HEIDELBERG LAUREATE FORUM

IIIT Dharwad ज्ञानेन विकासः

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

acm India Council

Pingala Interactions In Computing

Cloud Platforms and Frameworks

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Pingala Interaction in Computing

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur

KING'S College LONDON

The Alan Turing Institute

Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर

PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

IaaS and Hybrid Cloud

1. **IaaS and Hybrid Cloud**
   - **Orchestration & Virtualization: Eucalyptus & Amazon**
   - **Content Delivery Network (CDN): Facebook and Akamai**
2. PaaS and Container as a Service (CaaS)
   - PaaS: Google App Engine (GAE) and Ruby on Rails
   - CaaS: DockerHub
3. SaaS and Distributed Version Control (DVC)
   - SaaS: Facebook Testing (Infer and Sapienz)
   - DVC: GitHub and Git-LFS

IaaS and Hybrid Cloud

A. **Orchestration and Virtualization**
   - **Orchestration**
   - **Virtual machines**
     - **Type-1, native or bare-metal hypervisors**
     - **Type-2 or hosted hypervisors**
     - **Both Type 1 and Type 2**

B. Hybrid Eucalyptus and Amazon-EC2

A. Content Delivery Network, or
   Content Distribution Network (CDN)

B. Hybrid Akamai and Facebook

Infrastructure as a Service (IaaS)

- IaaS are online services
  - provide high-level APIs used to dereference (or abstract) various low-level details of infrastructure like
    - physical computing resources,
    - location,
    - data partitioning,
    - scaling,
    - security,
    - backup
- IaaS deals with Cloud Orchestration and Virtualization

https://en.wikipedia.org/wiki/Infrastructure_as_a_service

Examples of IaaS Clouds

IaaS involves the use of a Cloud Orchestration and Virtualization Technologies like

- Eucalyptus
  - Hypervisors (KVM, Xen, VMware)
  - Private and
  - Hybrid cloud computing with Amazon Elastic Compute Cloud (EC2)
- Open Stack,
  - OpenStack-native REST API and
  - Hybrid cloud computing with CloudFormation-compatible Query API
- Apache Cloudstack
  - virtualization, such as KVM, VMware vSphere, including ESXi and vCenter, and XenServer/XCP.
- OpenNebula.
  - Hypervisors (VMware vCenter, KVM, LXD and AWS Firecracker)

https://en.wikipedia.org/wiki/Infrastructure_as_a_service

Examples of IaaS Clouds

- Amazon Web Services
- AppScale
- Box
- Bluemix
- CloudBolt
- Cloud Foundry
- Cocaine (PaaS)
- Creatio
- EngineYard
- GreenQloud
- IBM Cloud
- iland
- Joyent
- Linode
- Lunacloud
- Microsoft Azure
- Mirantis
- Netlify
- Nimbula
- Nimbus
- OpenIO
- OpenNebula
- OpenStack
- Oracle Cloud
- OrionVM
- Rackspace Cloud
- Safe Swiss Cloud
- SoftLayer
- Zadara Storage
- libvirt
- libguestfs
- OVirt
- Virtual Machine Manager
- Wakame-vdc
- Virtual Private Cloud OnDemand

https://en.wikipedia.org/wiki/Infrastructure_as_a_service

Orchestration

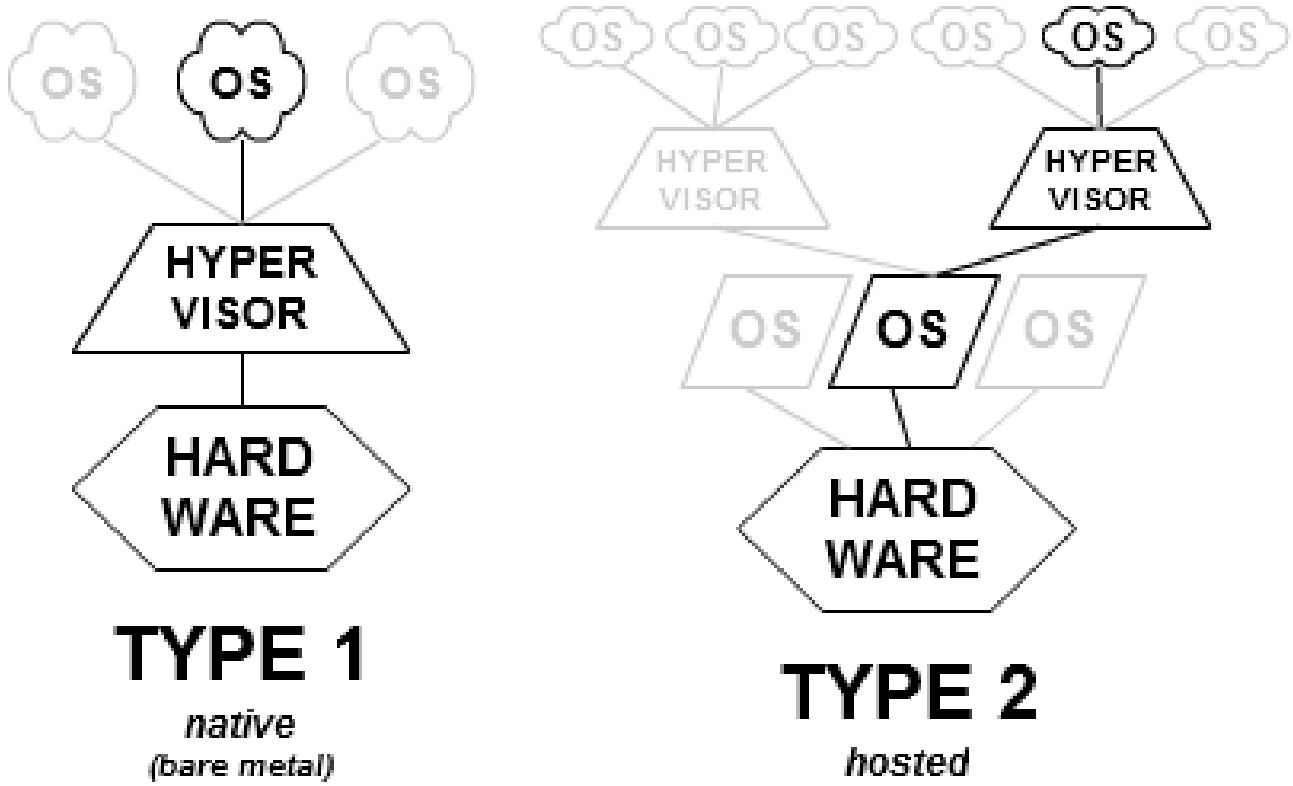
- Automated configuration, coordination, and management of
  - service-oriented architecture,
  - virtualization,
  - provisioning,
  - converged infrastructure and
  - dynamic datacenter.
- To meet requirements of Applications, Data, and Infrastructure.
- Difference Between
  - Workflows Automation are processed and completed as processes within a single domain for automation purposes.
  - Orchestration includes a workflow and provides a directed action towards larger goals and objectives.

https://en.wikipedia.org/wiki/Orchestration_(computing)

Virtualization

Hypervisor: Runs one or more virtual machines is called a *host machine*, and each virtual machine is called a *guest machine*.

- Type-1, native or bare-metal hypervisors
- Type-2 or hosted hypervisors:

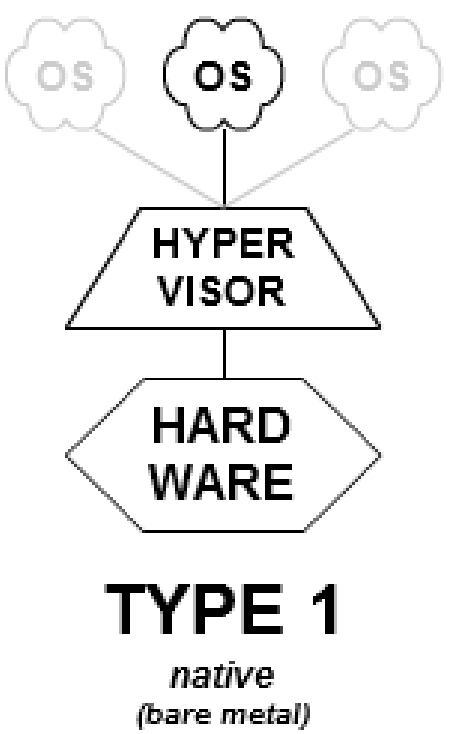


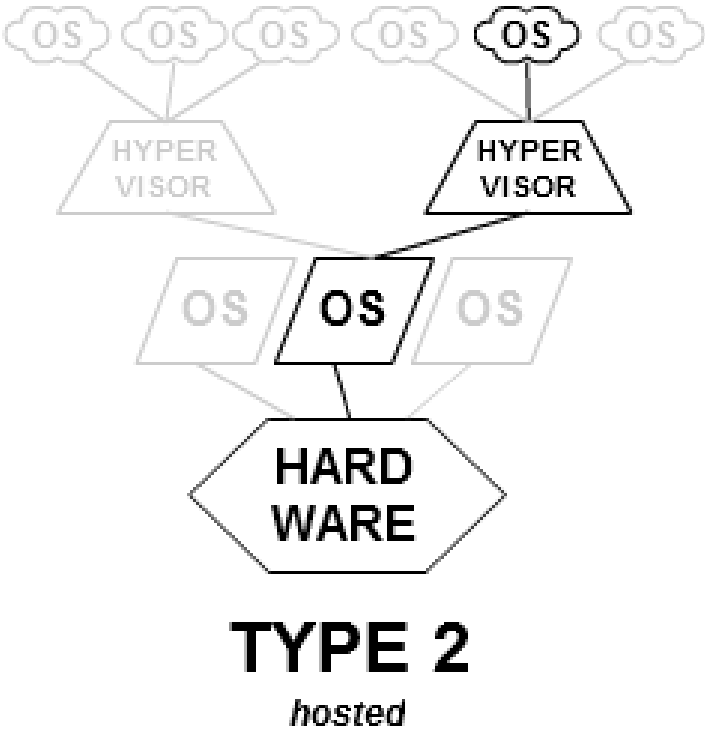
https://en.wikipedia.org/wiki/Hypervisor

Type-1, native or bare-metal hypervisors:

run directly on the host's hardware to control the hardware and to manage guest operating systems.

- SIMMON and IBM's CP-40 (first Hypervisors late 1960s)
- Microsoft Hyper-V and Xbox One system software
- VMware ESXi (formerly ESX),
- Nutanix AHV,
- XCP-ng,
- Oracle VM Server for SPARC and x86,
- POWER Hypervisor, and
- Xen.



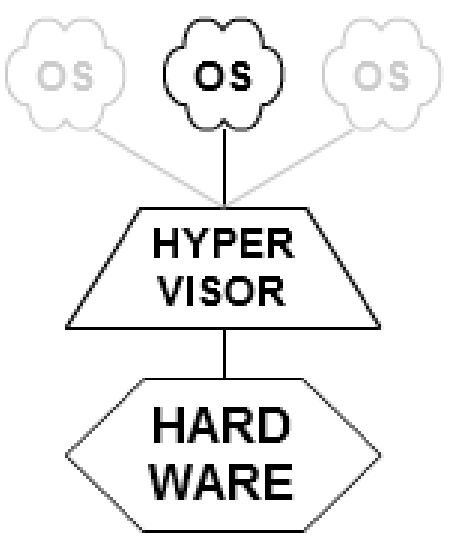


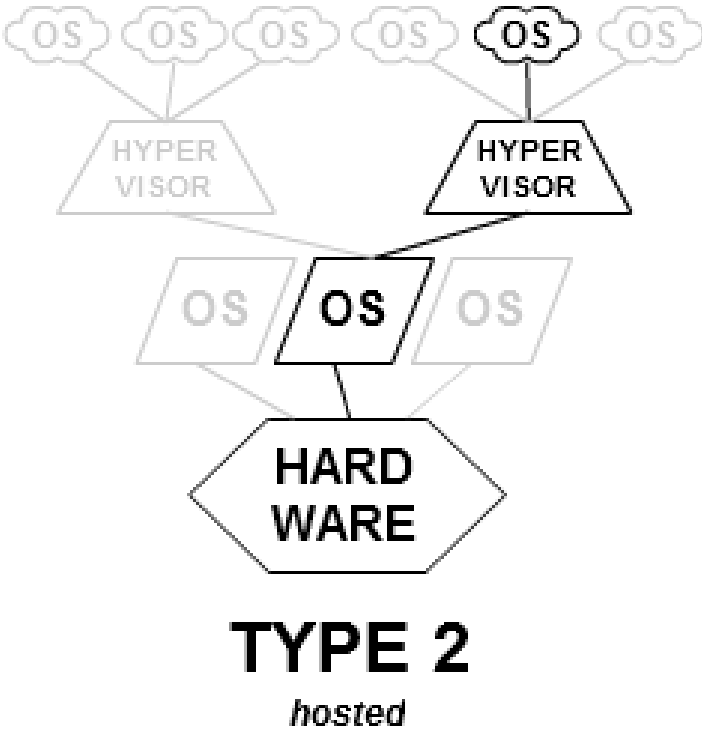
https://en.wikipedia.org/wiki/Hypervisor

Type-2 or hosted hypervisors:

run on a conventional Operating-system of a host machine like a process as a guest machine

- Oracle VirtualBox,
- Parallels Desktop for Mac,
- QEMU,
- VirtualBox,
- VMware Player and
- VMware Workstation



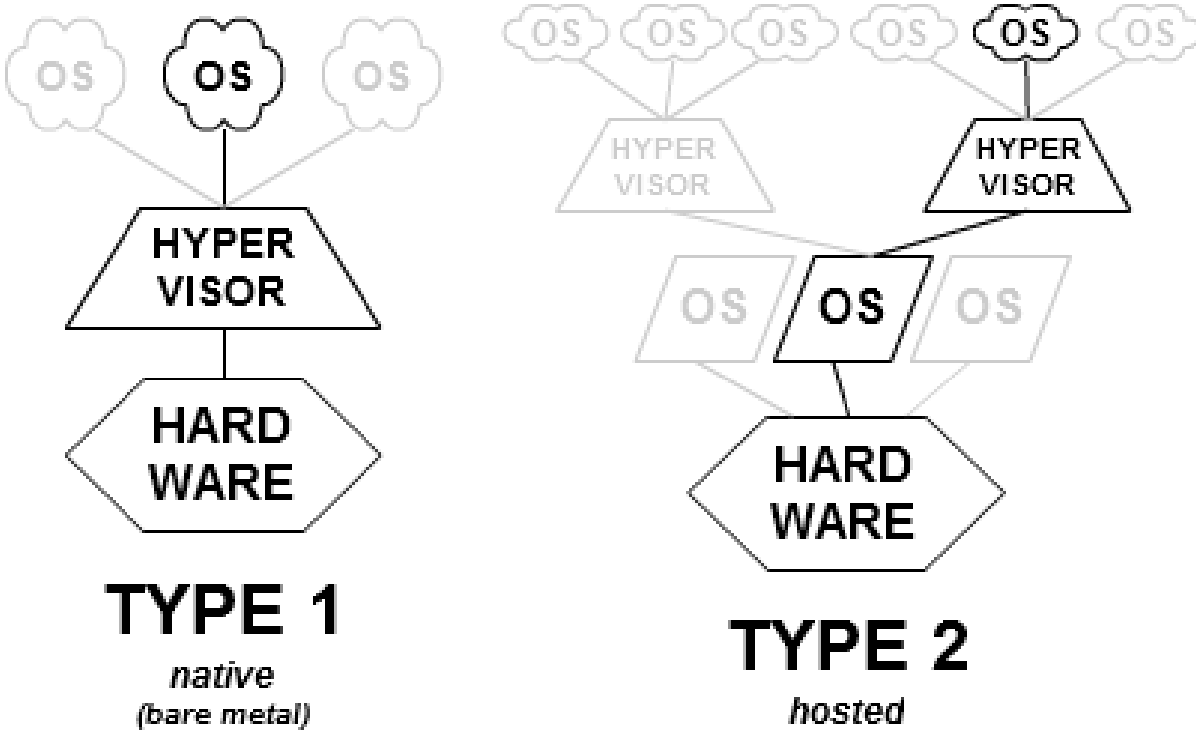


https://en.wikipedia.org/wiki/Hypervisor

Both Type 1 and Type 2

are kernel modules

- Linux's KVM (Kernel-based Virtual Machine)
- FreeBSD's bhyve (Berkeley Software Distribution BSD)



https://en.wikipedia.org/wiki/Hypervisor

IaaS and Hybrid Cloud

A. Orchestration and Virtualization
   - Orchestration
   - Virtual machines
     - Type-1, native or bare-metal hypervisors
     - Type-2 or hosted hypervisors
     - Both Type 1 and Type 2

B. Hybrid Eucalyptus and Amazon-EC2

A. Content Delivery Network, or
   Content Distribution Network (CDN)

B. Hybrid Akamai and Facebook

Amazon EC2, EBS, and S3

Amazon Elastic Compute Cloud (EC2):

- Pay per usage: rent Virtual Machines (VMs) to run applications
- Elastic: scalable deployment of applications by web services
- Amazon Machine Image (AMI) instance contains desired software
- Create, launch, and terminate instances,
- Amazon switched its own retail website platform to EC2 and AWS

Amazon Elastic Block Store (EBS) provides raw block-level storage that can be attached to Amazon EC2 instances used by Amazon Relational Database Service (RDS).

