State of California

California Department of Technology

Office of Information Security

Cloud Security Guide

SIMM 140

August 2020

REVISION HISTORY

REVISION	DATE OF RELEASE	OWNER	SUMMARY OF CHANGES
Initial Release	August 2020	Office of Information Security	New guide to supplement SIMM 5315-B, Cloud Security Standard.

TABLE OF CONTENTS

I. INTRODUCTION

This guide is intended to supplement Statewide Information Management Manual (SIMM) 5315-B, Cloud Security Standard, by providing supporting implementation guidance. While much of this guidance applies to all major cloud service providers (CSPs), you should investigate the details of the services you are using. In order to provide actionable guidance, this document provides examples and references best practices from CSPs in use by California State entities including: Amazon Web Services (AWS), Microsoft Azure, IBM GovCloud, Oracle Cloud, and Google Cloud Platform (GCP). This should not be construed as an endorsement of any vendor.

It is essential that security of information systems begin at inception and remain a vigilant effort through its lifecycle. The landscape of threats is more sophisticated than ever – fortunately, cloud computing puts modern defense tools in the hands of every developer. The utilization of cloud services does not exempt state agencies/entities from security provisions of the State Administrative Manual (SAM) and SIMM, but it does change the paradigm of responsibility.

The responsibility to secure cloud resources is shared between the customer and the cloud service provider, with varying responsibilities depending on the resource type and service model. Broadly, the cloud provider is responsible for “security of the cloud,” while the customer is responsible for “security in the cloud.” Even with this shared responsibility, state agencies/entities remain ultimately responsible for selecting and configuring cloud services commensurate with risk tolerance and regulatory requirements.

Modern cloud security may challenge some of your current practices, so cloud migration is a good opportunity to reevaluate them. One key change is that there is no longer a strict physical boundary between your system and other systems. Rather than focusing on boundary protections at a single layer, cloud security encourages security at all layers. For example, since you do not physically control network and storage devices in the cloud, you should always protect data by encrypting it in transit and at rest. Fortunately, cloud services often provide much easier and cost-effective ways to employ strong encryption.

Another recent shift in the security landscape is increased focus on detection and response. Protective controls remain essential, but sophisticated threats have demonstrated that despite our best efforts to protect systems, compromises happen. By establishing mechanisms for timely detection and response, we can mitigate the impact. This is another area that the cloud enables us – cloud services offer powerful ways to detect unexpected or unauthorized behavior and even automate response and remediation. However, these controls must be configured before an incident occurs – so security needs to be built into every layer of every application throughout its lifecycle, not as an afterthought.

II. CLOUD SECURITY CONTROLS

A. Identify

All cloud resources are logical from the consumer perspective, so you do not need to inventory hardware devices. Instead, ensure you have a complete picture of the services you have provisioned from each provider and outline their security needs, so you can develop controls accordingly.

1. Maintain an inventory of accounts with cloud service providers including root email addresses, account IDs, and points of contact.

In this context, accounts do not refer to user accounts, but rather the entity associated with the cloud services provisioned. Accounts therefore provide the strictest segmentation boundary, and you will create separate accounts over time for separate purposes. Strategies for organizing these multiple accounts are discussed further in Section B.1. It is easy for the number of accounts to grow out of control over time, so maintaining an up-to-date inventory is critical to ensuring visibility of your workloads and security exposure. A simple account inventory could look as follows:

Account Name	CSP	Account ID	Root Email	Description	Point of Contact
example-master	AWS	111111111111	aws.master@example.ca.gov	Organization account. Has no resources, just used to establish account hierarchy	engineering@example.ca.gov
example-security	AWS	222222222222	aws.security@example.ca.gov	Security for the department	security@example.ca.gov
example-shared-services	AWS	333333333333	aws.shared-services@example.ca.gov	Shared services for example-department	engineering@example.ca.gov
example-network	AWS	444444444444	aws.network@example.ca.gov	Networking for example-department	engineering@example.ca.gov
example-logging	AWS	555555555555	aws.logging@example.ca.gov	Logging for example-department	engineering@example.ca.gov
gb-prod	AWS	666666666666	grizzlybear.prod@example.ca.gov	Hosts the GrizzlyBear app	grizzlybear.engineering@example.ca.gov
gb-stage	AWS	777777777777	grizzlybear.stage@example.ca.gov	Staging for GrizzlyBear quality assurance	grizzlybear.engineering@example.ca.gov
gb-dev	AWS	888888888888	grizzlybear.dev@example.ca.gov	Dev environment for GrizzlyBear	grizzlybear.engineering@example.ca.gov

If you are unsure what accounts you already have with cloud service providers, you can ask a Technical Account Manager to identify accounts with your department's email domain. Ensure that all accounts are associated with email accounts under your control – it is critical that you do not use personal accounts. If you suspect this may be happening in your organization, contact the Office of Information Security for assistance in identifying "shadow" accounts.

The root email address associated with each account should be unique and associated with an email distribution list of personnel responsible and accountable for the resources in that account. In the event the cloud service provider needs to notify the account owner, they will contact this email address. By using a distribution list, you can ensure critical messages are seen in a timely manner by all relevant personnel. Additionally, this allows you to change the personnel in your email system, as will inevitably be required with personnel changes over time.

You should also note any trust relationships between accounts, where the segmentation boundary between account is bridged.

2. Maintain a method of tracking configuration changes and viewing inventory and configuration history of cloud services.

Cloud services provide a way to view resource inventory and configuration history, but it must be enabled and configured. For example, AWS Config[1], Azure Resource Manager[2], or GCP Resource Manager[3].

You should strive to define all your cloud infrastructure as code and deploy changes through an automated pipeline with immutable logging. This will make it easier to keep track of what changes were made, who submitted them, and who approved them.

3. Apply resource tags to data and applications according to their categorization and criticality.

SAM Section 5305.5 describes classification processes and requirements. Cloud resources can be marked with these classifications by applying tags, where you define keys and values. Your organization should adopt a tagging strategy that meets your needs. Then, you can use these tags in protective, detective, and responsive controls.
AWS – Tagging Strategies[4]
Azure – Use tags to organize resources[5]
GCP – Labelling and grouping resources[6]

B. Protect

Protection of cloud resources can be organized into three categories: identity and access management, infrastructure protection, and data protection. In the cloud, all data sources and computing services are considered resources and all access is granted through an identity and access management (IAM) system. Cloud resources may not always resemble on-premises hardware systems, so traditional techniques may not apply. IAM is especially important in these cases, as there are fewer layers of defense-in-depth protections. For example, “serverless” functions may not reside on a virtual machine or inside a network. Instead, these resources rely on

[1] https://aws.amazon.com/config/

[2] https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview

[3] https://cloud.google.com/resource-manager/

[4] https://d0.awsstatic.com/aws-answers/AWS_Tagging_Strategies.pdf

[5] https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-using-tags

[6] https://cloud.google.com/blog/products/gcp/labelling-and-grouping-your-google-cloud-platform-resources

policies that define who can execute them and what other resources they have access to. Effective IAM relies on: (1) maintaining fine-grained and role-based authorization based on least-privilege and (2) protecting access credentials. Infrastructure and data protection controls employ layered safeguards to reduce the attack surface exposed and limit the impact of unintended or unauthorized access.

Identity and Access Management

1. Maintain a tiered account management structure and apply restrictions on subordinate accounts (e.g., denying the removal of logging and security features, denying access to services that do not comply with regulatory requirements).

