks.

Use Google-owned and Google-managed encryption keys that are FIPS-140-2 compliant and support FedRAMP Moderate compliance. For an added layer of control and for the separation of duties, you can use customer-managed encryption keys (CMEK). For more information about keys, see Encrypt data at rest and in transit.

In addition to Assured Workloads, you can use Google Cloud blueprints that are relevant to your compliance regime. You can modify these blueprints to incorporate your security policies into your infrastructure deployments.

To help you build an environment that supports your compliance requirements, Google's blueprints and solution guides include recommended configurations and provide Terraform modules. The following table lists blueprints that address security and alignment with compliance requirements.

Requirement	Blueprints and solution guides
FedRAMP	Google Cloud FedRAMP implementation guide
	Setting up a FedRAMP Aligned Three-Tier Workload on Google Cloud
HIPAA	Protecting healthcare data on Google Cloud
	Setting up a HIPAA-aligned workload using Data Protection Toolkit

Monitor your compliance

This recommendation is relevant to the following focus areas:

Cloud governance, risk, and compliance

Logging, monitoring, and auditing

Most regulations require that you monitor particular activities, which include access-related activities. To help with your monitoring, you can use the following:

Access Transparency: View near real-time logs when Google Cloud administrators access your content.

Firewall Rules Logging: Record TCP and UDP connections inside a VPC network for any rules that you create. These logs can be useful for auditing network access or for providing early warning that the network is being used in an unapproved manner.

VPC Flow Logs: Record network traffic flows that are sent or received by VM instances.

Security Command Center Premium: Monitor for compliance with various standards.

OSSEC (or another open source tool): Log the activity of individuals who have administrator access to your environment.

Key Access Justifications: View the reasons for a key-access request.

Security Command Center notifications: Get alerts when noncompliance issues occur. For example, get alerts when users disable two-step verification or when service accounts are over-privileged. You can also set up automatic remediation for specific notifications.

Recommendations to manage your data sovereignty

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

Data sovereignty provides you with a mechanism to prevent Google from accessing your data. You approve access only for provider behaviors that you agree are necessary. For example, you can manage your data sovereignty in the following ways:

Store and manage encryption keys outside the cloud.

Grant access to these keys based on detailed access justifications.

Protect data in use by using Confidential Computing.

Manage your operational sovereignty

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

Operational sovereignty provides you with assurances that Google personnel can't compromise your workloads. For example, you can manage operational sovereignty in the following ways:

Restrict the deployment of new resources to specific provider regions.

Limit Google personnel access based on predefined attributes such as their citizenship or geographic location.

Manage software sovereignty

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

Software sovereignty provides you with assurances that you can control the availability of your workloads and run them wherever you want. Also, you can have this control without being dependent or locked in with a single cloud provider. Software sovereignty includes the ability to survive events that require you to quickly change where your workloads are deployed and what level of outside connection is allowed.

For example, to help you manage your software sovereignty, Google Cloud supports hybrid and multicloud deployments. If you choose on-premises deployments for data sovereignty reasons, Google Distributed Cloud is a combination of hardware and software that brings Google Cloud into your data center.

Recommendations to address privacy requirements

Google Cloud includes the following controls that promote privacy:

Default encryption of all data when it's at rest, when it's in transit, and while it's being processed.

Safeguards against insider access.

Support for numerous privacy regulations.

The following recommendations address additional controls that you can implement. For more information, see Privacy Resource Center.

Control data residency

This recommendation is relevant to the following focus area: Cloud governance, risk, and compliance.

Data residency describes where your data is stored at rest. Data residency requirements vary based on system design objectives, industry regulatory concerns, national law, tax implications, and even culture.

Controlling data residency starts with the following:

Understand your data type and its location.

Determine what risks exist for your data and which laws and regulations apply.

Control where your data is stored or where it goes.

To help you comply with data residency requirements, Google Cloud lets you control where your data is stored, how it's accessed, and how it's processed. You can use resource location policies to restrict where resources are created and to limit where data is replicated between regions. You can use the location property of a resource to identify where the service is deployed and who maintains it. For more information, see Resource locations supported services.

Classify your confidential data

This recommendation is relevant to the following focus area: Data security.

You must define what data is confidential, and then ensure that the confidential data is properly protected. Confidential data can include credit card numbers, addresses, phone numbers, and other personally identifiable information (PII). Using Sensitive Data Protection, you can set up appropriate classifications. You can then tag and tokenize your data before you store it in Google Cloud. Additionally, Dataplex Universal Catalog offers a catalog service that provides a platform for storing, managing, and accessing your metadata. For more information and an example of data classification and de-identification, see De-identification and re-identification of PII using Sensitive Data Protection.

Lock down access to sensitive data

This recommendation is relevant to the following focus areas:

Data security

Identity and access management

Place sensitive data in its own service perimeter by using VPC Service Controls. VPC Service Controls improves your ability to mitigate the risk of unauthorized copying or transferring of data (data exfiltration) from Google-managed services. With VPC Service Controls, you configure security perimeters around the resources of your Google-managed services to control the movement of data across the perimeter. Set Google Identity and Access Management (IAM) access controls for that data. Configure multifactor authentication (MFA) for all users who require access to sensitive data.

Shared responsibilities and shared fate on Google Cloud

This document describes the differences between the shared responsibility model and shared fate in Google Cloud. It discusses the challenges and nuances of the shared responsibility model. This document describes what shared fate is and how we partner with our customers to address cloud security challenges.

Understanding the shared responsibility model is important when determining how to best protect your data and workloads on Google Cloud. The shared responsibility model describes the tasks that you have when it comes to security in the cloud and how these tasks are different for cloud providers.

Understanding shared responsibility, however, can be challenging. The model requires an in-depth understanding of each service you utilize, the configuration options that each service provides, and what Google Cloud does to secure the service. Every service has a different configuration profile, and it can be difficult to determine the best security configuration. Google believes that the shared responsibility model stops short of helping cloud customers achieve better security outcomes. Instead of shared responsibility, we believe in shared fate.

Shared fate includes us building and operating a trusted cloud platform for your workloads. We provide best practice guidance and secured, attested infrastructure code that you can use to deploy your workloads in a secure way. We release solutions that combine various Google Cloud services to solve complex security problems and we offer innovative insurance options to help you measure and mitigate the risks that you must accept. Shared fate involves us more closely interacting with you as you secure your resources on Google Cloud.

Shared responsibility

You're the expert in knowing the security and regulatory requirements for your business, and knowing the requirements for protecting your confidential data and resources. When you run your workloads on Google Cloud, you must identify the security controls that you need to configure in Google Cloud to help protect your confidential data and each workload. To decide which security controls to implement, you must consider the following factors:

Your regulatory compliance obligations

Your organization's security standards and risk management plan

Security requirements of your customers and your vendors

Defined by workloads

Traditionally, responsibilities are defined based on the type of workload that you're running and the cloud services that you require. Cloud services include the following categories:

Cloud service Description

Infrastructure as a service (IaaS) IaaS services include Compute Engine, Cloud Storage, and networking services such as Cloud VPN, Cloud Load Balancing, and Cloud DNS.

IaaS provides compute, storage, and network services on demand with pay-as-you-go pricing. You can use IaaS if you plan on migrating an existing on-premises workload to the cloud using lift-and-shift, or if you want to run your application on particular VMs, using specific databases or network configurations.

In IaaS, the bulk of the security responsibilities are yours, and our responsibilities are focused on the underlying infrastructure and physical security.

Platform as a service (PaaS) PaaS services include App Engine, Google Kubernetes Engine (GKE), and BigQuery.

PaaS provides the runtime environment that you can develop and run your applications in. You can use PaaS if you're building an application (such as a website), and want to focus on development not on the underlying infrastructure.

In PaaS, we're responsible for more controls than in IaaS. Typically, this will vary by the services and features that you use. You share responsibility with us for application-level controls and IAM management. You remain responsible for your data security and client protection.

Software as a service (SaaS) SaaS applications include Google Workspace, Google Security Operations, and third-party SaaS applications that are available in Google Cloud Marketplace.

SaaS provides online applications that you can subscribe to or pay for in some way. You can use SaaS applications when your enterprise doesn't have the internal expertise or business requirement to build the application themselves, but does require the ability to process workloads.

In SaaS, we own the bulk of the security responsibilities. You remain responsible for your access controls and the data that you choose to store in the application.

Function as a service (FaaS) or serverless

FaaS provides the platform for developers to run small, single-purpose code (called functions) that run in response to particular events. You would use FaaS when you want particular things to occur based on a particular event. For example, you might create a function that runs whenever data is uploaded to Cloud Storage so that it can be classified.

FaaS has a similar shared responsibility list as SaaS. Cloud Run functions is a FaaS application.

The following diagram shows the cloud services and defines how responsibilities are shared between the cloud provider and customer.

Shared security responsibilities

As the diagram shows, the cloud provider always remains responsible for the underlying network and infrastructure, and customers always remain responsible for their access policies and data.

Defined by industry and regulatory framework

Various industries have regulatory frameworks that define the security controls that must be in place. When you move your workloads to the cloud, you must understand the following:

Which security controls are your responsibility

Which security controls are available as part of the cloud offering

Which default security controls are inherited

Inherited security controls (such as our default encryption and infrastructure controls) are controls that you can provide as part of your evidence of your security posture to auditors and regulators. For example, the Payment Card Industry Data Security Standard (PCI DSS) defines regulations for payment processors. When you move your business to the cloud, these regulations are shared between you and your CSP. To understand how PCI DSS responsibilities are shared between you and Google Cloud, see Google Cloud: PCI DSS Shared Responsibility Matrix.

As another example, in the United States, the Health Insurance Portability and Accountability Act (HIPAA) has set standards for handling electronic personal health information (PHI). These responsibilities are also shared between the CSP and you. For more information on how Google Cloud meets our responsibilities under HIPAA, see HIPAA - Compliance.

Other industries (for example, finance or manufacturing) also have regulations that define how data can be gathered, processed, and stored. For more information about shared responsibility related to these, and how Google Cloud meets our responsibilities, see Compliance resource center.

Defined by location

Depending on your business scenario, you might need to consider your responsibilities based on the location of your business offices, your customers, and your data. Different countries and regions have created regulations that inform how you can process and store your customer's data. For example, if your business has customers who reside in the European Union, your business might need to abide by the requirements that are described in the General Data Protection Regulation (GDPR), and you might be obligated to keep your customer data in the EU itself. In this circumstance, you are responsible for ensuring that the data that you collect remains in the Google Cloud regions in the EU. For more information about how we meet our GDPR obligations, see GDPR and Google Cloud.

For information about the requirements related to your region, see Compliance offerings. If your scenario is particularly complicated, we recommend speaking with our sales team or one of our partners to help you evaluate your security responsibilities.

Challenges for shared responsibility

Though shared responsibility helps define the security roles that you or the cloud provider has, relying on shared responsibility can still create challenges. Consider the following scenarios:

Most cloud security breaches are the direct result of misconfiguration (listed as number 3 in the Cloud Security Alliance's Pandemic 11 Report) and this trend is expected to increase. Cloud products are constantly changing, and new ones are constantly being launched. Keeping up with constant change can seem overwhelming. Customers need cloud providers to provide them with opinionated best practices to help keep up with the change, starting with best practices by default and having a baseline secure configuration.

Though dividing items by cloud services is helpful, many enterprises have workloads that require multiple cloud services types. In this circumstance, you must consider how various security controls for these services interact, including whether they overlap between and across services. For example, you might have an on-premises application that you're migrating to Compute Engine, use Google Workspace for corporate email, and also run BigQuery to analyze data to improve your products.

Your business and markets are constantly changing; as regulations change, as you enter new markets, or as you acquire other companies. Your new markets might have different requirements, and your new acquisition might host their workloads on another cloud. To manage the constant changes, you must constantly re-assess your risk profile and be able to implement new controls quickly.

How and where to manage your data encryption keys is an important decision that ties with your responsibilities to protect your data. The option that you choose depends on your regulatory requirements, whether you're running a hybrid cloud environment or still have an on-premises environment, and the sensitivity of the data that you're processing and storing.

Incident management is an important, and often overlooked, area where your responsibilities and the cloud provider responsibilities aren't easily defined. Many incidents require close collaboration and support from the cloud provider to help investigate and mitigate them. Other incidents can result from poorly configured cloud resources or stolen credentials, and ensuring that you meet the best practices for securing your resources and accounts can be quite challenging.

Advanced persistent threats (APTs) and new vulnerabilities can impact your workloads in ways that you might not consider when you start your cloud transformation. Ensuring that you remain up-to-date on the changing landscape, and who is responsible for threat mitigation is difficult, particularly if your business doesn't have a large security team.

Shared fate

We developed shared fate in Google Cloud to start addressing the challenges that the shared responsibility model doesn't address. Shared fate focuses on how all parties can better interact to continuously improve security. Shared fate builds on the shared responsibility model because it views the relationship between cloud provider and customer as an ongoing partnership to improve security.

Shared fate is about us taking responsibility for making Google Cloud more secure. Shared fate includes helping you get started with a secured landing zone and being clear, opinionated, and transparent about recommended security controls, settings, and associated best practices. It includes helping you better quantify and manage your risk with cyber-insurance, using our Risk Protection Program. Using shared fate, we want to evolve from the standard shared responsibility framework to a better model that helps you secure your business and build trust in Google Cloud.

The following sections describe various components of shared fate.

Help getting started

A key component of shared fate is the resources that we provide to help you get started, in a secure configuration in Google Cloud. Starting with a secure configuration helps reduce the issue of misconfigurations which is the root cause of most security breaches.

Our resources include the following:

Enterprise foundations blueprint that discuss top security concerns and our top recommendations.

Secure blueprints that let you deploy and maintain secure solutions using infrastructure as code (IaC). Blueprints have our security recommendations enabled by default. Many blueprints are created by Google security teams and managed as products. This support means that they're updated regularly, go through a rigorous testing process, and receive attestations from third-party testing groups. Blueprints include the enterprise foundations blueprint and the secured data warehouse blueprint.

Google Cloud Well-Architected Framework best practices that address the top recommendations for building security into your designs. The Well-Architected Framework includes a security section and a community zone that you can use to connect with experts and peers.

Landing zone navigation guides that step you through the top decisions that you need to make to build a secure foundation for your workloads, including resource hierarchy, identity onboarding, security and key management, and network structure.

Risk Protection Program

Shared fate also includes the Risk Protection Program (currently in preview), which helps you use the power of Google Cloud as a platform to manage risk, rather than just seeing cloud workloads as another source of risk that you need to manage. The Risk Protection Program is a collaboration between Google Cloud and two leading cyber insurance companies, Munich Re and Allianz Global & Corporate Speciality.

The Risk Protection Program includes Cyber Insurance Hub, which provides data-driven insights that you can use to better understand your cloud security posture. If you're looking for cyber insurance coverage, you can share these insights from Cyber Insurance Hub directly with our insurance partners to obtain a quote. For more information, see Google Cloud Risk Protection Program now in Preview.

Help with deployment and governance

Shared fate also helps with your continued governance of your environment. For example, we focus efforts on products such as the following:

Assured Workloads, which helps you meet your compliance obligations.

Security Command Center Premium, which uses threat intelligence, threat detection, web scanning, and other advanced methods to monitor and detect threats. It also provides a way to resolve many of these threats quickly and automatically.

Organization policies and resource settings that let you configure policies throughout your hierarchy of folders and projects.

Policy Intelligence tools that provide you with insights on access to accounts and resources.

Confidential Computing, which allows you to encrypt data in use.

Sovereign Controls by Partners, which is available in certain countries and helps enforce data residency requirements.

Putting shared responsibility and shared fate into practice

As part of your planning process, consider the following actions to help you understand and implement appropriate security controls:

Create a list of the type of workloads that you will host in Google Cloud, and whether they require IaaS, PaaS, and SaaS services. You can use the shared responsibility diagram as a checklist to ensure that you know the security controls that you need to consider.

Create a list of regulatory requirements that you must comply with, and access resources in the Compliance resource center that relate to those requirements.

Review the list of available blueprints and architectures in the Architecture Center for the security controls that you require for your particular workloads. The blueprints provide a list of recommended controls and the IaC code that you require to deploy that architecture.

Use the landing zone documentation and the recommendations in the enterprise foundations guide to design a resource hierarchy and network architecture that meets your requirements. You can use the opinionated workload blueprints, like the secured data warehouse, to accelerate your development process.

After you deploy your workloads, verify that you're meeting your security responsibilities using services such as the Cyber Insurance Hub, Assured Workloads, Policy Intelligence tools, and Security Command Center Premium.

For more information, see the CISO's Guide to Cloud Transformation paper.

What's next

Review the core security principles.

Keep up to date with shared fate resources.

Familiarize yourself with available blueprints, including the security foundations blueprint and workload examples like the secured data warehouse.

Read more about shared fate.

Read about our underlying secure infrastructure in the Google infrastructure security design overview.

Read how to implement NIST Cybersecurity Framework best practices in Google Cloud
Well-Architected Framework: Reliability pillar

Last reviewed 2025-02-14 UTC

This page provides a one-page view of all of the pages in the reliability pillar of the Well-Architected Framework. You can print this page or save it in PDF format by using your browser's print function.

This page doesn't have a table of contents. You can't use the links on this page to navigate within the page.

The reliability pillar in the Google Cloud Well-Architected Framework provides principles and recommendations to help you design, deploy, and manage reliable workloads in Google Cloud.

This document is intended for cloud architects, developers, platform engineers, administrators, and site reliability engineers.

Reliability is a system's ability to consistently perform its intended functions within the defined conditions and maintain uninterrupted service. Best practices for reliability include redundancy, fault-tolerant design, monitoring, and automated recovery processes.

As a part of reliability, resilience is the system's ability to withstand and recover from failures or unexpected disruptions, while maintaining performance. Google Cloud features, like multi-regional deployments, automated backups, and disaster recovery solutions, can help you improve your system's resilience.

Reliability is important to your cloud strategy for many reasons, including the following:

Minimal downtime: Downtime can lead to lost revenue, decreased productivity, and damage to reputation. Resilient architectures can help ensure that systems can continue to function during failures or recover efficiently from failures.

Enhanced user experience: Users expect seamless interactions with technology. Resilient systems can help maintain consistent performance and availability, and they provide reliable service even during high demand or unexpected issues.

Data integrity: Failures can cause data loss or data corruption. Resilient systems implement mechanisms such as backups, redundancy, and replication to protect data and ensure that it remains accurate and accessible.

Business continuity: Your business relies on technology for critical operations. Resilient architectures can help ensure continuity after a catastrophic failure, which enables business functions to continue without significant interruptions and supports a swift recovery.

Compliance: Many industries have regulatory requirements for system availability and data protection. Resilient architectures can help you to meet these standards by ensuring systems remain operational and secure.

Lower long-term costs: Resilient architectures require upfront investment, but resiliency can help to reduce costs over time by preventing expensive downtime, avoiding reactive fixes, and enabling more efficient resource use.

Organizational mindset

To make your systems reliable, you need a plan and an established strategy. This strategy must include education and the authority to prioritize reliability alongside other initiatives.

Set a clear expectation that the entire organization is responsible for reliability, including development, product management, operations, platform engineering, and site reliability engineering (SRE). Even the business-focused groups, like marketing and sales, can influence reliability.

Every team must understand the reliability targets and risks of their applications. The teams must be accountable to these requirements. Conflicts between reliability and regular product feature development must be prioritized and escalated accordingly.

Plan and manage reliability holistically, across all your functions and teams. Consider setting up a Cloud Centre of Excellence (CCoE) that includes a reliability pillar. For more information, see Optimize your organization's cloud journey with a Cloud Center of Excellence.

Focus areas for reliability

The activities that you perform to design, deploy, and manage a reliable system can be categorized in the following focus areas. Each of the reliability principles and recommendations in this pillar is relevant to one of these focus areas.

Scoping: To understand your system, conduct a detailed analysis of its architecture. You need to understand the components, how they work and interact, how data and actions flow through the system, and what could go wrong. Identify potential failures, bottlenecks, and risks, which helps you to take actions to mitigate those issues.

Observation: To help prevent system failures, implement comprehensive and continuous observation and monitoring. Through this observation, you can understand trends and identify potential problems proactively.

Response: To reduce the impact of failures, respond appropriately and recover efficiently. Automated responses can also help reduce the impact of failures. Even with planning and controls, failures can still occur.

Learning: To help prevent failures from recurring, learn from each experience, and take appropriate actions.

Core principles

The recommendations in the reliability pillar of the Well-Architected Framework are mapped to the following core principles:

Define reliability based on user-experience goals

Set realistic targets for reliability

Build highly available systems through resource redundancy

Take advantage of horizontal scalability

Detect potential failures by using observability

Design for graceful degradation

Perform testing for recovery from failures

Perform testing for recovery from data loss

Conduct thorough postmortems

Note: To learn about the building blocks of infrastructure reliability in Google Cloud, see Google Cloud infrastructure reliability guide.

Contributors

Authors:

Laura Hyatt | Customer Engineer, FSI

Jose Andrade | Customer Engineer, SRE Specialist

Gino Pelliccia | Principal Architect

Other contributors:

Andrés-Leonardo Martínez-Ortiz | Technical Program Manager

Brian Kudzia | Enterprise Infrastructure Customer Engineer

Daniel Lees | Cloud Security Architect

Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist

Gary Harmson | Principal Architect

Kumar Dhanagopal | Cross-Product Solution Developer

Marwan Al Shawi | Partner Customer Engineer

Nicolas Pintaux | Customer Engineer, Application Modernization Specialist

Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework

Ryan Cox | Principal Architect

Samantha He | Technical Writer

Wade Holmes | Global Solutions Director

Zach Seils | Networking Specialist

Define reliability based on user-experience goals

This principle in the reliability pillar of the Google Cloud Well-Architected Framework helps you to assess your users' experience, and then map the findings to reliability goals and metrics.

This principle is relevant to the scoping focus area of reliability.

Principle overview

Observability tools provide large amounts of data, but not all of the data directly relates to the impacts on the users. For example, you might observe high CPU usage, slow server operations, or even crashed tasks. However, if these issues don't affect the user experience, then they don't constitute an outage.

To measure the user experience, you need to distinguish between internal system behavior and user-facing problems. Focus on metrics like the success ratio of user requests. Don't rely solely on server-centric metrics, like CPU usage, which can lead to misleading conclusions about your service's reliability. True reliability means that users can consistently and effectively use your application or service.

Recommendations

To help you measure user experience effectively, consider the recommendations in the following sections.

Measure user experience

To truly understand your service's reliability, prioritize metrics that reflect your users' actual experience. For example, measure the users' query success ratio, application latency, and error rates.

Ideally, collect this data directly from the user's device or browser. If this direct data collection isn't feasible, shift your measurement point progressively further away from the user in the system. For example, you can use the load balancer or frontend service as the measurement point. This approach helps you identify and address issues before those issues can significantly impact your users.

Analyze user journeys

To understand how users interact with your system, you can use tracing tools like Cloud Trace. By following a user's journey through your application, you can find bottlenecks and latency issues that might degrade the user's experience. Cloud Trace captures detailed performance data for each hop in your service architecture. This data helps you identify and address performance issues more efficiently, which can lead to a more reliable and satisfying user experience.

Set realistic targets for reliability

This principle in the reliability pillar of the Google Cloud Well-Architected Framework helps you define reliability goals that are technically feasible for your workloads in Google Cloud.

This principle is relevant to the scoping focus area of reliability.

Principle overview

Design your systems to be just reliable enough for user happiness. It might seem counterintuitive, but a goal of 100% reliability is often not the most effective strategy. Higher reliability might result in a significantly higher cost, both in terms of financial investment and potential limitations on innovation. If users are already happy with the current level of service, then efforts to further increase happiness might yield a low return on investment. Instead, you can better spend resources elsewhere.

You need to determine the level of reliability at which your users are happy, and determine the point where the cost of incremental improvements begin to outweigh the benefits. When you determine this level of sufficient reliability, you can allocate resources strategically and focus on features and improvements that deliver greater value to your users.

Recommendations

To set realistic reliability targets, consider the recommendations in the following subsections.

Accept some failure and prioritize components

Aim for high availability such as 99.99% uptime, but don't set a target of 100% uptime. Acknowledge that some failures are inevitable.

The gap between 100% uptime and a 99.99% target is the allowance for failure. This gap is often called the error budget. The error budget can help you take risks and innovate, which is fundamental to any business to stay competitive.

Prioritize the reliability of the most critical components in the system. Accept that less critical components can have a higher tolerance for failure.

Balance reliability and cost

To determine the optimal reliability level for your system, conduct thorough cost-benefit analyses.

Consider factors like system requirements, the consequences of failures, and your organization's risk tolerance for the specific application. Remember to consider your disaster recovery metrics, such as the recovery time objective (RTO) and recovery point objective (RPO). Decide what level of reliability is acceptable within the budget and other constraints.

Look for ways to improve efficiency and reduce costs without compromising essential reliability features.

Build highly available systems through resource redundancy

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to plan, build, and manage resource redundancy, which can help you to avoid failures.

This principle is relevant to the scoping focus area of reliability.

Principle overview

After you decide the level of reliability that you need, you must design your systems to avoid any single points of failure. Every critical component in the system must be replicated across multiple machines, zones, and regions. For example, a critical database can't be located in only one region, and a metadata server can't be deployed in only one single zone or region. In those examples, if the sole zone or region has an outage, the system has a global outage.

Recommendations

To build redundant systems, consider the recommendations in the following subsections.

Identify failure domains and replicate services

Map out your system's failure domains, from individual VMs to regions, and design for redundancy across the failure domains.

To ensure high availability, distribute and replicate your services and applications across multiple zones and regions. Configure the system for automatic failover to make sure that the services and applications continue to be available in the event of zone or region outages.

For examples of multi-zone and multi-region architectures, see Design reliable infrastructure for your workloads in Google Cloud.

Detect and address issues promptly

Continuously track the status of your failure domains to detect and address issues promptly.

You can monitor the current status of Google Cloud services in all regions by using the Google Cloud Service Health dashboard. You can also view incidents relevant to your project by using Personalized Service Health. You can use load balancers to detect resource health and automatically route traffic to healthy backends. For more information, see Health checks overview.

Test failover scenarios

Like a fire drill, regularly simulate failures to validate the effectiveness of your replication and failover strategies.

For more information, see Simulate a zone outage for a regional MIG and Simulate a zone failure in GKE regional clusters.

Take advantage of horizontal scalability

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to help you use horizontal scalability. By using horizontal scalability, you can help ensure that your workloads in Google Cloud can scale efficiently and maintain performance.

This principle is relevant to the scoping focus area of reliability.

Principle overview

Re-architect your system to a horizontal architecture. To accommodate growth in traffic or data, you can add more resources. You can also remove resources when they're not in use.

To understand the value of horizontal scaling, consider the limitations of vertical scaling.

A common scenario for vertical scaling is to use a MySQL database as the primary database with critical data. As database usage increases, more RAM and CPU is required. Eventually, the database reaches the memory limit on the host machine, and needs to be upgraded. This process might need to be repeated several times. The problem is that there are hard limits on how much a database can grow. VM sizes are not unlimited. The database can reach a point when it's no longer possible to add more resources.

Even if resources were unlimited, a large VM can become a single point of failure. Any problem with the primary database VM can cause error responses or cause a system-wide outage that affects all users. Avoid single points of failure, as described in Build highly available systems through resource redundancy.

Besides these scaling limits, vertical scaling tends to be more expensive. The cost can increase exponentially as machines with greater amounts of compute power and memory are acquired.

Horizontal scaling, by contrast, can cost less. The potential for horizontal scaling is virtually unlimited in a system that's designed to scale.

Recommendations

To transition from a single VM architecture to a horizontal multiple-machine architecture, you need to plan carefully and use the right tools. To help you achieve horizontal scaling, consider the recommendations in the following subsections.

Use managed services

Managed services remove the need to manually manage horizontal scaling. For example, with Compute Engine managed instance groups (MIGs), you can add or remove VMs to scale your application horizontally. For containerized applications, Cloud Run is a serverless platform that can automatically scale your stateless containers based on incoming traffic.

Promote modular design

Modular components and clear interfaces help you scale individual components as needed, instead of scaling the entire application. For more information, see Promote modular design in the performance optimization pillar.

Implement a stateless design

Design applications to be stateless, meaning no locally stored data. This lets you add or remove instances without worrying about data consistency.

Detect potential failures by using observability

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to help you proactively identify areas where errors and failures might occur.

This principle is relevant to the observation focus area of reliability.

Principle overview

To maintain and improve the reliability of your workloads in Google Cloud, you need to implement effective observability by using metrics, logs, and traces.

Metrics are numerical measurements of activities that you want to track for your application at specific time intervals. For example, you might want to track technical metrics like request rate and error rate, which can be used as service-level indicators (SLIs). You might also need to track application-specific business metrics like orders placed and payments received.

Logs are time-stamped records of discrete events that occur within an application or system. The event could be a failure, an error, or a change in state. Logs might include metrics, and you can also use logs for SLIs.

A trace represents the journey of a single user or transaction through a number of separate applications or the components of an application. For example, these components could be microservices. Traces help you to track what components were used in the journeys, where bottlenecks exist, and how long the journeys took.

Metrics, logs, and traces help you monitor your system continuously. Comprehensive monitoring helps you find out where and why errors occurred. You can also detect potential failures before errors occur.

Recommendations

To detect potential failures efficiently, consider the recommendations in the following subsections.

Gain comprehensive insights

To track key metrics like response times and error rates, use Cloud Monitoring and Cloud Logging. These tools also help you to ensure that the metrics consistently meet the needs of your workload.

To make data-driven decisions, analyze default service metrics to understand component dependencies and their impact on overall workload performance.

To customize your monitoring strategy, create and publish your own metrics by using the Google Cloud SDK.

Perform proactive troubleshooting

Implement robust error handling and enable logging across all of the components of your workloads in Google Cloud. Activate logs like Cloud Storage access logs and VPC Flow Logs.

When you configure logging, consider the associated costs. To control logging costs, you can configure exclusion filters on the log sinks to exclude certain logs from being stored.

Optimize resource utilization

Monitor CPU consumption, network I/O metrics, and disk I/O metrics to detect under-provisioned and over-provisioned resources in services like GKE, Compute Engine, and Dataproc. For a complete list of supported services, see Cloud Monitoring overview.

Prioritize alerts

For alerts, focus on critical metrics, set appropriate thresholds to minimize alert fatigue, and ensure timely responses to significant issues. This targeted approach lets you proactively maintain workload reliability. For more information, see Alerting overview.

Design for graceful degradation

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to help you to design your Google Cloud workloads to fail gracefully.

This principle is relevant to the response focus area of reliability.

Principle overview

Graceful degradation is a design approach where a system that experiences a high load continues to function, possibly with reduced performance or accuracy. Graceful degradation ensures continued availability of the system and prevents complete failure, even if the system's work isn't optimal. When the load returns to a manageable level, the system resumes full functionality.

For example, during periods of high load, Google Search prioritizes results from higher-ranked web pages, potentially sacrificing some accuracy. When the load decreases, Google Search recomputes the search results.

Recommendations

To design your systems for graceful degradation, consider the recommendations in the following subsections.

Implement throttling

Ensure that your replicas can independently handle overloads and can throttle incoming requests during high-traffic scenarios. This approach helps you to prevent cascading failures that are caused by shifts in excess traffic between zones.

Use tools like Apigee to control the rate of API requests during high-traffic times. You can configure policy rules to reflect how you want to scale back requests.

Drop excess requests early

Configure your systems to drop excess requests at the frontend layer to protect backend components. Dropping some requests prevents global failures and enables the system to recover more gracefully.With this approach, some users might experience errors. However, you can minimize the impact of outages, in contrast to an approach like circuit-breaking, where all traffic is dropped during an overload.

Handle partial errors and retries

Build your applications to handle partial errors and retries seamlessly. This design helps to ensure that as much traffic as possible is served during high-load scenarios.

Test overload scenarios

To validate that the throttle and request-drop mechanisms work effectively, regularly simulate overload conditions in your system. Testing helps ensure that your system is prepared for real-world traffic surges.

Monitor traffic spikes

Use analytics and monitoring tools to predict and respond to traffic surges before they escalate into overloads. Early detection and response can help maintain service availability during high-demand periods.

Perform testing for recovery from failures

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to help you design and run tests for recovery in the event of failures.

This principle is relevant to the learning focus area of reliability.

Principle overview

To be sure that your system can recover from failures, you must periodically run tests that include regional failovers, release rollbacks, and data restoration from backups.

This testing helps you to practice responses to events that pose major risks to reliability, such as the outage of an entire region. This testing also helps you verify that your system behaves as intended during a disruption.

In the unlikely event of an entire region going down, you need to fail over all traffic to another region. During normal operation of your workload, when data is modified, it needs to be synchronized from the primary region to the failover region. You need to verify that the replicated data is always very recent, so that users don't experience data loss or session breakage. The load balancing system must also be able to shift traffic to the failover region at any time without service interruptions. To minimize downtime after a regional outage, operations engineers also need to be able to manually and efficiently shift user traffic away from a region, in as less time as possible. This operation is sometimes called draining a region, which means you stop the inbound traffic to the region and move all the traffic elsewhere.

Recommendations

When you design and run tests for failure recovery, consider the recommendations in the following subsections.

Define the testing objectives and scope

Clearly define what you want to achieve from the testing. For example, your objectives can include the following:

Validate the recovery time objective (RTO) and the recovery point objective (RPO). For details, see Basics of DR planning.

Assess system resilience and fault tolerance under various failure scenarios.

Test the effectiveness of automated failover mechanisms.

Decide which components, services, or regions are in the testing scope. The scope can include specific application tiers like the frontend, backend, and database, or it can include specific Google Cloud resources like Cloud SQL instances or GKE clusters. The scope must also specify any external dependencies, such as third-party APIs or cloud interconnections.

Prepare the environment for testing

Choose an appropriate environment, preferably a staging or sandbox environment that replicates your production setup. If you conduct the test in production, ensure that you have safety measures ready, like automated monitoring and manual rollback procedures.

Create a backup plan. Take snapshots or backups of critical databases and services to prevent data loss during the test. Ensure that your team is prepared to do manual interventions if the automated failover mechanisms fail.

To prevent test disruptions, ensure that your IAM roles, policies, and failover configurations are correctly set up. Verify that the necessary permissions are in place for the test tools and scripts.

Inform stakeholders, including operations, DevOps, and application owners, about the test schedule, scope, and potential impact. Provide stakeholders with an estimated timeline and the expected behaviors during the test.

Simulate failure scenarios

Plan and execute failures by using tools like Chaos Monkey. You can use custom scripts to simulate failures of critical services such as a shutdown of a primary node in a multi-zone GKE cluster or a disabled Cloud SQL instance. You can also use scripts to simulate a region-wide network outage by using firewall rules or API restrictions based on your scope of test. Gradually escalate the failure scenarios to observe system behavior under various conditions.

Introduce load testing alongside failure scenarios to replicate real-world usage during outages. Test cascading failure impacts, such as how frontend systems behave when backend services are unavailable.

To validate configuration changes and to assess the system's resilience against human errors, test scenarios that involve misconfigurations. For example, run tests with incorrect DNS failover settings or incorrect IAM permissions.

Monitor system behavior

Monitor how load balancers, health checks, and other mechanisms reroute traffic. Use Google Cloud tools like Cloud Monitoring and Cloud Logging to capture metrics and events during the test.

Observe changes in latency, error rates, and throughput during and after the failure simulation, and monitor the overall performance impact. Identify any degradation or inconsistencies in the user experience.

Ensure that logs are generated and alerts are triggered for key events, such as service outages or failovers. Use this data to verify the effectiveness of your alerting and incident response systems.

Verify recovery against your RTO and RPO

Measure how long it takes for the system to resume normal operations after a failure, and then compare this data with the defined RTO and document any gaps.

Ensure that data integrity and availability align with the RPO. To test database consistency, compare snapshots or backups of the database before and after a failure.

Evaluate service restoration and confirm that all services are restored to a functional state with minimal user disruption.

Document and analyze results

Document each test step, failure scenario, and corresponding system behavior. Include timestamps, logs, and metrics for detailed analyses.

Highlight bottlenecks, single points of failure, or unexpected behaviors observed during the test. To help prioritize fixes, categorize issues by severity and impact.

Suggest improvements to the system architecture, failover mechanisms, or monitoring setups. Based on test findings, update any relevant failover policies and playbooks. Present a postmortem report to stakeholders. The report should summarize the outcomes, lessons learned, and next steps. For more information, see Conduct thorough postmortems.

Iterate and improve

To validate ongoing reliability and resilience, plan periodic testing (for example, quarterly).

Run tests under different scenarios, including infrastructure changes, software updates, and increased traffic loads.

Automate failover tests by using CI/CD pipelines to integrate reliability testing into your development lifecycle.

During the postmortem, use feedback from stakeholders and end users to improve the test process and system resilience.

Perform testing for recovery from data loss

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to help you design and run tests for recovery from data loss.

This principle is relevant to the learning focus area of reliability.

Principle overview

To ensure that your system can recover from situations where data is lost or corrupted, you need to run tests for those scenarios. Instances of data loss might be caused by a software bug or some type of natural disaster. After such events, you need to restore data from backups and bring all of the services back up again by using the freshly restored data.

We recommend that you use three criteria to judge the success or failure of this type of recovery test: data integrity, recovery time objective (RTO), and recovery point objective (RPO). For details about the RTO and RPO metrics, see Basics of DR planning.

The goal of data restoration testing is to periodically verify that your organization can continue to meet business continuity requirements. Besides measuring RTO and RPO, a data restoration test must include testing of the entire application stack and all the critical infrastructure services with the restored data. This is necessary to confirm that the entire deployed application works correctly in the test environment.

Recommendations

When you design and run tests for recovering from data loss, consider the recommendations in the following subsections.

Verify backup consistency and test restoration processes

You need to verify that your backups contain consistent and usable snapshots of data that you can restore to immediately bring applications back into service. To validate data integrity, set up automated consistency checks to run after each backup.

To test backups, restore them in a non-production environment. To ensure your backups can be restored efficiently and that the restored data meets application requirements, regularly simulate data recovery scenarios. Document the steps for data restoration, and train your teams to execute the steps effectively during a failure.

Schedule regular and frequent backups

To minimize data loss during restoration and to meet RPO targets, it's essential to have regularly scheduled backups. Establish a backup frequency that aligns with your RPO. For example, if your RPO is 15 minutes, schedule backups to run at least every 15 minutes. Optimize the backup intervals to reduce the risk of data loss.

Use Google Cloud tools like Cloud Storage, Cloud SQL automated backups, or Spanner backups to schedule and manage backups. For critical applications, use near-continuous backup solutions like point-in-time recovery (PITR) for Cloud SQL or incremental backups for large datasets.

Define and monitor RPO

Set a clear RPO based on your business needs, and monitor adherence to the RPO. If backup intervals exceed the defined RPO, use Cloud Monitoring to set up alerts.

Monitor backup health

Use Google Cloud Backup and DR service or similar tools to track the health of your backups and confirm that they are stored in secure and reliable locations. Ensure that the backups are replicated across multiple regions for added resilience.

Plan for scenarios beyond backup

Combine backups with disaster recovery strategies like active-active failover setups or cross-region replication for improved recovery time in extreme cases. For more information, see Disaster recovery planning guide.

Conduct thorough postmortems

This principle in the reliability pillar of the Google Cloud Well-Architected Framework provides recommendations to help you conduct effective postmortems after failures and incidents.

This principle is relevant to the learning focus area of reliability.

Principle overview

A postmortem is a written record of an incident, its impact, the actions taken to mitigate or resolve the incident, the root causes, and the follow-up actions to prevent the incident from recurring. The goal of a postmortem is to learn from mistakes and not assign blame.

The following diagram shows the workflow of a postmortem:

The workflow of a postmortem.

The workflow of a postmortem includes the following steps:

Create postmortem

Capture the facts

Identify and analyze the root causes

Plan for the future

Execute the plan

Conduct postmortem analyses after major events and non-major events like the following:

User-visible downtimes or degradations beyond a certain threshold.

Data losses of any kind.

Interventions from on-call engineers, such as a release rollback or rerouting of traffic.

Resolution times above a defined threshold.

Monitoring failures, which usually imply manual incident discovery.

Recommendations

Define postmortem criteria before an incident occurs so that everyone knows when a post mortem is necessary.

To conduct effective postmortems, consider the recommendations in the following subsections.

Conduct blameless postmortems

Effective postmortems focus on processes, tools, and technologies, and don't place blame on individuals or teams. The purpose of a postmortem analysis is to improve your technology and future, not to find who is guilty. Everyone makes mistakes. The goal should be to analyze the mistakes and learn from them.

The following examples show the difference between feedback that assigns blame and blameless feedback:

Feedback that assigns blame: "We need to rewrite the entire complicated backend system! It's been breaking weekly for the last three quarters and I'm sure we're all tired of fixing things piecemeal. Seriously, if I get paged one more time I'll rewrite it myself..."

Blameless feedback: "An action item to rewrite the entire backend system might actually prevent these pages from continuing to happen. The maintenance manual for this version is quite long and really difficult to be fully trained up on. I'm sure our future on-call engineers will thank us!"

Make the postmortem report readable by all the intended audiences

For each piece of information that you plan to include in the report, assess whether that information is important and necessary to help the audience understand what happened. You can move supplementary data and explanations to an appendix of the report. Reviewers who need more information can request it.

Avoid complex or over-engineered solutions

Before you start to explore solutions for a problem, evaluate the importance of the problem and the likelihood of a recurrence. Adding complexity to the system to solve problems that are unlikely to occur again can lead to increased instability.

Share the postmortem as widely as possible

To ensure that issues don't remain unresolved, publish the outcome of the postmortem to a wide audience and get support from management. The value of a postmortem is proportional to the learning that occurs after the postmortem. When more people learn from incidents, the likelihood of similar failures recurring is reduced. (PDF).

Well-Architected Framework: Cost optimization pillar

Last reviewed 2025-02-14 UTC

This page provides a one-page view of all of the pages in the cost optimization pillar of the Well-Architected Framework. You can print this page or save it in PDF format by using your browser's print function.

This page doesn't have a table of contents. You can't use the links on this page to navigate within the page.

The cost optimization pillar in the Google Cloud Well-Architected Framework describes principles and recommendations to optimize the cost of your workloads in Google Cloud.

The intended audience includes the following:

- CTOs, CIOs, CFOs, and other executives who are responsible for strategic cost management.
- Architects, developers, administrators, and operators who make decisions that affect cost at all the stages of an organization's cloud journey.

The cost models for on-premises and cloud workloads differ significantly. On-premises IT costs include capital expenditure (CapEx) and operational expenditure (OpEx). On-premises hardware and software assets are acquired and the acquisition costs are depreciated over the operating life of the assets. In the cloud, the costs for most cloud resources are treated as OpEx, where costs are incurred when the cloud resources are consumed. This fundamental difference underscores the importance of the following core principles of cost optimization.

Note: You might be able to classify the cost of some Google Cloud services (like Compute Engine sole-tenant nodes) as capital expenditure. For more information, see Sole-tenancy accounting FAQ.

For cost optimization principles and recommendations that are specific to AI and ML workloads, see AI and ML perspective: Cost optimization in the Well-Architected Framework.

Core principles

The recommendations in the cost optimization pillar of the Well-Architected Framework are mapped to the following core principles:

- **Align cloud spending with business value**: Ensure that your cloud resources deliver measurable business value by aligning IT spending with business objectives.
- **Foster a culture of cost awareness**: Ensure that people across your organization consider the cost impact of their decisions and activities, and ensure that they have access to the cost information required to make informed decisions.
- **Optimize resource usage**: Provision only the resources that you need, and pay only for the resources that you consume.
- **Optimize continuously**: Continuously monitor your cloud resource usage and costs, and proactively make adjustments as needed to optimize your spending. This approach involves identifying and addressing potential cost inefficiencies before they become significant problems.

These principles are closely aligned with the core tenets of cloud FinOps. FinOps is relevant to any organization, regardless of its size or maturity in the cloud. By adopting these principles and following the related recommendations, you can control and optimize costs throughout your journey in the cloud.

Contributors

Author: Nicolas Pintaux | Customer Engineer, Application Modernization Specialist

Other contributors:

- Anuradha Bajpai | Solutions Architect
- Daniel Lees | Cloud Security Architect

- Eric Lam | Head of Google Cloud FinOps
- Fernando Rubbo | Cloud Solutions Architect
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Gary Harmson | Principal Architect
- Jose Andrade | Customer Engineer, SRE Specialist
- Kent Hua | Solutions Manager
- Kumar Dhanagopal | Cross-Product Solution Developer
- Marwan Al Shawi | Partner Customer Engineer
- Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework
- Samantha He | Technical Writer
- Steve McGhee | Reliability Advocate
- Sergei Lilichenko | Solutions Architect
- Wade Holmes | Global Solutions Director
- Zach Seils | Networking Specialist

Align cloud spending with business value

This principle in the cost optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to align your use of Google Cloud resources with your organization's business goals.

Principle overview

To effectively manage cloud costs, you need to maximize the business value that the cloud resources provide and minimize the total cost of ownership (TCO). When you evaluate the resource options for your cloud workloads, consider not only the cost of provisioning and using the resources, but also the cost of managing them. For example, virtual machines (VMs) on Compute Engine might be a cost-effective option for hosting applications. However, when you consider the overhead to maintain, patch, and scale the VMs, the TCO can increase. On the other hand, serverless services like Cloud Run can offer greater business value. The lower operational overhead lets your team focus on core activities and helps to increase agility.

To ensure that your cloud resources deliver optimal value, evaluate the following factors:

- **Provisioning and usage costs**: The expenses incurred when you purchase, provision, or consume resources.

- **Management costs**: The recurring expenses for operating and maintaining resources, including tasks like patching, monitoring and scaling.
- **Indirect costs**: The costs that you might incur to manage issues like downtime, data loss, or security breaches.
- **Business impact**: The potential benefits from the resources, like increased revenue, improved customer satisfaction, and faster time to market.

By aligning cloud spending with business value, you get the following benefits:

- **Value-driven decisions**: Your teams are encouraged to prioritize solutions that deliver the greatest business value and to consider both short-term and long-term cost implications.
- **Informed resource choice**: Your teams have the information and knowledge that they need to assess the business value and TCO of various deployment options, so they choose resources that are cost-effective.
- **Cross-team alignment**: Cross-functional collaboration between business, finance, and technical teams ensures that cloud decisions are aligned with the overall objectives of the organization.

Recommendations

To align cloud spending with business objectives, consider the following recommendations.

Prioritize managed services and serverless products

Whenever possible, choose managed services and serverless products to reduce operational overhead and maintenance costs. This choice lets your teams concentrate on their core business activities. They can accelerate the delivery of new features and functionalities, and help drive innovation and value.

The following are examples of how you can implement this recommendation:

- To run PostgreSQL, MySQL, or Microsoft SQL Server server databases, use Cloud SQL instead of deploying those databases on VMs.
- To run and manage Kubernetes clusters, use Google Kubernetes Engine (GKE) Autopilot instead of deploying containers on VMs.
- For your Apache Hadoop or Apache Spark processing needs, use Dataproc and Dataproc Serverless. Per-second billing can help to achieve significantly lower TCO when compared to on-premises data lakes.

Balance cost efficiency with business agility

Controlling costs and optimizing resource utilization are important goals. However, you must balance these goals with the need for flexible infrastructure that lets you innovate rapidly, respond quickly to changes, and deliver value faster. The following are examples of how you can achieve this balance:

- Adopt DORA metrics for software delivery performance. Metrics like change failure rate (CFR), time to detect (TTD), and time to restore (TTR) can help to identify and fix bottlenecks in your development and deployment processes. By reducing downtime and accelerating delivery, you can achieve both operational efficiency and business agility.
- Follow Site Reliability Engineering (SRE) practices to improve operational reliability. SRE's focus on automation, observability, and incident response can lead to reduced downtime, lower recovery time, and higher customer satisfaction. By minimizing downtime and improving operational reliability, you can prevent revenue loss and avoid the need to overprovision resources as a safety net to handle outages.

Enable self-service optimization

Encourage a culture of experimentation and exploration by providing your teams with self-service cost optimization tools, observability tools, and resource management platforms. Enable them to provision, manage, and optimize their cloud resources autonomously. This approach helps to foster a sense of ownership, accelerate innovation, and ensure that teams can respond quickly to changing needs while being mindful of cost efficiency.

Adopt and implement FinOps

Adopt FinOps to establish a collaborative environment where everyone is empowered to make informed decisions that balance cost and value. FinOps fosters financial accountability and promotes effective cost optimization in the cloud.

Promote a value-driven and TCO-aware mindset

Encourage your team members to adopt a holistic attitude toward cloud spending, with an emphasis on TCO and not just upfront costs. Use techniques like value stream mapping to visualize and analyze the flow of value through your software delivery process and to identify areas for improvement. Implement unit costing for your applications and services to gain a granular understanding of cost drivers and discover opportunities for cost optimization. For more information, see Maximize business value with cloud FinOps.

Foster a culture of cost awareness

This principle in the cost optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to promote cost awareness across your organization and ensure that team members have the cost information that they need to make informed decisions.

Conventionally, the responsibility for cost management might be centralized to a few select stakeholders and primarily focused on initial project architecture decisions. However, team members across all cloud user roles (analyst, architect, developer, or administrator) can help to reduce the cost of your resources in Google Cloud. By sharing cost data appropriately, you can empower team members to make cost-effective decisions throughout their development and deployment processes.

Principle overview

Stakeholders across various roles – product owners, developers, deployment engineers, administrators, and financial analysts – need visibility into relevant cost data and its relationship to business value. When provisioning and managing cloud resources, they need the following data:

- **Projected resource costs**: Cost estimates at the time of design and deployment.
- **Real-time resource usage costs**: Up-to-date cost data that can be used for ongoing monitoring and budget validation.
- **Costs mapped to business metrics**: Insights into how cloud spending affects key performance indicators (KPIs), to enable teams to identify cost-effective strategies.

Every individual might not need access to raw cost data. However, promoting cost awareness across all roles is crucial because individual decisions can affect costs.

By promoting cost visibility and ensuring clear ownership of cost management practices, you ensure that everyone is aware of the financial implications of their choices and everyone actively contributes to the organization's cost optimization goals. Whether through a centralized FinOps team or a distributed model, establishing accountability is crucial for effective cost optimization efforts.

Recommendations

To promote cost awareness and ensure that your team members have the cost information that they need to make informed decisions, consider the following recommendations.

Provide organization-wide cost visibility

To achieve organization-wide cost visibility, the teams that are responsible for cost management can take the following actions:

- **Standardize cost calculation and budgeting**: Use a consistent method to determine the full costs of cloud resources, after factoring in discounts and shared costs. Establish clear and standardized budgeting processes that align with your organization's goals and enable proactive cost management.
- **Use standardized cost management and visibility tools**: Use appropriate tools that provide real-time insights into cloud spending and generate regular (for example, weekly) cost progression snapshots. These tools enable proactive budgeting, forecasting, and identification of optimization opportunities. The tools could be cloud provider tools (like the Google Cloud Billing dashboard), third-party solutions, or open-source solutions like the Cost Attribution solution.
- **Implement a cost allocation system**: Allocate a portion of the overall cloud budget to each team or project. Such an allocation gives the teams a sense of ownership over cloud spending and encourages them to make cost-effective decisions within their allocated budget.
- **Promote transparency**: Encourage teams to discuss cost implications during the design and decision-making processes. Create a safe and supportive environment for sharing ideas and concerns related to cost optimization. Some organizations use positive reinforcement mechanisms like leaderboards or recognition programs. If your organization has restrictions on sharing raw cost data due to business concerns, explore alternative approaches for sharing cost information and insights. For example, consider sharing aggregated metrics (like the total cost for an environment or feature) or relative metrics (like the average cost per transaction or user).

Understand how cloud resources are billed

Pricing for Google Cloud resources might vary across regions. Some resources are billed monthly at a fixed price, and others might be billed based on usage. To understand how Google Cloud resources are billed, use the Google Cloud pricing calculator and product-specific pricing information (for example, Google Kubernetes Engine (GKE) pricing).

Understand resource-based cost optimization options

For each type of cloud resource that you plan to use, explore strategies to optimize utilization and efficiency. The strategies include rightsizing, autoscaling, and adopting serverless technologies where appropriate. The following are examples of cost optimization options for a few Google Cloud products:

- Cloud Run lets you configure always-allocated CPUs to handle predictable traffic loads at a fraction of the price of the default allocation method (that is, CPUs allocated only during request processing).
- You can purchase BigQuery slot commitments to save money on data analysis.
- GKE provides detailed metrics to help you understand cost optimization options.
- Understand how network pricing can affect the cost of data transfers and how you can optimize costs for specific networking services. For example, you can reduce the data transfer costs for external Application Load Balancers by using Cloud CDN or Google Cloud Armor. For more information, see Ways to lower external Application Load Balancer costs.

Understand discount-based cost optimization options

Familiarize yourself with the discount programs that Google Cloud offers, such as the following examples:

- **Committed use discounts (CUDs)**: CUDs are suitable for resources that have predictable and steady usage. CUDs let you get significant reductions in price in exchange for committing to specific resource usage over a period (typically one to three years). You can also use CUD auto-renewal to avoid having to manually repurchase commitments when they expire.
- **Sustained use discounts**: For certain Google Cloud products like Compute Engine and GKE, you can get automatic discount credits after continuous resource usage beyond specific duration thresholds.
- **Spot VMs**: For fault-tolerant and flexible workloads, Spot VMs can help to reduce your Compute Engine costs. The cost of Spot VMs is significantly lower than regular VMs. However, Compute Engine might preemptively stop or delete Spot VMs to reclaim capacity. Spot VMs are suitable for batch jobs that can tolerate preemption and don't have high availability requirements.
- **Discounts for specific product options**: Some managed services like BigQuery offer discounts when you purchase dedicated or autoscaling query processing capacity.

Evaluate and choose the discounts options that align with your workload characteristics and usage patterns.

Incorporate cost estimates into architecture blueprints

Encourage teams to develop architecture blueprints that include cost estimates for different deployment options and configurations. This practice empowers teams to compare costs proactively and make informed decisions that align with both technical and financial objectives.

Use a consistent and standard set of labels for all your resources

You can use labels to track costs and to identify and classify resources. Specifically, you can use labels to allocate costs to different projects, departments, or cost centers. Defining a formal labeling policy that aligns with the needs of the main stakeholders in your organization helps to make costs visible more widely. You can also use labels to filter resource cost and usage data based on target audience.

Use automation tools like Terraform to enforce labeling on every resource that is created. To enhance cost visibility and attribution further, you can use the tools provided by the open-source cost attribution solution.

Share cost reports with team members

By sharing cost reports with your team members, you empower them to take ownership of their cloud spending. This practice enables cost-effective decision making, continuous cost optimization, and systematic improvements to your cost allocation model.

Cost reports can be of several types, including the following:

- **Periodic cost reports**: Regular reports inform teams about their current cloud spending. Conventionally, these reports might be spreadsheet exports. More effective methods include automated emails and specialized dashboards. To ensure that cost reports provide relevant and actionable information without overwhelming recipients with unnecessary detail, the reports must be tailored to the target audiences. Setting up tailored reports is a foundational step toward more real-time and interactive cost visibility and management.
- **Automated notifications**: You can configure cost reports to proactively notify relevant stakeholders (for example, through email or chat) about cost anomalies, budget thresholds, or opportunities for cost optimization. By providing timely information directly to those who can act on it, automated alerts encourage prompt action and foster a proactive approach to cost optimization.
- **Google Cloud dashboards**: You can use the built-in billing dashboards in Google Cloud to get insights into cost breakdowns and to identify opportunities for cost optimization. Google Cloud also provides FinOps hub to help you monitor savings and get recommendations for cost optimization. An AI engine powers the FinOps hub to recommend cost optimization opportunities for all the resources that are currently deployed. To control access to these recommendations, you can implement role-based access control (RBAC).

- **Custom dashboards**: You can create custom dashboards by exporting cost data to an analytics database, like BigQuery. Use a visualization tool like Looker Studio to connect to the analytics database to build interactive reports and enable fine-grained access control through role-based permissions.
- **Multicloud cost reports**: For multicloud deployments, you need a unified view of costs across all the cloud providers to ensure comprehensive analysis, budgeting, and optimization. Use tools like BigQuery to centralize and analyze cost data from multiple cloud providers, and use Looker Studio to build team-specific interactive reports.

Optimize resource usage

This principle in the cost optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to help you plan and provision resources to match the requirements and consumption patterns of your cloud workloads.

Principle overview

To optimize the cost of your cloud resources, you need to thoroughly understand your workloads resource requirements and load patterns. This understanding is the basis for a well defined cost model that lets you forecast the total cost of ownership (TCO) and identify cost drivers throughout your cloud adoption journey. By proactively analyzing and forecasting cloud spending, you can make informed choices about resource provisioning, utilization, and cost optimization. This approach lets you control cloud spending, avoid overprovisioning, and ensure that cloud resources are aligned with the dynamic needs of your workloads and environments.

Recommendations

To effectively optimize cloud resource usage, consider the following recommendations.

Choose environment-specific resources

Each deployment environment has different requirements for availability, reliability and scalability. For example, developers might prefer an environment that lets them rapidly deploy and run applications for short durations, but might not need high availability. On the other hand, a production environment typically needs high availability. To maximize the utilization of your resources, define environment-specific requirements based on your business needs. The following table lists examples of environment-specific requirements.

Note: The requirements that are listed in this table are not exhaustive or prescriptive. They're meant to serve as examples to help you understand how requirements can vary based on the environment type.

Environment	Requirements
Production	• High availability • Predictable performance • Operational stability • Security with robust resources
Development and testing	• Cost efficiency • Flexible infrastructure with burstable capacity • Ephemeral infrastructure when data persistence is not necessary
Other environments (like staging and QA)	• Tailored resource allocation based on environment-specific requirements

Choose workload-specific resources

Each of your cloud workloads might have different requirements for availability, scalability, security, and performance. To optimize costs, you need to align resource choices with the specific requirements of each workload. For example, a stateless application might not require the same level of availability or reliability as a stateful backend. The following table lists more examples of workload-specific requirements.

Note: The requirements that are listed in this table are not exhaustive or prescriptive. They're meant to serve as examples to help you understand how requirements can vary based on the workload type.

Workload type	Workload requirements	Resource options
Mission-critical	Continuous availability, robust security, and high performance	Premium resources and managed services like Spanner for high availability and global consistency of data.
Non-critical	Cost-efficient and autoscaling infrastructure	Resources with basic features and ephemeral resources like Spot VMs.

Event-driven	Dynamic scaling based on the current demand for capacity and performance	Serverless services like Cloud Run and Cloud Run functions.
Experimental workloads	Low cost and flexible environment for rapid development, iteration, testing, and innovation	Resources with basic features, ephemeral resources like Spot VMs, and sandbox environments with defined spending limits.

A benefit of the cloud is the opportunity to take advantage of the most appropriate computing power for a given workload. Some workloads are developed to take advantage of processor instruction sets, and others might not be designed in this way. Benchmark and profile your workloads accordingly. Categorize your workloads and make workload-specific resource choices (for example, choose appropriate machine families for Compute Engine VMs). This practice helps to optimize costs, enable innovation, and maintain the level of availability and performance that your workloads need.

The following are examples of how you can implement this recommendation:

- For mission-critical workloads that serve globally distributed users, consider using Spanner. Spanner removes the need for complex database deployments by ensuring reliability and consistency of data in all regions.
- For workloads with fluctuating load levels, use autoscaling to ensure that you don't incur costs when the load is low and yet maintain sufficient capacity to meet the current load. You can configure autoscaling for many Google Cloud services, including Compute Engine VMs, Google Kubernetes Engine (GKE) clusters, and Cloud Run. When you set up autoscaling, you can configure maximum scaling limits to ensure that costs remain within specified budgets.

Select regions based on cost requirements

For your cloud workloads, carefully evaluate the available Google Cloud regions and choose regions that align with your cost objectives. The region with lowest cost might not offer optimal latency or it might not meet your sustainability requirements. Make informed decisions about where to deploy your workloads to achieve the desired balance. You can use the Google Cloud Region Picker to understand the trade-offs between cost, sustainability, latency, and other factors.

Use built-in cost optimization options

Google Cloud products provide built-in features to help you optimize resource usage and control costs. The following table lists examples of cost optimization features that you can use in some Google Cloud products:

Product	Cost optimization feature
Compute Engine	• Automatically add or remove VMs based on the current load by using autoscaling. • Avoid overprovisioning by creating and using custom machine types that match your workload's requirements. • For non-critical or fault-tolerant workloads, reduce costs by using Spot VMs. • In development environments, reduce costs by limiting the run time of VMs or by suspending or stopping VMs when you don't need them.
GKE	• Automatically adjust the size of GKE clusters based on the current load by using cluster autoscaler. • Automatically create and manage node pools based on workload requirements and ensure optimal resource utilization by using node auto-provisioning.
Cloud Storage	• Automatically transition data to lower-cost storage classes based on the age of data or based on access patterns by using Object Lifecycle Management. • Dynamically move data to the most cost-effective storage class based on usage patterns by using Autoclass.
BigQuery	• Reduce query processing costs for steady-state workloads by using capacity-based pricing. • Optimize query performance and costs by using partitioning and clustering techniques.
Google Cloud VMware Engine	• Reduce VMware costs by using cost-optimization strategies like CUDs, optimizing storage consumption, and rightsizing ESXi clusters.

Optimize resource sharing

To maximize the utilization of cloud resources, you can deploy multiple applications or services on the same infrastructure, while still meeting the security and other requirements of the applications. For example, in development and testing environments, you can use the same cloud infrastructure to test all the components of an application. For the production environment, you can deploy each component on a separate set of resources to limit the extent of impact in case of incidents.

The following are examples of how you can implement this recommendation:

- Use a single Cloud SQL instance for multiple non-production environments.
- Enable multiple development teams to share a GKE cluster by using the fleet team management feature in GKE with appropriate access controls.
- Use GKE Autopilot to take advantage of cost-optimization techniques like bin packing and autoscaling that GKE implements by default.
- For AI and ML workloads, save GPU costs by using GPU-sharing strategies like multi-instance GPUs, time-sharing GPUs, and NVIDIA MPS.

Develop and maintain reference architectures

Create and maintain a repository of reference architectures that are tailored to meet the requirements of different deployment environments and workload types. To streamline the design and implementation process for individual projects, the blueprints can be centrally managed by a team like a Cloud Center of Excellence (CCoE). Project teams can choose suitable blueprints based on clearly defined criteria, to ensure architectural consistency and adoption of best practices. For requirements that are unique to a project, the project team and the central architecture team should collaborate to design new reference architectures. You can share the reference architectures across the organization to foster knowledge sharing and expand the repository of available solutions. This approach ensures consistency, accelerates development, simplifies decision-making, and promotes efficient resource utilization.

Review the reference architectures provided by Google for various use cases and technologies. These reference architectures incorporate best practices for resource selection, sizing, configuration, and deployment. By using these reference architectures, you can accelerate your development process and achieve cost savings from the start.

Enforce cost discipline by using organization policies

Consider using organization policies to limit the available Google Cloud locations and products that team members can use. These policies help to ensure that teams adhere to cost-effective solutions and provision resources in locations that are aligned with your cost optimization goals.

Estimate realistic budgets and set financial boundaries

Develop detailed budgets for each project, workload, and deployment environment. Make sure that the budgets cover all aspects of cloud operations, including infrastructure costs, software licenses, personnel, and anticipated growth. To prevent overspending and ensure alignment with your financial goals, establish clear spending limits or thresholds for projects, services, or specific resources. Monitor cloud spending regularly against these limits. You can use proactive quota alerts to identify potential cost overruns early and take timely corrective action.

In addition to setting budgets, you can use quotas and limits to help enforce cost discipline and prevent unexpected spikes in spending. You can exercise granular control over resource consumption by setting quotas at various levels, including projects, services, and even specific resource types.

The following are examples of how you can implement this recommendation:

- **Project-level quotas**: Set spending limits or resource quotas at the project level to establish overall financial boundaries and control resource consumption across all the services within the project.
- **Service-specific quotas**: Configure quotas for specific Google Cloud services like Compute Engine or BigQuery to limit the number of instances, CPUs, or storage capacity that can be provisioned.
- **Resource type-specific quotas**: Apply quotas to individual resource types like Compute Engine VMs, Cloud Storage buckets, Cloud Run instances, or GKE nodes to restrict their usage and prevent unexpected cost overruns.
- **Quota alerts**: Get notifications when your quota usage (at the project level) reaches a percentage of the maximum value.

By using quotas and limits in conjunction with budgeting and monitoring, you can create a proactive and multi-layered approach to cost control. This approach helps to ensure that your cloud spending remains within defined boundaries and aligns with your business objectives. Remember, these cost controls are not permanent or rigid. To ensure that the cost controls remain aligned with current industry standards and reflect your evolving business needs, you must review the controls regularly and adjust them to include new technologies and best practices.

Optimize continuously

This principle in the cost optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to help you optimize the cost of your cloud deployments based on constantly changing and evolving business goals.

As your business grows and evolves, your cloud workloads need to adapt to changes in resource requirements and usage patterns. To derive maximum value from your cloud spending, you must maintain cost-efficiency while continuing to support business objectives. This requires a proactive and adaptive approach that focuses on continuous improvement and optimization.

Principle overview

To optimize cost continuously, you must proactively monitor and analyze your cloud environment and make suitable adjustments to meet current requirements. Focus your monitoring efforts on key performance indicators (KPIs) that directly affect your end users' experience, align with your business goals, and provide insights for continuous improvement. This approach lets you identify and address inefficiencies, adapt to changing needs, and continuously align cloud spending with strategic business goals. To balance comprehensive observability with cost effectiveness, understand the costs and benefits of monitoring resource usage and use appropriate process-improvement and optimization strategies.

Recommendations

To effectively monitor your Google Cloud environment and optimize cost continuously, consider the following recommendations.

Focus on business-relevant metrics

Effective monitoring starts with identifying the metrics that are most important for your business and customers. These metrics include the following:

- **User experience metrics**: Latency, error rates, throughput, and customer satisfaction metrics are useful for understanding your end users' experience when using your applications.
- **Business outcome metrics**: Revenue, customer growth, and engagement can be correlated with resource usage to identify opportunities for cost optimization.
- **DevOps Research & Assessment (DORA) metrics**: Metrics like deployment frequency, lead time for changes, change failure rate, and time to restore provide insights into the efficiency and reliability of your software delivery process. By improving these metrics, you can increase productivity, reduce downtime, and optimize cost.
- **Site Reliability Engineering (SRE) metrics**: Error budgets help teams to quantify and manage the acceptable level of service disruption. By establishing clear expectations for reliability, error budgets empower teams to innovate and deploy changes more confidently, knowing their safety margin. This proactive approach promotes a balance between innovation and stability, helping prevent excessive operational costs associated with major outages or prolonged downtime.

Use observability for resource optimization

The following are recommendations to use observability to identify resource bottlenecks and underutilized resources in your cloud deployments:

- **Monitor resource utilization**: Use resource utilization metrics to identify Google Cloud resources that are underutilized. For example, use metrics like CPU and memory utilization to identify idle VM resources. For Google Kubernetes Engine (GKE), you can view a detailed breakdown of costs and cost-related optimization metrics. For Google Cloud VMware Engine, review resource utilization to optimize CUDs, storage consumption, and ESXi right-sizing.
- **Use cloud recommendations**: Active Assist is a portfolio of intelligent tools that help you optimize your cloud operations. These tools provide actionable recommendations to reduce costs, increase performance, improve security and even make sustainability-focused decisions. For example, VM rightsizing insights can help to optimize resource allocation and avoid unnecessary spending.
- **Correlate resource utilization with performance**: Analyze the relationship between resource utilization and application performance to determine whether you can downgrade to less expensive resources without affecting the user experience.

Balance troubleshooting needs with cost

Detailed observability data can help with diagnosing and troubleshooting issues. However, storing excessive amounts of observability data or exporting unnecessary data to external monitoring tools can lead to unnecessary costs. For efficient troubleshooting, consider the following recommendations:

- **Collect sufficient data for troubleshooting**: Ensure that your monitoring solution captures enough data to efficiently diagnose and resolve issues when they arise. This data might include logs, traces, and metrics at various levels of granularity.
- **Use sampling and aggregation**: Balance the need for detailed data with cost considerations by using sampling and aggregation techniques. This approach lets you collect representative data without incurring excessive storage costs.
- **Understand the pricing models of your monitoring tools and services**: Evaluate different monitoring solutions and choose options that align with your project's specific needs, budget, and usage patterns. Consider factors like data volume, retention requirements, and the required features when making your selection.
- **Regularly review your monitoring configuration**: Avoid collecting excessive data by removing unnecessary metrics or logs.

Tailor data collection to roles and set role-specific retention policies

Consider the specific data needs of different roles. For example, developers might primarily need access to traces and application-level logs, whereas IT administrators might focus on system logs and infrastructure metrics. By tailoring data collection, you can reduce unnecessary storage costs and avoid overwhelming users with irrelevant information.

Additionally, you can define retention policies based on the needs of each role and any regulatory requirements. For example, developers might need access to detailed logs for a shorter period, while financial analysts might require longer-term data.

Consider regulatory and compliance requirements

In certain industries, regulatory requirements mandate data retention. To avoid legal and financial risks, you need to ensure that your monitoring and data retention practices help you adhere to relevant regulations. At the same time, you need to maintain cost efficiency. Consider the following recommendations:

- Determine the specific data retention requirements for your industry or region, and ensure that your monitoring strategy meets the requirements of those requirements.
- Implement appropriate data archival and retrieval mechanisms to meet audit and compliance needs while minimizing storage costs.

