

Dell Cloud Solution  
for Web Applications

# **Administrator Guide**



# Notes



**NOTE:** A NOTE indicates important information that helps you make better use of your computer.

---

**Information in this document is subject to change without notice.**

**© 2010 Dell Inc. All rights reserved.**

Reproduction of these materials in any manner whatsoever without the written permission of Dell Inc. is strictly forbidden.

Trademarks used in this text: Dell™, the DELL logo, PowerEdge™, and PowerConnect™ are trademarks of Dell Inc. UNIX® is a registered trademark of The Open Group in the United States and other countries.

Other trademarks and trade names may be used in this publication to refer to either the entities claiming the marks and names or their products. Dell Inc. disclaims any proprietary interest in trademarks and trade names other than its own.

**December 2010    Rev. A00**

# Contents

|   |           |
|---|-----------|
| <b>INTRODUCTION .....</b>                               | <b>7</b>  |
| Cloud Definition .....                                  | 7         |
| Joyent SmartMachines .....                              | 7         |
| Cloud Management .....                                  | 8         |
| Documentation .....                                     | 8         |
| <b>Solution Components .....</b>                        | <b>8</b>  |
| Hardware Components .....                               | 9         |
| Software Components .....                               | 10        |
| Measuring Consumption .....                             | 14        |
| Cloud Provisioning.....                                 | 14        |
| <b>NETWORK CONFIGURATION AND VLANS.....</b>             | <b>15</b> |
| Network Overview .....                                  | 15        |
| Default Networks .....                                  | 15        |
| Triangle Looped Configuration (Fallback) .....          | 21        |
| Switch Port Assignments .....                           | 23        |
| <b>BIOS AND RAID CONFIGURATIONS .....</b>               | <b>25</b> |
| RAID Settings for Provisioned Nodes .....               | 25        |
| RAID Settings for Compute Nodes .....                   | 25        |
| BIOS Settings for PowerEdge C Servers (C1100/2100)..... | 25        |
| BIOS Settings for PowerEdge Servers (R610/R710) .....   | 26        |
| <b>INSTALLING THE SOFTWARE .....</b>                    | <b>27</b> |
| Installation Steps .....                                | 27        |
| Deploying Compute Nodes .....                           | 27        |
| Setting Up Redundant Administration Server.....         | 29        |
| Installing Standard Templates .....                     | 29        |
| Checking Status of Compute Nodes .....                  | 30        |
| <b>Accessing Cloud Control Under SSL .....</b>          | <b>31</b> |

|   |           |
|---|-----------|
| <b>COMPUTE NODES .....</b>                                  | <b>33</b> |
| <b>Accessing the Global Zone .....</b>                      | <b>33</b> |
| <b>Common Commands .....</b>                                | <b>33</b> |
| Zlogin .....  | 33        |
| ZFS .....   | 33        |
| Zpool .....   | 34        |
| ZoneAdm.....  | 35        |
| Prstat -Z.....  | 35        |
| <b>ZEUS TRAFFIC MANAGER .....</b>                           | <b>37</b> |
| <b>Provisioning a Zeus SmartMachine.....</b>                | <b>37</b> |
| <b>Accessing Zeus .....</b>                                 | <b>38</b> |
| <b>Zeus Traffic Manager Documentation .....</b>             | <b>38</b> |
| <b>CUSTOM TEMPLATES .....</b>                               | <b>39</b> |
| <b>Working with Custom Templates.....</b>                   | <b>39</b> |
| Creating Custom Templates on the Server .....               | 39        |
| Transferring the Template to Other Servers .....            | 40        |
| Preparing Cloud Control to Recognize the New Template ..... | 41        |
| <b>FAILOVER PROCEDURE.....</b>                              | <b>45</b> |
| <b>CUSTOMIZING NOTIFICATIONS .....</b>                      | <b>47</b> |
| <b>BACKING UP ZONE DATA .....</b>                           | <b>48</b> |
| What Gets Backed Up .....                                   | 48        |
| <b>API REFERENCE.....</b>                                   | <b>51</b> |

|  |           |
|--|-----------|
| <b>Private Cloud API .....</b>                         | <b>51</b> |
| General Considerations .....                           | 51        |
| Differences Between Public and Private Cloud APIs..... | 53        |
| Resource Models.....                                   | 53        |
| <b>Resource /customers.....</b>                        | <b>54</b> |
| General Methods Description .....                      | 54        |
| Index Action.....                                      | 55        |
| Customers Collection Pagination .....                  | 56        |
| Show Action .....                                      | 57        |
| Create Action.....                                     | 58        |
| Update Action.....                                     | 59        |
| Delete Action .....                                    | 60        |
| Automatic Provision and RAM Quotas .....               | 60        |
| <b>Resource /templates .....</b>                       | <b>61</b> |
| General Methods Description .....                      | 61        |
| Index Action.....                                      | 61        |
| Templates Collection Pagination.....                   | 62        |
| Show Action .....                                      | 63        |
| <b>Resource /servers.....</b>                          | <b>64</b> |
| General Methods Description .....                      | 64        |
| Index Action.....                                      | 64        |
| Servers Collection Pagination.....                     | 65        |
| Show Action .....                                      | 66        |
| <b>Container's State Machine and REST.....</b>         | <b>66</b> |
| Transition Representation Format.....                  | 67        |
| XML.....   | 68        |
| JSON .....   | 68        |
| Containers Representation Format .....                 | 68        |
| JSON .....   | 69        |
| XML.....   | 69        |
| The Special Property <i>Running_Status</i> .....       | 70        |
| <b>Container Tasks .....</b>                           | <b>70</b> |
| Task: Create a New Container.....                      | 70        |
| Task: Reboot a Container.....                          | 71        |
| Task: Shutdown a Container.....                        | 72        |
| Task: Start Up a Container .....                       | 72        |

|  |            |
|--|------------|
| Task: Destroy a Container .....                          | 73         |
| Zones/Containers Finite State Machine .....              | 73         |
| RESTful Resource "Containers" .....                      | 74         |
| General Methods Description .....                        | 74         |
| Transitions.....   | 75         |
| General Response Consideration .....                     | 75         |
| Determining the Current State for a Zone/Container ..... | 76         |
| Containers Collection Pagination .....                   | 78         |
| <b>API XML Schemas .....</b>                             | <b>78</b>  |
| Customers Collection .....                               | 78         |
| <b>Resource /search .....</b>                            | <b>85</b>  |
| General Methods Description .....                        | 85         |
| Search Containers (Zones).....                           | 86         |
| Search Customers.....                                    | 87         |
| Search IP Addresses.....                                 | 88         |
| Search Servers.....                                      | 89         |
| Search Templates (Zone Configurations) .....             | 90         |
| <b>Collector Agent .....</b>                             | <b>90</b>  |
| Issuing Collector Agent Commands .....                   | 91         |
| Command Example .....                                    | 93         |
| Using a Client With the Collector Agent .....            | 94         |
| <b>GLOSSARY.....</b>                                     | <b>95</b>  |
| <b>GETTING HELP .....</b>                                | <b>99</b>  |
| <b>Contacting Dell.....</b>                              | <b>99</b>  |
| <b>INDEX.....</b>  | <b>101</b> |

# Introduction

The Dell Cloud Solution for Web Applications (DCSWA) is an optimized private cloud solution for running virtualized web applications, databases, and other compute nodes efficiently. The solution includes hardware, software, and services and is highly scalable from the lab to massive data centers.

The solution is offered as a turnkey package containing software, hardware, and core services (installation and support).

## Cloud Definition

In DCSWA, a cloud is defined as a collection of interconnected pods, racks, and nodes. Each pod is a collection of up to 12 racks, with each rack containing up to 15 compute nodes to handle a multitude of web applications.

Every pod has an Administration/Provisioning server (PS) that manages the compute nodes. In DCSWA, the Joyent Cloud Control (CC) software component is housed in the PS of one of the pods in the cloud. Performance and redundancy are handled by scaling out the application to multiple nodes in the cloud.

## Joyent SmartMachines

A SmartMachine is the Joyent SmartOS Unix para-virtualized virtual machine (VM). SmartMachines are zones (ZFS datasets and zone configurations). Formerly called “accelerators”, SmartMachines are optimized VMs running on the compute nodes. They use dedicated server virtualization to manage fluctuating loads by bursting onto additional CPUs that have available resources. Customers use SmartMachines to run components of their web applications.

In addition to customer SmartMachines, the solution provides the virtual Zeus Load Balancer to enable scaling of the web application across multiple SmartMachines. With a traditional multi-tiered web application, the load balancers can be inserted between all three tiers (web, applications, and database layers) to provide performance and redundancy. An additional MySQL optimized SmartMachine is provided to facilitate rapid deployment of database components for applications. This set of three SmartMachines represents the basis of the platform as a service (PaaS) environment.

## **Cloud Management**

The cloud is managed, maintained, secured, and backed up through software components. The management components provide reporting, monitoring, operating, and diagnostic functions at both the cloud administrator and cloud user levels. The solution defines a maintenance and backup/disaster recovery methodology for both the SmartMachines and infrastructure components.

Wrapped around all of these features are security features and methodologies that allow for secure multi-tenant operations.

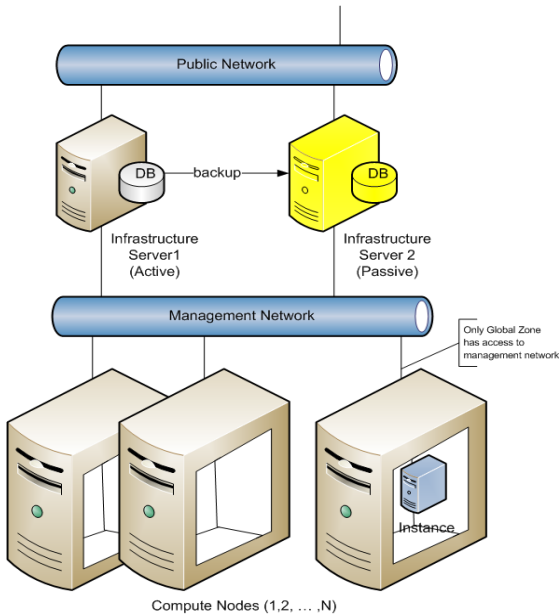
## **Documentation**

The solution provides guides for troubleshooting and maintaining the cloud. Documentation supporting the developers is provided to assist with migrating web applications to the cloud. Dell Services will assist in sizing, deployment, and installation.

## **Solution Components**

The solution is comprised of hardware and software components, which are described below.





**Figure 1. High-Level Solution Architecture**

## Hardware Components

Three hardware components form the solution:

- Administration/Provisioning Server
- Compute Nodes
- Network Infrastructure

### Administration/Provisioning Server

The Administration/Provisioning Server provides the multiple cloud control, user portal, and provisioning functions for the solution. The management network provides connectivity between the infrastructure servers and the nodes.

### Compute Nodes

The compute nodes are a collection of services, referred to as *client services* that run on each physical server except the Infrastructure Server. The services

implement the components of the stack that provision and monitor user SmartMachines running on the nodes.

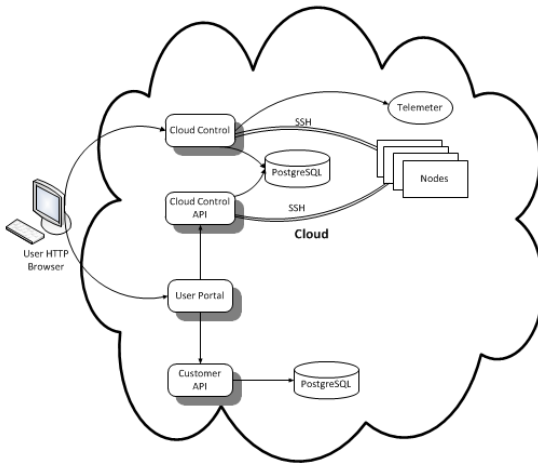
### Network Infrastructure

The network infrastructure consists of the collection of top-of-rack (ToR) switches and core switches that facilitate the inner-connectivity between the Administration/Provisioning Server and respective compute nodes, and the customer's networking infrastructure.

## Software Components

Joyent Cloud Software is the suite of cloud management software that includes:

- Joyent Cloud Control
- Joyent Cloud Management API
- User Portal
- Joyent Telemeter



**Figure 2. Cloud Components**

### Joyent Cloud Control

In the Dell Cloud Solution, the administration portal is referred to as Cloud Control. Cloud Control manages cloud operations, which include locations (datacenters), racks, sets of racks (pods), rack-mountable devices (load

balancers, servers, console servers, storage devices, switches, appliances and routers), the network (IPs, subnets, and virtual IPs), zones/SmartMachines (ZFS datasets and zone configurations), and customers.

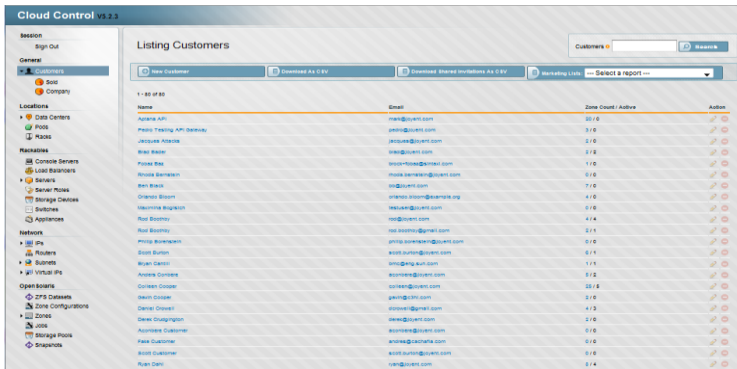
Cloud Control also includes reporting for various objects in the cloud infrastructure. Most Cloud Control functions focus on inventory management, IP assignment, and associations to other objects in the datacenter: that is, managing servers and zones.



**Figure 3. Cloud Control**

### Cloud Management API

This programmatic interface enables system integrators to access Cloud Control’s functionality via a RESTful API.




