

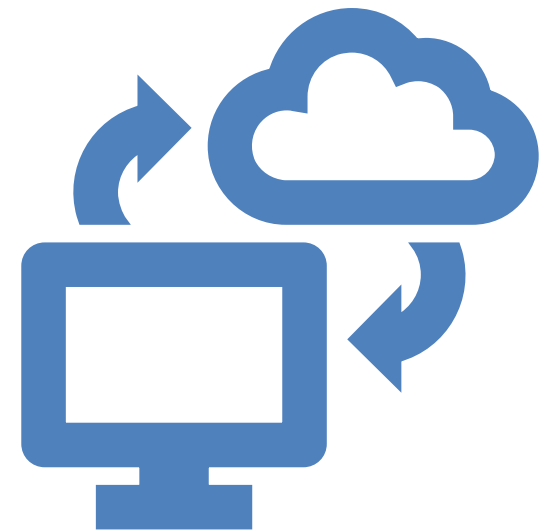


Dr. Chuck Easttom, M.Ed, MSDS, MBA, MSSE, Ph.D.², D.Sc.

What is Cloud Computing?

Cloud Computing is a general term used to describe a new class of network based computing that takes place over the Internet, a collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform). Using the Internet for communication and transport provides hardware, software and networking services to clients

These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API (Applications Programming Interface).



What is cloud computing?

Cloud computing means storing and accessing data and programs over the Internet instead of your computer's hard drive. – PC Mag

Cloud computing is the delivery of computing services—servers, storage, databases, networking, software, analytics, and more—over the Internet (“the cloud”). – Microsoft

Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing. –Amazon

NIST

The National Institute of Standards and Technology (NIST) defines cloud computing as “a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

Cloud Types

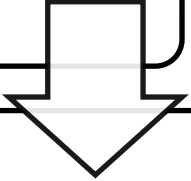
Public clouds are defined by the NIST as simply clouds that offer their infrastructure or services to either the general public or at least a large industry group.

Private clouds are those clouds used specifically by a single organization without offering the services to an outside party. There are of course hybrid clouds which combine the elements of a private and public cloud. These are essentially private clouds that have some limited public access.

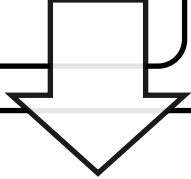
Community clouds is a midway point between private and public. These are systems wherein several organizations share a cloud for specific community needs. For example several computer companies might join to create a cloud devoted to common security issues.

Basic Cloud Concepts

In all clouds, someone else is providing the physical machines

A white downward-pointing arrow with a black outline, indicating a flow from the first concept to the second.

You aren't concerned about power, bandwidth, maintenance, physical security, or (sometimes) scaling

A white downward-pointing arrow with a black outline, indicating a flow from the second concept to the third.

You only pay for what you use

Multi-cloud

Multiple different cloud vendors are used heterogeneously. This mitigates dependency on a single vendor. Cloud assets (applications, virtual servers, etc.) are hosted across multiple different public clouds. One can also include private clouds in the architecture.

Poly cloud is similar, but in this case the different public clouds are being utilized not for flexibility and redundancy, but rather for specific services each provider offers.



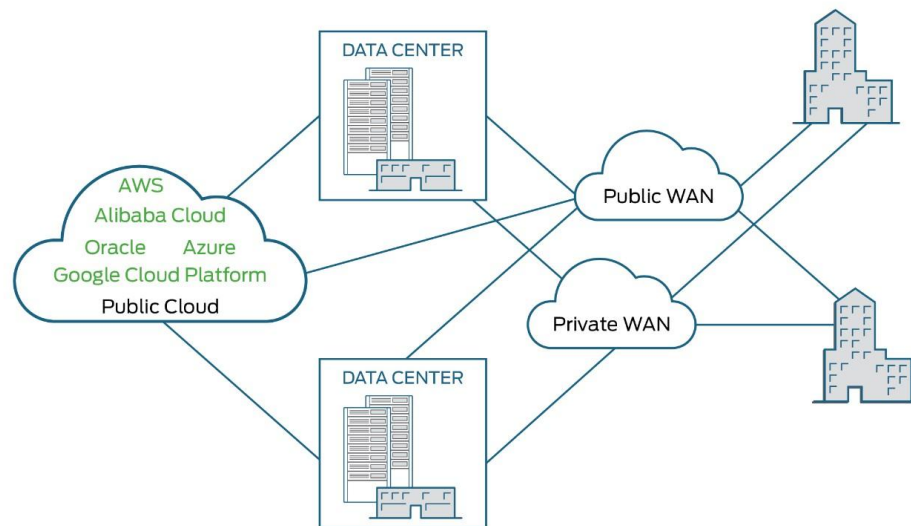
Multi cloud – Hybrid Cloud - Intercloud

Multicloud: As mentioned above, multicloud is a setup where multiple cloud vendors are used by a single organization. Ideally, the clouds offer interoperability and connectivity to ensure that users can easily access both. However, this is one of the most important challenges in setting up a successful multicloud. An easy way to think about multicloud is on your TV: Netflix, Amazon Prime, and HBO Max are all different cloud streaming services subscribed by an individual household.

Hybrid cloud: Hybrid clouds use a combination of on-premises hardware and cloud hosting. There are a number of reasons why an enterprise may choose a hybrid cloud. This may be part of an incremental transition to a public cloud. This may also be a regulatory necessity by keeping sensitive material on the on-premise hardware.

Intercloud: While a multicloud brings two or more different clouds together for use within the same organization, an intercloud configuration actively merges data within multiple cloud solutions. This is similar to a hybrid cloud environment in that two or more systems are working together, but the difference here stems from the fact that all systems reside in the cloud.

Multi cloud - Juniper



- Multicloud is a cloud computing deployment model that enables organizations to deliver application services across multiple private and public clouds containing some or any combination of the following: multiple cloud vendors, multiple cloud accounts, multiple cloud availability zones, or multiple cloud regions or premises.

• <https://www.juniper.net/us/en/research-topics/what-is-multicloud.html>

Multi-cloud issues

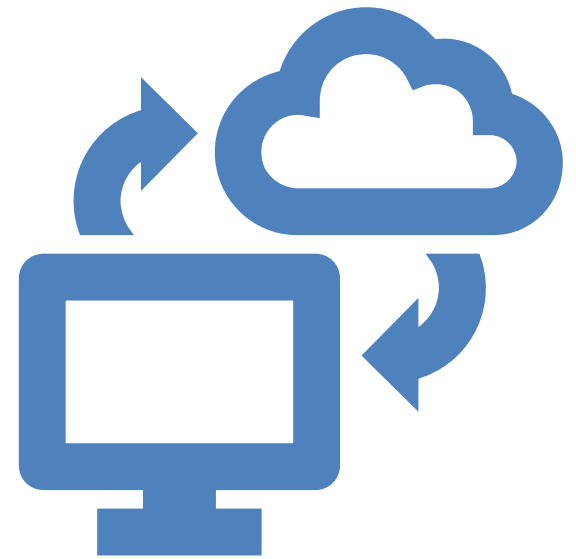
1. Configuration Errors
2. User Access Controls
3. Workload Freshness
4. Visibility
5. Application Hardening
6. Data Governance
7. Shared Security Model

<https://www.copado.com/devops-hub/blog/7-multi-cloud-security-challenges-and-how-to-combat-them>

HPC Cloud

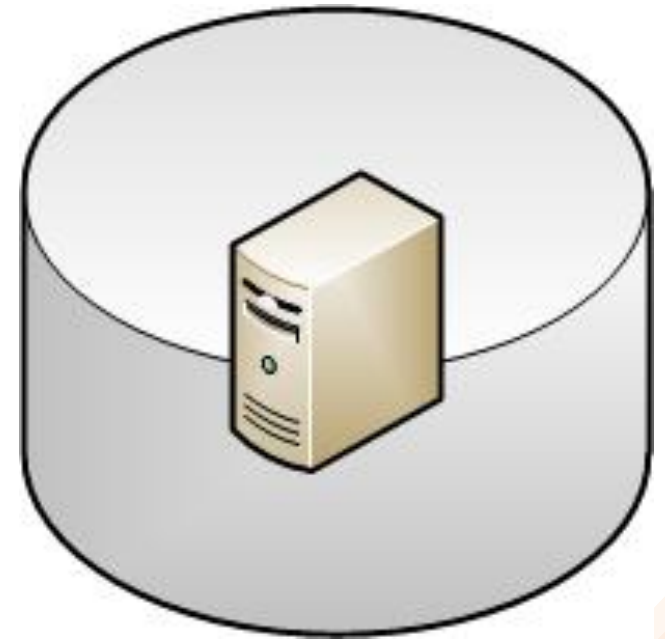
HPC Cloud

This is the use of cloud services for high performance computing. Such HPC applications would normally require clusters of computers or a supercomputer. There are several companies including Amazon Web Services that offer HPC cloud computing.



Virtualization

Started in 1967 with the IBM CP-40
Virtual machine (VM) software is a program that emulates a physical machine
A VM should behave as if it were an independent physical machine. You see this on desktops with Oracle Virtual Box and VMWare workstations



Virtual Systems

Virtual Machine

Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)

Desktop as a Service (DaaS)

Information Technology
Management as a Service
(ITMaaS).



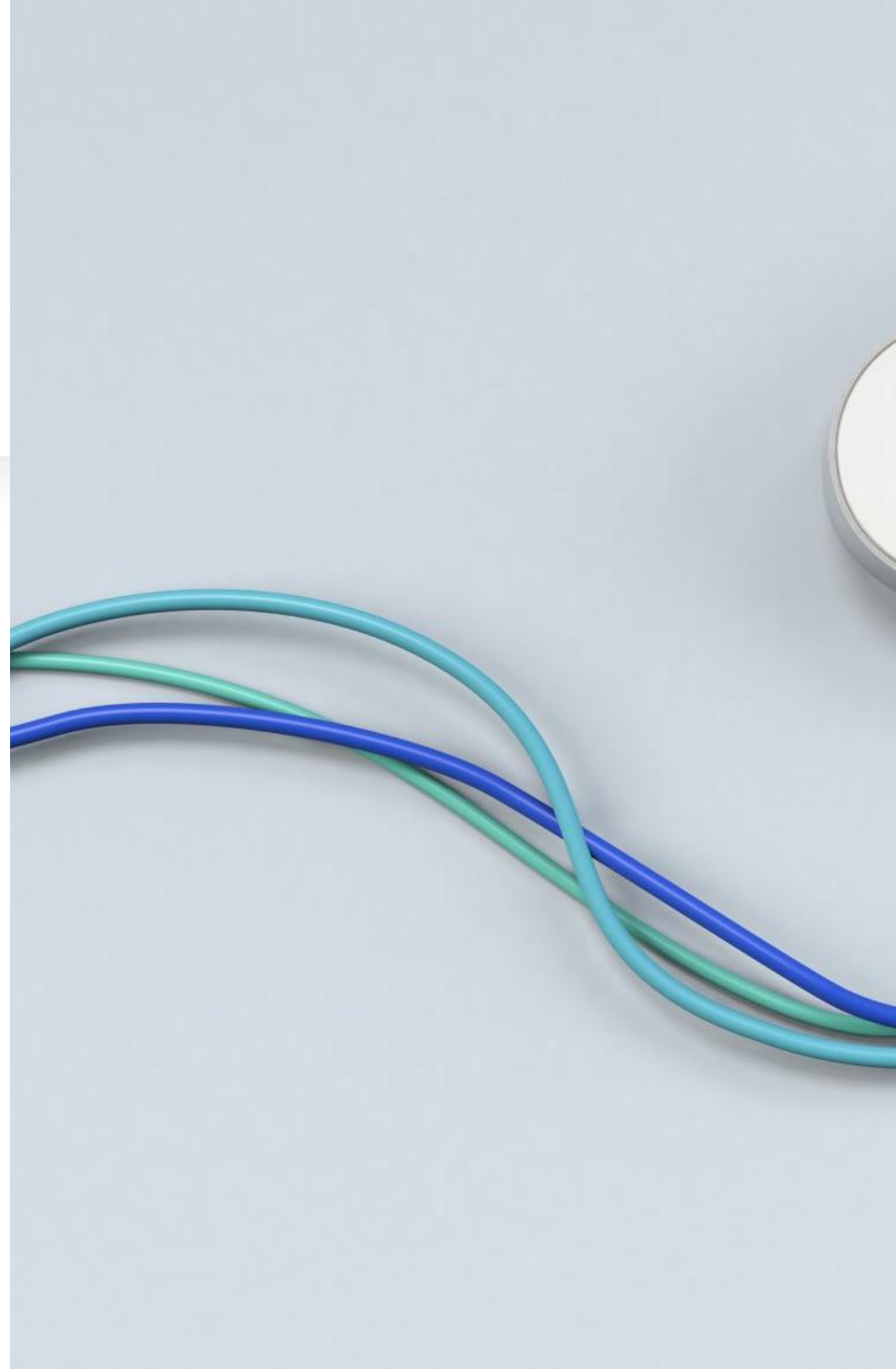
IAAS

You can't tell if you are on a cloud machine or not

From the perspective of the software (or an administrator), a cloud machine is identical to a physical machine. The process is:

1. Determine your operating system
2. Determine how much computing you need
3. Find an instance in your cloud provider library of machines
4. Start that instance

Automatically scale up / down machines as needed





PaaS

CPU resources scale up or down as needed

No need to spin up new machines, manage load balancing, etc.

There are several types of PaaS, including public, private and hybrid.

There are variations such as Communications Platform as a Service (CPaaS) and Mobile Platform as a Service (mPaaS).

SaaS

Basically renting an application instead of setting it up on your own server.

Usually users access SaaS apps via some thin client, often web browsers. There are a wide range of applications available in this fashion. The applications are provided by application service providers (ASP). There are subsets of SaaS such as DbaaS (database as a service)

There are two main variations of SaaS:

Vertical SaaS

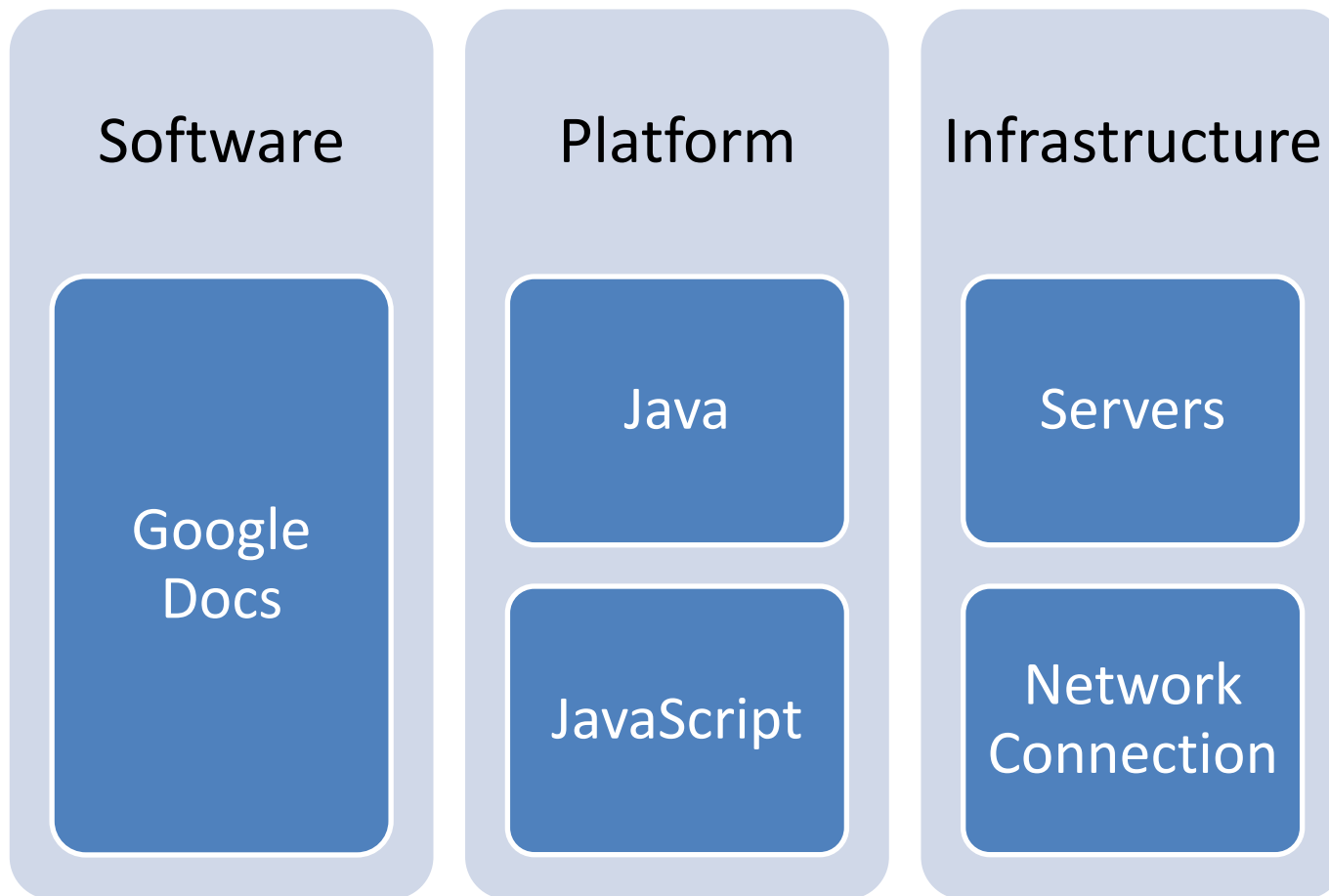
Software which is for a specific industry such as healthcare, finance, etc.

Horizontal SaaS

The products which focus on a particularly category of software such as software development, sales, etc., but is not for a particular industry

OpenSaaS refers to software as a service (SaaS) based on open source code.

Example SaaS: Google Docs



Variations

There are numerous variations such as
Security as a service (SECaaS or SaaS)

Knowledge as a service (KaaS)

data as a service (DaaS)

Mobile backend as a service (MBaaS)

Artificial intelligence as a service (AlaaS)

Content as a Service (CaaS)

These are typically specialized variation of PaaS, IaaS , SaaS.

Critical Security Areas in Cloud Computing (CSA)

Governing in the Cloud

Governance and Enterprise Risk Management

Legal and Electronic Discovery

Compliance and Audit

Information Lifecycle Management

Portability and Interoperability

Operating in the Cloud

Traditional Security, Business Continuity, and Disaster Recovery

Data Center Operations

Incident Response, Notification, and Remediation

Application Security

Encryption and Key Management

Identity and Access Management

Virtualization



Top 10 Customer Issues Eroding Cloud Confidence (from CSA)

Government regulations keeping pace with the market

Exit strategies

International data privacy

Legal issues

Contract lock in

Data ownership and custodian responsibilities

Longevity of suppliers

Integration of cloud with internal systems

Credibility of suppliers

Testing and assurance

ISO 27017

ISO 27017 is guidance for cloud security. It does apply the guidance of ISO 27002 to the cloud, but then adds 7 new controls.

CLD.6.3.1: Agreement on shared or divided security responsibilities between the customer and cloud provider

CLD.8.1.5: Addresses how assets are returned or removed from the cloud when the contract is terminated

CLD.9.5.1: This control states that the cloud provider must separate the customers virtual environment from other customers or outside parties.

CLD.9.5.2: This control states that the customer and the cloud provider both must ensure the virtual machines are hardened.

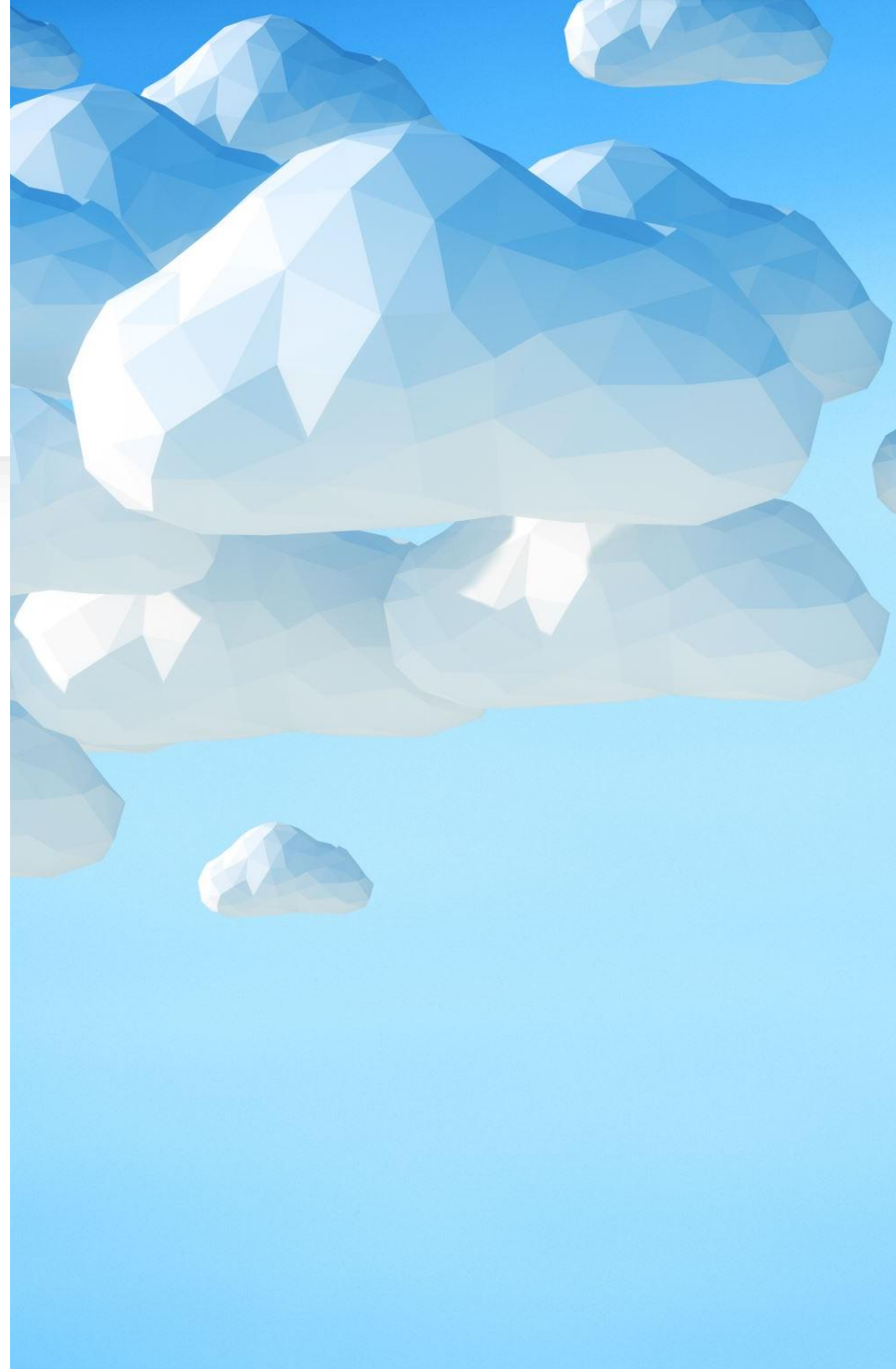
CLD.12.1.5: It is solely the customer's responsibility to define and manage administrative operations.

CLD.12.4.5: The cloud providers capabilities must enable the customer to monitor their own cloud environment.

CLD.13.1.4: The virtual network environment must be configured so that it least meets the security policies of the physical environment.

ISO 27018

ISO 27018 is closely related to ISO 27017. ISO 27018 defines privacy requirements in a cloud environment. Particularly how the customer and cloud provider must protect personally identifiable information (PII)



NSA Guidance

The NSA offers guidance on cloud security

https://media.defense.gov/2020/Jan/22/2002237484/-1/-1/0/CSI-MITIGATING-CLOUD-VULNERABILITIES_20200121.PDF

While not a base component of cloud architectures, encryption and key management (KM) form a critical aspect of protecting information in the cloud.

While CSPs are generally responsible for detecting threats to the underlying cloud platform, customers bear the responsibility of detecting threats to their own cloud resources.

Incident Response: CSPs are uniquely positioned to respond to incidents internal to the cloud infrastructure and bear responsibility for doing so. Incidents internal to customer cloud environments are generally the customer's responsibility, but CSPs may provide support to incident response teams.

Patching/Updating: CSPs are responsible for ensuring that their cloud offerings are secure and rapidly patch software within their purview but usually do not patch software managed by the customer (e.g., operating systems in IaaS offerings). Because of this, customers should vigilantly deploy patches to mitigate software vulnerabilities in the cloud. In some cases CSPs offer managed solutions in which they perform operating system patching as well.



NSA Guidance

Cloud Threat Actors

Threat actors may target the same types of weaknesses in both cloud and traditional system architectures. This section focuses on cloud-specific activities, but administrators should be aware that traditional tactics still apply. For example, an unpatched web application in the cloud bears similar risk of compromise as one served from an on-premises network. The following threat actors are relevant to cloud computing:

Malicious CSP Administrators

- Leverage privileged credentials or position to access, modify, or destroy information stored on the cloud platform;
- Leverage privileged credentials or position to modify the cloud platform in order to gain access to networks connected to or consuming cloud resources;

Malicious Customer Cloud Administrators

- Leverage privileged credentials to access, modify, or destroy information stored on the cloud platform;

Cyber Criminals and/or Nation State-Sponsored Actors

- Leverage a weakness in the cloud architecture or configuration to obtain sensitive data or consume cloud resources at the victim's expense;
- Exploit weak cloud-based authentication mechanisms to obtain user credentials (e.g., password spray attacks);
- Leverage compromised credentials or incorrect access privileges to gain access to cloud resources;
- Gain privileged access to the cloud environment to compromise tenant resources;
- Leverage the trust relationship between an organization's networks and cloud resources to pivot from clouds into protected networks or vice versa;

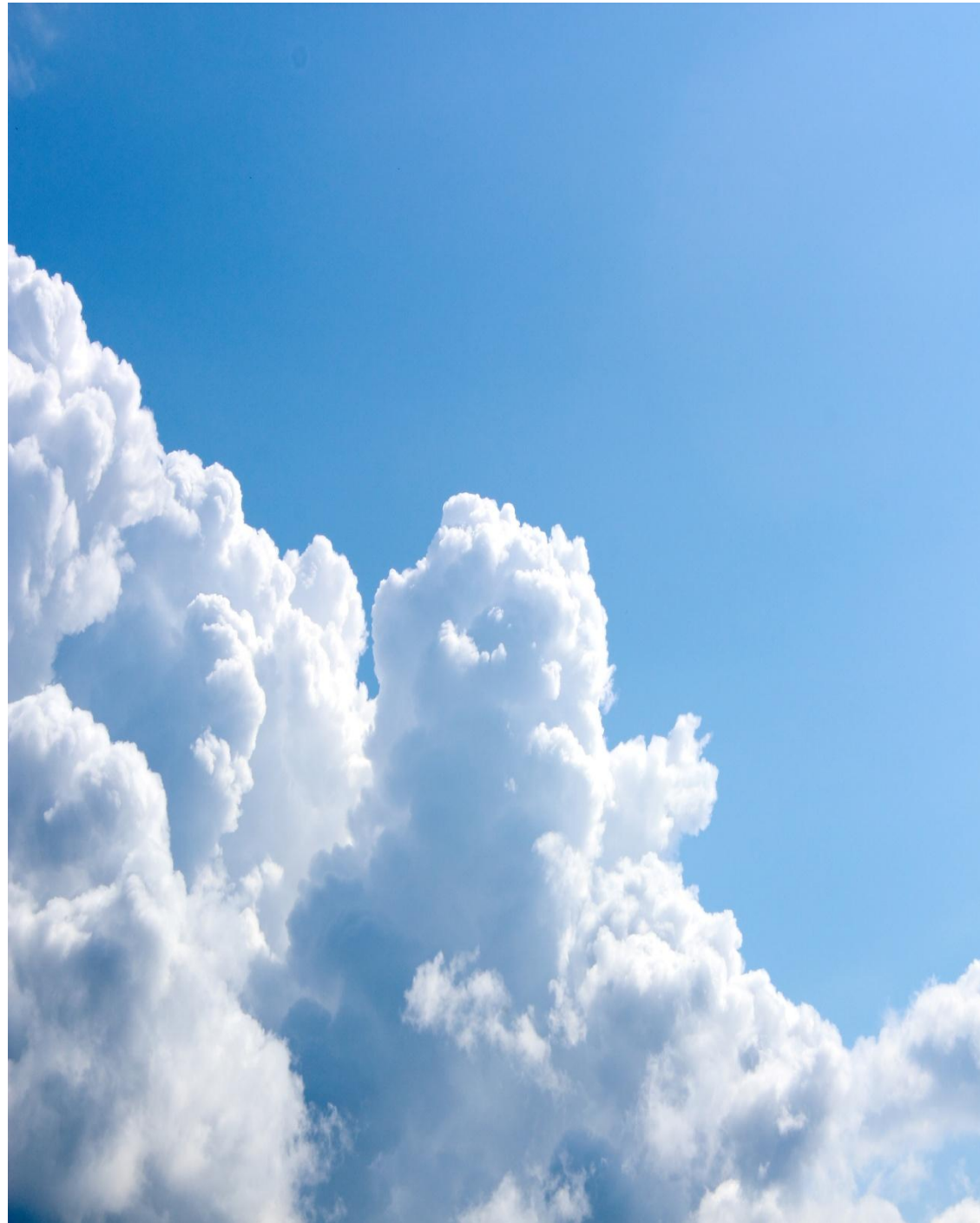
Untrained or Neglectful Customer Cloud Administrators

- Expose sensitive data or cloud resources unintentionally.

Cloud DDoS Defense

a Google Cloud Armor customer was targeted with a series of HTTPS DDoS attacks which peaked at 46 million requests per second. This is the largest Layer 7 DDoS reported to date—at least 76% larger than the previously reported record.

<https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps>



TeamTNT

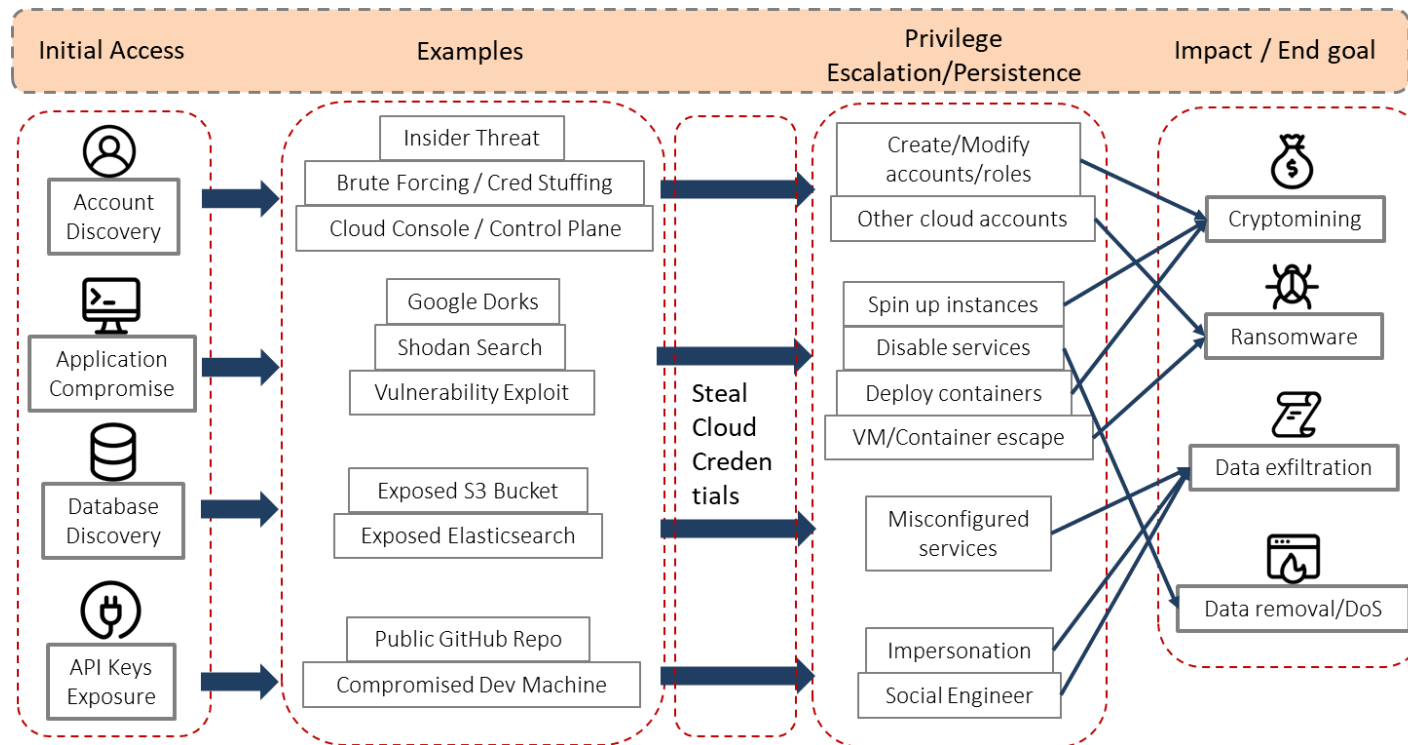


Attacking containers in the cloud
<https://www.avertium.com/blog/teamtnt-attacks-cloud-environments>

