University of CINCINNATI

Concepts of Virtualization…

(Mastering Cloud Computing: Chapter#3)

Based on materials from Lalit Kumar

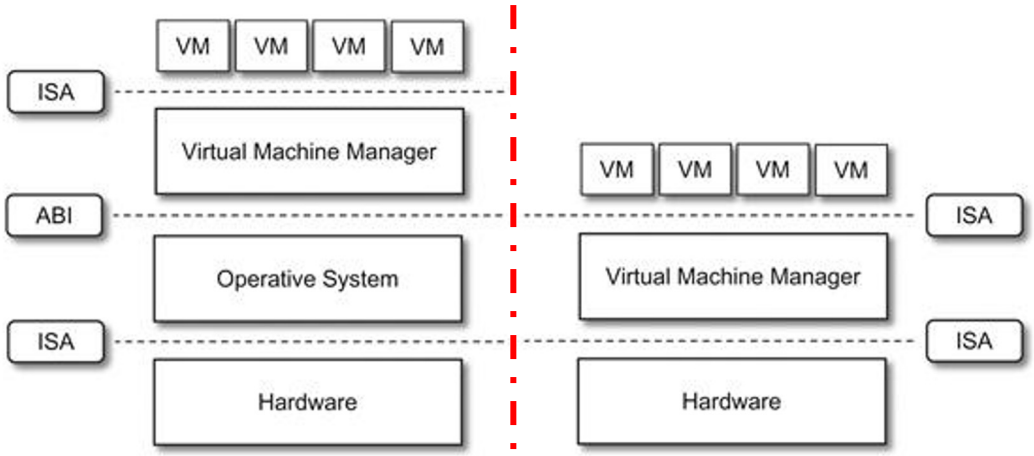
Rashmi Kansakar

Virtual Machine Manager (*VMM*)

University of CINCINNATI

2 types of VMM

Operates on top of OS



Operates on top of hardware

Operates on top of OS:

- Easy to maintain/modify
- Safe to operate
- Less intrusive

Operates on top of Hardware:

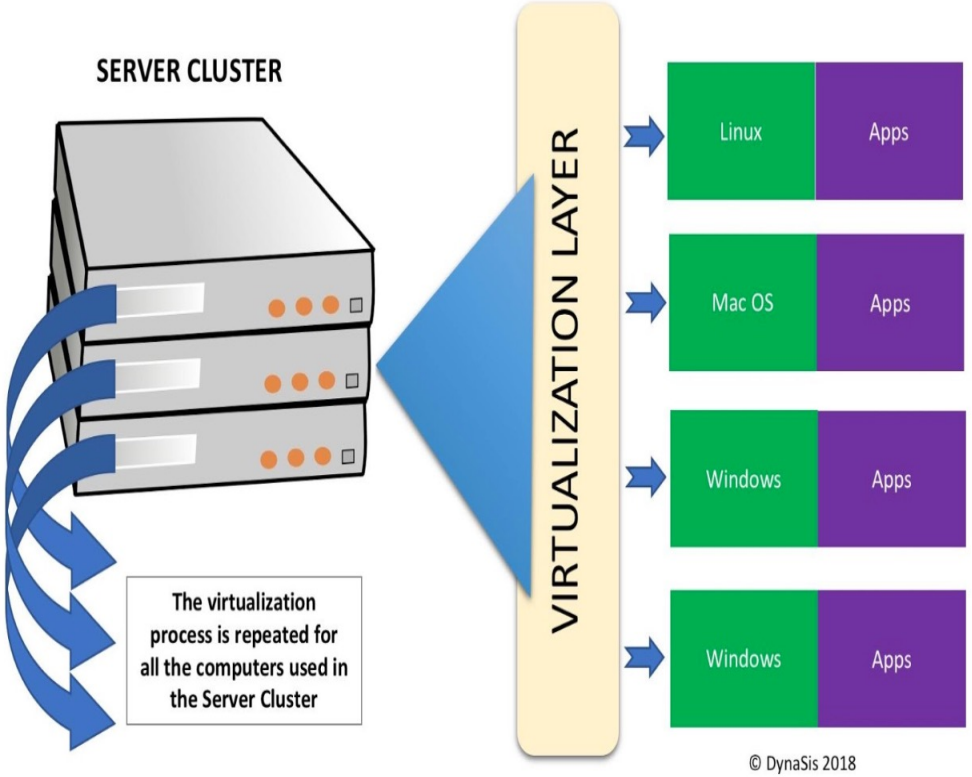
- More involved
- Higher efficiency
- Harder to maintain

Criteria	Type 1 hypervisor	Type 2 hypervisor
AKA	Bare-metal or Native	Hosted
Definition	Runs directly on the system with VMs running on them	Runs on a conventional Operating System
Virtualization	Hardware Virtualization	OS Virtualization
Operation	Guest OS and applications run on the hypervisor	Runs as an application on the host OS
Scalability	Better Scalability	Not so much, because of its reliance on the underlying OS.
Setup/Installation	Simple, as long as you have the necessary hardware support	Lot simpler setup, as you already have an Operating System.
System Independence	Has direct access to hardware along with virtual machines it hosts	Are not allowed to directly access the host hardware and its resources
Speed	Faster	Slower because of the system's dependency
Performance	Higher-performance as there's no middle layer	Comparatively has reduced performance rate as it runs with extra overhead
Security	More Secure	Less Secure, as any problem in the base operating system affects the entire system including the protected Hypervisor
Examples	• VMware ESXi • Microsoft Hyper-V • Citrix XenServer	• VMware Workstation Player • Microsoft Virtual PC • Sun's VirtualBox

Organization Chart

University of CINCINNATI

- **PARTITIONING**
  - Run multiple operating systems on one physical machine
  - Share physical resources between virtual machines
- **PORTABILITY**
  - Entire virtual machine is saved as a file, so…
  - Move, copy, or export as easily as a file
- **SECURITY**
  - Hardware is isolated from the operating system
  - Recovery as easily as restoring a file
- **AGNOSTIC**
  - Migrate a virtual machine between similar, or different, physical servers



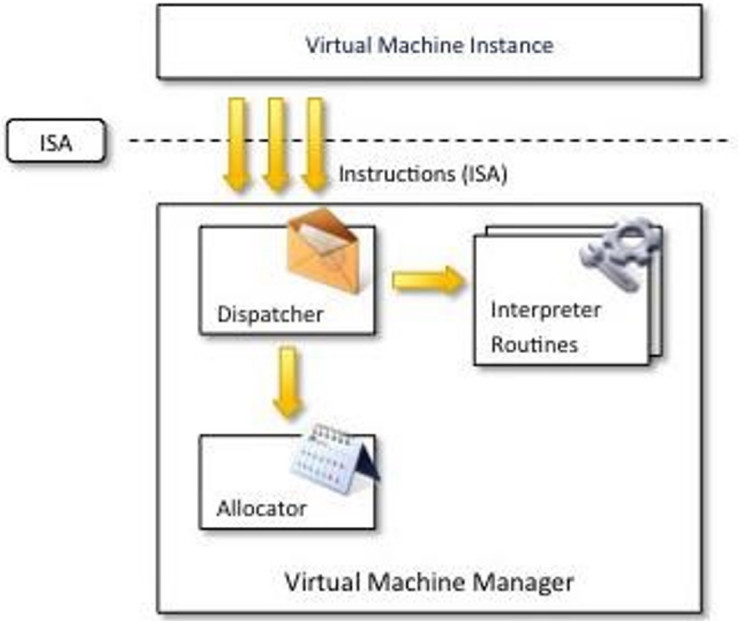
VMM Hierarchy

University of CINCINNATI

3 major components:

- Dispatcher
  - route instructions
- Allocator
  - provides system resources
- Interpreter routines
  - invoked for execution of privileged instructions

Click to add text



VMM Requirements

University of CINCINNATI

- **Equivalence**
  - Guest should behave exactly as it did on physical hardware
- **Resource control**
  - VMM should be in complete control
- **Efficiency**
  - Significant amount of guest instructions untouched by VMM

Goldberg and Popek (1974)

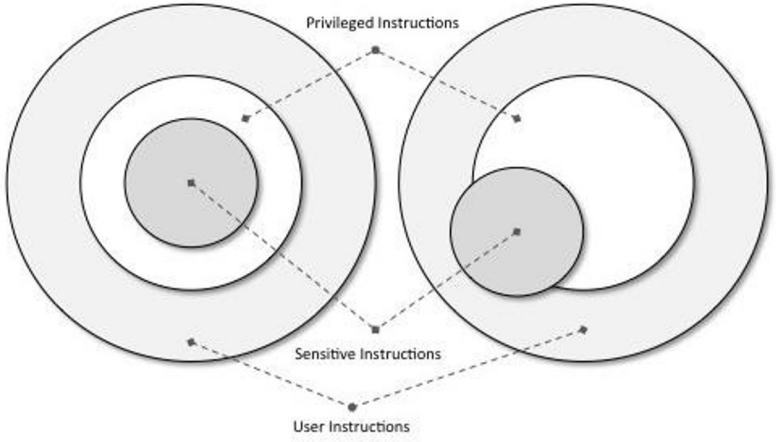
Theorem#1 (3 Theorems for VMM)

University of CINCINNATI

For any conventional third-generation computer (Integrated Circuits), a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

- All the instructions that changes the configuration of the system resources should generate trap in user mode and be executed under the **control of VMM**

- Always guarantees the resource control property when hypervisor is in **Ring 0** and non privileged instructions must be executed without intervention of hypervisor



Theorem#2 (3 Theorems for VMM)

University of CINCINNATI

A conventional third-generation computer is recursively virtualizable if:

a. **It is virtualizable**

b. **A VMM without any timing dependencies can be constructed for it.**

- ➢ Ability to run virtual machine manager on top of a virtual machine manager.
- ➢ Allow nesting hypervisor as long as the capacity of the underlying resources can accommodate.

Nesting makes the virtualization extremely slow.

Theorem#3 (3 Theorems for VMM)

University of CINCINNATI

A hybrid VMM may be constructed for any conventional third-generation machine in which the set of user-sensitive instructions is a subset of the set of privileged instructions.

- In HVM, more instructions are interpreted rather than being executed directly
- All instructions in virtual supervisor mode are interpreted.
- Overall execution in HVM is much slower than VMM
- Behavior/control sensitive instruction are managed by HVM either directly or through system trap.

