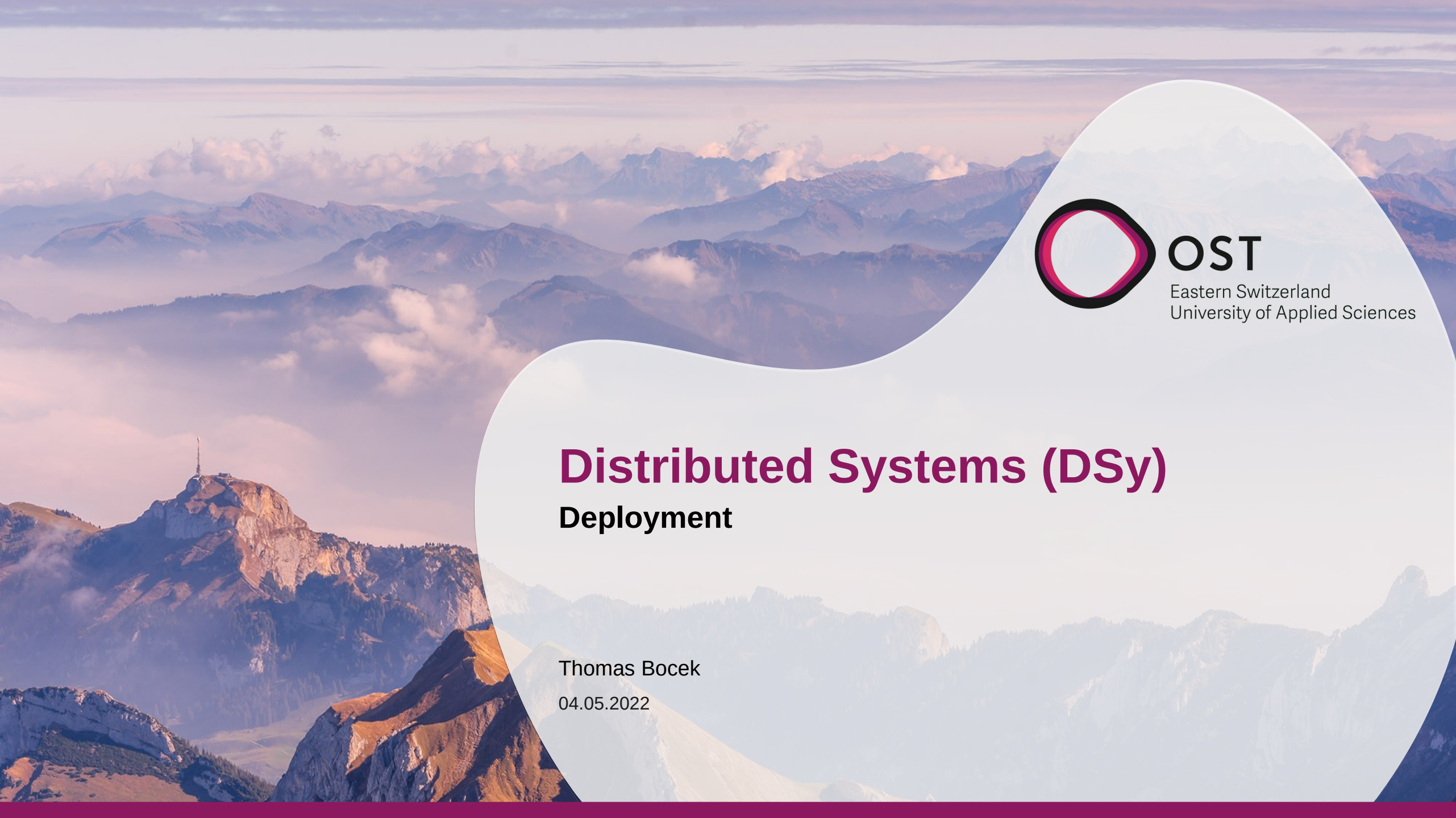


OST
Eastern Switzerland
University of Applied Sciences

Distributed Systems (DSy)

