

SMC: An Atos solution for HPC infrastructure

18 January 2021

Laboratório Nacional de Computação Científica (LNCC)

Presented by: Giacomo Mc Evoy Valenzano <giacomo.valenzano@atos.net>

Contents

- 1 Introduction
- 2 The foundation:
Ansible and BlueBanquise
- 3 SMC: Extending
BlueBanquise
- 4 Beyond SMC: xScale
- 5 Live demo:
BlueBanquise cluster
- 6 Final Remarks

Introduction

Escola Supercomputador SDumont - LNCC

Introduction – The idea behind SMC

Smart Management Center

- ▶ Cluster management stack, based on Ansible aimed at deploying and managing HPC clusters.
- ▶ Main features
 - Maximize efficiency of administrative tasks: be useful for 10 nodes and for 1k nodes
 - Easy to extend regardless of cluster size up to 1200 nodes
 - Use open-source libraries, comply with RHEL recommendations
 - Configuration/Infrastructure as code, no DB
 - Optional High Availability setup

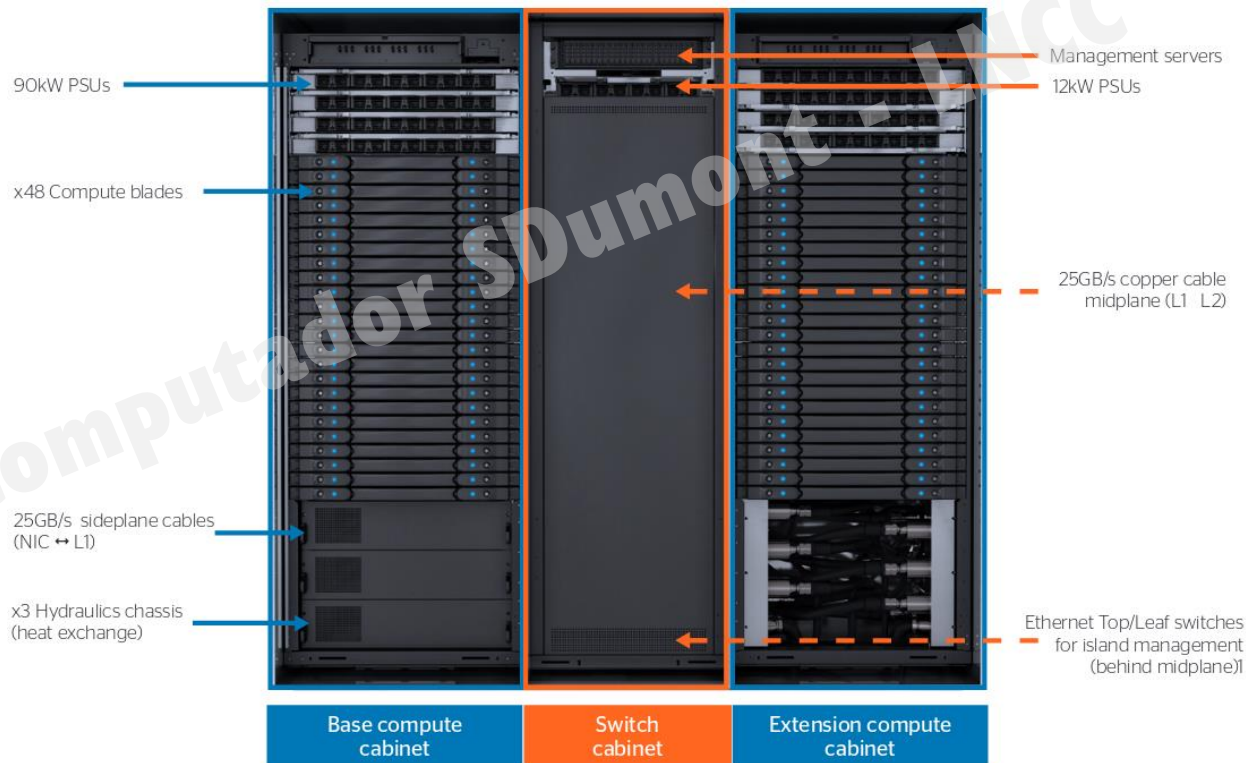
 Smart Management Center



Motivation: bring (and keep) HPC systems online

How to configure a complex cluster?

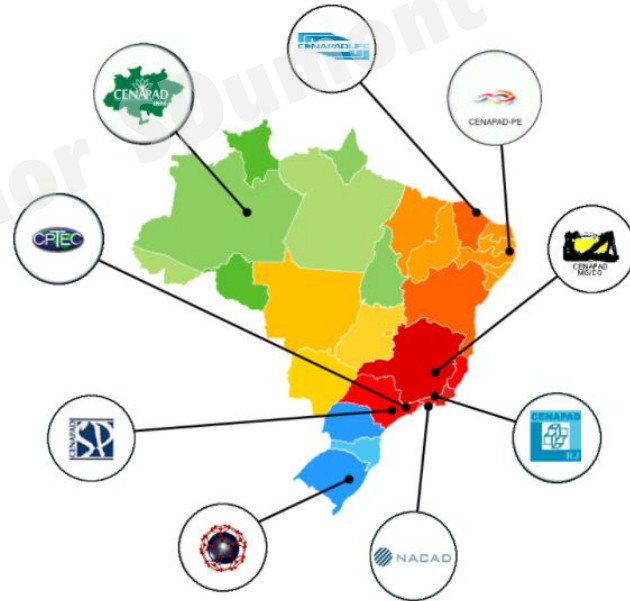
- ▶ Santos Dumont: expanded with BullSequana X1000
 - Management nodes
 - Compute nodes
 - Power management
- ▶ Need to configure:
 - Management nodes
 - Compute nodes
 - Power management
- ▶ Complex software:
 - Networking
 - Performance libs
 - User/job management
 - Monitoring



What about small/medium clusters?

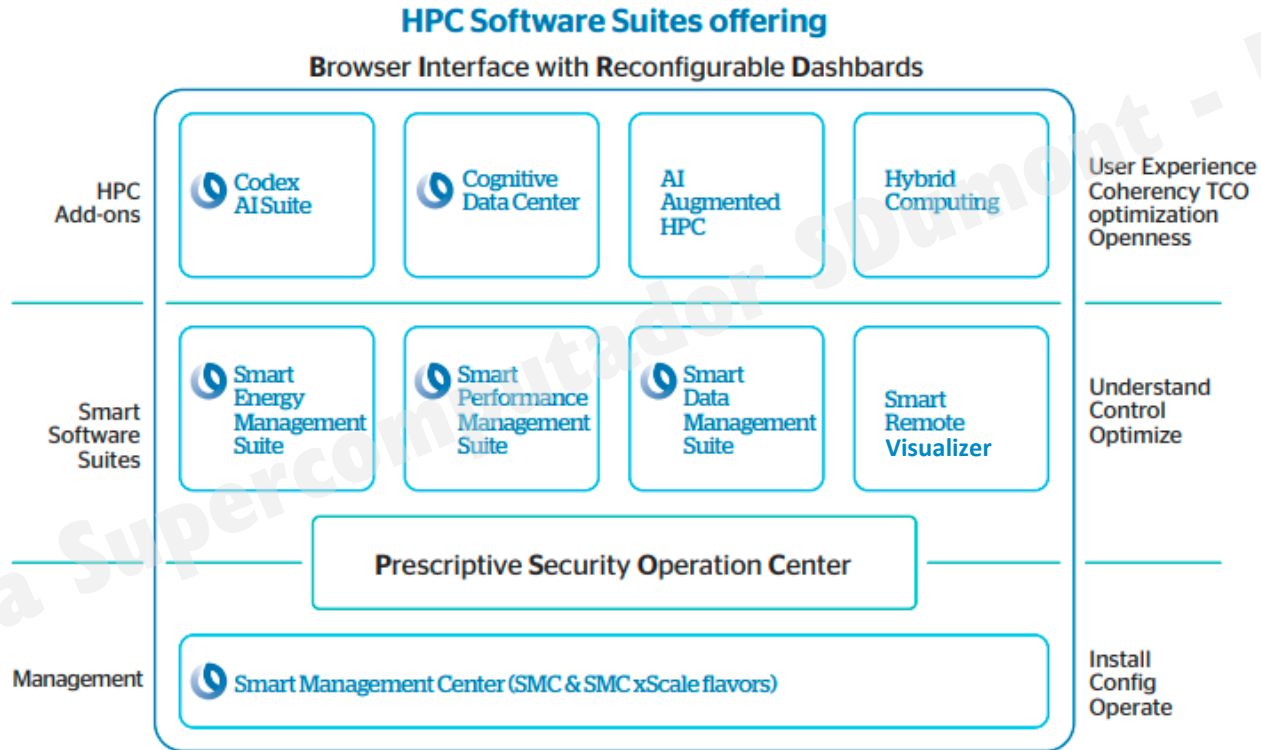
Not only about installing a single large cluster

- ▶ Do you work on system administration for the CPD or these research groups?



A path towards exascale

Scalable, flexible, efficient and secure while still optimizing performance

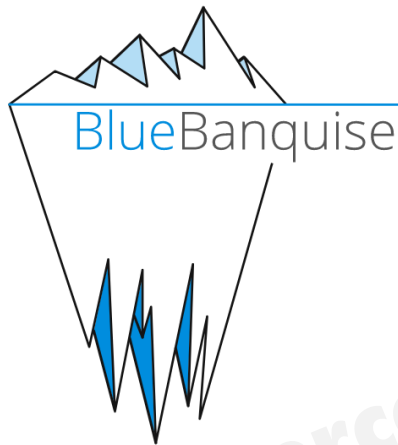


The foundation: Ansible and BlueBanquise

Escola Supercomputador SDumont - LNCC

What is BlueBanquise?

The open-source basis for SMC



- ▶ “Generic stack, based on Ansible, whose purpose is to deploy and manage clusters of hosts.”
- ▶ Coherent collection of Ansible roles, with working configurations provided as examples
- ▶ Open-source with MIT license

Ansible in a nutshell

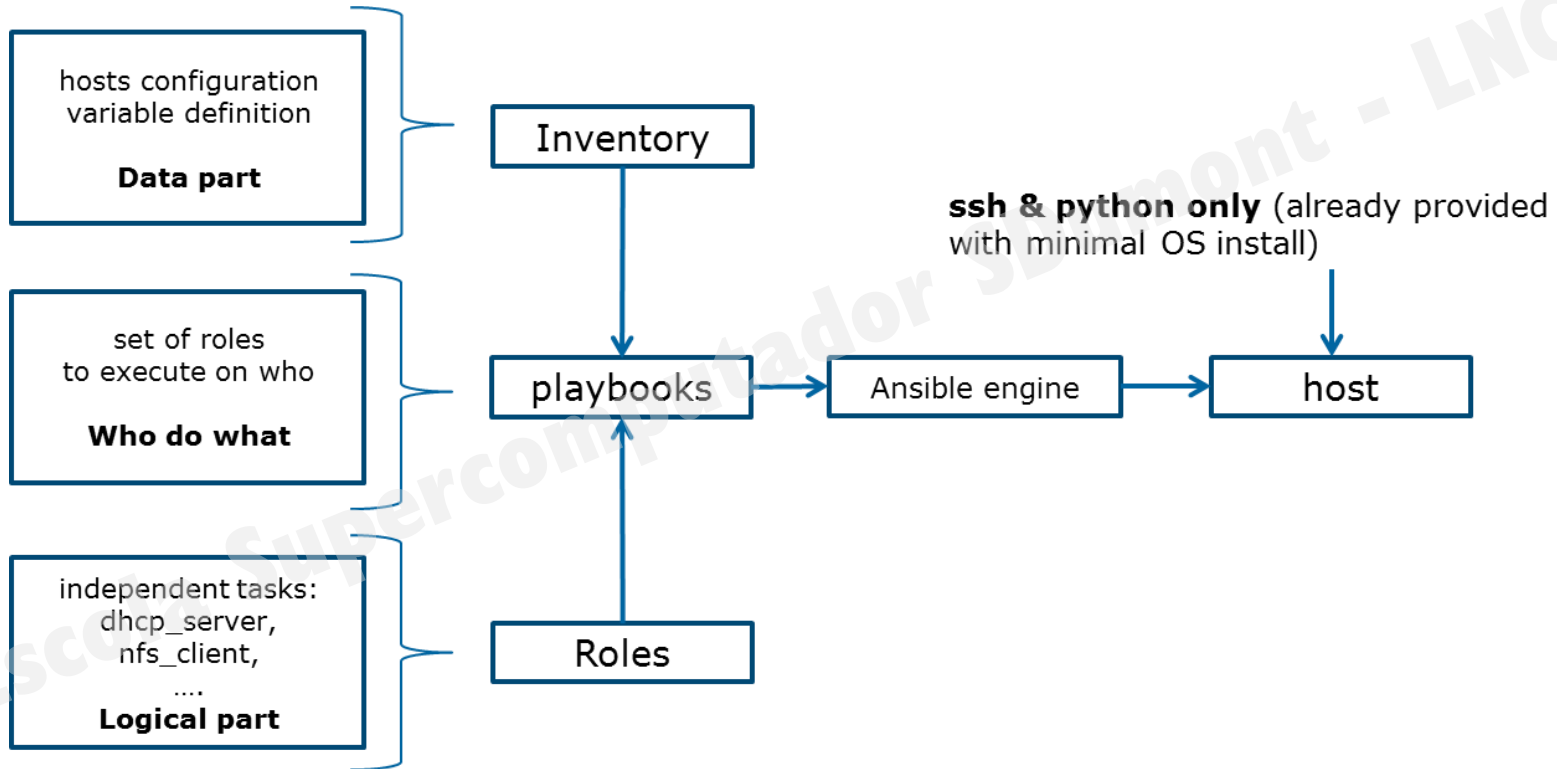
Automate All the Things

- ▶ Wikipedia:
“Ansible is an open-source software provisioning, configuration management, and application-deployment tool...”
- ▶ Key ideas
 - Describe the **desired state**, not the process
 - Decompose configuration in “roles” and “tasks”
 - Machines can be “grouped”
 - Agentless: nodes only need SSH



Ansible rendering process

Use playbooks to map tasks to hosts



How do we tell Ansible what to do?