“TeamTNT is using compromised AWS credentials to attack AWS cloud environments via the cloud platform’s application programming interface. The threat actors are now also targeting the credentials of 16 additional applications, including the AWS apps as well as Google Cloud credentials”

<https://www.scmagazine.com/news/cloud/teamtnt-attacks-iam-credentials-of-aws-and-google-cloud>

TrendMicro – Cloud Attack Overview



https://www.trendmicro.com/en_us/research/21/i/1h-2021-security-review-shows-active-cloud-attacks.html

Palo Alto Networks Cloud Security Report

"...our latest research shows that threat actors are finding ways to take advantage of that trust to make their attacks extremely difficult to detect and prevent. The latest campaigns conducted by an advanced persistent threat (APT) that we track as Cloaked Ursa (also known as APT29, Nobelium or Cozy Bear) demonstrate sophistication and the ability to rapidly integrate popular cloud storage services to avoid detection."

<https://unit42.paloaltonetworks.com/cloaked-ursa-online-storage-services-campaigns/>

Microsoft Nobelium Report

(MSP), and other IT services organizations (referred to as “service providers” for the rest of this blog) that have been granted administrative or privileged access by other organizations. The targeted activity has been observed against organizations based in the United States and across Europe since May 2021. MSTIC assesses that NOBELIUM has launched a campaign against these organizations to exploit existing technical trust relationships between the provider organizations and the governments, think tanks, and other companies they serve. NOBELIUM is the same actor behind the SolarWinds compromise in 2020, and this latest activity shares the hallmarks of the actor’s compromise-one-to-compromise-many approach. Microsoft has notified known victims of these activities through our nation-state notification process and worked with them and other industry partners to expand our investigation, resulting in new insights and disruption of the threat actor throughout stages of this campaign.

<https://www.microsoft.com/security/blog/2021/10/25/nobelium-targeting-delegated-administrative-privileges-to-facilitate-broader-attacks/>

IBM Force Report

—

Two-thirds of cloud security incidents could have been avoided if the configuration of apps, databases, and security policies were correct

"These two elements trickled down to the most frequently observed initial infection vectors for organizations: improperly configured assets, password spraying, and pivoting from on-premises infrastructure," IBM says. "In addition, API configuration and security issues, remote exploitation and accessing confidential data were common ways for threat actors to take advantage of lax security in cloud environments."

<https://www.zdnet.com/article/two-thirds-of-cloud-attacks-could-be-stopped-by-checking-configurations-research-finds/>

OWASP Cloud-Native Application Security Top 10

- **CNAS-1: Insecure cloud, container or orchestration configuration**
- **CNAS-2: Injection flaws (app layer, cloud events, cloud services)**
- **CNAS-3: Improper authentication & authorization**
- **CNAS-4: CI/CD pipeline & software supply chain flaws**
- **CNAS-4: CI/CD pipeline & software supply chain flaws**
- **CNAS-6: Over-permissive or insecure network policies**
- **CNAS-7: Using components with known vulnerabilities**
- **CNAS-8: Improper assets management**
- **CNAS-9: Inadequate 'compute' resource quota limits**
- **CNAS-10: Ineffective logging & monitoring (e.g. runtime activity)**

OWASP Top 10 Cloud Security Risks (Draft)

- R1. Accountability & Data Risk
- R2. User Identity Federation
- R3. Legal & Regulatory Compliance
- R4. Business Continuity & Resiliency
- R5. User Privacy & Secondary Usage of Data
- R6. Service & Data Integration
- R7. Multi-tenancy & Physical Security
- R8. Incidence Analysis & Forensics
- R9. Infrastructure Security
- R10. Non-production Environment Exposure

[https://owasp.org/www-pdf-archive/OWASP Cloud Top 10.pdf](https://owasp.org/www-pdf-archive/OWASP_Cloud_Top_10.pdf)

Checkpoint Cloud Security

- Misconfiguration
- Unauthorized Access
- Insecure Interfaces/APIs
- Hijacking of Accounts
- Lack of Visibility
- External Sharing of Data
- Insider threats
- Cyberattacks
- Denial of Service Attacks

<https://www.checkpoint.com/cyber-hub/cloud-security/what-is-cloud-security/top-cloud-security-issues-threats-and-concerns/>

FedRAMIP

The Federal Risk and Authorization Management Program (FedRAMP) is a government-wide program that provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services. Third-party assessment organizations (3PAOs) play a critical role in the FedRAMP security assessment process, as they are the independent assessment organizations that verify cloud providers' security implementations and provide the overall risk posture of a cloud environment for a security authorization decision

<https://www.fedramp.gov/>

NIST Special Publication 800-144, Guidelines on Security and Privacy in Public Cloud Computing

- NIST Special Publication 800-144, **Guidelines on Security and Privacy in Public Cloud Computing**, December 2011
- NIST Special Publication 800-145, **NIST Definition of Cloud Computing**, September 2011
- NIST Special Publication 800-146, **Cloud Computing Synopsis and Recommendations**, May 2012
- NIST Special Publication 500-291, **NIST Cloud Computing Standards Roadmap**, July 2011
- NIST Special Publication 500-292, **NIST Cloud Computing Reference Architecture**, September 2011
- NIST Special Publication 500-299, **NIST Cloud Computing Security Reference Architecture (Draft)**

NIST Special Publication 800-144, Guidelines on Security and Privacy in Public Cloud Computing

- This standard emphasizes the importance of the service level agreement (section 3.1).
- NIST 800-144 discusses governance as a security issue (section 4.1)
- Virtual Network Protection is also emphasized (section 4.4)
Authentication is addressed (section 4.5).
Many cloud providers are using SAML (We will discuss SAML in some depth later in this workshop)

**NIST Special Publication
800-144, Guidelines on
Security
and Privacy in Public Cloud
Computing**

SLA should cover:

Personnel requirements, including clearances, roles, and responsibilities

Regulatory requirements

Service availability

Problem reporting, review, and resolution

Information handling and disclosure agreements and procedures

Physical and logical access controls

Network access control, connectivity, and filtering

Data protection

System configuration and patch management

Backup and recovery

Data retention and sanitization

Security and vulnerability scanning

Risk management

Incident reporting, handling, and response

Continuity of operations

Resource management

Certification and accreditation

Assurance levels

Independent auditing of services

NIST Special Publication 800-144, Guidelines on Security and Privacy in Public Cloud Computing

Table 1: Security and Privacy Issues and Recommendations

Areas	Recommendations
Governance	<p>Extend organizational practices pertaining to the policies, procedures, and standards used for application development and service provisioning in the cloud, as well as the design, implementation, testing, use, and monitoring of deployed or engaged services.</p> <p>Put in place audit mechanisms and tools to ensure organizational practices are followed throughout the system lifecycle.</p>
Compliance	<p>Understand the various types of laws and regulations that impose security and privacy obligations on the organization and potentially impact cloud computing initiatives, particularly those involving data location, privacy and security controls, records management, and electronic discovery requirements.</p> <p>Review and assess the cloud provider's offerings with respect to the organizational requirements to be met and ensure that the contract terms adequately meet the requirements.</p> <p>Ensure that the cloud provider's electronic discovery capabilities and processes do not compromise the privacy or security of data and applications.</p>
Trust	<p>Ensure that service arrangements have sufficient means to allow visibility into the security and privacy controls and processes employed by the cloud provider, and their performance over time.</p> <p>Establish clear, exclusive ownership rights over data.</p> <p>Institute a risk management program that is flexible enough to adapt to the constantly evolving and shifting risk landscape for the lifecycle of the system.</p> <p>Continuously monitor the security state of the information system to support on-going risk management decisions.</p>

NIST Special Publication 800-144, Guidelines on Security and Privacy in Public Cloud Computing

	Recommendations
re	Understand the underlying technologies that the cloud provider uses to services, including the implications that the technical controls involved security and privacy of the system, over the full system lifecycle and ac components.
id Access ent	Ensure that adequate safeguards are in place to secure authentication, and other identity and access management functions, and are suitable for organization.
solution	Understand virtualization and other logical isolation techniques that the employs in its multi-tenant software architecture, and assess the risks in organization.
action	<p>Evaluate the suitability of the cloud provider's data management solutio organizational data concerned and the ability to control access to data, while at rest, in transit, and in use, and to sanitize data.</p> <p>Take into consideration the risk of collating organizational data with the organizations whose threat profiles are high or whose data collectively r significant concentrated value.</p> <p>Fully understand and weigh the risks involved in cryptographic key man facilities available in the cloud environment and the processes establish provider.</p>
/	<p>Understand the contract provisions and procedures for availability, data recovery, and disaster recovery, and ensure that they meet the organiza and contingency planning requirements.</p> <p>Ensure that during an intermediate or prolonged disruption or a serious operations can be immediately resumed, and that all operations can be reinstated in a timely and organized manner.</p>
esponse	<p>Understand the contract provisions and procedures for incident respons that they meet the requirements of the organization.</p> <p>Ensure that the cloud provider has a transparent response process in pl mechanisms to share information during and after an incident.</p> <p>Ensure that the organization can respond to incidents in a coordinated f cloud provider in accordance with their respective roles and responsibili computing environment.</p>

**NIST Special Publication
800-144, Guidelines on
Security
and Privacy in Public
Cloud Computing**

- The standard lists specific concerns:
- Inadequate Policies and Practices.
- Weak Confidentiality and Integrity Sureties.
- Weak Availability Sureties.
- Principal-Agent Problem
- Attenuation of Expertise.

Infrastructure as Code

IaC treats all infrastructure pieces as code, using different programming languages or tools. By codifying infrastructure, IaC allows for automation, consistency, and repeatable processes in deploying and managing resources. This approach not only reduces human error but also enhances efficiency and scalability

“The management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning that the DevSecOps team uses for source code. Infrastructure as Code evolved to solve the problem of environment drift in the release pipeline.”