Cloud account structures evolve along with usage, so you should not expect to establish a permanent structure from the outset. Instead:

- Create accounts for centralized monitoring and management
- Create separate accounts for distinct purposes
- Define security policies and restrict account usage to approved activities
- Develop and document an account management process (discussed in item 8 below)

Accounts are the strictest access boundary available, as no access is allowed across them without explicit authorization. Each account must be configured to trust other accounts, but only as necessary. A compartmentalized account structure could look as follows:

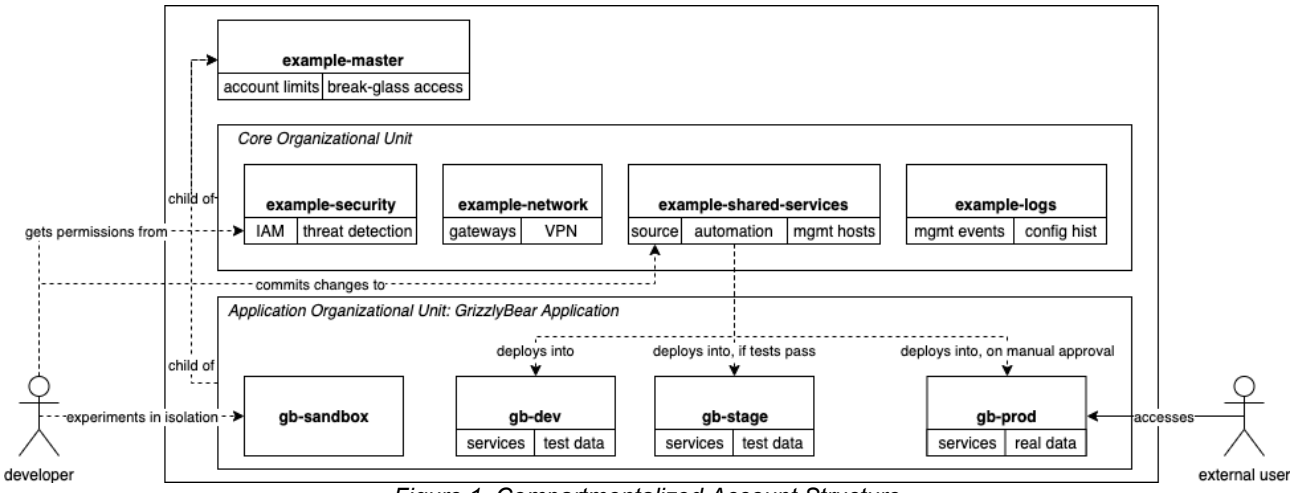


Figure 1. Compartmentalized Account Structure

A master account is used to organize the other accounts for example-department. A dedicated security account defines the access policies to the other accounts, manages security policies, and threat detection. A centralized logging account stores management event logs and configuration history. A network account controls routing between applications and other networks, potentially to on-premises networks. A fictional application, GrizzlyBear, uses the shared-services account to deploy into separate accounts for development, staging, and production environments. These accounts are grouped into an organizational unit. If example-department were to begin another

project, it would create a new organizational unit and separate accounts, to maintain isolation from the GrizzlyBear project.

Consider applying the following security policies to each subordinate account:[7]

- Prevent users from disabling service logging
- Prevent users from disabling or altering alerting
- Prevent users from deleting network flow logging
- Prevent users from disabling or altering configuration management controls
- Prevent users from creating new ways for networks to access the internet
- Deny access to non-approved regions
- Require encryption on data storage services
- Limit compute resources to specific types
- Require multi-factor authentication for specific actions like stopping compute instances on production workloads
- Require tagging upon resource creation

AWS – Multiple Account Security Strategy[8]

Azure – Organize Azure subscriptions into management groups[9]

GCP – Cloud Platform Resource Hierarchy[10]

2. Restrict usage of superuser access (i.e., root users) to the creation of less-privileged users for role-based access and administrative actions that can only be performed with superuser access.

Some user accounts have unlimited access to the services in the account. AWS calls this user account the *root user* and recommends only using it to create less-privileged accounts and a limited

---

[7] https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_example-scps.html

[8] https://d0.awsstatic.com/aws-answers/AWS_Multi_Account_Security_Strategy.pdf

[9] https://docs.microsoft.com/en-us/azure/security/fundamentals/operational-best-practices#organize-azure-subscriptions-into-management-groups

[10] https://cloud.google.com/resource-manager/docs/cloud-platform-resource-hierarchy

set of administrative tasks that require the root user.[11] The only way to limit this user is to limit the account itself, using a Service Control Policy applied to the organizational unit.[12] The root user account should have multi-factor authentication (MFA) enabled, a very strong password, and no access keys. Your organization should establish procedures for controlling and monitoring access to the password and MFA device.[13]

When creating permissions for users, pay special attention to the permissions that may allow that user to escalate their own permissions. Tools may be available to scan for permissions that could be used to escalate, such as aws_escalate[14]. For example, the following AWS permissions present risks:

Permission	**Risk**
iam:CreatePolicyVersion	With the flag -set-as-default, this action can override the current default policy even without permission to iam:SetDefaultPolicyVersion.
iam:SetDefaultPolicyVersion	This action can set an otherwise inactive policy as default, assigning it unintentionally.
iam:PassRole	Users who also have ec2:RunInstances or cloudformation:CreateStack can create new resources with an existing service role that they otherwise wouldn't have access to. They can then use these resources to inherit permissions or take actions beyond their intended scope.
iam:CreateAccessKey	This action can create a second set of access keys for an existing user, allowing an attacker access to that user's permissions.
iam:AttachUserPolicy, iam:AttachGroupPolicy, iam:AttachRolePolicy, iam:PutUserPolicy, iam:PutGroupPolicy, iam:PutRolePolicy	These permissions allow escalation by attaching policies to existing users, groups, or roles that were not intended to have those policies.

3. Require multi-factor authentication for all (1) privileged access, (2) user access to sensitive or confidential data, and (3) accounts representing official communications from state

[11] https://docs.aws.amazon.com/general/latest/gr/aws_tasks-that-require-root.html

[12] https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scp.html

[13] https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#lock-away-credentials

[14] https://github.com/RhinoSecurityLabs/Security-Research/blob/master/tools/aws-pentest-tools/aws_escalate.py

departments.

Requiring multi-factor authentication (MFA) is a simple but effective way to prevent unwanted access.[15][16] However, not all MFA mechanisms are equal. Avoid using text-messages based on Short Message Service (SMS), as they can be intercepted by malicious users through various means. Prefer software-based or hardware MFA from trusted sources when possible.

You can further enforce this requirement by creating policies that validate that the user has used MFA for certain actions.[17]

4. Configure fine-grained user permissions according to least privilege.

Create individual user accounts for those with authorized access, give them only permissions required to perform their duties, and avoid sharing account credentials. Cloud Service Providers (CSP) may use different taxonomy for account management settings. The following are provided as examples.

AWS – Grant least privilege[18], avoid sharing credentials[19]

Azure – Role-based access control[20]

GCP – Access control for organizations[21]

5. Periodically audit access and remove unused credentials and permissions.

As your system evolves, you may end up with credentials and permissions that are no longer necessary. By removing these, you reduce the risk of abuse. There are various tools that may assist you in this process:

- AWS
    - AWS IAM Access Advisor[22]

---

[15] https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#enable-mfa-for-privileged-users

[16] https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/howto-conditional-access-policy-admin-mfa

[17] https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-role.html#cli-configure-role-mfa

[18] https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

[19] https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#sharing-credentials

[20] https://docs.microsoft.com/en-us/azure/role-based-access-control/overview

[21] https://cloud.google.com/resource-manager/docs/access-control-org

[22] https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_access-advisor.html

- CloudTracker[23] helps find over-privileged IAM users and roles by comparing CloudTrail logs with current IAM policies.
- Principal Mapper (PMapper)[24] is a script and library for identifying risks in the configuration of AWS IAM.
- Aaia[25] helps visualize IAM policies and identify anomalies.
- Policy Sentry[26] helps prevent this issue by generating least-privilege policies

- Azure
  - Azure Active Directory portal[27]
- GCP
  - Viewing cloud audit logs[28]

6. Maintain logical perimeters between production and non-production environments (e.g., development, test). Prohibit using the same credentials across environments, except where single sign-on technologies generate unique credentials for federated access.

