Hybrid clouds &
Container management platforms

Pelle Jakovits

April 2022, Tartu
Outline

• Hybrid Clouds
  – Use cases and necessity
  – Benefits and disadvantages
  – AWS Hybrid Cloud services
  – VMware

• Container management platforms and Hybrid Clouds
  – Kubernetes and Hybrid Clouds
Private vs Hybrid vs Multi Clouds

• Private Clouds
  – Set up cloud under a single organization hosted in a dedicated physical infrastructure
  – OpenStack, OpenNebula, Kubernetes (containers)
  – Risks: overprovisioning and underprovisioning.

• Hybrid Clouds
  – Scale the private clouds by utilizing Public Cloud resources when needed
  – Risks: Issues with latency and synchronization of data

• Multi-Clouds
  – Utilize multiple public clouds. Perhaps together with a private cloud.
  – Risks: Overly complicated architecture and deployments
  – Benefits: Avoid vendor lock-in by design.
Hybrid clouds

• Hybrid clouds combine on premise infrastructure with off premise resources (e.g., public cloud)
  – Such as AWS, IBM Cloud, Google Cloud or MS Azure.
• Local cloud bursting or shrinking
  – Provision or remove external resources when required.
• Vendor specific hybrid clouds
  – Microsoft azure stack, VMware Hybrid Cloud
Hybrid cloud architecture

Ref: Rajkumar et.el Mastering Cloud Computing
Necessity of hybrid clouds

• Untested workloads with uncertain demand
• Cloud bursting
• Disaster recovery and high availability
• Improving the geo-locality of services. Bringing services closer to end-users
• Complying with data-localization and privacy demands
• Reducing the effect of Cloud vendor-lockin
Use cases

• **Health care sector**: Security and compliance in storing and movement of health data

• **Finance sector**: Flexibility, power, scale, and seamless connectivity

• **Government**: Use cloud email and collaboration tools and a mixture of public and private cloud infrastructure

• **Retail**: Always in (99.999%) uptime support in ecommerce operations. Reduce the risk of one Cloud provider failures.

• **eCommerce**: Enormous web traffic and scalability. Utilize availability zones of multiple clouds.

AWS Hybrid Cloud

• AWS hybrid cloud solutions and services for running and managing applications and resources wherever they reside
• Can use the same services, APIs, and tools wherever need it – from the cloud, to on premises, and to the edge.
• Need to deploy services on premises or in a specific location due to low latency or data locality requirements.
• Hybrid Infrastructure solutions - AWS outposts, AWS wavelength, AWS local zones
• Rugged & Disconnected Edge – AWS Snowball, AWS Snowcone
AWS Outposts

- Fully managed service offers the same AWS infrastructure, AWS services, APIs, and tools.
- AWS compute, storage, database, and other services run locally on Outposts.
- Outposts is available as a 42U rack that can scale from 1 rack to 96 racks to create pools of compute and storage capacity.
- Outposts will be available in two smaller form factors, 1U and 2U rack-mountable servers for locations with limited space or capacity requirements.
How it works?

1. Order
   From the AWS Console select your Outposts configuration consisting of a mix of compute and storage capacity

2. Install
   AWS personnel deliver Outposts to your site, connect Outposts to power, set up network connectivity to AWS Region and your local networks

3. Launch
   Use standard AWS APIs or Management Console to launch EC2 instances on your Outpost

4. Build
   Build and run apps using native AWS services running on Outposts or in the AWS Region

AWS Outposts
Run AWS infrastructure and services on premises
Rugged & Disconnected Edge – AWS Snowball

