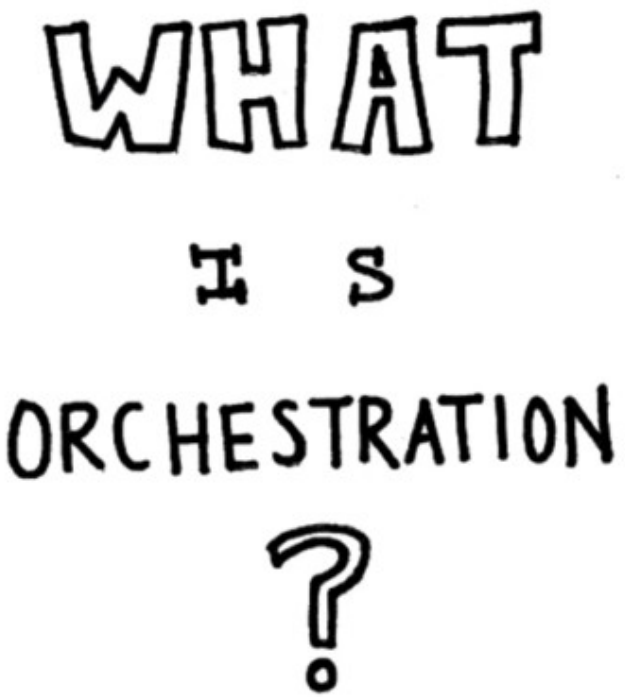
Cloud Orchestrators

- *Open Data Management & the Cloud*
- (Data Science & Scientific Computing / UniTS – DMG)

What are orchestrators?

★ Orchestrator is a management tool or platform that automates provisioning of services.

Why orchestrators?



THE SIMILITUDE OF THE SYMPHONY ORCHESTRA

What are orchestrators?

Just as for an orchestra conductor the challenge is to make sure that all instruments play the same piece of music with the right timing, so a cloud orchestartor has to control and synchronize the release of services.

The similitude of the symphony orchestra

SUPPOSE IT IS AN INFAMOUS SYNPHONY ORCHESTRA

BUT...WHY IS THE ORCHESTRA INFAMOUS?

BECAUSE ...

THE STRINGS PLAY A DIFFERENT NOTE THAN THE BRASS,

THE BRASS PLAY A DIFFERENT NOTE THAN THE WOODWINGS,

THE PERCUSSINISTS PLAYS A RIDICULOUS TEMPO

THAT IS MUCH TOO FAST FOR ANYONE TO KEEP

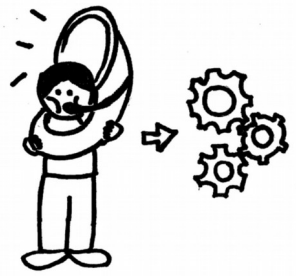
BOB DECIDES TO BECOME A CONDUCTOR

FOR THE INFAMOUS ORCHESTRA.

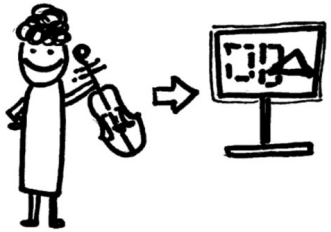
SO BOB HAS TO TELL EACH DEPARTMENT WHICH NOTES TO PLAY AND WHICH TEMPO TO PLAY AT.

IMAGINE THE PERCUSSIONISTS REPRESENT HARDWARE ...

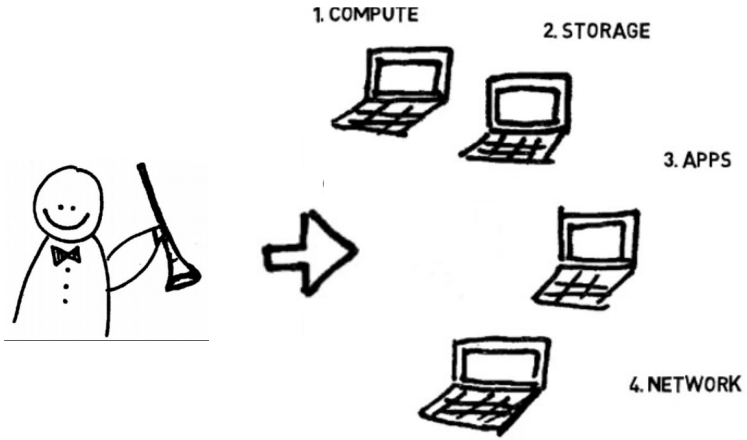
... AND THE BRASS REPRESENTS THE CONFIGURATION ELEMENTS, WHICH TELLS US HOW TO UTILIZE THE VMS AND RESOURCES



THE STRINGS REPRESENT THE SOFTWERE, SUCH AS THE VIRTUAL MACHINES

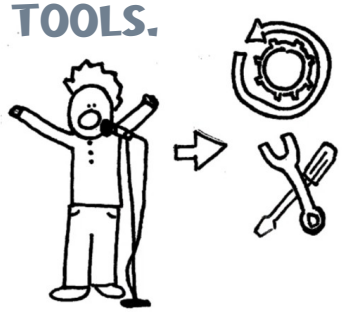


THE WOODWINDS REPRESENT THE RESOURCES PLANE ...

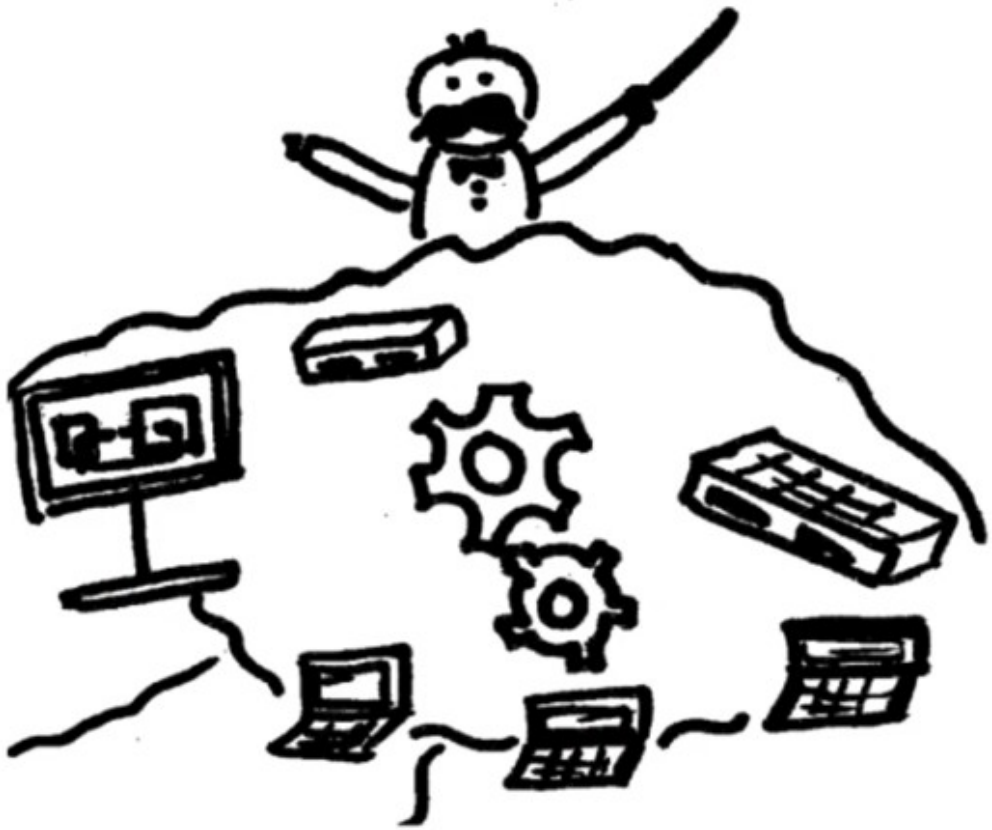


THE VOCALIST REPRESENT AUTOMATION AND ORCHESTRATION TOOLS.

DEV OPS MANAGEMENT AND MONITORING TOOLS



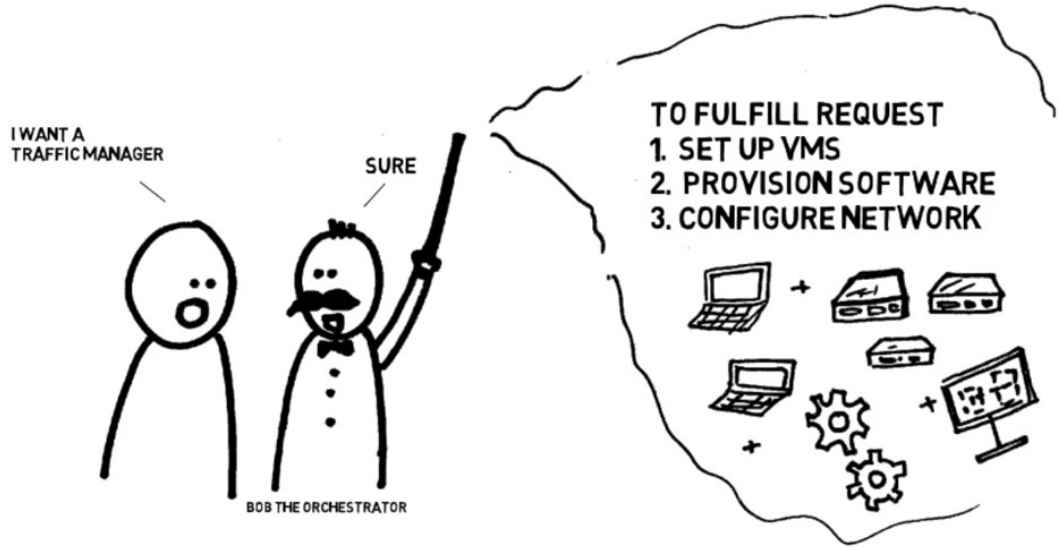
JUST LIKE HOW BOB CONDUCTS THE INSTRUMENTS IN THE ORCHESTRA TO PLAY A PIECE, ORCHESTRATION CONDUCTS ALL INDIVIDUAL ELEMENTS THAT ARE NEEDED TO COMPLETE AN OPERATIONAL TASK.



ORCHESTRATION HELPS TO ARRANGE AND DIRECT THE SERVICE DELIVERY OF APPLICATIONS, NETWORK, AND CLOUD SERVICES.

WITHOUT ORCHESTRATION OR BOB, THE CONDUCTOR, LIFE BECOMES FRUSTRATING AND TIME CONSUMING IN ORDER TO SATISFY EFFICIENT SERVICE DELIVERY.

ORCHESTRATION HELPS TO ARRANGE AND DIRECT THE SERVICE DELIVERY OF APPLICATIONS, NETWORK AND CLOUD SERVICES.



BY PROVIDING A HIGHER LEVEL OF AGILITY, MANAGEABILYTY, CONTROL AND VISIBILITY, COMPLEX MATTERS SUCH AS SECURITY AND INTERDEPARTMENTAL ISSUES ARE EASILY RESOLVED

THIS IS WHY A FAMOUS ORCHESTRA NEEDS BOB AND ...

WHY THE CLOUD SEVICE PROVISONING NEEDS ORCHESTRATORS

Orchestration vs. automation

★Automation describes a task or function accomplished without human intervention

★Orchestration describes the arranging and coordination of automated tasks, resulting in a consolidated process or workflow

★Cloud automation is a technical task, cloud orchestration is an IT workflow composed of tasks

Cloud orchestration

★ Cloud orchestration is the process to manage multiple workloads, in an automated fashion, across several cloud solutions (private, public, hybrid), with the goal being to synthesize this into a single workflow

Different Orchestrators

We are going to describe three different orchestrators:

- ★Amazon CloudFormation
- ★Heat (OpenStack)
- ★Kubernetes (k8s)

AWS CloudFormation

AWS CloudFormation

enables users to orchestrate the instantiation of multi-instance services via templates.

CloudFormation's sample templates or user's created templates can be used to describe the AWS resources, and any associated dependencies or runtime parameters, required to run the desired applications

AWS CloudFormation: how it works

Code your infrastructure from scratch with the CloudFormation template language, in either YAML or JSON format, or start from many available sample templates

Check out your template code locally, or upload it into an S3 bucket

Use AWS CloudFormation via the browser console, command line tools or APIs to create a stack based on your template code

AWS CloudFormation provisions and configures the stacks and resources you specified on your template

Heat the OpenStack cloud orchestrator (1)

★The mission of the OpenStack Orchestration program is to create a human- and machine-accessible service for managing the entire lifecycle of infrastructures and applications within OpenStack clouds

Heat the OpenStack cloud orchestrator (2)

- ★Heat is an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files
- ★Native Heat template format exists
- ★Heat Orchestration Template (OHT) provides compatibility with the AWS CloudFormation template format so existing CloudFormation templates can be launched on OpenStack

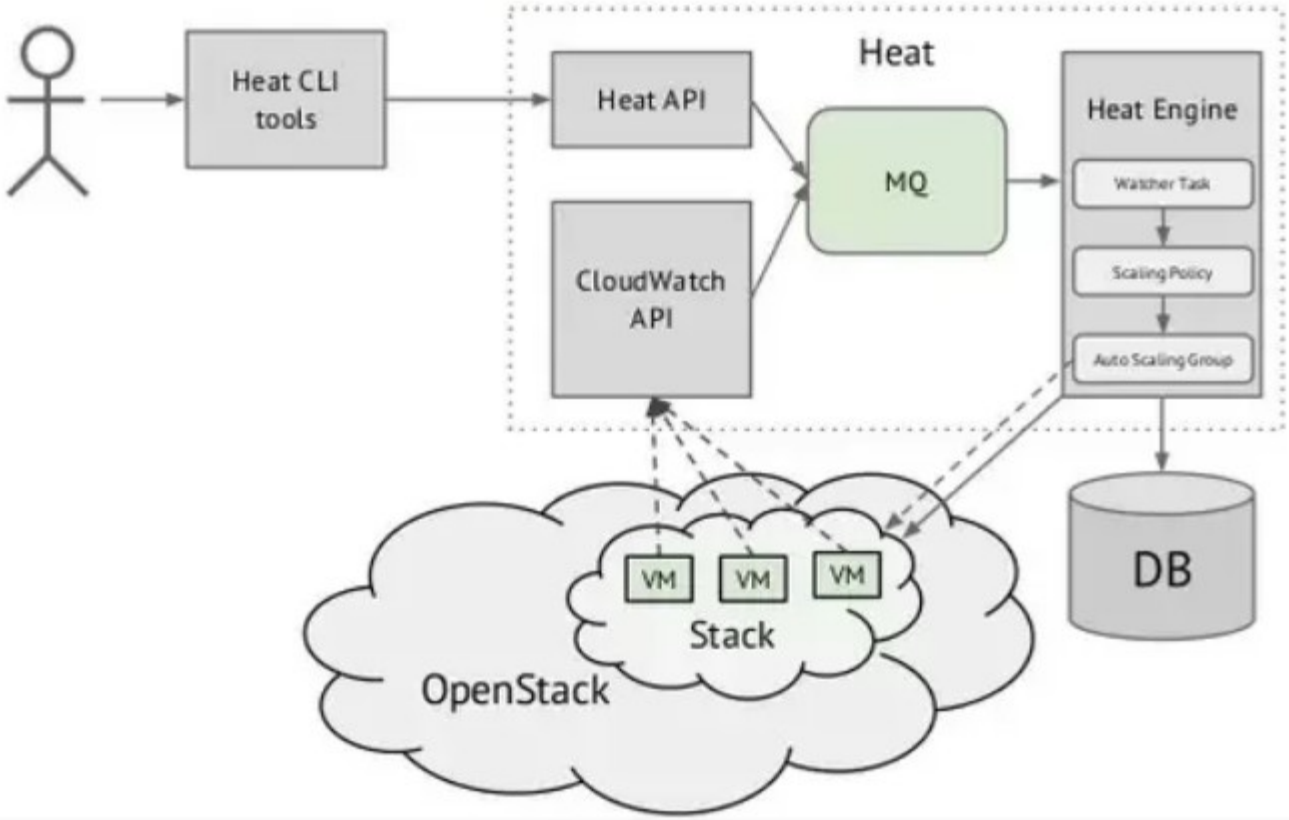
How Heat works (1)

- ★A Heat template describes the infrastructure for a cloud application in a text file readable and writable by humans, and can be treated as code (checked into version control, diffed, &c.)
- ★Infrastructure resources that can be described include: servers, floating ips, volumes, security groups, users, etc.
- ★Heat also provides an autoscaling service integrated with Telemetry, to include a scaling group as a resource in a template.

How Heat works (2)

- ★Templates can also specify the relationships between resources (e.g. this volume is connected to this server). This enables Heat to call out to the OpenStack APIs to create all the infrastructure in the correct order to completely launch the requested applications.
- ★Heat manages the whole lifecycle of the application. To change the infrastructure, it must simply modify the template and use it to update the desired stack.
- ★Heat templates integrate well with software configuration management tools such as Puppet

Heat basic workflow



Heat components

- ★heat: CLI that communicates with the heat-api.
- ★heat-api: component that provides an OpenStack-native ReST API that processes the requests and sends them to the heat-engine.
- ★heat-api-cfn: provides an AWS-style Query API that is compatible with AWS CloudFormation and processes the requests and send them to the heat-engine.
- ★heat-engine: is the brains of the operation and does the main work of orchestrating the launch of templates and providing events back to the API consumer.

Kubernates

Kubernetes, or k8s (k, 8 characters, s), or “kube” is an open source platform that automates Linux container operations.

Groups of hosts running Linux containers can be clustered spanning hosts across public, private, or hybrid clouds and Kubernetes helps to easily and efficiently manage those clusters.

Kubernates features

Kubernates is a platform to schedule and run containers on clusters of physical or virtual machines

- ★Orchestrate containers across multiple hosts.
- ★Make better use of hardware to maximize resources needed to run the enterprise apps.
- ★Control and automate application deployments and updates.
- ★Mount and add storage to run stateful apps.
- ★Scale containerized applications and their resources on the fly.
- ★Declaratively manage services, which guarantees the deployed applications are always running how you deployed them.
- ★Health-check and self-heal your apps with autoplacement, autorestart, autoreplication, and autoscaling.

ANALOGY OF SHIPS

TO UNDERSTAND THE

ARCHITECTURE OF KUBERNETES

Purpose of kubernetes is

- host user's applications in the form of containers,
- in an automated fasion
  so it can easily deploy as many instances of user's applications as required and easily enable communication

Two kind of sheeps in this example:

- cargo ships: do the actual work of carrying containers across the see

- control ships: responsible for monitoring and managing the cargo ships

The kubernetes cluster consists of a set of nodes

- physical or virtual
- on a private or public cloud
- hosting applications in the form of containers

In analogy with cargo ships,The worker nodes are ships that can load containers.

Two kind of sheeps in this example:

- cargo ships: do the actual work of carrying containers across the see

- control ships: responsible for monitoring and managing the cargo ships

The kubernetes cluster consists of a set of nodes

- physical or virtual
- on a private or public cloud
- hosting applications in the form of containers

In analogy with cargo ships,The worker nodes are ships that can load containers.

the control ships relate with the master node in kubernetes cluster. The master node is responsible for

- managing the kubernets cluster,
- storing information regarding different nodes,
- planning which containers goes where,
- monitoring nodes and containers on each ship etc.

The master node does all of this using a set of components toghether known as the control plane components.

There are many containers loaded and unloaded from the ships on a dayly basis
it is neededto maintain information about the different ships,
what container is in which ship,
at which time it has been loaded and so on.
All this information is maintained by a high available key-value store (database) known as etcd.

When ship arrive, you load containers on them using crame.

The crame identify the container that needs to be placed on ship,

identify the right ship based on

- its size,
- its capacity,
- the number of containers over the other ships

and any other condition such as

- the definition of the ship,
- the type of containers it is allowed to carry
- etc.

those are schedulers in kubernetes cluster.

The scheduler identify the right node to play a specific container on based on the container resource requirements, the worker node capacities or any other policies or constraints such as node affinity rules

There are different offices that are assigned to special tasks or departments, for example the operations team take care of the ships handling, traffic control etc.
they take care of issues due to damages, the rules that different ships state etc.
the cargo team takes care of containers, when containers are damaged or destroyed they mature new containers are made availabe
service offices they take care of IT and communications between different ships

similarly in kubernetes we have controllers that take care of different areas.

the node controllers take care of nodes,they are responsible of

- on-boarding new nodes to the cluster,
- handling situations where nodes bacome unavailable or are get sketcy destroyed

and the replication controllers ensure that the desired number of containers are running at all times in the replication group

In this architecture we have:
different components,
different offices,
different ships,
the data store
the cranes

but how do this communicate with each other?
How does one office reach another office and who manages the all at the high level?

the kube-apiserver is responsible for orchestrating all operations within the cluster.

it Exposes the kubernates api
which is used

- by external users to perform management operations on the cluster
- by the various controllers to monitor the state of the cluster and make necessary changes as required and
- by the worker nodes to communicate to the servers.

they are working with containers here.

Containers are everywhere so we need everything to be container compatible. Our applications are in the form of containers.

the different components that form the entire management system on the master node could be hosted in the form of containers.

The DNS service, networking solutions can all be deployed in the form of containers, so we need the software that can run containers and that's the container runtime engine a popular one being docker.

So we need docker installed in all the nodes in the cluster including the master node If you wish to host the controlling components as container.

It does not always to be docker,
Kubernetes supports other runtime engine as well like containerD or rocket.

let now us to turn the focus to cargo ships. Every ship has a captain. the captain is responsible for managing all activities on the ship.

The captain is responsible for liaising with the master ship

- starting with letting the master ship know that they are interested joining the group,
- receiving information about the containers to be loaded on the ship,
- loading the appropriate containers as required
- sending report back to the master about the status of this ship and the status of the containers on the ship

The capitain of the ship is the cubelet in kubernetes.

A cubelet is an agent that runs on each node on a cluster.

It listens for instructions from the cube api server and deploys or desroys containers on the nodes as required.

The cube-api server periodically fetches status reports from the cubelet to monitor the status of nodes and containers on them.

the applications running on the worker node need to be able to communicate with each other.

for example you might have a web server running in one container on one of the nodes and a database server running on another container on another node. how would the web server reach the database server on the other node?

Communications between worker nodes are enabled by another component that runs on the worker nodes known as the kube-proxy service.

The kube-proxy service ensures that the necessary rules are in place on the worker nodes to allow the containers running on them to reach each others.

TO SUMMARIZE:

- MASTER NODES
- WORKER NODES

ON THE MASTER:
ETCD WHICH STORES INFORMATION ABOUT THE CLUSTER
CUBE SCHEDULER THAT IS RESPONSIBLE FOR SCHEDULING APPLICATIONS OR CONTAINERS ON NODES
DIFFERENT CONTROLLERS THAT TAKE CARE OF DIFFERENT FUNCTIONS LIKE THE NODE CONTROLLER REPLICATION ETC.
CUBE-API-SERVER THAT IS RESPONSIBLE FOR ORCHESTRATING ALL OPERATIONS WITHIN THE CLUSTER

ON THE WORKER NODE:
THE CUBELET THAT LISTENS FOR INSTRUCTIONS FROM THE CUBE-API-SERVER AND MANAGES CONTAINERS
THE CUBE-PROXY THAT HELPS ENABLING COMMUNICATION BETWEEN THE SERVICES WITHIN THE CLUSTER

Kubernetes components in a picture