

Hyperscalers RackN Appliance

Reference Guide

Hyperscalers with RackN



Tuesday, 7 February 2023

1 TABLE OF CONTENTS

2	Introduction.....	3
3	Hyperscalers Featured Hardware.....	6
4	Features of Hyperscalers RackN Appliance	7
	For Metal as a Service Providers	7
	For Infrastructure as a Service Providers.....	8
5	Who we are?	10
6	Audience and Purpose.....	11
	Important Considerations.....	11
7	Digital IP Appliance Design Process	12
	Appliance Optimizer Utility AOU	12
8	Features of our hardware.....	13
9	Infrastructure Setup.....	14
	Terminologies.....	15
10	Base Product Deployment.....	16
11	Configure the Appliance	20
12	Testing the Appliance	26
13	Updating the Appliance	37
14	Additional Setup and Deployment.....	38
	Troubleshooting Appliance	38
15	Addendum.....	39
16	References.....	56

2 INTRODUCTION

Hyperscalers RackN Appliance solves the problem of Infrastructure Automation and Orchestration (IAO)/ Bare Metal Provisioning (BMP) and Infrastructure as a Service (IaaS)/ Infrastructure as Code (IaC) within data centres. Hyperscalers in partnership with RackN offers Digital Rebar Platform (DRP) to solve this problem and ease the management of physical hardware of Enterprises, Cloud providers and Co-location operators.

The purpose of Digital Rebar is removing the common, repetitive, non-value-added toil that most IT operations teams struggle with every day. It does not require complex components, infrastructure, and knowledge because we think that's the way it should be.

We help operations teams two ways:

1. by providing an automation catalog that covers the most common hard IT challenges like installing operating systems, updating firmware and building inventories.
2. by including a strong Infrastructure as Code process that encourages teams to reuse and standardize processes beyond our catalog

We know that a strong infrastructure foundation leaves companies to focus on the value-added work [1].

Streamlined Multi-O/S Provisioning

- Network Boot via PXE, IPXE
- Multi-Operating System – Linux, Windows, VMware
- Multi-Architecture – Intel, AMD, ARM64
- Image Based Deployment – Linux, Windows, VMware
- Immutable Operating Systems
- Boot via Media Attach
- Secure Boot Enabled [2]

Full Bare Metal Lifecycle with or without Out-of-Band Management

- BMC protocols including IPMI, Redfish, Vendor API, and none
- RAID configuration
- BIOS & Firmware configuration
- Automatic Discovery, Inventory and Classification
- VMware ESXi and VCF installation [2]

RackN Digital Rebar Platform tries to fill in the following three unique gaps left by other automation tools.

Integrating Provisioning and Configuration

Filling the gaps in the industry required RackN to create a workflow system that seamlessly blends provisioning and configuration actions throughout a components lifecycle. *For bare metal infrastructure*, that means coordinating the many protocols and management operations needed to

bootstrap, flash, and install a server into what appears to be a single process. *For virtual and cloud infrastructure*, that means normalizing and coordinating the many APIs required with post-provisioning operations such as handing off between Terraform and Ansible [3].

Reuse from Modular and Portable Infrastructure as Code (IaC)

Building an integrated workflow platform is not sufficient; we had to ensure that the automation we created could be used broadly and shared across different customers. This presented two distinct challenges. First, our target customer sites are secured so our solution needed to be portable and versioned in a way that allowed customers to download and install it themselves. Second, we had to accommodate the heterogeneous, multi-vendor nature of infrastructure in a stackable modular way that allowed customers to select just the components they needed [3].

Connecting Sites with Decentralized Management

Our *on-premises, air-gap ready* platform played a critical role in our distributed management design. Site autonomy is a customer requirement that creates resilient architectures when applied generally. Without the option of centralized control, RackN built a federated system based on loosely coupled sites that mirror each other's data. Our modular IaC system is the keystone to a model's success because it allows operators to maintain a consistent automation baseline throughout the system [3].

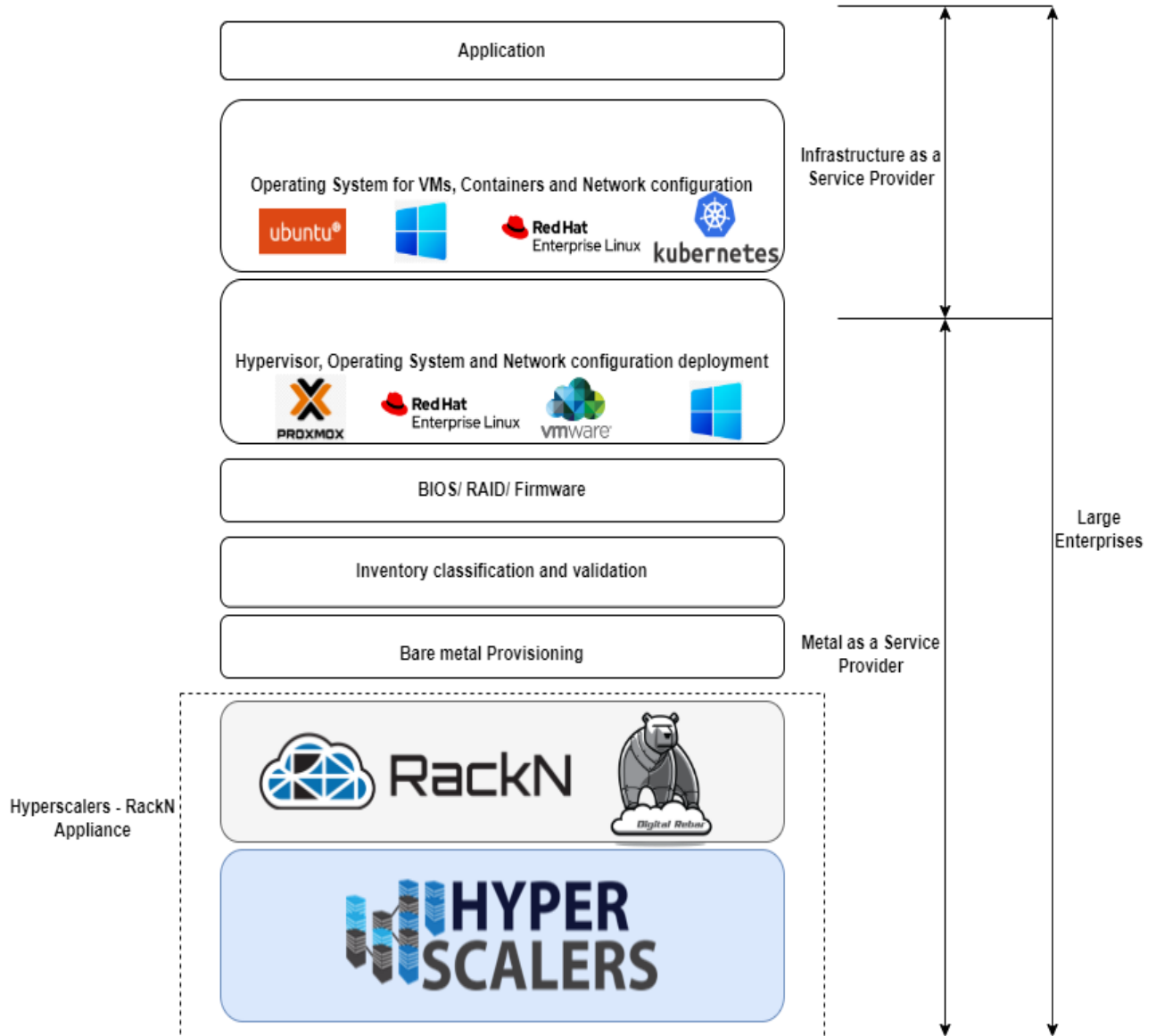


Figure 1 Hyperscalers - RackN Appliance

3 HYPERSCALERS FEATURED HARDWARE

Hyperscalers have identified the hardware configurations that can be categorised based on the customer use cases as the below. RackN appliance can be deployed in any of these high-performance servers.

S5X – 1U

S5K-1U



<p>S5X-1U [4]</p>	<p>CPU – 2 x 3rd gen Intel® Xeon Scalable processor (Up to 270W)</p> <p>Memory – Up to 8TB in 32 Slots</p> <p>System Management - Redfish v1.1, IPMI v2.0 Compliant, on board "KVM over IP" support</p> <p>Storage - 12 x 2.5" drive bays + 2 x M.2 slots (All flash)</p>
<p>S5K-1U [5]</p>	<p>CPU – 2 x AMD EPYC™ 7002/7003 Series Processors (Up to 280 W)</p> <p>Memory – Up to 4TB in 32 Slots</p> <p>System Management - Redfish v1.1, IPMI v2.0 Compliant, on board "KVM over IP" support</p> <p>Storage - 12 x 2.5" drive bays + 2 x M.2 slots (All flash)</p>

4 FEATURES OF HYPERSCALERS RACKN APPLIANCE

Hyperscalers RackN Appliance provides best in class Infrastructure Automation and Orchestration (IAO)/ Bare Metal Provisioning (BMP) and Infrastructure as a Service (IaaS)/ Infrastructure as Code (IaC) supporting a wide range of processes and systems. RackN is also able to integrate with existing MaaS by having them hand off to the RackN agent to complete installation of operating system.

For Metal as a Service Providers

RackN Features:

- BMP is only a small stage as part of system commissioning, so integration and handoffs are critical.
- BMP should not be treated as a onetime activity. Best practice data centres assume a constant refresh cycle for operating systems [6].

RackN Differentiators:

- RackN offers highly available and high scale configurations for IAO. Unlike vendor server management wrappers, our designs assume customers are driving IAO from other automated processes that rely on consistent results with accurate feedback.
- RackN small footprint can be run and managed from a top of rack switch to minimize data center overhead.
- RackN provides many performance enhancements for BMP that improve provisioning speed 10x or better.
- RackN offers image-based deployments for a wide range of O/S.
- RackN has advanced integrations for VMware ESXi that provides unmatched control of the process.
- RackN discovery and templating process simplifies BMP processes by allowing configurations to be automatically customized for each system
- RackN DHCP integrations enable improved performance and control compared to other BMP systems that rely on an external DHCP system.
- RackN supports advanced security configuration that reduce or eliminate insecure protocols during IAO.
- RackN workflows and classifiers allow systems to be provisioned without any human input when needed [6].

For Bare metal Provisioning

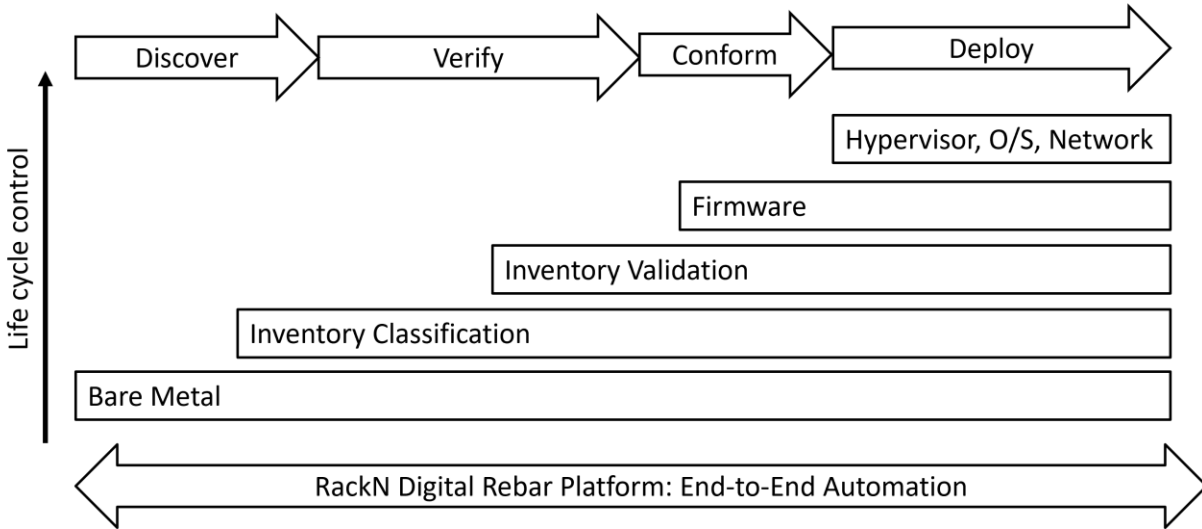


Figure 2 Generic Pipeline for Bare Metal Provisioning [7]

For Infrastructure as a Service Providers

Multi-cloud Infrastructure as a Service (IaaS)/ Infrastructure as Code (IaC) is a tooling category that allows developers to automate infrastructure building and teardown operations against cloud Application Programming Interfaces (API). This allows teams to define multisystem environments in a declarative way to be built automatically using a cloud API (aka an Infrastructure as a Service or IaaS). These tools are designed as a middle layer between an IaaS and Configuration Management (CM) systems, so they require customers to maintain both systems. While RackN provides bare metal IaaS APIs to enable IaC integrations, it also offers a complete IaC feature capability that integrates with our other control features [6].

RackN Features:

- Infrastructure as Code is an important design discipline for scale operators and is highly encouraged by RackN.
- RackN content and workflows provide all the features of an IaaS platform with significantly more control and state management.
- RackN bare metal APIs are designed for easy self-service by IaaS tools while also being highly transparent for operators to monitor and support self-service users [6].

RackN Differentiators:

- Since RackN API keeps running service, it is able to maintain the state of the system in addition to providing declarative configuration.

- RackN events and plugins allow operators to integrate IaaS actions into other workflows and notifications.
- RackN content management system is both versioned and code controlled to be an ideal IaaS format.
- The unique RackN composable architecture allows teams to collaborate on IaaS components with clear separation of duties [6].

For Infrastructure as a Service Providers

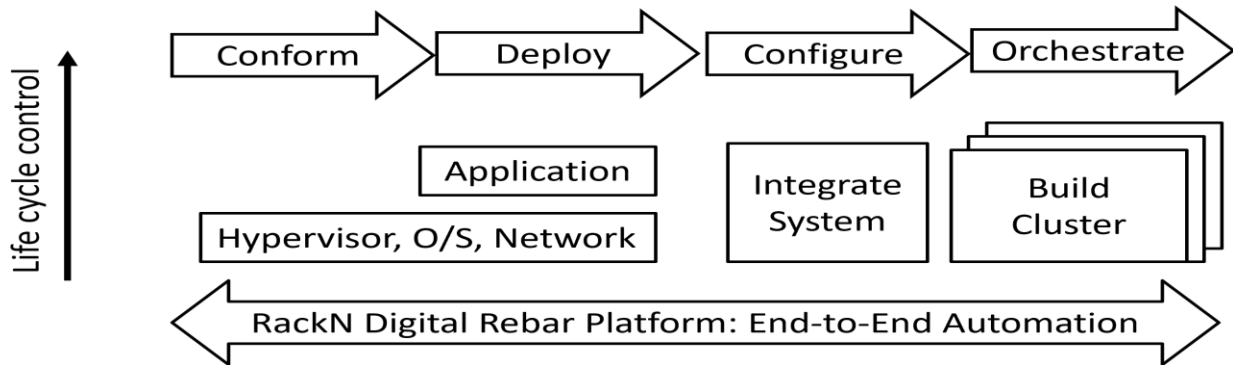


Figure 3 Generic Pipeline for Infrastructure as a Service Providers [7]

For Large Enterprises

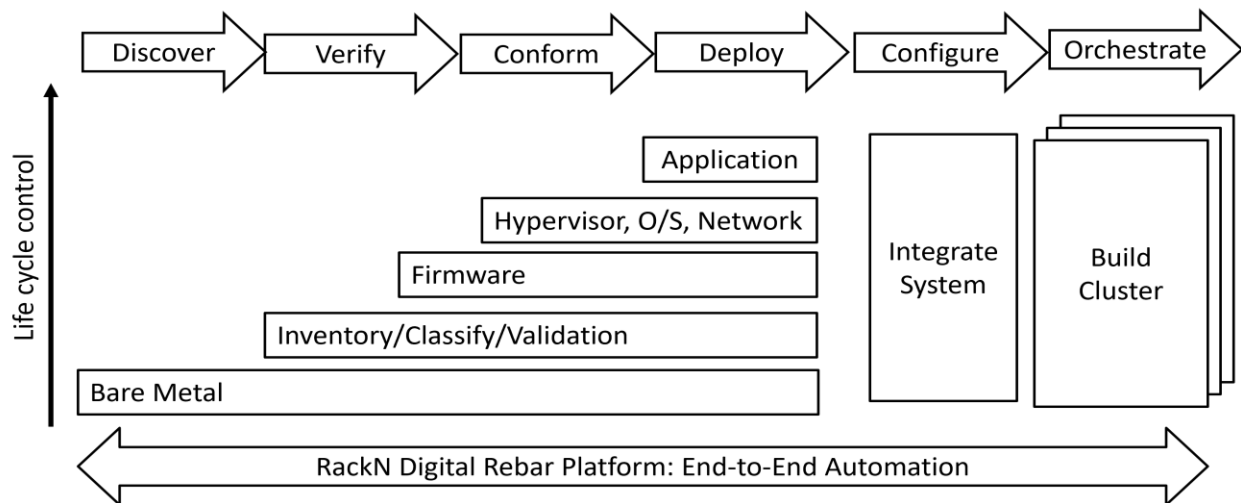


Figure 4 Generic pipeline for large enterprises [7]

5 WHO WE ARE?

Hyperscalers [8] is the world's first open supply chain Original Equipment Manufacturer- OEM, solving Information Technology challenges through standardization of best practices and hyperscale inspired practices and efficiencies. Hyperscalers offers choice across two open hardware architectures:

- Hyperscale - high efficiency open compute equipment as used by macro service providers
- Tier 1 Original – conventional equipment as per established Tier 1 OEM suppliers.

Each architecture is complete with network, compute, storage, and converged GP GPU infrastructure elements, and is open / free from vendor lock-in.

Hyperscalers' appliance solutions are packaged complete with hardware, software and pre-built (customisable) configurations. These were all pre-engineered using an in-house IP Appliance Design Process and validated in partnership with associated major software manufacturers. Many can be "test-driven" using Hyperscalers Lab as a Service (LaaS). Hyperscalers appliance solutions are ideally suited to IaaS PaaS and SaaS providers looking to implement their services from anywhere.

RackN [9] is founder-owned, profitable software company specializing in data center automation based in Austin, Texas. The RackN team has made careers in data center infrastructure and automation. We were founded by engineers who experienced the challenge of data center operations with companies like Dell, IBM, Netapp, Symantec, USMC. And we've been founders at start-ups in the infrastructure space building leading cloud, networking, and storage capabilities. Our vision comes from a belief in the transformational power of automation for completely managing the IT foundational layer (aka bare metal). We improve data center operations by making advanced techniques from cloud operators accessible at any scale site and between sites.

6 AUDIENCE AND PURPOSE

Engineers, Enthusiasts, Executives and IT professionals with background in Computer Science/ Electronics/ Information Technology with understanding in Linux commands, Python, JSON language and basic electronics who intend to study, explore, deploy Hyperscalers – RackN Appliance in Ubuntu 20.04 [10].

The purpose of this document is to create a RackN DRP appliance with various O/S images in QuantaGrid D53X-1U servers using Ubuntu 20.04 operating system [10].

Important Considerations

1. We recommend contacting Hyperscalers at info@hyperscalers.com to lay out a plan on Hardware, Network and Software Infrastructure with stability, robustness, and scalability in mind.
2. It is recommended to have a dedicated boot network for the production environment
3. In case of hardware lifecycle management, thorough testing on your use cases (Stable BIOS/ Firmware catering to your Operating system and applications) is necessary before the production deployment.

7 DIGITAL IP APPLIANCE DESIGN PROCESS

Hyperscalers has developed a Digital-IP-Appliance Design Process and associated Appliance Optimizer Utility which can enable the productization of IT-appliances for Digital-IP owners needing to hyperscale their services very quickly, reliably and at a fraction of traditional costs.

Appliance Optimizer Utility AOU

The Appliance Optimizer Utility (AOU) automates the discovery of appliance bottlenecks by pinging all layers in the proposed solution stack. A live dashboard unifies all key performance characteristics to provide a head-to-head performance assessment between all data-path layers in the appliance, as well as a comparison between holistic appliances.

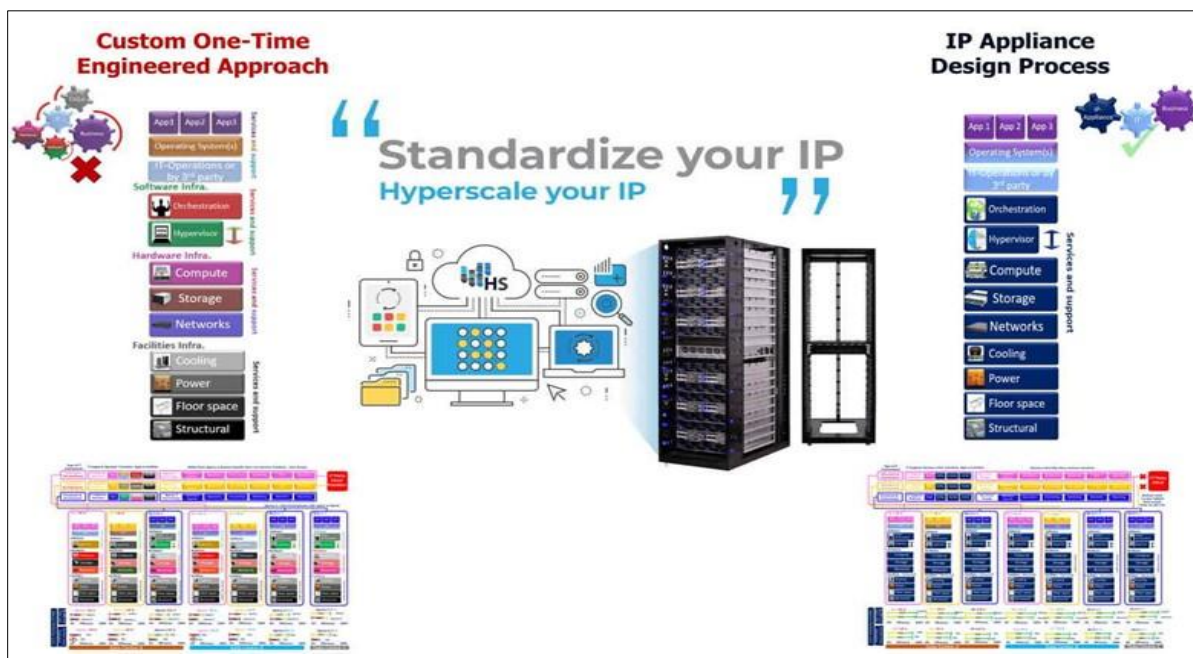


Figure 5 Digital IP-Appliance Design Process

8 FEATURES OF OUR HARDWARE

S5X -1U

The S5X 2.5" (D53X-1U) [4] based on PCIe Gen 4.0 and Intel's 3rd Generation Processor Family (Ice Lake) offers: Two (2) CPU Sockets for up to 80 cores using Intel® Xeon® Platinum 8380 Processor 40 cores each. 32 Memory slots for up to 8TB DIMM or Up to 12TB DIMM+DCPM (PMEM 200 series). 12 Front Storage drive bays 2.5" hot-plug U.2 NVMe or SATA/SAS. Five (5) x PCIe 4.0 expansions slots for Network Interface Cards NIC. Two (2) M.2 onboard storage. Three (3) accelerators like NVIDIA T4 GPU.



S5K – 1U

The S5K – 1U (D53K-1U) [5] native design for AMD EPYC™ 7003 Processors, ready for PCIe 4.0 eco-system deployment. It offers up to 128 cores within 1U form factor, optimized for HPC workloads. With 4 AMD xGMI-2 between dual EPYC™ processors up to 16GT/sec of CPU interconnect speed.



9 INFRASTRUCTURE SETUP

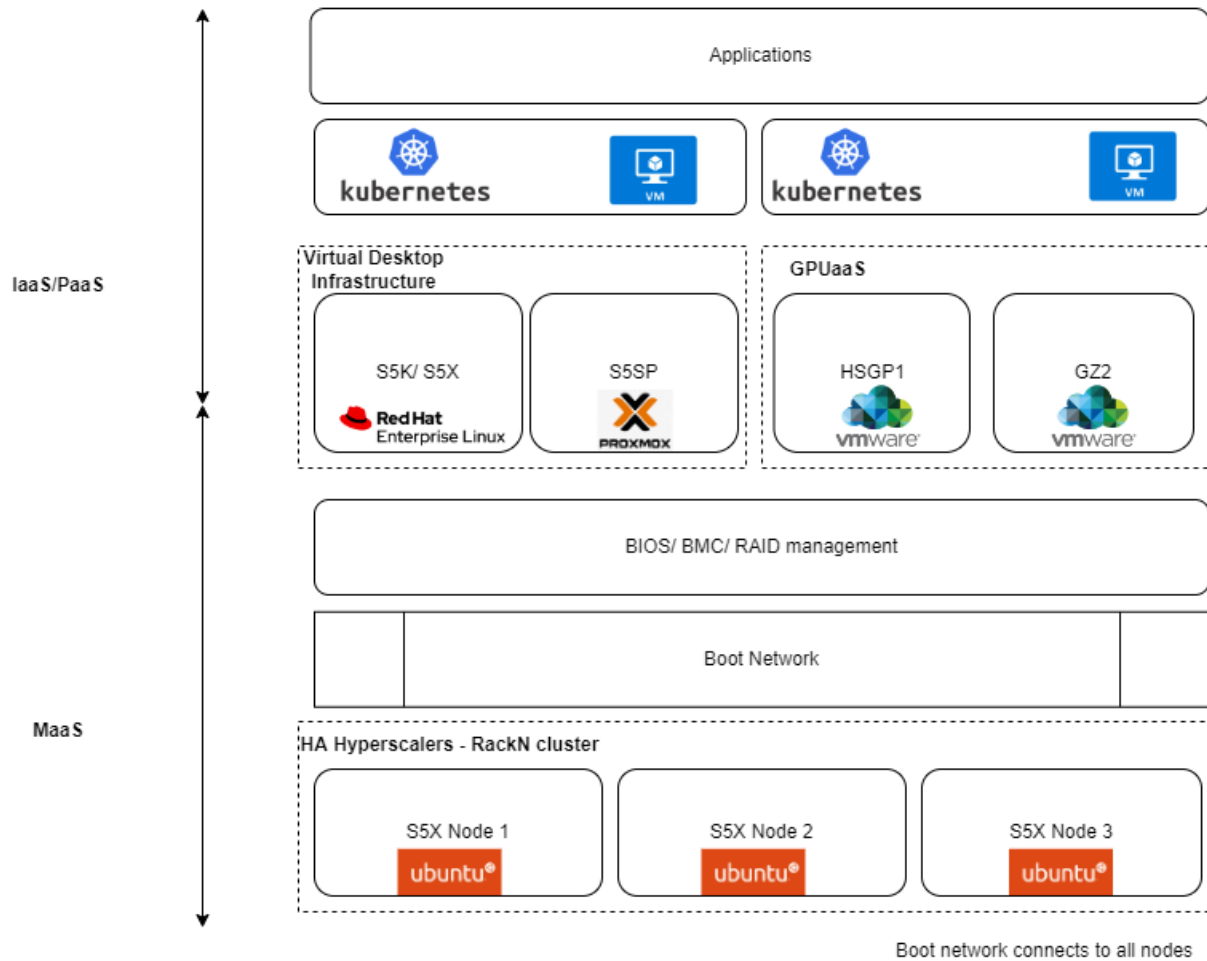


Figure 6 Hardware and software Infrastructure for MaaS, IaaS/ PaaS Providers

Server	Number of nodes	CPU	RAM	NIC Mezz	Storage Drives	OS
QuantaGrid D53X-1U (S5X) [11]	3	Intel Xeon 4310 x 1	32 GB/2933 MHz x1 unit	ConnectX 6 10/25G	Samsung 970 Evo (250 GB) x 3	Ubuntu 20.04 (5.15.0-48-generic)

Note Please consider adding additional storage for operating system images and repositories (if any) in production environment. Contact Hyperscalers for more information and strategies.

Terminologies

DRP – Digital Rebar Platform

API – Application Programming Interface

CM – Configuration Management

IaaS – Infrastructure as a Service

MaaS – Metal as a Service

PaaS – Platform as a Service

Burnin - Stress testing for system verification and validation before production deployment.

Profile – A configuration of a machine to be customised to client's needs

Pipeline – A series of workflows that will help a machine to be deployed

Workflow – A term to define the process of deployment of an operating system along with applications (if any) to a machine

Stages – Every workflow constitutes stages which will help in smooth transition towards the deployment

Tasks – Every stage consists of various tasks that need to be performed at any given stage.

Templates – Every task follows a certain template (differs between operating systems) to complete any given task (Eg: Placing SSH public keys into the machine)

HA – High Availability

DHCP - Dynamic Host Configuration Protocol

10 BASE PRODUCT DEPLOYMENT

RackN Digital Rebar Platform (DRP) endpoint can be deployed in various ways

- Bare metal
- Virtual Machine
- Cloud
- Terraform

In this document, we will be deploying the DRP endpoint to a Bare metal/ virtual machine with Ubuntu 20.04 desktop [10] installed to it.

Preinstallation Requirements

While installing Ubuntu 20.04 desktop, all the options offered by the installation window is set to default. Upon installation of the operating system, execute the following command to change firewall permissions.

```
firewall-cmd --permanent --add-port=22/tcp --add-port=8090-8092/tcp && firewall-cmd --reload  
firewall-cmd --add-port=22/tcp --add-port=8090-8092/tcp && firewall-cmd --reload
```

Networking Considerations

DHCP Broadcast Traffic

DRP provides PXE boot provisioning and DHCP IP addressing for systems. As such, your DRP Endpoint will need network connectivity via the locally connected network interfaces, and routing to any of the network subnets that you wish to provide Provisioning services for. As long as standard IP connectivity and reachability can be accomplished, DRP should be able to successfully provision systems.

If DRP is the IP Address Management (IPAM) service for your systems via the built-in DHCP server, you must also ensure IP Helpers or DHCP Relay options on your network switches/routers are correctly set up to forward Broadcast DHCP traffic to the DRP Endpoint IP address. This is a standard requirement for any DHCP and PXE provisioning system and is not unique to DRP.

IP Helper / DHCP Relay issues must be addressed both in Physical environments and Virtual Environments. Ultimately, your switch (be it virtual or physical) must forward Broadcast DHCP traffic to the DRP Endpoint [12].

Interface Speed and Duplex

DRP's communication to support Workflow does not require much bandwidth. Generally, these control messages are small packets, and do not consume much network resources. However, the

act of provisioning systems will require bandwidth dependent on the number of systems being provisioned in parallel, and the size of the provisioning artifacts (packages, images, etc). If these resources are hosted on the DRP Endpoints web service (the standard configuration), then you will need to consider these activities in your sizing.

For small lab and test environments, a single 1 Gbps full duplex network link is likely sufficient. For larger production environments, we recommend 2x 10 Gbps bonded links for both bandwidth, and reliability [12].

Provisioning -vs- Baseboard Management Network

The Baseboard Management Network (BMC), often times referred to inaccurately as the “IPMI” network, provides out-of-band control path and management of the physical machines in your environment. Typically, these networks are isolated from production network traffic. DRP can be configured to interact with, and control physical machine hardware via the BMC. The only requirement that DRP has is network reachability to the IP addresses of the BMC systems themselves.

To accomplish this, the DRP Endpoint can be “multi-homed”, or connected via NICs/Network Segments to both the in-band provisioning network, and the out-of-band provisioning network. Alternatively, a single network interface will suffice, if that network routes to the BMC interfaces. In this case, it would be prudent to ensure that Firewall or Access Control List rules block access to unknown systems. If firewall/Access Control Lists are in place, ensure that your DRP Endpoint(s) IP addresses are whitelisted/allowed access [12].

Multiple Network Connections

It is not necessary to add multiple network interfaces on the DRP endpoint to each network, assuming that the network switches and/or routers are appropriately forwarding DHCP Broadcast traffic to the DRP Endpoint.

If you do have multiple network connections on the DRP endpoint, it is critical that you evaluate your networks Layer 3 (routed) topology and ensure there are not asymmetric routing issues. If a packet ingresses one interface, but the DRP Endpoints operating system routing rules forward replies out a different interface, this will almost always break provisioning services [12].

You may need to install/use IP Policy Based Routing (PBR) rules on the DRP Endpoint host operating system to insure inbound/outbound routing of traffic conforms correctly to the required network paths.

DHCP Services / IP Addressing

The DRP Endpoint services is also a full fledged DHCP server. DRP works extremely hard to provide clean, fast, and accurate DHCP services that are tightly integrated with the Provisioning process. The service is also designed to reduce as much complexity as possible in the setup, operation, and runtime of the DHCP services. We encourage and recommend that customers use the DRP based DHCP services whenever possible.

However, in environments with existing (legacy) based DHCP services, or with very complex network topologies or hardware, it may not be possible to use DRP's DHCP services. DRP provisioning does support use of external DHCP services. The basic mechanisms of "nextserver" and "bootfile" configurations must be setup correctly in the external DHCP server. That configuration is generally extremely specific to the hardware and the DHCP server implementation. Please consult your documentation on how to forward the DHCP PXE traffic appropriately.

DRP controls its internal DHCP services via the definition of Subnets which define the start and ending ranges for IP Address handout during the DHCP negotiations (often times referred to as DORA). The DHCP server will NOT interfere with other DHCP traffic, as long as a Subnet for that Layer 3 network is not configured.

Ultimately - there can only be ONE authoritative DHCP server of record for a Layer 3 Subnet/Network. You must ensure that there are no other competing DHCP servers or services on the network, otherwise provisioning activities will likely fail.

DRP does support providing Proxy DHCP responses for limited DHCP servers that do not understand how to provide the PXE required nextserver and bootfile [12].

Provisioning Targets (Machines) Requirements

Physical Machine Requirements

The vast majority of physical hardware that is provisioned and managed with DRP are server class systems, with a Baseboard Management Controller (BMC, iDRAC, iLO, XCC, etc.). Typically, you will want to set these systems in BIOS to boot PXE first on the primary NIC that you designate as your in-band provisioning network. Systems with a BMC are not required, as long as they can be set to PXE boot, in those cases, DRP can take control via the PXE boot path, and in-band reboots of the system. External power control will have to be used/implemented outside of DRP in these cases

DRP is capable of managing switches (via ONIE/ZTP boot/install control), and storage devices. For these devices, please contact Hyperscalers/RackN for further details [12].

Using Virtual Machines

DRP will provision Virtual Machines equally well as physical hardware. Similar to physical hardware, the Virtual Machine vBIOS boot order needs to be configured with PXE boot first.

If you are utilizing virtual machines, you are generally free to size your VMs to whatever virtual hardware sizes you need. However, note that some Operating Systems that you might provision will have requirements that may dictate the lower bounds of your VMs sizing configuration.

For example, CentOS/Redhat VMs should be configured with at least 2 GB of vMemory. This is a requirement of the CentOS installation and dracut tooling. In general, we would recommend not configuring vMemory on your Virtual Machines with less than 2 GB as a safety precaution [12].

Installation Components

Deployment

To deploy stable version of RackN Digital Rebar Platform, execute the following command in an elevated terminal [13]

```
sudo apt update  
sudo apt install curl  
curl -fsSL get.rebar.digital/stable | sudo bash -s -- install -universal
```

A sample output to the above `curl` command is attached to the Addendum section of this document. Our deployment used v4.10.8 of DRP endpoint and v4.10.7 of UX.

11 CONFIGURE THE APPLIANCE

Upon deployment of the RackN Digital Rebar Platform, one can access the DRP endpoint at `https://<ip-address>:8092`. Accept the self-generated SSL certificate to access the DRP dashboard. The default username is `rocketskates` and password is `r0cketsk8ts`.

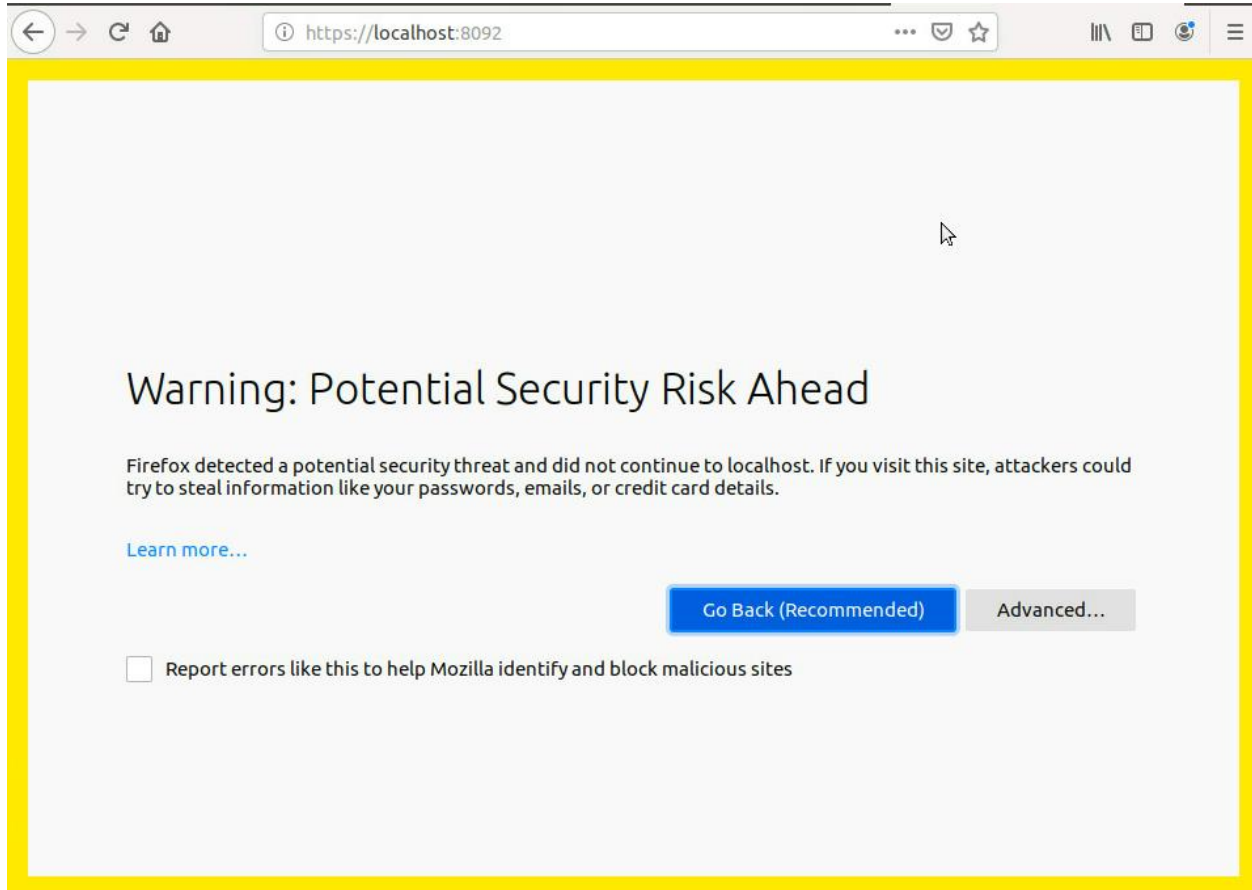


Figure 7 Initial DRP Endpoint screen

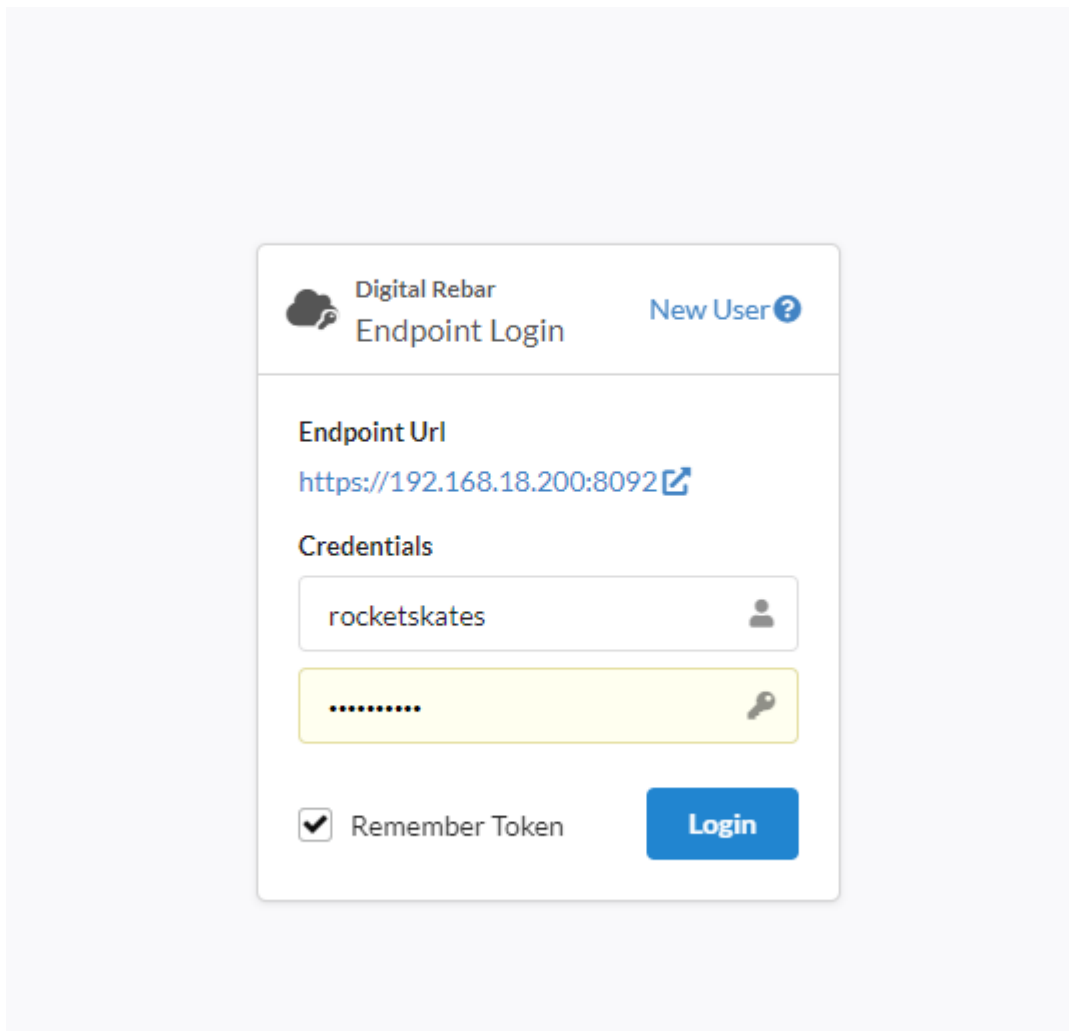


Figure 8 DRP Endpoint login

When you change the default password of the endpoint, execute the following in elevated terminal for drpcli to have access to the DRP endpoint [13]

```
export RS_ENDPOINT=https://<ip_address_of_your_endpoint>:8092/  
export RS_KEY=rocketskates:<your-password>
```

Registration

Upon logging into the RackN DRP endpoint, a registration is necessary to continue where you can upload license/ trial license (expires in 7 days) received from Hyperscalers/ RackN. Firstly, you would need to complete the tasks at System → Info and Preferences → System Bootstrap Wizard like changing default password, subnet configuration and adding administrator SSH public key to download sledgehammer (a lightweight OS based on centOS for discovery of machines) files for PXE deployment of machines.

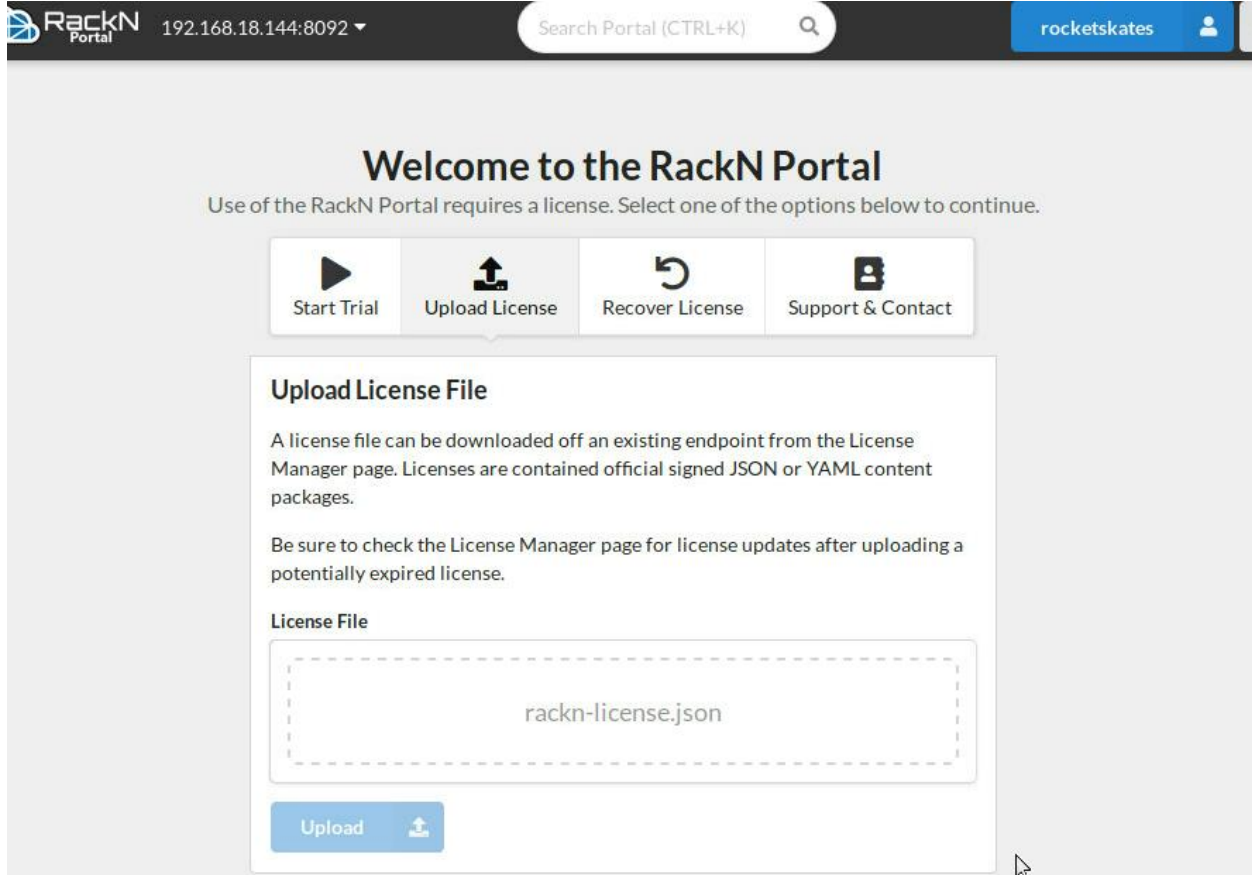
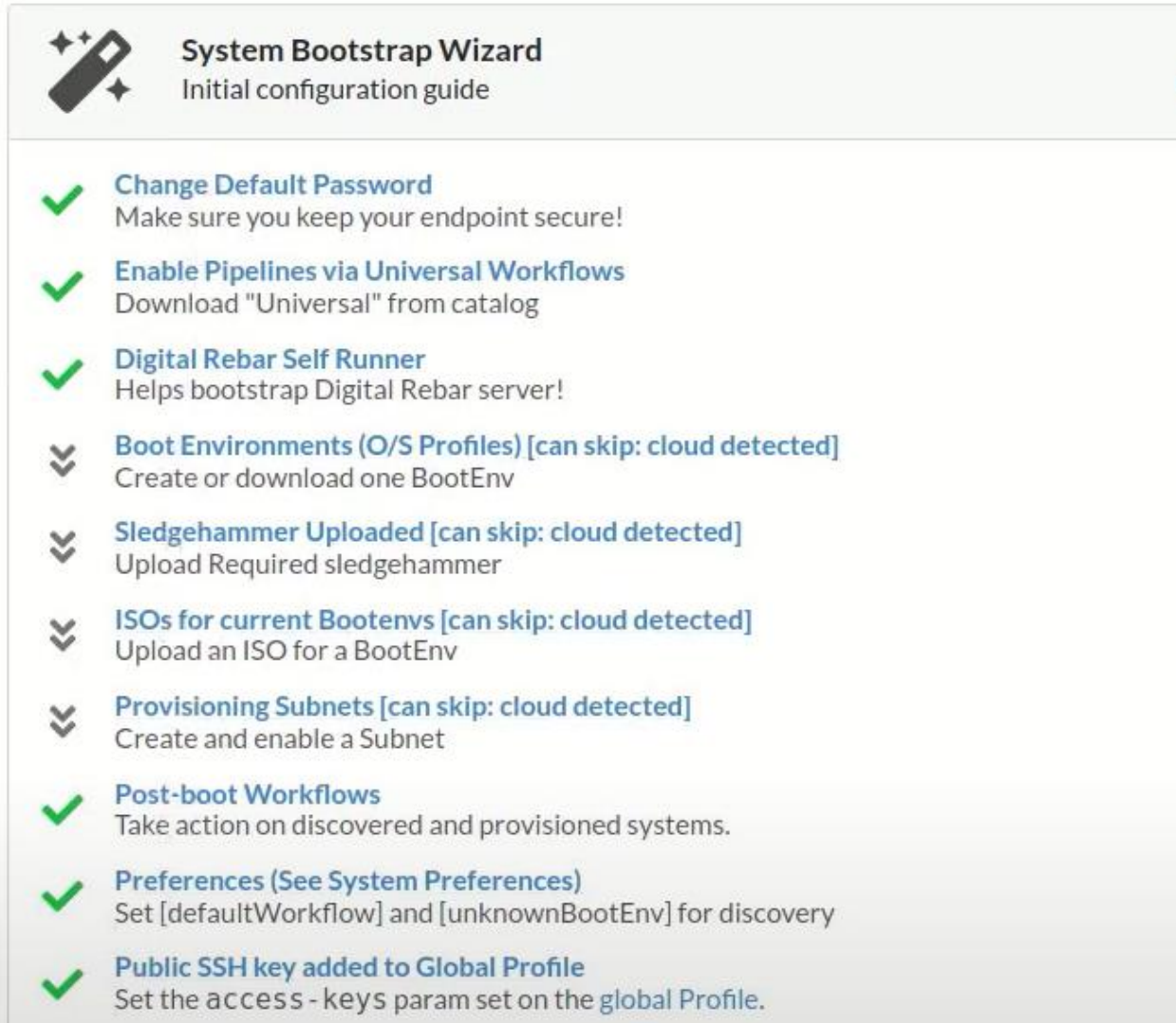


Figure 9 Upload license file for RackN DRP



The image shows a screenshot of the 'System Bootstrap Wizard' interface. At the top, there is a header with a pencil icon and the text 'System Bootstrap Wizard' and 'Initial configuration guide'. Below this, a list of tasks is displayed, each with a green checkmark icon and a blue title. The tasks are: 'Change Default Password' (Make sure you keep your endpoint secure!), 'Enable Pipelines via Universal Workflows' (Download "Universal" from catalog), 'Digital Rebar Self Runner' (Helps bootstrap Digital Rebar server!), 'Boot Environments (O/S Profiles) [can skip: cloud detected]' (Create or download one BootEnv), 'Sledgehammer Uploaded [can skip: cloud detected]' (Upload Required sledgehammer), 'ISOs for current Bootenvs [can skip: cloud detected]' (Upload an ISO for a BootEnv), 'Provisioning Subnets [can skip: cloud detected]' (Create and enable a Subnet), 'Post-boot Workflows' (Take action on discovered and provisioned systems.), 'Preferences (See System Preferences)' (Set [defaultWorkflow] and [unknownBootEnv] for discovery), and 'Public SSH key added to Global Profile' (Set the access - keys param set on the global Profile.).

- ✓ **Change Default Password**
Make sure you keep your endpoint secure!
- ✓ **Enable Pipelines via Universal Workflows**
Download "Universal" from catalog
- ✓ **Digital Rebar Self Runner**
Helps bootstrap Digital Rebar server!
- ⇓ **Boot Environments (O/S Profiles) [can skip: cloud detected]**
Create or download one BootEnv
- ⇓ **Sledgehammer Uploaded [can skip: cloud detected]**
Upload Required sledgehammer
- ⇓ **ISOs for current Bootenvs [can skip: cloud detected]**
Upload an ISO for a BootEnv
- ⇓ **Provisioning Subnets [can skip: cloud detected]**
Create and enable a Subnet
- ✓ **Post-boot Workflows**
Take action on discovered and provisioned systems.
- ✓ **Preferences (See System Preferences)**
Set [defaultWorkflow] and [unknownBootEnv] for discovery
- ✓ **Public SSH key added to Global Profile**
Set the access - keys param set on the global Profile.

Figure 10 System Bootstrap Wizard

Subnet Configuration

The DRP endpoint will need a range of IPs to be served in its DHCP server, which will be handed out to the machines while PXE booting them in `universal-discover` workflow.

✕ ens3 Locally managed

🔧 Editor ℹ️ Meta 📄 API

The Subnet Object defines the configuration of a single subnet for the DHCP server to process. Multiple subnets are allowed. The Subnet can represent a local subnet attached to a local interface (Broadcast Subnet) to the Digital Rebar Provision server or a subnet that is being forwarded or relayed (Relayed Subnet) to the Digital Rebar Provision server.

name	Reserved Time
500	21

DESCRIPTION

Definition

SUBNET ℹ️
READ ONLY

ACTIVE START ℹ️

ACTIVE END ℹ️

ACTIVE LEASE TIME ℹ️

RESERVED LEASE TIME ℹ️

Configuration

STRATEGY ℹ️

NEXT SERVER ℹ️

ONLY RESERVATIONS ℹ️

ENABLED ℹ️

UNMANAGED ℹ️

PROXY ℹ️

DHCP Options [More Options](#)

SUBNET MASK ℹ️

Activate Windows
Go to Settings to activate Windows.

Figure 11 Subnet Configuration

System Bootstrap Wizard
Initial configuration guide Cloud: N/A

Bootstrapping Completed
Ready for advanced content!

Figure 12 Final view of System Bootstrap Wizard

High Availability

In our deployment, we will be deploying 3 stand-alone DRP endpoints and associating them to behave with High Availability. In order to build strategies towards complex scenarios like network configuration and load balancing, contact Hyperscalers.

Prerequisites for High Availability

There are a few conditions that need to be met in order to set up an HA cluster of dr-provision nodes:

1. A fast network between the nodes you want to run dr-provision on. Data replication between HA nodes uses synchronous log replay and file transfer, so a fast network (at least gigabit Ethernet) is required to not induce too much lag into the system.
2. Enough storage space on each node to store a complete copy of all replicated data. dr-provision will wind up replicating all file, ISO, job log, content bundle, plugin, and writable data to all nodes participating in the HA cluster. The underlying storage should be fast enough that write speeds do not become a bottleneck – we recommend backing the data with a fast SSD or NVMe device.
3. A high-availability entitlement in your license. High-availability is a licensed enterprise feature. If you are not sure if your license includes high-availability support, contact info@hyperscalers.com.
4. All endpoints and the cluster ID must be registered in your license. High-availability is a licensed enterprise feature. If you are not sure if your license includes high-availability support, contact info@hyperscalers.com.
5. A virtual IP address that client and external traffic can be directed to. If using dr-provision's internal IL address management (i.e., not using the `-load-balanced` command line option), dr-provision will handle adding and removing the virtual IP from a specified interface and sending out gratuitous ARP packets on failover events, and the nodes forming the HA cluster must be on the same layer 2 network. If using an external load balancer, then the virtual IP must point to the load balancer, and that address will be used by everything outside of the cluster to communicate with whichever cluster node is the active one [13].

Self-enroll the initial active node

To start the initial active node, you can use the `drpcli system ha enroll` command to have it enroll itself. The form of the command to run is as follows:

```
drpcli system ha enroll $RS_ENDPOINT username password \  
  ConsensusAddr address:port \  
  Observer true/false \  
  VirtInterface interface \  
  VirtInterfaceScript /path/to/script \  
  HaID ha-identifier \  
  LoadBalanced true/false \  
  VirtAddr virtualaddr
```

The last 3 of those settings can only be specified during self-enroll, and even then, they can only be specified if the system you are self-enrolling is not already in a synchronous replication cluster.

You also can only specify `VirtInterface` and `VirtInterfaceScript` if `LoadBalanced` is false.

If any errors are returned during that call, they should be addressed, and the command retried. Once the command finished without error, the chosen system will be in a single node Raft cluster that is ready to have other nodes added to the cluster.

Adding additional nodes

To add additional nodes to an existing cluster, you also use `drpcli system ha enroll` against the current active node in that cluster:

```
drpcli system ha enroll https://ApiURL_of_target target_username target_password \  
  ConsensusAddr address:port \  
  Observer true/false \  
  VirtInterface interface \  
  VirtInterfaceScript /path/to/script
```

This will get the global HA settings from the active node in the cluster, merge those settings with the per-node settings from the target node and the rest of the settings passed in on the command line, and direct the target node to join the cluster using the merged configuration.

NOTE The current data on the target node will be backed up, and once the target node has joined the cluster it will mirror all data from the existing cluster. All backed up data will be inaccessible from that point.

12 TESTING THE APPLIANCE

The testing of the appliance was carried out in bare metal machines with the view of complete functionality of the operating system upon deployment, control with DRP endpoint at any given time, repurposing the hardware and deploying another workflow to it.

Infrastructure Pipelines built from Workflows drive this style.

1. `universal-discover` Workflow is main entry point for hardware
2. `universal-start` Workflow is main entry point for cloud-based resources

An example Pipeline would involve:

1. Discovers/Inventories/Classifies machines
2. Configures hardware (physical)
3. Installs OS (physical / some virtual)
4. Configures OS
5. Deploys applications

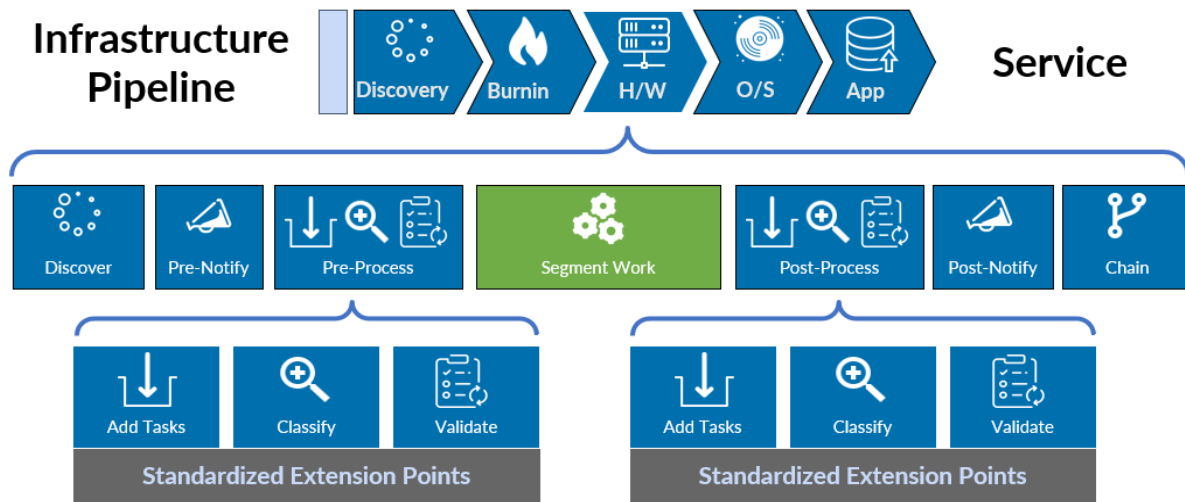


Figure 13 Generic deployment pipeline

Default username and Passwords

use	username	password
drp endpoint auth	rocketskates	r0cketsk8ts
sledgehammer	root	rebar1
most bootenvs (*)	root	RocketSkates
debian / ubuntu	rocketskates	RocketSkates
cloud-init images	<varies> (*)	RocketSkates
VMware ESXi (**)	root	RocketSkates

Note

(*) “most bootenvs” and cloud-init images refers to CentOS, Ubuntu, CoreOS, ESXi, etc. Generally speaking, this is the default “installed” credentials. Note that each distro has its own rules about root versus installed default user accounts. DRP follows most vendors “patterns” with regards to root -vs- unprivileged user creation, with the username changed to “rocketskates”. Some vendor specific notes are below.

(**) ESXi passwords by default will be assigned the r0cketsk8ts password. However, if the Param esxi/generate-random-password is set to true, then a randomly generated password will be used, and recorded on the Machine object Param named esxi/insecure-password [13].

Baremetal discovery

The following flowchart shows the bare metal universal discovery process which involves picking up hardware data through IPMI and BIOS and placing the hardware in “Ready” state to accept our

target operating systems. The workflows can be chained to create a definitive pipeline to set up an environment with specific set of client's applications in client's preferred operating system.

Baremetal discovery

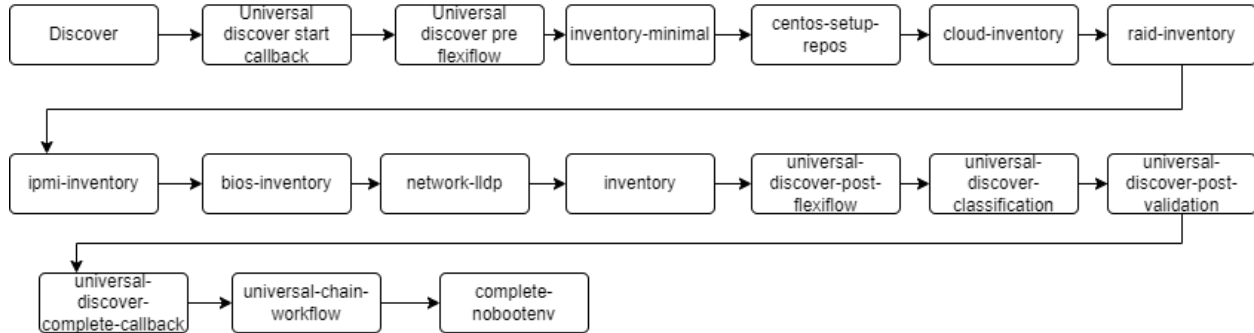


Figure 14 Baremetal discovery flowchart

The following images shows various machines taking up various operating systems as their target operating system and deploy them through “universal-application” or “universal-image-deploy” or custom workflows to setup RAID, Network, and application stack in the operating system.

During image-build phase, one can use the existing workflow within RackN (You might need to enable image-build and image-deploy in Catalog) or clone the existing workflows and add additional stages (To install packages and applications) that can be deployed as an image (All the packages will be installed with chroot root filesystem).

d08-00-27-c8-6c-9c.hyperscalers.com Locally managed

Lock Object Start Workflow

Editor Params Profiles Meta Activity API

IP ADDRESS: 192.168.18.11

Collections

PROFILES: universal-application-ubuntu-18.04

POOL: default Allocate Status: Free

CLUSTER: Empty

Operational Controls

RUNNABLE:

WORK ORDER MODE:

CONTEXT: Machine

BASE CONTEXT: Machine

Workflow Management

PIPELINE: ubuntu-18.04

WORKFLOW: ubuntu-base

BOOT ENVIRONMENT: local

STAGE (READ ONLY): complete

CURRENT JOB: Workflow Complete

TASK QUEUE:

- 1 stage:ubuntu-18.04-install Activate Windows
- 2 bootenv:ubuntu-18.04-install Go to Settings to activate Windows.
- 3 ubuntu-drp-only-repos
- 4 ssh-access
- 5 configure-network

Figure 15 Ubuntu 18.04 server workflow

fjhfb

d08-00-27-9a-65- Locally managed
Od.hyperscalers.com Lock Object Start Workflow

Editor Params Profiles Meta Activity API

IP ADDRESS 192.168.18.14

Collections

PROFILES universal-application-ubuntu-20.04

POOL default Allocate Status Free

CLUSTER Empty

Operational Controls

RUNNABLE

WORK ORDER MODE

CONTEXT Machine

BASE CONTEXT Machine

Workflow Management

PIPELINE ubuntu-20.04 -

WORKFLOW ubuntu-20.04-base -

BOOT ENVIRONMENT local

STAGE complete
READ ONLY

CURRENT JOB ✓ Workflow Complete

TASK QUEUE

- 1 stage:ubuntu-20.04-install Activate Windows
- 2 bootenv:ubuntu-20.04-install Go to Settings to activate Windows.
- 3 ssh-access
- 4 configure-network

Figure 16 Ubuntu 20.04 server workflow

d08-00-27-50-53-
1c.hyperscalers.com

Locally managed

Lock Object Start Workflow

Editor Params Profiles Meta Activity API

PROFILES universal-application-ubuntu-22.04

POOL default Status Free

CLUSTER Empty

Operational Controls

RUNNABLE

WORK ORDER MODE

CONTEXT Machine

BASE CONTEXT Machine

Workflow Management

PIPELINE ubuntu-22.04

WORKFLOW ubuntu-22.04-base

BOOT ENVIRONMENT local

STAGE READ ONLY complete

CURRENT JOB Workflow Complete

TASK QUEUE

```
1 stage:ubuntu-22.04-install
2 bootenv:ubuntu-22.04-install
3 ssh-access
4 configure-network
5 stage:drp-agent
6 drp-agent-install
7 stage:finish-install
8 bootenv:local
9 stage:complete
10 context:
```

Activate Windows
Go to Settings to activate Windows.

Figure 17 Ubuntu 22.04 server workflow

d54-ab-3a-d7-4a-
ce.hyperscalers.com

Locally managed

Lock Object Start Workflow

Editor Params Profiles Meta Activity API

Add Params Remove Params

```
6 |         "type": "mac",
7 |         "value": "a8:1e:84:b3:aa:d4"
8 |     },
9 |     },
10 |     },
11 |     "port": {
12 |       "id": {
13 |         "type": "local",
14 |         "value": "0/6"
15 |       },
16 |       "ttl": "120"
```

provisioner-default-password-hash A \$6\$7k02hvxSspC3yzfN\$DLyk.C8RCfmYKMeHBtWQLrWv7TLI6abz

raid-available-utilities [] [] []

raid-current-config []

universal/hardware A Quanta_Computer_Inc-QuantaPlex_T41S-2U-uefi-rc0

validation/errors []

Vmware

vmware/esxi-generic

vmware/esxi-version A esxi_700u1-16850804_vmware

vmware/esxi-version-override A esxi_700u2a-17867351_rkn_vmware

Figure 18 VMware custom configuration

d54-ab-3a-d7-4a-
ce.hyperscalers.com Locally managed

Lock Object Start Workflow

Editor Params Profiles Meta Activity API

PROFILES + []

POOL [] default Allocate ▶ Status Free

CLUSTER Empty

Operational Controls

RUNNABLE

WORK ORDER MODE

CONTEXT [] Machine

BASE CONTEXT [] Machine []

Workflow Management

PIPELINE [] Select a Pipeline

WORKFLOW [] esxi-install -

BOOT ENVIRONMENT [] esxi_700u2a-17867351_rkn_vmware-install

STAGE READ ONLY [] esxi-preserve-logs

CURRENT JOB [1ed114ed-e6a9-6f41-b663-526359910a2a](#)

TASK QUEUE

```
1 stage:prep-install
2 bootenv:sledgehammer
3 erase-hard-disks-for-os-install
4 stage:vmware-esxi-clear-patch-index
5 vmware-esxi-clear-patch-index
6 stage:vmware-esxi-set-password
7 vmware-esxi-set-password
8 stage:vmware-esxi-selector
9 vmware-esxi-selector
```

Figure 19 Vmware 7.0u2a deployment

d08-00-27-e0-ff-2e.hyperscalers.com Locally managed

[Lock Object](#) [Start Workflow](#)

[Editor](#) [Params](#) [Profiles](#) [Meta](#) [Activity](#) [API](#)

PROFILES [+](#) windows-install

POOL [default](#) [Allocate](#) [Status](#) Free

CLUSTER Empty

Operational Controls

RUNNABLE

WORK ORDER MODE

CONTEXT [Machine](#)

BASE CONTEXT [Machine](#)

Workflow Management

PIPELINE [Select a Pipeline](#)

WORKFLOW [universal-image-deploy](#) [-](#)

BOOT ENVIRONMENT [local](#)

STAGE [complete](#)
READ ONLY

CURRENT JOB ✓ Workflow Complete

TASK QUEUE

- 1 stage:discover
- 2 bootenv:sledgehammer
- 3 update-pipeline
- 4 enforce-sledgehammer
- 5 set-machine-ip-in-sledgehammer
- 6 reserve-dhcp-address
- 7 ssh-access
- 8 record-current-uefi-boot-entry
- 9 stage:universal-image-deploy-start-callback
- 10 callback-task
- 11 stage:universal-image-deploy-post-flowflow

Activate Windows
Go to Settings to activate Windows.

Figure 20 Windows 10 Enterprise deployment

Configuration parameters

Any machine can be customised and deployed as per the needs of the client. All these parameters can be associated with a profile for easier handling and management. Some notable parameters that provide control over access and deployment of the machine include

SSH control

access-ssh-parameters- Allows you to change a specific parameter in sshd_config file (Eg: Cipher)

access-ssh-template- Allows you to send customised sshd_config file with a custom template access-ssh-template (manually created) in Control → Template



```
1 # $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $
2
3 # This is the sshd server system-wide configuration file. See
4 # sshd_config(5) for more information.
5
6 # This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
7
8 # The strategy used for options in the default sshd_config shipped with
9 # OpenSSH is to specify options with their default value where
10 # possible, but leave them commented. Uncommented options override the
11 # default value.
12
13 Include /etc/ssh/sshd_config.d/*.conf
14
15 #Port 22
16 #AddressFamily any
17 #ListenAddress 0.0.0.0
18 #ListenAddress ::
19
20 #HostKey /etc/ssh/ssh_host_rsa_key
21 #HostKey /etc/ssh/ssh_host_ecdsa_key
22 #HostKey /etc/ssh/ssh_host_ed25519_key
23
24 # Ciphers and keying
25 #RekeyLimit default none
26
27 # Logging
28 #SyslogFacility AUTH
29 #LogLevel INFO
30
31 # Authentication:
32
33 #LoginGraceTime 2m
34 PermitRootLogin yes
35 #StrictModes yes
36 #MaxAuthTries 6
37 #MaxSessions 10
38
39 #PubkeyAuthentication yes
40
41 # Expect .ssh/authorized_keys2 to be disregarded by default in future.
42 #AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
```

Figure 21 access-ssh-template

image-builder parameters

For RHEL, these parameters are necessary due to RedHat Subscription management.

```
image-builder/skip-package-reset: true
redhat/subscription-username: "[Insert RHSM Username]"
redhat/subscription-password: "[Insert RHSM Password]"
redhat/rhsm-activation-key: "[Insert RHSP Activation Key]"
redhat/rhsm-organization: "[Insert RHSM Organization ID]"
```

image-deploy parameters

These are necessary parameters for any custom image deployment. The values should reflect the nature of the image that is being deployed. The default preferred location of images is files/images/<image-name>

```
image-deploy/image-installed-size: 60G
image-deploy/image-os: windows
image-deploy/image-type: dd-gz
image-deploy/admin-password: R0cketSk8ts
image-deploy/admin-username: rocketskates
image-deploy/image-file: files/images/win10ent-1909.box.gz
```

VMWare parameters

VMWare can be successfully deployed by changing/ tweaking the following parameters. VMWare custom ISOs are available with Hyperscalers/ RackN for the deployment of VMWare machines. Please contact Hyperscalers for more details.

```
"vmware/esxi-generic": true,
"vmware/esxi-version": "esxi_700-15843807_vmware",
"vmware/esxi-version-override": "esxi_700u2a-17867351_rkn_vmware"
"esxi/insecure-password": "<your-password>",
"esxi/selected-vendor": "vmware"
```

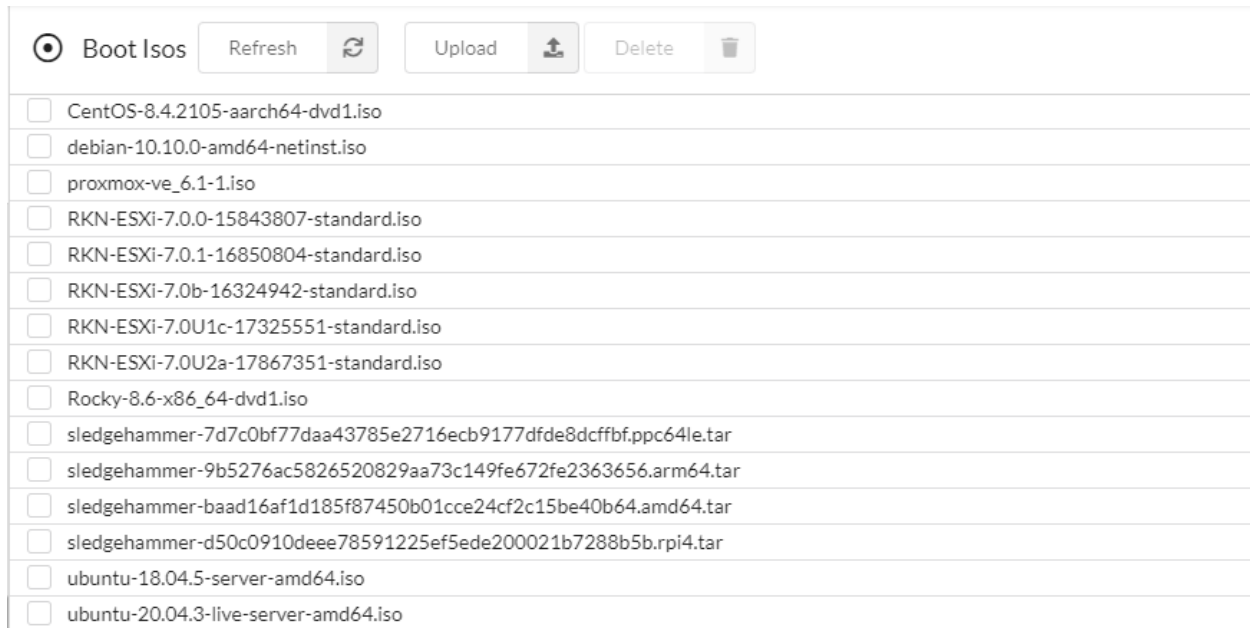


Figure 22 Boot ISOs

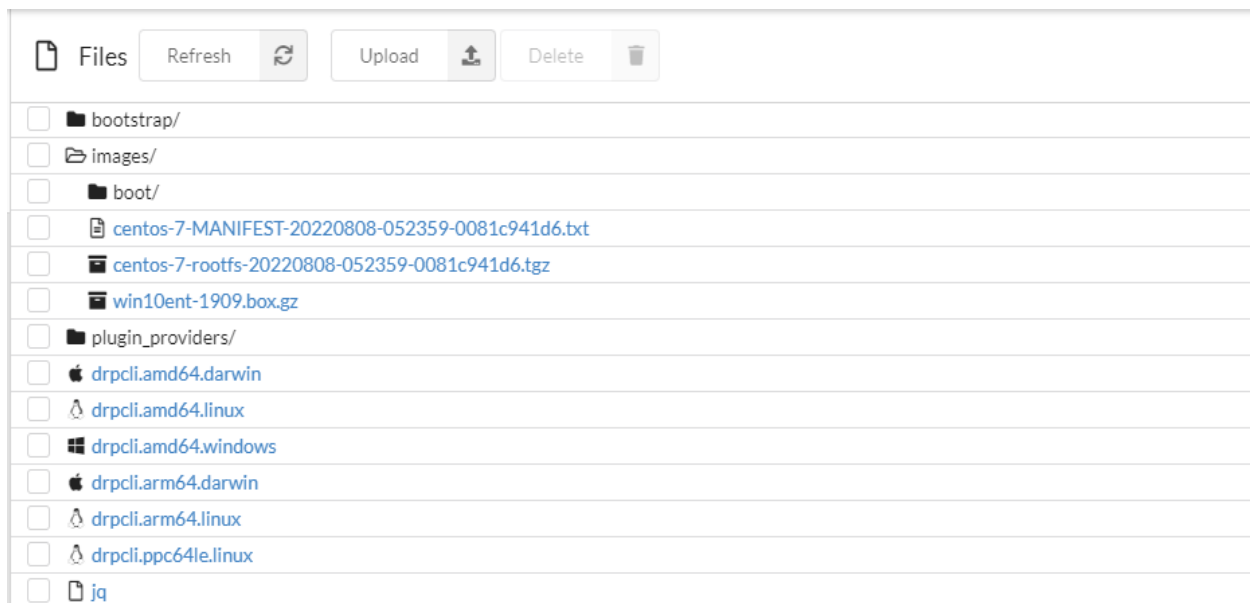


Figure 23 Custom boot images

13 UPDATING THE APPLIANCE

Any update/ upgrade to Digital Rebar's various features can be done through Catalog section within DRP endpoint. Please note while updating / upgrading do not refresh the page.

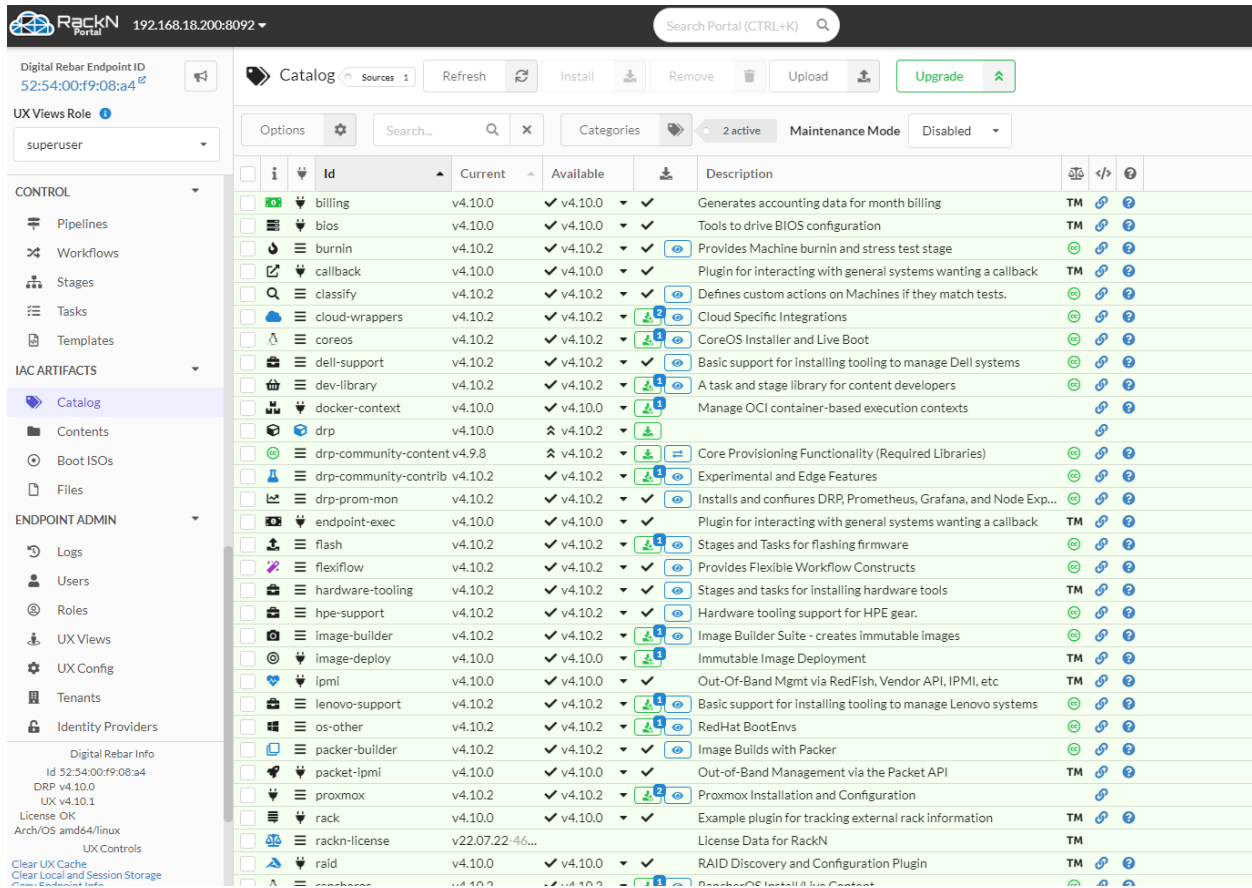


Figure 24 Catalog Iac Artifacts

14 ADDITIONAL SETUP AND DEPLOYMENT

Troubleshooting Appliance

In case of failure of deployment of OS caused by following factors,

Failed at any stage –

1. Check the task logs (the briefcase icon near the failed task).
2. If the failed task is due to lack of internet connectivity, ensure name server parameters are sent to the machine.
3. If the failed task is due to curtin image deployment, ensure that the image being deployed meets the specified requirements from RackN DRP [14] [15]. Contact Hyperscalers for more help.
4. SSH Access – In case of failed SSH access please ensure the public key of the host / guest is added to DRP endpoint or SSH configuration file format is sent during the deployment.

15 ADDENDUM

```
root@ubuntu-Standard-PC-i440FX-PIIX-1996:/home/ubuntu# curl -fsSL get.rebar.digital/stable | bash -s
-- install --universal
>>> Downloading file: rackn-catalog.json from https://rebar-catalog.s3-us-west-2.amazonaws.com/rackn-catalog.json
>>> Downloading file: drpcli from https://s3-us-west-2.amazonaws.com/rebar-catalog/drpcli/v4.9.10/amd64/linux/drpcli
### Checking service ports.
### 'dr-provision' service is not running, beginning install process ...
>>> Ensuring required tools are installed
### Using temp directory to extract artifacts to and install from ('/tmp/drp.installer.LoZgwT').
>>> Installing Version stable of Digital Rebar (dr-provision) with v22.07.25-1 install.sh
### Using MAC based Endpoint ID. Consider using '--drp-id' to set during installation.
>>> Downloading file: rackn-catalog.json from https://rebar-catalog.s3-us-west-2.amazonaws.com/rackn-catalog.json
>>> Downloading file: v4.9.11.zip from https://s3-us-west-2.amazonaws.com/rebar-catalog/drp/v4.9.11.zip
>>> Downloading file: v4.9.11.amd64.linux.zip from https://s3-us-west-2.amazonaws.com/rebar-catalog/drp/v4.9.11.amd64.linux.zip
./tools/install.sh: OK
./assets/startup/dr-provision.service: OK
./assets/startup/dr-provision.sysv: OK
./assets/startup/dr-provision.unit: OK
./bin/linux/amd64/drpjoin: OK
./bin/linux/amd64/drpcli: OK
./bin/linux/amd64/drundler: OK
./bin/linux/amd64/dr-provision: OK
./bin/linux/amd64/dr-waltool: OK
>>> Installing Version stable of Digital Rebar Community Content
>>> Downloading file: v4.9.8.json from https://s3-us-west-2.amazonaws.com/rebar-catalog/drp-community-content/v4.9.8.json
### Will attempt to execute startup procedures ('--startup' specified)
systemctl daemon-reload && systemctl start dr-provision
systemctl daemon-reload && systemctl enable dr-provision
### Install script saved to '/usr/local/bin/drp-install.sh'
### You can uninstall DRP with ' /usr/local/bin/drp-install.sh remove' - must be root)
Created symlink /etc/systemd/system/multi-user.target.wants/dr-provision.service → /etc/systemd/system/dr-provision.service.
### Waiting for dr-provision to start ....
Digital Rebar "version": "v4.9.11" started as Endpoint "id": "52:54:00:f9:08:a4",
{
  "Counts": {
    "blueprints": 1,
    "bootenvs": 34,
    "params": 85,
    "profiles": 7,
    "stages": 48,
    "tasks": 31,
    "templates": 39,
    "triggers": 1,
    "workflows": 3
  },
  "Warnings": [],
  "meta": {
    "Author": "RackN and Digital Rebar Community",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "green",
    "Copyright": "RackN 2019",
    "Description": "Core Provisioning Functionality (Required Libraries)",
    "DisplayName": "Community Core",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/drp-community-content.html",
```

```

    "Documentation": "Community Content\n-----\n\nThe drp-community-content is a
required initial content package for Digital Rebar Provision if you plan to use **any**\nof the
digital rebar content.\n\nThis contains the basic building blocks of all the rest of the system.
From basic bootenvs, stages,\nand workflows to templates and tasks, this content pack starts the
whole process.",
    "Icon": "creative commons",
    "License": "APLv2",
    "Name": "drp-community-content",
    "Order": "100",
    "Overwritable": false,
    "Prerequisites": "BasicStore",
    "RequiredFeatures": "sane-exit-codes, job-exit-states, fsm-runner, workflows, default-workflow,
http-range-header, roles, tenants, sprig, multiarch, overridable-bootloaders, centos-8, virtual-
media-boot, secure-file-server",
    "Source": "https://gitlab.com/rackn/provision-content",
    "Tags": "base,community,linux,APL,quick-start,ssh",
    "Type": "dynamic",
    "Version": "v4.9.8",
    "Writable": false
  }
}
{
  "Counts": {
    "blueprints": 5,
    "params": 106,
    "profiles": 9,
    "stages": 20,
    "tasks": 48,
    "templates": 21,
    "triggers": 1,
    "version_sets": 1,
    "workflows": 11
  },
  "Warnings": [],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "green",
    "Copyright": "RackN 2020",
    "Description": "Essential task and stage library",
    "DisplayName": "Core Task Library",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/task-library.html",
    "Documentation": ".. _component_task_library:\n\nRackN Task Library\n-----\n\nThis
content package is a collection of useful stages and operations for Digital Rebar. It also includes
several handy workflows like the CentOS base, Fedora base, and Ubuntu base workflows.\nYou can also
find many handy stages in this like the network-lldp stage which can be added to your discovery to
add additional networking information discovered by lldp.\n\n..
_component_task_library_cluster:\n\nCluster Stages\n=====\n\nThese stages implement the
:ref:`rs_cluster_pattern`.\n\nAllows operators to orchestrate machines into sequential or parallel
operation.\n\n.. _component_task_library_inventory:\n\nInventory Stage\n=====\n\nConvert
gohai and other JSON data into a map that can be used for analysis, classification or filtering.",
    "Icon": "shopping basket",
    "License": "APLv2",
    "Name": "task-library",
    "Order": "100",
    "Overwritable": false,
    "Prerequisites": "BasicStore:>=4.8.0,drp-community-content:>=4.8.0",
    "RequiredFeatures": "",
    "Source": "RackN",
    "Tags": "base,rackn,tasks,linux,library",
    "Type": "dynamic",
    "Version": "v4.9.8",
    "Writable": false
  }
}
}

```



```
>>> Installing content item 'universal'
{
  "Counts": {
    "params": 9,
    "stages": 2,
    "tasks": 3,
    "templates": 7,
    "workflows": 1
  },
  "Warnings": [],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "black",
    "Copyright": "RackN 2019",
    "Description": "Defines custom actions on Machines if they match tests.",
    "DisplayName": "Classify",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/classify.html",
    "Documentation": "The Classifier System allows an operator to flexibly define groups
of *test*/ *action* sequences to perform on a Machine. These can be used\nto do any number of
tasks. Typically the classifier allows an operator\nto add additional information to a Machine
object that will alter the\nbehavior of subsequent Workflow stages/tasks.\n\nBy default the
classifier provides a limited set of *tests* and *actions*\nthat can be performed. However, the
operator can specify additional\ngroups of *test*/ *actions* pairs by use of the ``classify/custom-
functions``\nParam. See the Param ``classify/custom-functions`` documentation on how to\nuse
it.\n\nBy default, the classifier is designed to exit on the first test match\nthat succeeds with a
`pass` status. The operator can override this\nbehavior by setting the optional ``continue``
value set to `true`\nin the ``classify/classification-data`` structure.\n\nIf you are extending
the Classifier to add custom groups of *test*/ *action*\nsequences, please review the ``function ...
{}`` BASH functions in the\n``classify.sh.tpl`` **template** for examples.\n\n\nGetting Started
with the Classifier\n-----\n\nThe Classifier rulesets of
*test*/ *action* sequences are configured by setting the Param\n``classify/classification-data`` on a
Machine. The usual rules and orders of precedence\nare followed on this Param (Param on Machine;
Profile on Machine; Global Profile; etc...).\n\nThe classification data structure is a list of YAML
or JSON objects, with two required and one\noptional set of values. The structure is as follows
(example in YAML):\n\n ::\n\n   - test: "<TEST_TYPE>"\n     actions:\n       - "<ACTION>"\n
- "<ACTION>"\n     continue: true\n\nThe ``test`` and ``actions`` are required elements.
``continue`` is optional based on the\noperators need.\n\nHere are some example usage scenarios to
help you get started.\n\nSimple MAC Address Match and
Classify\n=====\n\nIn this scenario, you will use the MAC Address as
a simple *test* matcher\nto apply *actions*. Demonstrated in this example are adding two Params,
and two profiles\nto the machine.\n\nThe Params in this example sets hostname, and
``universal/application`` value to specify the\n*role* that this machine will perform. Subsequent
content/plugins can react accordingly to\nthis. The profiles might be responsible for carrying
configuration data for subsequent\ncontent/plugins to operate on. For example, setting BIOS ``bios-
target-configuraion``,\ndetermining what Firmware/Flash versions to use, Operating System
configuration profile\ndata (eg VMware ESXi configuration, etc.).\n\nLast, this classify test/action
sequence will continue on in the list to any subsequent\ntest/action sequences after a match is
found (``continue: true``).\n\n ::\n\n   classify/classification-data:\n     - test:
\"has_mac 24:6e:96:6a:40:34\"\n     actions:\n       - \"set_parameter hostname r3r6-
u03\"\n       - \"set_parameter universal/application vcf381\"\n       - \"add_profile
demo-r3r6-machine-r3r6-u03\"\n       - \"add_profile demo-r3r6-application-vcf381\"\n
continue: true\n\n\nChange Workflow\n=====\n\nIn this scenario, the Classify system will
be used to change the Machines Workflow. We\nwill match on the Inventory field
``inventory/Manufacturer`` with a match of ``QEMU``\n(which is typically how a KVM hypervisor will
present a KVM backed Virtual Machine). The\nMachine will be changed to the ``centos-install``
workflow to install CentOS operating\nsystem.\n\n.. note:: When the Machine's workflow is changed,
any remaining Stages/Tasks from the **current**\nworkflow will be immediately replaced by
the new Workflow.\n\nThis example assumes that the operator has already set the ``classify/custom-
functions`` to include\nthe ``has_inventory_value.sh.tpl`` template included already.\n\n ::\n\n   classify/classification-data:\n     - test: \"has_inventory_value Manufacturer QEMU\"\n
actions:\n       - \"change_workflow centos-install\"\n\n\nSince there is no ``continue: true``
setting, the classifier will exit after this rule, and\nwill not process any further rules listed
after this one.\n\n.. note:: The ``Flexiflow`` content pack is designed to provide controlled ways
```

```

to inject new\n          **tasks** and switch **workflows**. You may wish to utilize the constructs
in that\n          content pack over the ``change_workflow`` action in the Classifier.",
  "Icon": "search",
  "License": "APLv2",
  "Name": "classify",
  "Order": "1000",
  "Overwritable": false,
  "Prerequisites": "",
  "RequiredFeatures": "",
  "Source": "RackN",
  "Tags": "advanced,enterprise,rackn,classify,linux,library",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "path": "ipmi",
  "size": 12488704
}
{
  "Counts": {
    "params": 9,
    "stages": 4,
    "tasks": 7
  },
  "Warnings": [
    "Layer ipmi has hardware-tooling as a prerequisite, but hardware-tooling does not exist!"
  ],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "black",
    "Copyright": "RackN 2019",
    "Description": "Provides Machine burnin and stress test stage",
    "DisplayName": "System Burn In",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/burnin.html",
    "Documentation": "",
    "Icon": "fire",
    "License": "APLv2",
    "Name": "burnin",
    "Order": "2000",
    "Overwritable": false,
    "Prerequisites": "BasicStore:>=4.6.0",
    "RequiredFeatures": "",
    "Source": "RackN",
    "Tags": "advanced,hardware,rackn,burnin,validation,physical",
    "Type": "dynamic",
    "Version": "v4.9.8",
    "Writable": false
  }
}
}
{
  "path": "rack",
  "size": 11399168
}
{
  "path": "raid",
  "size": 13189120
}
}
{
  "path": "bios",
  "size": 14114816
}
}
{

```

```
"Counts": {
  "params": 7,
  "stages": 1,
  "tasks": 3
},
"Warnings": [
  "Layer ipmi has hardware-tooling as a prerequisite, but hardware-tooling does not exist!"
],
"meta": {
  "Author": "RackN",
  "CodeSource": "https://gitlab.com/rackn/provision-content",
  "Color": "black",
  "Copyright": "RackN 2019",
  "Description": "Basic support for installing tooling to manage Dell systems",
  "DisplayName": "Dell Support",
  "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/dell-support.html",
  "Documentation": "",
  "Icon": "travel",
  "License": "APLv2",
  "Name": "dell-support",
  "Order": "2000",
  "Overwritable": false,
  "Prerequisites": "",
  "RequiredFeatures": "",
  "Source": "RackN",
  "Tags": "hardware,rackn,dell-support,validation,physical",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "Counts": {
    "params": 5,
    "tasks": 2
  },
  "Warnings": [
    "Layer ipmi has hardware-tooling as a prerequisite, but hardware-tooling does not exist!"
  ],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "black",
    "Copyright": "RackN 2019",
    "Description": "Hardware tooling support for HPE gear.",
    "DisplayName": "HPE Support",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/hpe-support.html",
    "Documentation": "",
    "Icon": "travel",
    "License": "APLv2",
    "Name": "hpe-support",
    "Order": "2000",
    "Overwritable": false,
    "Prerequisites": "",
    "RequiredFeatures": "",
    "Source": "",
    "Tags": "hardware,rackn,hpe-support,validation,physical",
    "Type": "dynamic",
    "Version": "v4.9.8",
    "Writable": false
  }
}
{
  "Counts": {
    "params": 2,
```

```
"tasks": 1
},
"Warnings": [
  "Layer ipmi has hardware-tooling as a prerequisite, but hardware-tooling does not exist!"
],
"meta": {
  "Author": "RackN",
  "CodeSource": "https://gitlab.com/rackn/provision-content",
  "Color": "black",
  "Copyright": "RackN 2019",
  "Description": "Basic support for installing tooling to manage Lenovo systems",
  "DisplayName": "Lenovo Support",
  "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/lenovo-support.html",
  "Documentation": "",
  "Icon": "travel",
  "License": "APLv2",
  "Name": "lenovo-support",
  "Order": "2000",
  "Overwritable": false,
  "Prerequisites": "drp-community-content: >=1.13.0",
  "RequiredFeatures": "",
  "Source": "RackN",
  "Tags": "hardware,rackn,lenovo-support,validation,physical",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "Counts": {
    "params": 2,
    "tasks": 1
  },
  "Warnings": [
    "Layer ipmi has hardware-tooling as a prerequisite, but hardware-tooling does not exist!"
  ],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "black",
    "Copyright": "RackN 2021",
    "Description": "Hardware tooling support for Supermicro gear.",
    "DisplayName": "Supermicro Support",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/supermicro-
support.html",
    "Documentation": "",
    "Icon": "travel",
    "License": "APLv2",
    "Name": "supermicro-support",
    "Order": "2000",
    "Overwritable": false,
    "Prerequisites": "",
    "RequiredFeatures": "",
    "Source": "",
    "Tags": "hardware,rackn,supermicro-support,validation,physical",
    "Type": "dynamic",
    "Version": "v4.9.8",
    "Writable": false
  }
}
{
  "Counts": {
    "profiles": 1,
    "stages": 2,
    "tasks": 2
  }
}
```

```
},
"Warnings": [],
"meta": {
  "Author": "RackN",
  "CodeSource": "https://gitlab.com/rackn/provision-content",
  "Color": "black",
  "Copyright": "RackN 2019",
  "Description": "Stages and tasks for installing hardware tools",
  "DisplayName": "Hardware Tooling",
  "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/hardware-
tooling.html",
  "Documentation": "",
  "Icon": "travel",
  "License": "RackN 2019",
  "Name": "hardware-tooling",
  "Order": "2000",
  "Overwritable": false,
  "Prerequisites": "dell-support, hpe-support, lenovo-support, supermicro-support",
  "RequiredFeatures": "",
  "Source": "",
  "Tags": "rackn,hardware,validation,physical",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "Counts": {
    "params": 11,
    "stages": 1,
    "tasks": 7,
    "templates": 2
  },
  "Warnings": [],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "black",
    "Copyright": "RackN 2019",
    "Description": "Stages and Tasks for flashing firmware",
    "DisplayName": "Firmware Flash",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/flash.html",
    "Documentation": "",
    "Icon": "upload",
    "License": "APLv2",
    "Name": "flash",
    "Order": "2000",
    "Overwritable": false,
    "Prerequisites": "hardware-tooling: >=1.13.0,drp-community-content",
    "RequiredFeatures": "",
    "Source": "RackN",
    "Tags": "hardware,rackn,flash,validation,physical",
    "Type": "dynamic",
    "Version": "v4.9.8",
    "Writable": false
  }
}
{
  "path": "vmware",
  "size": 13193216
}
{
  "path": "callback",
  "size": 11362304
}
}
```

```
{
  "Counts": {
    "params": 11,
    "stages": 4,
    "tasks": 6,
    "templates": 1
  },
  "Warnings": [],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "green",
    "Copyright": "RackN 2019",
    "Description": "Provides Validation Tasks and Workflow Elements",
    "DisplayName": "Validation",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/validation.html",
    "Documentation": "The Validation System allows an operator to flexibly define a set of tasks to
do validation operations. Validation allows an operator to execute tests to verify that certain
conditions have been met either in the environment or as completed tasks by other workflow
steps. This allows for a composable set of rules to be run to either verify the system has been
provisioned successfully or an error may have occurred that needs addressed. The validation
system is designed to be flexibly extended in the field. The current set of Stages, Library
functions, and Tasks are curated capabilities by RackN. Custom Stages and Tasks can be flexibly
added to allow for unique use cases. If you have developed a set of Tasks and Library functions
that may be useful to other operators, please consider contributing back to the content pack to
enhance it for other users. Prebuilt Stages
-----
The validation system provides
three prebuilt base sets of validation stages to run the validation engine in:
1. ``validation-post-discover``
2. ``validation-post-hardware``
3. ``validation-post-install``
The intent is to have a set of tasks that validate after the initial discover (*gohai-
inventory*, *inventory*, or other "discover" related tasks), post hardware configuration
(typically after BIOS settings, Firmware Flash, and RAID volume creation), and in the final
installed operating system. The validation system is run in Workflow as a standard *Stage*
allowing the operator to place the stage anywhere in the Workflow that makes sense. The above
three pre-built stages define RackN defined useful points during the provisioning lifecycle of a
_Machine_. The ``validation-start`` and ``validation-stop`` tasks are the only tasks that should
be listed in the Stage, do not add Tasks to the Stage. Tasks will dynamically be added based on
the control Parameters. Defining Validation Tasks
-----
Control over
which specific validation tasks are executed at any given validation stage is defined by the Param
``validation/list-parameter``. This parameter is a reference to the Param name that the
``validation-start`` task uses to dynamically update the task list with a list of validation
tasks. The Tasks that are defined in the parameter (specified by the ``validation/list-
parameter``) will be composed and added to the task list. These will be dynamically inserted
between the ``validation-start`` and ``validation-stop`` tasks. *Composed* means the value of
that parameter is the aggregation of all occurrences of that parameter at all levels of the system,
(e.g. From the parameters on the machine, then profiles on machines, then parameters on the stage,
...). This allows multiple resources to provide the validation tests that will eventually be run
on the system. Typically, Parameter orders of precedence would override earlier occurrences of the
Parameter values. Validation does not follow that standard path. The Param that is referenced by
``validation/list-parameter`` should be a simple array type Param. It lists each *Task* that
should be executed in the listed sequence by the validation process at that *Stage* in
the *Workflow*. Task Exit Conditions
-----
Validation actions can exit one
of several ways:
1. ``validation-success`` = validation succeeded, exit 0
2. ``validation-
fail-immediately`` = fail and exit 1, stopping workflow immediately at task
3. ``validation-fail-
at-stage-end`` = exit 0, collect logs in validation/errors param, stop at end of Stage
4. ``validation-fail-
and-ignore`` = log validation error to validation/errors-ignored, but exit 0 - no
workflow stop will occur at all futures return conditions (not yet implemented):
``validation-
fail-and-remediate`` = exit 1, and if the task specifies an additional remediation task, run this
new task, plus oneself again - hoping the remediation task solved the issue - may need number
tracking and break the cycle after N failures
Remediating and Continuing Failed Tasks
-----
The failure exit conditions allow the developer of the validation
task to determine if the failure is a "soft" failure which can be remediated, and then the
Workflow can be resumed, or if the failure implies a "hard" fail that can not be corrected during
workflow operation. Additionally, some failures may not impact final workflow completion, but the
operator would like to have them recorded for future review or remediation. If a failure that is
external to machine and can be remediated (for example a DNS record has the wrong information, and
```

DNS can be updated quickly)\nthe test should be `*validation-fail-immediately*`. This allows the failure to\nbe remediated and Workflow can be continued without failing validation.\n\nIf the failure is not able to be remediated immediately, a log message should be\nadded to the `validation/errors` logging Param, and success returned so\nthat additional issues can be found and aggregated.\n\n\nExample Usage\n-----\n\nHere is a brief outline of example usage of the validation system utilizing\nthe RackN defined stages. In this example, we will define tasks to run\nin\nthe `validation-post-discovery` and `validation-post-hardware` stages.\nMost of the Tasks referenced are valid tasks that exist across several\ncontent and plugin components.\n\nWorkflow\n=====\n\nIn this example we will create a `"my-complete-discovery"` workflow that utilizes several\ncomponents of the Digital Rebar content and plugins capability.\n\n.. note:: Note the addition of the 'validation-post-discovery' and 'validation-post-hardware' Stages, these define the point in the workflow\n where the validation process will run.\n\n ::\n\nMeta:\n color: grey\n icon: money\n Name: my-complete-discovery\n Description: My complete discovery workflow\n Stages:\n - discover\n - setup-repos\n - ipmi-inventory\n - raid-inventory\n - network-lldp\n - inventory\n - rack-discover\n - classify\n - validation-post-discover\n - ipmi-configure\n - flash\n - raid-enable-encryption\n - raid-configure\n - bios-configure\n - ilo-config\n - burnin\n - burnin-reboot\n - validation-post-hardware\n - notify-hardware-complete\n - sledgehammer-wait\n\n\nStages\n=====\n\nIn this example, we are not creating any new stages, simply using the RackN\ndefined stages in the above workflow example. Please review the `"Adding\nNew Stages"` section if you'd like to create custom stages.\n\n\nTasks\n=====\n\nThere are several tasks that are defined to be used by the Validation system\nin this example. You can find them in the following profile section. Note\nthat several of these tasks do exist in various content and plugins. Each of\nthese tasks would need to exist and perform the required functions.\n\n\nDefine the Tasks for Validation\n=====\n\nIn this example, we will utilize a profile which contains the defined Params\nfor the `validation-post-discover` and `validation-post-hardware` stages.\n\nThe operator can apply the Profile to the appropriate machines, you could\npotentially use the `classify` stage to execute a classification task to\ndetermine whether or not to programmatically apply the Profile to the given\nmachine as it executes the `classify` stage.\n\nThe profile defines the tasks to run. The profile YAML configuration would\nlook like the following:\n\n ::\n\n ---\n Meta:\n color: grey\n icon: money\n Name: validations\n Params:\n validation/post-discover:\n - "validation-machine-name-dns-to-ip"\n - "in-subnet-check-render"\n - "in-subnet-check-validate"\n - "ipmi-network-validation"\n - "validate-nic-counts"\n - "validate-jumbo-frames"\n validation/post-hardware:\n - "validate-ipmi-hostname-ip"\n\n\nThe type definition for the Params `validation/post-discover` are defined\nin the Validation content pack, and the appropriate stage contains the\ncontrol mapping for each.\n\nFor the `validation-post-discover` Stage, the stage carries the mapping\nwithin the stage, as follows:\n\n ::\n\n Params:\n "validation/list-parameter": "validation/post-discover"\n\n\nSimilarly, the `validation-post-hardware` stage defines the mapping to\npoint to the Param `validation/post-discover`.\n\nThese are the values we use in our profile above to add the list of Tasks\nto execute at the appropriate Stage run.\n\n\nOperating The Validation\n=====\n\nOnce you have completed the above tasks, you can choose to make the\n`"my-complete-discovery"` Workflow your default preference setting for all\nnew machines, or manually choose to place machines in to the Workflow.\n\nTo make this the default workflow, change your preferences (from `**Info\n&Preferences**` in the Portal), or the following CLI command:\n\n * `drpcli prefs set defaultWorkflow my-complete-discovery`\n\n\n\nReviewing Any Failures\n=====\n\nAs with all jobs executed in Workflows, there will be a job log available to\nreview the status output of any tasks that have run. If you receive a\nValidation failure, refer to the appropriate job log.\n\nIf the failure includes the use of the `add_validation_error` function, you\ncan review the Param value that was added to the Machine, with the name\n`validation/errors`. The Param will be added to the machine that has\nfailed the validation process.\n\nThe next run of any validation stage will empty the contents of this Param,\nprior to starting executing the tasks. Please insure that you review the\nerror values after a failure, but before you re-run any more validation\nstages.\n\nIt is required to include the `{{ template "setup.tpl" }}` in each of the\nvalidation tasks. With the inclusion of this template, you can set the\n`rs-debug-param` on a Machine, and the validation tasks will contain a lot\nmore debug output in the job log. If you need to verify/debug the actions in\nthe validation task, this is a good way to review more detailed output logging\ninformation.\n\n\nSkipping Validation Without Changing Your Workflow\n=====\n\nThe validation system was designed to allow you to add it to the Workflow, and\nif no validation tasks are defined (by use of the reference\n`validation/list-parameter` Param), then the system will skip any validation\nattempts for the given stage.\n\nThis allows you to leave the Stage defined control parameter blank and\nnot require changing the Workflow. This effectively becomes a `"noop"` of the\nvalidation


```

system.\n\n\nDeveloping New Validation Capabilities\n-----\n\nThis
section provides information on how to build custom validation capabilities\nutilizing the existing
Validation system.\n\n\nNamespace Definitions\n=====\n\nValidation system stages,
control params, and other content parts will utilize\nthe prefix namespace of ``validation/``.
Individual Tasks, Params, etc. that\nimplement functional use of the Validation system components
will utilize the\nprefix namespace of ``validate/``.\n\nFor example, the Start Stage for the
Validation system is ``validation-start``,\nwhile the NIC Count and UP link state checking uses a
Task namespaced as\n``validate-nic-counts``.\n\nStandard Stage/Task and Param semantics still apply
(eg ``validate/nic-required-up``\nis the Param namespace name).\n\nNote that traditionally, RackN
uses the \"/" naming standard for Params,\nwhile Stages, Taks, and Templates generally utilize
a "dash" to separate the\nparent namespace. Don't ask us why.\n\n\nLibrary of
Functions\n=====\n\nThe Validation System provides a template which contains a
library of currated\ntools for the system; the template is called *validation-lib.tpl*. Only
Bash\nfunctions which implement broadly useful capabilities are placed in this\ntemplate, which is
managed by RackN.\n\nIf your validation tools require a common library, they can be built as
a\nstandard template, and included in any appropriate custom tasks.\n\nCurrent functions that can be
incorporated in to your Validation custom tasks\nare as follows:\n\n * ``validation_add_error()`` =
Adds a validation error to the ``validation/errors`` Param\n * ``validation_add_error_ignore()`` =
Adds a validation error to the ``validation/errors-ignore`` Param\n *
``validation_clear_errors_ignore()`` = Removes the ``validation/errors-ignore`` Param from the
Machine\n * ``validation_msg_prefix()`` = builds a prefix of Stage/Task/CurrentJob for output
messages\n * ``validation_success()`` = marks a validation task successful and exits 0\n *
``validation_fail_at_stage_end()`` = marks a validation task failed, but exits 0, and ultimately
exits the Stage with exit code 1\n * ``validation_fail_immediately()`` = marks validation task
failed and exits 1 immediately\n * ``validation_fail_and_ignore()`` = marks validation as failed,
but ignores the failure without exiting the workflow\n * ``#validation_fail_and_remediate()`` = Not
implemented yet\n * ``validate_machine_name_dns_by_ip()`` = verifies the machine has a DNS record
based on the Machines IP address\n * ``validate_ipv4_ip_syntax()`` = simple helper to verify string
is in IPv4 dotted quad notation\n * ``validate_check_same_subnet()`` = verifies that given 2 ip
address and a subnet mask, both addresses are in the same subnet network\n *
``validate_ping_dest_from_src()`` = verifies that Machine can ping an IP address given a source
interface to use\n * ``validate_nic_counts()`` = verifies the number of NICs and number of NICs
that can be brought to an "UP" state\n\n\nAdding New Stages\n=====\n\nStages define
the primary grouping of Tasks that an operator runs at a given\npoint in the Workflow sequence.
These stages can be placed anywhere that\nmakes operational sense to a given Workflow.\n\nThe Stage
should **only** contain the ``validation-start`` and ``validation-stop``\ntasks, no other tasks
should be added to your stage, as the ``validation-start``\nstage will dynamically inject the
desired stages in to the Workflow for the\noperator.\n\nExample Stage in YAML:\n\n  ::\n  ---\n  Name: "my-validation"\n  Description: "Perform tasks defined by the 'my-validation' Param."\n  Documentation: |\n  The Param 'my-validation' is an array that will contain\n  the list of\n  validation tasks to run.\n  Params:\n  "validation/list-parameter": "my-validation"\n  Tasks:\n  - "validation-start"\n  - "validation-stop"\n  Meta:\n  color:\n  "orange"\n  icon: "search"\n  title: "RackN Content"\n\nNote that as the
``Documentation`` field says; the ``validation/list-parameter``\nfor this Stage is defined as ``my-
validation``. This Parameter must be defined\nas an array; which is a list of the Tasks to execute
during this Stage. The\nParam can be defined on the Machine in all of the normal ways (directly as
a\nParam, as part of a Profile, or as a Param in the Global Profile.\n\n\nAdding New
Tasks\n=====\n\nTasks are the heart of the validation system, and perform the actual
validation\nimplementation for the system to execute. Validation tasks are standard RackN\nWorkflow
Tasks, and can carry embedded templates, or refer to external templates\nto perform the actual
Task(s) defined.\n\nA validation task should include both the ``setup.tpl`` template and
the\n``validation-lib.tpl`` template. Both templates are required for a successful\nvalidation
task.\n\nValidation tasks are only limited by the requirements (and perhaps creativity) of\nthe
author. They should be individual and discreet items that the system should\ncheck for and return
results on. Please review the "Task Exit Conditions" above\nto insure you handle exit codes
correctly for the system.\n\nHere is an example Task that implements a hypothetical validation
function.\n\n  ::\n  ---\n  Name: "validation-test-fail"\n  Description: "Example failing
validation task"\n  Meta:\n  color: "blue"\n  icon: "bug"\n  title: "RackN
Content"\n  feature-flags: "sane-exit-codes"\n  Templates:\n  - Name:\n  "validation_fail.sh"\n  Contents: |\n  #!/usr/bin/env bash\n  {{ template
"setup.tpl" . }}\n  {{ template "validation-lib.tpl" . }}\n\n  {{ if eq (.Param
"margarita-time" ) true }}\n  echo "This is a failure. Calling add_validation_error"\n
add_validation_error "Margarita time! We'll finish provisioning later."\n  exit 0\n
{{ else -}}\n  echo "Sadly, it's not margarita time. Provision on!"\n  exit 0\n
{{ end -}}\n\nIn this example, some other component or process would be responsible for\nsetting the

```



```

check Param `margarita-time` to either `true` or `false`.

Utilizing the Error Logging
Param\=====
The library also contains a helper routine to add errors
to the `validation/errors` logging Param. The Bash function name is `add_validation_error`
which should be called with a single string which contains the text of the error message. This
function can safely called multiple times, and each subsequent error message will be appended to
the `validation/errors` array.

Separating Validate Tasks from the Validation
System\=====
Validation tasks or extensions can
be built and used and added to other Content Packs nor Plugins. The validation system itself is a
framework for executing and providing the method to run and manage the validation
process.

Examples of validation in use can be found throughout some of the other RackN managed
content and plugin systems. For example, review the VMware Plugin content:
https://gitlab.com/rackn/provision-plugins/blob/v4/cmds/vmware/content/tasks/validation-machine-
name-dns-to-ip.yaml
For a validation task that is defined and used outside of the Validation
content pack.",
  "Icon": "bug",
  "License": "APLv2",
  "Name": "validation",
  "Order": "1000",
  "Overwritable": false,
  "Prerequisites": "",
  "RequiredFeatures": "",
  "Source": "RackN",
  "Tags": "advanced,validation,physical",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "Counts": {
    "params": 3,
    "stages": 6,
    "tasks": 7,
    "workflows": 2
  },
  "Warnings": [],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "purple",
    "Copyright": "RackN 2019",
    "Description": "Provides Flexible Workflow Constructs",
    "DisplayName": "FlexiFlow",
    "DocUrl": "https://provision.readthedocs.io/en/latest/doc/content-packages/flexiflow.html",
    "Documentation": "The FlexiFlow system allows an operator to dynamically inject new tasks
    an existing workflow based on a Parameter. This allows for other content to build up Tasks to add
    an existing workflow flexibly. Some examples may include adding specific tasks based on
    classification results from the Classify system, or other paths.

    **Stage Based Manipulation**
    The Tasks that are defined in the parameter (specified by the `flexiflow/list-
    parameter`) will be composed and added to the task list. These will be dynamically inserted
    between the `flexiflow-start` and `flexiflow-stop` tasks.

    *Composed* means the value of that parameter is the aggregation of all occurrences of that parameter at all levels of the system, (e.g.
    From the parameters on the machine, then profiles on machines, then parameters on the stage, ...).
    This allows multiple resources to provide the FlexiFlow tasks that will eventually be run on the
    system. Typically, Parameter orders of precedence would override earlier occurrences of the
    Parameter values.

    FlexiFlow does not follow that standard path.

    The Param that is referenced by `flexiflow/list-parameter` should be a simple array type Param. It lists each *Task* that should
    be executed in the listed sequence by the FlexiFlow process at that *Stage* in
    the *Workflow*.

    **Workflow Based Manipulation**
    The Workflow that is defined in the Param
    named `flexiflow-workflow` will be run when the stage of the same name is run. This allows the
    operator to jump in to another Workflow in an existing workflow.

    Operating FlexiFlow Stage Based Manipulation
    -----
    Here is a brief outline of
    example usage of the FlexiFlow system utilizing the RackN defined stage. In this example, we will
    define tasks to run in the `flexiflow-stage`.

    Using the Stage and Task
    Injection\=====
    In this example we will create a `my-flexiflow-
  
```

```

discovery\" workflow.\n\n ::\n\n  Meta:\n    color: grey\n    icon: magic\n    Name: my-
flexiflow-discovery\n  Description: My flexiFlow Discovery Workflow\n  Stages:\n    -
discover\n    - ipmi-inventory\n    - raid-inventory\n    - network-lldp\n    - inventory\n
- classify\n    - flexiflow-stage\n    - sledgehammer-wait\n\nNote that the ``flexiflow-stage``
is run after the ``classify`` stage. A FlexiFlow stage can be added anywhere that conceptually
makes sense.\n\n\nUsing the FlexiFlow Stage\n=====\n\nThe FlexiFlow system is
comprised of a single stage named ``flexiflow-stage``.\n\nThe stage must contain ONLY the ``flexiflow-
start`` and ``flexiflow-stop``\n\nTasks. The ``flexiflow-start`` stage will dynamically inject the
desired tasks\n\nin to the Workflow for the operator. **DO NOT** add any tasks between these\n\nstart
and stop tasks.\n\nYou can choose to use the provided ``flexiflow-stage``, and customize\n\nit's use
by adding Params to the Machine that define the Tasks that will run,\n\nor you can create a custom
Stage. The provided FlexiFlow stage is used the\n\nsame way that a custom named stage would be
used.\n\n\nTo define the custom list of tasks to execute, simply follow the below steps:\n\n  * Add
the ``flexiflow/list-parameter`` param to the Machine\n  * The value of which is another Parameter
that is an Array of Strings that lists tasks to add\n  * An example is ``superflex`` (which would be
the Array of Strings)\n  * Insure you create a Param with Array of Strings as the type definition
(eg ``superflex``, defined as Array of Strings)\n  * Add the Param (eg ``superflex``) to the Machine
with a list of tasks in the array\n\nIf you create a custom stage, you can *optionally* embed the
Params that control\ndefining the list of Tasks to insert and/or the actual task list to
insert.\n\n\nHere is an example Stage in YAML that defines which specific Param will contain\n\nthe list
of tasks to be executed. Note that we are not defining the actual\n\nlist of task in this example.
Presumably a previous Stage (possibly Classify)\n\nwould build up the value of ``superflex`` tasks to
execute dynamically.\n\n  ::\n\n    ---\n    Name: \"my-superflex-stage\"\n    Description:
\"Perform tasks defined by the 'superflex' Param.\"\n    Documentation: |\n    The Param
'superflex' is an array that will contain\n    the list of flexiflow tasks to run.\n    Params:\n
flexiflow/list-parameter: \"superflex\"\n    Tasks:\n    - \"flexiflow-start\"\n    -
\"flexiflow-stop\"\n    Meta:\n    color: \"orange\"\n    icon: \"magic\"\n    title: \"RackN
Content\"\n\nNote that as the ``Documentation`` field says; the ``flexiflow/list-parameter``\n\nfor
this Stage is defined as ``superflex``. This Parameter must be defined\n\nas an array; which is a
list of the Tasks to execute during this Stage. The\n\nParam can be defined on the Machine in all of
the normal ways (directly as a\n\nParam, as part of a Profile, or as a Param in the Global Profile,
dynamically by\n\nanother content, etc).\n\n\nHere is an example stage that defines the custom FlexiFlow
stage, which sets\n\nthe control Parameter name to ``superflex``, and also defines the Tasks
that\n\n``superflex`` will reference and run.\n\n  ::\n\n    ---\n    Name: \"my-superflex-stage-
v2\"\n    Description: \"Perform tasks to run that are defined by the 'superflex' Param.\"\n\n
Documentation: |\n    The Param 'superflex' is an array that will contain\n    the list of
additional tasks to run. The actual tasks that\n    are run are also listed in this Stage.\n\n
Params:\n    flexiflow/list-parameter: superflex\n    superflex:\n    - task1\n    -
task2\n    - task3\n    Tasks:\n    - \"flexiflow-start\"\n    - \"flexiflow-stop\"\n\n
Meta:\n    color: \"orange\"\n    icon: \"magic\"\n    title: \"RackN Content\"\n\n.n.note::
Warning!! The Params that are used in FlexiFlow MUST BE TYPE DEFINED correctly\n\nand added
to the system before they are used. This use case pattern does NOT\n\nsupport using adhoc
Params.\n\n\nCreate Parameter Definition\n=====\n\nIt is critical that you
correctly type define the Parameter that defines the\n\nlist of tasks to run in advance of using the
Parameter. If you do not, you will\n\nreceive a warning message and the task injection will
fail.\n\n\nAn example type definition for the Parameter (using our ``superflex`` example\n\nabove),
would look like:\n\n  ::\n\n    ---\n    Name: \"superflex\"\n    Description: \"Defines the
flexiflow list parameter.\"\n    Documentation: |\n    This param defines the list of Tasks to
execute for flexiflow\n    during the stage run. If left empty, no tasks will be run.\n\n
Schema:\n    type: \"array\"\n    items:\n    type: \"string\"\n    default: []\n\n
Meta:\n    color: \"blue\"\n    icon: \"magic\"\n    title: \"RackN Content\"\n\nPossible
Errors\n=====\n\nHere is a list of some of the possible errors you may encounter.\n\n  ::\n\n
Failed to render actions: : template: :35:30: executing \"flexiflow_start.sh\" at
<.ComposeParam>: error calling ComposeParam: Cannot compose freeform param superflex\n\nUpdated
job for flexiflow-stage:flexiflow:flexiflow-start to incomplete\n\n\nIn this case, the ``superflex``
list parameter must be defined in advance of using it,\n\nthe operator has attempted to use the
Parameter as an adhoc param. Create a Param\ntype definition as specified in the *Create Parameter
Definition* section above.\n\n\nOperating FlexiFlow Workflow Based Manipulation\n-----\n\n\nThe FlexiFlow content also allows you to add a Stage to a given
workflow. At that Stage,\n\nthe system will reference the Param named ``flexiflow/workflow``. The
value of the\n\nparam should be the name of an existing Workflow. That workflow will then be
executed\n\non the system.\n\n\nPlease review the ``test-flexiflow-workflow-main`` Workflow which will
dynamically add the\ntest workflow named ``test-flexiflow-workflow-include``. This provides a
working example\n\nof using the include workflow capability.",
"Icon": "magic",

```

```
"License": "APLv2",
  "Name": "flexiflow",
  "Order": "1000",
  "Overwritable": false,
  "Prerequisites": "",
  "RequiredFeatures": "",
  "Source": "RackN",
  "Tags": "advanced,flexiflow,physical",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "path": "image-deploy",
  "size": 14352384
}
{
  "path": "docker-context",
  "size": 11382784
}
{
  "Counts": {
    "blueprints": 1,
    "contexts": 2,
    "params": 55,
    "profiles": 7,
    "tasks": 3,
    "templates": 22
  },
  "Warnings": [
    "Layer docker-context has rackn-license as a prerequisite, but rackn-license does not exist!"
  ],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "blue",
    "Copyright": "RackN 2020",
    "Description": "Cloud Specific Integrations",
    "DisplayName": "Cloud Wrappers",
    "DocUrl": "https://provision.readthedocs.io/en/tip/doc/content-packages/cloud-wrappers.html",
    "Documentation": "This library contains items that help run Digital Rebar manage machines on public clouds. It uses Terraform tasks to create/delete machines and Ansible tasks join the machine to install the Digital Rebar runner. Once the runners starts, it will collect ncloud specific data if a Metadata API is available.\n\nTL;DR: `cloud-provision` uses the v4.8 Resource Brokers to create and attach machines to Terraform accessible platform.\n\nRequirements\n-----\n\nInbound Access\n=====\n\nThe Digital Rebar Server must be at a location that is accessible to the machines being provisioned. This is required because the machines must be able to download the `join-up` script from the server using port 8090.\n\nOutbound Access\n=====\n\nIs NOT required unless you are using a cloud provider that requires SSH into the newly created machines.\n\nAs of v4.8, none of the major cloud providers (AWS, Azure, Google, Linode, Digital Ocean) required SSH to join-up.\n\nCatalog Items\n=====\n\nThe Cloud Wrapper requires Contexts because it uses Runner and Terraform. If SSH is required then the Ansible Context is used.\n\nSetting Up Cloud Brokers\n-----\n\nWhen you create a Cloud Broker, you must set Security credentials for each cloud.\n\nThe `cloud-profiles` script<https://gitlab.com/rackn/provision-content/-/blob/v4/tools/cloud-profiles.sh> in the RackN provision-content repo can be used to create the\n\nAWS\n=====\n\n* aws/access-secret\n* aws/access-key-id\n\nAdditional values, e.g. region, image and instance type, have safe defaults but should be reviewed.\n\nGoogle\n=====\n\n* google/credential - this is a copy of contents from the JSON file Google provides\n\nAdditional values, e.g. region, image and instance type, have safe defaults but should be reviewed.\n\nLinode\n=====\n\n* linode/token\n\nAdditional values, e.g. region, image and instance type, have safe defaults but should be reviewed.\n\nOptional Values\n=====\n\nWhen possible, the machine on the cloud provider is given the name of the machine in Digital Rebar.\n\nThe reference terraform plan will create tags on the cloud provider
```

```
based on the assigned profiles. It also creates one called \"digitalrebar.\" This can be handy to
find or manage the machines on the cloud provider.",
  "Icon": "cloud",
  "License": "APLv2",
  "Name": "cloud-wrappers",
  "Order": "1000",
  "Overwritable": false,
  "Prerequisites": "BasicStore:>=4.8.0,drp-community-content:>=4.8.0,task-library:>=4.8.0,docker-
context:>=4.8.0",
  "RequiredFeatures": "sane-exit-codes, job-exit-states, fsm-runner, workflows, default-workflow,
http-range-header, roles, tenants, sprig, multiarch, overridable-bootloaders, centos-8, virtual-
media-boot, secure-file-server",
  "Source": "RackN",
  "Tags": "advanced,cloud,integration",
  "Type": "dynamic",
  "Version": "v4.9.8",
  "Writable": false
}
}
{
  "Counts": {
    "contexts": 1,
    "params": 7,
    "stages": 2,
    "tasks": 9,
    "workflows": 2
  },
  "Warnings": [
    "Layer docker-context has rackn-license as a prerequisite, but rackn-license does not exist!"
  ],
  "meta": {
    "Author": "RackN",
    "CodeSource": "https://gitlab.com/rackn/provision-content",
    "Color": "black",
    "Copyright": "RackN",
    "Description": "Solidfire discovery and baseline support",
    "DisplayName": "Solidfire",
    "DocUrl": "https://provision.readthedocs.io/en/tip/doc/content-packages/solidfire.html",
    "Documentation": "",
    "Icon": "ship",
    "License": "RackN",
    "Name": "solidfire",
    "Order": "1000",
    "Overwritable": false,
    "Prerequisites": "docker-context",
    "RequiredFeatures": "",
    "Source": "RackN",
    "Tags": "jpmc,solidfire,enterprise",
    "Type": "dynamic",
    "Version": "v4.9.0",
    "Writable": false
  }
}
{
  "Counts": {
    "blueprints": 4,
    "params": 132,
    "profiles": 46,
    "stages": 149,
    "tasks": 7,
    "templates": 3,
    "ux_views": 1,
    "version_sets": 2,
    "workflows": 19
  },
},
```

```
"Warnings": [
  "Layer docker-context has rackn-license as a prerequisite, but rackn-license does not exist!"
],
"meta": {
  "Author": "RackN+universal",
  "CodeSource": "RackN Tree",
  "Color": "olive",
  "Copyright": "RackN 2021",
  "Description": "universal Workflows and Data",
  "DisplayName": "Universal Workflow Content",
  "DocUrl": "",
  "Documentation": "Universal Workflows (aka Pipelines)\n-----
\n\nUniversal Workflow architectural, :ref:`rs_universal_arch`, and Universal Workflow operations,
:ref:`rs_universal_ops`,\ncontain more details about the universal workflow system.",
  "Icon": "map signs",
  "License": "RackN",
  "Name": "universal",
  "Order": "1000",
  "Overwritable": false,
  "Prerequisites": "BasicStore:>=4.9.0,drp-community-
content:>=4.9.0,classify,ipmi,burnin,rack,raid,bios,flash,dell-support,hpe-support,hardware-
tooling,task-library:>=4.9.0,vmware,callback,lenovo-support,validation,flexiflow,image-deploy,cloud-
wrappers,solidfire",
  "RequiredFeatures": "",
  "Source": "RackN",
  "Tags": "universal,enterprise,pipelines",
  "Type": "dynamic",
  "Version": "v4.9.3",
  "Writable": false
}
}
>>> Installing initial profiles...
{
  "Address": "",
  "Arch": "amd64",
  "Available": true,
  "BootEnv": "local",
  "Bundle": "",
  "Context": "",
  "CurrentJob": "",
  "CurrentTask": 0,
  "Description": "",
  "Endpoint": "",
  "Errors": [],
  "Fingerprint": {
    "CSNHash": "",
    "CloudInstanceID": "",
    "MemoryIds": [],
    "SSNHash": "",
    "SystemUUID": ""
  },
  "HardwareAddr": [],
  "Locked": false,
  "Meta": {
    "feature-flags": "change-stage-v2",
    "machine-role": "machine"
  },
  "Name": "52-54-00-f9-08-a4",
  "OS": "",
  "Params": {
    "machine-self-runner": true
  },
  "Partial": false,
  "PendingWorkOrders": 0,
  "Pool": "self-runners",
```

```
"PoolAllocated": false,
"PoolStatus": "Free",
"Profiles": [
  "bootstrap-drp-endpoint"
],
"ReadOnly": false,
"RetryTaskAttempt": 0,
"Runnable": true,
"RunningWorkOrders": 0,
"Secret": "Jywt6TrYbc1437o",
"Stage": "none",
"TaskErrorStacks": [],
"Tasks": [],
"Uuid": "ffffffff-ffff-4b58-9def-bc4ef1aca9fa",
"Validated": true,
"WorkOrderMode": false,
"Workflow": "",
"WorkflowComplete": true
}
Warning: Profile bootstrap-cloud-wrappers doesn't exist - skipping
Warning: Profile bootstrap-solidfire doesn't exist - skipping
>>> Setting initial workflow to 'universal-bootstrap' for Machine '52-54-00-f9-08-a4'
{
  "Address": "",
  "Arch": "amd64",
  "Available": true,
  "BootEnv": "local",
  "Bundle": "",
  "Context": "",
  "CurrentJob": "",
  "CurrentTask": -1,
  "Description": "",
  "Endpoint": "",
  "Errors": [],
  "Fingerprint": {
    "CSNHash": "",
    "CloudInstanceID": "",
    "MemoryIds": [],
    "SSNHash": "",
    "SystemUUID": ""
  },
  "HardwareAddrs": [],
  "Locked": false,
  "Meta": {
    "feature-flags": "change-stage-v2",
    "machine-role": "machine"
  },
  "Name": "52-54-00-f9-08-a4",
  "OS": "",
  "Params": {
    "machine-self-runner": true
  },
  "Partial": false,
  "PendingWorkOrders": 0,
  "Pool": "self-runners",
  "PoolAllocated": false,
  "PoolStatus": "Free",
  "Profiles": [
    "bootstrap-drp-endpoint"
  ],
  "ReadOnly": false,
  "RetryTaskAttempt": 0,
  "Runnable": true,
  "RunningWorkOrders": 0,
  "Secret": "Jywt6TrYbc1437o",
```

```
"Stage": "start",
"TaskErrorStacks": [],
"Tasks": [
  "stage:start",
  "update-pipeline",
  "gohai",
  "set-machine-ip-in-joinup",
  "ssh-access",
  "stage:universal-bootstrap-start-callback",
  "callback-task",
  "stage:universal-bootstrap-pre-flexiflow",
  "flexiflow-start",
  "flexiflow-stop",
  "stage:universal-bootstrap",
  "universal-bootstrap-prefs",
  "bootstrap-discovery-iso",
  "bootstrap-ssh",
  "bootstrap-network",
  "stage:universal-bootstrap-post-flexiflow",
  "flexiflow-start",
  "flexiflow-stop",
  "stage:universal-bootstrap-classification",
  "classify-stage-list-start",
  "classify-stage-list-stop",
  "stage:universal-bootstrap-post-validation",
  "validation-start",
  "validation-stop",
  "stage:universal-bootstrap-complete-callback",
  "callback-task",
  "stage:universal-chain-workflow",
  "universal-chain-workflow",
  "stage:complete",
  "bootenv:local",
  "context:"
],
"Uuid": "ffffffff-ffff-4b58-9def-bc4ef1aca9fa",
"Validated": true,
"WorkOrderMode": false,
"Workflow": "universal-bootstrap",
"WorkflowComplete": false
}
{
  "Path": "/bootstrap/dr-provision.zip",
  "Size": 82669736
}
{
  "Path": "/bootstrap/install.sh",
  "Size": 79141
}
}
### With Digital Rebar started, complete the setup using the System Install Wizard
open https://[host ip]:8092 (accept self-signed TLS certificate)

### Remember to wait for bootstrapping to complete by watching the self-runner machine.

### Optionally, upload popular community operating system ISOs
drpccli bootenvs uploadiso ubuntu-20.04-install
drpccli bootenvs uploadiso centos-8-install
```


16 REFERENCES

- [1] RackN, “Digital Rebar is Self-Managed Software,” [Online]. Available: <https://docs.rackn.io/en/latest/doc/install/self-managed.html>. [Accessed 2022].
- [2] RackN, “Bare metal Provisioning,” [Online]. Available: <https://rackn.com/solutions/bare-metal/>. [Accessed 2022].
- [3] RackN, “RackN - Market Positioning,” 2021.
- [4] Hyperscalers, “S5X 2.5" | D53X-1U,” [Online]. Available: <https://www.hyperscalers.com/storage/storage-servers/hyperscalers-S5X-D53X-1U-ice-lake-densest-hyperscale-server-nvme-drives-buy>. [Accessed 2022].
- [5] Hyperscalers, “S5K D43K-1U,” [Online]. Available: <https://www.hyperscalers.com/quanta-qct-server-1u/AMD-hyperscale-server-3rd-generation-milan-CPU-S5K-D43K-1U>. [Accessed 2022].
- [6] RackN, “Introduction - Key features and Differentiators,” 2020.
- [7] RackN, “Infrastructure Pipelines and Bootstrapping Data centers,” 2021.
- [8] Hyperscalers, “About HS,” [Online]. Available: <https://www.hyperscalers.com/about-us-hyperscalers>.
- [9] RackN, “About RackN,” [Online]. Available: <https://rackn.com/about/about-rackn/>. [Accessed 2022].
- [10] Canonical, “Ubuntu 20.04.4 LTS (Focal Fossa),” [Online]. Available: <https://releases.ubuntu.com/20.04.4/>. [Accessed 2022].
- [11] Hyperscalers, “S5Z | T43Z-2U,” [Online]. Available: <https://www.hyperscalers.com/quanta-qct-servers-distribution-USA/buy-S5Z-T43Z-2U-4node-distributor-usa-aus-ocp-qct-distribution-lease-icelake>. [Accessed 2022].
- [12] RackN, “Preparing to Run Digital Rebar,” [Online]. Available: <https://docs.rackn.io/en/latest/doc/setup/preparing.html>.
- [13] RackN, “RackN Digital Rebar documentation,” [Online]. Available: <https://docs.rackn.io/en/latest/doc/quickstart.html>. [Accessed 2022].
- [14] RackN, “Image deploy,” [Online]. Available: <https://docs.rackn.io/en/latest/doc/content-packages/image-deploy.html?highlight=image%20deploy>. [Accessed 2022].
- [15] RackN, “Image builder,” [Online]. Available: <https://docs.rackn.io/en/latest/doc/content-packages/image-builder.html?highlight=image%20build>. [Accessed 2022].

p +61 1300 113 112
e info@hyperscalers.com

Solving Information Technology's
Complexity



Index

A	
Addendum	36
Additional Setup and Deployment.....	35
Appliance Optimizer Utility AOU	12
Audience and Purpose	11
B	
Base Product Deployment	16
D	
Digital IP Appliance Design Process	12
I	
Important Considerations.....	11
Infrastructure Setup	14
Introduction	3
P	
Prerequisites for updating.....	35
T	
Testing the Appliance	35
Troubleshooting DPX Appliance	36
U	
Updating the Appliance.....	35