By delineating between production and non-production environments and requiring separate access credentials, you can limit the impact of unwanted actions, whether accidental or malicious. As demonstrated previously in Figure 1, the preferred way to separate environments is with isolated accounts. This ensures that credentials which provide access to one environment do not also provide access to another environment, unless specifically configured.

7. Prohibit embedding credentials directly into code – configure applications to retrieve necessary credentials programmatically. Where feasible, programmatically generate temporary credentials instead of long-term credentials like passwords or access keys.

Cloud applications will require credentials to access various services, but these should always be stored in a secrets management system and never in an application's code. Secrets management services should store secrets encrypted and provide the ability to rotate keys and/or generate temporary credentials on-demand. Cloud providers offer services for this, such as AWS Secrets

---

[23] https://github.com/duo-labs/cloudtracker

[24] https://github.com/nccgroup/PMapper

[25] https://github.com/rams3sh/Aaia

[26] https://github.com/salesforce/policy_sentry

[27] https://docs.microsoft.com/en-us/azure/active-directory/reports-monitoring/concept-audit-logs

[28] https://cloud.google.com/logging/docs/audit/best-practices#viewing-best-practices

Manager[29] or Azure Key Vault[30], but you may also choose third-party solutions, such as Hashicorp Vault[31].

8. Provide access to cloud services by federated authentication through a centralized identity and access management system.

Cloud services work well with single sign-on technologies, where a user is granted access based on federated authentication by a centralized system. This provides several advantages, such as: adding/removing users in a single system rather than multiple, users do not need to remember additional passwords, and the ability to employ adaptive access control techniques (discussed below). You should consider the impact that unavailability of this system would cause and plan accordingly. You may need to make separate accounts that do not rely on an external identity provider – if so, these accounts must be carefully protected as they could provide an avenue of attack that bypasses controls.

9. Deploy adaptive access control technologies to dynamically adjust authentication requirements based on additional information (e.g., endpoint security posture, user or entity behavior, location).

Modern identity providers provide capabilities to adapt access based on information beyond simply a username and password. For example, it may require the user's device to demonstrate it has been updated with the latest security patches and/or is running an endpoint protection service. It may deny access based on impossible travel, where an account is attempting to access a system from multiple locations without enough time to travel between them. These capabilities must be carefully balanced to ensure availability to legitimate users and deny potential attackers.

Infrastructure Protection

10. Establish network topologies to limit traffic routing only between resources as necessary.

By partitioning resources into logical segments and only allowing traffic between them as necessary, you can limit the exposure of your resources to potential attack vectors, even from within your network. In the event an attacker gains control of one resource, the ability to send malicious traffic to another resource will be limited. For defense-in-depth, employ this technique using subnets, route tables, network access control lists, and virtual firewalls.

The first three work together – divide resources into subnets, define route tables that allow traffic between those subnets only as necessary, and define network access control lists to allow or deny traffic based on address range and port. These concepts are not unique to cloud but remain necessary when building cloud networks.

The cloud offers another feature, virtual firewalls, which are the most powerful way to partition cloud networks. AWS calls this functionality security groups, Azure calls it network security groups, and GCP calls it firewall rules. These firewalls are stateful, so you only specify one traffic direction (ingress or egress) and the return traffic is also allowed. You should define rules that **allow** intended

[29] https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

[30] https://azure.microsoft.com/en-us/services/key-vault/

[31] https://www.hashicorp.com/products/vault/

traffic and deny all other traffic. When defining rules, you should specify resources by name when possible, as opposed to IP address ranges. Names are more specific, less likely to change, and more readable.

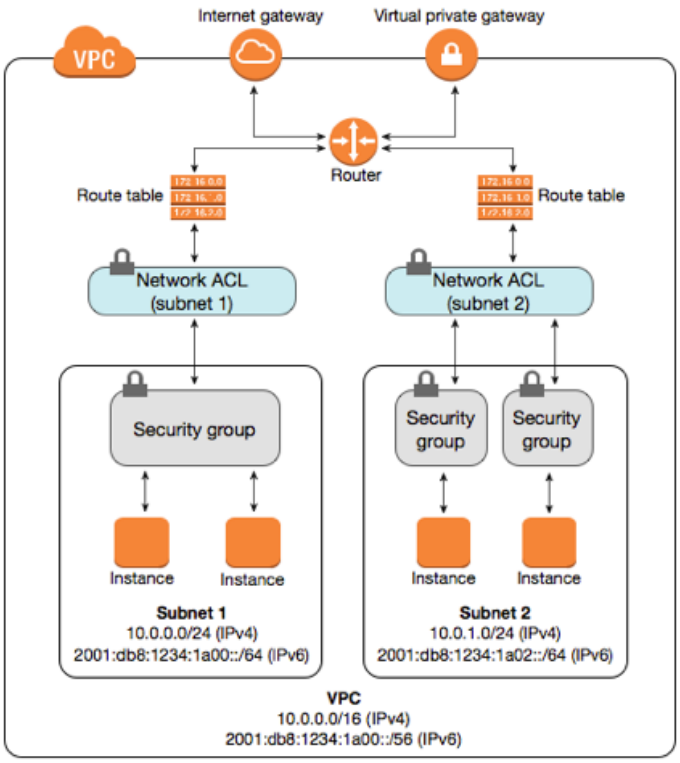


Figure 2. Network partitioning

AWS – Use VPC flow logs to diagnose rejected traffic, determine initiation direction, and determine what ports and protocols are necessary.[32]

11. Limit resource exposure to the public internet to only those resources intended to be publicly accessible and protected accordingly, including deployment of endpoint defense capabilities in accordance with SAM Section 5355.

As described above, you will partition virtual networks into subnets and route traffic between them only as necessary. One of the most important decisions to make is which subnets, and therefore resources, will be publicly reachable. Any resources that do not need to be reached over the internet should be placed in private subnets, where they only receive private addresses and do not have a route to the internet. Place additional focus on protecting those resources which do need to be publicly reachable, limiting the exposed services and deploying endpoint defense capabilities.
Azure – Restrict direct internet connectivity[33]
You should always be aware of the resources that are publicly accessible and monitor for changes. You may accidentally expose a resource, or, in the event of a breach, attackers may create a new way back into the environment by creating public resources. Consider developing programmatic

[32] https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html

[33] https://docs.microsoft.com/en-us/azure/security/fundamentals/iaas#restrict-direct-internet-connectivity

ways to monitor publicly accessible resources. For example, you can list all public IP addresses of compute instances using the AWS command line:

```
aws ec2 describe-instances --query "Reservations[*].Instances[*].PublicIpAddress[]" --
output=text
```

Not all cloud resources are networked in this way, but still require consideration to limit exposure. PaaS and SaaS services may be configured using policies that potentially allow public access. If not needed, utilize features to block public access to these services to prevent accidental exposure. For example, AWS Simple Storage Service (S3) can be configured to block public access.[34]

12. Deploy Web Application Firewalls and/or Distributed Denial of Service protection services to protect public-facing applications.

