Middleware Architectures 2

Lecture 5: Cloud Architectures

doc. Ing. Tomáš Vitvar, Ph.D.
tomas@vitvar.com • @TomasVitvar • https://vitvar.com

Czech Technical University in Prague
Faculty of Information Technologies • Software and Web Engineering • https://vitvar.com/lectures

Modified: Sun Mar 10 2024, 20:40:09
Humla v1.0

Overview

- Introduction
- Cloud Architecture
- Infrastructure as a Service

Terminology

- Cloud computing
- *aaS
- DevOps
- Cloud Native, Microservices
- Serverless

What is a Cloud?

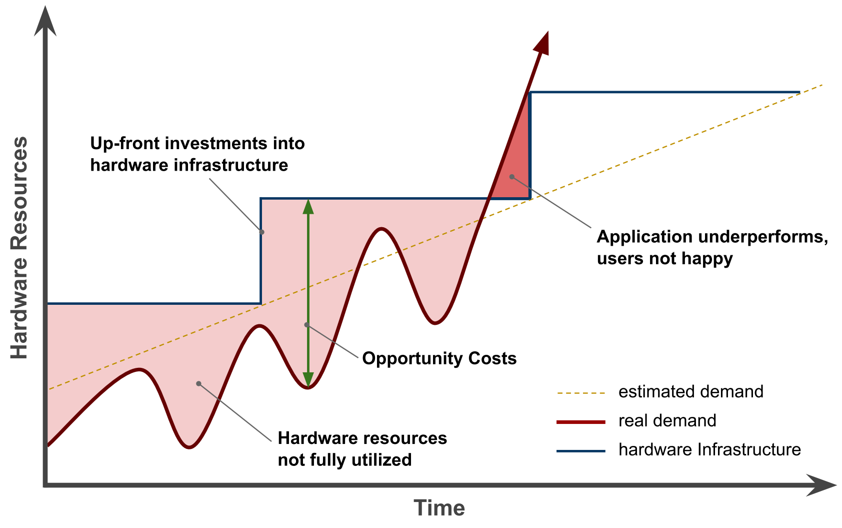
- A different way of thinking
  – *Got your grand mum's savings under your pillow?*
    → probably not, you better have them in your bank
  – *Data is your major asset*
  – *you better have them in a "bank" too*
  – *Someone can abuse your data?*
  – *banks bankrupt too, sometimes – it is a risk you take*
  – *there is a market and a competition*
- Outsourcing of application infrastructure
  – *Reliability and availability*
  – *Low costs – pay-per-use*
  – *Elasticity – can dynamically grow with your apps*
  – *CAPEX vs. OPEX*

What is a Cloud?

- Any app you access over the web?
- A datacenter?
  – *Offers virtualization*
  – *Any company having a datacenter wants to move to*
- Cloud provider should also offer services, such as:
  – *scalability, storage*
  – *Possible to configure programmatically*
    → *integration to enterprise administration processes*
    → *usually REST interface*

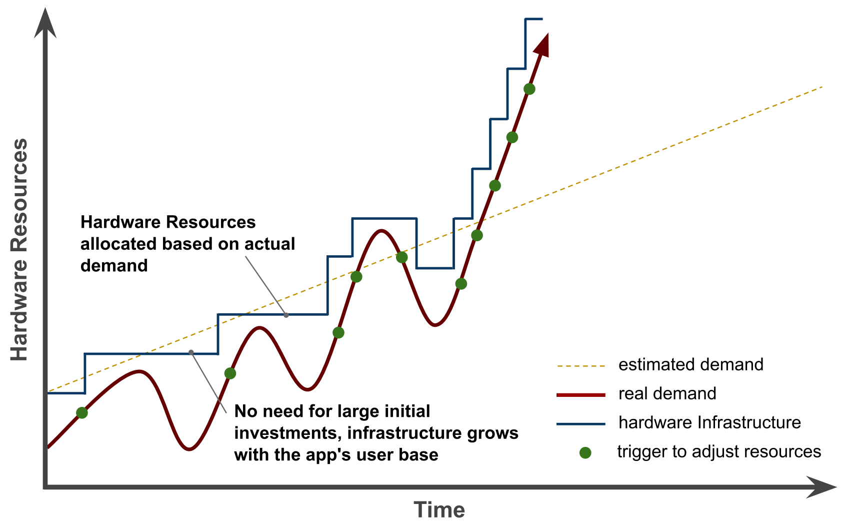
Traditional Solution to Infrastructure

- Traditional hardware model
  – *Up-front hardware investments*
  – *Hardware not optimally utilized*