-DoD Enterprise DevSecOpsReference Design Version 1.0 12 August 2019

Infrastructure as Code

Infrastructure as Code (IaC) is infrastructure definition and configuration that is defined with text files that are checked-in to a source code repository and kept under configuration management. It includes the management of networks, storage, virtual machines, load balancers, and even connection topologies. IaC evolved to solve a real-world problem referred to as environment drift in the release pipeline. Succinctly, the development environment fails to align with the production environment configuration. The goal is to automate all infrastructure provisioning and configuration in a repeatable, consistent way that also lends itself to peer reviews of the changes prior to any configuration changes actually being made.

IaC can take many forms. One is a template for instantiating a cloud service in a secure way. Another is through configuration files or scripts. It is important to consider vendor lock-in versus product lock-in when selecting technology or IaC formats. Blueprints and Cloud Formation only apply to Microsoft Azure and Amazon Web Services (AWS) respectively, creating a degree of vendor lock-in; Cloud-agnostic solutions, such as those provided by popular tools like Ansible and Terraform, avoid vendor lock-in but create product lock-in. In all cases, the IaC is specified via one or more text files.

Infrastructure as Code

Infrastructure as Code (IaC) is infrastructure definition and configuration that is defined with text files that are checked-in to a source code repository and kept under configuration management. It includes the management of networks, storage, virtual machines, load balancers, and even connection topologies. IaC evolved to solve a real-world problem referred to as environment drift in the release pipeline. Succinctly, the development environment fails to align with the production environment configuration. The goal is to automate all infrastructure provisioning and configuration in a repeatable, consistent way that also lends itself to peer reviews of the changes prior to any configuration changes actually being made.

GitOps is a paradigm where systems are described and observed declaratively, using code to specify the desired state. The benefits of GitOps build upon IaC, emphasizing the role of git and a git driven workflow. IaC is one of the three core practices of GitOps, along with merge requests and the reliance upon a CI/CD pipeline.

DevSecOps Fundamentals Playbook March 2021 Version 2.0

Infrastructure as Code

Infrastructure-as-Code (IaC) scanning is the process of creating the code and configuration files that are used to manage and provision infrastructure resources, such as virtual machines, networks, and storage, in a cloud environment. IaC is a method of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Sehgal, Vandana Verma. Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST (p. 257). Packt Publishing. Kindle Edition.

Infrastructure As Code

Part of the “DevOps” paradigm of infrastructure management.

Treats infrastructure as if it’s application code.

All infrastructure changes are committed to revision control (Git, SVN, etc).

Infrastructure is “self-documenting” as all details are contained in the source code.

Benefits of IaC in DevSecOps

Automation and consistency: IaC allows us to automate infrastructure provisioning and management, ensuring a consistent and reproducible environment. This consistency is crucial in DevSecOps, where we need to ensure that security controls are uniformly applied. For example, by defining security groups and rules in an IaC script, we can ensure that these settings are consistently applied across all instances, reducing the chance of human error leading to a security vulnerability.

Speed and agility: IaC allows rapid provisioning and de-provisioning of environments, which supports the agile methodologies used in DevSecOps. For example, developers can quickly spin up a replica of the production environment to reproduce and fix a security bug, then tear it down again when done, minimizing the cost and effort involved.

Version control and auditability: With IaC, the infrastructure is defined in code and can be version-controlled, just like application code. This provides a clear audit trail of what changes were made, when, and by whom, which is valuable for security audits and incident response. If a security incident occurs, we can easily review the infrastructure code to see what changed and potentially led to the incident.

Collaboration and communication: IaC helps to break down silos between development, operations, and security teams by providing a common language and set of tools. This supports the core DevSecOps principle of fostering collaboration and shared responsibility for security. For instance, a security expert could review a Terraform script as part of a pull request, providing feedback directly in the developers' workflow.

Policy as code: Just as IaC allows us to manage infrastructure as code, we can also manage security policies as code. This includes access controls, network policies, and compliance rules. By defining these policies as code, we can automate their enforcement and reduce non-compliance risk. For example, we could use a tool like Open Policy Agent to enforce policies, such as all S3 buckets must be private in our IaC scripts.

Early detection and remediation of issues: By integrating IaC with CI/CD pipelines and automated testing, we can catch and fix security issues early in the development cycle, before they reach production.

IaC

Infrastructure as Code (IaC) Best Practices:

Crafting Secure and Compliant Digital Blueprints In the field of modern infrastructure management, Infrastructure as Code (IaC) emerges as a cornerstone, providing a systematic and scalable approach to provisioning and managing resources. This section delves into the essential best practices of IaC, emphasizing the paramount importance of secure and compliant code in the digital blueprints that define an organization's infrastructure.

1. **Version Control for IaC: Git and Versioning:** Leveraging version control systems, such as Git, is foundational for IaC. Maintaining a comprehensive history of changes ensures traceability, collaboration, and the ability to roll back to known-good states. **Branching Strategies:** Adopting effective branching strategies facilitates parallel development efforts. Feature branches, release branches, and a robust merging strategy contribute to organized and controlled changes in the IaC codebase.

2. **Code Reviews in IaC Development: Collaborative Code Reviews:** Code reviews are critical for identifying security vulnerabilities, misconfigurations, and adherence to best practices. Enabling collaborative code reviews ensures that IaC blueprints benefit from the collective knowledge and expertise of the development team. **Automated Code Analysis:** Integrating automated tools for static code analysis during code reviews enhances the efficiency of identifying issues. These tools can analyse IaC scripts for security vulnerabilities and compliance with coding standards.

3. **Testing and Validation: Unit Testing for IaC:** Implementing unit tests for IaC scripts verifies the correctness of individual components. Unit tests serve as a proactive measure to catch errors early in the development cycle, reducing the likelihood of deploying insecure configurations. **Integration Testing:** Conducting integration tests ensures that various components of the IaC ecosystem work seamlessly together. Integration testing encompasses the entire infrastructure stack, uncovering potential issues arising from the interaction between different components.

IFEROUDJENE, IZEM. DevSecOps: A complete implementation guide (Cyber Security Book 3)

Infrastructure as Code

Traditionally, organizations have provisioned their own hardware, onto which they install operating systems and configure services to support their business value applications hosted within these environments. The arrival of cloud computing has changed this model. Instead of procuring and hosting hardware in their own data centers, organizations can take advantage of cloud service providers who manage the hardware on their behalf. There are three main types of cloud services:

- Software as a service (SaaS), in which third-party applications are hosted in the cloud and accessed by users through a browser (eg Google Docs)
- Platform as a service (PaaS) provides a platform consisting of an infrastructure, OS and services that can be used to build software (eg OpenShift)
- Infrastructure as a service (IaaS) offers the same capabilities as an on-premise data centre without the need to physically maintain the hardware, meaning that organizations are responsible for the OS, middleware and data hosted in the infrastructure

Wilson, Glenn. DevSecOps: A leader's guide to producing secure software without compromising flow, feedback and continuous improvement (pp. 188-189). Rethink Press. Kindle Edition.

Infrastructure as Code

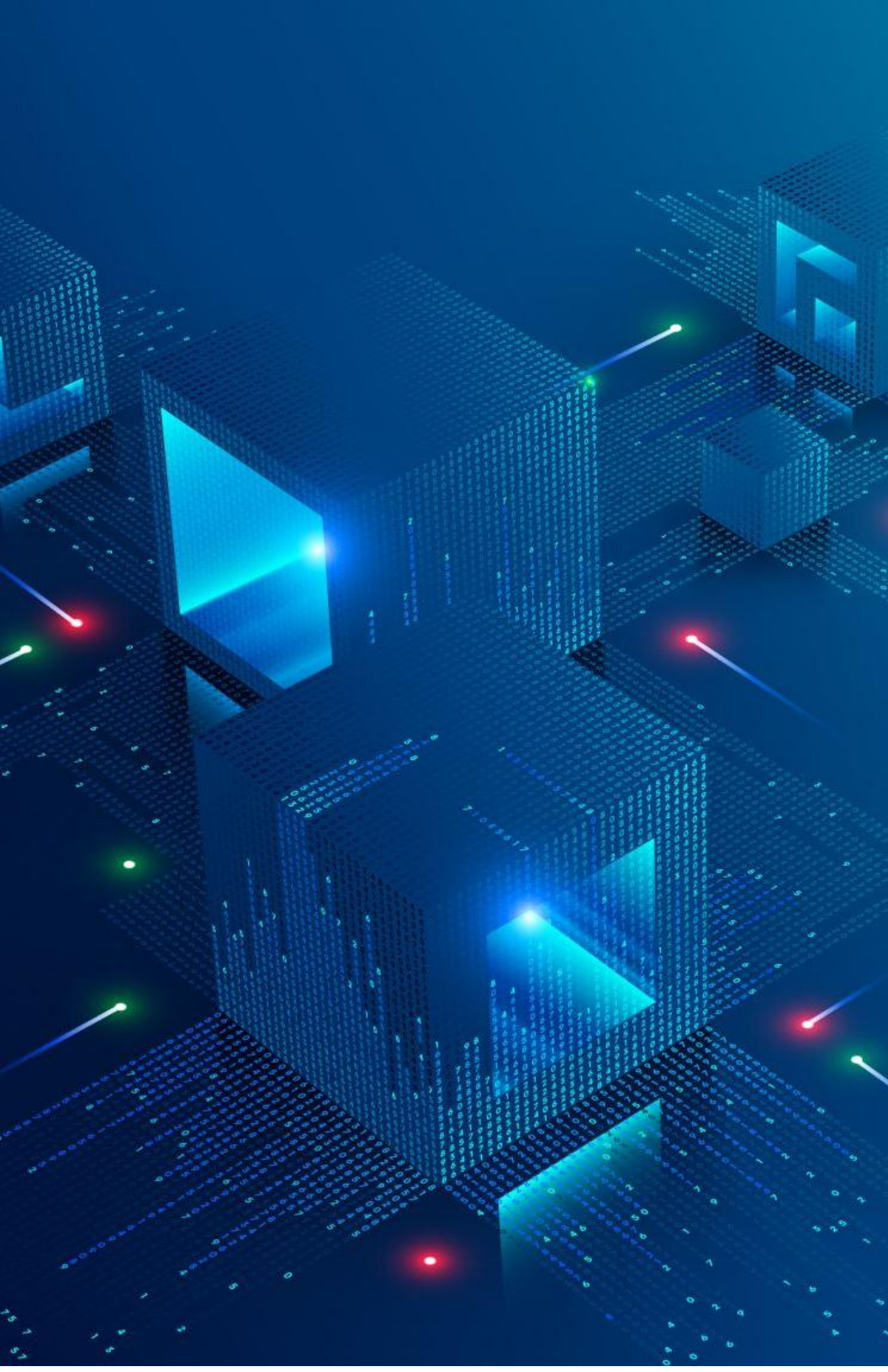
A key component of DevOps is applying principles of software development to create and manage infrastructure using code that can be managed like application source code: it is versioned controlled, tested and deployed within an automated CI/CD pipeline. This has led to the widespread adoption of infrastructure as code (IaC). In traditional provisioning of infrastructure, security teams managed security, including firewall configuration, access control and identity management. This would entail operations engineers sending detailed documentation to the various security departments to review and approve the infrastructure design before allowing a change to take place, sometimes in the development and testing environments as well as the production environment. If there was a problem, the design would need to be resubmitted and go through the change process all over again. Once approved, the changes were manual, following a set of written instructions, and often had to be carried out during a change window in unsociable hours to avoid major disruption to customers. This process would take several days or weeks, which hampered the ability to deliver new products in a timely manner. IaC has changed this model. Infrastructure code can be written, tested and deployed in just a few minutes. Although there is an obvious benefit of speed, the major problem with this approach is that it can propagate errors or security issues to many servers quickly. Security engineers should review the IaC files, but they may not have sufficient understanding of the IaC technology and the manual review would take a long time. The solution to this is to automate the testing of infrastructure code, particularly for potential security flaws.

Wilson, Glenn. DevSecOps: A leader's guide to producing secure software without compromising flow, feedback and continuous improvement (pp. 188-189). Rethink Press. Kindle Edition.

Infrastructure as Code

Key Advantages

- IT infrastructure supports and enables change, rather than being an obstacle or a constraint.
- Mitigates drift between environments by leveraging automation and push-button deployment.
- Enforces change management through GitOps with multiple approvers, as needed.
- Environmental changes are routine and fully automated, pivoting staff to focus on other tasks.
- Quicker recovery from failures, rather than assuming failure can be completely prevented.
- Empowers a continuous improvement ecosystem rather than “big bang” one and done activities.



Infrastructure as Code - IaC Security Best Practices

- **Infrastructure as Code (IaC)** is a key practice in modern IT and DevOps that involves managing and provisioning computing infrastructure through machine-readable definition files, rather than through physical hardware configuration or interactive configuration tools. Infrastructure as Code enables infrastructure setup and management using code instead of manual processes. This means that instead of clicking through a GUI or manually configuring servers, you write code to define the desired state of your infrastructure.
 - For example, you can describe:
 - Networks
 - Virtual machines
 - Load balancers
 - Databases
 - User permissions
 - All of this can be stored, version-controlled, reviewed, and deployed using standard software development practices.

Infrastructure as Code - IaC Security Best Practices

Popular Tools

Terraform – Open-source tool by HashiCorp, widely used for managing cloud infrastructure.

AWS CloudFormation – Amazon’s native IaC service for provisioning AWS resources.

Ansible – Agentless tool that uses YAML for configuration management and deployment.

Pulumi – Uses general-purpose programming languages (Python, TypeScript, etc.) for infrastructure.

Chef & Puppet – Traditionally used for configuration management, with some IaC capabilities

Infrastructure as Code – IaC Security Best Practices

Implement the Following IaC Security Best Practices to Remediate the IaC Security Issues:

Scan	Regularly scan IaC code for misconfiguration
Integrate	Integrate an automatic IaC security scanner in developer workflows to detect and rapidly remediate security issues
Detect and rectify	Detect and rectify environment drift using IaC security scanning tools
Stop	Stop the permeation of hard-coded secrets into IaC by implementing scanning on every commit
Source	Since source code and IaC code exist in the same repository, minimize the time and impact of code leaks by enforcing robust protocols, investigate suspicious user activity, implement an alerting mechanism, and quickly address source code leaks
Set up	Set up consistent centralized governance of tools to stop a single compromised account from compromising other accounts or IaC tools
Stop	Stop code tampering in IaC by regularly comparing the phases of the SDLC and detecting points where the code in the source control does not match the code in the build system

IaC & CDK Best Practices

- Follow patterns we, as developers, hopefully already follow; modularity, reusability, testing
- AWS Docs provide examples of unit testing CDK
- Ensure appropriate secrets management
- Use IaC to its full capability and don't use environment variables as that is now a redundant step
- Do not be tempted to make "one off" changes to infrastructure through a GUI – if a change is needed then refactor the CDK and deploy it
- Implement the *AWS Well-Architected* guidelines and documentation
- Separate stateless and stateful stacks

Have a stack containing databases so that you are free to redeploy your stack containing stateless microservices as often as you like without impact on your database stacks

Infrastructure as Code - Tools

- Terraform
- Ansible
- Chef
- OpenTofu
- SpaceLift
- Cross Plane
- TerraGrunt





AWS Guard Duty

Example Tool: AWS Guard Duty detects and flags potential threats in cloud environments. Automating security in CI/CD pipelines is a crucial step toward achieving consistent, efficient, and accurate security coverage. While challenges such as tool selection and false positives exist, they can be mitigated through careful planning and implementation. By automating tasks like code analysis, dependency scanning, and infrastructure validation, organizations can shift security left, ensuring vulnerabilities are addressed early and without disrupting delivery timelines. Automation not only strengthens security but also empowers teams to deliver secure software at the speed demanded by modern development practices.

Learning, Maxwell. *Advanced DevSecOps: A Guide to Advanced DevSecOps Practices*

SpaceLift

Spacelift is an infrastructure-as-code (IaC) management platform designed to help DevOps teams and cloud engineers automate, govern, and collaborate on infrastructure provisioning. It acts as a CI/CD platform purpose-built for IaC, offering flexible workflows, deep integrations, and strong policy controls.

Spacelift integrates with a wide variety of tools and platforms, such as:

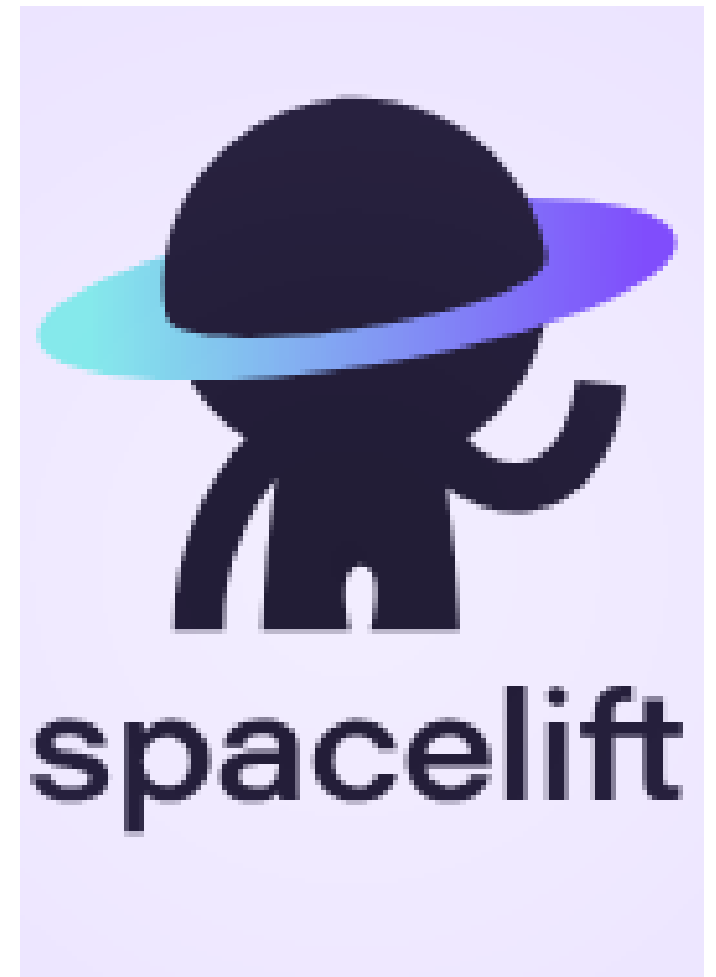
Git providers (GitHub, GitLab, Bitbucket)

Cloud providers (AWS, GCP, Azure)

Secret managers (Vault, AWS Secrets Manager)

ChatOps tools (Slack, MS Teams)

<https://spacelift.io/pricing> there is a free version to experiment with.



Chef



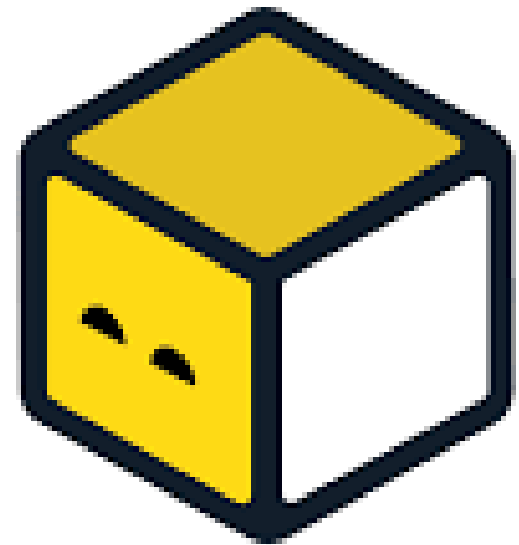
Chef was originally developed by Opscode and is now part of **Progress Software**. It automates infrastructure by turning it into code, allowing system administrators and DevOps engineers to define the desired state of their infrastructure and ensure it remains in that state consistently. Chef components:

- Cookbook: A collection of recipes, attributes, libraries, and files to define system configurations.
- Recipe: A script that defines specific steps to configure a part of the system.
- Resource: The basic building block (e.g., package, service, file) used in recipes.
- Run-list: The order in which recipes are applied to a node.

<https://www.chef.io/products/chef-infra>

OpenTofu

This is a fork of Terraform. OpenTofu is designed as a **drop-in replacement for Terraform**, aiming to maintain full **backward compatibility** while also accelerating community-led innovation. It allows developers and operations teams to describe infrastructure in a high-level configuration language (HCL) and automatically deploy and manage it across various platforms like AWS, Azure, GCP, and more



OpenTofu

<https://opentofu.org/>

SaltStack

SaltStack, often simply referred to as Salt, is an open-source configuration management and remote execution tool designed to automate the management and provisioning of infrastructure in a secure and scalable way.

SaltStack is a Python-based automation and configuration management platform developed to manage and maintain IT infrastructure at scale. It supports configuration automation, remote command execution, orchestration, and more, making it useful for DevOps and system administrators.

Originally developed by Thomas S. Hatch in 2011, SaltStack (the company) was later acquired by VMware in 2020. It has both an open-source version and a commercial enterprise offering.

<https://saltproject.io/>



Aikido Security

“Scan every Terraform, CloudFormation, and Helm change for critical misconfigurations.”

<https://www.aikido.dev>

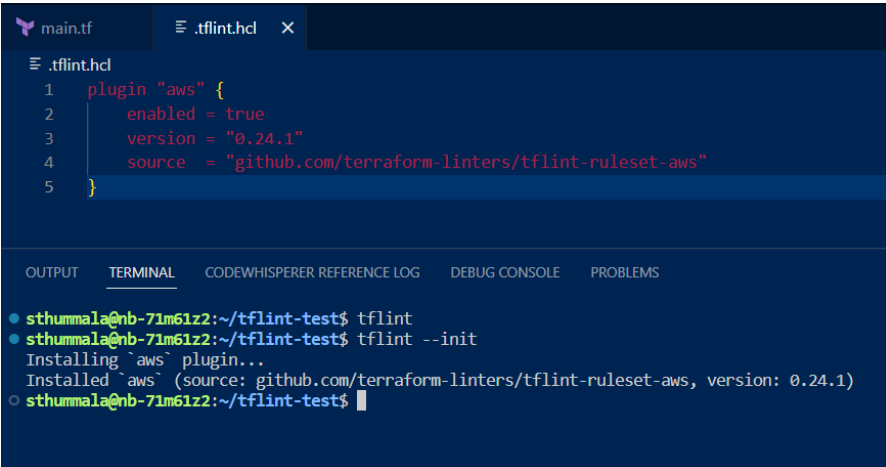
The screenshot shows the Aikido Security dashboard. On the left is a dark sidebar with the Aikido logo and navigation items: MindMeld AI, Feed, My Work, Hot Links, Critical (2), Auto Patches, Ingored, Repos, core, frontend, and rest-api. The main content area has a greeting 'Hello, Amber!' and search, support, and docs links. It features three summary cards: a total of 61 issues (2 Critical, 23 Medium, 24 High, 12 Low), 16 issues solved up from 12 last month, and 18 new issues in the last month. Below these is a table of issues.

Type	Name	Severity	Status	Assignee
	Unsanitized input At risk of arbitrary code injection	Critical	To Do	
	json-schema Abuse of JavaScript's prototype API possible (prototype pollution)	Critical	New	
	SVN Repo exposure This is why that sucks	High	Task Open	BertBirdz
	MFA Root account should have MFA enabled	High	Task Open	BertBirdz

TFLint

A scanner for Terraform

<https://github.com/terraform-linters/tflint>



```
main.tf  .tflint.hcl  X
.tflint.hcl
1 plugin "aws" {
2   enabled = true
3   version = "0.24.1"
4   source  = "github.com/terraform-linters/tflint-ruleset-aws"
5 }
```

OUTPUT TERMINAL CODEWHISPERER REFERENCE LOG DEBUG CONSOLE PROBLEMS

```
● sthumma1a@nb-71m61z2:~/tflint-test$ tflint
● sthumma1a@nb-71m61z2:~/tflint-test$ tflint --init
Installing `aws` plugin...
Installed `aws` (source: github.com/terraform-linters/tflint-ruleset-aws, version: 0.24.1)
○ sthumma1a@nb-71m61z2:~/tflint-test$
```

Principles of IaC

Make	Make all changes via definition file <ul style="list-style-type: none">•Exception: Testing a concept in the lab, with what you do being over-written immediately
Document	Document systems & processes in code, not diagrams
Use	Use VCS for all files
Apply	Apply CI/DD <ul style="list-style-type: none">•Run all tests with each change
Apply	Apply the small batches principle
Keep	Keep services available continuously

Declarative & Consistent Environments via Code



Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes. With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations.



Infrastructure as code (IaC) uses DevOps methodology and versioning with a descriptive model to define and deploy infrastructure, such as networks, virtual machines, load balancers, and connection topologies. Just as the same source code always generates the same binary, an IaC model generates the same environment every time it deploys.

The problems with manual infrastructure approaches



Human error. It's easy to create infrastructure with any method; but it's not easy to consistently produce the same result when running scripts ad-hoc, "one-off" deployments, and natural changes over time.



Lack of consistency. Multiple tools, scripts, tools, and methodologies require more time spend on ensuring correctness and consistency.



Scaling becomes too hard and has limited reproducibility.



Slower feedback loop. Deployment becomes slower, feedback becomes slower, development becomes slower.

Automated, declarative, maintainable infrastructure as code



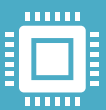
Infrastructure should not be at risk of becoming fragile, resistant to change, or practically impossible to redeploy



Infrastructure should be able to be version controlled, expressed in written language form, able to be shared, changed, and updated



Infrastructure should be able to be created and destroyed in minutes – not months



Infrastructure should be *declared as code* and anyone that needs should be able to make infrastructure changes and have it deployed automatically

Popular IaC solutions



- In general terms there are two broad categories of IaC: imperative and declarative.
- Imperative IaC is using existing languages and frameworks to create infrastructure step by step by indicating *how* the infrastructure should exist
- Declarative IaC is using schema-based Domain Specific Languages (DSLs) to indicate *what* infrastructure must exist



Popular IaC solutions

Popular IaC tools have different design philosophies; imperative vs declarative, DSLs vs existing language, cloud agnostic vs cloud specific, stateful vs stateless

	CONVENTIONS	DECLARATIVE	DSL	CLOUD AGNOSTIC	STATELESS
Nix/NixOS	✓	✓	✓	✓	✓
Terraform	✓	✓	✓	✓	✗
AWS Cloudformation	✓	✓	✓	✗	✓
Azure ARM/Bicep	✓	✓	✓	✗	✓
AWS CDK	✓	✗	✗	✗	✗
Pulumi	✓	✗	✗	✗	✗
PowerShell, Bash, Ansible, Chef, etc	✗ / 🤔	✗	✗	✗	✗

IaC maturity rank

Rank	Solution
A	Automated CI/CD, fast feedback loops, declarative infrastructure as code, unique environments can be rapidly created with PRs in source control, frequent and low ceremony releases, multiple releases daily if desired, unit testing of IaC
B	Automated CI, some manual processes, releases have a degree of ceremony, infrequent releases, some deployment scripts
C	Deployments involve SSH or RDP with servers and manually deploying files, “big bang” releases every months or quarters, slow feedback loop
D	No automation at all, adoption of infrastructure as code requires fully rearchitecting the stack, configuration is not in source control, requires constant usage of GUIs, lack of understanding of how the platform operates

AWS Cloud Developer Kit (CDK)



We'll be using CDK for this project



We should use existing CDK based projects in the D&G GitHub as a reference to familiarise ourselves with existing conventions and practices



Following existing usage in other projects in our GitHub, we will be using TypeScript with the CDK library

CDK Example: Lambda Function processing SQS messages

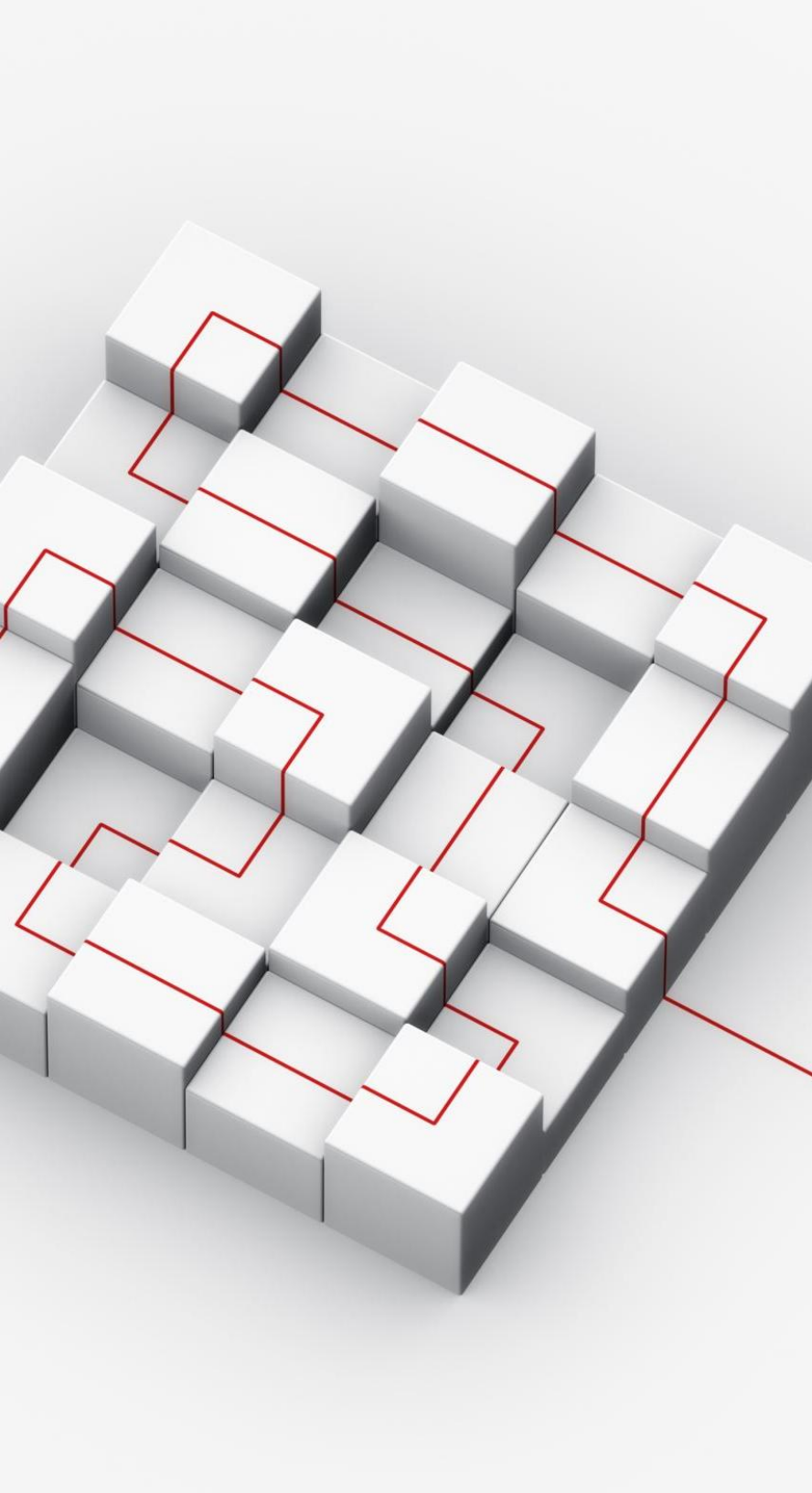
```
export class LambdaToSqsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // Create an SQS queue
    const queue = new sqs.Queue(this, 'dg-b2b-portals-prod-plan-purchased-queue', {
      visibilityTimeout: cdk.Duration.seconds(300)
    });

    // Create an AWS Lambda function
    const myLambda = new lambda.Function(this, 'dg-b2b-portal-prod-plan-purchased', {
      runtime: lambda.Runtime.DOTNET_7,
      handler: 'B2BPortals::EntryPoint::Main',
      code: lambda.Code.fromAsset('<.csproj> path')
      environment: {
        QUEUE_URL: queue.queueUrl
      }
    });

    // Grant necessary permissions for the Lambda function to send messages to the SQS queue
    queue.grantSendMessages(myLambda);

    // Configure the Lambda function to be triggered by an SQS event source
    myLambda.addEventSource(new events.SqsEventSource(queue));
  }
}
```



CDK Constructs and Stacks

- Not native AWS terminology, it is CDK specific terminology
- At a high level, constructs are types available in the CDK library for use by developers
Lambda, S3 Bucket, API Gateway are examples of constructs
- Stacks are written by developers to define their infrastructure needs
- In our project, some examples of stacks could be:
FrontendStack – the React SPA
BackendStack – the .NET Lambdas
ConfigurationStack – the database for our various portals and their products
DashboardStack – the database and dashboard for the team to see how many plans are sold daily, how many errors occur, application performance, etc

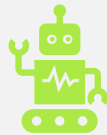
What is Terraform



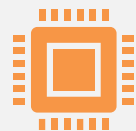
Terraform is an open source infrastructure as code tool.



Maintained by HashiCorp, written in GO.



Uses HCL (HashiCorp Configuration Language).



Is platform agnostic, capable deploying to many different providers.

Terraform IaC

Here is an example of how we might use Terraform to define an AWS S3 bucket:

Kumar Rath, Ashwini. Concepts and Practices of DevSecOps: Crack the DevSecOps interviews (English Edition) (p. 60). BPB Publications. Kindle Edition.

```
provider "aws" {  
    region = "us-west-2"  
}  
  
resource "aws_s3_bucket" "bucket"  
  
    bucket = "my-bucket"  
  
    acl = "private"  
  
    tags = {  
  
        Name = "My bucket"  
  
        Environment = "Dev"  
  
    }  
}
```

What is Terragrunt



Terragrunt is a wrapper for the Terraform executable.



Fills in functionality gaps of the vanilla Terraform tool.



Maintained by Gruntworks, written in GO



Virtually transparent after initial configuration.

Folder Structure

Recommended practice from Gruntworks is to use a two tree folder structure.

“Live” folder contains all input data and represents actively deployed state.

“Modules” folders contains resource deployment code that will use “Live” variables.

Providers



Providers abstract each unique platforms' functionality into common calls for Terraform to perform.



Providers define how API calls to each platform are made, and what resources can be managed on each platform.



Providers are maintained independently of the Terraform core application.

HCL Syntax



HCL uses stanzas or blocks to define resources and their variables.



Blocks use key value pairs to define input data for a resource.



Data types include **bool**, **number**, **string**, **list**, **map**



Interpolation syntax uses “`${...}`” to escape string sequences.

HCL Resource Blocks

Terraform has four primary blocks that are used. Each is declared by its type.

Resource blocks define a resource that will be created.

Variable blocks define an input variable for a module.

Output blocks define outputs of resource properties to use by other modules.

Data blocks define remote data lookups to query properties of existing resources.

Security Challenges of IaC

- **Organizations that fail to adhere to IaC best practices can expose their cloud ecosystem to attackers, leading to the following security issues:**
- **Network Exposure:** Most online attacks become successful because of insecure IaC practices in the organization. Some examples of IaC misconfiguration that can compromise the cloud environment include the misconfiguration of cloud storage services, publicly accessible SSH, misconfiguration of open-security groups, and Internet-accessible databases
- **Drifting Configuration:** In an organization, even if the developers have adhered to the IaC best practices, emergencies may compel the operations team to change the configuration in the production environment. This type of modification after deploying the infrastructure can break the immutability of cloud infrastructure
- **Unauthorized Privilege Escalation:** Most organizations use IaC for running their cloud environment, which encompasses microservices, Docker, Kubernetes, etc. The use of privileged accounts by developers for running cloud applications and other essential software causes escalated privilege risks
- **Compliance Violations:** While creating resources using IaC, untagged resources can lead to ghost resources that are difficult to visualize and detect in the cloud ecosystem. This leads to drift in the cloud posture. If remain undetected for a long period, compliance violations may occur

The screenshot shows the Snyk Cloud Config interface. The left sidebar contains navigation options: GROUP, Company Name / Line of Business, ORGANIZATION, Organization Name, Dashboard, Reports, Projects, Cloud, Integrations, Members, and Settings. The main content area is titled 'Organization Name > Manage > Cloud Config'. Under 'ORGANIZATION SETTINGS', 'Infrastructure as code' is highlighted. The 'Infrastructure as code' section includes a help icon and the title 'Infrastructure as code'. Below this, there is explanatory text: 'Snyk can now detect security misconfigurations in your configuration files when importing Git projects. You can find the full details in the [documentation](#) and enable the functionality with the toggle below. See the documentation for [details of individual security rules](#).' The 'Detect configuration files' section is highlighted with a red border and contains the text 'Snyk can detect configuration files and report any misconfigurations that may lead to security issues.' Below this text is a toggle switch labeled 'Enabled', which is currently turned on.

Snyk: Scan Terraform Configuration Files

- Snyk secures the cloud ecosystem by monitoring and testing the **Terraform files** in the source-code repository and helps in detecting and fixing misconfigurations

Terrascan: Scan Terraform Configurations

- Terrascan is a tool designed to analyze Infrastructure as Code (IaC) files and identify potential security vulnerabilities, compliance issues, and best practice violations
- It leverages a set of predefined rules and policies to perform static code analysis on IaC code

```
violations across Infrastructure as Code to  
sit https://runterrascan.io/
```

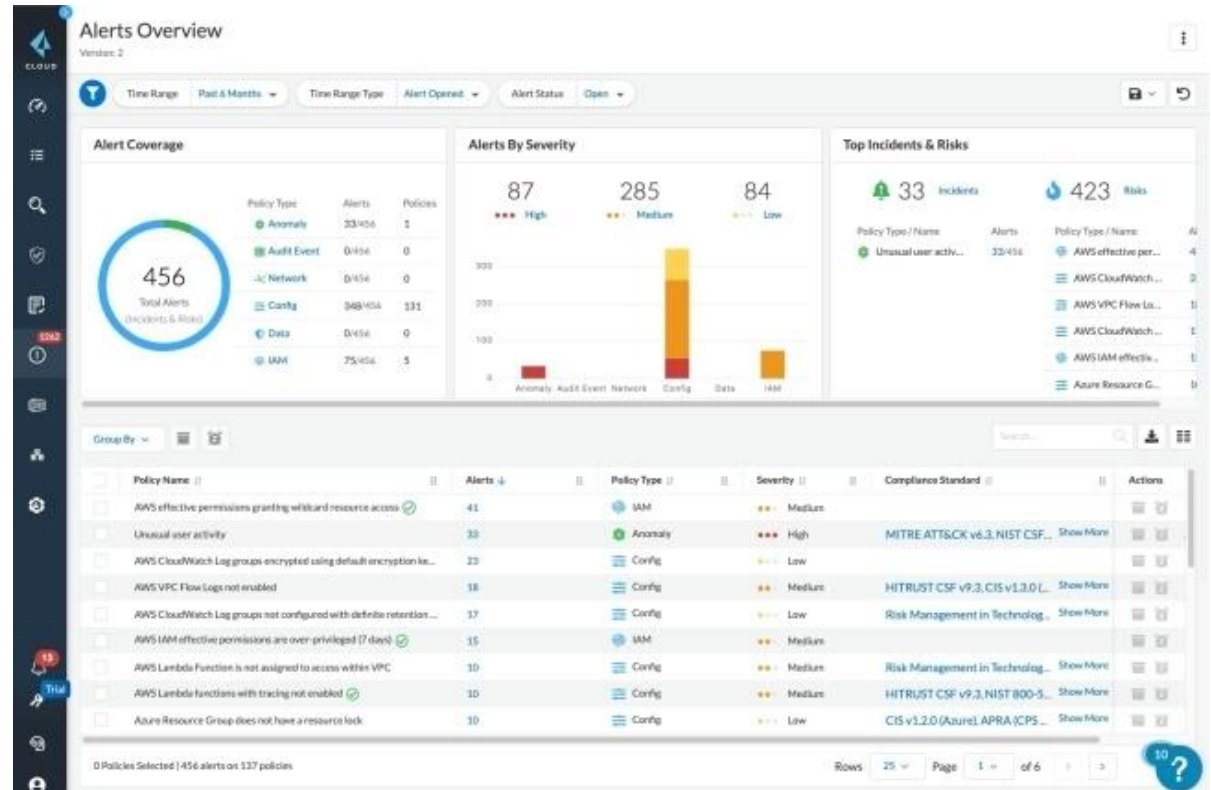
```
fo about any command  
can  
ect compliance and security violations across  
an API server  
an version you are currently using.
```

```
config file path  
help for terrascan  
log level (debug, info, warn, error, panic,  
directory path to write the log and output  
log output type (console, json) (default "c  
output type (human, json, yaml, xml, junit-  
temporary directory path to download remote
```

```
" for more information about a command.|
```

PrismaCloud: Scan Terraform Files in GitHub

- The integration of GitHub with PrismaCloud enables DevSecOps teams to identify misconfigurations and policy violations by scanning Terraform files



PrismaCloud: Scan CloudFormati on Templates

- By integrating PrismaCloud with AWS, an infrastructure scan can be performed on CloudFormation templates to detect security issues
 - **AWS Read Access:** This permits PrismaCloud to execute read-only API calls, which enable basic configuration scanning. The integration of AWS read access creates an IAM role with the AWS managed SecurityAudit policy. It then creates a message that notifies BridgeCrew about the integration
 - **AWS Remediation Stack:** This helps in mitigating policy violations. The integration of remediation creates resources such as an SSM parameter for PrismaCloud tokens and Lambda functions



Logs

- The Role of Logs in Tracking and Investigating Security Incidents Logs are the lifeblood of any security monitoring system. They provide a detailed record of events and activities across an organization's infrastructure, offering insights into system health, user behavior, and potential security incidents. In a DevSecOps environment, logs are essential for identifying and investigating suspicious activity, enabling teams to trace the root cause of incidents and respond effectively.
- Learning, Maxwell. Advanced DevSecOps: A Guide to Advanced DevSecOps Practices

Bet Practices for Avoiding Alert Fatigue

Overloading teams with alerts can lead to missed critical events. To avoid this, organizations must adopt a thoughtful approach to alert configuration.

- **Fine-Tune Alert Triggers:** Regularly review and adjust thresholds based on system changes and emerging threats. Example: Update thresholds for resource usage alerts after scaling up infrastructure.
- **Use Noise-Reduction Techniques:** Filter out low-priority alerts or group related notifications into a single message. Example: Combine multiple low-severity alerts into one periodic summary.
- **Test Alerts Regularly:** Validate that alerts trigger as intended and provide actionable information. Example: Simulate an unauthorized access attempt to ensure the alert system works effectively.
- **Adopt Escalation Policies:** Implement escalation mechanisms to ensure critical alerts are addressed promptly. Example: Notify a senior security engineer if a critical alert is unresolved for more than 30 minutes.
- **Leverage AI and Machine Learning:** Use advanced analytics to identify meaningful patterns and reduce noise. Example: Employ machine learning algorithms to identify unusual user behavior indicative of insider threats.