Amazon Simple Storage Service (S3) provides scalable storage infrastructure through a web service interface.

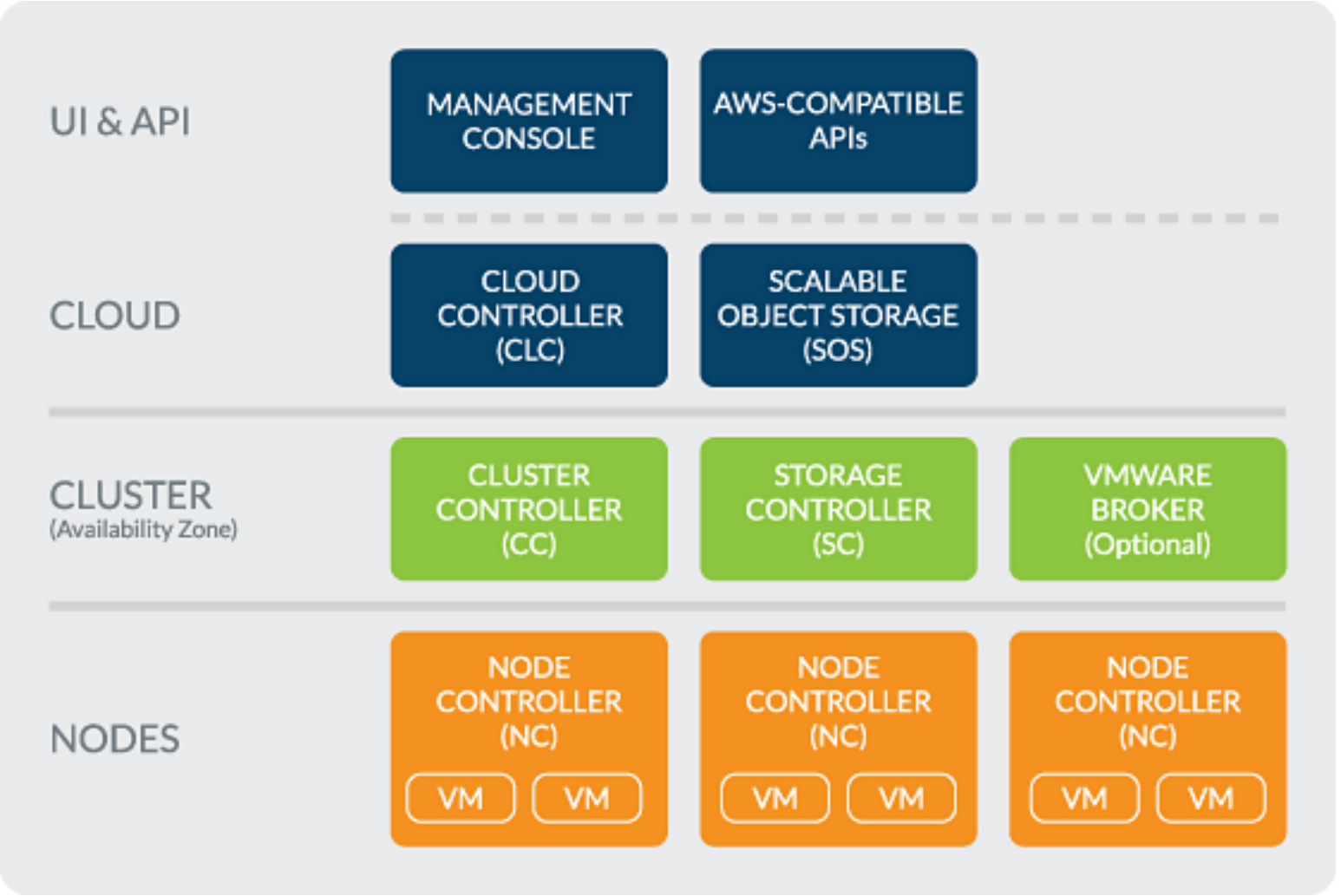
https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
https://en.wikipedia.org/wiki/Amazon_S3
https://en.wikipedia.org/wiki/Amazon_Elastic_Block_Store

Eucalyptus Cloud

- Eucalyptus **cloud** is highly scalable.
- **Eucalyptus Cloud Components:** The six components are grouped into three separate levels.
- Cloud Level: The Cloud level of the computing architecture is comprised of only two components and while used by many users, the transactions at each component are typically small.
  - Cloud Controller (CLC)
  - Scalable Object Storage (SOS)
- Cluster Level (i.e., Availability Zone)
  - Cluster Controller (CC)
  - Storage Controller (SC)
  - VMware Broker (Optional)
- Node Level: The Node level may have many components, but each component only supports a few users, even though the transactions are larger. This **distributed cloud architecture** is flexible enough to support businesses of any size.
  - Node Controller (NC)

https://www.eucalyptus.com/eucalyptus-cloud/iaas/architecture

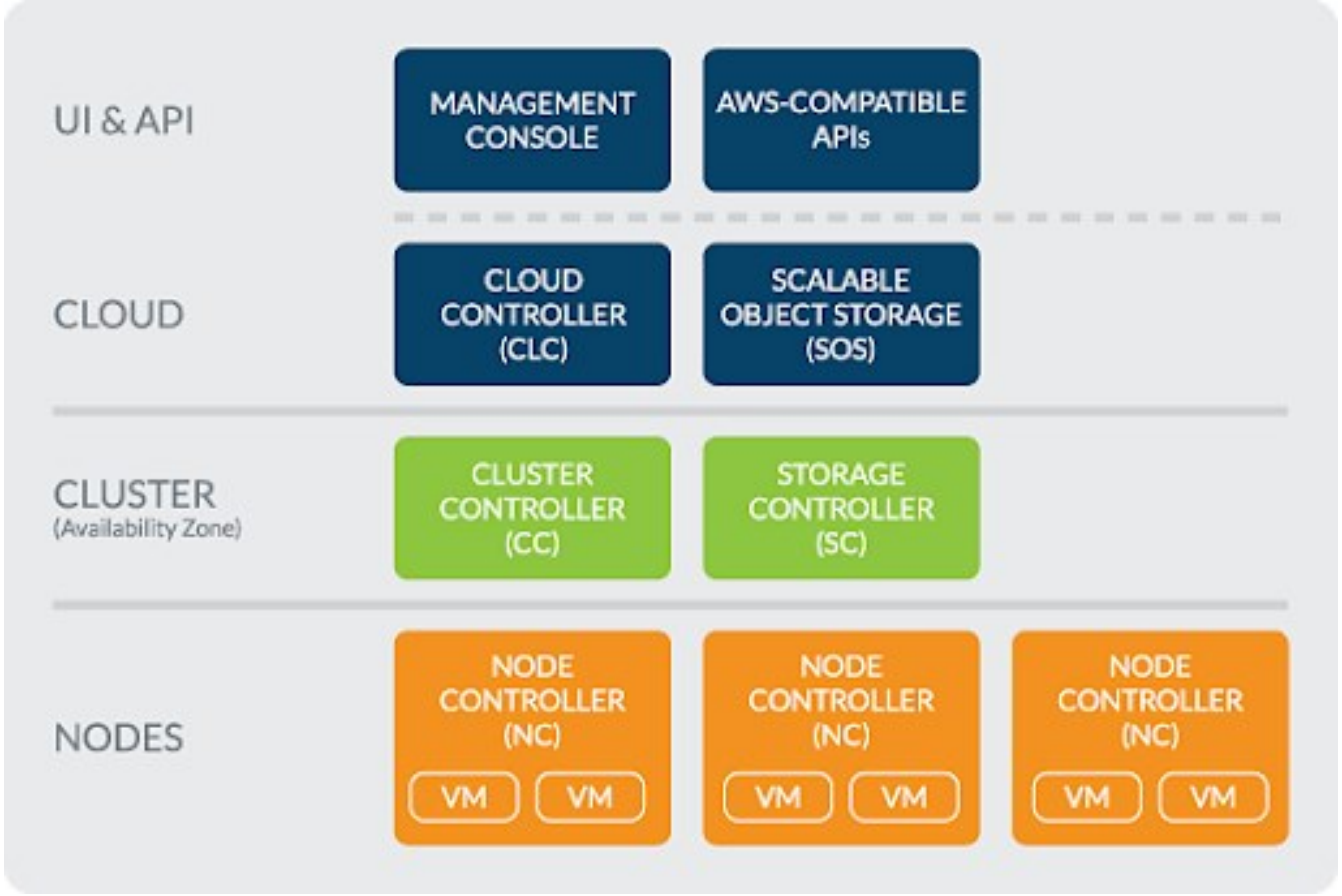
Eucalyptus Cloud Architecture



https://www.eucalyptus.com/eucalyptus-cloud/iaas/architecture

Web Service Components in Eucalyptus

Cloud Controller (CLC): For administrators, developers, project managers, and end-users to manage virtualized resources (servers, network, and storage), and high-level scheduling decisions.



https://docs.eucalyptus.cloud/eucalyptus/4.4.5/admin-guide/system_concepts.html

Web Service Components in Eucalyptus

- **Cluster Controller (CC)** executes on a machine that has network connectivity with machines running the Node Controller (NC) and the CLC. It schedules and manages VM network.
- **Storage Controller (SC)** interface with various storage systems (NFS, iSCSI, SAN devices, etc.).
- **Walrus** allows users to store persistent data, organized as buckets and objects for Virtual Machine (VM) images and user data.
- **Node Controller (NC)** executes on any machine that hosts VM instances, and controls VM activities: execution, inspection, and termination.

https://docs.eucalyptus.cloud/eucalyptus/4.4.5/admin-guide/system_concepts.html

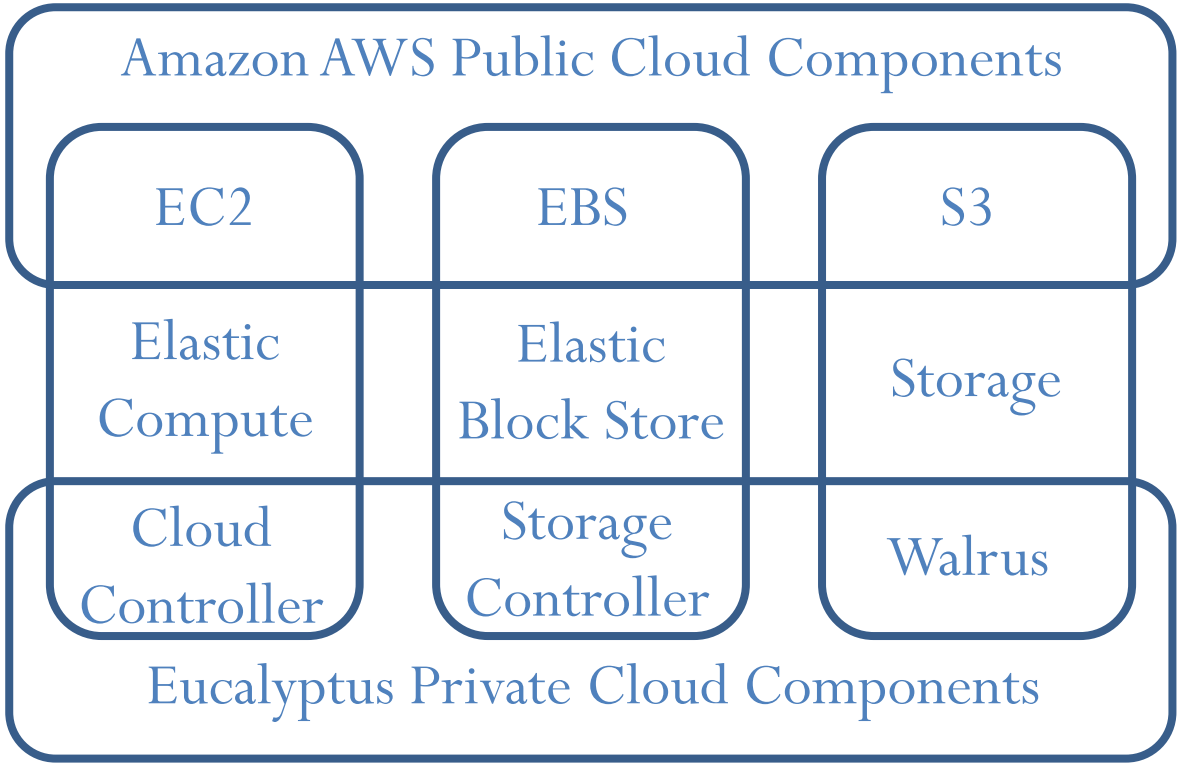
AWS-Eucalyptus Compatibility

Feature: AWS →→ Eucalyptus

Elastic: EC2 → → CC

Elastic Block Store: EBS → → Storage Controller

Storage: S3 → → Walrus



https://docs.eucalyptus.cloud/eucalyptus/4.4.5/admin-guide/system_concepts.html

AWS-Eucalyptus Compatibility

- AWS-Compatible API is implemented on top of Eucalyptus.
- AWS Tools — how to configure and use with Eucalyptus,
  - https://github.com/eucalyptus/eucalyptus/wiki/AWS-Tools
- Consumer use or reuse AWS-Compatible tools, images, and scripts.
- Tools in the eucalyptus cloud ecosystem can communicate with AWS.