Hardware Virtualization Techniques

University of CINCINNATI

- Hardware Assisted
- Full virtualization
- Paravirtualization
- Partial virtualization

Example

Hardware Assisted

University of CINCINNATI

- Hardware provides architectural support for virtualization
  - Intel VT & AMD V
- Adds extensions to x86-64
  - Emulate x86 architecture on x86-64
- Originally used in IBM System/370
- Examples: Kernel-based VM (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels

Full Virtualization

University of CINCINNATI

- Ability to run guest with no modifications.
- Guest has no idea it is virtualized.
- All privileged instructions need to be trapped
- Required for Windows OSs when HW not available.
- Works on all hardware, but SLOW

Paravirtualization

University of CINCINNATI

- Non-transparent virtualization
- Guest must be changed
  - changes required for sensitive instructions to VMM API calls
- Simpler VMM
- Much faster:
  - No traps! Code runs on bare hardware
  - OS/driver optimizations for greater speed
- Installing guest drivers
  - makes full virtualization in to paravirtualization
- Examples: Xen, IBM VM OS, VMware, Parallels

Para vs. Full VM

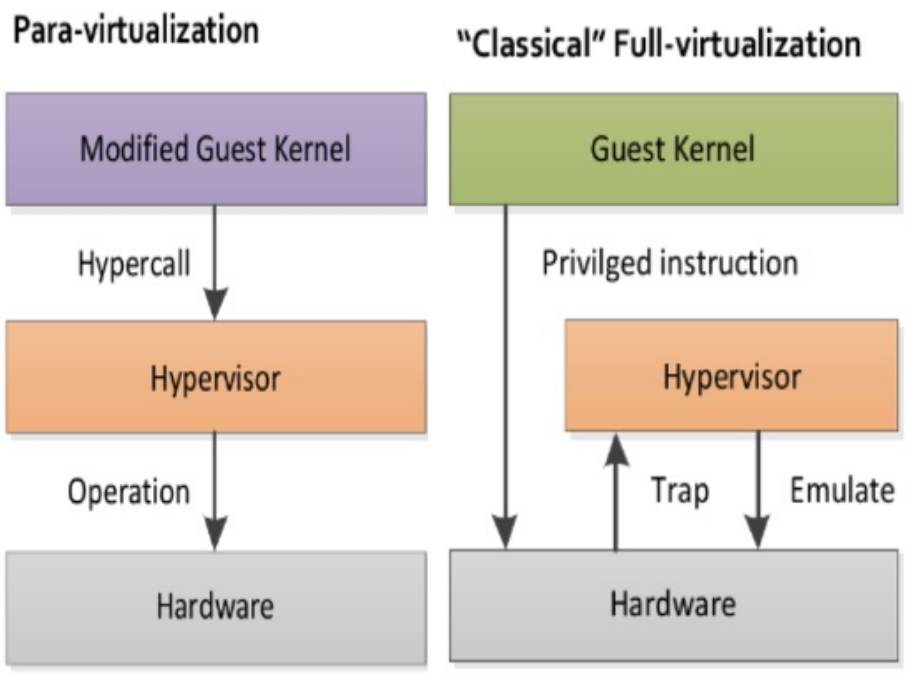
PARAVIRTUALIZATION

An enhancement of virtualization technology in which a guest OS is recompiled prior to installation inside a virtual machine

Allows guest operating systems to communicate with the hypervisor

Guest operating system directly communicate with the hypervisor using drivers

Higher Performance



FULL VIRTUALIZATION

A common and cost effective type of virtualization in which computer service requests are separated from the physical hardware that facilitates them

Allows the guest operating systems to execute independently

Guest operating system issues hardware calls to access hardware

Lower Performance

Full Virtualization	Paravirtualization
In Full virtualization, virtual machines permit the execution of the instructions with the running of unmodified OS in an entirely isolated way.	In paravirtualization, a virtual machine does not implement full isolation of OS but rather provides a different API which is utilized when OS is subjected to alteration.
Full Virtualization is less secure.	While the Paravirtualization is more secure than the Full.
Full Virtualization uses binary translation and a direct approach as a technique for operations.	While Paravirtualization uses hypercalls at compile time for operations.
Full Virtualization is slow than paravirtualization in operation.	Paravirtualization is faster in operation as compared to full virtualization.
Full Virtualization is more portable and compatible.	Virtualization is less portable and compatible.
Examples of full virtualization are Microsoft and Parallels systems.	Examples of paravirtualization are Microsoft Hyper-V, Citrix Xen, etc.
It supports all guest operating systems without modification.	The guest operating system has to be modified and only a few operating systems support it.
The guest operating system will issue hardware calls.	Using the drivers, the guest operating system will directly communicate with the hypervisor.
It is less streamlined compared to para-virtualization.	It is more streamlined.
It provides the best isolation.	It provides less isolation compared to full virtualization.

Partial Virtualization

University of CINCINNATI

- Not all features of hardware/OS are virtualized
- Examples:
  - Address space virtualization - virtual mem
  - RAID - disk
  - Network bonding
  - VPNs
  - Same hardware otherwise

Link

Practical Issues

University of CINCINNATI

- Hardware & Full Virtualization
  - Can overprovision CPU
  - Dedicated Ram
  - Dedicated disk (sometimes, delta drives possible)
- Paravirtualization
  - Can overprovision CPU & RAM!
  - Can set RAM ranges and let OSs self manage!

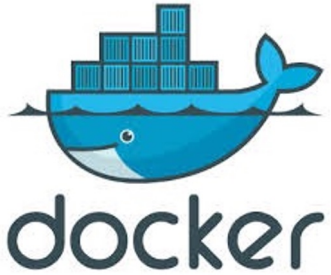
OS-Level Virtualization

University of CINCINNATI

- Separate user-spaces
- No hypervisor needed
- Different filesystem, network configurations, software, access to devices
- Extension of the chroot unix concept
  - Can not modify any resource beyond its scope
- **Little to no overhead**
- Can use any hardware, any software
- But, must all use same OS
- Ex: FreeBSD Jails, Parallels Virtuozzo Containers, OpenVZ, EC2, Docker, others..

Docker

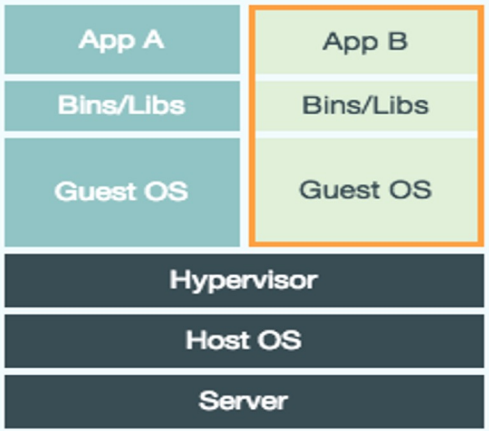
University of CINCINNATI



- Open Sourced Mar 2013 - Jan ‘14 capital
- Extension of LXC (Linux Containers)
- Uses Images to build other Images (Containers)
- Git feel
- Simple config file to generate images
- Lightweight & processes run on bare hardware (fast!)

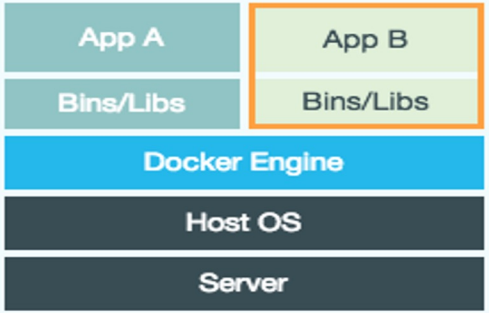
Virtual Machine *Vs.* Docker

University of CINCINNATI



Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.



Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

A Colony of Bungalows

On Premise Deployment:

1) Every app is hosted on a separate sever (bungalow)
2) No optimization of resources at all.
3) No shared services used.
4) High maintenance

An Apartment Complex

Virtualization:

1) Some resources or amenities shared (hyper-visor, host -OS) by apps
2) Some amenities are still app specific (guest OS)
3) Moderate maintenance needed.

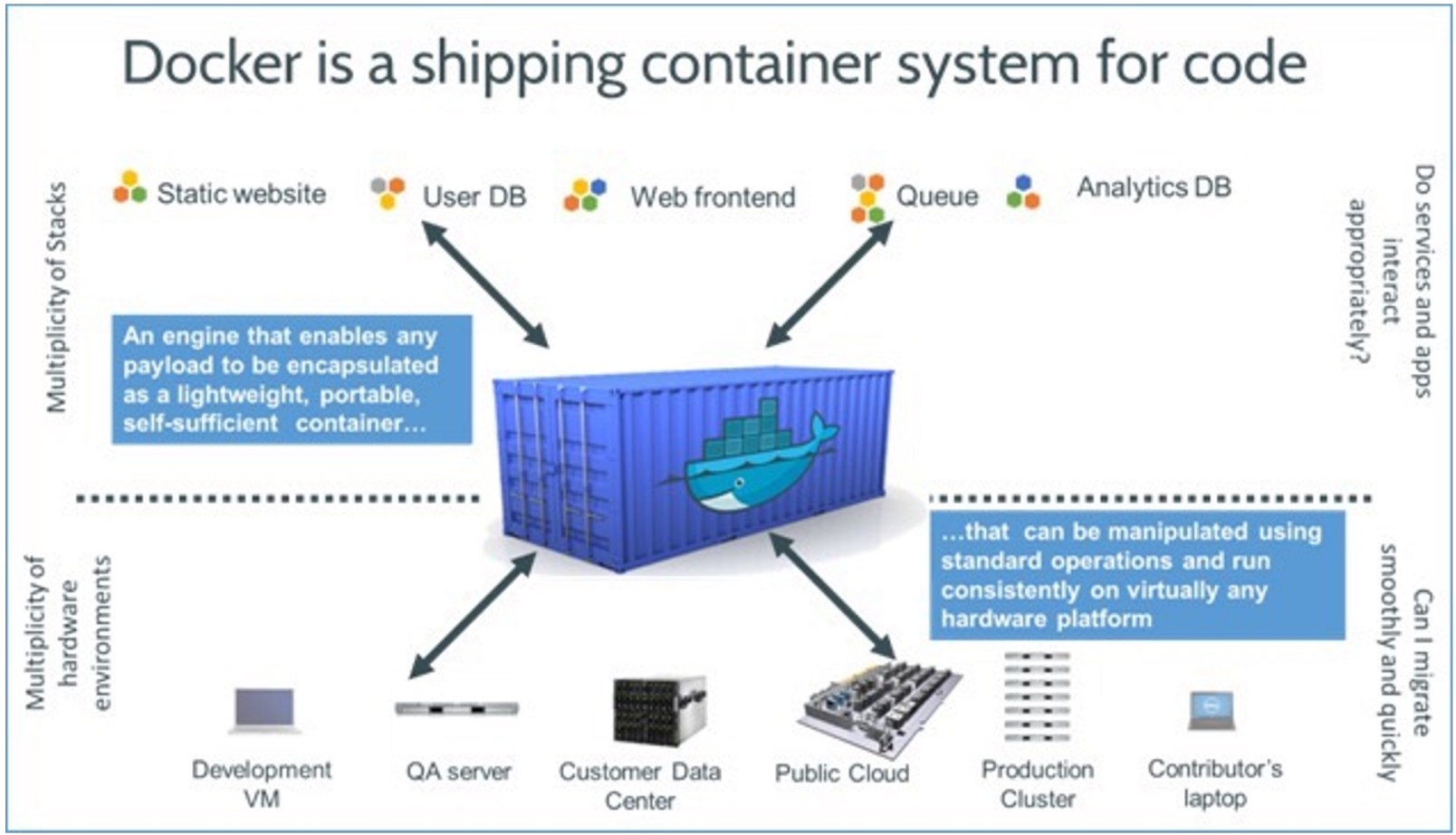
A Hotel

Containerization:

1) Most amenities are shared
2) Guests have only a subset of amenities to use
3) Hotel has better control over managing resources

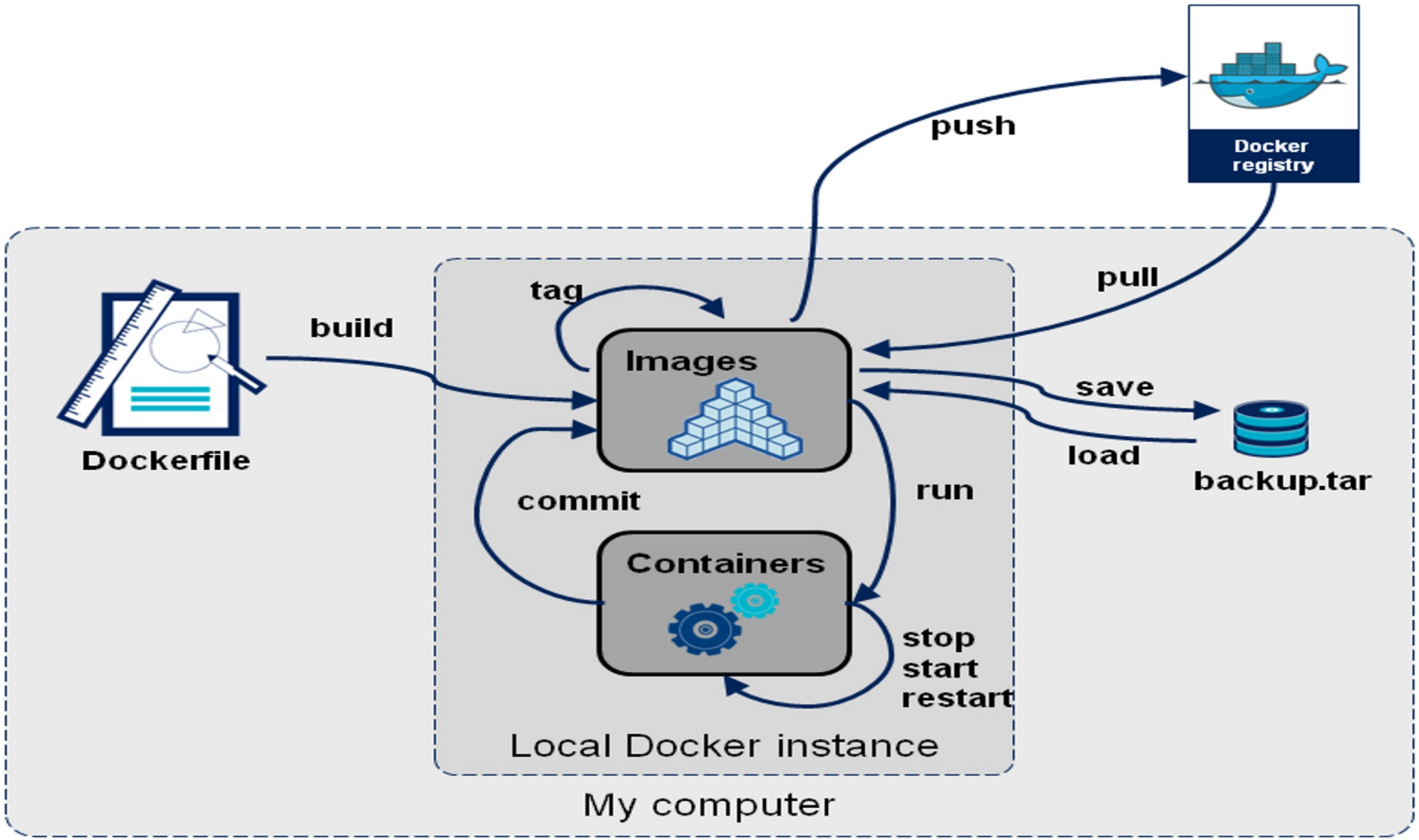
Docker Concept

University of CINCINNATI



Docker Operation

University of CINCINNATI

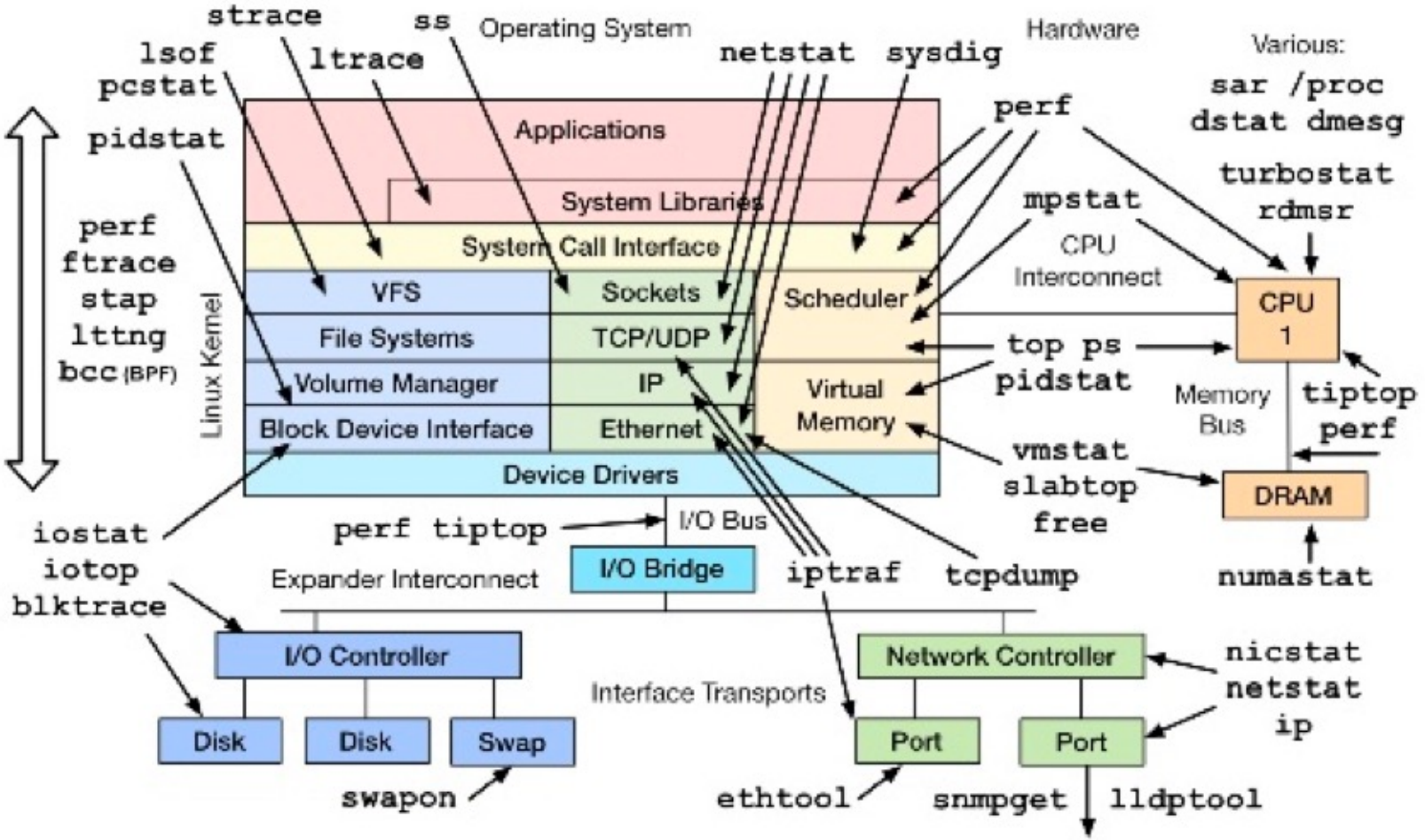


Virtual Machines	Docker
Each VM runs its own OS	All containers share the same Kernel of the host
Boot up time is in minutes	Containers instantiate in seconds
VMs snapshots are used sparingly	Images are built incrementally on top of another like layers. Lots of images/snapshots
Not effective diffs. Not version controlled	Images can be diffed and can be version controlled. Dockerhub is like GITHUB
Cannot run more than couple of VMs on an average laptop	Can run many Docker containers in a laptop.
Only one VM can be started from one set of VMX and VMDK files	Multiple Docker containers can be started from one Docker image

What Kernels Do

- Respond to messages from the hardware
- Start and schedule programs
- Control and organize storage
- Pass messages between programs
- Allocates resource, memory, CPU, network, and so on
- Create containers by Docker configuring the kernel

Linux Perf Tools



Where can we begin?

What is Docker Compose?

University of CINCINNATI

- Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.
- Using Compose is basically a three-step process:

1. Define your app's environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run docker-compose up and Compose starts and runs your entire app.