Web Application Firewalls can protect public-facing applications by filtering out common web vulnerabilities before they reach your application. This can prevent potential vulnerabilities in your application from being exploited, as well as reduce the traffic that reaches your applications and consumes its resources. The scalable nature of cloud services can present some non-obvious risks. If you don't architect an application to scale with traffic, you are susceptible to being overwhelmed by traffic and having your application's availability denied (i.e., a Distributed Denial of Service attack). If you do architect for scalability, however, you may be susceptible to an attacker causing you to utilize excessive resources and increase your costs (i.e., an Economic Denial of Service attack). To address this challenge, architect for scalability but utilize Content Delivery Networks for delivering web content, configure Web Application Firewalls to filter out common attack patterns, and configure additional Denial of Service protections offered by your cloud provider.

13. Limit virtual machine access to instance metadata services.

Cloud virtual machines can retrieve potentially sensitive data about their environment by sending requests to an internal address, usually 169.254.169.254. For example, the virtual machine may use this service to retrieve access keys that allow it to perform actions in the cloud. The instance metadata service is supposed to be accessible only from the virtual machine. However, attackers may be able manipulate an application into retrieving sensitive data and returning it to the attacker, for example through Server Side Request Forgery (SSRF).[35]
To mitigate this risk:

- Disable the metadata service for any virtual machines which do not require it.
- Limit the permissions providing to the virtual machine to least privilege, to limit the impact of compromise.
- Utilize local firewall rules to deny sending traffic to the metadata service except from the processes that require it.
- Pay particular attention to virtual machines which are widely reachable, for example from the internet.

14. Limit deployments and maintenance to automated technologies as much as possible, disabling services used for manual administration.

Services for manual administration, such as SSH or RDP, are a common attack vector and should

[34] https://aws.amazon.com/blogs/aws/amazon-s3-block-public-access-another-layer-of-protection-for-your-accounts-and-buckets/

[35] https://attack.mitre.org/techniques/T1522/

be disabled or limited as much as possible. Consider establishing an immutable infrastructure model where manual administration is not required, discussed further in Section IV. This approach may not be realistic for your system in the near term, however. You can still limit the risk of manual maintenance by using agent-based session management tools which alleviate the need to manage SSH keys and provide better auditing. For example, AWS Systems Manager can provide shell access to compute instances or run commands across a fleet of servers.[36] If you can eliminate the need for manual administration tools, you can eliminate the need for a bastion host exposed to the internet and thus reduce your attack surface.

15. Employ hardening practices designed for cloud virtual machines.

For example, the Center for Internet Security provides recommended configurations and pre-configured virtual machine images for Amazon Linux distributions.[37]

Data Protection

16. Select and configure storage services according to data availability (i.e., resilience to system downtime) and durability (i.e., resilience to data loss) requirements, which may include replication across cloud service provider zones and/or regions.

You should select data storage services and develop failover strategies based on a Business Impact Assessment. Cloud services offer simplified ways to fulfill your business needs, including object-based storage and managed databases with automated replication and failover. However, you must select and configure these services to balance cost with business needs.
When selecting services, identify the cloud service provider's commitment in the service's SLA. Durability refers to expected loss of data. Availability refers to the expected downtime of your data. Often, these are expressed as a percentage in a year by stating the "number of nines."

Availability %	Downtime per year	Downtime per month
99% ("two nines")	3.65 days	7.31 hours
99.9% ("three nines")	8.77 hours	43.83 minutes
99.5% ("two nines five")	1.83 days	3.65 hours
99.99% ("four nines")	52.60 minutes	4.38 minutes
99.999% ("five nines")	5.26 minutes	26.30 seconds
99.9999999% ("nine nines")	31.56 milliseconds	2.63 milliseconds

[36] https://aws.amazon.com/systems-manager/

[37] https://www.cisecurity.org/cis-hardened-images/

Some storage classes offer reduced cost in exchange for reduced availability while retaining high durability. For example, Amazon Simple Storage Service (S3) offers an "Infrequent Access" storage class with a service level agreement for two nines of availability, so the data may be unavailable for 3 days in a year without violating their agreement.[38] This may be acceptable for some of your data, but a significant business disruption for others. When selecting a storage class, consider the impact that downtime would have on your business and carefully evaluate the service level agreements available.

17. Configure fine-grained data access policies.

Avoid providing broad policies that allow users access to sensitive data. Implement least privilege by configuring policies that:

- Grant read-only access to certain items, attributes, tables, and/or indexes based on their role and the resource tags applied to the data.
- Grant write access to certain items, attributes, tables, and/or indexes based on their role.
- Require resource tags to be applied at time of creation, to maintain data classification standards.
- Prevent users from deleting database instances, especially with certain resource tags.

AWS – Using Tags in IAM[39], Relational Database Policies[40], DynamoDB Policies[41], S3 Policies[42]
Azure – Role-based access control[43], Secure blob storage with RBAC[44]
GCP – Using resource hierarchy for access control[45], Managing conditional policies[46]

18. Protect data at rest by employing SAM Section 5350.1 compliant encryption and/or tokenization methods to transform confidential, sensitive, or personal data into a form that is unreadable to unauthorized users.

One benefit to cloud services is the easy access to strong encryption services and their integration

---

[38] https://aws.amazon.com/s3/sla/

[39] https://aws.amazon.com/premiumsupport/knowledge-center/iam-ec2-resource-tags/

[40] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/security_iam_id-based-policy-examples.html

[41] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/using-identity-based-policies.html

[42] https://docs.aws.amazon.com/AmazonS3/latest/dev/example-policies-s3.html

[43] https://docs.microsoft.com/en-us/azure/role-based-access-control/overview

[44] https://docs.microsoft.com/en-us/azure/storage/common/storage-auth-aad-rbac-cli

[45] https://cloud.google.com/iam/docs/resource-hierarchy-access-control

[46] https://cloud.google.com/iam/docs/managing-conditional-policies

with other cloud services. AWS Key Management Service[47], Azure Key Vault[48], and Google Cloud Key Management Service[49], Oracle Cloud Key Management[50], and IBM Key Protect[51] utilize Hardware Security Modules which are validated at FIPS 140-2 Level 2, at least.
Where possible, configure storage encryption to be enabled by default. For example, AWS allows you to opt-in to default encryption for EBS volumes[52] and S3 buckets[53].
Azure – Encrypt virtual hard disk files[54]

19. Protect data encryption keys from unauthorized use by defining restrictive policies for key use that enforce the principles of least privilege and separation of duties.

For example:

- separate users with key administration permissions from users with key use permissions
- separate applications that require permission to encrypt data from applications that require permission to decrypt data
- require decryption requests to come from a trusted network path

AWS – Key Management Service Best Practices[55]

20. Establish encryption key rotation policies to limit the impact of a single compromised key.

Utilizing your cloud provider's native key management services makes this easy, as they can be configured to automatically rotate keys.

21. Employ SAM Section 5350.1 compliant encryption methods to protect data in transit outside trusted network boundaries, even across dedicated network connections to cloud service providers.

Dedicated network connections to cloud service providers may be private, but do not encrypt data in transit by default. You may be able to establish a Virtual Private Network (VPN) across the connection to provide encryption across the link. Otherwise, you may need to configure application-level encryption.

---

[47] https://aws.amazon.com/kms/features/

[48] https://azure.microsoft.com/en-us/services/key-vault/

[49] https://cloud.google.com/kms/

[50] https://docs.cloud.oracle.com/iaas/Content/KeyManagement/Concepts/keyoverview.htm

[51] https://cloud.ibm.com/catalog/services/key-protect#about

[52] https://aws.amazon.com/blogs/aws/new-opt-in-to-default-encryption-for-new-ebs-volumes/

[53] https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html

[54] https://docs.microsoft.com/en-us/azure/security/fundamentals/iaas#encrypt-your-virtual-hard-disk-files

[55] https://d0.awsstatic.com/whitepapers/aws-kms-best-practices.pdf

AWS – Direct Connect[56]

Azure – ExpressRoute Encryption[57]

GCP – Cloud Interconnect[58]

22. Configure data retention policies to automatically destroy data when it is no longer needed.

Cloud storage can be configured for automatic archival and/or destruction according to your requirements. By archiving data, you can limit the likelihood that an attacker can quickly retrieve it. By deleting data when it is no longer necessary, you can prevent unintentional disclosure.

AWS – S3 Lifecycle Policies[59]

Azure – Blob storage lifecycle management[60]

GCP – Cloud storage lifecycle management[61]

23. Employ techniques for automated discovery and classification of sensitive data.

Each cloud provider offers services to support data classification, particularly discovery of sensitive data:

- AWS Macie[62]
- Azure Information Protection[63]
- Google Cloud Data Loss Prevention[64]

24. Employ encryption methods to protect data in transit within trusted network boundaries.

Cloud services are designed to maintain separation between tenants, even when operating on shared hardware. Nevertheless, you should employ defense in depth by designing your systems to

---

[56] https://aws.amazon.com/directconnect/faqs/?nc=sn&loc=6

[57] https://docs.microsoft.com/en-us/azure/expressroute/expressroute-about-encryption

[58] https://cloud.google.com/interconnect/docs/how-to/choose-type

[59] https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-lifecycle.html

[60] https://docs.microsoft.com/en-us/azure/storage/blobs/storage-lifecycle-management-concepts?tabs=azure-portal

[61] https://cloud.google.com/storage/docs/lifecycle

[62] https://aws.amazon.com/macie/

[63] https://azure.microsoft.com/en-us/services/information-protection/

[64] https://cloud.google.com/dlp/

encrypt data in transit within your networks. For example, encrypt connections between application services and databases, even if they are in private subnets.

AWS – Encrypt a Connection to a Relational Database Service (RDS) Database[65]
Azure – Protect SQL databases using Always Encrypted[66]
GCP – Configuring Encryption for Cloud SQL[67]

C. Detect

Despite our best efforts to protect resources, security events happen – timely detection enables us to respond and mitigate potential impacts. As a cyber defense practitioner, you should assume your system is compromised and develop capabilities to discover malicious behavior. Cloud services can improve our ability to detect vulnerabilities and potential security events by providing rich native logging capabilities, modern threat detection services, and automated response.

1. Log cloud management events to centralized log storage for each cloud service provider, maintaining audit records in accordance with SAM Section 5335.2.

Protect logs by configuring immutable storage and preventing accidental or malicious deletion through separation of privilege.
AWS – Creating an organization trail in CloudTrail[68], enabling MFA delete[69]
Azure – Platform logs[70], Secure critical data with immutable storage[71]
GCP – Cloud Audit Logs Best Practices[72]

2. Establish threat prevention capabilities to identify weak configurations and notify security personnel. Configure automated remediation where possible.

Each cloud provider has offerings to assist with this.

---

[65] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html

[66] https://docs.microsoft.com/en-us/azure/sql-database/sql-database-always-encrypted-azure-key-vault?tabs=azure-powershell

[67] https://cloud.google.com/sql/docs/sqlserver/configure-ssl-instance

[68] https://docs.aws.amazon.com/awscloudtrail/latest/userguide/creating-trail-organization.html

[69] https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMFADelete.html

[70] https://docs.microsoft.com/en-us/azure/azure-monitor/platform/platform-logs-overview

[71] https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blob-immutable-storage

[72] https://cloud.google.com/logging/docs/audit/best-practices

- AWS: Trusted Advisor[73], Config[74], Security Hub[75]
- Azure: Advisor[76], Policy[77], Security Center[78]
- GCP: Security Health Analytics[79], Security Command Center[80]

There are also free and open-source tools available, such as:

- Multi-cloud: Scoutsuite[81]
- AWS: Prowler[82]
- Azure: Microburst[83]
- GCP: G-Scout[84]

Third-party vendors also offer products to support this effort, sometimes termed Cloud Security Posture Management.

3. Establish threat detection capabilities informed by threat intelligence providers and configure alerts to notify security personnel.

Each cloud provider has a threat detection service informed by threat intelligence partners and experience with their platform. These services can help you detect anomalies, but they require configuration to reduce false positives, hone-in on likely security threats, and notify the correct personnel with helpful information.

---

[73] https://aws.amazon.com/premiumsupport/technology/trusted-advisor/

[74] https://aws.amazon.com/config/

[75] https://aws.amazon.com/security-hub/

[76] https://azure.microsoft.com/en-us/services/advisor/

[77] https://docs.microsoft.com/en-us/azure/governance/policy/overview

[78] https://azure.microsoft.com/en-us/services/security-center/

[79] https://cloud.google.com/security-command-center/docs/how-to-manage-security-health-analytics

[80] https://cloud.google.com/security-command-center

[81] https://github.com/nccgroup/ScoutSuite

[82] https://github.com/toniblyx/prowler

[83] https://github.com/NetSPI/MicroBurst

[84] https://github.com/nccgroup/G-Scout

In a multi-account environment, you should centralize the alerts, feed them into a Security Information and Event Management (SIEM) system, and work with your security team to detect advanced risks.

AWS - GuardDuty[85], CloudTrail Insights[86]

Azure – Advanced Threat Detection[87]

GCP – Event Threat Detection[88], Anomaly Detection[89]

4. Implement tools to detect access credentials being stored in source control repositories.

Access credentials may be accidentally leaked by storing them in source control repositories, giving others unintentional access. You can prevent access credentials from being committed to a source repository by having developers configure a tool on their computer such as detect-secrets[90] or git-secrets[91]. These rely on the tool being properly configured on a developer's computer, however, so they can be easily bypassed. You can also configure a tool on the server-side to detect credential leaks, such as detect-secrets-server[92], so you can revoke those credentials in a timely manner.

5. Publish logs to the Security Information and Event Management (SIEM) system operated by the California Department of Technology (CDT) Security Operations Center (SOC).

The CDT SOC coordinates with the California Cybersecurity Integration Center to maintain threat detection capabilities informed by intelligence based on observations from state agencies and private sector partners.

D. Respond

The worst time to develop a response strategy is when you've already discovered a potential security event. Instead, prepare for incident response as you develop your system and, when possible, automate response and remediation. Cloud services enable automated response by exposing APIs for programmatic interaction, but you must configure these capabilities before an

---

[85] https://aws.amazon.com/guardduty/

[86] https://docs.aws.amazon.com/console/awscloudtrail/insights

[87] https://docs.microsoft.com/en-us/azure/security/fundamentals/threat-detection

[88] https://cloud.google.com/event-threat-detection/

[89] https://cloud.google.com/security-command-center/docs/how-to-view-vulnerabilities-threats#anomaly_detection

[90] https://github.com/Yelp/detect-secrets

[91] https://github.com/awslabs/git-secrets

[92] https://github.com/Yelp/detect-secrets-server

incident. When responding to security incidents in the cloud, try to identify the domain of the incident, as this will inform your remediation process. For example, the incident may affect the cloud service (e.g., cloud account, IAM policies), your cloud infrastructure (e.g., virtual machines or networks), or the application (e.g., in the application code or running software).

1. Ensure incident response plans include procedures for notifying and coordinating with cloud service providers.

Your incident response plan should identify how your cloud provider would notify you in the event they detect an incident affecting your account. Maintaining updated contact information, including root email addresses tied to email distribution lists with the correct personnel, is critical to receiving these notifications in a timely manner.

AWS – Managing account contact information[93]

Azure – Receive incident notifications from Microsoft[94]

GCP – Security Bulletins[95]

Depending on your support plan, you may be able to request assistance from your cloud service provider. For instance, AWS Shield Advanced provides additional protection from Distributed Denial of Service attacks which includes support from their response team. Your incident response plan should outline what support you have available and how to utilize it.

AWS – Submit support request[96]

Azure – Create support request[97]

GCP – Support center[98]

You should also include procedures for reporting abuse detected in your account.

AWS – Report abuse[99]

---

[93] https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/manage-account-payment.html

[94] https://docs.microsoft.com/en-us/azure/security/fundamentals/operational-best-practices#receive-incident-notifications-from-microsoft

[95] https://cloud.google.com/support/bulletins

[96] https://console.aws.amazon.com/support/home#/case/create?issueType=technical

[97] https://docs.microsoft.com/en-us/azure/azure-portal/supportability/how-to-create-azure-support-request

[98] https://enterprise.google.com/supportcenter/managecases

[99] https://support.aws.amazon.com/#/contacts/report-abuse

Azure – Submit Abuse Report[100]

GCP – Report suspected abuse[101]

2. Ensure incident response procedures include providing access to external incident responders and protecting this access from unintentional use.

During a security incident, you may require assistance from external incident responders, for example the California Cybersecurity Integration Center. You should prepare access mechanisms and permissions in advance so they can help you perform remediation actions in your cloud accounts.

Types of access to consider providing incident responders include:

- read-only access to resources and/or security configurations, so responders can view your environment and provide advice but not alter any configurations
- access to isolated environments for forensic analysis
- privileged access to take remediation actions on your behalf

3. Configure automated remediation of weak configurations.

Security events may arise from accidental or malicious configurations. Consider using services from your cloud provider or third-party vendors to identify weak configurations and automatically remediate them. For example, you can automatically cut off access to databases or virtual machines with administration services found to be reachable from the open internet. Where possible, make response mechanisms safe to execute more than once and on unknown states. This is a significant feature of defining cloud infrastructure as code, discussed further below.

AWS – Automated Response and Remediation with Security Hub[102]

Azure – Remediate non-compliant resources with Azure Policy[103]

GCP – Remediating Security Health Analytics findings[104]

4. Configure automated responses for incident investigation, such as isolating resources from network access but preserving resource state.

In some cases, you may be able to detect a security event but the remediation steps may not be clear. Consider automating actions that limit the ongoing impact and enable further investigation. For example, if your monitoring solution detects anomalies coming from a virtual machine, you can

---

[100] https://portal.msrc.microsoft.com/en-us/engage/cars

[101] https://support.google.com/code/contact/cloud_platform_report

[102] https://aws.amazon.com/blogs/security/automated-response-and-remediation-with-aws-security-hub/

[103] https://docs.microsoft.com/en-us/azure/governance/policy/how-to/remediate-resources

[104] https://cloud.google.com/security-command-center/docs/how-to-remediate-security-health-analytics

automate recording the virtual machine metadata, changing its associated security group to deny network traffic, disconnect it from load balancers or auto-scaling, snapshot the storage volume attached, and tag it as under investigation. You can then coordinate with incident responders to perform offline forensic analysis of the storage volume.

E. Recover

If services are impaired due a security event, you should be prepared to restore those services in a timely manner. Cloud services provide extensive backup and recovery capabilities, like snapshots of virtual machines and databases. However, you should consider the likelihood of the backup being compromised at the same time as the primary and architect to avoid correlated failures. Cloud datacenters are typically organized into availability zones and regions, where a region is a collection of availability zones. Although failure of a region is unlikely, you can prepare for this by sending backups to another region and preparing your applications to run in another region.

1. Provision for data preservation and retrieval in agreements with cloud service providers, including but not limited to:

    a. Identification of requisite formats for transfer of data to state entity or subsequent service provider.
    b. A defined transition period to enable a successful transfer of data from service provider to state entity.

As you prepare to import and/or create data in the cloud, consider if you may eventually need to migrate that data to another datacenter or service provider. You may want to avoid using proprietary formats and technologies for critical data if extraction and translation would be difficult or costly. Major cloud service providers offer ways to migrate data out using physical devices like AWS Snowball[105] or Azure Data Box[106], so you should select data formats that are compatible with these options.

2. Configure automated data backups and virtual machine snapshots across zones and/or regions, according to recovery time and recovery point objectives.

Cloud datacenters may become unavailable due to various reasons, so you should consider the impact that an outage would have on your workloads and plan accordingly. For some very low criticality applications and data, an outage may not be a concern so locating them in a single zone may be acceptable. For most applications, however, you should architect for availability across zones at a minimum and consider sending backups to another region. Consider creating IAM policies that separate the permissions needed to send backups from the permissions needed to modify those backups, to prevent accidental or malicious deletion.

3. Define and deploy cloud infrastructure using code-like methods, back up configurations and scripts, and protect them from deletion.

If your cloud infrastructure was accidentally or maliciously destroyed, how much would you be able to recover? If you configure cloud services through the web user interface, you may lose everything.

---

[105] https://aws.amazon.com/snowball/

[106] https://azure.microsoft.com/en-us/services/databox/

If you define infrastructure as code, protect the source code repository, and maintain immutable data backups, you may be able to redeploy it all within a few hours.

Each cloud provider has a native service for Infrastructure as Code and there are third-party options available. Each option involves trade-offs and continues to mature, so you should evaluate alternatives for your specific use case.

AWS - CloudFormation[107]

Azure – Resource Manager[108]

GCP – Deployment Manager[109]

Terraform[110]

Ansible[111]

[107] https://aws.amazon.com/cloudformation/

[108] https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview

[109] https://cloud.google.com/deployment-manager/

[110] https://www.terraform.io/

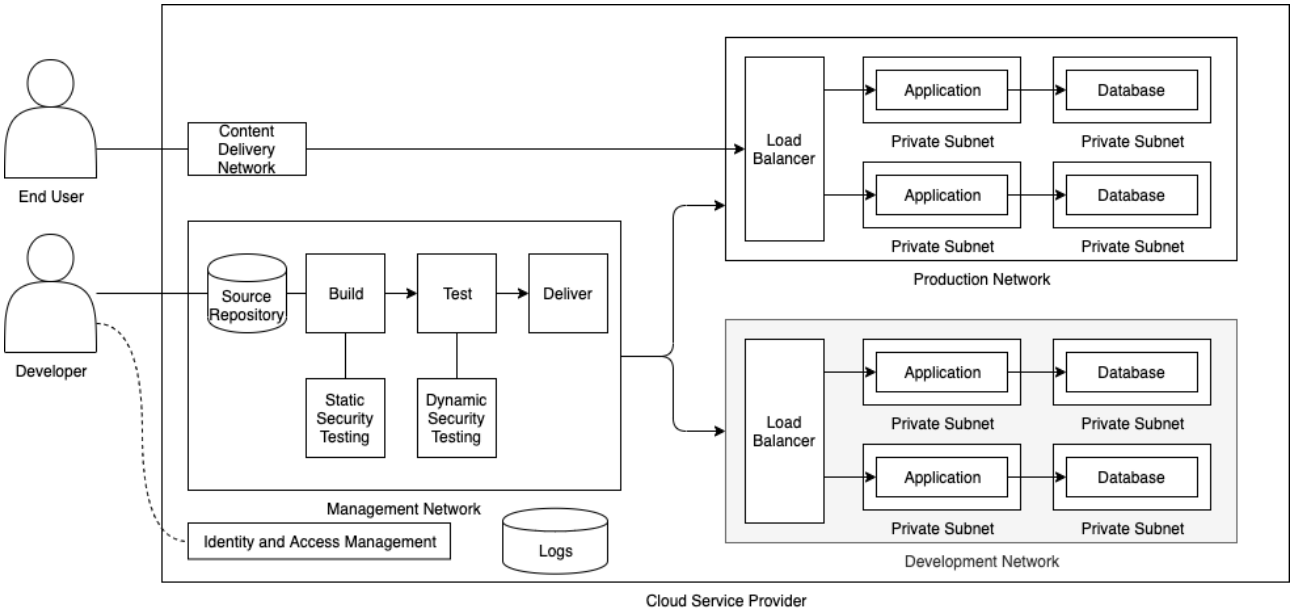
[111] https://docs.ansible.com/ansible/latest/index.html

III. REFERENCE ARCHITECTURES

Systems must be architected based on their unique requirements, but the following architectures can provide a reference during design and development.