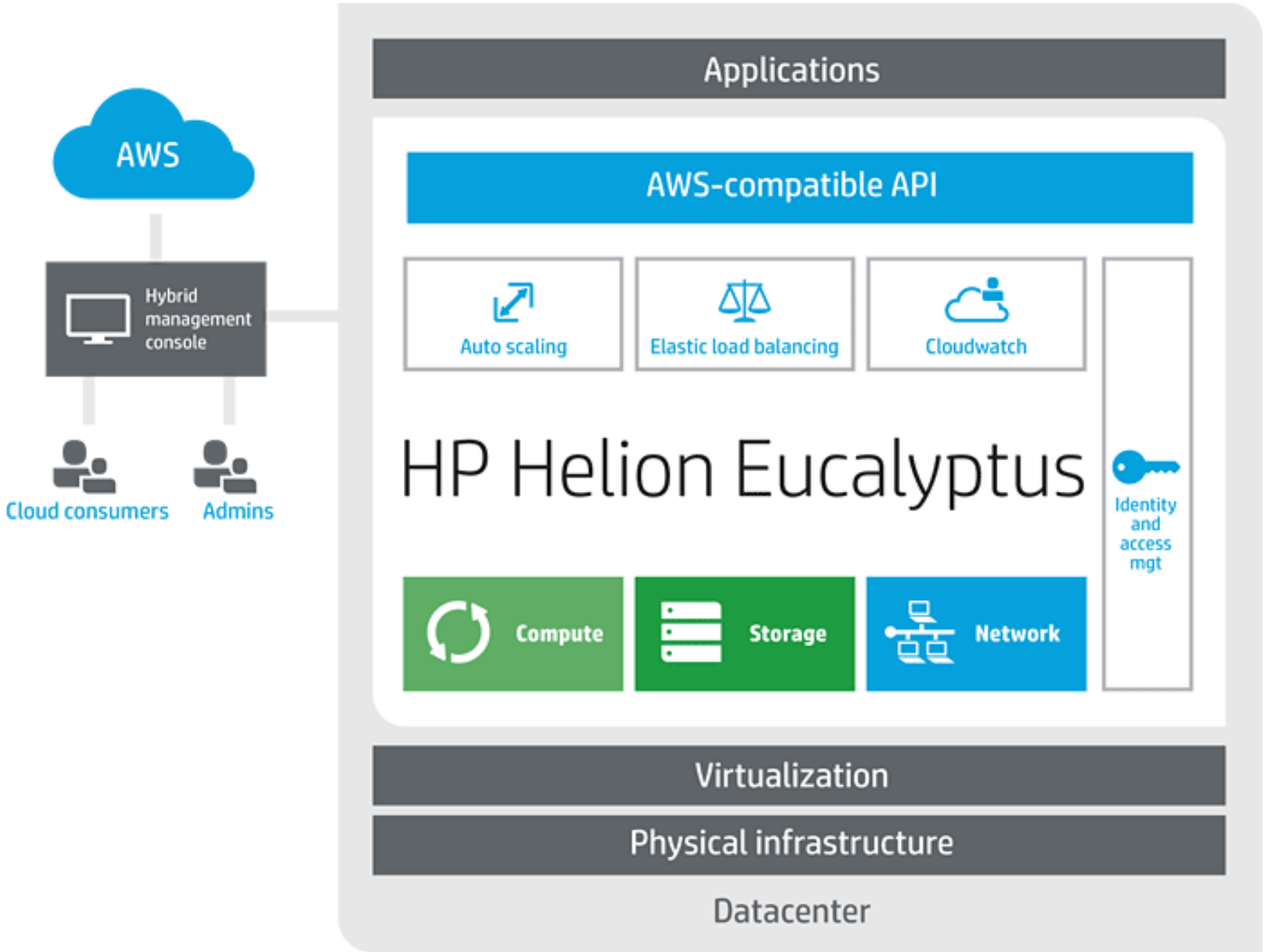
https://github.com/eucalyptus/eucalyptus/wiki

AWS-Eucalyptus Compatibility

Add instances and virtual machines as traffic demands increases

- *Autoscaling* – to scale Eucalyptus cloud resources up or down
  - to maintain performance and meet SLAs.
  - Amazon EC2-compatible APIs and tools.
- *Elastic Load Balancing* –Distributes incoming application traffic and service calls across multiple Eucalyptus workload instances.
- *CloudWatch* – Resources and Applications Monitoring tool like Amazon CloudWatch
  - The collection of metrics, set alarms, and identify trends
  - take action to ensure applications continue to run smoothly.

https://en.wikipedia.org/wiki/Eucalyptus_(software)



https://en.wikipedia.org/wiki/Eucalyptus_(software)

IaaS and Hybrid Cloud

A. Service Orchestration and Virtualization
   - Abstraction layer or abstraction level
   - Virtual machines
   - Type-1, native or bare-metal hypervisors
   - Type-2 or hosted hypervisors
   - Both Type 1 and Type 2

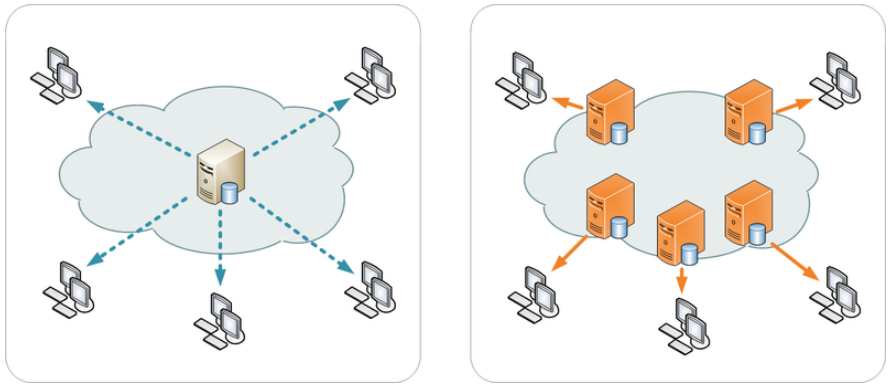
B. Hybrid Eucalyptus and Amazon-EC2

A. Content Delivery Network, or Content Distribution Network (CDN)

B. Hybrid Akamai and Facebook

Content Delivery Network, or Content Distribution Network (CDN)

- CDN is a geographically distributed network of proxy servers and their data centers
  - to provide high availability and performance by distributing the service spatially relative to end users.
  - came in late 1990s to improve the performance bottlenecks of the Internet
- Peer-to-peer CDNs: Consumers provide resources as well as use them. Different from client-server
- Private CDNs: Content owners create their own CDN.



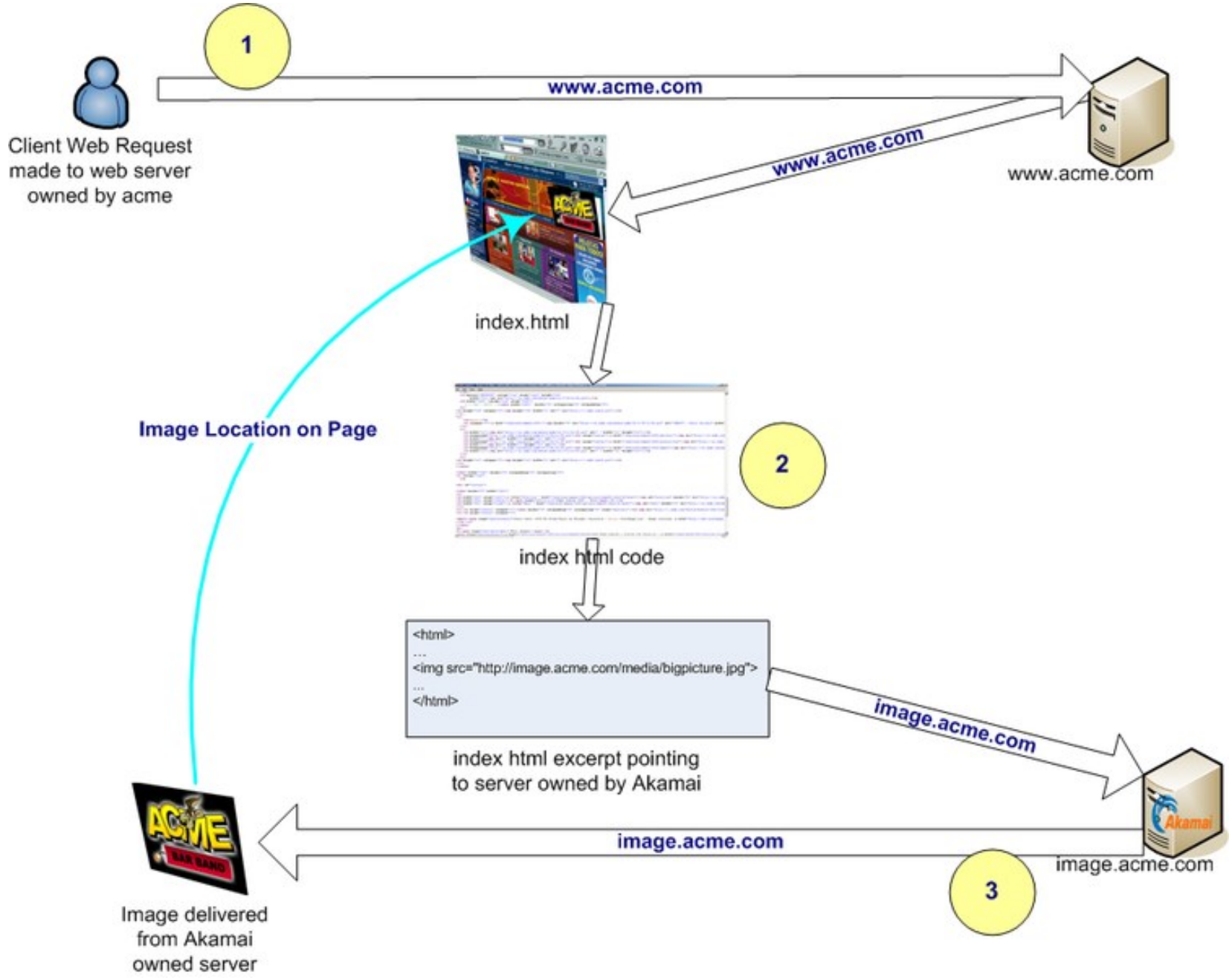
(Left) Single server distribution
(Right) CDN scheme of distribution

https://en.wikipedia.org/wiki/Content_delivery_network

Akamai CDN

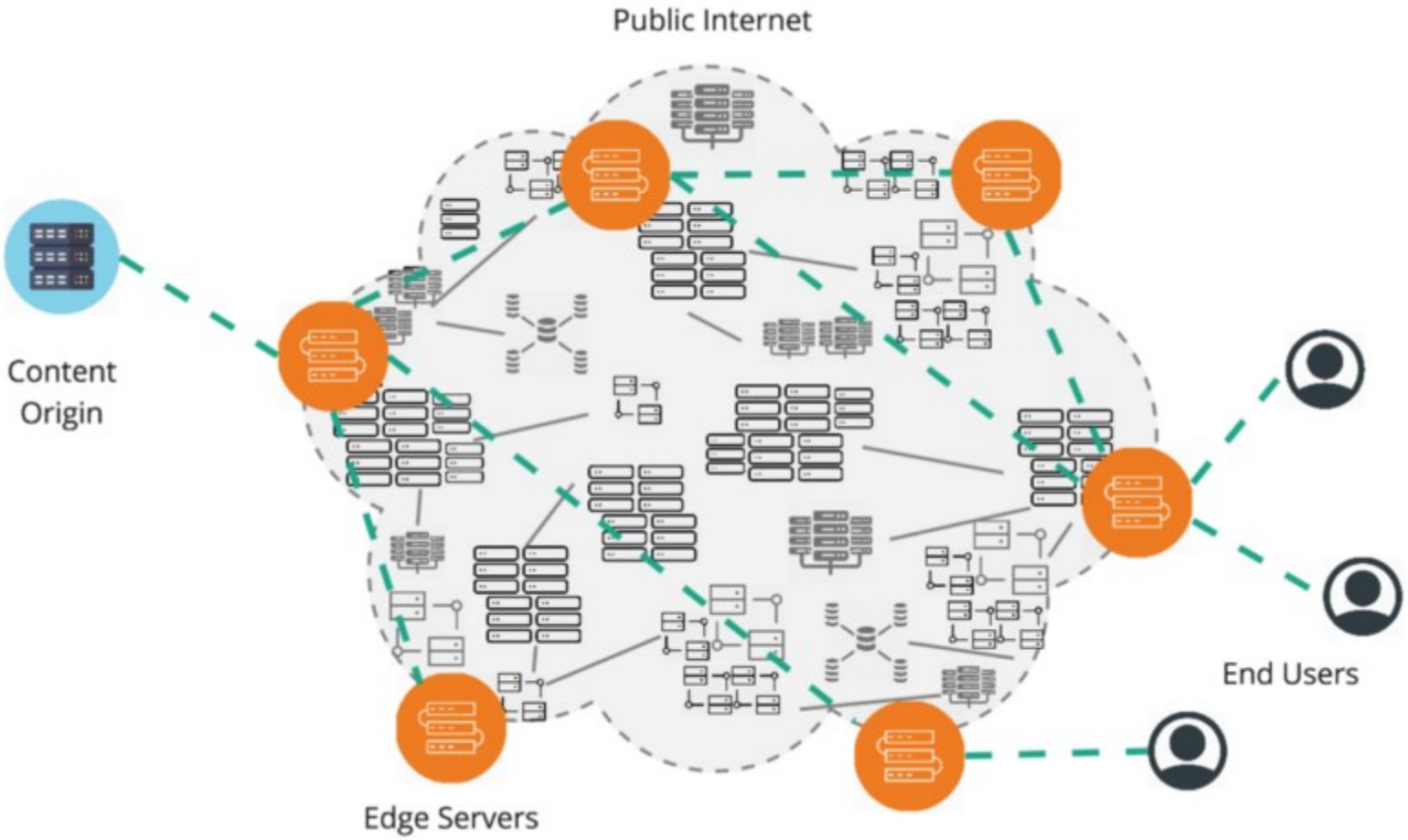
- Global CDN, Cybersecurity, and Cloud service provider.
- Akamai CDN is one of the largest distributed computing platform
  - Serving between 15% and 30% of all web traffic
  - Operates and rents worldwide network of servers
  - Consumer websites work faster by distributing content from locations near the end-user.
  - End-user navigates to the URL of an Akamai's consumer; their browser is redirected to one of Akamai's copies of the website.
- Facebook as Consumer of Akamai CDN uses many edge servers closer to FB consumer (i.e. FB users).

https://en.wikipedia.org/wiki/Akamai_Technologies



https://en.wikipedia.org/wiki/Akamai_Technologies

Akamai Content Delivery Network



https://www.freecodecamp.org/news/an-introduction-to-the-akamai-content-delivery-network-806aa16d8781/

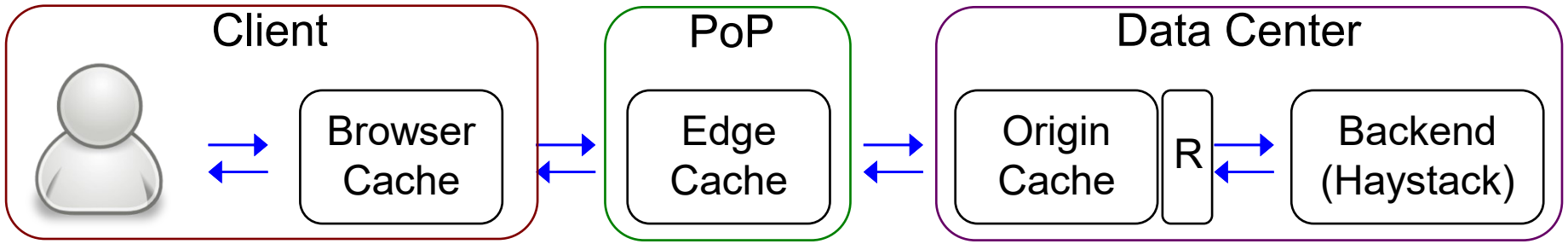
Hybrid Akamai and Facebook

- Facebook Photo Serving Stack: components are linked to show the photo retrieval work-flow.
- Desktop and Mobile clients initiate request traffic, which routes either directly to the Facebook Edge or via Akamai CDN depending on the fetch path.
- The Origin Cache collects traffic from both paths, serving images from its cache and resizing them if needed.
- The Haystack backend holds the actual image content.