a docker-compose.yml looks like this

University of CINCINNATI

```
version: '3'
services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```

Backend Service

Specify Volumes/Network

Environmental variables

Frontend Service

Specify Volumes/Network

Environmental variables

docker

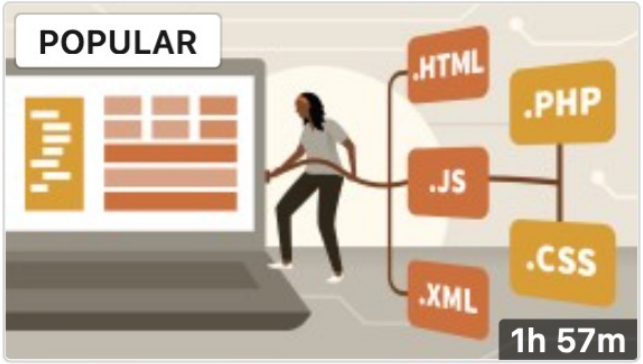
Container

- ✓ A container is a lightweight virtual machine that provides address space, network, file isolation
- ✓ Docker allows building images in layers and deployment of a new version just requires deploying layers that have changed.
- ✓ Containers can be managed either on VMs through autoscaling or on preallocated pool for short duration, quick loading
- ✓ Development workflow is supported through an image repository.

Docker by Doing

University of CINCINNATI

LEARNING OBJECTIVES:



COURSE

Learning Docker

LinkedIn · By: Carlos Nunez · 1 month ago

4.6 ★★★★½ (165) · 4,366 learners · Beginner

https://www.linkedin.com/learning/learning-docker-17236240

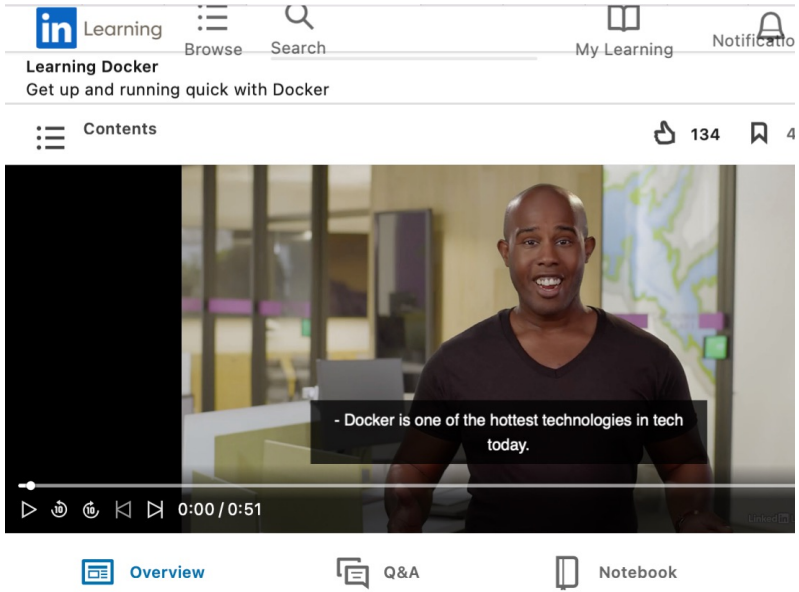
Docker by Doing

University of CINCINNATI

LEARNING OBJECTIVES:

COURSE **Learning Docker**

By: Carlos Nunez



Course details

1h 57m
Beginner
Released: 12/14/2022

Docker is an open-source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system libraries and dependencies required to run that code in any environment. In this course, Carlos Nunez introduces the basics of Docker, including its containers, Dockerfiles (or base images), and capabilities. Watch and learn how to build your own containers.

https://www.linkedin.com/learning/learning-docker-2/what-is-docker?u=2133849

Docker Commands

University of CINCINNATI

Additional Information (Docker): Link#1 & Link#2

- docker search centos
- docker pull centos (or any other image)
- docker images
- docker run -it <image name>
- docker run -v </mymachine>:</vm> <image name> <cmd>
- docker run -d -p 8080:80 <image name> <cmd>
- docker ps (-a)
- docker commit <image id> <repo name>
- docker build <folder with Docker file> (link)
- docker save <container id> > out.tar
- docker import out.tar
- docker save <container id> | gzip > out.tar.gz
- docker rm <container id>

Allow regular users to use Docker on a system: *usermod -a -G docker <username>*

Programming Language-Level Virtualization

- Allows programs to be run on different machines with different OS/architectures.
- Uses virtual machine to ‘run’ code: slow.
- Java (JVM), .Net (CLI), Python, Pascal, Groovy, Ruby
- Popular with enterprise apps
- Can be interpreted or jitted
- Stack-based (Java, CLI) or register-based
- Security a big win: easy to sandbox

Application-Level Virtualization

University of CINCINNATI

- Allows applications to run in an environment it shouldn't
- Extra layer emulates missing pieces:
  - filesystem, libraries, OS
- 2 methods:
  - Interpretation: every ISA instruction is handled - minimal startup, huge overhead
  - Binary translation: Instructions translated to native, then cached and reused.
- Ex: WINE, CrossOver (Mac), VMware ThinApp packages an exe

Storage Virtualization

University of CINCINNATI

- Physical location of data not important
- Accessed through logical path or similar
- Management of write access important
- Ex:
  - SAN - Virtualize SATA/SCSI cable - single computer at a time connected
  - NAS - Expose virtual filesystem over a network (NFS, AFP, SMB)
  - Dropbox.com, Box.com, SkyDrive

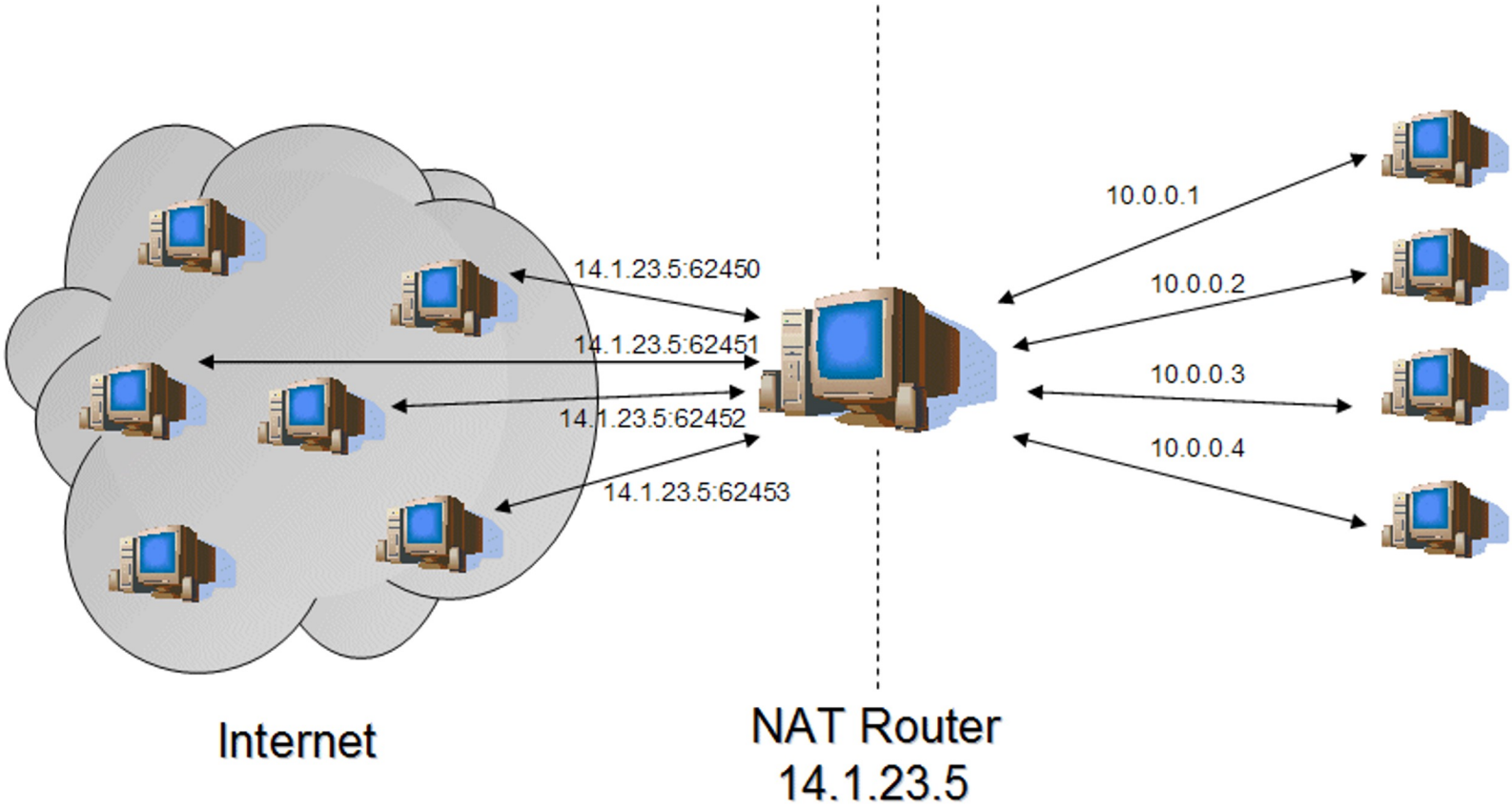
Network Virtualization

University of CINCINNATI

- External net virtualization - VLAN - Separate layer 2 networks (broadcast domain) on same physical wire
- Internal net virtualization - simulated network in computer, as in to connect multiple VMs on in one host.
  - NAT (Network Address Translation)

NAT Diagram

University of CINCINNATI



Desktop Virtualization

University of CINCINNATI

- Allows a desktop environment remotely
- Pros: high availability, persistence, accessibility, ease of management
- Neg: lower refresh, lag
- Old idea… from mainframe era
- Implemented with IaaS or in single OS

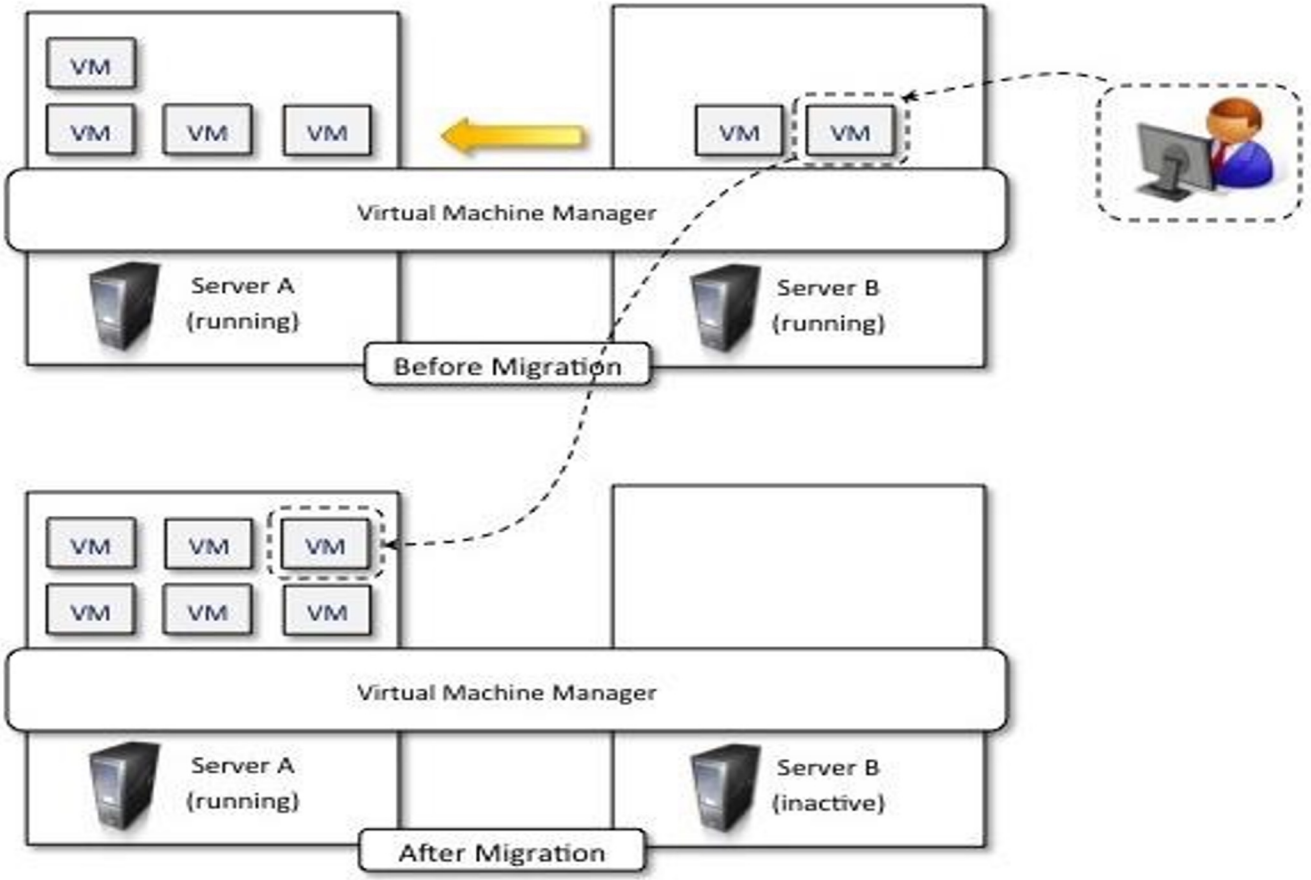
Ex: RDP (Windows Remote Services), VNC, X Server.

Virtualization & Cloud Computing

- Facilitates IaaS
- Facilitates PaaS
- Customization, security, isolation, manageability
- Allows better use of abstraction
- Opens new market-based computing opportunities

Virtual Machine Migration

University of CINCINNATI



Virtualization Pros/Cons

University of CINCINNATI

- Pros:
  - Managed execution
  - Isolation
  - Portability / Migration
  - Efficiency
- Cons:
  - Efficiency - hypervisor or degraded environment
  - Security
    - Phishing
    - Malicious hypervisor: BluPill, SubVirt (rootkits)
    - Replace VM with compromised one (JVM)

Technology Examples - FYI

University of CINCINNATI

Not tested - Details given in the literature

- Xen: paravirtualization
  - XenSource - Citrix (commercial)
  - Xen Cloud Platform - XCP - open source
- VMware: Full virtualization
  - Desktop
    - VMware Workstation
    - VMware Fusion (Mac)
- Server
  - VMware GSX
  - VMware ESXi
  - VMware vSphere - cloud management solution

Technology Examples - FYI

University of CINCINNATI

Not tested - Details given in the literature

- Microsoft Hyper-V
  - Part of Windows Server 2008 R2 onward
  - Windows Server Core (2008) Optimized for Hyper-V (removed extra MS components)
  - System Center Virtual Machine Manager (SCVMM) - manage VM's on multiple servers - can interoperate with VMware

Virtualization Summary

University of CINCINNATI

- **Ability to provide illusion of an environment**
  - Runtime:
    - Hardware: IaaS - VMware, VirtualBox
    - OS Level: CHROOT, Docker
    - Programing language: JVM, Python
    - Application: WINE
  - Storage: SAN, NAS, Dropbox.com
  - Network: VLAN, Virtual Networking, NAT, VPN
  - Remote Desktop: RDP, X Server
  - Pros/cons

Cloud Computing	Grid Computing
Cloud Computing follows client-server computing architecture.	Grid computing follows a distributed computing architecture.
Scalability is high.	Scalability is normal.
Cloud Computing is more flexible than grid computing.	Grid Computing is less flexible than cloud computing.
Cloud operates as a centralized management system.	Grid operates as a decentralized management system.
In cloud computing, cloud servers are owned by infrastructure providers.	In Grid computing, grids are owned and managed by the organization.
Cloud computing uses services like Iaas, PaaS, and SaaS.	Grid computing uses systems like distributed computing, distributed information, and distributed pervasive.
Cloud Computing is Service-oriented.	Grid Computing is Application-oriented.
It is accessible through standard web protocols.	It is accessible through grid middleware.

PARAMETER	FULL VIRTUALIZATION	PARA VIRTUALIZATION	HARDWARE ASSISTED VIRTUALIZATION
Generation	1st	2nd	3rd
Performance	Good	Better in certain cases	Fair
Used By	VMware, Microsoft, KVM	VMware, Xen	VMware, Xen, Microsoft, Parallels
Guest OS modification	Unmodified	Codified to issue hypercalls	Unmodified
Guest OS hypervisor independent?	Yes	XenLinux runs only on Hypervisor	Yes
Technique	Direct execution	Hypercalls	Exit to root mode on privileged instruction
Compatibility	Excellent	Poor	Excellent

Set up Google Cloud Platform Account

University of CINCINNATI

Check out the 'Always free' and 300 USD bonus to learn GCP within the first 12 months, details can be found here:

https://cloud.google.com/free

https://cloud.google.com/free

University of CINCINNATI

Solve real business challenges on Google Cloud

Get started for free | Contact sales

Run workloads for free

20+ free products for all customers

All customers get free hands-on experience with popular products, including Compute Engine and Cloud Storage, up to monthly limits.

$300 in free credits for new customers

New customers get $300 in free credits to fully explore and conduct an assessment of Google Cloud. You won't be charged until you upgrade.

Additional free credits for businesses

New customers who verify their business email will get additional free credits to use while exploring and evaluating Google Cloud.

https://cloud.google.com/free/docs/free-cloud-features

University of CINCINNATI

Google Cloud Free Program >

Was this helpful?

Free Google Cloud features and trial offer

Send feedback

Whether you're new to Google Cloud and are looking to learn the basics or you are an established customer exploring new product innovations, the Google Cloud Free Program has you covered.

The Google Cloud Free Program comprises the following:

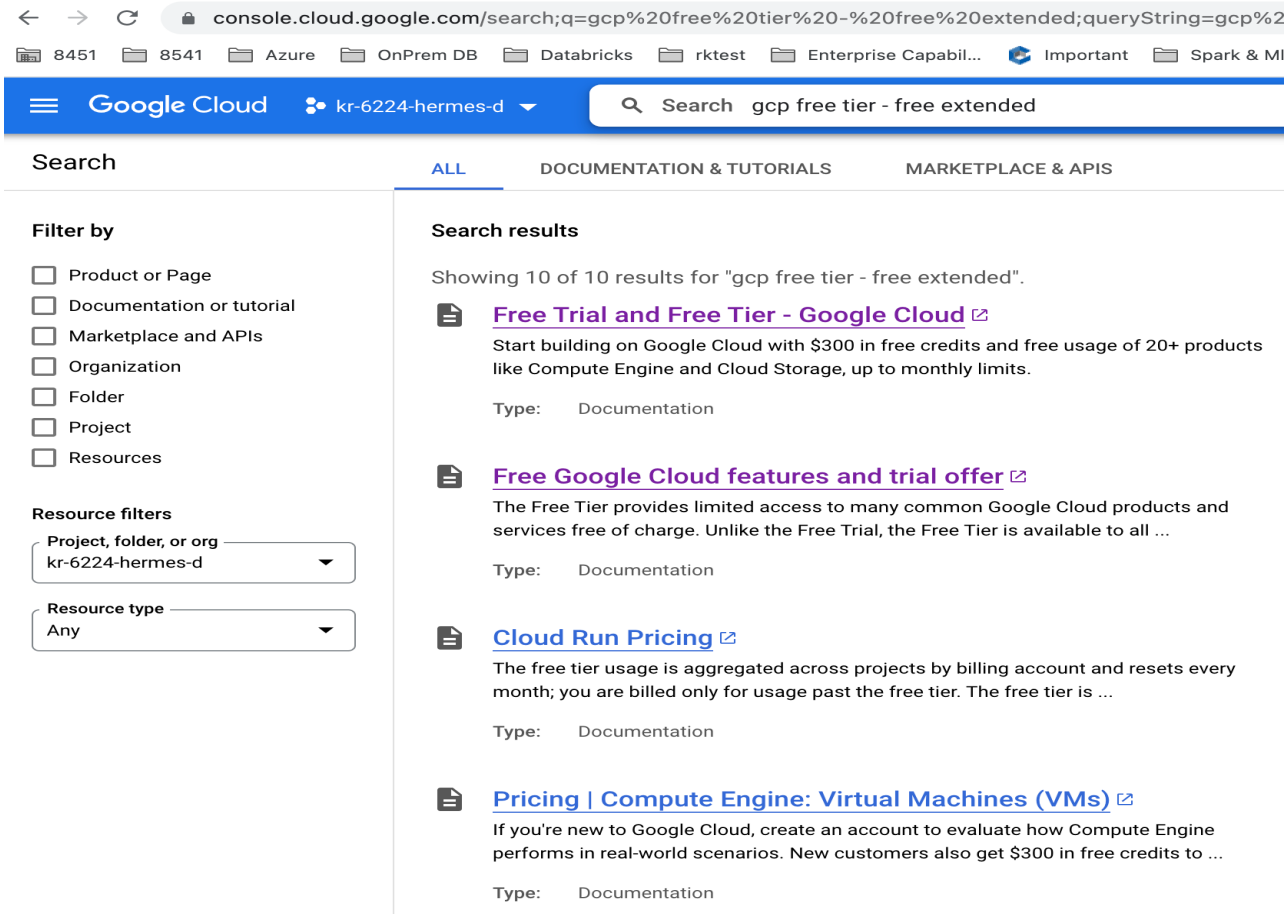
- **90-day, $300 Free Trial**: New Google Cloud and Google Maps Platform users can take advantage of a 90-day trial period that includes $300 in free Cloud Billing credits to explore and evaluate Google Cloud and Google Maps Platform products and services. You can use these credits toward one or a combination of products.
- **Free Tier**: All Google Cloud customers can use select Google Cloud products—like Compute Engine, Cloud Storage, and BigQuery—free of charge, within specified monthly usage limits. When you stay within the Free Tier limits, these resources are not charged against your Free Trial credits or to your Cloud Billing account's payment method after your trial ends.
- **Google Maps Platform monthly credit**: Google Maps Platform features a recurring $200 monthly credit (see Pricing for Maps, Routes, and Places). The monthly credit applies towards each Maps-related Cloud Billing account you create. Learn more about Google Maps Platform Billing Account Credits.

This article describes these components of the Google Cloud Free Program.

90-day, $300 Free Trial offer

Search: Free Google Cloud features and trial offer

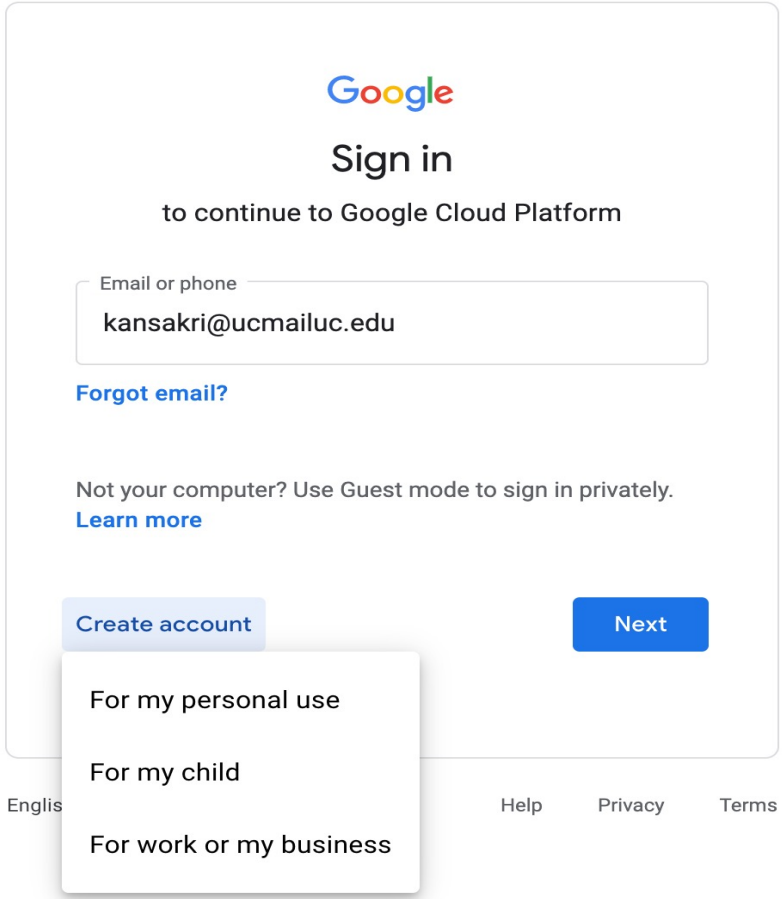
University of CINCINNATI



Search: Free Google Cloud features and trial offer

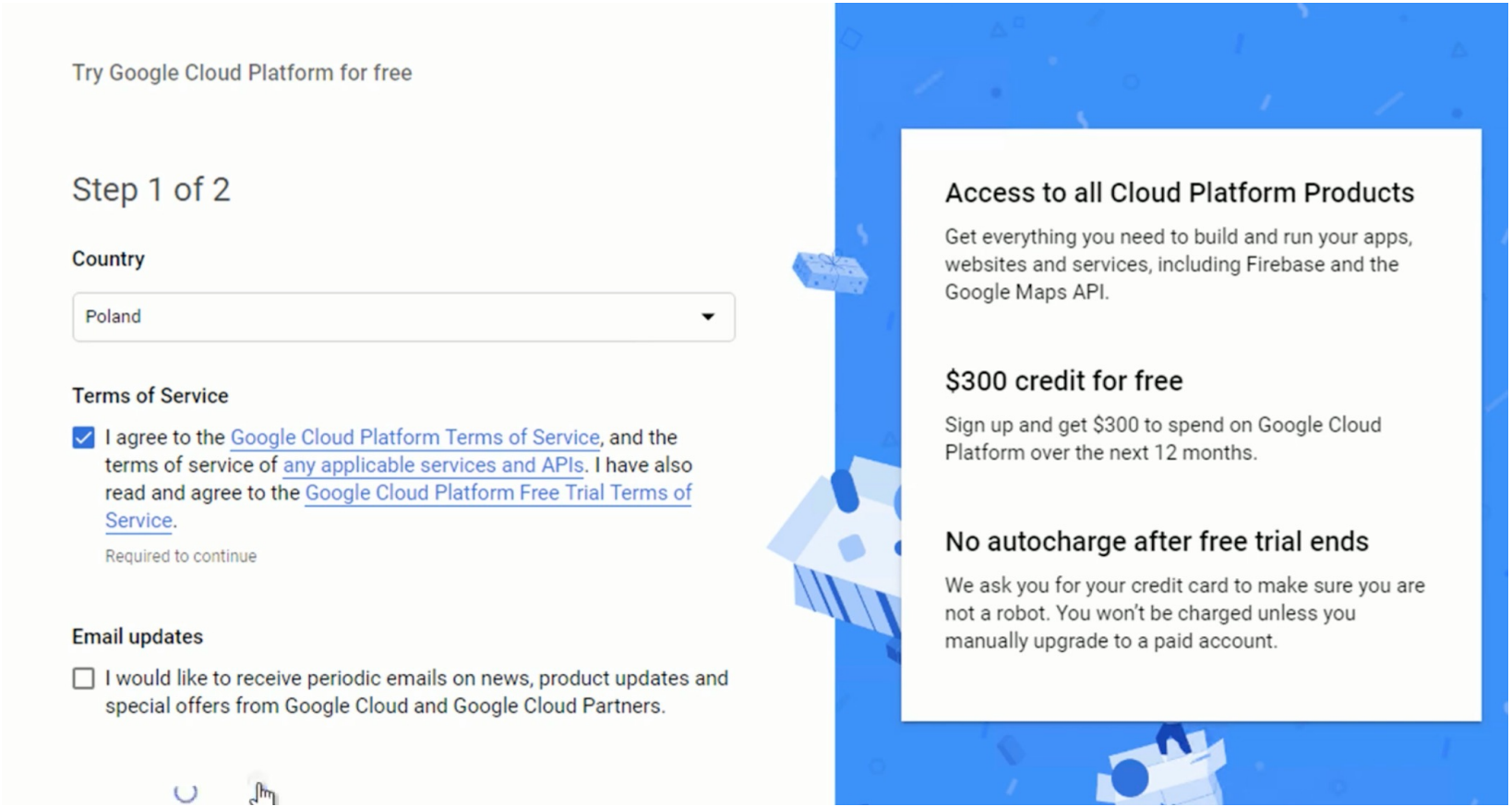
University of CINCINNATI

Get started for free



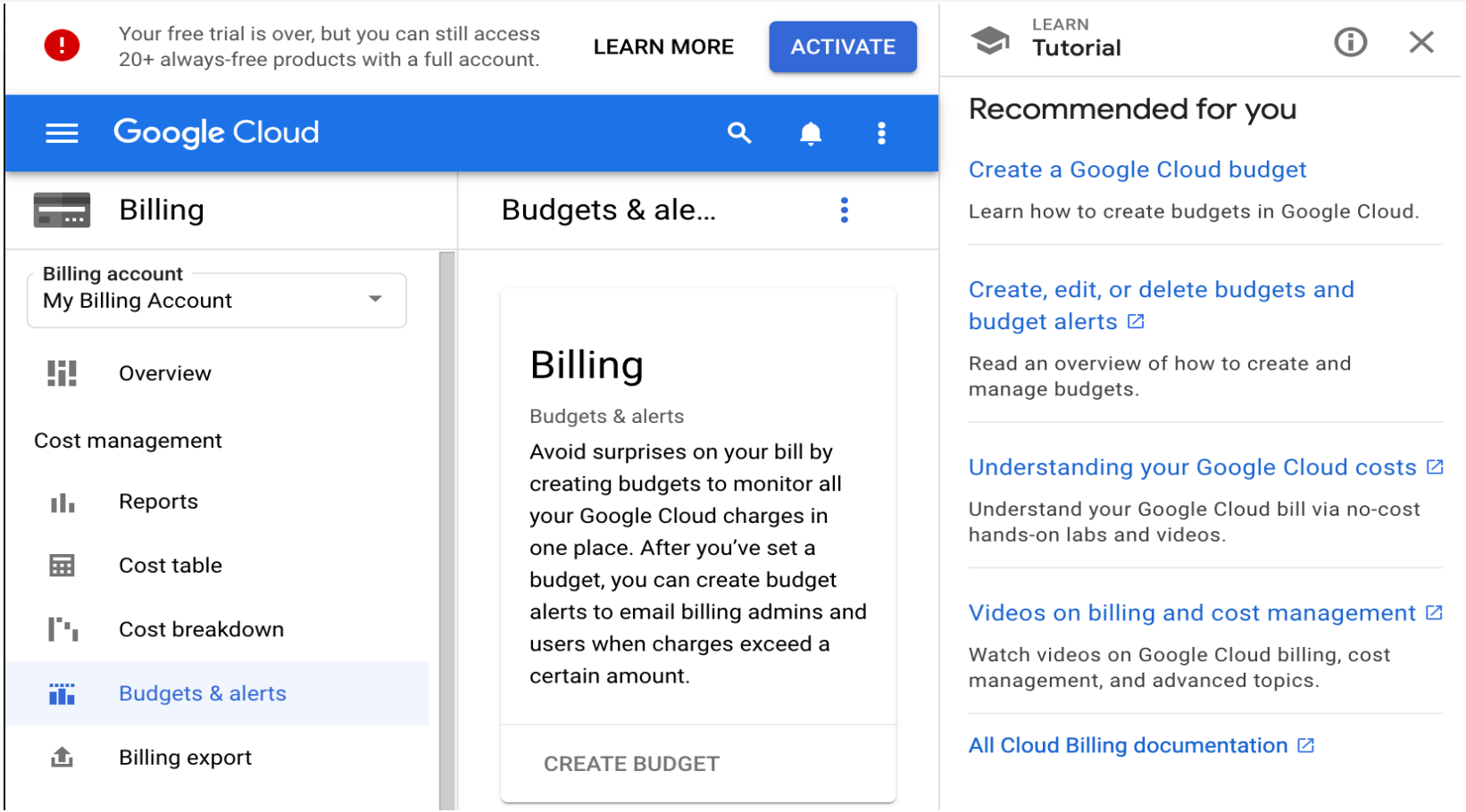