Deployment

Thomas Bocek

04.05.2022

Learning Goals

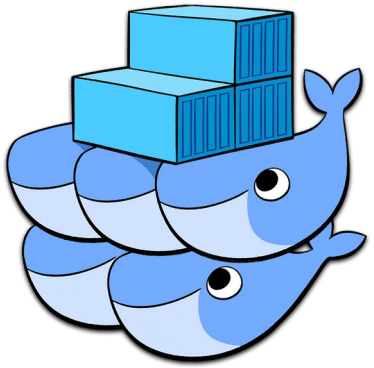
- Lecture 9 (Deployment)
  - Different ways to deploy your service
  - Cloud Operations [link]

Back in the old days…

- OTS: apt-get / yum / pacman install package, e.g., Apache – configure – run
- Custom SW: Java: war, provide custom /etc/init.d script with binary or script
- Problem:
  - It runs on my machine, who installs Java in the right version?
  - What happens on crashes?
  - Scaling?
  - HW defect?
  - Misconfiguration - access to complete PC?
- VMs / Containers help a lot
  - No access to complete PC, can scale, move to another machine, pre-install the right Java version
- So, how to deploy your app?
  - Ansible (Progress Chef, Puppet) - and more
    - Playbooks with ssh host list – your host needs to run the same OS (apt/yum)
  - Docker Swarm
    - Works with docker-compose.yml – with docker you package your application the same way on any platform
  - Kubernetes
    - Widespread

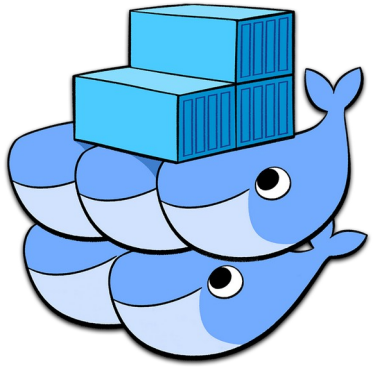
Docker Swarm

- Use docker --context to run/maintain containers on other machines
  - Does not work for docker-compose, could be used with Ansible… “Ansible is also great for bootstrapping Docker itself” [source]
- Docker Swarm
  - Deploy with docker-compose.yml (deploy:)
  - Built into docker
    - docker swarm – manage swarm
    - docker node – manage nodes
  - Scheduler is responsible for placement of containers to nodes
  - Can use the same files, easy to setup?
    - Azure, Google cloud, Amazon



- Kubernetes vs. Docker Swarm
- “Docker Swarm has already lost the battle against Kubernetes for supremacy in the container orchestration space” [link]
- “Kubernetes supports higher demands with more complexity while Docker Swarm offers a simple solution that is quick to get started with.” [link]

Docker Swarm



- 3 “Machines”
  - KVM instances, alpine running
    - Workers: 192.168.1.238, 192.168.1.103, 192.168.1.173
    - Manager: 192.168.1.166
- Run on manager
  - docker swarm init --advertise-addr 192.168.1.166
- To add a worker to this swarm, run the following command:
  - docker swarm join --token .. 192.168.1.166
  - docker info
  - docker node ls
- Manager are setup
- Join nodes
  - Run the docker swarm join command
  - docker node ls
- Workers are setup

Docker Swarm

- Create service
  - docker service create --name registry --publish 5000:5000 registry:2
  - Where to find the docker image
- Check service
  - docker service ls
- Many options in docker-compose
  - docker stack deploy --compose-file docker-compose.yml

```
worker:
  image: gaiadocker/example-voting-app-worker:latest
  networks:
    voteapp:
      aliases:
        - workers
  depends_on:
    - db
    - redis
  # service deployment
  deploy:
    mode: replicated
    replicas: 2
    labels: [APP=VOTING]
    # service resource management
    resources:
      # Hard limit - Docker does not allow to allocate more
      limits:
        cpus: '0.25'
        memory: 512M
      # Soft limit - Docker makes best effort to return to it
      reservations:
        cpus: '0.25'
        memory: 256M
    # service restart policy
    restart_policy:
      condition: on-failure
      delay: 5s
      max_attempts: 3
      window: 120s
    # service update configuration
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: continue
      monitor: 60s
      max_failure_ratio: 0.3
    # placement constraint - in this case on 'worker' nodes only
    placement:
      constraints: [node.role == worker]
```