Huang, Qi, et al. "An analysis of Facebook photo caching." *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. 2013.

Facebook's Photo-Serving Stack

- **Browser:** The first cache layer is in the client's browser
- **Edge:** The Facebook Edge is comprised of a set of Edge Caches that each run inside points of presence (PoPs) close to end users.
- **Haystack:** The backend layer accessed when there is a miss in the Origin cache (co-located with storage servers), the image can often be retrieved from a local Haystack server.

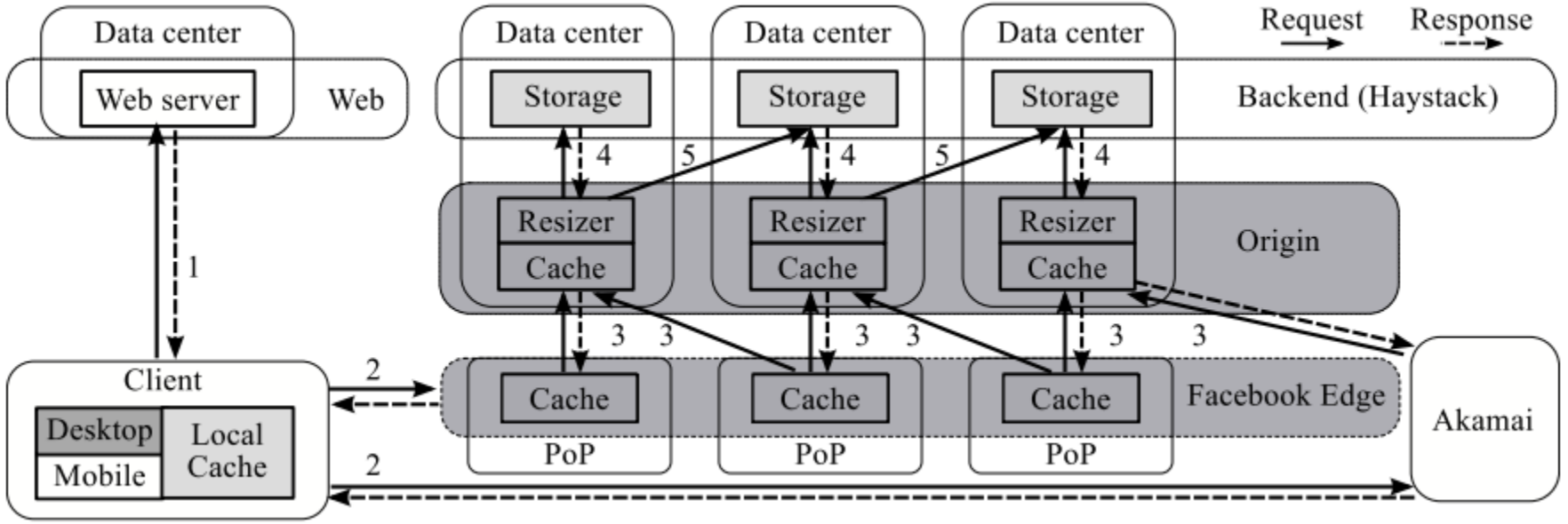


Huang, Qi, et al. "An analysis of Facebook photo caching." *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. 2013.

Hybrid Akamai and Facebook

Architecture and full life cycle of a photo request

- Photo Transformations: Resizers transform photos that are requested by the Akamai CDN,
- Caching Stack



Huang, Qi, et al. "An analysis of Facebook photo caching." *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. 2013.

PaaS and Container as a Service (CaaS) (Google App Engine and DockerHub)

1. IaaS and Hybrid Cloud
   - Orchestration & Virtualization: Eucalyptus & Amazon
   - Content Delivery Network (CDN): Facebook and Akamai
2. **PaaS and Container as a Service (CaaS)**
   - **PaaS: Google App Engine (GAE) and Ruby on Rails**
   - **CaaS: DockerHub**
3. SaaS and Distributed Version Control (DVC)
   - SaaS: Facebook Testing (Infer and Sapienz)
   - DVC: GitHub and Git-LFS

Platform as a service (PaaS)

- PaaS allow customers to do application
  - development,
  - execution, and
  - management

- PaaS removes the complexity of building and maintaining the infrastructure for app
  - development and
  - deployment

https://en.m.wikipedia.org/wiki/Platform_as_a_service

Examples of PaaS Clouds

- Amazon Web Services
- Abiquo Enterprise Edition
- CloudStack
- Citrix Cloud
- CtrlS
- DigitalOcean
- EMC Atmos
- Eucalyptus
- Fujitsu
- GoGrid
- Google Cloud Platform
- GreenButton
- Helion
- GE Predix
- Google App Engine
- GreenQloud
- Heroku
- IBM Cloud
- Inktank
- Jelastic
- Mendix
- Microsoft Azure
- MindSphere
- Netlify
- Oracle Cloud
- OutSystems
- openQRM
- OpenShift
- PythonAnywhere
- RightScale
- Scalr
- Force.com
- SAP Cloud Platform
- VMware vCloud Air
- WaveMaker

https://en.m.wikipedia.org/wiki/Platform_as_a_service

PaaS and Container as a Service (CaaS)

A. **PaaS – Google App Engine (GAE)** and Ruby on Rails
B. Container as a Service (CaaS) - DockerHub

Google File System

- A scalable distributed file system for large distributed data-intensive applications.
- It provides fault tolerance. High aggregate performance to a large number of clients.
- It is widely deployed within Google as the storage platform for the generation and processing of data used by Google service as well as research and development efforts that require large data sets.
- The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients. Earlier Google effort, "BigFiles", developed by Larry Page and Sergey Brin. Files are divided into fixed-size chunks of 64 megabytes.
- GFS is not implemented in the kernel of an operating system, but is instead provided as a userspace library.

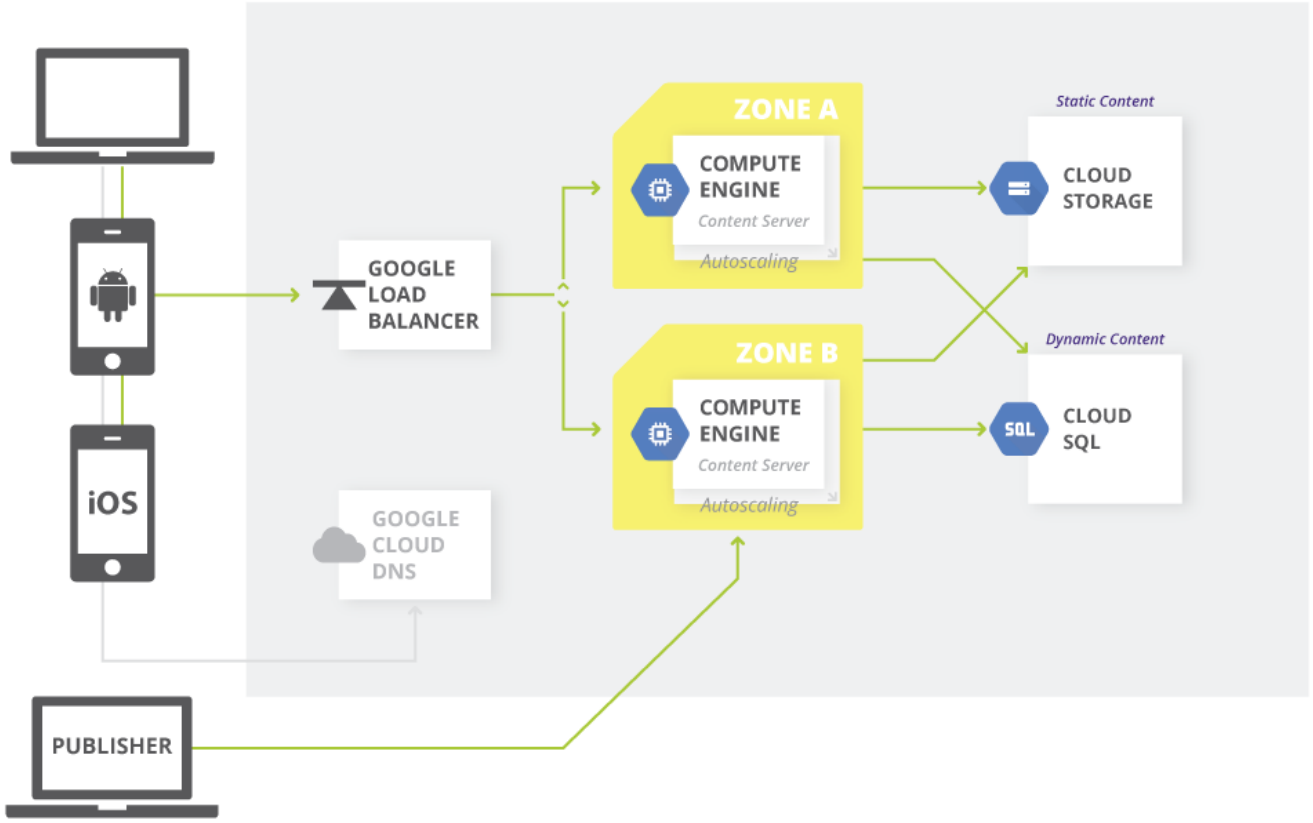
http://research.google.com/archive/gfs.html
http://en.wikipedia.org/wiki/Google_File_System

Google Content Management

- Web information, marketing campaigns or social media.
- Personalized for individual users or groups.
- Google Cloud Platform (GCP) components and services to create a Content Management system.
- Google Load Balancer to support traffic routed to multiple zones for high availability.
- Google's Cloud DNS provides a robust DNS manages the domain.
- Static content → Cloud Storage
- Dynamic content→ Cloud SQL implementation.

https://cloud.google.com/solutions/architecture/contentmanagement accessed 15 Jan 2015

Google Content Management



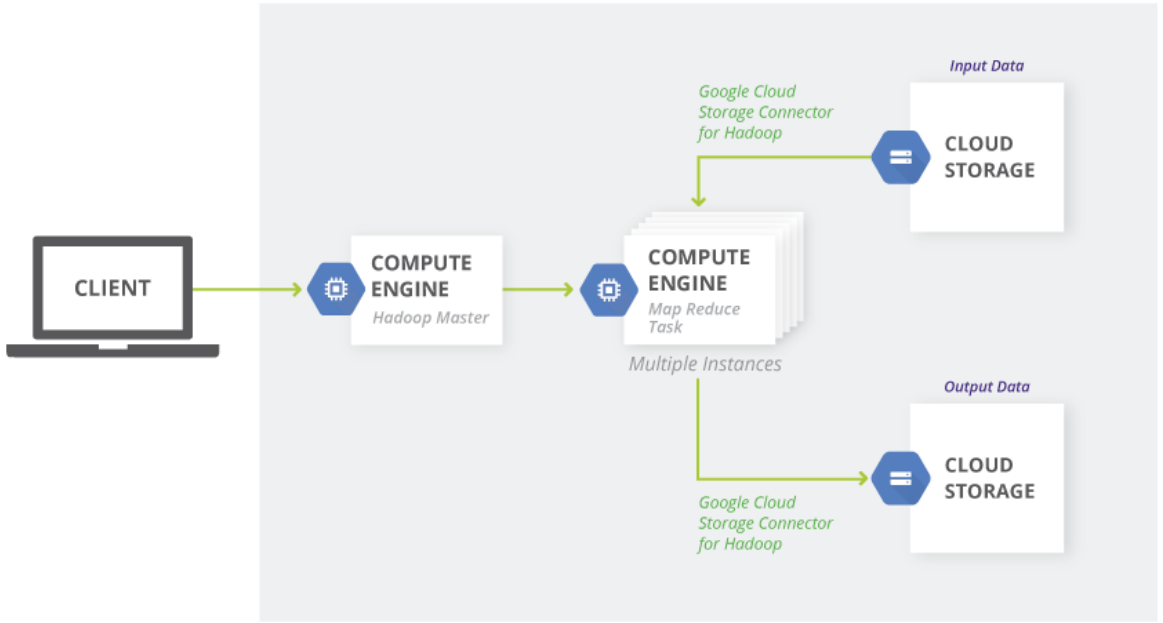
https://cloud.google.com/solutions/architecture/contentmanagement accessed 15 Jan 2015

Architecture: Hadoop on Google Cloud Platform

- Infrastructure for MapReduce using Hadoop.
- Compute power and Cloud Storage to store the input and output of the MapReduce jobs.
- Hadoop Master: includes the HDFS NameNode and the MapReduce JobTracker.
- Nodes in the cluster will run MapReduce tasks with DataNode and MapReduce TaskTracker.
- Backing-up the storage through Google Cloud Storage Connector for Hadoop. HDFS, can be used, Google's Cloud Storage.

https://cloud.google.com/solutions/architecture/hadoop accessed 15 Jan 2015

Architecture: Hadoop on Google Cloud Platform



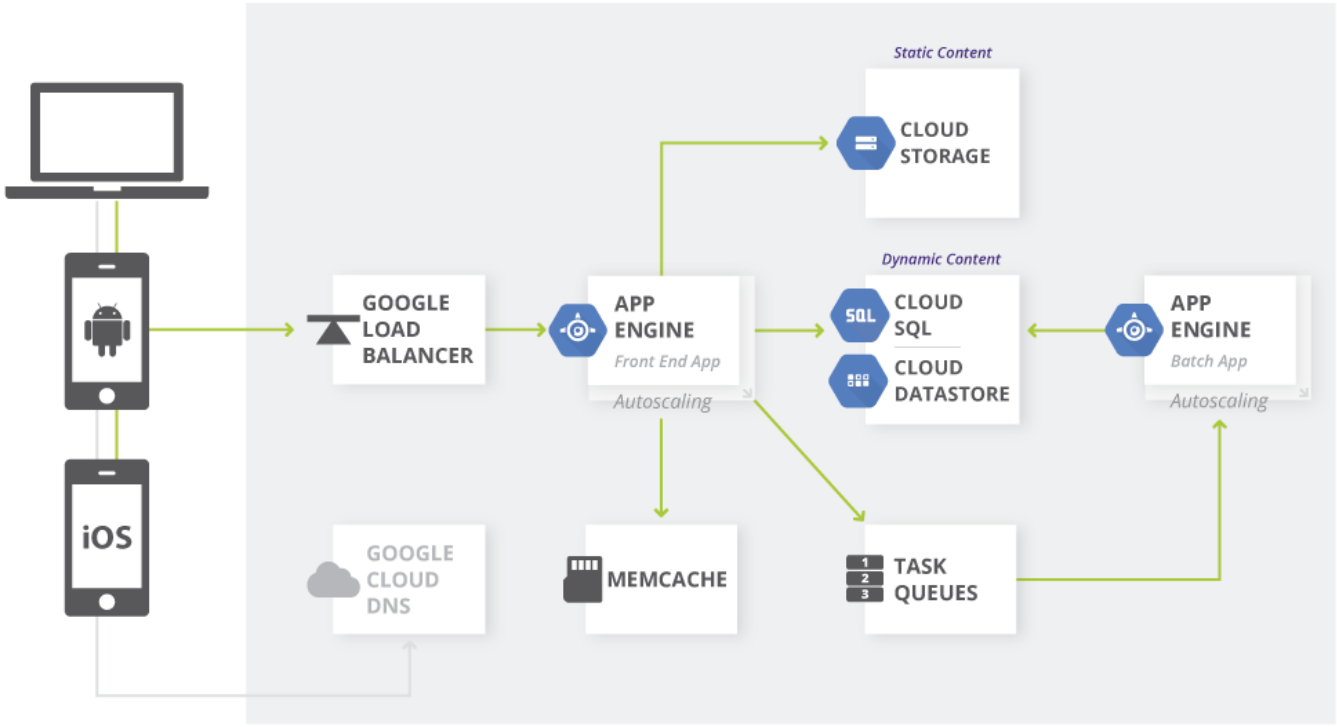
https://cloud.google.com/solutions/architecture/hadoop accessed 15 Jan 2015

Architecture: Web Application on Google App Engine

- Simple development and deployment of Web Applications with autoscaling compute power as well as the integrated features like distributed in-memory cache, task queues and datastore, to create robust applications quickly and easily.
- For applications written in Java, Python, PHP and Go.
- Supports multiple application versions which support A/B testing.
- Memcache is an in-memory cache to provide extremely high speed access to information cached by the web server (e.g. authentication or account information).
- Task Queues provide a mechanism to offload longer running tasks to backend servers, freeing the front end servers to service new user requests.
- Google Load Balancer which provides transparent load balancing to applications.
- Google's Cloud DNS is used to manage DNS domain of user.

https://cloud.google.com/solutions/architecture/webapp accessed 15 Jan 2015

Architecture: Web Application on Google App Engine



https://cloud.google.com/solutions/architecture/webapp accessed 15 Jan 2015

Google Web Tool - Kit

- GWT gives us API to design rich web applications.
- GWT is a Swing-like Java framework. Developer can write web application without writing HTML or JavaScript code.
- GWT is a development environment similar to any Web-Server-Code or Desktop-app development environment.
- GWT helps to debug, re-factor and unit test a Web-Client.
- GWT provides a so-called hosted mode, that allows developers to debug Java code, as well as a web mode which executes the GWT-generated JavaScript code.
- **Google uses GWT for its Sites:** Google Docs, Google AdSense, Google Wallet
- **Other Sites:** gogrid.com, Scenechronize, Google Moderator, Whirled. See more at http://gwtgallery.appspot.com/

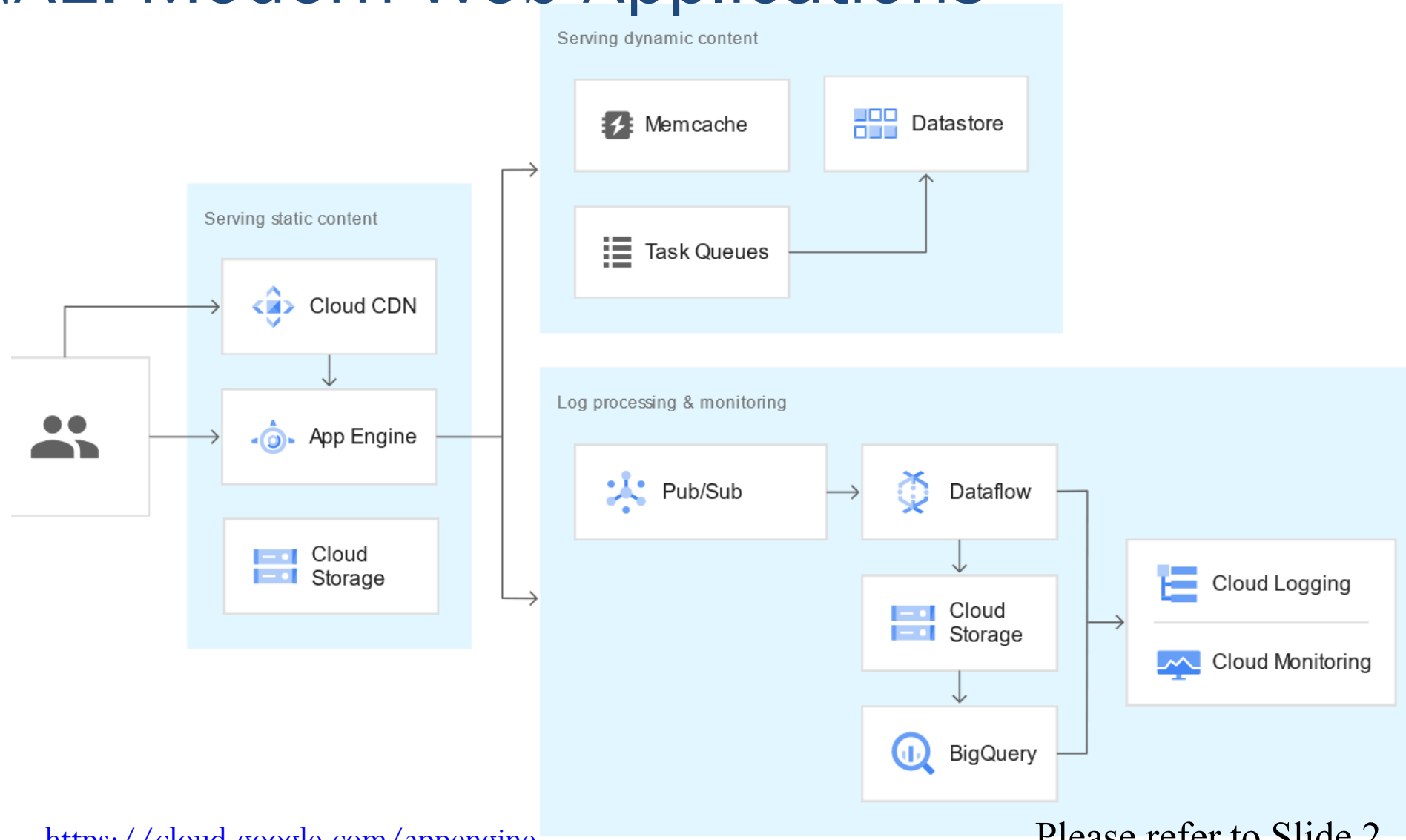
GAE: Modern Web Applications

Web app using Google App Engine (GAE) and Google Cloud.

- Quickly reach customers and end users by deploying web apps on App Engine.
- Zero-config deployments and zero server management.
- Only writing code for App Engine.
- App Engine automatically scales to support sudden traffic spikes without provisioning, patching, or monitoring.

https://cloud.google.com/appengine

GAE: Modern Web Applications



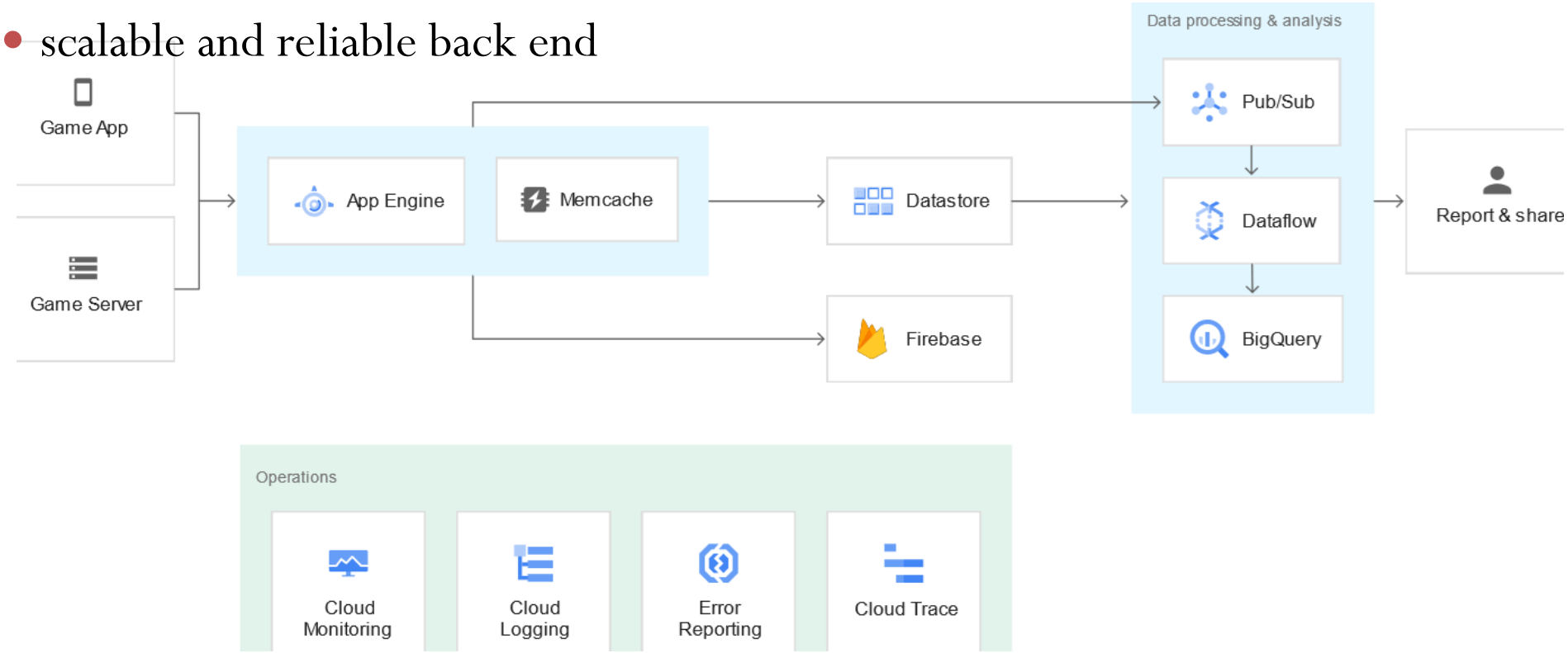
https://cloud.google.com/appengine

Please refer to Slide 2

GAE: Scalable Mobile Back-ends

Mobile app built with Firebase and Google services.

- automatically scales the hosting environment.
- an easy-to-use frontend mobile platform
- scalable and reliable back end



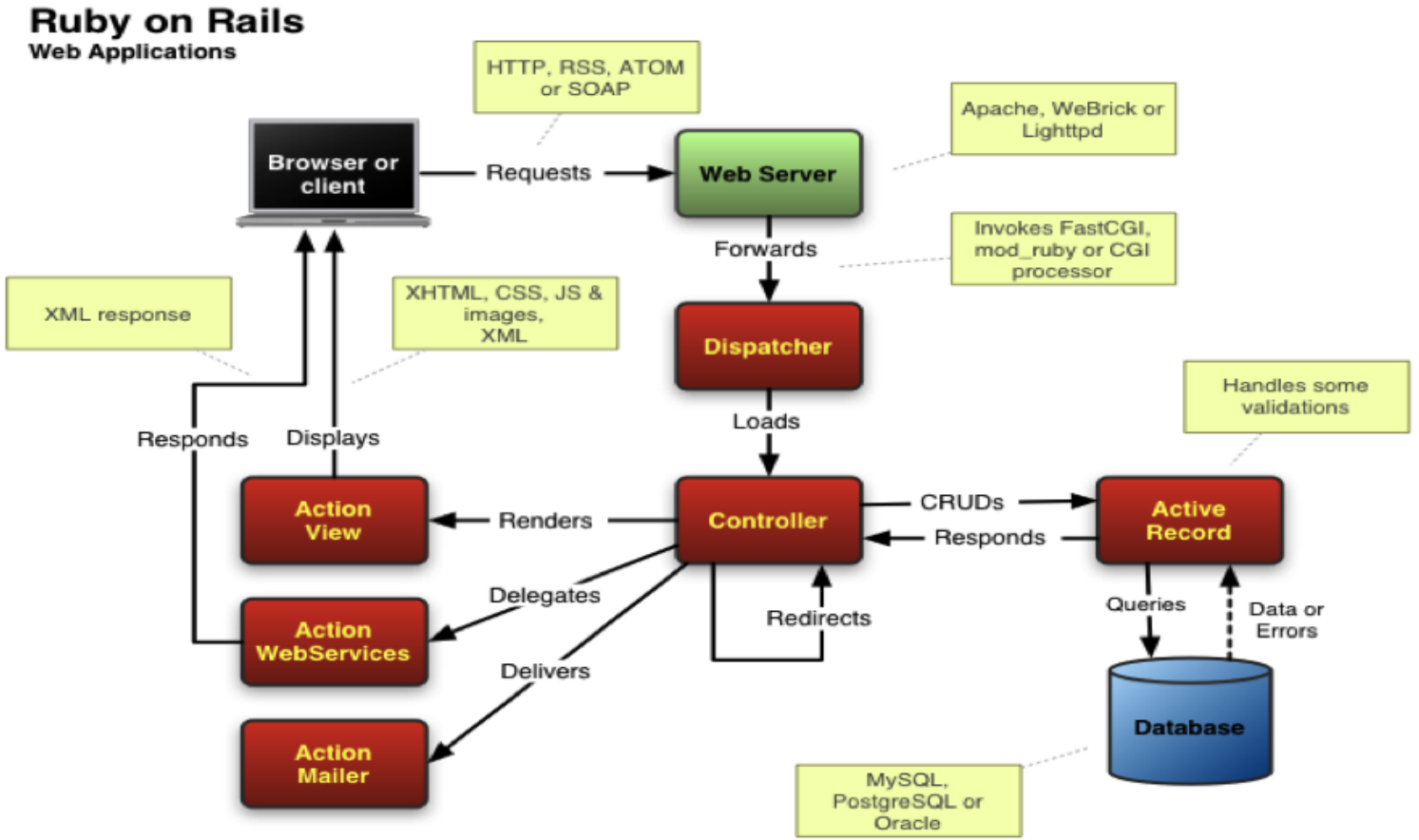
https://cloud.google.com/appengine

PaaS and Container as a Service (CaaS)

1. PaaS – Google App Engine (GAE) and **Ruby on Rails**
2. Container as a Service (CaaS) – DockerHub

Ruby on Rails

- A web applications development framework. Very popular in the agile development with the help of convention over configuration management. Ruby on Rails uses the Model-View-Controller (MVC)



http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/

Ruby on Rails

RESTful Architecture

- Representational State Transfer (REST) is an alternative to web services, such as SOAP and WSDL.
- Works on HTTP protocol for operations: Create, Read, Update and Delete (CRUD).
- RESTful is useful when it is required stateless, limited bandwidth (specially for mobile devices no overhead of SOAP), and when service provider and user have complete information of operations.

Ruby on Rails

- MVC separates business logic from HTML views. Architectural pattern in order to improve the maintainability of the application.
- **Model:** Carries the business logic and rules to manipulate the data.
  - Models represent the information in the database.
  - Manages interaction with database.
- **View:** Front-end of the application, representing the user interface.
  - HTML files with embedded Ruby code. Used to display data in the form of views
  - Formats, such as HTML, PDF, XML, RSS and more.
- **Controller:** it interact with models and views.
  - Incoming requests are processed by the controllers.
  - Controller process the data from the models and pass it to the views for presentation.

PaaS and Container as a Service (CaaS)

1. PaaS – Google App Engine (GAE) and Ruby on Rails
2. **Container as a Service (CaaS) – DockerHub**

OS-level virtualization

OS-level virtual image looks like real computers from the point of view of programs running on

- **Containers** (LXC, Solaris containers, Docker),
- **Zones** (Solaris containers), **Partitions**,
- **Virtual private servers** (OpenVZ), **Virtual environments** (VEs), **Virtual kernels** (DragonFly BSD), or
- All resources (connected devices, files and folders, network shares, CPU power, quantifiable hardware capabilities).
- Programs running inside of a container can only see the container's contents and devices assigned to the container.
- *Container* refer to OS-level virtualization systems, e.g. Microsoft's Hyper-V containers.

https://en.m.wikipedia.org/wiki/OS-level_virtualization

Container image

- Container is a running process interacting with its own private filesystem provided by a **Container image**.
- Container image runs an application with
  - the code or binary,
  - runtime dependencies, and
  - any other filesystem objects required.
- Docker is a platform for developers and sysadmins to **build, run, and share** applications with containers.

https://docs.docker.com/get-started/

DockerHub is a CaaS

- Solomon Hykes and Sebastien Pahl
- OS-level virtualization to deliver software in packages called containers.
  - hosts the containers is called **Docker Engine**
- Docker can package an application and its dependencies in a virtual container that can run on any Linux server.
- Containers are isolated from one another and bundle their own software, libraries, and configuration files;
- All containers are run by a single operating system kernel and therefore use fewer resources than VMs.

https://en.m.wikipedia.org/wiki/Docker_(software)

Docker Computational reproducibility

Docker implement and deployment of containers, includes:

(1) cross-platform portability,

(2) modular/component re-using and sharing,

(3) Linux container (LXC) based OS level virtualization,

(4) portable across platforms, and

(5) archiving and versioning of container images.

Boettiger, Carl. "An introduction to Docker for reproducible research." *ACM SIGOPS Operating Systems Review* 49.1 (2015): 71-79.

Dependency and Coupling

- Interdependence and relationships between software modules;
  - Connected two or more routines or modules;
  - Coupling can be "low/loose" or "high/tight"
  - Low coupling provides a well-structure and good design software,
- Disadvantage of High/Tightly coupled:
  - Change in one module forces changes in other dependent modules.
  - Modularity require effort or time due to inter-module dependency.
  - Harder to reuse and test modules due to dependent modules.
- Performance reduction by message and parameter
  - request/response messages require CPU and memory in
  - message creation, transmission, translation (e.g. marshaling) and interpretation (string, array or data structure)

https://en.wikipedia.org/wiki/Coupling_(computer_programming)

Docker images: resolve Dependencies

- Docker image is based on a Linux system with Linux-compatible software including
  - R,
  - Python,
  - Matlab, and
  - most other programming environments.

- Docker image and Virtual Machine (VM) image
  - Similarity: resolves the dependency with a pre-installed and pre-configured binary image of dependencies.
  - Dissimilarities: Docker images share the Linux kernel with the host machine.

Boettiger, Carl. "An introduction to Docker for reproducible research." *ACM SIGOPS Operating Systems Review* 49.1 (2015): 71-79.

Dockerfile

- Docker can build images automatically by reading the instructions from a *Dockerfile*.
- Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- Docker build create an automated build that executes several command-line instructions.

  docker build **-f** /path/to/a/Dockerfile

- Copy and paste to pull this image

  docker pull hello-world

Boettiger, Carl. "An introduction to Docker for reproducible research." *ACM SIGOPS Operating Systems Review* 49.1 (2015): 71-79.
https://hub.docker.com/_/hello-world

Docker concepts

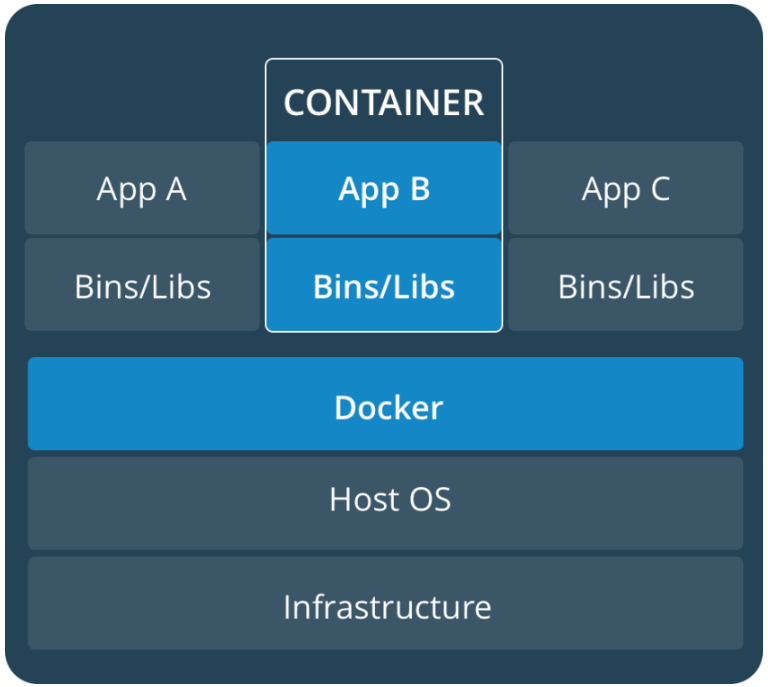
The use of containers to deploy applications is called *containerization*.

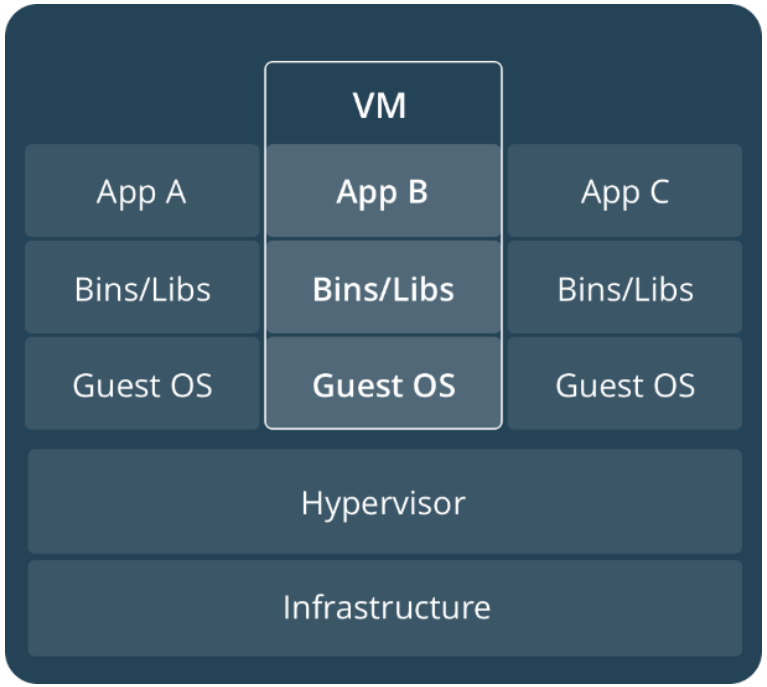
- **Flexible**: Complex applications can be containerized.
- **Lightweight**: Containers leverage and share the host kernel, making them efficient resource usage as compared to VMs.
- **Portable**: Build locally, deploy to the cloud, and run anywhere.
- **Loosely coupled**: Containers are highly self sufficient and encapsulated allow to replace or upgrade.
- **Scalable**: Distribute container replicas across a datacenter.
- **Secure**: Constraints and isolations to processes without any configuration required on the part of the user.

https://docs.docker.com/get-started/

Containers & Virtual machine (VM)

- Container runs *natively* on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking memory like an executable.
- VM runs a full "guest OS" with hypervisor access to host resources. VM uses overhead resources that are not consumed by application logic.





https://docs.docker.com/get-started/

Build and run your Docker image

git clone https://github.com/dockersamples/node-bulletin-board

cd node-bulletin-board/bulletin-board-app

docker build **--tag** bulletinboard:1.0

docker run **--publish 8000:8080 --detach --name bb** bulletinboard:1.0

```
# Use the official image as a parent image.
FROM node:current-slim

# Set the working directory.
WORKDIR /usr/src/app

# Copy the file from your host to your current location.
COPY package.json .

# Run the command inside your image filesystem.
RUN npm install

# Add metadata to the image to describe which port the container is listening on at runtime.
EXPOSE 8080

# Run the specified command within the container.
CMD [ "npm", "start" ]

# Copy the rest of your app's source code from your host to your image filesystem.
COPY . .
```

https://docs.docker.com/get-started/part2/

Why PaaS or CaaS

Programming tools, Deployment tools, and Application hosting

- CaaS include stack of tools for deploying containers.
- CaaS do not include development tools.
- CaaS is a subset of PaaS functionality
- CaaS main use case is containerized applications
- CaaS is an incomplete form of a PaaS.
- PaaS and CaaS use-cases:
  - PaaS: integrated solution for develop and deploy applications
  - CaaS: easy to set-up and manage a container environment
- Benefit from a development at PaaS or not? Yes, then PaaS
- Pre-developed application for public deployment or not?Yes, then CaaS

https://containerjournal.com/features/paas-vs-caas-wrong-question-ask/

PaaS and CaaS Hybrid

- PaaS-CaaS hybrid provides both development environment and deployment container
  - Amazon Elastic Container Service (ECS),
  - Ruby-on-Rails (EngineYard) and
  - Google App Engine etc.

https://containerjournal.com/features/paas-vs-caas-wrong-question-ask/
https://aws.amazon.com/ecs/
https://www.engineyard.com/
https://cloud.google.com/appengine

SaaS and Distributed Version Control (Facebook Testing, GitHub and Git-LFS)

1. IaaS and Hybrid Cloud
   - Orchestration & Virtualization: Eucalyptus & Amazon
   - Content Delivery Network (CDN): Facebook and Akamai
2. PaaS and Container as a Service (CaaS)
   - PaaS: Google App Engine (GAE) and Ruby on Rails
   - CaaS: DockerHub
3. **SaaS and Distributed Version Control (DVC)**
   - **SaaS: Facebook Testing (Infer and Sapienz)**
   - **DVC: GitHub and Git-LFS**

SaaS and Distributed Version Control

A. **Scalable Facebook Testing:**
   - **Infern and**
   - **Sapienz**
   - **FBLearner**
   - **Facebook's OneWorld platform**

B. Distributed Version Control:
   - Version Control System (VCS), Central Version Control (CVC), Distributed Version Control (DVC)
   - CVC v/s DVC
   - Git, GitHub, GitHub Desktop (RaaS Client), and
   - Git-LFS

Scalable Facebook Testing

- Scale means different things to different techniques to scale (or fail to scale) in different dimensions
- Scalability needed to assess the degree of deployability
- Challenges and Opportunities when deploying
  - Static analysis and Dynamic analysis at scale
- Facebook deploy two technologies
  - the Infer for static analysis and
  - the Sapienz dynamic analysis
  - Both are research-led start-up that was subsequently deployed at scale, impacting billions of people worldwide.

Harman, Mark, and Peter O'Hearn. "From start-ups to scale-ups: Opportunities and open problems for static and dynamic program analysis." *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2018.

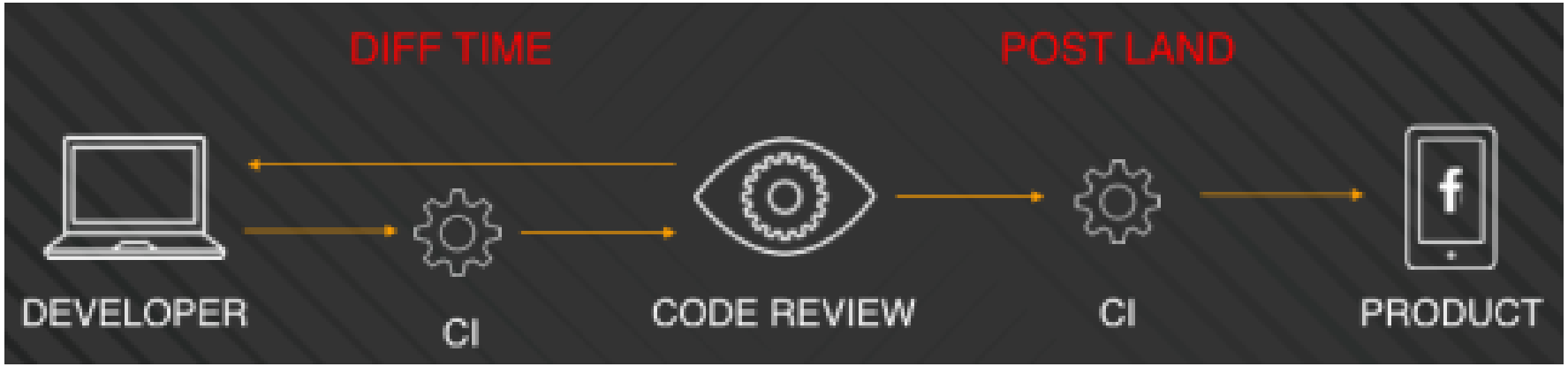
Infer: Static Analysis at Scale

- At Facebook, Infer is a ‘deep’ static analysis tool applied to
  - Java, Objective C, and C++ code bases
- Infer at Facebook with acquisition of startup Monoidics in 2013
- Infer is open source and used at several companies
  - AWS, Mozilla, JD.com and Spotify
- Based on Separation Logic to scale algorithms for reasoning "states"
  - of a *store* and a *heap* about memory,
  - local (or stack-allocated) variables and dynamically-allocated,
  - implements a compositional program analysis: inter-procedural analysis,
  - follows pointer chains, safety of programs with embedded pointers,
  - scales to large code bases
- Result: 10s of 1000s bugs fix by Facebook developers before production.

Harman, Mark, and Peter O'Hearn. "From start-ups to scale-ups: Opportunities and open problems for static and dynamic program analysis." *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2018.

Infer: Static Analysis at Scale

- Facebook practices Continuous Development and Deployment i.e. Continuous Integration (CI) of "Diffs" (a code change) for code-review.
- Developers shares a single code base to alter and commit. Then, prepares a diff and submits it for the code review.
- Infer runs as a bot during diff time, writing comments for the developer and other human reviewers. Post-land incorporate the master build of the system.



Harman, Mark, and Peter O'Hearn. "From start-ups to scale-ups: Opportunities and open problems for static and dynamic program analysis." *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2018.