Search: Free Google Cloud features and trial offer

University of CINCINNATI



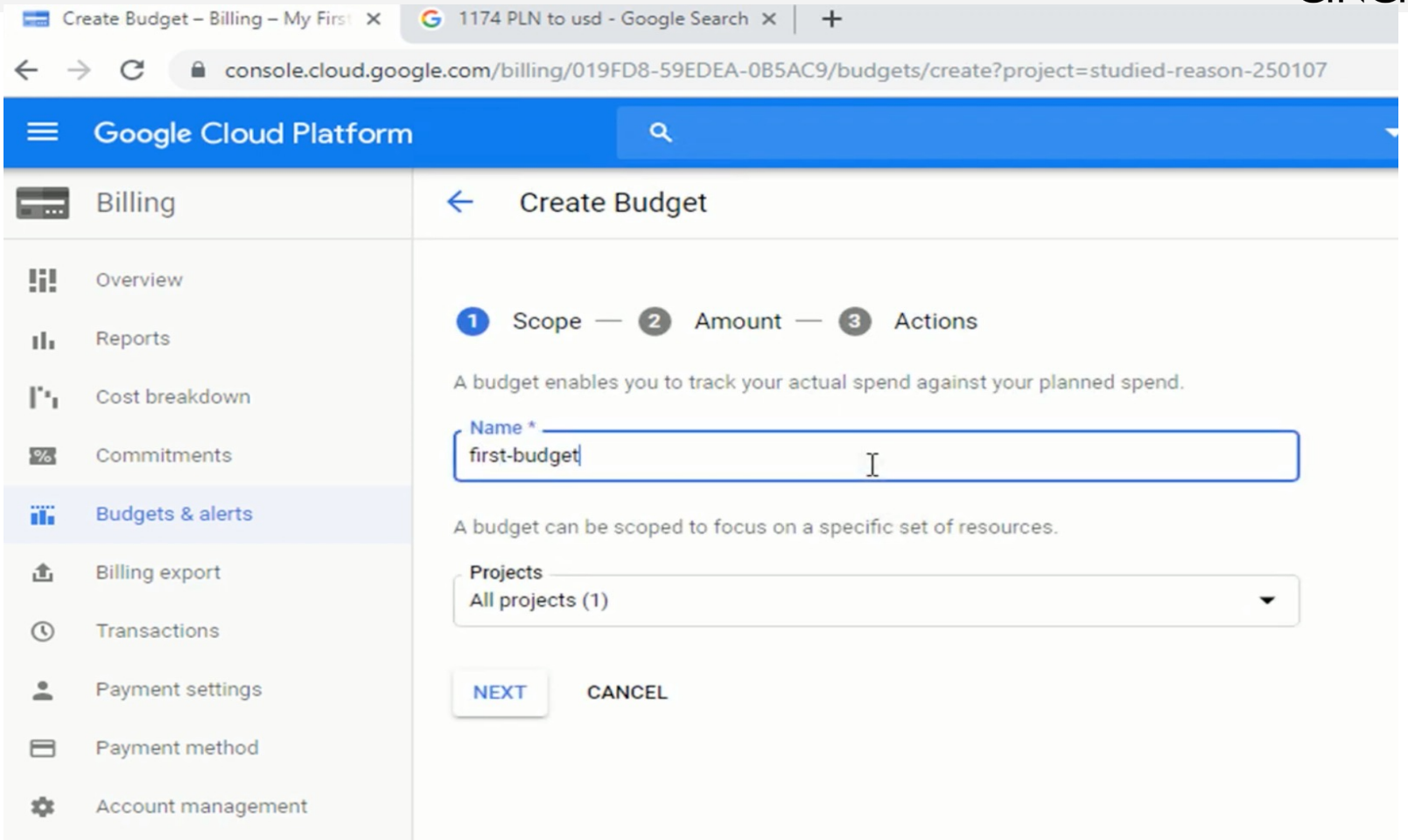
Billing details

University of CINCINNATI

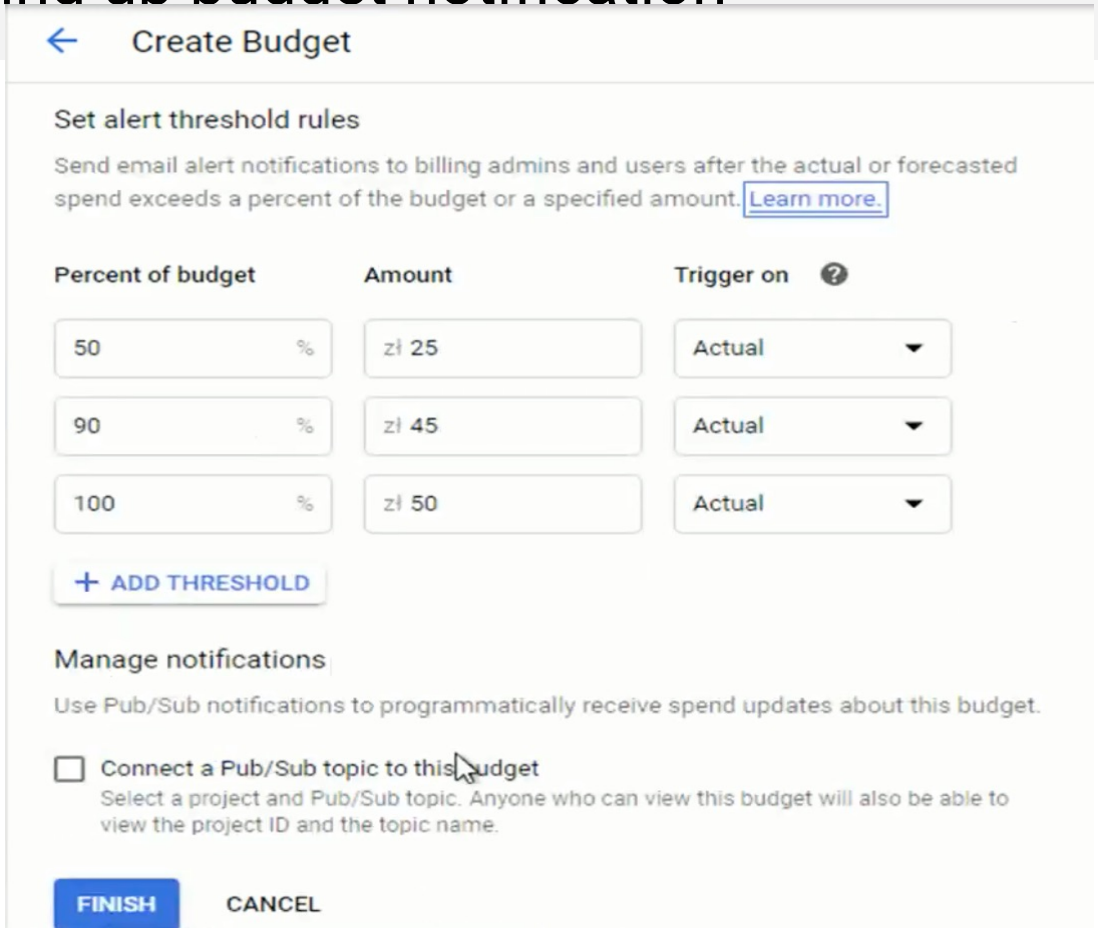


Creating a Budget

University of CINCINNATI



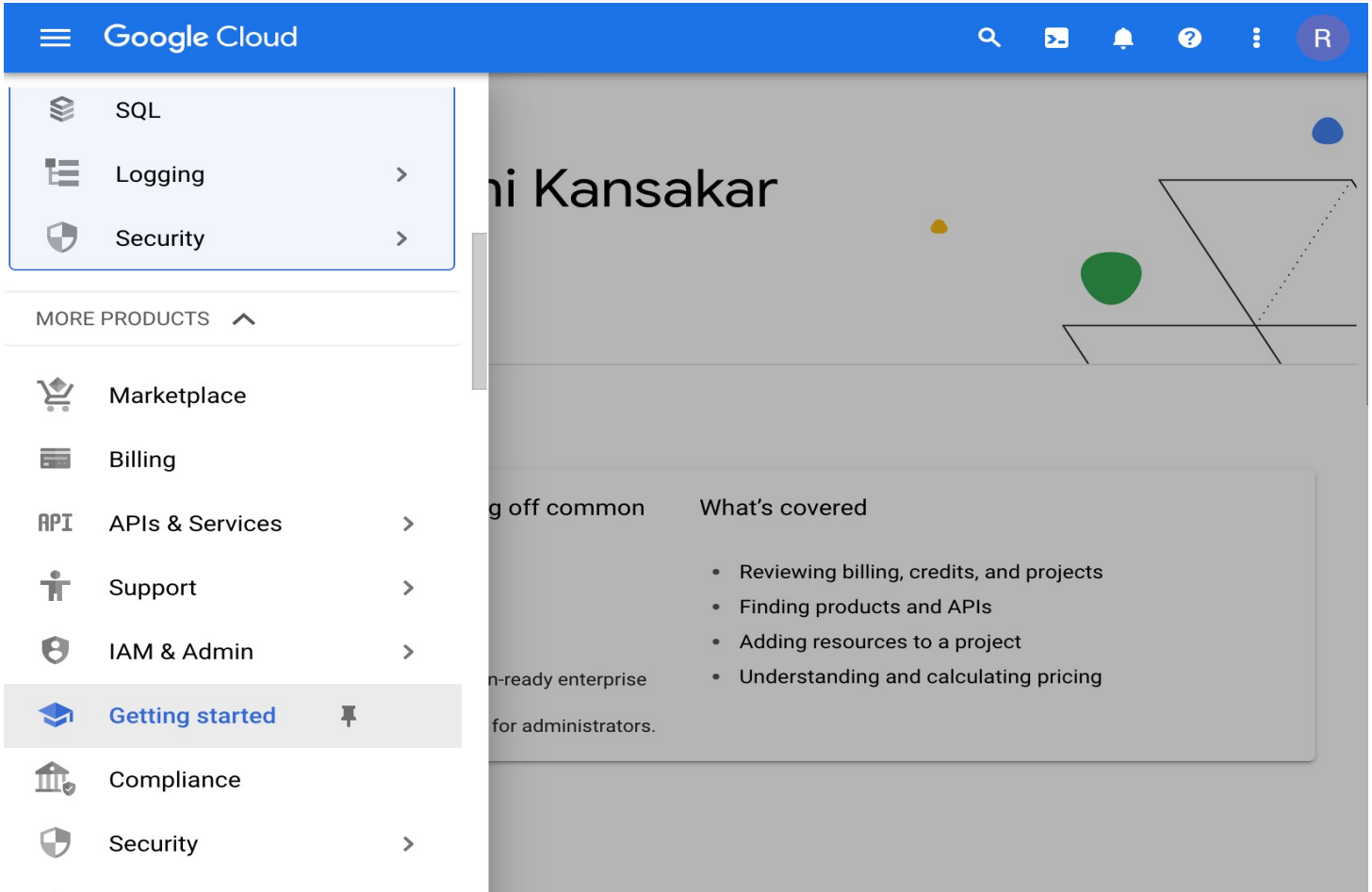
Setting up budget notification



Exceeding budget does not start charging the fee. It is just a notification.

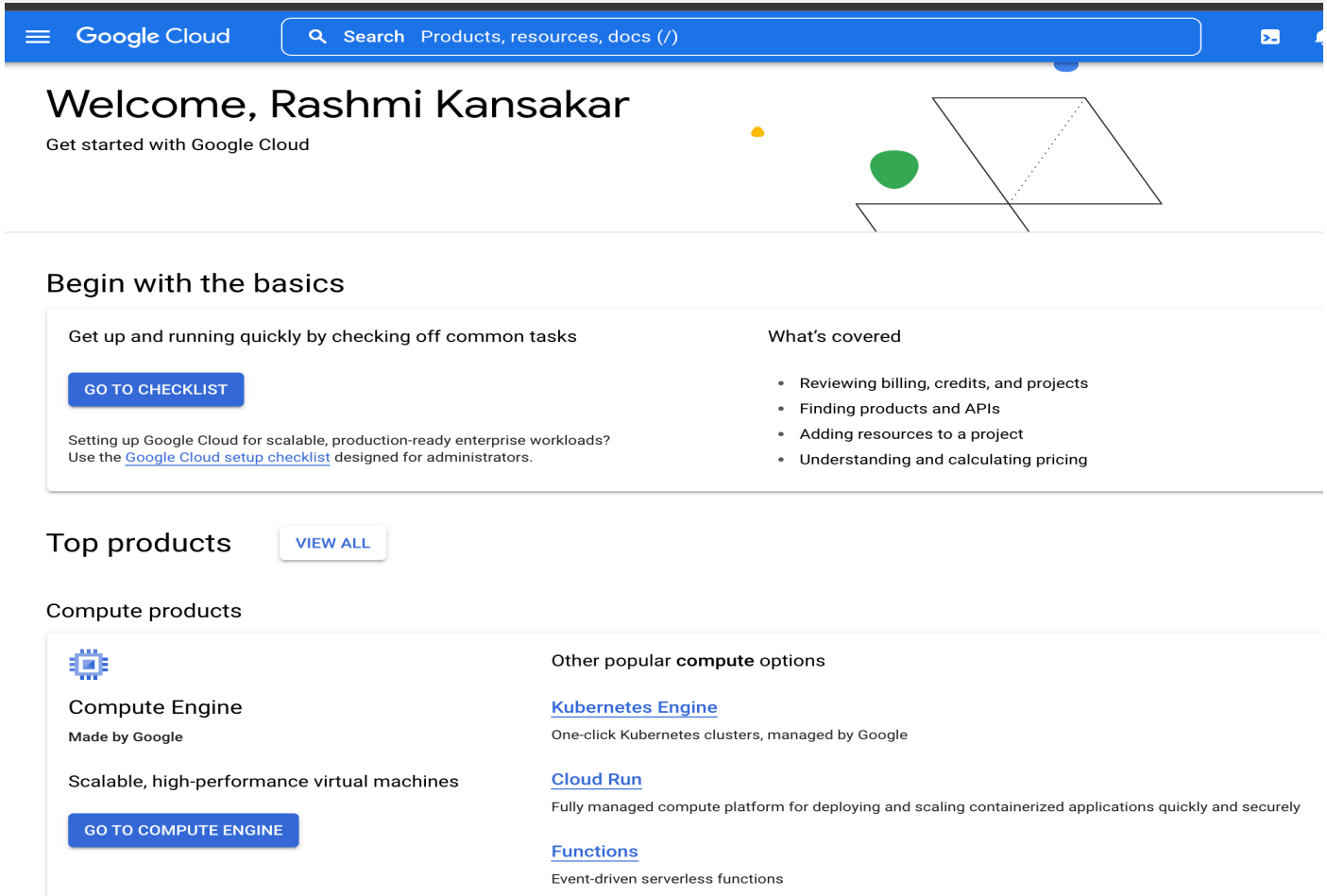
Getting Started

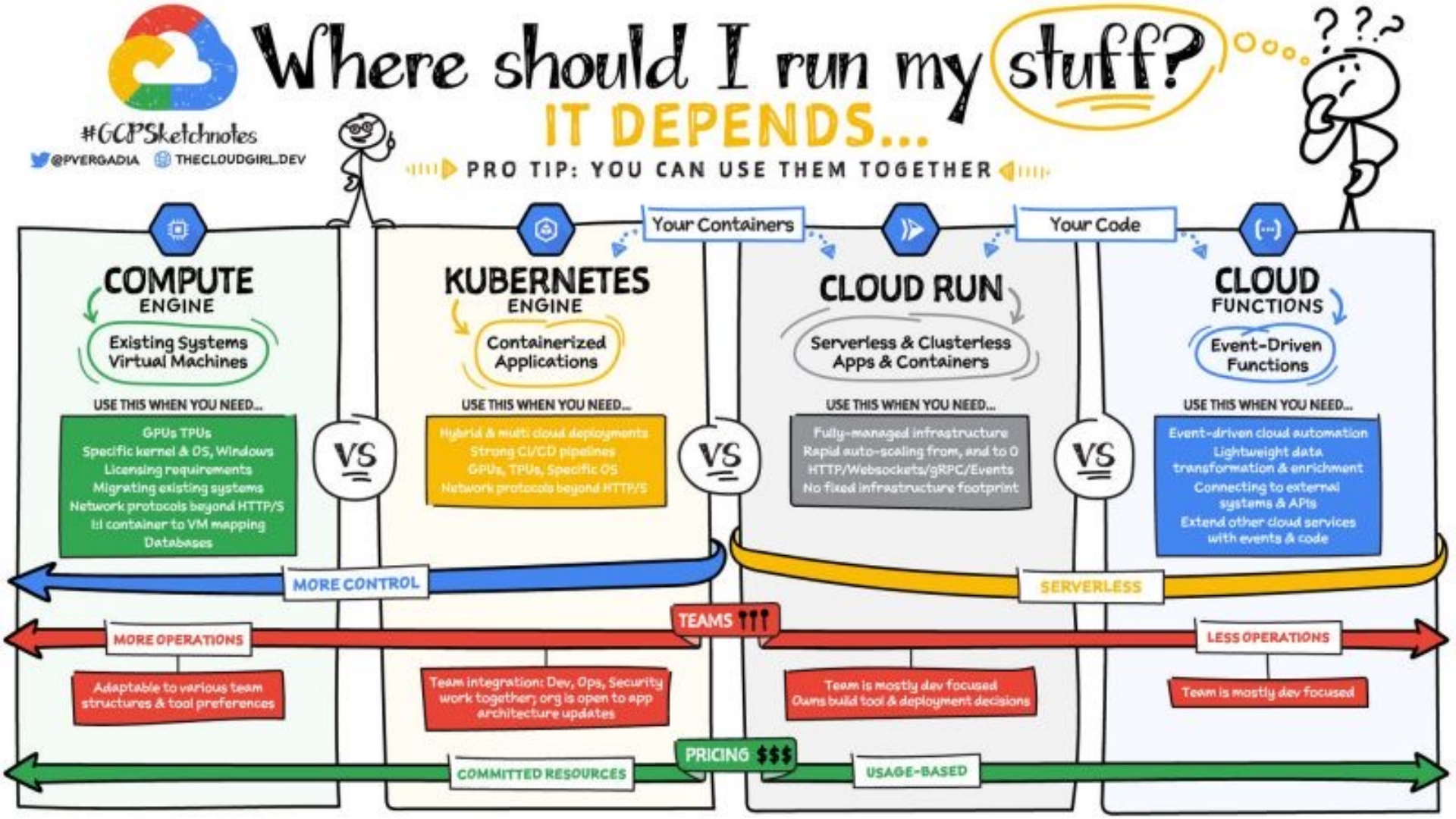
University of CINCINNATI

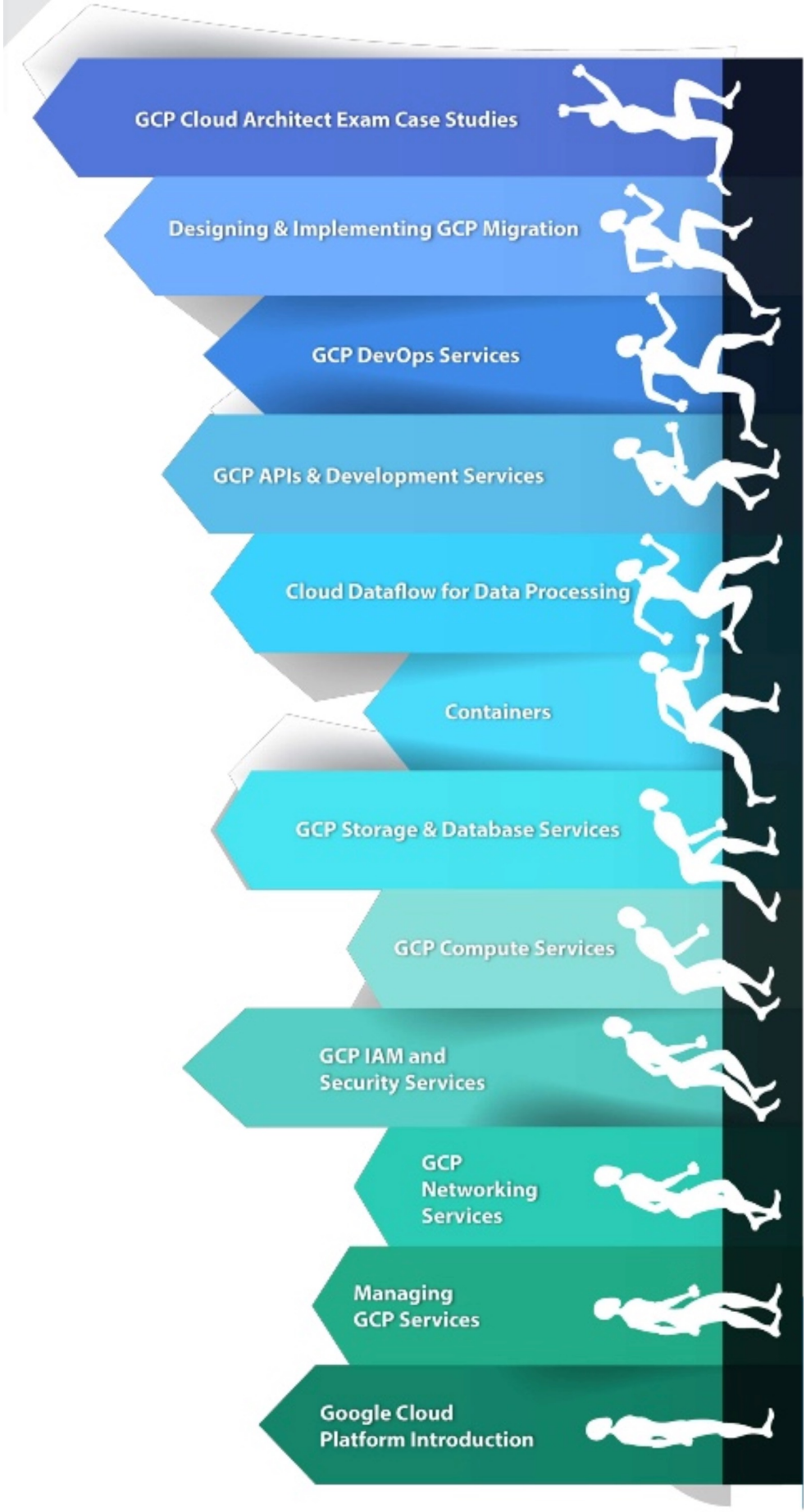


Begin with the basics

University of CINCINNATI




#GCPSketchnotes
@PVERGADIA
THECLOUDGIRL.DEV
Where should I run my stuff?
IT DEPENDS...
PRO TIP: YOU CAN USE THEM TOGETHER
Your Containers
Your Code
COMPUTE ENGINE
Existing Systems
Virtual Machines
USE THIS WHEN YOU NEED...
GPUs TPUs
Specific kernel & OS, Windows
Licensing requirements
Migrating existing systems
Network protocols beyond HTTP/S
1:1 container to VM mapping
Databases
VS
KUBERNETES ENGINE
Containerized Applications
USE THIS WHEN YOU NEED...
Hybrid & multi cloud deployments
Strong CI/CD pipelines
GPUs, TPUs, Specific OS
Network protocols beyond HTTP/S
VS
CLOUD RUN
Serverless & Clusterless Apps & Containers
USE THIS WHEN YOU NEED...
Fully-managed infrastructure
Rapid auto-scaling from, and to 0
HTTP/Websockets/gRPC/Events
No fixed infrastructure footprint
VS
CLOUD FUNCTIONS
Event-Driven Functions
USE THIS WHEN YOU NEED...
Event-driven cloud automation
Lightweight data transformation & enrichment
Connecting to external systems & APIs
Extend other cloud services with events & code
MORE CONTROL
SERVERLESS
TEAMS
MORE OPERATIONS
LESS OPERATIONS
Adaptable to various team structures & tool preferences
Team integration: Dev, Ops, Security work together; org is open to app architecture updates
Team is mostly dev focused
Owns build tool & deployment decisions
Team is mostly dev focused
PRICING $$$
COMMITTED RESOURCES
USAGE-BASED


GCP Cloud Architect Exam Case Studies
Designing & Implementing GCP Migration
GCP DevOps Services
GCP APIs & Development Services
Cloud Dataflow for Data Processing
Containers
GCP Storage & Database Services
GCP Compute Services
GCP IAM and Security Services
GCP Networking Services
Managing GCP Services
Google Cloud Platform Introduction

QWIKLABS Learn. Engage. Solve

University of CINCINNATI

- https://www.cloudskillsboost.google
- https://go.qwiklabs.com/
- https://github.com/topics/qwiklabs
- Youtube: Introduction to GCP for Students