Use YAML files to declare configuration

► Create inventory

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          network_interfaces:
            - interface: enp0s9
              ip4: 10.10.3.1
              mac: 08:00:27:0d:44:90
              network: ice1-1
            - interface: ib0
              ip4: 10.20.3.1
              network: interconnect-1
```

► Assign roles

```
- name: Computes play
  hosts: "mg_computes"

  roles:
    - role: set_hostname
      tags: set_hostname
    - role: repositories_client
      tags: repositories_client
    - role: nic
      tags: nic
    - role: hosts_file
      tags: hosts_file
    - role: dns_client
      tags: dns_client
    - role: time
      tags: time
  vars:
    time_profile: client
```

How do we tell Ansible what to do?

Use YAML files to declare configuration

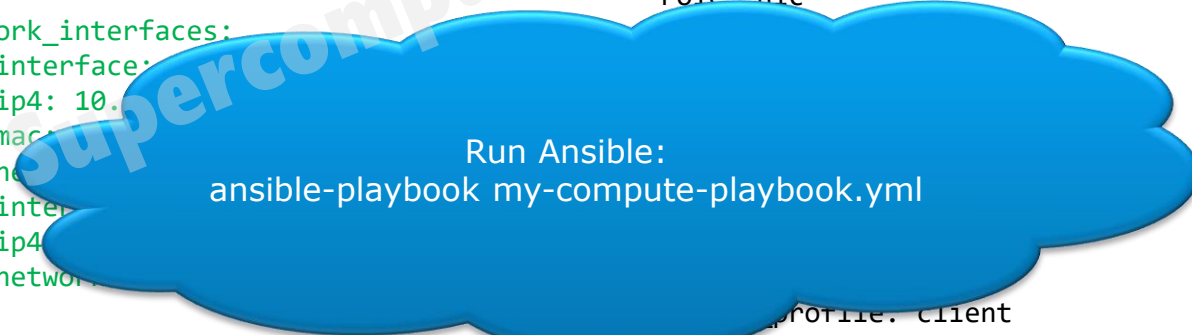
► Create inventory

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          network_interfaces:
            - interface:
              ip4: 10.
              mac:
              ne
            - inter
              ip4
              network
```

► Assign roles

```
- name: Computes play
  hosts: "mg_computes"

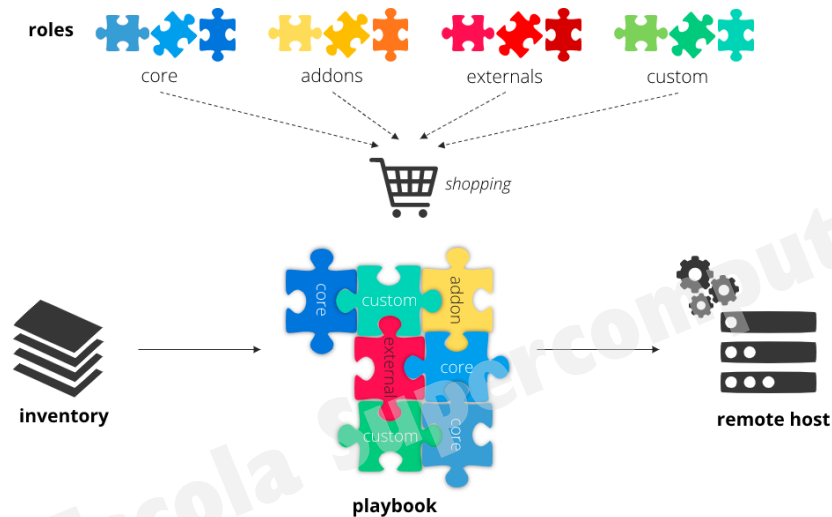
  roles:
    - role: set_hostname
      tags: set_hostname
    - role: repositories_client
      tags: repositories_client
    - role: nic
```



Run Ansible:
ansible-playbook my-compute-playbook.yml

BlueBanquise components

Stack is fully modular



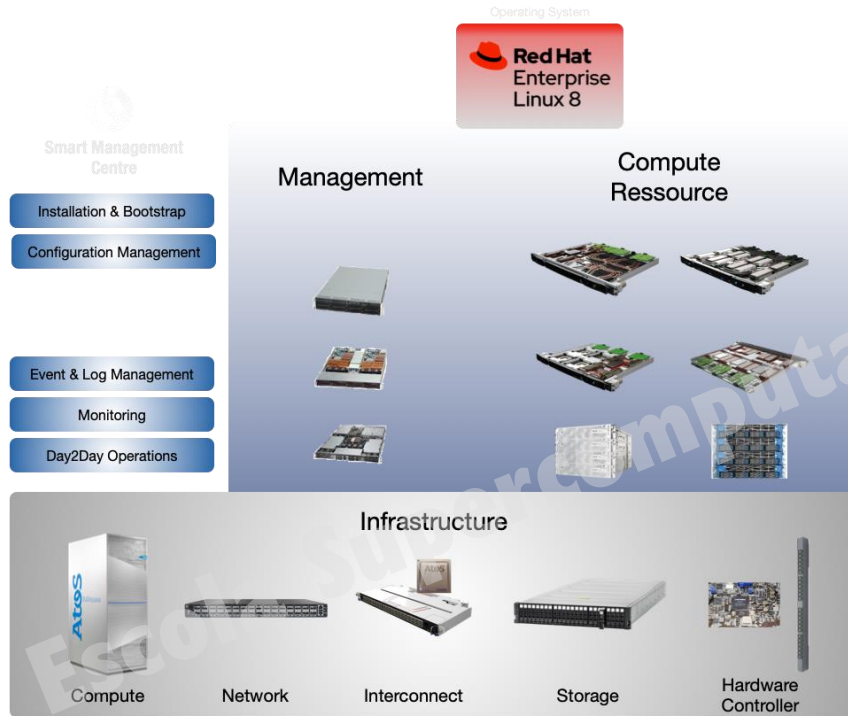
- ▶ Only python3 and Ansible
- ▶ Core components
 - Deploy: PXE, kickstart
 - Manage: DNS, IPMI
 - Security: firewall, SSH
- ▶ Support
 - Focus on RHEL
 - x86_64 and arm64

SMC: Extending BlueBanquise

Escola Supercomputador SDumont - LNCC

Why do we need SMC?

Specific needs for BullSequana

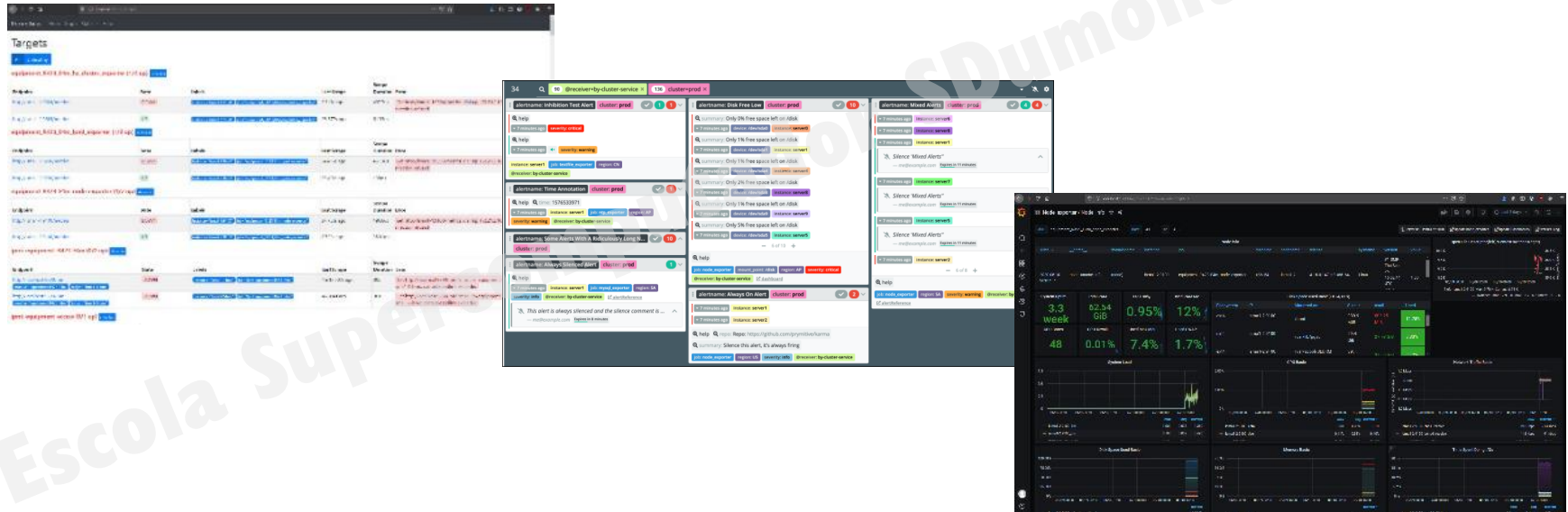


- ▶ Full integration with BullSequana XH2000 servers
- ▶ High Availability with pacemaker
- ▶ Additional roles
 - Monitoring and alerting with Prometheus and Grafana
 - BMC configuration
 - Power management
 - and more...

Cluster monitoring with SMC

Detect anomalies and reduce down time

- ▶ Monitor nodes with Prometheus
- ▶ Alert management with Karma
- ▶ Dedicated dashboards with Grafana



Showcasing SMC by example

Automagically configure new compute nodes

- ▶ Scenario: after physically installing a new diskless node, how much effort until we can monitor power consumption during a job submission?
- ▶ The steps:
 1. Add node to inventory
 2. Run management playbook
 3. Prepare diskless image
 4. Boot diskless node remotely
 5. Run compute playbook

* Some caveats mentioned later

SMC by example: 1. Add node to inventory

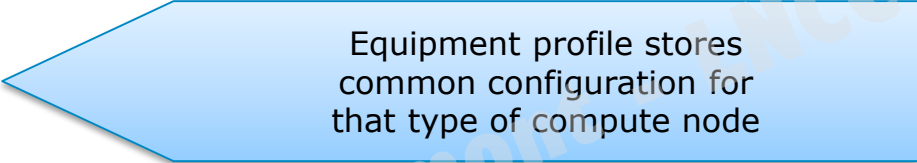
Edit /etc/bluebanquise/inventory/cluster/nodes/computes.yml

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          [...]
        c010:
          bmc:
            name: bc010
            ip4: 10.10.103.10
            mac: 08:00:27:dc:44:91
            network: ice1-1
          network_interfaces:
            - interface: enp0s9
              ip4: 10.10.3.10
              mac: 08:00:27:0d:44:90
              network: ice1-1
            - interface: ib0
              ip4: 10.20.3.10
              network: interconnect-1
```

SMC by example: 1. Add node to inventory

Edit /etc/bluebanquise/inventory/cluster/nodes/computes.yml

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          [...]
        c010:
          bmc:
            name: bc010
            ip4: 10.10.103.10
            mac: 08:00:27:dc:44:91
            network: ice1-1
          network_interfaces:
            - interface: enp0s9
              ip4: 10.10.3.10
              mac: 08:00:27:0d:44:90
              network: ice1-1
            - interface: ib0
              ip4: 10.20.3.10
              network: interconnect-1
```



Equipment profile stores common configuration for that type of compute node

SMC by example: 1. Add node to inventory

Edit /etc/bluebanquise/inventory/cluster/nodes/computes.yml

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          [...]
        c010:
          bmc:
            name: bc010
            ip4: 10.10.103.10
            mac: 08:00:27:dc:44:91
            network: ice1-1
          network_interfaces:
            - interface: enp0s9
              ip4: 10.10.3.10
              mac: 08:00:27:0d:44:90
              network: ice1-1
            - interface: ib0
              ip4: 10.20.3.10
              network: interconnect-1
```

Name in inventory will be used as hostname, should use numeric suffix

SMC by example: 1. Add node to inventory

Edit /etc/bluebanquise/inventory/cluster/nodes/computes.yml

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          [...]
        c010:
          bmc:
            name: bc010
            ip4: 10.10.103.10
            mac: 08:00:27:dc:44:91
            network: ice1-1
          network_interfaces:
            - interface: enp0s9
              ip4: 10.10.3.10
              mac: 08:00:27:0d:44:90
              network: ice1-1
            - interface: ib0
              ip4: 10.20.3.10
              network: interconnect-1
```

BMC can be configured
using its MAC address and DHCP

SMC by example: 1. Add node to inventory

Edit /etc/bluebanquise/inventory/cluster/nodes/computes.yml

```
mg_computes:
  children:
    equipment_typeC:
      hosts:
        c001:
          [...]
        c010:
          bmc:
            name: bc010
            ip4: 10.10.103.10
            mac: 08:00:27:dc:44:91
            network: ice1-1
          network_interfaces:
            - interface: enp0s9
              ip4: 10.10.3.10
              mac: 08:00:27:0d:44:90
              network: ice1-1
            - interface: ib0
              ip4: 10.20.3.10
              network: interconnect-1
```

Node is characterized by its networking,
if we need more details then use another
equipment profile!

SMC by example: 2. Run management playbook

It should not break anything!

- ▶ Actual command:

```
cd /etc/bluebanquise; ansible-playbook playbooks/managements.yml
```

- ▶ What SMC will do for you
 - DHCP: register MAC/IP addresses of node and BMC
 - PXE: ensure node retrieves correct boot script (UEFI, iPXE)
 - SLURM: add node to compute partition
 - Prometheus: update configuration to include node and IPMI exporters

SMC by example: 3. Prepare diskless image

Image is prepared once, then reused

- ▶ Actual commands (first will interactively ask how to create an image):

```
/usr/bin/disklessset  
/usr/bin/bootset -n c010 -b diskless -i my_livenet_image
```

- ▶ What SMC will do for you
 - Prepare kernel/ramdisk combination based on user choice
 - Prepare diskless image based on user choice (**livenet**/NFS)
 - Customize diskless image (configure SELinux, run custom playbook...)
 - Configure PXE to ensure node will boot prepared diskless image

SMC by example: 4. Boot diskless node remotely

Leverage accessible BMC

- ▶ Actual command:

```
ipmitool -I lanplus -H bc010 -U admin -P pass chassis power on
```

- ▶ This is possible because:
 - BMC is connected to the network via DHCP
 - The management node knows about bc010 hostname
 - PXE server is ready to serve diskless image to node

Showcasing SMC by example (cont.)

5. Run compute playbook

▶ Actual command

```
cd /etc/bluebanquise; ansible-playbook playbooks/computes.yml -l c010
```

▶ What SMC will do for you

- Configure basic software stack (networks, SSH, etc)
- SLURM: Start slurmd service and connect securely to controller
- Prometheus: configure alerts and start monitoring service

Everything should be ready!

Can repeat for 100s of nodes



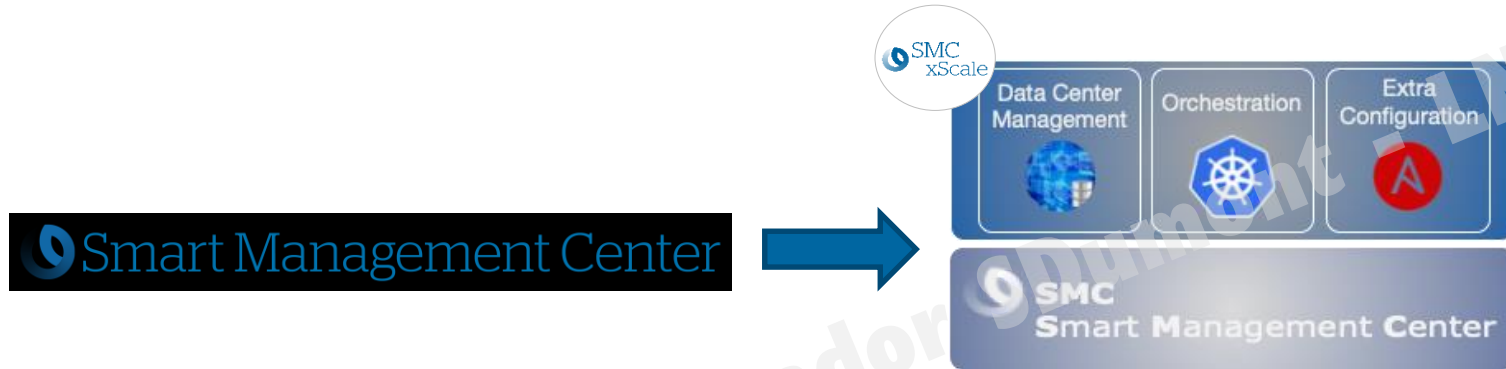
- ▶ What we were able to do:
 - Install and configure a new compute diskless node
 - Job submission by user
 - Monitoring by sysadmin
- ▶ Some caveats:
 - We ignored security mechanisms
 - Few additional commands may be needed, e.g. restart DHCP and reconfigure SLURM

Beyond SMC: xScale

Escola Supercomputador SDumont - LNCC

Smart Management Center xScale

Atos next-gen HPC Management Stack



- Supports wide range of hardware
- Cluster < 1200 nodes
- Centralized management
- Reduce downtime
- Optimized for BullSequana XH2000 platform
- Cluster \geq 1200 nodes (\geq 12 BullSequana XH2000 racks)
- Orchestrated management
- Aim for 0 downtime

Live demo:
BlueBanquise cluster

Escola Supercomputador SDumont - LNCC

Installing BlueBanquise live!

In a virtual cluster of... containers

- ▶ Purpose of demo: to show how we can configure a SLURM cluster from scratch
- ▶ What we start with
 - Very basic RHEL 8.2 Docker image (with systemd support)
 - Repositories: RHEL, BlueBanquise, Ansible from EPEL
 - Ansible playbook with a management play and a compute play
 - An open-source tool called dcluster
- ▶ What we get after only 4 commands
 - Docker cluster with a management node and two compute nodes
 - SLURM fully configured to submit jobs to the compute nodes

Demo Time

Create cluster, let Ansible do the rest

- ▶ Step 1: Create a cluster of containers with 2 compute nodes:

```
dcluster create --profile bluebanquise bbcluster 2
```

- ▶ Step 2: Deploy some ansible playbooks on the cluster (we get inventory for free!)

```
dcluster ansible --cluster bbcluster \  
  dcluster-repo dcluster-ssh dcluster-ansible setup-bb \  
  -e @bluebanquise.json
```

- ▶ Step 3: Login to our management1 container

```
dcluster ssh bbcluster management1
```

- ▶ Step 4: Run a BlueBanquise management+compute playbook

```
cd /etc/bluebanquise; ansible-playbook bbcluster-playbook.yml
```

Explaining dcluster in (very) few minutes

A tool for deploying a cluster of containers and running Ansible on it

Parse a profile to...

call docker-compose

and create inventory

Explaining dcluster in (very) few minutes

A tool for deploying a cluster of containers and running Ansible on it

Parse a profile to...

call docker-compose

and create inventory

```
bluebanquise:
  extend: simple
  compute:
    hostname:
      prefix: c
    image: rhel82:init
    systemctl: true
    static:
      expose:
        - '6818'
  head:
    hostname: management1
    image: rhel82:init
  ...
```

Explaining dcluster in (very) few minutes

A tool for deploying a cluster of containers and running Ansible on it

Parse a profile to...

```
bluebanquise:
  extend: simple
  compute:
    hostname:
      prefix: c
    image: rhel82:init
    systemctl: true
    static:
      expose:
        - '6818'
  head:
    hostname: management1
    image: rhel82:init
    ...
```

call docker-compose

```
services:
  bbcluster-c001:
    container_name: bbcluster-c001
    image: rhel82:init
    init: false
    entrypoint: "/sbin/init"
    cap_add:
      - SYS_ADMIN
    hostname: c001
    networks:
      dcluster-bbcluster:
        ipv4_address: 172.30.0.1
    expose:
      - '6818'
    ...
```

and create inventory

Explaining dcluster in (very) few minutes

A tool for deploying a cluster of containers and running Ansible on it

Parse a profile to...

```
bluebanquise:
  extend: simple
  compute:
    hostname:
      prefix: c
    image: rhel82:init
    systemctl: true
    static:
      expose:
        - '6818'
  head:
    hostname: management1
    image: rhel82:init
    ...
```

call docker-compose

```
services:
  bbcluster-c001:
    container_name: bbcluster-c001
    image: rhel82:init
    init: false
    entrypoint: "/sbin/init"
    cap_add:
      - SYS_ADMIN
    hostname: c001
    networks:
      dcluster-bbcluster:
        ipv4_address: 172.30.0.1
    expose:
      - '6818'
    ...
```

and create inventory

```
all:
  children:
    compute:
      hosts:
        172.30.0.1:
        172.30.0.2:
    head:
      hosts:
        172.30.0.253:
  hosts:
    172.30.0.1:
      container: bbcluster-c001
      hostname: c001
      image: rhel82:init
    ...
  vars:
    cluster_name: bbcluster
    ...
```

Final Remarks

Escola Supercomputador SDumont - LNCC

Summary

- ▶ Configuration and maintenance of HPC clusters can be made easy with SMC.
- ▶ SMC is based on BlueBanquise, an open-source Ansible stack ready to use.
- ▶ Configuration management can be both simple to use and expose a lot of complexity if needed.
- ▶ SMC xScale is the next generation of SMC and is targeted for exascale and hybrid solutions.
- ▶ dcluster open-source tool can be used to test distributed applications.

Additional resources

▶ Learn more about BullSequana and SMC

- <https://atos.net/en/solutions/high-performance-computing-hpc/bullsequana-x-supercomputers>
- https://atos.net/wp-content/uploads/2020/12/HPC_Software_Suites_position_paper.pdf
- https://atos.net/en/2020/press-release_2020_11_19/new-hpc-software-suites

▶ BlueBanquise

- <https://bluebanquise.com>
- <https://github.com/bluebanquise>

▶ dcluster on GitHub

- <https://github.com/ginomcevoy/dcluster>

Thank you

For more information please contact:
Giacomo Mc Evoy Valenzano
giacomo.valenzano@atos.net

Atos, the Atos logo, Atos|Syntel are registered trademarks of the Atos group. January 2021. © 2021 Atos. Confidential information owned by Atos, to be used by the recipient only. This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval from Atos.

The Atos logo is displayed in a bold, blue, sans-serif font. The letter 'o' is stylized with a white dot in the center, creating a circular effect. The background of the slide features a large, light gray circular graphic with a white, teardrop-shaped cutout.