**Google Cloud Services Checklist**

IAM Checklist:

🗹Never Use personal account: Avoid using personal accounts for accessing and managing GCP resources to ensure proper segregation of personal and work-related activities.

🗹Enforce MFA for all users: Enable Multi-Factor Authentication (MFA) for all GCP user accounts to add an extra layer of security during authentication.

🗹Ensure Security Key enforcement for admin accounts: For admin accounts, enforce the use of security keys (e.g., FIDO2 keys) as a more secure authentication method.

🗹Disable Unused Service Accounts: Regularly review and disable any unused or unnecessary service accounts to minimize potential security risks.

🗹Delete Unused Service Account Keys: Remove any unused or expired keys associated with service accounts to reduce the risk of unauthorized access.

🗹Set session length for Cloud Console and CLI: Configure session length limits to ensure that inactive sessions automatically terminate, reducing the risk of unauthorized access.

🗹Limit the number of admin accounts: Minimize the number of admin-level accounts to reduce the attack surface and potential points of compromise.

🗹Grant roles to a group instead of individual users: Assign roles and permissions to groups rather than individual users for easier management and better access control.

🗹Avoid the use of Basic IAM Roles: Basic IAM roles grant broad permissions and should be avoided to ensure fine-grained control and adherence to the principle of least privilege.

🗹Make sure no project-level IAM member is assigned Service Account User or Service Account Token 🗹Creator roles: These roles provide excessive privileges and should be avoided at the project level.

🗹Use GCP IAM Recommender: Leverage GCP IAM Recommender, a machine learning-driven policy tool, to analyze IAM logs and receive recommendations for optimizing IAM policies and removing unnecessary permissions.

🗹Have corporate login credentials, not personal accounts: Use corporate accounts for managing GCP resources to ensure better control, auditability, and access management.

🗹Don’t associate any API keys to your GCP projects: Avoid using API keys for authentication and prefer more secure mechanisms like service accounts or OAuth.

🗹Enable Multi-Factor Authentication (MFA) for GCP user accounts: Enable MFA for GCP user accounts to enhance security and protect against unauthorized access.

🗹Limit GCP IAM primitive roles to within Google Cloud projects: Avoid assigning GCP IAM primitive roles at the organization level and instead assign them within specific Google Cloud projects to minimize the scope of their impact.

KMS Checklist:

🗹Enable key rotation: Regularly rotate your cryptographic keys to minimize the potential impact of a key compromise.

🗹Implement least privilege access: Grant only necessary permissions to users and roles for accessing and managing keys.

🗹Monitor and audit key usage: Track and analyze key usage logs to identify any suspicious activities or potential security breaches.

🗹Centralize Key Management: Use a centralized key management system like Google Cloud KMS to simplify key management, monitoring, and auditing.

🗹Implement strong protections for key material: Utilize secure storage mechanisms such as Hardware Security Modules (HSMs) or Google-managed key storage for safeguarding key material.

🗹Avoid storing keys in code or configuration files: Keys should not be hardcoded or stored directly in code or configuration files to minimize the risk of accidental exposure or unauthorized access.

Kubernetes Checklist:

🗹Dashboard Disabled: Disable the Kubernetes dashboard in production environments to minimize the risk of unauthorized access.

🗹Private Endpoint: Utilize private endpoints for accessing Kubernetes clusters, ensuring that they are not directly exposed to the public internet.

🗹Private Cluster Enabled: Enable private clusters to restrict access to Kubernetes API server from within the VPC network only.

🗹Pod Security Policy Enabled: Enforce pod security policies to define and enforce security standards for pods running in the cluster.

🗹Network Policy Enabled: Implement network policies to control network traffic between pods and enforce segmentation and isolation.

🗹Monitoring Enabled: Enable monitoring and observability solutions for your Kubernetes clusters to detect and respond to potential security threats or performance issues.

🗹Master Authorized Network: Restrict access to the Kubernetes master API endpoint by allowing connections only from specific authorized networks or IP ranges.

🗹Logging Enabled: Enable logging for your Kubernetes cluster to capture and analyze logs for security analysis and troubleshooting.

🗹Alias IP ranges Enabled: Utilize Alias IP ranges to enable direct communication between pods and other GCP services within the same VPC network.

🗹Default Service Account: Avoid using the default service account and create specific service accounts with limited permissions for each workload or application.

🗹Container Optimized OS Images enabled: Use optimized and hardened container images for running workloads in your Kubernetes clusters.

🗹Automatic Node upgrade: Enable automatic node upgrades to ensure that your cluster nodes are running the latest security patches and updates.

🗹Disabled Legacy Authentication Method: Disable legacy authentication methods like x509 client certificates and use more secure authentication mechanisms.

🗹Don’t Use Kubernetes Secrets: Avoid storing sensitive data directly in Kubernetes secrets and utilize external secret management systems like HashiCorp Vault or Azure Key Vault.

🗹Adopt VPC logging: Enable VPC flow logs and cluster-level logging to capture network traffic and Kubernetes-related logs for security analysis and monitoring.

🗹Limit Control Panel Exposure: Restrict access to the Kubernetes control panel (kubelet) to authorized users or IP ranges to minimize the risk of unauthorized access.

VPC Checklist:

🗹Define and implement a well-structured VPC architecture: Design your VPC architecture with appropriate network segmentation using subnets and IP ranges based on your requirements.

🗹Use separate VPCs for different environments or projects: Isolate resources and limit access by creating separate VPCs for different environments or projects.

🗹Use Google Cloud Identity and Access Management (IAM): Control access to your VPC and associated resources using IAM policies and roles.

🗹Follow the principle of least privilege: Grant only necessary permissions to users and roles based on the principle of least privilege.

🗹Use subnets and firewall rules for segmentation: Segment your VPC using subnets and configure firewall rules to control traffic flow and restrict access.

🗹Implement network-level firewalls: Use network-level firewalls to restrict inbound and outbound traffic based on specific rules and policies.

🗹Enable logging and monitoring: Enable VPC flow logs, firewall logs, and VPC flow logs for capturing and analyzing network activities. Set up monitoring and alerts for suspicious activities.

🗹Utilize bastion hosts or secure VPN connections: Use bastion hosts or establish secure VPN connections for remote access to your VPC resources.

🗹Regularly update and patch VPC components: Keep your VPC components (subnets, firewall rules, VPN gateways) up to date with the latest security patches and updates.

🗹Stay informed about new features and security updates: Regularly review Google Cloud documentation and security advisories to stay updated on VPC security considerations and best practices.

Storage Checklist:

🗹Storage Permission Logging: Enable permission logging for your storage buckets to track and monitor access to storage objects and identify any unauthorized access attempts.

🗹Bucket Logging: Enable logging for storage buckets to capture detailed information about object-level operations, access attempts, and changes.

🗹Storage Bucket All User Policy: Avoid granting public access to storage buckets and ensure that proper access controls are in place.

🗹Bucket Versioning: Enable versioning for your storage buckets to retain a history of object versions and protect against accidental deletion or modification.

Compute Service Checklist:

🗹OS login Enabled: Utilize OS Login feature to manage SSH access to virtual machine instances using IAM identities instead of SSH keys.

🗹IP Forwarding Disabled: Disable IP forwarding on virtual machine instances to prevent unauthorized traffic redirection.

🗹Instance Level SSH Only: Restrict SSH access to virtual machine instances at the instance level, allowing access only to authorized users or groups.

🗹VM Instance Least Privilege: Assign the least privilege level required for virtual machine instances by granting only necessary roles and permissions.

🗹CSEK Encryption Enabled: Encrypt virtual machine boot disks using Customer-Managed Encryption Keys (CSEK) for added security.

🗹Connect Serial Port Disabled: Disable serial port access to virtual machine instances unless it is explicitly required for troubleshooting.

🗹Cryptographic Keys: Properly manage and protect cryptographic keys used for encryption, authentication, and secure communication within virtual machine instances.

🗹Use Identity-Aware Proxy and OS Login: Leverage Identity-Aware Proxy (IAP) and OS Login for secure remote access to virtual machine instances.

🗹Avoid Using Default Service Account: Create dedicated service accounts with limited privileges instead of relying on the default service account.

🗹Avoid Public IP addresses: Minimize the use of public IP addresses for virtual machine instances to reduce the exposure to the public internet.

🗹Regularly Update and Patch VM instances: Keep virtual machine instances up to date with the latest security patches and updates to mitigate vulnerabilities.

🗹Use trusted and Secured/Hardened OS Images: Utilize trusted and hardened operating system images for virtual machine instances to minimize security risks.

🗹Regularly Deprecate Older OS Versions: Avoid using outdated or unsupported operating system versions on virtual machine instances and regularly update to newer versions.

🗹Avoid Storing Sensitive Data in VM Custom Metadata: Prevent storing sensitive information in VM custom metadata as it can be accessible to users with sufficient permissions.

GCP codebuild Checklist

🗹1.Utilize GCP Identity and Access Management (IAM) to control access to CodeBuild resources.

🗹2.Follow the principle of least privilege by granting only the necessary permissions to users and roles.

🗹3.Enable multi-factor authentication (MFA) for IAM users with access to CodeBuild.

🗹4.Store your source code in a secure and version-controlled repository, such as GCP CodeCommit or a trusted external source control system.

🗹5.Regularly update and patch the operating systems and software packages in your build environments.

🗹6.Use dedicated build environments for different projects to minimize the risk of code contamination.

🗹7.Restrict network access to build environments by placing them in private subnets and configuring security groups and network ACLs accordingly.

🗹8.Use separate build environments for different stages of the software development lifecycle (e.g., development, testing, production) to minimize the impact of any security breaches.

🗹9.Avoid hard-coding sensitive information, such as API keys or credentials, directly in build specifications or source code.

🗹10.Enable logging for CodeBuild builds and configure logs to be stored centrally in services like Logs or storage.

🗹11.Stay up to date with the latest features, patches, and updates provided by GCP for CodeBuild.

🗹12.Implement build-time checks and tests for security-related requirements, such as secure coding practices and adherence to security policies.

🗹13.Encrypt environment variables that contain sensitive data in transit and at rest.

🗹14.Utilize code analysis tools and security scanners to identify vulnerabilities or potential security issues in your code.

🗹15.Stay up to date with the latest features, patches, and updates provided by GCP for CodeBuild.

codepipeline Checklist

🗹1.Ensure IAM privilege are least privileged in CI/CD

🗹2.Ensure the secuirty of the CI/CD pipeline

🗹3.The production enviourment should not allow manual changes or SSH access

🗹4.Github should have proper policy in place for github actions

🗹5. Github action to create and approve pull request should be allowed(in this github itself approves the pull request)

🗹6.Every pull request should require approval before merging into the main branch.