A. Standardized Architecture

From a cloud-agnostic perspective, a typical system architecture is illustrated below.



Separate virtual (i.e., software-defined) networks are used for management, development and production purposes. Traffic is routed between networks and applications only as necessary; all other traffic is denied.

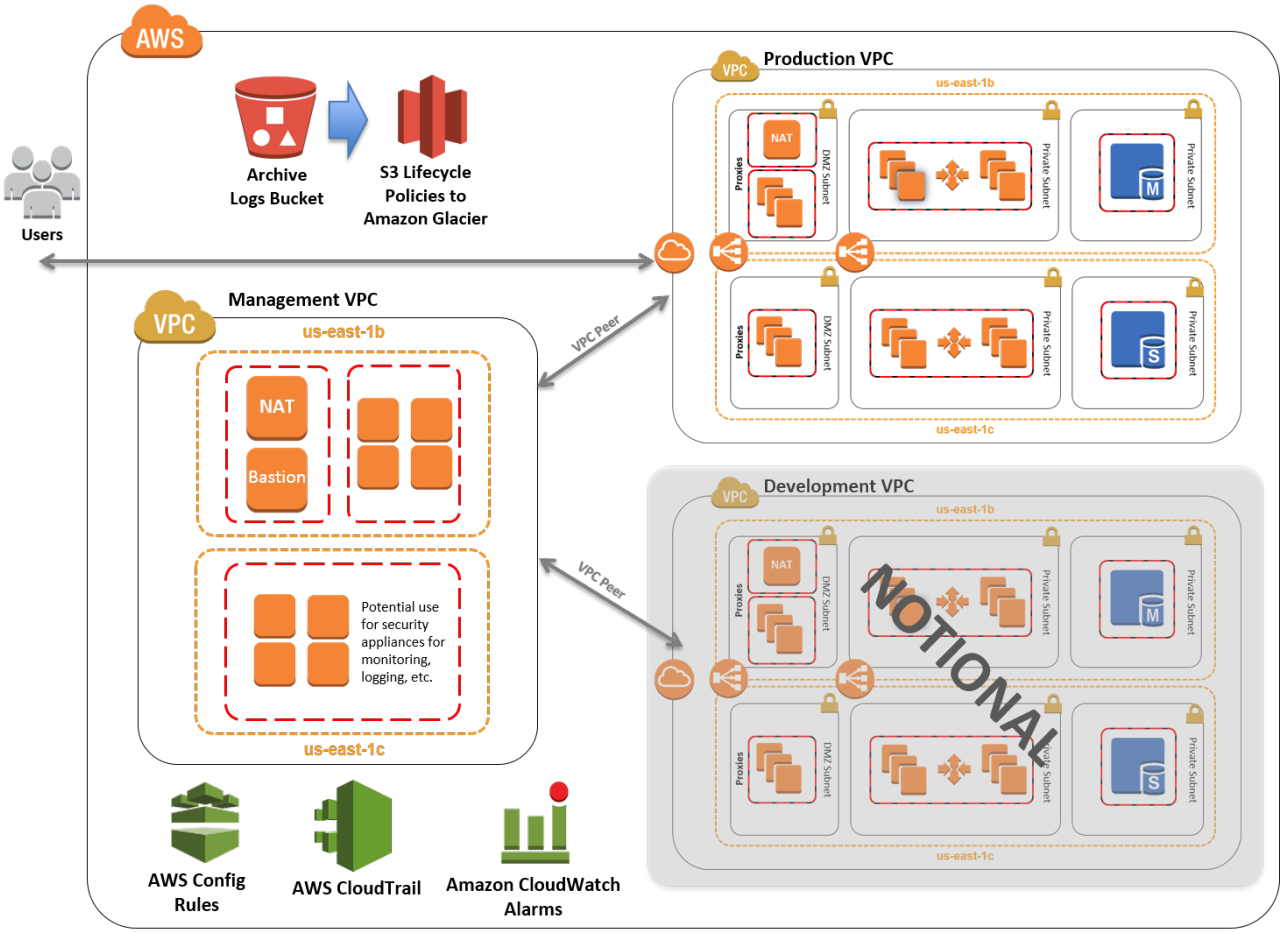
End users access the production network through a load balancer service, which is configured to direct traffic to the application. Application are in private subnets, which only allow traffic from the load balancer. Databases are in separate private subnets, which only allow traffic from their applications. A development network closely matches the production network so testing best reflects the impact on production.

The cloud service provider's identity and access management service restricts management access to designated users with fine-grained permissions. Developers make changes to application and/or infrastructure source code, which is stored in a repository inside the management network. Changes to the source code trigger a pipeline to build the application, perform various tests, and deliver updates to the various environments. Testing includes security analysis of the source code, compiled application before runtime, and the running application. Changes to the production environment are limited to the delivery service – developers do not have direct access to the production application or data.

Logs are collected in a centralized service and transmitted to a security incident and event management platform. Alarms and notifications are configured to alert personnel of potential incidents.

B. Standardized Architecture on AWS

The above architecture may be constructed using AWS services as depicted below.



This architecture demonstrates the following components:

1. Identity and Access Management (IAM) configuration with custom IAM policies, with associated groups, roles, and instance profiles.
2. Standard, external-facing Amazon Virtual Private Cloud (Amazon VPC) Multi-AZ architecture with separate subnets for different application tiers and private (back-end) subnets for application and database. The Multi-AZ architecture helps ensure high availability.
3. Amazon Simple Storage Service (Amazon S3) buckets for encrypted web content, logging, and backup data.
4. Standard Amazon VPC security groups for Amazon Elastic Compute Cloud (Amazon EC2) instances and load balancers used in the sample application stack. The security groups limit access to only necessary services.
5. Three-tier Linux web application using Auto Scaling and Elastic Load Balancing, which can be modified or bootstrapped with customer applications.
6. A secured bastion login host to facilitate command-line Secure Shell (SSH) access to Amazon EC2 instances for troubleshooting and systems administration activities.

7. Encrypted, Multi-AZ Amazon Relational Database Service (Amazon RDS) MySQL database.
8. Logging, monitoring, and alerts.

AWS – Standardized Architecture for National Institute of Standards and Technology (NIST) Compliance[112]

[112] https://aws.amazon.com/quickstart/architecture/compliance-nist/

IV. CONTINUOUS SECURITY

Systems evolve, so security must be maintained and evaluated throughout the system lifecycle. By building security into a system each step of the way, you can prevent major issues being discovered either late in development or, worse, in production. Additionally, by automating security checks, developers can quickly identify security issues and learn from the feedback, developing their skills along the way.