Implement smart alerting

Alerting helps to detect and resolve issues in a timely manner. However, a balance is necessary between an approach that keeps you informed, and one that overwhelms you with notifications. By designing intelligent alerting systems, you can prioritize critical issues that have higher business impact. Consider the following recommendations:

- **Prioritize issues that affect customers**: Design alerts that trigger rapidly for issues that directly affect the customer experience, like website outages, slow response times, or transaction failures.
- **Tune for temporary problems**: Use appropriate thresholds and delay mechanisms to avoid unnecessary alerts for temporary problems or self-healing system issues that don't affect customers.
- **Customize alert severity**: Ensure that the most urgent issues receive immediate attention by differentiating between critical and noncritical alerts.
- **Use notification channels wisely**: Choose appropriate channels for alert notifications (email, SMS, or paging) based on the severity and urgency of the alerts.

Well-Architected Framework: Performance optimization pillar

Last reviewed 2025-02-14 UTC

This page provides a one-page view of all of the pages in the performance optimization pillar of the Well-Architected Framework. You can print this page or save it in PDF format by using your browser's print function.

This page doesn't have a table of contents. You can't use the links on this page to navigate within the page.

This pillar in the Google Cloud Well-Architected Framework provides recommendations to optimize the performance of workloads in Google Cloud.

This document is intended for architects, developers, and administrators who plan, design, deploy, and manage workloads in Google Cloud.

The recommendations in this pillar can help your organization to operate efficiently, improve customer satisfaction, increase revenue, and reduce cost. For example, when the backend processing time of an application decreases, users experience faster response times, which can lead to higher user retention and more revenue.

The performance optimization process can involve a trade-off between performance and cost. However, optimizing performance can sometimes help you reduce costs. For example, when the load increases, autoscaling can help to provide predictable performance by ensuring that the system resources aren't overloaded. Autoscaling also helps you to reduce costs by removing unused resources during periods of low load.

Performance optimization is a continuous process, not a one-time activity. The following diagram shows the stages in the performance optimization process:

The performance optimization process is an ongoing cycle that includes the following stages:

1. **Define requirements**: Define granular performance requirements for each layer of the application stack before you design and develop your applications. To plan resource allocation, consider the key workload characteristics and performance expectations.
2. **Design and deploy**: Use elastic and scalable design patterns that can help you meet your performance requirements.
3. **Monitor and analyze**: Monitor performance continually by using logs, tracing, metrics, and alerts.

4. **Optimize**: Consider potential redesigns as your applications evolve. Rightsize cloud resources and use new features to meet changing performance requirements.

As shown in the preceding diagram, continue the cycle of monitoring, re-assessing requirements, and adjusting the cloud resources.

For performance optimization principles and recommendations that are specific to AI and ML workloads, see AI and ML perspective: Performance optimization in the Well-Architected Framework.

Core principles

The recommendations in the performance optimization pillar of the Well-Architected Framework are mapped to the following core principles:

- Plan resource allocation
- Take advantage of elasticity
- Promote modular design
- Continuously monitor and improve performance

Contributors

Authors:

- Daniel Lees | Cloud Security Architect
- Gary Harmson | Principal Architect
- Luis Urena | Developer Relations Engineer
- Zach Seils | Networking Specialist

Other contributors:

- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Jose Andrade | Customer Engineer, SRE Specialist
- Kumar Dhanagopal | Cross-Product Solution Developer
- Marwan Al Shawi | Partner Customer Engineer
- Nicolas Pintaux | Customer Engineer, Application Modernization Specialist
- Ryan Cox | Principal Architect
- Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework
- Samantha He | Technical Writer

- Wade Holmes | Global Solutions Director

Plan resource allocation

This principle in the performance optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to help you plan resources for your workloads in Google Cloud. It emphasizes the importance of defining granular requirements before you design and develop applications for cloud deployment or migration.

Principle overview

To meet your business requirements, it's important that you define the performance requirements for your applications, before design and development. Define these requirements as granularly as possible for the application as a whole and for each layer of the application stack. For example, in the storage layer, you must consider the throughput and I/O operations per second (IOPS) that the applications need.

From the beginning, plan application designs with performance and scalability in mind. Consider factors such as the number of users, data volume, and potential growth over time.

Performance requirements for each workload vary and depend on the type of workload. Each workload can contain a mix of component systems and services that have unique sets of performance characteristics. For example, a system that's responsible for periodic batch processing of large datasets has different performance demands than an interactive virtual desktop solution. Your optimization strategies must address the specific needs of each workload.

Select services and features that align with the performance goals of each workload. For performance optimization, there's no one-size-fits-all solution. When you optimize each workload, the entire system can achieve optimal performance and efficiency.

Consider the following workload characteristics that can influence your performance requirements:

- **Deployment archetype**: The deployment archetype that you select for an application can influence your choice of products and features, which then determine the performance that you can expect from your application.
- **Resource placement**: When you select a Google Cloud region for your application resources, we recommend that you prioritize low latency for end users, adhere to data-locality regulations, and ensure the availability of required Google Cloud products and services.

- **Network connectivity**: Choose networking services that optimize data access and content delivery. Take advantage of Google Cloud's global network, high-speed backbones, interconnect locations, and caching services.
- **Application hosting options**: When you select a hosting platform, you must evaluate the performance advantages and disadvantages of each option. For example, consider bare metal, virtual machines, containers, and serverless platforms.
- **Storage strategy**: Choose an optimal storage strategy that's based on your performance requirements.
- **Resource configurations**: The machine type, IOPS, and throughput can have a significant impact on performance. Additionally, early in the design phase, you must consider appropriate security capabilities and their impact on resources. When you plan security features, be prepared to accommodate the necessary performance trade-offs to avoid any unforeseen effects.

Recommendations

To ensure optimal resource allocation, consider the recommendations in the following sections.

Configure and manage quotas

Ensure that your application uses only the necessary resources, such as memory, storage, and processing power. Over-allocation can lead to unnecessary expenses, while under-allocation might result in performance degradation.

To accommodate elastic scaling and to ensure that adequate resources are available, regularly monitor the capacity of your quotas. Additionally, track quota usage to identify potential scaling constraints or over-allocation issues, and then make informed decisions about resource allocation.

Educate and promote awareness

Inform your users about the performance requirements and provide educational resources about effective performance management techniques.

To evaluate progress and to identify areas for improvement, regularly document the target performance and the actual performance. Load test your application to find potential breakpoints and to understand how you can scale the application.

Monitor performance metrics

Use Cloud Monitoring to analyze trends in performance metrics, to analyze the effects of experiments, to define alerts for critical metrics, and to perform retrospective analyses.

Active Assist is a set of tools that can provide insights and recommendations to help optimize resource utilization. These recommendations can help you to adjust resource allocation and improve performance.

Take advantage of elasticity

This principle in the performance optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to help you incorporate elasticity, which is the ability to adjust resources dynamically based on changes in workload requirements.

Elasticity allows different components of a system to scale independently. This targeted scaling can help improve performance and cost efficiency by allocating resources precisely where they're needed, without over provisioning or under provisioning your resources.

Principle overview

The performance requirements of a system directly influence when and how the system scales vertically or scales horizontally. You need to evaluate the system's capacity and determine the load that the system is expected to handle at baseline. Then, you need to determine how you want the system to respond to increases and decreases in the load.

When the load increases, the system must scale out horizontally, scale up vertically, or both. For horizontal scaling, add replica nodes to ensure that the system has sufficient overall capacity to fulfill the increased demand. For vertical scaling, replace the application's existing components with components that contain more capacity, more memory, and more storage.

When the load decreases, the system must scale down (horizontally, vertically, or both).

Define the *circumstances* in which the system scales up or scales down. Plan to manually scale up systems for known periods of high traffic. Use tools like autoscaling, which responds to increases or decreases in the load.

Recommendations

To take advantage of elasticity, consider the recommendations in the following sections.

Plan for peak load periods

You need to plan an efficient scaling path for known events, such as expected periods of increased customer demand.

Consider scaling up your system ahead of known periods of high traffic. For example, if you're a retail organization, you expect demand to increase during seasonal sales. We recommend that you manually scale up or scale out your systems before those sales to ensure that your system can immediately handle the increased load or immediately adjust existing limits. Otherwise, the system might take several minutes to add resources in response to real-time changes. Your application's capacity might not increase quickly enough and cause some users to experience delays.

For unknown or unexpected events, such as a sudden surge in demand or traffic, you can use autoscaling features to trigger elastic scaling that's based on metrics. These metrics can include CPU utilization, load balancer serving capacity, latency, and even custom metrics that you define in Cloud Monitoring.

For example, consider an application that runs on a Compute Engine managed instance group (MIG). This application has a requirement that each instance performs optimally until the average CPU utilization reaches 75%. In this example, you might define an autoscaling policy that creates more instances when the CPU utilization reaches the threshold. These newly-created instances help absorb the load, which helps ensure that the average CPU utilization remains at an optimal rate until the maximum number of instances that you've configured for the MIG is reached. When the demand decreases, the autoscaling policy removes the instances that are no longer needed.

Plan resource slot reservations in BigQuery or adjust the limits for autoscaling configurations in Spanner by using the managed autoscaler.

Use predictive scaling

If your system components include Compute Engine, you must evaluate whether predictive autoscaling is suitable for your workload. Predictive autoscaling forecasts the future load based on your metrics' historical trends—for example, CPU utilization. Forecasts are recomputed every few minutes, so the autoscaler rapidly adapts its forecast to very recent changes in load. Without predictive autoscaling, an autoscaler can only scale a group reactively, based on observed real-time changes in load. Predictive autoscaling works with both real-time data and historical data to respond to both the current and the forecasted load.

Implement serverless architectures

Consider implementing a serverless architecture with serverless services that are inherently elastic, such as the following:

- Cloud Run
- Cloud Run functions
- BigQuery
- Spanner
- Eventarc

- Workflows
- Pub/Sub

Unlike autoscaling in other services that require fine-tuning rules (for example, Compute Engine), serverless autoscaling is instant and can scale down to zero resources.

Use Autopilot mode for Kubernetes

For complex applications that require greater control over Kubernetes, consider Autopilot mode in Google Kubernetes Engine (GKE). Autopilot mode provides automation and scalability by default. GKE automatically scales nodes and resources based on traffic. GKE manages nodes, creates new nodes for your applications, and configures automatic upgrades and repairs.

Promote modular design

This principle in the performance optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to help you promote a modular design. Modular components and clear interfaces can enable flexible scaling, independent updates, and future component separation.

Principle overview

Understand the dependencies between the application components and the system components to design a scalable system.

Modular design enables flexibility and resilience, regardless of whether a monolithic or microservices architecture was initially deployed. By decomposing the system into well-defined, independent modules with clear interfaces, you can scale individual components to meet specific demands.

Targeted scaling can help optimize resource utilization and reduce costs in the following ways:

- Provisions only the necessary resources to each component, and allocates fewer resources to less-demanding components.
- Adds more resources during high-traffic periods to maintain the user experience.
- Removes under-utilized resources without compromising performance.

Modularity also enhances maintainability. Smaller, self-contained units are easier to understand, debug, and update, which can lead to faster development cycles and reduced risk.

While modularity offers significant advantages, you must evaluate the potential performance trade-offs. The increased communication between modules can introduce latency and overhead. Strive for a balance between modularity and performance. A highly modular design might not be universally suitable. When performance is critical, a more tightly coupled approach might be appropriate. System design is an iterative process, in which you continuously review and refine your modular design.

Recommendations

To promote modular designs, consider the recommendations in the following sections.

Design for loose coupling

Design a loosely coupled architecture. Independent components with minimal dependencies can help you build scalable and resilient applications. As you plan the boundaries for your services, you must consider the availability and scalability requirements. For example, if one component has requirements that are different from your other components, you can design the component as a standalone service. Implement a plan for graceful failures for less-important subprocesses or services that don't impact the response time of the primary services.

Design for concurrency and parallelism

Design your application to support multiple tasks concurrently, like processing multiple user requests or running background jobs while users interact with your system. Break large tasks into smaller chunks that can be processed at the same time by multiple service instances. Task concurrency lets you use features like autoscaling to increase the resource allocation in products like the following:

- Compute Engine
- GKE
- BigQuery
- Spanner

Balance modularity for flexible resource allocation

Where possible, ensure that each component uses only the necessary resources (like memory, storage, and processing power) for specific operations. Resource over-allocation can result in unnecessary costs, while resource under-allocation can compromise performance.

Use well-defined interfaces

Ensure modular components communicate effectively through clear, standardized interfaces (like APIs and message queues) to reduce overhead from translation layers or from extraneous traffic.

Use stateless models

A stateless model can help ensure that you can handle each request or interaction with the service independently from previous requests. This model facilitates scalability and recoverability, because you can grow, shrink, or restart the service without losing the data necessary for in-progress requests or processes.

Choose complementary technologies

Choose technologies that complement the modular design. Evaluate programming languages, frameworks, and databases for their modularity support.

For more information, see the following resources:

- Re-architecting to cloud native
- Introduction to microservices

Continuously monitor and improve performance

This principle in the performance optimization pillar of the Google Cloud Well-Architected Framework provides recommendations to help you continuously monitor and improve performance.

After you deploy applications, continuously monitor their performance by using logs, tracing, metrics, and alerts. As your applications grow and evolve, you can use the trends in these data points to re-assess your performance requirements. You might eventually need to redesign parts of your applications to maintain or improve their performance.

Principle overview

The process of continuous performance improvement requires robust monitoring tools and strategies. Cloud observability tools can help you to collect key performance indicators (KPIs) such as latency, throughput, error rates, and resource utilization. Cloud environments offer a variety of methods to conduct granular performance assessments across the application, the network, and the end-user experience.

Improving performance is an ongoing effort that requires a multi-faceted approach. The following key mechanisms and processes can help you to boost performance:

- To provide clear direction and help track progress, define performance objectives that align with your business goals. Set SMART goals: specific, measurable, achievable, relevant, and time-bound.
- To measure performance and identify areas for improvement, gather KPI metrics.

- To continuously monitor your systems for issues, use visualized workflows in monitoring tools. Use architecture process mapping techniques to identify redundancies and inefficiencies.
- To create a culture of ongoing improvement, provide training and programs that support your employees' growth.
- To encourage proactive and continuous improvement, incentivize your employees and customers to provide ongoing feedback about your application's performance.

Recommendations

To promote modular designs, consider the recommendations in the following sections.

Define clear performance goals and metrics

Define clear performance objectives that align with your business goals. This requires a deep understanding of your application's architecture and the performance requirements of each application component.

As a priority, optimize the most critical components that directly influence your core business functions and user experience. To help ensure that these components continue to run efficiently and meet your business needs, set specific and measurable performance targets. These targets can include response times, error rates, and resource utilization thresholds.

This proactive approach can help you to identify and address potential bottlenecks, optimize resource allocation, and ultimately deliver a seamless and high-performing experience for your users.

Monitor performance

Continuously monitor your cloud systems for performance issues and set up alerts for any potential problems. Monitoring and alerts can help you to catch and fix issues before they affect users. Application profiling can help to identify bottlenecks and can help to optimize resource use.

You can use tools that facilitate effective troubleshooting and network optimization. Use Google Cloud Observability to identify areas that have high CPU consumption, memory consumption, or network consumption. These capabilities can help developers improve efficiency, reduce costs, and enhance the user experience. Network Intelligence Center shows visualizations of the topology of your network infrastructure, and can help you to identify high-latency paths.

Incentivize continuous improvement

Create a culture of ongoing improvement that can benefit both the application and the user experience.

Provide your employees with training and development opportunities that enhance their skills and knowledge in performance techniques across cloud services. Establish a community of practice (CoP) and offer mentorship and coaching programs to support employee growth.

To prevent *reactive* performance management and encourage proactive performance management, encourage ongoing feedback from your employees, your customers, and your stakeholders. You can consider gamifying the process by tracking KPIs on performance and presenting those metrics to teams on a frequent basis in the form of a league table.

To understand your performance and user happiness over time, we recommend that you measure user feedback quantitatively and qualitatively. The HEART framework can help you capture user feedback across five categories:

- Happiness
- Engagement
- Adoption
- Retention
- Task success

By using such a framework, you can incentivize engineers with data-driven feedback, user-centered metrics, actionable insights, and a clear understanding of goals.

Design for environmental sustainability

Last reviewed 2024-08-27 UTC

This document in the Google Cloud Well-Architected Framework summarizes how you can approach environmental sustainability for your workloads in Google Cloud. It includes information about how to minimize your carbon footprint on Google Cloud.

Understand your carbon footprint

To understand the carbon footprint from your Google Cloud usage, use the Carbon Footprint dashboard. The Carbon Footprint dashboard attributes emissions to the Google Cloud projects that you own and the cloud services that you use.

Choose the most suitable cloud regions

One effective way to reduce carbon emissions is to choose cloud regions with lower carbon emissions. To help you make this choice, Google publishes carbon data for all Google Cloud regions.

When you choose a region, you might need to balance lowering emissions with other requirements, such as pricing and network latency. To help select a region, use the Google Cloud Region Picker.

Choose the most suitable cloud services

To help reduce your existing carbon footprint, consider migrating your on-premises VM workloads to Compute Engine.

Consider serverless options for workloads that don't need VMs. These managed services often optimize resource usage automatically, reducing costs and carbon footprint.

Minimize idle cloud resources

Idle resources incur unnecessary costs and emissions. Some common causes of idle resources include the following:

Unused active cloud resources, such as idle VM instances.

Over-provisioned resources, such as larger VM instances machine types than necessary for a workload.

Non-optimal architectures, such as lift-and-shift migrations that aren't always optimized for efficiency. Consider making incremental improvements to these architectures.

The following are some general strategies to help minimize wasted cloud resources:

Identify idle or overprovisioned resources and either delete them or rightsize them.

Refactor your architecture to incorporate a more optimal design.

Migrate workloads to managed services.

Reduce emissions for batch workloads

Run batch workloads in regions with lower carbon emissions. For further reductions, run workloads at times that coincide with lower grid carbon intensity when possible.

What's next

Learn how to use Carbon Footprint data to measure, report, and reduce your cloud carbon emissions.

Well-Architected Framework: AI and ML perspective

Last reviewed 2025-02-14 UTC

This page provides a one-page view of all of the pages in the AI and ML perspective of the Well-Architected Framework. You can print this page or save it in PDF format by using your browser's print function.

This page doesn't have a table of contents. You can't use the links on this page to navigate within the page.

This document in the Google Cloud Well-Architected Framework describes principles and recommendations to help you to design, build, and manage AI and ML workloads in Google Cloud that meet your operational, security, reliability, cost, and performance goals.

The target audience for this document includes decision makers, architects, administrators, developers, and operators who design, build, deploy, and maintain AI and ML workloads in Google Cloud.

The following pages describe principles and recommendations that are specific to AI and ML, for each pillar of the Well-Architected Framework:

- AI and ML perspective: Operational excellence
- AI and ML perspective: Security
- AI and ML perspective: Reliability
- AI and ML perspective: Cost optimization
- AI and ML perspective: Performance optimization

Contributors

Authors:

- Benjamin Sadik | AI and ML Specialist Customer Engineer

- Charlotte Gistelinck, PhD | Partner Engineer
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Isaac Lo | AI Business Development Manager
- Kamilla Kurta | GenAI/ML Specialist Customer Engineer
- Mohamed Fawzi | Benelux Security and Compliance Lead
- Rick (Rugui) Chen | AI Infrastructure Field Solutions Architect
- Sannya Dang | AI Solution Architect

Other contributors:

- Daniel Lees | Cloud Security Architect
- Gary Harmson | Principal Architect
- Jose Andrade | Customer Engineer, SRE Specialist
- Kumar Dhanagopal | Cross-Product Solution Developer
- Marwan Al Shawi | Partner Customer Engineer
- Nicolas Pintaux | Customer Engineer, Application Modernization Specialist
- Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework
- Ryan Cox | Principal Architect
- Samantha He | Technical Writer
- Stef Ruinard | Generative AI Field Solutions Architect
- Wade Holmes | Global Solutions Director
- Zach Seils | Networking Specialist

AI and ML perspective: Operational excellence

This document in the Well-Architected Framework: AI and ML perspective provides an overview of the principles and recommendations to build and operate robust AI and ML systems on Google Cloud. These recommendations help you set up foundational elements like observability, automation, and scalability. The recommendations in this document align with the operational excellence pillar of the Google Cloud Well-Architected Framework.

Operational excellence within the AI and ML domain is the ability to seamlessly deploy, manage, and govern the AI and ML systems and pipelines that help drive your organization's strategic objectives. Operational excellence lets you respond efficiently to changes, reduce operational complexity, and ensure that your operations remain aligned with business goals.

The recommendations in this document are mapped to the following core principles:

- Build a robust foundation for model development
- Automate the model development lifecycle
- Implement observability
- Build a culture of operational excellence
- Design for scalability

Build a robust foundation for model development

To develop and deploy scalable, reliable AI systems that help you achieve your business goals, a robust model-development foundation is essential. Such a foundation enables consistent workflows, automates critical steps in order to reduce errors, and ensures that the models can scale with demand. A strong model-development foundation ensures that your ML systems can be updated, improved, and retrained seamlessly. The foundation also helps you to align your models' performance with business needs, deploy impactful AI solutions quickly, and adapt to changing requirements.

To build a robust foundation to develop your AI models, consider the following recommendations.

Define the problems and the required outcomes

Before you start any AI or ML project, you must have a clear understanding of the business problems to be solved and the required outcomes. Start with an outline of the business objectives and break the objectives down into measurable key performance indicators (KPIs). To organize and document your problem definitions and hypotheses in a Jupyter notebook environment, use tools like Vertex AI Workbench. To implement versioning for code and documents and to document your projects, goals, and assumptions, use tools like Git. To develop and manage prompts for generative AI applications, you can use Vertex AI Studio.

Collect and preprocess the necessary data

To implement data preprocessing and transformation, you can use Dataflow (for Apache Beam), Dataproc (for Apache Spark), or BigQuery if an SQL-based process is appropriate. To validate schemas and detect anomalies, use TensorFlow Data Validation (TFDV) and take advantage of automated data quality scans in BigQuery where applicable.

For generative AI, data quality includes accuracy, relevance, diversity, and alignment with the required output characteristics. In cases where real-world data is insufficient or imbalanced, you can generate synthetic data to help improve model robustness and generalization. To create synthetic datasets based on existing patterns or to augment training data for better model performance, use BigQuery DataFrames and Gemini. Synthetic data is particularly valuable for generative AI because it can help improve prompt diversity and overall model robustness. When you build datasets for fine-tuning generative AI models, consider using the synthetic data generation capabilities in Vertex AI.

For generative AI tasks like fine-tuning or reinforcement learning from human feedback (RLHF), ensure that labels accurately reflect the quality, relevance, and safety of the generated outputs.

Select an appropriate ML approach

When you design your model and parameters, consider the model's complexity and computational needs. Depending on the task (such as classification, regression, or generation), consider using Vertex AI custom training for custom model building or AutoML for simpler ML tasks. For common applications, you can also access pretrained models through Vertex AI Model Garden. You can experiment with a variety of state-of-the-art foundation models for various use cases, such as generating text, images, and code.

You might want to fine-tune a pretrained foundation model to achieve optimal performance for your specific use case. For high-performance requirements in custom training, configure Cloud Tensor Processing Units (TPUs) or GPU resources to accelerate the training and inference of deep-learning models, like large language models (LLMs) and diffusion models.

Set up version control for code, models, and data

To manage and deploy code versions effectively, use tools like GitHub or GitLab. These tools provide robust collaboration features, branching strategies, and integration with CI/CD pipelines to ensure a streamlined development process.

Use appropriate solutions to manage each artifact of your ML system, like the following examples:

- For code artifacts like container images and pipeline components, Artifact Registry provides a scalable storage solution that can help improve security. Artifact Registry also includes versioning and can integrate with Cloud Build and Cloud Deploy.
- To manage data artifacts, like datasets used for training and evaluation, use solutions like BigQuery or Cloud Storage for storage and versioning.

- To store metadata and pointers to data locations, use your version control system or a separate data catalog.

To maintain the consistency and versioning of your feature data, use Vertex AI Feature Store. To track and manage model artifacts, including binaries and metadata, use Vertex AI Model Registry, which lets you store, organize, and deploy model versions seamlessly.

To ensure model reliability, implement Vertex AI Model Monitoring. Detect data drift, track performance, and identify anomalies in production. For generative AI systems, monitor shifts in output quality and safety compliance.

Automate the model-development lifecycle

Automation helps you to streamline every stage of the AI and ML lifecycle. Automation reduces manual effort and standardizes processes, which leads to enhanced operational efficiency and a lower risk of errors. Automated workflows enable faster iteration, consistent deployment across environments, and more reliable outcomes, so your systems can scale and adapt seamlessly.

To automate the development lifecycle of your AI and ML systems, consider the following recommendations.

Use a managed pipeline orchestration system

Use Vertex AI Pipelines to automate every step of the ML lifecycle—from data preparation to model training, evaluation, and deployment. To accelerate deployment and promote consistency across projects, automate recurring tasks with scheduled pipeline runs, monitor workflows with execution metrics, and develop reusable pipeline templates for standardized workflows. These capabilities extend to generative AI models, which often require specialized steps like prompt engineering, response filtering, and human-in-the-loop evaluation. For generative AI, Vertex AI Pipelines can automate these steps, including the evaluation of generated outputs against quality metrics and safety guidelines. To improve prompt diversity and model robustness, automated workflows can also include data augmentation techniques.

Implement CI/CD pipelines

To automate the building, testing, and deployment of ML models, use Cloud Build. This service is particularly effective when you run test suites for application code, which ensures that the infrastructure, dependencies, and model packaging meet your deployment requirements.

ML systems often need additional steps beyond code testing. For example, you need to stress test the models under varying loads, perform bulk evaluations to assess model performance across diverse datasets, and validate data integrity before retraining. To simulate realistic workloads for stress tests, you can use tools like Locust, Grafana k6, or Apache JMeter. To identify bottlenecks, monitor key metrics like latency, error rate, and resource utilization through Cloud Monitoring. For generative AI, the testing must also include evaluations that are specific to the type of generated content, such as text quality, image fidelity, or code functionality. These evaluations can involve automated metrics like perplexity for language models or human-in-the-loop evaluation for more nuanced aspects like creativity and safety.

To implement testing and evaluation tasks, you can integrate Cloud Build with other Google Cloud services. For example, you can use Vertex AI Pipelines for automated model evaluation, BigQuery for large-scale data analysis, and Dataflow pipeline validation for feature validation.

You can further enhance your CI/CD pipeline by using Vertex AI for continuous training to enable automated retraining of models on new data. Specifically for generative AI, to keep the generated outputs relevant and diverse, the retraining might involve automatically updating the models with new training data or prompts. You can use Vertex AI Model Garden to select the latest base models that are available for tuning. This practice ensures that the models remain current and optimized for your evolving business needs.

Implement safe and controlled model releases

To minimize risks and ensure reliable deployments, implement a model release approach that lets you detect issues early, validate performance, and roll back quickly when required.

To package your ML models and applications into container images and deploy them, use Cloud Deploy. You can deploy your models to Vertex AI endpoints.

Implement controlled releases for your AI applications and systems by using strategies like canary releases. For applications that use managed models like Gemini, we recommend that you gradually release new application versions to a subset of users before the full deployment. This approach lets you detect potential issues early, especially when you use generative AI models where outputs can vary.

To release fine-tuned models, you can use Cloud Deploy to manage the deployment of the model versions, and use the canary release strategy to minimize risk. With managed models and fine-tuned models, the goal of controlled releases is to test changes with a limited audience before you release the applications and models to all users.

For robust validation, use Vertex AI Experiments to compare new models against existing ones, and use Vertex AI model evaluation to assess model performance. Specifically for generative AI, define evaluation metrics that align with the intended use case and the potential risks. You can use the Gen AI evaluation service in Vertex AI to assess metrics like toxicity, coherence, factual accuracy, and adherence to safety guidelines.

To ensure deployment reliability, you need a robust rollback plan. For traditional ML systems, use Vertex AI Model Monitoring to detect data drift and performance degradation. For generative AI models, you can track relevant metrics and set up alerts for shifts in output quality or the emergence of harmful content by using Vertex AI model evaluation along with Cloud Logging and Cloud Monitoring. Configure alerts based on generative AI-specific metrics to trigger rollback procedures when necessary. To track model lineage and revert to the most recent stable version, use insights from Vertex AI Model Registry.

Implement observability

The behavior of AI and ML systems can change over time due to changes in the data or environment and updates to the models. This dynamic nature makes observability crucial to detect performance issues, biases, or unexpected behavior. This is especially true for generative AI models because the outputs can be highly variable and subjective. Observability lets you proactively address unexpected behavior and ensure that your AI and ML systems remain reliable, accurate, and fair.

To implement observability for your AI and ML systems, consider the following recommendations.

Monitor performance continuously

Use metrics and success criteria for ongoing evaluation of models after deployment.

You can use Vertex AI Model Monitoring to proactively track model performance, identify training-serving skew and prediction drift, and receive alerts to trigger necessary model retraining or other interventions. To effectively monitor for training-serving skew, construct a golden dataset that represents the ideal data distribution, and use TFDV to analyze your training data and establish a baseline schema.

Configure Model Monitoring to compare the distribution of input data against the golden dataset for automatic skew detection. For traditional ML models, focus on metrics like accuracy, precision, recall, F1-score, AUC-ROC, and log loss. Define custom thresholds for alerts in Model Monitoring. For generative AI, use the Gen AI evaluation service to continuously monitor model output in production. You can also enable automatic evaluation metrics for response quality, safety, instruction adherence, grounding, writing style, and verbosity. To assess the generated outputs for quality, relevance, safety, and adherence to guidelines, you can incorporate human-in-the-loop evaluation.

Create feedback loops to automatically retrain models with Vertex AI Pipelines when Model Monitoring triggers an alert. Use these insights to improve your models continuously.

Evaluate models during development

Before you deploy your LLMs and other generative AI models, thoroughly evaluate them during the development phase. Use Vertex AI model evaluation to achieve optimal performance and to mitigate risk. Use Vertex AI rapid evaluation to let Google Cloud automatically run evaluations based on the dataset and prompts that you provide.

You can also define and integrate custom metrics that are specific to your use case. For feedback on generated content, integrate human-in-the-loop workflows by using Vertex AI Model Evaluation.

Use adversarial testing to identify vulnerabilities and potential failure modes. To identify and mitigate potential biases, use techniques like subgroup analysis and counterfactual generation. Use the insights gathered from the evaluations that were completed during the development phase to define your model monitoring strategy in production. Prepare your solution for continuous monitoring as described in the Monitor performance continuously section of this document.

Monitor for availability

To gain visibility into the health and performance of your deployed endpoints and infrastructure, use Cloud Monitoring. For your Vertex AI endpoints, track key metrics like request rate, error rate, latency, and resource utilization, and set up alerts for anomalies. For more information, see Cloud Monitoring metrics for Vertex AI.

Monitor the health of the underlying infrastructure, which can include Compute Engine instances, Google Kubernetes Engine (GKE) clusters, and TPUs and GPUs. Get automated optimization recommendations from Active Assist. If you use autoscaling, monitor the scaling behavior to ensure that autoscaling responds appropriately to changes in traffic patterns.

Track the status of model deployments, including canary releases and rollbacks, by integrating Cloud Deploy with Cloud Monitoring. In addition, monitor for potential security threats and vulnerabilities by using Security Command Center.

Set up custom alerts for business-specific thresholds

For timely identification and rectification of anomalies and issues, set up custom alerting based on thresholds that are specific to your business objectives. Examples of Google Cloud products that you can use to implement a custom alerting system include the following:

- Cloud Logging: Collect, store, and analyze logs from all components of your AI and ML system.
- Cloud Monitoring: Create custom dashboards to visualize key metrics and trends, and define custom metrics based on your needs. Configure alerts to get

notifications about critical issues, and integrate the alerts with your incident management tools like PagerDuty or Slack.

- Error Reporting: Automatically capture and analyze errors and exceptions.
- Cloud Trace: Analyze the performance of distributed systems and identify bottlenecks. Tracing is particularly useful for understanding latency between different components of your AI and ML pipeline.
- Cloud Profiler: Continuously analyze the performance of your code in production and identify performance bottlenecks in CPU or memory usage.

Build a culture of operational excellence

Shift the focus from just building models to building sustainable, reliable, and impactful AI solutions. Empower teams to continuously learn, innovate, and improve, which leads to faster development cycles, reduced errors, and increased efficiency. By prioritizing automation, standardization, and ethical considerations, you can ensure that your AI and ML initiatives consistently deliver value, mitigate risks, and promote responsible AI development.

To build a culture of operational excellence for your AI and ML systems, consider the following recommendations.

Champion automation and standardization

To emphasize efficiency and consistency, embed automation and standardized practices into every stage of the AI and ML lifecycle. Automation reduces manual errors and frees teams to focus on innovation. Standardization ensures that processes are repeatable and scalable across teams and projects.

Prioritize continuous learning and improvement

Foster an environment where ongoing education and experimentation are core principles. Encourage teams to stay up-to-date with AI and ML advancements, and provide opportunities to learn from past projects. A culture of curiosity and adaptation drives innovation and ensures that teams are equipped to meet new challenges.

Cultivate accountability and ownership

Build trust and alignment with clearly defined roles, responsibilities, and metrics for success. Empower teams to make informed decisions within these boundaries, and establish transparent ways to measure progress. A sense of ownership motivates teams and ensures collective responsibility for outcomes.

Embed AI ethics and safety considerations

Prioritize considerations for ethics in every stage of development. Encourage teams to think critically about the impact of their AI solutions, and foster discussions on fairness, bias, and societal impact. Clear principles and accountability mechanisms ensure that your AI systems align with organizational values and promote trust.

Design for scalability

To accommodate growing data volumes and user demands and to maximize the value of AI investments, your AI and ML systems need to be scalable. The systems must adapt and perform optimally to avoid performance bottlenecks that hinder effectiveness. When you design for scalability, you ensure that the AI infrastructure can handle growth and maintain responsiveness. Use scalable infrastructure, plan for capacity, and employ strategies like horizontal scaling and managed services.

To design your AI and ML systems for scalability, consider the following recommendations.

Plan for capacity and quotas

Assess future growth, and plan your infrastructure capacity and resource quotas accordingly. Work with business stakeholders to understand the projected growth and then define the infrastructure requirements accordingly.

Use Cloud Monitoring to analyze historical resource utilization, identify trends, and project future needs. Conduct regular load testing to simulate workloads and identify bottlenecks.

Familiarize yourself with Google Cloud quotas for the services that you use, such as Compute Engine, Vertex AI, and Cloud Storage. Proactively request quota increases through the Google Cloud console, and justify the increases with data from forecasting and load testing. Monitor quota usage and set up alerts to get notifications when the usage approaches the quota limits.

To optimize resource usage based on demand, rightsize your resources, use Spot VMs for fault-tolerant batch workloads, and implement autoscaling.

Prepare for peak events

Ensure that your system can handle sudden spikes in traffic or workload during peak events. Document your peak event strategy and conduct regular drills to test your system's ability to handle increased load.

To aggressively scale up resources when the demand spikes, configure autoscaling policies in Compute Engine and GKE. For predictable peak patterns, consider using predictive autoscaling. To trigger autoscaling based on application-specific signals, use custom metrics in Cloud Monitoring.

Distribute traffic across multiple application instances by using Cloud Load Balancing. Choose an appropriate load balancer type based on your application's needs. For geographically distributed users, you can use global load balancing to route traffic to the nearest available instance. For complex microservices-based architectures, consider using Cloud Service Mesh.

Cache static content at the edge of Google's network by using Cloud CDN. To cache frequently accessed data, you can use Memorystore, which offers a fully managed in-memory service for Redis, Valkey, or Memcached.

Decouple the components of your system by using Pub/Sub for real-time messaging and Cloud Tasks for asynchronous task execution

Scale applications for production

To ensure scalable serving in production, you can use managed services like Vertex AI distributed training and Vertex AI Inference. Vertex AI Inference lets you configure the machine types for your prediction nodes when you deploy a model to an endpoint or request batch predictions. For some configurations, you can add GPUs. Choose the appropriate machine type and accelerators to optimize latency, throughput, and cost.

To scale complex AI and Python applications and custom workloads across distributed computing resources, you can use Ray on Vertex AI. This feature can help optimize performance and enables seamless integration with Google Cloud services. Ray on Vertex AI simplifies distributed computing by handling cluster management, task scheduling, and data transfer. It integrates with other Vertex AI services like training, prediction, and pipelines. Ray provides fault tolerance and autoscaling, and helps you adapt the infrastructure to changing workloads. It offers a unified framework for distributed training, hyperparameter tuning, reinforcement learning, and model serving. Use Ray for distributed data preprocessing with Dataflow or Dataproc, accelerated model training, scalable hyperparameter tuning, reinforcement learning, and parallelized batch prediction.

Contributors

Authors:

- Charlotte Gistelinck, PhD | Partner Engineer
- Sannya Dang | AI Solution Architect
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist

Other contributors:

- Gary Harmson | Principal Architect
- Kumar Dhanagopal | Cross-Product Solution Developer

- Marwan Al Shawi | Partner Customer Engineer
- Ryan Cox | Principal Architect
- Stef Ruinard | Generative AI Field Solutions Architect

AI and ML perspective: Security

This document in the Well-Architected Framework: AI and ML perspective provides an overview of principles and recommendations to ensure that your AI and ML deployments meet the security and compliance requirements of your organization. The recommendations in this document align with the security pillar of the Google Cloud Well-Architected Framework.

Secure deployment of AI and ML workloads is a critical requirement, particularly in enterprise environments. To meet this requirement, you need to adopt a holistic security approach that starts from the initial conceptualization of your AI and ML solutions and extends to development, deployment, and ongoing operations. Google Cloud offers robust tools and services that are designed to help secure your AI and ML workloads.

Define clear goals and requirements

It's easier to integrate the required security and compliance controls early in your design and development process, than to add the controls after development. From the start of your design and development process, make decisions that are appropriate for your specific risk environment and your specific business priorities.

Consider the following recommendations:

- Identify potential attack vectors and adopt a security and compliance perspective from the start. As you design and evolve your AI systems, keep track of the attack surface, potential risks, and obligations that you might face.
- Align your AI and ML security efforts with your business goals and ensure that security is an integral part of your overall strategy. Understand the effects of your security choices on your main business goals.

Keep data secure and prevent loss or mishandling

Data is a valuable and sensitive asset that must be kept secure. Data security helps you to maintain user trust, support your business objectives, and meet your compliance requirements.

Consider the following recommendations:

- Don't collect, keep, or use data that's not strictly necessary for your business goals. If possible, use synthetic or fully anonymized data.

- Monitor data collection, storage, and transformation. Maintain logs for all data access and manipulation activities. The logs help you to audit data access, detect unauthorized access attempts, and prevent unwanted access.
- Implement different levels of access (for example, no-access, read-only, or write) based on user roles. Ensure that permissions are assigned based on the principle of least privilege. Users must have only the minimum permissions that are necessary to let them perform their role activities.
- Implement measures like encryption, secure perimeters, and restrictions on data movement. These measures help you to prevent data exfiltration and data loss.
- Guard against data poisoning for your ML training systems.

Keep AI pipelines secure and robust against tampering

Your AI and ML code and the code-defined pipelines are critical assets. Code that isn't secured can be tampered with, which can lead to data leaks, compliance failure, and disruption of critical business activities. Keeping your AI and ML code secure helps to ensure the integrity and value of your models and model outputs.

Consider the following recommendations:

- Use secure coding practices, such as dependency management or input validation and sanitization, during model development to prevent vulnerabilities.
- Protect your pipeline code and your model artifacts, like files, model weights, and deployment specifications, from unauthorized access. Implement different access levels for each artifact based on user roles and needs.
- Enforce lineage and tracking of your assets and pipeline runs. This enforcement helps you to meet compliance requirements and to avoid compromising production systems.

Deploy on secure systems with secure tools and artifacts

Ensure that your code and models run in a secure environment that has a robust access control system with security assurances for the tools and artifacts that are deployed in the environment.

Consider the following recommendations:

- Train and deploy your models in a secure environment that has appropriate access controls and protection against unauthorized use or manipulation.
- Follow standard Supply-chain Levels for Software Artifacts (SLSA) guidelines for your AI-specific artifacts, like models and software packages.

- Prefer using validated prebuilt container images that are specifically designed for AI workloads.

Protect and monitor inputs

AI systems need inputs to make predictions, generate content, or automate actions. Some inputs might pose risks or be used as attack vectors that must be detected and sanitized. Detecting potential malicious inputs early helps you to keep your AI systems secure and operating as intended.

Consider the following recommendations:

- Implement secure practices to develop and manage prompts for generative AI systems, and ensure that the prompts are screened for harmful intent.
- Monitor inputs to predictive or generative systems to prevent issues like overloaded endpoints or prompts that the systems aren't designed to handle.
- Ensure that only the intended users of a deployed system can use it.

Monitor, evaluate, and prepare to respond to outputs

AI systems deliver value because they produce outputs that augment, optimize, or automate human decision-making. To maintain the integrity and trustworthiness of your AI systems and applications, you need to make sure that the outputs are secure and within expected parameters. You also need a plan to respond to incidents.

Consider the following recommendations:

- Monitor the outputs of your AI and ML models in production, and identify any performance, security, and compliance issues.
- Evaluate model performance by implementing robust metrics and security measures, like identifying out-of-scope generative responses or extreme outputs in predictive models. Collect user feedback on model performance.
- Implement robust alerting and incident response procedures to address any potential issues.

Contributors

Authors:

- Kamilla Kurta | GenAI/ML Specialist Customer Engineer
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Mohamed Fawzi | Benelux Security and Compliance Lead

Other contributors:

- Daniel Lees | Cloud Security Architect
- Kumar Dhanagopal | Cross-Product Solution Developer
- Marwan Al Shawi | Partner Customer Engineer
- Wade Holmes | Global Solutions Director

AI and ML perspective: Reliability

This document in the Google Cloud Well-Architected Framework: AI and ML perspective provides an overview of the principles and recommendations to design and operate reliable AI and ML systems on Google Cloud. It explores how to integrate advanced reliability practices and observability into your architectural blueprints. The recommendations in this document align with the reliability pillar of the Google Cloud Well-Architected Framework.

In the fast-evolving AI and ML landscape, reliable systems are essential in order to ensure customer satisfaction and achieve business goals. To meet the unique demands of both predictive ML and generative AI, you need AI and ML systems that are robust, reliable, and adaptable. To handle the complexities of MLOps—from development to deployment and continuous improvement—you need to use a reliability-first approach. Google Cloud offers a purpose-built AI infrastructure that's aligned with site reliability engineering (SRE) principles and that provides a powerful foundation for reliable AI and ML systems.

The recommendations in this document are mapped to the following core principles:

- Ensure that infrastructure is scalable and highly available
- Use a modular and loosely coupled architecture
- Build an automated end-to-end MLOps platform
- Maintain trust and control through data and model governance
- Implement holistic observability and reliability practices

Ensure that ML infrastructure is scalable and highly available

Reliable AI and ML systems in the cloud require scalable and highly available infrastructure. These systems have dynamic demands, diverse resource needs, and critical dependencies on model availability. Scalable architectures adapt to fluctuating loads and variations in data volume or inference requests. High availability (HA) helps to ensure resilience against failures at the component, zone, or region level.

To build scalable and highly available ML infrastructure, consider the following recommendations.

Implement automatic and dynamic scaling capabilities

AI and ML workloads are dynamic, with demand that fluctuates based on data arrival rates, training frequency, and the volume of inference traffic. Automatic and dynamic scaling adapts infrastructure resources seamlessly to demand fluctuations. Scaling your workloads effectively helps to prevent downtime, maintain performance, and optimize costs.

To autoscale your AI and ML workloads, use the following products and features in Google Cloud:

- **Data processing pipelines**: Create data pipelines in Dataflow. Configure the pipelines to use Dataflow's horizontal autoscaling feature, which dynamically adjusts the number of worker instances based on CPU utilization, pipeline parallelism, and pending data. You can configure autoscaling parameters through pipeline options when you launch jobs.
- **Training jobs**: Automate the scaling of training jobs by using Vertex AI custom training. You can define worker pool specifications such as the machine type, the type and number of accelerators, and the number of worker pools. For jobs that can tolerate interruptions and for jobs where the training code implements checkpointing, you can reduce costs by using Spot VMs.
- **Online inference**: For online inference, use Vertex AI endpoints. To enable autoscaling, configure the minimum and maximum replica count. Specify a minimum of two replicas for HA. Vertex AI automatically adjusts the number of replicas based on traffic and the configured autoscaling metrics, such as CPU utilization and replica utilization.
- **Containerized workloads in Google Kubernetes Engine**: Configure autoscaling at the node and Pod levels. Configure the cluster autoscaler and node auto-provisioning to adjust the node count based on pending Pod resource requests like CPU, memory, GPU, and TPU. Use Horizontal Pod Autoscaler (HPA) for deployments to define scaling policies based on metrics like CPU and memory utilization. You can also scale based on custom AI and ML metrics, such as GPU or TPU utilization and prediction requests per second.
- **Serverless containerized services**: Deploy the services in Cloud Run and configure autoscaling by specifying the minimum and maximum number of container instances. Use best practices to autoscale GPU-enabled instances by specifying the accelerator type. Cloud Run automatically scales instances between the configured minimum and maximum limits based on incoming requests. When there are no requests, it scales efficiently to zero instances. You can leverage the automatic, request-driven scaling of Cloud Run to deploy Vertex

AI agents and to deploy third-party workloads like quantized models using Ollama, LLM model inference using vLLM, and Huggingface Text Generation Inference (TGI).

Design for HA and fault tolerance

For production-grade AI and ML workloads, it's crucial that you ensure continuous operation and resilience against failures. To implement HA and fault tolerance, you need to build redundancy and replication into your architecture on Google Cloud. This approach helps to ensure that a failure of an individual component doesn't cause a failure of the complete system.

- For HA and low latency in model serving, particularly for real-time inference and generative AI models, distribute your deployments across multiple locations.
- For global availability and resilience, deploy the models to multiple Vertex AI endpoints across Google Cloud regions or use the global endpoint.
- Use global load balancing to route traffic.
- For training on GKE or Compute Engine MIGs, implement monitoring for Xid errors. When you identify Xid errors, take appropriate remedial action. For example, reset GPUs, reset Compute Engine instances, or trigger hardware replacement by using the gcloud CLI report faulty host command.
- Explore fault-tolerant or elastic and resilient training solutions like recipes to use the Google Resiliency Library or integration of the Resilient training with Pathways logic for TPU workloads.

Implement redundancy for critical AI and ML components in Google Cloud. The following are examples of products and features that let you implement resource redundancy:

- Deploy GKE regional clusters across multiple zones.
- Ensure data redundancy for datasets and checkpoints by using Cloud Storage multi-regional or dual-region buckets.
- Use Spanner for globally consistent, highly available storage of metadata.
- Configure Cloud SQL read replicas for operational databases.
- Ensure that vector databases for retrieval augmented generation (RAG) are highly available and multi-zonal or multi-regional.

Manage resources proactively and anticipate requirements

Effective resource management is important to help you optimize costs, performance, and reliability. AI and ML workloads are dynamic and there's high demand for specialized hardware like GPUs and TPUs. Therefore, it's crucial that you apply proactive resource management and ensure resource availability.

Plan for capacity based on historical monitoring data, such as GPU or TPU utilization and throughput rates, from Cloud Monitoring and logs in Cloud Logging. Analyze this telemetry data by using BigQuery or Looker Studio and forecast future demand for GPUs based on growth or new models. Analysis of resource usage patterns and trends helps you to predict when and where you need critical specialized accelerators.

- Validate capacity estimates through rigorous load testing. Simulate traffic on AI and ML services like serving and pipelines by using tools like Apache JMeter or LoadView.
- Analyze system behavior under stress.
    - To anticipate and meet increased workload demands in production, proactively identify resource requirements. Monitor latency, throughput, errors, and resource utilization, especially GPU and TPU utilization. Increase resource quotas as necessary.
    - For generative AI serving, test under high concurrent loads and identify the level at which accelerator availability limits performance.
- Perform continuous monitoring for model queries and set up proactive alerts for agents.
    - Use the model observability dashboard to view metrics that are collected by Cloud Monitoring, such as model queries per second (QPS), token throughput, and first token latencies.

Optimize resource availability and obtainability

Optimize costs and ensure resource availability by strategically selecting appropriate compute resources based on workload requirements.

- For stable 24x7 inference or for training workloads with fixed or predictable capacity requirements, use committed use discounts (CUDs) for VMs and accelerators.
- For GKE nodes and Compute Engine VMs, use Spot VMs and Dynamic Workload Scheduler (DWS) capabilities:
    - For fault-tolerant tasks such as evaluation and experimentation workloads, use Spot VMs. Spot VMs can be preempted, but they can help reduce your overall costs.
    - To manage preemption risk for high-demand accelerators, you can ensure better obtainability by using DWS.

        - For complex batch training that needs high-end GPUs to run up to seven days, use the DWS Flex-Start mode.
        - For longer running workloads that run up to three months, use the Calendar mode to reserve specific GPUs (H100 and H200) and TPUs (Trillium).
- To optimize AI inference on GKE, you can run a vLLM engine that dynamically uses TPUs and GPUs to address fluctuating capacity and performance needs. For more information, see vLLM GPU/TPU Fungibility.
- For advanced scenarios with complex resource and topology needs that involve accelerators, use tools to abstract resource management.
    - Cluster Director lets you deploy and manage accelerator groups with colocation and scheduling for multi-GPU training (A3 Ultra H200 and A4 B200). Cluster Director supports GKE and Slurm clusters.
    - Ray on Vertex AI abstracts distributed computing infrastructure. It enables applications to request resources for training and serving without the need for direct management of VMs and containers.

Distribute incoming traffic across multiple instances

Effective load balancing is crucial for AI applications that have fluctuating demands. Load balancing distributes traffic, optimizes resource utilization, provides HA and low latency, and helps to ensure a seamless user experience.

- **Inference with varying resource needs**: Implement load balancing based on model metrics. GKE Inference Gateway lets you deploy models behind a load balancer with model-aware routing. The gateway prioritizes instances with GPU and TPU accelerators for compute-intensive tasks like generative AI and LLM inference. Configure detailed health checks to assess model status. Use serving frameworks like vLLM or Triton for LLM metrics and integrate the metrics into Cloud Monitoring by using Google Cloud Managed Service for Prometheus.
- **Inference workloads that need GPUs or TPUs**: To ensure that critical AI and ML inference workloads consistently run on machines that are suitable to the workloads' requirements, particularly when GPU and TPU availability is constrained, use GKE custom compute classes. You can define specific compute *profiles* with fallback policies for autoscaling. For example, you can define a profile that specifies a higher priority for reserved GPU or TPU instances. The profile can include a fallback to use cost-efficient Spot VMs if the reserved resources are temporarily unavailable.

- **Generative AI on diverse orchestration platforms**: Use a centralized load balancer. For example, for cost and management efficiency, you can route requests that have low GPU needs to Cloud Run and route more complex, GPU-intensive tasks to GKE. For inter-service communication and policy management, implement a service mesh by using Cloud Service Mesh. Ensure consistent logging and monitoring by using Cloud Logging and Cloud Monitoring.
- **Global load distribution**: To load balance traffic from global users who need low latency, use a global external Application Load Balancer. Configure geolocation routing to the closest region and implement failover. Establish regional endpoint replication in Vertex AI or GKE. Configure Cloud CDN for static assets. Monitor global traffic and latency by using Cloud Monitoring.
- **Granular traffic management**: For requests that have diverse data types or complexity and long-running requests, implement granular traffic management.
    - Configure content-based routing to direct requests to specialized backends based on attributes like URL paths and headers. For example, direct requests to GPU-enabled backends for image or video models and to CPU-optimized backends for text-based models.
    - For long-running generative AI requests or batch workloads, use WebSockets or gRPC. Implement traffic management to handle timeouts and buffering. Configure request timeouts and retries and implement rate limiting and quotas by using API Gateway or Apigee.

Use a modular and loosely coupled architecture

In a modular, loosely coupled AI and ML architecture, complex systems are divided into smaller, self-contained components that interact through well-defined interfaces. This architecture minimizes module dependencies, simplifies development and testing, enhances reproducibility, and improves fault tolerance by containing failures. The modular approach is crucial for managing complexity, accelerating innovation, and ensuring long-term maintainability.

To design a modular and loosely coupled architecture for AI and ML workloads, consider the following recommendations.

Implement small self-contained modules or components

Separate your end-to-end AI and ML system into small, self-contained modules or components. Each module or component is responsible for a specific function, such as data ingestion, feature transformation, model training, inference serving, or evaluation. A modular design provides several key benefits for AI and ML systems: improved maintainability, increased scalability, reusability, and greater flexibility and agility.

The following sections describe Google Cloud products, features, and tools that you can use to design a modular architecture for your AI and ML systems.

Containerized microservices on GKE

For complex AI and ML systems or intricate generative AI pipelines that need fine-grained orchestration, implement modules as microservices that are orchestrated by using GKE. Package each distinct stage as an individual microservice within Docker containers. These distinct stages include data ingestion that's tailored for diverse formats, specialized data preprocessing or feature engineering, distributed model training or fine tuning of large foundation models, evaluation, or serving.

Deploy the containerized microservices on GKE and leverage automated scaling based on CPU and memory utilization or custom metrics like GPU utilization, rolling updates, and reproducible configurations in YAML manifests. Ensure efficient communication between the microservices by using GKE service discovery. For asynchronous patterns, use message queues like Pub/Sub.

The microservices-on-GKE approach helps you build scalable, resilient platforms for tasks like complex RAG applications where the stages can be designed as distinct services.

Serverless event-driven services

For event-driven tasks that can benefit from serverless, automatic scaling, use Cloud Run or Cloud Run functions. These services are ideal for asynchronous tasks like preprocessing or for smaller inference jobs. Trigger Cloud Run functions on events, such as a new data file that's created in Cloud Storage or model updates in Artifact Registry. For web-hook tasks or services that need a container environment, use Cloud Run.

Cloud Run services and Cloud Run functions can scale up rapidly and scale down to zero, which helps to ensure cost efficiency for fluctuating workloads. These services are suitable for modular components in Vertex AI Agents workflows. You can orchestrate component sequences with Workflows or Application Integration.

Vertex AI managed services

Vertex AI services support modularity and help you simplify the development and deployment of your AI and ML systems. The services abstract the infrastructure complexities so that you can focus on the application logic.

- To orchestrate workflows that are built from modular steps, use Vertex AI Pipelines.
- To run custom AI and ML code, package the code in Docker containers that can run on managed services like Vertex AI custom training and Vertex AI prediction.

- For modular feature engineering pipelines, use Vertex AI Feature Store.
- For modular exploration and prototyping, use notebook environments like Vertex AI Workbench or Colab Enterprise. Organize your code into reusable functions, classes, and scripts.

Agentic applications

For AI agents, the Agent Development Kit (ADK) provides modular capabilities like Tools and State. To enable interoperability between frameworks like LangChain, LangGraph, LlamaIndex, and Vertex AI, you can combine the ADK with the Agent2Agent (A2A) protocol and the Model Context Protocol (MCP). This interoperability lets you compose agentic workflows by using diverse components.

You can deploy agents on Vertex AI Agent Engine, which is a managed runtime that's optimized for scalable agent deployment. To run containerized agents, you can leverage the autoscaling capabilities in Cloud Run.

Design well-defined interfaces

To build robust and maintainable software systems, it's crucial to ensure that the components of a system are loosely coupled and modularized. This approach offers significant advantages, because it minimizes the dependencies between different parts of the system. When modules are loosely coupled, changes in one module have minimal impact on other modules. This isolation enables independent updates and development workflows for individual modules.

The following sections provide guidance to help ensure seamless communication and integration between the modules of your AI and ML systems.

Protocol choice

- For universal access, use HTTP APIs, adhere to RESTful principles, and use JSON for language-agnostic data exchange. Design the API endpoints to represent actions on resources.
- For high-performance internal communication among microservices, use gRPC with Protocol Buffers (ProtoBuf) for efficient serialization and strict typing. Define data structures like ModelInput, PredictionResult, or ADK Tool data by using .proto files, and then generate language bindings.
- For use cases where performance is critical, leverage gRPC streaming for large datasets or for continuous flows such as live text-to-speech or video applications. Deploy the gRPC services on GKE.

Standardized and comprehensive documentation

Regardless of the interface protocol that you choose, standardized documentation is crucial. The OpenAPI Specification describes RESTful APIs. Use OpenAPI to document your AI and ML APIs: paths, methods, parameters, request-response formats that are linked to JSON schemas, and security. Comprehensive API documentation helps to improve discoverability and client integration. For API authoring and visualization, use UI tools like Swagger Editor. To accelerate development and ensure consistency, you can generate client SDKs and server stubs by using AI-assisted coding tools like Gemini Code Assist. Integrate OpenAPI documentation into your CI/CD flow.

Interaction with Google Cloud managed services like Vertex AI

Choose between the higher abstraction of the Vertex AI SDK, which is preferred for development productivity, and the granular control that the REST API provides.

- The Vertex AI SDK simplifies tasks and authentication. Use the SDK when you need to interact with Vertex AI.
- The REST API is a powerful alternative especially when interoperability is required between layers of your system. It's useful for tools in languages that don't have an SDK or when you need fine-grained control.

Use APIs to isolate modules and abstract implementation details

For security, scalability, and visibility, it's crucial that you implement robust API management for your AI and ML services. To implement API management for your defined interfaces, use the following products:

- **API Gateway**: For APIs that are externally exposed and managed, API Gateway provides a centralized, secure entry point. It simplifies access to serverless backend services, such as prediction, training, and data APIs. API Gateway helps to consolidate access points, enforce API contracts, and manage security capabilities like API keys and OAuth 2.0. To protect backends from overload and ensure reliability, implement rate limiting and usage quotas in API Gateway.
- **Cloud Endpoints**: To streamline API development and deployment on GKE and Cloud Run, use Cloud Endpoints, which offers a developer-friendly solution for generating API keys. It also provides integrated monitoring and tracing for API calls and it automates the generation of OpenAPI specs, which simplifies documentation and client integration. You can use Cloud Endpoints to manage access to internal or controlled AI and ML APIs, such as to trigger training and manage feature stores.
- **Apigee**: For enterprise-scale AI and ML, especially sophisticated generative AI APIs, Apigee provides advanced, comprehensive API management. Use Apigee for advanced security like threat protection and OAuth 2.0, for traffic management like caching, quotas, and mediation, and for analytics. Apigee can

help you to gain deep insights into API usage patterns, performance, and engagement, which are crucial for understanding generative AI API usage.

Plan for graceful degradation

In production AI and ML systems, component failures are unavoidable, just like in other systems. Graceful degradation ensures that essential functions continue to operate, potentially with reduced performance. This approach prevents complete outages and improves overall availability. Graceful degradation is critical for latency-sensitive inference, distributed training, and generative AI.

The following sections describe techniques that you use to plan and implement graceful degradation.

Fault isolation

- To isolate faulty components in distributed architectures, implement the circuit breaker pattern by using resilience libraries, such as Resilience4j in Java and CircuitBreaker in Python.
- To prevent cascading failures, configure thresholds based on AI and ML workload metrics like error rates and latency and define fallbacks like simpler models and cached data.

Component redundancy

For critical components, implement redundancy and automatic failover. For example, use GKE multi-zone clusters or regional clusters and deploy Cloud Run services redundantly across different regions. To route traffic to healthy instances when unhealthy instances are detected, use Cloud Load Balancing.

Ensure data redundancy by using Cloud Storage multi-regional buckets. For distributed training, implement asynchronous checkpointing to resume after failures. For resilient and elastic training, use Pathways.

Proactive monitoring

Graceful degradation helps to ensure system availability during failure, but you must also implement proactive measures for continuous health checks and comprehensive monitoring. Collect metrics that are specific to AI and ML, such as latency, throughput, and GPU utilization. Also, collect model performance degradation metrics like model and data drift by using Cloud Monitoring and Vertex AI Model Monitoring.

Health checks can trigger the need to replace faulty nodes, deploy more capacity, or automatically trigger continuous retraining or fine-tuning of pipelines that use updated data. This proactive approach helps to prevent both accuracy-based degradation and system-level graceful degradation and it helps to enhance overall reliability.

SRE practices

To monitor the health of your systems, consider adopting SRE practices to implement service level objectives (SLOs). Alerts on error budget loss and burn rate can be early indicators of reliability problems with the system. For more information about SRE practices, see the Google SRE book.

Build an automated end-to-end MLOps platform

A robust, scalable, and reliable AI and ML system on Google Cloud requires an automated end-to-end MLOps platform for the model development lifecycle. The development lifecycle includes initial data handling, continuous model training, deployment, and monitoring in production. By automating these stages on Google Cloud, you establish repeatable processes, reduce manual toil, minimize errors, and accelerate the pace of innovation.

An automated MLOps platform is essential for establishing production-grade reliability for your applications. Automation helps to ensure model quality, guarantee reproducibility, and enable continuous integration and delivery of AI and ML artifacts.

To build an automated end-to-end MLOps platform, consider the following recommendations.

Automate the model development lifecycle

A core element of an automated MLOps platform is the orchestration of the entire AI and ML workflow as a series of connected, automated steps: from data preparation and validation to model training, evaluation, deployment, and monitoring.

- Use Vertex AI Pipelines as your central orchestrator:
    - Define end-to-end workflows with modular components for data processing, training, evaluation, and deployment.
    - Automate pipeline runs by using schedules or triggers like new data or code changes.
    - Implement automated parameterization and versioning for each pipeline run and create a version history.
    - Monitor pipeline progress and resource usage by using built-in logging and tracing, and integrate with Cloud Monitoring alerts.
- Define your ML pipelines programmatically by using the Kubeflow Pipelines (KFP) SDK or TensorFlow Extended SDK. For more information, see Interfaces for Vertex AI Pipelines.

- Orchestrate operations by using Google Cloud services like Dataflow, Vertex AI custom training, Vertex AI Model Registry, and Vertex AI endpoints.
- For generative AI workflows, orchestrate the steps for prompt management, batched inference, human-in-the-loop (HITL) evaluation, and coordinating ADK components.

Manage infrastructure as code

Infrastructure as code (IaC) is crucial for managing AI and ML system infrastructure and for enabling reproducible, scalable, and maintainable deployments. The infrastructure needs of AI and ML systems are dynamic and complex. The systems often require specialized hardware like GPUs and TPUs. IaC helps to mitigate the risks of manual infrastructure management by ensuring consistency, enabling rollbacks, and making deployments repeatable.

To effectively manage your infrastructure resources as code, use the following techniques.

Automate resource provisioning

To effectively manage IaC on Google Cloud, define and provision your AI and ML infrastructure resources by using Terraform. The infrastructure might include resources such as the following:

- GKE clusters that are configured with node pools. The node pools can be optimized based on workload requirements. For example, you can use A100, H100, H200, or B200 GPUs for training, and use L4 GPUs for inference.
- Vertex AI endpoints that are configured for model serving, with defined machine types and scaling policies.
- Cloud Storage buckets for data and artifacts.

Use configuration templates

Organize your Terraform configurations as modular templates. To accelerate the provisioning of AI and ML resources, you can use Cluster Toolkit. The toolkit provides example blueprints, which are Google-curated Terraform templates that you can use to deploy ready-to-use HPC, AI, and ML clusters in Slurm or GKE. You can customize the Terraform code and manage it in your version control system. To automate the resource provisioning and update workflow, you can integrate the code into your CI/CD pipelines by using Cloud Build.

Automate configuration changes

After you provision your infrastructure, manage the ongoing configuration changes declaratively:

- In Kubernetes-centric environments, manage your Google Cloud resources as Kubernetes objects by using Config Connector.
- Define and manage Vertex AI resources like datasets, models, and endpoints, Cloud SQL instances, Pub/Sub topics, and Cloud Storage buckets by using YAML manifests.
- Deploy the manifests to your GKE cluster in order to integrate the application and infrastructure configuration.
- Automate configuration updates by using CI/CD pipelines and use templating to handle environment differences.
- Implement configurations for Identity and Access Management (IAM) policies and service accounts by using IaC.

Integrate with CI/CD

- Automate the lifecycle of the Google Cloud infrastructure resources by integrating IaC into CI/CD pipelines by using tools like Cloud Build and Infrastructure Manager.
- Define triggers for automatic updates on code commits.
- Implement automated testing and validation within the pipeline. For example, you can create a script to automatically run the Terraform validate and plan commands.
- Store the configurations as artifacts and enable versioning.
- Define separate environments, such as dev, staging, and prod, with distinct configurations in version control and automate environment promotion.

Validate model behavior

To maintain model accuracy and relevance over time, automate the training and evaluation process within your MLOps platform. This automation, coupled with rigorous validation, helps to ensure that the models behave as expected with relevant data before they're deployed to production.

- Set up continuous training pipelines, which are either triggered by new data and monitoring signals like data drift or that run on a schedule.
    - To manage automated training jobs, such as hyperparameter tuning trials and distributed training configurations for larger models, use Vertex AI custom training.
    - For fine-tuning foundation models, automate the fine-tuning process and integrate the jobs into your pipelines.

- Implement automated model versioning and securely store trained model artifacts after each successful training run. You can store the artifacts in Cloud Storage or register them in Model Registry.
- Define evaluation metrics and set clear thresholds, such as minimum accuracy, maximum error rate, and minimum F1 score.
    - Ensure that a model meets the thresholds to automatically pass the evaluation and be considered for deployment.
    - Automate evaluation by using services like model evaluation in Vertex AI.
    - Ensure that the evaluation includes metrics that are specific to the quality of generated output, factual accuracy, safety attributes, and adherence to specified style or format.
- To automatically log and track the parameters, code versions, dataset versions, and results of each training and evaluation run, use Vertex AI Experiments. This approach provides a history that's useful for comparison, debugging, and reproducibility.
- To optimize hyperparameter tuning and automate searching for optimal model configurations based on your defined objective, use Vertex AI Vizier.
- To visualize training metrics and to debug during development, use Vertex AI TensorBoard.

Validate inputs and outputs of AI and ML pipelines

To ensure the reliability and integrity of your AI and ML systems, you must validate data when it enters the systems and moves through the pipelines. You must also verify the inputs and outputs at the component boundaries. Robust validation of all inputs and outputs—raw data, processed data, configurations, arguments, and files—helps to prevent unexpected behavior and maintain model quality throughout the MLOps lifecycle. When you integrate this proactive approach into your MLOps platform, it helps detect errors before they are propagated throughout a system and it saves time and resources.

To effectively validate the inputs and outputs of your AI and ML pipelines, use the following techniques.

Automate data validation

- Implement automated data validation in your data ingestion and preprocessing pipelines by using TensorFlow Data Validation (TFDV).
    - For large-scale, SQL-based data quality checks, leverage scalable processing services like BigQuery.

    - For complex, programmatic validation on streaming or batch data, use Dataflow.
- Monitor data distributions over time with TFDV capabilities.
    - Visualize trends by using tools that are integrated with Cloud Monitoring to detect data drift. You can automatically trigger model retraining pipelines when data patterns change significantly.
- Store validation results and metrics in BigQuery for analysis and historical tracking and archive validation artifacts in Cloud Storage.

Validate pipeline configurations and input data

To prevent pipeline failures or unexpected behavior caused by incorrect settings, implement strict validation for all pipeline configurations and command-line arguments:

- Define clear schemas for your configuration files like YAML or JSON by using schema validation libraries like jsonschema for Python. Validate configuration objects against these schemas before a pipeline run starts and before a component executes.
- Implement input validation for all command-line arguments and pipeline parameters by using argument-parsing libraries like argparse. Validation should check for correct data types, valid values, and required arguments.
- Within Vertex AI Pipelines, define the expected types and properties of component parameters by using the built-in component input validation features.
- To ensure reproducibility of pipeline runs and to maintain an audit trail, store validated, versioned configuration files in Cloud Storage or Artifact Registry.

Validate input and output files

Validate input and output files such as datasets, model artifacts, and evaluation reports for integrity and format correctness:

- Validate file formats like CSV, Parquet, and image types by using libraries.
- For large files or critical artifacts, validate file sizes and checksums to detect corruption or incomplete transfers by using Cloud Storage data validation and change detection.
- Perform file validation by using Cloud Run functions (for example, based on file upload events) or within Dataflow pipelines.
- Store validation results in BigQuery for easier retrieval and analysis.

Automate deployment and implement continuous monitoring

Automated deployment and continuous monitoring of models in production helps to ensure reliability, perform rapid updates, and detect issues promptly. This involves managing model versions, controlled deployment, automated deployment using CI/CD, and comprehensive monitoring as described in the following sections.

Manage model versions

Manage model iterations and associated artifacts by using versioning tools:

- To track model versions and metadata and to link to underlying model artifacts, use Model Registry.
- Implement a clear versioning scheme (such as, semantic versioning). For each model version, attach comprehensive metadata such as training parameters, evaluation metrics from validation pipelines, and dataset version.
- Store model artifacts such as model files, pretrained weights, and serving container images in Artifact Registry and use its versioning and tagging features.
- To meet security and governance requirements, define stringent access-control policies for Model Registry and Artifact Registry.
- To programmatically register and manage versions and to integrate versions into automated CI/CD pipelines, use the Vertex AI SDK or API.

Perform controlled deployment

Control the deployment of model versions to endpoints by using your serving platform's traffic management capabilities.

- Implement a rolling deployment by using the traffic splitting feature of Vertex AI endpoints.
- If you deploy your model to GKE, use advanced traffic management techniques like canary deployment:
    1. Route a small subset of the production traffic to a new model version.
    2. Continuously monitor performance and error rates through metrics.
    3. Establish that the model is reliable.
    4. Roll out the version to all traffic.
- Perform A/B testing of AI agents:
    1. Deploy two different model-agent versions or entirely different models to the same endpoint.