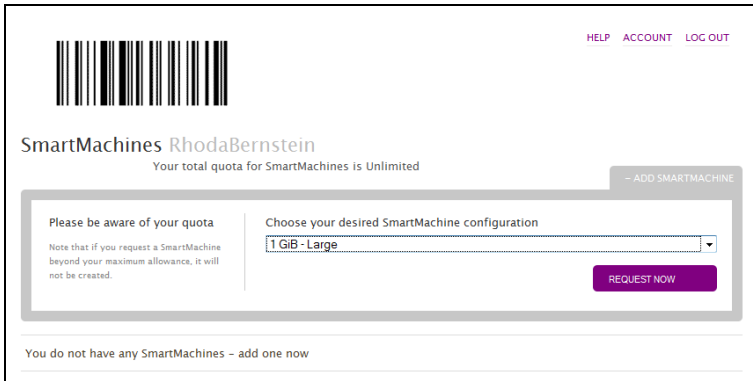
**Figure 4. Main Cloud Control Screen**

## User Portal

The self-service User Portal provides customers the means to perform certain tasks themselves. Through the User Portal, customers can:

- Create SmartMachines (up to their quota)
- Reboot a SmartMachine
- Shutdown a SmartMachine
- Delete a SmartMachine

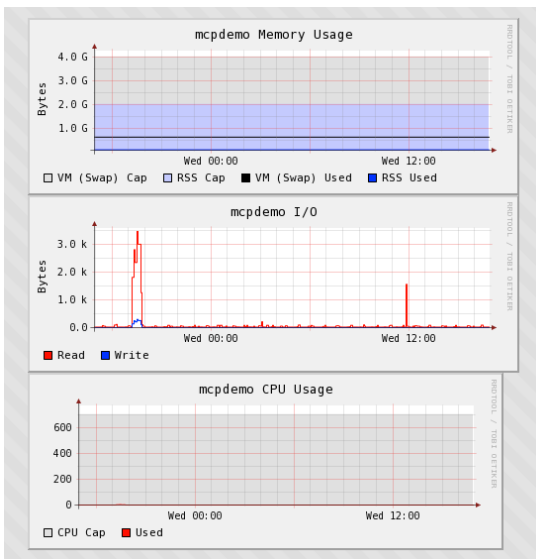
 **NOTE:** Capacity can be added to an existing SmartMachine by an administrator using Cloud Control, provided that there is available quota for the user.



**Figure 5. Sample User Portal Screen**

## Joyent Telemeter

The telemeter measures consumption. All telemeter information is provided in graphical form for each zone in Cloud Control.



**Figure 6. Sample Telemeter Graphs**

## Measuring Consumption

Measuring consumption is important in both private and public clouds. In private clouds, consumption measurement supports departmental charge-backs. In public clouds, a consumption measurement supports both the pay-in-advance (subscriptions) and pay-afterwards (invoicing) models.

Because consumption requirements change over time, initially allocated resources may no longer match customer needs. Therefore, it is critical to meter resource allocations on an ongoing basis so that customer invoices and reports reflect actual use.

Consumption measurement is also valuable for tracking the amount of resources actually consumed (subset) versus the amount allocated. Comparing the subset to the full allocation can enable greater efficiencies. This is especially important for service providers in capacity planning. The system provides measures of consumption for both customers and service providers.

## Cloud Provisioning

Clouds must be created. This process, called *cloud provisioning*, requires several steps.

- 1 Install the software.
- 2 Provision the network.
- 3 Execute the JumpStart.
- 4 Configure the head node.
- 5 Set up the global zone.
- 6 Set up compute node(s).

Cloud Control uses the Ubuntu Linux operating system because it is fast and secure. Ubuntu is integrated into the software and is installed automatically.

# Network Configuration and vLANs

## Network Overview

The basic premise of the architecture is to provide physical redundancy (excluding the POC/Lab Cloud deployment) and reduced configuration requirements. This is handled by defining a basic switching topology that is extensible and creates a stacked logical switch from multiple physical switches, which can be configured as a group. These physical topologies allow for redundancy and scale while maintaining high network bandwidth between SmartMachines, regardless of their placement within the environment. Larger deployments (greater than three racks) require a core switch that takes over the layer 3 routing from the ToR switches.

## Default Networks

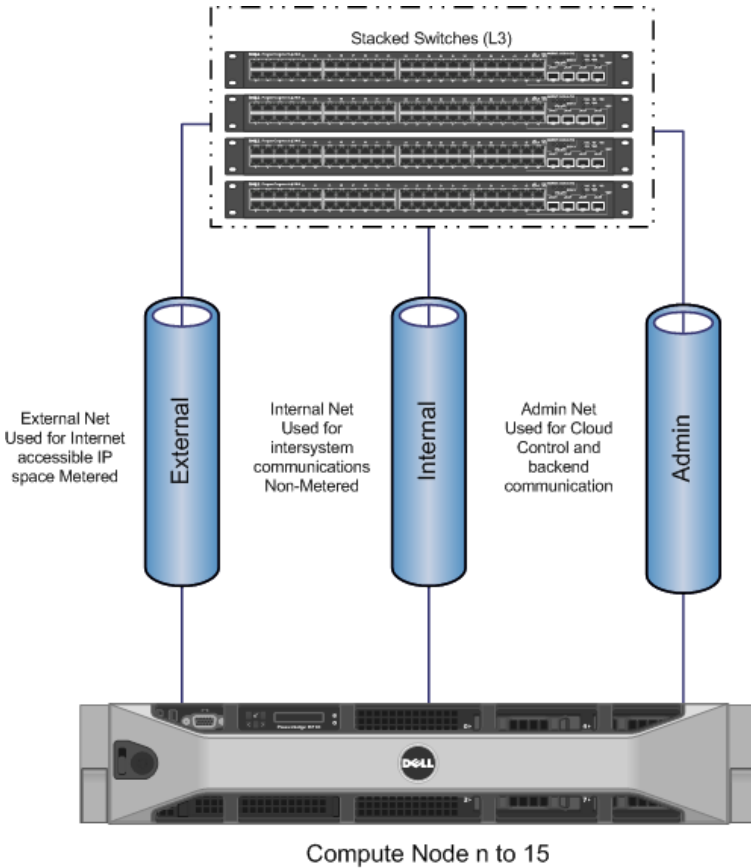
The solution relies heavily on vLAN technologies to separate the different networks. Three networks are configured by default, and all exist in all configurations.

*External* vLANS—these vLANs are typically addressed with internet routable addresses and are used for connections to devices that are external to the cloud infrastructure. They include externally visible services such as load balancers and web servers. There can be one or many external vLANs depending on the need to segregate traffic among groups of SmartMachines.

*Internal* vLANs—these vLANs are typically addressed with private addresses and are used for communications between SmartMachines. SmartMachines that perform front-end and back-end functions should be configured to communicate via internal vLANs. There can be one or many of these networks depending on the need to segregate traffic among groups of SmartMachines.

*Administrative* vLAN—This vLAN is used for administrative functions, such as Cloud Control tasks, keyboard/video/mouse (KVM) switches, intelligent platform management interface (IPMI), baseboard management controller (BMC) architecture, system logs, backups, and other monitoring or administrative processes. SmartMachines do *not* have access to this network. There is only one vLAN set up for the administrative function and it spans across the entire pod. Unlike the ports for the external and internal vLANs, ports for the administrative vLAN are not tagged.

*Non-default* vLANs—to use other than the default external, internal, or administrative vLANs, SmartMachines *must* use dedicated Crossbow NICs.



**Figure 7. Logical Network Diagram**



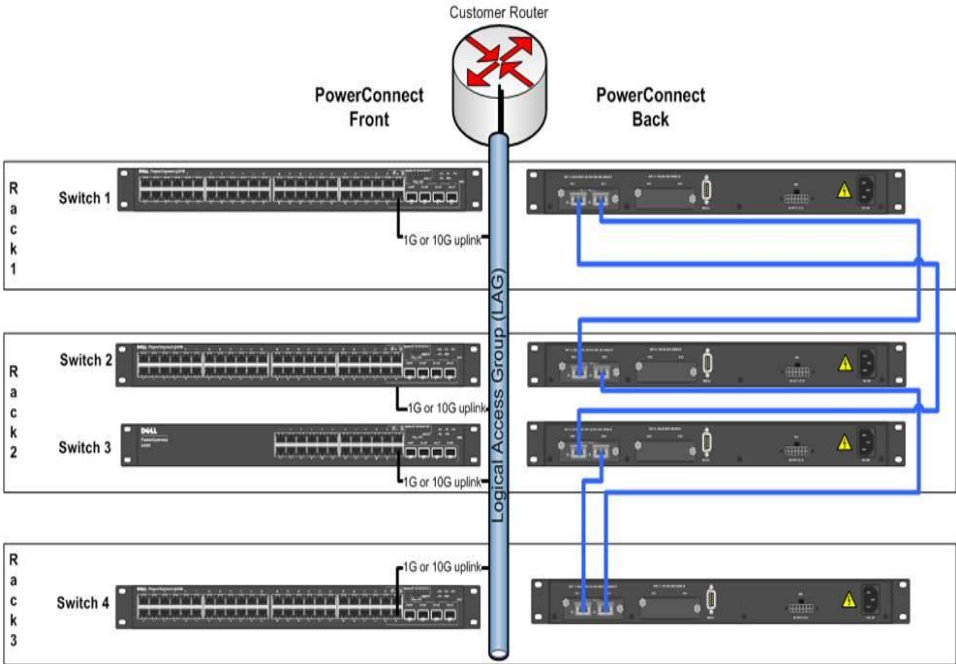
When more than one rack of equipment (greater than fourteen compute nodes) is deployed, the top-of-rack switches continue to perform layer 3 routing and are logically stacked together. Stacking the switches offers some significant benefits.

## Benefits

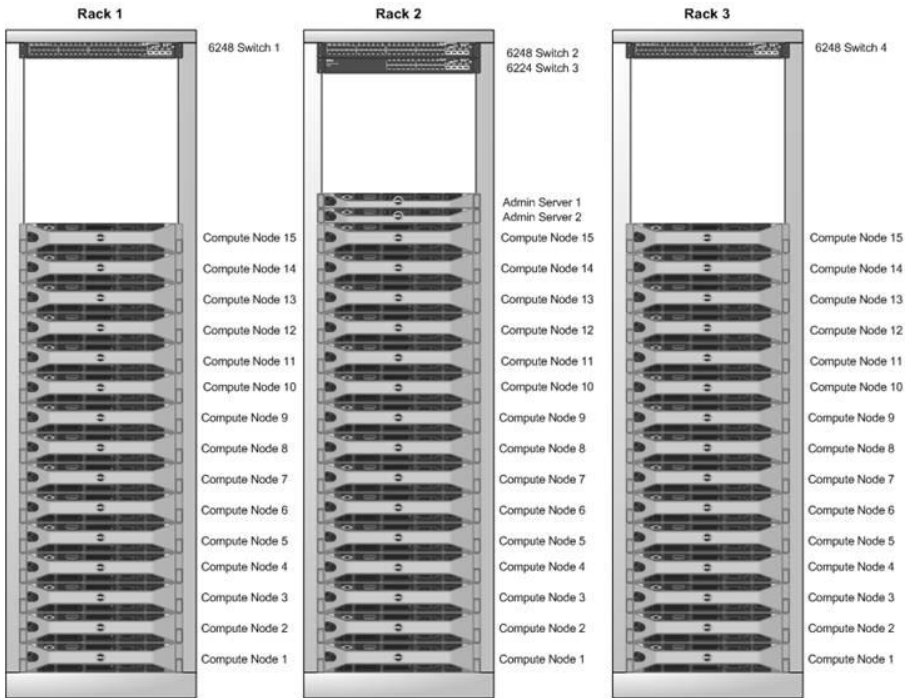
- **Improved Manageability:** All switches in the stack are managed as a single switch.
- **Efficient Spanning Tree:** The stack is viewed as a single switch by the Spanning Tree Protocol.
- **Link Aggregation:** Stacking multiple switches in a chassis allows a LAG across ports on different switches in the stack.
- **Reduced Network Traffic:** Traffic between the individual switches in a stack is passed across the stacking cable, reducing the amount of traffic passed upstream to network distribution switches.
- **Higher Speed:** The stacking module supports a higher data rate than the 10GbE uplink module (supports 12Gb per stack port offering 24Gb between switches).
- **Lower Cost:** Uplink ports are shared by all switches in the stack, reducing the number of distribution switch ports necessary to connect modular servers to the network.
- **Simplified Updates:** The basic firmware management commands propagate new firmware versions and boot image settings to all switch stack members.

## Drawbacks

- Stacking cables are proprietary and only come in 1m and 3m lengths, which limits the distance between switches.
- Stacking requires a ring topology for redundancy, which may be impacted by this distance limitation.
- Errors in configuration propagate throughout the stack immediately.



**Figure 8. Three Rack Stacked Network Configuration**



**Figure 9. Node Placement in Three Rack Configuration**

## Hybrid Looped Configuration

The hybrid looped configuration takes the stacked three-rack configuration and combines multiple iterations using a triangle looped configuration.

It is expected that most traffic will be between nodes within the cloud (i.e., between front-end hosts and DB servers). These systems are distributed throughout the environment without consideration of their logical proximity. A potential problem with stacking a large number of switches together is the possible need for traffic to traverse the ring. As the ring extends, the traffic must pass through more switches to reach the destination. In a traditional triangle looped configuration, all traffic travelling between racks has to travel up to the core router and back down to the destination rack. As this is only one Layer 2 hop away, the uplinks have to share this traffic with traffic originating outside the environment.

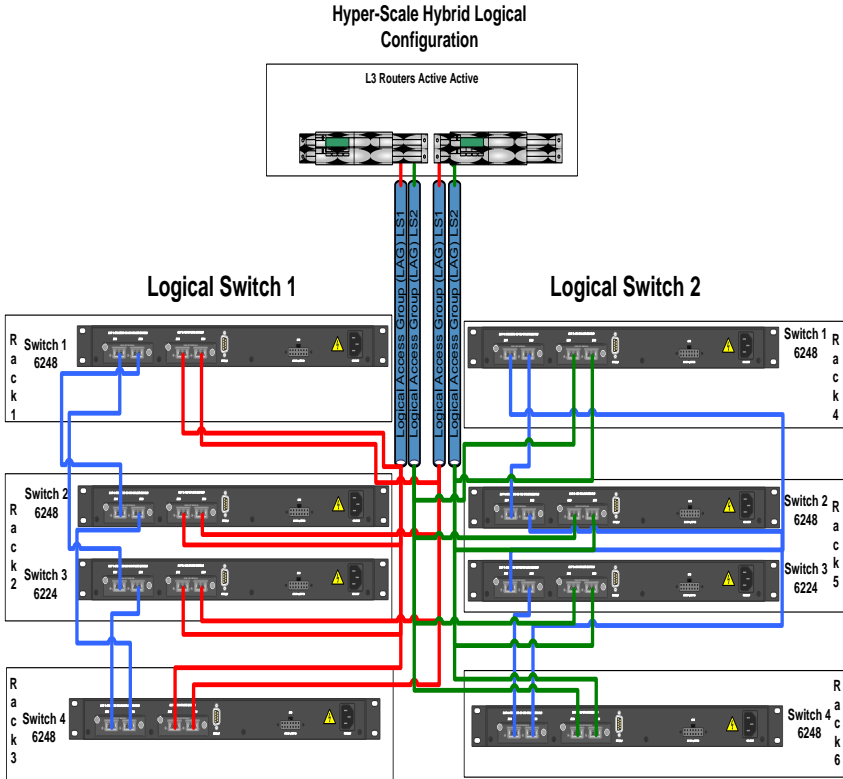
Under these circumstances, the hybrid looped configuration is more efficient than the triangle looped configuration.

This configuration is expanded utilizing the three rack stacks (four switches each). The maximum recommended limit is 12 total racks of 15 compute nodes each for a maximum pod configuration of 180 compute nodes.

### Hyperscale Physical Diagram



Figure 10. Hyperscale Rack Diagram



**Figure 11. Hybrid Looped Wiring Diagram**

### Triangle Looped Configuration (Fallback)

This configuration is a classic triangle looped configuration. The triangle topology is currently the most widely implemented in the enterprise data center. It provides a deterministic design that makes it easy to troubleshoot while providing a high level of flexibility. This is the fallback configuration used when distance limitations prevent the stacking of ToR switches. There are 2 core switches, and each ToR switch uses a 10Gb uplink to each switch.



**NOTE:** Due to port limitations, Admin nodes must be placed in racks 2 and/or 5. Subsequent racks do not require a PowerConnect 6224.

### Hyper-Scale Triangle Looped Configuration

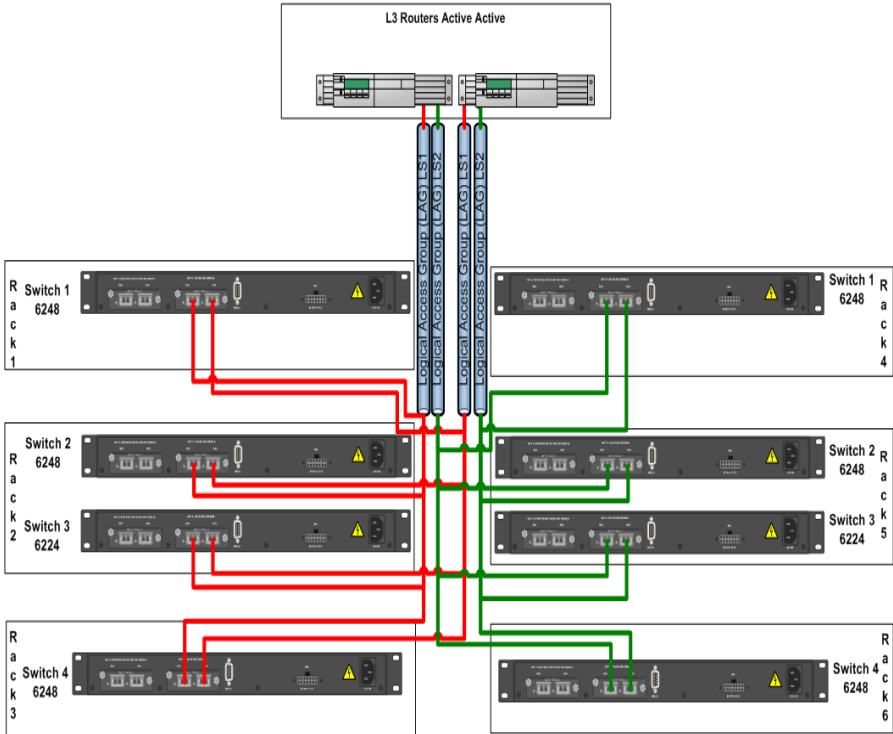


Figure 12. Hyperscale Triangle Looped Configuration Switch

## Switch Port Assignments

Regardless of the deployment, the ToR switch is either a PowerConnect 6224 or 6248. Following are the vLAN port assignments.

### 6248 Ports

| Network          | Ports       | Notes                   |
|------------------|-------------|-------------------------|
| Admin            | X/g1-X/g15  | Untagged                |
| External/Public  | X/g16-X/g30 | Tagged                  |
| Internal/Private | X/g31-X/g45 | Tagged                  |
| External/Public  | X/g46       | Untagged for admin node |
| Admin            | X/g47       | Untagged                |
| Uplink           | X/g48       | Only if required        |

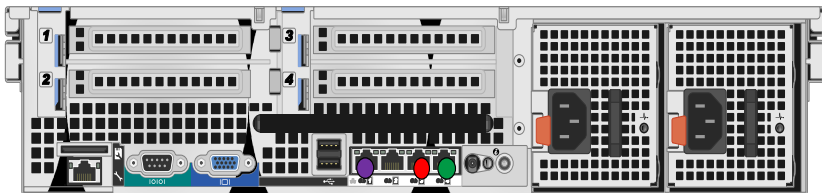
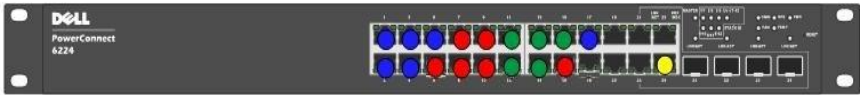
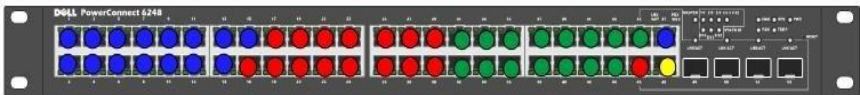
For the admin node, port g47 is used for the admin network and port g46 is used for the external network. The switch configuration is the same for all deployments. Depending on the setup, some ports are not populated. In all cases, at least two ports remain unused to allow for quick re-routing around port failures. The uplink and cross-switch connections vary depending on deployment.

### 6224 Ports

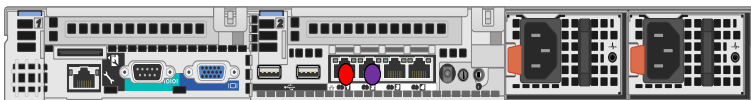
| Network          | Ports       | Notes            |
|------------------|-------------|------------------|
| Admin            | X/g1-X/g5   | Untagged         |
| External/Public  | X/g6-X/g10  | Tagged           |
| Internal/Private | X/g11-x/g15 | Tagged           |
| External/Public  | X/g16       | Untagged         |
| Admin            | X/g17       | Untagged         |
| Uplink           | X/g24       | Only if required |

For the admin node, port g17 is used for the admin network and port g16 is used for the external network. The switch configuration is the same for all deployments. Depending on the setup, some ports are not populated. The uplink and cross-switch connections vary based on deployment.

- Blue: Admin
- Red: Public
- Green: Private
- Yellow: Uplink



Compute Node



Cloud Control/ Provisioning Server

Figure 13. Port Assignments



# BIOS and RAID Configurations

## RAID Settings for Provisioned Nodes

Configure all provisioning nodes in the solution as follows:

- All drives—Labeled: “OS Volume” RAID config: RAID 10

## RAID Settings for Compute Nodes

Configure all compute nodes in the solution as follows:

- First two drives (as indicated by the server drive numbering on the front)—Labeled: “OS Volume” RAID config: RAID 1
- Remaining drives: Labeled: “Data Volume” RAID config: RAID 5—Default settings for all remaining settings

## BIOS Settings for PowerEdge C Servers (C1100/2100)

Console redirection:

- Enabled
- Serial port = COM2 (BMC or DRAC) serial port
- Serial port settings: 115200 bps, 8 data bits, 1 stop bit, no parity
- Redirection Enabled After Boot

Quick boot:

- Boot order: PERC first, then PXE NIC
- Action after power failure: Power On

VT:

- CPU setting: Enable VT
- PCI setting: Enable VT-d

BMC:

- Shared NIC
- Set IP/Gateway (done at install time by the pre-config script)
- Turn off DHCP

Disable **Force PXE boot**.

## **BIOS Settings for PowerEdge Servers (R610/R710)**

The following is a collection of all BIOS settings needed for the DCSWA solution. The following settings should *not* be considered deviations from the default settings.


- Processor Settings-> Virtualization -> **On**
- Serial Communication:
  - **ON with redirection to COM2**
  - **Port device 1=1, Port device 2=2**
  - **Fail Safe Baud = 115200**
- **Redirect After Boot = Enable**
- **Remote type = VT100/VT220**
- **Power management = Max. Perf**
- iDRAC6 settings: **DRAC mode = shared mode**

# Installing the Software

Cloud Control software is installed along with Ubuntu on the head node.


## Installation Steps

- 1 Insert the DVD into the CD ROM and reboot.
- 2 On the PS, log in as `jill/joyent`.

 **NOTE:** You will always log in as `jill/joyent`. To run a command as root, use `sudo`.

- 3 Edit `/opt/cloudcontrol/bootstrap/config.json`.

Change the settings in `config.json` to reflect your network environment.

 **NOTE:** During Cloud Control software installation, `config.json` is moved to `/data/config/config.json`.

- 4 Run the following command:

```
sudo -s /opt/cloudcontrol/install.sh
```

Cloud Control is now installed and running.

## Deploying Compute Nodes

For each compute node, complete the following:

- 1 Add the compute node to the system:
  - run: `cd /opt/joyent/bin`
  - run: `./add-host /data/config/config.json c8:0a:a9:1f:94:84 dell_c2100* joyent <hostname>`

- \* Rack server choices are `dell_c2100` or `dell_r710`.
  - The MAC should be the MAC address of the admin interface on the compute node.
  - The `joyent` at the end is the root password of the compute node.
  - You may specify an optional hostname to override the default naming scheme of `admin-1`, `admin-2`, etc.
- 2 Run the `instigate` command: `./instigate`.
    - Without arguments, it will ask you questions.
      - Use the BMC IP address
      - `c2100` is `root/root`
      - `R710` is `root/calvin`
    - It also takes arguments ( `-h [host bmc IP]` `-u [root]` `-p [password]`) to eliminate the questions
    - You can watch through this tool by adding `-w`.
    - To verify that the node is installing, use `showmount -a` to see if the node is attached to the admin.
  - 3 Edit the newly added server in Cloud Control.
    - Fill in the rack information (remember to leave space if you have more than 1 U servers).
    - Click **Update**.
  - 4 Deploy tools.
    - The server appears in the **Requiring Setup** list.
    - Each server has a **Deploy Joyprovision Tools** button at the top of its view page.
    - You can verify that the tools have been deployed if you can `ssh joyprovsn@[node ip]` directly without password or RSA key challenges.
  - 5 Mark as setup.
    - Each server has a **Mark as Setup** button at the top of its view page.
    - This step removes a server from the waiting for setup list and allows it to be provisioned for zones.

## Setting Up Redundant Administration Server

- 1 PXE boot the redundant head node from the second interface (this must be set up in the BIOS).
- 2 Wait for the OS to be installed and the device to come online.
- 3 On the currently active head node, run:  
`sudo /opt/joyent/bin/run_backups`

This command pushes the backup files to the redundant node. If the “on” field in `config.json` is set to 1, this will run automatically on an hourly basis.

## Installing Standard Templates

Three templates are included as compressed files:

- Protemplate
- MySQL
- Zeus

These templates are located on the head node in `/opt/cloudcontrol/templates`. Run the following command to deploy the templates:

- 1 Log into to head node:  
`/opt/dell/bin/finalize_servers.rb`

The script will push all the templates to all configured compute nodes and set up the metering system. This script can be run repeatedly without harm.

- 2 Verify that all the zones were properly received:

```
zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
zones                                5.24G  14.3G   35K    /zones
zones/mysql-1.0.2                    745M  14.3G   745M    /zones/mysql-
1.0.2
zones/protemplate-2.3.2              1.07G  14.3G   1.07G
/zones/protemplate-2.3.2
zones/zxtm-6.0.4                     363M  14.3G   363M    /zones/zxtm-6.0.4
```

## Checking Status of Compute Nodes

On the admin node, run:

```
/opt/dell/bin/collect_information.rb
```

This script prints to standard out a comma-separated stream of data for each server. This is intended to be used to validate and store the status of the compute nodes in the cloud.

## Accessing Cloud Control Under SSL

Cloud Control makes its services available over unencrypted HTTP. Cloud Control runs on port 8080, and the user portal runs on port 8083. In a production environment, it is a good idea to use secure socket layer (SSL) encryption for this traffic. The most convenient way to do this is to use a Zeus Traffic Manager as a proxy.

You will need to provision a Zeus SmartMachine, create a Zeus SSL virtual server on it, and configure it to handle SSL encryption. The Zeus Traffic Manager provides a wizard to assist you.

- Configure the SSL Virtual Server to decrypt HTTPS traffic to the Cloud Control head node.
- Configure signed or self-signed SSL certificates.
- Create self-signed certificates.

Instructions for this process are located in Chapter 12, “SSL Encryption” of the *Zeus User Manual*. You can find the manual as well as the rest of the Zeus documentation in the `/opt/zeus` directory of your Zeus Smart Machine.





# Compute Nodes

Compute nodes are the physical servers that contain the virtual servers, called SmartMachines. In Cloud Control, compute nodes are called servers and SmartMachines are called zones.

## Accessing the Global Zone

The global zone is the zone that contains all the other zones. You can think of the global zone as the physical machine that contains the virtual machines.

In general, the only reason to access a global zone is to update configurations or to create new SmartMachine templates.

## Common Commands

### Zlogin

The **zlogin** command allows you to log into a zone. This is the equivalent of getting root access to the virtual machine that the zone represents.



**NOTE:** Complete details for zlogin can be found at <http://www.unix.com/man-page/OpenSolaris/1/zlogin/>

### ZFS

The **zfs** command is used to work with zfs data sets.



**NOTE:** Complete details for zfs can be found at <http://www.unix.com/man-page/OpenSolaris/1m/zfs/>

A zfs dataset can be the file system for a zone or a template used to provision a new zone.

```
[root@admin-1 ~]# zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
zones                               3.56G 16.0G   34K    /zones
zones/develop-1.0                   116K 16.0G  1.07G  /zones/develop-1.0
zones/develop-1.0.0                  0    16.0G  1.07G  /zones/develop-1.0.0
zones/ffc2vjaa                       239M 4.77G  1.27G  /zones/ffc2vjaa
zones/ffc2vjab                       239M 4.77G  1.27G  /zones/ffc2vjab
zones/ffc2vjac                       239M 1.77G  1.27G  /zones/ffc2vjac
zones/ffc2vjad                       239M 1.77G  1.27G  /zones/ffc2vjad
zones/ffc2vjae                       239M 1.77G  1.27G  /zones/ffc2vjae
zones/mysql-1.0.2                    745M 16.0G  745M   /zones/mysql-1.0.2
zones/protemplate-2.3.2              1.07G 16.0G  1.07G  /zones/protemplate-
2.3.2
zones/testdevel                      239M 1.77G  1.27G  /zones/testdevel
zones/zxtm-6.0.4                     363M 16.0G  363M   /zones/zxtm-6.0.4
```

One of the command zfs subcommands is snapshot. It allows you to create a snapshot of the current state of a zone:

```
[root@admin-1 ~]# zfs snapshot zones/ffc2vjae@20101006
```

In this case, the command created a snapshot of the zone ffc2vjae. The @ sign indicates that it is a snapshot.

## Zpool

Zpool configures ZFS storage pools. A storage pool is a collection of devices that provide physical storage and data replication for ZFS datasets. All datasets within a storage pool share the same space.



**NOTE:** Complete details for zpool can be found at <http://www.unix.com/man-page/OpenSolaris/1m/zpool/>

You can use the zpool list command to see the status of the pools that your global zone uses.

```
[root@admin-1 ~]# zpool list
NAME      SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
zones    19.9G  3.56G  16.3G   17%  ONLINE  -
```

## ZoneAdm

ZoneAdm is used to administer system zones. A zone is an application container that is maintained by the operating system run-time.



**NOTE:** Complete details for ZoneAdm can be found at <http://www.unix.com/man-page/OpenSolaris/1m/zoneadm/>

```
[root@admin-1 ~]# zoneadm list -v
ID NAME          STATUS      PATH                                BRAND  IP
0  global         running    /                                    native shared
2  ffc2vjaa       running    /zones/ffc2vjaa                    native shared
7  testdevel      running    /zones/testdevel                    native excl
9  ffc2vjac       running    /zones/ffc2vjac                     native shared
11 ffc2vjad        running    /zones/ffc2vjad                     native shared
13 ffc2vjae        running    /zones/ffc2vjae                     native shared
14 ffc2vjab        running    /zones/ffc2vjab                     native shared
```

## Prstat -Z

Prstat iteratively examines all active processes on the system and reports statistics based on the selected output mode and sort order.



**NOTE:** Complete details for prstat can be found at <http://www.unix.com/man-page/OpenSolaris/1m/prstat>



# Zeus Traffic Manager

The Zeus Traffic Manager is a software traffic manager and load balancer that runs as an appliance on a SmartMachine. Typically, Zeus is used to provide the public-facing IP addresses for a pool of application servers.



**NOTE:** The Zeus Traffic Manager requires additional license fees.

## Provisioning a Zeus SmartMachine

To provision a Zeus SmartMachine, choose one of the Zeus templates when creating a new zone. Zeus templates typically begin with zxtm.

Figure 14. Creating a Zeus Zone from the Customer Panel

Figure 15. Creating a Zeus Zone from the New Zone Panel

# Accessing Zeus

To access the Zeus Traffic Manager, go to port 9090 on the Zeus SmartMachine:

<https://<zeus-smart-machine-name>:9090/apps/zxtm/login.cgi>

Log in with the credentials you received when you provisioned the SmartMachine.

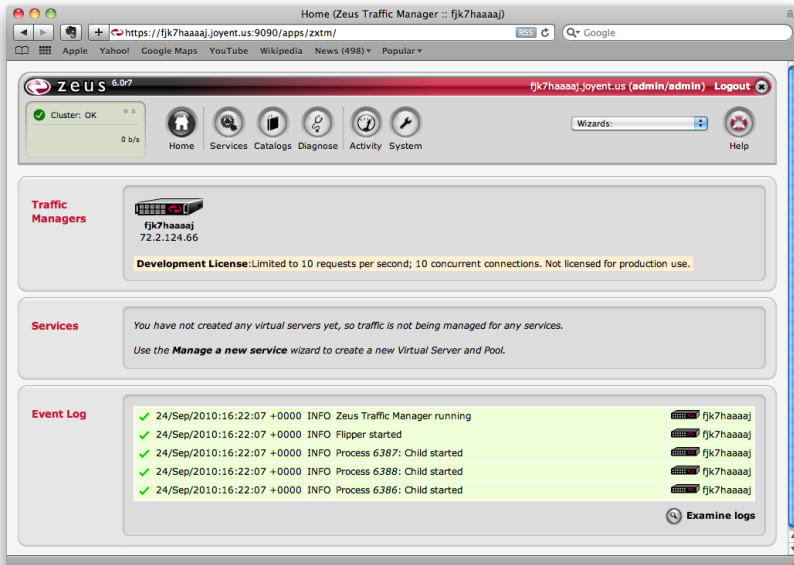


Figure 16. Accessing the Zeus Traffic Manager

## Zeus Traffic Manager Documentation

Documentation for Zeus Traffic Manager is located in `/opt/zeus/*.pdf`.

# Custom Templates

## Working with Custom Templates

You can create a custom template to provision zones. You can use an existing template as the basis for the new template or you can use the supplied template as a base.

The **tpl** tool:

- Creates the new zone.
- Customizes it with a customization script.
- Creates the snapshot that can be used as a template in Cloud Control.

### Creating Custom Templates on the Server

Cloud Control relies on a particular naming convention for zone templates:

`basename-x.y.z`

where `x.y.z` is a version number of the form `major.minor.update`.

The **tpl** tool uses this naming convention to find the existing base template, to create the new template, and to find the directory that holds the customization scripts.

This section assumes that you are creating a new zone template named `develop-1.0.0` and it is based on the template `base-1.3.4`.

To use **tpl** you need:

- An existing zone template (in this case `base-1.3.4`).
- A directory named `develop-1.0` in the directory you are using to run **tpl**. This directory holds the customization script for your zone.
- An executable bash script named *customize* inside the directory `develop-1.0`

Log into the server that you want to use to create your custom template and invoke **tpl** like this:

```
tpl -b base-1.3.4 develop-1.0.0
```

**Tpl** verifies that the required directories and files are present, begins creating the zone, and asks for an IP address and a NIC. The IP should be an address in the server's external subnet, and the NIC should typically be external.



**NOTE:** Use the Subnets page in Cloud Control to determine which subnets your server uses. If your server is actively provisioning zones, reserve the IP address you use in the IP page of Cloud Control.

After cloning the base zone, **tpl** copies a directory named `develop-1.0` to the root of the zone and executes the customize script in it.

The customize script does nothing but set the message of the day. At login, all newly provisioned zones that use this template will display:

```
#!/usr/bin/bash
# Replace the MOTD message and exit
```

```
cat - >/etc/motd <<EOF
Welcome to a brand new zone!
EOF
exit 0
```

In most cases, your customize script will be more complex. Your script runs as the root user of your zone, and the network is functional. You can download software and install it, place configuration files in the proper place, and so on.

When **tpl** finishes, you will have a zone and a snapshot of the zone. In the example above, the zone would be named `zones/develop-1.0.0` and the snapshot would be named [zones/develop-1.0.0@final](#).

## Transferring the Template to Other Servers

The snapshot is used to create a stream file that you can transfer to other servers.

To create the stream file use the `zfs send` command:

```
zfs send zones/develop-1.0.0@final > develop-1.0.0.zfs
gzip develop-1.0.0.zfs
```

To transfer the zipped file to the other server, use the `zfs recv` command to create the template zone there:

```
gzcat develop-1.0.0.zfs.gz | zfs receive zones/ develop-1.0.0
```



## Preparing Cloud Control to Recognize the New Template

Once you have created the zone template on the server, you need to let Cloud Control know that it exists.

- 1 In Cloud Control, navigate to the **ZFS Datasets** page.
- 2 Click **New ZFS Dataset**. In the **Name** field, enter the name of the zone template you created.
- 3 Click **Create**. The new zone appears on the **Listing Zfs Datasets** page in the **Name** column.
- 4 From the side menu, click **Zone Configurations** to access the **Zone Configurations** page, then click **New Zone Configuration**.



**NOTE:** The zone configuration specifies the amount of RAM a zone can use, how much disk space is allocated to it, etc. You can create more than one zone configuration for each template.

- 5 Enter the information (bolded fields are required).

**New zone\_configuration** Zone configurations

|                                   |                                     |
|-----------------------------------|-------------------------------------|
| <b>Name</b>                       | med-develop-1.0.0                   |
| <b>Pretty Name</b>                | development medium                  |
| <b>RAM in MB</b>                  | 512                                 |
| <b>CPU Shares</b>                 | 1                                   |
| CPU Cap                           | <input type="checkbox"/>            |
| <b>Swap in MB</b>                 | 512                                 |
| <b>Max. Lightweight Processes</b> |                                     |
| <b>Disk in GB</b>                 | 2                                   |
| DNS Parent Domain                 | <input type="checkbox"/>            |
| Load Balancing Available?         | <input type="checkbox"/>            |
| Similar Zone Names Per Customer?  | <input checked="" type="checkbox"/> |
| <b>ZFS Dataset</b>                | develop-1.0.0                       |

- 6 Choose the template you just created from the **ZFS Dataset** popup menu and click **Create**.

- 7 Verify that the template appears in the **ZFS Datasets** list.

Zone Configuration was successfully created.

development medium

Zone configurations  Search

Edit

|                                  |  |
|----------------------------------|--|
| Pretty Name                      | development medium                                 |
| RAM in MB                        | 512  |
| CPU Shares                       | 1  |
| CPU Cap                          | <input checked="" type="checkbox"/> server default |
| Swap In MB                       | 512  |
| Lightweight Procs                | 2000   |
| Disk in GB                       | 2  |
| DNS Parent Domain                |  |
| Load Balancing Available?        | false  |
| Similar Zone Names Per Customer? | true   |

ZFS Datasets (1)

| Name          | Action |
|---------------|--------|
| develop-1.0.0 |        |

- 8 From the side menu, click **Server Roles** to access the **Listing Server Roles** page. Click the server role name (in the sample below, “PRO”).

Listing Server Roles

Server roles  Search

+ New Server Role

1 - 1 of 1

| Name | Supported Zone Configurations  | Actions |
|------|--|---------|
| PRO  | small protemplate-2.3.2 (small protemplate-2.3.2), small zxtm-6.0.4 (small zxtm-6.0.4), small mysql-1.0.2 (small mysql-1.0.2), small development (small develop-1.0.0) |         |

1 - 1 of 1

- 9 From the **Zone Configuration** popup menu under **Supported Zone Configurations**, choose the configurations supported on this server.

| Name  | RAM in MB | Load Balancing Available? | Action |
|---|-----------|---------------------------|--------|
| small proteplate-2.3.2 (small proteplate-2.3.2) | 256       | no                        | ⊖      |
| small zxtm-6.0.4 (small zxtm-6.0.4)             | 256       | no                        | ⊖      |
| small mysql-1.0.2 (small mysql-1.0.2)           | 256       | no                        | ⊖      |
| small development (small develop-1.0.0)         | 256       | no                        | ⊖      |

Zone Configuration:

- 10 Click the **Assign** button. You are now ready to provision zones using the template and zone configurations you created.



# Failover Procedure

These steps outline the process for enacting failover to the backup head node.

- 1 Disconnect the current admin node, or shut it down if it is not already shut down:

```
sudo shutdown -h now.
```

- 2 On the console of redundant admin node, run:  
`/opt/dell/bin/restore_admin.sh`.

This runs a restore using the latest backup file from `/data/backups`, and performs an IP failover assuming the IP address of the original admin node.

- 3 If a file name is provided, that file is used as the backup file; otherwise, the most recent one from `/data/backups` is used.
- 4 When finished, the redundant IP becomes unavailable and the original IP of the admin node is assumed by the redundant server. You should experience no difference in operations.



# Customizing Notifications

When Cloud Control creates a new customer zone, it sends the customer a welcome email. The email includes information such as IP addresses and default passwords as well as information on where to get support. You can customize this email for your specific environment.

The welcome email is located on the head node in the directory:

```
/opt/joyent/apps/cloud_control/app/views/notifier in  
the file generic_customer_welcome.html.erb
```

This is an embedded Ruby (ERB) file that you can edit with any text editor. The important thing to remember when editing an ERB file is to avoid changing the text enclosed in Ruby commands such as `<% if @zone.public_ips.size > 0 %>`.

# Backing Up Zone Data

Cloud Control provides a script that runs once an hour to back up important configuration and zone data. You can use these backups to restore a Cloud Control head node that has become corrupted.

## What Gets Backed Up

The script `run_backups` backs up the following files:

- `/data/config/config.json`: The configuration file that Cloud Control uses to discover where the physical servers that it controls are located.
- `/home/jill/.ssh`: The SSH directory that contains the keys the head node uses to authenticate communication with the servers. In the backup, this directory is named `ssh` without the initial dot so that you can see it clearly.
- `pg_data.sql`: A dump of the Cloud Control database, which describes how zones are set up on each server.



**NOTE:** The backup script does not back up the data contained in any zone.

You can find the backups in the head node in the directory `/data/backups`. They are stored as compressed tar archives.

Hourly backups are named `backup_hourly_HH.tar.bz2` where `HH` is the hour that the backup was performed.

Daily backups occur at midnight and are named `backup_daily_WW.tar.bz2` where `WW` is the day of the week that the backup was performed (0 = Sunday).



**NOTE:** The daily backup contains the first backup of day. For example, the backup named `backup_daily_02.tar.bz2` is the same as the backup made at midnight on Tuesday.



Weekly backups occur every Sunday and are named `backup_weekly.tar.bz2`.



**NOTE:** The weekly backup contains the same backup as the first backup made on Sunday. The weekly backups are overwritten every week.



# API Reference

This document describes Joyent's RESTful API for private clouds using Cloud Control. This API is used to connect Cloud Control with third-party applications.

## Private Cloud API

### General Considerations

The Base URL for all requests depends on the application environment. For example,

Production    <https://api.joyent.com/mcp/<version>>

Staging        <https://api.staging.joyent.us/mcp/<version>>

Development: <http://127.0.0.1:4567/>

Where

**<version>** is the release date of the version of the API being called — e.g., 2009-10-15 (ISO-8601 date format including year, month, date).

- All API calls must use HTTP Basic authentication over SSL. All requests have to include a shared username and password. (Add `-u username:password` to all the `curl` request examples).
- **Content-Type** header is not required or supported for this release, but it will be in a future release.
- Additional extension given to the URL will override any **Accept** header provided, and return the requested format. In case of no **Accept** header provided and neither extension supplied to the URL, the default **Content-Type** of the response will be `application/xml`.

- All resources and collections provide an **HTTP Vary** header for GET requests, intended to be used by caching agents, in order to fully determine whether a cache is permitted to use the response to reply to a subsequent request without revalidation.
- GET requests to the API Base URL will retrieve information regarding the service, including available resources, version date, and the estimated expiration date for this API version.

### Service Information XML Sample CURL Request/Response

```
$ curl -i --url http://api.host.tld/mcp/2009-11-30/ -u
username:password
HTTP/1.1 200 OK
Content-Length: 484
Connection: keep-alive
Server: Joyent Web Server 2.0
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<service_info version="2009-11-30">
  <resources>
    <resource uri="http://api.host.tld/mcp/2009-11-30/customers"
name="customers">
      <resource uri="/containers" name="containers"/>
    </resource>
    <resource uri="http://api.host.tld/mcp/2009-11-30/templates"
name="templates"/>
    <resource uri="http://api.host.tld/mcp/2009-11-30/servers"
name="servers"/>
  </resources>
</service_info>
```

## Service Information JSON Sample CURL Request/Response

```
$ curl -i --url http://api.host.tld/mcp/2009-11-30/ -u
username:password -H "Accept: application/json"
HTTP/1.1 200 OK
Content-Length: 375
Connection: keep-alive
Server: Joyent Web Server 2.0
Content-Type: application/json

{
  "version":"2009-11-30",
  "resources":[
    {
      "uri":"http://api.host.tld/mcp/2009-11-30/customers",
      "name":"customers",
      "resource":{"uri":"/containers","name":"containers"}
    },{
      "uri":"http://api.host.tld/mcp/2009-11-30/templates",
      "name":"templates"
    },{
      "uri":"http://api.host.tld/mcp/2009-11-30/servers",
      "name":"servers"
    }
  ]
}
```

## Differences Between Public and Private Cloud APIs

Using the Public Cloud API:

- Customers cannot access other customers' information or Cloud resources.
- Customers can access only those resources assigned to them. For example, a customer will only have access to information for owned containers.
- The Public API requires customer authentication and authorization using **OAuth**, while the Control API uses HTTP Basic Authentication over SSL.

## Resource Models

This API provides access to the following entities:

- Customers: Company or department with associated containers.
- Containers: Solaris Zones.
- Templates: Predefined templates used to create a new container.
- Servers: Physical servers where the containers can be created.

Additionally, the following entities are associated with the ones listed above, especially containers:

- **IPs:** IPv4 addresses associated with a container.
- **Credentials:** Access credentials associated with a container.
- **Hostnames:** Any hostname associated with a container.
- **Transitions:** Detailed information about any of the operations allowed for a container.
- **Errors:** Problems associated with any of the allowed operations.

All of these resource models can have either XML or JSON representation, depending on the Content-type accepted for a given request. Also, all of them may have a different representation (lightweight) when requesting a collection of resources rather than when requesting a single resource (detailed).

## Resource */customers*

### General Methods Description

| HTTP Method | URI            | Response Body                            | Status Codes  |
|-------------|----------------|--|---------------|
| GET         | COLLECTION_URI | Customers Collection Format              | 200, 204, 304 |
| POST        | COLLECTION_URI | Customers Collection Format/Error Format | 201, 409      |
| GET         | RESOURCE_URI   | Customer Format                          | 200, 304, 404 |
| PUT         | RESOURCE_URI   | ————/Error Format                        | 200, 409      |
| DELETE      | RESOURCE_URI   | ————                                     | 405           |

Allowed actions: Index[GET], Show[GET], Update[PUT], Create[POST].

Destroy[DELETE] action is not allowed and will return 405 Status Code.

## Index Action

|                     |  |
|---------------------|--|
| Path                | <code>/customers(. [xml json])?</code> |
| Request Method      | <b>GET</b>                             |
| Parameters          | None                                   |
| Success HTTP Code   | 200 OK                                 |
| Response Body (XML) |  |

```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
  <customer>
    <uri>/customers/Integer</uri>
    <ram_quota_in_megabytes>Integer</ram_quota_in_megabytes>
    <auto_provisionable>Boolean</auto_provisionable>
    <email_address>somebody@example.com</email_address>
    <updated_at>[ISO-8601 Date Time, Complete date plus hours,
minutes and seconds]</updated_at>
  </customer>
  <customer>...</customer>
  ...
</customers>
```

## Sample CURL Requests

```
curl -i --url http(s)://api.base.uri/customers.[xml|json]
curl -i --url http(s)://api.base.uri/customers -H 'Accept:
application/json'
curl -i --url http(s)://api.base.uri/customers -H 'Accept:
application/xml'
```

## Provides If-Modified-Since HTTP Header (Last Modified Customer)

```
curl -i --url http(s)://api.base.uri/customers.[xml|json] -H 'If-Modified-Since: HTTP-Date'  
curl -i --url http(s)://api.base.uri/customers -H 'Accept: application/json' -H 'If-Modified-Since: HTTP-Date'  
curl -i --url http(s)://api.base.uri/customers -H 'Accept: application/xml' -H 'If-Modified-Since: HTTP-Date'
```

## Customers Collection Pagination

It is possible to limit the number of customers retrieved for a GET `COLLECTION_URI` request. By default there is no limit, and all customers will be returned. In order to limit the collection size:

- HTTP request headers **X-Joyent-Collection-Offset** and **X-Joyent-Collection-Limit** can be used with the traditional SQL meaning.
- The same result can be achieved by using `offset` and `limit` query string parameters.
- The preferred way to retrieve a limited list of customers is using HTTP headers. Hence, when both options are given (HTTP headers and query string parameters), the HTTP headers will take precedence.
- When `offset` is provided — either using headers or query string — and `limit` is not, the default limit is 20.
- When `limit` is provided — either using headers or query string — and `offset` is not, the default offset is 0.
- The **customers** collection is always sorted by URI, (from older to newer).
- The total number of customers is always returned as the HTTP header **X-Joyent-Resource-Count**.
- `Last-Modified` is provided in the response, which can be taken advantage of in future requests with `If-Modified-Since`.

```
curl -i --url http(s)://api.base.uri/customers?offset=100&limit=10  
curl -i --url http(s)://api.base.uri/customers -H "X-Joyent-Collection-Offset: 100" -H "X-Joyent-Collection-Limit: 10"
```



## Show Action

Path `/customers/:customer_id(. [xml|json])?`

HTTP Method `GET`

Parameters `None`

Success HTTP Code `200 OK`

Response Body  
(XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<customer>
  <uri>/customers/Integer</uri>
  <email_address>somebody@example.com</email_address>
  <alternate_email_address></alternate_email_address>
  <company_name>String</company_name>
  <street_1>String</street_1>
  <street_2>String</street_2>
  <city>String</city>
  <state></state>
  <postal_code>String</postal_code>
  <country>String</country>
  <phone_number>String</phone_number>
  <updated_at>[ISO-8601 Date Time, Complete date plus hours,
minutes and seconds]</updated_at>
</customer>
```

## Sample CURL Requests

```
curl -i --url
http(s)://api.base.uri/customers/#{customer_id}. [xml|json]
```

## Provides If-Modified-Since HTTP Header

```
curl -i --url
http(s)://api.base.uri/customers/#{customer_id}. [xml|json] \
-H 'If-Modified-Since: #{customer.updated_at} (HTTP-Date)'
```

## Create Action

|                   |  |
|-------------------|--|
| Path              | /customers(. [xml json])?  |
| HTTP Method       | POST   |
| Parameters        | :customer: the same at Show action<br>response body without uri and<br>updated_at. first_name,<br>last_name and email_address are<br>required fields.  |
| Request Body      | Currently not supported a Content-Type of<br><b>application/json</b> or <b>application/xml</b> header,<br>but only application/x-www-form-<br>urlencoded even when the proper <b>Content-</b><br><b>type</b> header should be one of the<br>aforementioned ones. |
| Success HTTP Code | 201 Created  |
| Error HTTP Code   | 409, including validation errors on the response<br>body   |



**NOTE:** Validation errors include only the error messages, not the attribute names.

### Response Body

```
<?xml version="1.0" encoding="UTF-8"?>
<customer>
  <uri>/customers/Integer</uri>
</customer>
```

### Sample CURL Requests

```
curl --url http(s)://api.base.uri/customers.[xml|json] --data
customer="<?xml version="1.0"?>..."
```

## XML

```
curl --url http(s)://api.base.uri/customers.xml --customer= \
"=xml version="1.0" encoding="UTF-8"?"
  <customer>
    <first_name>John</first_name>
    <last_name>Doe</last_name>
    <email_address>john@doe.com</email_address>

  <alternate_email_address>doe@john.com</alternate_email_address>
    <company_name>John Doe Inc.</company_name>
  </customer>"
```

## JSON

```
curl --url http(s)://api.base.uri/customers.json --customer= \
'{"first_name":"John","last_name":"Doe","email_address":"john@doe.
com",\
"alternate_email_address":"doe@john.com","company_name":"John Doe
Inc."}'
```

## Update Action

|                   |  |
|-------------------|--|
| Path              | /customers/:customer_id(. [xml json])?   |
| HTTP Method       | PUT<br>Optionally, allow method override with POST + <code>_method=put</code> param                              |
| Parameters        | The same at Show action response body but <code>uri</code> and <code>updated_at</code>                           |
| Success HTTP Code | 200 OK   |
| Error HTTP Codes  | 404 when trying to update a resource that does not exist<br>409 including validation errors on the response body |



**NOTE:** Validation errors include only the error messages not the attribute names.

Response Body      N/A

## Sample CURL Requests

```
curl -X PUT -i --url
http(s)://api.base.uri/customers/#{:customer_id}.[xml|json] \
--data customer="<?xml version="1.0"?>...."
```

## Delete Action

- No delete action allowed for this gateway version.
- Any DELETE request to `/customers/:customer_id(. [xml|json])` ? should return 405 Method Not Allowed

## Sample CURL Request

```
curl -X DELETE -i --url
http(s)://api.base.uri/customers/#{customer_id}.[xml|json]
```

## Automatic Provision and RAM Quotas

When the value for the customer attribute `auto_provisionable` is set to false, any attempt to create a new container for this customer will not be queued for automatic provision. Instead, the container record will be created, and it will appear in the **to setup** list on Cloud Control. The response status code for this case on `POST /customers/:customer_id/containers` will be 204 No Content, with the appropriate information on response headers. (The default value for this attribute, when not specified, is true.)

The customer attribute `ram_quota_in_megabytes` will not have any effect when set to zero, which is the default value. When it is set to an integer value, any attempt to create a container for the customer will verify that the total RAM used by the customer's containers plus the RAM required by the new container will not be greater than the quota assigned to the customer.

Any attempt to overtake this value will result in a response to the action `POST /customers/:customer_id/containers` with code 403, and response body including the message `RAM Quota Exceeded`.

# Resource */templates*

All the containers are created from a predefined template dataset.

## General Methods Description

| HTTP Method | URI            | Response Body               | Status Codes  |
|-------------|----------------|-----------------------------|---------------|
| GET         | COLLECTION_URI | Templates Collection Format | 200, 204, 304 |
| GET         | RESOURCE_URI   | Templates Format            | 201, 304; 404 |

Allowed actions: Index[GET], Show[GET].

## Index Action

Path `/templates(. [xml|json])?`

HTTP Method `GET`

Parameters `None`

Success HTTP Code `200 OK`

Response Body

```
<?xml version="1.0" encoding="UTF-8"?>
<templates>
  <template>
    <uri>/templates/Integer</uri>
    <pretty_name>String</pretty_name>
    <ram_in_megabytes>Integer</ram_in_megabytes>
    <disk_in_gigabytes>Integer</disk_in_gigabytes>
  </template>
  <template>...</template>
  ...
</templates>
```

## Sample CURL Requests

```
curl -i --url http(s)://api.base.uri/templates.[xml|json]
curl -i --url http(s)://api.base.uri/templates -H 'Accept:
application/json'
curl -i --url http(s)://api.base.uri/templates -H 'Accept:
application/xml'
```

## Templates Collection Pagination

It is possible to limit the number of templates retrieved for a `GET COLLECTION_URI` request. By default there is no limit, and all templates will be returned. In order to limit the collection size:

- HTTP request headers **X-Joyent-Collection-Offset** and **X-Joyent-Collection-Limit** can be used with the traditional SQL meaning.
- The same result can be achieved by using `offset` and `limit` query string parameters.
- The preferred way to retrieve a limited list of templates is using HTTP headers. Hence, when both options are given (HTTP headers and query string parameters), the HTTP headers will take precedence.
- When `offset` is provided — either using headers or query string — and `limit` is not, the default limit is 20.
- When `limit` is provided — either using headers or query string — and `offset` is not, the default offset is 0.
- The templates collection is always sorted by URI (from older to newer).
- The total number of templates is always returned as the HTTP header **X-Joyent-Resource-Count**.
- `Last-Modified` is provided in the response, which can be taken advantage of in future requests with `If-Modified-Since`.

```
curl -i --url http(s)://api.base.uri/templates?offset=100&limit=10
curl -i --url http(s)://api.base.uri/templates -H "X-Joyent-
Collection-Offset: 100" -H "X-Joyent-Collection-Limit: 10"
```

## Show Action

|                   |   |
|-------------------|---|
| Path              | /templates/:template_id(.[xml json])?                       |
| HTTP Method       | GET   |
| Parameters        | None  |
| Success HTTP Code | 200 OK  |
| Error HTTP Code   | 404 when trying to retrieve non-existent zone configuration |
| Response Body     |   |

```
<?xml version="1.0" encoding="UTF-8"?>
<template>
  <uri>/templates/Integer</uri>
  <name>String</name>
  <pretty_name>String</pretty_name>
  <ram_in_megabytes>Integer</ram_in_megabytes>
  <disk_in_gigabytes>Integer</disk_in_gigabytes>
  <cpu_shares>Integer</cpu_shares>
  <swap_in_megabytes>Integer</swap_in_megabytes>
  <lightweight_processes>Integer</lightweight_processes>
</template>
```

## Sample CURL Requests

```
curl -i --url
\http(s)://api.base.uri/templates/#{template_id}.[xml|json]
curl -i --url \
http(s)://api.base.uri/templates/#{template_id} \
-H 'Accept: application/json'
curl -i --url \
http(s)://api.base.uri/templates/#{template_id} \
-H 'Accept: application/xml'
```

Error Response Code: 404 Not Found (when given a non-existent uri)



**NOTE:** Both methods provide **ETag** headers on the response. It is pending to use it in order to handle requests including **If-None-Match** headers.

# Resource `/servers`

This resource represents the servers that can be used from the API.



**NOTE:** Available servers need to be flagged as `api_provisionable` from Cloud Control before they can be used from the API.

## General Methods Description

| HTTP Method | URI                         | Response Body             | Status Codes  |
|-------------|-----------------------------|---------------------------|---------------|
| GET         | <code>COLLECTION_URI</code> | Servers Collection Format | 200, 204, 304 |
| GET         | <code>RESOURCE_URI</code>   | Servers Format            | 201, 304; 404 |

Allowed actions: `Index[GET]`, `Show[GET]`.

## Index Action

Path `/servers(. [xml|json])?`

HTTP Method `GET`

Parameters `None`

Success HTTP Code `200 OK`

Response Body

## Sample CURL Requests

```
curl -i --url http(s)://api.base.uri/servers.[xml|json]
curl -i --url http(s)://api.base.uri/servers -H 'Accept:
application/json'
curl -i --url http(s)://api.base.uri/servers -H 'Accept:
application/xml'
```



## Provides If-Modified-Since HTTP Header (Last Modified Customer)

```
curl -i --url http(s)://api.base.uri/servers.[xml|json] -H 'If-Modified-Since: HTTP-Date'
curl -i --url http(s)://api.base.uri/servers -H 'Accept: application/json' -H 'If-Modified-Since: HTTP-Date'
curl -i --url http(s)://api.base.uri/servers -H 'Accept: application/xml' -H 'If-Modified-Since: HTTP-Date'
```

## Servers Collection Pagination

It is possible to limit the number of servers retrieved for a GET `COLLECTION_URI` request. By default there is no limit, and all templates will be returned. In order to limit the collection size:

- HTTP request headers **X-Joyent-Collection-Offset** and **X-Joyent-Collection-Limit** can be used with the traditional SQL meaning.
- The same result can be achieved by using `offset` and `limit` query string parameters.
- The preferred way to retrieve a limited list of servers is using HTTP headers. Hence, when both options are given (HTTP headers and query string parameters), the HTTP headers will take precedence.
- When `offset` is provided — either using headers or query string — and `limit` is not, the default limit is 20.
- When `limit` is provided — either using headers or query string — and `offset` is not, the default offset is 0.
- The `servers` collection is always sorted by URI, (from older to newer).
- The total number of servers is always returned as the HTTP header **X-Joyent-Resource-Count**.
- `Last-Modified` is provided in the response, which can be taken advantage of in future requests with `If-Modified-Since`.

```
curl -i --url http(s)://api.base.uri/servers?offset=100&limit=10
curl -i --url http(s)://api.base.uri/servers -H "X-Joyent-Collection-Offset: 100" -H "X-Joyent-Collection-Limit: 10"
```

## Show Action

|                   |  |
|-------------------|--|
| Path              | /servers/:server_id(.[xml json])?                |
| HTTP Method       | GET  |
| Parameters        | None   |
| Success HTTP Code | 200 OK   |
| Error HTTP Code   | 404, when trying to retrieve non-existent server |
| Response Body     |  |

## Sample CURL Requests

```
curl -i --url
http(s)://api.base.uri/servers/#{server_id}.[xml|json]
curl -i --url http(s)://api.base.uri/servers/#{server_id} -H
'Accept: application/json'
curl -i --url http(s)://api.base.uri/servers/#{server_id} -H
'Accept: application/xml'
```

Error Response Code: 404 Not Found (when given an non-existent uri)

## Provides If-Modified-Since HTTP Header

```
curl -i --url
http(s)://api.base.uri/servers/#{server_id}.[xml|json] \
-H 'If-Modified-Since: #{server.updated_at} (HTTP-Date)'
```



**NOTE:** Both methods provide **ETag** headers on the response. It's pending to use it in order to handle requests including **If-None-Match** headers.

# Container's State Machine and REST

The REST representation of a container has some complexities due to the dualism introduced by trying to represent a file system object as a traditional HTTP resource. Containers are tracked with records in a database, but they are merely a representation of an underlying complex system, imposing constraints upon the RESTful design.

For example, when a new container is instantiated, the database record is created before the container is created on the file system. The database record exists, but the existence of the object that the database record represents cannot be verified until later. Care must be taken to ensure that the RESTful representation is correct.

Alongside the requirement for dualism in the system is the requirement for asynchronous tasks. In many cases, the time required to perform a task exceeds the time that is generally available to complete an HTTP request. To work around this limitation, the container API uses asynchronous transitions.

Every time a successful request is made to transition the container from one state to another (a task), the HTTP status code is 202 Accepted.

Additionally, a header is included in the response with the name **X-Joyent-Transition-URI** which provides a URI that can be monitored for progress.

This section describes all the possible transitions during the life cycle of a container, including information about any preconditions required by either the REST resource or the underlying file system, and how to retrieve information about the progress of a task being monitored.

These are the tasks we can perform for a given zone/container:

- Create (aka Provision)
- Shutdown
- Destroy
- Reboot
- Startup

## Transition Representation Format

The Transition Representation is either an XML or JSON formatted document containing the progress and completion information for a given transition. When the value of the progress key reaches 100, the transition is complete and the value of success is set to true. Transitions can be monitored by repeatedly requesting a transition URI.

Once the value of progress is 100 and the value of success has been set to true, the response will include the **Location** header with the value of the `RESOURCE_URI` that was used to start the transition.

All requests to the transition URI will include an HTTP header **X-Joyent-Target-URI** pointing to the URI which has triggered the current transition.



NOTE: Only the most recent transaction might be available for a given `RESOURCE_URI`, while all the transitions created from `COLLECTION_URI` will be available.

The format for the response body of these requests is:

## XML

```
<?xml version="1.0" encoding="UTF-8"?>
<transition>
  <progress>[Integer between 0-100]</progress>
  <success>[Boolean true|false]</success>
  <message>[String]</message>
  <name>[String]</name>
</transition>
```

## JSON

```
{
  "success": [Boolean true|false],
  "message": "[String]",
  "progress": [Integer between 0-100],
  "name": "[String]"
}
```

## Containers Representation Format

Containers can be provided to the API consumer at any time by requesting either the `COLLECTION_URI`, an endpoint that returns a collection of resources, or by visiting the `RESOURCE_URI`, which represents a single resource.

The Containers format is fairly simple, consisting of either a JSON or XML formatted document containing sets of pairs. The format selection is made by the API, based on the HTTP **Accept** header being set by the client as `application/json` or `application/xml`. If there is no **Accept** header provided, then the API defaults to the XML representation.

In the case of the `ram` key the relevant unit is megabytes. In each of the following examples there may well be key/value pairs missing, but these examples will serve for the purposes of this document.

## JSON

```
{
  "uri": RESOURCE_URI,
  "name": String,
  "ram": Integer,
  "dataset name": String,
  "updated": String [ISO-8601 Date Time, Complete date plus hours,
minutes and seconds: YYYY-MM-DDThh:mm:ssTZD (eg 1997-07-
16T19:20:30+01:00) ],
  "created": String [ISO-8601 Date Time, Complete date plus hours,
minutes and seconds: YYYY-MM-DDThh:mm:ssTZD (eg 1997-07-
16T19:20:30+01:00) ],
  "running_status": String ["configured", "incomplete",
"installed", "ready", "running", "shutting_down", "down",
"unavailable"],
  "ssh_rsa_fingerprint": String [SSH RSA Key Fingerprint],
  "ssh_dsa_fingerprint": String [SSH RSA Key Fingerprint - Default
key used]
}
```

## XML

```
<?xml version="1.0" encoding="UTF-8"?>
<container>
  <uri>RESOURCE_URI</uri>
  <name>String</name>
  <ram>Integer</ram>
  <dataset_name>String</dataset_name>
  <running_status>String ["configured", "incomplete", "installed",
"ready", "running", "shutting_down", "down",
"unavailable"]</running_status>
  <updated>
    String [ISO-8601 Date Time, Complete date plus hours, minutes
and seconds: YYYY-MM-DDThh:mm:ssTZD (eg 1997-07-16T19:20:30+01:00)
]
  </updated>
  <created>
    String [ISO-8601 Date Time, Complete date plus hours, minutes
and seconds: YYYY-MM-DDThh:mm:ssTZD (eg 1997-07-16T19:20:30+01:00)
]
  </created>
  <ssh_rsa_fingerprint>String [SSH RSA Key
Fingerprint]</ssh_rsa_fingerprint>
  <ssh_dsa_fingerprint>String [SSH RSA Key Fingerprint - Default
key used]</ssh_dsa_fingerprint>
</container>
```

## The Special Property *Running\_Status*

A container's `running_status` property being displayed into resource representation matches the latest known status for that container. In other words, the value of this property is cached.

This API provides a way to update the cached value using `GET RESOURCE_URI/running_status`. This schedules a new check of the container status, and properly updates the container representation when that value is available.



**NOTE:** The intended use of the API methods provided to either retrieve the `running_status` of a container, or reboot it is a hypothetical scenario where the container `SSH` daemon is not responding. Otherwise, the reboot operation could be simply triggered from an `SSH` session.

## Container Tasks

Container tasks manage the transition of a container from one state to another. The tasks the container is capable of carrying out are limited by its current state. Once a task has started, any request to start a different task will yield a `409 Conflict` response. Repeated requests to start a transition that is already running are idempotent.

A key difference between the new container API and the previous incarnation is that the new API does not allow tasks to be queued.

### Task: Create a New Container

Create a container with an `HTTP POST` request to the `COLLECTION_URI`. The `POST` body contains an `x-www-urlencoded` body with a single parameter `template` with a relative URI to a template resource.

Possible errors from the request are `503 Service Unavailable` (when there are no servers capable of holding the zone) or `409 Conflict` (when there is a client error such as an invalid template). In both cases details of the error will be provided in the body of the response.

Successful responses will have a `202 Accepted` status and the database record will be created. The response will include the **X-Joyent-Transition-URI** header.

The complexity of creating a container file system entity may cause failures due to operating system errors. In this case, the container is in a failed state. No tasks can be performed and the transition's success property is false. The message property notes the failure reasons.

Tasks that can be performed on a newly created container are:

- Reboot
- Shutdown

## Implementation

The approach to build `RESOURCE_URI`, `COLLECTION_URI` and `X-Joyent-Transition-URI` uses underlying database records identifiers, so for example:

```
COLLECTION_URI = /containers
RESOURCE_URI = /containers/:id
X-Joyent-Transition-URI = /transitions/RESOURCE_URI (Create
transition)
X-Joyent-Transition-URI = RESOURCE_URI/transition (Any other
transition)
```

So you might try to retrieve the `zone/container` format using the `:id` obtained from the `create transition` URI when this task is either waiting to be executed, during execution, or after a failure. In such cases, any request to `RESOURCE_URI` will return 404 Not Found.

## Task: Reboot a Container

Only **running** containers can be rebooted. In other words, the previous task must be one of the following:

- Create with success result.
- Startup with success result.
- Reboot whatever the result.

Request a container reboot using the HTTP PUT verb on the URI `RESOURCE_URI/running_status`. A successful request will yield the standard 202 Accepted status. It will also include the **X-Joyent-Transition-URI** header.

Once a reboot task has completed, the container enters the same running state that it was in prior to the reboot.

## Task: Shutdown a Container

Only **running** containers can be halted. In other words, the previous task must be one of the following:

- Create with success result.
- Startup with success result.
- Reboot whatever the result.

Request a container shutdown using: `PUT RESOURCE_URI/shutdown`. The expected server response will be `202 Accepted` and will, like all transition requests, also include the header **X-Joyent-Transition-URI**.

One possible reason for a failure during a container shutdown is a *wedged* container. This is a Solaris error that occasionally prevents the container from halting. Experience suggests that subsequent requests to shutdown may succeed, but the only safe way to ensure shutdown is to complete a server reboot.

If a failure occurs during shutdown, the only task that can be attempted is another shutdown. Because the system considers the container to still be active, it disallows any reboot attempts.

Allowed tasks for a successfully halted zone/container are:

- Startup
- Destroy

## Task: Start Up a Container

Only **halted** containers can be started up. In other words, the previous task must to be shutdown with success result.

Request a container startup using: `PUT RESOURCE_URI/startup`.

The shutdown logic is applicable.

Tasks allowed on a successfully running zone/container are the same as those allowed after a successful container creation:

- Reboot
- Shutdown



## Task: Destroy a Container

Only **halted** containers can be destroyed.

Request a zone/container deletion using: `DELETE RESOURCE_URI`. Response code will be `202 Accepted` and it will include the header **X-Joyent-Transition-URI**.

No further tasks can be performed on a successfully destroyed zone/container.

## Zones/Containers Finite State Machine

During any of the following operations, the zone/container is in a locked or in-transition state and will not accept any other operation.

| Task Name | Initial State | Success State | Failure State |
|-----------|---------------|---------------|---------------|
| Create    | —             | Running       | —             |
| Reboot    | Running       | Running       | Running       |
| Shutdown  | Running       | Halted        | Wedged        |
|           | Wedged        | Halted        | Wedged        |
| Startup   | Halted        | Running       | Halted        |
| Destroy   | Halted        | —             | Halted        |

## RESTful Resource "Containers"

### General Methods Description

| HTTP Method | URI                             | Response Body                      | Status Codes   | HTTP Transition Header                                 |
|-------------|---------------------------------|------------------------------------|--|--|
| GET         | COLLECTION_URI                  | Containers<br>Collection<br>Format | 200, 204<br>(Still no<br>container<br>created),<br>304 | —  |
| POST        | COLLECTION_URI                  | ---/Error<br>Format                | 202, 204,<br>503, 403,<br>409                          | X-Joyent-Transition-URI =<br>/transitions/RESOURCE_URI |
| GET         | RESOURCE_URI                    | Container<br>Format                | 200, 304,<br>404, 410                                  | —  |
| PUT         | RESOURCE_URI<br>/running_status | ---/Error<br>Format                | 202, 404,<br>409, 410                                  | X-Joyent-Transition-URI =<br>RESOURCE_URI/transition   |
| GET         | RESOURCE_URI<br>/running_status | ---/Error<br>Format                | 202, 404,<br>409, 410                                  | —  |
| PUT         | RESOURCE_URI<br>/shutdown       | ---/Error<br>Format                | 202, 404,<br>409, 410                                  | X-Joyent-Transition-URI =<br>RESOURCE_URI/transition   |
| PUT         | RESOURCE_URI<br>/startup        | ---/Error<br>Format                | 202, 404,<br>409, 410                                  | X-Joyent-Transition-URI =<br>RESOURCE_URI/transition   |
| DELETE      | RESOURCE_URI                    | ---/Error<br>Format                | 202, 404,<br>409, 410                                  | X-Joyent-Transition-URI =<br>RESOURCE_URI/transition   |



**NOTE:** Shutdown and startup actions use the same `PUT RESOURCE_URI` because the tasks are mutually exclusive for a given zone/container. Any `PUT` request to `RESOURCE_URI` will attempt a shutdown task if the zone/container is running; otherwise, it will attempt a startup task.

## Transitions

| HTTP Method | URI                           | Response Body        | Status Codes                  | HTTP Target Header               | HTTP Location Header         |
|-------------|-------------------------------|----------------------|-------------------------------|----------------------------------|------------------------------|
| GET         | /transitions<br>/RESOURCE_URI | Transition<br>Format | 200,<br>404                   | COLLECTION_URI                   | RESOURCE_URI<br>(On success) |
| GET         | RESOURCE_URI/<br>transition   | Transition<br>Format | 200,<br>(304),<br>404,<br>410 | RESOURCE_URI<br>[/last_task_uri] |                              |

## General Response Consideration

If the URI for the resources is modified, any response will return:

- Code: 301 Moved Permanently
- Response Header: **Location** new location for the resource
- Response Body: N/A.

Any method trying to access a resource may return:

- Code: 410 Gone, when trying to access a destroyed resource. Additionally, the response headers will include **Last-Modified** with the exact time the resource was destroyed.
- Response Body: N/A.
- Code: 404 Not Found, when trying to access a resource which never existed, or when trying to access a new resource being created, or a resource whose creation failed.
- Response Body: N/A.
- Code: 304 Not Modified, when **If-Modified-Since** HTTP Request header is older than or equal to the last time the resource or collection representation was updated.
- Response Body: N/A.
- Code: 409 Conflict, when trying to perform any task over a zone/container with a different task transition being executed.

## Response Body

```
<errors>
  <error>
    A different transition for this container has been requested.
    Please, wait until that transition is finished before to request a
    new task.
  </error>
</errors>
{
  "errors": "A different transition for this container has been
  requested. Please, wait until that transition is finished before
  to request a new task."
}
```

All responses will include the appropriate **Content-Type** header, either `application/xml` or `application/json`. (Notable exceptions are when the response code is either 204 or 304. The HTTP specification explicitly disallows an entity body and a **Content-Type** header).

Additionally, the response to the request `GET RESOURCE_URI` will include the **Content-MD5** HTTP header with the value of the MD5 hash for the current response body when the response status code is 200 OK.

Finally, any POST request to `RESOURCE_URI` will return 405 Method Not Allowed.

## Determining the Current State for a Zone/Container

Every container has a unique `RESOURCE_URI` which matches a relative URI like `/customers/:customer_identifier/containers/:container_identifier`. Use this `RESOURCE_URI` to determine the current state for a container as follows:

- 1 Perform an HTTP GET request to `/transitions/RESOURCE_URI`.

To determine if the container was a failure or is still being created, analyze the request's Response Body and Headers to review the container state:

- **Success property has a value of false and progress property has a value of 100.** The container creation is a failure. Any request to `RESOURCE_URI` will return 404 Not Found, and the container will not be retrieved into the list of containers for a given customer.
- **Success property has a value of false and progress property has a value other than 100.** The container creation is in progress.

- **Success property has a value of true and progress property has a value of 100.** The container was created. The time the container was created will match the value of the **Last-Modified** header. Additionally, the response will include a **Location** header with the value of `RESOURCE_URI`.

Only if a container was created is there reason to continue trying to determine its current state. If the container was not created, requests will return `404 Not Found`.

2 Perform an HTTP GET Request to `RESOURCE_URI`.

There are 3 possibilities depending on the HTTP Response status codes:

- `404` -- The container is still being created, or the creation was a failure, (i.e., the container does not exist).
- `410` -- This container was destroyed exactly at the time given by the HTTP **Last-Modified** header.
- `200` -- The container exists and hasn't been destroyed.

Only if the response status for this request is `200 OK` is there reason to continue trying to determine the current state.

3 Perform an HTTP GET Request to `RESOURCE_URI/transition` in order to retrieve latest container transition. Depending on the response status code and the values for the transition properties are:

- Response status code is `404` -- Once the container was created, no other transition has been requested. The container should be running. (Review the value of `running_status` to confirm).
- Latest transition progress is not `100` -- The container is in the middle of the transition with the given name.
- Latest transition progress is `100` -- Depending on the name of the transition and the value of `success` property:

| Name     | Success | State   |
|----------|---------|---------|
| Shutdown | False   | Wedged  |
| Shutdown | True    | Halted  |
| Startup  | False   | Halted  |
| Startup  | True    | Running |
| Reboot   | False   | Running |
| Reboot   | True    | Running |

## Containers Collection Pagination

By default, there is no limit to the number of containers retrieved for a GET `COLLECTION_URI` request; all non-destroyed containers for a given customer are returned. The following methods can be used to limit the number of containers retrieved:

- HTTP Request Headers **X-Joyent-Collection-Offset** and **X-Joyent-Collection-Limit** can be used with the traditional SQL meaning.
- The same result can be achieved by using `offset` and `limit` Query String parameters.
- The preferred way to retrieve a limited list of containers is using HTTP Headers. When both options (HTTP Headers and Query String parameters) are given, HTTP Headers will take precedence.
- When `offset` is provided — either using headers or query string — and `limit` is not, the default limit is 20.
- When `limit` is provided — either using headers or query string — and `offset` is not, the default offset is 0.
- Containers collection is always sorted by URI (from older to newer).
- The total number of a customer's non-destroyed containers is always retrieved by the HTTP **X-Joyent-Resource-Count** header.
- **Last-Modified** is provided in the response, which can be taken advantage of in future requests with `If-Modified-Since`.

```
curl -i --url
http(s)://api.base.uri/customers/1234/containers?offset=100&limit=
10
curl -i --url http(s)://api.base.uri/customers/1234/containers -H
"X-Joyent-Collection-Offset: 100" -H "X-Joyent-Collection-Limit:
10"
```

## API XML Schemas

### Customers Collection

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customers" type="customersType" />
  <xsd:complexType name="customersType">
    <xsd:sequence>
```

```

    <xsd:element maxOccurs="unbounded" name="customer"
type="customerType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="customerType">
  <xsd:sequence>
    <xsd:element name="uri" type="xsd:string" />
    <xsd:element name="first_name" type="xsd:string" />
    <xsd:element name="last_name" type="xsd:string" />
    <xsd:element name="email_address" type="xsd:string" />
    <xsd:element name="auto_provisionable" type="xsd:boolean" />
    <xsd:element name="ram_quota_in_megabytes" type="xsd:int" />
    <xsd:element name="updated_at" type="xsd:dateTime" />
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
Customer
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customerType" />
  <xsd:complexType name="customerType">
    <xsd:sequence>
      <xsd:element name="uri" type="xsd:string" />
      <xsd:element name="first_name" type="xsd:string" />
      <xsd:element name="last_name" type="xsd:string" />
      <xsd:element name="email_address" type="xsd:string" />
      <xsd:element name="auto_provisionable" type="xsd:boolean" />
      <xsd:element name="ram_quota_in_megabytes" type="xsd:int" />
      <xsd:element name="alternate_email_address"
type="xsd:string" />
      <xsd:element name="company_name" type="xsd:string" />
      <xsd:element name="street_1" type="xsd:string" />
      <xsd:element name="street_2" type="xsd:string" />
      <xsd:element name="city" type="xsd:string" />
      <xsd:element name="state" type="xsd:string" />
      <xsd:element name="postal_code" type="xsd:string" />
      <xsd:element name="country" type="xsd:string" />
      <xsd:element name="phone_number" type="xsd:string" />
      <xsd:element name="updated_at" type="xsd:dateTime" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
Servers Collection
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="servers" type="serversType" />
  <xsd:complexType name="serversType">
    <xsd:sequence>

```

```

    <xsd:element maxOccurs="unbounded" name="server"
type="serverType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="serverType">
  <xsd:sequence>
    <xsd:element name="uri" type="xsd:string" />
    <xsd:element name="hostname" type="xsd:string" />
    <xsd:element name="api_provisionable" type="xsd:boolean" />
    <xsd:element name="updated_at" type="xsd:dateTime" />
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
Server
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="server" type="serverType" />
  <xsd:complexType name="serverType">
    <xsd:sequence>
      <xsd:element name="uri" type="xsd:string" />
      <xsd:element name="hostname" type="xsd:string" />
      <xsd:element name="api_provisionable" type="xsd:boolean" />
      <xsd:element name="updated_at" type="xsd:dateTime" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
Templates Collection
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="templates" type="templatesType" />
  <xsd:complexType name="templatesType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="template"
type="templateType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="templateType">
    <xsd:sequence>
      <xsd:element name="uri" type="xsd:string" />
      <xsd:element name="pretty_name" type="xsd:string" />
      <xsd:element name="ram_in_megabytes" type="xsd:int" />
      <xsd:element name="disk_in_gigabytes" type="xsd:int" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
Template
<?xml version="1.0" encoding="utf-8"?>

```



```

<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="template" type="templateType" />
  <xsd:complexType name="templateType">
    <xsd:sequence>
      <xsd:element name="uri" type="xsd:string" />
      <xsd:element name="pretty_name" type="xsd:string" />
      <xsd:element name="ram_in_megabytes" type="xsd:int" />
      <xsd:element name="disk_in_gigabytes" type="xsd:int" />
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="load_balancing_available"
type="xsd:boolean" />
      <xsd:element name="cpu_shares" type="xsd:int" />
      <xsd:element name="swap_in_megabytes" type="xsd:int" />
      <xsd:element name="lightweight_processes" type="xsd:int" />
      <xsd:element name="dns_parent_domain" type="xsd:string" />
      <xsd:element name="target_zone_count" type="xsd:int" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
Containers Collection
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="containers" type="containersType" />
  <xsd:complexType name="containersType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="container"
type="containerType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="containerType">
    <xsd:sequence>
      <xsd:element name="uri" type="xsd:string" />
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="dataset_name" type="xsd:string" />
      <xsd:element name="ram" type="xsd:int" />
      <xsd:element name="updated" type="xsd:dateTime" />
      <xsd:element name="created" type="xsd:dateTime" />
      <xsd:element name="ssh_rsa_fingerprint" type="xsd:string" />
      <xsd:element name="ssh_dsa_fingerprint" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
Container
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="container" type="containerType" />

```

```

<xsd:complexType name="containerType">
  <xsd:sequence>
    <xsd:element name="uri" type="xsd:string" />
    <xsd:element name="name" type="xsd:string" />
    <xsd:element name="dataset_name" type="xsd:string" />
    <xsd:element name="ram" type="xsd:int" />
    <xsd:element name="updated" type="xsd:dateTime" />
    <xsd:element name="created" type="xsd:dateTime" />
    <xsd:element name="running_status" type="xsd:string" />
    <xsd:element name="ssh_rsa_fingerprint" type="xsd:string" />
    <xsd:element name="ssh_dsa_fingerprint" type="xsd:string" />
    <xsd:element name="ips" type="ipsType" />
    <xsd:element name="server" type="serverType" />
    <xsd:element name="credentials" type="credentialsType" />
    <xsd:element name="hostnames" type="hostnamesType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="hostnamesType">
  <xsd:sequence>
    <xsd:element name="hostname" type="hostnameType" />
  </xsd:sequence>
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="hostnameType">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="credentialsType">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="credential"
type="credentialType" />
  </xsd:sequence>
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="credentialType">
  <xsd:sequence>
    <xsd:element name="system" type="xsd:string" />
    <xsd:element name="username" type="xsd:string" />
    <xsd:element name="password" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="serverType">
  <xsd:sequence>
    <xsd:element name="uri" type="xsd:string" />
    <xsd:element name="hostname" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ipsType">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="ip" type="ipType"
/>
  </xsd:sequence>
</xsd:complexType>

```

```

</xsd:sequence>
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
<xsd:complexType name="ipType">
  <xsd:sequence>
    <xsd:element name="number" type="xsd:decimal" />
    <xsd:element name="address" type="xsd:string" />
    <xsd:element name="updated_at" type="xsd:dateTime" />
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
Transition
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="transition" type="transitionType" />
  <xsd:complexType name="transitionType">
    <xsd:sequence>
      <xsd:element name="progress" />
      <xsd:simpleType>
        <xsd:restriction base="xsd:int">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="100"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="success" type="xsd:boolean" />
    <xsd:element name="message" type="xsd:string" />
    <xsd:element name="name">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="create" />
          <xsd:enumeration value="destroy" />
          <xsd:enumeration value="startup" />
          <xsd:enumeration value="shutdown" />
          <xsd:enumeration value="reboot" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
Errors
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="errors" type="errorsType" />
  <xsd:complexType name="errorsType">
    <xsd:sequence>

```

```
<xsd:element maxOccurs="unbounded" name="error"  
type="xsd:string" />  
</xsd:sequence>  
</xsd:complexType>  
</xsd:schema>
```

# Resource */search*

## General Methods Description

| HTTP Method | URI                               | Response Body                   | Status Codes |
|-------------|-----------------------------------|---------------------------------|--------------|
| GET         | <i>/search</i>                    | Short description of search API | 200          |
| GET         | <i>/search/containers?...=...</i> | Results of containers query     | 200, 400     |
| GET         | <i>/search/customers?...=...</i>  | Results of customers query      | 200, 400     |
| GET         | <i>/search/ips?...=...</i>        | Results of ips query            | 200, 400     |
| GET         | <i>/search/servers?...=...</i>    | Results of servers query        | 200, 400     |
| GET         | <i>/search/templates?...=...</i>  | Results of templates query      | 200, 400     |

HTTP code 400 occurs unless only one constraint is provided in the request. For example, */search/customers?first\_name=John&last\_name=Doe* will return a 400 because it has two constraints: *first\_name=John* and *last\_name=Doe*.

Just */search/customers* will also return 400 because it has no constraint at all.

Plain */search* is an exception to this rule on number of queries since it doesn't perform a search.



**NOTE:** You can only use one constraint with the */search* capability.

## Search Containers (Zones)

Path /search/containers?name=.+

Request Method GET

Success HTTP Code 200 OK

### Response Body (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<zones>
  <zone>
    <uri>/customers/598/containers/127</uri>
    <name>db6640c2</name>
  </zone>
</zones>
```

### Sample CURL Requests

```
curl -Gu user:password --url
http(s)://api.base.uri/search/containers?name=db6640c2
curl -Gu user:password --url
http(s)://api.base.uri/search/containers?name=db6640c2 -H 'Accept:
application/json'
curl -Gu user:password --url
http(s)://api.base.uri/search/containers?name=db6640c2 -H 'Accept:
application/xml'
```



**NOTE:** If you request partial names, all similar container names will be returned.

## Search Customers

### Path

/search/customers?[email\_address|first\_name|last\_name]=.+

### HTTP Method

GET

### Success HTTP Code

200 OK

### Response Body

```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
  <customer>
    <uri>/customers/4907</uri>
    <first_name>John</first_name>
    <last_name>Doe</last_name>
    <email_address>john@example.com</email_address>
    <updated_at>2010-01-14T20:29:29-07:00</updated_at>
  </customer>
</customers>
```

## Sample CURL Requests

```
curl -Gu user:password --url
http(s)://api.base.uri/search/customers?first_name=John
curl -Gu user:password --url
http(s)://api.base.uri/search/customers?last_name=Doe
curl -Gu user:password --url
http(s)://api.base.uri/search/customers?email_address=john@example
.com
curl -Gu user:password --url
http(s)://api.base.uri/search/customers?first_name=John -H
'Accept: application/json'
curl -Gu user:password --url
http(s)://api.base.uri/search/customers?first_name=John -H
'Accept: application/xml'
...
```



**NOTE:** If you request partial first names, last names or email addresses, all similar customers or email addresses will be returned. It's currently not possible to provide multiple constraints in a single request, e.g. /customers?first\_name=John&last\_name=Doe.

## Search IP Addresses

Path /search/ips?address=.+

HTTP Method GET

Success HTTP Code 200 OK

### Response Body

```
<?xml version="1.0" encoding="UTF-8"?>
<ips>
  <ip>
    <number>3232251904</number>
    <address>192.168.64.0</address>
    <updated_at>2010-03-07T00:00:09-07:00</updated_at>
  </ip>
</ips>
```

### Sample CURL Requests

```
curl -Gu user:password --url
http(s)://api.base.uri/search/ips?address=192.168.64.0
curl -Gu user:password --url
http(s)://api.base.uri/search/ips?address=192.168.64.0 -H 'Accept:
application/json'
curl -Gu user:password --url
http(s)://api.base.uri/search/ips?address=192.168.64.0 -H 'Accept:
application/xml'
```



**NOTE:** Unlike the other queries in the search API, this one is currently precise. The full IP address must be provided; an entry like `/search/ips?address=192.168.64` (note the missing final octet) will not work. Sometimes a result returned by `ips?address=...` will not have a URI. This occurs when an IP exists but is associated with something other than a server or zone, e.g., a router.



## Search Servers

Path `/search/servers?hostname=.`

HTTP Method `GET`

Success HTTP Code `200 OK`

### Response Body

```
<?xml version="1.0" encoding="UTF-8"?>
<servers>
  <server>
    <uri>/servers/3</uri>
    <hostname>foo.example.com</hostname>
    <api_provisionable/>
    <updated_at>2009-03-19T23:22:57-07:00</updated_at>
  </server>
</servers>
```

## Sample CURL Requests

```
curl -Gu user:password --url
http(s)://api.base.uri/search/servers?hostname=foo.example.com
curl -Gu user:password --url
http(s)://api.base.uri/search/servers?hostname=foo.example.com -H
'Accept: application/json'
curl -Gu user:password --url
http(s)://api.base.uri/search/servers?hostname=foo.example.com -H
'Accept: application/xml'
```



**NOTE:** If you request partial server names, all servers with a similar hostname will be returned.

## Search Templates (Zone Configurations)

Path /search/templates?[name|pretty\_name]=.+

HTTP Method GET

Success HTTP Code 200 OK

### Response Body

```
<?xml version="1.0" encoding="UTF-8"?>
<zone_configurations>
  <zone_configuration>
    <uri>/templates/1</uri>
    <name>example</name>
    <pretty_name>Example Template</pretty_name>
    <ram_in_megabytes>512</ram_in_megabytes>
    <disk_in_gigabytes>10</disk_in_gigabytes>
  </zone_configuration>
</zone_configurations>
```

### Sample CURL Requests

```
curl -Gu user:password --url
http(s)://api.base.uri/search/templates?name=example
curl -Gu user:password --url
http(s)://api.base.uri/search/templates?pretty_name=Example%20Temp
late
curl -Gu user:password --url
http(s)://api.base.uri/search/templates?name=example -H 'Accept:
application/json'
curl -Gu user:password --url
http(s)://api.base.uri/search/templates?name=example -H 'Accept:
application/xml'
```



**NOTE:** If you request partial names or pretty names, all similar templates will be returned. It's currently not possible to provide multiple constraints in a single request, e.g. /templates?name=foo&pretty\_name=bar

## Collector Agent

The Collector Agent is a node daemon run on global zones. It watches for changes in values for swap, cpu, zfs, and memory usage on zones and records. Values are stored in an SQLite3 database along with an ISO 8601 timestamp. It also responds to simple queries to retrieve that data.

If you have an interest in integrating the solution with a billing application, use this API.

## Issuing Collector Agent Commands

To issue a command to a certain host, publish a message to the `amq.topic` exchange of type `topic`. The routing key will take the following form:

```
collector.<command>.<hostname>
```

To issue a command, use the `hostname` (not FQDN) of the global zone of the machine to be administered. For example, to send a `query` command to `ev1-dev-02.joyent.us`, run:

```
collector.query.ev1-dev-02
```

Commands are issued to the agent via AMQP as string messages with a JSON-encoded object. In addition to command-specific JSON members, all commands are required to include the following fields:

| Field                  | Description   |
|------------------------|---|
| <code>timestamp</code> | Time the request was sent, in ISO 8601 format.  |
| <code>id</code>        | A 4-byte number in hex-string form (e.g., <code>1a2b3c4d</code> ).  |
| <code>client id</code> | A 4-byte number in hex-string form. The client ID is used by the client for this session of commands and is used by the collector agent to reply to the appropriate client. |

### ACK Response

On error or successful execution of a command, the agent publishes an ACK message to the `amq.topic` exchange, routing key — `collector.ack<client_id>.<hostname>` where `<client_id>` is the `client_id` specified in the original message and `<hostname>` is the hostname of the machine the agent is running on.

This is done so that a client only binds to one queue per host, regardless of the number of messages it transmits.

If an error occurs over the course of executing a command, the ACK response will contain an error field with a message indicating what went wrong.

## Collector Agent Command

---

|                   |                      |   |
|-------------------|----------------------|---|
| <b>Command</b>    |                      | Query   |
| <b>Example</b>    |                      | <code>collector.query.&lt;hostname&gt;</code>                                   |
| <b>Use</b>        |                      | Query the machine   |
| <b>Parameters</b> | <code>name</code>    | The statistic to be queried.  |
|                   | <code>zone_id</code> | The numeric <code>zone_id</code> to be queried.                                 |
|                   | <code>start</code>   | ISO8601 date string. Only show events occurring later than this date. Optional. |
|                   | <code>end</code>     | ISO8601 date string. Only show events occurring before this date. Optional.     |

---

## Supported Statistics and Values

| <b>swapresv</b> | <b>cpucaps</b> | <b>rcap</b> |       | <b>zfs_usage</b> |
|-----------------|----------------|-------------|-------|------------------|
| timezone        | timezone       | timezone    | cap   | timezone         |
| zone_id         | zone_id        | zone_id     | at    | zone_id          |
| value           | usage          | zone        | avgat | zone             |
| usage           | value          | nproc       | pg    | path             |
|                 | maxusage       | vm          | avgpg | referenced       |
|                 | nwait          | rss         |       | available        |
|                 |                |             |       | used             |

---

## Command Example

The ACK response for queries contains a **data** field with a time-ordered array of events for that particular statistic.

### Query

```
{ name: 'swapresv'  
, start: '2010-07-30T20:25:54.774Z'  
, zone_id: '1556'  
, timestamp: '2010-07-30T21:59:13.519Z'  
, client_id: 'ad03717'  
, id: '787d1605'  
}
```

### Response

```
{ req_id: '787d1605'  
, timestamp: '2010-07-30T21:59:14.402Z'  
, _routingKey: 'collector.ackad03717.angel'  
, _deliveryTag: 1  
, data:  
  [ { timestamp: '2010-07-30T21:44:55.401Z'  
    , zone_id: '1556'  
    , usage: '834494464'  
    , value: '2147483648'  
    }  
    , { timestamp: '2010-07-30T21:45:06.532Z'  
    , zone_id: '1556'  
    , usage: '834502656'  
    , value: '2147483648'  
    }  
    , { timestamp: '2010-07-30T21:45:07.546Z'  
    , zone_id: '1556'  
    , usage: '834519040'  
    , value: '2147483648'  
    }  
    , { timestamp: '2010-07-30T21:45:36.921Z'  
    , zone_id: '1556'  
    , usage: '834535424'  
    , value: '2147483648'  
    }  
    , { timestamp: '2010-07-30T21:46:07.310Z'  
    , zone_id: '1556'  
    , usage: '834543616'  
    , value: '2147483648'  
    }  
  ]  
}
```

## Using a Client With the Collector Agent

You can find an example collector agent client on your head node in `/opt/joyent/apps/cloud_control/script/collector_agent_client.rb`.

# Glossary

## **BMC**

Baseboard Management Controller. Applies to PowerEdge R series servers only; not present in C series servers.

## **CC / Cloud Control**

The web administration portal for managing the DCSWA operations. This provides the core management functionality of users, SmartMachines and the system in general.

## **Cloud Control API**

Provides a REST programmatic interface into a subset of the features provided by Cloud Control. The API runs parallel with Cloud Control and inherits all of the redundancy features of Cloud. This is a web-service requiring XML/JSON style communications.

## **Cloud Software**

The suite of Cloud management software that includes Cloud Control, Telemeter, and user portal.

## **CN / Node**

Compute Node. A server in the pod that runs SmartMachines

## **Core Router**

Enables communication between the racks of a pod and the external internet to a pod.

## **Customer API**

Provides a REST programmatic interface to manage customers as seen from the user portal. This is a thin layer that allows for multi-tenancy enforcement within the user portal. This is a web-service requiring XML/JSON style communications.

## **DCSWA**

Dell Cloud Solution for Web Applications

## **DNS Server**

Required by the cloud to operate properly. This can be provided by the customer or as part of the cloud and tied into the customer DNS system. In either case, the customer's DNS system must be modified to point to the cloud DNS server or host the cloud's address spaces and name spaces. The suggested implementation is for the cloud to host the DNS server with a cloud domain under the customer's domain. The customer's domain then points to the cloud-based DNS server running on the IS.

**DOP / Department Ordering Portal**

See **User Portal**

**Head Node**

See **Infrastructure Server**

**IaaS**

Infrastructure as a Service

**Instance**

See **SmartMachine**

**IPMI**

Intelligent Platform Management Interface. A common interface to monitor server temperature, voltage, power supplies, and chassis intrusion.

**IS / Infrastructure Server**

Operates the provision and management capabilities of the cloud, including CC and DOP.

**Jumpstart Server**

The core provisioning component of the IS. It provides the base installation image and configuration tools needed to install and update the compute nodes.

**KVM**

Keyboard Video Mouse Switch

**Load Balancer**

An application that directs network traffic to other applications that handle the actual processing of the request.

**MCP**

Master Control Portal: The legacy acronym for Cloud Control.

**MySQL**

A common database used in web applications.

**NSF Server**

An optional customer provided hardware component used to provide back-up and disaster recovery support. It can also be a way for the compute nodes to access a shared storage area. It is not intended for cloud user storage. Dell Services may choose to sell a component per Pod or Cloud depending on size and scale.

**NTP Server**

Runs on the IS and the compute nodes synchronized to that system. The IS can be configured to synchronize to an external time source.

**PaaS**

Platform as a Service

**PDU**

Power Distribution Unit

**POC**

Proof of Concept



**Pod**

A collection of up to 12 racks managed by a provisioning server (PS).

**PowerEdge Rack Enclosure**

Dell Cloud Solution for Web Applications is housed in PowerEdge server rack enclosures.

**PowerEdge Server**

Dell PowerEdge servers are used to host Cloud software and SmartMachines.

**Provisioning Tools**

These tools reside on the compute nodes and provide the services that Cloud Control uses to manage SmartMachines. The tools are deployed to the compute nodes by Cloud Control as part of installation.

**PS / Provisioning Server**

Server that performs the installation of compute nodes. For DCSWA, combined with the CC head node.

**Rack**

Physical enclosure where hardware is placed.

**Repository Server**

Serves Solaris packages to the SmartMachines for update and maintenance. The default SmartMachines provide a set of tools, but they might not be current or contain everything a developer needs. The repository provides tools and updates that can be applied to the SmartMachines after initial installation. The repository server resides on the Infrastructure server.

**RU**

Rack unit equals 1¾ inch

**SmartMachine**

The product name for the DCSWA virtual machine or compute instance. SmartMachines are self-contained virtual operating system instances with supporting libraries.

**SmartOS**

The general purpose UNIX-like operating environment. SmartOS is optimized to provide SmartMachines with minimum guaranteed access to compute resources with automatic bursting as needed.

**Telemeter**

The reporting and monitoring system for Cloud Control. This service runs on the compute nodes and provides Cloud Control with monitoring data/graphs accessible though an external web page. It is used for diagnostics and loading information, and can also provide billing data.

**ToR Switch**

Top of Rack switch. Provides networking for servers in the rack.

**User Portal**

Provides the cloud user a web interface to provision, manage, and decommission SmartMachines.

**Zeus Load Balancer**

Controls application traffic. inspects, transforms, and routes requests across the application infrastructure. It can be provisioned by end users from the user portal. Zeus Load Balancer is provided by the Zeus corporation and runs as a SmartMachine.

**Zone**

A virtual machine or compute instance running on a node within the cloud. Zones are referenced by the Cloud Control Admin.

# Getting Help

## Contacting Dell

Customers in the United States can call 800-WWW-DELL (800-999-3355).



**NOTE:** If you do not have an active Internet connection, you can find contact information on your purchase invoice, packing slip, bill, or Dell product catalog.

Dell provides several online and telephone-based support and service options. Availability varies by country and product, and some services may not be available in your area. To contact Dell for sales, technical support, or customer service issues:

- 1 Visit [support.dell.com](http://support.dell.com).
- 2 Click your country/region at the bottom of the page. For a complete country/region listing, click **All**.
- 3 Click **All Support Options** from the **Support** list at the bottom of the page.
- 4 Select the appropriate service or support link based on your need.
- 5 Choose the method of contacting Dell that is convenient for you.



# Index

## A

Accessing Cloud Control, 31  
API Reference, 51

## B

Backing Up Zone Data, 48  
BIOS Settings, 25

## C

Cloud Provisioning, 14  
Cloud Solution Components  
    Hardware Components, 9  
Cloud Solutions Components  
    Software Components, 10  
Collector Agent, 90  
Contacting Dell. *See* Getting Help  
Container Tasks, 70  
    Create a Container, 70  
    Destroy a Container, 73  
    Reboot a Container, 71  
    Shutdown a Container, 72  
    Start Up a Container, 72  
Custom Templates  
    Working with, 39  
Customizing Notifications, 47

## D

DCSWA, 7  
Dell Cloud Solution for Web  
    Applications. *See* DCSWA  
Determining the Current State for a  
    Zone/Container, 76

## G

Getting Help, 99  
Glossary, 95

## H

Hybrid Looped Configuration, 19

## I

Installing Standard Templates, 29  
Installing the Software, 27

## J

Joyent SmartMachines, 7  
Joyent Telemeter, 13

## M

Measuring Consumption, 14

## **P**

Pods, 7

## **R**

racks, 7

RAID Settings, 25

REST, 66

## **S**

Software Installation. *See Installing the Software*

Switch Port Assignments, 23

## **T**

Templates. *See Installing Standard Templates*

Triangle Looped Configuration, 21

## **U**

User Portal, 12

## **V**

vLANs, 15

## **Z**

Zeus Load Balancer, 7

Zeus SmartMachine  
Provisioning, 37