A. Infrastructure Management

The practice of defining Infrastructure as Code (IaC) has improved reliability of infrastructure management in many ways, security included. Teams who uses IaC can improve the clarity, transparency, and stability of their environments. By limiting changes to code, you can also enable the security experts on your teams to evaluate significant changes prior to approval, without slowing development down significantly. Each cloud provider has a native service for Infrastructure as Code and there are third-party options available. Each option involves trade-offs and continues to mature, so you should evaluate alternatives for your specific use case.
AWS – Infrastructure as Code Whitepaper[113]
Azure – What is infrastructure as code?[114]
GCP – Infrastructure as Code[115]

B. Application Build

You should automate application build processes, not only for improved development speed and feedback, but also to integrate security testing. Include static code security analysis tools into your integration processes, and configure them to run frequently, fail builds if serious issues are found, and provide immediate feedback to developers.[116] This feedback is critical to developers understanding what vulnerabilities they may be introducing so they can avoid doing it in the future. Additionally, automated build processes should check application dependencies for vulnerabilities, as many applications rely on a multitude of external libraries.[117]

C. Application Containers

Containerization of applications, though not necessarily limited to cloud computing, has risen along with the adoption of cloud services. Traditional security tools and techniques may not apply to some aspects of containers, so you should evaluate what changes may be required to maintain coverage while using containers.
There are five layers of container security to consider: images, registries, orchestrators, containers, and host operating systems. Consider the following practices:

---

[113] https://d0.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf

[114] https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

[115] https://cloud.google.com/solutions/infrastructure-as-code/

[116] For more, see https://owasp.org/www-community/controls/Static_Code_Analysis

[117] For example, the OWASP Dependency Check tools from http://www.owasp.org/index.php/OWASP_Dependency_Check

1. Security of Images

    a. Adopt tools and processes to validate and enforce compliance with secure configuration policies.[118]
    b. Configure images to run as non-privileged users.
    c. Prohibit the use of remote administration tools and services that provide interactive control within containers (e.g., SSH). Instead, remote management of containers should be performed through container runtime APIs (which may be accessed via orchestration tools), cloud service APIs (if using cloud-based container services), or by creating remote shell sessions to the host on which the container is running.
    d. Continuously monitor all images for embedded malware.
    e. Prohibit the storage of secrets (e.g., access keys) in images, instead programmatically retrieving them at runtime. Container orchestrators may provide this capability natively or images can be configured to use secrets managements tools used for non-container environments.
    f. Ensure that secrets are only provided to the specific containers that require them.
    g. Ensure that secrets are always encrypted at rest and in transit using Federal Information Processing Standard (FIPS) 140 approved cryptographic algorithms.
    h. Maintain a set of trusted images and registries and ensure that only images from this set are run in their environment.
    i. Employ tools that use a pipeline-based build and deployment approach to maintain immutable containers and images.

2. Security of Registries

    a. Configure development tools, orchestrators, and container runtimes to only connect to registries over encrypted channels.
    b. All access to registries that contain proprietary or sensitive images should require authentication.
    c. Any write access to a registry should require authentication to ensure that only images from trusted entities can be added to it.
    d. All write access to registries should be audited and any read actions for sensitive images should similarly be logged.
    e. Consider federating access to container registries with existing access management directories to take advantage of security controls already in place.
    f. Configure continuous integration processes to allow images to be signed by authorized personnel and pushed to a registry only after they have passed a vulnerability scan and compliance assessment.

3. Security of Orchestrators

    a. Choose orchestrators that provide mutually authenticated network connections between cluster members and end-to-end encryption of intra-cluster traffic.
    b. Choose orchestration platforms designed to be resilient to compromise of individual nodes without compromising the overall security of the cluster. A compromised node must be able to be isolated and removed from the cluster without disrupting or degrading overall cluster operations.
    c. Configure orchestrators to use a least privilege access model in which users are only

[118] For example, perform container static analysis using Clair. https://github.com/quay/clair

granted the ability to perform the specific actions on the specific hosts, containers, and images their job roles require.
d. Enforce strong authentication methods, including multi-factor mechanisms, on all access to cluster-wide administrative accounts, as these accounts provide ability to affect all resources in the environment.
e. Configure orchestrators to separate network traffic into discrete virtual networks by sensitivity level. Communication between networks of different sensitivity levels should occur through a limited, well-defined interfaces.
f. Configure orchestrators to isolate deployments to specific sets of hosts by sensitivity levels. This can be accomplished through host 'pinning' within the orchestrator or simply by having separate, individually managed clusters for each sensitivity level.
g. Organizations should implement single sign-on to existing directory systems where applicable.

For Kubernetes applications, consider the practices outlined in the CIS Benchmark[119].

4. Security of Containers

a. Control the egress network traffic sent by containers, similar to the patterns used in traditional architectures. At minimum, ensure containers are not able to send traffic across networks of differing sensitivity levels except using limited, well-defined interfaces.
b. Ensure that containers are run with the default profiles provided by their runtime and should consider using additional profiles for high-risk applications.
c. Implement container-aware security tools that are designed to operate at the scale and change rate typically seen with containers. Existing host-based intrusion detection processes and tools are often unable to detect and prevent attacks within containers due to the differing technical architecture.
d. Run containers with their root filesystems in read-only mode.
e. Institute separate environments for development, test, production, and other scenarios, each with specific controls to provide role-based access control for container deployment and management activities.
f. Ensure all container creation is associated with individual user identities and logged to provide an audit trail of activity.
g. Use tools to look for Common Vulnerabilities and Exposures (CVEs) vulnerabilities in container runtimes deployed and upgrade any instances at risk.
h. Automate compliance with container runtime configuration standards, such as the Center for Internet Security Docker Benchmark.[120]
i. Deploy app-aware network filtering tools to evaluate inter-container traffic.
j. Use security tools that enforce baseline requirements for vulnerability management and compliance prior to allowing an image to be run.

5. Security of Host Operating Systems

a. Use minimalistic operating systems designed for containers whenever possible, to reduce the attack surface and mitigate the typical risks and hardening activities

[119] https://www.cisecurity.org/benchmark/kubernetes/

[120] https://www.cisecurity.org/benchmark/docker/

associated with general-purpose operating systems.

b. Prohibit running containers on hosts that run other applications, like a web server or database, outside of containers.
c. Use tools provided by the operating system vendor or other trusted organizations to regularly check for and apply updates to all software components used within the operating system.
d. Operate host operating systems in an immutable manner with no data or state stored uniquely and persistently on the host and no application-level dependencies provided by the host. Instead, all app components and dependencies should be packaged and deployed in containers.
e. Ensure that all authentication to the OS is audited, login anomalies are monitored, and any escalation to perform privileged operations is logged.
f. Ensure that containers are run with the minimal set of file system permissions required. Any file changes that containers need to persist to disk should be made within storage volumes specifically allocated for this purpose. In no case should containers be able to mount sensitive directories on a host's file system.
g. Employ tools that use a pipeline-based build and deployment approach to maintain immutable containers, rather than updating existing systems.

D. Application Delivery

Your application should be delivered through tiers of environments with different levels of testing. For example, an application may promote through development, staging, and production environments. The development environment includes dynamic/runtime security testing to quickly identify potential vulnerabilities and provide feedback to developers. The staging environment includes security configuration checks, which are effective because staging closely mirrors the production environment, although the resources are smaller to reduce cost.

V. QUESTIONS

Questions regarding this guide may be sent to:

California Department of Technology
Office of Information Security
Security@state.ca.gov