    2. Split traffic across the deployments.
    3. Analyze the results against business objectives.

- Implement automated rollback mechanisms that can quickly revert endpoint traffic to a previous stable model version if monitoring alerts are triggered or performance thresholds are missed.
- Configure traffic splitting and deployment settings programmatically by using the Vertex AI SDK or API.
- Use Cloud Monitoring to track performance and traffic across versions.
- Automate deployment with CI/CD pipelines. You can use Cloud Build to build containers, version artifacts, and trigger deployment to Vertex AI endpoints.
- Ensure that the CI/CD pipelines manage versions and pull from Artifact Registry.
- Before you shift traffic, perform automated endpoint testing for prediction correctness, latency, throughput, and API function.
- Store all configurations in version control.

Monitor continuously

- Use Model Monitoring to automatically detect performance degradation, data drift (changes in input distribution compared to training), and prediction drift (changes in model outputs).
    - Configure drift detection jobs with thresholds and alerts.
    - Monitor real-time performance: prediction latency, throughput, error rates.
- Define custom metrics in Cloud Monitoring for business KPIs.
- Integrate Model Monitoring results and custom metrics with Cloud Monitoring for alerts and dashboards.
- Configure notification channels like email, Slack, or PagerDuty and configure automated remediation.
- To debug prediction logs, use Cloud Logging.
- Integrate monitoring with incident management.

For generative AI endpoints, monitor output characteristics like toxicity and coherence:

- Monitor feature serving for drift.

- Implement granular prediction validation: validate outputs against expected ranges and formats by using custom logic.
- Monitor prediction distributions for shifts.
- Validate output schema.
- Configure alerts for unexpected outputs and shifts.
- Track and respond to real-time validation events by using Pub/Sub.

Ensure that the output of comprehensive monitoring feeds back into continuous training.

Maintain trust and control through data and model governance

AI and ML reliability extends beyond technical uptime. It includes trust and robust data and model governance. AI outputs might be inaccurate, biased, or outdated. Such issues erode trust and can cause harm. Comprehensive traceability, strong access control, automated validation, and transparent practices help to ensure that AI outputs are reliable, trustworthy, and meet ethics standards.

To maintain trust and control through data and model governance, consider the following recommendations.

Establish data and model catalogs for traceability

To facilitate comprehensive tracing, auditing, and understanding the lineage of your AI and ML assets, maintain a robust, centralized record of data and model versions throughout their lifecycle. A reliable data and model catalog serves as the single source of truth for all of the artifacts that are used and produced by your AI and ML pipelines–from raw data sources and processed datasets to trained model versions and deployed endpoints.

Use the following products, tools, and techniques to create and maintain catalogs for your data assets:

- Build an enterprise-wide catalog of your data assets by using Dataplex Universal Catalog. To automatically discover and build inventories of the data assets, integrate Dataplex Universal Catalog with your storage systems, such as BigQuery, Cloud Storage, and Pub/Sub.
- Ensure that your data is highly available and durable by storing it in Cloud Storage multi-region or dual-region buckets. Data that you upload to these buckets is stored redundantly across at least two separate geographic locations. This redundancy provides built-in resilience against regional outages and it helps to ensure data integrity.

- Tag and annotate your datasets with relevant business metadata, ownership information, sensitivity levels, and lineage details. For example, link a processed dataset to its raw source and to the pipeline that created the dataset.
- Create a central repository for model versions by using Model Registry. Register each trained model version and link it to the associated metadata. The metadata can include the following:
    - Training parameters.
    - Evaluation metrics from validation pipelines.
    - Dataset version that was used for training, with lineage traced back to the relevant Dataplex Universal Catalog entry.
    - Code version that produced the dataset.
    - Details about the framework or foundation model that was used.
- Before you import a model into Model Registry, store model artifacts like model files and pretrained weights in a service like Cloud Storage. Store custom container images for serving or custom training jobs in a secure repository like Artifact Registry.
- To ensure that data and model assets are automatically registered and updated in the respective catalogs upon creation or modification, implement automated processes within your MLOps pipelines. This comprehensive cataloging provides end-to-end traceability from raw data to prediction, which lets you audit the inputs and processes that led to a specific model version or prediction. The auditing capability is vital for debugging unexpected behavior, ensuring compliance with data usage policies, and understanding the impact of data or model changes over time.
- For Generative AI and foundation models, your catalog must also track details about the specific foundation model used, fine-tuning parameters, and evaluation results that are specific to the quality and safety of the generated output.

Implement robust access controls and audit trails

To maintain trust and control in your AI and ML systems, it's essential that you protect sensitive data and models from unauthorized access and ensure accountability for all changes.

- Implement strict access controls and maintain detailed audit trails across all components of your AI and ML systems in Google Cloud.

- Define granular permissions in IAM for users, groups, and service accounts that interact with your AI and ML resources.
- Follow the principle of least privilege rigorously.
- Grant only the minimum necessary permissions for specific tasks. For example, a training service account needs read access to training data and write access for model artifacts, but the service might not need write access to production serving endpoints.

Apply IAM policies consistently across all relevant assets and resources in your AI and ML systems, including the following:

- Cloud Storage buckets that contain sensitive data or model artifacts.
- BigQuery datasets.
- Vertex AI resources, such as model repositories, endpoints, pipelines, and Feature Store resources.
- Compute resources, such as GKE clusters and Cloud Run services.

Use auditing and logs to capture, monitor, and analyze access activity:

- Enable Cloud Audit Logs for all of the Google Cloud services that are used by your AI and ML system.
- Configure audit logs to capture detailed information about API calls, data access events, and configuration changes made to your resources. Monitor the logs for suspicious activity, unauthorized access attempts, or unexpected modifications to critical data or model assets.
- For real-time analysis, alerting, and visualization, stream the audit logs to Cloud Logging.
- For cost-effective long-term storage and retrospective security analysis or compliance audits, export the logs to BigQuery.
- For centralized security monitoring, integrate audit logs with your security information and event management (SIEM) systems. Regularly review access policies and audit trails to ensure they align with your governance requirements and detect potential policy violations.
- For applications that handle sensitive data, such as personally identifiable information (PII) for training or inference, use Sensitive Data Protection checks within pipelines or on data storage.
- For generative AI and agentic solutions, use audit trails to help track who accessed specific models or tools, what data was used for fine-tuning or

prompting, and what queries were sent to production endpoints. The audit trails help you to ensure accountability and they provide crucial data for you to investigate misuse of data or policy violations.

Address bias, transparency, and explainability

To build trustworthy AI and ML systems, you need to address potential biases that are inherent in data and models, strive for transparency in system behavior, and provide explainability for model outputs. It's especially crucial to build trustworthy systems in sensitive domains or when you use complex models like those that are typically used for generative AI applications.

- Implement proactive practices to identify and mitigate bias throughout the MLOps lifecycle.
- Analyze training data for bias by using tools that detect skew in feature distributions across different demographic groups or sensitive attributes.
- Evaluate the overall model performance and the performance across predefined slices of data. Such evaluation helps you to identify disparate performance or bias that affects specific subgroups.

For model transparency and explainability, use tools that help users and developers understand why a model made a particular prediction or produced a specific output.

- For tabular models that are deployed on Vertex AI endpoints, generate feature attributions by using Vertex Explainable AI. Feature attributions indicate the input features that contributed most to the prediction.
- Interactively explore model behavior and potential biases on a dataset by using model-agnostic tools like the What-If Tool, which integrates with TensorBoard.
- Integrate explainability into your monitoring dashboards. In situations where understanding the model's reasoning is important for trust or decision-making, provide explainability data directly to end users through your application interfaces.
- For complex models like LLMs that are used for generative AI models, explain the *process* that an agent followed, such as by using trace logs. Explainability is relatively challenging for such models, but it's still vital.
- In RAG applications, provide citations for retrieved information. You can also use techniques like prompt engineering to guide the model to provide explanations or show its reasoning steps.
- Detect shifts in model behavior or outputs that might indicate emerging bias or unfairness by implementing continuous monitoring in production. Document

model limitations, intended use cases, and known potential biases as part of the model's metadata in the Model Registry.

Implement holistic AI and ML observability and reliability practices

Holistic observability is essential for managing complex AI and ML systems in production. It's also essential for measuring the reliability of complex AI and ML systems, especially for generative AI, due to its complexity, resource intensity, and potential for unpredictable outputs. Holistic observability involves observing infrastructure, application code, data, and model behavior to gain insights for proactive issue detection, diagnosis, and response. This observability ultimately leads to high-performance, reliable systems. To achieve holistic observability you need to do the following:

- Adopt SRE principles.
- Define clear reliability goals.
- Track metrics across system layers.
- Use insights from observability for continuous improvement and proactive management.

To implement holistic observability and reliability practices for AI and ML workloads in Google Cloud, consider the following recommendations.

Establish reliability goals and business metrics

Identify the key performance indicators (KPIs) that your AI and ML system directly affects. The KPIs might include revenue that's influenced by AI recommendations, customer churn that the AI systems predicted or mitigated, and user engagement and conversion rates that are driven by generative AI features.

For each KPI, define the corresponding technical reliability metrics that affect the KPI. For example, if the KPI is "customer satisfaction with a conversational AI assistant," then the corresponding reliability metrics can include the following:

- The success rate of user requests.
- The latency of responses: time to first token (TTFT) and token streaming for LLMs.
- The rate of irrelevant or harmful responses.
- The rate of successful task completion by the agent.

For AI and ML training, reliability metrics can include model FLOPS utilization (MFU), iterations per second, tokens per second, and tokens per device.

To effectively measure and improve AI and ML reliability, begin by setting clear reliability goals that are aligned with the overarching business objectives. Adopt the SRE approach by defining SLOs that quantify acceptable levels of reliability and performance for your AI and ML services from the users' perspective. Quantify these technical reliability metrics with specific SLO targets.

The following are examples of SLO targets:

- 99.9% of API calls must return a successful response.
- 95th percentile inference latency must be below 300 ms.
- TTFT must be below 500 ms for 99% of requests.
- Rate of harmful output must be below 0.1%.

Aligning SLOs directly with business needs ensures that reliability efforts are focused on the most critical system behavior that affects users and the business. This approach helps to transform reliability into a measurable and actionable engineering property.

Monitor infrastructure and application performance

Track infrastructure metrics across all of the resources that are used by your AI and ML systems. The metrics include processor usage (CPU, GPU, and TPU), memory usage, network throughput and latency, and disk I/O. Track the metrics for managed environments like Vertex AI training and serving and for self-managed resources like GKE nodes and Cloud Run instances.

Monitor the four golden signals for your AI and ML applications:

- **Latency**: Time to respond to requests.
- **Traffic**: Volume of requests or workload.
- **Error rate**: Rate of failed requests or operations.
- **Saturation**: Utilization of critical resources like CPU, memory, and GPU or TPU accelerators, which indicates how close your system is to capacity limits.

Perform monitoring by using the following techniques:

- Collect, store, and visualize the infrastructure and application metrics by using Cloud Monitoring. You can use pre-built dashboards for Google Cloud services and create custom dashboards that are tailored based on your workload's specific performance indicators and infrastructure health.
    - Collect and integrate metrics from specialized serving frameworks like vLLM or NVIDIA Triton Inference Server into Cloud Monitoring by using Google Cloud Managed Service for Prometheus.

    - Create dashboards and configure alerts for metrics that are related to custom training, endpoints, and performance, and for metrics that Vertex AI exports to Cloud Monitoring.

- Collect detailed logs from your AI and ML applications and the underlying infrastructure by using Cloud Logging. These logs are essential for troubleshooting and performance analysis. They provide context around events and errors.
- Pinpoint latency issues and understand request flows across distributed AI and ML microservices by using Cloud Trace. This capability is crucial for debugging complex Vertex AI Agents interactions or multi-component inference pipelines.
- Identify performance bottlenecks within function blocks in application code by using Cloud Profiler. Identifying performance bottlenecks can help you optimize resource usage and execution time.
- Gather specific accelerator-related metrics like detailed GPU utilization per process, memory usage per process, and temperature, by using tools like NVIDIA Data Center GPU Manager (DCGM).

Implement data and model observability

Reliable generative AI systems require robust data and model observability, which starts with end-to-end pipeline monitoring.

- Track data ingestion rates, processed volumes, and transformation latencies by using services like Dataflow.
- Monitor job success and failure rates within your MLOps pipelines, including pipelines that are managed by Vertex AI Pipelines.

Continuous assessment of data quality is crucial.

- Manage and govern data by using Dataplex Universal Catalog:
    - Evaluate accuracy by validating against ground truth or by tracking outlier detection rates.
    - Monitor freshness based on the age of data and frequency of updates against SLAs.
    - Assess completeness by tracking null-value percentages and required field-fill rates.
    - Ensure validity and consistency through checks for schema-adherence and duplication.

- Proactively detect anomalies by using Cloud Monitoring alerting and through clear data lineage for traceability.
- For RAG systems, examine the relevance of the retrieved context and the groundedness (attribution to source) of the responses.
- Monitor the throughput of vector database queries.

Key model observability metrics include input-output token counts and model-specific error rates, such as hallucination or query resolution failures. To track these metrics, use Model Monitoring.

- Continuously monitor the toxicity scores of the output and user-feedback ratings.
- Automate the assessment of model outputs against defined criteria by using the Gen AI evaluation service.
- Ensure sustained performance by systematically monitoring for data and concept drift with comprehensive error-rate metrics.

To track model metrics, you can use TensorBoard or MLflow. For deep analysis and profiling to troubleshoot performance issues, you can use PyTorch XLA profiling or NVIDIA Nsight.

Contributors

Authors:

- Rick (Rugui) Chen | AI Infrastructure Field Solutions Architect
- Stef Ruinard | Generative AI Field Solutions Architect

Other contributors:

- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Hossein Sarshar | AI Infrastructure Field Solution Architect
- Jose Andrade | Customer Engineer, SRE Specialist
- Kumar Dhanagopal | Cross-Product Solution Developer
- Laura Hyatt | Customer Engineer, FSI
- Olivier Martin | AI Infrastructure Field Solution Architect
- Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework

AI and ML perspective: Cost optimization

This document in Well-Architected Framework: AI and ML perspective provides an overview of principles and recommendations to optimize the cost of your AI systems throughout the ML lifecycle. By adopting a proactive and informed cost management approach, your organization can realize the full potential of AI and ML systems and also maintain financial discipline. The recommendations in this document align with the cost optimization pillar of the Google Cloud Well-Architected Framework.

AI and ML systems can help you unlock valuable insights and predictive capabilities from data. For example, you can reduce friction in internal processes, improve user experiences, and gain deeper customer insights. The cloud offers vast amounts of resources and quick time-to-value without large up-front investments for AI and ML workloads. To maximize business value and to align the spending with your business goals, you need to understand the cost drivers, proactively optimize costs, set up spending controls, and adopt FinOps practices.

The recommendations in this document are mapped to the following core principles:

- Define and measure costs and returns
- Optimize resource allocation
- Enforce data management and governance practices
- Automate and streamline with MLOps
- Use managed services and pre-trained models

Define and measure costs and returns

To effectively manage AI and ML costs in Google Cloud, you must define and measure the cloud resource costs and the business value of your AI and ML initiatives. To help you track expenses granularly, Google Cloud provides comprehensive billing and cost management tools, such as the following:

- Cloud Billing reports and tables
- Looker Studio dashboards, budgets, and alerts
- Cloud Monitoring
- Cloud Logging

To make informed decisions about resource allocation and optimization, consider the following recommendations.

Establish business goals and KPIs

Align the technical choices in your AI and ML projects with business goals and key performance indicators (KPIs).

Define strategic objectives and ROI-focused KPIs

Ensure that AI and ML projects are aligned with strategic objectives like revenue growth, cost reduction, customer satisfaction, and efficiency. Engage stakeholders to understand the business priorities. Define AI and ML objectives that are specific, measurable, attainable, relevant, and time-bound (SMART). For example, a SMART objective is: "Reduce chat handling time for customer support by 15% in 6 months by using an AI chatbot".

To make progress towards your business goals and to measure the return on investment (ROI), define KPIs for the following categories of metrics:

- Costs for training, inference, storage, and network resources, including specific unit costs (such as the cost per inference, data point, or task). These metrics help you gain insights into efficiency and cost optimization opportunities. You can track these costs by using Cloud Billing reports and Cloud Monitoring dashboards.
- Business value metrics like revenue growth, cost savings, customer satisfaction, efficiency, accuracy, and adoption. You can track these metrics by using BigQuery analytics and Looker dashboards.
- Industry-specific metrics like the following:
    - Retail industry: measure revenue lift and churn
    - Healthcare industry: measure patient time and patient outcomes
    - Finance industry: measure fraud reduction
- Project-specific metrics. You can track these metrics by using Vertex AI Experiments and evaluation.
    - Predictive AI: measure accuracy and precision
    - Generative AI: measure adoption, satisfaction, and content quality
    - Computer vision AI: measure accuracy

Foster a culture of cost awareness and continuous optimization

Adopt FinOps principles to ensure that each AI and ML project has estimated costs and has ways to measure and track actual costs throughout its lifecycle. Ensure that the costs and business benefits of your projects have assigned owners and clear accountability.

For more information, see Foster a culture of cost awareness in the Cost Optimization pillar of the Google Cloud Well-Architected Framework.

Drive value and continuous optimization through iteration and feedback

Map your AI and ML applications directly to your business goals and measure the ROI.

To validate your ROI hypotheses, start with pilot projects and use the following iterative optimization cycle:

1. **Monitor continuously and analyze data**: Monitor KPIs and costs to identify deviations and opportunities for optimization.
2. **Make data-driven adjustments**: Optimize strategies, models, infrastructure, and resource allocation based on data insights.
3. **Refine iteratively**: Adapt business objectives and KPIs based on the things you learned and the evolving business needs. This iteration helps you maintain relevance and strategic alignment.
4. **Establish a feedback loop**: Review performance, costs, and value with stakeholders to inform ongoing optimization and future project planning.

Manage billing data with Cloud Billing and labels

Effective cost optimization requires visibility into the source of each cost element. The recommendations in this section can help you use Google Cloud tools to get granular insights into your AI and ML costs. You can also attribute costs to specific AI and ML projects, teams, and activities. These insights lay the groundwork for cost optimization.

Organize and label Google Cloud resources

- Structure your projects and resources in a hierarchy that reflects your organizational structure and your AI and ML workflows. To track and analyze costs at different levels, organize your Google Cloud resources by using organizations, folders, and projects. For more information, see Decide a resource hierarchy for your Google Cloud landing zone.
- Apply meaningful labels to your resources. You can use labels that indicate the project, team, environment, model name, dataset, use case, and performance requirements. Labels provide valuable context for your billing data and enable granular cost analysis.
- Maintain consistency in your labeling conventions across all of your AI and ML projects. Consistent labeling conventions ensure that your billing data is organized and can be readily analyzed.

Use billing-related tools

- To facilitate detailed analysis and reporting, export the billing data to BigQuery. BigQuery has powerful query capabilities that let you analyze the billing data to help you understand your costs.
- To aggregate costs by labels, projects, or specific time periods, you can write custom SQL queries in BigQuery. Such queries let you attribute costs to specific AI and ML activities, such as model training, hyperparameter tuning, or inference.
- To identify cost anomalies or unexpected spending spikes, use the analytic capabilities in BigQuery. This approach can help you detect potential issues or inefficiencies in your AI and ML workloads.
- To identify and manage unexpected costs, use the anomaly detection dashboard in Cloud Billing.
- To distribute costs across different teams or departments based on resource usage, use Google Cloud's cost allocation feature. Cost allocation promotes accountability and transparency.
- To gain insights into spending patterns, explore the prebuilt Cloud Billing reports. You can filter and customize these reports to focus on specific AI and ML projects or services.

Monitor resources continuously with dashboards, alerts, and reports

To create a scalable and resilient way to track costs, you need continuous monitoring and reporting. Dashboards, alerts, and reports constitute the foundation for effective cost tracking. This foundation lets you maintain constant access to cost information, identify areas of optimization, and ensure alignment between business goals and costs.

Create a reporting system

Create scheduled reports and share them with appropriate stakeholders.

Use Cloud Monitoring to collect metrics from various sources, including your applications, infrastructure, and Google Cloud services like Compute Engine, Google Kubernetes Engine (GKE), and Cloud Run functions. To visualize metrics and logs in real time, you can use the prebuilt Cloud Monitoring dashboard or create custom dashboards. Custom dashboards let you define and add metrics to track specific aspects of your systems, like model performance, API calls, or business-level KPIs.

Use Cloud Logging for centralized collection and storage of logs from your applications, systems, and Google Cloud services. Use the logs for the following purposes:

- Track costs and utilization of resources like CPU, memory, storage, and network.

- Identify cases of over-provisioning (where resources aren't fully utilized) and under-provisioning (where there are insufficient resources). Over-provisioning results in unnecessary costs. Under-provisioning slows training times and might cause performance issues.
- Identify idle or underutilized resources, such as VMs and GPUs, and take steps to shut down or rightsize them to optimize costs.
- Identify cost spikes to detect sudden and unexpected increases in resource usage or costs.

Use Looker or Looker Studio to create interactive dashboards and reports. Connect the dashboards and reports to various data sources, including BigQuery and Cloud Monitoring.

Set alert thresholds based on key KPIs

For your KPIs, determine the thresholds that should trigger alerts. Meaningful alert thresholds can help you avoid alert fatigue. Create alerting policies in Cloud Monitoring to get notifications related to your KPIs. For example, you can get notifications when accuracy drops below a certain threshold or latency exceeds a defined limit. Alerts based on log data can notify you about potential cost issues in real time. Such alerts let you take corrective actions promptly and prevent further financial loss.

Optimize resource allocation

To achieve cost efficiency for your AI and ML workloads in Google Cloud, you must optimize resource allocation. To help you avoid unnecessary expenses and ensure that your workloads have the resources that they need to perform optimally, align resource allocation with the needs of your workloads.

To optimize the allocation of cloud resources to AI and ML workloads, consider the following recommendations.

Use autoscaling to dynamically adjust resources

Use Google Cloud services that support autoscaling, which automatically adjusts resource allocation to match the current demand. Autoscaling provides the following benefits:

- **Cost and performance optimization**: You avoid paying for idle resources. At the same time, autoscaling ensures that your systems have the necessary resources to perform optimally, even at peak load.
- **Improved efficiency**: You free up your team to focus on other tasks.
- **Increased agility**: You can respond quickly to changing demands and maintain high availability for your applications.

The following table summarizes the techniques that you can use to implement autoscaling for different stages of your AI projects.

Stage	Autoscaling techniques
Training	• Use managed services like Vertex AI or GKE, which offer built-in autoscaling capabilities for training jobs. • Configure autoscaling policies to scale the number of training instances based on metrics like CPU utilization, memory usage, and job queue length. • Use custom scaling metrics to fine-tune autoscaling behavior for your specific workloads.
Inference	• Deploy your models on scalable platforms like Vertex AI Inference, GPUs on GKE, or TPUs on GKE. • Use autoscaling features to adjust the number of replicas based on metrics like request rate, latency, and resource utilization. • Implement load balancing to distribute traffic evenly across replicas and ensure high availability.

Start with small models and datasets

To help reduce costs, test ML hypotheses at a small scale when possible and use an iterative approach. This approach, with smaller models and datasets, provides the following benefits:

- **Reduced costs from the start**: Less compute power, storage, and processing time can result in lower costs during the initial experimentation and development phases.
- **Faster iteration**: Less training time is required, which lets you iterate faster, explore alternative approaches, and identify promising directions more efficiently.
- **Reduced complexity**: Simpler debugging, analysis, and interpretation of results, which leads to faster development cycles.
- **Efficient resource utilization**: Reduced chance of over-provisioning resources. You provision only the resources that are necessary for the current workload.

Consider the following recommendations:

- **Use sample data first**: Train your models on a representative subset of your data. This approach lets you assess the model's performance and identify potential issues without processing the entire dataset.
- **Experiment by using notebooks**: Start with smaller instances and scale as needed. You can use Vertex AI Workbench, a managed Jupyter notebook environment that's well suited for experimentation with different model architectures and datasets.
- **Start with simpler or pre-trained models**: Use Vertex AI Model Garden to discover and explore the pre-trained models. Such models require fewer computational resources. Gradually increase the complexity as needed based on performance requirements.
    - Use pre-trained models for tasks like image classification and natural language processing. To save on training costs, you can fine-tune the models on smaller datasets initially.
    - Use BigQuery ML for structured data. BigQuery ML lets you create and deploy models directly within BigQuery. This approach can be cost-effective for initial experimentation, because you can take advantage of the pay-per-query pricing model for BigQuery.
- **Scale for resource optimization**: Use Google Cloud's flexible infrastructure to scale resources as needed. Start with smaller instances and adjust their size or number when necessary.

Discover resource requirements through experimentation

Resource requirements for AI and ML workloads can vary significantly. To optimize resource allocation and costs, you must understand the specific needs of your workloads through systematic experimentation. To identify the most efficient configuration for your models, test different configurations and analyze their performance. Then, based on the requirements, right-size the resources that you used for training and serving.

We recommend the following approach for experimentation:

1. **Start with a baseline**: Begin with a baseline configuration based on your initial estimates of the workload requirements. To create a baseline, you can use the cost estimator for new workloads or use an existing billing report. For more information, see Unlock the true cost of enterprise AI on Google Cloud.
2. **Understand your quotas**: Before launching extensive experiments, familiarize yourself with your Google Cloud project quotas for the resources and APIs that you plan to use. The quotas determine the range of configurations that you can

realistically test. By becoming familiar with quotas, you can work within the available resource limits during the experimentation phase.

3. **Experiment systematically**: Adjust parameters like the number of CPUs, amount of memory, number and type of GPUs and TPUs, and storage capacity. Vertex AI training and Vertex AI predictions let you experiment with different machine types and configurations.
4. **Monitor utilization, cost, and performance**: Track the resource utilization, cost, and key performance metrics such as training time, inference latency, and model accuracy, for each configuration that you experiment with.
    - To track resource utilization and performance metrics, you can use the Vertex AI console.
    - To collect and analyze detailed performance metrics, use Cloud Monitoring.
    - To view costs, use Cloud Billing reports and Cloud Monitoring dashboards.
    - To identify performance bottlenecks in your models and optimize resource utilization, use profiling tools like Vertex AI TensorBoard.
5. **Analyze costs**: Compare the cost and performance of each configuration to identify the most cost-effective option.
6. **Establish resource thresholds and improvement targets based on quotas**: Define thresholds for when scaling begins to yield diminishing returns in performance, such as minimal reduction in training time or latency for a significant cost increase. Consider project quotas when setting these thresholds. Determine the point where the cost and potential quota implications of further scaling are no longer justified by performance gains.
7. **Refine iteratively**: Repeat the experimentation process with refined configurations based on your findings. Always ensure that the resource usage remains within your allocated quotas and aligns with established cost-benefit thresholds.

Use MLOps to reduce inefficiencies

As organizations increasingly use ML to drive innovation and efficiency, managing the ML lifecycle effectively becomes critical. ML operations (MLOps) is a set of practices that automate and streamline the ML lifecycle, from model development to deployment and monitoring.

Align MLOps with cost drivers

To take advantage of MLOps for cost efficiency, identify the primary cost drivers in the ML lifecycle. Then, you can adopt and implement MLOps practices that are aligned with the cost drivers. Prioritize and adopt the MLOps features that address the most impactful cost drivers. This approach helps ensure a manageable and successful path to significant cost savings.

Implement MLOps for cost optimization

The following are common MLOps practices that help to reduce cost:

- **Version control**: Tools like Git can help you to track versions of code, data, and models. Version control ensures reproducibility, facilitates collaboration, and prevents costly rework that can be caused by versioning issues.
- **Continuous integration and continuous delivery (CI/CD)**: Cloud Build and Artifact Registry let you implement CI/CD pipelines to automate building, testing, and deployment of your ML models. CI/CD pipelines ensure efficient resource utilization and minimize the costs associated with manual interventions.
- **Observability**: Cloud Monitoring and Cloud Logging let you track model performance in production, identify issues, and trigger alerts for proactive intervention. Observability lets you maintain model accuracy, optimize resource allocation, and prevent costly downtime or performance degradation.
- **Model retraining**: Vertex AI Pipelines simplifies the processes for retraining models periodically or when performance degrades. When you use Vertex AI Pipelines for retraining, it helps ensure that your models remain accurate and efficient, which can prevent unnecessary resource consumption and maintain optimal performance.
- **Automated testing and evaluation**: Vertex AI helps you accelerate and standardize model evaluation. Implement automated tests throughout the ML lifecycle to ensure the quality and reliability of your models. Such tests can help you catch errors early, prevent costly issues in production, and reduce the need for extensive manual testing.

For more information, see MLOps: Continuous delivery and automation pipelines in machine learning.

Enforce data management and governance practices

Effective data management and governance practices are critical to cost optimization. Well organized data can encourage teams to reuse datasets, avoid needless duplication, and reduce the effort to obtain high quality data. By proactively managing data, you can reduce storage costs, enhance data quality, and ensure that your ML models are trained on the most relevant and valuable data.

To implement data management and governance practices, consider the following recommendations.

Establish and adopt a data governance framework

The growing prominence of AI and ML has made data the most valuable asset for organizations that are undergoing digital transformation. A robust framework for data governance is a crucial requirement for managing AI and ML workloads cost-effectively at scale. A data governance framework with clearly defined policies, procedures, and roles provides a structured approach for managing data throughout its lifecycle. Such a framework helps to improve data quality, enhance security, improve utilization, and reduce redundancy.

Establish a data governance framework

There are many pre-existing frameworks for data governance, such as the frameworks published by the EDM Council, with options available for different industries and organization sizes. Choose and adapt a framework that aligns with your specific needs and priorities.

Implement the data governance framework

Google Cloud provides the following services and tools to help you implement a robust data governance framework:

- Dataplex Universal Catalog is an intelligent data fabric that helps you unify distributed data and automate data governance without the need to consolidate data sets in one place. This helps to reduce the cost to distribute and maintain data, facilitate data discovery, and promote reuse.
    - To organize data, use Dataplex Universal Catalog abstractions and set up logical data lakes and zones.
    - To administer access to data lakes and zones, use Google Groups and Dataplex Universal Catalog roles.
    - To streamline data quality processes, enable auto data quality.
- Dataplex Universal Catalog is also a fully managed and scalable metadata management service. The catalog provides a foundation that ensures that data assets are accessible and reusable.
    - Metadata from the supported Google Cloud sources is automatically ingested into the universal catalog. For data sources outside of Google Cloud, create custom entries.

    - To improve the discoverability and management of data assets, enrich technical metadata with business metadata by using aspects.
    - Ensure that data scientists and ML practitioners have sufficient permissions to access Dataplex Universal Catalog and use the search function.
- BigQuery sharing lets you efficiently and securely exchange data assets across your organizations to address challenges of data reliability and cost.
    - Set up data exchanges and ensure that curated data assets can be viewed as listings.
    - Use data clean rooms to securely manage access to sensitive data and efficiently partner with external teams and organizations on AI and ML projects.
    - Ensure that data scientists and ML practitioners have sufficient permissions to view and publish datasets to BigQuery sharing.

Make datasets and features reusable throughout the ML lifecycle

For significant efficiency and cost benefits, reuse datasets and features across multiple ML projects. When you avoid redundant data engineering and feature development efforts, your organization can accelerate model development, reduce infrastructure costs, and free up valuable resources for other critical tasks.

Google Cloud provides the following services and tools to help you reuse datasets and features:

- Data and ML practitioners can publish data products to maximize reuse across teams. The data products can then be discovered and used through Dataplex Universal Catalog and BigQuery sharing.
- For tabular and structured datasets, you can use Vertex AI Feature Store to promote reusability and streamline feature management through BigQuery.
- You can store unstructured data in Cloud Storage and govern the data by using BigQuery object tables and signed URLs.
- You can manage vector embeddings by including metadata in your Vector Search indexes.

Automate and streamline with MLOps

A primary benefit of adopting MLOps practices is a reduction in costs for technology and personnel. Automation helps you avoid the duplication of ML activities and reduce the workload for data scientists and ML engineers.

To automate and streamline ML development with MLOps, consider the following recommendations.

Automate and standardize data collection and processing

To help reduce ML development effort and time, automate and standardize your data collection and processing technologies.

Automate data collection and processing

This section summarizes the products, tools, and techniques that you can use to automate data collection and processing.

Identify and choose the relevant data sources for your AI and ML tasks:

- Database options such as Cloud SQL, Spanner, AlloyDB for PostgreSQL, Firestore, and BigQuery. Your choice depends on your requirements, such as latency on write access (static or dynamic), data volume (high or low), and data format (structured, unstructured, or semi-structured). For more information, see Google Cloud databases.
- Data lakes such as Cloud Storage with BigLake.
- Dataplex Universal Catalog for governing data across sources.
- Streaming events platforms such as Pub/Sub, Dataflow, or Apache Kafka.
- External APIs.

For each of your data sources, choose an ingestion tool:

- Dataflow: For batch and stream processing of data from various sources, with ML-component integration. For an event-driven architecture, you can combine Dataflow with Eventarc to efficiently process data for ML. To enhance MLOps and ML job efficiency, use GPU and right-fitting capabilities.
- Cloud Run functions: For event-driven data ingestion that gets triggered by changes in data sources for real-time applications.
- BigQuery: For classical tabular data ingestion with frequent access.

Choose tools for data transformation and loading:

- Use tools such as Dataflow or Dataform to automate data transformations like feature scaling, encoding categorical variables, and creating new features in batch, streaming, or real time. The tools that you select depend upon your requirements and chosen services.
- Use Vertex AI Feature Store to automate feature creation and management. You can centralize features for reuse across different models and projects.

Standardize data collection and processing

To discover, understand, and manage data assets, use metadata management services like Dataplex Universal Catalog. It helps you standardize data definitions and ensure consistency across your organization.

To enforce standardization and avoid the cost of maintaining multiple custom implementations, use automated training pipelines and orchestration. For more information, see the next section.

Automate training pipelines and reuse existing assets

To boost efficiency and productivity in MLOps, automated training pipelines are crucial. Google Cloud offers a robust set of tools and services to build and deploy training pipelines, with a strong emphasis on reusing existing assets. Automated training pipelines help to accelerate model development, ensure consistency, and reduce redundant effort.

Automate training pipelines

The following table describes the Google Cloud services and features that you can use to automate the different functions of a training pipeline.

Function	Google Cloud services and features
Orchestration: Define complex ML workflows consisting of multiple steps and dependencies. You can define each step as a separate containerized task, which helps you manage and scale individual tasks with ease.	• To create and orchestrate pipelines, use Vertex AI Pipelines or Kubeflow Pipelines. These tools support simple data transformation, model training, model deployment, and pipeline versioning. They let you define dependencies between steps, manage data flow, and automate the execution of the entire workflow. • For complex operational tasks with heavy CI/CD and extract, transform, and load (ETL) requirements, use Cloud Composer. If you prefer Airflow for data orchestration, Cloud Composer is a compatible managed service that's built on Airflow. • For pipelines that are managed outside of Vertex AI Pipelines, use Workflows for infrastructure-focused tasks like starting and stopping VMs or integrating with external systems.

Function	Google Cloud services and features
	• To automate your CI/CD process, use Cloud Build with Pub/Sub. You can set up notifications and automatic triggers for when new code is pushed or when a new model needs to be trained. • For a fully-managed, scalable solution for pipeline management, use Cloud Data Fusion.
Versioning: Track and control different versions of pipelines and components to ensure reproducibility and auditability.	Store Kubeflow pipeline templates in a Kubeflow Pipelines repository in Artifact Registry.
Reusability: Reuse existing pipeline components and artifacts, such as prepared datasets and trained models, to accelerate development.	Store your pipeline templates in Cloud Storage and share them across your organization.
Monitoring: Monitor pipeline execution to identify and address any issues.	Use Cloud Logging and Cloud Monitoring. For more information, see Monitor resources continuously with dashboards, alerts, and reports.

Expand reusability beyond pipelines

Look for opportunities to expand reusability beyond training pipelines. The following are examples of Google Cloud capabilities that let you reuse ML features, datasets, models, and code.

- Vertex AI Feature Store provides a centralized repository for organizing, storing, and serving ML features. It lets you reuse features across different projects and models, which can improve consistency and reduce feature engineering effort. You can store, share, and access features for both online and offline use cases.
- Vertex AI datasets enable teams to create and manage datasets centrally, so your organization can maximize reusability and reduce data duplication. Your teams can search and discover the datasets by using Dataplex Universal Catalog.

- Vertex AI Model Registry lets you store, manage, and deploy your trained models. Model Registry lets you reuse the models in subsequent pipelines or for online prediction, which helps you take advantage of previous training efforts.
- Custom containers let you package your training code and dependencies into containers and store the containers in Artifact Registry. Custom containers let you provide consistent and reproducible training environments across different pipelines and projects.

Use Google Cloud services for model evaluation and tuning

Google Cloud offers a powerful suite of tools and services to streamline and automate model evaluation and tuning. These tools and services can help you reduce your time to production and reduce the resources required for continuous training and monitoring. By using these services, your AI and ML teams can enhance model performance with fewer expensive iterations, achieve faster results, and minimize wasted compute resources.

Use resource-efficient model evaluation and experimentation

Begin an AI project with experiments before you scale up your solution. In your experiments, track various metadata such as dataset version, model parameters, and model type. For further reproducibility and comparison of the results, use metadata tracking in addition to code versioning, similar to the capabilities in Git. To avoid missing information or deploying the wrong version in production, use Vertex AI Experiments before you implement full-scale deployment or training jobs.

Vertex AI Experiments lets you do the following:

- Streamline and automate metadata tracking and discovery through a user friendly UI and API for production-ready workloads.
- Analyze the model's performance metrics and compare metrics across multiple models.

After the model is trained, continuously monitor the performance and data drift over time for incoming data. To streamline this process, use Vertex AI Model Monitoring to directly access the created models in Model Registry. Model Monitoring also automates monitoring for data and results through online and batch predictions. You can export the results to BigQuery for further analysis and tracking.

Choose optimal strategies to automate training

For hyperparameter tuning, we recommend the following approaches:

- To automate the process of finding the optimal hyperparameters for your models, use Vertex AI hyperparameter tuning. Vertex AI uses advanced algorithms to explore the hyperparameter space and identify the best configuration.

- For efficient hyperparameter tuning, consider using Bayesian optimization techniques, especially when you deal with complex models and large datasets.

For distributed training, we recommend the following approaches:

- For large datasets and complex models, use the distributed training infrastructure of Vertex AI. This approach lets you train your models on multiple machines, which helps to significantly reduce training time and associated costs. Use tools like the following:
    - Vertex AI tuning to perform supervised fine-tuning of Gemini, Imagen, and other models.
    - Vertex AI training or Ray on Vertex AI for custom distributed training.
- Choose optimized ML frameworks, like Keras and PyTorch, that support distributed training and efficient resource utilization.

Use explainable AI

It's crucial to understand why a model makes certain decisions and to identify potential biases or areas for improvement. Use Vertex Explainable AI to gain insights into your model's predictions. Vertex Explainable AI offers a way to automate feature-based and example-based explanations that are linked to your Vertex AI experiments.

- Feature-based: To understand which features are most influential in your model's predictions, analyze feature attributions. This understanding can guide feature-engineering efforts and improve model interpretability.
- Example-based: To return a list of examples (typically from the training set) that are most similar to the input, Vertex AI uses nearest neighbor search. Because similar inputs generally yield similar predictions, you can use these explanations to explore and explain a model's behavior.

Use managed services and pre-trained models

Adopt an incremental approach to model selection and model development. This approach helps you avoid excessive costs that are associated with starting afresh every time. To control costs, use ML frameworks, managed services, and pre-trained models.

To get the maximum value from managed services and pre-trained models, consider the following recommendations.

Use notebooks for exploration and experiments

Notebook environments are crucial for cost-effective ML experimentation. A notebook provides an interactive and collaborative space for data scientists and engineers to explore data, develop models, share knowledge, and iterate efficiently. Collaboration and knowledge sharing through notebooks significantly accelerates development, code reviews, and knowledge transfer. Notebooks help streamline workflows and reduce duplicated effort.

Instead of procuring and managing expensive hardware for your development environment, you can use the scalable and on-demand infrastructure of Vertex AI Workbench and Colab Enterprise.

- Vertex AI Workbench is a Jupyter notebook development environment for the entire data science workflow. You can interact with Vertex AI and other Google Cloud services from within an instance's Jupyter notebook. Vertex AI Workbench integrations and features help you do the following:
    - Access and explore data from a Jupyter notebook by using BigQuery and Cloud Storage integrations.
    - Automate recurring updates to a model by using scheduled executions of code that runs on Vertex AI.
    - Process data quickly by running a notebook on a Dataproc cluster.
    - Run a notebook as a step in a pipeline by using Vertex AI Pipelines.
- Colab Enterprise is a collaborative, managed notebook environment that has the security and compliance capabilities of Google Cloud. Colab Enterprise is ideal if your project's priorities include collaborative development and reducing the effort to manage infrastructure. Colab Enterprise integrates with Google Cloud services and AI-powered assistance that uses Gemini. Colab Enterprise lets you do the following:
    - Work in notebooks without the need to manage infrastructure.
    - Share a notebook with a single user, Google group, or Google Workspace domain. You can control notebook access through Identity and Access Management (IAM).
    - Interact with features built into Vertex AI and BigQuery.

To track changes and revert to previous versions when necessary, you can integrate your notebooks with version control tools like Git.

Start with existing and pre-trained models

Training complex models from scratch, especially deep-learning models, requires significant computational resources and time. To accelerate your model selection and development process, start with existing and pre-trained models. These models, which are trained on vast datasets, eliminate the need to train models from scratch and significantly reduce cost and development time.

Reduce training and development costs

Select an appropriate model or API for each ML task and combine them to create an end-to-end ML development process.

Vertex AI Model Garden offers a vast collection of pre-trained models for tasks such as image classification, object detection, and natural language processing. The models are grouped into the following categories:

- Google models like the Gemini family of models and Imagen for image generation.
- Open-source models like Gemma and Llama.
- Third-party models from partners like Anthropic and Mistral AI.

Google Cloud provides AI and ML APIs that let developers integrate powerful AI capabilities into applications without the need to build models from scratch.

- Cloud Vision API lets you derive insights from images. This API is valuable for applications like image analysis, content moderation, and automated data entry.
- Cloud Natural Language API lets you analyze text to understand its structure and meaning. This API is useful for tasks like customer feedback analysis, content categorization, and understanding social media trends.
- Speech-to-Text API converts audio to text. This API supports a wide range of languages and dialects.
- Video Intelligence API analyzes video content to identify objects, scenes, and actions. Use this API for video content analysis, content moderation, and video search.
- Document AI API processes documents to extract, classify, and understand data. This API helps you automate document processing workflows.
- Dialogflow API enables the creation of conversational interfaces, such as chatbots and voice assistants. You can use this API to create customer service bots and virtual assistants.
- Gemini API in Vertex AI provides access to Google's most capable and general-purpose AI model.

Reduce tuning costs

To help reduce the need for extensive data and compute time, fine-tune your pre-trained models on specific datasets. We recommend the following approaches:

- **Learning transfer**: Use the knowledge from a pre-trained model for a new task, instead of starting from scratch. This approach requires less data and compute time, which helps to reduce costs.
- **Adapter tuning (parameter-efficient tuning)**: Adapt models to new tasks or domains without full fine-tuning. This approach requires significantly lower computational resources and a smaller dataset.
- **Supervised fine tuning**: Adapt model behavior with a labeled dataset. This approach simplifies the management of the underlying infrastructure and the development effort that's required for a custom training job.

Explore and experiment by using Vertex AI Studio

Vertex AI Studio lets you rapidly test, prototype, and deploy generative AI applications.

- **Integration with Model Garden**: Provides quick access to the latest models and lets you efficiently deploy the models to save time and costs.
- **Unified access to specialized models**: Consolidates access to a wide range of pre-trained models and APIs, including those for chat, text, media, translation, and speech. This unified access can help you reduce the time spent searching for and integrating individual services.

Use managed services to train or serve models

Managed services can help reduce the cost of model training and simplify the infrastructure management, which lets you focus on model development and optimization. This approach can result in significant cost benefits and increased efficiency.

Reduce operational overhead

To reduce the complexity and cost of infrastructure management, use managed services such as the following:

- Vertex AI training provides a fully managed environment for training your models at scale. You can choose from various prebuilt containers with popular ML frameworks or use your own custom containers. Google Cloud handles infrastructure provisioning, scaling, and maintenance, so you incur lower operational overhead.
- Vertex AI predictions handles infrastructure scaling, load balancing, and request routing. You get high availability and performance without manual intervention.

- Ray on Vertex AI provides a fully managed Ray cluster. You can use the cluster to run complex custom AI workloads that perform many computations (hyperparameter tuning, model fine-tuning, distributed model training, and reinforcement learning from human feedback) without the need to manage your own infrastructure.

Use managed services to optimize resource utilization

For details about efficient resource utilization, see Optimize resource utilization.

Contributors

Authors:

- Isaac Lo | AI Business Development Manager
- Anastasia Prokaeva | Field Solutions Architect, Generative AI
- Amy Southwood | Technical Solutions Consultant, Data Analytics & AI

Other contributors:

- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Kumar Dhanagopal | Cross-Product Solution Developer
- Marwan Al Shawi | Partner Customer Engineer
- Nicolas Pintaux | Customer Engineer, Application Modernization Specialist

AI and ML perspective: Performance optimization

This document in the Well-Architected Framework: AI and ML perspective provides an overview of principles and recommendations to help you to optimize the performance of your AI and ML workloads on Google Cloud. The recommendations in this document align with the performance optimization pillar of the Google Cloud Well-Architected Framework.

AI and ML systems enable new automation and decision-making capabilities for your organization. The performance of these systems can directly affect your business drivers like revenue, costs, and customer satisfaction. To realize the full potential of your AI and ML systems, you need to optimize their performance based on your business goals and technical requirements. The performance optimization process often involves certain trade-offs. For example, a design choice that provides the required performance might lead to higher costs. The recommendations in this document prioritize performance over other considerations like costs.

To optimize AI and ML performance, you need to make decisions regarding factors like the model architecture, parameters, and training strategy. When you make these decisions, consider the entire lifecycle of the AI and ML systems and their deployment environment. For example, LLMs that are very large can be highly performant on massive training infrastructure, but very large models might not perform well in capacity-constrained environments like mobile devices.

Translate business goals to performance objectives

To make architectural decisions that optimize performance, start with a clear set of business goals. Design AI and ML systems that provide the technical performance that's required to support your business goals and priorities. Your technical teams must understand the mapping between performance objectives and business goals.

Consider the following recommendations:

- **Translate business objectives into technical requirements**: Translate the business objectives of your AI and ML systems into specific technical performance requirements and assess the effects of not meeting the requirements. For example, for an application that predicts customer churn, the ML model should perform well on standard metrics, like accuracy and recall, *and* the application should meet operational requirements like low latency.
- **Monitor performance at all stages of the model lifecycle**: During experimentation and training after model deployment, monitor your key performance indicators (KPIs) and observe any deviations from business objectives.
- **Automate evaluation to make it reproducible and standardized**: With a standardized and comparable platform and methodology for experiment evaluation, your engineers can increase the pace of performance improvement.

Run and track frequent experiments

To transform innovation and creativity into performance improvements, you need a culture and a platform that supports experimentation. Performance improvement is an ongoing process because AI and ML technologies are developing continuously and quickly. To maintain a fast-paced, iterative process, you need to separate the experimentation space from your training and serving platforms. A standardized and robust experimentation process is important.

Consider the following recommendations:

- **Build an experimentation environment**: Performance improvements require a dedicated, powerful, and interactive environment that supports the experimentation and collaborative development of ML pipelines.

- **Embed experimentation as a culture**: Run experiments before any production deployment. Release new versions iteratively and always collect performance data. Experiment with different data types, feature transformations, algorithms, and hyperparameters.

Build and automate training and serving services

Training and serving AI models are core components of your AI services. You need robust platforms and practices that support fast and reliable creation, deployment, and serving of AI models. Invest time and effort to create foundational platforms for your core AI training and serving tasks. These foundational platforms help to reduce time and effort for your teams and improve the quality of outputs in the medium and long term.

Consider the following recommendations:

- **Use AI-specialized components of a training service**: Such components include high-performance compute and MLOps components like feature stores, model registries, metadata stores, and model performance-evaluation services.
- **Use AI-specialized components of a prediction service**: Such components provide high-performance and scalable resources, support feature monitoring, and enable model performance monitoring. To prevent and manage performance degradation, implement reliable deployment and rollback strategies.

Match design choices to performance requirements

When you make design choices to improve performance, carefully assess whether the choices support your business requirements or are wasteful and counterproductive. To choose the appropriate infrastructure, models, or configurations, identify performance bottlenecks and assess how they're linked to your performance measures. For example, even on very powerful GPU accelerators, your training tasks can experience performance bottlenecks due to data I/O issues from the storage layer or due to performance limitations of the model itself.

Consider the following recommendations:

- **Optimize hardware consumption based on performance goals**: To train and serve ML models that meet your performance requirements, you need to optimize infrastructure at the compute, storage, and network layers. You must measure and understand the variables that affect your performance goals. These variables are different for training and inference.
- **Focus on workload-specific requirements**: Focus your performance optimization efforts on the unique requirements of your AI and ML workloads. Rely on managed services for the performance of the underlying infrastructure.

- **Choose appropriate training strategies**: Several pre-trained and foundational models are available, and more such models are released often. Choose a training strategy that can deliver optimal performance for your task. Decide whether you should build your own model, tune a pre-trained model on your data, or use a pre-trained model API.
- **Recognize that performance-optimization strategies can have diminishing returns**: When a particular performance-optimization strategy doesn't provide incremental business value that's measurable, stop pursuing that strategy.

Link performance metrics to design and configuration choices

To innovate, troubleshoot, and investigate performance issues, establish a clear link between design choices and performance outcomes. In addition to experimentation, you must reliably record the lineage of your assets, deployments, model outputs, and the configurations and inputs that produced the outputs.

Consider the following recommendations:

- **Build a data and model lineage system**: All of your deployed assets and their performance metrics must be linked back to the data, configurations, code, and the choices that resulted in the deployed systems. In addition, model outputs must be linked to specific model versions and how the outputs were produced.
- **Use explainability tools to improve model performance**: Adopt and standardize tools and benchmarks for model exploration and explainability. These tools help your ML engineers understand model behavior and improve performance or remove biases.

Contributors

Authors:

- Benjamin Sadik | AI and ML Specialist Customer Engineer
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist

Other contributors:

- Kumar Dhanagopal | Cross-Product Solution Developer
- Marwan Al Shawi | Partner Customer Engineer
- Zach Seils | Networking Specialist

Well-Architected Framework: Financial services industry (FSI) perspective

Last reviewed 2025-07-28 UTC

This page provides a one-page view of all of the pages in the FSI perspective of the Well-Architected Framework. You can print this page or save it in PDF format by using your browser's print function.

This page doesn't have a table of contents. You can't use the links on this page to navigate within the page.

This document in the Google Cloud Well-Architected Framework describes principles and recommendations to help you to design, build, and manage financial services industry (FSI) applications in Google Cloud that meet your operational, security, reliability, cost, and performance goals.

The target audience for this document includes decision makers, architects, administrators, developers, and operators who design, build, deploy, and maintain FSI workloads in Google Cloud. Examples of FSI organizations that could benefit from this guidance include banks, payment infrastructure players, insurance providers, and capital market operators.

FSI organizations have specific considerations, particularly for architecture and resilience. These considerations are primarily driven by regulatory, risk, and performance requirements. This document provides high-level guidance that's based on design considerations that we've observed across a wide range of FSI customers globally. Whether your workloads are fully in the cloud or transitioning to hybrid or multi-cloud deployments, the guidance in this document helps you design workloads on Google Cloud to meet your regulatory requirements and diverse risk perspectives. The guidance might not address the unique challenges of every organization. It provides a foundation that addresses many of the primary regulatory requirements of FSI organizations.

A primary challenge in designing cloud workloads involves aligning cloud deployments with on-premises environments, especially when you aim for consistent approaches to security, reliability, and resilience. Cloud services create opportunities to fundamentally rethink your architecture in order to reduce management overhead, optimize cost, enhance security, and improve reliability and resilience.

The following pages describe principles and recommendations that are specific to FSI workloads for each pillar of the Well-Architected Framework:

- FSI perspective: Operational excellence
- FSI perspective: Security
- FSI perspective: Reliability
- FSI perspective: Cost optimization
- FSI perspective: Performance optimization

Contributors

Authors:

- Gino Pelliccia | Principal Architect
- Alex Stepney | Lead Principal Architect
- Phil Bryan | EMEA FSI Lead Principal Architect
- Stathis Onasoglou | EMEA FSI Principal Architect
- Sam Moss | EMEA FinOps Professional Services Lead

Other contributors:

- Daniel Lees | Cloud Security Architect
- Danielle Fisla | US FS Portfolio Lead, PSO
- Filipe Gracio, PhD | Customer Engineer, AI/ML Specialist
- Henry Cheng | Principal Architect
- John Bacon | Partner Solutions Architect
- Jose Andrade | Customer Engineer, SRE Specialist
- Kumar Dhanagopal | Cross-Product Solution Developer
- Laura Hyatt | Customer Engineer, FSI
- Michael Yang | Industry Solutions AI Consulting Lead, FSI
- Nicolas Pintaux | Customer Engineer, Application Modernization Specialist
- Omar Saenz | EMEA Partner Engineer, Security
- Radhika Kanakam | Program Lead, Google Cloud Well-Architected Framework
- Steve McGhee | Reliability Advocate
- Tarun Sharma | Principal Architect
- Yuriy Babenko | Customer Engineer, FSI

FSI perspective: Operational excellence

This document in the Google Cloud Well-Architected Framework: FSI perspective provides an overview of the principles and recommendations to build, deploy, and operate robust financial services industry (FSI) workloads in Google Cloud. These recommendations help you set up foundational elements like observability, automation, and scalability. The recommendations in this document align with the operational excellence pillar of the Well-Architected Framework.

Operational excellence is critical for FSI workloads in Google Cloud due to the highly regulated and sensitive nature of such workloads. Operational excellence ensures that cloud solutions can adapt to evolving needs *and* meet your requirements for value, performance, security, and reliability. Failures in these areas could result in significant financial losses, regulatory penalties, and reputational damage.

Operational excellence provides the following benefits for FSI workloads:

- **Maintain trust and reputation**: Financial institutions rely heavily on their customers' trust. Operational disruptions or security breaches can severely erode this trust and cause customer attrition. Operational excellence helps to minimize these risks.
- **Meet stringent regulatory compliance requirements**: The FSI is subject to numerous and complex regulations, such as the following:
    - EU General Data Protection Regulation (GDPR)
    - EU Digital Operational Resilience Act (DORA)
    - California Consumer Privacy Act (CCPA)
    - Industry-specific regulations

Robust operational processes, monitoring, and incident management are essential for demonstrating compliance with regulations and avoiding penalties.

- **Ensure business continuity and resilience**: Financial markets and services often operate continuously. Therefore, high availability and effective disaster recovery are paramount. Operational excellence principles guide the design and implementation of resilient systems. The reliability pillar provides more guidance in this area.
- **Protect sensitive data**: Financial institutions handle vast amounts of highly sensitive customer and financial data. Strong operational controls, security monitoring, and rapid incident response are crucial in order to prevent data breaches and maintain privacy. The security pillar provides more guidance in this area.
- **Optimize performance for critical applications**: Many financial applications, such as trading platforms and real-time analytics, demand high performance and low latency. To meet these performance requirements, you need highly optimized compute, networking, and storage design. The performance optimization pillar provides more guidance in this area.

- **Manage costs effectively**: In addition to security and reliability, financial institutions are also concerned with cost efficiency. Operational excellence includes practices for optimizing resource utilization and managing cloud spending. The cost optimization pillar provides more guidance in this area.

The operational excellence recommendations in this document are mapped to the following core principles:

- Define SLAs and corresponding SLOs and SLIs
- Define and test incident management processes
- Continuously improve and innovate

Define SLAs and corresponding SLOs and SLIs

Across many FSI organizations, the availability of applications is typically classified based on recovery time objective (RTO) and recovery point objective (RPO) metrics. For business-critical applications that serve external customers, a service level agreement (SLA) might also be defined.

SLAs need a framework of metrics that represents the behavior of the system from the user-satisfaction perspective. Site reliability engineering (SRE) practices offer a way to achieve the level of system reliability that you want. Creating a framework of metrics involves defining and monitoring key numerical indicators to understand system health from the user's perspective. For example, metrics like latency and error rates quantify how well a service is performing. These metrics are called service level indicators (SLIs). Developing effective SLIs is crucial, because they provide the raw data that's necessary to objectively assess reliability.

To define meaningful SLAs, SLIs, and SLOs, consider the following recommendations:

- Develop and define SLIs for each critical service. Set target values that define the acceptable performance levels.
- Develop and define the service level objectives (SLO) that correspond to the SLIs. For example, an SLO might state that 99.9% of requests must have a latency that's less than 200 milliseconds.
- Identify the internal remedial actions that must be taken if a service doesn't meet the SLOs. For example, to improve the resilience of the platform, you might need to focus development resources on fixing issues.
- Validate the SLA requirement for each service and recognize the SLA as the formal contract with the service users.

Examples of service levels

The following table provides examples of SLIs, SLOs, and SLAs for a payment platform:

Business metric	SLI	SLO	SLA
Payment transaction success	A quantitative measure of the percentage of all initiated payment transactions that are successfully processed and confirmed. Example: (number of successful transactions ÷ total number of valid transactions) × 100, measured over a rolling 5-minute window.	An internal target to maintain a high percentage of successful payment transactions over a specific period. Example: Maintain a 99.98% payment transaction success rate over a rolling 30-day window, excluding invalid requests and planned maintenance.	A contractual guarantee for the success rate and speed of payment transaction processing. Example: The service provider guarantees that 99.0% of payment transactions initiated by the client will be successfully processed and confirmed within one second.
Payment processing latency	The average time taken for a payment transaction to be processed from initiation by the client to final confirmation. Example: Average response time in milliseconds for transaction confirmation, measured over a rolling 5-minute window.	An internal target for the speed at which payment transactions are processed. Example: Ensure that 99.5% of payment transactions are processed within 400 milliseconds over a rolling 30-day window.	A contractual commitment to resolve critical payment processing issues within a specified timeframe. Example: For critical payment processing issues (defined as an outage that affects more than 1% of transactions), the service provider commits to a resolution time of within two hours from the time when the issue is reported or detected.

Business metric	SLI	SLO	SLA
Platform availability	The percentage of time when the core payment processing API and user interface are operational and accessible to clients. Example: (total operational time − downtime) ÷ total operational time × 100, measured per minute.	An internal target for the uptime of the core payment platform. Example: Achieve 99.995% platform availability per calendar month, excluding scheduled maintenance windows.	A formal, legally binding commitment to clients regarding the minimum uptime of the payment platform, including consequences for failure to meet. Example: The platform will maintain a minimum of 99.9% availability per calendar month, excluding scheduled maintenanc windows. If the availability falls below the minimum level, the client will receive a service credit of 5% of the monthly service fee for each 0.1% drop.

Use SLI data to monitor whether systems are within the defined SLOs and to ensure that the SLAs are met. By using a set of well-defined SLIs, engineers and developers can monitor FSI applications at the following levels:

- Directly within the service that the applications are deployed on, such as GKE or Cloud Run.
- By using logs that are provided by infrastructure components, such as the load balancer.

OpenTelemetry provides an open source standard and a set of technologies to capture all types of telemetry including metrics, traces, and logs. Google Cloud Managed Service for Prometheus provides a fully-managed, highly scalable backend for metrics and operation of Prometheus at scale.

For more information about SLI, SLO, and error budgets, see the SRE handbook.

To develop effective alerting and monitoring dashboards and mechanisms, use Google Cloud Observability tools together with Google Cloud Monitoring. For information about security-specific monitoring and detection capabilities, see the security pillar.

Define and test incident management processes

Well-defined and regularly tested incident management processes contribute directly to the value, performance, security, and reliability of the FSI workloads in Google Cloud. These processes help financial institutions meet their stringent regulatory requirements, protect sensitive data, maintain business continuity, and uphold customer trust.

Regular testing of incident management processes provides the following benefits:

- **Maintain performance under peak loads**: Regular performance and load testing help financial institutions ensure that their cloud-based applications and infrastructure can handle peak transaction volumes, market volatility, and other high-demand scenarios without performance degradation. This capability is crucial for maintaining a seamless user experience and meeting the demands of financial markets.
- **Identify potential bottlenecks and limitations**: Stress testing pushes systems to their limits, and it enables financial institutions to identify potential bottlenecks and performance limitations before they affect critical operations. This proactive approach enables financial institutions to adjust their infrastructure and applications for optimal performance and scalability.
- **Validate reliability and resilience**: Regular testing, including chaos engineering or simulated failures, helps to validate the reliability and resilience of financial systems. This testing ensures that the systems can recover gracefully from failures and maintain high availability, which is essential for business continuity.
- **Perform effective capacity planning**: Performance testing provides valuable data on resource utilization under different load conditions, which is crucial for accurate capacity planning. Financial institutions can use this data to proactively anticipate future capacity needs and to avoid performance issues due to resource constraints.
- **Deploy new features and code changes successfully**: Integrating automated testing into CI/CD pipelines helps to ensure that changes and new deployments are thoroughly validated before they're released into production environments. This approach significantly reduces the risk of errors and regressions that could lead to operational disruptions.
- **Meet regulatory requirements for system stability**: Financial regulations often require institutions to have robust testing practices to ensure the stability and reliability of their critical systems. Regular testing helps to demonstrate compliance with these requirements.

To define and test your incident management processes, consider the following recommendations.

Establish clear incident response procedures

A well-established set of incident response procedures involves the following elements:

- Roles and responsibilities that are defined for incident commanders, investigators, communicators, and technical experts to ensure effective and coordinated response.
- Communication protocols and escalation paths that are defined to ensure that information is shared promptly and effectively during incidents.
- Procedures that are documented in a runbook or playbook that outlines the steps for communication, triage, investigation, and resolution.
- Regular training and preparation that equips teams with the knowledge and skills to respond effectively.

Implement performance and load testing regularly

Regular performance and load testing helps to ensure that cloud-based applications and infrastructure can handle peak loads and maintain optimal performance. Load testing simulates realistic traffic patterns. Stress testing exercises the system to its limits to identify potential bottlenecks and performance limitations. You can use products like Cloud Load Balancing and load testing services to simulate real-world traffic. Based on the test results, you can adjust your cloud infrastructure and applications for optimal performance and scalability. For example, you can adjust resource allocation or tune application configurations.

Automate testing within CI/CD pipelines

Incorporating automated testing into your CI/CD pipelines helps to ensure the quality and reliability of cloud applications by validating changes before deployment. This approach significantly reduces the risk of errors and regressions and it helps you to build a more stable and robust software system. You can incorporate different types of testing in your CI/CD pipelines, including unit testing, integration testing, and end-to-end testing. Use products like Cloud Build and Cloud Deploy to create and manage your CI/CD pipelines.

Continuously improve and innovate

For financial services workloads in the cloud, migrating to the cloud is merely the initial step. Ongoing enhancement and innovation are essential for the following reasons:

- **Accelerate innovation**: Take advantage of new technologies like AI to improve your services.
- **Reduce costs**: Eliminate inefficiencies and optimize resource use.
- **Enhance agility**: Adapt to market and regulatory changes quickly.

- **Improve decision making**: Use data analytics products like BigQuery and Looker to make informed choices.

To ensure continuous improvement and innovation, consider the following recommendations.

Conduct regular retrospectives

Retrospectives are vital for continuously improving incident response procedures, and for optimizing testing strategies based on the outcomes of regular performance and load testing. To ensure that retrospectives are effective, do the following:

- Give teams an opportunity to reflect on their experiences, identify what went well, and pinpoint areas for improvement.
- Hold retrospectives after project milestones, major incidents, or significant testing cycles. Teams can learn from both successes and failures and continuously refine their processes and practices.
- Use a structured approach like the start-stop-continue model to ensure that the retrospective sessions are productive and lead to actionable steps.
- Use retrospectives to identify areas where automation of change management can be further enhanced to improve reliability and reduce risks.

Foster a culture of learning

A culture of learning facilitates safe exploration of new technologies in Google Cloud, such as AI and ML capabilities to enhance services like fraud detection and personalized financial advice. To promote a culture of learning, do the following:

- Encourage teams to experiment, share knowledge, and learn continuously.
- Adopt a blameless culture, where failures are viewed as opportunities for growth and improvement.
- Create a psychologically safe environment that lets teams take risks and consider innovative solutions. Teams learn from both successes and failures, which leads to a more resilient and adaptable organization.
- Develop a culture that facilitates sharing of knowledge gained from incident management processes and testing exercises.

Stay up-to-date with cloud technologies

Continuous learning is essential for understanding and implementing new security measures, leveraging advanced data analytics for better insights, and adopting innovative solutions that are relevant to the financial industry.

- Maximize the potential of Google Cloud services by staying informed about the latest advancements, features, and best practices.
- When new Google Cloud features and services are introduced, identify opportunities to further automate processes, enhance security, and improve the performance and scalability of your applications.
- Participate in relevant conferences, webinars, and training sessions to expand your knowledge and understand new capabilities.
- Encourage team members to obtain Google Cloud certifications to help ensure that the organization has the necessary skills for success in the cloud.

FSI perspective: Security, privacy, and compliance

This document in the Google Cloud Well-Architected Framework: FSI perspective provides an overview of the principles and recommendations to address the security, privacy, and compliance requirements of financial services industry (FSI) workloads in Google Cloud. The recommendations help you build resilient and compliant infrastructure, safeguard sensitive data, maintain customer trust, navigate the complex landscape of regulatory requirements, and effectively manage cyber threats. The recommendations in this document align with the security pillar of the Well-Architected Framework.

Security in cloud computing is a critical concern for FSI organizations, which are highly attractive to cybercriminals due to the vast amounts of sensitive data that they manage, including customer details and financial records. The consequences of a security breach are exceptionally severe, including significant financial losses, long-term reputational damage, and significant regulatory fines. Therefore, FSI workloads need stringent security controls.

To help ensure comprehensive security and compliance, you need to understand the shared responsibilities between you (FSI organizations) and Google Cloud. Google Cloud is responsible for securing the underlying infrastructure, including physical security and network security. You are responsible for securing data and applications, configuring access control, and configuring and managing security services. To support you in your security efforts, the Google Cloud partner ecosystem offers security integration and managed services.

The security recommendations in this document are mapped to the following core principles:

- Implement security by design
- Implement zero trust
- Implement shift-left security

- Implement preemptive cyber defense
- Use AI securely and responsibly, and use AI for security
- Meet regulatory, compliance, and privacy needs
- Prioritize security initiatives

Implement security by design

Financial regulations like the Payment Card Industry Data Security Standard (PCI DSS), the Gramm-Leach-Bliley Act (GLBA) in the United States, and various national financial data protection laws mandate that security is integrated into systems from the outset. The security-by-design principle emphasizes the integration of security throughout the development lifecycle to help ensure that vulnerabilities are minimized from the outset.

To apply the security-by-design principle for your FSI workloads in Google Cloud, consider the following recommendations:

- Ensure that only necessary permissions are granted by applying the principle of least privilege through granular role-based access control (RBAC) in Identity and Access Management (IAM). The use of RBAC is a key requirement in many financial regulations.
- Enforce security perimeters around your sensitive services and data within Google Cloud by using VPC Service Controls. The security perimeters help to segment and protect sensitive data and resources, and help to prevent data exfiltration and unauthorized access, as required by regulations.
- Define security configurations as code by using infrastructure as code (IaC) tools like Terraform. This approach embeds security controls from the initial deployment phase, which helps to ensure consistency and auditability.
- Scan your application code by integrating Static Application Security Testing (SAST) into the CI/CD pipeline with Cloud Build. Establish automated security gates to prevent the deployment of non-compliant code.
- Provide a unified interface for security insights by using Security Command Center. The use of Security Command Center enables continuous monitoring and early detection of misconfigurations or threats that could lead to regulatory breaches. To meet the requirements of standards such as ISO 27001 and NIST 800-53, you can use posture management templates.
- Track the reduction in vulnerabilities that are identified in production deployments and the percentage of IaC deployments that adhere to security best practices. You can detect and view vulnerabilities and information about

compliance to security standards by using Security Command Center. For more information, see Vulnerability findings.

Implement zero trust

Modern financial regulations increasingly emphasize the need for stringent access controls and continuous verification. These requirements reflect the principle of zero trust, which aims to protect workloads against both internal and external threats and bad actors. The zero-trust principle advocates for continuous verification of every user and device, which eliminates implicit trust and mitigates lateral movement.

To implement zero trust, consider the following recommendations:

- Enable context-aware access based on user identity, device security, location, and other factors by combining IAM controls with Chrome Enterprise Premium. This approach ensures continuous verification before access to financial data and systems is granted.
- Provide secure and scalable identity and access management by configuring Identity Platform (or your external identity provider if you use Workforce Identity Federation). Set up multi-factor authentication (MFA) and other controls that are crucial to implement zero trust and help ensure regulatory compliance.
- Implement MFA for all user accounts, especially for accounts with access to sensitive data or systems.
- Support audits and investigations related to regulatory compliance by establishing comprehensive logging and monitoring of user access and network activity.
- Enable private and secure communication between services within Google Cloud and on-premises environments without exposing the traffic to the public internet by using Private Service Connect.
- Implement granular identity controls and authorize access at the application level by using Identity-Aware Proxy (IAP) rather than relying on network-based security mechanisms like VPN tunnels. This approach helps to reduce lateral movement within the environment.

Implement shift-left security

Financial regulators encourage proactive security measures. Identifying and addressing vulnerabilities early in the development lifecycle helps to reduce the risk of security incidents and the potential for non-compliance penalties. The principle of shift-left security promotes early security testing and integration, which helps to reduce the cost and complexity of remediation.

To implement shift-left security, consider the following recommendations:

- Ensure automated security checks early in the development process by integrating security scanning tools, such as container vulnerability scanning and static code analysis, into the CI/CD pipeline with Cloud Build.
- Ensure that only secure artifacts are deployed by using Artifact Registry to provide a secure and centralized repository for software packages and container images with integrated vulnerability scanning. Use virtual repositories to mitigate dependency confusion attacks by prioritizing your private artifacts over remote repositories.
- Automatically scan web applications for common vulnerabilities by integrating Web Security Scanner, which is a part of Security Command Center, into your development pipelines.
- Implement security checks for the source code, build process, and code provenance by using the Supply-chain Levels for Software Artifacts (SLSA) framework. Enforce the provenance of the workloads that run in your environments by using solutions such as Binary Authorization. Ensure that your workloads use only verified open-source software libraries by using Assured Open Source.
- Track the number of vulnerabilities that are identified and remediated in your development lifecycle, the percentage of code deployments that pass security scans, and the reduction in security incidents caused by software vulnerabilities. Google Cloud provides tools to help with this tracking for different kinds of workloads. For example, for containerized workloads, use the container scanning feature of Artifact Registry.

Implement preemptive cyber defense

Financial institutions are prime targets for sophisticated cyberattacks. Regulations often require robust threat intelligence and proactive defense mechanisms. Preemptive cyber defense focuses on proactive threat detection and response by using advanced analytics and automation.

Consider the following recommendations:

- Proactively identify and mitigate potential threats, by using the threat intelligence, incident response, and security validation services of Mandiant.
- Protect web applications and APIs from web exploits and DDoS attacks at the network edge by using Google Cloud Armor.

- Aggregate and prioritize security findings and recommendations by using Security Command Center, which enables security teams to proactively address potential risks.
- Validate preemptive defenses and incident response plans by conducting regular security simulations and penetration testing.
- Measure the time to detect and respond to security incidents, the effectiveness of DDoS mitigation efforts, and the number of prevented cyberattacks. You can get the required metrics and data from Google Security Operations SOAR and SIEM dashboards.

Use AI securely and responsibly, and use AI for security

AI and ML are increasingly used for financial services use cases such as fraud detection and algorithmic trading. Regulations require that these technologies be used ethically, transparently, and securely. AI can also help to enhance your security capabilities. Consider the following recommendations for using AI:

- Develop and deploy ML models in a secure and governed environment by using Vertex AI. Features like model explainability and fairness metrics can help to address responsible-AI concerns.
- Leverage the security analytics and operations capabilities of Google Security Operations, which uses AI and ML to analyze large volumes of security data, detect anomalies, and automate threat response. These capabilities help to enhance your overall security posture and aid in compliance monitoring.
- Establish clear governance policies for AI and ML development and deployment, including security and ethics-related considerations.
- Align with the elements of the Secure AI Framework (SAIF), which provides a practical approach to address the security and risk concerns of AI systems.
- Track the accuracy and effectiveness of AI-powered fraud detection systems, the reduction in false positives in security alerts, and the efficiency gains from AI-driven security automation.

Meet regulatory, compliance, and privacy needs

Financial services are subject to a vast array of regulations, including data residency requirements, specific audit trails, and data protection standards. To ensure that sensitive data is properly identified, protected, and managed, FSI organizations need robust data governance policies and data classification schemes. Consider the following recommendations to help you meet regulatory requirements:

- Set up data boundaries in Google Cloud for sensitive and regulated workloads by using Assured Workloads. Doing so helps you to meet government and industry-specific compliance requirements such as FedRAMP and CJIS.
- Identify, classify, and protect sensitive data, including financial information, by implementing Cloud Data Loss Prevention (Cloud DLP). Doing so helps you to meet data privacy regulations like GDPR and CCPA.
- Track details of administrative activities and access to resources by using Cloud Audit Logs. These logs are crucial for meeting audit requirements that are stipulated by many financial regulations.
- When you choose Google Cloud regions for your workloads and data, consider local regulations that are related to data residency. Google Cloud global infrastructure lets you choose regions that can help you to meet your data residency requirements.
- Manage the keys that are used to encrypt sensitive financial data at rest and in transit by using Cloud Key Management Service. Such encryption is a fundamental requirement of many security and privacy regulations.
- Implement the controls that are necessary to address your regulatory requirements. Validate that the controls work as expected. Get the controls validated again by an external auditor to prove to the regulator that your workloads are compliant with the regulations.

Prioritize security initiatives

Given the breadth of security requirements, financial institutions must prioritize initiatives that are based on risk assessment and regulatory mandates. We recommend the following phased approach:

1. **Establish a strong security foundation**: Focus on the core areas of security, including identity and access management, network security, and data protection. This focus helps to build a robust security posture and helps to ensure comprehensive defense against evolving threats.
2. **Address critical regulations**: Prioritize compliance with key regulations like PCI DSS, GDPR, and relevant national laws. Doing so helps to ensure data protection, mitigates legal risks, and builds trust with customers.
3. **Implement advanced security**: Gradually adopt advanced security practices like zero trust, AI-driven security solutions, and proactive threat hunting.

FSI perspective: Reliability

This document in the Google Cloud Well-Architected Framework: FSI perspective provides an overview of the principles and recommendations to design, deploy, and operate reliable financial services industry (FSI) workloads in Google Cloud. The document explores how to integrate advanced reliability practices and observability into your architectural blueprints. The recommendations in this document align with the reliability pillar of the Well-Architected Framework.

For financial institutions, reliable and resilient infrastructure is both a business need and a regulatory imperative. To ensure that FSI workloads in Google Cloud are reliable, you must understand and mitigate potential failure points, deploy resources redundantly, and plan for recovery. Operational resilience is an outcome of reliability. It's the ability to absorb, adapt to, and recover from disruptions. Operational resilience helps FSI organizations meet strict regulatory requirements. It also helps avoid intolerable harm to customers.

The key building blocks of reliability in Google Cloud are regions, zones, and the various location scopes of cloud resources: zonal, regional, multi-regional, global. You can improve availability by using managed services, distributing resources, implementing high-availability patterns, and automating processes.

Regulatory requirements

FSI organizations operate under strict reliability mandates by regulatory agencies such as the Federal Reserve System in the US, the European Banking Authority in the EU, and the Prudential Regulation Authority in the UK. Globally, regulators emphasize operational resilience, which is vital for financial stability and consumer protection. Operational resilience is the ability to withstand disruptions, recover effectively, and maintain critical services. This requires a harmonized approach for managing technological risks and dependencies on third parties.

The regulatory requirements across most jurisdictions have the following common themes:

- **Cybersecurity and technological resilience**: Strengthening defenses against cyber threats and ensuring the resilience of IT systems.
- **Third-party risk management**: Managing the risks associated with outsourcing services to providers of information and communication technology (ICT).
- **Business continuity and incident response**: Robust planning to maintain critical operations during disruptions and to recover effectively.
- **Protecting financial stability**: Ensuring the soundness and stability of the broader financial system.

The reliability recommendations in this document are mapped to the following core principles:

- Prioritize multi-zone and multi-region deployments
- Eliminate single points of failure (SPOFs)
- Understand and manage aggregate availability
- Implement a robust DR strategy
- Leverage managed services
- Automate the infrastructure provisioning and recovery processes

Prioritize multi-zone and multi-region deployments

For critical financial services applications, we recommend that you use a multi-region topology that's distributed across at least two regions and across three zones within each region. This approach is important for resilience against zone and region outages. Regulations often prescribe this approach, because if a failure occurs in one zone or region, most jurisdictions consider a severe disruption to a second zone to be a plausible consequence. The rationale is that when one location fails, the other location might receive an exceptionally high amount of additional traffic.

Consider the following recommendations to build resilience against zone and region outages:

- Prefer resources that have a wider locational scope. Where possible, use regional resources instead of zonal resources, and use multi-regional or global resources instead of regional resources. This approach helps to avoid the need to restore operations by using backups.
- In each region, leverage three zones rather than two. To handle failovers, overprovision capacity by a third more than the estimate.
- Minimize manual recovery steps by implementing active-active deployments like the following examples:
    - Distributed databases like Spanner provide built-in redundancy and synchronisation across regions.
    - The HA feature of Cloud SQL provides a topology that's near active-active, with read replicas across zones. It provides a recovery point objective (RPO) between regions that's close to 0.
- Distribute user traffic across regions by using Cloud DNS, and deploy a regional load balancer in each region. A global load balancer is another option that you can consider depending on your requirements and criticality. For more

information, see Benefits and risks of global load balancing for multi-region deployments.

- To store data, use multi-region services like Spanner and Cloud Storage.

Eliminate single points of failure

Distribute resources across different locations and use redundant resources to prevent any single point of failure (SPOF) from affecting the entire application stack.

Consider the following recommendations to avoid SPOFs:

- Avoid deploying just a single application server or database.
- Ensure automatic recreation of failed VMs by using managed instance groups (MIGs).
- Distribute traffic evenly across the available resources by implementing load balancing.
- Use HA configurations for databases such as Cloud SQL.
- Improve data availability by using regional persistent disks with synchronous replication.

For more information, see Design reliable infrastructure for your workloads in Google Cloud.

Understand and manage aggregate availability

Be aware that the overall or *aggregate* availability of a system is affected by the availability of each tier or component of the system. The number of tiers in an application stack has an inverse relationship with the aggregate availability of the stack. Consider the following recommendations for managing aggregate availability:

- Calculate the aggregate availability of a multi-tier stack by using the formula *tier1_availability* × *tier2_availability* × *tierN_availability*.

The following diagram shows the calculation of aggregate availability for a multi-tier system that consists of four services:

In the preceding diagram, the service in each tier provides 99.9% availability, but the aggregate availability of the system is lower at 99.6% (0.999 × 0.999 × 0.999 × 0.999). In general, the aggregate availability of a multi-tier stack is lower than the availability of the tier that provides the least availability.

- Where feasible, choose *parallelization* over *chaining*. With parallelized services, the end-to-end availability is higher than the availability of each individual service.

The following diagram shows two services, A and B, that are deployed by using the chaining and parallelization approaches:

In the preceding examples, both services have an SLA of 99%, which results in the following aggregate availability depending on the implementation approach:

  - **Chained services** yield an aggregate availability of only 98% (.99 × .99).
  - **Parallelized services** yield a higher aggregate availability at 99.99% because each service runs independently and individual services aren't affected by the availability of the other services. The formula for aggregated parallelized services is *1 − (1 − A) × (1 − B)*.

- Choose Google Cloud services with uptime SLAs that can help meet the required level of overall uptime for your application stack.
- When you design your architecture, consider the trade-offs between availability, operational complexity, latency, and cost. Increasing the number of nines of availability generally costs more, but doing so helps you meet regulatory requirements.

For example, 99.9% availability (three nines) means a potential downtime of 86 seconds in a 24-hour day. In contrast, 99% (two nines) means a downtime of 864 seconds over the same period, which is 10 times more downtime than with three nines of availability.

For critical financial services, the architecture options might be limited. However, it's critical to identify the availability requirements and accurately calculate availability. Performing such an assessment helps you to assess the implications of your design decisions on your architecture and budget.

Implement a robust DR strategy

Create well-defined plans for different disaster scenarios, including zonal and regional outages. A well-defined disaster recovery (DR) strategy lets you recover from a disruption and resume normal operations with minimal impact.

DR and high availability (HA) are different concepts. With cloud deployments, in general, DR applies to multi-region deployments and HA applies to regional deployments. These deployment archetypes support different replication mechanisms.

- **HA**: Many managed services provide synchronous replication between zones within a single region by default. Such services support a zero or near-zero recovery time objective (RTO) and recovery point objective (RPO). This support lets you create an active-active deployment topology that doesn't have any SPOF.
- **DR**: For workloads that are deployed across two or more regions, if you don't use multi-regional or global services, you must define a replication strategy. The replication strategy is typically asynchronous. Carefully assess how such replication affects the RTO and RPO for critical applications. Identify the manual or semi-automated operations that are necessary for failover.

For financial institutions, your choice of failover region might be limited by regulations about data sovereignty and data residency. If you need an active-active topology across two regions, we recommend that you choose managed multi-regional services, like Spanner and Cloud Storage, especially when data replication is critical.

Consider the following recommendations:

- Use managed multi-regional storage services for data.
- Take snapshots of data in persistent disks and store the snapshots in multi-region locations.
- When you use regional or zonal resources, set up data replication to other regions.
- Validate that your DR plans are effective by testing the plan regularly.
- Be aware of the RTO and RPO and their correlation to the impact tolerance that's stipulated by financial regulations in your jurisdiction.

For more information, see Architecting disaster recovery for cloud infrastructure outages.

Leverage managed services

Whenever possible, use managed services to take advantage of the built-in features for backups, HA, and scalability. Consider the following recommendations for using managed services:

- Use managed services in Google Cloud. They provide HA that's backed by SLAs. They also offer built-in backup mechanisms and resilience features.
- For data management, consider services like Cloud SQL, Cloud Storage, and Spanner,

- For compute and application hosting, consider Compute Engine managed instance groups (MIGs) and Google Kubernetes Engine (GKE) clusters. Regional MIGs and GKE regional clusters are resilient to zone outages.
- To improve resilience against region outages, use managed multi-regional services.
- Identify the need for *exit plans* for services that have unique characteristics and define the required plans. Financial regulators like the FCA, PRA, and EBA require firms to have strategies and contingency plans for data retrieval and operational continuity if the relationship with a cloud provider ends. Firms must assess the exit feasibility before entering into cloud contracts and they must maintain the ability to change providers without operational disruption.
- Verify that the services that you choose support exporting data to an open format like CSV, Parquet, and Avro. Verify whether the services are based on open technologies, like GKE support for the Open Container Initiative (OCI) format or Cloud Composer built on Apache Airflow.

Automate the infrastructure provisioning and recovery processes

Automation helps to minimize human errors and helps to reduce the time and resources that are necessary to respond to incidents. The use of automation can help to ensure faster recovery from failures and more consistent results. Consider the following recommendations to automate how you provision and recover resources:

- Minimize human errors by using infrastructure as code (IaC) tools like Terraform.
- Reduce manual intervention by automating failover processes. Automated responses can also help to reduce the impact of failures. For example, you can use Eventarc or Workflows to automatically trigger remedial actions in response to issues observed through audit logs.
- Increase the capacity of your cloud resources during failover by using autoscaling.
- Automatically apply policies and guardrails for regulatory requirements across your cloud topology during service deployment by adopting platform engineering.

FSI perspective: Cost optimization

This document in the Google Cloud Well-Architected Framework: FSI perspective provides an overview of principles and recommendations to optimize the cost of your financial services industry (FSI) workloads in Google Cloud. The recommendations in this document align with the cost optimization pillar of the Well-Architected Framework.

Robust cost optimization for financial services workloads requires the following fundamental elements:

- The ability to identify wasteful versus value-driving resource utilization.
- An embedded culture of financial accountability.

To optimize cost, you need a comprehensive understanding of the cost drivers and resource needs across your organization. In some large organizations, especially those that are early in the cloud journey, a single team is often responsible for optimizing spend across a large number of domains. This approach assumes that a central team is best placed to identify high-value opportunities to improve efficiency.

The centralized approach might yield some success during the initial stages of cloud adoption or for non-critical workloads. However, a single team can't drive cost optimization across an entire organization. When the resource usage or the level of regulatory scrutiny increases, the centralized approach isn't sustainable. Centralized teams face scalability challenges particularly when dealing with a large number of financial products and services. The project teams that own the products and services might resist changes that are made by an external team.

For effective cost optimization, spend-related data must be highly visible, and engineers and other cloud users who are close to the workloads must be motivated to take action to optimize cost. From an organizational standpoint, the challenge for cost optimization is to identify what areas should be optimized, identify the engineers who are responsible for those areas, and then convince them to take the required optimization action. This document provides recommendations to address this challenge.

The cost optimization recommendations in this document are mapped to the following core principles:

- Identify waste by using Google Cloud tools
- Identify value by analyzing and enriching spend data
- Allocate spend to drive accountability
- Drive accountability and motivate engineers to take action
- Focus on value and TCO rather than cost

Identify waste by using Google Cloud tools

Google Cloud provides several products, tools, and features to help you identify waste. Consider the following recommendations.

Use automation and AI to systematically identify what to optimize

Active Assist provides intelligent recommendations across services that are critical to FSI, such as Cloud Run for microservices, BigQuery for data analytics, Compute Engine for core applications, and Cloud SQL for relational databases. Active Assist recommendations are provided at no cost and without any configuration by you. The recommendations help you to identify idle resources and underutilized commitments.

Centralize FinOps monitoring and control through a unified interface

Cloud Billing reports and the FinOps hub let you implement comprehensive cost monitoring. This comprehensive view is vital for financial auditors and internal finance teams to track cloud spend, assess the financial posture, evaluate FinOps maturity across various business units or cost centers, and provide a consistent financial narrative.

Identify value by analyzing and enriching spend data

Active Assist is effective at identifying obvious waste. However, pinpointing value can be more challenging, particularly when workloads are on unsuitable products or when the workloads lack clear alignment with business value. For FSI workloads, business value extends beyond cost reduction. The value includes risk mitigation, regulatory adherence, and gaining competitive advantages.

To understand cloud spend and value holistically, you need a complete understanding at multiple levels: where the spend is coming from, what business function the spend is driving, and the technical feasibility of refactoring or optimizing the workload in question.

The following diagram shows how you can apply the data-information-knowledge-wisdom (DIKW) pyramid and Google Cloud tools to get a holistic understanding of cloud costs and value.

The preceding diagram shows how you can use the DIKW approach to refine raw cloud spending data into actionable insights and decisions that drive business value.

- **Data**: In this layer, you collect raw, unprocessed streams of usage and cost data for your cloud resources. Your central FinOps team uses tools like Cloud Billing invoices, billing exports, and Cloud Monitoring to get granular, detailed data. For example, a data point could be that a VM named app1-test-vmA ran for 730 hours in the us-central1 region and cost USD 70.
- **Information**: In this layer, your central FinOps team uses tools like Cloud Billing reports and the FinOps Hub to structure the raw data to help answer questions like "What categories of resources are people spending money on?" For example,

you might find out that a total of USD 1,050 was spent on VMs of the machine type n4-standard-2 across two regions in the US.

- **Knowledge**: In this layer, your central FinOps team enriches information with appropriate business context about *who* spent money and for *what purpose*. You use mechanisms like tagging, labeling, resource hierarchy, billing accounts, and custom Looker dashboards. For example, you might determine that the app1 testing team in the US spent USD 650 during the second week of July as part of a stress testing exercise.
- **Wisdom**: In this layer, your product and application teams use the contextualized knowledge to assess the business value of cloud spending and to make informed, strategic decisions. Your teams might answer questions like the following:
    - Is the USD 5,000 that was spent on a data analytics pipeline generating business value?
    - Could we re-architect the pipeline to be more efficient without reducing performance?

Consider the following recommendations for analyzing cloud spend data.

Analyze spend data that's provided by Google Cloud

Start with detailed Cloud Billing data that's exported to BigQuery and data that's available in Monitoring logs. To derive actionable insights and make decisions, you need to structure this data and enrich it with business context.

Visualize data through available tooling

Augment the built-in Google Cloud dashboards with custom reporting by using tools like Looker Studio on top of BigQuery exports. Finance teams can build custom dashboards that contextualize cloud spend against financial metrics, regulatory reporting requirements, and business unit profitability. They can then provide a clear financial narrative for analysis and decision making by executive stakeholders.

Allocate spend to drive accountability

After you understand what's driving the cloud spend, you need to identify who is spending money and why. This level of understanding requires a robust cost-allocation practice, which involves attaching business-relevant metadata to cloud resources. For example, if a particular resource is used by the Banking-AppDev team, you can attach a tag like team=banking_appdev to the resource to track the cost that the team incurs on that resource. Ideally, you should allocate 100% of your cloud costs to the source of the spending. In practice, you might start with a lower target because building a metadata structure to support 100% cost allocation is a complex effort.

Consider the following recommendations to develop a metadata strategy to support cost allocation:

- **Validity**: Ensure that the tags help to identify business-related key performance indicators (KPIs) and regulatory requirements. This association is critical for internal chargebacks, regulatory reporting, and aligning cloud spend with business-unit goals. For example, the following tags clearly identify a spending team, their region, and the product that they work on: team=banking_appdev, region=emea, product=frontend.
- **Automation**: To achieve a high level of tagging compliance, enforce tagging through automation. Manual tagging is prone to errors and inconsistency, which are unacceptable in FSI environments where auditability and financial accuracy are paramount. Automated tagging ensures that resources are correctly categorized when they're created.
- **Simplicity**: Measure simple, uncorrelated factors. FSI environments are complex. To ensure that cost-allocation rules in such an environment are easy to understand and enforce, the rules must be as simple as possible. Avoid overengineering the rules for highly specific (*edge*) cases. Complex rules can lead to confusion and resistance from operational teams.

After you define an allocation strategy by using tags, you need to decide the level of granularity at which the strategy should be implemented. The required granularity depends on your business needs. For example, some organizations might need to track cost at the product level, some might need cost data for each cost center, and others might need cost data per environment (development, staging, and production).

Consider the following approaches to achieve the appropriate level of cost-allocation granularity for your organization:

- Use the project hierarchy in Google Cloud as a natural starting point for cost allocation. Projects represent points of policy enforcement in Google Cloud. By default, IAM permissions, security policies, and cost are attributed to projects and folders. When you review cost data that's exported from Cloud Billing, you can view the folder hierarchy and the projects that are associated with the cost data. If your Google Cloud resource hierarchy reflects your organization's accountability structure for spend, then this is the simplest way to implement cost allocation.
- Use tags and labels for additional granularity. They provide flexible ways to categorize resources in billing exports. Tags and labels facilitate detailed cost breakdowns by application and environment.

Often, you might need to use the project hierarchy combined with tagging and labeling for effective cost allocation. Regardless of the cost-allocation approach that you choose, follow the recommendations that were described earlier for developing a robust metadata strategy: validitation, automation, and simplicity.

Drive accountability and motivate engineers to take action

The cloud FinOps team is responsible for driving an organization to be conscious of costs and value. The individual product teams and engineering teams must take the required actions for cost optimization. These teams are also accountable for the cost behavior of the financial services workloads and for ensuring that their workloads provide the required business value.

Consider the following recommendations to drive accountability and motivate teams to optimize cost.

Establish a centralized FinOps team for governance

Cloud FinOps practices don't grow organically. A dedicated FinOps team must define and establish FinOps practices by doing the following:

- Build the required processes, tools, and guidance.
- Create, communicate, and enforce the necessary policies, such as mandatory tagging, budget reviews, and optimization processes.
- Encourage engineering teams to be accountable for cost.
- Intervene when the engineering teams don't take on ownership for costs.

Get executive sponsorship and mandates

Senior leadership, including the CTO, CFO, and CIO, must actively champion an organization-wide shift to a FinOps culture. Their support is crucial for prioritizing cost accountability, allocating resources for the FinOps program, ensuring cross-functional participation, and driving compliance with FinOps requirements.

Incentivize teams to optimize cost

Engineers and engineering teams might not be self-motivated to focus on cost optimization. It's important to align team and individual goals with cost efficiency by implementing incentives such as the following:

- Reinvest a portion of the savings from cost optimization in the teams that achieved the optimization.
- Publicly recognize and celebrate cost optimization efforts and successes.
- Use gamification techniques to reward teams that effectively optimize cost.

- Integrate efficiency metrics into performance goals.

Implement showback and chargeback techniques

Ensure that teams have clear visibility into the cloud resources and costs that they own. Assign financial responsibility to the appropriate individuals within the teams. Use formal mechanisms to enforce rigorous tagging and implement transparent rules for allocating shared costs.

Focus on value and TCO rather than cost

When you evaluate cloud solutions, consider the long-term total cost of ownership (TCO). For example, self-hosting a database for an application might seem to be cheaper than using a managed database service like Cloud SQL. However, to assess the long-term value and TCO, you must consider the hidden costs that are associated with self-hosted databases. Such costs include the dedicated engineering effort for patching, scaling, security hardening, and disaster recovery, which are critical requirements for FSI workloads. Managed services provide significantly higher long-term value, which offsets the infrastructure costs. Managed services provide robust compliance capabilities, have built-in reliability features, and can help to reduce your operational overhead.

Consider the following recommendations to focus on value and TCO.

Use product-specific techniques and tools for resource optimization

Leverage cost-optimization tools and features that are provided by Google Cloud products, such as the following:

- Compute Engine: Autoscaling, custom machine types, and Spot VMs
- GKE: Cluster autoscaler and node auto-provisioning
- Cloud Storage: Object Lifecycle Management and Autoclass
- BigQuery: Capacity-based pricing and cost-optimization techniques
- Google Cloud VMware Engine: committed use discounts (CUDs), optimized storage, and other cost optimization strategies

Take advantage of discounts

Ensure that the billing rate for your cloud resources is as low as possible by using discounts that Google offers. The individual product and engineering teams typically manage resource optimization. The central FinOps team is responsible for optimizing billing rates because they have visibility into resource requirements across the entire organization. Therefore, they can aggregate the requirements and maximize the commitment-based discounts.

You can take advantage of the following types of discounts for Google Cloud resources:

- Enterprise discounts are negotiated discounts based on your organization's commitment to a minimum total spend on Google Cloud at a reduced billing rate.
- Resource-based CUDs are in exchange for a commitment to use a minimum quantity of Compute Engine resources over a one-year period or a three-year period. Resource-based CUDs are applicable to the resources that are in a specific project and region. To share CUDs across multiple projects, you can enable discount sharing.
- Spend-based CUDs are in exchange for a commitment to spend a minimum amount of money on a particular product over a one-year period or a three-year period. Spend-based discounts are applicable at the billing account level. The discounts are applied regionally or globally depending on the product.

You can achieve significant savings by using CUDs on top of enterprise discounts.

In addition to CUDs, use the following approaches to reduce billing rates:

- Use Spot VMs for fault-tolerant and flexible workloads. Spot VMs are more than 80% cheaper than regular VMs.
- BigQuery offers multiple pricing models, which include on-demand pricing and edition-based pricing that's based on commitments and autoscaling requirements. If you use a significant volume of BigQuery resources, choose an appropriate edition to reduce the cost per slot for analytics workloads.
- Carefully evaluate the available Google Cloud regions for the services that you need to use. Choose regions that align with your cost objectives and factors like latency and compliance requirements. To understand the trade-offs between cost, sustainability, and latency, use the Google Cloud Region Picker.

FSI perspective: Performance optimization

This document in the Google Cloud Well-Architected Framework: FSI perspective provides an overview of principles and recommendations to optimize the performance of your financial services industry (FSI) workloads in Google Cloud. The recommendations in this document align with the performance optimization pillar of the Well-Architected Framework.

Performance optimization has a long history in financial services. It has helped FSI organizations surpass technical challenges and it's nearly always been an enabler or accelerator for the creation of new business models. For example, ATMs (introduced in 1967) automated the cash dispensation process and they helped banks to decrease the cost of their core business. Techniques like bypassing the OS kernel and pinning application threads to compute cores helped to achieve deterministic and low latency for trading applications. The reduction in latency facilitated higher and firmer liquidity with tighter spreads in the financial markets.

The cloud creates new opportunities for performance optimization. It also challenges some of the historically accepted optimization patterns. Specifically, the following trade-offs are more transparent and controllable in the cloud:

- Time to market versus cost.
- End-to-end performance at the system level versus performance at the node level.
- Talent availability versus agility of technology-related decision making.

For example, adapting hardware and IT resources to specific skill requirements is a trivial task in the cloud. To support GPU programming, you can easily create GPU-based VMs. You can scale capacity in the cloud to accommodate demand spikes without over-provisioning resources. This capability helps to ensure that your workloads can handle peak loads, such as on nonfarm payroll days and when trading volumes are significantly greater than historical levels. Instead of spending on writing highly optimized code at the level of individual servers (like highly fine-tuned code in the C language) or writing code for conventional high performance computing (HPC) environments, you can scale out optimally by using a well-architected Kubernetes-based distributed system.

The performance optimization recommendations in this document are mapped to the following core principles:

- Align technology performance metrics with key business indicators
- Prioritize security without sacrificing performance for unproven risks
- Rethink your architecture to adapt to new opportunities and requirements
- Future-proof your technology to meet present and future business needs

Align technology performance metrics with key business indicators

You can map performance optimization to business-value outcomes in several ways. For example, in a buy-side research desk, a business objective could be to optimize the output per research hour or to prioritize experiments from teams that have a proven track record, such as higher Sharpe ratios. On the sell side, you can use analytics to track client interest and accordingly prioritize the throughput to AI models that support the most interesting research.

Connecting performance goals to business key performance indicators (KPIs) is also important for funding performance improvements. Business innovation and transformation initiatives (sometimes called *change-the-bank* efforts) have different budgets and they have potentially different degrees of access to resources when compared to business-as-usual (BAU) or *run-the-bank* operations. For example, Google Cloud helped the risk management and technology teams of a G-SIFI to collaborate with the front-office quantitative analysts on a solution to perform risk analytics calculations (such as XVA) in minutes instead of hours or days. This solution helped the organization to meet relevant compliance requirements. It also enabled the traders to have higher quality conversations with their clients, potentially offering tighter spreads, firmer liquidity, and more cost-effective hedging.

When you align your performance metrics with business indicators, consider the following recommendations:

- Connect each technology initiative to the relevant business objectives and key results (OKRs), such as increasing revenue or profit, reducing costs, and mitigating risk more efficiently or holistically.
- Focus on optimizing performance at the system level. Look beyond the conventional change-the-bank versus run-the-bank separation and the front-office versus back-office silos.

Prioritize security without sacrificing performance for unproven risks

Security and regulatory compliance in FSI organizations must be unequivocally of a high standard. Maintaining a high standard is essential to avoid losing clients and to prevent irreparable damage to an organization's brand. Often, the highest value is derived through technology innovations such as generative AI and unique, managed services like Spanner. Don't automatically discard such technology options due to a blanket misconception about prohibitive operational risk or inadequate regulatory compliance posture.

Google Cloud has worked closely with G-SIFIs to make sure that an AI-based approach for Anti-Money Laundering (AML) can be used across the jurisdictions where the institutions serve customers. For example, HSBC significantly enhanced the performance of its financial crime (Fincrime) unit with the following results:

- Nearly two to four times more confirmed suspicious activity.
- Lower operational costs due to elimination of over 60% of false positives and focused investigation time only on high-risk, actionable alerts.
- Auditable and explainable outputs to support regulatory compliance.

Consider the following recommendations:

- Confirm that the products that you intend to use can help meet the security, resilience, and compliance requirements for the jurisdictions where you operate.

To achieve this objective, work with Google Cloud account teams, risk teams, and product teams.

- Create more powerful models *and* provide transparency to customers by leveraging AI explainability (for example, Shapley value attribution). Techniques like Shapley value attribution can attribute model decisions to particular features at the input level.
- Achieve transparency for generative AI workloads by using techniques like citations to sources, grounding, and RAG.
- When explainability isn't enough, separate out the decision making steps in your value streams and use AI to automate only the non-decision making steps. In some cases, explainable AI might not be sufficient or a process might require human intervention due to regulatory concerns (for example, GDPR, Article 22). In such cases, present all the information that the human agent needs for decision making in a single control pane, but automate the data gathering, ingestion, manipulation, and summarization tasks.

Rethink your architecture to adapt to new opportunities and requirements

Augmenting your current architectures with cloud-based capabilities can provide significant value. To achieve more transformative outcomes, you need to periodically rethink your architecture by using a cloud-first approach.

Consider the following recommendations to periodically rethink the architecture of your workloads to further optimize performance.

Use cloud-based alternatives to on-premises HPC systems and schedulers

To take advantage of higher elasticity, improved security posture, and extensive monitoring and governance capabilities, you can run HPC workloads in the cloud or burst on-premises workloads to the cloud. However, for certain numerical modeling use cases like simulation of investment strategies or XVA modeling, combining Kubernetes with Kueue might offer a more powerful solution.

Switch to graph-based programming for simulations

Monte Carlo simulations might be much more performant in a graph-based execution system such as Dataflow. For example, HSBC uses Dataflow to run risk calculations 16 times faster compared to their previous approach.

Run cloud-based exchanges and trading platforms

Conversations with Google Cloud customers reveal that the 80/20 Pareto principle applies to the performance requirements of markets and trading applications.

- More than 80% of trading applications don't need extremely low latency. However, they get significant benefits from the resilience, security, and elasticity capabilities of the cloud. For example, BidFX, a foreign exchange multi-dealer platform uses the cloud to launch new products quickly and to significantly increase their availability and footprint without increasing resources.
- The remaining applications (less than 20%) need low latency (less than a millisecond), determinism, and fairness in the delivery of messages. Conventionally, these systems run in rigid and expensive colocated facilities. Increasingly, even this category of applications is being replatformed on the cloud, either at the edge or as cloud-first applications.

Future-proof your technology to meet present and future business needs

Historically, many FSI organizations built proprietary technologies to gain a competitive edge. For example, in the early 2000s, successful investment banks and trading firms had their own implementations of foundational technologies such as pub-sub systems and message brokers. With the evolution of open source technologies and the cloud, such technologies have become commodities and don't offer incremental business value.

Consider the following recommendations to future-proof your technology.

Adopt a data-as-a-service (DaaS) approach for faster time to market and cost transparency

FSI organizations often evolve through a combination of organic growth and mergers and acquisitions (M&A). As a result, the organizations need to integrate disparate technologies. They also need to manage duplicate resources, such as data vendors, data licenses, and integration points. Google Cloud provides opportunities to create differentiated value in post-merger integrations.

For example, you can use services like BigQuery sharing to build an analysis-ready data-as-a-service (DaaS) platform. The platform can provide both market data and inputs from alternative sources. This approach eliminates the need to build redundant data pipelines and it lets you focus on more valuable initiatives. Further, the merged or acquired companies can quickly and efficiently rationalize their post-merger data licensing and infrastructure needs. Instead of spending effort on adapting and merging legacy data estates and operations, the combined business can focus on new business opportunities.

Build an abstraction layer to isolate existing systems and address emerging business models

Increasingly, the competitive advantage for banks isn't the core banking system but their customer experience layer. However, legacy banking systems often use monolithic applications that were developed in languages like Cobol and are integrated across the entire banking value chain. This integration made it difficult to separate the layers of the value chain, so it was nearly impossible to upgrade and modernize such systems.

One solution to address this challenge is to use an isolation layer such as an API management system or a staging layer like Spanner that duplicates the book of record and facilitates the modernization of services with advanced analytics and AI. For example, Deutsche Bank used Spanner to isolate their legacy core banking estate and start their innovation journey.

Google Cloud deployment archetypes

Release Notes

Last reviewed 2024-11-20 UTC

As a cloud architect or decision maker, when you plan to deploy an application in Google Cloud, you need to choose a *deployment archetype*[1] that's suitable for your application. This guide describes six deployment archetypes—zonal, regional, multi-regional, global, hybrid, and multicloud, and presents use cases and design considerations for each deployment archetype. The guide also provides a comparative analysis to help you choose the deployment archetypes that meet your requirements for availability, cost, performance, and operational efficiency.

What is a deployment archetype?

A deployment archetype is an abstract, provider-independent model that you use as the foundation to build application-specific *deployment architectures* that meet your business and technical requirements. Each deployment archetype specifies a combination of failure domains where an application can run. These failure domains can be one or more Google Cloud zones or regions, and they can extend to include your on-premises data centers or failure domains in other cloud providers.

The following diagram shows six applications deployed in Google Cloud. Each application uses a deployment archetype that meets its specific requirements.

As the preceding diagram shows, in an architecture that uses the hybrid or multicloud deployment archetype, the cloud topology is based on one of the *basic* archetypes: zonal, regional, multi-regional, or global. In this sense, the hybrid and multicloud deployment archetypes can be considered as *composite* deployment archetypes that include one of the basic archetypes.

Note: Deployment archetypes are different from location scopes. The location scope of a Google Cloud resource defines its availability boundary. For example, the location scope of a Compute Engine VM is *zonal*. This means that if the Google Cloud zone in which a VM is provisioned has an outage, the availability of the VM is affected. However, by distributing VMs across multiple zones, you can build a highly available architecture that's based on the *regional* deployment archetype.

Choosing a deployment archetype helps to simplify subsequent decisions regarding the Google Cloud products and features that you should use. For example, for a highly available containerized application, if you choose the regional deployment archetype, then regional Google Kubernetes Engine (GKE) clusters are more appropriate than zonal GKE clusters.

When you choose a deployment archetype for an application, you need to consider tradeoffs between factors like availability, cost, and operational complexity. For example, if an application serves users in multiple countries and needs high availability, you might choose the multi-regional deployment archetype. But for an internal application that's used by employees in a single geographical region, you might prioritize cost over availability and, therefore, choose the regional deployment archetype.

Overview of the deployment archetypes

The following tabs provide definitions for the deployment archetypes and a summary of the use cases and design considerations for each.

ZonalRegionalMulti-regionalGlobalHybridMulticloud

Your application runs within a single Google Cloud zone, as shown in the following diagram:

Use cases	• Development and test environments. • Applications that don't need high availability. • Low-latency networking between application components. • Migrating commodity workloads. • Applications that use license-restricted software.
Design considerations	• Downtime during zone outages. For business continuity, you can provision a passive replica of the application in another zone in the same region. If a zone outage occurs, you can restore the application to production by using the passive replica.

More information	See the following sections: • Zonal deployment archetype • Comparative analysis of all the deployment archetypes

Contributors

Author: Kumar Dhanagopal | Cross-Product Solution Developer

Other contributors:

- Anna Berenberg | Engineering Fellow
- Anshu Kak | Distinguished Engineer
- Jeff Welsch | Director, Product Management
- Marwan Al Shawi | Partner Customer Engineer
- Sekou Page | Outbound Product Manager
- Steve McGhee | Reliability Advocate
- Victor Moreno | Product Manager, Cloud Networking

---

1. Anna Berenberg and Brad Calder, Deployment Archetypes for Cloud Applications, ACM Computing Surveys, Volume 55, Issue 3, Article No.: 61, pp 1-48 ↩

Google Cloud zonal deployment archetype

Last reviewed 2024-11-20 UTC

This section of the Google Cloud deployment archetypes guide describes the zonal deployment archetype.

In a cloud architecture that uses the basic zonal deployment archetype, the application runs in a single Google Cloud zone, as shown in the following diagram:

To be able to recover from zone outages, you can use a dual-zone architecture where a passive replica of the application stack is provisioned in a second (failover) zone, as shown in the following diagram:

If an outage occurs in the primary zone, you can promote the standby database to be the primary (write) database and update the load balancer to send traffic to the frontend in the failover zone.

Note: For more information about region-specific considerations, see Geography and regions.

Use cases

The following are examples of use cases for which the zonal deployment archetype is an appropriate choice:

- **Cloud development and test environments**: You can use the zonal deployment archetype to build a low-cost environment for development and testing.
- **Applications that don't need high availability**: The zonal deployment archetype might be sufficient for applications that can tolerate downtime.
- **Low-latency networking between application components**: A single-zone architecture might be well suited for applications such as batch computing that need low-latency and high-bandwidth network connections among the compute nodes.
- **Migration of commodity workloads**: The zonal deployment archetype provides a cloud migration path for commodity on-premises apps for which you have no control over the code or that can't support architectures beyond a basic active-passive topology.
- **Running license-restricted software**: The zonal deployment archetype might be well suited for license-restricted systems where running more than one instance at a time is either too expensive or isn't permitted.

Design considerations

When you build an architecture that's based on the zonal deployment archetype, consider the potential downtime during zone and region outages.

Zone outages

If the application runs in a single zone with no failover zone, then when a zone outage occurs, the application can't serve requests. To prevent this situation, you must maintain a passive replica of the infrastructure stack in another (failover) zone in the same region. If an outage occurs in the primary zone, you can promote the database in the failover zone to be the primary database, and ensure that incoming traffic is routed to the frontend in the failover zone. After Google resolves the outage, you can choose to either *fail back* to the primary zone or make it the new failover zone.

Note: For more information about region-specific considerations, see Geography and regions.

Region outages

If a region outage occurs, you must wait for Google to resolve the outage and then verify that the application works as expected. If you need robustness against region outages, consider using the multi-regional deployment archetype.

Reference architecture

For a reference architecture that you can use to design a zonal deployment on Compute Engine VMs, see Single-zone deployment on Compute Engine.

Google Cloud regional deployment archetype

Last reviewed 2024-11-14 UTC

This section of the Google Cloud deployment archetypes guide describes the regional deployment archetype.

In a cloud architecture that uses the regional deployment archetype, instances of the application run in two or more zones within a single Google Cloud region. All the application instances use a centrally managed, shared repository of configuration files. Application data is replicated synchronously across all the zones in the architecture.

The following diagram shows the cloud topology for a highly available application that runs independently in three zones within a single Google Cloud region:

The preceding diagram shows an application with frontend and backend components that run independently in three zones in a Google Cloud region. An external load balancer forwards user requests to one of the frontends. An internal load balancer forwards traffic from the frontends to the backends. The application uses a database that's replicated across the zones. If a zone outage occurs, the database fails over to a replica in another zone.

The topology in the preceding diagram is robust against zone outages, but not against region outages. To be able to recover from region outages, you must have deployed a passive replica of the application in a second (failover) region, as shown in the following diagram:

When an outage occurs in the primary region, you must promote the database in the failover region, and let the DNS routing policies route traffic to the load balancer in the failover region.

To optimize the cost of the failover infrastructure, you can operate the failover region at a lower capacity by deploying fewer resources.

Use cases

The following sections provide examples of use cases for which the regional deployment archetype is an appropriate choice.

Highly available application with users within a geographic area

We recommend the regional deployment archetype for applications that need robustness against zone outages but can tolerate some downtime caused by region outages. If any part of the application stack fails, the application continues to run if at least one functioning component with adequate capacity exists in every tier. If a zone outage occurs, the application stack continues to run in the other zones.

Low latency for application users

If users of an application are within a geographic area, such as a single country, the regional deployment archetype can help improve the user-perceived performance of the application. You can optimize network latency for user requests by deploying the application in the Google Cloud region that's closest to your users.

Low-latency networking between application components

A single-region architecture might be well suited for applications such as batch computing that need low-latency and high-bandwidth network connections among the compute nodes. All resources are in a single Google Cloud region, so inter-resource network traffic remains within the region. The inter-resource network latency is low, and you don't incur cross-region data transfer costs. Intra-region network costs still apply.

Compliance with data residency and sovereignty requirements

The regional deployment archetype can help you meet regulatory requirements for data residency and operational sovereignty. For example, a country in Europe might require that all user data be stored and accessed in data centers that are located physically within the country. To help meet this requirement, you can deploy the application to a Google Cloud region in Europe.

Design considerations

When you build an architecture that's based on the regional deployment archetype, consider the following design factors.

Downtime during region outages

When a region outage occurs, the application is down. You can reduce the downtime caused by region outages by maintaining a passive (failover) replica of the infrastructure stack in another Google Cloud region. If an outage occurs in the primary region, you can activate the stack in the failover region and use DNS routing policies to route traffic to the load balancer in the failover region.

Cost of redundant resources

A multi-zone architecture typically has more cloud resources than a single-zone deployment. Consider the cost of these cloud resources when you build your architecture. For applications that need robustness against zone outages, the availability advantage of a multi-zone architecture might justify the higher cost.

Note: For more information about region-specific considerations, see Geography and regions.

Reference architecture

For a reference architecture that you can use to design a regional deployment on Compute Engine VMs, see Regional deployment on Compute Engine.

Google Cloud multi-regional deployment archetype

Last reviewed 2024-11-20 UTC

This section of the Google Cloud deployment archetypes guide describes the multi-regional deployment archetype.

In a cloud architecture that uses the multi-regional deployment archetype, the application runs in two or more Google Cloud regions. Application data is replicated across all the regions in the architecture. To ensure fast and synchronous replication of data, the regions are typically within a continent.

The following diagram shows the cloud topology for an application that runs in two Google Cloud regions:

Multi-regional deployment archetype.

The preceding diagram shows two isolated multi-tier application stacks that run independently in two Google Cloud regions. In each region, the application runs in three zones. The databases in the two regions are replicated. If the workload has a low recovery point objective (RPO) or if it requires strong cross-region consistency of data, then the database replication needs to be synchronous. Otherwise, the databases can be replicated asynchronously. User requests are routed to regional load balancers by using a DNS routing policy. If an outage occurs in any one of the two regions, DNS routes user requests to the load balancer in the other region.

Use cases

The following sections provide examples of use cases for which the multi-regional deployment archetype is an appropriate choice.

High availability for geographically dispersed users

We recommend a multi-regional deployment for applications that are business-critical and where high availability and robustness against region outages are essential. If a region becomes unavailable for any reason (even a large-scale disruption caused by a natural disaster), users of the application don't experience any downtime. Traffic is routed to the application in the other available regions. If data is replicated synchronously, the recovery time objective (RTO) is near zero.

Low latency for application users

If your users are within a specific geographical area, such as a continent, you can use a multi-regional deployment to achieve an optimal balance between availability and performance. When one of the regions has an outage, the global load balancer sends requests that originate in that region to another region. Users don't perceive significant performance impact because the regions are within a geographical area.

Compliance with data residency and sovereignty requirements

The multi-regional deployment archetype can help you meet regulatory requirements for data residency and operational sovereignty. For example, a country in Europe might require that all user data be stored and accessed in data centers that are located physically within the country. You can deploy the application to Google Cloud regions in Europe and use DNS with a geofenced routing policy to route traffic to the appropriate region.

Design considerations

When you provision and manage redundant resources across locations, the volume of cross-location network traffic can be high. You also store and replicate data across multiple regions. When you build an architecture that uses the multi-regional deployment archetype, consider the potentially higher cost of cloud resources and network traffic, and the complexity of operating the deployment. For business-critical applications, the availability advantage of a multi-region architecture might outweigh the increased cost and operational complexity.

Reference architecture

For a reference architecture that you can use to design a multi-regional deployment on Compute Engine VMs, see Multi-regional deployment on Compute Engine.

Google Cloud global deployment archetype

Last reviewed 2024-11-20 UTC

This section of the Google Cloud deployment archetypes guide describes the global deployment archetype.

In an architecture that's based on the global deployment archetype, the application runs in multiple Google Cloud regions across the globe. You can deploy the application either as a distributed location-unaware stack or as multiple regionally isolated stacks. In either case, a global anycast load balancer distributes traffic to the appropriate region. The application writes data to, and reads from, a synchronously replicated database that's available in all the regions, like Spanner with multi-region configuration. Other components of the application stack can also be global, such as the cache and object store.

The following diagram shows the distributed location-unaware variant of the global deployment archetype:

Global deployment archetype with a globally distributed application stack.

The preceding diagram shows a location-unaware application stack, with frontend and backend instances (typically microservices) that are distributed across multiple zones in three Google Cloud regions. A global anycast load balancer distributes incoming traffic to an appropriate frontend instance. This distribution is based on the availability and capacity of the instances and their geographical proximity to the source of the traffic. Cross-region internal load balancers distribute traffic from the frontend instances to appropriate backend instances based on their availability and capacity. The application uses a database that is synchronously replicated and available across regions.

The following diagram shows a variant of the global deployment archetype with regionally isolated application stacks:

Global deployment archetype with regionally isolated application stacks.

The preceding diagram shows regionally isolated application stacks that run in multiple zones in two Google Cloud regions. This topology is similar to the multi-regional deployment archetype, but it uses a global anycast load balancer instead of DNS routing. The global load balancer distributes incoming traffic to a frontend in the region that's nearest to the user. Both the application stacks write data to, and read from, a database that is synchronously replicated and available across both the regions. If an outage occurs in any one of the two regions, the global load balancer sends user requests to a frontend in the other region.

Use cases

The following sections provide examples of use cases for which the global deployment archetype is an appropriate choice.

Highly available application for a global audience

We recommend the global deployment archetype for applications that serve users across the world and, therefore, need high availability and robustness against outages in multiple regions.

Opportunity to optimize cost and simplify operations

With the global deployment archetype, you can use highly available global resources like a global load balancer and a global database. Compared to a multi-regional deployment, a global deployment can help lower costs and simplify operations because you provision and manage fewer resources.

Design considerations

When you build an architecture that's based on the global deployment archetype, consider the following design factors.

Storage, replication, and networking costs

In a globally distributed architecture, the volume of cross-location network traffic can be high compared to a regional deployment. You might also store and replicate more data. When you build an architecture that's based on the global deployment archetype, consider the potentially higher cost for data storage and networking. For business-critical applications, the availability advantage of a globally distributed architecture might outweigh the higher networking and storage costs.

Managing changes to global resources

The opportunity to use highly available global resources can help you to optimize cost and simplify operations. However, to ensure that the global resources don't become single points of failure (SPOF), you must carefully manage configuration changes to global resources.

Reference architecture

For a reference architecture that you can use to design a global deployment, see Global deployment with Compute Engine and Spanner.

Google Cloud hybrid deployment archetype

Last reviewed 2024-11-20 UTC

This section of the Google Cloud deployment archetypes guide describes the hybrid deployment archetype, provides examples of use cases, and discusses design considerations.

In an architecture that's based on the hybrid deployment archetype, some parts of the application are deployed in Google Cloud, and other parts run on-premises.

Use cases

The following sections provide examples of use cases for which the hybrid deployment archetype is an appropriate choice.

Note: For each of these use cases, the Google Cloud part of the architecture can use the zonal, regional, multi-regional, or global deployment archetype.

Disaster recovery (DR) site for an on-premises application

For mission-critical applications that you run on-premises, you can back up the data to Google Cloud and maintain a replica in the cloud, as shown in the following diagram. The backup frequency and whether the replica needs to be active or passive depends on your recovery time objective (RTO) and recovery point objective (RPO). When the on-premises application is down due to planned or unplanned events, you can activate the replica in Google Cloud to restore the application to production.

Hybrid deployment archetype: DR site for an on-premises application.

On-premises development for cloud applications

For an application that runs in Google Cloud, you can keep the development environments on-premises, and use a CI/CD pipeline to push updates to the cloud, as shown in the following diagram. This architecture lets you retain control over your development activities while enjoying the benefits that Google Cloud offers for scalability, cost optimization, and reliability.

Hybrid deployment archetype: On-premises development for cloud applications.

Enhancing on-premises applications with cloud capabilities

Google Cloud offers advanced capabilities in many areas, including storage, artificial intelligence (AI) and machine learning (ML), big data, and analytics. The hybrid deployment archetype lets you use these advanced Google Cloud capabilities even for applications that you run on-premises. The following are examples of these capabilities:

Low-cost, unlimited archive storage in the cloud for an on-premises application.

AI and ML applications in the cloud for data generated by an on-premises application.

Cloud-based data warehouse and analytics processes using BigQuery for data ingested from on-premises data sources.

Cloud bursting, to handle overflow traffic when the load on the on-premises application reaches peak capacity.

The following diagram shows a hybrid topology where data from an on-premises application is uploaded to Google Cloud. Data analysts analyze the uploaded data by using advanced AI, ML, big data, and analytics capabilities in Google Cloud.

Hybrid deployment archetype: Enhancing on-premises applications with cloud capabilities.

Tiered hybrid topology

In this topology, which is sometimes called a split-stack deployment, the application's frontend is in Google Cloud, and the backend is on-premises. The frontend might include capabilities like load balancing, CDN, DDoS protection, and access policies. The frontend sends traffic to the on-premises backend for processing, as shown in the following diagram:

Hybrid deployment archetype: Tiered hybrid topology.

This architecture might be suitable when an application is used globally but the backend needs to be within a single, controlled environment. A variation of this use case is to run the frontend on-premises and deploy the backend in Google Cloud.

More information

For more information about the rationale and use cases for the hybrid deployment archetype, see Build hybrid and multicloud architectures using Google Cloud.

Design considerations

When you build an architecture that's based on the hybrid deployment archetype, consider the following design factors.

On-premises to cloud network connection

For efficient network communication between your on-premises environment and the resources in Google Cloud, you need a network connection that's reliable and secure. For more information about hybrid connectivity options offered by Google Cloud, see Choosing a Network Connectivity product.

Setup effort and operational complexity

Setting up and operating a hybrid topology requires more effort than an architecture that uses only Google Cloud. To operate this topology, you need to manage resources consistently across the on-premises and Google Cloud environments.

Cost of redundant resources

A hybrid deployment is potentially more expensive than a cloud-only deployment, because data might need to be stored redundantly on-premises and in the cloud. Also, some of the redundant resources might be underutilized. When you build an architecture that's based on the hybrid deployment archetype, consider the potentially higher overall cost of resources.

Example architectures

For examples of architectures that use the hybrid deployment archetype, see Build hybrid and multicloud architectures using Google Cloud.

Google Cloud multicloud deployment archetype

Last reviewed 2024-11-20 UTC

This section of the Google Cloud deployment archetypes guide describes the multicloud deployment archetype, provides examples of use cases, and discusses design considerations.

In an architecture that uses the multicloud deployment archetype, some parts of the application run in Google Cloud while others are deployed in other cloud platforms.

Use cases

The following sections provide examples of use cases for which the multicloud deployment archetype is an appropriate choice.

Note: For each of these use cases, the architecture in each cloud can use the zonal, regional, multi-regional, or global deployment archetype.

Google Cloud as the primary site and another cloud as a DR site

To manage disaster recovery (DR) for mission-critical applications in Google Cloud, you can back up the data and maintain a passive replica in another cloud platform, as shown in the following diagram. If the application in Google Cloud is down, you can use the external replica to restore the application to production.

Multicloud deployment archetype: Google Cloud as the primary site and another cloud as a DR site.

Enhancing applications with Google Cloud capabilities

Google Cloud offers advanced capabilities in areas like storage, artificial intelligence (AI) and machine learning (ML), big data, and analytics. The multicloud deployment archetype lets you take advantage of these advanced capabilities in Google Cloud for applications that you want to run on other cloud platforms. The following are examples of these capabilities:

Low-cost, unlimited archive storage.

AI and ML applications for data generated by applications deployed in other cloud platforms.

Data warehousing and analytics processes using BigQuery for data ingested from applications that run in other cloud platforms.

The following diagram shows a multicloud topology that enhances an application running on another cloud platform with advanced data-processing capabilities in Google Cloud.

Multicloud deployment archetype: Enhancing applications with Google Cloud capabilities.

More information

For more information about the rationale and use cases for the multicloud deployment archetype, see Build hybrid and multicloud architectures using Google Cloud.

Design considerations

When you build an architecture that's based on the multicloud deployment archetype, consider the following design factors.

Cost of redundant resources

A multicloud architecture often costs more than an architecture where the application runs entirely in Google Cloud, due to the following factors:

Data might need to be stored redundantly within each cloud rather than in a single cloud. The storage and data transfer costs might be higher.

If an application runs in multiple cloud platforms, some of the redundant resources might be underutilized, leading to higher overall cost of the deployment.

Inter-cloud connectivity

For efficient network communication between your resources in multiple cloud platforms, you need secure and reliable cross-cloud connectivity. For example, you can use Google Cloud Cross-Cloud Interconnect to establish high-bandwidth dedicated connectivity between Google Cloud and another cloud service provider. For more information, see Patterns for connecting other cloud service providers with Google Cloud.

Setup effort and operational complexity

Setting up and operating a multicloud topology requires significantly more effort than an architecture that uses only Google Cloud:

Security features and tools aren't standard across cloud platforms. Your security administrators need to learn the skills and knowledge that are necessary to manage security for resources distributed across all the cloud platforms that you use.

You need to efficiently provision and manage resources across multiple public cloud platforms. Tools like Terraform can help reduce the effort to provision and manage resources. To manage containerized multicloud applications, you can use GKE attached clusters.

Example architectures

For examples of architectures that use the multicloud deployment archetype, see Build hybrid and multicloud architectures using Google Cloud.

Comparative analysis of Google Cloud deployment archetypes

Last reviewed 2024-11-20 UTC

This section of the Google Cloud deployment archetypes guide compares the deployment archetypes in terms of availability, robustness against outages, cost, and operational complexity.

The following table summarizes the comparative analysis for the basic deployment archetypes: zonal, regional, multi-regional, and global. For the hybrid and multicloud topologies, the deployment archetype that's used for the Google Cloud part of the topology influences the availability, robustness against outages, cost, and operational complexity.

Design consideration	Zonal	Regional	Multi-regional	Global
Infrastructure availability	99.9% (3 nines)	99.99% (4 nines)	99.999% (5 nines)	99.999% (5 nines)
Robustness of infrastructure against zone outages	RTO of hours or days	Near-zero RTO if replication is synchronous	Near-zero RTO if replication is synchronous	Near-zero RTO if replication is synchronous
Robustness of infrastructure against region outages	RTO of hours or days	RTO of hours or days	Near-zero RTO if replication is synchronous	Near-zero RTO if replication is synchronous
Cost of Google Cloud resources	Low	Medium	High	Medium
Operational complexity	Simpler than the other deployment archetypes	More complex than zonal	More complex than regional	Potentially simpler than multi-regional

Note: For more information about region-specific considerations, see Geography and regions.

The following sections describe the comparative analysis that's summarized in the preceding table.

Infrastructure availability

The following sections describe the differences in infrastructure availability between the deployment archetypes.

Zonal, regional, multi-regional, and global deployment archetypes

Google Cloud infrastructure is built to support a target availability of 99.9% for your workload when you use the zonal deployment archetype, 99.99% for regional deployments, and 99.999% for multi-regional and global deployments. These availability numbers are targets for the platform-level infrastructure.

The availability that you can expect from an application that's deployed in Google Cloud depends on the following factors besides the deployment archetype:

Design of the application

Number of interdependent tiers in the application stack

Uptime Service Level Agreements (SLAs) for the Google Cloud services used

Amount of redundant resources

Location scopes of the resources

For more information, see Building blocks of reliability in Google Cloud.

Hybrid and multicloud deployment archetypes

For a hybrid or multicloud topology, the overall availability depends on the infrastructure in each environment and the interdependencies between the environments.

If critical interdependencies exist between components in Google Cloud and components outside Google Cloud, the overall availability is lower than the availability of the component that provides the least availability across all the environments.

If every component of the application is deployed redundantly across Google Cloud and on-premises or in other cloud platforms, the redundancy ensures high availability.

Robustness of infrastructure against zone and region outages

The following sections describe the differences between the deployment archetypes in terms of the ability of the infrastructure to continue to support your workloads in the event of Google Cloud zone and region outages.

Zonal deployment archetype

An architecture that uses the basic single-zone deployment archetype isn't robust against zone outages. You must plan for recovering from zone outages based on your recovery point objective (RPO) and recovery time objective (RTO). For example, you can maintain a passive or scaled-down replica of the infrastructure in another (failover) zone. If an outage occurs in the primary zone, you can promote the database in the failover zone to be the primary database and update the load balancer to send traffic to the frontend in the failover zone.

Regional deployment archetype

An architecture that uses the regional deployment archetype is robust against zone outages. A failure in one zone is unlikely to affect infrastructure in other zones. The RTO is near zero if data is replicated synchronously. However, when an outage affects an entire Google Cloud region, the application becomes unavailable. Plan for recovering from outages according to your RPO and RTO for the application. For example, you can provision a passive replica of the infrastructure in a different region, and activate the replica during region outages.

Multi-regional and global deployment archetypes

An architecture that uses the multi-regional or global deployment archetype is robust against zone and region outages. The RTO is near zero if data is replicated synchronously. An architecture where the application runs as a globally distributed location-unaware stack provides the highest level of robustness against region outages.

Hybrid and multicloud deployment archetypes

The robustness of a hybrid and multicloud architecture depends on the robustness of each environment (Google Cloud, on-premises, and other cloud platforms), and the interdependencies between the environments.

For example, if every component of an application runs redundantly across both Google Cloud and another environment (on-premises or another cloud platform), then the application is robust against any Google Cloud outages. If critical interdependencies exist between components in Google Cloud and components that are deployed on-premises or on other cloud platforms, the robustness against Google Cloud outages depends on the robustness of the deployment archetype that you use for the Google Cloud part of the architecture.

Cost of Google Cloud resources

The cost of the Google Cloud resources that are required for an application depends on the Google Cloud services that you use, the number of resources that you provision, the period for which you retain or use resources, and the deployment archetype that you choose. To estimate the cost of Google Cloud resources in an architecture based on any deployment archetype, you can use the Google Cloud Pricing Calculator.

The following sections describe the differences in the cost of the Google Cloud resources between the various deployment archetypes.

Zonal versus regional and multi-regional deployment archetypes

When compared with an architecture that uses the zonal deployment archetype, an architecture that uses the multi-regional deployment archetype might incur extra costs for redundant storage. Also, for any network traffic that crosses region boundaries, you need to consider the cross-region data transfer costs.

Global deployment archetype

With this archetype, you have the opportunity to use highly available global resources, like a global load balancer. The cost of setting up and operating the cloud resources can be lower than a multi-regional deployment where you provision and configure multiple instances of regional resources. However, global resources might entail higher costs in some cases. For example, the global load balancer requires Premium Tier networking, but for regional load balancers, you can choose Standard Tier.

Hybrid and multicloud deployment archetypes

In a hybrid or multicloud deployment architecture, you need to consider additional costs along with the cost of the resources that you provision. For example, consider costs like hybrid or cross-cloud networking, and the cost of monitoring and managing the resources across multiple environments.

Considerations for all the deployment archetypes

When you assess the cost of running a cloud workload, you need to consider additional costs along with the cost of the Google Cloud resources that you provision. For example, consider personnel expenses and the overhead costs to design, build, and maintain your cloud deployment.

To compare the cost of Google Cloud resources across the deployment archetypes, also consider the cost per unit of work that the application performs. Identify units of work that reflect the business drivers of the application, like the number of users the application serves or the number of requests processed.

By carefully managing the utilization of your Google Cloud resources and adopting Google-recommended best practices, you can optimize the cost of your cloud deployments. For more information, see Google Cloud Well-Architected Framework: Cost optimization.

Operational complexity

This following sections describe the differences in operational complexity between the deployment archetypes, which depends on the number of infrastructure resources, features, and application stacks that you need to operate.

Zonal versus regional and multi-regional deployment archetypes

An architecture that's based on the zonal deployment archetype is easier to set up and operate when compared with the other deployment architectures. An application that runs redundantly in multiple zones or regions requires higher operational effort, due to the following reasons:

The status of the application stacks in multiple locations must be monitored, both at the stack level and for each component of the application.

If a component becomes unavailable in any location, in-process requests must be handled gracefully.

Application changes must be rolled out carefully.

The databases must be synchronized across all the locations.

Global deployment archetype

The global deployment archetype lets you use highly available global resources like a global load balancer and a global database. The effort to set up and operate cloud resources can be lower than a multi-regional deployment where you need to manage multiple instances of regional resources. However, you must carefully manage changes to global resources.

The effort to operate an architecture that uses the global deployment archetype also depends on whether you deploy a distributed location-unaware stack or multiple regionally isolated stacks:

A distributed, location-unaware application can be expanded and scaled with greater flexibility. For example, if certain components have critical end-user latency requirements in only specific locations, you can deploy these components in the required locations and operate the remainder of the stack in other locations.

An application that's deployed as multiple regionally isolated stacks requires higher effort to operate and maintain, due to the following factors:

The status of the application stacks in multiple locations must be monitored, both at the stack level and for each component.

If a component becomes unavailable in any location, in-process requests must be handled gracefully.

Application changes must be rolled out carefully.

The databases must be synchronized across all the locations.

Hybrid and multicloud deployment archetypes

Hybrid or multicloud topologies require more effort to set up and operate than an architecture that uses only Google Cloud.

Resources must be managed consistently across the on-premises and Google Cloud topologies.

You need a way to efficiently provision and manage resources across multiple platforms. Tools like Terraform can help to reduce the provisioning effort.

Security features and tools aren't standard across cloud platforms. Your security administrators need to acquire skills and expertise to manage the security of resources distributed across all the cloud platforms that you use.

Google Cloud deployment archetypes: What's next

Last reviewed 2025-02-25 UTC

This section of the Google Cloud deployment archetypes guide lists resources that you can use to learn more about cloud deployment archetypes and best practices for building architectures for your cloud workloads.

Deployment Archetypes for Cloud Applications: Research paper that this guide is based on.

Google Cloud Well-Architected Framework: Recommendations to help you design and operate a cloud topology that's secure, efficient, resilient, high-performing, and cost-effective.

Hybrid and multicloud architecture patterns: Common architectural patterns for the hybrid and multicloud deployment archetypes.

Patterns for connecting other cloud service providers with Google Cloud: Options for network connections between Google Cloud and other cloud platforms.

Design reliable infrastructure for your workloads in Google Cloud: Design guidance to help to protect your applications against failures at the resource, zone, and region level.

Enterprise foundations blueprint: An opinionated enterprise foundations blueprint for a landing zone, with step-by-step guidance to configure and deploy your Google Cloud estate.

Single-zone deployment on Compute Engine

Release Notes

Last reviewed 2025-08-12 UTC

This document provides a reference architecture for a multi-tier application that runs on Compute Engine VMs in a single zone in Google Cloud. You can use this reference architecture to efficiently rehost (lift and shift) on-premises applications to the cloud with minimal changes to the applications. The document also describes the design factors that you should consider when you build a zonal architecture for your cloud applications. The intended audience for this document is cloud architects.

Architecture

The following diagram shows an architecture for an application that runs in a single Google Cloud zone. This architecture is aligned with the Google Cloud zonal deployment archetype.

The architecture is based on the infrastructure as a service (IaaS) cloud model. You provision the required infrastructure resources (compute, networking, and storage) in Google Cloud. You retain full control over the infrastructure and responsibility for the operating system, middleware, and higher layers of the application stack. To learn more about IaaS and other cloud models, see PaaS vs. IaaS vs. SaaS vs. CaaS: How are they different?

The preceding diagram includes the following components:

Component	Purpose
Regional external load balancer	The regional external load balancer receives and distributes user requests to the web tier VMs.
Zonal managed instance group (MIG) for the web tier	The web tier of the application is deployed on Compute Engine VMs that are part of a zonal MIG. The MIG is the backend for the regional external load balancer. Each VM in the MIG hosts an independent instance of the web tier of the application.
Regional internal load balancer	The regional internal load balancer distributes traffic from the web tier VMs to the application tier VMs.
Zonal MIG for the application tier	The application tier is deployed on Compute Engine VMs that are part of a zonal MIG, which is the backend for the internal load balancer. Each VM in the MIG hosts an independent instance of the application tier.

Component	Purpose
Third-party database deployed on a Compute Engine VM	The architecture in this document shows a third-party database (like PostgreSQL) that's deployed on a Compute Engine VM. You can deploy a standby database in another zone. The database replication and failover capabilities depend on the database that you use. Installing and managing a third-party database involves additional effort and operational cost for applying updates, monitoring, and ensuring availability. You can avoid the overhead of installing and managing a third-party database and take advantage of built-in high availability (HA) features by using a fully managed database service like Cloud SQL or AlloyDB for PostgreSQL. For more information about managed database options, see Database services.
Virtual Private Cloud network and subnet	All the Google Cloud resources in the architecture use a single VPC network and subnet. Depending on your requirements, you can choose to build an architecture that uses multiple VPC networks or multiple subnets. For more information, see Deciding whether to create multiple VPC networks
Cloud Storage regional bucket	Application and database backups are stored in a regional Cloud Storage bucket. If a zone outage occurs, your application and data aren't lost. Alternatively, you can use Backup and DR Service to create, store, and manage the database backups.

Products used

This reference architecture uses the following Google Cloud products:

- Compute Engine: A secure and customizable compute service that lets you create and run VMs on Google's infrastructure.
- Cloud Load Balancing: A portfolio of high performance, scalable, global and regional load balancers.
- Cloud Storage: A low-cost, no-limit object store for diverse data types. Data can be accessed from within and outside Google Cloud, and it's replicated across locations for redundancy.

- Virtual Private Cloud (VPC): A virtual system that provides global, scalable networking functionality for your Google Cloud workloads. VPC includes VPC Network Peering, Private Service Connect, private services access, and Shared VPC.

Use cases

This section describes use cases for which a single-zone deployment on Compute Engine is an appropriate choice.

- **Cloud development and testing**: You can use a single-zone deployment architecture to build a low-cost cloud environment for development and testing.
- **Applications that don't need HA**: A single-zone architecture might be sufficient for applications that can tolerate downtime due to infrastructure outages.
- **Low-latency, low-cost networking between application components**: A single-zone architecture might be well suited for applications such as batch computing that need low-latency and high-bandwidth network connections among the compute nodes. With a single-zone deployment, there's no cross-zone network traffic, and you don't incur costs for intra-zone traffic.
- **Migration of commodity workloads**: The zonal deployment architecture provides a cloud-migration path for commodity on-premises applications for which you have no control over the code or that can't support architectures beyond a basic active-passive topology.
- **Running license-restricted software**: A single-zone architecture might be well suited for license-restricted systems where running more than one instance at a time is either too expensive or isn't permitted.

Design considerations

This section provides guidance to help you use this reference architecture to develop an architecture that meets your specific requirements for system design, security, reliability, operational efficiency, cost, and performance.

Note: The guidance in this section isn't exhaustive. Depending on the specific requirements of your application and the Google Cloud products and features that you use, there might be additional design factors and trade-offs that you should consider.

When you build the architecture for your workload, consider best practices and recommendations in the Google Cloud Well-Architected Framework.

System design

This section provides guidance to help you to choose Google Cloud regions and zones for your zonal deployment and to select appropriate Google Cloud services.

Region selection

When you choose the Google Cloud regions where your applications must be deployed, consider the following factors and requirements:

- Availability of Google Cloud services in each region. For more information, see Products available by location.
- Availability of Compute Engine machine types in each region. For more information, see Regions and zones.
- End-user latency requirements.
- Cost of Google Cloud resources.
- Cross-regional data transfer costs.
- Regulatory requirements.

Some of these factors and requirements might involve trade-offs. For example, the most cost-efficient region might not have the lowest carbon footprint. For more information, see Best practices for Compute Engine regions selection.

Compute infrastructure

The reference architecture in this document uses Compute Engine VMs for certain tiers of the application. Depending on the requirements of your application, you can choose from other Google Cloud compute services:

- **Containers**: You can run containerized applications in Google Kubernetes Engine (GKE) clusters. GKE is a container-orchestration engine that automates deploying, scaling, and managing containerized applications.
- **Serverless**: If you prefer to focus your IT efforts on your data and applications instead of setting up and operating infrastructure resources, then you can use serverless services like Cloud Run.

The decision of whether to use VMs, containers, or serverless services involves a trade-off between configuration flexibility and management effort. VMs and containers provide more configuration flexibility, but you're responsible for managing the resources. In a serverless architecture, you deploy workloads to a preconfigured platform that requires minimal management effort. For more information about choosing appropriate compute services for your workloads in Google Cloud, see Hosting Applications on Google Cloud.

Storage services

The architecture shown in this document uses zonal Persistent Disk volumes for all the tiers. For more durable persistent storage, you can use regional Persistent Disk volumes, which provide synchronous replication of data across two zones within a region.

Google Cloud Hyperdisk provides better performance, flexibility, and efficiency than Persistent Disk. With Hyperdisk Balanced, you can provision IOPS and throughput separately and dynamically, which lets you tune the volume to a wide variety of workloads.

For low-cost storage that's replicated across multiple locations, you can use Cloud Storage regional, dual-region, or multi-region buckets.

- Data in regional buckets is replicated synchronously across the zones in the region.
- Data in dual-region or multi-region buckets is stored redundantly in at least two separate geographic locations. Metadata is written synchronously across regions, and data is replicated asynchronously. For dual-region buckets, you can use turbo replication, which ensures that objects are replicated across region pairs, with a recovery point objective (RPO) of 15 minutes. For more information, see Data availability and durability.

To store data that's shared across multiple VMs in a region, such as across all the VMs in the web tier or application tier, you can use a Filestore regional instance. The data that you store in a Filestore regional instance is replicated synchronously across three zones within the region. This replication ensures high availability and robustness against zone outages. You can store shared configuration files, common tools and utilities, and centralized logs in the Filestore instance, and mount the instance on multiple VMs. For robustness against region outages, you can replicate a Filestore instance to a different region. For more information, see Instance replication.

If your database is Microsoft SQL Server, we recommend using Cloud SQL for SQL Server. In scenarios when Cloud SQL doesn't support your configuration requirements, or if you need access to the operating system, you can deploy a Microsoft SQL Server failover cluster instance (FCI). In this scenario, you can use the fully managed Google Cloud NetApp Volumes to provide continuous availability (CA) SMB storage for the database.

When you design storage for your workloads, consider the functional characteristics, resilience requirements, performance expectations, and cost goals. For more information, see Design an optimal storage strategy for your cloud workload.

Database services

The reference architecture in this document uses a third-party database that's deployed on Compute Engine VMs. Installing and managing a third-party database involves effort and cost for operations like applying updates, monitoring and ensuring availability, performing backups, and recovering from failures.

You can avoid the effort and cost of installing and managing a third-party database by using a fully managed database service like Cloud SQL, AlloyDB for PostgreSQL, Bigtable, Spanner, or Firestore. These Google Cloud database services provide uptime service-level agreements (SLAs), and they include default capabilities for scalability and observability.

If your workload needs an Oracle database, you can deploy the database on a Compute Engine VM or use Oracle Database@Google Cloud. For more information, see Oracle workloads in Google Cloud.

Network design

Choose a network design that meets your business and technical requirements. You can use a single VPC network or multiple VPC networks. For more information, see the following documentation:

- Deciding whether to create multiple VPC networks
- Decide the network design for your Google Cloud landing zone

Security, privacy, and compliance

This section describes factors that you should consider when you use this reference architecture to design and build a zonal topology in Google Cloud that meets the security and compliance requirements of your workloads.

Protection against external threats

To protect your application against threats like distributed-denial-of-service (DDoS) attacks and cross-site scripting (XSS), you can use Google Cloud Armor security policies. Each policy is a set of rules that specifies certain conditions that should be evaluated and actions to take when the conditions are met. For example, a rule could specify that if the source IP address of the incoming traffic matches a specific IP address or CIDR range, then the traffic must be denied. You can also apply preconfigured web application firewall (WAF) rules. For more information, see Security policy overview.

External access for VMs

In the reference architecture that this document describes, the Compute Engine VMs don't need inbound access from the internet. Don't assign external IP addresses to the VMs. Google Cloud resources that have only a private, internal IP address can still access certain Google APIs and services by using Private Service Connect or Private Google Access. For more information, see Private access options for services.

To enable secure outbound connections from Google Cloud resources that have only private IP addresses, like the Compute Engine VMs in this reference architecture, you can use Secure Web Proxy or Cloud NAT.

Service account privileges

For the Compute Engine VMs in the architecture, instead of using the default service accounts, we recommend that you create dedicated service accounts and specify the resources that the service account can access. The default service account has a broad range of permissions, including some that might not be necessary. You can tailor dedicated service accounts to have only the essential permissions. For more information, see Limit service account privileges.

SSH security

To enhance the security of SSH connections to the Compute Engine VMs in your architecture, implement Identity-Aware Proxy (IAP) and Cloud OS Login API. IAP lets you control network access based on user identity and Identity and Access Management (IAM) policies. Cloud OS Login API lets you control Linux SSH access based on user identity and IAM policies. For more information about managing network access, see Best practices for controlling SSH login access.

Network security

To control network traffic between the resources in the architecture, you must configure appropriate Cloud Next Generation Firewall (NGFW) policies.

Each firewall rule lets you control traffic based on parameters like the protocol, IP address, and port. For example, you can configure a firewall rule to allow TCP traffic from the web server VMs to a specific port of the database VMs, and block all other traffic.

More security considerations

When you build the architecture for your workload, consider the platform-level security best practices and recommendations that are provided in the Enterprise foundations blueprint and Google Cloud Well-Architected Framework: Security, privacy, and compliance.

Reliability

This section describes design factors that you should consider when you use this reference architecture to build and operate reliable infrastructure for your zonal deployments in Google Cloud.

Robustness against infrastructure outages

In a single-zone deployment architecture, if any component in the infrastructure stack fails, the application can process requests if each tier contains at least one functioning component with adequate capacity. For example, if a web server instance fails, the load balancer forwards user requests to the other available web server instances. If a VM that hosts a web server or app server instance crashes, the MIG recreates the VM automatically. If the database crashes, you must manually activate the second database and update the app server instances to connect to the database.

A zone outage or region outage affects all the Compute Engine VMs in a single-zone deployment. A zone outage doesn't affect the load balancer in this architecture because it's a regional resource. However, the load balancer can't distribute traffic, because there are no available backends. If a zone or region outage occurs, you must wait for Google to resolve the outage, and then verify that the application works as expected.

You can reduce the downtime caused by zone or region outages by maintaining a passive (failover) replica of the infrastructure stack in another Google Cloud zone or region. If an outage occurs in the primary zone, you can activate the stack in the failover zone or region, and use a DNS routing policy to route traffic to the load balancer in the failover zone or region.

For applications that require robustness against zone or region outages, consider using a regional or multi-regional architecture. See the following reference architectures:

- Regional deployment on Compute Engine
- Multi-regional deployment on Compute Engine

MIG autoscaling

The autoscaling capability of stateless MIGs lets you maintain application availability and performance at predictable levels.

To control the autoscaling behavior of your stateless MIGs, you can specify target utilization metrics, such as average CPU utilization. You can also configure schedule-based autoscaling for stateless MIGs. Stateful MIGs can't be autoscaled. For more information, see Autoscaling groups of instances.

MIG size limit

When you decide the size of your MIGs, consider the default and maximum limits on the number of VMs that can be created in a MIG. For more information, see Add and remove VMs from a MIG.

VM autohealing

Sometimes the VMs that host your application might be running and available, but there might be issues with the application itself. The application might freeze, crash, or not have sufficient memory. To verify whether an application is responding as expected, you can configure application-based health checks as part of the autohealing policy of your MIGs. If the application on a particular VM isn't responding, the MIG autoheals (repairs) the VM. For more information about configuring autohealing, see About repairing VMs for high availability.

VM placement

In the architecture that this document describes, the application tier and web tier run on Compute Engine VMs within a single zone.

To improve the robustness of the architecture, you can create a spread placement policy and apply it to the MIG template. When the MIG creates VMs, it places the VMs within each zone on different physical servers (called *hosts*), so your VMs are robust against failures of individual hosts. For more information, see Create and apply spread placement policies to VMs.

VM capacity planning

To make sure that capacity for Compute Engine VMs is available when VMs need to be provisioned, you can create *reservations*. A reservation provides assured capacity in a specific zone for a specified number of VMs of a machine type that you choose. A reservation can be specific to a project, or shared across multiple projects. For more information about reservations, see Choose a reservation type.

Stateful storage

A best practice in application design is to avoid the need for stateful local disks. But if the requirement exists, you can configure your persistent disks to be stateful to ensure that the data is preserved when the VMs are repaired or recreated. However, we recommend that you keep the boot disks stateless, so that you can update them to the latest images with new versions and security patches. For more information, see Configuring stateful persistent disks in MIGs.

Data durability

You can use Backup and DR to create, store, and manage backups of the Compute Engine VMs. Backup and DR stores backup data in its original, application-readable format. When required, you can restore your workloads to production by directly using data from long-term backup storage and avoid the need to prepare or move data.

Compute Engine provides the following options to help you to ensure the durability of data that's stored in Persistent Disk volumes:

- You can use snapshots to capture the point-in-time state of Persistent Disk volumes. The snapshots are stored redundantly in multiple regions, with automatic checksums to ensure the integrity of your data. Snapshots are incremental by default, so they use less storage space and you save money. Snapshots are stored in a Cloud Storage location that you can configure. For more recommendations about using and managing snapshots, see Best practices for Compute Engine disk snapshots.
- To ensure that data in Persistent Disk remains available if a zone outage occurs, you can use Regional Persistent Disk or Hyperdisk Balanced High Availability. Data in these disk types is replicated synchronously between two zones in the same region. For more information, see About synchronous disk replication.

Database availability

If you use a managed database service like Cloud SQL in HA configuration, then in the event of a failure of the primary database, Cloud SQL fails over automatically to the standby database. You don't need to change the IP address for the database endpoint. If you use a self-managed third-party database that's deployed on a Compute Engine VM, then you must use an internal load balancer or other mechanism to ensure that the application can connect to another database if the primary database is unavailable.

To implement cross-zone failover for a database that's deployed on a Compute Engine VM, you need a mechanism to identify failures of the primary database and a process to fail over to the standby database. The specifics of the failover mechanism depend on the database that you use. You can set up an observer instance to detect failures of the primary database and orchestrate the failover. You must configure the failover rules appropriately to avoid a split-brain situation and prevent unnecessary failover. For example architectures that you can use to implement failover for PostgreSQL databases, see Architectures for high availability of PostgreSQL clusters on Compute Engine.

More reliability considerations

When you build the cloud architecture for your workload, review the reliability-related best practices and recommendations that are provided in the following documentation:

- Google Cloud infrastructure reliability guide
- Patterns for scalable and resilient apps
- Designing resilient systems
- Google Cloud Well-Architected Framework: Reliability

Cost optimization

This section provides guidance to optimize the cost of setting up and operating a zonal Google Cloud topology that you build by using this reference architecture.

VM machine types

To help you optimize the resource utilization of your VM instances, Compute Engine provides machine type recommendations. Use the recommendations to choose machine types that match your workload's compute requirements. For workloads with predictable resource requirements, you can customize the machine type to your needs and save money by using custom machine types.

VM provisioning model

If your application is fault tolerant, then Spot VMs can help to reduce your Compute Engine costs for the VMs in the application and web tiers. The cost of Spot VMs is significantly lower than regular VMs. However, Compute Engine might preemptively stop or delete Spot VMs to reclaim capacity.

Spot VMs are suitable for batch jobs that can tolerate preemption and don't have high availability requirements. Spot VMs offer the same machine types, options, and performance as regular VMs. However, when the resource capacity in a zone is limited, MIGs might not be able to scale out (that is, create VMs) automatically to the specified target size until the required capacity becomes available again.

VM resource utilization

The autoscaling capability of stateless MIGs enables your application to handle increases in traffic gracefully, and it helps you to reduce cost when the need for resources is low. Stateful MIGs can't be autoscaled.

Third-party licensing

When you migrate third-party workloads to Google Cloud, you might be able to reduce cost by bringing your own licenses (BYOL). For example, to deploy Microsoft Windows Server VMs, instead of using a premium image that incurs additional cost for the third-party license, you can create and use a custom Windows BYOL image. You then pay only for the VM infrastructure that you use on Google Cloud. This strategy helps you continue to realize value from your existing investments in third-party licenses. If you decide to use the BYOL approach, then the following recommendations might help to reduce cost:

- Provision the required number of compute CPU cores independently of memory by using custom machine types. By doing this, you limit the third-party licensing cost to the number of CPU cores that you need.
- Reduce the number of vCPUs per core from 2 to 1 by disabling simultaneous multithreading (SMT).

If you deploy a third-party database like Microsoft SQL Server on Compute Engine VMs, then you must consider the license costs for the third-party software. When you use a managed database service like Cloud SQL, the database license costs are included in the charges for the service.

More cost considerations

When you build the architecture for your workload, also consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Cost optimization.

Operational efficiency

This section describes the factors that you should consider when you use this reference architecture to design and build a zonal Google Cloud topology that you can operate efficiently.

VM configuration updates

To update the configuration of the VMs in a MIG (such as the machine type or boot-disk image), you create a new instance template with the required configuration and then apply the new template to the MIG. The MIG updates the VMs by using the update method that you choose: automatic or selective. Choose an appropriate method based on your requirements for availability and operational efficiency. For more information about these MIG update methods, see Apply new VM configurations in a MIG.

VM images

For your VMs, instead of using Google-provided public images, we recommend that you create and use custom OS images that contain the configurations and software that your applications require. You can group your custom images into a custom image family. An image family always points to the most recent image in that family, so your instance templates and scripts can use that image without you having to update references to a specific image version. You must regularly update your custom images to include the security updates and patches that are provided by the OS vendor.

Deterministic instance templates

If the instance templates that you use for your MIGs include startup scripts to install third-party software, make sure that the scripts explicitly specify software-installation parameters such as the software version. Otherwise, when the MIG creates the VMs, the software that's installed on the VMs might not be consistent. For example, if your instance template includes a startup script to install Apache HTTP Server 2.0 (the apache2 package), then make sure that the script specifies the exact apache2 version that should be installed, such as version 2.4.53. For more information, see Deterministic instance templates.

More operational considerations

When you build the architecture for your workload, consider the general best practices and recommendations for operational efficiency that are described in Google Cloud Well-Architected Framework: Operational excellence.

Performance optimization

This section describes the factors that you should consider when you use this reference architecture to design and build a zonal topology in Google Cloud that meets the performance requirements of your workloads.

Compute performance

Compute Engine offers a wide range of predefined and customizable machine types for the workloads that you run on VMs. Choose an appropriate machine type based on your performance requirements. For more information, see Machine families resource and comparison guide.

VM multithreading

Each virtual CPU (vCPU) that you allocate to a Compute Engine VM is implemented as a single hardware multithread. By default, two vCPUs share a physical CPU core. For applications that involve highly parallel operations or that perform floating point calculations (such as genetic sequence analysis, and financial risk modeling), you can improve performance by reducing the number of threads that run on each physical CPU core. For more information, see Set the number of threads per core.

VM multithreading might have licensing implications for some third-party software, like databases. For more information, read the licensing documentation for the third-party software.

Network Service Tiers

Network Service Tiers lets you optimize the network cost and performance of your workloads. You can choose Premium Tier or Standard Tier. Premium Tier delivers traffic on Google's global backbone to achieve minimal packet loss and low latency. Standard Tier delivers traffic using peering, internet service providers (ISP), or transit networks at an edge point of presence (PoP) that's closest to the region where your Google Cloud workload runs. To optimize performance, we recommend using Premium Tier. To optimize cost, we recommend using Standard Tier.

Network performance

For workloads that need low inter-VM network latency within the application and web tiers, you can create a compact placement policy and apply it to the MIG template that's used for those tiers. When the MIG creates VMs, it places the VMs on physical servers that are close to each other. While a compact placement policy helps improve inter-VM network performance, a spread placement policy can help improve VM availability as described earlier. To achieve an optimal balance between network performance and availability, when you create a compact placement policy, you can specify how far apart the VMs must be placed. For more information, see Placement policies overview.

Compute Engine has a per-VM limit for egress network bandwidth. This limit depends on the VM's machine type and whether traffic is routed through the same VPC network as the source VM. For VMs with certain machine types, to improve network performance, you can get a higher maximum egress bandwidth by enabling Tier_1 networking.

More performance considerations

When you build the architecture for your workload, consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Performance optimization.

What's next

- Learn more about the Google Cloud products used in this reference architecture:
  - Cloud Load Balancing overview
  - Instance groups
- Get started with migrating your workloads to Google Cloud.
- Explore and evaluate deployment archetypes that you can choose to build architectures for your cloud workloads.
- Review architecture options for designing reliable infrastructure for your workloads in Google Cloud.
- For more reference architectures, diagrams, and best practices, explore the Cloud Architecture Center.

Contributors

Authors:

- Kumar Dhanagopal | Cross-Product Solution Developer
- Samantha He | Technical Writer

Other contributors:

- Ben Good | Solutions Architect

- Carl Franklin | Director, PSO Enterprise Architecture
- Daniel Lees | Cloud Security Architect
- Gleb Otochkin | Cloud Advocate, Databases
- Mark Schlagenhauf | Technical Writer, Networking
- Pawel Wenda | Group Product Manager
- Sean Derrington | Group Product Manager, Storage
- Sekou Page | Outbound Product Manager
- Simon Bennett | Group Product Manager
- Steve McGhee | Reliability Advocate
- Victor Moreno | Product Manager, Cloud Networking

Regional deployment on Compute Engine

Release Notes

Last reviewed 2025-08-12 UTC

This document provides a reference architecture for a multi-tier application that runs on Compute Engine VMs in multiple zones within a Google Cloud region. You can use this reference architecture to efficiently rehost (lift and shift) on-premises applications to the cloud with minimal changes to the applications. The document also describes the design factors that you should consider when you build a regional architecture for your cloud applications. The intended audience for this document is cloud architects.

Architecture

The following diagram shows an architecture for an application that runs in active-active mode in isolated stacks that are deployed across three Google Cloud zones within a region. The architecture is aligned with the regional deployment archetype.

An application runs in active-active-mode in isolated stacks that are deployed across three Google Cloud zones within a region.

The architecture is based on the infrastructure as a service (IaaS) cloud model. You provision the required infrastructure resources (compute, networking, and storage) in Google Cloud. You retain full control over the infrastructure and responsibility for the operating system, middleware, and higher layers of the application stack. To learn more about IaaS and other cloud models, see PaaS vs. IaaS vs. SaaS vs. CaaS: How are they different?.

The preceding diagram includes the following components:

Component   Purpose

Regional external load balancer

The regional external load balancer receives and distributes user requests to the web tier VMs.

Use an appropriate load balancer type depending on the traffic type and other requirements. For example, if the backend consists of web servers (as shown in the preceding architecture), then use an Application Load Balancer to forward HTTP(S) traffic. To load-balance TCP traffic, use a Network Load Balancer. For more information, see Choose a load balancer.

Regional managed instance group (MIG) for the web tier

The web tier of the application is deployed on Compute Engine VMs that are part of a regional MIG. The MIG is the backend for the regional external load balancer.

The MIG contains Compute Engine VMs in three different zones. Each of these VMs hosts an independent instance of the web tier of the application.

Regional internal load balancer

The regional internal load balancer distributes traffic from the web tier VMs to the application tier VMs.

Depending on your requirements, you can use a regional internal Application Load Balancer or Network Load Balancer. For more information, see Choose a load balancer.

Regional MIG for the application tier

The application tier is deployed on Compute Engine VMs that are part of a regional MIG, which is the backend for the internal load balancer.

The MIG contains Compute Engine VMs in three different zones. Each VM hosts an independent instance of the application tier.

Third-party database deployed on a Compute Engine VM

The architecture in this document shows a third-party database (like PostgreSQL) that's deployed on a Compute Engine VM. You can deploy a standby database in another zone. The database replication and failover capabilities depend on the database that you use.

Installing and managing a third-party database involves additional effort and operational cost for applying updates, monitoring, and ensuring availability. You can avoid the overhead of installing and managing a third-party database and take advantage of built-in high availability (HA) features by using a fully managed database service like Cloud SQL or AlloyDB for PostgreSQL. For more information about managed database options, see Database services later in this guide.

Virtual Private Cloud network and subnet

All the Google Cloud resources in the architecture use a single VPC network and subnet.

Depending on your requirements, you can choose to build an architecture that uses multiple VPC networks or multiple subnets. For more information, see Deciding whether to create multiple VPC networks in "Best practices and reference architectures for VPC design."

Cloud Storage dual-region bucket

Application and database backups are stored in a dual-region Cloud Storage bucket. If a zone or region outage occurs, your application and data aren't lost.

Alternatively, you can use Backup and DR Service to create, store, and manage the database backups.

Products used

This reference architecture uses the following Google Cloud products:

Compute Engine: A secure and customizable compute service that lets you create and run VMs on Google's infrastructure.

Cloud Load Balancing: A portfolio of high performance, scalable, global and regional load balancers.

Cloud Storage: A low-cost, no-limit object store for diverse data types. Data can be accessed from within and outside Google Cloud, and it's replicated across locations for redundancy.

Virtual Private Cloud (VPC): A virtual system that provides global, scalable networking functionality for your Google Cloud workloads. VPC includes VPC Network Peering, Private Service Connect, private services access, and Shared VPC.

Use cases

This section describes use cases for which a regional deployment on Compute Engine is an appropriate choice.

Efficient migration of on-premises applications

You can use this reference architecture to build a Google Cloud topology to rehost (lift and shift) on-premises applications to the cloud with minimal changes to the applications. All the tiers of the application in this reference architecture are hosted on Compute Engine VMs. This approach lets you migrate on-premises applications efficiently to the cloud and take advantage of the cost benefits, reliability, performance, and operational simplicity that Google Cloud provides.

Highly available application with users within a geographic area

We recommend a regional deployment architecture for applications that need robustness against zone outages but can tolerate some downtime caused by region outages. If any part of the application stack fails, the application continues to run if at least one functioning component with adequate capacity exists in every tier. If a zone outage occurs, the application stack continues to run in the other zones.

Low latency for application users

If all the users of an application are within a single geographic area, such as a single country, a regional deployment architecture can help improve the user-perceived performance of the application. You can optimize network latency for user requests by deploying the application in the Google Cloud region that's closest to your users.

Low-latency networking between application components

A single-region architecture might be well suited for applications such as batch computing that need low-latency and high-bandwidth network connections among the compute nodes. All the resources are in a single Google Cloud region, so inter-resource network traffic remains within the region. The inter-resource network latency is low, and you don't incur cross-region data transfer costs. Intra-region network costs still apply.

Compliance with data residency requirements

You can use a single-region architecture to build a topology that helps you to meet data residency requirements. For example, a country in Europe might require that all user data be stored and accessed in data centers that are located physically within Europe. To meet this requirement, you can run the application in a Google Cloud region in Europe.

Design considerations

This section provides guidance to help you use this reference architecture to develop an architecture that meets your specific requirements for system design, security and compliance, reliability, operational efficiency, cost, and performance.

Note: The guidance in this section isn't exhaustive. Depending on the specific requirements of your application and the Google Cloud products and features that you use, there might be additional design factors and trade-offs that you should consider.

System design

This section provides guidance to help you to choose Google Cloud regions for your regional deployment and to select appropriate Google Cloud services.

Region selection

When you choose the Google Cloud regions where your applications must be deployed, consider the following factors and requirements:

Availability of Google Cloud services in each region. For more information, see Products available by location.

Availability of Compute Engine machine types in each region. For more information, see Regions and zones.

End-user latency requirements.

Cost of Google Cloud resources.

Cross-regional data transfer costs.

Regulatory requirements.

Some of these factors and requirements might involve trade-offs. For example, the most cost-efficient region might not have the lowest carbon footprint. For more information, see Best practices for Compute Engine regions selection.

Compute infrastructure

The reference architecture in this document uses Compute Engine VMs for certain tiers of the application. Depending on the requirements of your application, you can choose from other Google Cloud compute services:

Containers: You can run containerized applications in Google Kubernetes Engine (GKE) clusters. GKE is a container-orchestration engine that automates deploying, scaling, and managing containerized applications.

Serverless: If you prefer to focus your IT efforts on your data and applications instead of setting up and operating infrastructure resources, then you can use serverless services like Cloud Run.

The decision of whether to use VMs, containers, or serverless services involves a trade-off between configuration flexibility and management effort. VMs and containers provide more configuration flexibility, but you're responsible for managing the resources. In a serverless architecture, you deploy workloads to a preconfigured platform that requires minimal management effort. For more information about choosing appropriate compute services for your workloads in Google Cloud, see Hosting Applications on Google Cloud.

Storage services

The architecture shown in this document uses regional Persistent Disk volumes for all the tiers. Persistent disks provide synchronous replication of data across two zones within a region.

Google Cloud Hyperdisk provides better performance, flexibility, and efficiency than Persistent Disk. With Hyperdisk Balanced, you can provision IOPS and throughput separately and dynamically, which lets you tune the volume to a wide variety of workloads.

For low-cost storage that's replicated across multiple locations, you can use Cloud Storage regional, dual-region, or multi-region buckets.

Data in regional buckets is replicated synchronously across the zones in the region.

Data in dual-region or multi-region buckets is stored redundantly in at least two separate geographic locations. Metadata is written synchronously across regions, and data is replicated asynchronously. For dual-region buckets, you can use turbo replication, which ensures that objects are replicated across region pairs, with a recovery point objective (RPO) of 15 minutes. For more information, see Data availability and durability.

To store data that's shared across multiple VMs in a region, such as across all the VMs in the web tier or application tier, you can use a Filestore regional instance. The data that you store in a Filestore regional instance is replicated synchronously across three zones within the region. This replication ensures high availability and robustness against zone outages. You can store shared configuration files, common tools and utilities, and centralized logs in the Filestore instance, and mount the instance on multiple VMs. For robustness against region outages, you can replicate a Filestore instance to a different region. For more information, see Instance replication.

If your database is Microsoft SQL Server, we recommend using Cloud SQL for SQL Server. In scenarios when Cloud SQL doesn't support your configuration requirements, or if you need access to the operating system, you can deploy a Microsoft SQL Server failover cluster instance (FCI). In this scenario, you can use the fully managed Google Cloud NetApp Volumes to provide continuous availability (CA) SMB storage for the database.

When you design storage for your workloads, consider the functional characteristics, resilience requirements, performance expectations, and cost goals. For more information, see Design an optimal storage strategy for your cloud workload.

Database services

The reference architecture in this document uses a third-party database that's deployed on Compute Engine VMs. Installing and managing a third-party database involves effort and cost for operations like applying updates, monitoring and ensuring availability, performing backups, and recovering from failures.

You can avoid the effort and cost of installing and managing a third-party database by using a fully managed database service like Cloud SQL, AlloyDB for PostgreSQL, Bigtable, Spanner, or Firestore. These Google Cloud database services provide uptime service-level agreements (SLAs), and they include default capabilities for scalability and observability.

If your workload needs an Oracle database, you can deploy the database on a Compute Engine VM or use Oracle Database@Google Cloud. For more information, see Oracle workloads in Google Cloud.

Network design

Choose a network design that meets your business and technical requirements. You can use a single VPC network or multiple VPC networks. For more information, see the following documentation:

Deciding whether to create multiple VPC networks

Decide the network design for your Google Cloud landing zone

Security, privacy, and compliance

This section describes factors that you should consider when you use this reference architecture to design and build a regional topology in Google Cloud that meets the security, privacy, and compliance requirements of your workloads.

Protection against external threats

To protect your application against threats like distributed-denial-of-service (DDoS) attacks and cross-site scripting (XSS), you can use Google Cloud Armor security policies. Each policy is a set of rules that specifies certain conditions that should be evaluated and actions to take when the conditions are met. For example, a rule could specify that if the source IP address of the incoming traffic matches a specific IP address or CIDR range, then the traffic must be denied. You can also apply preconfigured web application firewall (WAF) rules. For more information, see Security policy overview.

External access for VMs

In the reference architecture that this document describes, the Compute Engine VMs don't need inbound access from the internet. Don't assign external IP addresses to the VMs. Google Cloud resources that have only a private, internal IP address can still access certain Google APIs and services by using Private Service Connect or Private Google Access. For more information, see Private access options for services.

To enable secure outbound connections from Google Cloud resources that have only private IP addresses, like the Compute Engine VMs in this reference architecture, you can use Secure Web Proxy or Cloud NAT.

Service account privileges

For the Compute Engine VMs in the architecture, instead of using the default service accounts, we recommend that you create dedicated service accounts and specify the resources that the service account can access. The default service account has a broad range of permissions, including some that might not be necessary. You can tailor dedicated service accounts to have only the essential permissions. For more information, see Limit service account privileges.

SSH security

To enhance the security of SSH connections to the Compute Engine VMs in your architecture, implement Identity-Aware Proxy (IAP) and Cloud OS Login API. IAP lets you control network access based on user identity and Identity and Access Management (IAM) policies. Cloud OS Login API lets you control Linux SSH access based on user identity and IAM policies. For more information about managing network access, see Best practices for controlling SSH login access.

Network security

To control network traffic between the resources in the architecture, you must configure appropriate Cloud Next Generation Firewall (NGFW) policies.

More security considerations

When you build the architecture for your workload, consider the platform-level security best practices and recommendations that are provided in the Enterprise foundations blueprint and Google Cloud Well-Architected Framework: Security, privacy, and compliance.

Reliability

This section describes design factors that you should consider when you use this reference architecture to build and operate reliable infrastructure for your regional deployments in Google Cloud.

Infrastructure outages

In a regional architecture, if any individual component in the infrastructure stack fails, the application can process requests if at least one functioning component with adequate capacity exists in each tier. For example, if a web server instance fails, the load balancer forwards user requests to the other available web server instances. If a VM that hosts a web server or app server instance crashes, the MIG recreates the VM automatically.

If a zone outage occurs, the load balancer isn't affected, because it's a regional resource. A zone outage might affect individual Compute Engine VMs. But the application remains available and responsive because the VMs are in a regional MIG. A regional MIG ensures that new VMs are created automatically to maintain the configured minimum number of VMs. After Google resolves the zone outage, you must verify that the application runs as expected in all the zones where it's deployed.

If all the zones in this architecture have an outage or if a region outage occurs, then the application is unavailable. You must wait for Google to resolve the outage, and then verify that the application works as expected.

You can reduce the downtime caused by region outages by maintaining a passive (failover) replica of the infrastructure stack in another Google Cloud region. If an outage occurs in the primary region, you can activate the stack in the failover region and use DNS routing policies to route traffic to the load balancer in the failover region.

For applications that require robustness against region outages, consider using a multi-regional architecture. For more information, see Multi-regional deployment on Compute Engine.

MIG autoscaling

To control the autoscaling behavior of your stateless MIGs, you can specify target utilization metrics, such as average CPU utilization. You can also configure schedule-based autoscaling for stateless MIGs. Stateful MIGs can't be autoscaled. For more information, see Autoscaling groups of instances.

MIG size limit

When you decide the size of your MIGs, consider the default and maximum limits on the number of VMs that can be created in a MIG. For more information, see Add and remove VMs from a MIG.

VM autohealing

Sometimes the VMs that host your application might be running and available, but there might be issues with the application itself. The application might freeze, crash, or not have sufficient memory. To verify whether an application is responding as expected, you can configure application-based health checks as part of the autohealing policy of your MIGs. If the application on a particular VM isn't responding, the MIG autoheals (repairs) the VM. For more information about configuring autohealing, see About repairing VMs for high availability.

VM placement

In the architecture that this document describes, the application tier and web tier run on Compute Engine VMs that are distributed across multiple zones. This distribution ensures that your application is robust against zone outages.

To improve the robustness of the architecture, you can create a spread placement policy and apply it to the MIG template. When the MIG creates VMs, it places the VMs within each zone on different physical servers (called hosts), so your VMs are robust against failures of individual hosts. For more information, see Create and apply spread placement policies to VMs.

VM capacity planning

To make sure that capacity for Compute Engine VMs is available when VMs need to be provisioned, you can create reservations. A reservation provides assured capacity in a specific zone for a specified number of VMs of a machine type that you choose. A reservation can be specific to a project, or shared across multiple projects. For more information about reservations, see Choose a reservation type.

Stateful storage

A best practice in application design is to avoid the need for stateful local disks. But if the requirement exists, you can configure your persistent disks to be stateful to ensure that the data is preserved when the VMs are repaired or recreated. However, we recommend that you keep the boot disks stateless, so that you can update them to the latest images with new versions and security patches. For more information, see Configuring stateful persistent disks in MIGs.

Data durability

You can use Backup and DR to create, store, and manage backups of the Compute Engine VMs. Backup and DR stores backup data in its original, application-readable format. When required, you can restore your workloads to production by directly using data from long-term backup storage and avoid the need to prepare or move data.

Compute Engine provides the following options to help you to ensure the durability of data that's stored in Persistent Disk volumes:

You can use snapshots to capture the point-in-time state of Persistent Disk volumes. The snapshots are stored redundantly in multiple regions, with automatic checksums to ensure the integrity of your data. Snapshots are incremental by default, so they use less storage space and you save money. Snapshots are stored in a Cloud Storage location that you can configure. For more recommendations about using and managing snapshots, see Best practices for Compute Engine disk snapshots.

To ensure that data in Persistent Disk remains available if a zone outage occurs, you can use Regional Persistent Disk or Hyperdisk Balanced High Availability. Data in these disk types is replicated synchronously between two zones in the same region. For more information, see About synchronous disk replication.

If you use a managed database service like Cloud SQL, backups are taken automatically based on the retention policy that you define. You can supplement the backup strategy with additional logical backups to meet regulatory, workflow, or business requirements.

If you use a third-party database and you need to store database backups and transaction logs, you can use regional Cloud Storage buckets. Regional Cloud Storage buckets provide low-cost backup storage that's redundant across zones.

Database availability

If you use a managed database service like Cloud SQL in HA configuration, then in the event of a failure of the primary database, Cloud SQL fails over automatically to the standby database. You don't need to change the IP address for the database endpoint. If you use a self-managed third-party database that's deployed on a Compute Engine VM, then you must use an internal load balancer or other mechanism to ensure that the application can connect to another database if the primary database is unavailable.

To implement cross-zone failover for a database that's deployed on a Compute Engine VM, you need a mechanism to identify failures of the primary database and a process to fail over to the standby database. The specifics of the failover mechanism depend on the database that you use. You can set up an observer instance to detect failures of the primary database and orchestrate the failover. You must configure the failover rules appropriately to avoid a split-brain situation and prevent unnecessary failover. For example architectures that you can use to implement failover for PostgreSQL databases, see Architectures for high availability of PostgreSQL clusters on Compute Engine.

More reliability considerations

When you build the cloud architecture for your workload, review the reliability-related best practices and recommendations that are provided in the following documentation:

Google Cloud infrastructure reliability guide

Patterns for scalable and resilient apps

Designing resilient systems

Google Cloud Well-Architected Framework: Reliability

Cost optimization

This section provides guidance to optimize the cost of setting up and operating a regional Google Cloud topology that you build by using this reference architecture.

VM machine types

To help you optimize the resource utilization of your VM instances, Compute Engine provides machine type recommendations. Use the recommendations to choose machine types that match your workload's compute requirements. For workloads with predictable resource requirements, you can customize the machine type to your needs and save money by using custom machine types.

VM provisioning model

If your application is fault tolerant, then Spot VMs can help to reduce your Compute Engine costs for the VMs in the application and web tiers. The cost of Spot VMs is significantly lower than regular VMs. However, Compute Engine might preemptively stop or delete Spot VMs to reclaim capacity.

Spot VMs are suitable for batch jobs that can tolerate preemption and don't have high availability requirements. Spot VMs offer the same machine types, options, and performance as regular VMs. However, when the resource capacity in a zone is limited, MIGs might not be able to scale out (that is, create VMs) automatically to the specified target size until the required capacity becomes available again.

VM resource utilization

The autoscaling capability of stateless MIGs enables your application to handle increases in traffic gracefully, and it helps you to reduce cost when the need for resources is low. Stateful MIGs can't be autoscaled.

Third-party licensing

When you migrate third-party workloads to Google Cloud, you might be able to reduce cost by bringing your own licenses (BYOL). For example, to deploy Microsoft Windows Server VMs, instead of using a premium image that incurs additional cost for the third-party license, you can create and use a custom Windows BYOL image. You then pay only for the VM infrastructure that you use on Google Cloud. This strategy helps you continue to realize value from your existing investments in third-party licenses. If you decide to use the BYOL approach, then the following recommendations might help to reduce cost:

Provision the required number of compute CPU cores independently of memory by using custom machine types. By doing this, you limit the third-party licensing cost to the number of CPU cores that you need.

Reduce the number of vCPUs per core from 2 to 1 by disabling simultaneous multithreading (SMT).

If you deploy a third-party database like Microsoft SQL Server on Compute Engine VMs, then you must consider the license costs for the third-party software. When you use a managed database service like Cloud SQL, the database license costs are included in the charges for the service.

More cost considerations

When you build the architecture for your workload, also consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Cost optimization.

Operational efficiency

This section describes the factors that you should consider when you use this reference architecture to design and build a regional Google Cloud topology that you can operate efficiently.

VM configuration updates

To update the configuration of the VMs in a MIG (such as the machine type or boot-disk image), you create a new instance template with the required configuration and then apply the new template to the MIG. The MIG updates the VMs by using the update method that you choose: automatic or selective. Choose an appropriate method based on your requirements for availability and operational efficiency. For more information about these MIG update methods, see Apply new VM configurations in a MIG.

VM images

For your VMs, instead of using Google-provided public images, we recommend that you create and use custom OS images that contain the configurations and software that your applications require. You can group your custom images into a custom image family. An image family always points to the most recent image in that family, so your instance templates and scripts can use that image without you having to update references to a specific image version. You must regularly update your custom images to include the security updates and patches that are provided by the OS vendor.

Deterministic instance templates

If the instance templates that you use for your MIGs include startup scripts to install third-party software, make sure that the scripts explicitly specify software-installation parameters such as the software version. Otherwise, when the MIG creates the VMs, the software that's installed on the VMs might not be consistent. For example, if your instance template includes a startup script to install Apache HTTP Server 2.0 (the apache2 package), then make sure that the script specifies the exact apache2 version that should be installed, such as version 2.4.53. For more information, see Deterministic instance templates.

More operational considerations

When you build the architecture for your workload, consider the general best practices and recommendations for operational efficiency that are described in Google Cloud Well-Architected Framework: Operational excellence.

Performance optimization

This section describes the factors that you should consider when you use this reference architecture to design and build a regional topology in Google Cloud that meets the performance requirements of your workloads.

Compute performance

Compute Engine offers a wide range of predefined and customizable machine types for the workloads that you run on VMs. Choose an appropriate machine type based on your performance requirements. For more information, see Machine families resource and comparison guide.

VM multithreading

Each virtual CPU (vCPU) that you allocate to a Compute Engine VM is implemented as a single hardware multithread. By default, two vCPUs share a physical CPU core. For applications that involve highly parallel operations or that perform floating point calculations (such as genetic sequence analysis, and financial risk modeling), you can improve performance by reducing the number of threads that run on each physical CPU core. For more information, see Set the number of threads per core.

VM multithreading might have licensing implications for some third-party software, like databases. For more information, read the licensing documentation for the third-party software.

Network Service Tiers

Network Service Tiers lets you optimize the network cost and performance of your workloads. You can choose Premium Tier or Standard Tier. Premium Tier delivers traffic on Google's global backbone to achieve minimal packet loss and low latency. Standard Tier delivers traffic using peering, internet service providers (ISP), or transit networks at an edge point of presence (PoP) that's closest to the region where your Google Cloud workload runs. To optimize performance, we recommend using Premium Tier. To optimize cost, we recommend using Standard Tier.

Network performance

For workloads that need low inter-VM network latency within the application and web tiers, you can create a compact placement policy and apply it to the MIG template that's used for those tiers. When the MIG creates VMs, it places the VMs on physical servers that are close to each other. While a compact placement policy helps improve inter-VM network performance, a spread placement policy can help improve VM availability as described earlier. To achieve an optimal balance between network performance and availability, when you create a compact placement policy, you can specify how far apart the VMs must be placed. For more information, see Placement policies overview.

Compute Engine has a per-VM limit for egress network bandwidth. This limit depends on the VM's machine type and whether traffic is routed through the same VPC network as the source VM. For VMs with certain machine types, to improve network performance, you can get a higher maximum egress bandwidth by enabling Tier_1 networking.

More performance considerations

When you build the architecture for your workload, consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Performance optimization.

What's next

Learn more about the Google Cloud products used in this reference architecture:

Cloud Load Balancing overview

Instance groups

Get started with migrating your workloads to Google Cloud.

Explore and evaluate deployment archetypes that you can choose to build architectures for your cloud workloads.

Review architecture options for designing reliable infrastructure for your workloads in Google Cloud.

For more reference architectures, diagrams, and best practices, explore the Cloud Architecture Center.

Contributors

Authors:

Kumar Dhanagopal | Cross-Product Solution Developer

Samantha He | Technical Writer

Other contributors:

Ben Good | Solutions Architect

Carl Franklin | Director, PSO Enterprise Architecture

Daniel Lees | Cloud Security Architect

Gleb Otochkin | Cloud Advocate, Databases

Mark Schlagenhauf | Technical Writer, Networking

Pawel Wenda | Group Product Manager

Sean Derrington | Group Product Manager, Storage

Sekou Page | Outbound Product Manager

Simon Bennett | Group Product Manager

Steve McGhee | Reliability Advocate

Victor Moreno | Product Manager, Cloud Networking

Multi-regional deployment on Compute Engine

Release Notes

Last reviewed 2025-08-11 UTC

This document provides a reference architecture for a multi-tier application that runs on Compute Engine VMs in multiple regions in Google Cloud. You can use this reference architecture to efficiently rehost (lift and shift) on-premises applications to the cloud with minimal changes to the applications. The document also describes the design factors that you should consider when you build a multi-regional architecture for your cloud applications. The intended audience for this document is cloud architects.

Architecture

The following diagram shows an architecture for an application that runs in active-active mode in isolated stacks that are deployed across two Google Cloud regions. In each region, the application runs independently in three zones. The architecture is aligned with the Google Cloud multi-regional deployment archetype, which ensures that your Google Cloud topology is robust against zone and region outages and that it provides low latency for application users.

Multi-regional architecture using a global load balancer

The architecture is based on the infrastructure as a service (IaaS) cloud model. You provision the required infrastructure resources (compute, networking, and storage) in Google Cloud, and you retain full control over and responsibility for the operating system, middleware, and higher layers of the application stack. To learn more about IaaS and other cloud models, see PaaS vs. IaaS vs. SaaS vs. CaaS: How are they different?

The preceding diagram includes the following components:

Component	Purpose
Global external load balancer	The global external load balancer receives and distributes user requests to the application. The global external load balancer advertises a single anycast IP address, but it's implemented as a large number of proxies on Google Front Ends (GFEs). Client requests are directed to the GFE that's closest to the client.
Regional managed instance groups (MIGs) for the web tier	The web tier of the application is deployed on Compute Engine VMs that are part of regional MIGs. These MIGs are the backends for the global load balancer. Each MIG contains Compute Engine VMs in three different zones. Each of these VMs hosts an independent instance of the web tier of the application.

Regional internal load balancers

The internal load balancer in each region distributes traffic from the web tier VMs to the application tier VMs in that region.

Regional MIGs for the application tier

The application tier is deployed on Compute Engine VMs that are part of regional MIGs. The MIG in each region is the backend for the internal load balancer in that region.

Each MIG contains Compute Engine VMs in three different zones. Each VM hosts an independent instance of the application tier.

Third-party database deployed on Compute Engine VMs

The architecture in this document shows a third-party database (such as PostgreSQL) that's deployed on Compute Engine VMs in the two regions. You can set up cross-region replication for the databases and configure the database in each region to fail over to the database in the other region. The replication and failover capabilities depend on the database that you use.

Installing and managing a third-party database involves additional effort and operational cost for replication, applying updates, monitoring, and ensuring availability. You can avoid the overhead of installing and managing a third-party database and take advantage of built-in high availability (HA) features by using a fully managed database like a multi-region Spanner instance.

Virtual Private Cloud network and subnets

All the Google Cloud resources in the architecture use a single VPC network that has subnets in two different regions.

Depending on your requirements, you can choose to build an architecture that uses multiple VPC networks and subnets. For more information, see Deciding whether to create multiple VPC networks.

Cloud Storage dual-region buckets

Database backups are stored in dual-region Cloud Storage buckets. Alternatively, you can use Backup and DR Service to create, store, and manage the database backups.

Products used

This reference architecture uses the following Google Cloud products:

Compute Engine: A secure and customizable compute service that lets you create and run VMs on Google's infrastructure.

Cloud Load Balancing: A portfolio of high performance, scalable, global and regional load balancers.

Cloud Storage: A low-cost, no-limit object store for diverse data types. Data can be accessed from within and outside Google Cloud, and it's replicated across locations for redundancy.

Virtual Private Cloud (VPC): A virtual system that provides global, scalable networking functionality for your Google Cloud workloads. VPC includes VPC Network Peering, Private Service Connect, private services access, and Shared VPC.

Use cases

This section describes use cases for which a multi-regional deployment on Compute Engine is an appropriate choice.

Efficient migration of on-premises applications

You can use this reference architecture to build a Google Cloud topology to rehost (lift and shift) on-premises applications to the cloud with minimal changes to the applications. All the tiers of the application in this reference architecture are hosted on Compute Engine VMs. This approach lets you migrate on-premises applications efficiently to the cloud and take advantage of the cost benefits, reliability, performance, and operational simplicity that Google Cloud provides.

High availability for geographically dispersed users

We recommend a multi-regional deployment for applications that are business-critical and where high availability and robustness against region outages are essential. If a region becomes unavailable for any reason (even a large-scale disruption caused by a natural disaster), users of the application don't experience any downtime. Traffic is routed to the application in the other available regions. If data is replicated synchronously, the recovery time objective (RTO) is near zero.

Low latency for application users

If your users are within a specific geographical area, such as a continent, you can use a multi-regional deployment to achieve an optimal balance between availability and performance. When one of the regions has an outage, the global load balancer sends requests that originate in that region to another region. Users don't perceive significant performance impact because the regions are within a geographical area.

Design alternative

The preceding architecture uses a global load balancer, which supports certain features to enhance the reliability of your deployments, such as edge caching using Cloud CDN. This section presents an alternative architecture that uses regional load balancers and Cloud DNS. This alternative architecture supports the following additional features:

Transport Layer Security (TLS) termination in specified regions.

Ability to serve content from the region that you specify. However, that region might not be the best performing region at a given time.

A wider range of connection protocols if you use a Passthrough Network Load Balancer.

For more information about the differences between regional and global load balancers, see Global versus regional load balancing and Modes of operation.

Multi-regional architecture using regional load balancers and DNS.

The alternative architecture in the preceding diagram is robust against zone and region outages. A Cloud DNS public zone routes user requests to the appropriate region. Regional external load balancers receive user requests and distribute them across the web tier instances of the application within each region. The other components in this architecture are identical to the components in the global load balancer-based architecture.

Design considerations

This section provides guidance to help you use this reference architecture to develop an architecture that meets your specific requirements for system design, security, reliability, operational efficiency, cost, and performance.

Note: The guidance in this section isn't exhaustive. Depending on the specific requirements of your application and the Google Cloud products and features that you use, there might be additional design factors and trade-offs that you should consider.

When you build an architecture for your workload, consider the best practices and recommendations in the Google Cloud Well-Architected Framework.

System design

This section provides guidance to help you to choose Google Cloud regions for your multi-regional deployment and to select appropriate Google Cloud services.

Region selection

When you choose the Google Cloud regions where your applications must be deployed, consider the following factors and requirements:

Availability of Google Cloud services in each region. For more information, see Products available by location.

Availability of Compute Engine machine types in each region. For more information, see Regions and zones.

End-user latency requirements.

Cost of Google Cloud resources.

Cross-regional data transfer costs.

Regulatory requirements.

Some of these factors and requirements might involve trade-offs. For example, the most cost-efficient region might not have the lowest carbon footprint. For more information, see Best practices for Compute Engine regions selection.

Compute infrastructure

The reference architecture in this document uses Compute Engine VMs for certain tiers of the application. Depending on the requirements of your application, you can choose from other Google Cloud compute services:

Containers: You can run containerized applications in Google Kubernetes Engine (GKE) clusters. GKE is a container-orchestration engine that automates deploying, scaling, and managing containerized applications.

Serverless: If you prefer to focus your IT efforts on your data and applications instead of setting up and operating infrastructure resources, then you can use serverless services like Cloud Run.

The decision of whether to use VMs, containers, or serverless services involves a trade-off between configuration flexibility and management effort. VMs and containers provide more configuration flexibility, but you're responsible for managing the resources. In a serverless architecture, you deploy workloads to a preconfigured platform that requires minimal management effort. For more information about choosing appropriate compute services for your workloads in Google Cloud, see Hosting Applications on Google Cloud.

Storage services

The architectures shown in this document use regional Persistent Disk volumes for all the tiers. Persistent disks provide synchronous replication of data across two zones within a region.

Google Cloud Hyperdisk provides better performance, flexibility, and efficiency than Persistent Disk. With Hyperdisk Balanced, you can provision IOPS and throughput separately and dynamically, which lets you tune the volume to a wide variety of workloads.

For low-cost storage that's replicated across multiple locations, you can use Cloud Storage regional, dual-region, or multi-region buckets.

Data in regional buckets is replicated synchronously across the zones in the region.

Data in dual-region or multi-region buckets is stored redundantly in at least two separate geographic locations. Metadata is written synchronously across regions, and data is replicated asynchronously. For dual-region buckets, you can use turbo replication, which ensures that objects are replicated across region pairs, with a recovery point objective (RPO) of 15 minutes. For more information, see Data availability and durability.

To store data that's shared across multiple VMs in a region, such as across all the VMs in the web tier or application tier, you can use a Filestore regional instance. The data that you store in a Filestore regional instance is replicated synchronously across three zones within the region. This replication ensures high availability and robustness against zone outages. You can store shared configuration files, common tools and utilities, and centralized logs in the Filestore instance, and mount the instance on multiple VMs. For robustness against region outages, you can replicate a Filestore instance to a different region. For more information, see Instance replication.

If your database is Microsoft SQL Server, we recommend using Cloud SQL for SQL Server. In scenarios when Cloud SQL doesn't support your configuration requirements, or if you need access to the operating system, you can deploy a Microsoft SQL Server failover cluster instance (FCI). In this scenario, you can use the fully managed Google Cloud NetApp Volumes to provide continuous availability (CA) SMB storage for the database.

When you design storage for your workloads, consider the functional characteristics, resilience requirements, performance expectations, and cost goals. For more information, see Design an optimal storage strategy for your cloud workload.

Database services

The reference architecture in this document uses a third-party database that's deployed on Compute Engine VMs. Installing and managing a third-party database involves effort and cost for operations like applying updates, monitoring and ensuring availability, performing backups, and recovering from failures.

You can avoid the effort and cost of installing and managing a third-party database by using a fully managed database service like Cloud SQL, AlloyDB for PostgreSQL, Bigtable, Spanner, or Firestore. These Google Cloud database services provide uptime service-level agreements (SLAs), and they include default capabilities for scalability and observability.

If your workload needs an Oracle database, you can deploy the database on a Compute Engine VM or use Oracle Database@Google Cloud. For more information, see Oracle workloads in Google Cloud.

When you choose and set up the database for a multi-regional deployment, consider your application's requirements for cross-region data consistency, and be aware of the performance and cost trade-offs.

If the application requires strong consistency (all users must read the same data at all times), then the data must be replicated synchronously across all regions in the architecture. However, synchronous replication can lead to higher cost and decreased performance, because any data that's written must be replicated in real time across the regions before the data is available for read operations.

If your application can tolerate eventual consistency, then you can replicate data asynchronously. This can help improve performance because the data doesn't need to be replicated synchronously across regions. However, users in different regions might read different data because the data might not have been fully replicated at the time of the request.

Network design

Choose a network design that meets your business and technical requirements. You can use a single VPC network or multiple VPC networks. For more information, see the following documentation:

Deciding whether to create multiple VPC networks

Decide the network design for your Google Cloud landing zone

Security, privacy, and compliance

This section describes factors that you should consider when you use this reference architecture to design and build a multi-regional topology in Google Cloud that meets the security, privacy, and compliance requirements of your workloads.

Protection against external threats

To protect your application against threats like distributed-denial-of-service (DDoS) attacks and cross-site scripting (XSS), you can use Google Cloud Armor security policies. Each policy is a set of rules that specifies certain conditions that should be evaluated and actions to take when the conditions are met. For example, a rule could specify that if the source IP address of the incoming traffic matches a specific IP address or CIDR range, then the traffic must be denied. You can also apply preconfigured web application firewall (WAF) rules. For more information, see Security policy overview.

External access for VMs

In the reference architecture that this document describes, the Compute Engine VMs don't need inbound access from the internet. Don't assign external IP addresses to the VMs. Google Cloud resources that have only a private, internal IP address can still access certain Google APIs and services by using Private Service Connect or Private Google Access. For more information, see Private access options for services.

To enable secure outbound connections from Google Cloud resources that have only private IP addresses, like the Compute Engine VMs in this reference architecture, you can use Secure Web Proxy or Cloud NAT.

Service account privileges

For the Compute Engine VMs in the architecture, instead of using the default service accounts, we recommend that you create dedicated service accounts and specify the resources that the service account can access. The default service account has a broad range of permissions, including some that might not be necessary. You can tailor dedicated service accounts to have only the essential permissions. For more information, see Limit service account privileges.

SSH security

To enhance the security of SSH connections to the Compute Engine VMs in your architecture, implement Identity-Aware Proxy (IAP) and Cloud OS Login API. IAP lets you control network access based on user identity and Identity and Access Management (IAM) policies. Cloud OS Login API lets you control Linux SSH access based on user identity and IAM policies. For more information about managing network access, see Best practices for controlling SSH login access.

Disk encryption

By default, the data that's stored in Persistent Disk volumes is encrypted using Google-owned and Google-managed encryption keys. As an additional layer of protection, you can choose to encrypt the Google-owned and managed key by using keys that you own and manage in Cloud Key Management Service (Cloud KMS). For more information, see About disk encryption for Hyperdisk volumes and Encrypt data with customer-managed encryption keys.

Network security

To control network traffic between the resources in the architecture, you must configure appropriate Cloud Next Generation Firewall (NGFW) policies.

Data residency considerations

You can use regional load balancers to build a multi-regional architecture that helps you to meet data residency requirements. For example, a country in Europe might require that all user data be stored and accessed in data centers that are located physically within Europe. To meet this requirement, you can use the regional load balancer-based architecture. In that architecture, the application runs in Google Cloud regions in Europe and you use Cloud DNS with a geofenced routing policy to route traffic through regional load balancers. To meet data residency requirements for the database tier, use a sharded architecture instead of replication across regions. With this approach, the data in each region is isolated, but you can't implement cross-region high availability and failover for the database.

More security considerations

When you build the architecture for your workload, consider the platform-level security best practices and recommendations that are provided in the Enterprise foundations blueprint and Google Cloud Well-Architected Framework: Security, privacy, and compliance.

Reliability

This section describes design factors that you should consider when you use this reference architecture to build and operate reliable infrastructure for your multi-regional deployments in Google Cloud.

Robustness against infrastructure outages

In a multi-regional deployment architecture, if any individual component in the infrastructure stack fails, the application can process requests if at least one functioning component with adequate capacity exists in each tier. For example, if a web server instance fails, the load balancer forwards user requests to the other available web server instances. If a VM that hosts a web server or app server instance crashes, the MIG recreates the VM automatically.

If a zone outage occurs, the load balancer isn't affected, because it's a regional resource. A zone outage might affect individual Compute Engine VMs. But the application remains available and responsive because the VMs are in a regional MIG. A regional MIG ensures that new VMs are created automatically to maintain the configured minimum number of VMs. After Google resolves the zone outage, you must verify that the application runs as expected in all of the zones where it's deployed.

If all of the zones in one of the regions have an outage or if a region-wide outages occurs, then the application in the other region remains available and responsive. The global external load balancer directs traffic to the region that isn't affected by the outage. After Google resolves the region outage, you must verify that the application runs as expected in the region that had the outage.

If both of the regions in this architecture have outages, then the application is unavailable. You must wait for Google to resolve the outage, and then verify that the application works as expected.

MIG autoscaling

When you run your application on multiple regional MIGs, the application remains available during isolated zone outages or region outages. The autoscaling capability of stateless MIGs lets you maintain application availability and performance at predictable levels.

To control the autoscaling behavior of your stateless MIGs, you can specify target utilization metrics, such as average CPU utilization. You can also configure schedule-based autoscaling for stateless MIGs. Stateful MIGs can't be autoscaled. For more information, see Autoscaling groups of instances.

MIG size limit

When you decide the size of your MIGs, consider the default and maximum limits on the number of VMs that can be created in a MIG. For more information, see Add and remove VMs from a MIG.

VM autohealing

Sometimes the VMs that host your application might be running and available, but there might be issues with the application itself. The application might freeze, crash, or not have sufficient memory. To verify whether an application is responding as expected, you can configure application-based health checks as part of the autohealing policy of your MIGs. If the application on a particular VM isn't responding, the MIG autoheals (repairs) the VM. For more information about configuring autohealing, see About repairing VMs for high availability.

VM placement

In the architecture that this document describes, the application tier and web tier run on Compute Engine VMs that are distributed across multiple zones. This distribution ensures that your application is robust against zone outages.

To improve the robustness of the architecture, you can create a spread placement policy and apply it to the MIG template. When the MIG creates VMs, it places the VMs within each zone on different physical servers (called hosts), so your VMs are robust against failures of individual hosts. For more information, see Create and apply spread placement policies to VMs.

VM capacity planning

To make sure that capacity for Compute Engine VMs is available when VMs need to be provisioned, you can create reservations. A reservation provides assured capacity in a specific zone for a specified number of VMs of a machine type that you choose. A reservation can be specific to a project, or shared across multiple projects. For more information about reservations, see Choose a reservation type.

Stateful storage

A best practice in application design is to avoid the need for stateful local disks. But if the requirement exists, you can configure your persistent disks to be stateful to ensure that the data is preserved when the VMs are repaired or recreated. However, we recommend that you keep the boot disks stateless, so that you can update them to the latest images with new versions and security patches. For more information, see Configuring stateful persistent disks in MIGs.

Data durability

You can use Backup and DR to create, store, and manage backups of the Compute Engine VMs. Backup and DR stores backup data in its original, application-readable format. When required, you can restore your workloads to production by directly using data from long-term backup storage and avoid the need to prepare or move data.

Compute Engine provides the following options to help you to ensure the durability of data that's stored in Persistent Disk volumes:

You can use snapshots to capture the point-in-time state of Persistent Disk volumes. The snapshots are stored redundantly in multiple regions, with automatic checksums to ensure the integrity of your data. Snapshots are incremental by default, so they use less storage space and you save money. Snapshots are stored in a Cloud Storage location that you can configure. For more recommendations about using and managing snapshots, see Best practices for Compute Engine disk snapshots.

To ensure that data in Persistent Disk remains available if a zone outage occurs, you can use Regional Persistent Disk or Hyperdisk Balanced High Availability. Data in these disk types is replicated synchronously between two zones in the same region. For more information, see About synchronous disk replication.

Database availability

To implement cross-zone failover for a database that's deployed on a Compute Engine VM, you need a mechanism to identify failures of the primary database and a process to fail over to the standby database. The specifics of the failover mechanism depend on the database that you use. You can set up an observer instance to detect failures of the primary database and orchestrate the failover. You must configure the failover rules appropriately to avoid a split-brain situation and prevent unnecessary failover. For example architectures that you can use to implement failover for PostgreSQL databases, see Architectures for high availability of PostgreSQL clusters on Compute Engine.

More reliability considerations

When you build the cloud architecture for your workload, review the reliability-related best practices and recommendations that are provided in the following documentation:

Google Cloud infrastructure reliability guide

Patterns for scalable and resilient apps

Designing resilient systems

Google Cloud Well-Architected Framework: Reliability

Cost optimization

This section provides guidance to optimize the cost of setting up and operating a multi-regional Google Cloud topology that you build by using this reference architecture.

VM machine types

To help you optimize the resource utilization of your VM instances, Compute Engine provides machine type recommendations. Use the recommendations to choose machine types that match your workload's compute requirements. For workloads with predictable resource requirements, you can customize the machine type to your needs and save money by using custom machine types.

VM provisioning model

If your application is fault tolerant, then Spot VMs can help to reduce your Compute Engine costs for the VMs in the application and web tiers. The cost of Spot VMs is significantly lower than regular VMs. However, Compute Engine might preemptively stop or delete Spot VMs to reclaim capacity.

Spot VMs are suitable for batch jobs that can tolerate preemption and don't have high availability requirements. Spot VMs offer the same machine types, options, and performance as regular VMs. However, when the resource capacity in a zone is limited, MIGs might not be able to scale out (that is, create VMs) automatically to the specified target size until the required capacity becomes available again.

VM resource utilization

The autoscaling capability of stateless MIGs enables your application to handle increases in traffic gracefully, and it helps you to reduce cost when the need for resources is low. Stateful MIGs can't be autoscaled.

Third-party licensing

When you migrate third-party workloads to Google Cloud, you might be able to reduce cost by bringing your own licenses (BYOL). For example, to deploy Microsoft Windows Server VMs, instead of using a premium image that incurs additional cost for the third-party license, you can create and use a custom Windows BYOL image. You then pay only for the VM infrastructure that you use on Google Cloud. This strategy helps you continue to realize value from your existing investments in third-party licenses. If you decide to use the BYOL approach, then the following recommendations might help to reduce cost:

Provision the required number of compute CPU cores independently of memory by using custom machine types. By doing this, you limit the third-party licensing cost to the number of CPU cores that you need.

Reduce the number of vCPUs per core from 2 to 1 by disabling simultaneous multithreading (SMT).

If you deploy a third-party database like Microsoft SQL Server on Compute Engine VMs, then you must consider the license costs for the third-party software. When you use a managed database service like Cloud SQL, the database license costs are included in the charges for the service.

More cost considerations

When you build the architecture for your workload, also consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Cost optimization.

Operational efficiency

This section describes the factors that you should consider when you use this reference architecture to design and build a multi-regional Google Cloud topology that you can operate efficiently.

VM configuration updates

To update the configuration of the VMs in a MIG (such as the machine type or boot-disk image), you create a new instance template with the required configuration and then apply the new template to the MIG. The MIG updates the VMs by using the update method that you choose: automatic or selective. Choose an appropriate method based on your requirements for availability and operational efficiency. For more information about these MIG update methods, see Apply new VM configurations in a MIG.

VM images

For your VMs, instead of using Google-provided public images, we recommend that you create and use custom OS images that contain the configurations and software that your applications require. You can group your custom images into a custom image family. An image family always points to the most recent image in that family, so your instance templates and scripts can use that image without you having to update references to a specific image version. You must regularly update your custom images to include the security updates and patches that are provided by the OS vendor.

Deterministic instance templates

If the instance templates that you use for your MIGs include startup scripts to install third-party software, make sure that the scripts explicitly specify software-installation parameters such as the software version. Otherwise, when the MIG creates the VMs, the software that's installed on the VMs might not be consistent. For example, if your instance template includes a startup script to install Apache HTTP Server 2.0 (the apache2 package), then make sure that the script specifies the exact apache2 version that should be installed, such as version 2.4.53. For more information, see Deterministic instance templates.

More operational considerations

When you build the architecture for your workload, consider the general best practices and recommendations for operational efficiency that are described in Google Cloud Well-Architected Framework: Operational excellence.

Performance optimization

This section describes the factors that you should consider when you use this reference architecture to design and build a multi-regional topology in Google Cloud that meets the performance requirements of your workloads.

Compute performance

Compute Engine offers a wide range of predefined and customizable machine types for the workloads that you run on VMs. Choose an appropriate machine type based on your performance requirements. For more information, see Machine families resource and comparison guide.

VM multithreading

Each virtual CPU (vCPU) that you allocate to a Compute Engine VM is implemented as a single hardware multithread. By default, two vCPUs share a physical CPU core. For applications that involve highly parallel operations or that perform floating point calculations (such as genetic sequence analysis, and financial risk modeling), you can improve performance by reducing the number of threads that run on each physical CPU core. For more information, see Set the number of threads per core.

VM multithreading might have licensing implications for some third-party software, like databases. For more information, read the licensing documentation for the third-party software.

Network Service Tiers

Network Service Tiers lets you optimize the network cost and performance of your workloads. You can choose Premium Tier or Standard Tier. Premium Tier delivers traffic on Google's global backbone to achieve minimal packet loss and low latency. Standard Tier delivers traffic using peering, internet service providers (ISP), or transit networks at an edge point of presence (PoP) that's closest to the region where your Google Cloud workload runs. To optimize performance, we recommend using Premium Tier. To optimize cost, we recommend using Standard Tier.

Network performance

For workloads that need low inter-VM network latency within the application and web tiers, you can create a compact placement policy and apply it to the MIG template that's used for those tiers. When the MIG creates VMs, it places the VMs on physical servers that are close to each other. While a compact placement policy helps improve inter-VM network performance, a spread placement policy can help improve VM availability as described earlier. To achieve an optimal balance between network performance and availability, when you create a compact placement policy, you can specify how far apart the VMs must be placed. For more information, see Placement policies overview.

Compute Engine has a per-VM limit for egress network bandwidth. This limit depends on the VM's machine type and whether traffic is routed through the same VPC network as the source VM. For VMs with certain machine types, to improve network performance, you can get a higher maximum egress bandwidth by enabling Tier_1 networking.

Caching

If your application serves static website assets and if your architecture includes a global external Application Load Balancer, then you can use Cloud CDN to cache regularly accessed static content closer to your users. Cloud CDN can help to improve performance for your users, reduce your infrastructure resource usage in the backend, and reduce your network delivery costs. For more information, see Faster web performance and improved web protection for load balancing.

More performance considerations

When you build the architecture for your workload, consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Performance optimization.

What's next

Learn more about the Google Cloud products used in this reference architecture:

Cloud Load Balancing overview

Instance groups

Cloud DNS overview

Get started with migrating your workloads to Google Cloud.

Explore and evaluate deployment archetypes that you can choose to build architectures for your cloud workloads.

Review architecture options for designing reliable infrastructure for your workloads in Google Cloud.

For more reference architectures, design guides, and best practices, explore the Cloud Architecture Center.

Contributors

Authors:

Kumar Dhanagopal | Cross-Product Solution Developer

Samantha He | Technical Writer

Other contributors:

Ben Good | Solutions Architect

Carl Franklin | Director, PSO Enterprise Architecture

Daniel Lees | Cloud Security Architect

Gleb Otochkin | Cloud Advocate, Databases

Mark Schlagenhauf | Technical Writer, Networking

Pawel Wenda | Group Product Manager

Sean Derrington | Group Product Manager, Storage

Sekou Page | Outbound Product Manager

Shobhit Gupta | Solutions Architect

Simon Bennett | Group Product Manager

Steve McGhee | Reliability Advocate

Victor Moreno | Product Manager, Cloud Networking

Global deployment with Compute Engine and Spanner

Release Notes

Last reviewed 2025-08-12 UTC

This document provides a reference architecture for a multi-tier application that runs on Compute Engine VMs and Spanner in a global topology in Google Cloud. The document also provides guidance to help you build an architecture that uses other Google Cloud infrastructure services. It describes the design factors that you should consider when you build a global architecture for your cloud applications. The intended audience for this document is cloud architects.

This architecture is aligned with the global deployment archetype. We recommend this archetype for applications that serve users across the world and need high availability and robustness against outages in multiple regions. This architecture supports elastic scaling at the network, application, and database levels. It lets you align costs with usage without having to compromise on performance, availability, or scalability.

Architecture

The following diagram shows an architecture for an application that runs on infrastructure that's distributed globally across multiple Google Cloud regions.

Global deployment architecture using Compute Engine and Spanner.

In this architecture, a global load balancer distributes incoming requests to web servers in appropriate regions based on their availability, capacity, and proximity to the source of the traffic. A cross-regional internal load balancing layer handles distribution of traffic from the web servers to the appropriate application servers based on their availability and capacity. The application servers write data to, and read from, a synchronously replicated database that's available in all the regions.

The architecture includes the following Google Cloud resources:

Component	Purpose

Global external load balancer

The global external load balancer receives and distributes user requests to the application. The global external load balancer advertises a single anycast IP address, but the load balancer is implemented as a large number of proxies on Google Front Ends (GFEs). Client requests are directed to the GFE that's closest to the client.

Depending on your requirements, you can use a global external Application Load Balancer or a global external proxy Network Load Balancer. For more information, see Choose a load balancer.

To protect your application against threats like distributed denial-of-service (DDoS) attacks and cross-site scripting (XSS), you can use Google Cloud Armor security policies.

Regional managed instance groups (MIGs) for the web tier

The web tier of the application is deployed on Compute Engine VMs that are part of regional MIGs. These MIGs are the backends for the global load balancer.

Each MIG contains Compute Engine VMs in three different zones. Each of these VMs hosts an independent instance of the web tier of the application.

Cross-region internal load balancing layer

Internal load balancers with cross-regional backends handle the distribution of traffic from the web tier VMs in any region to the application tier VMs across all the regions.

Depending on your requirements, you can use a cross-region internal Application Load Balancer or a cross-region internal proxy Network Load Balancer. For more information, see Choose a load balancer.

Regional MIGs for the application tier

The application tier is deployed on Compute Engine VMs that are part of regional MIGs. These MIGs are the backends for the internal load balancing layer.

Each MIG contains Compute Engine VMs in three different zones. Each VM hosts an independent instance of the application tier.

Spanner multi-region instance

The application writes data to and reads from a multi-region Spanner instance. The multi-region configuration in this architecture includes the following replicas:

Four read-write replicas in separate zones across two regions.

A witness replica in a third region.

Virtual Private Cloud (VPC) network and subnets

All the resources in the architecture use a single VPC network. The VPC network has the following subnets:

A subnet in each region for the web server VMs.

A subnet in each region for the application server VMs.

(Not shown in the architecture diagram) A proxy-only subnet in each region for the cross-region internal load balancer.

Instead of using a single VPC network, you can create a separate VPC network in each region and connect the networks by using Network Connectivity Center.

Products used

This reference architecture uses the following Google Cloud products:

Compute Engine: A secure and customizable compute service that lets you create and run VMs on Google's infrastructure.

Cloud Load Balancing: A portfolio of high performance, scalable, global and regional load balancers.

Spanner: A highly scalable, globally consistent, relational database service.

Design considerations

This section provides guidance to help you use this reference architecture to develop an architecture that meets your specific requirements for system design, security and compliance, reliability, cost, operational efficiency, and performance.

Note: The guidance in this section isn't exhaustive. Depending on the specific requirements of your application and the Google Cloud products and features that you use, there might be additional design factors and trade-offs that you should consider.

System design

This section provides guidance to help you to choose Google Cloud regions for your global deployment and to select appropriate Google Cloud services.

Region selection

When you choose the Google Cloud regions where your applications must be deployed, consider the following factors and requirements:

Availability of Google Cloud services in each region. For more information, see Products available by location.

Availability of Compute Engine machine types in each region. For more information, see Regions and zones.

End-user latency requirements.

Cost of Google Cloud resources.

Cross-regional data transfer costs.

Regulatory requirements.

Some of these factors and requirements might involve trade-offs. For example, the most cost-efficient region might not have the lowest carbon footprint. For more information, see Best practices for Compute Engine regions selection.

Compute infrastructure

The reference architecture in this document uses Compute Engine VMs for certain tiers of the application. Depending on the requirements of your application, you can choose from other Google Cloud compute services:

Containers: You can run containerized applications in Google Kubernetes Engine (GKE) clusters. GKE is a container-orchestration engine that automates deploying, scaling, and managing containerized applications.

Serverless: If you prefer to focus your IT efforts on your data and applications instead of setting up and operating infrastructure resources, then you can use serverless services like Cloud Run.

The decision of whether to use VMs, containers, or serverless services involves a trade-off between configuration flexibility and management effort. VMs and containers provide more configuration flexibility, but you're responsible for managing the resources. In a serverless architecture, you deploy workloads to a preconfigured platform that requires minimal management effort. For more information about choosing appropriate compute services for your workloads in Google Cloud, see Hosting Applications on Google Cloud.

Storage services

The architecture shown in this document uses regional Persistent Disk volumes for the VMs. Regional Persistent Disk volumes provide synchronous replication of data across two zones within a region. Data in Persistent Disk volumes is not replicated across regions.

Google Cloud Hyperdisk provides better performance, flexibility, and efficiency than Persistent Disk. With Hyperdisk Balanced, you can provision IOPS and throughput separately and dynamically, which lets you tune the volume to a wide variety of workloads.

For low-cost storage that's replicated across multiple locations, you can use Cloud Storage regional, dual-region, or multi-region buckets.

Data in regional buckets is replicated synchronously across the zones in the region.

Data in dual-region or multi-region buckets is stored redundantly in at least two separate geographic locations. Metadata is written synchronously across regions, and data is replicated asynchronously. For dual-region buckets, you can use turbo replication, which ensures that objects are replicated across region pairs, with a recovery point objective (RPO) of 15 minutes. For more information, see Data availability and durability.

To store data that's shared across multiple VMs in a region, such as across all the VMs in the web tier or application tier, you can use a Filestore regional instance. The data that you store in a Filestore regional instance is replicated synchronously across three zones within the region. This replication ensures high availability and robustness against zone outages. You can store shared configuration files, common tools and utilities, and centralized logs in the Filestore instance, and mount the instance on multiple VMs. For robustness against region outages, you can replicate a Filestore instance to a different region. For more information, see Instance replication.

If your database is Microsoft SQL Server, we recommend using Cloud SQL for SQL Server. In scenarios when Cloud SQL doesn't support your configuration requirements, or if you need access to the operating system, you can deploy a Microsoft SQL Server failover cluster instance (FCI). In this scenario, you can use the fully managed Google Cloud NetApp Volumes to provide continuous availability (CA) SMB storage for the database.

When you design storage for your workloads, consider the functional characteristics, resilience requirements, performance expectations, and cost goals. For more information, see Design an optimal storage strategy for your cloud workload.

Database services

The reference architecture in this document uses Spanner, a fully managed, horizontally scalable, globally distributed, and synchronously-replicated database. We recommend a multi-regional Spanner configuration for mission-critical deployments that require strong cross-region consistency. Spanner supports synchronous cross-region replication without downtime for failover, maintenance, or resizing.

For information about other managed database services that you can choose from based on your requirements, see Google Cloud databases. When you choose and configure the database for a multi-regional deployment, consider your application's requirements for cross-region data consistency, and be aware of the performance and cost trade-offs.

Network design

Choose a network design that meets your business and technical requirements. You can use a single VPC network or multiple VPC networks. For more information, see the following documentation:

Deciding whether to create multiple VPC networks

Decide the network design for your Google Cloud landing zone

External load balancing options

An architecture that uses a global external load balancer, such as the architecture in this document, supports certain features that help you to enhance the reliability of your deployments. For example, if you use the global external Application Load Balancer, you can implement edge caching by using Cloud CDN.

If your application requires Transport Layer Security (TLS) to be terminated in a specific region, or if you need the ability to serve content from specific regions, you can use regional load balancers with Cloud DNS to route traffic to different regions. For information about the differences between regional and global load balancers, see the following documentation:

Global versus regional load balancing in "Choose a load balancer"

Modes of operation in "External Application Load Balancer overview"

Security, privacy, and compliance

This section describes factors that you should consider when you use this reference architecture to design and build a global topology in Google Cloud that meets the security, privacy, and compliance requirements of your workloads.

Protection against external threats

To protect your application against threats like distributed-denial-of-service (DDoS) attacks and cross-site scripting (XSS), you can use Google Cloud Armor security policies. Each policy is a set of rules that specifies certain conditions that should be evaluated and actions to take when the conditions are met. For example, a rule could specify that if the source IP address of the incoming traffic matches a specific IP address or CIDR range, then the traffic must be denied. You can also apply preconfigured web application firewall (WAF) rules. For more information, see Security policy overview.

External access for VMs

In the reference architecture that this document describes, the Compute Engine VMs don't need inbound access from the internet. Don't assign external IP addresses to the VMs. Google Cloud resources that have only a private, internal IP address can still access certain Google APIs and services by using Private Service Connect or Private Google Access. For more information, see Private access options for services.

To enable secure outbound connections from Google Cloud resources that have only private IP addresses, like the Compute Engine VMs in this reference architecture, you can use Secure Web Proxy or Cloud NAT.

Service account privileges

For the Compute Engine VMs in the architecture, instead of using the default service accounts, we recommend that you create dedicated service accounts and specify the resources that the service account can access. The default service account has a broad range of permissions, including some that might not be necessary. You can tailor dedicated service accounts to have only the essential permissions. For more information, see Limit service account privileges.

SSH security

To enhance the security of SSH connections to the Compute Engine VMs in your architecture, implement Identity-Aware Proxy (IAP) and Cloud OS Login API. IAP lets you control network access based on user identity and Identity and Access Management (IAM) policies. Cloud OS Login API lets you control Linux SSH access based on user identity and IAM policies. For more information about managing network access, see Best practices for controlling SSH login access.

More security considerations

When you build the architecture for your workload, consider the platform-level security best practices and recommendations that are provided in the Enterprise foundations blueprint and Google Cloud Well-Architected Framework: Security, privacy, and compliance.

Reliability

This section describes design factors that you should consider when you use this reference architecture to build and operate reliable infrastructure for a global deployment in Google Cloud.

MIG autoscaling

When you run your application on multiple regional MIGs, the application remains available during isolated zone outages or region outages. The autoscaling capability of stateless MIGs lets you maintain application availability and performance at predictable levels.

To control the autoscaling behavior of your stateless MIGs, you can specify target utilization metrics, such as average CPU utilization. You can also configure schedule-based autoscaling for stateless MIGs. Stateful MIGs can't be autoscaled. For more information, see Autoscaling groups of instances.

MIG size limit

When you decide the size of your MIGs, consider the default and maximum limits on the number of VMs that can be created in a MIG. For more information, see Add and remove VMs from a MIG.

VM autohealing

Sometimes the VMs that host your application might be running and available, but there might be issues with the application itself. The application might freeze, crash, or not have sufficient memory. To verify whether an application is responding as expected, you can configure application-based health checks as part of the autohealing policy of your MIGs. If the application on a particular VM isn't responding, the MIG autoheals (repairs) the VM. For more information about configuring autohealing, see About repairing VMs for high availability.

VM placement

In the architecture that this document describes, the application tier and web tier run on Compute Engine VMs that are distributed across multiple zones. This distribution ensures that your application is robust against zone outages.

To improve the robustness of the architecture, you can create a spread placement policy and apply it to the MIG template. When the MIG creates VMs, it places the VMs within each zone on different physical servers (called hosts), so your VMs are robust against failures of individual hosts. For more information, see Create and apply spread placement policies to VMs.

VM capacity planning

To make sure that capacity for Compute Engine VMs is available when VMs need to be provisioned, you can create reservations. A reservation provides assured capacity in a specific zone for a specified number of VMs of a machine type that you choose. A reservation can be specific to a project, or shared across multiple projects. For more information about reservations, see Choose a reservation type.

Stateful storage

A best practice in application design is to avoid the need for stateful local disks. But if the requirement exists, you can configure your persistent disks to be stateful to ensure that the data is preserved when the VMs are repaired or recreated. However, we recommend that you keep the boot disks stateless, so that you can update them to the latest images with new versions and security patches. For more information, see Configuring stateful persistent disks in MIGs.

Data durability

You can use Backup and DR to create, store, and manage backups of the Compute Engine VMs. Backup and DR stores backup data in its original, application-readable format. When required, you can restore your workloads to production by directly using data from long-term backup storage and avoid the need to prepare or move data.

Compute Engine provides the following options to help you to ensure the durability of data that's stored in Persistent Disk volumes:

You can use snapshots to capture the point-in-time state of Persistent Disk volumes. The snapshots are stored redundantly in multiple regions, with automatic checksums to ensure the integrity of your data. Snapshots are incremental by default, so they use less storage space and you save money. Snapshots are stored in a Cloud Storage location that you can configure. For more recommendations about using and managing snapshots, see Best practices for Compute Engine disk snapshots.

To ensure that data in Persistent Disk remains available if a zone outage occurs, you can use Regional Persistent Disk or Hyperdisk Balanced High Availability. Data in these disk types is replicated synchronously between two zones in the same region. For more information, see About synchronous disk replication.

Database reliability

Data that's stored in a multi-region Spanner instance is replicated synchronously across multiple regions. The Spanner configuration that's shown in the preceding architecture diagram includes the following replicas:

Four read-write replicas in separate zones across two regions.

A witness replica in a third region.

A write operation to a multi-region Spanner instance is acknowledged after at least three replicas—in separate zones across two regions—have committed the operation. If a zone or region failure occurs, Spanner has access to all of the data, including data from the latest write operations, and it continues to serve read and write requests.

Spanner uses disaggregated storage where the compute and storage resources are decoupled. You don't have to move data when you add compute capacity for HA or scaling. The new compute resources get data when they need it from the closest Colossus node. This makes failover and scaling faster and less risky.

Spanner provides external consistency, which is a stricter property than serializability for transaction-processing systems. For more information, see the following:

Spanner: TrueTime and external consistency

Demystifying Spanner multi-region configurations

Inside Spanner and the CAP Theorem

More reliability considerations

When you build the cloud architecture for your workload, review the reliability-related best practices and recommendations that are provided in the following documentation:

Google Cloud infrastructure reliability guide

Patterns for scalable and resilient apps

Designing resilient systems

Google Cloud Well-Architected Framework: Reliability

Cost optimization

This section provides guidance to optimize the cost of setting up and operating a global Google Cloud topology that you build by using this reference architecture.

VM machine types

To help you optimize the resource utilization of your VM instances, Compute Engine provides machine type recommendations. Use the recommendations to choose machine types that match your workload's compute requirements. For workloads with predictable resource requirements, you can customize the machine type to your needs and save money by using custom machine types.

VM provisioning model

If your application is fault tolerant, then Spot VMs can help to reduce your Compute Engine costs for the VMs in the application and web tiers. The cost of Spot VMs is significantly lower than regular VMs. However, Compute Engine might preemptively stop or delete Spot VMs to reclaim capacity.

Spot VMs are suitable for batch jobs that can tolerate preemption and don't have high availability requirements. Spot VMs offer the same machine types, options, and performance as regular VMs. However, when the resource capacity in a zone is limited, MIGs might not be able to scale out (that is, create VMs) automatically to the specified target size until the required capacity becomes available again.

VM resource utilization

The autoscaling capability of stateless MIGs enables your application to handle increases in traffic gracefully, and it helps you to reduce cost when the need for resources is low. Stateful MIGs can't be autoscaled.

Database cost

Spanner helps ensure that your database costs are predictable. The compute capacity that you specify (number of nodes or processing units) determines the storage capacity. The read and write throughputs scale linearly with compute capacity. You pay for only what you use. When you need to align costs with the needs of your workload, you can adjust the size of your Spanner instance.

Third-party licensing

When you migrate third-party workloads to Google Cloud, you might be able to reduce cost by bringing your own licenses (BYOL). For example, to deploy Microsoft Windows Server VMs, instead of using a premium image that incurs additional cost for the third-party license, you can create and use a custom Windows BYOL image. You then pay only for the VM infrastructure that you use on Google Cloud. This strategy helps you continue to realize value from your existing investments in third-party licenses. If you decide to use the BYOL approach, then the following recommendations might help to reduce cost:

Provision the required number of compute CPU cores independently of memory by using custom machine types. By doing this, you limit the third-party licensing cost to the number of CPU cores that you need.

Reduce the number of vCPUs per core from 2 to 1 by disabling simultaneous multithreading (SMT).

If you deploy a third-party database like Microsoft SQL Server on Compute Engine VMs, then you must consider the license costs for the third-party software. When you use a managed database service like Cloud SQL, the database license costs are included in the charges for the service.

More cost considerations

When you build the architecture for your workload, also consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Cost optimization.

Operational efficiency

This section describes the factors that you should consider when you use this reference architecture to design and build a global Google Cloud topology that you can operate efficiently.

VM configuration updates

To update the configuration of the VMs in a MIG (such as the machine type or boot-disk image), you create a new instance template with the required configuration and then apply the new template to the MIG. The MIG updates the VMs by using the update method that you choose: automatic or selective. Choose an appropriate method based on your requirements for availability and operational efficiency. For more information about these MIG update methods, see Apply new VM configurations in a MIG.

VM images

For your VMs, instead of using Google-provided public images, we recommend that you create and use custom OS images that contain the configurations and software that your applications require. You can group your custom images into a custom image family. An image family always points to the most recent image in that family, so your instance templates and scripts can use that image without you having to update references to a specific image version. You must regularly update your custom images to include the security updates and patches that are provided by the OS vendor.

Deterministic instance templates

If the instance templates that you use for your MIGs include startup scripts to install third-party software, make sure that the scripts explicitly specify software-installation parameters such as the software version. Otherwise, when the MIG creates the VMs, the software that's installed on the VMs might not be consistent. For example, if your instance template includes a startup script to install Apache HTTP Server 2.0 (the apache2 package), then make sure that the script specifies the exact apache2 version that should be installed, such as version 2.4.53. For more information, see Deterministic instance templates.

Migration to Spanner

You can migrate your data to Spanner from other databases like MySQL, SQL Server, and Oracle Database. The migration process depends on factors like the source database, the size of your data, downtime constraints, and complexity of the application code. To help you plan and implement the migration to Spanner efficiently, we provide a range of Google Cloud and third-party tools. For more information, see Migration overview.

Database administration

With Spanner, you don't need to configure or monitor replication or failover. Synchronous replication and automatic failover are built-in. Your application experiences zero downtime for database maintenance and failover. To further reduce operational complexity, you can configure autoscaling. With autoscaling enabled, you don't need to monitor and scale the instance size manually.

More operational considerations

When you build the architecture for your workload, consider the general best practices and recommendations for operational efficiency that are described in Google Cloud Well-Architected Framework: Operational excellence.

Performance optimization

This section describes the factors that you should consider when you use this reference architecture to design and build a global topology in Google Cloud that meets the performance requirements of your workloads.

Network performance

For workloads that need low inter-VM network latency within the application and web tiers, you can create a compact placement policy and apply it to the MIG template that's used for those tiers. When the MIG creates VMs, it places the VMs on physical servers that are close to each other. While a compact placement policy helps improve inter-VM network performance, a spread placement policy can help improve VM availability as described earlier. To achieve an optimal balance between network performance and availability, when you create a compact placement policy, you can specify how far apart the VMs must be placed. For more information, see Placement policies overview.

Compute Engine has a per-VM limit for egress network bandwidth. This limit depends on the VM's machine type and whether traffic is routed through the same VPC network as the source VM. For VMs with certain machine types, to improve network performance, you can get a higher maximum egress bandwidth by enabling Tier_1 networking.

Compute performance

Compute Engine offers a wide range of predefined and customizable machine types for the workloads that you run on VMs. Choose an appropriate machine type based on your performance requirements. For more information, see Machine families resource and comparison guide.

VM multithreading

Each virtual CPU (vCPU) that you allocate to a Compute Engine VM is implemented as a single hardware multithread. By default, two vCPUs share a physical CPU core. For applications that involve highly parallel operations or that perform floating point calculations (such as genetic sequence analysis, and financial risk modeling), you can improve performance by reducing the number of threads that run on each physical CPU core. For more information, see Set the number of threads per core.

VM multithreading might have licensing implications for some third-party software, like databases. For more information, read the licensing documentation for the third-party software.

Network Service Tiers

Network Service Tiers lets you optimize the network cost and performance of your workloads. You can choose Premium Tier or Standard Tier. Premium Tier delivers traffic on Google's global backbone to achieve minimal packet loss and low latency. Standard Tier delivers traffic using peering, internet service providers (ISP), or transit networks at an edge point of presence (PoP) that's closest to the region where your Google Cloud workload runs. To optimize performance, we recommend using Premium Tier. To optimize cost, we recommend using Standard Tier.

The architecture in this document uses a global external load balancer with an external IP address and backends in multiple regions. This architecture requires you to use Premium Tier, which uses Google's highly reliable global backbone to help you achieve minimal packet loss and latency.

If you use regional external load balancers and route traffic to regions by using Cloud DNS, then you can choose Premium Tier or Standard Tier depending on your requirements. The pricing for Standard Tier is lower than Premium Tier. Standard Tier is suitable for traffic that isn't sensitive to packet loss and that doesn't have low latency requirements.

Spanner performance

When you provision a Spanner instance, you specify the compute capacity of the instance in terms of the number of nodes or processing units. Monitor the resource utilization of your Spanner instance, and scale the capacity based on the expected load and your application's performance requirements. You can scale the capacity of a Spanner instance manually or automatically. For more information, see Autoscaling overview.

With a multi-region configuration, Spanner replicates data synchronously across multiple regions. This replication enables low-latency read operations from multiple locations. The trade-off is higher latency for write operations, because the quorum replicas are spread across multiple regions. To minimize the latency for read-write transactions in a multi-region configuration, Spanner uses leader-aware routing (enabled by default).

For recommendations to optimize the performance of your Spanner instance and databases, see the following documentation:

Performance best practices for multi-region configurations

Schema design best practices

Bulk loading best practices

Data Manipulation Language best practices

SQL best practices

Caching

If your application serves static website assets and if your architecture includes a global external Application Load Balancer, then you can use Cloud CDN to cache regularly accessed static content closer to your users. Cloud CDN can help to improve performance for your users, reduce your infrastructure resource usage in the backend, and reduce your network delivery costs. For more information, see Faster web performance and improved web protection for load balancing.

More performance considerations

When you build the architecture for your workload, consider the general best practices and recommendations that are provided in Google Cloud Well-Architected Framework: Performance optimization.

What's next

Learn more about the Google Cloud products used in this reference architecture:

Cloud Load Balancing

Compute Engine managed instance groups

Spanner multi-region configurations

Learn about replication and consistency in Spanner:

Demystifying Spanner multi-region configurations

Inside Spanner and the CAP Theorem

Get started with migrating your workloads to Google Cloud.

Explore and evaluate deployment archetypes that you can choose to build architectures for your cloud workloads.

Review architecture options for designing reliable infrastructure for your workloads in Google Cloud.

Deploy programmable GFEs using Cloud Armor, load balancing, and Cloud CDN.

For more reference architectures, diagrams, and best practices, explore the Cloud Architecture Center.

Contributors

Authors:

Kumar Dhanagopal | Cross-Product Solution Developer

Samantha He | Technical Writer

Other contributors:

Ben Good | Solutions Architect

Daniel Lees | Cloud Security Architect

Gleb Otochkin | Cloud Advocate, Databases

Justin Makeig | Product Manager

Mark Schlagenhauf | Technical Writer, Networking

Sekou Page | Outbound Product Manager

Steve McGhee | Reliability Advocate

Victor Moreno | Product Manager, Cloud Networking

Landing zone design in Google Cloud

Release Notes

Last reviewed 2024-10-31 UTC

This document provides an overview on how to design landing zones in Google Cloud. A landing zone, also called a cloud foundation, is a modular and scalable configuration that enables organizations to adopt Google Cloud for their business needs. A landing zone is often a prerequisite to deploying enterprise workloads in a cloud environment.

A landing zone is not a zone or zonal resources.

This document is aimed at solutions architects, technical practitioners, and executive stakeholders who want an overview of the following:

Typical elements of landing zones in Google Cloud

Where to find detailed information on landing zone design

How to deploy a landing zone for your enterprise, including options to deploy pre-built solutions

This document is part of a series that helps you understand how to design and build a landing zone. The other documents in this series help guide you through the high-level decisions that you need to make when you design your organization's landing zone. In this series, you learn about the following:

Landing zone design in Google Cloud (this document)

Decide how to onboard identities to Google Cloud

Decide the resource hierarchy for your Google Cloud landing zone

Decide the network design for your Google Cloud landing zone

Decide the security for your Google Cloud landing zone

This series does not specifically address compliance requirements from regulated industries such as financial services or healthcare.

What is a Google Cloud landing zone?

Landing zones help your enterprise deploy, use, and scale Google Cloud services more securely. Landing zones are dynamic and grow as your enterprise adopts more cloud-based workloads over time.

To deploy a landing zone, you must first create an organization resource and create a billing account, either online or invoiced.

A landing zone spans multiple areas and includes different elements, such as identities, resource management, security, and networking. Many other elements can also be part of a landing zone, as described in Elements of a landing zone.

The following diagram shows a sample implementation of a landing zone. It shows an Infrastructure as a Service (IaaS) use case with hybrid cloud and on-premises connectivity in Google Cloud:

Example architecture for a landing zone.

The example architecture in the preceding diagram shows a Google Cloud landing zone that includes the following Google Cloud services and features:

Resource Manager defines a resource hierarchy with organizational policies.

A Cloud Identity account synchronizes with an on-premises identity provider and Identity and Access Management (IAM) providing granular access to Google Cloud resources.

A network deployment that includes the following:

A Shared VPC network for each environment (production, development, and testing) connects resources from multiple projects to the VPC network.

Virtual Private Cloud (VPC) firewall rules control connectivity to and from workloads in the Shared VPC networks.

A Cloud NAT gateway allows outbound connections to the internet from resources in these networks without external IP addresses.

Cloud Interconnect connects on-premises applications and users. (You can choose between different Cloud Interconnect options, including Dedicated Interconnect or Partner Interconnect.)

Cloud VPN connects to other cloud service providers.

A Cloud DNS private zone hosts DNS records for your deployments in Google Cloud.

Multiple service projects are configured to use the Shared VPC networks. These service projects host your application resources.

Google Cloud Observability includes Cloud Monitoring for monitoring and Cloud Logging for logging. Cloud Audit Logs, Firewall Rules Logging and VPC Flow Logs help ensure all necessary data is logged and available for analysis.

A VPC Service Controls perimeter includes Shared VPC and the on-premises environment. A security perimeter isolates service and resources, which helps to mitigate the risk of data exfiltration from supported Google Cloud services.

The diagram above is only an example, because there is no single or standard implementation of a landing zone. Your business must make many design choices, depending on different factors, including the following:

Your industry

Your organizational structure and processes

Your security and compliance requirements

The workloads that you want to move to Google Cloud

Your existing IT infrastructure and other cloud environments

The location of your business and customers

When to build a landing zone

We recommend that you build a landing zone before you deploy your first enterprise workload on Google Cloud, because a landing zone provides the following:

A foundation that's designed to be secure

The network for enterprise workloads

The tools that you require to govern your internal cost distribution

However, because a landing zone is modular, your first iteration of a landing zone is often not your final version. Therefore, we recommend that you design a landing zone with scalability and growth in mind. For example, if your first workload does not require access to on-premises network resources, you could build connectivity to your on-premises environment later.

Depending on your organization and the type of workloads that you plan to run on Google Cloud, some workloads might have very different requirements. For example, some workloads might have unique scalability or compliance requirements. In these cases, you might require more than one landing zone for your organization: one landing zone to host most of the workloads and a separate landing zone to host the unique workloads. You can share some elements such as identities, billing, and the organization resource across your landing zones. However, other elements, such as the network setup, deployment mechanisms, and folder-level policies, might vary.

Elements of a landing zone

A landing zone requires you to design the following core elements on Google Cloud:

Identity provisioning

Resource hierarchy

Network

Security controls

In addition to these core elements, your business might have additional requirements. The following table describes these elements and where you can find more information about them.

Landing zone element	Description
Monitoring and logging	Design a monitoring and logging strategy that helps ensure all relevant data is logged and that you have dashboards that visualize the data and alerts that notify you of any actionable exceptions. For more information, see Google Cloud Observability documentation
Backup and disaster recovery	Design a strategy for backups and disaster recovery. For more information, see the following: Disaster recovery planning guide Backup and DR Service
Compliance	Follow the compliance frameworks that are relevant to your organization. For more information, see the Compliance resource center.
Cost efficiency and control	Design capabilities to monitor and optimize cost for workloads in your landing zone. For more information, see the following: Overview of cloud billing concepts

Google Cloud Well-Architected Framework: Cost optimization

Cost management

API management	Design a scalable solution for APIs that you develop. For more information, see Apigee API Management.

Cluster management

Design Google Kubernetes Engine (GKE) clusters that follow best practices to build scalable, resilient, and observable services.

For more information, see the following:

GKE best practices

GKE Autopilot mode

Multi-cluster Services

About Cloud Service Mesh

Best practices for designing and deploying a landing zone

Designing and deploying a landing zone requires planning. You must have the right team to perform the tasks, and use a project management process. We also recommend that you follow the technical best practices that are described in this series.

Build a team

Bring together a team that includes people from multiple technical functions across the organization. The team must include people who can build all landing zone elements, including security, identity, networks, and operations. Identify a cloud practitioner who understands Google Cloud to lead the team. Your team should include members who manage the project and track achievements, and members who collaborate with application or business owners.

Make sure that all stakeholders are involved early in the process. Your stakeholders must come to a common understanding of the scope of the process and make high-level decisions when the project gets kicked off.

Apply project management to your landing zone deployment

Designing and deploying your landing zone can take multiple weeks, so project management is essential. Ensure that project goals are clearly defined and communicated to all stakeholders and that all parties receive updates on any project changes. Define regular checkpoints and agree on milestones with realistic timelines that take operational processes and unexpected delays into account.

To best align with business requirements, plan the initial landing zone deployment around the use cases that you want to deploy first in Google Cloud. We recommend that you first deploy workloads that can most easily run on Google Cloud, such as horizontally scaling multi-tier web applications. These workloads might be new or existing workloads. To assess existing workloads for migration readiness, see Migration to Google Cloud: Getting started.

Because landing zones are modular, center the initial design around the elements that are required to migrate your first workloads and plan to add other elements later.

Follow technical best practices

Consider using Infrastructure as Code (IaC), with, for example, Terraform. IaC helps you make your deployment repeatable and modular. Having a CI/CD pipeline that deploys cloud infrastructure changes using GitOps helps you ensure that you follow internal guidelines and put the right controls in place.

When you design your landing zone, ensure that you and your team take technical best practices into consideration. For more information on decisions to make in your landing zone, see the other guides in this series.

In addition to this series, the following table describes frameworks, guides, and blueprints that can also help you follow best practices, depending on your use cases.

Related documentation	Description
Google Cloud Setup	A high-level guided flow to help you set up Google Cloud for scalable, production-ready, enterprise workloads.

Enterprise foundation blueprint	An opinionated view of Google Cloud security best practices, aimed at CISO, security practitioners, risk managers, or compliance officers.
Google Cloud Well-Architected Framework	Recommendations and best practices to help architects, developers, administrators, and other cloud practitioners design and operate a cloud topology that's secure, efficient, resilient, high-performing, and cost-effective.
Terraform blueprints	A list of blueprints and modules that are packaged as Terraform modules and that you can use to create resources for Google Cloud.

Identify resources to help implement your landing zone

Google Cloud offers the following options to help you set up your landing zone:

Design and deploy a landing zone that is customized to your requirements with Google Cloud partners or Google Cloud professional services.

Onboard a workload with the Google Cloud Customer Onboarding program.

Deploy a generic landing zone with the setup guide in the Google Cloud console.

Deploy a highly opinionated landing zone that is aligned to the security foundations blueprint by using the Terraform example foundation.

All these offerings have approaches that are designed specifically to meet the needs of different industries and business sizes, across the globe. To help you make the best selection for your use case, we recommend that you work with your Google Cloud account team to make the selection and help to ensure a successful project.

What's next

Decide how to onboard identities to Google Cloud (next document in this series).

Decide the resource hierarchy for your Google Cloud landing zone.

Decide the network design for your Google Cloud landing zone.

Decide the security for your Google Cloud landing zone.

Decide how to onboard identities to Google Cloud

Release Notes

Last reviewed 2024-10-31 UTC

This document describes identity provisioning options for Google Cloud and the decisions that you must make when you onboard your users to Cloud Identity or Google Workspace. This document also provides guidance on where to find more information on how to deploy each option.

This document is part of a series about landing zones, and is intended for architects and technical practitioners who are involved in managing identities for your organization and your Google Cloud deployment.

Overview

To let your organization's users access your Google Cloud resources, you must provide a way for them to authenticate themselves. Google Cloud uses Google Sign-In to authenticate users, which is the same identity provider (IdP) that other Google services such as Gmail or Google Ads use.

Although some users in your organization might already have a private Google user account, we strongly advise against letting them use their private accounts when they access Google Cloud. Instead, you can onboard your users to Cloud Identity or Google Workspace, which lets you control the lifecycle and security of user accounts.

Provisioning identities in Google Cloud is a complex topic and your exact strategy might require more detail than is in scope for this decision guide. For more best practices, planning, and deployment information, see overview of identity and access management.

Decision points for identity onboarding

To choose the best identity provisioning design for your organization, you must make the following decisions:

Your identity architecture

How to consolidate existing user accounts

Decide on your identity architecture

Managing the lifecycle and security of user accounts plays an important role in securing your Google Cloud deployment. A key decision that you must make is the role that Google Cloud should play in relation to your existing identity management systems and applications. The options are as follows:

Use Google as your primary identity provider (IdP).

Use federation with an external identity provider.

The following sections provide more information about each option.

Option 1: Use Google as your primary source for identities (no federation)

When you create user accounts directly in Cloud Identity or Google Workspace, you can make Google your source of identities and primary IdP. Users can then use these identities and credentials to sign in to Google Cloud and other Google services.

Cloud Identity and Google Workspace provide a large selection of ready-to-use integrations for popular third-party applications. You can also use standard protocols such as SAML, OAuth, and OpenID Connect to integrate your custom applications with Cloud Identity or Google Workspace.

Use this strategy when the following is true:

Your organization already has user identities provisioned in Google Workspace.

Your organization doesn't have an existing IdP.

Your organization has an existing IdP but wants to start quickly with a small subset of users and federate identities later.

Avoid this strategy when you have an existing IdP that you want to use as an authoritative source for identities.

For more information, see the following:

Preparing your Google Workspace or Cloud Identity Account

Using Google as an IdP

Option 2: Use federation with an external identity provider

You can integrate Google Cloud with an existing external IdP by using federation. Identity federation establishes trust between two or more IdPs so that the multiple identities that a user might have in different identity management systems can be linked.

When you federate a Cloud Identity or Google Workspace account with an external IdP, you let users use their existing identity and credentials to sign in to Google Cloud and other Google services.

Important: Super administrator accounts always bypass single-sign-on (SSO).

Use this strategy when the following is true:

You have an existing IdP such as Active Directory, Azure AD, ForgeRock, Okta, or Ping Identity.

You want employees to use their existing identity and credentials to sign in to Google Cloud and other Google services such as Google Ads and Google Marketing Platform.

Avoid this strategy when your organization doesn't have an existing IdP.

For more information, see the following:

External identities - Overview of Google identity management

Reference architectures: Using an external IdP

Best practices for federating Google Cloud with an external identity provider

Federating Google Cloud with Active Directory

Federating Google Cloud with Azure AD

Decide how to consolidate existing user accounts

If you haven't been using Cloud Identity or Google Workspace, it's possible that your organization's employees are using consumer accounts to access your Google services.

Consumer accounts are accounts that are fully owned and managed by the people who created them. Because those accounts are not under your organization's control and might include both personal and corporate data, you must decide how to consolidate these accounts with other corporate accounts.

For details on consumer accounts, how to identify them, and what risk they might pose to your organization, see Assessing existing user accounts.

The options for consolidating the accounts are as follows:

Consolidate a relevant subset of consumer accounts.

Consolidate all accounts through migration.

Consolidate all accounts through eviction, by not migrating accounts before creating new ones.

The following sections provide more information about each option.

Option 1: Consolidate a relevant subset of consumer accounts

If you want to keep consumer accounts and manage them and their data under corporate policies, you must migrate them to Cloud Identity or Google Workspace. However, the process of consolidating consumer accounts can be time consuming. Therefore, we recommend that you first evaluate which subset of users are relevant for your planned Google Cloud deployment, and then consolidate only those user accounts.

Use this strategy when the following is true:

The transfer tool for unmanaged user accounts shows many consumer user accounts in your domain, but only a subset of your users will use Google Cloud.

You want to save time in the consolidation process.

Avoid this strategy when the following is true:

You don't have consumer user accounts in your domain.

You want to ensure that all data from all consumer user accounts in your domain is consolidated to managed accounts before you start using Google Cloud.

For more information, see Overview of consolidating accounts.

Option 2: Consolidate all accounts through migration

If you want to manage all user accounts in your domain, you can consolidate all consumer accounts by migrating them to managed accounts.

Use this strategy when the following is true:

The transfer tool for unmanaged user accounts shows only a few consumer accounts in your domain.

You want to restrict the use of consumer accounts in your organization.

Avoid this strategy when you want to save time in the consolidation process.

For more information, see Migrating consumer accounts.

Option 3: Consolidate all accounts through eviction

You can evict consumer accounts in the following circumstances:

You want users who created consumer accounts to keep full control over their accounts and data.

You don't want to transfer any data to be managed by your organization.

To evict consumer accounts, create a managed user identity of the same name without migrating the user account first.

Use this strategy when the following is true:

You want to create new managed accounts for your users without transferring any of the data that exists in their consumer accounts.

You want to restrict the Google services that are available in your organization. You also want users to keep their data and keep using these services for the consumer accounts that they created.

Avoid this strategy when consumer accounts have been used for corporate purposes and might have access to corporate data.

For more information, see Evicting consumer accounts.

Best practices for onboarding identities

After you choose your identity architecture and your method to consolidate existing consumer accounts, consider the following identity best practices.

Select a suitable onboarding plan that works for your organization

Select a high-level plan to onboard your organization's identities to Cloud Identity or Google Workspace. For a selection of proven onboarding plans, along with guidance on how to select the plan that best suits your needs, see Assessing onboarding plans.

If you plan to use an external IdP and have identified user accounts that need to be migrated, you might have additional requirements. For more information, see Assessing the impact of user account consolidation on federation.

Protect user accounts

After you've onboarded users to Cloud Identity or Google Workspace, you must put measures in place to help protect their accounts from abuse. For more information, see the following:

Implement security best practices for Cloud Identity administrative accounts.

Enforce uniform multifactor authentication rules and follow best practices when combined with federating identities.

Export your Google Workspace or Cloud Identity audit logs to Cloud Logging by enabling data sharing.

What's next

Decide your resource hierarchy (next document in this series).

Find out more about how users, Cloud Identity accounts, and Google Cloud organizations relate.

Review recommended best practices for planning accounts and organizations.

Read about best practices for federating Google Cloud with an external identity provider.

Decide a resource hierarchy for your Google Cloud landing zone

Release Notes

Last reviewed 2024-10-31 UTC

A resource hierarchy helps to organize your resources in Google Cloud. This document describes how to choose your resource hierarchy as part of your landing zone design. It's intended for cloud system administrators, architects, and technical practitioners who are involved in designing the resource hierarchy. This document is part of a series on landing zones. It includes sample hierarchies that demonstrate common ways that businesses can structure resources in Google Cloud.

Design factors for resource hierarchy

When you define your resource hierarchy in Google Cloud, you must consider how your organization works today and the ideal end state of your cloud transformation. The best way to manage resources is based on your organization's intended way of working in the cloud. As every organization is different, there is no single, best approach for resource hierarchy.

However, we recommend that you avoid mapping your corporate organization structure to the resource hierarchy. Instead, focus on your business needs and operations in Google Cloud.

Google Cloud resource hierarchy

The resource hierarchy in Google Cloud starts at the root node, which is called the *organization*. We recommend that businesses have only one root node, except for in specific situations. You define lower levels of the hierarchy using *folders* and *projects*, and you create folders within folders to build your hierarchy. You can create the projects that host your workloads at any level of the hierarchy.

The following diagram shows a root node called *Organization*, and folders at levels one, two, and three. Projects and subfolders are created under the folders in level two.

Resource hierarchy factors

When you decide your resource hierarchy, consider the following factors:

- Who is responsible for cloud resources? Is it your departments, subsidiaries, technical teams, or legal entities?
- What are your compliance needs?
- Do you have upcoming business events, such as mergers, acquisitions, and spin-offs?

Understand resource interactions throughout the hierarchy

Organization policies are inherited by descendants in the resource hierarchy, but can be superseded by policies defined at a lower level. For more information, see understanding hierarchy evaluation. You use organization policy constraints to set guidelines around the whole organization or significant parts of it and still allow for exceptions.

Allow policies, formerly known as IAM policies are inherited by descendants, and allow policies at lower levels are additive. However, allow policies can be superseded by deny policies, which let you restrict permissions at the project, folder, and organization level. Deny policies are applied before allow policies.

You also need to consider the following:

- Cloud Logging includes aggregated sinks that you can use to aggregate logs at the folder or organization level. For more information, see decide the security for your Google Cloud landing zone.
- Billing is not directly linked to the resource hierarchy, but assigned at the project level. However, to get aggregated information at the folder level, you can analyze your costs by project hierarchy using billing reports.
- Hierarchical firewall policies let you implement consistent firewall policies throughout the organization or in specific folders. Inheritance is implicit, which means that you can allow or deny traffic at any level or you can delegate the decision to a lower level.

Decision points for resource hierarchy design

The following flowchart shows the things that you must consider to choose the best resource hierarchy for your organization.

The preceding diagram outlines the following decision points:

1. Do different subsidiaries, regional groups, or business units have very different policy requirements?
    a. If yes, follow the design based on region or subsidiaries.
    b. If no, go to the next decision point.
2. Do your workload or product teams require strong autonomy over their policies? For example, you don't have a central security team that determines policies for all workload or product teams.
    a. If yes, see the design based on accountability framework.
    b. If no, see the design based on application environment.

Your specific use case might lead you to another resource hierarchy design than what the decision chart suggests. Most organizations choose a hybrid approach and select different designs at different levels of the resource hierarchy, starting with the design that most affects policies and access.

Option 1: Hierarchy based on application environments

In many organizations, you define different policies and access controls for different application environments, such as development, production, and testing. Having separate policies that are standardized across each environment eases management and configuration. For example, you might have security policies that are more stringent in production environments than in testing environments.

Use a hierarchy based on application environments if the following is true:

- You have separate application environments that have different policy and administration requirements.
- You have centralized security or audit requirements that a central security team must be able to enforce consistently on all production workloads and data.
- You require different Identity and Access Management (IAM) roles to access your Google Cloud resources in different environments.

Avoid this hierarchy if the following is true:

- You don't run multiple application environments.

- You don't have varying application dependencies and business processes across environments.
- You have strong policy differences for different regions, teams, or applications.

The following diagram shows a hierarchy for example.com, which is a fictitious financial technology company.

As shown in the preceding diagram, example.com has three application environments that have different policies, access controls, regulatory requirements, and processes. The environments are as follows:

- **Development and QA environment:** This environment is managed by developers who are both internal employees and consultants. They continuously push code and are responsible for quality assurance. This environment is never available to your business' consumers. The environment has less strict integration and authentication requirements than the production environment, and developers are assigned approved roles with suitable permissions. The Development and QA environment is designed only for standard application offerings from example.com.
- **Testing environment:** This environment is used for regression and application testing, and supports the business-to-business (B2B) offerings of example.com clients who use example.com APIs. For this purpose, example.com creates two project types:
    - Internal testing projects to complete internal regression, performance, and configuration for B2B offerings.
    - Client UAT projects with multi-tenant support so that B2B clients can validate their configurations and align with example.com requirements for user experience, branding, workflows, reports, and so on.
- **Production environment:** This environment hosts all product offerings that are validated, accepted, and launched. This environment is subject to Payment Card Industry Data Security Standard (PCI DSS) regulations, uses hardware security modules (HSMs), and integrates with third-party processors for items such as authentication and payment settlements. The audit and compliance teams are critical stakeholders of this environment. Access to this environment is tightly controlled and limited mostly to automated deployment processes.

For more information about designing a hierarchy that is based on application environments, see Organization structure in the enterprise foundations blueprint.

Option 2: Hierarchy based on regions or subsidiaries

Some organizations operate across many regions and have subsidiaries doing business in different geographies or have been a result of mergers and acquisitions. These organizations require a resource hierarchy that uses the scalability and management options in Google Cloud, and maintains the independence for different processes and policies that exist between the regions or subsidiaries. This hierarchy uses subsidiaries or regions as the highest folder level in the resource hierarchy. Deployment procedures are typically focused around the regions.

Use this hierarchy if the following is true:

- Different regions or subsidiaries operate independently.
- Regions or subsidiaries have different operational backbones, digital platform offerings, and processes.
- Your business has different regulatory and compliance standards for regions or subsidiaries.

The following diagram shows an example hierarchy of a global organization with two subsidiaries and a holding group with a regionalized structure.

The preceding diagram has the following hierarchy structure:

- The following folders are on the first level:
    - The Subsidiary 1 and Subsidiary 2 folders represent the two subsidiaries of the company that have different access permissions and policies from the rest of the organization. Each subsidiary uses IAM to restrict access to their projects and Google Cloud resources.
    - The Holding group folder represents the groups that have high-level common policies across regions.
    - The Bootstrap folder represents the common resources that are required to deploy your Google Cloud infrastructure, as described in the enterprise foundations blueprint.
- On the second level, within the *Group Holding* folder, there are the following folders:
    - The APAC, EMEA, and Americas folders represent the various regions within the group that have different governance, access permissions, and policies.

    - The Shared infrastructure folder represents resources that are used globally across all regions.
- Within each folder are various projects that contain the resources that these subsidiaries or regions are responsible for.

You can add more folders to separate different legal entities, departments, and teams within your company.

Option 3: Hierarchy based on an accountability framework

A hierarchy based on an accountability framework works best when your products are run independently or organizational units have clearly defined teams who own the lifecycle of the products. In these organizations, the product owners are responsible for the entire product lifecycle, including its processes, support, policies, security, and access rights. Your products are independent from each other, and organization-wide guidelines and controls are uncommon.

Use this hierarchy when the following is true:

- You run an organization that has clear ownership and accountability for each product.
- Your workloads are independent and don't share many common policies.
- Your processes and external developer platforms are offered as service or product offerings.

The following diagram shows an example hierarchy for an ecommerce platform provider.

The preceding diagram has the following hierarchy structure:

- The following folders are on the first level:
    - The folders that are named Ecommerce Modules and Logistics and Warehousing Modules represent the modules within the platform offering that require the same access permissions and policies during the product lifecycle.
    - The Reconciliation and Billing folder represents the product teams who are responsible for the end-to-end modules for specific business components within the platform offering.

    - The Bootstrap folder represents the common resources that are required to deploy your Google Cloud infrastructure, as described in the enterprise foundations blueprint.

- Within each folder are various projects that contain the independent modules that different product teams are responsible for.

For more information, see Fabric FAST Terraform framework resource hierarchy.

Best practices for resource hierarchy

The following sections describe the best practices for designing resource hierarchy that we recommend, regardless of the resource hierarchy that you choose.

For more best practices on how to configure your Cloud Identity and Google Workspace accounts and organizations, see Best practices for planning accounts and organizations.

Use a single organization node

To avoid management overhead, use a single organization node whenever possible. However, consider using multiple organization nodes to address the following use cases:

- You want to test major changes to your IAM levels or resource hierarchy.
- You want to experiment in a sandbox environment that doesn't have the same organization policies.
- Your organization includes sub-companies that are likely to be sold off or run as completely separate entities in the future.

Use standardized naming conventions

Use a standardized naming convention throughout your organization. The security foundations blueprint has a sample naming convention that you can adapt to your requirements.

Keep bootstrapping resources and common services separate

Keep separate folders for bootstrapping the Google Cloud environment using infrastructure-as-code (IaC) and for common services that are shared between environments or applications. Place the bootstrap folder right below the organization node in the resource hierarchy.

Place the folders for common services at different levels of the hierarchy, depending on the structure that you choose.

Place the folder for common services right below the organization node when the following is true:

- Your hierarchy uses application environments at the highest level and teams or applications at the second layer.
- You have shared services such as monitoring that are common between environments.

Place the folder for common services at a lower level, below the application folders, when the following is true:

- You have services that are shared between applications and you deploy a separate instance for each deployment environment.
- Applications share microservices that require development and testing for each deployment environment.

The following diagram shows an example hierarchy where there is a shared infrastructure folder that is used by all environments and shared services folders for each application environment at a lower level in the hierarchy:

The preceding diagram has the following hierarchy structure:

- The folders on the first level are as follows:
    - The Development environment and Production environment folders contain the application environments.
    - The Shared infrastructure folder contains common resources that are shared across environments, such as monitoring.
    - The Bootstrap folder contains the common resources required to deploy your Google Cloud infrastructure, as described in the enterprise foundations blueprint.
- On the second level, there are the following folders:
    - A folder in each environment for each application (App 1 and App 2) which contains the resources for these applications.
    - A Shared folder for both application environments that contains services that are shared between the applications but are independent for each environment. For example, you might have a folder-level secrets project so that you can apply different allow policies to your production secrets and non-production secrets.

- Within each application folder are various projects that contain the independent modules that are part of each application.

What's next

- Design the network for your landing zone (the next document in this series).
- Review the enterprise foundations blueprint.
- Read the blueprints and whitepapers that are available in the Google Cloud security best practices center.

Decide the network design for your Google Cloud landing zone

Release Notes

Last reviewed 2024-10-31 UTC

When you design your landing zone, you must choose a network design that works for your organization. This document describes four common network designs, and helps you choose the option that best meets your organization's requirements, and your organization's preference for centralized control or decentralized control. It's intended for network engineers, architects, and technical practitioners who are involved in creating the network design for your organization's landing zone.

This article is part of a series about landing zones.

Choose your network design

The network design that you choose depends primarily on the following factors:

- **Centralized or decentralized control:** Depending on your organization's preferences, you must choose one of the following:
    - Centralize control over the network including IP addressing, routing, and firewalling between different workloads.
    - Give your teams greater autonomy in running their own environments and building network elements within their environments themselves.
- **On-premises or hybrid cloud connectivity options:** All the network designs discussed in this document provide access from on-premises to cloud environments through Cloud VPN or Cloud Interconnect. However, some designs require you to set up multiple connections in parallel, while others use the same connection for all workloads.
- **Security requirements:** Your organization might require traffic between different workloads in Google Cloud to pass through centralized network appliances such

as next generation firewalls (NGFW). This constraint influences your Virtual Private Cloud (VPC) network design.

- **Scalability:** Some designs might be better for your organization than others, based on the number of workloads that you want to deploy, and the number of virtual machines (VMs), internal load balancers, and other resources that they will consume.

Decision points for network design

The following flowchart shows the decisions that you must make to choose the best network design for your organization.

The preceding diagram guides you through the following questions:

1. Do you require Layer 7 inspection using network appliances between different workloads in Google Cloud?
    - If yes, see Hub-and-spoke topology with centralized appliances.
    - If no, proceed to the next question.
2. Do many of your workloads require on-premises connectivity?
    - If yes, go to decision point 4.
    - If no, proceed to the next question.
3. Can your workloads communicate using private endpoints in a service producer and consumer model?
    - If yes, see Expose services in a consumer-producer model with Private Service Connect.
    - If no, proceed to the next question.
4. Do you want to administer firewalling and routing centrally?
    - If yes, see Shared VPC network for each environment.
    - If no, see Hub-and-spoke topology without appliances.

This chart is intended to help you make a decision, however, it can often be the case that multiple designs might be suitable for your organization. In these instances, we recommend that you choose the design that fits best with your use case.

Network design options

The following sections describe four common design options. We recommend option 1 for most use cases. The other designs discussed in this section are alternatives that apply to specific organizational edge-case requirements.

The best fit for your use case might also be a network that combines elements from multiple design options discussed in this section. For example, you can use Shared VPC networks in hub-and-spoke topologies for better collaboration, centralized control, and to limit the number of VPC spokes. Or, you might design most workloads in a Shared VPC topology but isolate a small number of workloads in separate VPC networks that only expose services through a few defined endpoints using Private Service Connect.

Note: When the design options refer to connections to on-premises networks, you can use the same concepts for connections to other cloud service providers (CSPs).

Option 1: Shared VPC network for each environment

We recommend this network design for most use cases. This design uses separate Shared VPC networks for each deployment environment that you have in Google Cloud (development, testing, and production). This design lets you centrally manage network resources in a common network and provides network isolation between the different environments.

Use this design when the following is true:

- You want central control over firewalling and routing rules.
- You need a simple, scalable infrastructure.
- You need centralized IP address space management.

Avoid this design when the following is true:

- You want developer teams to have full autonomy, including the ability to manage their own firewall rules, routing, and peering to other team networks.
- You need Layer 7 inspection using NGFW appliances.

The following diagram shows an example implementation of this design.

The preceding diagram shows the following:

- The on-premises network is spread across two geographical locations.
- The on-premises network connects through redundant Cloud Interconnect instances to two separate Shared VPC networks, one for production and one for development.

- The production and development environments are connected to both Cloud Interconnect instances with different VLAN attachments.
- Each Shared VPC has service projects that host the workloads.
- Firewall rules are centrally administered in the host project.
- The development environment has the same VPC structure as the production environment.

By design, traffic from one environment cannot reach another environment. However, if specific workloads must communicate with each other, you can allow data transfer through controlled channels on-premises, or you can share data between applications with Google Cloud services like Cloud Storage or Pub/Sub. We recommend that you avoid directly connecting separated environments through VPC Network Peering, because it increases the risk of accidentally mixing data between the environments. Using VPC Network Peering between large environments also increases the risk of hitting VPC quotas around peering and peering groups.

For more information, see the following:

- Shared VPC overview
- Shared VPC architecture in the enterprise foundations guide
- Reference architecture in VPC design best practices
- Terraform deployment stage: Networking with separate environments as part of Fabric FAST framework
- Network stage for Terraform example foundation using Cloud Foundation toolkit

To implement this design option, see Create option 1: Shared VPC network for each environment.

Option 2: Hub-and-spoke topology with centralized appliances

This network design uses hub-and-spoke topology. A hub VPC network contains a set of appliance VMs such as NGFWs that are connected to the spoke VPC networks that contain the workloads. Traffic between the workloads, on-premises networks, or the internet is routed through appliance VMs for inspection and filtering.

Use this design when the following is true:

- You require Layer 7 inspection between different workloads or applications.
- You have a corporate mandate that specifies the security appliance vendor for all traffic.

Avoid this design when the following is true:

- You don't require Layer 7 inspection for most of your workloads.
- You want workloads on Google Cloud to not communicate at all with each other.
- You only need Layer 7 inspection for traffic going to on-premises networks.

The following diagram shows an example implementation of this pattern.

The preceding diagram shows the following:

- A production environment which includes a hub VPC network and multiple spoke VPC networks that contain the workloads.
- The spoke VPC networks are connected with the hub VPC network by using VPC Network Peering.
- The hub VPC network has multiple instances of a virtual appliance in a managed instance group. Traffic to the managed instance group goes through an internal passthrough Network Load Balancer.
- The spoke VPC networks communicate with each other through the virtual appliances by using static routes with the internal load balancer as the next hop.
- Cloud Interconnect connects the transit VPC networks to on-premises locations.
- On-premises networks are connected through the same Cloud Interconnects using separate VLAN attachments.
- The transit VPC networks are connected to a separate network interface on the virtual appliances, which lets you inspect and filter all traffic to and from these networks by using your appliance.
- The development environment has the same VPC structure as the production environment.
- This setup doesn't use source network address translation (SNAT). SNAT isn't required because Google Cloud uses *symmetric hashing*. For more information see Symmetric hashing.

By design, traffic from one spoke network cannot reach another spoke network. However, if specific workloads must communicate with each other, you can set up direct peering between the spoke networks using VPC Network Peering, or you can share data between applications with Google Cloud services like Cloud Storage or Pub/Sub.

To maintain low latency when the appliance communicates between workloads, the appliance must be in the same region as the workloads. If you use multiple regions in your cloud deployment, you can have one set of appliances and one hub VPC for each environment in each region. Alternatively, you can use network tags with routes to have all instances communicate with the closest appliance.

Firewall rules can restrict the connectivity within the spoke VPC networks that contain workloads. Often, teams who administer the workloads also administer these firewall rules. For central policies, you can use hierarchical firewall policies. If you require a central network team to have full control over firewall rules, consider centrally deploying those rules in all VPC networks by using a GitOps approach. In this case, restrict the IAM permissions to only those administrators who can change the firewall rules. Spoke VPC networks can also be Shared VPC networks if multiple teams deploy in the spokes.

In this design, we recommend that you use VPC Network Peering to connect the hub VPC network and spoke VPC networks because it adds minimum complexity. However, the maximum number of spokes is limited by the following:

- The limit on VPC Network Peering connections from a single VPC network.
- Peering group limits such as the maximum number of forwarding rules for the internal TCP/UDP Load Balancing for each peering group.

If you expect to reach these limits, you can connect the spoke networks through Cloud VPN. Using Cloud VPN adds extra cost and complexity and each Cloud VPN tunnel has a bandwidth limit.

For more information, see the following:

- Hub and spoke transitivity architecture in the enterprise foundations guide
- Terraform deployment stage: Networking with Network Virtual Appliance as part of the Fabric FAST framework
- Terraform hub-and-spoke transitivity module as part of the example foundation

To implement this design option, see Create option 2: Hub-and-spoke topology with centralized appliances.

Option 3: Hub-and-spoke topology without appliances

This network design also uses a hub-and-spoke topology, with a hub VPC network that connects to on-premises networks and spoke VPC networks that contain the workloads. Because VPC Network Peering is non-transitive, spoke networks cannot communicate with each other directly.

Use this design when the following is true:

- You want workloads or environments in Google Cloud to not communicate with each other at all using internal IP addresses, but you do want them to share on-premises connectivity.
- You want to give teams autonomy in managing their own firewall and routing rules.

Avoid this design when the following is true:

- You require Layer 7 inspection between workloads.
- You want to centrally manage routing and firewall rules.
- You require communications from on-premises services to managed services that are connected to the spokes through another VPC Network Peering, because VPC Network Peering is non-transitive.

The following diagram shows an example implementation of this design.

The preceding diagram shows the following:

- A production environment which includes a hub VPC network and multiple spoke VPC networks that contain the workloads.
- The spoke VPC networks are connected with the hub VPC network by using VPC Network Peering.
- Connectivity to on-premises locations passes through Cloud Interconnect connections in the hub VPC network.
- On-premises networks are connected through the Cloud Interconnect instances using separate VLAN attachments.
- The development environment has the same VPC structure as the production environment.

By design, traffic from one spoke network cannot reach another spoke network. However, if specific workloads must communicate with each other, you can set up direct peering between the spoke networks using VPC Network Peering, or you can share data between applications with Google Cloud services like Cloud Storage or Pub/Sub.

This network design is often used in environments where teams act autonomously and there is no centralized control over firewall and routing rules. However, the scale of this design is limited by the following:

- The limit on VPC Network Peering connections from a single VPC network

- Peering group limits such as the maximum number of forwarding rules for the internal passthrough Network Load Balancer for each peering group

Therefore, this design is not typically used in large organizations that have many separate workloads on Google Cloud.

As a variation to the design, you can use Cloud VPN instead of VPC Network Peering. Cloud VPN lets you increase the number of spokes, but adds a bandwidth limit for each tunnel and increases complexity and costs. When you use custom advertised routes, Cloud VPN also allows for transitivity between the spokes without requiring you to directly connect all the spoke networks.

For more information, see the following:

- Hub-and-spoke network architecture
- Hub-and-spoke architecture in the enterprise foundations guide
- Terraform deployment stage: Networking with VPC Network Peering as part of the Fabric FAST framework
- Terraform deployment stage: Networking with Cloud VPN as part of Fabric FAST framework

To implement this design option, see Create option 3: Hub-and-spoke topology without appliances.

Option 4: Expose services in a consumer-producer model with Private Service Connect

In this network design, each team or workload gets their own VPC network, which can also be a Shared VPC network. Each VPC network is independently managed and uses Private Service Connect to expose all the services that need to be accessed from outside the VPC network.

Use this design when the following is true:

- Workloads only communicate with each other and the on-premises environment through defined endpoints.
- You want teams to be independent of each other, and manage their own IP address space, firewalls, and routing rules.

Avoid this design when the following is true:

- Communication between services and applications uses many different ports or channels, or ports and channels change frequently.
- Communication between workloads uses protocols other than TCP or UDP.
- You require Layer 7 inspection between workloads.

The following diagram shows an example implementation of this pattern.

The preceding diagram shows the following:

- Separate workloads are located in separate projects and VPC networks.
- A client VM in one VPC network can connect to a workload in another VPC network through a Private Service Connect endpoint.
- The endpoint is attached to a service attachment in the VPC network where the service is located. The service attachment can be in a different region from the endpoint if the endpoint is configured for global access.
- The service attachment connects to the workload through Cloud Load Balancing.
- Clients in the workload VPC can reach workloads that are located on-premises as follows:
    - The endpoint is connected to a service attachment in a transit VPC network.
    - The service attachment is connected to the on-premises network using Cloud Interconnect.
- An internal Application Load Balancer is attached to the service attachment and uses a hybrid network endpoint group to balance traffic load between the endpoints that are located on-premises.
- On-premises clients can also reach endpoints in the transit VPC network that connect to service attachments in the workload VPC networks.

For more information, see the following:

- Publish managed services using Private Service Connect
- Access published services through endpoints

To implement this design option, see Create option 4: Expose services in a consumer-producer model with Private Service Connect.

Best practices for network deployment

After you choose the best network design for your use case, we recommend implementing the following best practices:

- Use custom mode VPC networks and delete the default network to have better control over your network's IP addresses.

- Limit external access by using Cloud NAT for resources that need internet access and reducing the use of public IP addresses to resources accessible through Cloud Load Balancing. For more information, see Alternatives to using an external IP address.
- If you use Cloud Interconnect, make sure that you follow the recommended topologies for non-critical or production-level applications. Use redundant connections to meet the SLA for the service. Alternatively, you can connect Google Cloud to on-premises networks through Cloud VPN.
- Enforce the policies introduced in limit external access by using an organization policy to restrict direct access to the internet from your VPC.
- Use hierarchical firewall policies to inherit firewall policies consistently across your organization or folders.
- Follow DNS best practices for hybrid DNS between your on-premises network and Google Cloud.

For more information, see Best practices and reference architectures for VPC design.

What's next

- Implement your Google Cloud landing zone network design
- Decide the security for your Google Cloud landing zone (next document in this series).
- Read Best practices for VPC network design.
- Read more about Private Service Connect.

Implement your Google Cloud landing zone network design

Release Notes

Last reviewed 2024-10-31 UTC

This document provides steps and guidance to implement your chosen network design after you review Decide the network design for your Google Cloud landing zone. If you have not already done so, review Landing zone design in Google Cloud before you choose an option.

These instructions are intended for network engineers, architects, and technical practitioners who are involved in creating the network design for your organization's landing zone.

Network design options

Based on your chosen network design, complete one of the following:

- Create option 1: Shared VPC network for each environment
- Create option 2: Hub-and-spoke topology with centralized appliances
- Create option 3: Hub-and-spoke topology without appliances
- Create option 4: Expose services in a consumer-producer model with Private Service Connect

Create option 1: Shared VPC network for each environment

If you have chosen to create the Shared VPC network for each environment in "Decide the network design for your Google Cloud landing zone", follow this procedure.

The following steps create a single instance of a VPC. When you need multiple instances of a VPC, such as for development and production environments, repeat the steps for each VPC.

Limit external access by using an organization policy

We recommend that you limit direct access to the internet to only the resources that need it. Resources without external addresses can still access many Google APIs and services through Private Google Access. Private Google Access is enabled at the subnet level and lets resources interact with key Google services, while isolating them from the public internet.

For usability, the default functionality of Google Cloud lets users create resources in all projects, as long as they have the correct IAM permissions. For improved security, we recommend that you restrict the default permissions for resource types that can cause unintended internet access. You can then authorize specific projects only to allow the creation of these resources. Use the instructions at Creating and managing organization policies to set the following constraints.

Restrict Protocol Forwarding Based on type of IP Address

Protocol forwarding establishes a forwarding rule resource with an external IP address and lets you direct the traffic to a VM.

The **Restrict Protocol Forwarding Based on type of IP Address** constraint prevents the creation of forwarding rules with external IP addresses for the entire organization. For projects authorized to use external forwarding rules, you can modify the constraint at the folder or project level.

Set the following values to configure this constraint:

- *Applies to:* Customize
- *Policy enforcement:* Replace

- *Policy values:* Custom
- *Policy type:* Deny
- *Custom value:* IS:EXTERNAL

Define allowed external IPs for VM instances

By default, individual VM instances can acquire external IP addresses, which allows both outbound and inbound connectivity with the internet.

Enforcing the **Define allowed external IPs for VM instances** constraint prevents the use of external IP addresses with VM instances. For workloads that require external IP addresses on individual VM instances, modify the constraint at a folder or project level to specify the individual VM instances. Or, override the constraint for the relevant projects.

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Deny All

Disable VPC External IPv6 usage

The **Disable VPC External IPv6 usage** constraint, when set to True, prevents the configuration of VPC subnets with external IPv6 addresses for VM instances.

- *Applies to:* Customize
- *Enforcement:* On

Disable default network creation

When a new project is created, a default VPC is automatically created. This is useful for quick experiments that don't require specific network configuration or integration with a larger enterprise networking environment.

Configure the **Skip default network creation** constraint to disable default VPC creation for new projects. You can manually create the default network within a project, if needed.

- *Applies to:* Customize
- *Enforcement:* On

Design firewall rules

Firewall rules let you allow or deny traffic to or from your VMs based on a configuration you define. Hierarchical firewall policies are implemented at the organization and folder levels, and network firewall policies are implemented at the VPC network level in the resource hierarchy. Together, these provide an important capability to help secure your workloads.

Regardless of where the firewall policies are applied, use the following guidelines when designing and evaluating your firewall rules:

- Implement least-privilege (also referred to as microsegmentation) principles. Block all traffic by default and only allow the specific traffic you need. This includes limiting the rules to only the protocols and ports you need for each workload.
- Enable firewall rules logging for visibility into firewall behavior and to use Firewall Insights.
- Define a numbering methodology for allocating firewall rule priorities. For example, it's best practice to reserve a range of low numbers in each policy for rules needed during incident response. We also recommend that you prioritize more specific rules higher than more general rules, to ensure that the specific rules aren't shadowed by the general rules. The following example shows a possible approach for firewall rule priorities:

Firewall rule priority range	Purpose
0-999	Reserved for incident response
1000-1999	Always blocked traffic
2000-1999999999	Workload-specific rules
2000000000-2100000000	Catch-all rules
2100000001-2147483643	Reserved

Configure hierarchical firewall policies

Hierarchical firewall policies let you create and enforce a consistent firewall policy across your organization. For examples of using hierarchical firewall policies, see Hierarchical firewall policy examples.

Define hierarchical firewall policies to implement the following network access controls:

- **Identity-Aware Proxy (IAP) for TCP forwarding.** IAP for TCP forwarding is allowed through a security policy that permits ingress traffic from IP range 35.235.240.0/20 for TCP ports 22 and 3389.
- **Health checks for Cloud Load Balancing.** The well-known ranges that are used for health checks are allowed.
    - For most Cloud Load Balancing instances (including Internal TCP/UDP Load Balancing, Internal HTTP(S) Load Balancing, External TCP Proxy Load Balancing, External SSL Proxy Load Balancing, and HTTP(S) Load Balancing), a security policy is defined that allows ingress traffic from the IP ranges 35.191.0.0/16 and 130.211.0.0/22 for ports 80 and 443.
    - For Network Load Balancing, a security policy is defined that enables legacy health checks by allowing ingress traffic from IP ranges 35.191.0.0/16, 209.85.152.0/22, and 209.85.204.0/22 for ports 80 and 443.

Configure your Shared VPC environment

Before implementing a Shared VPC design, decide how to share subnets with service projects. You attach a service project to a host project. To determine which subnets are available for the service project, you assign IAM permissions to the host project or individual subnets. For example, you can choose to dedicate a different subnet to each service project, or share the same subnets between service projects.

1. Create a new project for the Shared VPC. Later in this process, this project becomes the host project and contains the networks and networking resources to be shared with the service projects.
2. Enable the Compute Engine API for the host project.
3. Configure Shared VPC for the project.
4. Create the custom-mode VPC network in the host project.
5. Create subnets in the region where you plan to deploy workloads. For each subnet, enable Private Google Access to allow VM instances without external IP addresses to reach Google services.

Configure Cloud NAT

Follow these steps if the workloads in specific regions require outbound internet access—for example, to download software packages or updates.

1. Create a Cloud NAT gateway in the regions where workloads require outbound internet access. You can customize the Cloud NAT configuration to only allow outbound connectivity from specific subnets, if needed.
2. At a minimum, enable Cloud NAT logging for the gateway to log ERRORS_ONLY. To include logs for translations performed by Cloud NAT, configure each gateway to log ALL.

Configure hybrid connectivity

You can use Dedicated Interconnect, Partner Interconnect, or Cloud VPN to provide hybrid connectivity to your landing zone. The following steps create the initial hybrid connectivity resources required for this design option:

1. If you're using Dedicated Interconnect, do the following. If you're using Partner Interconnect or Cloud VPN, you can skip these steps.
    a. Create a separate project for the physical interconnect ports.
    b. Enable the Compute Engine API for the project.
    c. Create Dedicated Interconnect connections.
2. For each region where you're terminating hybrid connectivity in the VPC network, do the following:
    a. Create two Dedicated or Partner VLAN attachments, one for each edge availability zone. As part of this process, you select Cloud Routers and create BGP sessions.
    b. Configure the peer network (on-premises or other cloud) routers.

Configure workload projects

Create a separate service project for each workload:

1. Create a new project to function as one of the service projects for the Shared VPC.
2. Enable the Compute Engine API for the service project.
3. Attach the project to the host project.
4. Configure access to all subnets in the host project or some subnets in the host project.