Sapienz: Dynamic Analysis at Scale

- Sapienz is deployed to run continuously testing the most recent internal builds of the Facebook apps.
- Sapienz is a multi-objective automated test case design system, that seeks to maximize fault revelation while minimizing the debug effort to fix.
- It uses
  - FBLearner (Facebook's Machine Learning infrastructure)
  - Facebook's OneWorld platform

Harman, Mark, and Peter O'Hearn. "From start-ups to scale-ups: Opportunities and open problems for static and dynamic program analysis." *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2018.

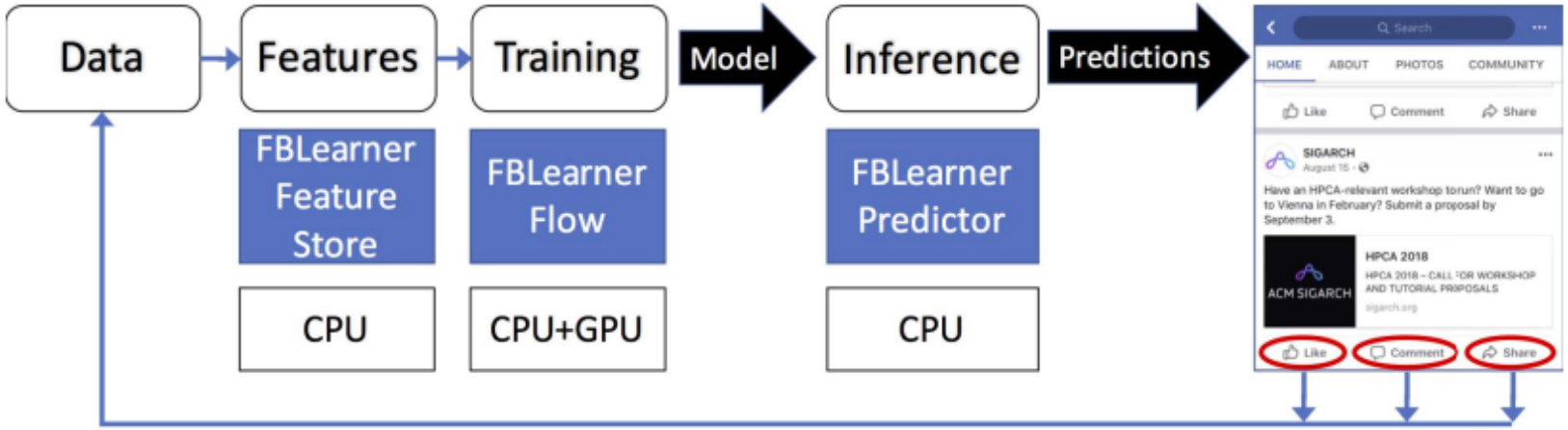
Sapienz: Dynamic Analysis at Scale

- FBLearner (Facebook's Machine Learning infrastructure)
  - Machine Learning (ML) perform at a scale, e.g., ML inference phase executions run into the tens of trillions per day.
  - FBLearner is a ML platform designed to support at a global scale
  - Sapienz is 1 of 100s services deployed on FBLearner framework
  - FBLearner Flow component in Sapienz
    - deploys detection of crashing behaviour directly into the work flow,
    - integrates with Phabricator for reporting and actioning fixes
    - corrects the failures detected
- Facebook's OneWorld platform
  - support scalable deployment on many of emulators, at a time
  - Sapienz deploys on 1000s of emulators to test the Android app alone

Harman, Mark, and Peter O'Hearn. "From start-ups to scale-ups: Opportunities and open problems for static and dynamic program analysis." *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2018.

FBLearner

ML-as-a-Service Inside Facebook: FBLearner Feature Store, FBLearner Flow, FBLearner Predictor.



MACHINE LEARNING ALGORITHMS LEVERAGED BY PRODUCT/SERVICE

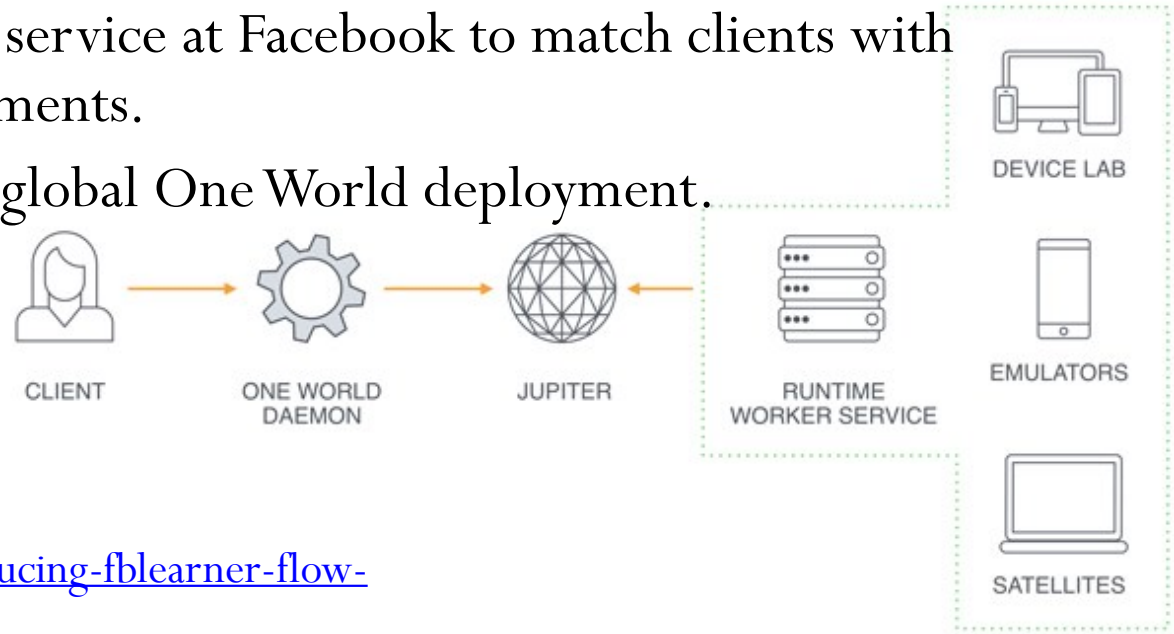
Models	Services
Support Vector Machines (SVM)	Facer (User Matching)
Gradient Boosted Decision Trees (GBDT)	Sigma
Multi-Layer Perceptron (MLP)	Ads, News Feed, Search, Sigma
Convolutional Neural Networks (CNN)	Lumos, Facer (Feature Extraction)
Recurrent Neural Networks (RNN)	Text Understanding, Translation, Speech Recognition

Hazelwood, Kim, et al. "Applied machine learning at facebook: A datacenter infrastructure perspective." *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018.

Facebook's OneWorld platform

Support standard communication mechanisms like ADB (Android Debug Bridge) and Provide the illusion that remote devices are connected locally

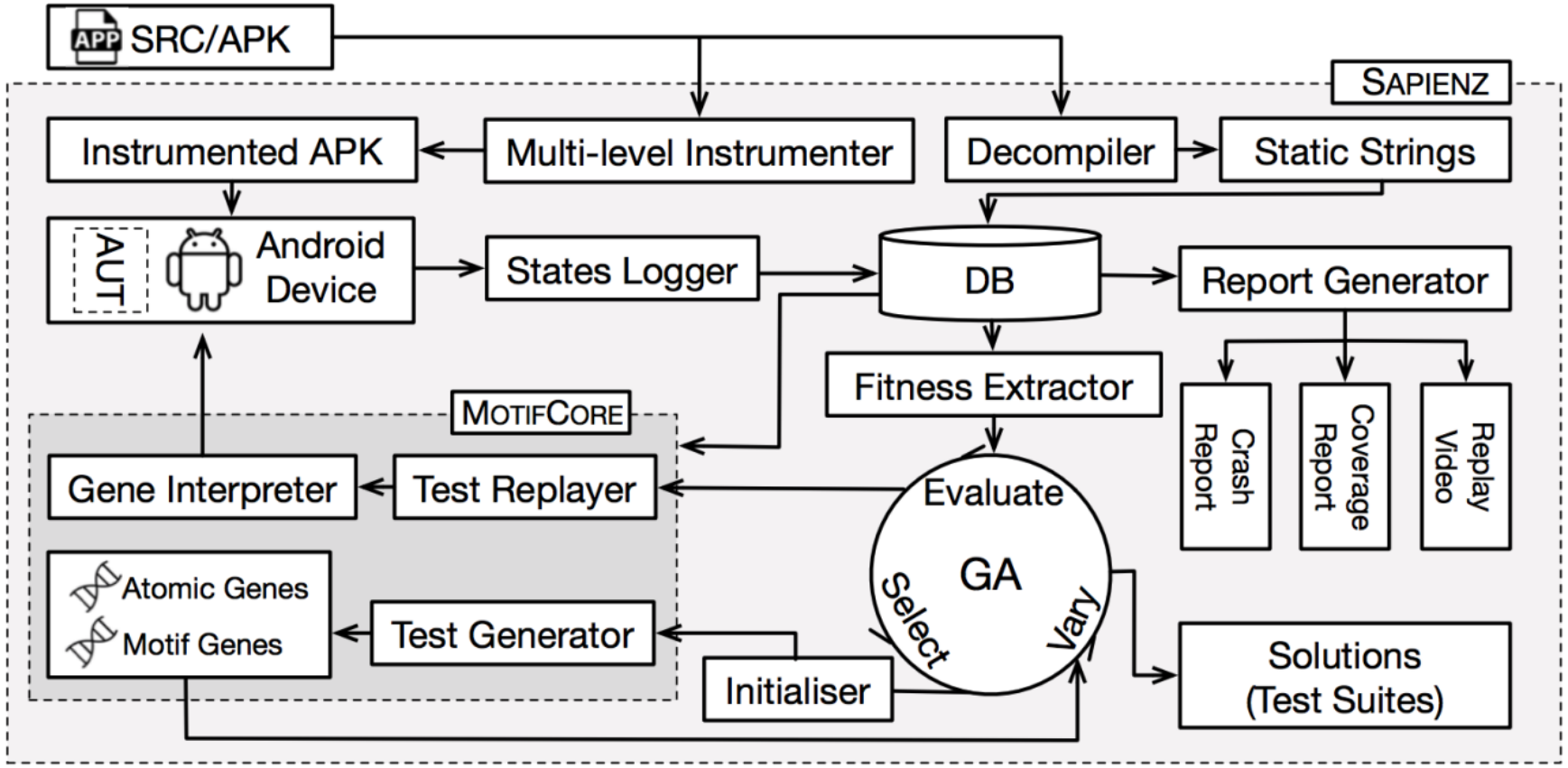
- **Runtime worker service:** it manages the life cycle of the resource and responds to requests from clients to use its resources.
- **One World daemon:** it connect to remote resources with protocol to communicate with workers and sets up the environment.
- **Scheduler:** Uses Jupiter, a job-scheduling service at Facebook to match clients with workers' resources as per specified requirements.
- **Satellite:** it connect local resources to the global One World deployment.



https://engineering.fb.com/2016/05/09/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/

Sapienz workflow

Sapienz Automated test design workflow



Harman, Mark, and Peter O'Hearn. "From start-ups to scale-ups: Opportunities and open problems for static and dynamic program analysis." *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2018.

SaaS and Distributed Version Control

A. Scalable Facebook Testing:
   - Infern and
   - Sapienz
   - FBLearner
   - Facebook's OneWorld platform

B. Distributed Version Control:
   - **Version Control System (VCS), Central Version Control (CVC), Distributed Version Control (DVC)**
   - **CVC v/s DVC**
   - **Git, GitHub, GitHub Desktop (RaaS Client), and**
   - **Git-LFS**

Version Control System (VCS)

- Version control is also known as
  - revision control,
  - source control, or
  - source code management
- It is responsible for managing changes to
  - computer programs,
  - documents,
  - large web sites, or
  - other collections of information.
- It is a component of Software Configuration Management (SCM).
- VCS are stand-alone applications.

https://en.wikipedia.org/wiki/Version_control

Centralised Version Control (CVC)

- Single server holds the code base
- Clients access the server with check-in/check-outs
- CVC available are
  - CVS,
  - Subversion,
  - Visual Source Safe.
- Pro: Simple to manage a single server
- Con: Single point of failure.

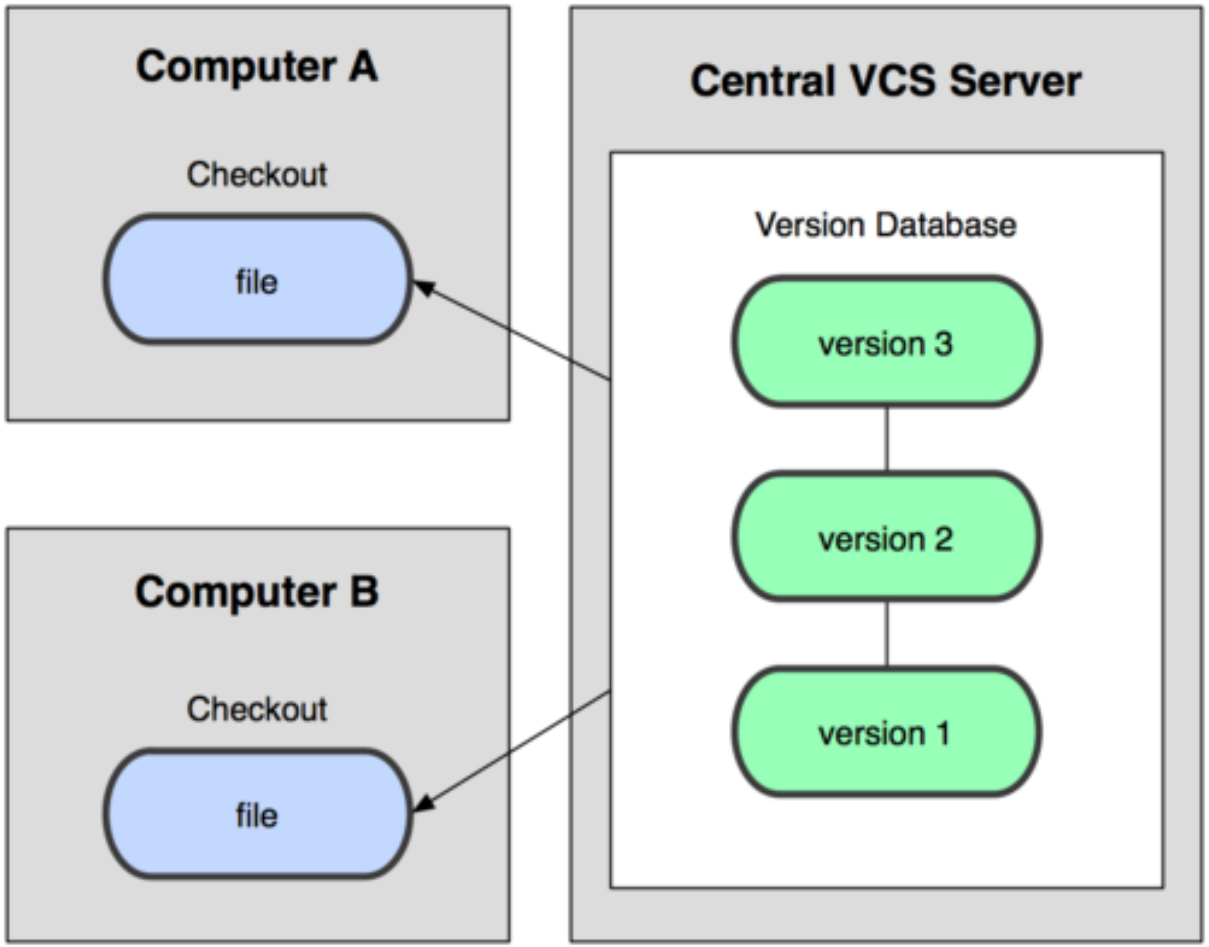
Distributed Version Control (DVC)

