ECS-CommunityEdition Documentation

Release 3.2.2.0

Dell-EMC

Sep 18, 2018

Contents

1 Building `ecs-install` Image From Sources 1

2 ECS Community Edition Installation Guides 7

3 Standard Installation 9

4 OVA Installation 15

5 Island Installation 19

6 ECS Administrative Web UI 27

7 Migration 33

8 deploy.yml 35

9 ECS Community Edition Utilities 41

10 ECS Software 3.x - Troubleshooting Tips 47

11 Frequently Asked Questions 53

12 Description 55

13 Quick Start Guide 57

14 Hardware Requirements 59

15 Deployment Scenarios 61

CHAPTER 1

Building `ecs-install` Image From Sources

The ECS-CommunityEdition git repository is also a build environment for the `ecs-install` image.

1.1 Building `ecs-install` Image During Bootstrap with `boostrap.sh`

If you're hacking around in the install node code, then you'll probably want to build your own install node image at some point. The `bootstrap.sh` script has options for accomplishing just this.

```
[Usage]
 -h, --help
    Display this help text and exit
 --help-build
    Display build environment help and exit
 --version
    Display version information and exit

[Build Options]
 --zero-fill-ova
    Reduce ephemera, defrag, and zerofill the instance after bootstrapping
 --build-from <URL>
    Use the Alpine Linux mirror at <URL> to build the ecs-install image locally.
    Mirror list: https://wiki.alpinelinux.org/wiki/Alpine_Linux:Mirrors
```

All you'll need is a URL that points to an Alpine Linux mirror. For a good default, you can use the GeoDB enabled CDN mirror, which should auto-select a nearby mirror for you based on your public edge IP: http://dl-cdn.alpinelinux.org/alpine/

To tell bootstrap to build the image for you, just include the `--build-from` argument on your command line, like so:

```
[admin@localhost ECS-CommunityEdition]$ ./bootstrap.sh --build-from http://
dl-cdn.alpinelinux.org/alpine/
```

1.2 Building `ecs-install` Image After Bootstrapping with `build_image`

If you need to build the `ecs-install` image after bootstrapping, then you'll need to give a valid Alpine Linux mirror to your install node:

```
[admin@installer-230 ECS-CommunityEdition]$ build_image --update-mirror http://cache.
↪local/alpine/
> Updating bootstrap.conf to use Alpine Linux mirror http://cache.local/alpine/
```

Once the mirror is configured, you can then build the image:

```
[admin@installer-230 ECS-CommunityEdition]$ build_image
> Building image ecs-install
> Build context is: local
> Using custom registry: cache.local:5000
> Tunneling through proxy: cache.local:3128
> Checking Alpine Linux mirror
> Generating Alpine Linux repositories file
> Collecting artifacts
> UI artifact is: ui/resources/docker/ecs-install.2.5.1-local.installer-230.4.tgz
INFO[0000] FROM cache.local:5000/alpine:3.6
INFO[0000] | Image sha256:37eec                          size=3.962 MB
INFO[0000] LABEL MAINTAINER='Travis Wichert <travis.wichert@emc.com>'
INFO[0000] ENV ANSIBLE_CONFIG="/etc/ansible/ansible.cfg"
INFO[0000] ENV ANSIBLE_HOSTS="/usr/local/src/ui/inventory.py"
INFO[0000] Commit changes
INFO[0000] | Cached! Take image sha256:302bc    size=3.962 MB (+0 B)
INFO[0000] COPY ui/resources/docker/ecs-install-requirements.txt /etc/ecs-install-
↪requirements.txt
INFO[0000] | Calculating tarsum for 1 files (465 B total)
INFO[0000] | Cached! Take image sha256:44a83    size=3.962 MB (+465 B)
INFO[0000] COPY ui/resources/docker/apk-repositories /etc/apk/repositories
INFO[0000] | Calculating tarsum for 1 files (239 B total)
INFO[0000] | Not cached
INFO[0000] | Created container 89e5a010f1b5 (image sha256:44a83)
INFO[0000] | Uploading files to container 89e5a010f1b5
INFO[0000] Commit changes
INFO[0001] | Result image is sha256:26c0f                size=3.962 MB (+239 B)
INFO[0001] | Removing container 89e5a010f1b5
INFO[0001] ENV http_proxy=http://cache.local:3128
INFO[0001] ENV pip_proxy=cache.local:3128
INFO[0001] Commit changes
INFO[0002] | Created container 49b210eacd7c (image sha256:26c0f)
INFO[0002] | Result image is sha256:d9d58                size=3.962 MB (+0 B)
INFO[0002] | Removing container 49b210eacd7c
INFO[0003] RUN apk -q update &&     apk -q --no-cache upgrade
INFO[0003] | Created container 856a966289a6 (image sha256:d9d58)
INFO[0005] Commit changes
INFO[0006] | Result image is sha256:a2978                size=6.855 MB (+2.893 MB)
INFO[0006] | Removing container 856a966289a6
INFO[0006] RUN apk -q --no-cache add python2 py-pip                                 ␣
↪openssh-client sshpass openssl ca-certificates libffi libressl@edge_main           ␣
↪                  pigz jq less                              opentracker aria2␣
↪mktorrent@edge_community                                ansible@edge_main
INFO[0006] | Created container 2c940cb6c2e6 (image sha256:a2978)
```

(continues on next page)

(continued from previous page)

```
INFO[0016] Commit changes
INFO[0026] | Result image is sha256:b806e                size=124.4 MB (+117.6 MB)
INFO[0026] | Removing container 2c940cb6c2e6
INFO[0026] RUN mv /etc/profile.d/color_prompt /etc/profile.d/color_prompt.sh     &&␣
↪ln -s /usr/local/src/ui/ansible /ansible     && ln -s /usr/local/src/ui /ui     &&␣
↪ln -s /usr/local/src /src     && ln -s /usr/bin/python /usr/local/bin/python     &&␣
↪mkdir -p /var/run/opentracker     && chown nobody:nobody /var/run/opentracker
INFO[0027] | Created container a5a35a59e61a (image sha256:b806e)
INFO[0027] Commit changes
INFO[0029] | Result image is sha256:55ae2                size=124.4 MB (+295 B)
INFO[0029] | Removing container a5a35a59e61a
INFO[0029] RUN apk -q --no-cache add --update --virtual .build-deps musl-dev python2-
↪dev libffi-dev                        build-base make openssl-dev linux-headers git␣
↪gcc git-perl     && if ! [ -z "$pip_proxy" ]; then             export pip_proxy="--
↪proxy $pip_proxy" &&             git config --global http.proxy "$http_proxy"      ␣
↪ ;fi     && pip install -q $pip_proxy --no-cache-dir -r /etc/ecs-install-
↪requirements.txt     && apk -q --no-cache --purge del .build-deps
INFO[0030] | Created container 4d07a461385a (image sha256:55ae2)
INFO[0184] Commit changes
INFO[0187] | Result image is sha256:79f09                size=151.1 MB (+26.68 MB)
INFO[0187] | Removing container 4d07a461385a
INFO[0187] RUN mkdir -p /etc/ansible
INFO[0188] | Created container 021968b10369 (image sha256:79f09)
INFO[0188] Commit changes
INFO[0190] | Result image is sha256:376dc                size=151.1 MB (+0 B)
INFO[0190] | Removing container 021968b10369
INFO[0191] COPY ui/resources/docker/ansible.cfg /etc/ansible/ansible.cfg
INFO[0191] | Calculating tarsum for 1 files (5.437 kB total)
INFO[0191] | Created container acf602cb1215 (image sha256:376dc)
INFO[0191] | Uploading files to container acf602cb1215
INFO[0191] Commit changes
INFO[0193] | Result image is sha256:a3b7d                size=151.1 MB (+5.437 kB)
INFO[0193] | Removing container acf602cb1215
INFO[0193] COPY ui/resources/docker/entrypoint.sh /usr/local/bin/entrypoint.sh
INFO[0193] | Calculating tarsum for 1 files (5.844 kB total)
INFO[0194] | Created container d2e1e94bba06 (image sha256:a3b7d)
INFO[0194] | Uploading files to container d2e1e94bba06
INFO[0194] Commit changes
INFO[0196] | Result image is sha256:c0530                size=151.1 MB (+5.844 kB)
INFO[0196] | Removing container d2e1e94bba06
INFO[0196] RUN chmod +x /usr/local/bin/entrypoint.sh
INFO[0197] | Created container 58814799d1c4 (image sha256:c0530)
INFO[0197] Commit changes
INFO[0199] | Result image is sha256:6fa79                size=151.1 MB (+0 B)
INFO[0199] | Removing container 58814799d1c4
INFO[0200] ENTRYPOINT [ "/usr/local/bin/entrypoint.sh" ]
INFO[0200] Commit changes
INFO[0200] | Created container dc4494fd062f (image sha256:6fa79)
INFO[0202] | Result image is sha256:481e1                size=151.1 MB (+0 B)
INFO[0202] | Removing container dc4494fd062f
INFO[0202] COPY ui/resources/docker/torrent.sh /usr/local/bin/torrent.sh
INFO[0202] | Calculating tarsum for 1 files (890 B total)
INFO[0203] | Created container 9f15d6413cd2 (image sha256:481e1)
INFO[0203] | Uploading files to container 9f15d6413cd2
INFO[0203] Commit changes
INFO[0205] | Result image is sha256:35f06                size=151.1 MB (+890 B)
INFO[0205] | Removing container 9f15d6413cd2
```

(continues on next page)

(continued from previous page)

```
INFO[0205] COPY ui/resources/docker/ecs-install.2.5.1-local.installer-230.4.tgz /usr/
↪local/src/ui.tgz
INFO[0205] | Calculating tarsum for 1 files (3.958 MB total)
INFO[0206] | Created container e6542b37ddc7 (image sha256:35f06)
INFO[0206] | Uploading files to container e6542b37ddc7
INFO[0206] Commit changes
INFO[0208] | Result image is sha256:161f5                size=155.1 MB (+3.958 MB)
INFO[0208] | Removing container e6542b37ddc7
INFO[0208] ENV http_proxy=
INFO[0208] ENV pip_proxy=
INFO[0208] VOLUME [ "/opt", "/usr", "/var/log", "/root", "/etc" ]
INFO[0208] LABEL VERSION=cache.local:5000/emccorp/ecs-install:2.5.1-local.installer-
↪230.4
INFO[0208] ENV VERSION=cache.local:5000/emccorp/ecs-install:2.5.1-local.installer-230.
↪4
INFO[0208] Commit changes
INFO[0213] | Created container 7beb4650354e (image sha256:161f5)
INFO[0216] | Result image is sha256:7bd3d                size=155.1 MB (+0 B)
INFO[0216] | Removing container 7beb4650354e
INFO[0217] TAG cache.local:5000/emccorp/ecs-install:2.5.1-local.installer-230.4
INFO[0217] | Tag sha256:7bd3d -> cache.local:5000/emccorp/ecs-install:2.5.1-local.
↪installer-230.4
INFO[0217] Cleaning up
INFO[0217] Successfully built sha256:7bd3d | final size 155.1 MB (+151.1 MB from the␣
↪base image)
> Tagging cache.local:5000/emccorp/ecs-install:2.5.1-local.installer-230.4 -> emccorp/
↪ecs-install:latest
```

The new image is automatically tagged :latest in the local repository and replaces any previous :latest images.

You'll then want to clean up the local Docker repository with this command:

```
[admin@installer-230 ECS-CommunityEdition]$ build_image --clean
> Cleaning up...
>       [build tmp containers]
>       [ecs-install data containers]
>       [exited containers]
>       [dangling layers]
```

1.3 Making Quick Iterative Changes to an Existing `ecs-install` Image with `update_image`

Building an image can take a long time. If you have not made any changes to files that are used in the `docker build` process, then you can update an existing `ecs-install` data container with code changes using the `update_image` macro:

```
[admin@installer-230 ECS-CommunityEdition]$ update_image
> Updating image: ecs-install
> Build context is: local
> Tunneling through proxy: cache.local:3128
> Cleaning up...
>       [build tmp containers]
>       [ecs-install data containers]
>       [exited containers]
```

(continues on next page)

(continued from previous page)

```
>      [dangling layers]
> Collecting artifacts
> UI is: ui/resources/docker/ecs-install.2.5.1-local.installer-230.5.tgz
> Creating new data container
> Image updated.
```

1.4 Quickly Testing Ansible Changes with `testbook`

If you're working with Ansible within ECS Community Edition, you might find yourself needing to test to see how your Ansible role is being played from within the `ecs-install` image. You can do this by modifying the files under the `testing` subdirectory of the Ansible `roles` directory: `ui/ansible/roles/testing`

After making your changes, run `update_image` as discussed above, and then run `testbook` to execute your role. The `testbook` command will automatically initialize a new data container, configure access with the install node, and test your role directives.

CHAPTER 2

ECS Community Edition Installation Guides

For **Standard** installations (Internet connected, from source) use this guide.

For **Island** installations (Isolated environment, from source) use this guide.

For **OVA** installations (connectivity agnostic, from OVA) use this guide.

For information on deploy.yml file available options use this guide.

CHAPTER 3

Standard Installation

The standard installation assumes an Internet connected VM which will be bootstrapped and become an install node. The ECS deployment will then proceed from the install node.

3.1 Prerequisites

Listed below are all necessary components for a successful ECS Community Edition installation. If they are not met the installation will likely fail.

3.1.1 Hardware Requirements

The installation process is designed to be performed from either a dedicated installation node. However, it is possible, if you so choose, for one of the ECS data nodes to double as the install node. The install node will bootstrap the ECS data nodes and configure the ECS instance. When the process is complete, the install node may be safely destroyed. Both single node and multi-node deployments require only a single install node.

The technical requirements for the installation node are minimal, but reducing available CPU, memory, and IO throughput will adversely affect the speed of the installation process:

- 1 CPU Core
- 2 GB Memory
- 10 GB HDD
- CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

The minimum technical requirements for each ECS data node are:

- 4 CPU Cores
- 16 GB Memory
- 16 GB Minimum system block storage device
- 104 GB Minimum additional block storage device in a raw, unpartitioned state.

- CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

The recommended technical requirements for each ECS data node are:

- 8 CPU Cores
- 64GB RAM
- 16GB root block storage
- 1TB additional block storage
- CentOS 7.4 Minimal installation

For multi-node installations each data node must fulfill these minimum qualifications. The installer will do a pre-flight check to ensure that the minimum qualifications are met. If they are not, the installation will not continue.

3.1.2 Environmental Requirements

The following environmental requirements must also be met to ensure a successful installation:

- **Network:** All nodes, including install node and ECS data node(s), must exist on the same IPv4 subnet. IPv6 networking *may* work, but is neither tested nor supported for ECS Community Edition at this time.
- **Remote Access:** Installation is coordinated via Ansible and SSH. However, public key authentication during the initial authentication and access configuration is not yet supported. Therefore, password authentication must be enabled on all nodes, including install node and ECS data node(s). *This is a known issue and will be addressed in a future release*
- **OS:** CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

3.1.3 All-in-One Single-Node Deployments

A single node *can* successfully run the installation procedure on itself. To do this simply input the node's own IP address as the installation node as well as the data node in the deploy.yml file.

3.2 1. Getting Started

It is recommended to use a non-root administrative user account with sudo privileges on the install node when performing the deployment. Deploying from the root account is supported, but not recommended.

Before data store nodes can be created, the install node must be prepared. If acquiring the software via the GitHub repository, run:

0. `cd $HOME`
1. `sudo yum install -y git`
2. `git clone https://github.com/EMCECS/ECS-CommunityEdition`.

If the repository is being added to the machine via usb drive, scp, or some other file-based means, please copy the archive into `$HOME/` and run:

- for .zip archive `unzip ECS-CommunityEdition.zip`
- for .tar.gz archive `tar -xzvf ECS-CommunityEdition.tar.gz`

If the directory created when unarchiving the release .zip or tarball has a different name than `ECS-CommunityEdition`, then rename it with the following command:

0. `mv <directory name> ECS-CommunityEdition`

This will help the documentation make sense as you proceed with the deployment.

3.3 2. Creating The Deployment Map (`deploy.yml`)

Installation requires the creation of a deployment map. This map is represented in a YAML configuration file called deploy.yml.

Below are steps for creating a basic deploy.yml. **All fields indicated below are required for a successful installation.**

0. From the `$HOME/ECS-CommunityEdition` directory, run the commmand: `cp docs/design/reference.deploy.yml deploy.yml`
1. Edit the file with your favorite editor on another machine, or use `vi deploy.yml` on the install node. Read the comments in the file and review the examples in the `examples/` directory.
2. Top-level deployment facts (`facts:`)
   () Enter the IP address of the install node into the `install_node:` field.
   (a) Enter into the `management_clients:` field the CIDR address/mask of each machine or subnet that will be whitelisted in node's firewalls and allowed to communicate with ECS management API.
   - `10.1.100.50/32` is *exactly* the IP address.
   - `192.168.2.0/24` is the entire /24 subnet.
   - `0.0.0.0/0` represents the entire Internet.
3. SSH login details (`ssh_defaults:`)
   () If the SSH server is bound to a non-standard port, enter that port number in the `ssh_port:` field, or leave it set at the default (22).
   (a) Enter the username of a user permitted to run commands as UID 0/GID 0 ("root") via the `sudo` command into the `ssh_username:` field. This must be the same across all nodes.
   (b) Enter the password for the above user in the `ssh_password:` field. This will only be used during the initial public key authentication setup and can be changed after. This must be the same across all nodes.
4. Node configuration (`node_defaults:`)
   () Enter the DNS domain for the ECS installation. This can simply be set to `localdomain` if you will not be using DNS with this ECS deployment.
   (a) Enter each DNS server address, one per line, into `dns_servers:`. This can be what's present in `/etc/resolv.conf`, or it can be a different DNS server entirely. This DNS server will be set to the primary DNS server for each ECS node.
   (b) Enter each NTP server address, one per line, into `ntp_servers:`.
5. Storage Pool configuration (`storage_pools:`)
   () Enter the storage pool `name:`.
   (a) Enter each member data node's IP address, one per line, in `members:`.
   (b) Under `options:`, enter each block device reserved for ECS, one per line, in `ecs_block_devices:`. All member data nodes of a storage pool must be identical.
6. Virtual Data Center configuration (`virtual_data_centers:`)
   () Enter each VDC `name:`.

(a) Enter each member Storage Pool name, one per line, in `members`:

7. Optional directives, such as those for Replication Groups and users, may also be configured at this time.
8. When you have completed the `deploy.yml` to your liking, save the file and exit the `vi` editor.
9. Move on to Bootstrapping

These steps quickly set up a basic deploy.yml file

3.3.1 More on deploy.yml

If you need to make changes to your deploy.yml after bootstrapping, there are two utilities for this.

0. The `videploy` utility will update the installed `deploy.yml` file in place and is the preferred method.
1. The `update_deploy` utility will update the installed `deploy.yml` file with the contents of a different `deploy.yml` file.

See the [utilties][utilities] document for more information on these and other ECS CE utilities.

For more information on deploy.yml, please read the reference guide found here.

3.4 3. Bootstrapping the Install Node (`bootstrap.sh`)

The bootstrap script configures the installation node for ECS deployment and downloads the required Docker images and software packages that all other nodes in the deployment will need for successful installation.

Once the deploy.yml file has been created, the installation node must be bootstrapped. To do this `cd` into the ECS-CommunityEdition directory and run `./bootstrap.sh -c deploy.yml`. Be sure to add the `-g` flag if building the ECS deployment in a virtual environment and the `-y` flag if you're okay accepting all defaults.

The bootstrap script accepts many flags. If your environment uses proxies, including MitM SSL proxies, custom nameservers, or a local Docker registry or CentOS mirror, you may want to indicate that on the `bootstrap.sh` command line.

```
[Usage]
 -h, --help
    Display this help text and exit
 --help-build
    Display build environment help and exit
 --version
    Display version information and exit

[General Options]
 -y / -n
    Assume YES or NO to any questions (may be dangerous).
 -v / -q
    Be verbose (also show all logs) / Be quiet (only show necessary output)
 -c <FILE>
    If you have a deploy.yml ready to go, give its path to this arg.

[Platform Options]
 --ssh-private-key <id_rsa | id_ed25519>
 --ssh-public-key <id_rsa.pub | id_ed25519.pub>
    Import SSH public key auth material and use it when authenticating to remote␣
↪nodes.
 -o, --override-dns <NS1,NS2,NS*>
```

(continues on next page)

(continued from previous page)

```
    Override DHCP-configured nameserver(s); use these instead. No spaces! Use of -o␣
↪is deprecated, please use --override-dns.
 -g, --vm-tools
    Install virtual machine guest agents and utilities for QEMU and VMWare.␣
↪VirtualBox is not supported at this time. Use of -g is deprecated, please use --vm-
↪tools.
 -m, --centos-mirror <URL>
    Use the provided package <mirror> when fetching packages for the base OS (but not␣
↪3rd-party sources, such as EPEL or Debian-style PPAs). The mirror is specified as '
↪<host>:<port>'. This option overrides any mirror lists the base OS would normally␣
↪use AND supersedes any proxies (assuming the mirror is local), so be warned that␣
↪when using this option it's possible for bootstrapping to hang indefinitely if the␣
↪mirror cannot be contacted. Use of -m is deprecated, please use --centos-mirror.

[Docker Options]
 -r, --registry-endpoint REGISTRY
    Use the Docker registry at REGISTRY instead of DockerHub. The connect string is␣
↪specified as '<host>:<port>[/<username>]'. You may be prompted for your credentials␣
↪if authentication is required. You may need to use -d (below) to add the registry's␣
↪cert to Docker. Use of -r is deprecated, please use --registry-endpoint.

 -l, --registry-login
    After Docker is installed, login to the Docker registry to access images which␣
↪require access authentication. This will authenticate with Dockerhub unless --
↪registry-endpoint is also used. Use of -l is deprecated, please use --registry-
↪login.

 -d, --registry-cert <FILE>
    [Requires --registry-endpoint] If an alternate Docker registry was specified with␣
↪-r and uses a cert that cannot be resolved from the anchors in the local system's␣
↪trust store, then use -d to specify the x509 cert file for your registry.

[Proxies & Middlemen]
 -p, --proxy-endpoint <PROXY>
    Connect to the Internet via the PROXY specified as '[user:pass@]<host>:<port>'.␣
↪Items in [] are optional. It is assumed this proxy handles all protocols.  Use of -
↪p is deprecated, please use --proxy-endpoint.
 -k, --proxy-cert <FILE>
    Install the certificate in <file> into the local trust store. This is useful for␣
↪environments that live behind a corporate HTTPS proxy.  Use of -k is deprecated,␣
↪please use --proxy-cert.
 -t, --proxy-test-via <HOSTSPEC>
    [Requires --proxy-endpoint] Test Internet connectivity through the PROXY by␣
↪connecting to HOSTSPEC. HOSTSPEC is specified as '<host>:<port>'. By default
↪'google.com:80' is used. Unless access to Google is blocked (or vice versa), there␣
↪is no need to change the default.

[Examples]
 Install VM guest agents and use SSH public key auth keys in the .ssh/ directory.
    $ bash bootstrap.sh --vm-tools --ssh-private-key .ssh/id_rsa --ssh-public-key .
↪ssh/id_rsa.pub

 Quietly use nlanr.peer.local on port 80 and test the connection using EMC's␣
↪webserver.
    $ bash bootstrap.sh -q --proxy-endpoint nlanr.peer.local:80 -proxy-test-via emc.
↪com:80
```

(continues on next page)

(continued from previous page)

```
 Assume YES to all questions. Use the CentOS mirror at http://cache.local/centos when␣
↪fetching packages. Use the Docker registry at registry.local:5000 instead of␣
↪DockerHub, and install the x509 certificate in certs/reg.pem into Docker's trust␣
↪store so it can access the Docker registry.
    $ bash bootstrap.sh -y --centos-mirror http://cache.local/centos --registry-
↪endpoint registry.local:5000 --registry-cert certs/reg.pem
```

The bootstrapping process has completed when the following message appears:

```
> All done bootstrapping your install node.
>
> To continue (after reboot if needed):
>     $ cd /home/admin/ECS-CommunityEdition
> If you have a deploy.yml ready to go (and did not use -c flag):
>     $ sudo cp deploy.yml /opt/emc/ecs-install/
> If not, check out the docs/design and examples directory for references.
> Once you have a deploy.yml, you can start the deployment
> by running:
>
> [WITH Internet access]
>     $ step1
>   [Wait for deployment to complete, then run:]
>     $ step2
>
> [WITHOUT Internet access]
>     $ island-step1
>   [Migrate your install node into the isolated environment and run:]
>     $ island-step2
>   [Wait for deployment to complete, then run:]
>     $ island-step3
>
```

After the installation node has successfully bootstrapped you will likely be prompted to reboot the machine. If so, then the machine MUST be rebooted before continuing to Step 4.

3.5 4. Deploying ECS Nodes (`step1`)

Once the deploy.yml file has been correctly written and the install node rebooted if needed, then the next step is to simply run `step1`.

After the installer initializes, the EMC ECS license agreement will appear on the screen. Press `q` to close the screen and type `yes` to accept the license and continue or `no` to abort the process. The install cannot continue until the license agreement has been accepted.

3.6 5. Deploying ECS Topology (`step2`)

If you would prefer to manually configure your ECS topology, you may skip this step entirely.

Once `step1` has completed, you may then direct the installer to configure the ECS topology by running `step2`. Once `step2` has completed, your ECS will be ready for use.

Assuming all went well, you now have a functioning ECS Community Edition instance and you may now proceed with your test efforts.

CHAPTER 4

OVA Installation

The OVA installation assumes deployment in a network-isolated environment. One clone of the OVA will become an install node. The ECS deployment will then proceed from the install node.

4.1 Prerequisites

Listed below are all necessary components for a successful ECS Community Edition installation. If they are not met the installation will likely fail.

4.1.1 Hardware Requirements

The installation process is designed to be performed from either a dedicated installation node. However, it is possible, if you so choose, for one of the ECS data nodes to double as the install node. The install node will bootstrap the ECS data nodes and configure the ECS instance. When the process is complete, the install node may be safely destroyed. Both single node and multi-node deployments require only a single install node.

The technical requirements for the installation node are minimal, but reducing available CPU, memory, and IO throughput will adversely affect the speed of the installation process:

- 1 CPU Core
- 2 GB Memory
- 10 GB HDD
- CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

The minimum technical requirements for each ECS data node are:

- 4 CPU Cores
- 16 GB Memory
- 16 GB Minimum system block storage device
- 104 GB Minimum additional block storage device in a raw, unpartitioned state.

- CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

The recommended technical requirements for each ECS data node are:

- 8 CPU Cores
- 64GB RAM
- 16GB root block storage
- 1TB additional block storage
- CentOS 7.4 Minimal installation

For multi-node installations each data node must fulfill these minimum qualifications. The installer will do a pre-flight check to ensure that the minimum qualifications are met. If they are not, the installation will not continue.

4.1.2 Environmental Requirements

The following environmental requirements must also be met to ensure a successful installation:

- **Network:** All nodes, including install node and ECS data node(s), must exist on the same IPv4 subnet. IPv6 networking *may* work, but is neither tested nor supported for ECS Community Edition at this time.
- **Remote Access:** Installation is coordinated via Ansible and SSH. However, public key authentication during the initial authentication and access configuration is not yet supported. Therefore, password authentication must be enabled on all nodes, including install node and ECS data node(s). *This is a known issue and will be addressed in a future release*
- **OS:** CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

4.1.3 All-in-One Single-Node Deployments

A single node *can* successfully run the installation procedure on itself. To do this simply input the node's own IP address as the installation node as well as the data node in the deploy.yml file.

4.2 1. Getting Started

4.2.1 1.1. Download and deploy the OVA to a VM

The OVA is available for download from the release notes page. Select the most recent version of the OVA for the best experience.

4.2.2 1.2. Deploy a VM from the OVA and Adjust its resources to have a minimum of:

- 16GB RAM
- 4 CPU cores
- (Optional) Increase vmdk from the minimum 104GB

4.2.3 1.3. Clone the VM

Clone the VM you created enough times to reach the number of nodes desired for your deployment. The minimum number of nodes for basic functionality is one (1). The minimum number of nodes for erasure coding replication to be enabled is four (4).

4.2.4 1.4. Collect and Configure networking information

Power on the VMs and collect their DHCP assigned IP addresses from the vCenter client or from the VMs themselves

You may also assign static IP addresses by logging into each VM and running `nmtui` to set network the network variables (IP, mask, gateway, DNS, etc).

The information you collect in this step is crucial for step 2.

4.3 2. Creating The Deployment Map (`deploy.yml`)

Installation requires the creation of a deployment map. This map is represented in a YAML configuration file called deploy.yml.

Below are steps for creating a basic deploy.yml. **All fields indicated below are required for a successful installation.**

0. Log into the first VM and run `videploy`.
1. Edit this deploy.yml file with your favorite editor on another machine, or use `vi deploy.yml` on the install node. Read the comments in the file and review the examples in the `examples/` directory.
2. Top-level deployment facts (`facts:`)
   () Enter the IP address of the install node into the `install_node:` field.
   (a) Enter into the `management_clients:` field the CIDR address/mask of each machine or subnet that will be whitelisted in node's firewalls and allowed to communicate with ECS management API.
   - `10.1.100.50/32` is *exactly* the IP address.
   - `192.168.2.0/24` is the entire /24 subnet.
   - `0.0.0.0/0` represents the entire Internet.
3. SSH login details (`ssh_defaults:`)
   () If the SSH server is bound to a non-standard port, enter that port number in the `ssh_port:` field, or leave it set at the default (22).
   (a) Enter the username of a user permitted to run commands as UID 0/GID 0 ("root") via the `sudo` command into the `ssh_username:` field. This must be the same across all nodes.
   (b) Enter the password for the above user in the `ssh_password:` field. This will only be used during the initial public key authentication setup and can be changed after. This must be the same across all nodes.
4. Node configuration (`node_defaults:`)
   () Enter the DNS domain for the ECS installation. This can simply be set to `localdomain` if you will not be using DNS with this ECS deployment.
   (a) Enter each DNS server address, one per line, into `dns_servers:`. This can be what's present in `/etc/resolv.conf`, or it can be a different DNS server entirely. This DNS server will be set to the primary DNS server for each ECS node.
   (b) Enter each NTP server address, one per line, into `ntp_servers:`.

5. Storage Pool configuration (`storage_pools:`)
   () Enter the storage pool `name:`.
   (a) Enter each member data node's IP address, one per line, in `members:`.
   (b) Under `options:`, enter each block device reserved for ECS, one per line, in `ecs_block_devices:`. All member data nodes of a storage pool must be identical.
6. Virtual Data Center configuration (`virtual_data_centers:`)
   () Enter each VDC `name:`.
   (a) Enter each member Storage Pool name, one per line, in `members:`
7. Optional directives, such as those for Replication Groups and users, may also be configured at this time.
8. After completing the deploy.yml file to your liking, exit out of `videploy` as you would the `vim` editor (ESC, :, wq, ENTER). This will update the deploy.yml file.

4.3.1 More on deploy.yml

If you need to make changes to your deploy.yml after bootstrapping, there are two utilities for this.

0. The `videploy` utility will update the installed `deploy.yml` file in place and is the preferred method.
1. The `update_deploy` utility will update the installed `deploy.yml` file with the contents of a different `deploy.yml` file.

See the [utilties][utilities] document for more information on these and other ECS CE utilities.

For more information on deploy.yml, please read the reference guide found here.

4.4 4. Deploying ECS Nodes (`ova-step1`)

Once the deploy.yml file has been correctly written and the install node rebooted if needed, then the next step is to simply run `ova-step1`.

After the installer initializes, the EMC ECS license agreement will appear on the screen. Press `q` to close the screen and type `yes` to accept the license and continue or `no` to abort the process. The install cannot continue until the license agreement has been accepted.

4.5 5. Deploying ECS Topology (`ova-step2`)

If you would prefer to manually configure your ECS topology, you may skip this step entirely.

Once `ova-step1` has completed, you may then direct the installer to configure the ECS topology by running `ova-step2`. Once `ova-step2` has completed, your ECS will be ready for use.

Assuming all went well, you now have a functioning ECS Community Edition instance and you may now proceed with your test efforts.

CHAPTER 5

Island Installation

The island installation assumes an Internet connected VM which will be bootstrapped and become an install node. The install node will be migrated into a network-isolated environment and the ECS deployment will then proceed from the install node.

5.1 Prerequisites

Listed below are all necessary components for a successful ECS Community Edition installation. If they are not met the installation will likely fail.

5.1.1 Hardware Requirements

The installation process is designed to be performed from either a dedicated installation node. However, it is possible, if you so choose, for one of the ECS data nodes to double as the install node. The install node will bootstrap the ECS data nodes and configure the ECS instance. When the process is complete, the install node may be safely destroyed. Both single node and multi-node deployments require only a single install node.

The technical requirements for the installation node are minimal, but reducing available CPU, memory, and IO throughput will adversely affect the speed of the installation process:

- 1 CPU Core
- 2 GB Memory
- 10 GB HDD
- CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

The minimum technical requirements for each ECS data node are:

- 4 CPU Cores
- 16 GB Memory
- 16 GB Minimum system block storage device

- 104 GB Minimum additional block storage device in a raw, unpartitioned state.
- CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

The recommended technical requirements for each ECS data node are:

- 8 CPU Cores
- 64GB RAM
- 16GB root block storage
- 1TB additional block storage
- CentOS 7.4 Minimal installation

For multi-node installations each data node must fulfill these minimum qualifications. The installer will do a pre-flight check to ensure that the minimum qualifications are met. If they are not, the installation will not continue.

5.1.2 Environmental Requirements

The following environmental requirements must also be met to ensure a successful installation:

- **Network:** All nodes, including install node and ECS data node(s), must exist on the same IPv4 subnet. IPv6 networking *may* work, but is neither tested nor supported for ECS Community Edition at this time.
- **Remote Access:** Installation is coordinated via Ansible and SSH. However, public key authentication during the initial authentication and access configuration is not yet supported. Therefore, password authentication must be enabled on all nodes, including install node and ECS data node(s). *This is a known issue and will be addressed in a future release*
- **OS:** CentOS 7 Minimal installation (ISO- and network-based minimal installs are equally supported)

5.1.3 All-in-One Single-Node Deployments

A single node *can* successfully run the installation procedure on itself. To do this simply input the node's own IP address as the installation node as well as the data node in the deploy.yml file.

5.2 1. Getting Started

It is recommended to use a non-root administrative user account with sudo privileges on the install node when performing the deployment. Deploying from the root account is supported, but not recommended.

Before data store nodes can be created, the install node must be prepared. If acquiring the software via the GitHub repository, run:

0. `cd $HOME`
1. `sudo yum install -y git`
2. `git clone https://github.com/EMCECS/ECS-CommunityEdition`.

If the repository is being added to the machine via usb drive, scp, or some other file-based means, please copy the archive into `$HOME/` and run:

- for .zip archive `unzip ECS-CommunityEdition.zip`
- for .tar.gz archive `tar -xzvf ECS-CommunityEdition.tar.gz`

If the directory created when unarchiving the release .zip or tarball has a different name than `ECS-CommunityEdition`, then rename it with the following command:

0. `mv <directory name> ECS-CommunityEdition`

This will help the documentation make sense as you proceed with the deployment.

5.3 2. Creating The Deployment Map (`deploy.yml`)

Installation requires the creation of a deployment map. This map is represented in a YAML configuration file called deploy.yml.

Below are steps for creating a basic deploy.yml. **All fields indicated below are required for a successful installation.**

0. From the `$HOME/ECS-CommunityEdition` directory, run the commmand: `cp docs/design/reference.deploy.yml deploy.yml`
1. Edit the file with your favorite editor on another machine, or use `vi deploy.yml` on the install node. Read the comments in the file and review the examples in the `examples/` directory.
2. Top-level deployment facts (`facts:`)
   () Enter the IP address of the install node into the `install_node:` field.
   (a) Enter into the `management_clients:` field the CIDR address/mask of each machine or subnet that will be whitelisted in node's firewalls and allowed to communicate with ECS management API.
   - `10.1.100.50/32` is *exactly* the IP address.
   - `192.168.2.0/24` is the entire /24 subnet.
   - `0.0.0.0/0` represents the entire Internet.
3. SSH login details (`ssh_defaults:`)
   () If the SSH server is bound to a non-standard port, enter that port number in the `ssh_port:` field, or leave it set at the default (22).
   (a) Enter the username of a user permitted to run commands as UID 0/GID 0 ("root") via the `sudo` command into the `ssh_username:` field. This must be the same across all nodes.
   (b) Enter the password for the above user in the `ssh_password:` field. This will only be used during the initial public key authentication setup and can be changed after. This must be the same across all nodes.
4. Node configuration (`node_defaults:`)
   () Enter the DNS domain for the ECS installation. This can simply be set to `localdomain` if you will not be using DNS with this ECS deployment.
   (a) Enter each DNS server address, one per line, into `dns_servers:`. This can be what's present in `/etc/resolv.conf`, or it can be a different DNS server entirely. This DNS server will be set to the primary DNS server for each ECS node.
   (b) Enter each NTP server address, one per line, into `ntp_servers:`.
5. Storage Pool configuration (`storage_pools:`)
   () Enter the storage pool `name:`.
   (a) Enter each member data node's IP address, one per line, in `members:`.
   (b) Under `options:`, enter each block device reserved for ECS, one per line, in `ecs_block_devices:`. All member data nodes of a storage pool must be identical.

6. Virtual Data Center configuration (`virtual_data_centers:`)
   () Enter each VDC `name:`.
   (a) Enter each member Storage Pool name, one per line, in `members:`
7. Optional directives, such as those for Replication Groups and users, may also be configured at this time.
8. When you have completed the `deploy.yml` to your liking, save the file and exit the `vi` editor.
9. Move on to Bootstrapping

These steps quickly set up a basic deploy.yml file

5.3.1 More on deploy.yml

If you need to make changes to your deploy.yml after bootstrapping, there are two utilities for this.

0. The `videploy` utility will update the installed `deploy.yml` file in place and is the preferred method.
1. The `update_deploy` utility will update the installed `deploy.yml` file with the contents of a different `deploy.yml` file.

See the [utilties][utilities] document for more information on these and other ECS CE utilities.

For more information on deploy.yml, please read the reference guide found here.

5.4 3. Bootstrapping the install node (`bootstrap.sh`)

The bootstrap script configures the installation node for ECS deployment and downloads the required Docker images and software packages that all other nodes in the deployment will need for successful installation.

Once the deploy.yml file has been created, the installation node must be bootstrapped. To do this `cd` into the ECS-CommunityEdition directory and run `./bootstrap.sh -c deploy.yml`. Be sure to add the `-g` flag if building the ECS deployment in a virtual environment and the `-y` flag if you're okay accepting all defaults.

The bootstrap script accepts many flags. If your environment uses proxies, including MitM SSL proxies, custom nameservers, or a local Docker registry or CentOS mirror, you may want to indicate that on the `bootstrap.sh` command line.

```
[Usage]
 -h, --help
    Display this help text and exit
 --help-build
    Display build environment help and exit
 --version
    Display version information and exit

[General Options]
 -y / -n
    Assume YES or NO to any questions (may be dangerous).
 -v / -q
    Be verbose (also show all logs) / Be quiet (only show necessary output)
 -c <FILE>
    If you have a deploy.yml ready to go, give its path to this arg.

[Platform Options]
 --ssh-private-key <id_rsa | id_ed25519>
```

(continues on next page)

(continued from previous page)

```
 --ssh-public-key <id_rsa.pub | id_ed25519.pub>
    Import SSH public key auth material and use it when authenticating to remote␣
↪nodes.
 -o, --override-dns <NS1,NS2,NS*>
    Override DHCP-configured nameserver(s); use these instead. No spaces! Use of -o␣
↪is deprecated, please use --override-dns.
 -g, --vm-tools
    Install virtual machine guest agents and utilities for QEMU and VMWare.␣
↪VirtualBox is not supported at this time. Use of -g is deprecated, please use --vm-
↪tools.
 -m, --centos-mirror <URL>
    Use the provided package <mirror> when fetching packages for the base OS (but not␣
↪3rd-party sources, such as EPEL or Debian-style PPAs). The mirror is specified as '
↪<host>:<port>'. This option overrides any mirror lists the base OS would normally␣
↪use AND supersedes any proxies (assuming the mirror is local), so be warned that␣
↪when using this option it's possible for bootstrapping to hang indefinitely if the␣
↪mirror cannot be contacted. Use of -m is deprecated, please use --centos-mirror.

[Docker Options]
 -r, --registry-endpoint REGISTRY
    Use the Docker registry at REGISTRY instead of DockerHub. The connect string is␣
↪specified as '<host>:<port>[/<username>]'. You may be prompted for your credentials␣
↪if authentication is required. You may need to use -d (below) to add the registry's␣
↪cert to Docker. Use of -r is deprecated, please use --registry-endpoint.

 -l, --registry-login
    After Docker is installed, login to the Docker registry to access images which␣
↪require access authentication. This will authenticate with Dockerhub unless --
↪registry-endpoint is also used. Use of -l is deprecated, please use --registry-
↪login.

 -d, --registry-cert <FILE>
    [Requires --registry-endpoint] If an alternate Docker registry was specified with␣
↪-r and uses a cert that cannot be resolved from the anchors in the local system's␣
↪trust store, then use -d to specify the x509 cert file for your registry.

[Proxies & Middlemen]
 -p, --proxy-endpoint <PROXY>
    Connect to the Internet via the PROXY specified as '[user:pass@]<host>:<port>'.␣
↪Items in [] are optional. It is assumed this proxy handles all protocols.  Use of -
↪p is deprecated, please use --proxy-endpoint.
 -k, --proxy-cert <FILE>
    Install the certificate in <file> into the local trust store. This is useful for␣
↪environments that live behind a corporate HTTPS proxy.  Use of -k is deprecated,␣
↪please use --proxy-cert.
 -t, --proxy-test-via <HOSTSPEC>
    [Requires --proxy-endpoint] Test Internet connectivity through the PROXY by␣
↪connecting to HOSTSPEC. HOSTSPEC is specified as '<host>:<port>'. By default
↪'google.com:80' is used. Unless access to Google is blocked (or vice versa), there␣
↪is no need to change the default.

[Examples]
 Install VM guest agents and use SSH public key auth keys in the .ssh/ directory.
    $ bash bootstrap.sh --vm-tools --ssh-private-key .ssh/id_rsa --ssh-public-key .
↪ssh/id_rsa.pub

 Quietly use nlanr.peer.local on port 80 and test the connection using EMC's␣
↪webserver.
```

(continues on next page)

(continued from previous page)

```
    $ bash bootstrap.sh -q --proxy-endpoint nlanr.peer.local:80 -proxy-test-via emc.
↪com:80

 Assume YES to all questions. Use the CentOS mirror at http://cache.local/centos when␣
↪fetching packages. Use the Docker registry at registry.local:5000 instead of␣
↪DockerHub, and install the x509 certificate in certs/reg.pem into Docker's trust␣
↪store so it can access the Docker registry.
    $ bash bootstrap.sh -y --centos-mirror http://cache.local/centos --registry-
↪endpoint registry.local:5000 --registry-cert certs/reg.pem
```

The bootstrapping process has completed when the following message appears:

```
> All done bootstrapping your install node.
>
> To continue (after reboot if needed):
>     $ cd /home/admin/ECS-CommunityEdition
> If you have a deploy.yml ready to go (and did not use -c flag):
>     $ sudo cp deploy.yml /opt/emc/ecs-install/
> If not, check out the docs/design and examples directory for references.
> Once you have a deploy.yml, you can start the deployment
> by running:
>
> [WITH Internet access]
>     $ step1
>   [Wait for deployment to complete, then run:]
>     $ step2
>
> [WITHOUT Internet access]
>     $ island-step1
>   [Migrate your install node into the isolated environment and run:]
>     $ island-step2
>   [Wait for deployment to complete, then run:]
>     $ island-step3
>
```

After the installation node has successfully bootstrapped you will likely be prompted to reboot the machine. If so, then the machine MUST be rebooted before continuing to Step 4.

5.5 4. Deploying ECS Nodes (`island-step1`)

Once the deploy.yml file has been correctly written and the install node rebooted if needed, then the next step is to simply run `island-step1`.

After the installer initializes, the EMC ECS license agreement will appear on the screen. Press `q` to close the screen and type `yes` to accept the license and continue or `no` to abort the process. The install cannot continue until the license agreement has been accepted.

The first thing the installer will do is create an artifact cache of base operating system packages and the ECS software Docker image. The installer will stop after this step.

5.6 5. Migrate the Install Node

At this time, please shut down the install node VM and migrate it into your isolated environment.

Once the install node has been migrated into your island, you can begin deploying ECS by running `island-step2`. The next tasks the installer will perform are: configuring the ECS nodes, performing a pre-flight check to ensure ECS nodes are viable deployment targets, distributing the artifact cache to ECS nodes, installing necessary packages, and finally deploying the ECS software and init scripts onto ECS nodes.

5.7 6. Deploying ECS Topology (`island-step3`)

If you would prefer to manually configure your ECS topology, you may skip this step entirely.

Once `island-step2` has completed, you may then direct the installer to configure the ECS topology by running `island-step3`. Once `island-step3` has completed, your ECS will be ready for use.

Assuming all went well, you now have a functioning ECS Community Edition instance and you may now proceed with your test efforts.

CHAPTER 6

ECS Administrative Web UI

6.1 Login to the Web UI

The WebUI uses SSL and a self-signed certificate to help protect your session from casual eves-dropping. Take the IP of your first ECS node, fire up your browser, and point `https://` at it. For this example, the latest Google Chrome browser was used.

You cannot add, change, or remove administrative users in this build. Use the default below.

> Username: **root**Password: **ChangeMe**

6.2 Input License

Open *Settings*, then *Licensing* and upload the `license.xml` file located in the ecs-single-node / ecs-multi-node folder. **The UI will not automatically update the license view in this release.** Navigating away from page and returning will prompt it to update. You may need to try a few times before it updates. Once it does, you should see something like this:

6.3 Create Storage vPool

Open *Manage*, then *Storage Pools* and create a storage pool. Keep the name simple, and add all nodes to the pool. Click *Save*.

There's a known issue in this build that causes the Storage Pools view to appear frozen for about 1-2 minutes after provisioning begins. **Unlike with the license view case, this view will update on its own.** Once it's updated, you should see something similar to:

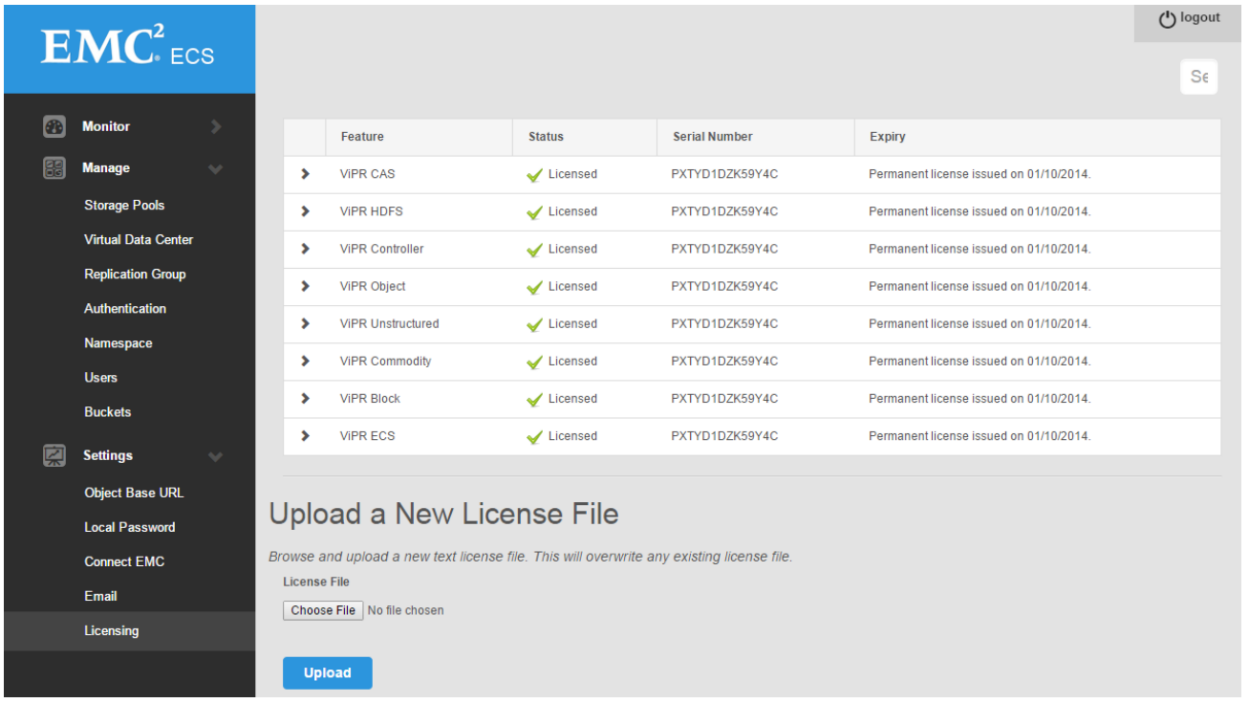


Fig. 1: Upload License file

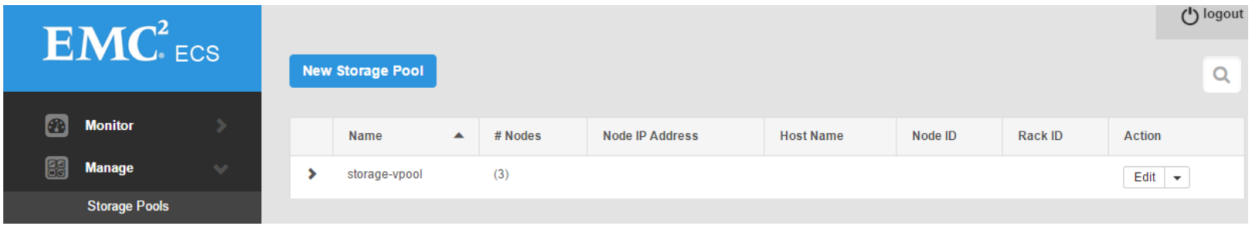


Fig. 2: Create Storage VPool

6.4 Create Virtual Data Center

Open *Manage*, then *Virtual Data Center* and create a Virtual Data Center using the below screenshot as a guide. **Please wait for up to 20 minutes after creating a Storage vPool before creating a Virtual Data Center.** There are several background tasks that must complete, and for object to fully initialize.

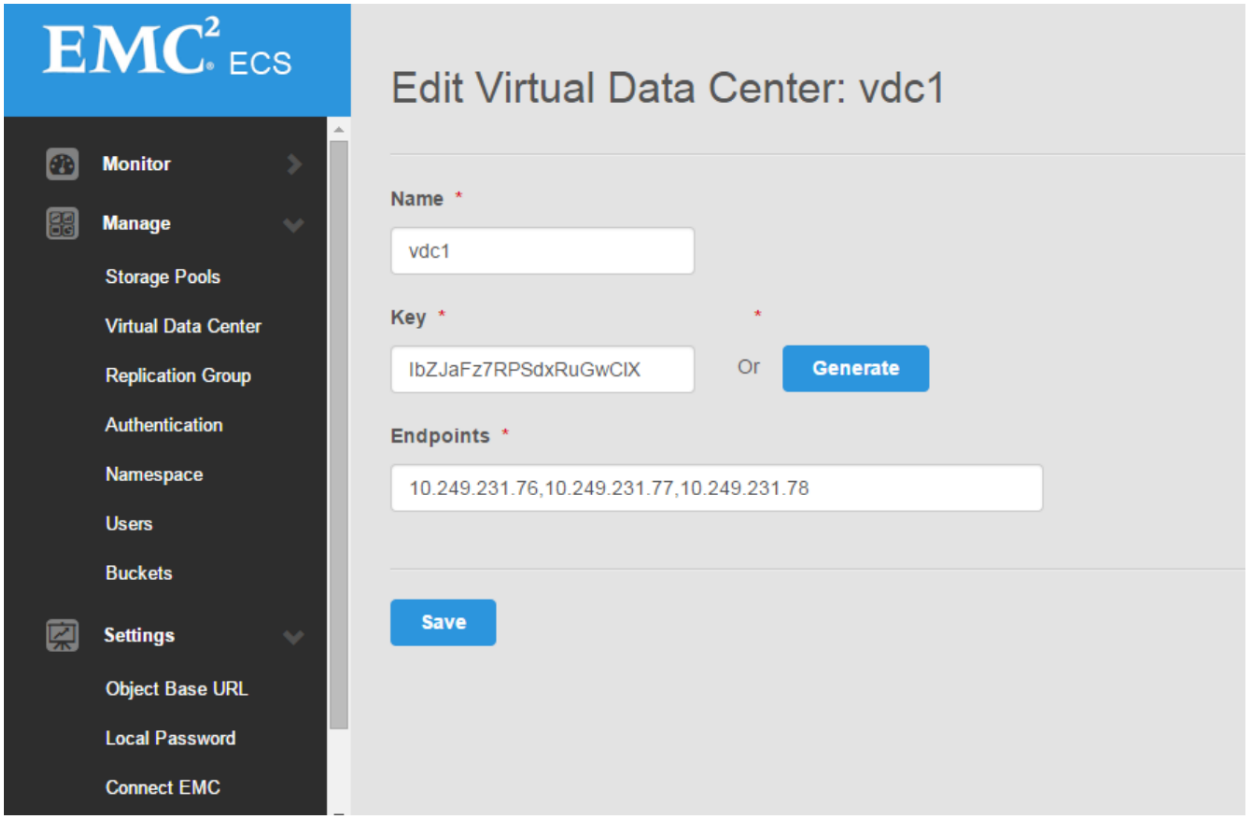


Fig. 3: Create Virtual Data Center

6.5 Create Replication Group

Open *Manage*, then *Replication Group* and create a Replication Group using the below as an example. Currently only one VDC in a replication group is supported.

6.6 Create Namespace

Open *Manage*, then *Namespace*. Set up a Simple Namespace with a name such as "ns". Input a namespace username to use with the namespace, such as "ecs_user". Select the replication group for the namespace, and click *Save* at the very bottom.

Namespace features available in this release

- Simple Namespace
- Retention Policies

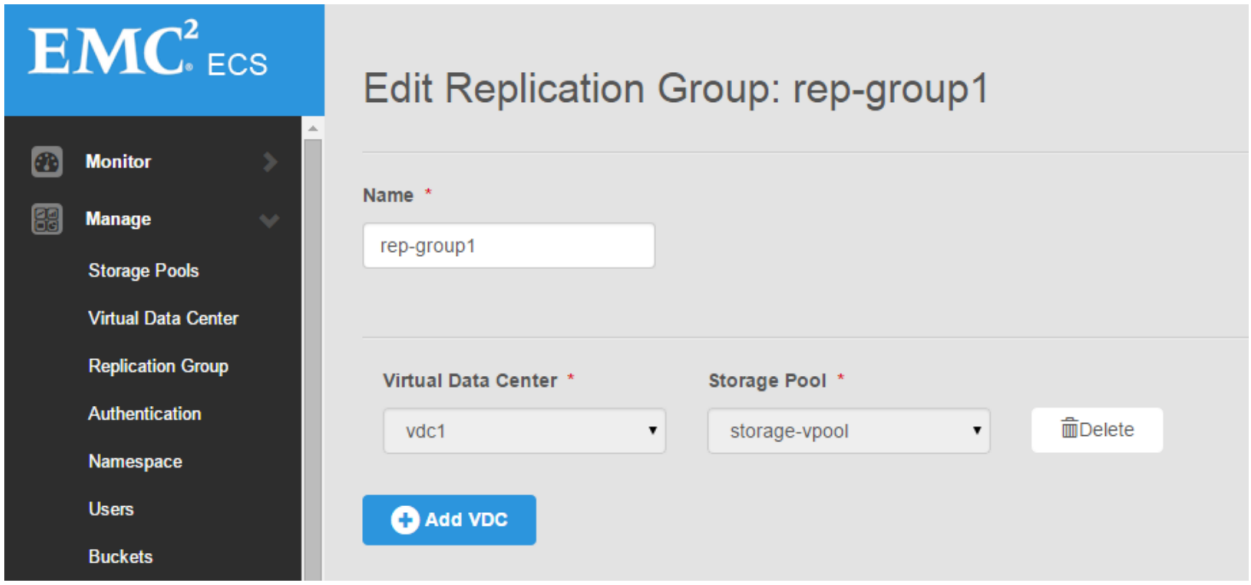


Fig. 4: Create Replication Group

- Quotas
- Authentication Domains

6.7 Create Object User Account

Open *Manage*, then *Users*, then click on *Object Users* and *New Object User* to set up object store credentials.

Create secrets by filling the fields and clicking the buttons.

- S3 Key: Click *Generate & Add Password* to retrieve the server-generated key.
- Swift Password: Enter your own password and click *Set Password*.

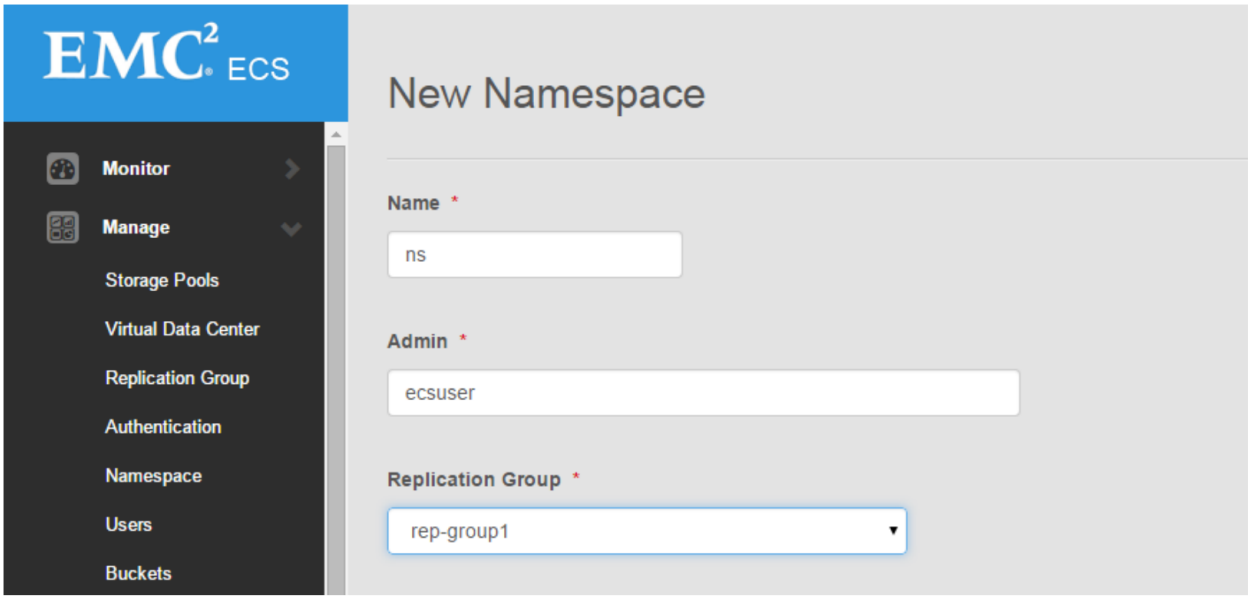


Fig. 5: Create Namespace

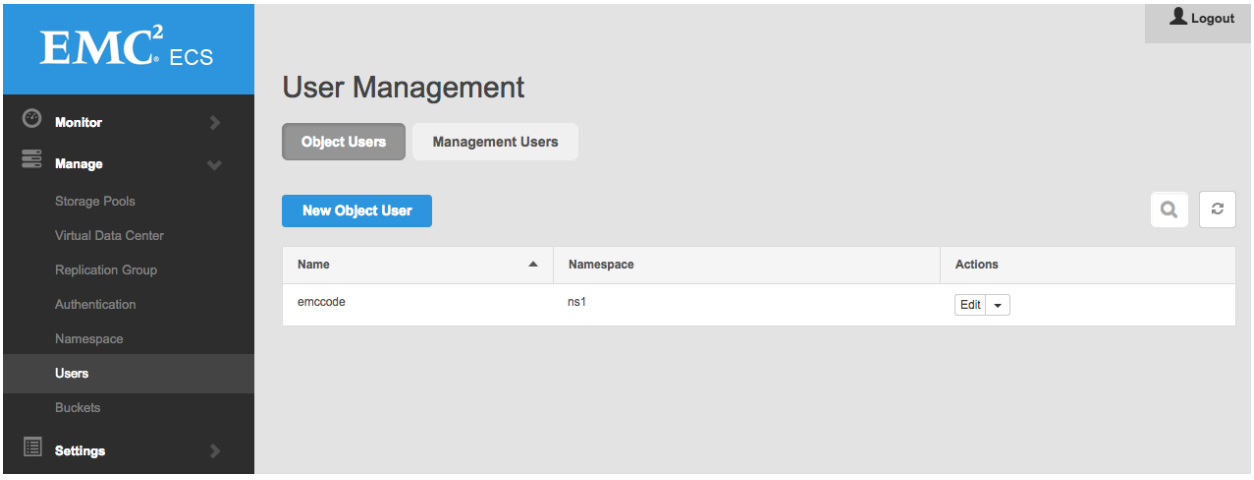


Fig. 6: Create Namespace

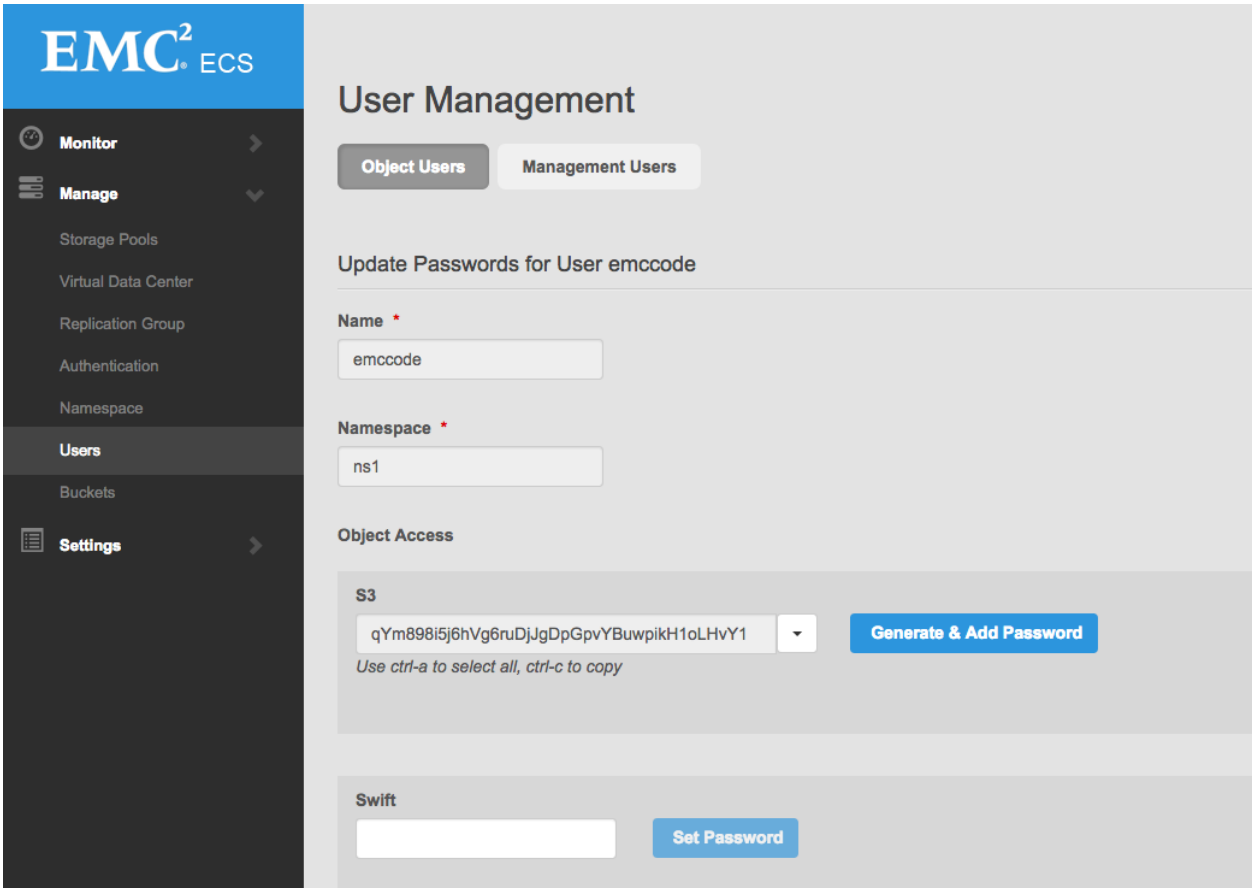


Fig. 7: Create User S3 and Swift Keys

CHAPTER 7

Migration

7.1 General Cases

Most migration cases can be handled by a great tool we wrote called ecs-sync, found here.

7.2 HDFS

An HDFS migration is possible with s3distcp or distcp. Please note that if using s3distcp with the s3a driver, it needs to be the latest version or you may run into issues. If using distcp, ECS's HCFS driver "viprfs" will need to be set up as a secondary FS and the distcp made from `hdfs://...` to `viprfs://...`. Instructions for installing the HCFS driver can be found here.

CHAPTER 8

deploy.yml

The installer works off a configuration file called `deploy.yml` placed in `/opt/emc/ecs-install`.

8.1 deploy.yml Reference Diagram

The following is a visual overview of the deployment configuration file

8.2 deploy.yml Template

The following deploy.yml reference template can be found in `docs/design/reference.deploy.yml` in the ECS-CommunityEdition repository on Github.

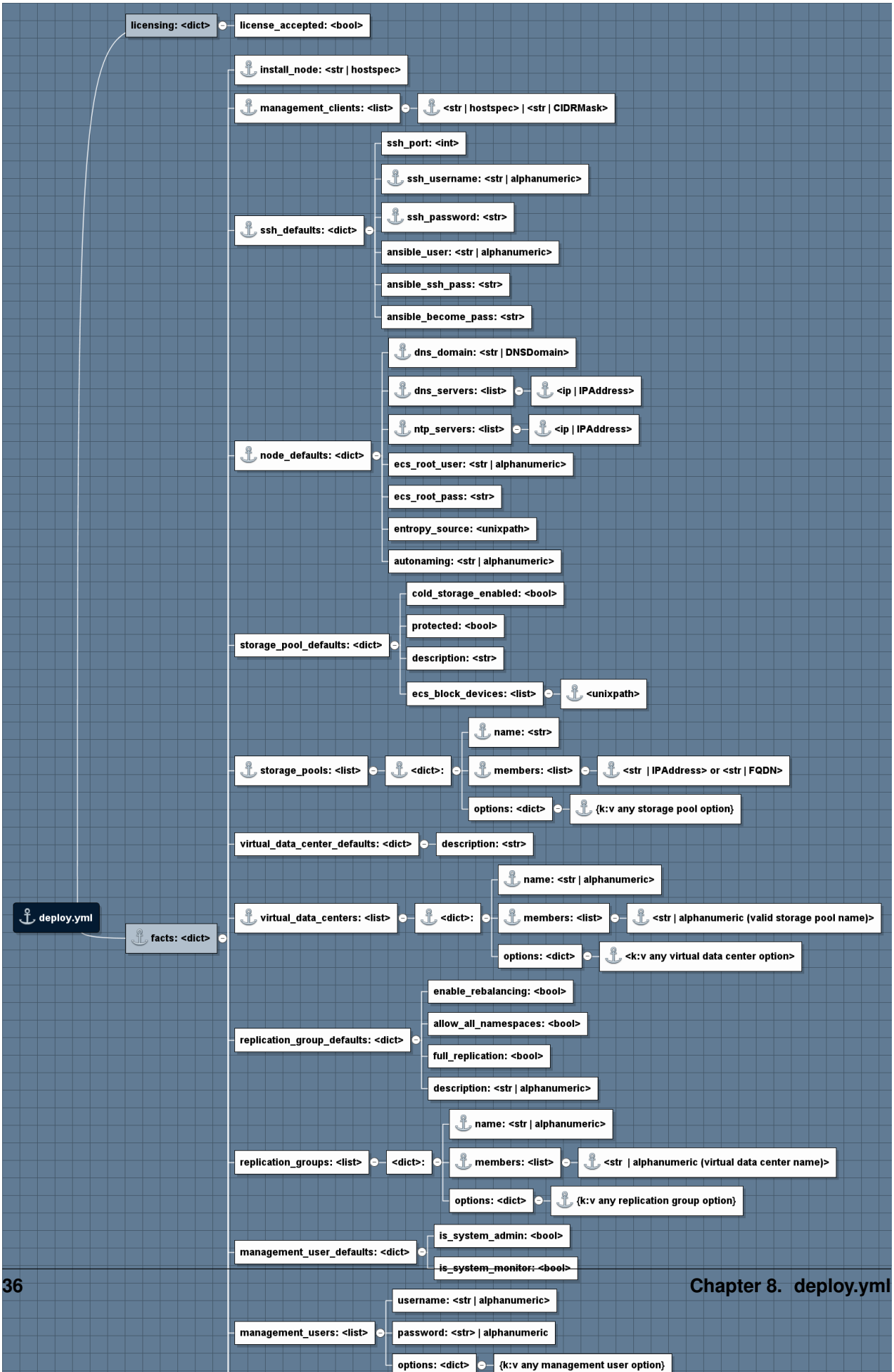
```
# deploy.yml reference implementation

# [Optional]
# By changing the license_accepted boolean value to "true" you are
# declaring your agreement to the terms of the license agreement
# contained in the license.txt file included with this software
# distribution.
licensing:
  license_accepted: false

# [Required]
# Deployment facts reference
facts:

  # [Required]
  # Node IP or resolvable hostname from which installations will be launched
  # The only supported configuration is to install from the same node as the
  # bootstrap.sh script is run.
  # NOTE: if the install node is to be migrated into an island environment,
```

(continues on next page)


deploy.yml
licensing: <dict>
license_accepted: <bool>
install_node: <str | hostspec>
management_clients: <list>
<str | hostspec> | <str | CIDRMask>
ssh_defaults: <dict>
ssh_port: <int>
ssh_username: <str | alphanumeric>
ssh_password: <str>
ansible_user: <str | alphanumeric>
ansible_ssh_pass: <str>
ansible_become_pass: <str>
node_defaults: <dict>
dns_domain: <str | DNSDomain>
dns_servers: <list>
<ip | IPAddress>
ntp_servers: <list>
<ip | IPAddress>
ecs_root_user: <str | alphanumeric>
ecs_root_pass: <str>
entropy_source: <unixpath>
autonaming: <str | alphanumeric>
storage_pool_defaults: <dict>
cold_storage_enabled: <bool>
protected: <bool>
description: <str>
ecs_block_devices: <list>
<unixpath>
storage_pools: <list>
<dict>:
name: <str>
members: <list>
<str | IPAddress> or <str | FQDN>
options: <dict>
{k:v any storage pool option}
facts: <dict>
virtual_data_center_defaults: <dict>
description: <str>
virtual_data_centers: <list>
<dict>:
name: <str | alphanumeric>
members: <list>
<str | alphanumeric (valid storage pool name)>
options: <dict>
<k:v any virtual data center option>
replication_group_defaults: <dict>
enable_rebalancing: <bool>
allow_all_namespaces: <bool>
full_replication: <bool>
description: <str | alphanumeric>
replication_groups: <list>
<dict>:
name: <str | alphanumeric>
members: <list>
<str | alphanumeric (virtual data center name)>
options: <dict>
{k:v any replication group option}
management_user_defaults: <dict>
is_system_admin: <bool>
is_system_monitor: <bool>
management_users: <list>
username: <str | alphanumeric>
password: <str> | alphanumeric
options: <dict>
{k:v any management user option}

(continued from previous page)

```
  #       the hostname or IP address listed here should be the one in the
  #       island environment.
  install_node: 192.168.2.200

  # [Required]
  # IPs of machines that will be whitelisted in the firewall and allowed
  # to access management ports of all nodes. If a member of this list is set to
  # the wildcard mask (0.0.0.0/0) then anyone can access management ports!
  management_clients:
    - 0.0.0.0/0

  # [Required]
  # These credentials must be the same across all nodes. Ansible uses these
  # credentials to gain initial access to each node in the deployment and set
  # up ssh public key authentication. If these are not correct, the deployment
  # will fail.
  ssh_defaults:
    # Username to login as
    ssh_username: admin
    # Password to use with SSH login
    ssh_password: ChangeMe

  # [Required]
  # Environment configuration for this deployment.
  node_defaults:
    dns_domain: local
    dns_servers:
      - 192.168.2.2
    ntp_servers:
      - 192.168.2.2
    #
    # [Optional]
    # VFS path to source of randomness
    # Defaults to /dev/urandom for speed considerations.  If you prefer
    # /dev/random, put that here.
    # If you have a /dev/srandom implementation or special entropy hardware,
    # you may use that too so long as it implements a /dev/random type device
    # abstraction.
    entropy_source: /dev/urandom
    #
    # [Optional]
    # Picklist for node names.
    # Available options:
    # - "moons" (ECS CE default)
    # - "cities" (ECS SKU-flavored)
    autonaming: moons

  # [Optional]
  # Storage pool defaults. Configure to your liking.
  # All block devices that will be consumed by ECS on ALL nodes must be listed
  # under the ecs_block_devices option. This can be overridden by the storage
  # pool configuration. At least ONE (1) block device is REQUIRED for a
  # successful install. More is typically better.
  storage_pool_defaults:
    is_cold_storage_enabled: false
    is_protected: false
    description: Default storage pool description
```

(continues on next page)

(continued from previous page)

```
    ecs_block_devices:
      - /dev/vda

  # [Required]
  # Storage pool layout. You MUST have at least ONE (1) storage pool for a
  # successful install.
  storage_pools:
    - name: sp1
      members:
        - 192.168.2.220
        - 192.168.2.221
        - 192.168.2.222
        - 192.168.2.223
      options:
        is_protected: false
        is_cold_storage_enabled: false
        description: My First SP
        ecs_block_devices:
          - /dev/vda

  # [Optional]
  # VDC defaults. Configure to your liking.
  virtual_data_center_defaults:
    description: Default virtual data center description

  # [Required]
  # Virtual data center layout. You MUST have at least ONE (1) VDC for a
  # successful install. WARNING: Multi-VDC deployments are not yet implemented.
  virtual_data_centers:
    - name: vdc1
      members:
        - sp1
      options:
        description: My First VDC

  # [Optional]
  # Replication group defaults. Configure to your liking.
  replication_group_defaults:
    description: Default replication group description
    enable_rebalancing: true
    allow_all_namespaces: true
    is_full_rep: false

  # [Optional, required for namespaces]
  # Replication group layout. At least one replication_group is required to also
  # provision namespaces.
  replication_groups:
    - name: rg1
      members:
        - vdc1
      options:
        description: My First RG
        enable_rebalancing: true
        allow_all_namespaces: true
        is_full_rep: false

  # [Optional]
```

(continues on next page)

(continued from previous page)

```
  # Namespace defaults.
  namespace_defaults:
    is_stale_allowed: false
    is_compliance_enabled: false

  # [Optional]
  # Namespace layout
  namespaces:
    - name: ns1
      replication_group: rg1
      administrators:
        - root
      options:
        is_stale_allowed: false
        is_compliance_enabled: false
```

CHAPTER 9

ECS Community Edition Utilities

9.1 ecsdeploy

The `ecsdeploy` utility responsible for executing Ansible playbooks and helper scripts responsible for deploying ECS Community Edition to member data nodes.

```
Usage: ecsdeploy [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  Command line interface to ecs-install installer

Options:
  -v, --verbose  Use multiple times for more verbosity
  --help         Show this message and exit.

Commands:
  access         Configure ssh access to nodes
  bootstrap      Install required packages on nodes
  cache          Build package cache
  check          Check data nodes to ensure they are in compliance
  deploy         Deploy ECS to nodes
  disable-cache  Disable datanode package cache handling
  enable-cache   Enable datanode package cache handling
  load           Apply deploy.yml
  reboot         Reboot data nodes that need it
  start          Start the ECS service
  stop           Stop the ECS service
```

9.2 ecsconfig

The `ecsconfig` utility responsible for communicating with the ECS management API and configuring an ECS deployment with administrative and organizational objects.

```
Usage: ecsconfig [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  Command line interface to configure ECS from declarations in deploy.yml

Options:
  -v, --verbose  Use multiple times for more verbosity
  --help         Show this message and exit.

Commands:
  licensing         Work with ECS Licenses
  management-user   Work with ECS Management Users
  namespace         Work with ECS Namespaces
  object-user       Work with ECS Object Users
  ping              Check ECS Management API Endpoint(s)
  rg                Work with ECS Replication Groups
  sp                Work with ECS Storage Pools
  trust             Work with ECS Certificates
  vdc               Work with ECS Virtual Data Centers
```

9.3 ecsremove

The `ecsremove` utility is responsible for removing ECS instances and artifacts from member data nodes and the install node.

```
Usage: ecsremove [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  Command line interface to remove ECS bits

Options:
  -v, --verbose  Use multiple times for more verbosity
  --help         Show this message and exit.

Commands:
  purge-all         Uninstall ECS and purge artifacts from all nodes
  purge-installer   Purge caches from install node
  purge-nodes       Uninstall ECS and purge artifacts from data nodes
```

9.4 enter

This utility has two functions: 1. To access member data nodes by name `enter luna` 2. To access the `ecs-install` image directly and the contents of the data container.

Accessing the `ecs-install` image directly

```
[admin@installer-230 ~]$ enter
installer-230 [/]$
```

Accessing a member node

```
[admin@installer-230 ~]$ enter luna
Warning: Identity file /opt/ssh/id_ed25519 not accessible: No such file or directory.
Warning: Permanently added 'luna,192.168.2.220' (ECDSA) to the list of known hosts.
```

(continues on next page)

(continued from previous page)

```
Last login: Thu Nov  9 16:44:31 2017 from 192.168.2.200
[admin@luna ~]$
```

9.5 catfacts

This utility displays all the facts Ansible has registered about a node in pretty-printed, colorized output from `jq` paged through `less`.

Running `catfacts` without an argument lists queryable nodes.

```
[admin@installer-230 ~]$ catfacts
Usage: $ catfacts <Ansible inventory host>
Here is a list of hosts you can query:
Data Node(s):
  hosts (1):
    192.168.2.220
Install Node:
  hosts (1):
    192.168.2.200
```

Querying a node

```
[admin@installer-230 ~]$ catfacts 192.168.2.200
{
  "ansible_all_ipv4_addresses": [
    "172.17.0.1",
    "192.168.2.200"
  ],
  "ansible_all_ipv6_addresses": [
    "fe80::42:98ff:fe85:2502",
    "fe80::f0c5:a7d1:6fff:205e"
  ],
  "ansible_apparmor": {
    "status": "disabled"
  },
  "ansible_architecture": "x86_64",
  "ansible_bios_date": "04/01/2014",
  "ansible_bios_version": "rel-1.8.2-0-g33fbe13 by qemu-project.org",
  "ansible_cmdline": {
    "BOOT_IMAGE": "/vmlinuz-3.10.0-693.5.2.el7.x86_64",
    "LANG": "en_US.UTF-8",

[... snip ...]
```

9.6 update_deploy

This utility updates the `/opt/emc/ecs-install/deploy.yml` file with the updated contents of the file `deploy.yml` provided during bootstrapping. It can also set the path to the `deploy.yml` file from which to fetch updates.

Running with no arguments

```
[admin@installer-230 ~]$ update_deploy
> Updating /opt/emc/ecs-install/deploy.yml from /home/admin/ecsce-lab-configs/local/
↪local-lab-1-node-1/deploy.yml
37c37
<     ssh_password: ChangeMe
---
>     ssh_password: admin
> Recreating ecs-install data container
ecs-install> Initializing data container, one moment ... OK
ecs-install> Applying deploy.yml
```

Updating the deploy.yml file to a different source.

```
[admin@installer-230 ~]$ update_deploy ~/ecsce-lab-configs/local/local-lab-1-node-2/
↪deploy.yml
> Updating bootstrap.conf to use deploy config from /home/admin/ecsce-lab-configs/
↪local/local-lab-1-node-2/deploy.yml
> Updating /opt/emc/ecs-install/deploy.yml from /home/admin/ecsce-lab-configs/local/
↪local-lab-1-node-2/deploy.yml
37c37
<     ssh_password: admin
---
>     ssh_password: ChangeMe
82c82
<         - 192.168.2.221
---
>         - 192.168.2.220
173a174
>
> Recreating ecs-install data container
ecs-install> Initializing data container, one moment ... OK
ecs-install> Applying deploy.yml
```

9.7 videploy

This utility modifies the `deploy.yml` file currently installed at `/opt/emc/ecs-install/deploy.yml`.

```
[admin@installer-230 ~]$ videploy
```

First, vim runs with the contents of `deploy.yml`, and then `videploy` calls `update_deploy`.

9.8 pingnodes

This utility pings nodes involved in the deployment using Ansible's `ping` module to verify connectivity. It can be used to ping groups or individual nodes.

Ping all data nodes (default)

```
[admin@installer-230 ~]$ pingnodes
192.168.2.220 | SUCCESS => {
    "changed": false,
    "failed": false,
```

(continues on next page)

(continued from previous page)

```
    "ping": "pong"
}
```

Ping all known nodes

```
[admin@installer-230 ~]$ pingnodes all
localhost | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
192.168.2.200 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
192.168.2.220 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
```

Ping the node identified as 192.168.2.220

```
[admin@installer-230 ~]$ pingnodes 192.168.2.220
192.168.2.220 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
```

Ping members of the install_node group

```
[admin@installer-230 ~]$ pingnodes install_node
192.168.2.200 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
```

9.9 inventory

This utility displays the known Ansible inventory and all registered group and host variables.

```
[admin@installer-230 ~]$ inventory
{
  "ecs_install": {
    "hosts": [
      "localhost"
    ],
    "vars": {
      "ansible_become": false,
      "ansible_python_interpreter": "/usr/local/bin/python",
      "ansible_connection": "local"
```

(continues on next page)

(continued from previous page)

```
    }
  },
  "install_node": {
    "hosts": [
      "192.168.2.200"
    ],

[... snip ...]
```

CHAPTER 10

ECS Software 3.x - Troubleshooting Tips

This is a list of troubleshooting tips and nuggets that will help with issues. If you still have problems, please use the support section.

10.1 Installation

10.1.1 If you change deploy.yml after running step1, you must run `update_deploy` before running step1 again. Otherwise you will likely get the following error:

```
{"failed": true, "msg": "An unhandled exception occurred while running the lookup␣
↪plugin 'file'.
 Error was a <class 'ansible.errors.AnsibleFileNotFound'>, original message: the file_
↪name
 '/opt/ssh/id_ed25519.pub' does not exist, or is not readable"}
```

10.1.2 `A block device configured in deploy.yml for data nodes is already partitioned.`

This error often shows up after a failed installation attempt. In order to clean up the block devices to start over run `ecsremove purge-nodes`.

10.2 Provisioning of ECS

It takes roughly 30 minutes to get the system provisioned for Step2. ECS creates Storage Pools, Replication Groups with the attached disks. If Step2 is successful, you should see something along these lines.

10.2.1 Checking Step 2 Object provisioning progress

If you want to see if system is making progress:

1. Log into one of ECS data nodes.
2. Navigate to the **/var/log/vipr/emcvipr-object/** directory
3. View the **/var/log/vipr/emc-viprobject/ssm.log** (tail -f /var/log/vipr/emcvipr-object/ssm.log)

Note: there are ~2k tables to be initialized for the provisioning to complete. You can check the following command to see if the tables are close to that number and if all tables are ready. Run this from the node.

```
curl -X GET "http://<YourIPAddress>:9101/stats/dt/DTInitStat"
```

10.3 ECS Services

10.3.1 Docker Container immediately exits on startup

If your docker instance immediately exits when started, please ensure that the entries in `/etc/hosts` on the host system and `network.json` in the install directory are correct (the latter should reflect the host's public IP and the corresponding network adapter).

10.3.2 ECS web portal will not start

The portal service will listen on ports 443 and 4443; check to make sure no other services (such as virtual hosts or additional instances of ECSCE) are not attempting to utilize these same ports.

For multiple-node installations, the `/etc/hosts` file on the host VM should include entries for each node and their hostname. Additionally, many services including the ECS web portal will not start until all nodes specified to the installation step 1 script have been successfully installed and concurrently running; the installation script should be run on all nodes in a cluster before attempting authentication or use of the GUI.

If attempting to authenticate results in a response of "Connection Refused", review the below section and ensure all necessary ports are open on all ECS nodes in the cluster.

10.4 NFS

10.4.1 Necessary NFS Ports

The following ports must be opened for NFS to function properly

Port Number
111
2049

10.4.2 NFS Volume Refuses to Mount

ECS does support the NFS file system. However, troubles can occur when ECS is installed on the full version, or "Everything" version, of CentOS 7. ***Note that the following solution is not necessary on CentOS 7 Minimal.***

The Problem

CentOS 7 Everything starts with NFS/RPC/Portmap components running in the root scope. This is a problem as the ECS-CE Docker container runs its own version of rpcbind. This is the instance of rpcbind that ECS is intended to communicate with. When CentOS is running rpcbind in root scope in addition to the ECS Docker container, a conflict is created and a NFS volume cannot be mounted.

This can be seen by `# rpcinfo -p` returning no NFS services.

The Solution

The conflict can be resolved by simply running `systemctl disable rpcbind`. This command will shut down the rpc service running on the host OS while leaving the Docker instance untouched.

To confirm the CentOS service is gone, run `rpcinfo -p` in the CentOS shell. This should return an error: `rpcinfo: can't contact portmapper: RPC: Remote system error - No such file or directory`

The same command, `rpcinfo-p`, can be run in the Docker container, which should return something similar to:

```
program vers proto   port  service
 100000    4   tcp    111  portmapper
 100000    3   tcp    111  portmapper
 100000    2   tcp    111  portmapper
 100000    4   udp    111  portmapper
 100000    3   udp    111  portmapper
 100000    2   udp    111  portmapper
 100005    3   tcp   2049  mountd
 100005    3   udp   2049  mountd
 100003    3   tcp   2049  nfs
 100024    1   tcp   2049  status
 100021    4   tcp  10000  nlockmgr
 100021    4   udp  10000  nlockmgr
```

NFS should now function correctly.

10.5 IBM Tivoli Monitoring

10.5.1 Issue

ECS Community edition will fail to completely initialize the storage pool on machines that have the IBM Tivoli Monitoring agent installed. The storage pool will forever stick in the "Initializing" state and attempts to create a VDC will result in HTTP 400 errors.

10.5.2 Analysis

Doing a `ps -ef` inside the container will show that dataheadsvc and metering are restarting frequently. Looking at `/opt/storageos/logs/metering.log` will show a bind exception on port 10110. This port is already bound by Tivoli's `k10agent` process.

10.5.3 Workaround

1. Uninstall Tivoli Monitoring or
2. Change the port on impacted nodes.

Changing the port on ECS

On *all* nodes, you will need to edit `/opt/storageos/conf/mt-var.xml` to change the bind port from 10110 to 10109. Edit the file and change the line:

```
<property name="serviceUrl" value="service:jmx:rmi://127.0.0.1:10110/jndi/rmi://127.0.
↪0.1:10111/sos" />
```

to:

```
<property name="serviceUrl" value="service:jmx:rmi://127.0.0.1:10109/jndi/rmi://127.0.
↪0.1:10111/sos" />
```

Then restart the metering service:

```
kill `pidof metering`
```

10.6 Network Troubleshooting

10.6.1 For those operating behind EMC firewall

To install ECS Community Edition under these conditions, please view the readme file under **/emc-ssl-cert** for further instructions in installing the necessary CA certificate.

10.6.2 Disabling IPv6

ECS Community Edition does not yet support IPv6. The following procedure can be used to disable IPv6 in CentOS 7.

10.6.3 To disable IPv6 on startup:

Add the following to /etc/sysctl.conf

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

10.6.4 To disable IPv6 running:

```
echo 1 > /proc/sys/net/ipv6/conf/all/disable_ipv6
echo 1 > /proc/sys/net/ipv6/conf/default/disable_ipv6
```

or

```
sysctl -w net.ipv6.conf.all.disable_ipv6=1
sysctl -w net.ipv6.conf.default.disable_ipv6=1
```

10.6.5 Get correct interface name

CentOS 7 does not assign network interface names as eth0, eth1, etc, but rather assigns "predictable" names to each interface that generally look like `ens32` or similar. There are many benefits to this that can be read about here.

This can be disabled as documented in the above link, however, these names can otherwise be simply found and used in the ECS-Community installer without issue. To find the names for each device enter the following command: `ip a`. This command will output a list of network devices. Simply find the corresponding device and substitute it for eth0 in the stage1 installation script.

10.6.6 Port Conflicts

It is possible that on multinode installations ECS may run into a port conflict. So far there exists a port conflict with the following:

- ScaleIO - Ports: 9011, 9099

In these instances the user can attempt to:

1. Enter the container
2. Change all instances of the conflicting ports to unused ports in `/opt/storageos/conf`
3. Reboot the nodes after altering the conf file.

10.6.7 List of open ports required on each ECS data node

Ensure the ports in the following table are open for communication. In the case of a multiple-node installation, additionally ensure that each node is trusted to itself and to other nodes in the system by using the following command on each node:

`firewall-cmd --permanent --zone=trusted --add-source=<ECS-node-IP>/32`

followed by `firewall-cmd --reload` for each host.

`fwd_settings.sh` in the main directory will invoke the `firewalld` service and permanently open necessary ports. In the case of a failure in this setup referencing `iptables`, please ensure that your docker network bridge is running and installed using `yum install bridge-utils`.

Port Name-Usage=Port Number
port.ssh=22
port.ecsportal=80
port.rcpbind=111
port.activedir=389
port.ecsportalsvc=443
port.activedirssl=636
port.ssm=1095
port.rm=1096
port.blob=1098
port.provision=1198

Continued on next page

Table 1 – continued from previous page

Port Name-Usage=Port Number
port.objhead=1298
port.nfs=2049
port.zookeeper=2181
port.coordinator=2889
port.cassvc=3218
port.ecsmgmtapi=4443
port.rmmvdcr=5120
port.rmm=5123
port.coordinator=7399
port.coordinatorsvc=7400
port.rmmcmd=7578
port.objcontrolUnsecure=9010
port.objcontrolSecure=9011
port.s3MinUnsecure=9020
port.s3MinSecure=9021
port.atmosMinUnsecure=9022
port.atmosMinSecure=9023
port.swiftMinUnsecure=9024
port.swiftMinSecure=9025
port.apiServerMinUnsecure=9028
port.apiServerMinSecure=9029
port.hdfssvc=9040
port.netserver=9069
port.cm=9091
port.geoCmdMinUnsecure=9094
port.geoCmdMinSecure=9095
port.geoDataMinUnsecure=9096
port.geoDataMinSecure=9097
port.geo=9098
port.ss=9099
port.dtquery=9100
port.dtqueryrecv=9101
port.georeplayer=9111
port.stat=9201
port.statWebServer=9202
port.vnest=9203
port.vnesthb=9204
port.vnestMinUnsecure=9205
port.vnestMinSecure=9206
port.hdfs=9208
port.event=9209
port.objcontrolsvc=9212
port.zkutils=9230
port.cas=9250
port.resource=9888
port.tcpIpcServer=9898

CHAPTER 11

Frequently Asked Questions

11.1 Can I add storage to ECS-CommunityEdition after initializing an installation?

No. Unfortunately because ECS Community Edition lacks the 'fabric' layer present in full featured ECS, it is not possible to add storage space after a completed installation.

11.2 I am locked out of my ECS management console, can I reset the root password?

Currently there is no procedure to reset the root password if you are locked out.

11.3 The storage capacity statistics presented on the ECS dashboard seem wrong, what's going on here?

ECS uses a data boxcarting strategy called "chunking". Chunks are pre-allocated when ECS is first initialized. When user data is written into ECS, pre-allocated chunks are filled and ECS pre-allocates however many new chunks ECS "thinks" would be best for the future based on what it knows about the past.

Capacity statistics are calculated based on allocated and pre-allocated chunks at the time statistics are requested, and don't exactly reflect the actual amount of user data stored within ECS. We do this because it is a performance-enhancing heuristic that is a "good enough" representation of capacities without having to churn through the whole system to figure out the actual user data capacity numbers. In short, the numbers you are seeing are not designed to be exact, but are close estimates.

11.4 Can I use a data store node as an NTP server for the rest of the nodes?

No, this is not a supported deployment option. An external NTP server is required.

11.5 My ECS functions but the storage pools never initialize.

If you can store objects in buckets without issue, then it's likely that your storage pools and data stores are fully initialized. ECS Community Edition is a bit weird in that there are some UI/display issues with storage pools showing "Not Ready" and data stores showing "initializing" even after they have become fully initialized. If you can create VDCs, replication groups, namespaces, and buckets, then your storage pools are certainly initialized as those higher level abstractions require a working storage pool.

ECS Community Edition

See changelog.md file for release notes.

CHAPTER 12

Description

EMC ECS is a stateful containerized cloud storage. It provides persistence for your applications that can access data through standardized Object protocols like AWS S3 or OpenStack Swift. ECS can be set up on one or more hosts / VMs in a single-site or a multi-site geo replicated configuration. We want the wider community to use ECS and provide feedback. Usage of this software is under the following End User License Agreement.

ECS Community Edition is a free, reduced footprint, version of Dell EMC's ECS software. Of course, this means there are some limitations to the use of the software, so the question arises; how is the Community Edition of ECS different from the production version?

12.1 License difference

As noted with the included license, ECS Community cannot be used in production environments and is intended to be used for trial and proof of concept purposes only. This software is still owned and protected by Dell EMC.

12.2 Feature differences

It it important to note that ECS-Community Edition is **not** the same as ECS software and as such lacks some features that are integral to the actual ECS software.

- ECS Community Edition does NOT support encryption.
- ECS Community Edition does NOT include ECS' system management, or "fabric", layer.

12.3 Notice

Because of these differences, ECS Community Edition is absolutely **not** qualified for testing failure scenarios. Failure scenarios can only be adequately mimicked on the full version of ECS Software.

CHAPTER 13

Quick Start Guide

If you have the following:

1. A CentOS 7.4 Minimal instance with:
   (a) 16GB RAM
   (b) 16GB block device for system
   (c) 104GB block device for ECS
2. Internet access
3. No proxies, local mirrors, or special Docker registries

Then you should be able to get up and going with a Single-Node All-in-One install using these commands on your VM:

```
# git clone https://github.com/EMCECS/ECS-CommunityEdition
# cd ECS-CommunityEdition
# cp docs/design/reference.deploy.yml deploy.yml
# echo "Edit this deploy.yml to match your VM's environment"
# vi deploy.yml
# ./bootstrap.sh -y -c deploy.yml
```

And then after the node reboots (you did use a clean minimal install from ISO or netinstall right?):

```
# step1
# step2
```

And if all went well, you now have a working stand-alone ECS, mostly configured, and ready for use.

CHAPTER 14

Hardware Requirements

Hardware or virtual machine with:

- 4 CPU Cores
- 16GB RAM
- 16GB root block storage
- 104GB additional block storage
- CentOS 7.4 Minimal installation

Hardware or virtual machine with:

- 8 CPU Cores
- 64GB RAM
- 16GB root block storage
- 1TB additional block storage
- CentOS 7.4 Minimal installation

CHAPTER 15

Deployment Scenarios

15.1 ECS Multi-Node All-in-One Deployment with Install Node (recommended, full-featured)

Deploy a multi-node ECS instance to two or more hardware or virtual machines and enable all ECS features. Three nodes are required for all ECS 3.0 and above features to be activated.

15.2 ECS Single-Node All-in-One Deployment (smallest footprint)

Deploy a stand-alone instance of a limited set of ECS kit to a single hardware or virtual machine.

15.2.1 Deployments into Soft-Isolated and Air-Gapped Island Environments

Important information regarding Island deployments

Please be aware that install node bootstrapping requires Internet access to the hardware or virtual machine that will become the install node, but once this step is complete, the machine can be removed from the Internet and migrated into the Island environment.

If you prefer to download a prefab install node as an OVF/OVA, follow one of the links below. Please note that OVAs are produced upon each release and do not necessarily have the most current software.

Please see the release page for OVA download links.

15.3 ECS Multi-Node Deployment with Install Node (recommended, most reusable, full-featured)

Using an install node for isolated environments, deploy a multi-node ECS instance to two or more hardware or virtual machines and enable all ECS features. Three nodes are required for all ECS 3.0 and above features to be activated.

15.4 ECS Single-Node Deployment with Install Node

Using an install node for isolated environments, deploy a stand-alone instance of a limited set of ECS kit to a single hardware or virtual machine.