GCP

* Google Cloud Platform
* Googles cloud service.
* 3rd largest in the world.
  + The fastest growing one. I believe\*\*\*\*\*
* GCP was built by Google developers.
  + AWS was built by Amazon IT Infrastructure

GCP Services

* SQL
  + Creates a VM that already has a version of SQL downloaded on it.
  + Opens the correct ports for connecting to the SQL database.
  + Optimized for database usage.
* GCP Storage
  + You create buckets that are uniquely named in the world.
  + These buckets can store any type of file.
    - Png
    - Jar
    - Html
  + You can folders in a bucket as well
  + Static Web Site hosting
    - A way of deploying front-ends to users
    - Put all the html,css,js in a bucket and make those files public.
    - Easy to do.
    - Very cost effective and reliable
      * Buckets are less likely to fail than your vm.
* Compute Engine
  + The primary service for creating virtual machines in GCP.
    - Advantages to hosting backends using GCP virtual machines.
      * It takes only a few minutes to create a virtual machine.
      * You can pick any operating system you want
        + Even choose a custom boot disk like a copy of a previous VM.
      * The VMs can easily be made to be publicly accessible.
        + Make sure you disable the firewall appropriately.
      * Delete instances whenever you need to.
      * Pay for only what you are using.
        + Really saves on cost.
  + GCP conveniently allows you to SSH to a VM via the browser.
* Disk Storage.
  + Hard drive disk that a VM uses.
  + You can detach a disk from a VM.
    - You can then attach it to a new VM.
  + Copy a disk (aka make a machine image)

Linux

* Operating system
  + Based on Unix and written by Linus Torvalds.
* Unix was designed in the early 1970s but AT&T
  + Operating system for mainframe computers.
  + Multiple users would the use the mainframe at the same time.
    - That is why you have to permissions on files.
    - sudo (super user do) execute a command as the root user which has complete access to the machine.
  + There is no graphics in a Unix system.
    - 1970 not really a thing anyway.
    - Everything is done in a text based terminal
* Why is it so popular for cloud computing and web server deployments?
  + It is essentially free.
  + Very lean in comparison to windows.
    - Less bloat on a Linux machine than a Windows.
  + Well established package managers
  + Most Linux distributions come with a package manager.
    - Yum
      * Yum install maven
        + Goes to the yum repository and install maven.
    - Deb
* Linux commands
  + Cd => change directory
  + Ls => list
  + Touch => make file
  + Mkdir => make directory
  + Text editors
    - Vim (objectively the best)
    - Nano
    - Emacs
  + Ps => shows all processes
  + Mv => move a file somewhere else
  + Cp => copy past
  + Rm => remove a file
  + Rmdir => remove a directory
  + Sudo => super user do
  + Pwd => print working directory
  + Cd ../ => move up a directory
  + Echo => echo
  + Man => manual
  + Cat => prints to the screen
  + More => keeps scrolling
  + Export => set an environment variable
  + Curl => makes an http request
    - Curl -O performs a download
  + Chmod => edit the permission on a file
    - Read
    - Write
    - Executable permissions
  + Tar
    - xzvf
    - Zip or unzip tar files
  + ./ => executes a file
  + Ls -l
    - -l is a flag

OWASP

Open Web Application Security Project

* Non profit software organization dedicate to promoting good security practices.
* Publish tutorials or hold meetings on how best to approach security.
  + General security guidelines
    - Security should be layered.
      * Don’t rely on one safe guard.
    - Design applications and permissions on the least privilege principle.
      * If the only IP address that can access a database is known.
      * Then make the restrictions such that the firewall only allows that IP address.
      * If a computer only needs access to a single bucket.
      * Ideally that computer only has access to that single bucket

OWASP top 10

* The most common security flaws in applications that year
* [**Injection**](https://owasp.org/www-project-top-ten/2017/A1_2017-Injection). Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
  + PreparedStatement vs Statement
* [**Broken Authentication**](https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication). Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities temporarily or permanently.
  + Making sure only the correct people have JWTs
  + Your endpoints should not be completed unprotected.
* [**Sensitive Data Exposure**](https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure). Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
  + Make sure you account for ALL of your endpoints
  + In very large applications someone might write an endpoint that bypassed security without even noticing.
* [**XML External Entities (XXE)**](https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE)). Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
  + Imagine if someone maliciously swapped out your cfg.xml
  + Someone got a hold of your xml they might sensitive details.
* [**Broken Access Control**](https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control). Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users’ accounts, view sensitive files, modify other users’ data, change access rights, etc.
  + Giving people too many permissions by mistake
* [**Security Misconfiguration**](https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration). Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
* [**Cross-Site Scripting (XSS)**](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)). XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
  + Imagine puts their username as a script tag with JS code
  + Then when another person loads the webpage and it shows that ‘’username’ it executes the script
* [**Insecure Deserialization**](https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization). Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
  + Serializable interface in Java is deprecated because of this issue.
  + Deserializing from Java Byte led to people executing malicious code.
* [**Using Components with Known Vulnerabilities**](https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities). Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
  + Using dependencies that have security flaws.
  + Imagine if Javalin had some security flaw.
* [**Insufficient Logging & Monitoring**](https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring). Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.
  + Logging lets you see suspicious activity and respond to it.
  + Perform damage control.