Configure observability

Network Intelligence Center provides a cohesive way to monitor, troubleshoot, and visualize your cloud networking environment. Use it to ensure that your design functions with the desired intent.

The following configurations support the analysis of logging and metrics enabled.

- You must enable the Network Management API before you can run Connectivity Tests. Enabling the API is required to use the API directly, the Google Cloud CLI, or the Google Cloud console.
- You must enable the Firewall Insights API before you can perform any tasks using Firewall Insights.

Next steps

The initial configuration for this network design option is now complete. You can now either repeat these steps to configure an additional instance of the landing zone environment, such as a staging or production environment, or continue to Decide the security for your Google Cloud landing zone.

Create option 2: Hub-and-spoke topology with centralized appliances

If you have chosen to create the hub-and-spoke topology with centralized appliances in "Decide the network design for your Google Cloud landing zone", follow this procedure.

The following steps create a single instance of a VPC. When you need multiple instances of a VPC, such as for development and production environments, repeat the steps for each VPC.

Limit external access by using an organization policy

We recommend that you limit direct access to the internet to only the resources that need it. Resources without external addresses can still access many Google APIs and services through Private Google Access. Private Google Access is enabled at the subnet level and lets resources interact with key Google services, while isolating them from the public internet.

For usability, the default functionality of Google Cloud lets users create resources in all projects, as long as they have the correct IAM permissions. For improved security, we recommend that you restrict the default permissions for resource types that can cause unintended internet access. You can then authorize specific projects only to allow the creation of these resources. Use the instructions at Creating and managing organization policies to set the following constraints.

Restrict Protocol Forwarding Based on type of IP Address

Protocol forwarding establishes a forwarding rule resource with an external IP address and lets you direct the traffic to a VM.

The **Restrict Protocol Forwarding Based on type of IP Address** constraint prevents the creation of forwarding rules with external IP addresses for the entire organization. For projects authorized to use external forwarding rules, you can modify the constraint at the folder or project level.

Set the following values to configure this constraint:

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Custom
- *Policy type:* Deny
- *Custom value:* IS:EXTERNAL

Define allowed external IPs for VM instances

By default, individual VM instances can acquire external IP addresses, which allows both outbound and inbound connectivity with the internet.

Enforcing the **Define allowed external IPs for VM instances** constraint prevents the use of external IP addresses with VM instances. For workloads that require external IP addresses on individual VM instances, modify the constraint at a folder or project level to specify the individual VM instances. Or, override the constraint for the relevant projects.

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Deny All

Disable VPC External IPv6 usage

The **Disable VPC External IPv6 usage** constraint, when set to True, prevents the configuration of VPC subnets with external IPv6 addresses for VM instances.

- *Applies to:* Customize
- *Enforcement:* On

Disable default network creation

When a new project is created, a default VPC is automatically created. This is useful for quick experiments that don't require specific network configuration or integration with a larger enterprise networking environment.

Configure the **Skip default network creation** constraint to disable default VPC creation for new projects. You can manually create the default network within a project, if needed.

- *Applies to:* Customize
- *Enforcement:* On

Design firewall rules

Firewall rules let you allow or deny traffic to or from your VMs based on a configuration you define. Hierarchical firewall policies are implemented at the organization and folder levels, and network firewall policies are implemented at the VPC network level in the resource hierarchy. Together, these provide an important capability to help secure your workloads.

Regardless of where the firewall policies are applied, use the following guidelines when designing and evaluating your firewall rules:

- Implement least-privilege (also referred to as microsegmentation) principles. Block all traffic by default and only allow the specific traffic you need. This includes limiting the rules to only the protocols and ports you need for each workload.
- Enable firewall rules logging for visibility into firewall behavior and to use Firewall Insights.
- Define a numbering methodology for allocating firewall rule priorities. For example, it's best practice to reserve a range of low numbers in each policy for rules needed during incident response. We also recommend that you prioritize more specific rules higher than more general rules, to ensure that the specific rules aren't shadowed by the general rules. The following example shows a possible approach for firewall rule priorities:

Firewall rule priority range	Purpose
0-999	Reserved for incident response
1000-1999	Always blocked traffic

Firewall rule priority range	Purpose
2000-1999999999	Workload-specific rules
2000000000-2100000000	Catch-all rules
2100000001-2147483643	Reserved

Configure hierarchical firewall policies

Hierarchical firewall policies let you create and enforce a consistent firewall policy across your organization. For examples of using hierarchical firewall policies, see Hierarchical firewall policy examples.

Define hierarchical firewall policies to implement the following network access controls:

- **Identity-Aware Proxy (IAP) for TCP forwarding.** IAP for TCP forwarding is allowed through a security policy that permits ingress traffic from IP range 35.235.240.0/20 for TCP ports 22 and 3389.
- **Health checks for Cloud Load Balancing.** The well-known ranges that are used for health checks are allowed.
    - For most Cloud Load Balancing instances (including Internal TCP/UDP Load Balancing, Internal HTTP(S) Load Balancing, External TCP Proxy Load Balancing, External SSL Proxy Load Balancing, and HTTP(S) Load Balancing), a security policy is defined that allows ingress traffic from the IP ranges 35.191.0.0/16 and 130.211.0.0/22 for ports 80 and 443.
    - For Network Load Balancing, a security policy is defined that enables legacy health checks by allowing ingress traffic from IP ranges 35.191.0.0/16, 209.85.152.0/22, and 209.85.204.0/22 for ports 80 and 443.

Configure your VPC environment

The transit and hub VPC networks provide the networking resources to enable connectivity between workload spoke VPC networks and on-premises or multi-cloud networks.

1. Create a new project for transit and hub VPC networks. Both VPC networks are part of the same project to support connectivity through the virtual network appliances.
2. Enable the Compute Engine API for the project.
3. Create the transit custom mode VPC network.
4. In the transit VPC network, create a subnet in the regions where you plan to deploy the virtual network appliances.
5. Create the hub custom mode VPC network.
6. In the hub VPC network, create a subnet in the regions where you plan to deploy the virtual network appliances.
7. Configure global or regional network firewall policies to allow ingress and egress traffic for the network virtual appliances.
8. Create a managed instance group for the virtual network appliances.
9. Configure the internal TCP/UDP load balancing resources for the transit VPC. This load balancer is used for routing traffic from the transit VPC to the hub VPC through the virtual network appliances.
10. Configure the internal TCP/UDP load balancing resources for the hub VPC. This load balancer is used for routing traffic from the hub VPC to the transit VPC through the virtual network appliances.
11. Configure Private Service Connect for Google APIs for the hub VPC.
12. Modify VPC routes to send all traffic through the network virtual appliances:
    a. Delete the 0.0.0.0/0 route with next-hop default-internet-gateway from the hub VPC.
    b. Configure a new route with destination 0.0.0.0/0 and a next-hop of the forwarding rule for the load balancer in the hub VPC.

Configure Cloud NAT

Follow these steps if the workloads in specific regions require outbound internet access—for example, to download software packages or updates.

1. Create a Cloud NAT gateway in the regions where workloads require outbound internet access. You can customize the Cloud NAT configuration to only allow outbound connectivity from specific subnets, if needed.

2. At a minimum, enable Cloud NAT logging for the gateway to log ERRORS_ONLY. To include logs for translations performed by Cloud NAT, configure each gateway to log ALL.

Configure hybrid connectivity

You can use Dedicated Interconnect, Partner Interconnect, or Cloud VPN to provide hybrid connectivity to your landing zone. The following steps create the initial hybrid connectivity resources required for this design option:

1. If you're using Dedicated Interconnect, do the following. If you're using Partner Interconnect or Cloud VPN, you can skip these steps.
    a. Create a separate project for the physical interconnect ports.
    b. Enable the Compute Engine API for the project.
    c. Create Dedicated Interconnect connections.
2. For each region where you're terminating hybrid connectivity in the VPC network, do the following:
    a. Create two Dedicated or Partner VLAN attachments, one for each edge availability zone. As part of this process, you select Cloud Routers and create BGP sessions.
    b. Configure the peer network (on-premises or other cloud) routers.
    c. Configure custom advertised routes in the Cloud Routers for the subnet ranges in the hub and workload VPCs.

Configure workload projects

Create a separate spoke VPC for each workload:

1. Create a new project to host your workload.
2. Enable the Compute Engine API for the project.
3. Configure VPC Network Peering between the workload spoke VPC and hub VPC with the following settings:
    - Enable custom route export on the hub VPC.
    - Enable custom route import on the workload spoke VPC.
4. Create subnets in the regions where you plan to deploy workloads. For each subnet, enable Private Google Access to allow VM instances with *only* internal IP addresses to reach Google services.
5. Configure Private Service Connect for Google APIs.

6. To route all traffic through the virtual network appliances in the hub VPC, delete the 0.0.0.0/0 route with next-hop default-internet-gateway from the workload spoke VPC.
7. Configure global or regional network firewall policies to allow ingress and egress traffic for your workload.

Configure observability

Network Intelligence Center provides a cohesive way to monitor, troubleshoot, and visualize your cloud networking environment. Use it to ensure that your design functions with the desired intent.

The following configurations support the analysis of logging and metrics enabled.

- You must enable the Network Management API before you can run Connectivity Tests. Enabling the API is required to use the API directly, the Google Cloud CLI, or the Google Cloud console.
- You must enable the Firewall Insights API before you can perform any tasks using Firewall Insights.

Next steps

The initial configuration for this network design option is now complete. You can now either repeat these steps to configure an additional instance of the landing zone environment, such as a staging or production environment, or continue to Decide the security for your Google Cloud landing zone.

Create option 3: Hub-and-spoke topology without appliances

If you have chosen to create the hub-and-spoke topology without appliances in "Decide the network design for your Google Cloud landing zone", follow this procedure.

The following steps create a single instance of a VPC. When you need multiple instances of a VPC, such as for development and production environments, repeat the steps for each VPC.

Limit external access by using an organization policy

We recommend that you limit direct access to the internet to only the resources that need it. Resources without external addresses can still access many Google APIs and services through Private Google Access. Private Google Access is enabled at the subnet level and lets resources interact with key Google services, while isolating them from the public internet.

For usability, the default functionality of Google Cloud lets users create resources in all projects, as long as they have the correct IAM permissions. For improved security, we recommend that you restrict the default permissions for resource types that can cause unintended internet access. You can then authorize specific projects only to allow the creation of these resources. Use the instructions at Creating and managing organization policies to set the following constraints.

Restrict Protocol Forwarding Based on type of IP Address

Protocol forwarding establishes a forwarding rule resource with an external IP address and lets you direct the traffic to a VM.

The **Restrict Protocol Forwarding Based on type of IP Address** constraint prevents the creation of forwarding rules with external IP addresses for the entire organization. For projects authorized to use external forwarding rules, you can modify the constraint at the folder or project level.

Set the following values to configure this constraint:

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Custom
- *Policy type:* Deny
- *Custom value:* IS:EXTERNAL

Define allowed external IPs for VM instances

By default, individual VM instances can acquire external IP addresses, which allows both outbound and inbound connectivity with the internet.

Enforcing the **Define allowed external IPs for VM instances** constraint prevents the use of external IP addresses with VM instances. For workloads that require external IP addresses on individual VM instances, modify the constraint at a folder or project level to specify the individual VM instances. Or, override the constraint for the relevant projects.

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Deny All

Disable VPC External IPv6 usage

The **Disable VPC External IPv6 usage** constraint, when set to True, prevents the configuration of VPC subnets with external IPv6 addresses for VM instances.

- *Applies to:* Customize

- *Enforcement:* On

Disable default network creation

When a new project is created, a default VPC is automatically created. This is useful for quick experiments that don't require specific network configuration or integration with a larger enterprise networking environment.

Configure the **Skip default network creation** constraint to disable default VPC creation for new projects. You can manually create the default network within a project, if needed.

- *Applies to:* Customize
- *Enforcement:* On

Design firewall rules

Firewall rules let you allow or deny traffic to or from your VMs based on a configuration you define. Hierarchical firewall policies are implemented at the organization and folder levels, and network firewall policies are implemented at the VPC network level in the resource hierarchy. Together, these provide an important capability to help secure your workloads.

Regardless of where the firewall policies are applied, use the following guidelines when designing and evaluating your firewall rules:

- Implement least-privilege (also referred to as microsegmentation) principles. Block all traffic by default and only allow the specific traffic you need. This includes limiting the rules to only the protocols and ports you need for each workload.
- Enable firewall rules logging for visibility into firewall behavior and to use Firewall Insights.
- Define a numbering methodology for allocating firewall rule priorities. For example, it's best practice to reserve a range of low numbers in each policy for rules needed during incident response. We also recommend that you prioritize more specific rules higher than more general rules, to ensure that the specific rules aren't shadowed by the general rules. The following example shows a possible approach for firewall rule priorities:

Firewall rule priority range	Purpose
0-999	Reserved for incident response
1000-1999	Always blocked traffic
2000-1999999999	Workload-specific rules
2000000000-2100000000	Catch-all rules
2100000001-2147483643	Reserved

Configure hierarchical firewall policies

Hierarchical firewall policies let you create and enforce a consistent firewall policy across your organization. For examples of using hierarchical firewall policies, see Hierarchical firewall policy examples.

Define hierarchical firewall policies to implement the following network access controls:

- **Identity-Aware Proxy (IAP) for TCP forwarding.** IAP for TCP forwarding is allowed through a security policy that permits ingress traffic from IP range 35.235.240.0/20 for TCP ports 22 and 3389.
- **Health checks for Cloud Load Balancing.** The well-known ranges that are used for health checks are allowed.
    - For most Cloud Load Balancing instances (including Internal TCP/UDP Load Balancing, Internal HTTP(S) Load Balancing, External TCP Proxy Load Balancing, External SSL Proxy Load Balancing, and HTTP(S) Load Balancing), a security policy is defined that allows ingress traffic from the IP ranges 35.191.0.0/16 and 130.211.0.0/22 for ports 80 and 443.
    - For Network Load Balancing, a security policy is defined that enables legacy health checks by allowing ingress traffic from IP ranges

35.191.0.0/16, 209.85.152.0/22, and 209.85.204.0/22 for ports 80 and 443.

Configure the hub VPC environment

The hub VPC provides the networking resources to enable connectivity between workload spoke VPC networks and on-premises or multi-cloud networks.

1. Create a new project for the hub VPC network.
2. Enable the Compute Engine API for the project.
3. Create the hub custom mode VPC network.
4. Configure Private Service Connect for Google APIs for the hub VPC.

Configure hybrid connectivity

You can use Dedicated Interconnect, Partner Interconnect, or Cloud VPN to provide hybrid connectivity to your landing zone. The following steps create the initial hybrid connectivity resources required for this design option:

1. If you're using Dedicated Interconnect, do the following. If you're using Partner Interconnect or Cloud VPN, you can skip these steps.
    a. Create a separate project for the physical interconnect ports.
    b. Enable the Compute Engine API for the project.
    c. Create Dedicated Interconnect connections.
2. For each region where you're terminating hybrid connectivity in the VPC network, do the following:
    a. Create two Dedicated or Partner VLAN attachments, one for each edge availability zone. As part of this process, you select Cloud Routers and create BGP sessions.
    b. Configure the peer network (on-premises or other cloud) routers.
    c. Configure custom advertised routes in the Cloud Routers for the subnet ranges in the hub and workload VPCs.

Configure workload projects

Create a separate spoke VPC for each workload:

1. Create a new project to host your workload.
2. Enable the Compute Engine API for the project.

3. Configure VPC Network Peering between the workload spoke VPC and hub VPC, with the following settings:
    - Enable custom route export on the hub VPC.
    - Enable custom route import on the workload spoke VPC.
4. Create subnets in the regions where you plan to deploy workloads. For each subnet, enable Private Google Access to allow VM instances with *only* internal IP addresses to reach Google services.
5. Configure Private Service Connect for Google APIs.

Configure Cloud NAT

Follow these steps if the workloads in specific regions require outbound internet access—for example, to download software packages or updates.

1. Create a Cloud NAT gateway in the regions where workloads require outbound internet access. You can customize the Cloud NAT configuration to only allow outbound connectivity from specific subnets, if needed.
2. At a minimum, enable Cloud NAT logging for the gateway to log ERRORS_ONLY. To include logs for translations performed by Cloud NAT, configure each gateway to log ALL.

Configure observability

Network Intelligence Center provides a cohesive way to monitor, troubleshoot, and visualize your cloud networking environment. Use it to ensure that your design functions with the desired intent.

The following configurations support the analysis of logging and metrics enabled.

- You must enable the Network Management API before you can run Connectivity Tests. Enabling the API is required to use the API directly, the Google Cloud CLI, or the Google Cloud console.
- You must enable the Firewall Insights API before you can perform any tasks using Firewall Insights.

Next steps

The initial configuration for this network design option is now complete. You can now either repeat these steps to configure an additional instance of the landing zone environment, such as a staging or production environment, or continue to Decide the security for your Google Cloud landing zone.

Create option 4: Expose services in a consumer-producer model with Private Service Connect

If you have chosen to expose services in a consumer-producer model with Private Service Connect for your landing zone, as described in "Decide the network design for your Google Cloud landing zone", follow this procedure.

The following steps create a single instance of a VPC. When you need multiple instances of a VPC, such as for development and production environments, repeat the steps for each VPC.

Limit external access by using an organization policy

We recommend that you limit direct access to the internet to only the resources that need it. Resources without external addresses can still access many Google APIs and services through Private Google Access. Private Google Access is enabled at the subnet level and lets resources interact with key Google services, while isolating them from the public internet.

For usability, the default functionality of Google Cloud lets users create resources in all projects, as long as they have the correct IAM permissions. For improved security, we recommend that you restrict the default permissions for resource types that can cause unintended internet access. You can then authorize specific projects only to allow the creation of these resources. Use the instructions at Creating and managing organization policies to set the following constraints.

Restrict Protocol Forwarding Based on type of IP Address

Protocol forwarding establishes a forwarding rule resource with an external IP address and lets you direct the traffic to a VM.

The **Restrict Protocol Forwarding Based on type of IP Address** constraint prevents the creation of forwarding rules with external IP addresses for the entire organization. For projects authorized to use external forwarding rules, you can modify the constraint at the folder or project level.

Set the following values to configure this constraint:

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Custom
- *Policy type:* Deny
- *Custom value:* IS:EXTERNAL

Define allowed external IPs for VM instances

By default, individual VM instances can acquire external IP addresses, which allows both outbound and inbound connectivity with the internet.

Enforcing the **Define allowed external IPs for VM instances** constraint prevents the use of external IP addresses with VM instances. For workloads that require external IP addresses on individual VM instances, modify the constraint at a folder or project level to specify the individual VM instances. Or, override the constraint for the relevant projects.

- *Applies to:* Customize
- *Policy enforcement:* Replace
- *Policy values:* Deny All

Disable VPC External IPv6 usage

The **Disable VPC External IPv6 usage** constraint, when set to True, prevents the configuration of VPC subnets with external IPv6 addresses for VM instances.

- *Applies to:* Customize
- *Enforcement:* On

Disable default network creation

When a new project is created, a default VPC is automatically created. This is useful for quick experiments that don't require specific network configuration or integration with a larger enterprise networking environment.

Configure the **Skip default network creation** constraint to disable default VPC creation for new projects. You can manually create the default network within a project, if needed.

- *Applies to:* Customize
- *Enforcement:* On

Design firewall rules

Firewall rules let you allow or deny traffic to or from your VMs based on a configuration you define. Hierarchical firewall policies are implemented at the organization and folder levels, and network firewall policies are implemented at the VPC network level in the resource hierarchy. Together, these provide an important capability to help secure your workloads.

Regardless of where the firewall policies are applied, use the following guidelines when designing and evaluating your firewall rules:

- Implement least-privilege (also referred to as microsegmentation) principles. Block all traffic by default and only allow the specific traffic you need. This includes limiting the rules to only the protocols and ports you need for each workload.
- Enable firewall rules logging for visibility into firewall behavior and to use Firewall Insights.
- Define a numbering methodology for allocating firewall rule priorities. For example, it's best practice to reserve a range of low numbers in each policy for rules needed during incident response. We also recommend that you prioritize more specific rules higher than more general rules, to ensure that the specific rules aren't shadowed by the general rules. The following example shows a possible approach for firewall rule priorities:

Firewall rule priority range	Purpose
0-999	Reserved for incident response
1000-1999	Always blocked traffic
2000-1999999999	Workload-specific rules
2000000000-2100000000	Catch-all rules
2100000001-2147483643	Reserved

Configure hierarchical firewall policies

Hierarchical firewall policies let you create and enforce a consistent firewall policy across your organization. For examples of using hierarchical firewall policies, see Hierarchical firewall policy examples.

Define hierarchical firewall policies to implement the following network access controls:

- **Identity-Aware Proxy (IAP) for TCP forwarding.** IAP for TCP forwarding is allowed through a security policy that permits ingress traffic from IP range 35.235.240.0/20 for TCP ports 22 and 3389.
- **Health checks for Cloud Load Balancing.** The well-known ranges that are used for health checks are allowed.
    - For most Cloud Load Balancing instances (including Internal TCP/UDP Load Balancing, Internal HTTP(S) Load Balancing, External TCP Proxy Load Balancing, External SSL Proxy Load Balancing, and HTTP(S) Load Balancing), a security policy is defined that allows ingress traffic from the IP ranges 35.191.0.0/16 and 130.211.0.0/22 for ports 80 and 443.
    - For Network Load Balancing, a security policy is defined that enables legacy health checks by allowing ingress traffic from IP ranges 35.191.0.0/16, 209.85.152.0/22, and 209.85.204.0/22 for ports 80 and 443.

Configure the VPC environment

The transit VPC provides the networking resources to enable connectivity between workload spoke VPC networks and on-premises or multi-cloud networks.

1. Create a new project for the transit VPC network.
2. Enable the Compute Engine API for the project.
3. Create the transit custom mode VPC network.
4. Create a Private Service Connect subnet in each region where you plan to publish services running in your hub VPC or on-premises environment. Consider Private Service Connect subnet sizing when deciding your IP addressing plan.
5. For each on-premises service you want to expose to workloads running in Google Cloud, create an internal HTTP(S) or TCP proxy load balancer and expose the services using Private Service Connect.
6. Configure Private Service Connect for Google APIs for the transit VPC.

Configure hybrid connectivity

You can use Dedicated Interconnect, Partner Interconnect, or Cloud VPN to provide hybrid connectivity to your landing zone. The following steps create the initial hybrid connectivity resources required for this design option:

1. If you're using Dedicated Interconnect, do the following. If you're using Partner Interconnect or Cloud VPN, you can skip these steps.

    a. Create a separate project for the physical interconnect ports.

    b. Enable the Compute Engine API for the project.

    c. Create Dedicated Interconnect connections.

2. For each region where you're terminating hybrid connectivity in the VPC network, do the following:

    a. Create two Dedicated or Partner VLAN attachments, one for each edge availability zone. As part of this process, you select Cloud Routers and create BGP sessions.

    b. Configure the peer network (on-premises or other cloud) routers.

Configure workload projects

Create a separate VPC for each workload:

1. Create a new project to host your workload.
2. Enable the Compute Engine API for the project.
3. Create a custom-mode VPC network.
4. Create subnets in the regions where you plan to deploy workloads. For each subnet, enable Private Google Access to allow VM instances with *only* internal IP addresses to reach Google services.
5. Configure Private Service Connect for Google APIs.
6. For each workload you're consuming from a different VPC or your on-premises environment, create a Private Service Connect consumer endpoint.
7. For each workload you're producing for a different VPC or your on-premises environment, create an internal load balancer and service attachment for the service. Consider Private Service Connect subnet sizing when deciding your IP addressing plan.
8. If the service should be reachable from your on-premises environment, create a Private Service Connect consumer endpoint in the transit VPC.

Configure Cloud NAT

Follow these steps if the workloads in specific regions require outbound internet access—for example, to download software packages or updates.

1. Create a Cloud NAT gateway in the regions where workloads require outbound internet access. You can customize the Cloud NAT configuration to only allow outbound connectivity from specific subnets, if needed.

2. At a minimum, enable Cloud NAT logging for the gateway to log ERRORS_ONLY. To include logs for translations performed by Cloud NAT, configure each gateway to log ALL.

Configure observability

Network Intelligence Center provides a cohesive way to monitor, troubleshoot, and visualize your cloud networking environment. Use it to ensure that your design functions with the desired intent.

The following configurations support the analysis of logging and metrics enabled.

- You must enable the Network Management API before you can run Connectivity Tests. Enabling the API is required to use the API directly, the Google Cloud CLI, or the Google Cloud console.
- You must enable the Firewall Insights API before you can perform any tasks using Firewall Insights.

Next steps

The initial configuration for this network design option is now complete. You can now either repeat these steps to configure an additional instance of the landing zone environment, such as a staging or production environment, or continue to Decide the security for your Google Cloud landing zone.

What's next

- Decide the security for your Google Cloud landing zone (next document in this series).
- Read Best practices for VPC network design.
- Read more about Private Service Connect.

Decide the security for your Google Cloud landing zone

Release Notes

Last reviewed 2025-06-07 UTC

This document introduces important security decisions and recommended options to consider when designing a Google Cloud landing zone. It's part of a series about landing zones, and is intended for security specialists, CISOs, and architects who want to understand the decisions that they need to make when designing a landing zone in Google Cloud.

In this document, it's assumed that a central team, such as the security team or the platform team, enforces these landing zone security controls. Because the focus of this document is the design of enterprise-scale environments, some strategies that it describes might be less relevant for small teams.

Decision points for securing your Google Cloud landing zone

To choose the best security design for your organization, you must make the following decisions:

- How to limit persistent credentials for service accounts
- How to mitigate data exfiltration through Google APIs
- How to continuously monitor for insecure configurations and threats
- How to centrally aggregate necessary logs
- How to meet compliance requirements for encryption at rest
- How to meet compliance requirements for encryption in transit
- Which additional controls are necessary for managing cloud service provider access

Architecture diagram

The example architecture described in this document uses common security design patterns. Your specific controls might vary based on factors such as your organization's industry, target workloads, or additional compliance requirements. The following diagram shows the security controls architecture that you apply in your landing zone when you follow the recommendations in this document.

The preceding diagram shows the following:

- Service account key management helps mitigate risk from long-lived service account credentials.
- VPC Service Controls defines a perimeter around sensitive resources that helps to restrict access from outside the perimeter.
- Security Command Center monitors the environment for insecure configurations and threats.
- A centralized log sink collects audit logs from all projects.
- Google default encryption at rest encrypts all data that persists to disk.
- Google default encryption in transit applies to layer 3 and layer 4 network paths.

- Access Transparency gives you visibility and control over how Google can access your environment.

Decide how to limit persistent credentials for service accounts

Service accounts are machine identities that you use to grant IAM roles to workloads and allow the workload to access Google Cloud APIs. A service account key is a persistent credential, and any persistent credentials are potentially high risk. We don't recommend that you let developers freely create service account keys.

For example, if a developer accidentally commits the service account key to a public Git repository, an external attacker can authenticate using those credentials. As another example, if the service account key is stored in an internal repository, a malicious insider who can read the key could use the credentials to escalate their own Google Cloud privileges.

To define a strategy to manage these persistent credentials, you must provide viable alternatives, limit the proliferation of persistent credentials, and manage how they are used. For information about alternatives to service account keys, see Choose the right authentication method for your use case.

The following sections describe the options to limit persistent credentials. We recommend option 1 for most use cases. The other options discussed in the following sections are alternatives that you can consider if option 1 doesn't apply to your specific organization.

All organizations created after May 23, 2024 have Google Cloud security baseline constraints enforced when the organization resource is first created. This change makes option 1 the default option.

Option 1: Restrict use of persistent service account keys

We recommend that you don't permit any users to download service account keys because exposed keys are a common attack vector. Restricting the use of persistent service account keys is an option that can help reduce the risk and overhead of manually managing service account keys.

To implement this option, consider the following:

- To prevent developers from creating and downloading persistent credentials, configure the organization policy constraint constraints/iam.disableServiceAccountKeyCreation.
- Educate your teams on more secure alternatives to service account keys. For example, when users and applications that are outside of your Google Cloud environment need to use a service account, they can authenticate with service

account impersonation or workload identity federation instead of a service account key.

- Design a process for teams to request an exception to this policy when downloading a service account key is the only viable option. For example, a third-party SaaS product might require a service account key to read logs from your Google Cloud environment.

Avoid this option when you already have tooling in place to generate short-lived API credentials for service accounts.

For more information, see the following:

- Organization policy constraints in the Google Cloud enterprise foundations blueprint
- Best practices for working with service accounts

Option 2: Use additional access management tools to generate short-lived credentials

As an alternative to Restrict use of persistent service account keys, you can generate short-lived credentials for service accounts. Short-lived credentials create less risk than persistent credentials such as service account keys. You can develop your own tooling or use third-party solutions such as Hashicorp Vault to generate short-lived access credentials.

Use this option when you already have invested in a third-party tool for generating short-lived credentials for access control, or have sufficient budget and capacity to develop your own solution.

Avoid using this option when you don't have existing tooling to grant short-lived credentials, or don't have the capacity to build your own solution.

For more information, see Create short-lived service account credentials.

Decide how to mitigate data exfiltration through Google APIs

Google APIs have public endpoints that are available to all customers. While every API resource in your Google Cloud environment is subject to IAM access controls, there is a risk that data could be accessed using stolen credentials, exfiltrated by malicious insiders or compromised code, or exposed through a misconfigured IAM policy.

VPC Service Controls is a solution that addresses these risks. However, VPC Service Controls also introduces complexity to your access model, so you must design VPC Service Controls to meet your unique environment and use case.

The following sections describe the options to mitigate data exfiltration through Google APIs. We recommend option 1 for most use cases. The other options discussed in the following sections are alternatives that you can consider if option 1 doesn't apply to your specific use case.

Option 1: Configure VPC Service Controls broadly across your environment

We recommend that you design your environment within one or more VPC Service Controls perimeters that restrict all supported APIs. Configure exceptions to the perimeter with access levels or ingress policies so that developers can access the services that they require, including console access where needed.

Use this option when the following is true:

- The services that you intend to use support VPC Service Controls, and your workloads don't require unrestricted internet access.
- You store sensitive data on Google Cloud that could be a significant loss if exfiltrated.
- You have consistent attributes for developer access that can be configured as exceptions to the perimeter, allowing users to access the data that they need.

Avoid this option when your workloads require unrestricted internet access or services that are not supported by VPC Service Controls.

For more information, see the following:

- Best practices for enabling VPC Service Controls
- Supported products and limitations for VPC Service Controls
- Design and architect service perimeters

Option 2: Configure VPC Service Controls for a subset of your environment

Instead of configuring VPC Service Controls broadly across your environment, you can configure VPC Service Controls only on the subset of projects that contain sensitive data and internal-only workloads. This option lets you use a simpler design and operation for most projects, while still prioritizing data protection for projects with sensitive data.

For example, you might consider this alternative when a limited number of projects contain BigQuery datasets with sensitive data. You can define a service perimeter around just these projects, and define ingress and egress rules to allow narrow exceptions for the analysts who need to use these datasets.

For another example, in an application with three-tier architecture, some components might be outside of the perimeter. The presentation tier that allows ingress from user traffic might be a project outside of the perimeter, and the application tier and data tier that contain sensitive data might be separate projects inside the service perimeter. You define ingress and egress rules to the perimeter so that the tiers can communicate across the perimeter with granular access.

Use this option when the following is true:

- Only limited and well-defined projects contain sensitive data. Other projects contain data of lower risk.
- Some workloads are internal only, but some workloads require public internet access or have dependencies on services that are not supported by VPC Service Controls.
- Configuring VPC Service Controls across all projects creates too much overhead or requires too many workarounds

Avoid this option when many projects could potentially contain sensitive data.

For more information, see Best practices for enabling VPC Service Controls.

Option 3: Don't configure VPC Service Controls

As another alternative to configuring VPC Service Controls broadly across your environment, you can choose not to use VPC Service Controls, particularly if the operational overhead outweighs the value of VPC Service Controls.

For example, your organization might not have a consistent pattern of developer access that could form the basis of an ingress policy. Perhaps your IT operations are outsourced to multiple third parties, so developers don't have managed devices or access from consistent IP addresses. In this scenario, you might not be able to define ingress rules to allow exceptions to the perimeter that developers need to complete their daily operations.

Use this option when:

- You use services that don't support VPC Service Controls.
- Workloads are internet facing and don't contain sensitive data.
- You don't have consistent attributes of developer access like managed devices or known IP ranges.

Avoid this option when you have sensitive data in your Google Cloud environment.

Decide how to continuously monitor for insecure configurations and threats

Adopting cloud services introduces new challenges and threats when compared to using services located on-premises. Your existing tools that monitor long-lived servers may not be appropriate for autoscaling or ephemeral services, and might not monitor serverless resources at all. Therefore, you should evaluate security tools that work with the full range of cloud services that you might adopt. You should also continuously monitor for cloud standards, like the CIS Benchmark for Google Cloud.

The following sections describe the options for continuous monitoring. We recommend option 1 for most use cases. The other options discussed in the following sections are alternatives that you can consider if option 1 doesn't apply to your specific use case.

Option 1: Use Security Command Center

We recommend that you activate Security Command Center at the organization level, which helps you strengthen your security posture by doing the following:

- Evaluating your security and data attack surface
- Providing asset inventory and discovery
- Identifying misconfigurations, vulnerabilities, and threats
- Helping you mitigate and remediate risks

When you enable Security Command Center at the beginning of your landing zone build, your organization's security team has near real-time visibility on insecure configurations, threats, and remediation options. This visibility helps your security team assess whether the landing zone meets their requirements and is ready for developers to start deploying applications.

Use this option when the following is true:

- You want a security posture management and threat detection tool that is integrated with all Google Cloud services without additional integration effort.
- You want to use the same threat intelligence, machine learning, and other advanced methods that Google uses to protect its own services.
- Your existing security operations center (SOC) doesn't have the skills or capacity to generate threat insights from a large volume of cloud logs.

Avoid this option when your existing security tools can fully address ephemeral or serverless cloud resources, monitor for insecure configurations, and identify threats at scale in a cloud environment.

Option 2: Use your existing security tools for cloud security posture management and threat detection

As an alternative option to Use Security Command Center, you might consider other cloud security posture management tools. Various third-party tools exist that have similar functions to Security Command Center, and you might already have invested in cloud-native tools that are focused on multi-cloud environments.

You can also use Security Command Center and third-party tools together. For example, you might ingest the finding notifications from Security Command Center to another tool, or you might add a third-party security service to the Security Command Center dashboard. As another example, you might have a requirement to store logs on an existing SIEM system for the SOC team to analyze for threats. You could configure your existing SIEM to ingest only the finding notifications that Security Command Center produces, instead of ingesting a large volume of logs and expecting a SOC team to analyze the raw logs for insight.

Use this option when your existing security tools can fully address ephemeral or serverless cloud resources, monitor for insecure configurations, and identify threats at scale in a cloud environment.

Avoid this option when the following is true:

- Your existing SOC doesn't have the skills or capacity to generate threat insights from the vast volume of cloud logs.
- Integrating multiple third-party tools with multiple Google Cloud services introduces more complexity than value.

For more information, see the following:

- Enable finding notifications for Pub/Sub
- Managing security sources using the Security Command Center API
- Add a third-party security service

Decide how to centrally aggregate necessary logs

Most audit logs are stored in the Google Cloud project that produced them. As your environment grows, it can be untenable for an auditor to check logs in every individual project. Therefore, you need to make a decision on how logs will be centralized and aggregated to help your internal audit and security operations.

The following sections describe the options for aggregating logs. We recommend option 1 for most use cases. The other options discussed in the following sections are alternatives that you can consider if option 1 doesn't apply to your specific use case.

Option 1: Retain logs in Google Cloud by using aggregated logs sinks

We recommend that you configure a centralized organization-wide log sink for audit logs and other logs that are required by your security team. You can reference the logs scoping tool to identify the logs that your security team requires and whether these log types require explicit enablement.

For example, the security team expects a central record of any resources that your users create so that the security team can monitor and investigate suspicious changes. The security team also requires an immutable record of data access for certain highly sensitive workloads. Therefore, the security team configures one log sink to aggregate administrator activity audit logs from all projects into a Log Analytics bucket in a central project that they can view for impromptu investigations. They then configure a second log sink for data access audit logs from projects with sensitive workloads into a Cloud Storage bucket for long-term retention.

Use this option when the following is true:

- Your security team expects a central record of all audit logs or other specific log types.
- Your security team needs to store logs in an environment with restricted access, outside the control of the workload or teams who produced the log.

Avoid this option when the following is true:

- Your organization doesn't have a central requirement for consistent audit logs across workloads.
- Individual project owners have full responsibility for managing their own audit logs.

For more information, see the following:

- Detective controls in the enterprise foundations blueprint
- Best practices for Cloud Audit Logs
- Configure aggregated sinks
- Log scoping tool
- Retention policies and retention policy locks

Option 2: Export required audit logs to storage outside of Google Cloud

As an alternative to storing logs in Google Cloud only, consider exporting audit logs outside of Google Cloud. After you centralize necessary log types into an aggregate log sink in Google Cloud, ingest the contents of that sink to another platform outside of Google Cloud for storing and analyzing logs.

For example, you might use a third-party SIEM to aggregate and analyze audit logs across multiple cloud providers. This tool has sufficient capabilities to work with serverless cloud resources, and your SOC team has the skills and capacity to generate insight from this large volume of logs.

This option can potentially be very expensive because of the network egress cost in Google Cloud, as well as the storage cost and capacity in the other environment. Rather than exporting every available log, we recommend that you be selective about which logs are required in the external environment.

Use this option when you have a requirement to store logs from all environments and cloud providers in a single central location.

Avoid this option when the following is true:

- Your existing systems don't have the capacity or budget to ingest a large volume of additional cloud logs.
- Your existing systems require integration efforts for each log type and format.
- You are collecting logs without a clear goal of how they will be used.

For more information, see Detective controls in the enterprise foundations blueprint.

Decide how to meet compliance requirements for encryption at rest

Google Cloud automatically encrypts all your content stored at rest, using one or more encryption mechanisms. Depending on your compliance requirements, you might have an obligation to manage the encryption keys yourself.

The following sections describe the options for encryption at rest. We recommend option 1 for most use cases. The other options discussed in the following sections are alternatives that you can consider if option 1 doesn't apply to your specific use case.

Option 1: Accept use of default encryption at rest

Default encryption at rest is sufficient for many use cases that don't have particular compliance requirements regarding encryption key management.

For example, the security team at an online gaming company requires all customer data to be encrypted at rest. They don't have regulatory requirements about key management, and after reviewing Google's default encryption at rest, they are satisfied that it's a sufficient control for their needs.

Use this option when the following is true:

- You don't have particular requirements around how to encrypt data or how encryption keys are managed.
- You prefer a managed service over the cost and operational overhead of managing your own encryption keys.

Avoid this option when you have compliance requirements to manage your own encryption keys.

For more information, see Encryption at rest in Google Cloud.

Option 2: Manage encryption keys using Cloud KMS

In addition to default encryption at rest, you might require more control over the keys used to encrypt data at rest within a Google Cloud project. Cloud Key Management Service (Cloud KMS) offers the ability to protect your data using customer-managed encryption keys (CMEK). For example, in the financial services industry, you might have a requirement to report to your external auditors how you manage your own encryption keys for sensitive data.

For additional layers of control, you can configure hardware security modules (HSM) or external key management (EKM) with CMEK. Customer-supplied encryption keys (CSEK) are not recommended; scenarios that historically were addressed by CSEK are now better addressed by Cloud External Key Manager (Cloud EKM) because Cloud EKM has support for more services and higher availability.

This option shifts some responsibility to application developers to follow the key management that your security team mandates. The security team can enforce the requirement by blocking the creation of non-compliant resources with CMEK organization policies.

Use this option when one or more of the following is true:

- You have a requirement to manage the lifecycle of your own keys.
- You have a requirement to generate cryptographic key material with a FIPS 140-2 Level 3 certified HSM.
- You have a requirement to store cryptographic key material outside of Google Cloud using Cloud EKM.

Avoid this option when the following is true:

- You don't have particular requirements for how to encrypt data or how encryption keys are managed.
- You prefer a managed service over the cost and operational overhead of managing your own encryption keys.

For more information, see the following:

- Manage encryption keys with Cloud Key Management Service in the enterprise foundations blueprint
- Customer-managed encryption keys (CMEK)
- Cloud HSM

- Cloud External Key Manager
- CMEK organization policies

Option 3: Tokenize data at the application layer before persisting in storage

In addition to the default encryption provided by Google, you can also develop your own solution to tokenize data before storing it in Google Cloud.

For example, a data scientist must analyze a dataset with PII information, but the data scientist shouldn't have access to read the raw data of some sensitive fields. To limit control access to raw data, you could develop an ingestion pipeline with Sensitive Data Protection to ingest and tokenize sensitive data, and then persist the tokenized data to BigQuery.

Tokenizing data is not a control that you can centrally enforce in the landing zone. Instead, this option shifts the responsibility to application developers to configure their own encryption, or to a platform team who develops an encryption pipeline as a shared service for application developers to use.

Use this option when particular workloads have highly sensitive data that must not be used in its raw form.

Avoid this option when Cloud KMS is sufficient to meet your requirements, as described in Manage encryption keys using Cloud KMS. Moving data through additional encryption and decryption pipelines adds cost, latency, and complexity to applications.

For more information, see Sensitive Data Protection overview.

Decide how to meet compliance requirements for encryption in transit

Google Cloud has several security measures to help ensure the authenticity, integrity, and privacy of data in transit. Depending on your security and compliance requirements, you might also configure application layer encryption.

The following sections describe the options for encryption in transit. We recommend option 1 for most use cases. The other options discussed in the following sections are additional features that you can consider if option 1 is insufficient for your specific use case.

Option 1: Evaluate whether Google Cloud encryption in transit is sufficient

Many traffic patterns in your landing zone benefit from Google Cloud default encryption in transit.

- VM-to-VM connections within VPC networks and peered VPC networks that are inside of Google's production network are integrity-protected and encrypted.

- Traffic from the internet to Google services and traffic from your VPC network to Google services terminate at the Google Front End (GFE). GFE also does the following:
    - Provides DDoS attack countermeasures
    - Routes and load balances traffic to the Google Cloud services
- Google's infrastructure uses ALTS for the authentication, integrity, and encryption of connections from the GFE to a Google Cloud service, and from one Google Cloud service to another Google Cloud service.

Use this option when Google Cloud default encryption in transit is sufficient for your internal workloads.

Add additional protection when the following is true:

- You allow internet ingress traffic into your VPC network.
- You're connecting to your on-premises environment.

In these cases, you should configure additional controls, as discussed in Option 2: Require applications to configure Layer 7 encryption in transit and Option 3: Configure additional encryption to your on-premises environment.

For more information about Google Cloud default encryption in transit, see Encryption in transit in Google Cloud.

Option 2: Configure addition encryption for traffic to or from your customer applications

In addition to default encryption in transit, you can configure Layer 7 encryption for application traffic to your customer applications. Google Cloud provides managed services to support applications that need application-layer encryption in transit, including managed certificates and Cloud Service Mesh.

For example, a developer is building a new application that accepts ingress traffic from the internet. They use an external Application Load Balancer with Google-managed SSL certificates to run and scale services behind a single IP address.

Application layer encryption is not a control that you can enforce centrally in the landing zone. Instead, this option shifts some responsibility to application developers to configure encryption in transit.

Use this option when applications require HTTPS or SSL traffic between components.

Consider allowing a limited exception to this option when you are migrating compute workloads to the cloud that did not previously require encryption in transit for internal traffic when the applications were on-premises. During a large-scale migration, forcing legacy applications to modernize before migration might cause an unacceptable delay to the program.

For more information, see the following:

- Using Google-managed SSL certificates
- Using self-managed SSL certificates
- Cloud Service Mesh

Option 3: Configure additional encryption to your on-premises environment

If you require a connection from Google Cloud to your on-premises environment, you can configure Cloud Interconnect, which sets up a direct physical connection between Google Cloud and your data centers. When using Cloud Interconnect, traffic from your on-premises environment to Google Cloud isn't encrypted in transit by default. Therefore, if you're using Cloud Interconnect, we recommend that you enable MACsec for Cloud Interconnect as part of your landing zone.

If you use HA VPN to connect private traffic between Google Cloud and your data centers, traffic uses IPsec encryption by default.

For more information, see Choosing a Network Connectivity Center product.

Decide which additional controls are necessary for managing cloud service provider access

The need to audit cloud service provider (CSP) access can be a concern during cloud adoption. Google Cloud provides multiple layers of control that can enable verification of cloud provider access.

The following sections describe the options for managing CSP access. We recommend option 1 for most use cases. The other option discussed in the following sections is an additional feature that you can consider if option 1 is insufficient for your specific use case.

Option 1: Enable Access Transparency logs

Access Transparency logs record the actions taken by Google Cloud personnel in your environment, such as when they troubleshoot a support case.

For example, your developer raises a troubleshooting issue to Cloud Customer Care, and asks the support agent to help troubleshoot their environment. An Access Transparency log is generated to show what actions the support staff took, including the support case number that justified it.

Use this option when the following is true:

- You have a requirement to verify that Google Cloud personnel are accessing your content only for valid business reasons, such as fixing an outage or attending to your support requests.
- You have a compliance requirement to track access to sensitive data.

Option 2: Enable Access Transparency logs and provider access management

If your business has a compliance requirement to grant explicit approval before a CSP can access your environment, in addition to Option 1, you can use Access Transparency with other controls that let you explicitly approve or deny access to your data.

Access Approval is a manual solution that ensures that Customer Care and Google engineering require your explicit approval (through email or through a webhook) whenever they need to access your content.

Key Access Justifications is a programmatic solution that adds a justification field to any requests for encryption keys that are configured with Cloud EKM.

Use this option when the following is true:

- You want a central team to directly manage access to your content by Google personnel.
- For Access Approval, you can accept the risk that the central capability to approve or deny each access request is more important than the operational trade-off, which could be a slower resolution of support cases.
- For Key Access Justifications, your business is already using Cloud External Key Manager as part of your encryption strategy, and wants programmatic control over all types of access to encrypted data (not just Google personnel access to data).

Avoid this option when the following is true:

- The operational delay that can result when a central team with Access Approval authority must respond is an unacceptable risk to workloads that require high availability and a rapid response from Customer Care.

For more information, see the following:

- Cloud provider access management
- Overview of Access Approval

Security best practices for Google Cloud landing zones

In addition to the decisions introduced in this document, consider the following security best practices:

- Provision the identities in your environment. For more information, see Decide how to onboard identities to Google Cloud.
- Design a network that meets your organization's use cases. For more information, see Decide the network design for your Google Cloud landing zone.
- Implement the organization policy constraints that are defined in the enterprise foundations blueprint. These constraints help to prevent common security issues such as creating unnecessary external IP addresses, or granting the Editor role to all service accounts.
- Review the full list of organization policy constraints to assess whether other controls are relevant for your requirements. All organizations created after May 23, 2024 have Google Cloud security baseline constraints enforced when the organization resource is first created. This change makes option 1 the default option.

What's next

- Review the enterprise foundations blueprint.
- Read more security best practices in the Google Cloud Well-Architected Framework.

Enterprise foundations blueprint

Last reviewed 2025-05-15 UTC

This document describes the best practices that let you deploy a foundational set of resources in Google Cloud. A cloud foundation is the baseline of resources, configurations, and capabilities that enable companies to adopt Google Cloud for their business needs. A well-designed foundation enables consistent governance, security controls, scale, visibility, and access to shared services across all workloads in your Google Cloud environment. After you deploy the controls and governance that are described in this document, you can deploy workloads to Google Cloud.

The *enterprise foundations blueprint* (formerly known as the *security foundations blueprint*) is intended for architects, security practitioners, and platform engineering teams who are responsible for designing an enterprise-ready environment on Google Cloud. This blueprint consists of the following:

- A terraform-example-foundation GitHub repository that contains the deployable Terraform assets.
- A guide that describes the architecture, design, and controls that you implement with the blueprint (this document).

You can use this guide in one of two ways:

- To create a complete foundation based on Google's best practices. You can deploy all the recommendations from this guide as a starting point, and then customize the environment to address your business' specific requirements.
- To review an existing environment on Google Cloud. You can compare specific components of your design against Google-recommended best practices.

Supported use cases

The enterprise foundation blueprint provides a baseline layer of resources and configurations that help enable all types of workloads on Google Cloud. Whether you're migrating existing compute workloads to Google Cloud, building containerized web applications, or creating big data and machine learning workloads, the enterprise foundation blueprint helps you build your environment to support enterprise workloads at scale.

After you deploy the enterprise foundation blueprint, you can deploy workloads directly or deploy additional blueprints to support complex workloads that require additional capabilities.

A defense-in-depth security model

Google Cloud services benefit from the underlying Google infrastructure security design. It is your responsibility to design security into the systems that you build on top of Google Cloud. The enterprise foundation blueprint helps you to implement a defense-in-depth security model for your Google Cloud services and workloads.

The following diagram shows a defense-in-depth security model for your Google Cloud organization that combines architecture controls, policy controls, and detective controls.

The diagram describes the following controls:

- **Policy controls** are programmatic constraints that enforce acceptable resource configurations and prevent risky configurations. The blueprint uses a combination of policy controls including infrastructure-as-code (IaC) validation in your pipeline and organization policy constraints.
- **Architecture controls** are the configuration of Google Cloud resources like networks and resource hierarchy. The blueprint architecture is based on security best practices.
- **Detective controls** let you detect anomalous or malicious behavior within the organization. The blueprint uses platform features such as Security Command Center, integrates with your existing detective controls and workflows such as a

security operations center (SOC), and provides capabilities to enforce custom detective controls.

Key decisions

This section summarizes the high-level architectural decisions of the blueprint.

The diagram describes how Google Cloud services contribute to key architectural decisions:

- **Cloud Build:** Infrastructure resources are managed using a GitOps model. Declarative IaC is written in Terraform and managed in a version control system for review and approval, and resources are deployed using Cloud Build as the continuous integration and continuous deployment (CI/CD) automation tool. The pipeline also enforces policy-as-code checks to validate that resources meet expected configurations before deployment.
- **Cloud Identity:** Users and group membership are synchronized from your existing identity provider. Controls for user account lifecycle management and single sign-on (SSO) rely on the existing controls and processes of your identity provider.
- **Identity and Access Management (IAM):** Allow policies (formerly known as IAM policies) allow access to resources and are applied to groups based on job function. Users are added to the appropriate groups to receive view-only access to foundation resources. All changes to foundation resources are deployed through the CI/CD pipeline which uses privileged service account identities.
- **Resource Manager:** All resources are managed under a single organization, with a resource hierarchy of folders that organizes projects by environments. Projects are labeled with metadata for governance including cost attribution.
- **Networking**: Network topologies use Shared VPC to provide network resources for workloads across multiple regions and zones, separated by environment, and managed centrally. All network paths between on-premises hosts, Google Cloud resources in the VPC networks, and Google Cloud services are private. No outbound traffic to or inbound traffic from the public internet is permitted by default.
- **Cloud Logging**: Aggregated log sinks are configured to collect logs relevant for security and auditing into a centralized project for long-term retention, analysis, and export to external systems.

- **Organization Policy Service:** Organization policy constraints are configured to prevent various high-risk configurations.
- **Secret Manager:** Centralized projects are created for a team responsible for managing and auditing the use of sensitive application secrets to help meet compliance requirements.
- **Cloud Key Management Service (Cloud KMS):** Centralized projects are created for a team responsible for managing and auditing encryption keys to help meet compliance requirements.
- **Security Command Center:** Threat detection and monitoring capabilities are provided using a combination of built-in security controls from Security Command Center and custom solutions that let you detect and respond to security events.

For alternatives to these key decisions, see alternatives.

What's next

- Read about authentication and authorization (next document in this series).

Authentication and authorization

Last reviewed 2025-05-15 UTC

This section introduces how to use Cloud Identity to manage the identities that your employees use to access Google Cloud services.

External identity provider as the source of truth

We recommend federating your Cloud Identity account with your existing identity provider. Federation helps you ensure that your existing account management processes apply to Google Cloud and other Google services.

If you don't have an existing identity provider, you can create user accounts directly in Cloud Identity.

Note: If you're already using Google Workspace, Cloud Identity uses the same console, administrative controls, and user accounts as your Google Workspace account.

The following diagram shows a high-level view of identity federation and single sign-on (SSO). It uses Microsoft Active Directory, located in the on-premises environment, as the example identity provider.

This diagram describes the following best practices:

- User identities are managed in an Active Directory domain that is located in the on-premises environment and federated to Cloud Identity. Active Directory uses Google Cloud Directory Sync to provision identities to Cloud Identity.
- Users attempting to sign in to Google services are redirected to the external identity provider for single sign-on with SAML, using their existing credentials to authenticate. No passwords are synchronized with Cloud Identity.

The following table provides links to setup guidance for identity providers.

Identity provider	Guidance
Active Directory	• Active Directory user account provisioning • Active Directory single sign-on
Microsoft Entra ID (formerly Azure AD)	• Federating Google Cloud with Microsoft Entra ID
Other external identity providers (for example, Ping or Okta)	• Integrating Ping Identity Solutions with Google Identity Services • Okta user provisioning and single sign-on • Best practices for federating Google Cloud with an external identity provider

We strongly recommend that you enforce multi-factor authentication at your identity provider with a phishing-resistant mechanism such as a Titan Security Key.

The recommended settings for Cloud Identity aren't automated through the Terraform code in this blueprint. See administrative controls for Cloud Identity for the recommended security settings that you must configure in addition to deploying the Terraform code.

Groups for access control

A *principal* is an identity that can be granted access to a resource. Principals include Google Accounts for users, Google groups, Google Workspace accounts, Cloud Identity domains, and service accounts. Some services also let you grant access to all users who authenticate with a Google Account, or to all users on the internet. For a principal to interact with Google Cloud services, you must grant them roles in Identity and Access Management (IAM).

To manage IAM roles at scale, we recommend that you assign users to groups based on their job functions and access requirements, then grant IAM roles to those groups. You should add users to groups using the processes in your existing identity provider for group creation and membership.

We don't recommend granting IAM roles to individual users because individual assignments can increase the complexity of managing and auditing roles.

The blueprint configures groups and roles for view-only access to foundation resources. We recommend that you deploy all resources in the blueprint through the foundation pipeline, and that you don't grant roles to users to groups to modify foundation resources outside of the pipeline.

The following table shows the groups that are configured by the blueprint for viewing foundation resources.

Name	Description	Roles	Scope
grp-gcp-org-admin@example.com	Highly privileged administrators who can grant IAM roles at the organization level. They can access any other role. This privilege is not recommended for daily use.	Organization Administrator	organization
grp-gcp-billing-admin@example.com	Highly privileged administrators who can modify the Cloud Billing account. This privilege is not recommended for daily use.	Billing Account Admin	organization
grp-gcp-billing-viewer@example.com	The team who is responsible for viewing and analyzing the spending across all projects.	Billing Account Viewer	organization
		BigQuery User	billing project
grp-gcp-audit-viewer@example.com	The team who is responsible for auditing security-related logs.	Logs Viewer BigQuery User	logging project
grp-gcp-security-reviewer@example.com	The team who is responsible for reviewing cloud security.	Security Reviewer	organization

Name	Description	Roles	Scope
grp-gcp-network-viewer@example.com	The team who is responsible for viewing and maintaining network configurations.	Compute Network Viewer	organization
grp-gcp-scc-admin@example.com	The team who is responsible for configuring Security Command Center.	Security Center Admin Editor	organization
grp-gcp-secrets-admin@example.com	The team who is responsible for managing, storing, and auditing credentials and other secrets that are used by applications.	Secret Manager Admin	secrets projects
grp-gcp-kms-admin@example.com	The team who is responsible for enforcing encryption key management to meet compliance requirements.	Cloud KMS Viewer	kms projects

As you build your own workloads on top of the foundation, you create additional groups and grant IAM roles that are based on the access requirements for each workload.

We strongly recommend that you avoid basic roles (such as Owner, Editor, or Viewer) and use predefined roles instead. Basic roles are overly permissive and a potential security risk. Owner and Editor roles can lead to privilege escalation and lateral movement, and the Viewer role includes access to read all data. For best practices on IAM roles, see Use IAM securely.

Super admin accounts

Cloud Identity users with the super admin account bypass the organization's SSO settings and authenticate directly to Cloud Identity. This exception is by design, so that the super admin can still access the Cloud Identity console in the event of an SSO misconfiguration or outage. However, it means you must consider additional protection for super admin accounts.

To protect your super admin accounts, we recommend that you always enforce 2-step verification with security keys in Cloud Identity. For more information, see Security best practices for administrator accounts.

Issues with consumer user accounts

If you didn't use Cloud Identity or Google Workspace before you onboarded to Google Cloud, it's possible that your organization's employees are already using consumer accounts that are associated with their corporate email identities to access other Google services such as Google Marketing Platform or YouTube. Consumer accounts are accounts that are fully owned and managed by the individuals who created them. Because those accounts aren't under your organization's control and might include both personal and corporate data, you must decide how to consolidate these accounts with other corporate accounts.

We recommend that you consolidate existing consumer user accounts as part of onboarding to Google Cloud. If you aren't using Google Workspace for all your user accounts already, we recommend blocking access to consumer accounts.

Administrative controls for Cloud Identity

Cloud Identity has various administrative controls that are not automated by Terraform code in the blueprint. We recommend that you enforce each of these best practice security controls early in the process of building your foundation.

Control	Description
Deploy 2-step verification	User accounts might be compromised through phishing, social engineering, password spraying, or various other threats. 2-step verification helps mitigate these threats. We recommend that you enforce 2-step verification for all user accounts in your organization with a phishing-resistant mechanism such as Titan Security Keys or other keys that are based on the phishing-resistant FIDO U2F (CTAP1) standards.
Set session length for Google Cloud services	Persistent OAuth tokens on developer workstations can be a security risk if exposed. We recommend that you set a reauthentication policy to require authentication every 16 hours using a security key.
Set session length for Google Services	*(Google Workspace customers only)* Persistent web sessions across other Google services can be a security risk if exposed. We recommend that you enforce a maximum web session length and align this with session length controls in your SSO provider.

Control	Description
Share data from Cloud Identity with Google Cloud services	Admin Activity audit logs from Google Workspace or Cloud Identity are ordinarily managed and viewed in the Admin Console, separately from your logs in your Google Cloud environment. These logs contain information that is relevant for your Google Cloud environment, such as user login events. We recommend that you share Cloud Identity audit logs to your Google Cloud environment to centrally manage logs from all sources.
Set up post SSO verification	The blueprint assumes that you set up SSO with your external identity provider. We recommend that you enable an additional layer of control based on Google's sign-in risk analysis. After you apply this setting, users might see additional risk-based login challenges at sign-in if Google deems that a user sign-in is suspicious.
Remediate issues with consumer user accounts	Users with a valid email address at your domain but no Google Account can sign up for unmanaged consumer accounts. These accounts might contain corporate data, but are not controlled by your account lifecycle management processes. We recommend that you take steps to ensure that all user accounts are managed accounts.
Disable account recovery for super admin accounts	Super admin account self-recovery is off by default for all new customers (existing customers might have this setting on). Turning this setting off helps to mitigate the risk that a compromised phone, compromised email, or social engineering attack could let an attacker gain super admin privileges over your environment. Plan an internal process for a super admin to contact another super admin in your organization if they have lost access to their account, and ensure that all super admins are familiar with the process for support-assisted recovery.
Enforce and monitor password	In most cases, user passwords are managed through your external identity provider, but super admin accounts bypass SSO and must use a password to sign in to Cloud Identity. Disable password reuse and monitor password

Control	Description
requirements for users	strength for any users who use a password to sign in to Cloud Identity, particularly super admin accounts.
Set organization-wide policies for using groups	By default, external user accounts can be added to groups in Cloud Identity. We recommend that you configure sharing settings so that group owners can't add external members. Note that this restriction doesn't apply to the super admin account or other delegated administrators with Groups admin permissions. Because federation from your identity provider runs with administrator privileges, the group sharing settings don't apply to this group synchronization. We recommend that you review controls in the identity provider and synchronization mechanism to ensure that non-domain members aren't added to groups, or that you apply group restrictions.

What's next

- Read about organization structure (next document in this series).

Organization structure

Last reviewed 2025-05-15 UTC

The root node for managing resources in Google Cloud is the organization. The Google Cloud organization provides a resource hierarchy that provides an ownership structure for resources and attachment points for organization policies and access controls. The resource hierarchy consists of folders, projects, and resources, and it defines the structure and use of Google Cloud services within an organization.

Resources lower in the hierarchy inherit policies such as IAM allow policies and organization policies. All access permissions are denied by default, until you apply allow policies directly to a resource or the resource inherits the allow policies from a higher level in the resource hierarchy.

The following diagram shows the folders and projects that are deployed by the blueprint.

The following sections describe the folders and projects in the diagram.

Folders

The blueprint uses folders to group projects based on their environment. This logical grouping is used to apply configurations like allow policies and organization policies at the folder level and then all resources within the folder inherit the policies. The following table describes the folders that are part of the blueprint.

Folder	Description
bootstrap	Contains the projects that are used to deploy foundation components.
common	Contains projects with resources that are shared by all environments.
production	Contains projects with production resources.
nonproduction	Contains a copy of the production environment to let you test workloads before you promote them to production.
development	Contains the cloud resources that are used for development.
networking	Contains the networking resources that are shared by all environments.

Projects

The blueprint uses projects to group individual resources based on their functionality and intended boundaries for access control. This following table describes the projects that are included in the blueprint.

Folder	Project	Description
bootstrap	prj-b-cicd	Contains the deployment pipeline that's used to build out the foundation components of the organization. For more information, see deployment methodology.
	prj-b-seed	Contains the Terraform state of your infrastructure and the Terraform service account that is required to run the pipeline. For more information, see deployment methodology.

Folder	Project	Description
common	prj-c-secrets	Contains organization-level secrets. For more information, see store application credentials with Secret Manager.
	prj-c-logging	Contains the aggregated log sources for audit logs. For more information, see centralized logging for security and audit.
	prj-c-scc	Contains resources to help configure Security Command Center alerting and other custom security monitoring. For more information, see threat monitoring with Security Command Center.
	prj-c-billing-export	Contains a BigQuery dataset with the organization's billing exports. For more information, see allocate costs between internal cost centers.
	prj-c-infra-pipeline	Contains an infrastructure pipeline for deploying resources like VMs and databases to be used by workloads. For more information, see pipeline layers.
	prj-c-kms	Contains encryption keys for encrypting shared services within the common folder. For more information, see manage encryption keys.
networking	prj-net-*{env}*-svpc	Contains the host project for a Shared VPC network. For more information, see network topology.
	prj-net-hub	Contains the Shared VPC network used as a hub between the on-premises environment and Google Cloud spokes. This project is created in the hub-

Folder	Project	Description
		and-spoke topology only. For more information, see network topology.
	prj-net-interconnect	Contains the Cloud Interconnect connections that provide connectivity between your on-premises environment and Google Cloud. For more information, see hybrid connectivity.
environments: - development (d) - non-production (n) - production (p)	prj-*{env}*-*{workload_name_or_id}*	Contains various workload projects in which you create resources for applications. For more information, see project deployment patterns and pipeline layers.
	prj-*{env}*-secrets	Contains folder-level secrets. For more information, see store and audit application credentials with Secret Manager.
	prj-*{env}*-kms	Contains encryption keys for encrypting services within each environment folder. For more information, see manage encryption keys.

Governance for resource ownership

We recommend that you apply labels consistently to your projects to assist with governance and cost allocation. The following table describes the project labels that are added to each project for governance in the blueprint.

Label	Description
application	The human-readable name of the application or workload that is associated with the project.
businesscode	A short code that describes which business unit owns the project. The code shared is used for common projects that are not explicitly tied to a business unit.

Label	Description
billingcode	A code that's used to provide chargeback information.
primarycontact	The username of the primary contact that is responsible for the project. Because project labels can't include special characters such as the ampersand (@), it is set to the username without the @example.com suffix.
secondarycontact	The username of the secondary secondary contact that is responsible for the project. Because project labels can't include special characters such as @, set only the username without the @example.com suffix.
environment	A value that identifies the type of environment, such as bootstrap, common, production, non-production,development, or network.
envcode	A value that identifies the type of environment, shortened to b, c, p, n, d, or net.
vpc	The ID of the VPC network that this project is expected to use.

Google might occasionally send important notifications such as account suspensions or updates to product terms. The blueprint uses Essential Contacts to send those notifications to the groups that you configure during deployment. Essential Contacts is configured at the organization node and inherited by all projects in the organization. We recommend that you review these groups and ensure that emails are monitored reliably.

Essential Contacts is used for a different purpose than the primarycontact and secondarycontact fields that are configured in project labels. The contacts in project labels are intended for internal governance. For example, if you identify non-compliant resources in a workload project and need to contact the owners, you could use the primarycontact field to find the person or team responsible for that workload.

What's next

- Read about networking (next document in this series).

Networking

Last reviewed 2025-05-15 UTC

Networking is required for resources to communicate within your Google Cloud organization and between your cloud environment and on-premises environment. This section describes the structure in the blueprint for VPC networks, IP address space, DNS, firewall policies, and connectivity to the on-premises environment.

Network topology

The blueprint repository provides the following options for your network topology:

- Use separate Shared VPC networks for each environment, with no network traffic directly allowed between environments.
- Use a hub-and-spoke model that adds a hub network to connect each environment in Google Cloud, with the network traffic between environments gated by a network virtual appliance (NVA).

Choose the Shared VPC network for each environment topology when you don't want direct network connectivity between environments. Choose the hub-and-spoke network topology when you want to allow network connectivity between environments that is filtered by an NVA, such as when you rely on existing tools that require a direct network path to every server in your environment.

Both topologies use Shared VPC as a principal networking construct because Shared VPC allows a clear separation of responsibilities. Network administrators manage network resources in a centralized host project, and workload teams deploy their own application resources and consume the network resources in service projects that are attached to the host project.

Shared VPC network for each environment topology

If you require network isolation between your development, non-production, and production networks on Google Cloud, we recommend the Shared VPC network for each environment topology. This topology allows no network traffic between environments.

The following diagram shows the Shared VPC network for each environment topology.

The diagram describes the following key concepts of the Shared VPC for each environment topology:

- Each environment (production, non-production, and development) has one Shared VPC network. This diagram shows only the production environment, but the same pattern is repeated for each environment.
- Each Shared VPC network has two subnets, with each subnet in a different region.

- Connectivity with on-premises resources is enabled through four VLAN attachments to the Dedicated Interconnect instance for each Shared VPC network, using four Cloud Router services (two in each region for redundancy). For more information, see Hybrid connectivity between on-premises environment and Google Cloud.

By design, this topology doesn't allow network traffic to flow directly between environments. If you do require network traffic to flow directly between environments, you must take additional steps to allow this network path. For example, you might configure Private Service Connect endpoints to expose a service from one VPC network to another VPC network. Alternatively, you might configure your on-premises network to let traffic flow from one Google Cloud environment to the on-premises environment and then to another Google Cloud environment.

Hub-and-spoke network topology

If you deploy resources in Google Cloud that require a direct network path to resources in multiple environments, we recommend the hub-and-spoke network topology.

The hub-and-spoke topology uses several of the concepts that are part of the Shared VPC network for each environment topology, but modifies the topology to add a hub network. The following diagram shows the hub-and-spoke topology.

The diagram describes these key concepts of hub-and-spoke network topology:

- This model adds a hub network, and each of the development, non-production, and production networks (spokes) are connected to the hub network through VPC Network Peering. Alternatively, if you anticipate exceeding the quota limit, you can use an HA VPN gateway instead.
- Connectivity to on-premises networks is allowed only through the hub network. All spoke networks can communicate with shared resources in the hub network and use this path to connect to on-premises networks.
- The hub networks include an NVA for each region, deployed redundantly behind internal Network Load Balancer instances. This NVA serves as the gateway to allow or deny traffic to communicate between spoke networks.
- The hub network also hosts tooling that requires connectivity to all other networks. For example, you might deploy tools on VM instances for configuration management to the common environment.

To enable spoke-to-spoke traffic, the blueprint deploys NVAs on the hub Shared VPC network that act as gateways between networks. Routes are exchanged from hub-to-spoke VPC networks through custom routes exchange. In this scenario, connectivity between spokes must be routed through the NVA because VPC Network Peering is non-transitive, and therefore, spoke VPC networks can't exchange data with each other directly. You must configure the virtual appliances to selectively allow traffic between spokes.

Project deployment patterns

When creating new projects for workloads, you must decide how resources in this project connect to your existing network. The following table describes the patterns for deploying projects that are used in the blueprint.

Pattern	Description	Example usage
Shared VPC service projects	These projects are configured as service projects to a Shared VPC host project. Use this pattern when resources in your project have the following criteria: • Require network connectivity to the on-premises environment or resources in the same Shared VPC topology.	example_shared_vpc_project.t
Floating projects	Floating projects are not connected to other VPC networks in your topology. Use this pattern when resources in your project have the following criteria: • Don't require full mesh connectivity to an on-premises environment or resources in the Shared VPC topology. • Don't require a VPC network, or you want to manage the VPC network for this project independently of your main VPC network topology (such as when you want to use an IP address range that clashes with the ranges already in use). You might have a scenario where you want to keep the VPC network of a floating project separate from the main VPC network topology but also want to expose a	example_floating_project.tf

Pattern	Description	Example usage
	limited number of endpoints between networks. In this case, publish services by using Private Service Connect to share network access to an individual endpoint across VPC networks without exposing the entire network.	
Peering projects	Peering projects create their own VPC networks and peer to other VPC networks in your topology. Use this pattern when resources in your project have the following criteria: • Require network connectivity in the directly peered VPC network, but don't require transitive connectivity to an on-premises environment or other VPC networks. • Must manage the VPC network for this project independently of your main network topology. If you create peering projects, it's your responsibility to allocate non-conflicting IP address ranges and plan for peering group quota.	example_peering_project.tf

IP address allocation

This section introduces how the blueprint architecture allocates IP address ranges. You might need to change the specific IP address ranges used based on the IP address availability in your existing hybrid environment.

The following table provides a breakdown of the IP address space that's allocated for the blueprint. The hub environment only applies in the hub-and-spoke topology.

Purpose	Region	Hub environment	Development environment	Non-production environment	Production environment
Primary subnet ranges	Region 1	10.8.0.0/18	10.8.64.0/18	10.8.128.0/18	10.8.192.0/18
	Region 2	10.9.0.0/18	10.9.64.0/18	10.9.128.0/18	10.9.192.0/18

Purpose	Region	Hub environment	Development environment	Non-production environment	Production environment
	Unallocated	10.*{10-15}*.0.0/18	10.*{10-15}*.64.0/18	10.*{10-15}*.128.0/18	10.*{10-15}*.192.0/18
Private services access	Global	10.16.32.0/21	10.16.40.0/21	10.16.48.0/21	10.16.56.0/21
Private Service Connect endpoints	Global	10.17.0.5/32	10.17.0.6/32	10.17.0.7/32	10.17.0.8/32
Proxy-only subnets	Region 1	10.26.0.0/23	10.26.2.0/23	10.26.4.0/23	10.26.6.0/23
	Region 2	10.27.0.0/23	10.27.2.0/23	10.27.4.0/23	10.27.6.0/23
	Unallocated	10.*{28-33}*.0.0/23	10.*{28-33}*.2.0/23	10.*{28-33}*.4.0/23	10.*{28-33}*.6.0/23
Secondary subnet ranges	Region 1	100.72.0.0/18	100.72.64.0/18	100.72.128.0/18	100.72.192.0/18
	Region 2	100.73.0.0/18	100.73.64.0/18	100.73.128.0/18	100.73.192.0/18
	Unallocated	100.*{74-79}*.0.0/18	100.*{74-79}*.64.0/18	100.*{74-79}*.128.0/18	100.*{74-79}*.192.0/18

The preceding table demonstrates these concepts for allocating IP address ranges:

- IP address allocation is subdivided into ranges for each combination of purpose, region, and environment.
- Some resources are global and don't require subdivisions for each region.
- By default, for regional resources, the blueprint deploys in two regions. In addition, there are unused IP address ranges so that you can can expand into six additional regions.

- The hub network is only used in the hub-and-spoke network topology, while the development, non-production, and production environments are used in both network topologies.

The following table introduces how each type of IP address range is used.

Purpose	Description
Primary subnet ranges	Resources that you deploy to your VPC network, such as virtual machine instances, use internal IP addresses from these ranges.
Private services access	Some Google Cloud services such as Cloud SQL require you to preallocate a subnet range for private services access. The blueprint reserves a /21 range globally for each of the Shared VPC networks to allocate IP addresses for services that require private services access. When you create a service that depends on private services access, you allocate a regional /24 subnet from the reserved /21 range.
Private Service Connect	The blueprint provisions each VPC network with a Private Service Connect endpoint to communicate with Google Cloud APIs. This endpoint lets your resources in the VPC network reach Google Cloud APIs without relying on outbound traffic to the internet or publicly advertised internet ranges.
Proxy-based load balancers	Some types of Application Load Balancers require you to preallocate proxy-only subnets. Although the blueprint doesn't deploy Application Load Balancers that require this range, allocating ranges in advance helps reduce friction for workloads when they need to request a new subnet range to enable certain load balancer resources.
Secondary subnet ranges	Some use cases, such as container-based workloads, require secondary ranges. Although the blueprint doesn't deploy services that require secondary ranges, it allocates ranges from the RFC 6598 IP address space for secondary ranges. Alternatively, if you struggle to allocate sufficiently large CIDR blocks for these services, you might consider deploying those services in the floating project pattern introduced in the project deployment patterns section.

Centralized DNS setup

For DNS resolution between Google Cloud and on-premises environments, we recommend that you use a hybrid approach with two authoritative DNS systems. In this approach, Cloud DNS handles authoritative DNS resolution for your Google Cloud environment and your existing on-premises DNS servers handle authoritative DNS resolution for on-premises resources. Your on-premises environment and Google Cloud environment perform DNS lookups between environments through forwarding requests.

The following diagram demonstrates the DNS topology across the multiple VPC networks that are used in the blueprint.

The diagram describes the following components of the DNS design that is deployed by the blueprint:

- The DNS hub is the central point of DNS exchange between the on-premises environment and the Google Cloud environment. DNS forwarding uses the same Dedicated Interconnect instances and Cloud Routers that are already configured in your network topology.
    - In the Shared VPC network for each environment topology, the DNS project is the same as the production Shared VPC host project.
    - In the hub-and-spoke topology, the DNS hub project is the same as the hub Shared VPC host project.
- Servers in each Shared VPC network can resolve DNS records from other Shared VPC networks through DNS forwarding, which is configured between Cloud DNS in each Shared VPC host project and the DNS hub.
- On-premises servers can resolve DNS records in Google Cloud environments using DNS server policies that allow queries from on-premises servers. The blueprint configures an inbound server policy in the DNS hub to allocate IP addresses, and the on-premises DNS servers forward requests to these addresses. All DNS requests to Google Cloud reach the DNS hub first, which then resolves records from DNS peers.
- Servers in Google Cloud can resolve DNS records in the on-premises environment using forwarding zones that query on-premises servers. All DNS requests to the on-premises environment originate from the DNS hub. The DNS request source is 35.199.192.0/19.

Firewall policies

Google Cloud has multiple firewall policy types. Hierarchical firewall policies are enforced at the organization or folder level to inherit firewall policy rules consistently across all resources in the hierarchy. In addition, you can configure network firewall policies for each VPC network. The blueprint combines these firewall policies to enforce common configurations across all environments using Hierarchical firewall policies and to enforce more specific configurations at each individual VPC network using network firewall policies.

The blueprint doesn't use legacy VPC firewall rules. We recommend using only firewall policies and avoid mixing use with legacy VPC firewall rules.

Hierarchical firewall policies

The blueprint defines a single hierarchical firewall policy and attaches the policy to each of the production, non-production, development, bootstrap, and common folders. This hierarchical firewall policy contains the rules that should be enforced broadly across all environments, and delegates the evaluation of more granular rules to the network firewall policy for each individual environment.

The following table describes the hierarchical firewall policy rules deployed by the blueprint.

Rule description	Direction of traffic	Filter (IPv4 range)	Protocols and ports	Action
Delegate the evaluation of inbound traffic from RFC 1918 to lower levels in the hierarchy.	Ingress	192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12	all	Go to next
Delegate the evaluation of outbound traffic to RFC 1918 to lower levels in the hierarchy.	Egress	192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12	all	Go to next
IAP for TCP forwarding	Ingress	35.235.240.0/20	tcp:22,3389	Allow
Windows server activation	Egress	35.190.247.13/32	tcp:1688	Allow
Health checks for Cloud Load Balancing	Ingress	130.211.0.0/22, 35.191.0.0/16, 209.85.152.0/22, 209.85.204.0/22	tcp:80,443	Allow

Network firewall policies

The blueprint configures a network firewall policy for each network. Each network firewall policy starts with a minimum set of rules that allow access to Google Cloud services and deny egress to all other IP addresses.

In the hub-and-spoke model, the network firewall policies contain additional rules to allow communication between spokes. The network firewall policy allows outbound traffic from one to the hub or another spoke, and allows inbound traffic from the NVA in the hub network.

The following table describes the rules in the global network firewall policy deployed for each VPC network in the blueprint.

Rule description	Direction of traffic	Filter	Protocols and ports
Allow outbound traffic to Google Cloud APIs.	Egress	The Private Service Connect endpoint that is configured for each individual network. See Private access to Google APIs.	tcp:443
Deny outbound traffic not matched by other rules.	Egress	all	all
Allow outbound traffic from one spoke to another spoke (for hub-and-spoke model only).	Egress	The aggregate of all IP addresses used in the hub-and-spoke topology. Traffic that leaves a spoke VPC is routed to the NVA in the hub network first.	all
Allow inbound traffic to a spoke from the NVA in the hub network (for hub-and-spoke model only).	Ingress	Traffic originating from the NVAs in the hub network.	all

When you first deploy the blueprint, a VM instance in a VPC network can communicate with Google Cloud services, but not to other infrastructure resources in the same VPC network. To allow VM instances to communicate, you must add additional rules to your network firewall policy and tags that explicitly allow the VM instances to communicate. Tags are added to VM instances, and traffic is evaluated against those tags. Tags additionally have IAM controls so that you can define them centrally and delegate their use to other teams.

Note: All references to tags in this document refer to tags with IAM controls. We don't recommend VPC firewall rules with legacy network tags.

The following diagram shows an example of how you can add custom tags and network firewall policy rules to let workloads communicate inside a VPC network.

The diagram demonstrates the following concepts of this example:

- The network firewall policy contains Rule 1 that denies outbound traffic from all sources at priority 65530.
- The network firewall policy contains Rule 2 that allows inbound traffic from instances with the service=frontend tag to instances with the service=backend tag at priority 999.
- The instance-2 VM can receive traffic from instance-1 because the traffic matches the tags allowed by Rule 2. Rule 2 is matched before Rule 1 is evaluated, based on the priority value.
- The instance-3 VM doesn't receive traffic. The only firewall policy rule that matches this traffic is Rule 1, so outbound traffic from instance-1 is denied.

Private access to Google Cloud APIs

To let resources in your VPC networks or on-premises environment reach Google Cloud services, we recommend private connectivity instead of outbound internet traffic to public API endpoints. The blueprint configures Private Google Access on every subnet, creates internal endpoints with Private Service Connect to communicate with Google Cloud services, and configures firewall policies and DNS records to allow traffic to those endpoints. Used together, these controls allow a private path to Google Cloud services, without relying on internet outbound traffic or publicly advertised internet ranges.

The blueprint configures Private Service Connect endpoints with the API bundle called vpc-sc, which allows access to the Google Cloud services supported by the restricted VIP. This control helps mitigate exfiltration risk by preventing access to other Google APIs not related to Google Cloud. This control is also a prerequisite step for enabling VPC Service Controls. For more information about the optional steps to enable VPC Service Controls, see protect your resources with VPC Service Controls.

The following table describes the Private Service Connect endpoints created for each network.

Environment	API bundle	Private Service Connect endpoint IP address
Common	vpc-sc	10.17.0.5/32
Development	vpc-sc	10.17.0.6/32
Non-production	vpc-sc	10.17.0.7/32
Production	vpc-sc	10.17.0.8/32

To ensure that traffic for Google Cloud services has a DNS lookup to the correct endpoint, the blueprint configures private DNS zones for each VPC network. The following table describes these private DNS zones.

Private zone name	DNS name	Record type	Data
googleapis.com.	*.googleapis.com.	CNAME	restricted.googleapis.com.
	restricted.googleapis.com	A	The Private Service Connect endpoint IP address for that VPC network.
gcr.io.	*.gcr.io	CNAME	gcr.io.
	gcr.io	A	The Private Service Connect endpoint IP address for that VPC network.
pkg.dev.	*.pkg.dev.	CNAME	pkg.dev.
	pkg.dev.	A	The Private Service Connect endpoint IP address for that VPC network.

The blueprint has additional configurations to enforce that these Private Service Connect endpoints are used consistently. Each Shared VPC network also enforces the following:

- A network firewall policy rule that allows outbound traffic from all sources to the IP address of the Private Service Connect endpoint on TCP:443.

- A network firewall policy rule that denies outbound traffic to 0.0.0.0/0, which includes the default domains that are used for access to Google Cloud services.

Internet connectivity

The blueprint doesn't allow inbound or outbound traffic between its VPC networks and the internet. For workloads that require internet connectivity, you must take additional steps to design the access paths required.

For workloads that require outbound traffic to the internet, we recommend that you manage outbound traffic through Cloud NAT to allow outbound traffic without unsolicited inbound connections, or through Secure Web Proxy for more granular control to allow outbound traffic to trusted web services only.

For workloads that require inbound traffic from the internet, we recommend that you design your workload with Cloud Load Balancing and Google Cloud Armor to benefit from DDoS and WAF protections.

We don't recommend that you design workloads that allow direct connectivity between the internet and a VM using an external IP address on the VM.

Hybrid connectivity between an on-premises environment and Google Cloud

To establish connectivity between the on-premises environment and Google Cloud, we recommend that you use Dedicated Interconnect to maximize security and reliability. A Dedicated Interconnect connection is a direct link between your on-premises network and Google Cloud.

The following diagram introduces hybrid connectivity between the on-premises environment and a Google Virtual Private Cloud network.

The diagram describes the following components of the pattern for 99.99% availability for Dedicated Interconnect:

- Four Dedicated Interconnect connections, with two connections in one metropolitan area (metro) and two connections in another metro. Within each metro, there are two distinct zones within the colocation facility.
- The connections are divided into two pairs, with each pair connected to a separate on-premises data center.
- VLAN attachments are used to connect each Dedicated Interconnect instance to Cloud Routers that are attached to the Shared VPC topology.

- Each Shared VPC network has four Cloud Routers, two in each region, with the dynamic routing mode set to global so that every Cloud Router can announce all subnets, independent of region.

With global dynamic routing, Cloud Router advertises routes to all subnets in the VPC network. Cloud Router advertises routes to remote subnets (subnets outside of the Cloud Router's region) with a lower priority compared to local subnets (subnets that are in the Cloud Router's region). Optionally, you can change advertised prefixes and priorities when you configure the BGP session for a Cloud Router.

Traffic from Google Cloud to an on-premises environment uses the Cloud Router closest to the cloud resources. Within a single region, multiple routes to on-premises networks have the same multi-exit discriminator (MED) value, and Google Cloud uses equal cost multi-path (ECMP) routing to distribute outbound traffic between all possible routes.

On-premises configuration changes

To configure connectivity between the on-premises environment and Google Cloud, you must configure additional changes in your on-premises environment. The Terraform code in the blueprint automatically configures Google Cloud resources but doesn't modify any of your on-premises network resources.

Some of the components for hybrid connectivity from your on-premises environment to Google Cloud are automatically enabled by the blueprint, including the following:

- Cloud DNS is configured with DNS forwarding between all Shared VPC networks to a single hub, as described in DNS setup. A Cloud DNS server policy is configured with inbound forwarder IP addresses.
- Cloud Router is configured to export routes for all subnets and custom routes for the IP addresses used by the Private Service Connect endpoints.

To enable hybrid connectivity, you must take the following additional steps:

1. Order a Dedicated Interconnect connection.
2. Configure on-premises routers and firewalls to allow outbound traffic to the internal IP address space defined in IP address space allocation.
3. Configure your on-premises DNS servers to forward DNS lookups bound for Google Cloud to the inbound forwarder IP addresses that is already configured by the blueprint.
4. Configure your on-premises DNS servers, firewalls, and routers to accept DNS queries from the Cloud DNS forwarding zone (35.199.192.0/19).

5. Configure on-premise DNS servers to respond to queries from on-premises hosts to Google Cloud services with the IP addresses defined in private access to Cloud APIs.
6. For encryption in transit over the Dedicated Interconnect connection, configure MACsec for Cloud Interconnect or configure HA VPN over Cloud Interconnect for IPsec encryption.

For more information, see Private Google Access for on-premises hosts.

What's next

- Read about detective controls (next document in this series).

Detective controls

Last reviewed 2025-05-15 UTC

Threat detection and monitoring capabilities are provided using a combination of built-in security controls from Security Command Center and custom solutions that let you detect and respond to security events.

Centralized logging for security and audit

The blueprint configures logging capabilities to track and analyze changes to your Google Cloud resources with logs that are aggregated to a single project.

The following diagram shows how the blueprint aggregates logs from multiple sources in multiple projects into a centralized log sink.

Logging structure for example.com.

The diagram describes the following:

Log sinks are configured at the organization node to aggregate logs from all projects in the resource hierarchy.

Multiple log sinks are configured to send logs that match a filter to different destinations for storage and analytics.

The prj-c-logging project contains all the resources for log storage and analytics.

Optionally, you can configure additional tooling to export logs to a SIEM.

The blueprint uses different log sources and includes these logs in the log sink filter so that the logs can be exported to a centralized destination. The following table describes the log sources.

Log source

Description

Admin Activity audit logs

You cannot configure, disable, or exclude Admin Activity audit logs.

System Event audit logs

You cannot configure, disable, or exclude System Event audit logs.

Policy Denied audit logs

You cannot configure or disable Policy Denied audit logs, but you can optionally exclude them with exclusion filters.

Data Access audit logs

By default, the blueprint doesn't enable data access logs because the volume and cost of these logs can be high.

To determine whether you should enable data access logs, evaluate where your workloads handle sensitive data and consider whether you have a requirement to enable data access logs for each service and environment working with sensitive data.

VPC Flow Logs

The blueprint enables VPC Flow Logs for every subnet. The blueprint configures log sampling to sample 50% of logs to reduce cost.

If you create additional subnets, you must ensure that VPC Flow Logs are enabled for each subnet.

Firewall Rules Logging

The blueprint enables Firewall Rules Logging for every firewall policy rule.

If you create additional firewall policy rules for workloads, you must ensure that Firewall Rules Logging is enabled for each new rule.

Cloud DNS logging

The blueprint enables Cloud DNS logs for managed zones.

If you create additional managed zones, you must enable those DNS logs.

Google Workspace audit logging

Requires a one-time enablement step that is not automated by the blueprint. For more information, see Share data with Google Cloud services.

Access Transparency logs

Requires a one-time enablement step that is not automated by the blueprint. For more information, see Enable Access Transparency.

The following table describes the log sinks and how they are used with supported destinations in the blueprint.

Sink

Destination

Purpose

sk-c-logging-la

Logs routed to Cloud Logging buckets with Log Analytics and a linked BigQuery dataset enabled

Actively analyze logs. Run ad hoc investigations by using Logs Explorer in the console, or write SQL queries, reports, and views using the linked BigQuery dataset.

sk-c-logging-bkt

Logs routed to Cloud Storage

Store logs long-term for compliance, audit, and incident-tracking purposes.

Optionally, if you have compliance requirements for mandatory data retention, we recommend that you additionally configure Bucket Lock.

sk-c-logging-pub

Logs routed to Pub/Sub

Export logs to an external platform such as your existing SIEM.

This requires additional work to integrate with your SIEM, such as the following mechanisms:

For many tools, third-party integration with Pub/Sub is the preferred method to ingest logs.

For Google Google Security Operations, you can ingest Google Cloud data to Google Security Operations without provisioning additional infrastructure.

For Splunk, you can stream logs from Google Cloud to Splunk using Dataflow.

For guidance on enabling additional log types and writing log sink filters, see the log scoping tool.

Threat monitoring with Security Command Center

We recommend that you activate Security Command Center Premium for your organization to automatically detect threats, vulnerabilities, and misconfigurations in your Google Cloud resources. Security Command Center creates security findings from multiple sources including the following:

Security Health Analytics: detects common vulnerabilities and misconfigurations across Google Cloud resources.

Attack path exposure: shows a simulated path of how an attacker could exploit your high-value resources, based on the vulnerabilities and misconfigurations that are detected by other Security Command Center sources.

Event Threat Detection: applies detection logic and proprietary threat intelligence against your logs to identify threats in near-real time.

Container Threat Detection: detects common container runtime attacks.

Virtual Machine Threat Detection: detects potentially malicious applications that are running on virtual machines.

Web Security Scanner: scans for OWASP Top Ten vulnerabilities in your web-facing applications on Compute Engine, App Engine, or Google Kubernetes Engine.

For more information on the vulnerabilities and threats addressed by Security Command Center, see Security Command Center sources.

You must activate Security Command Center after you deploy the blueprint. For instructions, see Activate Security Command Center for an organization.

After you activate Security Command Center, we recommend that you export the findings that are produced by Security Command Center to your existing tools or processes for triaging and responding to threats. The blueprint creates the prj-c-scc project with a Pub/Sub topic to be used for this integration. Depending on your existing tools, use one of the following methods to export findings:

If you use the console to manage security findings directly in Security Command Center, configure folder-level and project-level roles for Security Command Center to let teams view and manage security findings just for the projects for which they are responsible.

If you use Google SecOps as your SIEM, ingest Google Cloud data to Google SecOps.

If you use a SIEM or SOAR tool with integrations to Security Command Center, share data with Cortex XSOAR, Elastic Stack, ServiceNow, Splunk, or QRadar.

If you use an external tool that can ingest findings from Pub/Sub, configure continuous exports to Pub/Sub and configure your existing tools to ingest findings from the Pub/Sub topic.

Custom solution for automated log analysis

You might have requirements to create alerts for security events that are based on custom queries against logs. Custom queries can help supplement the capabilities of your SIEM by analyzing logs on Google Cloud and exporting only the events that merit investigation, especially if you don't have the capacity to export all cloud logs to your SIEM.

The blueprint helps enable this log analysis by setting up a centralized source of logs that you can query using a linked BigQuery dataset. To automate this capability, you must implement the code sample at bq-log-alerting and extend the foundation capabilities. The sample code lets you regularly query a log source and send a custom finding to Security Command Center.

The following diagram introduces the high-level flow of the automated log analysis.

Automated logging analysis.

The diagram shows the following concepts of automated log analysis:

Logs from various sources are aggregated into a centralized logs bucket with log analytics and a linked BigQuery dataset.

BigQuery views are configured to query logs for the security event that you want to monitor.

Cloud Scheduler pushes an event to a Pub/Sub topic every 15 minutes and triggers Cloud Run functions.

Cloud Run functions queries the views for new events. If it finds events, it pushes them to Security Command Center as custom findings.

Security Command Center publishes notifications about new findings to another Pub/Sub topic.

An external tool such as a SIEM subscribes to the Pub/Sub topic to ingest new findings.

The sample has several use cases to query for potentially suspicious behavior. Examples include a login from a list of super admins or other highly privileged accounts that you specify, changes to logging settings, or changes to network routes. You can extend the use cases by writing new query views for your requirements. Write your own queries or reference security log analytics for a library of SQL queries to help you analyze Google Cloud logs.

Custom solution to respond to asset changes

To respond to events in real time, we recommend that you use Cloud Asset Inventory to monitor asset changes. In this custom solution, an asset feed is configured to trigger notifications to Pub/Sub about changes to resources in real time, and then Cloud Run functions runs custom code to enforce your own business logic based on whether the change should be allowed.

The blueprint has an example of this custom governance solution that monitors for IAM changes that add highly sensitive roles including Organization Admin, Owner, and Editor. The following diagram describes this solution.

Automatically reverting an IAM policy change and sending a notification.

The previous diagram shows these concepts:

Changes are made to an allow policy.

The Cloud Asset Inventory feed sends a real-time notification about the allow policy change to Pub/Sub.

Pub/Sub triggers a function.

Cloud Run functions runs custom code to enforce your policy. The example function has logic to assess if the change has added the Organization Admin, Owner, or Editor roles to an allow policy. If so, the function creates a custom security finding and sends it to Security Command Center.

Optionally, you can use this model to automate remediation efforts. Write additional business logic in Cloud Run functions to automatically take action on the finding, such as reverting the allow policy to its previous state.

In addition, you can extend the infrastructure and logic used by this sample solution to add custom responses to other events that are important to your business.

What's next

Read about preventative controls (next document in this series).

Preventative controls for acceptable resource configurations

Last reviewed 2025-05-15 UTC

We recommend that you define policy constraints that enforce acceptable resource configurations and prevent risky configurations. The blueprint uses a combination of organization policy constraints and infrastructure-as-code (IaC) validation in your pipeline. These controls prevent the creation of resources that don't meet your policy guidelines. Enforcing these controls early in the design and build of your workloads helps you to avoid remediation work later.

Organization policy constraints

The Organization Policy service enforces constraints to ensure that certain resource configurations can't be created in your Google Cloud organization, even by someone with a sufficiently privileged IAM role.

The blueprint enforces policies at the organization node so that these controls are inherited by all folders and projects within the organization. This bundle of policies is designed to prevent certain high-risk configurations, such as exposing a VM to the public internet or granting public access to storage buckets, unless you deliberately allow an exception to the policy.

The following table introduces the organization policy constraints that are implemented in the blueprint:

Organization policy constraint	Description
compute.disableNestedVirtualization	Nested virtualization on Compute Engine VM evade monitoring and other security tools fo VMs if poorly configured. This constraint pre the creation of nested virtualization.
compute.disableSerialPortAccess	IAM roles like compute.instanceAdmin allow privileged access to an instance's serial port SSH keys. If the SSH key is exposed, an attac could access the serial port and bypass netw and firewall controls. This constraint prevent serial port access.
compute.disableVpcExternalIpv6	External IPv6 subnets can be exposed to unauthorized internet access if they are poo configured. This constraint prevents the crea external IPv6 subnets.

Organization policy constraint	Description
compute.requireOsLogin	The default behavior of setting SSH keys in metadata can allow unauthorized remote ac to VMs if keys are exposed. This constraint enforces the use of OS Login instead of meta based SSH keys.
compute.restrictProtocolForwardingCreationForTypes	VM protocol forwarding for external IP addre can lead to unauthorized internet egress if forwarding is poorly configured. This constra allows VM protocol forwarding for internal addresses only.
compute.restrictXpnProjectLienRemoval	Deleting a Shared VPC host project can be disruptive to all the service projects that use networking resources. This constraint preven accidental or malicious deletion of the Share host projects by preventing the removal of the project lien on these projects.
compute.setNewProjectDefaultToZonalDNSOnly	A legacy setting for global (project-wide) inte DNS is not recommended because it reduce service availability. This constraint prevents use of the legacy setting.
compute.skipDefaultNetworkCreation	A default VPC network and overly permissive default VPC firewall rules are created in ever project that enables the Compute Engine AP constraint skips the creation of the default n and default VPC firewall rules.
compute.vmExternalIpAccess	By default, a VM is created with an external I address that can lead to unauthorized intern access. This constraint configures an empty allowlist of external IP addresses that the VM use and denies all others.

Organization policy constraint	Description
essentialcontacts.allowedContactDomains	By default, Essential Contacts can be config to send notifications about your domain to a other domain. This constraint enforces that c email addresses in approved domains can b as recipients for Essential Contacts.
iam.allowedPolicyMemberDomains	By default, allow policies can be granted to a Google Account, including unmanaged acco and accounts belonging to external organiza This constraint ensures that allow policies in organization can only be granted to managed accounts from your own domain. Optionally, can allow additional domains.
iam.automaticIamGrantsForDefaultServiceAccounts	By default, default service accounts are automatically granted overly permissive role constraint prevents the automatic IAM role g to default service accounts.
iam.disableServiceAccountKeyCreation	Service account keys are a high-risk persiste credential, and in most cases a more secure alternative to service account keys can be us This constraint prevents the creation of servi account keys.
iam.disableServiceAccountKeyUpload	Uploading service account key material can increase risk if key material is exposed. This constraint prevents the uploading of service account keys.
sql.restrictAuthorizedNetworks	Cloud SQL instances can be exposed to unauthenticated internet access if the instan are configured to use authorized networks w a Cloud SQL Auth Proxy. This policy prevents configuration of authorized networks for data

Organization policy constraint	Description
	access and forces the use of the Cloud SQL Proxy instead.
sql.restrictPublicIp	Cloud SQL instances can be exposed to unauthenticated internet access if the instar are created with public IP addresses. This constraint prevents public IP addresses on C SQL instances.
storage.uniformBucketLevelAccess	By default, objects in Cloud Storage can be accessed through legacy Access Control Lis (ACLs) instead of IAM, which can lead to inconsistent access controls and accidental exposure if misconfigured. Legacy ACL acce not affected by the iam.allowedPolicyMemberDomains con This constraint enforces that access can onl configured through IAM uniform bucket-leve access, not legacy ACLs.
storage.publicAccessPrevention	Cloud Storage buckets can be exposed to unauthenticated internet access if misconfi This constraint prevents ACLs and IAM permissions that grant access to allUsers and allAuthenticatedUsers.

These policies are a starting point that we recommend for most customers and most scenarios, but you might need to modify organization policy constraints to accommodate certain workload types. For example, a workload that uses a Cloud Storage bucket as the backend for Cloud CDN to host public resources is blocked by storage.publicAccessPrevention, or a public-facing Cloud Run app that doesn't require authentication is blocked by iam.allowedPolicyMemberDomains. In these cases, modify the organization policy at the folder or project level to allow a narrow exception. You can also conditionally add constraints to organization policy by defining a tag that grants an exception or enforcement for policy, then applying the tag to projects and folders.

For additional constraints, see available constraints and custom constraints.

Pre-deployment validation of infrastructure-as-code

The blueprint uses a GitOps approach to manage infrastructure, meaning that all infrastructure changes are implemented through version-controlled infrastructure-as-code (IaC) and can be validated before deploying.

The policies enforced in the blueprint define acceptable resource configurations that can be deployed by your pipeline. If code that is submitted to your GitHub repository does not pass the policy checks, no resources are deployed.

For information on how pipelines are used and how controls are enforced through CI/CD automation, see deployment methodology.

What's next

- Read about deployment methodology (next document in this series)

Deployment methodology

Last reviewed 2025-05-15 UTC

We recommend that you use declarative infrastructure to deploy your foundation in a consistent and controllable manner. This approach helps enable consistent governance by enforcing policy controls about acceptable resource configurations into your pipelines. The blueprint is deployed using a GitOps flow, with Terraform used to define infrastructure as code (IaC), a Git repository for version control and approval of code, and Cloud Build for CI/CD automation in the deployment pipeline. For an introduction to this concept, see managing infrastructure as code with Terraform, Cloud Build, and GitOps.

The following sections describe how the deployment pipeline is used to manage resources in your organization.

Pipeline layers

To separate the teams and technology stack that are responsible for managing different layers of your environment, we recommend a model that uses different pipelines and different personas that are responsible for each layer of the stack.

The following diagram introduces our recommended model for separating a foundation pipeline, infrastructure pipeline, and application pipeline.

The diagram introduces the pipeline layers in this model:

- The *foundation pipeline* deploys the foundation resources that are used across the platform. We recommend that a single central team is responsible for managing the foundation resources that are consumed by multiple business units and workloads.

- The *infrastructure pipeline* deploys projects and infrastructure that are used by workloads, such as VM instances or databases. The blueprint sets up a separate infrastructure pipeline for each business unit, or you might prefer a single infrastructure pipeline used by multiple teams.
- The *application pipeline* deploys the artifacts for each workload, such as containers or images. You might have many different application teams with individual application pipelines.

The following sections introduce the usage of each pipeline layer.

The foundation pipeline

The foundation pipeline deploys the foundation resources. It also sets up the infrastructure pipeline that is used to deploy infrastructure used by workloads.

To create the foundation pipeline, you first clone or fork the terraform-example-foundation to your own Git repository. Follow the steps in the 0-bootstrap README file to configure your bootstrap folder and resources.

Stage	Description
0-bootstrap	Bootstraps a Google Cloud organization. This step also configures a CI/CD pipeline for the blueprint code in subsequent stages. • The CICD project contains the Cloud Build foundation pipeline for deploying resources. • The seed project includes the Cloud Storage buckets that contain the Terraform state of the foundation infrastructure and includes highly privileged service accounts that are used by the foundation pipeline to create resources. The Terraform state is protected through storage Object Versioning. When the CI/CD pipeline runs, it acts as the service accounts that are managed in the seed project.

After you create the foundation pipeline in the 0-bootstrap stage, the following stages deploy resources on the foundation pipeline. Review the README directions for each stage and implement each stage sequentially.

Stage	Description
1-org	Sets up top-level shared folders, projects for shared services, organization-level logging, and baseline security settings through organization policies.
2-environments	Sets up development, non-production, and production environments within the Google Cloud organization that you've created.
3-networks-dual-svpc or 3-networks-hub-and-spoke	Sets up shared VPCs in your chosen topology and the associated network resources.

The infrastructure pipeline

The infrastructure pipeline deploys the projects and infrastructure (for example, the VM instances and databases) that are used by workloads. The foundation pipeline deploys multiple infrastructure pipelines. This separation between the foundation pipeline and infrastructure pipeline allows for a separation between platform-wide resources and workload-specific resources.

The following diagram describes how the blueprint configures multiple infrastructure pipelines that are intended for use by separate teams.

The diagram describes the following key concepts:

- Each infrastructure pipeline is used to manage infrastructure resources independently of the foundation resources.
- Each business unit has its own infrastructure pipeline, managed in a dedicated project in the common folder.
- Each of the infrastructure pipelines has a service account with permission to deploy resources only to the projects that are associated with that business unit. This strategy creates a separation of duties between the privileged service accounts used for the foundation pipeline and those used by each infrastructure pipeline

This approach with multiple infrastructure pipelines is recommended when you have multiple entities inside your organization that have the skills and appetite to manage their infrastructure separately, particularly if they have different requirements such as the types of pipeline validation policy they want to enforce. Alternatively, you might prefer to have a single infrastructure pipeline managed by a single team with consistent validation policies.

In the terraform-example-foundation, stage 4 configures an infrastructure pipeline, and stage 5 demonstrates an example of using that pipeline to deploy infrastructure resources.

Stage	Description
4-projects	Sets up a folder structure, projects, and an infrastructure pipeline.
5-app-infra (optional)	Deploys workload projects with a Compute Engine instance using the infrastructure pipeline as an example.

The application pipeline

The application pipeline is responsible for deploying application artifacts for each individual workload, such as images or Kubernetes containers that run the business logic of your application. These artifacts are deployed to infrastructure resources that were deployed by your infrastructure pipeline.

The enterprise foundation blueprint sets up your foundation pipeline and infrastructure pipeline, but doesn't deploy an application pipeline. For an example application pipeline, see the enterprise application blueprint.

Automating your pipeline with Cloud Build

The blueprint uses Cloud Build to automate CI/CD processes. The following table describes the controls are built into the foundation pipeline and infrastructure pipeline that are deployed by the terraform-example-foundation repository. If you are developing your own pipelines using other CI/CD automation tools, we recommend that you apply similar controls.

Control	Description
Separate build configurations to	The blueprint uses two Cloud Build build configuration files for the entire pipeline, and each repository that is associated with a stage has two Cloud Build triggers that are associated with those build configuration files. When

Control	Description
validate code before deploying	code is pushed to a repository branch, the build configuration files are triggered to first run cloudbuild-tf-plan.yaml which validates your code with policy checks and Terraform plan against that branch, then cloudbuild-tf-apply.yaml runs terraform apply on the outcome of that plan.
Terraform policy checks	The blueprint includes a set of Open Policy Agent constraints that are enforced by the policy validation in Google Cloud CLI. These constraints define the acceptable resource configurations that can be deployed by your pipeline. If a build doesn't meet policy in the first build configuration, then the second build configuration doesn't deploy any resources. The policies enforced in the blueprint are forked from GoogleCloudPlatform/policy-library on GitHub. You can write additional policies for the library to enforce custom policies to meet your requirements.
Principle of least privilege	The foundation pipeline has a different service account for each stage with an allow policy that grants only the minimum IAM roles for that stage. Each Cloud Build trigger runs as the specific service account for that stage. Using different accounts helps mitigate the risk that modifying one repository could impact the resources that are managed by another repository. To understand the particular IAM roles applied to each service account, see the sa.tf Terraform code in the bootstrap stage.
Cloud Build private pools	The blueprint uses Cloud Build private pools. Private pools let you optionally enforce additional controls such as restricting access to public repositories or running Cloud Build inside a VPC Service Controls perimeter.
Cloud Build custom builders	The blueprint creates its own custom builder to run Terraform. For more information, see 0-bootstrap/Dockerfile. This control enforces that the pipeline consistently runs with a known set of libraries at pinned versions.
Deployment approval	Optionally, you can add a manual approval stage to Cloud Build. This approval adds an additional checkpoint after the build is triggered but before it runs so that a privileged user can manually approve the build.

Branching strategy

We recommend a persistent branch strategy for submitting code to your Git system and deploying resources through the foundation pipeline. The following diagram describes the persistent branch strategy.

The diagram describes three persistent branches in Git (development, non-production, and production) that reflect the corresponding Google Cloud environments. There are also multiple ephemeral feature branches that don't correspond to resources that are deployed in your Google Cloud environments.

We recommend that you enforce a pull request (PR) process into your Git system so that any code that is merged to a persistent branch has an approved PR.

To develop code with this persistent branch strategy, follow these high-level steps:

1. When you're developing new capabilities or working on a bug fix, create a new branch based off of the development branch. Use a naming convention for your branch that includes the type of change, a ticket number or other identifier, and a human-readable description, like feature/123456-org-policies.
2. When you complete the work in the feature branch, open a PR that targets the development branch.
3. When you submit the PR, the PR triggers the foundation pipeline to perform terraform plan and terraform validate to stage and verify the changes.
4. After you validate the changes to the code, merge the feature or bug fix into the development branch.
5. The merge process triggers the foundation pipeline to run terraform apply to deploy the latest changes in the development branch to the development environment.
6. Review the changes in the development environment using any manual reviews, functional tests, or end-to-end tests that are relevant to your use case. Then promote changes to the non-production environment by opening a PR that targets the non-production branch and merge your changes.
7. To deploy resources to the production environment, repeat the same process as step 6: review and validate the deployed resources, open a PR to the production branch, and merge.

What's next

- Read about operations best practices (next document in this series).

Operations best practices

Last reviewed 2025-05-15 UTC

This section introduces operations that you must consider as you deploy and operate additional workloads into your Google Cloud environment. This section isn't intended to be exhaustive of all operations in your cloud environment, but introduces decisions related to the architectural recommendations and resources deployed by the blueprint.

Update foundation resources

Although the blueprint provides an opinionated starting point for your foundation environment, your foundation requirements might grow over time. After your initial deployment, you might adjust configuration settings or build new shared services to be consumed by all workloads.

To modify foundation resources, we recommend that you make all changes through the foundation pipeline. Review the branching strategy for an introduction to the flow of writing code, merging it, and triggering the deployment pipelines.

Decide attributes for new workload projects

When creating new projects through the project factory module of the automation pipeline, you must configure various attributes. Your process to design and create projects for new workloads should include decisions for the following:

Which Google Cloud APIs to enable

Which Shared VPC to use, or whether to create a new VPC network

Which IAM roles to create for the initial project-service-account that is created by the pipeline

Which project labels to apply

The folder that the project is deployed to

Which billing account to use

Whether to add the project to a VPC Service Controls perimeter

Whether to configure a budget and billing alert threshold for the project

For a complete reference of the configurable attributes for each project, see the input variables for the project factory in the automation pipeline.

Manage permissions at scale

When you deploy workload projects on top of your foundation, you must consider how you will grant access to the intended developers and consumers of those projects. We recommend that you add users into a group that is managed by your existing identity provider, synchronize the groups with Cloud Identity, and then apply IAM roles to the groups. Always keep in mind the principle of least privilege.

We also recommend that you use IAM recommender to identify allow policies that grant over-privileged roles. Design a process to periodically review recommendations or automatically apply recommendations into your deployment pipelines.

Coordinate changes between the networking team and the application team

The network topologies that are deployed by the blueprint assume that you have a team responsible for managing network resources, and separate teams responsible for deploying workload infrastructure resources. As the workload teams deploy infrastructure, they must create firewall rules to allow the intended access paths between components of their workload, but they don't have permission to modify the network firewall policies themselves.

Plan how teams will work together to coordinate the changes to the centralized networking resources that are needed to deploy applications. For example, you might design a process where a workload team requests tags for their applications. The networking team then creates the tags and adds rules to the network firewall policy that allows traffic to flow between resources with the tags, and delegates the IAM roles to use the tags to the workload team.

Optimize your environment with the Active Assist portfolio

In addition to IAM recommender, Google Cloud provides the Active Assist portfolio of services to make recommendations about how to optimize your environment. For example, firewall insights or the unattended project recommender provide actionable recommendations that can help tighten your security posture.

Design a process to periodically review recommendations or automatically apply recommendations into your deployment pipelines. Decide which recommendations should be managed by a central team and which should be the responsibility of workload owners, and apply IAM roles to access the recommendations accordingly.

Grant exceptions to organization policies

The blueprint enforces a set of organization policy constraints that are recommended to most customers in most scenarios, but you might have legitimate use cases that require limited exceptions to the organization policies you enforce broadly.

For example, the blueprint enforces the iam.disableServiceAccountKeyCreation constraint. This constraint is an important security control because a leaked service account key can have a significant negative impact, and most scenarios should use more secure alternatives to service account keys to authenticate. However, there might be use cases that can only authenticate with a service account key, such as an on-premises server that requires access to Google Cloud services and cannot use Workload Identity Federation. In this scenario, you might decide to allow an exception to the policy, so long as additional compensating controls like best practices for managing service account keys are enforced.

Therefore, you should design a process for workloads to request an exception to policies, and ensure that the decision makers who are responsible for granting exceptions have the technical knowledge to validate the use case and consult on whether additional controls must be in place to compensate. When you grant an exception to a workload, modify the organization policy constraint as narrowly as possible. You can also conditionally add constraints to an organization policy by defining a tag that grants an exception or enforcement for policy, then applying the tag to projects and folders.

Protect your resources with VPC Service Controls

The blueprint helps prepare your environment for VPC Service Controls by enforcing prerequisite configurations like the restricted VIP. However, the blueprint initially deploys VPC Service Controls only in dry-run mode because cutting over to enforced mode can be a disruptive process that requires planning unique to your organization.

A perimeter denies access to restricted Google Cloud services from traffic that originates outside the perimeter, which includes the console, developer workstations, and the foundation pipeline used to deploy resources. If you use VPC Service Controls, you must design exceptions to the perimeter that allow the access paths that you intend.

A VPC Service Controls perimeter is intended for exfiltration controls between your Google Cloud organization and external sources. The perimeter isn't intended to replace or duplicate allow policies for granular access control to individual projects or resources. When you design and architect a perimeter, we recommend using a common unified perimeter for lower management overhead.

If you must design multiple perimeters to granularly control service traffic within your Google Cloud organization, we recommend that you clearly define the threats that are addressed by a more complex perimeter structure and the access paths between perimeters that are needed for intended operations.

To adopt VPC Service Controls, evaluate the following:

Which of your use cases require VPC Service Controls.

Whether the required Google Cloud services support VPC Service Controls.

How to configure breakglass access to modify the perimeter in case it disrupts your automation pipelines.

How to use best practices for enabling VPC Service Controls to design and implement your perimeter.

After you change the perimeter from dry-run to enforced mode, we recommend that you design a process to consistently add new projects to the correct perimeter, and a process to design exceptions when developers have a new use case that is denied by your current perimeter configuration.

Test organization-wide changes in a separate organization

We recommend that you never deploy changes to production without testing. For workload resources, this approach is facilitated by separate environments for development, non-production, and production. However, some resources at the organization don't have separate environments to facilitate testing.

For changes at the organization-level, or other changes that can affect production environments like the configuration between your identity provider and Cloud Identity, consider creating a separate organization for test purposes.

Control remote access to virtual machines

Because we recommend that you deploy immutable infrastructure through the foundation pipeline, infrastructure pipeline, and application pipeline, we also recommend that you only grant developers direct access to a virtual machine through SSH or RDP for limited or exceptional use cases.

For scenarios that require remote access, we recommend that you manage user access using OS Login where possible. This approach uses managed Google Cloud services to enforce access control, account lifecycle management, two-step verification, and audit logging. Alternatively, if you must allow access through SSH keys in metadata or RDP credentials, it is your responsibility to manage the credential lifecycle and store credentials securely outside of Google Cloud.

In any scenario, a user with SSH or RDP access to a VM can be a privilege escalation risk, so you should design your access model with this in mind. The user can run code on that VM with the privileges of the associated service account or query the metadata server to view the access token that is used to authenticate API requests. This access can then be a privilege escalation if you didn't deliberately intend for the user to operate with the privileges of the service account.

Mitigate overspending by planning budget alerts

The blueprint implements best practices introduced in the Google Cloud Well-Architected Framework: Cost Optimization for managing cost, including the following:

Use a single billing account across all projects in the enterprise foundation.

Assign each project a billingcode metadata label that is used to allocate cost between cost centers.

Set budgets and alert thresholds.

It's your responsibility to plan budgets and configure billing alerts. The blueprint creates budget alerts for workload projects when the forecasted spending is on track to reach 120% of the budget. This approach lets a central team identify and mitigate incidents of significant overspending. Significant unexpected increases in spending without a clear cause can be an indicator of a security incident and should be investigated from the perspectives of both cost control and security.

Note: Budget alerts cover a different type of notification than the Billing category of Essential Contacts. Budget alerts are related to the consumption of budgets that you define for each project. Billing notifications from Essential Contacts are related to pricing updates, errors, and credits.

Depending on your use case, you might set a budget that is based on the cost of an entire environment folder, or all projects related to a certain cost center, instead of setting granular budgets for each project. We also recommend that you delegate budget and alert setting to workload owners who might set more granular alerting threshold for their day-to-day monitoring.

For guidance on cost optimization, see Optimize costs with FinOps hub.

Allocate costs between internal cost centers

The console lets you view your billing reports to view and forecast cost in multiple dimensions. In addition to the prebuilt reports, we recommend that you export billing data to a BigQuery dataset in the prj-c-billing-export project. The exported billing records allow you to allocate cost on custom dimensions, such as your internal cost centers, based on project label metadata like billingcode.

The following SQL query is a sample query to understand costs for all projects that are grouped by the billingcode project label.

```
#standardSQL
SELECT
  (SELECT value from UNNEST(labels) where key = 'billingcode') AS costcenter,
  service.description AS description,
  SUM(cost) AS charges,
  SUM((SELECT SUM(amount) FROM UNNEST(credits))) AS credits
FROM PROJECT_ID.DATASET_ID.TABLE_NAME
GROUP BY costcenter, description
ORDER BY costcenter ASC, description ASC
```

To set up this export, see export Cloud Billing data to BigQuery.

If you require internal accounting or chargeback between cost centers, it's your responsibility to incorporate the data that is obtained from this query into your internal processes.

Ingest findings from detective controls into your existing SIEM

Although the foundation resources help you configure aggregated destinations for audit logs and security findings, it is your responsibility to decide how to consume and use these signals.

If you have a requirement to aggregate logs across all cloud and on-premise environments into an existing SIEM, decide how to ingest logs from the prj-c-logging project and findings from Security Command Center into your existing tools and processes. You might create a single export for all logs and findings if a single team is responsible for monitoring security across your entire environment, or you might create multiple exports filtered to the set of logs and findings needed for multiple teams with different responsibilities.

Alternatively, if log volume and cost are prohibitive, you might avoid duplication by retaining Google Cloud logs and findings only in Google Cloud. In this scenario, ensure that your existing teams have the right access and training to work with logs and findings directly in Google Cloud.

For audit logs, design log views to grant access to a subset of logs in your centralized logs bucket to individual teams, instead of duplicating logs to multiple buckets which increases log storage cost.

For security findings, grant folder-level and project-level roles for Security Command Center to let teams view and manage security findings just for the projects for which they are responsible, directly in the console.

Continuously develop your controls library

The blueprint starts with a baseline of controls to detect and prevent threats. We recommend that you review these controls and add additional controls based on your requirements. The following table summarizes the mechanisms to enforce governance policies and how to extend these for your additional requirements:

Policy controls enforced by the blueprint	Guidance to extend these controls
Security Command Center detects vulnerabilities and threats from multiple security sources.	Define custom modules for Security Health Analytics and custom modules for Event Threat Detection.
The Organization Policy service enforces a recommended set of organization policy constraints on Google Cloud services.	Enforce additional constraints from the premade list of available constraints or create custom constraints.
Open Policy Agent (OPA) policy validates code in the foundation pipeline for acceptable configurations before deployment.	

Develop additional constraints based on the guidance at GoogleCloudPlatform/policy-library.

Alerting on log-based metrics and performance metrics configures log-based metrics to alert on changes to IAM policies and configurations of some sensitive resources.

Design additional log-based metrics and alerting policies for log events that you expect shouldn't occur in your environment.

A custom solution for automated log analysis regularly queries logs for suspicious activity and creates Security Command Center findings.

Write additional queries to create findings for security events that you want to monitor, using security log analytics as a reference.

A custom solution to respond to asset changes creates Security Command Center findings and can automate remediation actions.

Create additional Cloud Asset Inventory feeds to monitor changes for particular asset types and write additional Cloud Run functions with custom logic to respond to policy violations.

These controls might evolve as your requirements and maturity on Google Cloud change.

Manage encryption keys with Cloud Key Management Service

Google Cloud provides default encryption at rest for all customer content, but also provides Cloud Key Management Service (Cloud KMS) to provide you additional control over your encryption keys for data at rest. We recommend that you evaluate whether the default encryption is sufficient, or whether you have a compliance requirement that you must use Cloud KMS to manage keys yourself. For more information, see decide how to meet compliance requirements for encryption at rest.

The blueprint provides a prj-c-kms project in the common folder and a prj-{env}-kms project in each environment folder for managing encryption keys centrally. This approach lets a central team audit and manage encryption keys that are used by resources in workload projects, in order to meet regulatory and compliance requirements.

Depending on your operational model, you might prefer a single centralized project instance of Cloud KMS under the control of a single team, you might prefer to manage encryption keys separately in each environment, or you might prefer multiple distributed instances so that accountability for encryption keys can be delegated to the appropriate teams. Modify the Terraform code sample as needed to fit your operational model.

Optionally, you can enforce customer-managed encryption keys (CMEK) organization policies to enforce that certain resource types always require a CMEK key and that only CMEK keys from an allowlist of trusted projects can be used.

Store and audit application credentials with Secret Manager

We recommend that you never commit sensitive secrets (such as API keys, passwords, and private certificates) to source code repositories. Instead, commit the secret to Secret Manager and grant the Secret Manager Secret Accessor IAM role to the user or service account that needs to access the secret. We recommend that you grant the IAM role to an individual secret, not to all secrets in the project.

When possible, you should generate production secrets automatically within the CI/CD pipelines and keep them inaccessible to human users except in breakglass situations. In this scenario, ensure that you don't grant IAM roles to view these secrets to any users or groups.

The blueprint provides a single prj-c-secrets project in the common folder and a prj-{env}-secrets project in each environment folder for managing secrets centrally. This approach lets a central team audit and manage secrets used by applications in order to meet regulatory and compliance requirements.

Depending on your operational model, you might prefer a single centralized instance of Secret Manager under the control of a single team, or you might prefer to manage secrets separately in each environment, or you might prefer multiple distributed instances of Secret Manager so that each workload team can manage their own secrets. Modify the Terraform code sample as needed to fit your operational model.

Plan breakglass access to highly privileged accounts

Although we recommend that changes to foundation resources are managed through version-controlled IaC that is deployed by the foundation pipeline, you might have exceptional or emergency scenarios that require privileged access to modify your environment directly. We recommend that you plan for breakglass accounts (sometimes called firecall or emergency accounts) that have highly privileged access to your environment in case of an emergency or when the automation processes break down.

The following table describes some example purposes of breakglass accounts.

Breakglass purpose	Description
Super admin	Emergency access to the Super admin role used with Cloud Identity, to, for example, fix issues that are related to identity federation or multi-factor authentication (MFA).
Organization administrator	Emergency access to the Organization Administrator role, which can then grant access to any other IAM role in the organization.
Foundation pipeline administrator	Emergency access to modify the resources in your CICD project on Google Cloud and external Git repository in case the automation of the foundation pipeline breaks down.

Operations or SRE

An operations or SRE team needs privileged access to respond to outages or incidents. This can include tasks like restarting VMs or restoring data.

Your mechanism to permit breakglass access depends on the existing tools and procedures you have in place, but a few example mechanisms include the following:

Use your existing tools for privileged access management to temporarily add a user to a group that is predefined with highly-privileged IAM roles or use the credentials of a highly-privileged account.

Pre-provision accounts intended only for administrator usage. For example, developer Dana might have an identity dana@example.com for daily use and admin-dana@example.com for breakglass access.

Use an application like just-in-time privileged access that allows a developer to self-escalate to more privileged roles.

Regardless of the mechanism you use, consider how you operationally address the following questions:

How do you design the scope and granularity of breakglass access? For example, you might design a different breakglass mechanism for different business units to ensure that they cannot disrupt each other.

How does your mechanism prevent abuse? Do you require approvals? For example, you might have split operations where one person holds credentials and one person holds the MFA token.

How do you audit and alert on breakglass access? For example, you might configure a custom Event Threat Detection module to create a security finding when a predefined breakglass account is used.

How do you remove the breakglass access and resume normal operations after the incident is over?

For common privilege escalation tasks and rolling back changes, we recommend designing automated workflows where a user can perform the operation without requiring privilege escalation for their user identity. This approach can help reduce human error and improve security.

For systems that require regular intervention, automating the fix might be the best solution. Google encourages customers to adopt a zero-touch production approach to make all production changes using automation, safe proxies, or audited breakglass. Google provides the SRE books for customers who are looking to adopt Google's SRE approach.

What's next

Read Deploy the blueprint (next document in this series).

Deploy the blueprint

Last reviewed 2025-05-15 UTC

This section describes the process that you can use to deploy the blueprint, its naming conventions, and alternatives to blueprint recommendations.

Bringing it all together

To deploy your own enterprise foundation in alignment with the best practices and recommendations from this blueprint, follow the high-level tasks summarized in this section. Deployment requires a combination of prerequisite setup steps, automated deployment through the terraform-example-foundation on GitHub, and additional steps that must be configured manually after the initial foundation deployment is complete.

Process	Steps
Prerequisites before deploying the foundation pipeline resources	Complete the following steps before you deploy the foundation pipeline: • Create a Cloud Identity account and verify domain ownership. • Apply for an invoiced billing account with your Google Cloud sales team or create a self-service billing account. • Enforce security best practices for administrator accounts. • Verify and reconcile issues with consumer user accounts.

Process	Steps
	• Configure your external identity provider as source of truth for synchronizing user accounts and SSO. • Provision the groups for access control that are required to run the blueprint. • Determine the network topology that you will use. • Decide your source code management tool. The instructions for terraform-example-foundation are written to a Git repository that is hosted in Cloud Source Repositories. • Decide your CI/CD automation tools. The terraform-example-foundation provides different sets of directions for different automation tools. To connect to an an existing on-premises environment, prepare the following: • Plan your IP address allocation based on the number and size of ranges that are required by the blueprint. • Order your Dedicated Interconnect connections.
Steps to deploy the terraform-example-foundation from GitHub	Follow the README directions for each stage to deploy the terraform-example-foundation from GitHub: • Stage 0-bootstrap to create a foundation pipeline. If using a self-service billing account, you must request additional project quota before proceeding to the next stage. • Stage 1-org to configure organization-level resources. • Stage 2-environments to create environments. • Stage either 3-networks-dual-svpc or 3-networks-hub-and-spoke to create networking resources in your preferred topology. • Stage 4-projects to create an infrastructure pipeline. • Optionally, stage 5-app-infra for sample usage of the infrastructure pipeline.

Process	Steps
Additional steps after IaC deployment	After you deploy the Terraform code, complete the following: • Complete the on-premises configuration changes. • Activate Security Command Center Premium. • Export Cloud Billing data to BigQuery. • Sign up for a Cloud Customer Care plan. • Enable Access Transparency logs. • Share data from Cloud Identity with Google Cloud. • Apply the administrative controls for Cloud Identity which aren't automated by the IaC deployment. • Assess which of the additional administrative controls for customers with sensitive workloads are appropriate for your use case. • Review operation best practices and plan how to connect your existing operations and capabilities to the foundation resources.

Additional administrative controls for customers with sensitive workloads

Google Cloud provides additional administrative controls that can help your security and compliance requirements. However, some controls involve additional cost or operational trade-offs that might not be appropriate for every customer. These controls also require customized inputs for your specific requirements that can't be fully automated in the blueprint with a default value for all customers.

This section introduces security controls that you apply centrally to your foundation. This section isn't intended to be exhaustive of all the security controls that you can apply to specific workloads. For more information on Google's security products and solutions, see Google Cloud security best practices center.

Evaluate whether the following controls are appropriate for your foundation based on your compliance requirements, risk appetite, and sensitivity of data.

Control	Description
Protect your resources with VPC Service Controls	VPC Service Controls lets you define security policies that prevent access to Google-managed services outside of a trusted perimeter, block access to data from untrusted locations, and mitigate data exfiltration risks. However, VPC Service Controls can cause existing services to break until you define exceptions to allow intended access patterns. Evaluate whether the value of mitigating exfiltration risks justifies the increased complexity and operational overhead of adopting VPC Service Controls. The blueprint prepares prerequisite network controls and a dry-run VPC Service Controls perimeter, but the perimeter isn't enforced until you take additional steps.
Restrict resource locations	You might have regulatory requirements that cloud resources must only be deployed in approved geographical locations. This organization policy constraint enforces that resources can only be deployed in the list of locations you define.
Enable Assured Workloads	Assured Workloads provides additional compliance controls that help you meet specific regulatory regimes. The blueprint provides optional variables in the deployment pipeline for enablement.
Enable data access logs	You might have a requirement to log all access to certain sensitive data or resources. Evaluate where your workloads handle sensitive data that requires data access logs, and enable the logs for each service and environment working with sensitive data.
Enable Access Approval	Access Approval ensures that Cloud Customer Care and engineering require your explicit approval whenever they need to access your customer content. Evaluate the operational process required to review Access Approval requests to mitigate possible delays in resolving support incidents.
Enable Key Access Justifications	Key Access Justifications lets you programmatically control whether Google can access your encryption keys, including for automated operations and for Customer Care to access your customer content.

Control	Description
	Evaluate the cost and operational overhead associated with Key Access Justifications as well as its dependency on Cloud External Key Manager (Cloud EKM).
Disable Cloud Shell	Cloud Shell is an online development environment. This shell is hosted on a Google-managed server outside of your environment, and thus it isn't subject to the controls that you might have implemented on your own developer workstations. If you want to strictly control which workstations a developer can use to access cloud resources, disable Cloud Shell. You might also evaluate Cloud Workstations for a configurable workstation option in your own environment.
Restrict access to the Google Cloud console	Google Cloud lets you restrict access to the Google Cloud console based on access level attributes like group membership, trusted IP address ranges, and device verification. Some attributes require an additional subscription to Chrome Enterprise Premium. Evaluate the access patterns that you trust for user access to web-based applications such as the console as part of a larger zero trust deployment.

Naming conventions

We recommend that you have a standardized naming convention for your Google Cloud resources. The following table describes recommended conventions for resource names in the blueprint.

Resource	Naming convention
Folder	fldr-*environment* *environment* is a description of the folder-level resources within the Google Cloud organization. For example, bootstrap, common, production, nonproduction, development, or network. For example: fldr-production

Resource	Naming convention
Project ID	prj-*environmentcode-description-randomid* • *environmentcode* is a short form of the environment field (one of b, c, p, n, d, or net). Shared VPC host projects use the environmentcode of the associated environment. Projects for networking resources that are shared across environments, like the interconnect project, use the net environment code. • *description* is additional information about the project. You can use short, human-readable abbreviations. • *randomid* is a randomized suffix to prevent collisions for resource names that must be globally unique and to mitigate against attackers guessing resource names. The blueprint automatically adds a random four-character alphanumeric identifier. For example: prj-c-logging-a1b2
VPC network	vpc-*environmentcode-vpctype* • *environmentcode* is a short form of the environment field (one of b, c, p, n, d, or net). • *vpctype* is one of svpc, float, or peer. For example: vpc-p-svpc
Subnet	sn-*environmentcode-vpctype-region*{-*description*} • *environmentcode* is a short form of the environment field (one of b, c, p, n, d, or net). • *vpctype* is one of svpc, float, or peer. • *region* is any valid Google Cloud region that the resource is located in. We recommend removing hyphens and using an abbreviated form of some regions and directions to avoid hitting character limits. For example, au (Australia), na (North America), sa (South America), eu (Europe), se (southeast), or ne (northeast). • *description* is additional information about the subnet. You can use short, human-readable abbreviations. For example: sn-p-svpc-uswest1

Resource	Naming convention
Firewall policies	fw-*firewalltype-scope-environmentcode*{*-description*} • *firewalltype* is hierarchical or network. • *scope* is global or the Google Cloud region that the resource is located in. We recommend removing hyphens and using an abbreviated form of some regions and directions to avoid reaching character limits. For example, au (Australia), na (North America), sa (South America), eu (Europe), se (southeast), or ne (northeast). • *environmentcode* is a short form of the environment field (one of b, c, p, n, d, or net) that owns the policy resource. • *description* is additional information about the hierarchical firewall policy. You can use short, human-readable abbreviations. For example: fw-hierarchical-global-c-01 fw-network-uswest1-p-svpc
Cloud Router	cr-*environmentcode-vpctype-region*{*-description*} • *environmentcode* is a short form of the environment field (one of b, c, p, n, d, or net). • *vpctype* is one of svpc, float, or peer. • *region* is any valid Google Cloud region that the resource is located in. We recommend removing hyphens and using an abbreviated form of some regions and directions to avoid reaching character limits. For example, au (Australia), na (North America), sa (South America), eu (Europe), se (southeast), or ne (northeast). • *description* is additional information about the Cloud Router. You can use short, human-readable abbreviations. For example: cr-p-svpc-useast1-cr1
Cloud Interconnect connection	ic-*dc-colo*

Resource	Naming convention
	<ul><li>*dc* is the name of your data center to which a Cloud Interconnect is connected.</li><li>*colo* is the colocation facility name that the Cloud Interconnect from the on-premises data center is peered with.</li></ul>For example: ic-mydatacenter-lgazone1
Cloud Interconnect VLAN attachment	vl-*dc-colo-environmentcode-vpctype-region*{*-description}*<ul><li>*dc* is the name of your data center to which a Cloud Interconnect is connected.</li><li>*colo* is the colocation facility name that the Cloud Interconnect from the on-premises data center is peered with.</li><li>*environmentcode* is a short form of the environment field (one of b, c, p, n, d, or net).</li><li>*vpctype* is one of svpc, float, or peer.</li><li>*region* is any valid Google Cloud region that the resource is located in. We recommend removing hyphens and using an abbreviated form of some regions and directions to avoid reaching character limits. For example, au (Australia), na (North America), sa (South America), eu (Europe), se (southeast), or ne (northeast).</li><li>*description* is additional information about the VLAN. You can use short, human-readable abbreviations.</li></ul>For example: vl-mydatacenter-lgazone1-p-svpc-useast1-cr1
Group	grp-gcp-*description*@example.com Where *description* is additional information about the group. You can use short, human-readable abbreviations. For example: grp-gcp-billingadmin@example.com
Custom role	rl-*description* Where *description* is additional information about the role. You can use short, human-readable abbreviations.

Resource	Naming convention
	For example: rl-customcomputeadmin
Service account	sa-*description*@*projectid*.iam.gserviceaccount.com Where: • *description* is additional information about the service account. You can use short, human-readable abbreviations. • *projectid* is the globally unique project identifier. For example: sa-terraform-net@prj-b-seed-a1b2.iam.gserviceaccount.com
Storage bucket	bkt-*projectid-description* Where: • *projectid* is the globally unique project identifier. • *description* is additional information about the storage bucket. You can use short, human-readable abbreviations. For example: bkt-prj-c-infra-pipeline-a1b2-app-artifacts

Alternatives to default recommendations

The best practices that are recommended in the blueprint might not work for every customer. You can customize any of the recommendations to meet your specific requirements. The following table introduces some of the common variations that you might require based on your existing technology stack and ways of working.

Decision area	Possible alternatives
Organization: The blueprint uses a single organization as the root node for all resources.	Decide a resource hierarchy for your Google Cloud landing zone introduces scenarios in which you might prefer multiple organizations, such as the following: • Your organization includes sub-companies that are likely to be sold in

Decision area	Possible alternatives
	the future or that run as completely separate entities. • You want to experiment in a sandbox environment with no connectivity to your existing organization.
Folder structure: The blueprint has a folder structure that divides workloads into production, non-production and development folders at the top layer.	Decide a resource hierarchy for your Google Cloud landing zone introduces other approaches for structuring folders based on how you want to manage resources and inherit policies, such as: • Folders based on application environments • Folders based on regional entities or subsidiaries • Folders based on accountability framework
Organization policies: The blueprint enforces all organization policy constraints at the organization node.	You might have different security policies or ways of working for different parts of the business. In this scenario, enforce organization policy constraints at a lower node in the resource hierarchy. Review the complete list of organization policy constraints that help meet your requirements.
Deployment pipeline tooling: The blueprint uses Cloud Build to run the automation pipeline.	You might prefer other products for your deployment pipeline, such as Terraform Enterprise, GitLab Runners, GitHub Actions, or Jenkins. The blueprint includes alternative directions for each product.

Decision area	Possible alternatives
Code repository for deployment: The blueprint uses Cloud Source Repositories as the managed private Git repository.	Use your preferred version control system for managing code repositories, such as GitLab, GitHub, or Bitbucket. If you use a private repository that is hosted in your on-premises environment, configure a private network path from your repository to your Google Cloud environment.
Identity provider: The blueprint assumes an on-premises Active Directory and federates identities to Cloud Identity using Google Cloud Directory Sync.	If you already use Google Workspace, you can use the Google identities that are already managed in Google Workspace. If you don't have an existing identity provider, you might create and manage user identities directly in Cloud Identity. If you have an existing identity provider, such as Okta, Ping, or Azure Entra ID, you might manage user accounts in your existing identity provider and synchronize to Cloud Identity. If you have data sovereignty or compliance requirements that prevent you from using Cloud Identity, and if you don't require managed Google user identities for other Google services such as Google Ads or Google Marketing Platform, then you might prefer workforce identity federation. In this scenario, be aware of limitations with supported services.
Multiple regions: The blueprint deploys regional resources into two different Google Cloud regions to help enable workload design with high availability and disaster recovery requirements in mind.	If you have end users in more geographical locations, you might configure more Google Cloud regions to create resources closer to the end user with less latency. If you have data sovereignty constraints or your availability needs can be met in a single region,

Decision area	Possible alternatives
	you might configure only one Google Cloud region.
IP address allocation: The blueprint provides a set of IP address ranges.	You might need to change the specific IP address ranges that are used based on the IP address availability in your existing hybrid environment. If you modify the IP address ranges, use the blueprint as guidance for the number and size of ranges required, and review the valid IP address ranges for Google Cloud.
Hybrid networking: The blueprint uses Dedicated Interconnect across multiple physical sites and Google Cloud regions for maximum bandwidth and availability.	Depending on your requirements for cost, bandwidth, and reliability requirements, you might configure Partner Interconnect or Cloud VPN instead. If you need to start deploying resources with private connectivity before a Dedicated Interconnect can be completed, you might start with Cloud VPN and change to using Dedicated Interconnect later. If you don't have an existing on-premises environment, you might not need hybrid networking at all.
VPC Service Controls perimeter: The blueprint recommends a single perimeter which includes all service projects for your Shared VPC topology.	You might have a use case that requires multiple perimeters for an organization or you might decide not to use VPC Service Controls at all. For information, see decide how to mitigate data exfiltration through Google APIs.
Secret Manager: The blueprint deploys a project for using Secret Manager in the common folder for organization-wide secrets, and a project in each	If you have a single team who is responsible for managing and auditing sensitive secrets across the organization, you might prefer to use only a single project for managing access to secrets.

Decision area	Possible alternatives
environment folder for environment-specific secrets.	If you let workload teams manage their own secrets, you might not use a centralized project for managing access to secrets, and instead let teams use their own instances of Secret Manager in workload projects.
Cloud KMS: The blueprint deploys a project for using Cloud KMS in the common folder for organization-wide keys, and a project for each environment folder for keys in each environment.	If you have a single team who is responsible for managing and auditing encryption keys across the organization, you might prefer to use only a single project for managing access to keys. A centralized approach can help meet compliance requirements like PCI key custodians. If you let workload teams manage their own keys, you might not use a centralized project for managing access to keys, and instead let teams use their own instances of Cloud KMS in workload projects.
Aggregated log sinks: The blueprint configures a set of log sinks at the organization node so that a central security team can review audit logs from across the entire organization.	You might have different teams who are responsible for auditing different parts of the business, and these teams might require different logs to do their jobs. In this scenario, design multiple aggregated sinks at the appropriate folders and projects and create filters so that each team receives only the necessary logs, or design log views for granular access control to a common log bucket.
Granularity of infrastructure pipelines: The blueprint uses a model where each business unit has a separate infrastructure pipeline to manage their workload projects.	You might prefer a single infrastructure pipeline that is managed by a central team if you have a central team who is responsible for deploying all projects and infrastructure. This central team can accept pull requests from workload teams to review and approve before project creation, or the team can create the pull

Decision area	Possible alternatives
	request themselves in response to a ticketed system. You might prefer more granular pipelines if individual workload teams have the ability to customize their own pipelines and you want to design more granular privileged service accounts for the pipelines.
SIEM exports:The blueprint manages all security findings in Security Command Center.	Decide whether you will export security findings from Security Command Center to tools such as Google Security Operations or your existing SIEM, or whether teams will use the console to view and manage security findings. You might configure multiple exports with unique filters for different teams with different scopes and responsibilities.

What's next

- Implement the blueprint using the Terraform example foundation on GitHub.
- Learn more about best practice design principles with the Google Cloud Well-Architected Framework.
- Review the library of blueprints to help you accelerate the design and build of common enterprise workloads, including the following:
    - Import data from Google Cloud into a secured BigQuery data warehouse
    - Import data from an external network into a secured BigQuery data warehouse
    - Deploy a secured serverless architecture using Cloud Run functions
    - Deploy a secured serverless architecture using Cloud Run
- See related solutions to deploy on top of your foundation environment.