OST

Docker Swarm

- Moved from docker-compose to swarm on digital ocean for THORWallet

  + It works, it was fast

  - Annoying limitations: show real IPs in our load balancer [issue] - probably there are more...

- Logtail.com for collecting logfiles from the containers – vector.dev to to send it from our containers to logtail
  - Logs are important for failure analysis, statistics. The more services, the more is aggregation important

```
2022-05-03 14:39:43.698 [digital_ocean_docker] [INFO] 200 10.0.0.2 - - [03/May/2022:12:39:37 +0000] "GET /v2/history/tvl?interval=hour&count=168 HTTP/2.0" 200 25208
2022-05-03 14:39:43.698 [digital_ocean_docker] [INFO] 200 10.0.0.2 - - [03/May/2022:12:39:35 +0000] "GET /v2/history/depths/BNB.AVA-645?interval=hour&count=168 HTTP/2.0" 200 63629
2022-05-03 14:39:43.698 [digital_ocean_docker] 2022-05-03T12:39:37Z INF Access duration_ms=1210.841177 ip=10.0.0.2 method=GET module=http req_id=c9oi5266nfomcajkq02g size=25208 status=200 url=/v2/history/tvl?interval=hour&count=168 user_agent=okhttp/4.9.1
2022-05-03 14:39:43.698 [digital_ocean_docker] [INFO] 302 10.0.0.2 - - [03/May/2022:12:39:35 +0000] "GET /v2/thorchain/inbound_addresses HTTP/2.0" 302 138
2022-05-03 14:39:43.698 [digital_ocean_docker] 2022-05-03T12:39:35Z INF Access duration_ms=351.714459 ip=10.0.0.2 method=GET module=http req_id=c9oi51qvl63ie64cn290 size=63111 status=200 url=/v2/history/depths/ETH.AAVE-0X7FC66500C84A76AD7E9C93437BFC5AC33E2DDAE9?interval=hour&count=168 user_agent=okhttp/4.9.1
2022-05-03 14:39:43.698 [digital_ocean_docker] [INFO] 200 10.0.0.2 - - [03/May/2022:12:39:34 +0000] "GET /v2/history/depths/ETH.YFI-0X0BC529C00C6401AEF6D220BE8C6EA1667F6AD93E?interval=hour&count=168 HTTP/2.0" 200 63733
```


OST

Kubernetes

- Kubernetes, K8s
  - Container orchestration (docker)
    - Automated deployment, scaling
  - Started by Google, now with CNCF
- Kubernetes-based PaaS
  - Google, Amazon, Azure (book), Digital Ocean, …
    - Difficult pricing schemes
  - 1.0 released in 2015
  - Package manager Helm released in 2016 (convert docker-compose)
- Why Kubernetes?
  - Containers can crash, machine that runs container can crash (e.g., out of memory)
  - Development: run on one machine, deployment how and where to distribute?
  - Kubernetes manages the lifecycle of containers