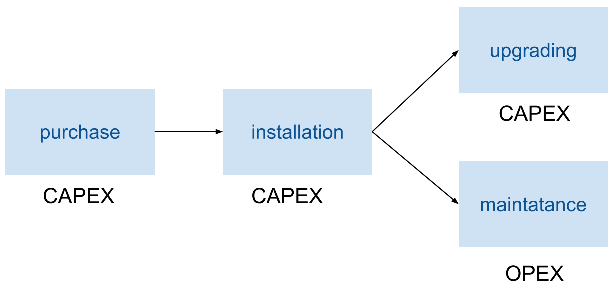
Good Performance – Cloud Solution

- Cloud Computing model
  - *No up-front hardware investments*
  - *Hardware optimally utilized*



CAPEX vs. OPEX

- Captial expenditure/capital expense (CAPEX)
  - *money to spend to buy, maintain or improve fixed assets*
    $\rightarrow$ *buildings, vehicels, equipment or land*
  - *have impact on costs vs. profit and tax*



- Operational expenditure (OPEX)
  - *ongoing costs for running a product, business, or systems*
  - *OPEX are entirely tax-deductible*
- Cloud lets you trade CAPEX for OPEX
  - *No investments in data centers and infrastructures*
  - *You pay only when you consume resources*

Cloud Computing Concepts

- **On-demand and self-service**
  - *Resources are provisioned as they are requested and when they are required*
  - *No human interaction, automatic*
- **Broad network access**
  - *Capabilities are available over the network*
- **Resource pooling**
  - *Provider's computing resources reused by multiple tenants (multitenancy)*
  - *Resourcces are dynamically assigned/re-assigned according to demand*
  - *Computing resources: CPU, memory, storage, network*
- **Scalability and elasticity**
  - *Infrastructure may grow and shrink according to needs*
  - *Automatic or manual*
- **Measured service**
  - *Resource usage can be monitored, controlled and reported*
- **Pay-per-use**
  - *Consumers only pay for resources when they use them*

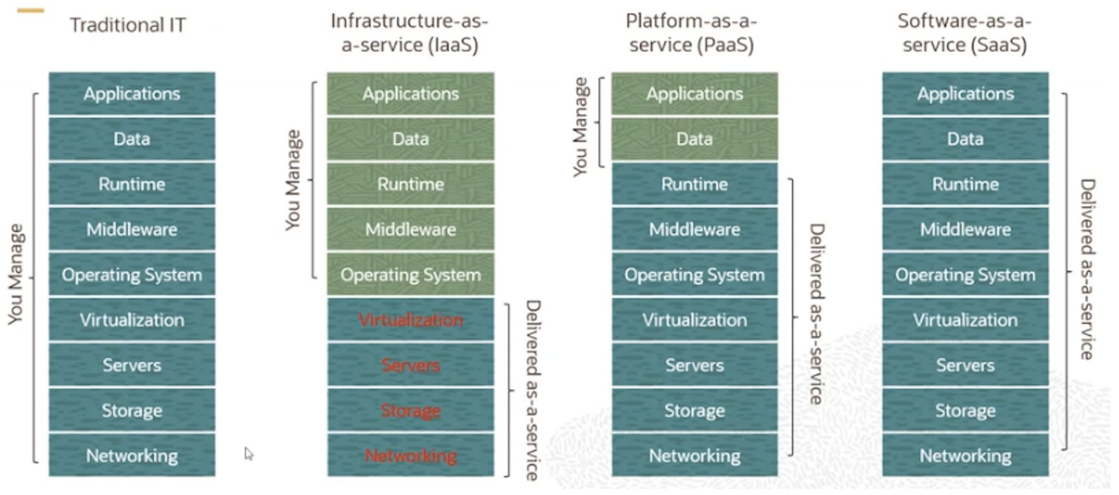
Cloud Computing Concepts (Cont.)

- Service Models (aka Cloud Layers)
  - *IaaS – Infrastructure as a Service*
  - *PaaS – Platform as a Service, Serverless*
    - *→ MWaaS, DBaaS, ...*
    - *→ FaaS*
  - *SaaS – Software as a Service*
- Deployment Models
  - *Public Cloud*
  - *Private Cloud*
  - *Hybrid Cloud*

Overview

- Introduction
- Cloud Architecture
  - *Service Models*
  - *Multitenancy*
- Infrastructure as a Service

Service Models



IaaS: Infrastructure as a Service

- Usage
  - *Predefined shapes of compute instances (e.g. micro, small, large, extra-large)*
    → for example: RedHat 7.8, 613 MB of memory, 1 TB block storage
  - *Pay-per-use – pay for resources you use (time or amount)*
    → no up-front costs
- IaaS Services Examples
  - *Load balancer*
  - *Autoscaling*
  - *Connectivity with on-premise network*
  - *Resource monitoring*
- IaaS providers
  - *Amazon EC2, GoGrid, Rackspace, OpenNebula, Google Cloud, Oracle OCI, ...*

PaaS: Platform as a Service

- Usage
  - *Choose software platform, e.g., JEE, .NET, Python, etc.*
  - *Pay-per-use – pay for the resources you use; no up-front costs*
  - *Cloud native, microservices, containers*
- PaaS features
  - *Serverless*
  - *Auto Scalling and Load balancing*
  - *Local development environment*
  - *Administration API*
- PaaS providers
  - *Google App Engine – first PaaS service*
  - *Today, mostly Kubernetes, Google, Heroku, Azure, AWS, Oracle*

SaaS: Software as a Service

- Software delivery model for applications hosted in the cloud
  - *typically software for end-users*
  - *services accessed using a web browser*
  - *provides API for programmatic access*
- SaaS characteristics
  - *Typically build on top of IaaS or PaaS*
  - *Configurable and customizable modern Web applications*
  - *Usually basic version for free, need to pay for "pro" version*
  - *Global availability - any computer, any device*
  - *Easy management - automatic and fast updates*
  - *Pay-per-use – pay for the time you use*
- SaaS providers
  - *Google Apps, Salesforce, ...*

Overview

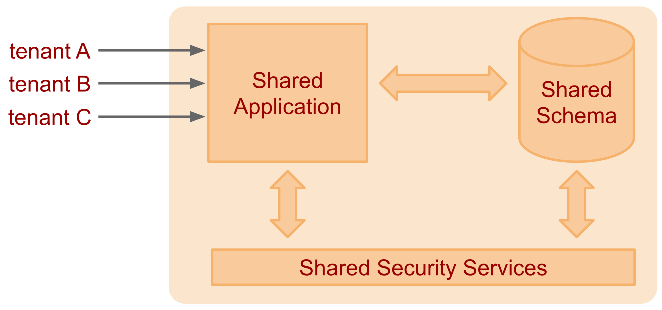
- Introduction
- Cloud Architecture
  - *Service Models*
  - *Multitenancy*
- Infrastructure as a Service

Multitenancy

- Architectural approach where resources are shared between multiple tenants or consumers
- Implications
  - *Centralization of infrastructure in locations with lower costs*
  - *Peak-load capacity increases*
  - *Utilisation and efficiency improvements for systems that are not well utilised*
- Sharing options
  - *Shared Everything*
  - *Shared Infrastructure*
    - → *Virtual Machines*
    - → *OS "virtualization"*

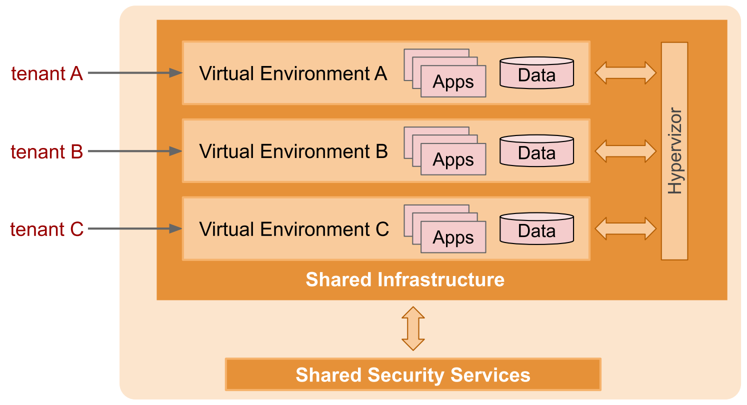
Shared Everything

- Resources are shared between all tenants or consumers
  - *tenant: a service consumer*
- Common for the SaaS model
- The application should provide tenant isolation
- Data for multiple tenants is stored in the same database tables



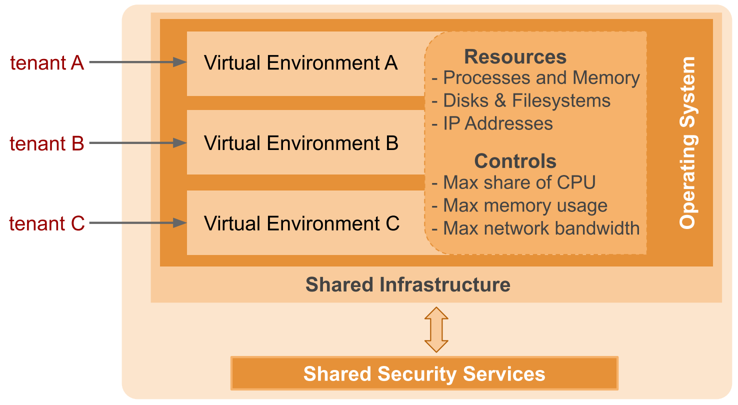
Shared Infrastructure: Virtual Machines

- Infrastructure shared via virtual machines
  - *each tenant has its own virtual environment*
  - *Isolation provided by hypervisor*
    - → *hypervisor: virtual machine manager, runs virtual machines*
  - *Resource contention depends on VM capability and configuration*
  - *Adds an additional layer and processes to run and manage*



Shared Infrastructure: OS Virtualization

- Infrastructure shared via OS Virtualization
  - *Each tenant has its own processing zone*
  - *Isolation provided by the operating system*
  - *Resource contention depends on zone configuration*
  - *No VMs to run and manage, no abstraction layer between app & OS*



Overview

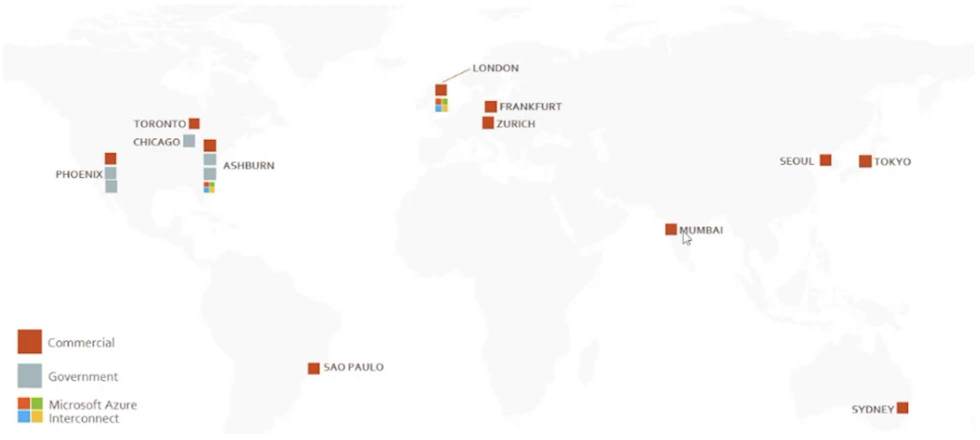
- Introduction
- Cloud Architecture
- Infrastructure as a Service
  – *Networking*
  – *Compute*
  – *Storage*
  – *Infrastructure as Code*

Overview

- Infrastructure = environment where your app is running
- Tenancy = your "space" in the cloud
- What you need
  – *Servers (compute instances) to run your app in a location (region)*
  – *Connectivity*
    → *Private network for intra-communication*
    → *Public network for internet communication*
    → *Firewall (security) rules*
    → *Route tables*
  – *Storage*
    → *Operating system*
    → *Your app data*
  – *Identity Management*
    → *Who and how can access and control your tenancy*
  – *Monitoring, Logging, Auditing*

Region

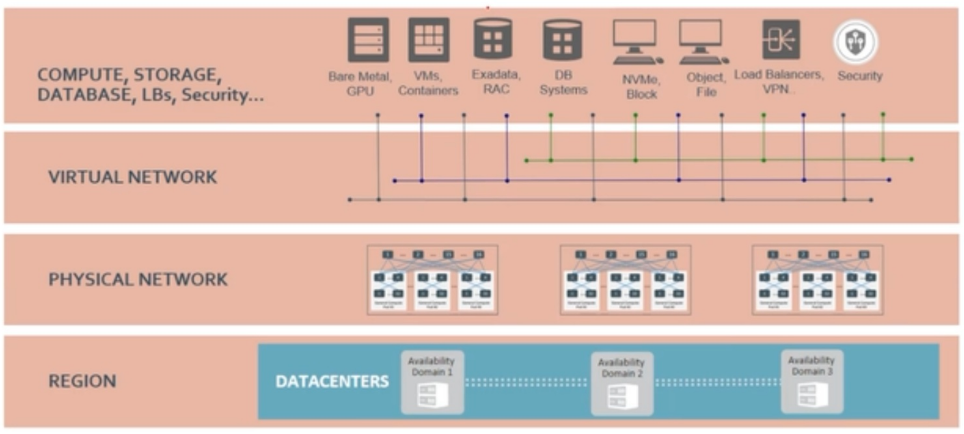
- Region = location on a planet where cloud data centers are located



- Why location matters...
  - *Latency - your users should be close to your app*
  - *Regulations - your data should be stored in EU*
  - *Connectivity to external providers*
    - → *Such as other cloud vendors*

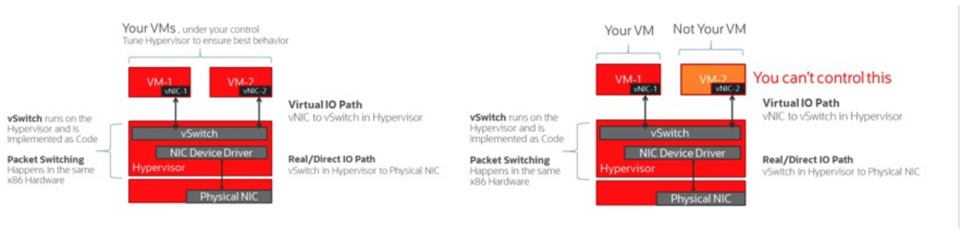
Datacenters

- Datacenter (aka Availability Domain – AD)
  - *Computing resources in a location within a region*
  - *One or more datacenters exsit in a region*
    - → *They are completely de-correlated, independent*
    - → *They have separated power supply, do not share underlying infrastructure*
    - → *If one DC fails, the other one is up and running*

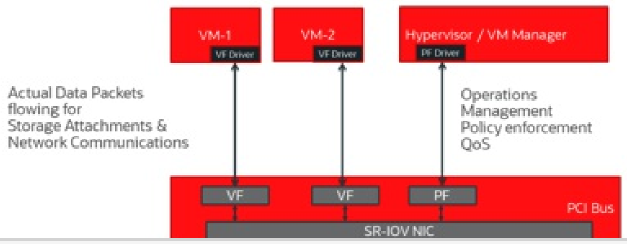


Off-box Network Virtualization

- Gen 1.0 Cloud, inefficient Resource Sharing
  - *On-Premise virtualization to share resources amongst multiple tenants in the cloud.*


Your VMs, under your control
Tune Hypervisor to ensure best behavior
VM-1
vNIC-1
VM-2
vNIC-2
Virtual IO Path
vNIC to vSwitch in Hypervisor
vSwitch runs on the Hypervisor and is implemented as Code
vSwitch
NIC Device Driver
Hypervisor
Packet Switching
Happens in the same x86 Hardware
Real/Direct IO Path
vSwitch in Hypervisor to Physical NIC
Physical NIC
Your VM
Not Your VM
VM-1
vNIC-1
VM-2
vNIC-2
You can't control this
Virtual IO Path
vNIC to vSwitch in Hypervisor
vSwitch runs on the Hypervisor and is implemented as Code
vSwitch
NIC Device Driver
Hypervisor
Packet Switching
Happens in the same x86 Hardware
Real/Direct IO Path
vSwitch in Hypervisor to Physical NIC
Physical NIC

- Gen 2.0 Cloud
  - *Smart-NIC accelerated SR-IOV (Single Root I/O Virtualization)*
    → Networking is a specialized function, needs hardware/sillicon to accelerate it

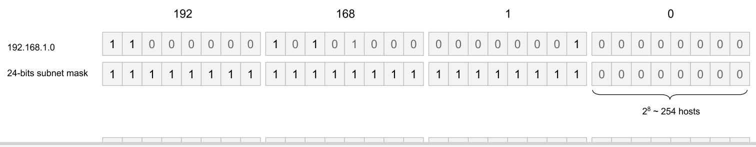


Overview

- Introduction
- Cloud Architecture
- Infrastructure as a Service
  - *Networking*
  - *Compute*
  - *Storage*
  - *Infrastructure as Code*

Virtual Cloud Network

- VCN = a private network in a **single region** in which your instances reside
- A a single and contigous IPV4 CIDR block of your choice
  – *CIDR (classless inter-domain routing) notation*
  – *IP address:*
    → *network prefix (the most significant bits) and*
    → *interfaces on the network (least significant bits) ~ network hosts*
  – *Example:* `192.168.1.0/24`
    → *IP range:* `192.168.1.0` - `192.168.1.255`
- You further create subnets on a VCN to organize your instances
  – *The subnets must be "within" the VCN, they can span across ADs*
  – *Example: using* `27` *bits for a subnet mask allow for 8 subnets*
    → `192.168.1.0/27`, `192.168.1.32/27`, `192.168.1.64/27`, ...
    → *Each subnet can have 32 hosts*

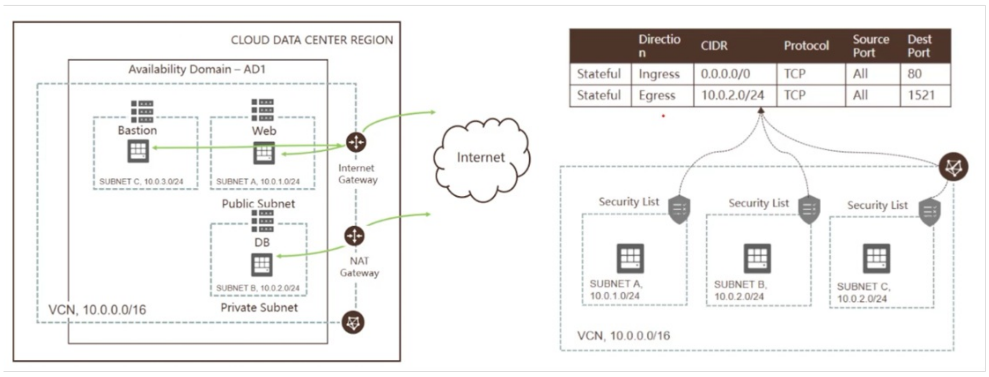


	192								168								1								0							
192.168.1.0	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
24-bits subnet mask	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

2^8 ~ 254 hosts

VCN Routing and Security

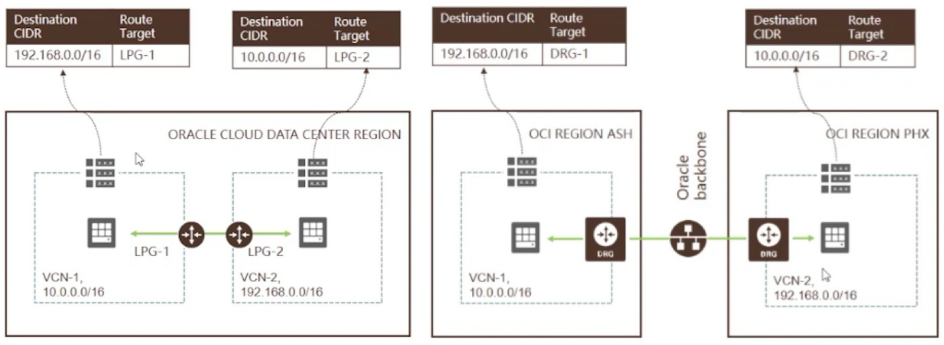
- Private and Public subnets
  – *Public can communicate in/out from/to Internet*
  – *Internet traffic routed to public subnet*
  – *Private can be completely isolated or communicate to Internet only*
- Route tables
  – *Required to route across subnets and in/out from the Internet*
- Security
  – *Control access to/from the subnet*



	Direction	CIDR	Protocol	Source Port	Dest Port
Stateful	Ingress	0.0.0.0/0	TCP	All	80
Stateful	Egress	10.0.2.0/24	TCP	All	1521

Peering

- Local Peering
  - *Connecting two VCNs in a region*
- Remote Peering
  - *Connecting two VCNs across regions*
- Connectivity with on-premise datacenter
  - *Fast connection needs to be in place*
  - *Secure VPN needs to be established*



Destination CIDR	Route Target
192.168.0.0/16	LPG-1

Destination CIDR	Route Target
10.0.0.0/16	LPG-2

Destination CIDR	Route Target
192.168.0.0/16	DRG-1

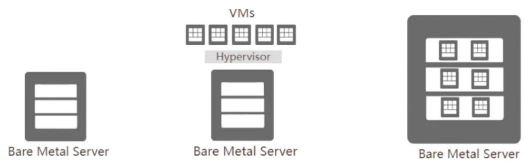
Destination CIDR	Route Target
10.0.0.0/16	DRG-2

Overview

- Introduction
- Cloud Architecture
- Infrastructure as a Service
  - *Networking*
  - *Compute*
  - *Storage*
  - *Infrastructure as Code*

Compute Instances

- Shape = amount of memory and CPU an instance is using
  - *There are classes of shapes that you can choose from*
  - *Standard and HPC/GPU shapes*
- Virtual Machine (VM) – multi-tenant model
  - *A hypervisor to virtualize the underlying Bare Metal server into smaller VMs*
- Bare Metal (BM) – single-tenant model
  - *Direct hardware access, full bare metal server*
  - *Types of workloads: performance intensive, require a specific hypervisor*
- Dedicated VM Hosts (DVM) – single-tenant model
  - *VM instances running on dedicated single-tenant servers*
  - *Not shared with other customers*



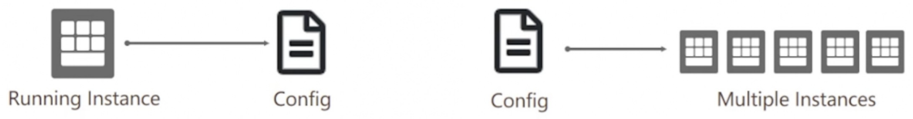
- States: start, stop, reboot, terminate
  - *Billing pauses in STOP state but depends on shape*

Image

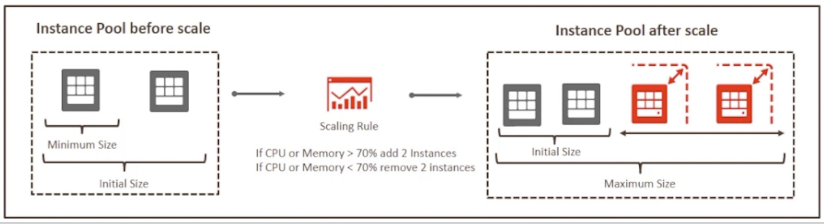
- Image
  - *A template of a virtual hard drive with OS*
  - *Other software, libraries, configurations, etc.*
- Stored on a boot volume
- Base images are provided
  - *CentOS, Ubuntu, Windows Server, Oracle Linux, RedHat, etc.*
  - *Some may require licence costs*
- Custom images
  - *You can create a custom image from the base image*
  - *Specific packages, libraries or custom configuration*
  - *You store the image in the object storage*

Autoscaling

- Instance configuration
  - *OS image, metadata, shape, vNICs, storage, subnets*
  - *Apply configuration to multiple instances at the same time*
    - → *You can manage them all together (start, stop, terminate)*

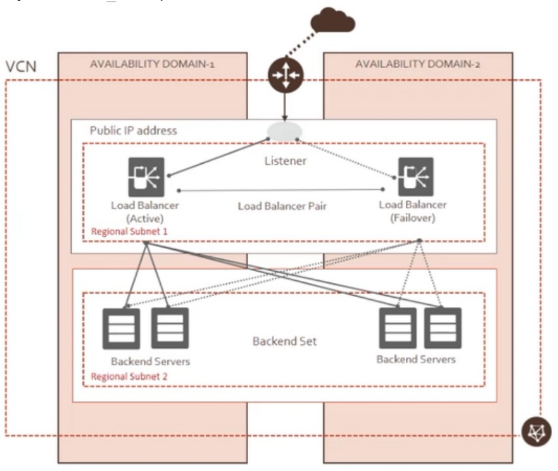


- Autoscaling
  - *Automatically adjust a number of compute instances in an instance pool*
  - *Control using performance metrics such CPU or memory utilization*
  - *cooldown period – time between scale in and scale out (e.g. 300 seconds)*


Instance Pool before scale
Minimum Size
Initial Size
Scaling Rule
If CPU or Memory > 70% add 2 Instances
If CPU or Memory < 70% remove 2 instances
Instance Pool after scale
Initial Size
Maximum Size

Load Balancer

- Managed service
  - *Health check – checks health status of backends (TCP, HTTP)*
  - *Algorithm – round-robin, IP hash, least connections*
- Supports protocols
  - *TCP, HTTP 1.1, HTTP/2, WebSocket, SSL termination, end-to-end SSL*
  - *Supports sticky sessions (sessions persistence)*
- High Availability
  - *Primary and stand-by LB*
  - *Each LB is in different AD*
  - *Failover uses floating IP*



Overview

- Introduction
- Cloud Architecture
- Infrastructure as a Service
  – *Networking*
  – *Compute*
  – *Storage*
  – *Infrastructure as Code*

Object Storage

- Types of data to store
  – *Storage for unstructured data (images, media files, logs, backups)*
  – *Data managed as objects, provides API using HTTP verbs*
- Namespace
  – *Logical entity that serves as top-level container for all bueckts and objects*
  – *Each tenancy is provided one unique namespace*
- Bucket
  – *A logical container for storing objects*
  – *Bucket names must be unique within tenancy*
  – *Hot bucket – standard, can be accessed immediatelly*
  – *Cold bucket – rarely accessed data, need to be restored*
    → *Minimum retention, such as 90 days*
    → *Time to First Byte (TTFB) is in hours, e.g. 4 hours*
- Object and metadata
  – *data managed as objects regardless data type*
- Example object URL path:

```
/n/<namespace>/b/<bucket>/o/<object_name>
```

Block Storage

- Local NVMe SSD device
  - *Locally attached device, provided by some shapes, 200K IOPS - 1M IOPS*
  - *Workloads that require high storage performance*
  - *usually no RAID, snapshots, backups*
- Block volumes
  - *Reside in storage servers*
  - *NVMe SSD based, up to 35K IOPS*
  - *Data stored on block volumes beyond the lifespan of compute instance*
  - *Multiple replicas across multiple storage servers*
- File Storage
  - *Network file server (NFS)*
  - *Client mounts a **mount target** (NFS endpoint) and an **export path***
  - *Example*

```
sudo mount 10.0.0.6/example1/path /mnt/mountpointA
```

Overview

- Introduction
- Cloud Architecture
- Infrastructure as a Service
  - *Networking*
  - *Compute*
  - *Storage*
  - *Infrastructure as Code*

Overview

- Definition
  – *Application envs (in a cloud) managed via definition files*
  – *Version control, team development, scripting, etc.*
- Major Technologies
  – ***Configuration Management Tools***
  → *install and manage software on machines that already exist*
  → *Examples: Ansible, Chef, Puppet*
  – ***Abstraction of cloud infrastructure***
  → *Terraform*

Terraform

- Higher-level abstraction of the datacenter and associated services
- Supports many service providers
  – *Google, Microsoft, Oracle, AWS*
- Steps
  1. *Description of resources in Hashicorp Configuration Language (HCL)*
     – *instances, networks, firewall rules, routing tables, etc.*
  2. *Terraform generates execution plan to reach the desired state*
  3. *Terraform executes the plan to reach the desired state; can generate incremental execution plan*