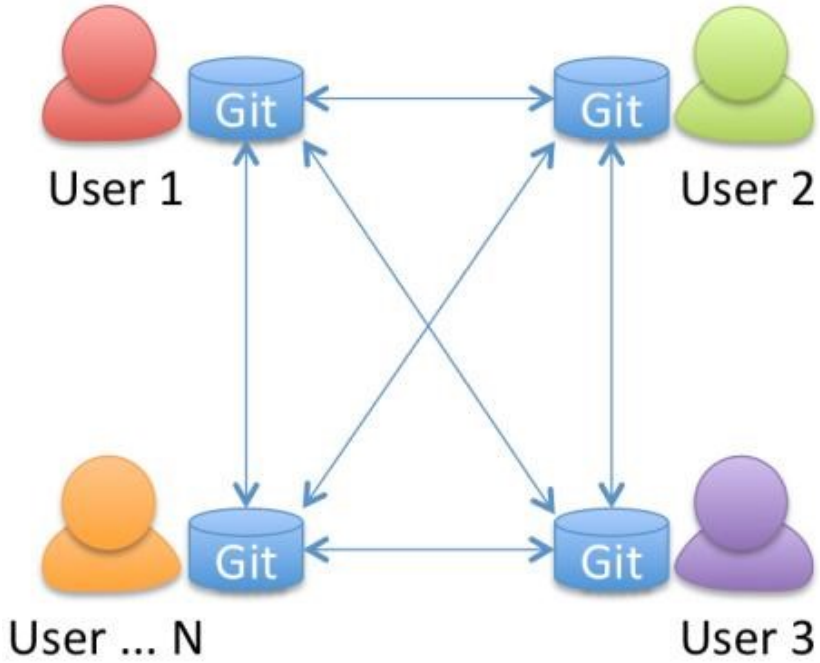
- aka, Distributed Revision Control
- It is a form of version control that complete codebase, including its full history, is mirrored on every developer's computer. Enables
  - automatic management branching and merging,
  - speeds up most operations (except pushing and pulling),
  - improves the ability to work offline and
  - does not rely on a single location for backups.
- Each client holds a copy of the code base.
- Code is shared between clients by push/pulls
  - Advantages: Many operations, no single point of failure
  - Disadvantages: Complex to manage

https://en.wikipedia.org/wiki/Distributed_version_control

Naisan Benatar "Version Control" G51FSE

CVC v/s DVC


Computer A
Checkout
file
Computer B
Checkout
file
Central VCS Server
Version Database
version 3
version 2
version 1


Git
Git
User 1
User 2
Git
Git
User ... N
User 3

Git

- **Git** is a distributed version-control system.
- It track changes in source code files during software development.
- It is designed for coordinating work among programmers,
- Its goal include speed, data integrity, and support for distributed, non-linear workflows.
- Git was created by Linus Torvalds in 2005 for development of the Linux kernel.
- Every Git directory on every developer's computer is a repository, and independent of network or central server.

https://en.wikipedia.org/wiki/Git

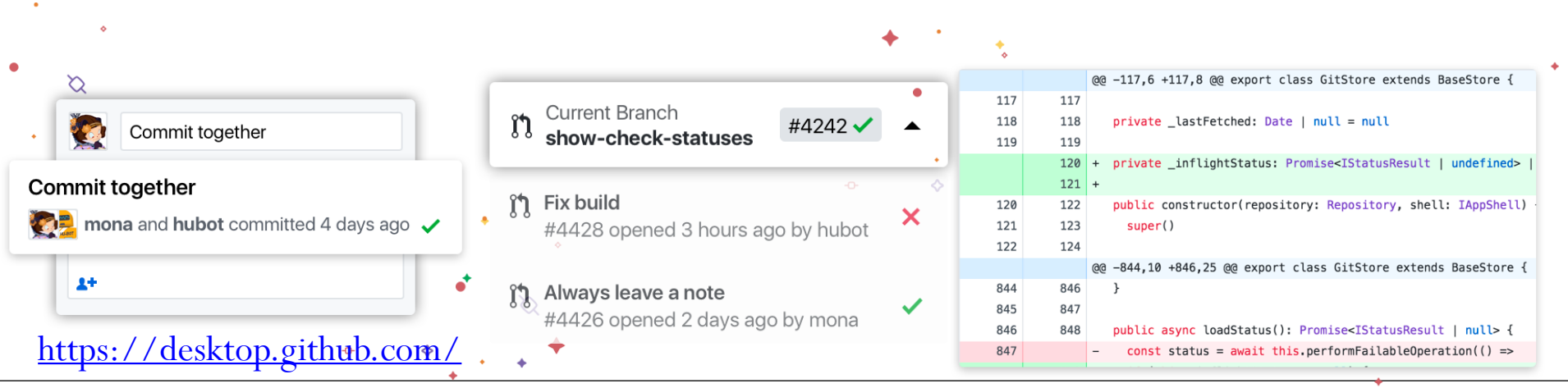
GitHub

- **GitHub** provides web-hosting functionalities for
  - Software development,
  - Distributed Version Control, and
  - Source Code Management (SCM).
- GitHub features
  - bug tracking, continuous integration, change management,
  - feature requests, task management, wikis,
  - *Branching*, *Merging*, *Fetch*, and *Commit* of new *push* files
  - *pull* or *clone* of whole existing repository or its file(s)
- January 2020, GitHub reports
  - over 40 million users and
  - more than 190 million repositories (28 million+ public repositories),
  - making it the largest host of source code in the world.

https://en.wikipedia.org/wiki/GitHub

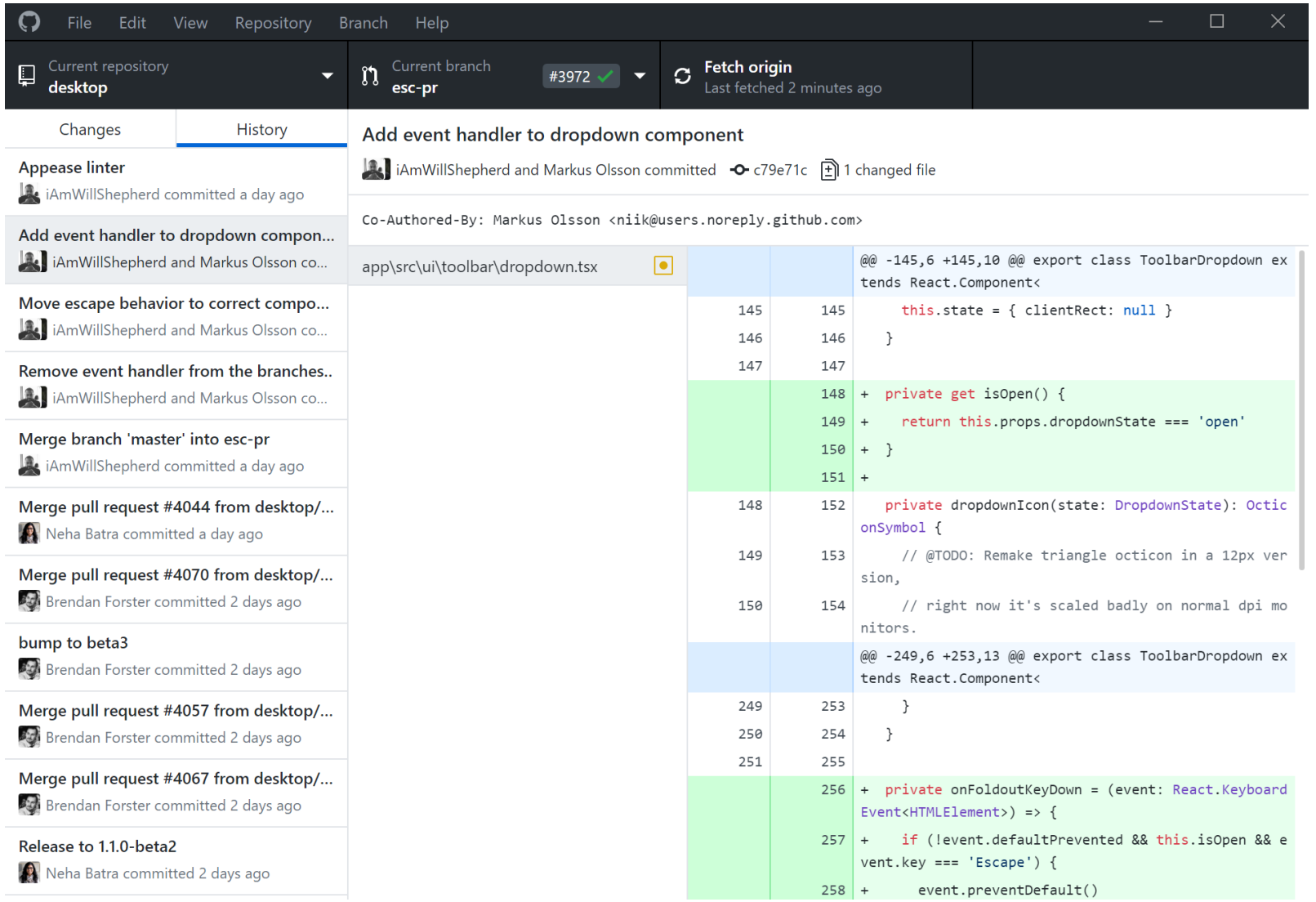
GitHub Desktop (RaaS Client)

- It simplifies development workflow by abstracting Git.
  - Attribute commits with collaborators: co-authors
  - Checkout branches with pull requests and view Commit statuses
  - Syntax highlighted diffs: highlighting when viewing diffs
  - Expanded image diff support: compare changed images
  - Extensive editor & shell integrations: use favorite editor or shell
  - Open-source Community support: contribute and collaboration



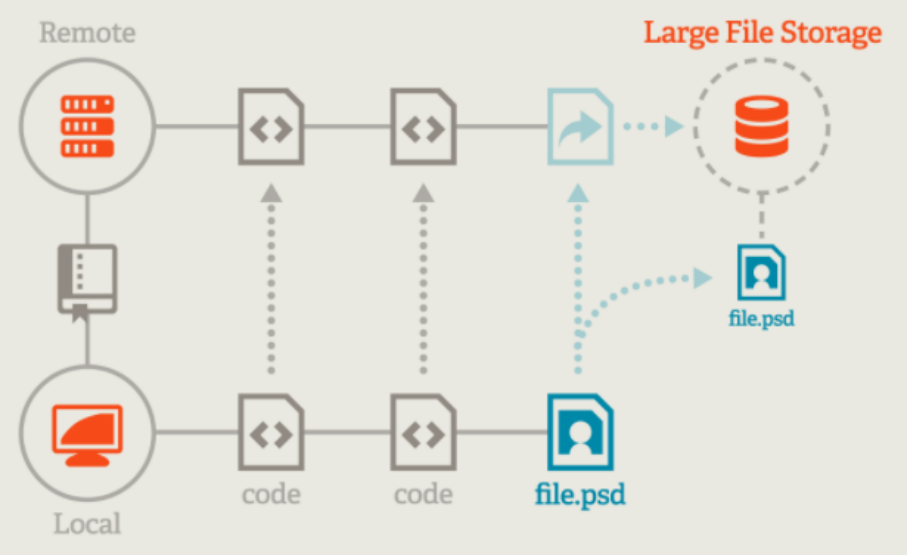
https://desktop.github.com/

GitHub Desktop (RaaS Client)



Git-LFS

- Git Large File Storage (LFS)
  - audio,
  - videos,
  - datasets, and
  - graphics,
- Git-LFS replaces large files with text pointers inside Git
- Storing the file contents on a remote server like
  - GitHub.com or GitHub Enterprise
  - *git lfs* install
  - *git add* file.psd
  - *git commit* **-m** "Add design file"
  - *git push* origin master



References

- Huang, Qi, et al. "An analysis of Facebook photo caching." *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. 2013.
- Boettiger, Carl. "An introduction to Docker for reproducible research." *ACM SIGOPS Operating Systems Review* 49.1 (2015): 71-79.
- Hazelwood, Kim, et al. "Applied machine learning at facebook: A datacenter infrastructure perspective." *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018.
- Naisan Benatar "Version Control" G51FSE *Foundations of Software Engineering*

References

- https://en.wikipedia.org/wiki/Infrastructure_as_a_service
- https://en.wikipedia.org/wiki/Orchestration_(computing)
- https://en.wikipedia.org/wiki/Hypervisor
- https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
- https://en.wikipedia.org/wiki/Amazon_S3
- https://en.wikipedia.org/wiki/Amazon_Elastic_Block_Store
- https://docs.eucalyptus.cloud/eucalyptus/4.4.5/admin-guide/system_concepts.html
- https://github.com/eucalyptus/eucalyptus/wiki
- https://en.wikipedia.org/wiki/Eucalyptus_(software)
- https://aws.amazon.com/ecs/

References

- https://en.wikipedia.org/wiki/Content_delivery_network
- https://en.wikipedia.org/wiki/Akamai_Technologies
- https://en.m.wikipedia.org/wiki/Platform_as_a_service
- https://cloud.google.com/appengine
- https://www.engineyard.com/
- https://en.m.wikipedia.org/wiki/OS-level_virtualization
- https://en.m.wikipedia.org/wiki/Docker_(software)
- https://hub.docker.com/_/hello-world
- https://docs.docker.com/get-started/
- https://containerjournal.com/features/paas-vs-caas-wrong-question-ask/

References

- https://engineering.fb.com/2016/05/09/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/
- https://en.wikipedia.org/wiki/Distributed_version_control
- https://en.wikipedia.org/wiki/Git
- https://en.wikipedia.org/wiki/GitHub
- https://desktop.github.com/
- https://en.wikipedia.org/wiki/General_Data_Protection_Regulation
- https://gdpr-info.eu/
- https://en.wikipedia.org/wiki/Content_delivery_network

תודה רבה
Hebrew

Ευχαριστώ
Greek

Спасибо
Russian

Danke
German

धन्यवादः
Sanskrit

Merci
French

ধন্যবাদ
Bangla

நன்றி
Tamil

شكراً
Arabic

ಧನ್ಯವಾದಗಳು
Kannada

Thank You
English

നന്ദി
Malayalam

多謝
Traditional Chinese

Grazie
Italian

ధన్యవాదాలు
Telugu

આભાર
Gujarati

ਧੰਨਵਾਦ
Punjabi

धन्यवाद
Hindi & Marathi

多谢
Simplified Chinese

Gracias
Spanish

https://sites.google.com/site/animeshchaturvedi07

Obrigado
Portuguese

ありがとうございました
Japanese

ขอบคุณ
Thai

감사합니다
Korean