On-premise data centers do not need to be legacy

Fabio Alessandro "Fale" Locati
EMEA Associate Principal Specialist Solutions Architect

Red Hat

TOC

- A little bit of history
- What is cloud
- Lessons we can learn
- Technologies considerations and bets
- Conclusions
- Q&A

About me

- GNU/Linux user since 2001
- Working with GNU/Linux since 2004
- Currently working for Red Hat

A little bit of history

Very brief "cloud" history

- 1998 - Rackspace is founded
- 2005 - SoftLayer is founded
- 2006 - AWS launches Simple Storage Service (S3)
- 2006 - AWS launches Elastic Compute Cloud (EC2)
- 2008 - Google launches Google App Engine
- 2021 - AWS has over 200 services

Very brief non-"cloud" history

- 1964 - IBM introduces the CP-40, the first mainframe with time-sharing technology
- late 1960s - IBM releases SIMMON, the first hypervisor
- 1974 - Gerald Popek and Robert Goldberg classify the hypervisors into two types:
  - Type 1: bare-metal virtualization
  - Type 2: hosted [on top of the host Operating System] virtualization
- 1998 - VMware founded
- 2001 - VMware releases ESX 1.0 Server
- 2003 - Xen first release
- 2003 - VMware releases Virtual Center 1.0 with vMotion
- 2008 - Microsoft releases Hyper-V

What is cloud

What is cloud?

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. (Wikipedia) ~~Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. (Wikipedia)~~
A business model where one party rents to a second party computer system resources, especially data storage (cloud storage) and computing power, with the smallest granularity possible.

- Time: month -> hour -> minute -> second -> millisecond
- Compute: CPU -> Core -> vCore -> fractional vCPU

Lessons we can learn

Separation of concerns

- Standardize the interface between infrastructure and workload
- Scalability at workload level
- Workload have an abstract concept of the physical architecture

Functional business model

- Standardize the interface between infrastructure and workload
- Bill back infrastructure costs to the workloads owners
- Keep the costs down

Maintain control

- Do not use third-party proprietary software
- Evaluate buy vs build decisions preferring the latter
- Be aware of lock-ins
  Product between the **probability** that a component will require substitution during the solution life and the **total costs** in case of substitution.

Technologies considerations and bets

KISS

- Reduce the complexity of your system to a minimum
- Prefer build-time complexity over run-time complexity
- Minimize the amount of services available

Containers

- Use a Kubernetes distribution
  - DIY/Community
  - Commercial
    - Fully open source
    - Trustworthy company
    - Long track record
    - Heavily involved in upstream development

Automation

- Use an immutable approach to infrastructure
- Version the infrastructure (eg: gitops)
- Automate the whole process

Conclusions

Putting all together

- Infrastructure
- API
- Workloads

Putting it all together - Infrastructure

- Create/Architect for multiple DataCenters (and multiple clusters) but hide them from the workload developer
- Deploy Kubernetes container platform clusters on bare-metal
- Use a tool to manage and abstract the clusters (eg: Open Cluster Management)
- Automate all the infrastracture pieces and configuration

Putting it all together - API

- Define discrete ”regions” based on legal frameworks (eg: eu, us)
- Standardize the Kubernetes APIs as the only interfaces between infrastructure and workload
- Start providing only: OCI registry, Object Storage, and a very limited subset of Kubernetes objects (eg: Pods, Deployments, Stateful SetsServices, PV, PVC, ConfigMaps, Secrets)
- Provide more services once you have a good strategy to support them and many of your users are already using the technology (eg: Databases)

Putting it all together - Workloads

- Create a simple UX to submit the creation/update/deletion of workloads objects
- Store workloads objects in a versioned storage (eg: git) and automate deployment
- Require (opt-out?) applications resilient to restarts, replications, etc.

Q&A

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat