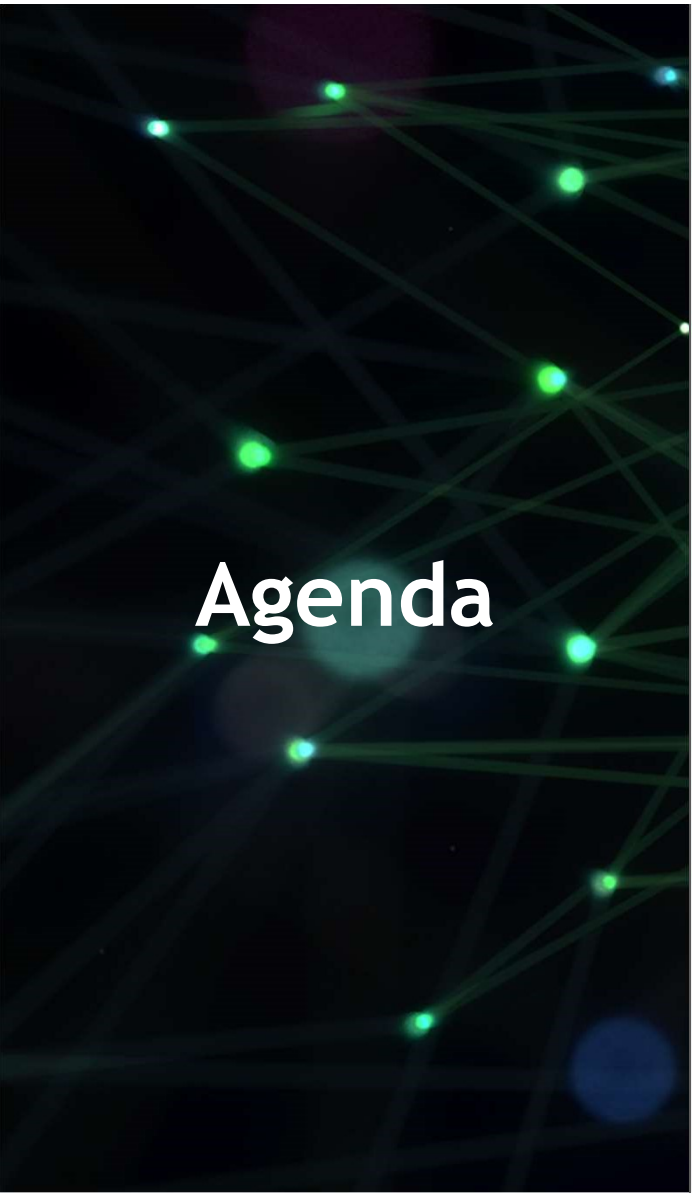
S9334: Building And Managing Scalable AI Infrastructure With NVIDIA DGX Pod And DGX Pod Management Software



Agenda

- **Building your AI Data Center with DGX Reference Architectures**
- **Creating Network Topologies**
- **DGX POD Management Software (DeepOps)**


Why Infrastructure
Matters to AI

Considerations For On-prem Vs Cloud

Keep Compute Where the Data Lives

CLOUD

- Early exploration
- Small datasets in cloud
- Few experiments
- **Careful prep for each run to save costs**

ON-PREM

- Deep Learning Enterprise
- Large datasets on-premises
- Frequent, rapid experimentation
- Creative exploration, frequent training runs
- **Fixed cost infrastructure = experiment freely**

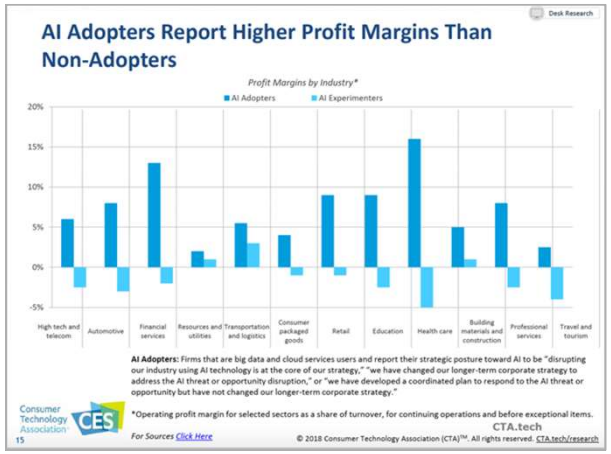
TRAIN CLOSEST TO WHERE YOUR DATA LIVES

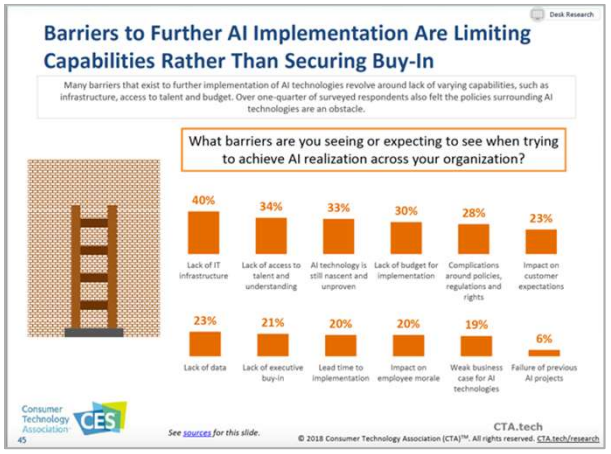
✓ Data Sovereignty and Security

✓ Lowest Cost per Training Run

✓ Fail fast, learn FASTER

AI Adopters Impeded By Infrastructure





AI Boosts Profit Margins up to 15%

40% see infrastructure as impeding AI

source: 2018 CTA Market Research


Considerations When
Selecting An AI Platform

AI Platform Considerations

Factors impacting deep learning platform decisions

DEVELOPER PRODUCTIVITY

“ Must get started now, line of business wants to deliver results yesterday

SCALING PERFORMANCE

“ I want the most GPU bang for the buck

TOTAL COST OF OWNERSHIP

“ I have limited budget, need lowest up-front cost possible

Comparing AI Compute Alternatives

Looking beyond the “spec sheet”

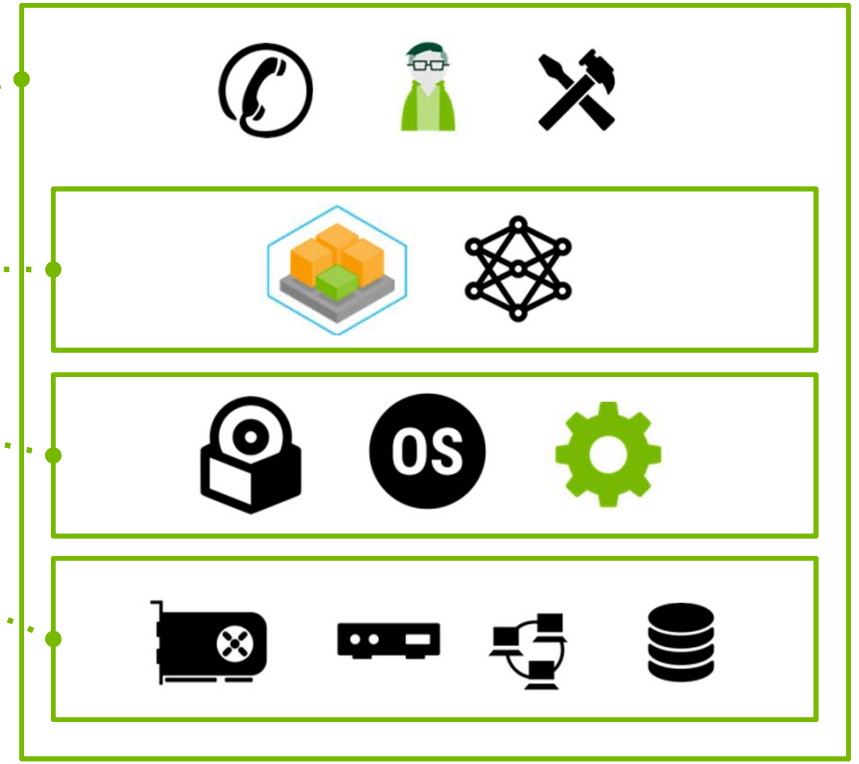
Evaluation Criteria

AI/DL Expertise & Innovation

AI/DL Software Stack

Operating System Image

Hardware Architecture



The Value Of AI Infrastructure With DGX Reference Architectures

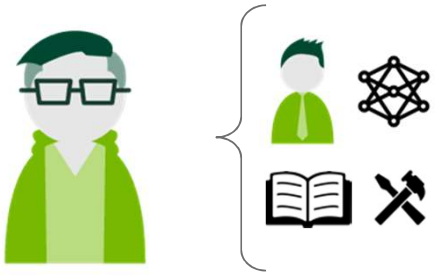
SCALABLE PERFORMANCE

Reference architectures from NVIDIA and leading storage partners

FASTER, SIMPLIFIED DEPLOYMENT

Simplified, validated, converged infrastructure offers

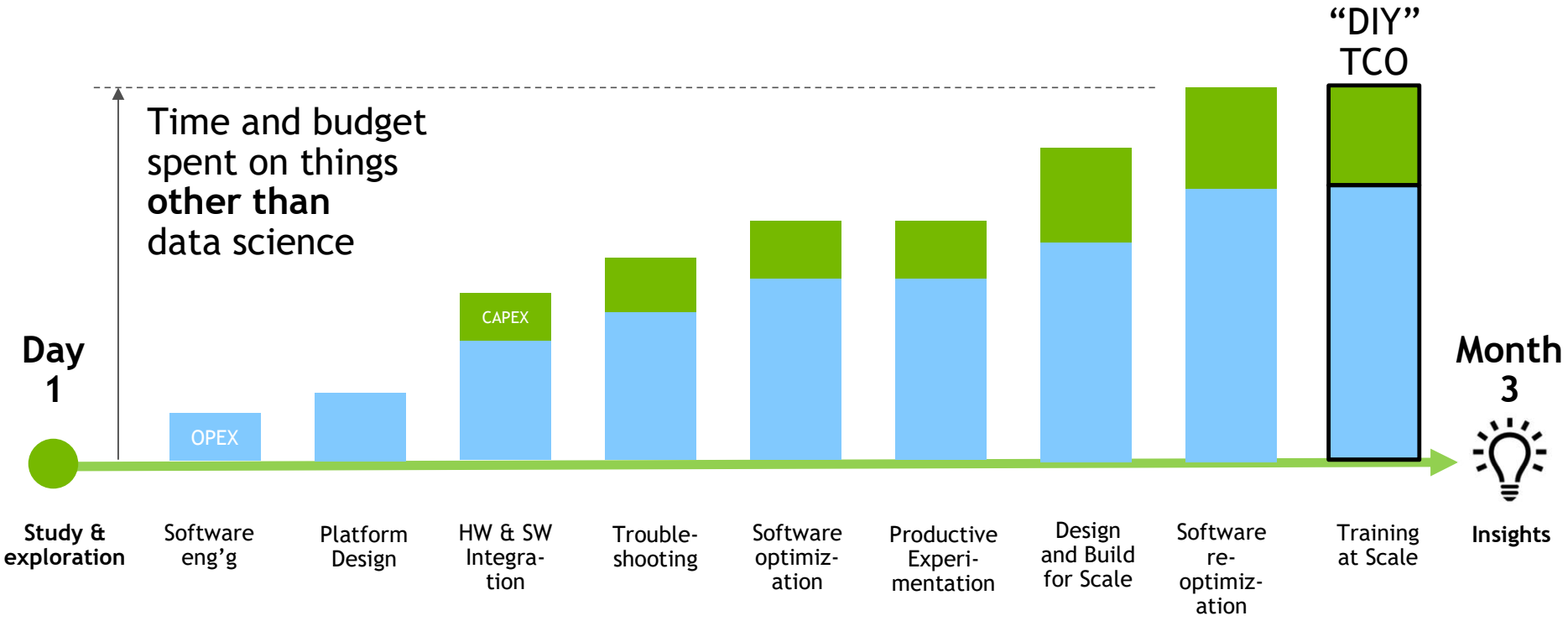
TRUSTED EXPERTISE AND SUPPORT



Available through select NPN partners as a turnkey solution

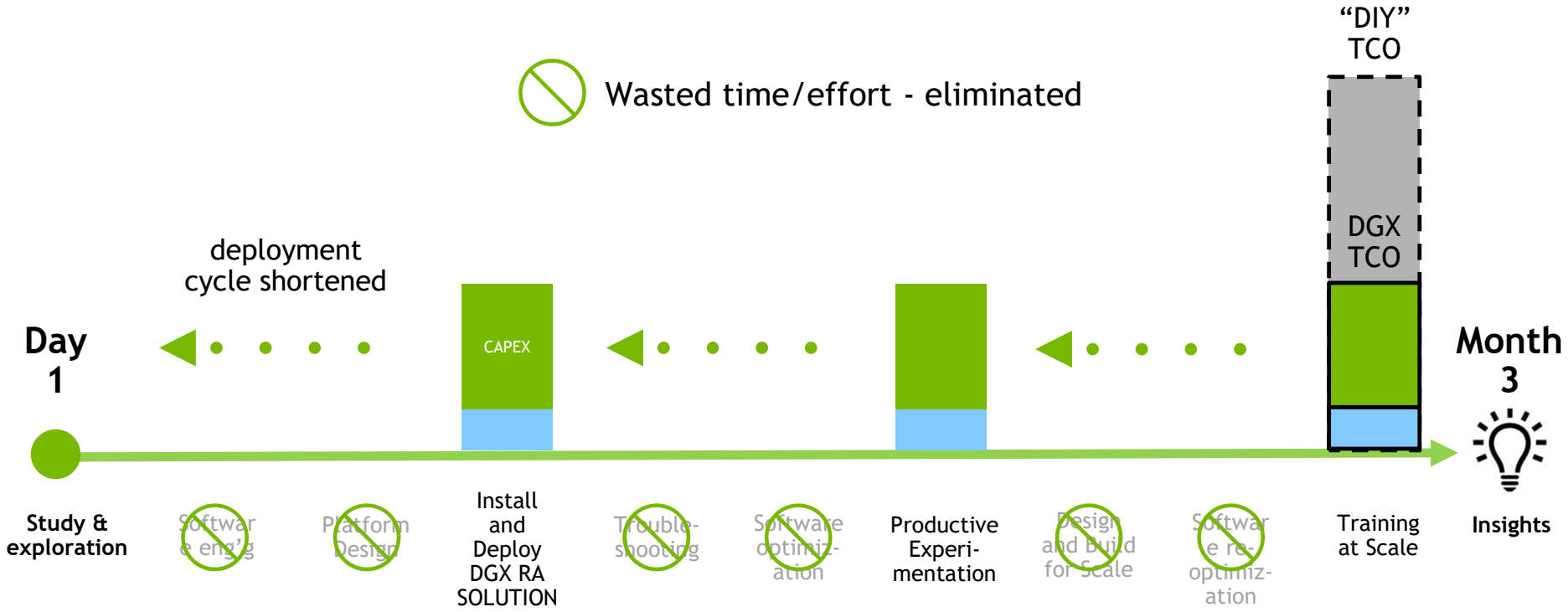

Simplifying Deployment

AI Success Delayed By Deployment Complexity



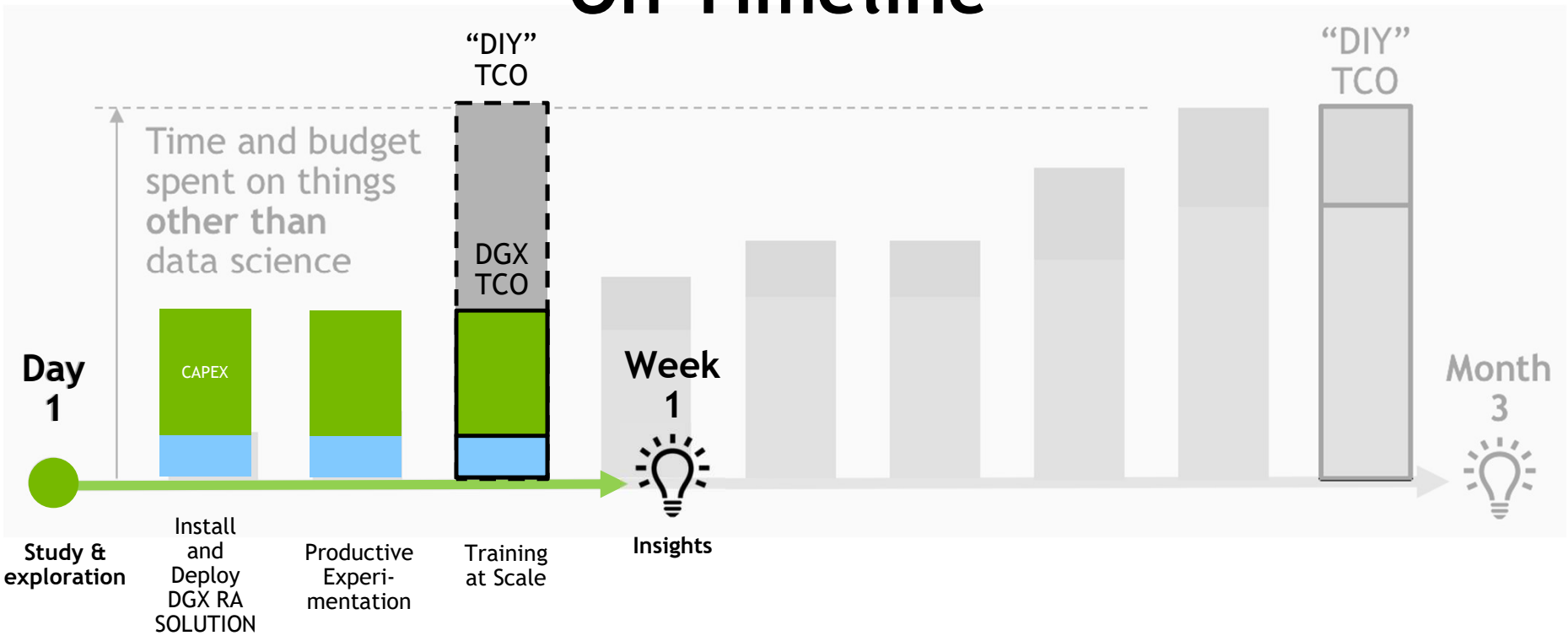
Designing, Building and Supporting an AI Infrastructure - from Scratch

The Impact Of DGX R/A Solutions On Timeline



2. Deploying an Integrated, Full-Stack AI Solution using a DGX Reference Architecture

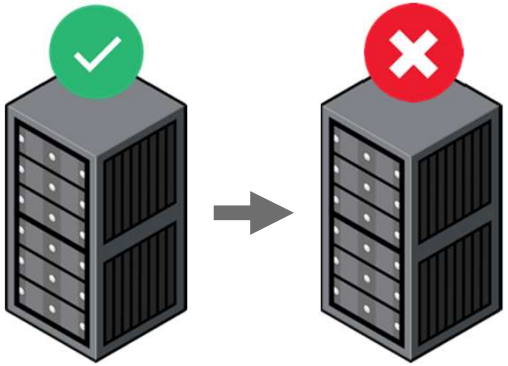
The Impact Of DGX R/A Solutions On Timeline



2. Deploying an Integrated, Full-Stack AI Solution using a DGX Reference Architecture


Supporting AI Infrastructure

Supporting AI: Alternative Approaches



Installed/ running

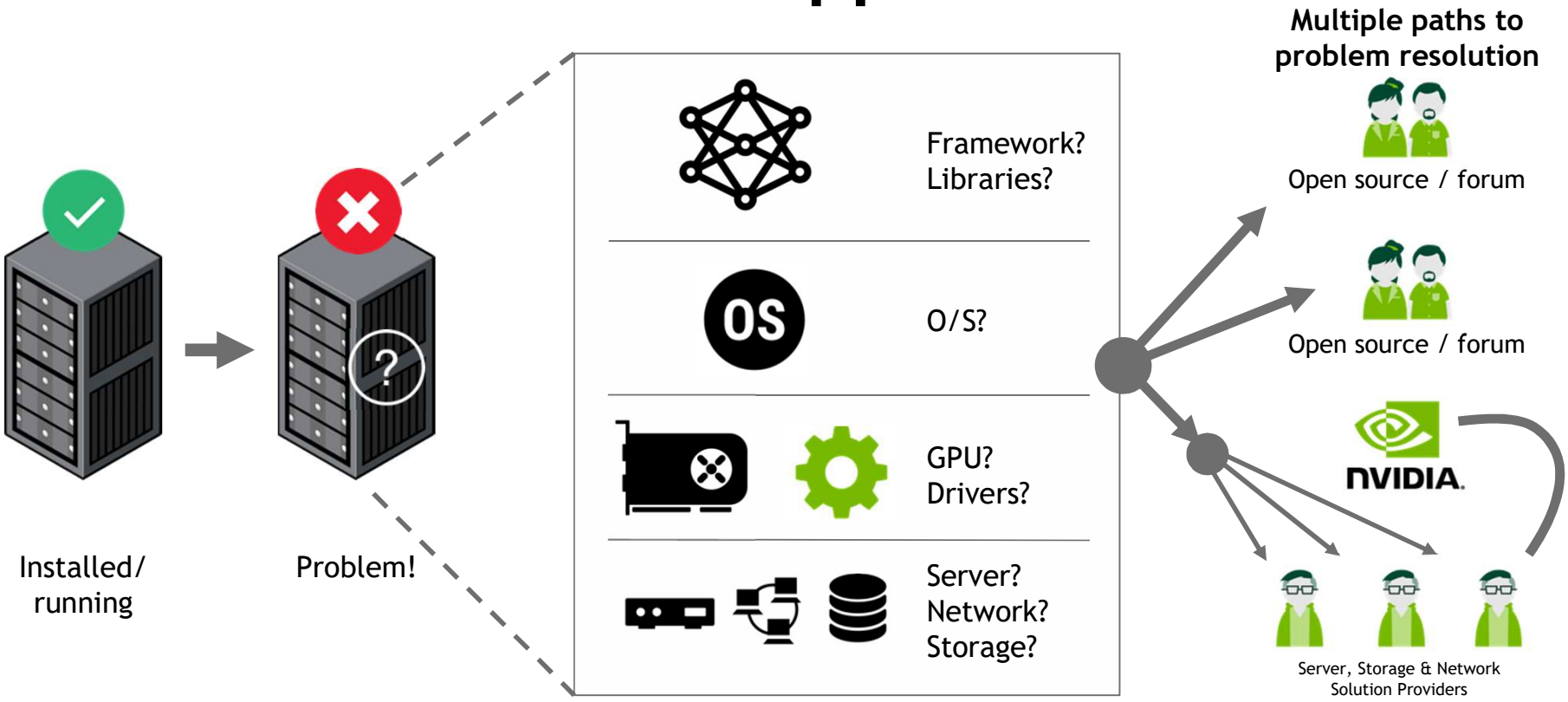
Problem!

"My PyTorch CNN model is running 30% slower than yesterday!"

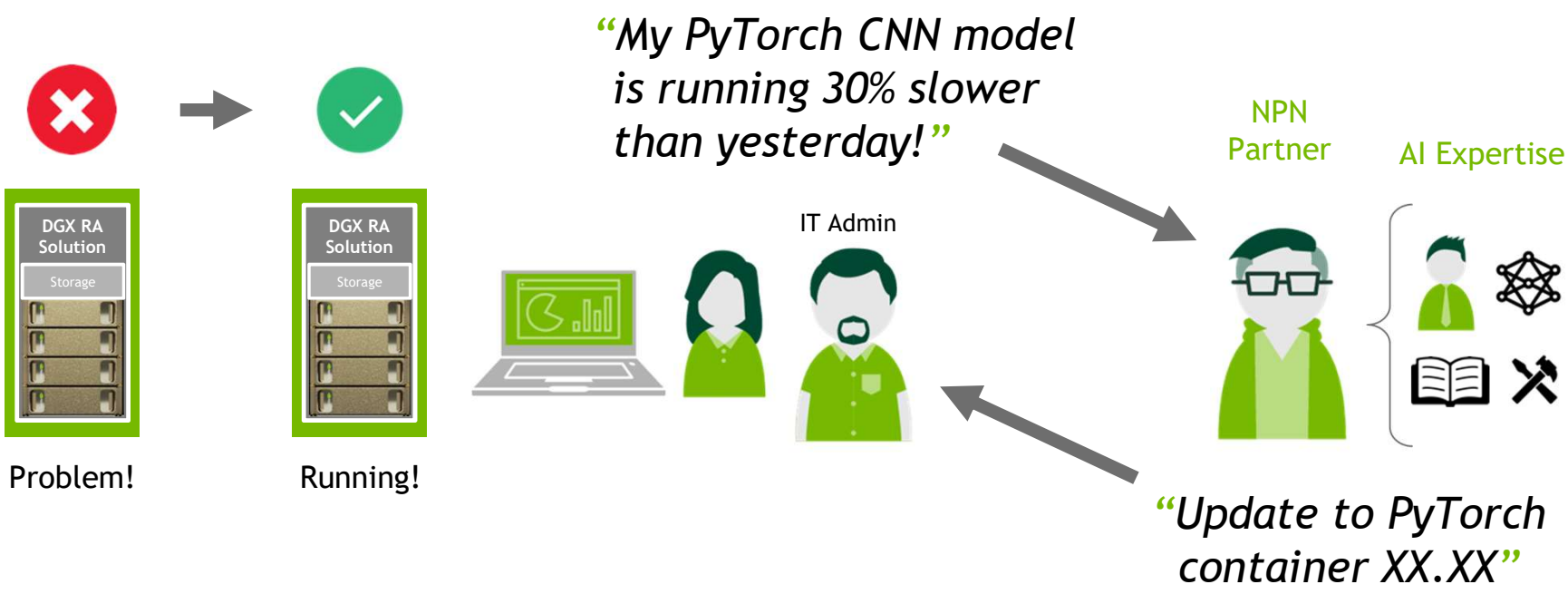
IT Admin

"OK let me look into it"

Supporting AI: Alternative Approaches



Supporting AI With DGX Reference Architecture Solutions




Creating Networking Topologies

DGX-1 POD Storage Partner Solutions

NetApp ONTAP AI

Simplify, Accelerate, and Scale the Data Pipeline for Deep Learning

HARDWARE

- **NVIDIA DGX-1** | **5**x DGX-1 Systems | 5 PFLOPS
- **NETAPP AFF A800** | HA Pair | 364TB | 1M IOPS
- **CISCO** | 2x 100Gb Ethernet Switches with RDMA

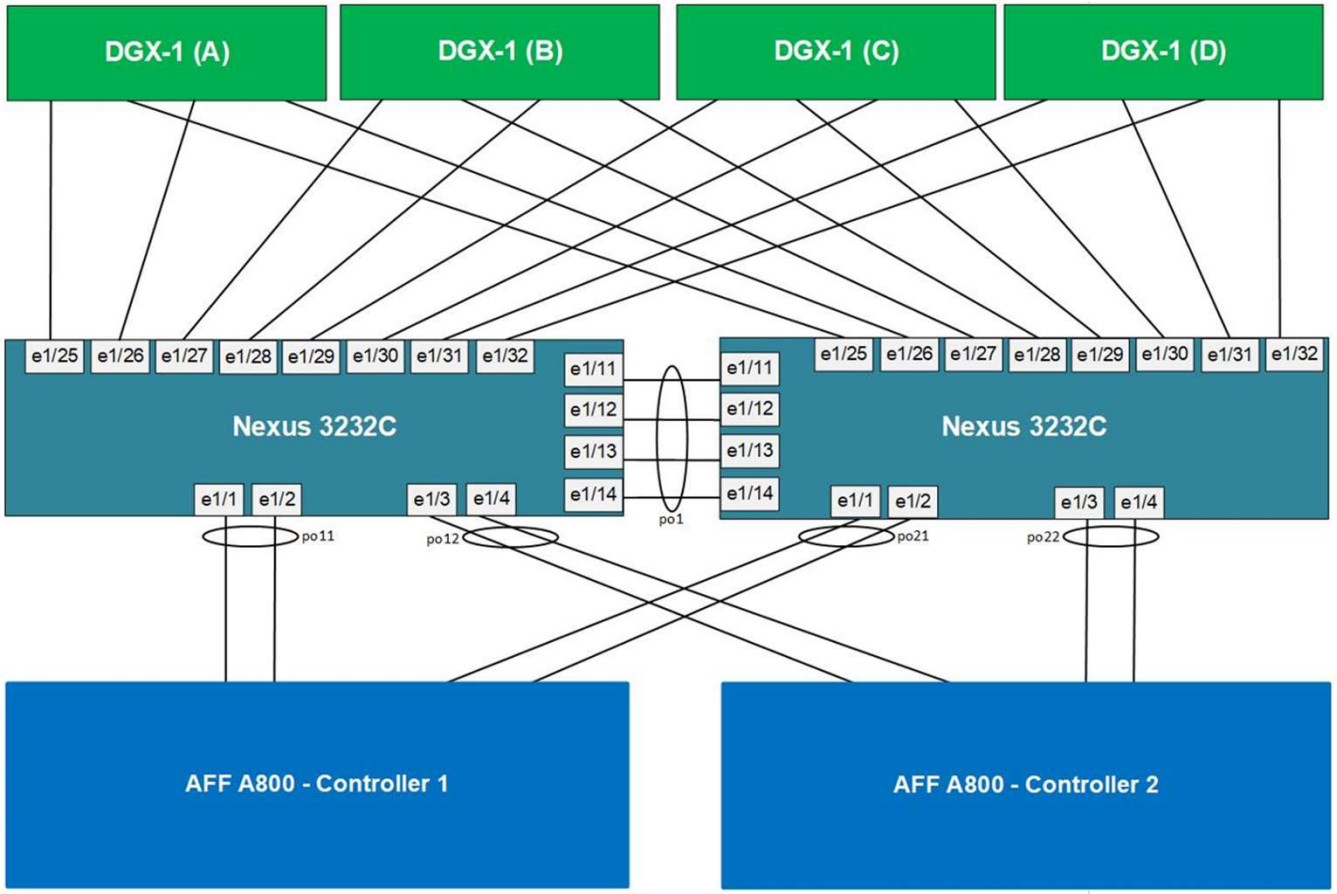
SOFTWARE

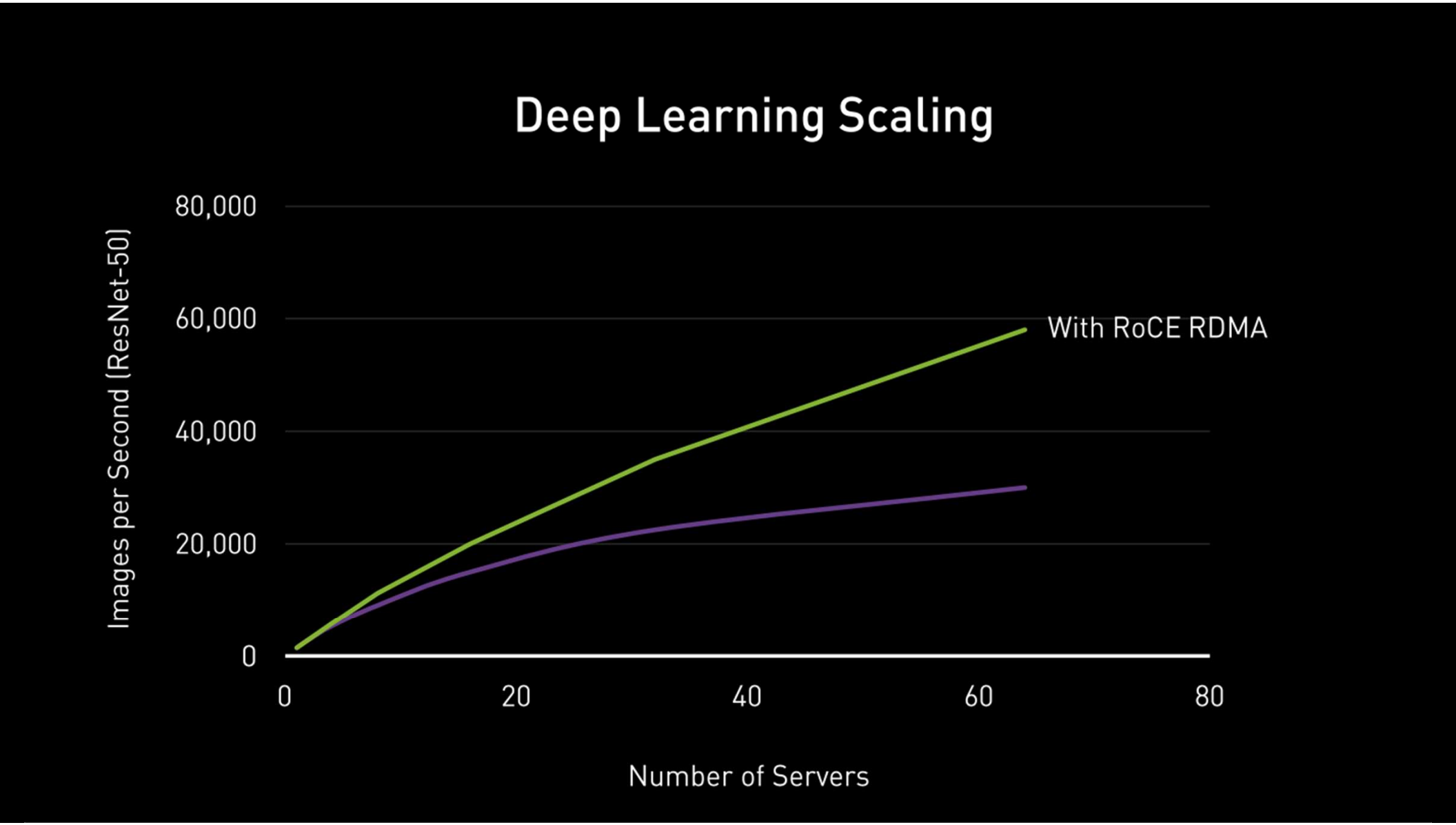
- **NVIDIA GPU CLOUD DEEP LEARNING STACK** | NVIDIA Optimized Frameworks
- **NETAPP ONTAP 9** | Simplified Data Management
- **TRIDENT** | Provision Persistent Storage for DL

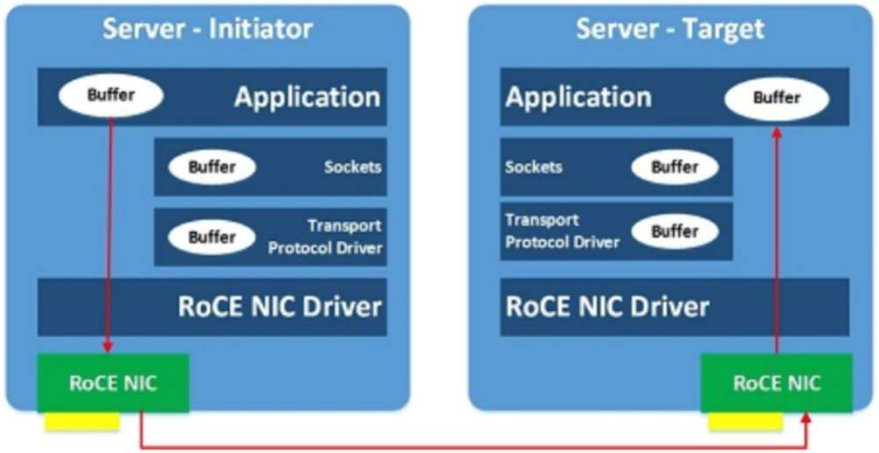
SUPPORT

- Single point of contact support
- Proven support model

NetApp Network Switch Port Configuration




Deep Learning Scaling
80,000
60,000
40,000
20,000
0
Images per Second (ResNet-50)
With RoCE RDMA
0
20
40
60
80
Number of Servers



RDMA over Converged Ethernet (RoCE) RoCE

Microsoft SMB Direct | Hewlett Packard Enterprise | SEAGATE xyratex | IBM xiv tms

ORACLE | TOSHIBA | DataDirect NETWORKS | FUJITSU

DELL EMC | Micron | PURESTORAGE

INFINIDAT | X-IO technologies BETTER BY DESIGN | TERADATA | Western Digital VIRIDENT | NetApp

- Ethernet
- InfiniBand

Mellanox TECHNOLOGIES

- RoCE (2010) defines how to perform RDMA (1999) over Ethernet
- InfiniBand architecture specification defines how to perform RDMA over an InfiniBand network

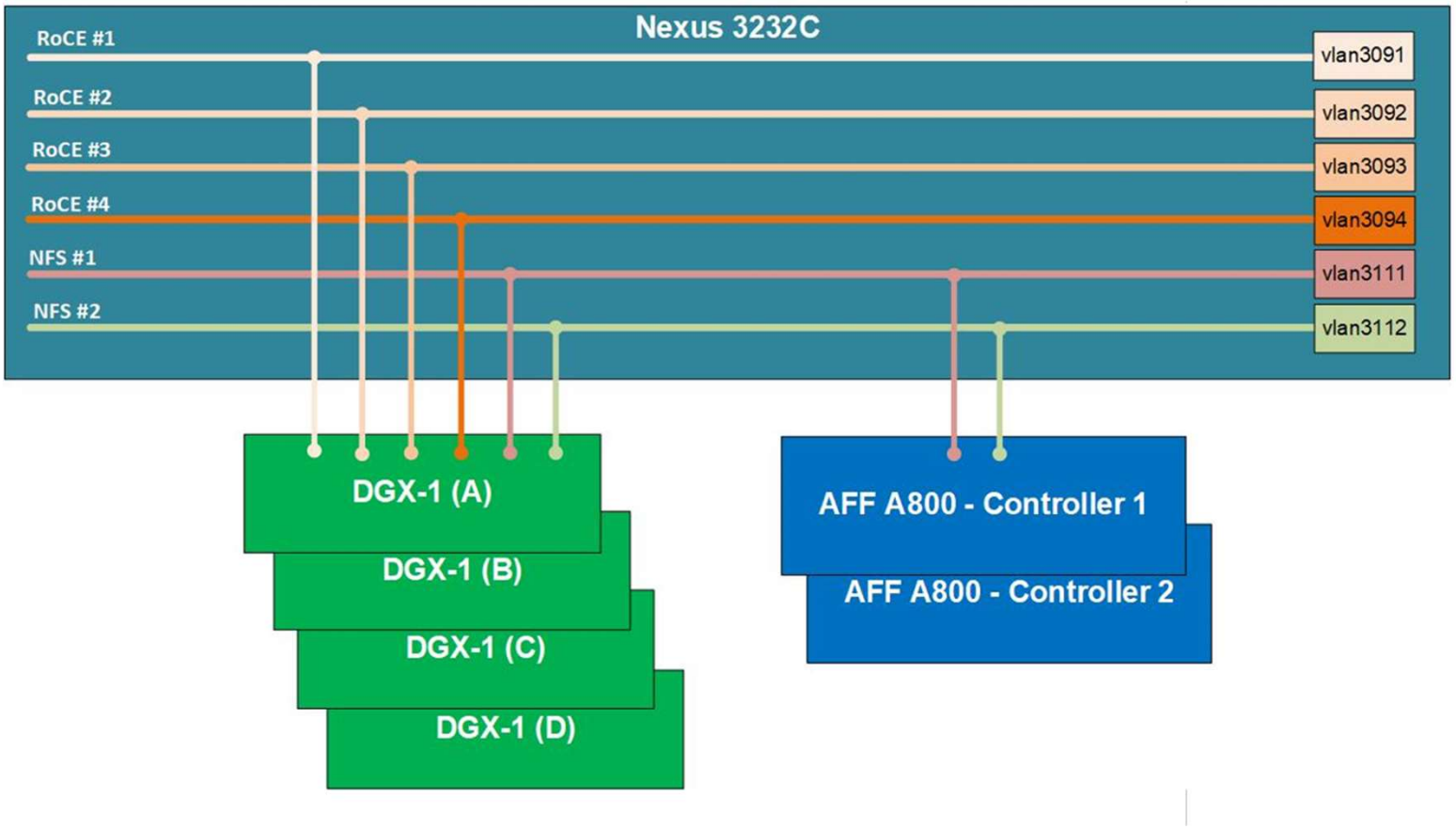
ONTAP AI with Ansible

Eliminate deployment and management complexity

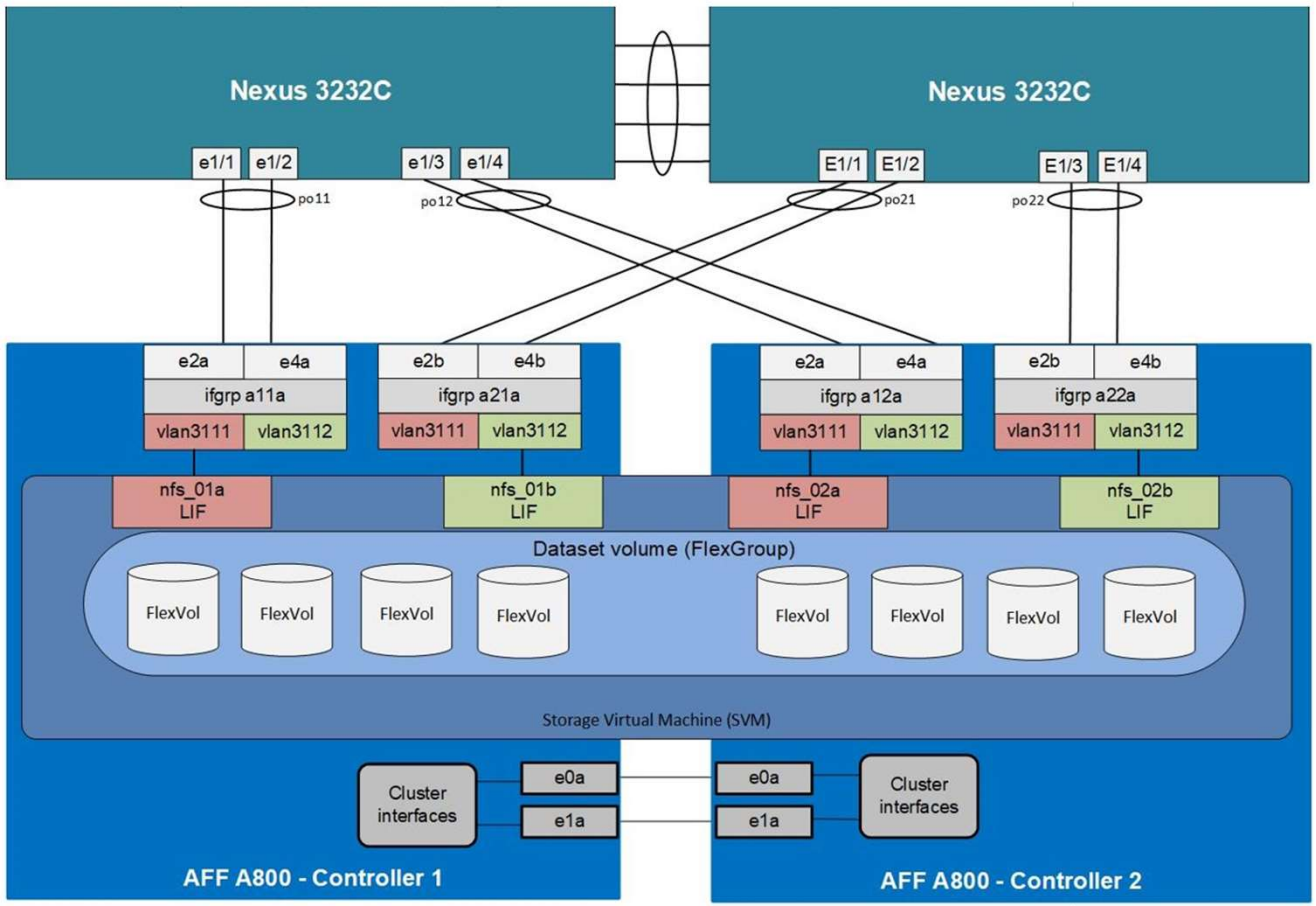
- Deploy the whole stack in ~20 minutes
- Network configuration – 20 discreet tasks
  - VLANs, ports & port-channels
  - Complete QoS configuration for RoCE support
- Storage configuration – 18 tasks
  - Logical and physical storage networking
  - FlexGroup logical storage configuration
- DGX Server configuration – 24 tasks
  - Basic OS management- packages, NTP, etc.
  - RoCE cluster interconnect & NFS Storage network configuration
  - NFS data volume mounts

https://blog.netapp.com/how-to-configure-ontap-ai-in-20-minutes-with-ansible-automation/

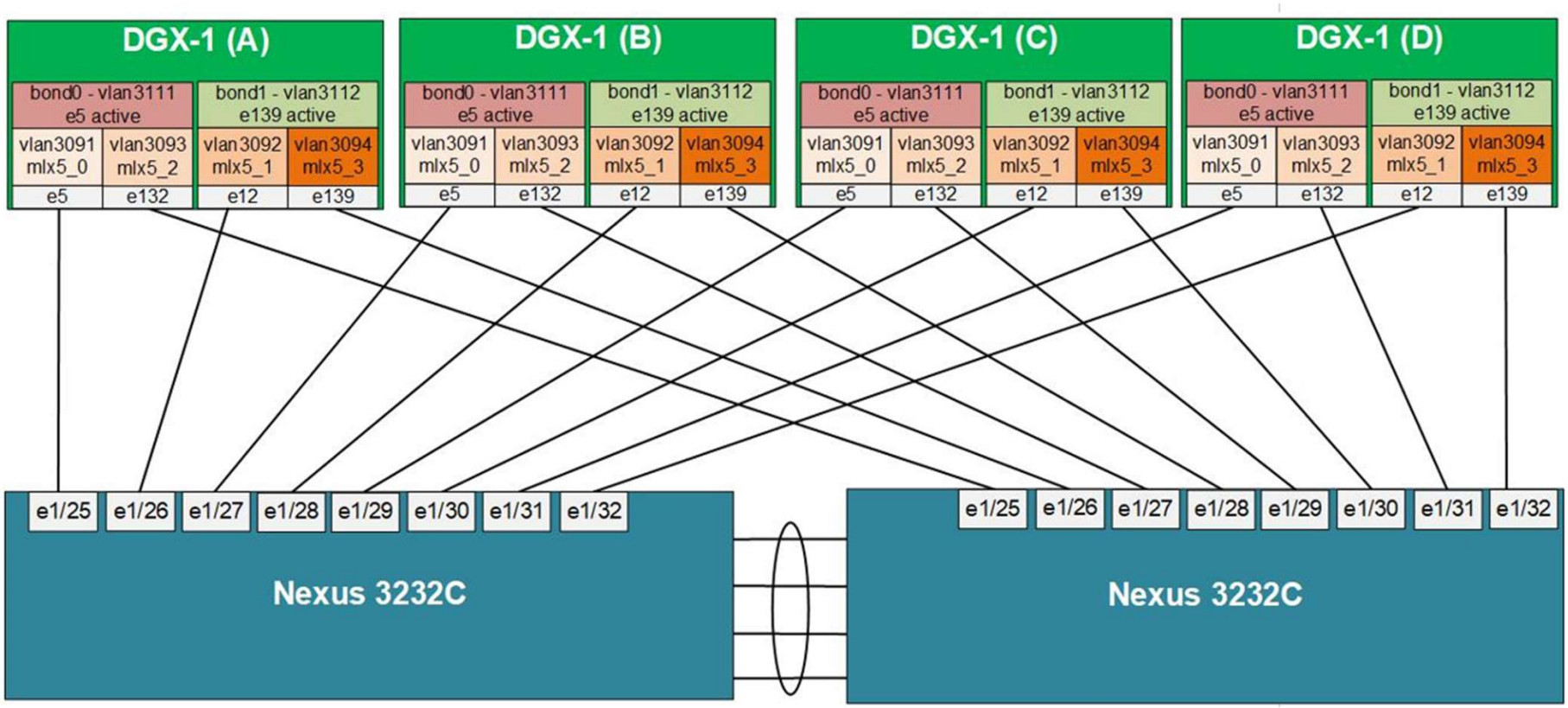
NetApp VLAN Connectivity for DGX-1 Servers and Storage System Ports



NetApp Storage System Configuration



NetApp Host Configuration



AIRI: AI-Ready Infrastructure

Extending the power of DGX-1 at-scale in every enterprise

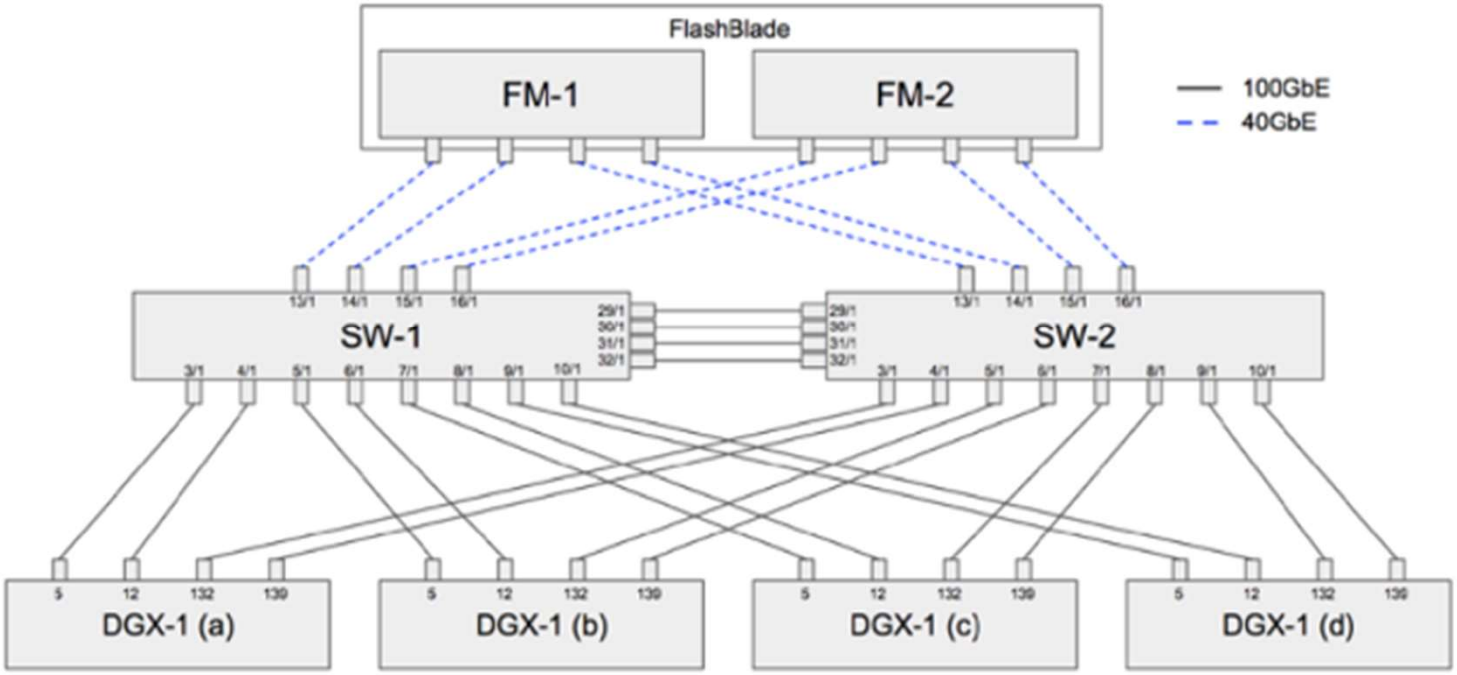
HARDWARE

- **NVIDIA DGX-1** | 4x DGX-1 Systems | 4 PFLOPS
- **PURE FLASHBLADE™** | 15x 17TB Blades | 1.5M IOPS
- **CISCO or ARISTA** | 2x 100Gb Ethernet Switches with RDMA

SOFTWARE

- **NVIDIA GPU CLOUD DEEP LEARNING STACK** | NVIDIA Optimized Frameworks
- **AIRI SCALING TOOLKIT** | Multi-node Training Made Simple

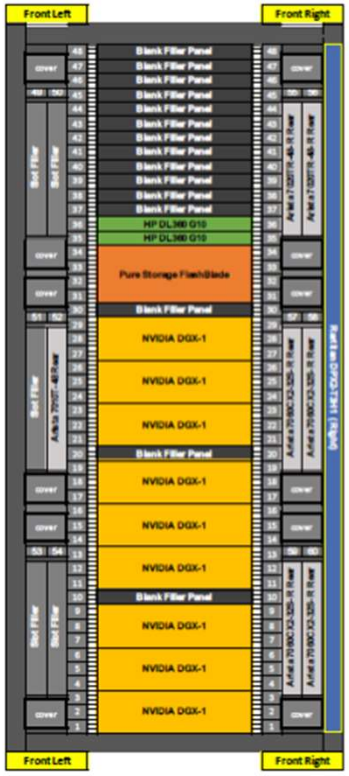
Pure Storage Network Topology



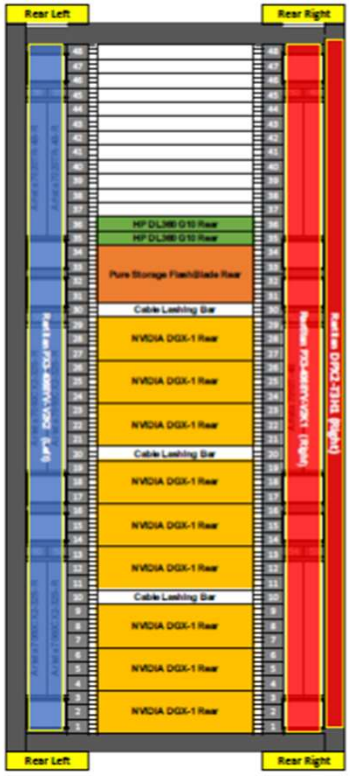
Rack Design & Builds

GROUPWARE TECHNOLOGY

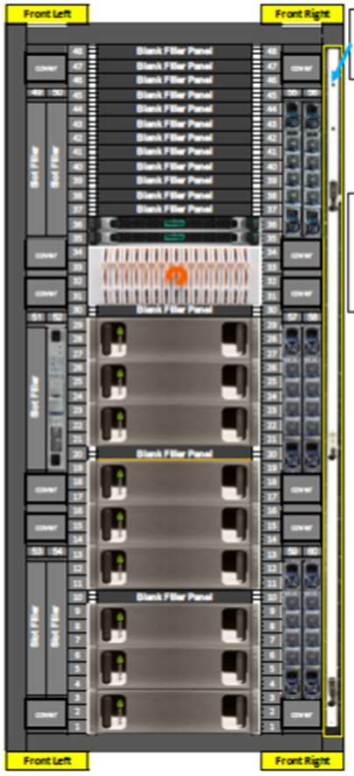
Logical Front View



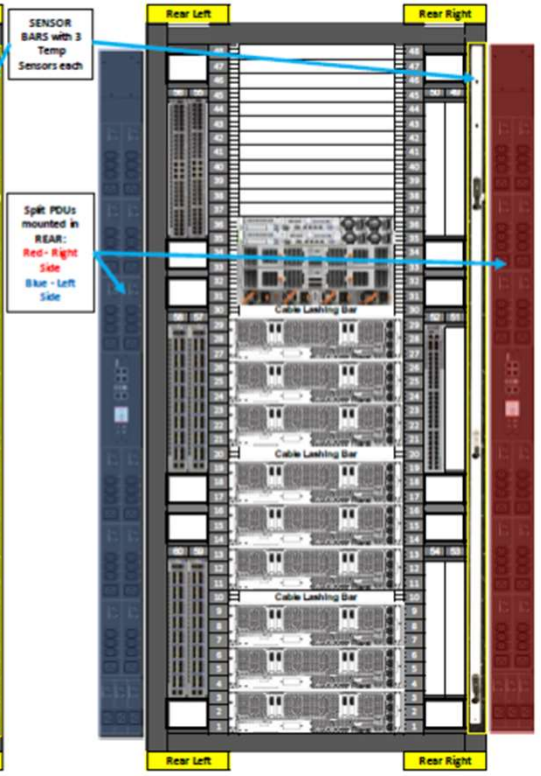
Logical Rear View



Physical Front View



Physical Rear View



DDN A3I with DGX-1

Making AI-Powered Innovation Easier

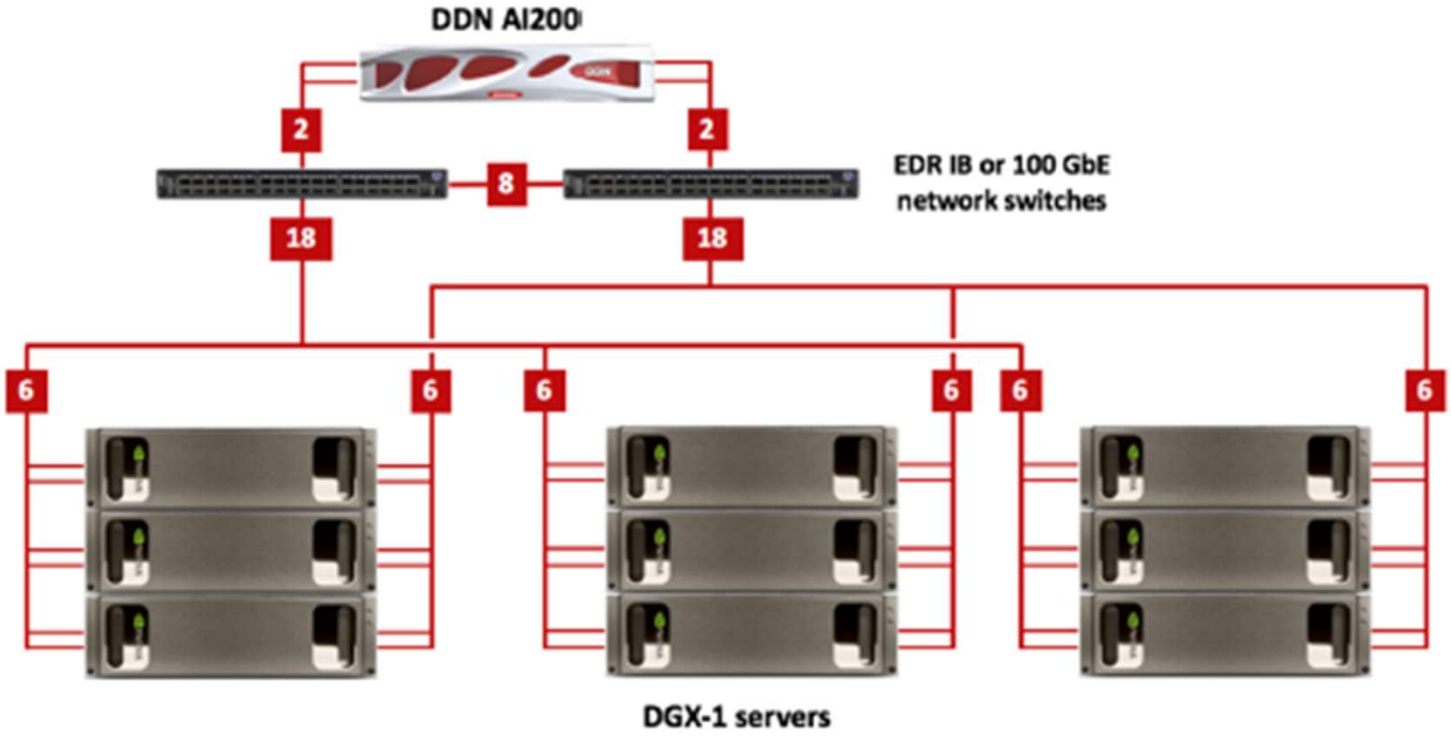
HARDWARE

- **NVIDIA DGX-1** | **4**x DGX-1 Systems | 4 PFLOPS
- **DDN AI200, AI7990** | 20GB/s | from 30TB | 350K IOPS
- **NETWORK: 2**x EDR IB or 100GbE Switches with RDMA

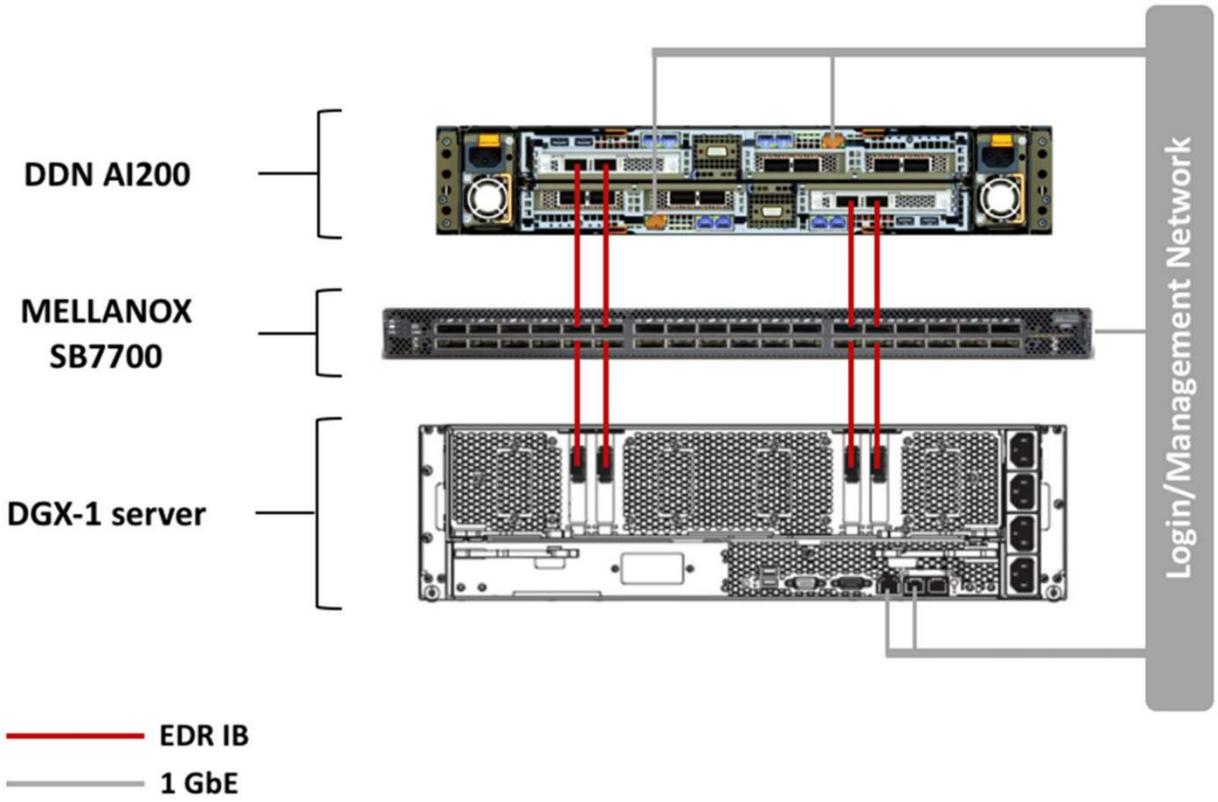
SOFTWARE

- **NVIDIA GPU CLOUD DEEP LEARNING STACK** | NVIDIA Optimized Frameworks
- **DDN:** High performance, low latency, parallel file system
- **DDN:** In-container client for easy deployment, efficiency, performance and reliability

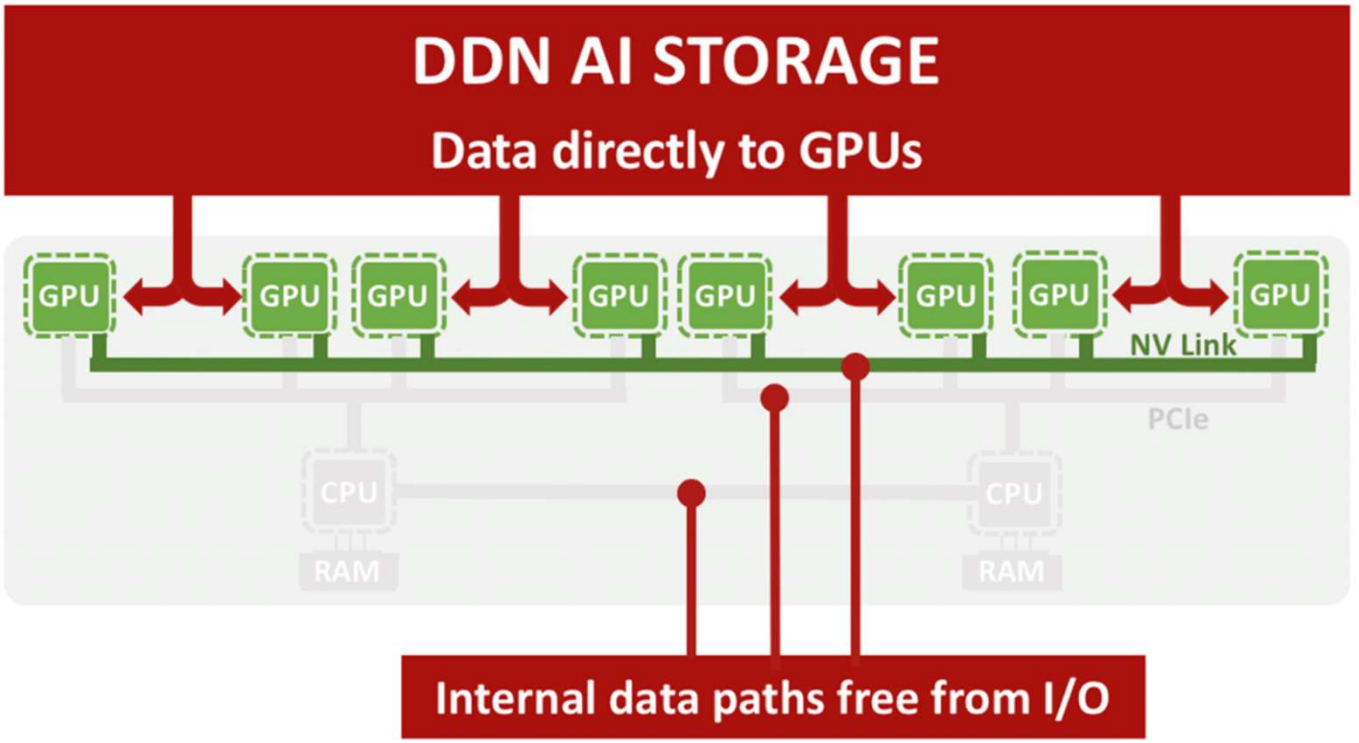
DDN A3I Reference Architecture
9:1 Configuration



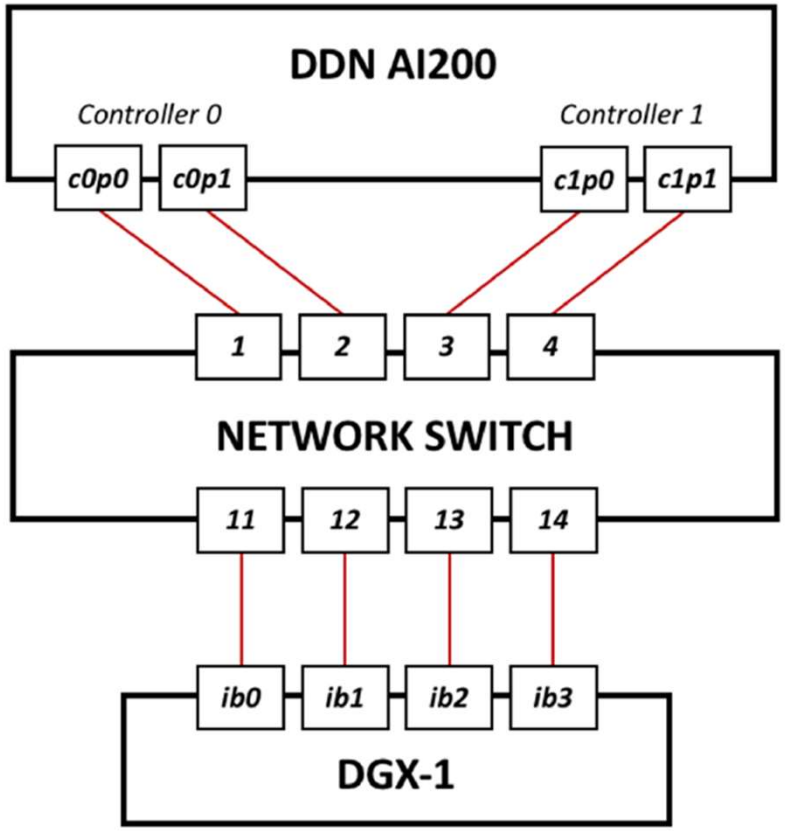
Network Diagram of DDN A3I Benchmark Testing Environment



Optimized Data Delivery for DGX-1 server with DDN A3I



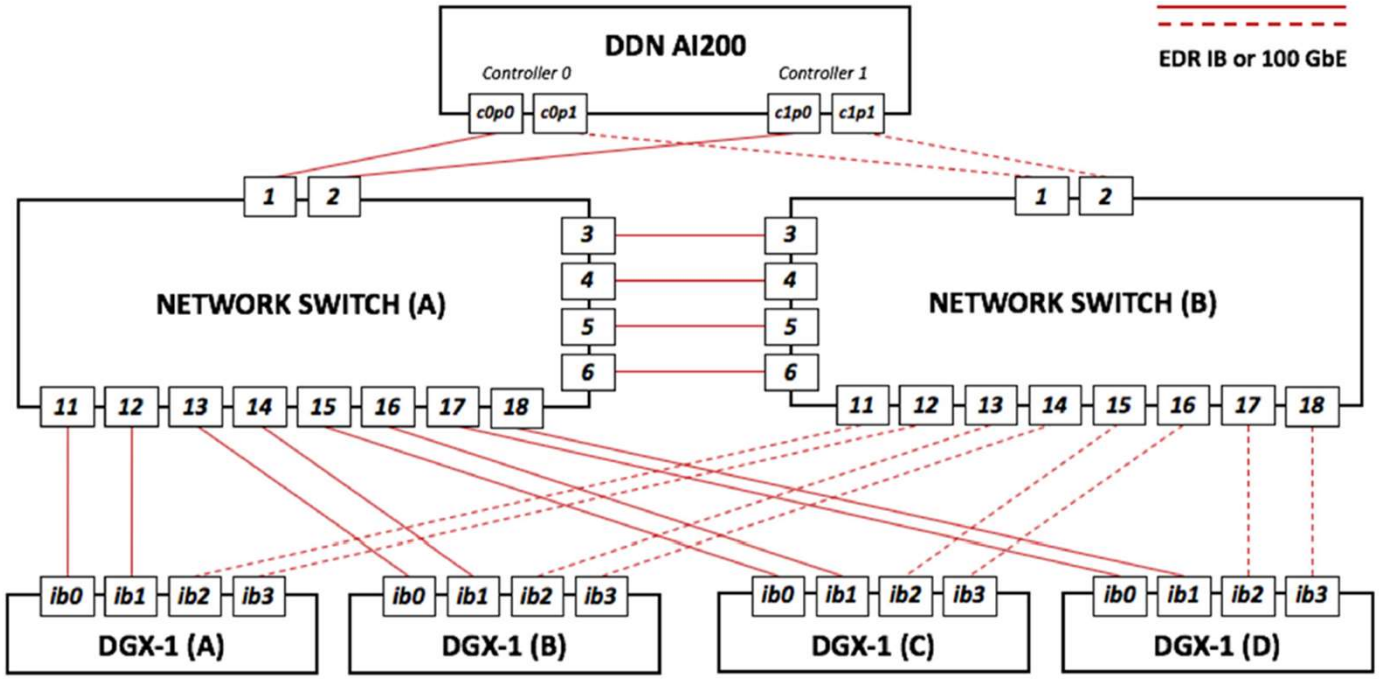
DDN Network Diagram of Port-Level Connectivity



EDR IB or 100 GbE

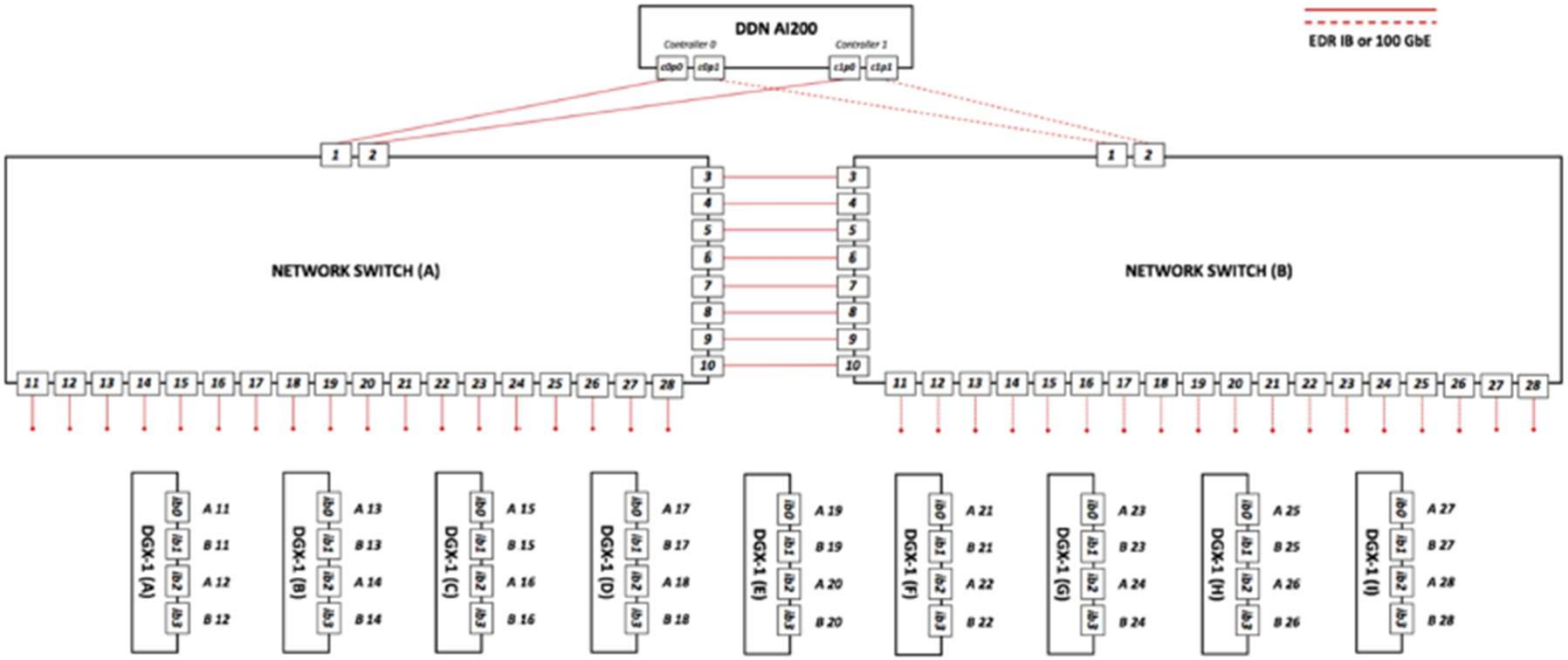
1:1 configuration

DDN Network Diagram of Port-Level Connectivity



4:1 configuration

DDN Network Diagram of Port-Level Connectivity

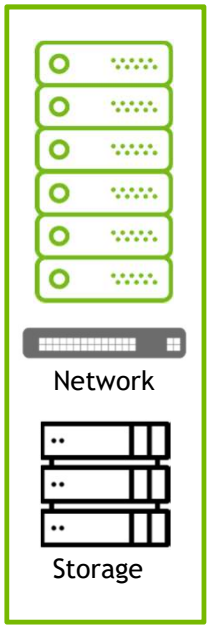


9:1 configuration


DGX POD Management Software
(DeepOps)

You've Got A Shiny New DGX POD!

What now?



Cluster Deployment & Maintenance

OS | Firmware | Monitoring | Security / Access | Job Scheduling

Other important considerations

Airgap | NGC Containers

DeepOps

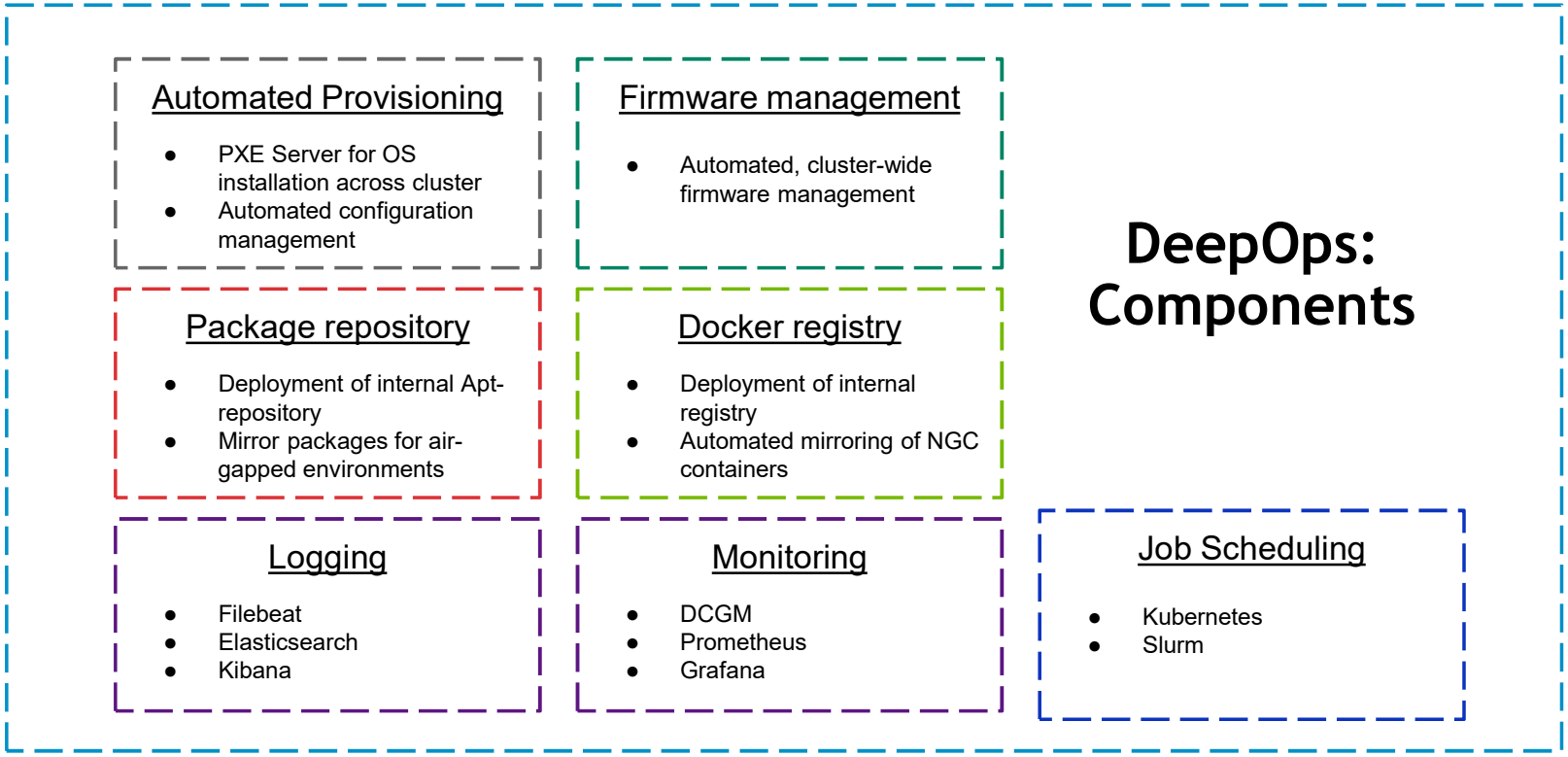
What is it?

- **Open-source** project to **facilitate deployment of multi-node GPU clusters** for Deep Learning and HPC environments, in an on-premise, optionally air-gapped datacenter or in the cloud
- DeepOps is also recognized as the **DGX POD Management Software**
- The **modular** nature of the project also allows more experienced administrators to pick and choose items that may be useful, making the process compatible with their existing software or infrastructure
- GitHub: https://github.com/NVIDIA/deepops

Note: You can use DeepOps to configure any NVIDIA GPU-Accelerated platform (and not just DGX servers).

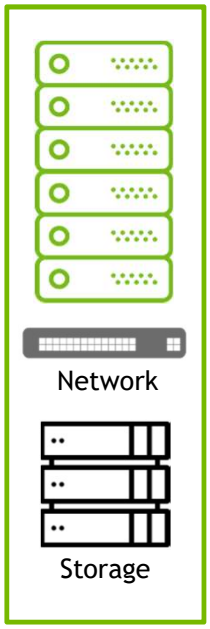
Building out your GPU cluster

DeepOps Components



Here's What We'll Build Today

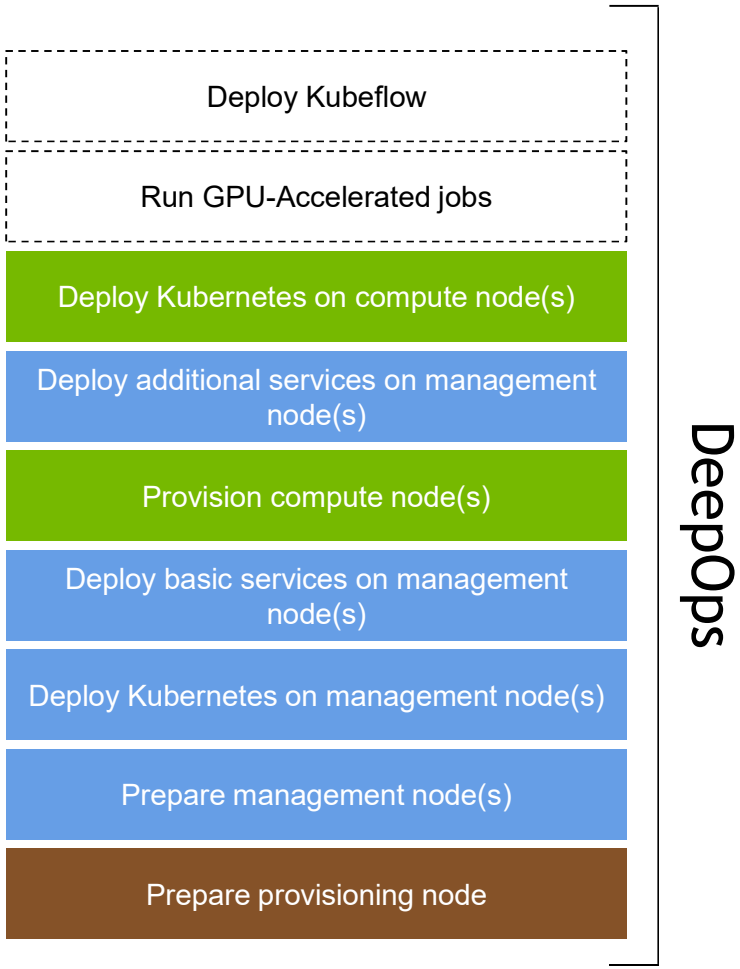
To cluster and beyond!



GPU Compute node(s):
- For high-performance compute workloads

Management node(s):
- Used for cluster management

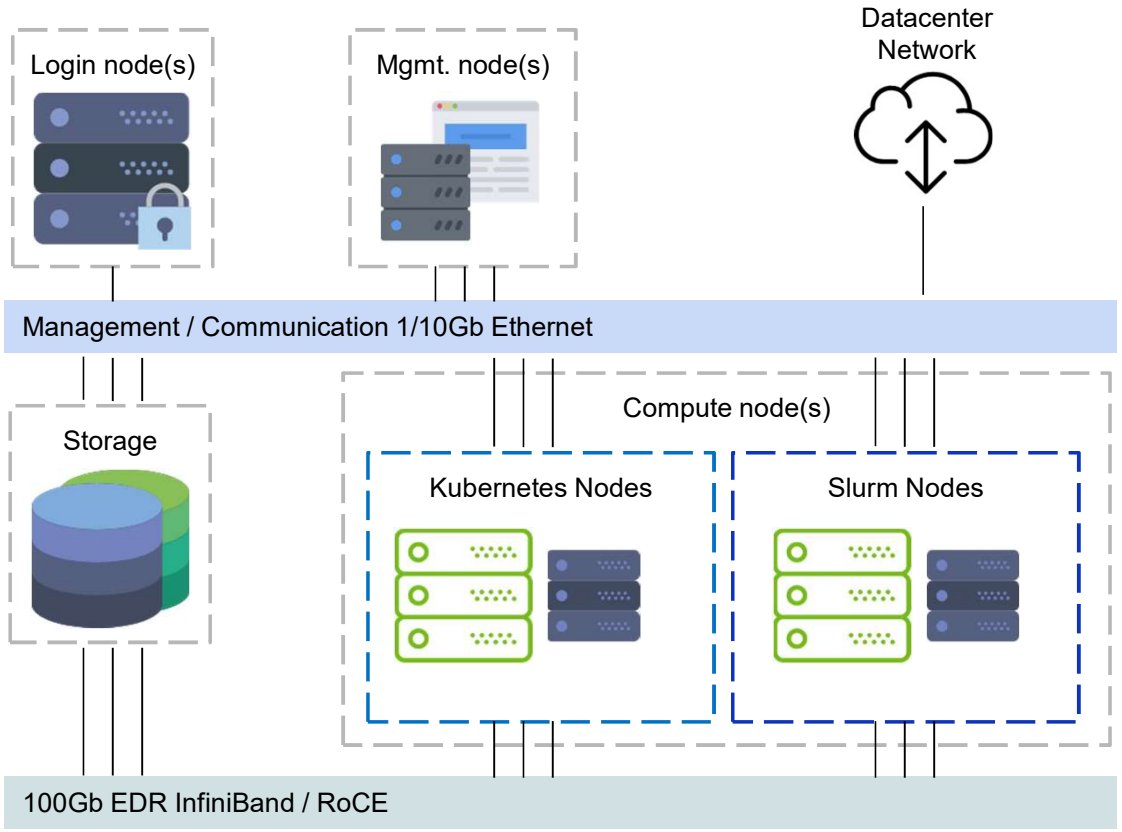
Provisioning node:
- Orchestrates the initial setup of the cluster



Architectural Considerations

ARCHITECTURE

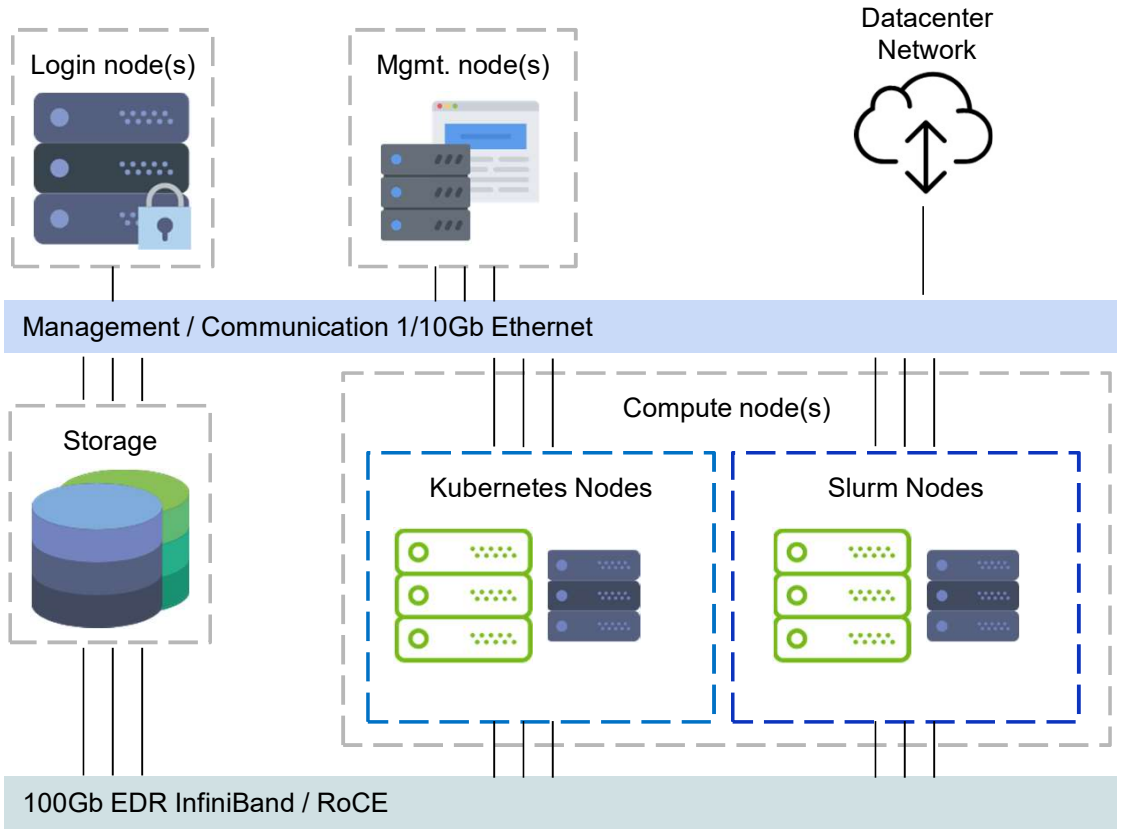
Building Multi-node GPU Clusters with DeepOps



- 1x CPU-only login node
- Odd number of CPU-only management nodes
  - required for etcd key-value store
- 1/10Gb Ethernet control & management networks
  - Management, connectivity, command & control
- Fully non-blocking fat-tree 100Gb EDR Infiniband topology
  - Use the biggest EDR IB core switch that fits

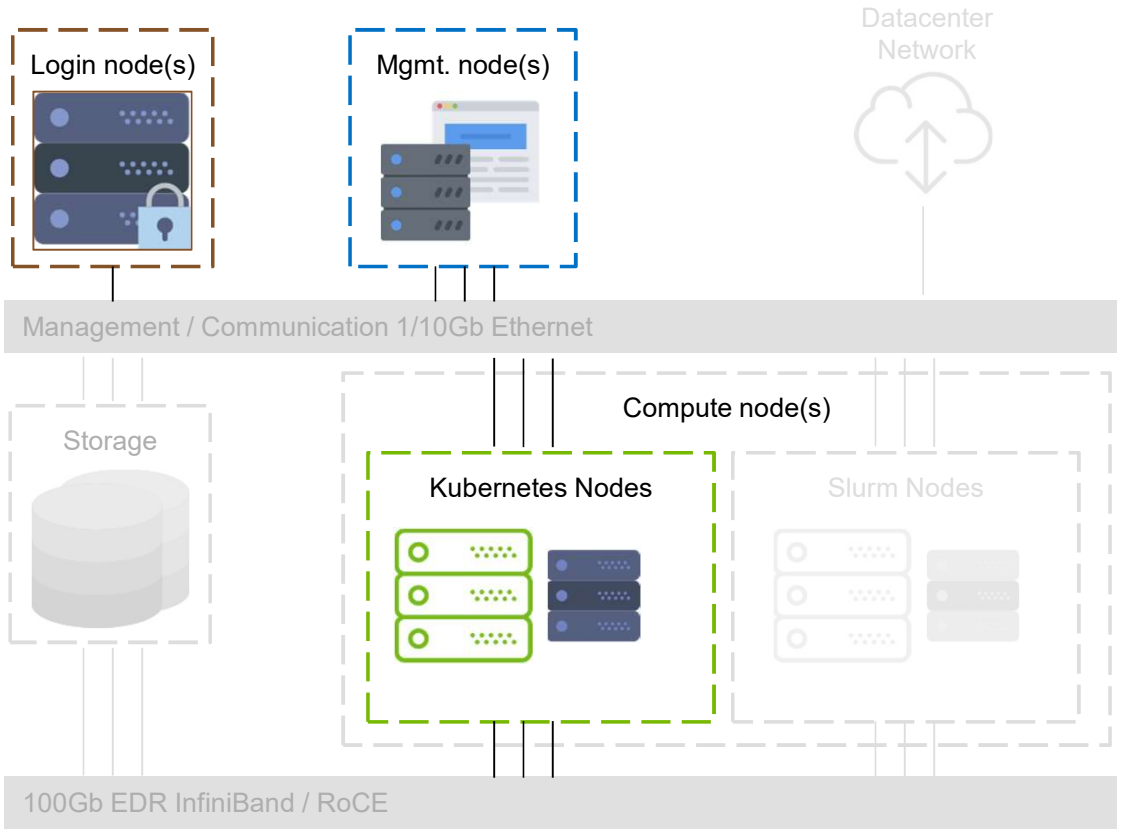
ARCHITECTURE

Benefits

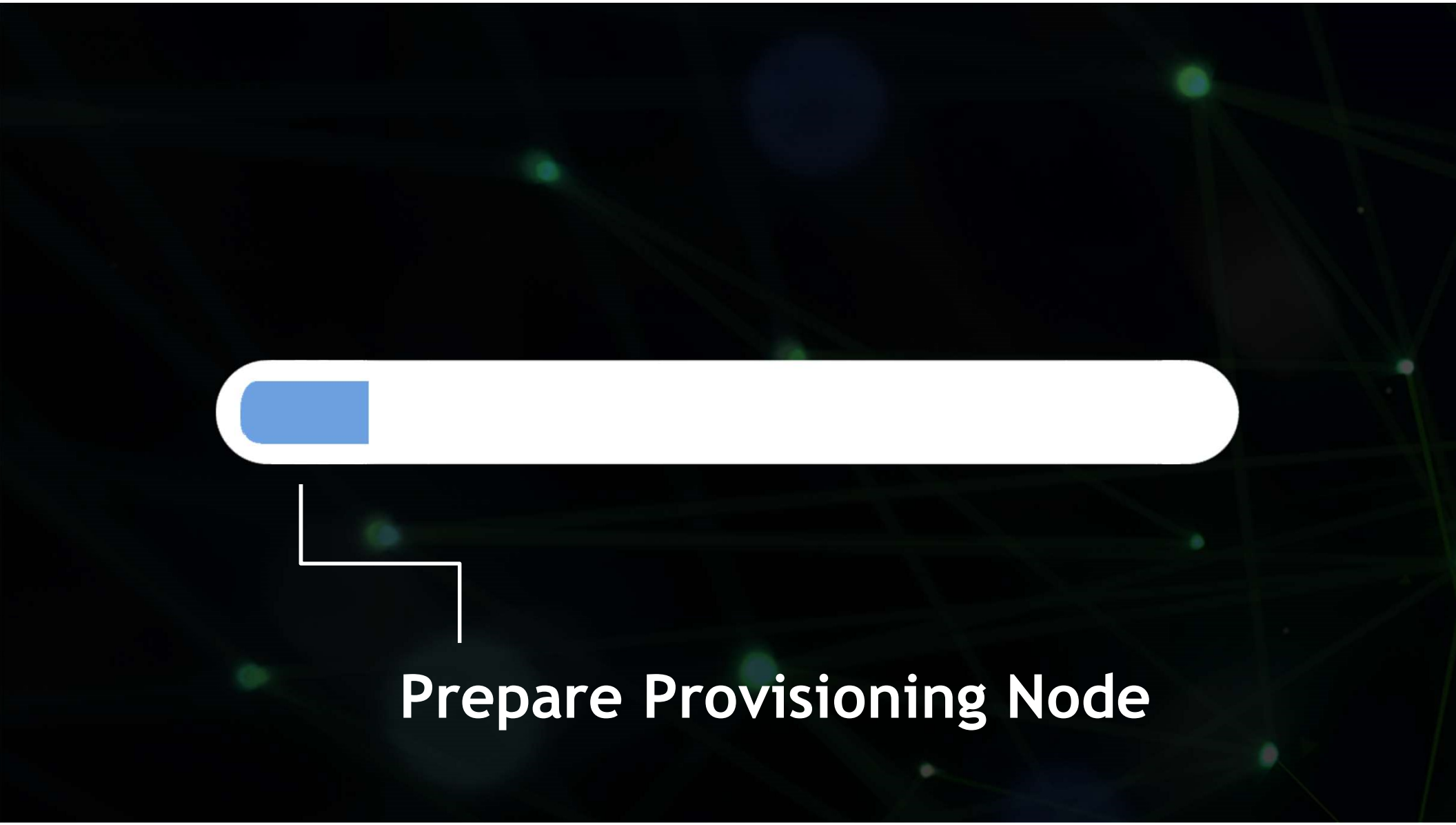


- **Augment legacy HPC schedulers** with new features:
  - Cluster management services
  - Jupyter notebooks
  - Deep learning inference deployments (TensorRT)
- **Keep data in the same place**, no need to have separate clusters
- **Free up deep learning researchers to do DL**, not become devsecops/sysadmin

ARCHITECTURE

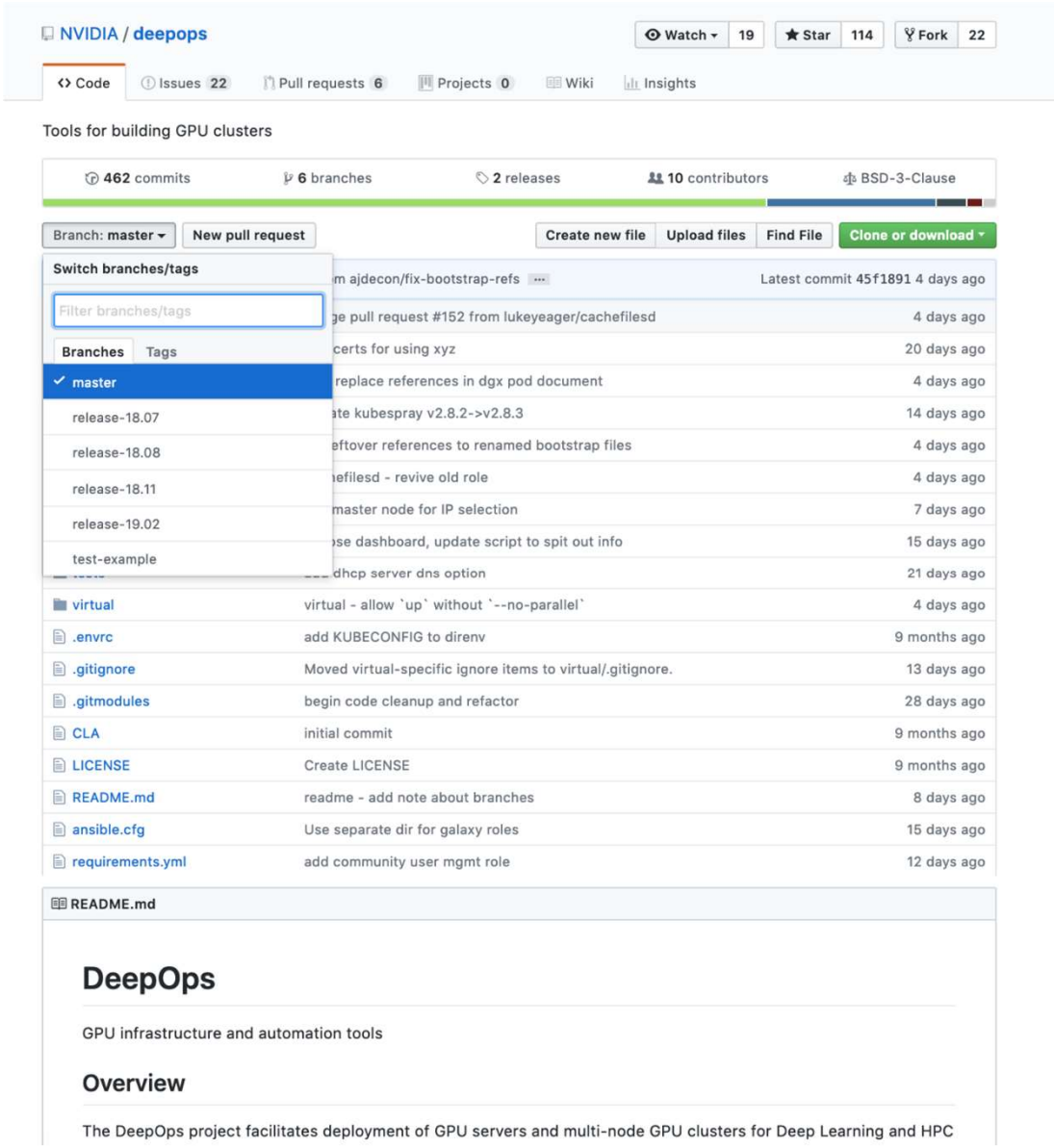


What we'll cover today


Prepare Provisioning Node

Clone the DeepOps Project on the Provisioning Server

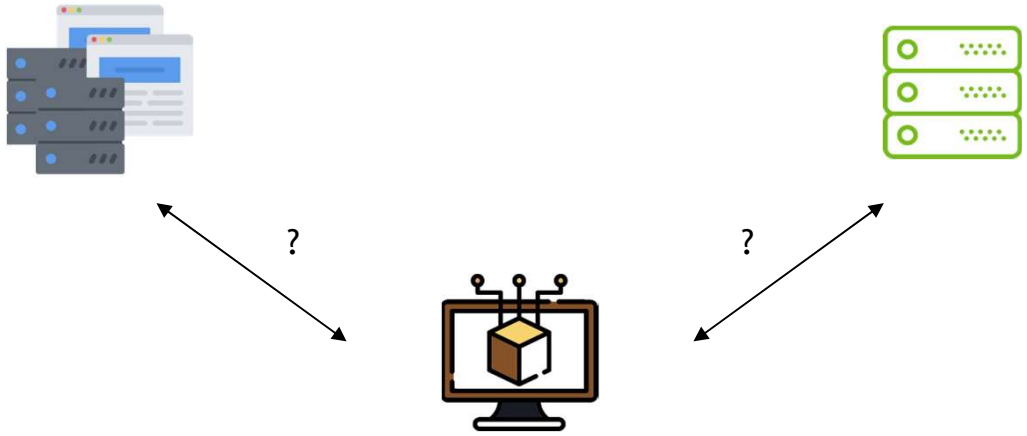
GitHub:
https://github.com/NVIDIA/deepops



Prep the provisioning node:

1. Install software dependencies
2. Setup config directory

```
1. ubuntu@deepops-mgmt4-2: ~/deepops (ssh)
ubuntu@deepops-mgmt4-2:~/deepops$ ./scripts/bootstrap-mgmt.sh
gpg: keyring `/tmp/tmpypsrqc7m/secring.gpg' created
gpg: keyring `/tmp/tmpypsrqc7m/pubring.gpg' created
gpg: requesting key 7BB9C367 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpypsrqc7m/trustdb.gpg: trustdb created
gpg: key 7BB9C367: public key "Launchpad PPA for Ansible, Inc." imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
Get:1 http://ppa.launchpad.net/ansible/ansible/ubuntu xenial InRelease [18.0 kB]
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://ppa.launchpad.net/ansible/ansible/ubuntu xenial/main amd64 Packages [540 B]
Get:5 http://ppa.launchpad.net/ansible/ansible/ubuntu xenial/main Translation-en [344 B]
Get:6 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:7 http://archive.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Get:8 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [911 kB]
Get:9 http://archive.ubuntu.com/ubuntu xenial-updates/main Translation-en [367 kB]
Get:10 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [727 kB]
64% [Waiting for headers]
```

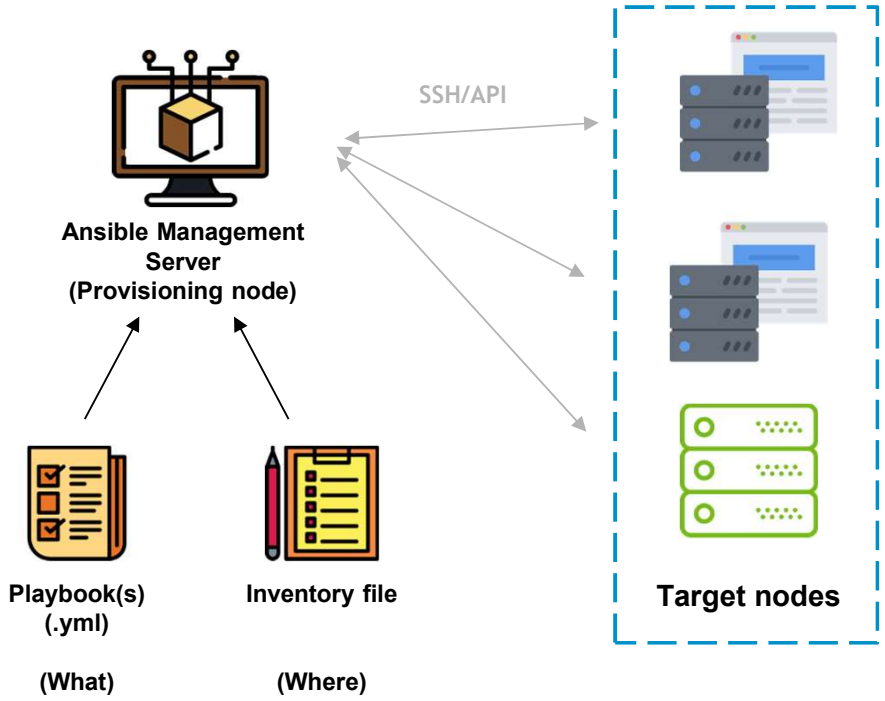


Our Provisioning node now has the necessary software.

But how will it orchestrate the cluster deployment?

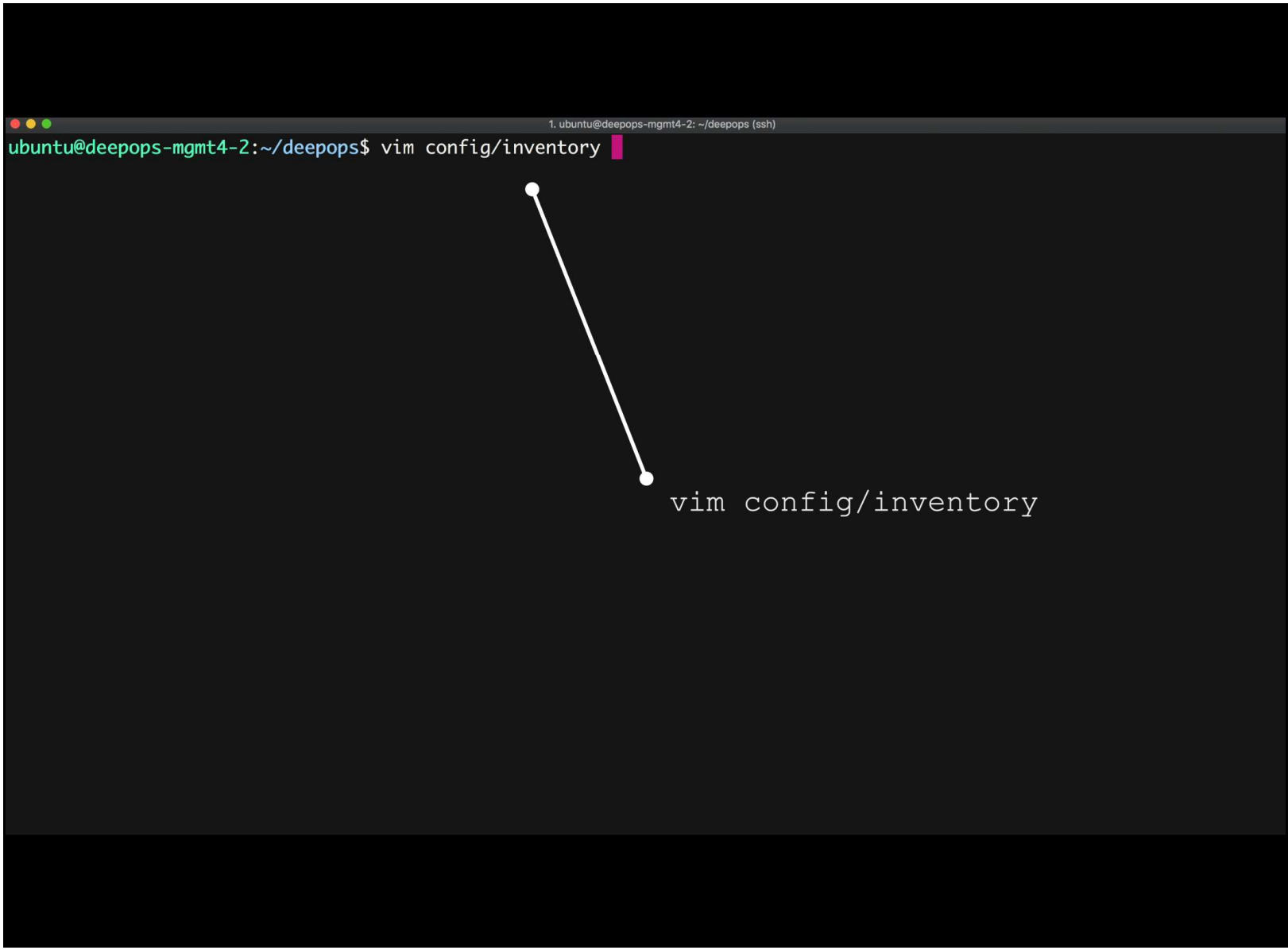
Automation: Ansible

- Open-source **automation and configuration management tool**
- **Agentless**
  (nothing to install on target nodes)
- **Idempotent**
  (run the same task over-and-over again without any repercussions)
- **Easier to maintain & scale** than custom scripts
- Playbooks use **YAML**: easy to learn and read



Prep the provisioning node:

3. Configure Ansible Inventory
4. Configure Cluster parameters



GPU Compute node(s):
- For high-performance compute workloads

Management node(s):
- Used for cluster management

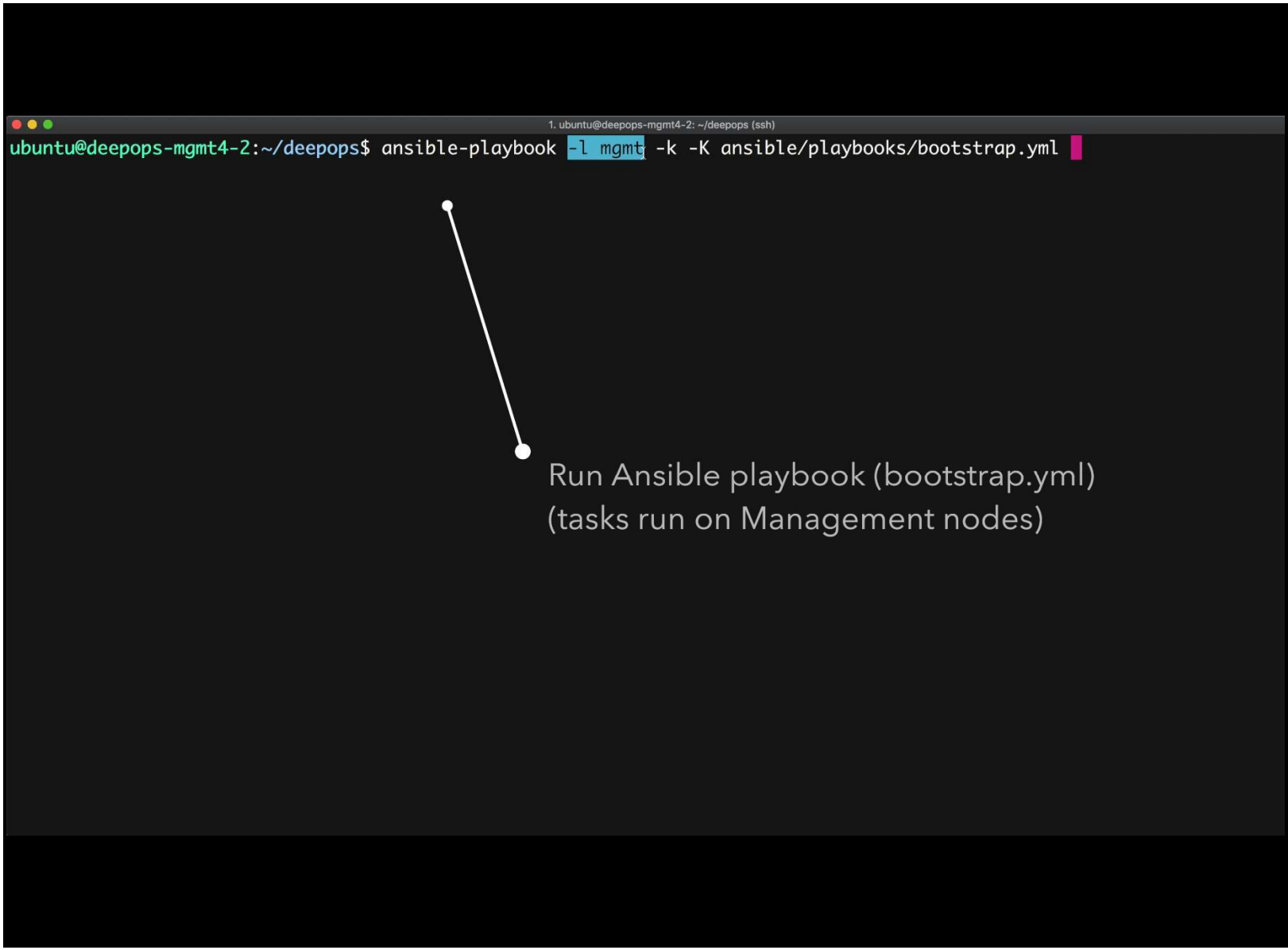
Provisioning node:
- Orchestrates the initial setup of the cluster

Prepare provisioning node

Prepare Management Node(s)

Prep management node(s):

1. **Bootstrap management node(s) with Ansible**



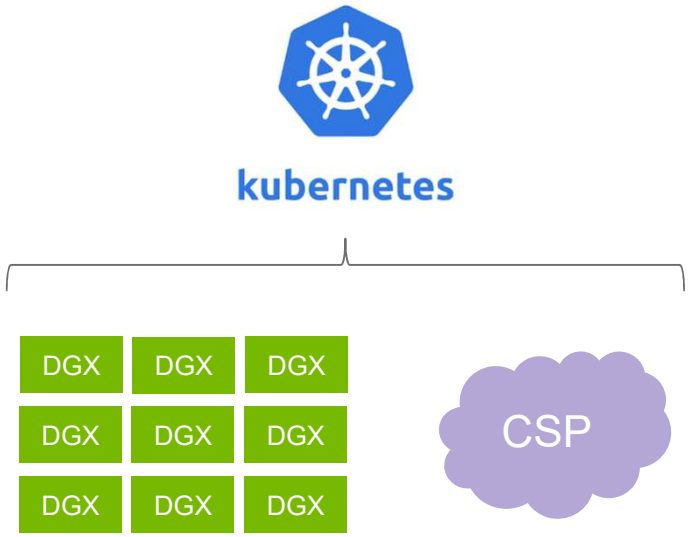
- **We will deploy a bunch of cluster services on our Management node(s).**

  For example:
  - Monitoring
  - Logging
  - PXE server
  - …

- **How can we make the services resilient, and easy to deploy & manage?**

Kubernetes

What is it?



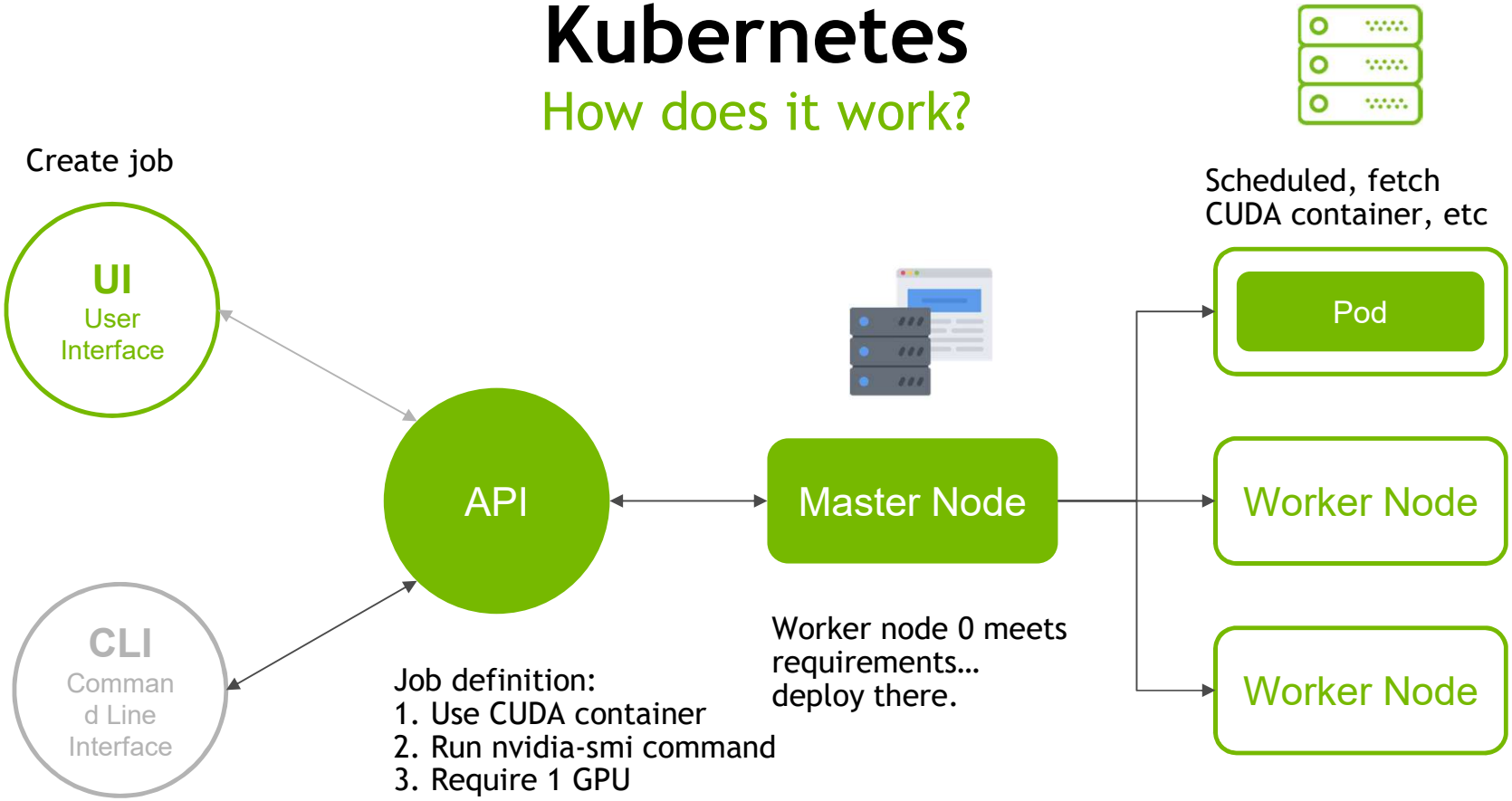
Think of Kubernetes as an operating system for a cluster

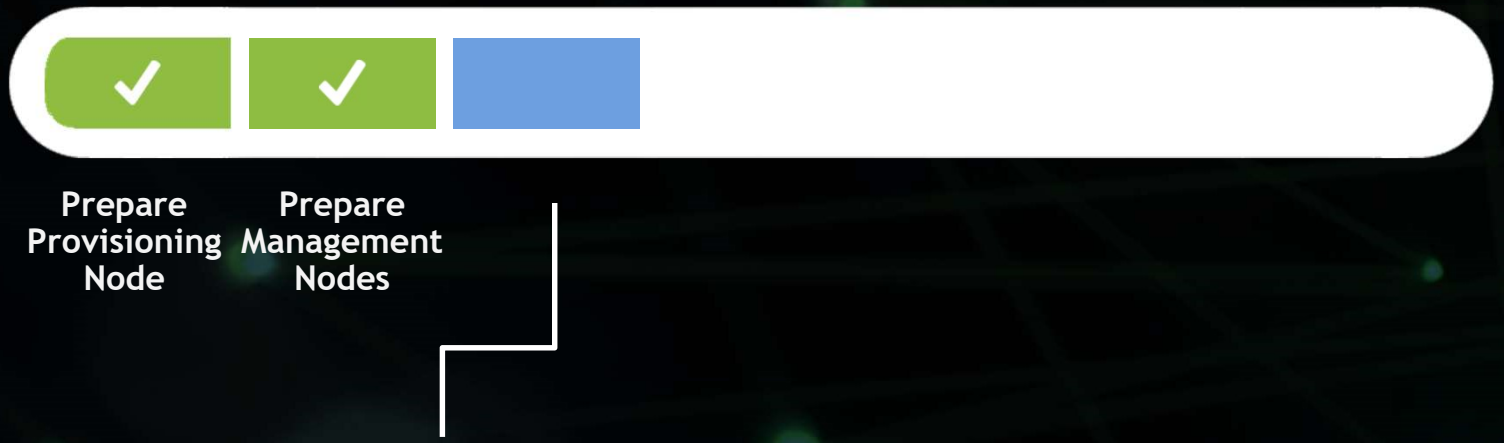
The cluster's servers can be on-prem, in the public cloud, or a mix (hybrid)

Use Kubernetes to manage nodes in the cluster, administer user access, launch jobs as containers, expose running services externally, and more

Kubernetes

How does it work?

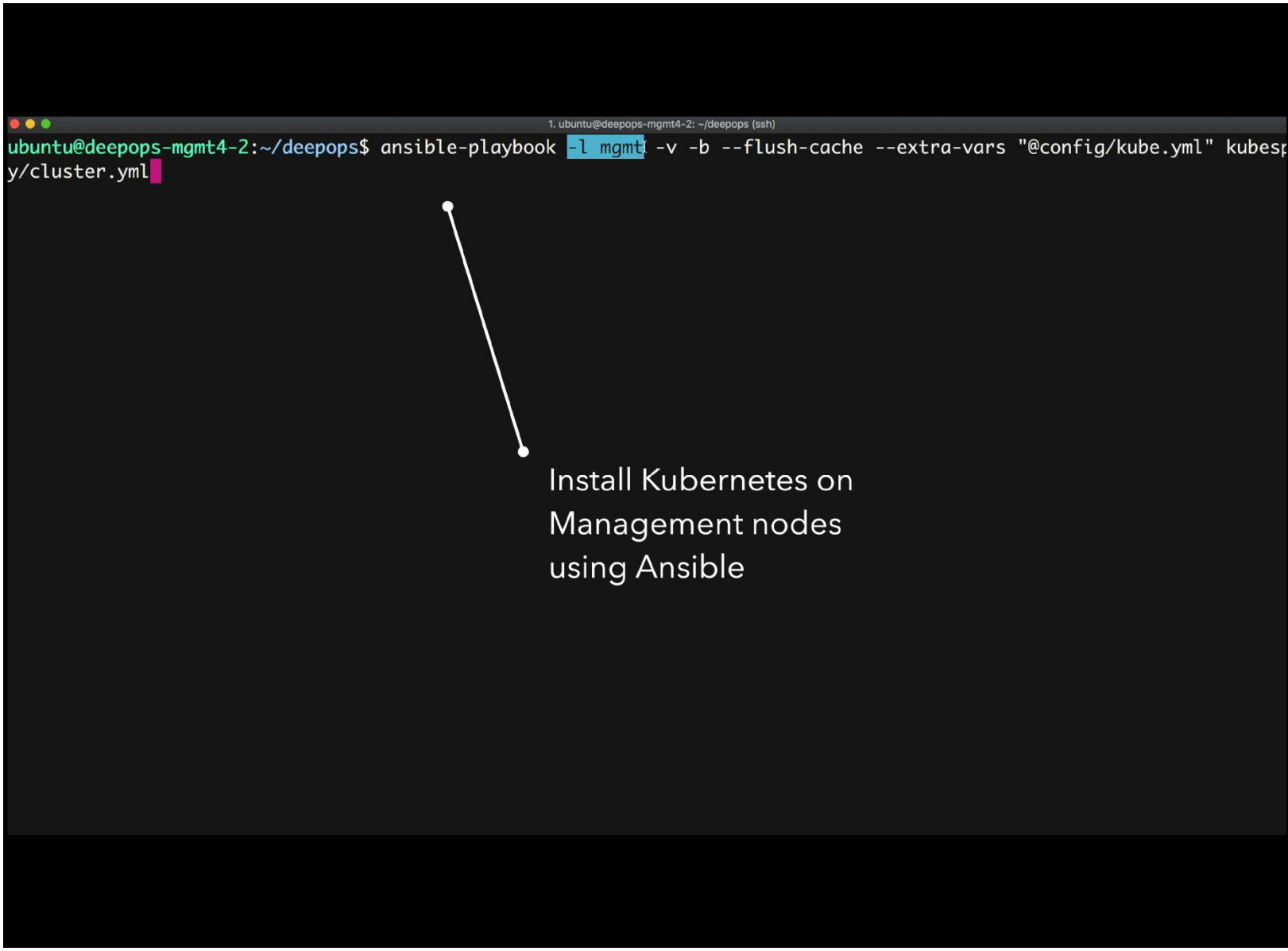




Deploy Kubernetes on Management Node(s)

Prep management node(s):

2. **Deploy Kubernetes on Management node(s)**



Deploy Basic Cluster Services

What are those initial basic services?

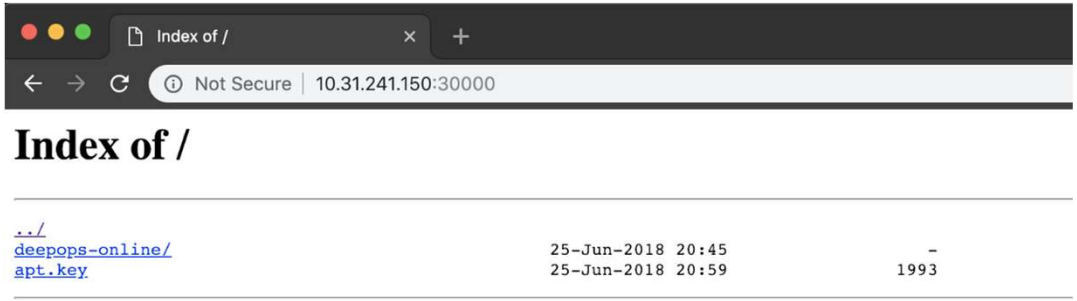
- Firmware Management for DGX-Servers
- Internal Package Repository
- Internal Docker Registry
- Scheduler
- ...

- **Optional:** DGXIE (DHCP, DNS, PXE-Server) for OS management

Deploying Internal Package Repository

For Air-Gapped Environments

```
$ kubectl apply -f services/apt.yml
```



Deploying Internal Docker Registry

For Air-Gapped Environments

```
$ helm repo add stable https://kubernetes-charts.storage.googleapis.com
$ helm install --values config/registry.yml stable/docker-registry --version 1.4.3
$ ansible-playbook -k ansible/playbooks/docker.yml
```

The container registry will be available to nodes in the cluster at registry.local

```
$ docker pull busybox:latest
$ docker tag busybox:latest registry.local/busybox
$ docker push registry.local/busybox
```

GPU Compute node(s):
- For high-performance compute workloads

Management node(s):
- Used for cluster management

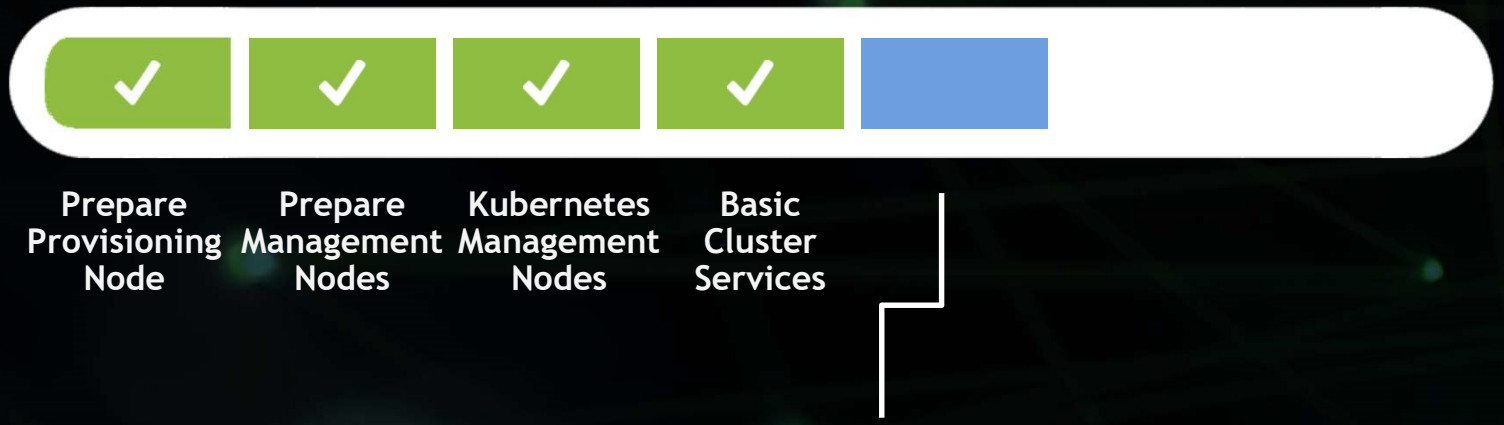
Provisioning node:
- Orchestrates the initial setup of the cluster

Deploy basic services on management node(s)

Deploy Kubernetes on management node(s)

Prepare management node(s)

Prepare provisioning node


Prepare Provisioning Node
Prepare Management Nodes
Kubernetes Management Nodes
Basic Cluster Services
Provision Compute Node(s)

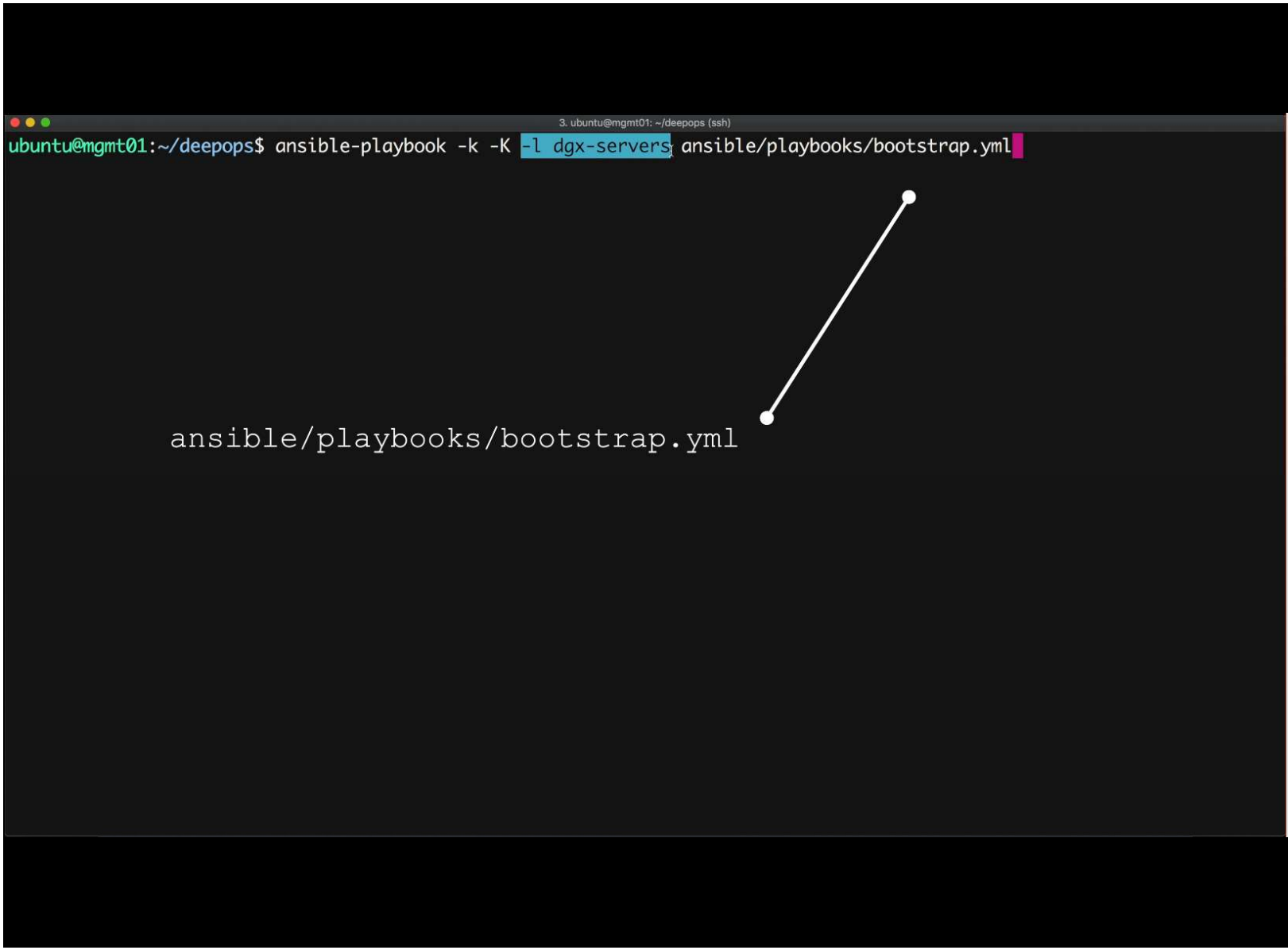
Provisioning Compute Nodes

Options

- **Manual:**
  - BMC Remote Console
  - Bootable USB
- **Automated:**
  - Third-party tools:
    - Foreman
    - MAAS
    - …
  - DeepOps:
    - OS installation container
    - Detailed steps available on GitHub: https://github.com/NVIDIA/deepops/

Provision compute node(s):

With the OS installed, we're ready to configure our nodes



Compute node(s):
- Used for computing stuff

Management node(s):
- Used for cluster management

Provisioning node:
- Orchestrates the initial setup of the cluster

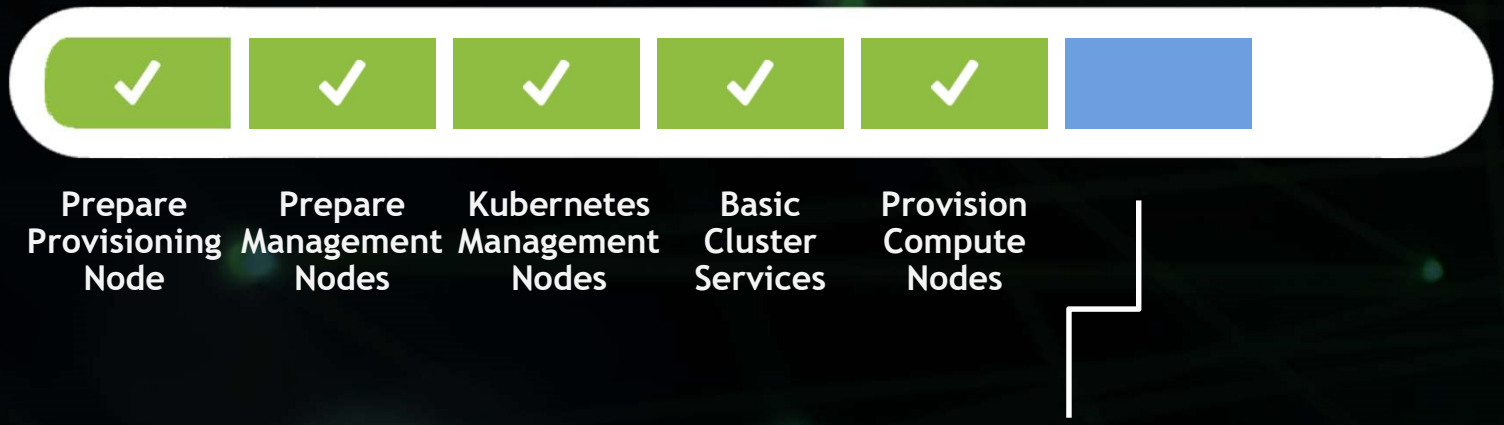
Provision compute node(s)

Deploy basic services on management node(s)

Deploy Kubernetes on management node(s)

Prepare management node(s)

Prepare provisioning node


Prepare Provisioning Node
Prepare Management Nodes
Kubernetes Management Nodes
Basic Cluster Services
Provision Compute Nodes
Deploy Additional Services

Other services:

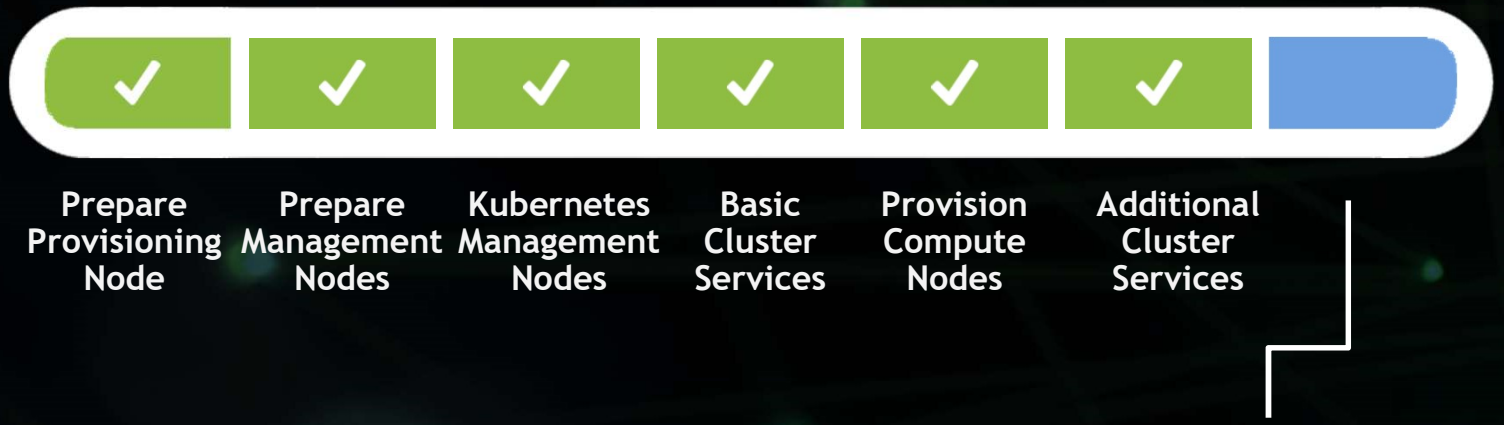
- Monitoring
- Logging
- Ingress controller
- Authentication
- …

See the DeepOps project for instructions:
https://github.com/NVIDIA/deepops

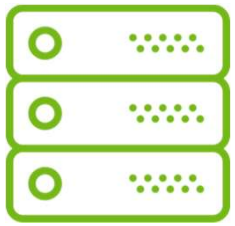
Deploy additional services:

1. Deploy monitoring service

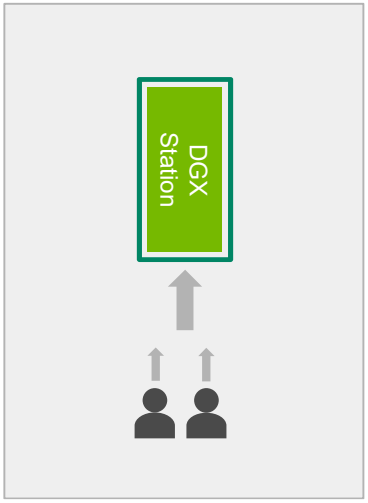
```
3. ubuntu@mgmt01: ~/deepops (ssh)
ubuntu@mgmt01:~/deepops$ cat deploy_monitoring.sh

#Deploy the monitoring and alerting stack:

helm repo add coreos https://s3-eu-west-1.amazonaws.com/coreos-charts/stable/
helm install coreos/prometheus-operator --name prometheus-operator --namespace monitoring --values config/prometheus-operator.yml
kubectl create configmap kube-prometheus-grafana-gpu --from-file=config/gpu-dashboard.json -n monitoring
helm install coreos/kube-prometheus --name kube-prometheus --namespace monitoring --values config/kube-prometheus.yml

#To collect GPU metrics, label each GPU node and deploy the DCGM Prometheus exporter:
kubectl label nodes dgx01 hardware-type=NVIDIAGPU
kubectl create -f services/dcgm-exporter.yml


ubuntu@mgmt01:~/deepops$
```



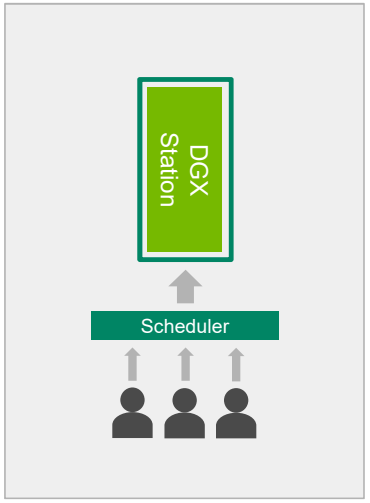
Deploy Kubernetes on Compute Node(s)



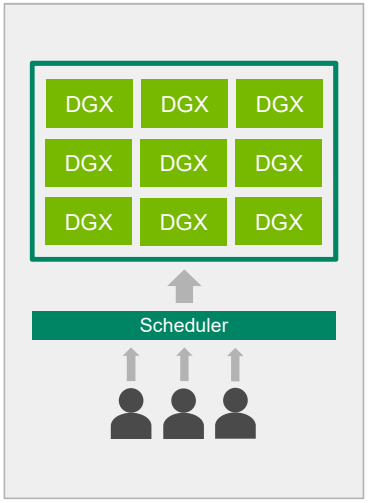
But why Kubernetes?



Many users, single node
On-prem



Many users, single node
On-prem



Many users, many nodes
On-prem

Direct Access through SSH

- Users submit jobs to scheduler & retrieve results (non-interactive; batch)
- Users request a limited interactive session through the scheduler

Scheduler

Kubernetes

Basic scheduling features

Share nodes, schedule jobs for GPUs on a node (current best solution: Excel spreadsheet)

Covers data permissions and security (LDAP, file permissions)

Adds analytics and monitoring (important also for justification of purchase)

SLURM

Advanced scheduling features

Multi-node jobs

Job dependencies, workflows, DAGs

Advanced reservations

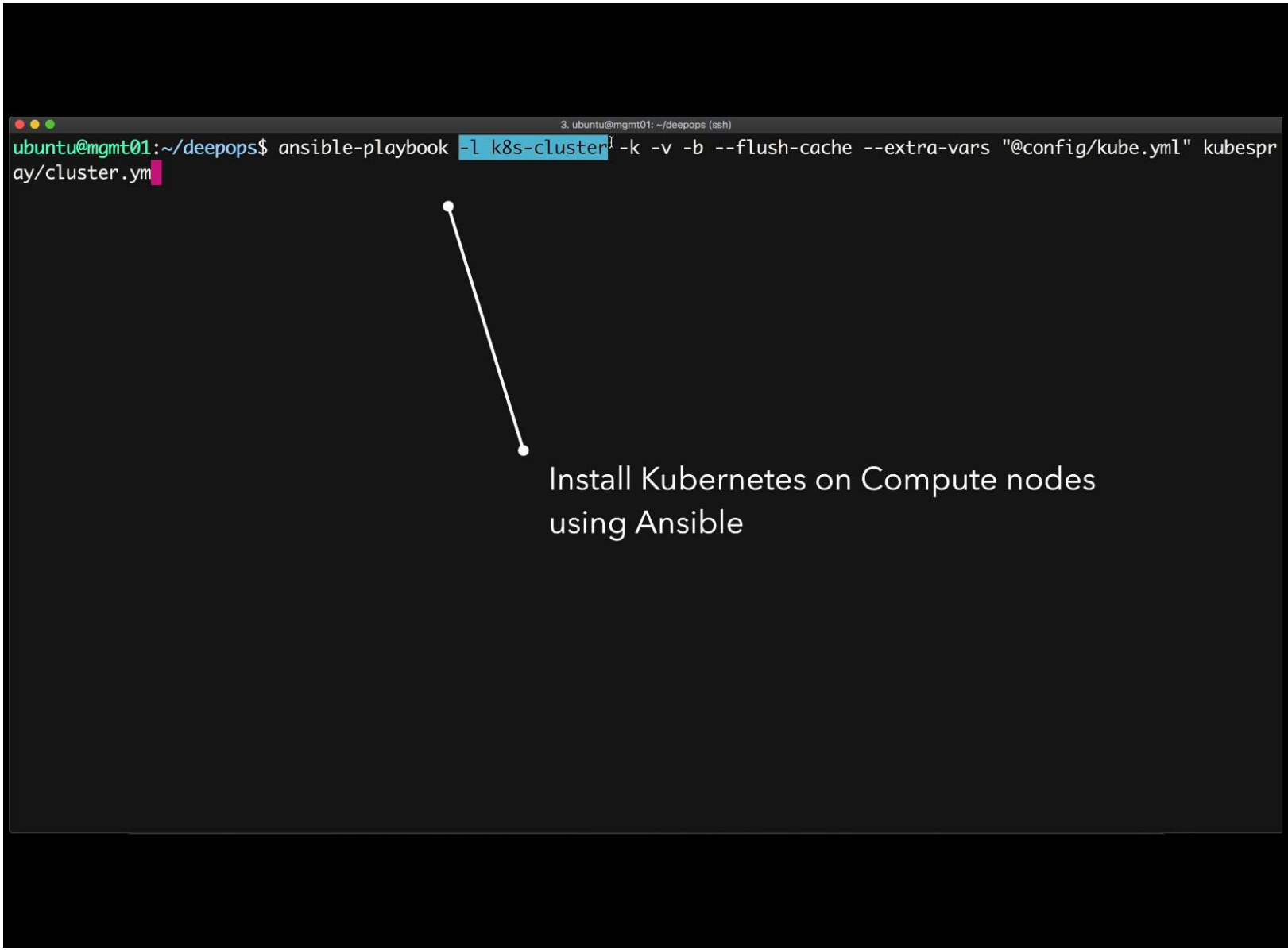
Intelligent scheduling (not just FIFO)

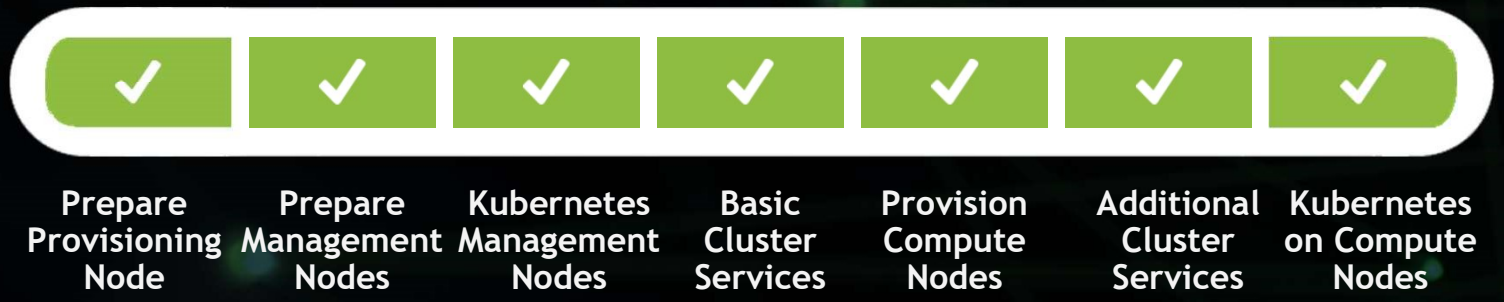
…

Other HPC-like scheduling functionality

Deploy additional services:

1. **Deploy Kubernetes on Compute node(s)**

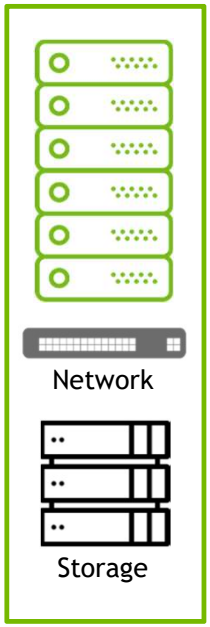



Prepare Provisioning Node
Prepare Management Nodes
Kubernetes Management Nodes
Basic Cluster Services
Provision Compute Nodes
Additional Cluster Services
Kubernetes on Compute Nodes

Submitting jobs to compute nodes

Submit a GPU-enabled job to the GPU compute node(s)

```
2. ubuntu@mgmt01: ~/deepops (ssh)
apiVersion: batch/v1
kind: Job
metadata:
  name: tensorflow-example
spec:
  backoffLimit: 5
  template:
    spec:
      containers:
        - name: tensorflow-container
          image: nvcr.io/nvidia/tensorflow:18.08-py3
          command: ["/bin/sh", "-c"]
          args:
            - mpiexec --allow-run-as-root --bind-to socket -np 8 python /opt/tensorflow/nvidia-examples/cnn/resnet.py --layers=5
0 --precision=fp16 --batch_size=256
          resources:
            limits:
              nvidia.com/gpu: 8
      restartPolicy: Never

~
~
~
~
~
~
~
~
~
"tensorflow-example.yml" 19L, 540C                                                    1,1           All
```



GPU Compute node(s):
- For high-performance compute workloads

Management node(s):
- Used for cluster management

Provisioning node:
- Orchestrates the initial setup of the cluster

Deploy Kubeflow

Deploy Kubernetes on compute node(s)

Deploy additional services on management node(s)

Provision compute node(s)

Deploy basic services on management node(s)

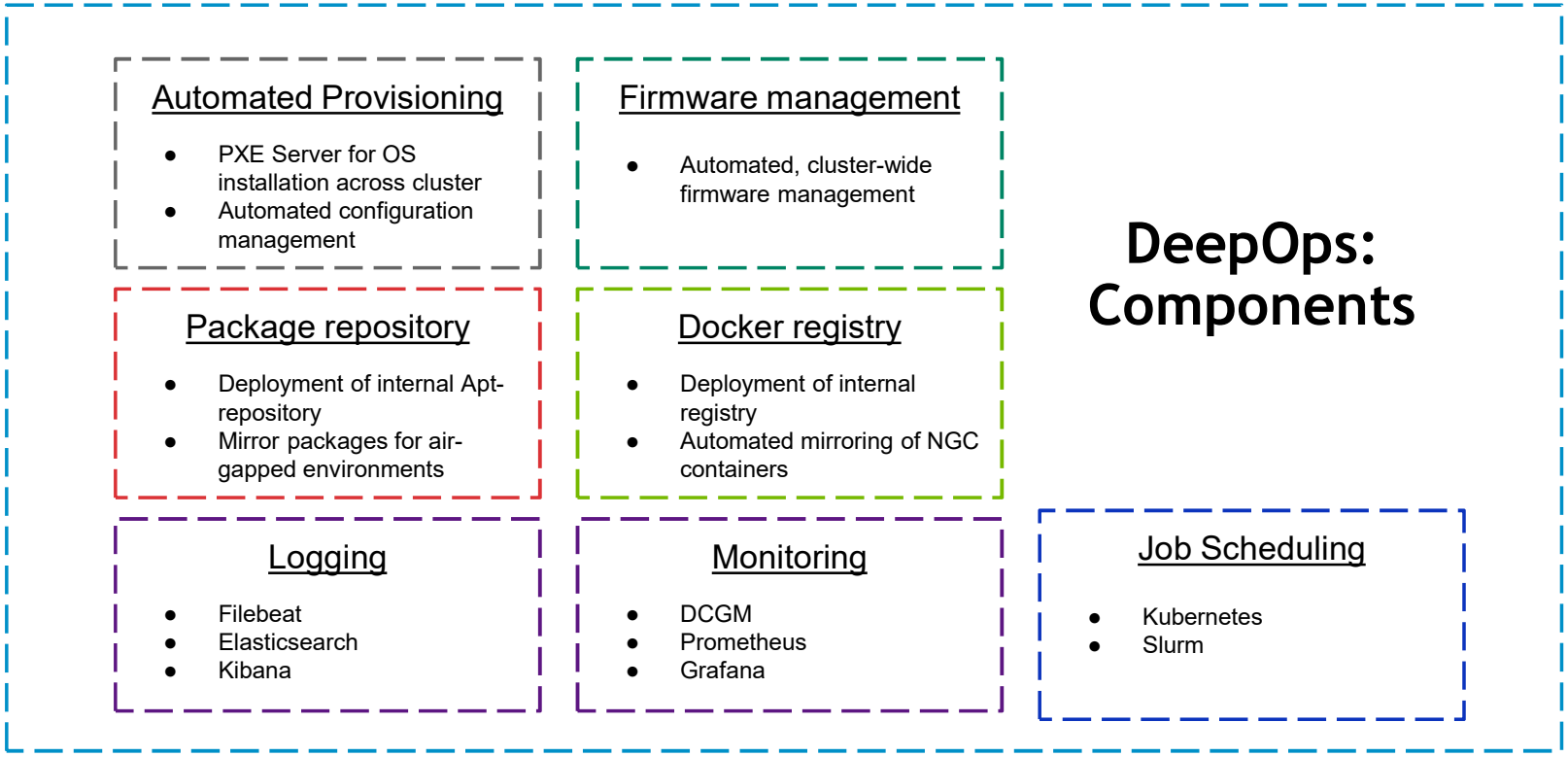
Deploy Kubernetes on management node(s)

Prepare management node(s)

Prepare provisioning node

Summary

DeepOps Components




NVIDIA