• Used for edge computing, data migration and edge storage.
• It has two options
  – Snowball Edge Storage Optimized devices - block storage and Amazon S3-compatible object storage, and 40 vCPUs.
  – Snowball Edge Compute Optimized - 52 vCPUs, block and object storage, and an optional GPU.
• These devices may also be rack mounted and clustered together to build larger temporary installations.
• Snowball supports specific Amazon EC2 instance types and AWS Lambda functions
• Develop and test in the AWS Cloud, then deploy applications on devices in remote locations to collect, pre-process, and ship the data to AWS
AWS Snow family

https://www.javatpoint.com/aws-snowball
AWS Snowball

- Snowball is a petabyte-scale data transport solution
- It is a streamline bringing the data into AWS and bypassing an internet.
- High network costs, long transfer time, and a security issue have been resolved by using Snowball

https://www.javatpoint.com/aws-snowball
AWS Snowball Edge

• Snowball Edge is a 100 TB data transfer device with on-board storage and compute capabilities.
• Snowball Edge is like an AWS data-center that you can bring on-premises.
• Run Lambda functions from Snowball edge
• Faster Data transfer: It transfers the data with a speed of up to 100 GB/second.
• Clustering: You can cluster Snowball edges together to form a local storage tier and process your data on-premises to achieve 99.99% data durability across 5-10 devices

Example: Aircraft engine manufacturer

• https://www.javatpoint.com/aws-snowball
AWS Snowball mobile

• A Snowmobile is an exabyte-scale data transfer service.
• It can transfer large amounts of data in and out of AWS.
• You can transfer 100 PB per Snowmobile, a 45-foot long ruggedized shipping container, pulled by a semi-trailer truck.
• Transferring data with Snowmobile is secure, fast and cost-effective.

https://www.javatpoint.com/aws-snowball
VMware Hybrid Cloud

• VMware hybrid cloud is backed by portfolio of services providing the software defined building blocks for virtualized infrastructure to build and run the applications.

• VMware has its vendor specific solutions to build hybrid cloud

• It primarily consists of Software defined data center components:
  – vSphere & vCenter for compute virtualization and management
  – vSAN for storage virtualization
  – NSX for network virtualization
VMware hybrid cloud solutions

- **VMware Cloud Foundation** with embedded Kubernetes capabilities
- **VMware Cloud Foundation on Dell EMC** physically installed in on-premise facilities and maintained as a fully-managed service
- **Hyperconverged Infrastructure (HCI) powered by VMware vSAN** available on Dell EMC VxRail and vSAN ready nodes from multiple hardware vendors
- **VMware Cloud on AWS jointly engineered by VMware and AWS** and powered by the VMware Cloud Foundation SDDC stack running on AWS infrastructure
- **Azure VMware Solution**
- **Google Cloud VMware Engine**
- **IBM Cloud for VMware Solutions**
- **Oracle Cloud VMware Solution**
Hybrid Cloud: VmWare and Microsoft Azure Integration

Hybrid cloud management platforms

• Automate, orchestrate the public and private clouds by enforcing the policies with automated workflow management.
  – ManageIQ, Redhat Cloudforms, Morpheus, Cloudbolt

• Features
  – Self service provisioning
  – Cloud workload management
  – Show back/Chargeback
  – Bursting workloads
  – Capacity planning and management
  – Leveraging Existing infrastructure
Benefits of hybrid clouds

• Flexibility and agility
• Elasticity
• Faster delivery of new products and services
• Cost control
• Avoidance of lock-in
• Access to the latest technology
Challenges of hybrid cloud

- Possibility of vendor lock in
- Service integration and Interoperability challenges
- Transparency
- Security and privacy issues
- Data locality issues
  - Data replication and sharing
- Resource management and scheduling in hybrid clouds
  - Metrics: QoS, SLA, Failure, profit
KUBERNETES
Kubernetes

- Open source platform for deploying and managing containerized services
- Based on the concept of Google Borg (~2004)
- Released in June 2014
- Big developer base and releases every 3 months
- Infrastructure as a Code (IaaS)
- Often used for automatic the deployment and scaling of microservices
Kubernetes building blocks

• K8s cluster consists of a set of worker nodes and control services (also running on some nodes)

• A set of management services are used to control the Kubernetes cluster
  – Control Plane Components
    • REST API server, Scheduler, Etcd database, Controller manager

• Worker nodes run:
  – Kubelet, k-proxy, container runtime
  – Responsible for running the containers
Kubernetes Components

https://kubernetes.io/docs/concepts/overview/components/
API server

• Exposes the Kubernetes API
• Front-end for the Kubernetes control plane
• Supports running several Kube-API servers in parallel and traffic is balanced between them
etcd

• Consistent and highly-available key-value database
• Used for coordination and distributed configuration store
  – kubectl get commands retrieve data from etcd
  – kubectl commands modify data in etcd
• "Distributed /etc directory"
  – /etc is used as configuration directory in Linux systems
• Leader based distributed database
  – But a node does not need to know who is leader
  – All requests to any node (that require consensus) are forwarded to leader
Controller Manager

• Control plane component that runs controller processes
  – Each sub-controller is a separate process
  – Can run on "any" node in the k8s cluster

• Controllers verify and ensure current vs desired state of the cluster and its resources

• Types of k8s controllers:
  – **Node controller**: Responsible for noticing and responding when nodes fail, are removed or added.
  – **Job controller**: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion
  – **Endpoints controller**: Populates the Endpoints object
  – **Service Account & Token controllers**: manage accounts and API access tokens
Scheduler

• Control plane service
• Listens to newly created and unallocated Pods
  – Chooses which node to run them on
• Scheduling decisions take into account:
  – individual and collective resource requirements,
  – hardware/software/policy constraints,
  – affinity and anti-affinity specifications,
  – data locality,
  – inter-workload interference
  – deadlines.
Kubelet

• Primary "node agent"
  – An agent that runs on each cluster node
• A PodSpec is a YAML or JSON object that describes a pod
• The kubelet takes a set of PodSpecs that should be deployed on the node
  – Checks the PodSpecs against the state of currently running containers on the node
  – ensures that the containers described in those PodSpecs are running and healthy on this node
kube-proxy

• Network proxy that implements the k8s Service concepts
• Deployed on each worker node
• Maintains network rules on nodes to enable communication between Pods and other nodes and outside world
• Either uses the operating system packet filtering (e.g. iptables) or forwards traffic by itself
Container runtime

• Software responsible for running containers

• Kubernetes supports
  – containerd
  – CRI-O (to support Open Container Initiative containers)
  – Docker Engine (Which internally uses containerd)
  – Mirantis Container Runtime (formerly Docker Enterprise)
  – ... any that implements Kubernetes Container Runtime Interface (CRI)
Kubernetes Components

https://kubernetes.io/docs/concepts/overview/components/
Kubernetes characteristics

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management
Kubernetes building blocks

• Namespace
• Pod
• Deployment
• ReplicaSet
• Service
• Ingress
• Configmaps
• Secret
Pod

- Smallest deployable k8s unit of computing
- Group of containers
- Pod has a unique k8s internal IP address
  - Logical micro-host
- When scaling in k8s – replicas of pods are scaled
- Two pods of the same Deployment usually run on different worker nodes
- Only tightly coupled containers should be in same pod
  - When they need to share some resources (e.g., Volumes)

https://kubernetes.io/docs/concepts/workloads/pods/
https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/
Deployment

• Contains a template of a pod
  – Specifies the desired state of the Pod
• Creates and manages Replica Sets
• Deployment can be scaled up and down
  – Changes the number of Pod replicas
• Roll out new versions of pods
• Can be rolled back
• Kubernetes Deployment Controller continuously monitors the state of the deployment
Services

• Each Pod in a Kubernetes cluster has a unique internal IP address
  – Services define how to group and access those pods (from outside)

• A k8s service defines how traffic is routed across a set of Pods

• Handles discovery and routing

• Services using labels and selectors to match Pods

• Labels can specify:
  – Different apps
  – Different versions of the services
  – Pods are targeted at development, test, or production

• Services define how to load balance traffic

https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/
Scaling

• Once we have defined a service, we can scale the number of Pods inside the service
  – Requires definition of ReplicaSet – marks Pods scalable. We would not want to scale non-scalable components – like single pod DB.

• Having multiple replicas enables rolling updates without downtime
  – New version Pod can be deployed one replica at a time

• Can scale service to 0

• Supports the implementation of auto-scaling

https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/
Rolling Update

https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/
Data and volumes

- Volume is a directory accessible to containers in a pod
- On-disk files in a container are **ephemeral**
  - K8s destroys ephemeral volumes when Pod is destroyed
- **K8s persistent volumes (PV)** exist after pod lifetime
  - PV is a unit of storage that has been provisioned
- **PersistentVolumeClaim** – request of using a PV storage unit – e.g., to attach it to a Pod
  - PV Storage classes:
    - AWSElasticBlockStore,
    - AzureFile,
    - NFS,
    - Glusterfs,
    - Local
    - Etc.
Kubernetes: Orchestration or choreography?

• Kubernetes tries to eliminate the need for precise orchestration
• A set of independent control processes continuously update the current state to achieve a desired state - as specified by the provided templates
• Centralized control is not strictly required.
• The drive is to provide an easier to use and a robust, resilient, and extensible platform.
• Define strict dependencies, but otherwise - let the schedulers do their work
Cloud controller manager

• It is an optional component
  – Beneficial when deploying Kubernetes directly on Cloud infrastructure.
  – Not needed when running Kubernetes on-premises or in a single node-mode.

• Provides a cloud provider specific control logic and allows Kubernetes to directly manage Cloud services through provider API.
  – Includes different controllers for each supported Cloud platform.

• K8s controllers, which benefit from cloud-controller features:
  – **Node controller**: Directly checks the state of cloud infrastructure through cloud API. E.g. to see if a node has been deleted when it no longer responds.
  – **Route controller**: Deploying Virtual cloud networks and routes.
  – **Service controller**: For configuring Cloud load balancer services. Adding, removing services to load balancing groups.
Kubernetes federation

KUBERNETES V2 DEPLOYMENT EXAMPLE

HOST CLUSTER

FEDERATION SYSTEM

FEDERATION CONTROL PLANE

PUSH RECONCILER

KUBECONFIG
M. CLUSTER-1
CR

KUBECONFIG
M. CLUSTER-2
CR

MEMBER CLUSTER 1

NS
TEST

SATELLITE
Deployment

MEMBER CLUSTER 2

NS
TEST

SATELLITE
Deployment

https://banzaicloud.com/blog/hybrid-cloud-kubernetes/
Service Mesh

[Diagram showing the components of a Service Mesh system, including PODs "A" and "B", envoy proxies, Mixer, Pilot, Citadel, Sidecar injector, and Istio Operator.]
Kubernetes service mesh based container Hybrid Cloud

Istio single mesh without flat network or vpn
Kubernetes service mesh based container Hybrid Cloud

https://techblog.cisco.com/blog/istio-multicluster-federation-2
Kubernetes alternatives and flavours

• Docker Swarm
• Red Hat OpenShift
• Nomad
• Amazon Elastic Container Service
Conclusions

• Hybrid Clouds can be used to dynamically augment on-premise resources
  – Enables to bring services closer to end users
  – Helps to deal with sudden spikes

• Muti-cloud approach helps to lessen dependance on a single provider
  – Extends the reach of your resources

• Kubernetes simplifies management containers and microservices
  – Largest community among container orchestration tools
  – Kubernetes can run on-premises bare metal, OpenStack, Google, Azure, AWS, ...
    • Most cloud providers have managed Kubernetes services
  – Helps to avoid vendor lock issues
That’s all for today

• This week’s practice session
  – Setting up Kubernetes
  – Deploying Flask application
  – Managing secrets, scaling and availability of services

• Next week’s lecture
  – Cloud deployment modelling

• In Two weeks
  – Guest lecture from Vjatšeslav Antipenko
  – Head of IoT in Telia Estonia
  – Everyone should attend
THANK YOU FOR YOUR ATTENTION

jakovits@ut.ee