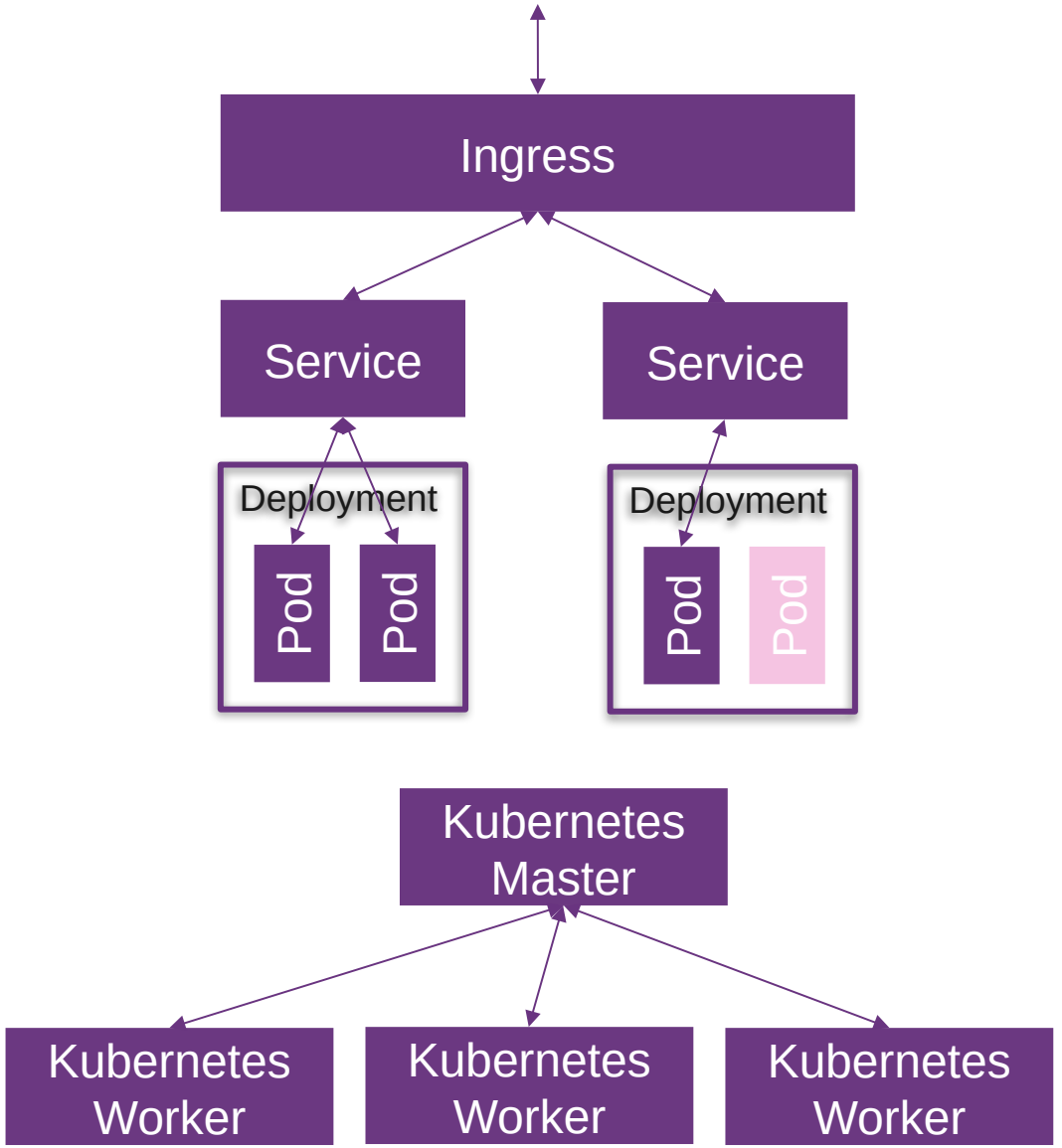
Kubernetes

- Design principles
  - Configuration is declarative – declare state with YAML/JSON
    - “self-healing”
  - Abstraction layer for distributed system
    - Provides interface to interact with containers
  - Immutable containers
    - Don’t store state in a container. If a health check fails, Kubernetes removes the container and starts a new one
    - Rollback applications, use older version of container
      - SQL – may need to change schema

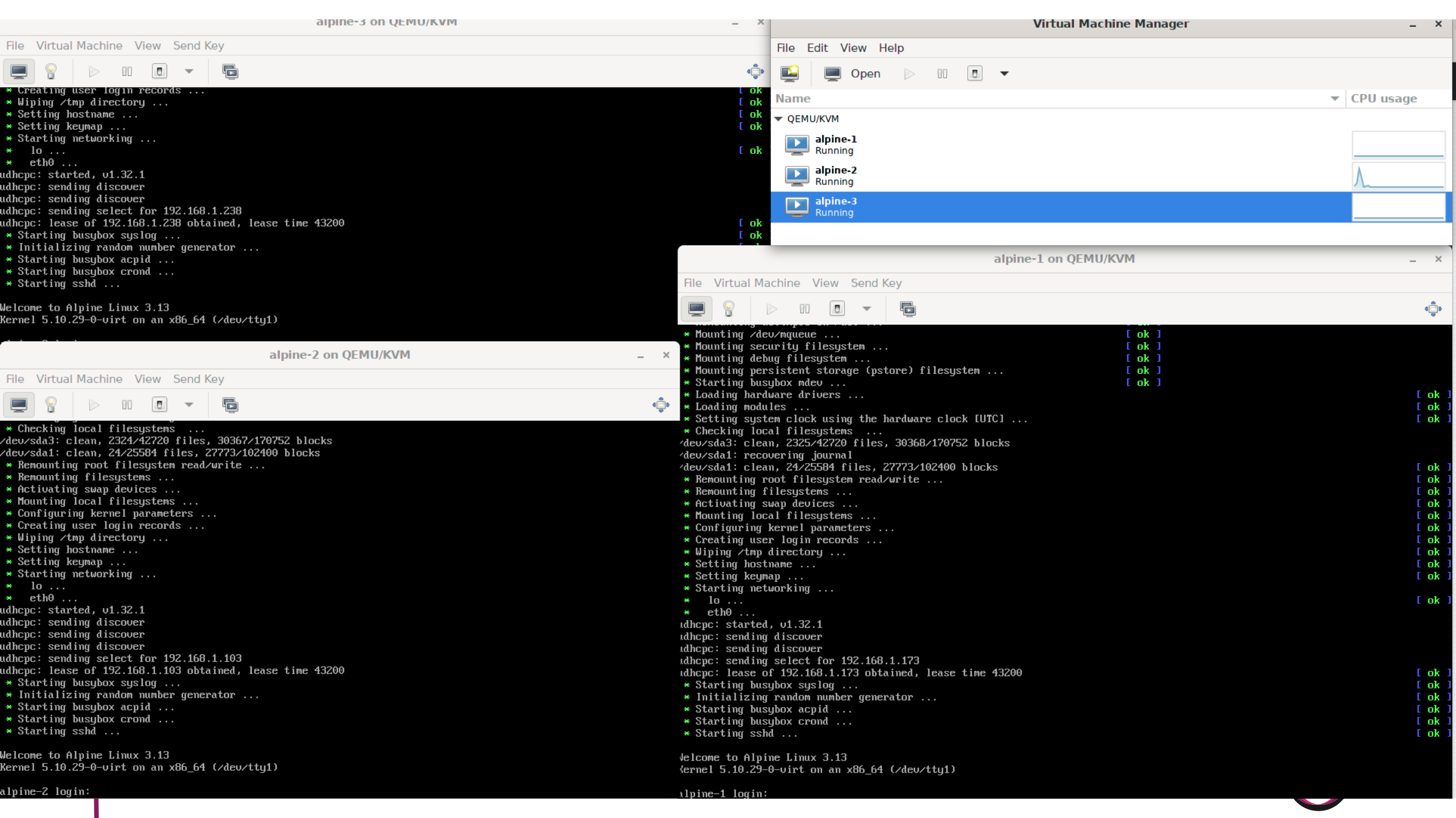
- Pod – one (or more close connected) container (long running)
  - Job – short running
  - Volume - directory accessible to all containers running in a Pod
- Deployment – define scale, HW limits
- Service – singe entry point (internal), define a set of Pods
- Ingress – expose end points / external access
- Namespaces – run multiple projects on one cluster, separate with namespaces

Kubernetes

- Minikube, k3s
  - Kubernetes master / server / control plane
  - Kubernetes worker / nodes / agent / compute machine
- Deploy any containerized application
  - Better use health endpoints
    - Liveness/Readiness
- Youtube course



OST


alpine-3 on QEMU/KVM
File Virtual Machine View Send Key
* Creating user login records ... [ok
* Wiping /tmp directory ... [ok
* Setting hostname ... [ok
* Setting keymap ... [ok
* Starting networking ... [ok
* lo ...
* eth0 ...
udhcpc: started, v1.32.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.1.238
udhcpc: lease of 192.168.1.238 obtained, lease time 43200 [ok
* Starting busybox syslog ... [ok
* Initializing random number generator ...
* Starting busybox acpid ...
* Starting busybox crond ...
* Starting sshd ...
Welcome to Alpine Linux 3.13
Kernel 5.10.29-0-virt on an x86_64 (/dev/tty1)
Virtual Machine Manager
File Edit View Help
Open
Name
CPU usage
QEMU/KVM
alpine-1
Running
alpine-2
Running
alpine-3
Running
alpine-2 on QEMU/KVM
File Virtual Machine View Send Key
* Checking local filesystems ...
/dev/sda3: clean, 2324/42720 files, 30367/170752 blocks
/dev/sda1: clean, 24/25584 files, 27773/102400 blocks
* Remounting root filesystem read/write ...
* Remounting filesystems ...
* Activating swap devices ...
* Mounting local filesystems ...
* Configuring kernel parameters ...
* Creating user login records ...
* Wiping /tmp directory ...
* Setting hostname ...
* Setting keymap ...
* Starting networking ...
* lo ...
* eth0 ...
udhcpc: started, v1.32.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.1.103
udhcpc: lease of 192.168.1.103 obtained, lease time 43200
* Starting busybox syslog ...
* Initializing random number generator ...
* Starting busybox acpid ...
* Starting busybox crond ...
* Starting sshd ...
Welcome to Alpine Linux 3.13
Kernel 5.10.29-0-virt on an x86_64 (/dev/tty1)
alpine-2 login:
alpine-1 on QEMU/KVM
File Virtual Machine View Send Key
* Mounting /dev/mqueue ... [ok]
* Mounting security filesystem ... [ok]
* Mounting debug filesystem ... [ok]
* Mounting persistent storage (pstore) filesystem ... [ok]
* Starting busybox mdev ... [ok]
* Loading hardware drivers ... [ok]
* Loading modules ... [ok]
* Setting system clock using the hardware clock [UTC] ... [ok]
* Checking local filesystems ...
/dev/sda3: clean, 2325/42720 files, 30368/170752 blocks
/dev/sda1: recovering journal
/dev/sda1: clean, 24/25584 files, 27773/102400 blocks [ok]
* Remounting root filesystem read/write ... [ok]
* Remounting filesystems ... [ok]
* Activating swap devices ... [ok]
* Mounting local filesystems ... [ok]
* Configuring kernel parameters ... [ok]
* Creating user login records ... [ok]
* Wiping /tmp directory ... [ok]
* Setting hostname ... [ok]
* Setting keymap ... [ok]
* Starting networking ...
* lo ... [ok]
* eth0 ...
udhcpc: started, v1.32.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.1.173
udhcpc: lease of 192.168.1.173 obtained, lease time 43200 [ok]
* Starting busybox syslog ... [ok]
* Initializing random number generator ... [ok]
* Starting busybox acpid ... [ok]
* Starting busybox crond ... [ok]
* Starting sshd ... [ok]
Welcome to Alpine Linux 3.13
Kernel 5.10.29-0-virt on an x86_64 (/dev/tty1)
alpine-1 login: