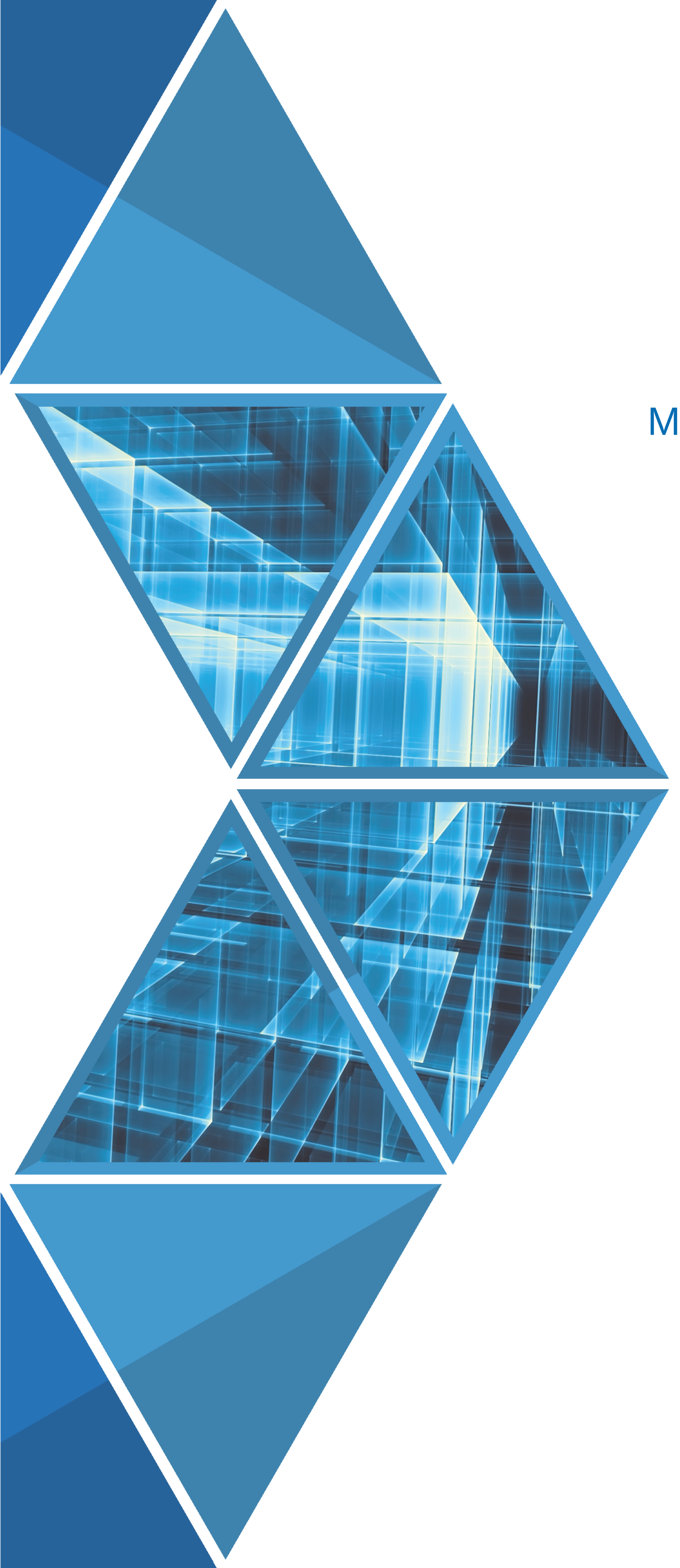


MemVerge

Memory Machine™ Cloud

User Guide

First Published: 2023-01-27

Last Revised: 2023-07-18

Contents

Introduction

What is Memory Machine™ Cloud?

Memory Machine Cloud (MMCloud) is a software platform that streamlines the way containerized applications are deployed in the cloud.

Based on user-supplied policy, MMCloud selects and instantiates cloud resources on behalf of the user. MMCloud has a built-in job scheduler so users can deploy Docker containers (and other containers that comply with the Open Container Initiative *image-spec*) across a group of virtual machines.

MMCloud includes AppCapsule, MemVerge's checkpoint/restore (C/R) capability. The AppCapsule is a moment-in-time snapshot of the application instance, including in-memory state and relevant files. AppCapsule is used to support workload mobility and workload continuity. Workload mobility means that a job can move from one virtual machine to another, for example, to a more powerful virtual machine that is a better fit for the next stage of execution. Workload mobility also provides high availability — if the underlying virtual machine is reclaimed, the workload automatically moves to a new virtual machine and resumes running.

Most of the time, hyperscale Cloud Service Providers (CSPs) have excess virtual machine capacity, which they offer as Spot Instances at varying discounts — the average discount is around 80%. The trade-off is that any Spot Instance can be reclaimed with only nominal warning (typically, two minutes or less). MMCloud’s AppCapsule feature is triggered automatically when the CSP signals that it is reclaiming the Spot Instance. Job execution pauses and then resumes on a new Spot Instance, allowing users to take advantage of the reduced cost of Spot Instances without the risk of losing intermediate results before the job has run to completion.

About This Guide

MMCloud OpCenter

MMCloud relies on the OpCenter, a server that you deploy on a virtual machine in a Cloud Service Provider (CSP) network. This guide provides instructions on how to install the OpCenter and how to use it to run containerized applications in the CSP's network.

Prerequisites

To install and run MMCloud, you need an account with a CSP (such as AWS) and a working knowledge of the basic services the CSP offers.

Most, but not all, jobs require a job script — usually in the form of a shell script — that you provide.

Supported Clouds

MMCloud is designed to work on any cloud infrastructure. The El Nido 2.2.1 Release supports AWS and Alibaba Cloud. Support for Google Cloud is considered experimental in the El Nido release.

How This Guide is Organized

The organization is as follows:

- Background *(on page 6)*

  Provides an overview of the MMCloud architecture, cloud computing, containers, and job schedulers.
- Getting Started *(on page 16)*

  Describes how to install the OpCenter and how to run a workload using the web interface or the CLI (called "float")
- Working with OpCenter *(on page 28)*

  Highlights major features and describes common usage scenarios.
- Working with Applications *(on page 55)*

  Details the configuration tasks required to support several popular applications to work with MMCloud.
- CLI Command Reference *(on page 89)*

  Presents detailed information on commands available through the MMCloud CLI.

Background

Introduction

Operating MMCloud entails interacting with cloud services and third party software, for example, software related to containers and workflow management systems. This part of the MMCloud User Guide provides background information on these topics. It is not an in-depth tutorial or reference; rather, this section highlights topics that you should be familiar with to use MMCloud.

Users who have a working knowledge of these topics can skip this part and proceed to Getting Started *(on page 16)*.

MMCloud Architecture

MMCloud has several components that interact with cloud services to ensure mobility for submitted jobs so that, for example, jobs run to completion on Spot Instances.

Design

MMCloud enables users to run containerized applications on virtual machines leased from Cloud Service Providers. The virtual machines are usually Spot Instances but they can also be On-demand Instances (this is a per-job configuration option). All functions are controlled by the MMCloud Operations Center (OpCenter), which receives CLI commands from users (clients) and manages resources in the cloud, as shown in Figure 1: MMCloud Architecture *(on page 7)*.

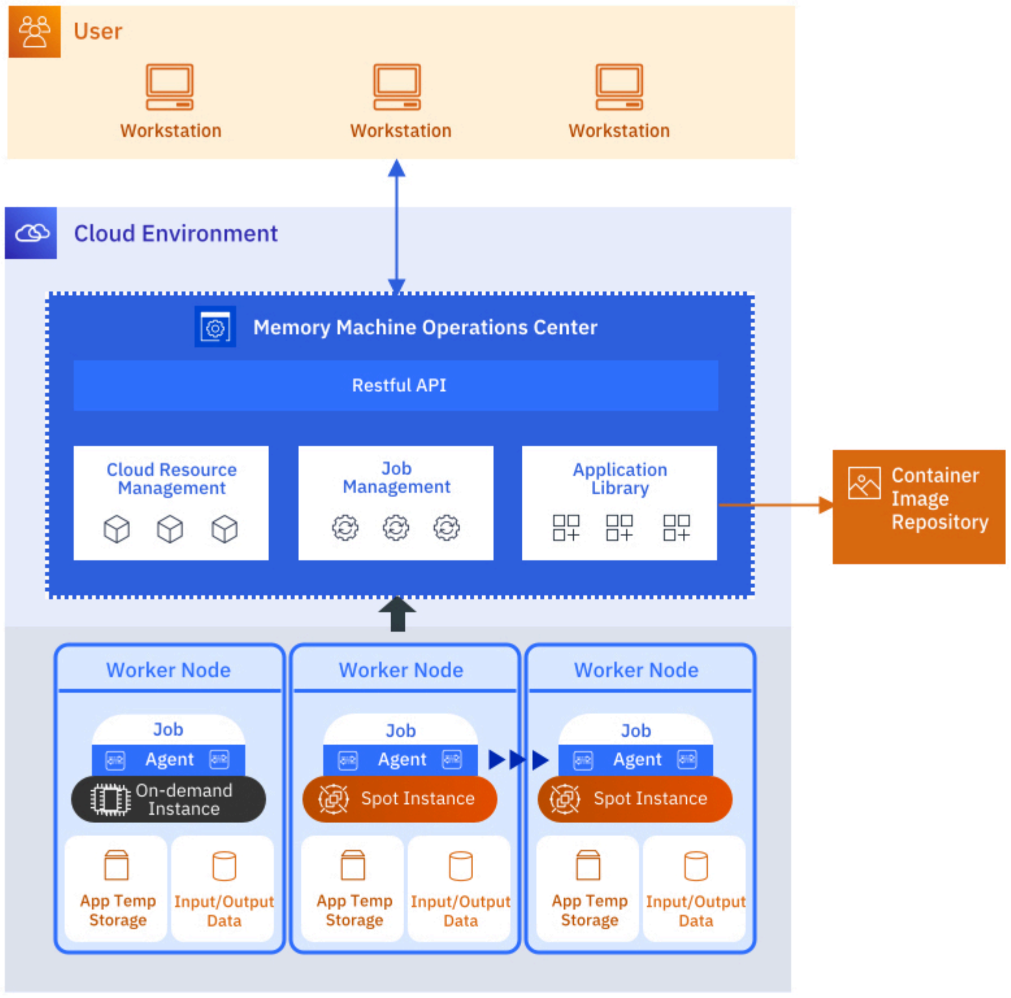


Figure 1. MMCloud Architecture

Components

The MMCloud architecture includes the following components.

- Users (Clients)

  Using the MMCloud web interface or the MMCloud CLI, clients interact with the MMCloud OpCenter
- MMCloud OpCenter

  This provides the core functionality that allows MMCloud to marshal resources for starting workloads and to migrate workloads if needed. If the OpCenter is not running, currently executing jobs continue (but are not migrated) and new jobs are not scheduled.
- Application Library

  A container image registry is a service for hosting and distributing images. The default registry for Docker images is Docker Hub. A repository is a collection of images within a registry (one registry hosts many repositories). A private repository requires a username and access token to post or retrieve images; a public repository does not. The Application Library contains a database of information for accessing container images in various repositories (public and private).
- Worker Nodes

  These are the compute engines provided by virtual machines running in the Cloud Service Provider's network. Worker nodes may have locally-mounted file systems and attached storage. On-demand Instances or Spot Instances can be used as worker nodes.

Workflow

The operation of MMCloud proceeds as follows. A client submits a job to the OpCenter, using CLI command options to select a container image from the Application Library and to specify the compute resources needed (the web interface can generate the CLI command line for you). The OpCenter uses this information to orchestrate the necessary resources in the Cloud Service Provider's network and schedules the job for execution. The cloud resources always include a compute node and may include block storage and file systems as well. One or more data sets usually accompany a job. The user-provided job script describes how these data sets are accessed, for example, by loading data from an AWS S3 bucket. The job script also describes where the output is placed — results are usually written to a persistent file system. When the job has run to completion, the user retrieves the results.

Workload Continuity

Using the AppCapsule feature, OpCenter automatically moves a job running on a Spot Instance to a new Spot Instance if the first Spot Instance is reclaimed, as illustrated in Figure 2: MMCloud Operation *(on page 9)*.

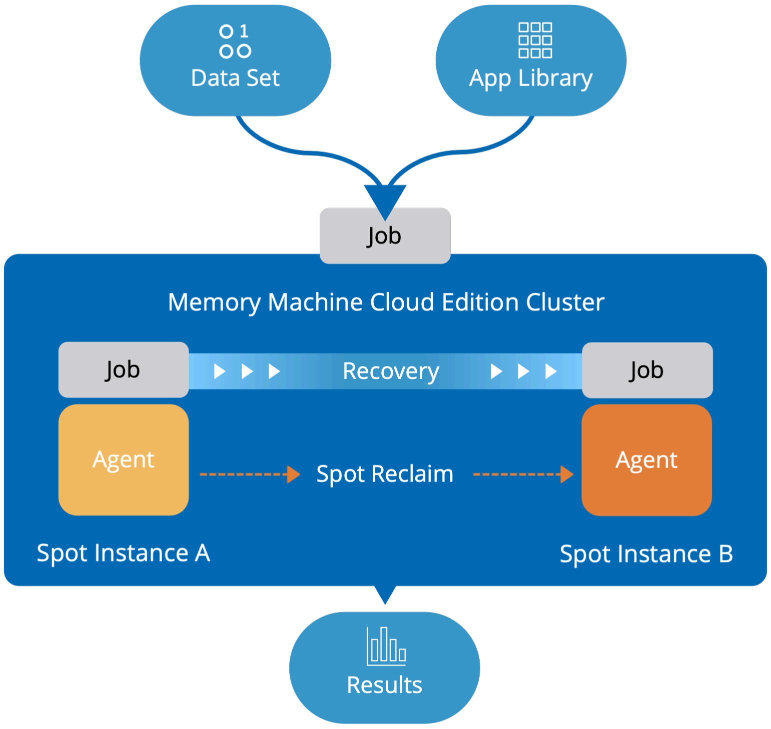


Figure 2. MMCloud Operation

In the example shown, the job starts executing on Spot Instance A. If the Cloud Service Provider signals that it intends to reclaim Spot Instance A, the OpCenter triggers the AppCapsule feature to capture the state of the running job and export the checkpoint image to persistent storage. A new Spot Instance is started (Spot Instance B in this case), the checkpoint image is imported from persistent storage, and the job resumes execution.

The job continues to run in this manner until completion, at which point the user retrieves the final results.

Workload Mobility

For jobs running on Spot Instances, job migration occurs automatically if the Spot Instance is reclaimed. The new Spot Instance is usually of the same type as the one that was reclaimed although the user can set a policy so that the new instance is different, for example, an On-demand Instance or a Spot Instance of a different size.

Using the CLI, you can manually migrate a job from one virtual machine to another, for example, from a Spot Instance to an On-demand Instance of a different type. Using CLI command options, you can specify the new On-demand Instance by the instance type (for example, c6xi.large in AWS) or by specifying ranges of memory size and number of virtual CPUs.

Workload mobility is useful for rightsizing compute platforms: jobs can pass through several execution stages where the compute requirements are different; for example, one stage might be memory-intensive whereas another stage might be compute-intensive. Workload mobility moves the job from one compute platform to another as the resource demands change.

Job migration can be initiated in three ways:

- Manually, using CLI commands or the OpCenter web interface.
- Automatically, using a rules-based policy driven by resource utilization, for example, CPU or memory utilization thresholds.
- Programatically, by inserting **`float migrate`** commands at breakpoints specified in the job script, for example, after loading data.

Licensing

MemVerge maintains an account server where you obtain licenses for MemVerge products. Subscribers to MMCloud choose from the following license plans.

- MMCloud Essentials: a free, perpetual license that allows licensees to launch jobs using OpCenter but does not enable advanced features.
- MMCloud Pro: a license that enables all features. Accrued charges are usage-based.
- MMCloud Enterprise: a license that enables all features. Charges are not usage-based — charges are based on a contract between MemVerge and the account holder.

When you log in to the OpCenter the first time, you click on a button to apply the license to your OpCenter instance.

Cloud Service Providers

To use MMCloud, you need an account with a Cloud Service Provider.

Introduction

Each of the global Cloud Service Providers (CSPs) offers hundreds of cloud-based services, ranging from general purpose computing and database systems to leading-edge capabilities such as Function-as-a-Service (serverless computing). The basic services are similar across CSPs although the terminology varies. To run containerized applications in the cloud, MMCloud uses several of these services. A working knowledge of the version of these services provided by your CSP is required to deploy MMCloud.

CSP Services Used by MMCloud

The following CSP services are used by MMCloud:

- Virtual Machines

  CSPs group virtual machines into classes, generally based on the type of processor used. Within a class, virtual machines are distinguished by the number of virtual CPUs and the amount of memory. Each virtual machine type has a name, for example, c6a.4xlarge in AWS.
- Virtual Private Cloud (VPC)

  The VPC is a network of cloud resources that act like a private data center belonging to one customer.
- Availability Zones and Regions

  To maximize the availability of cloud resources, CSPs distribute their data centers into geographically diverse regions. Within a single region, resources are further separated into zones — bounded by latency — so that if a resource fails in one zone, it can be replaced by an identical resource in another zone in the same region.
- Identity and Access Management (IAM)

  IAM is a service that allows you to control access to resources and authorization to perform certain tasks.
- Persistent file system

  Access to a file system is required to load the virtual machine's operating system and is useful for storing data and results. The file system persists after the virtual machine is deleted.
- Firewall

  Virtual firewalls control network access to resources by imposing inbound and outbound filtering rules. For some applications, such as RStudio, an inbound rule must be applied to allow access to the container running RStudio on a virtual machine.
- Management Console

  To deploy and operate MMCloud, users must configure and view resources in the CSP's network. The interface that acts as the entry point for users is the Management Console. From there, users can navigate to other service-specific consoles and dashboards.

Terminology used by CSPs

Table 1. Mapping of Services across CSPs

Service	AWS	Azure	Google Cloud Platform	AliCloud
Virtual Machine	Elastic Compute Cloud (EC2)	Virtual Machine	Compute Engine	Elastic Compute Service (ECS)
Object Storage	Simple Storage Service (S3)	Azure Blob Storage	Cloud Storage	Object Storage Service (OSS)
Block Storage	Elastic Block Store (EBS)	Azure Disk Storage	Persistent Disk	Elastic Block Storage
Virtual Private Cloud	Amazon Virtual Private Cloud	Azure Virtual Network	Virtual Private Cloud	Virtual Private Cloud
Regions and Availability Zones	Regions and Availability Zones	Regions and Zones	Azure Availability Zones and Regions	Regions and Zones
IAM	Amazon IAM	Azure Identity and Access Management	IAM	IDaaS
Firewall	Security Groups	Azure Firewall	Google Cloud Firewalls	Cloud Firewall
CLI	AWS CLI	Azure CLI	Cloud SDK	Alibaba Cloud CLI
Marketplace	AWS Marketplace	Azure Marketplace	Marketplace	Marketplace
Cloud Deployment	AWS CloudFormation	Azure Deployment Manager	Cloud Deployment Manager	Compute Nest
Management Portal	AWS Management Console	Azure Portal	Google Cloud Console	Alibaba Cloud Management Console

Using CSP Services with MMCloud

OpCenter runs on a virtual machine inside a VPC in the CSP's network. From there, OpCenter selects, configures, and deploys the resources required to run the jobs submitted to OpCenter. In some cases, the user must perform additional configuration tasks manually. Although the procedures are different for each CSP, the procedures always begin by requiring the user to log in to the CSP's Management Console.

Most OpCenter maintenance procedures can be performed by using `float` commands, but the procedures for instantiating or deleting OpCenter must be carried out using the CSP's Management Console.

Containers

Containers provide a standardized and repeatable way to package, deploy, and manage applications in the cloud.

Introduction

A container image is a template for creating a container, i.e., a container is the runnable instance of a container image. Applications submitted to Memory Machine Cloud must be packaged in images compliant with the Open Container Initiative (OCI) image format. An example of an OCI-compliant container image is Docker.

Docker

The company Docker was an early developer of Linux container technology and the Docker platform is often considered the de-facto standard. Docker uses a client-server architecture in which the client uses **`docker`** commands to interact with the docker daemon (**`dockerd`**). Docker is a complete container environment — images are built, and containers are deployed and managed using the **`docker`** CLI.

Container Image Registry

A container repository is a collection of container images. A registry is a collection of repositories and also a service that allows users to store, access, and share images in specific repositories. A repository may be public in which case all images in the repository are available to all users. A private repository restricts access so that only users who have a valid username and access token may access the repository.

MemVerge maintains a private repository at Docker Hub to store images that are automatically loaded by MMCloud.

Singularity

Singularity is an alternative to Docker. Singularity is commonly used in HPC (high-performance computing) environments whereas Docker is widely used in VM and cloud deployments. Although similar in many ways, differing design goals mean significant differences. Docker requires a separate daemon process that runs with root privileges whereas Singularity runs with user privileges and does not require a separate daemon. Docker stresses isolation from the host system, for example, processes running inside the Docker container do not have automatic access to the host file system. Singularity stresses integration, for example, some host file systems are automatically mounted inside the container.

Although it has its own container format called Singularity Image Format (SIF), Singularity can pull images from Docker Hub and convert them to SIF format on the fly. It is possible to convert Singularity images to Docker images but the conversion usually happens in the other direction.

Kubernetes

A container is a convenient way to package an application along with its dependencies so that the resulting image can be used to start multiple application instances with reproducible results. Deploying large numbers of containers requires a container orchestration and management platform. Originally developed by Google, Kubernetes, also known as k8s, is the most popular of these platforms. Red Hat provides a supported version called OpenShift.

The smallest execution unit in Kubernetes is the pod. A pod includes one or more containers and runs on a worker node. Kubernetes includes workload resources called controllers that create and manage pods in a cluster. For example, the controller can replicate pods to scale up or delete pods to scale down. Controllers can also ensure that a minimum number of pods is running at all times.

MMCloud is not a replacement for Kubernetes. Both platforms start and run Docker containers. Where Kubernetes manages and orchestrates containers in terms of pods running on worker nodes deployed in clusters, MMCloud deploys individual containers and manages at the job level.

Using Containers with MMCloud

If you use existing Docker images (accessible from a public or private repository) to submit workloads to MMCloud, you can complete all the necessary tasks using the **`float`** CLI. To build your own custom image and upload it to a repository, you can use **`docker`** or another container management tool such as **`podman`**.

If you build a container image on your local server, using **`docker`** or **`buildah`**, for example, you can bypass the image registry and upload the image directly to the OpCenter using **`float`** commands.

Job Schedulers and Workflow Management Systems

MMCloud includes a job scheduler for running batch jobs in the cloud.

Introduction

Workload managers, also known as job schedulers, are used to schedule, execute, and monitor batch jobs on clusters of compute nodes. Contention for resources is resolved by maintaining a queue of pending work. *Workflow* management systems are used to manage data-analytic pipelines, i.e., sequences of computational tasks such as found in genomic analyses.

Examples of job schedulers are:

- Slurm
- IBM Spectrum LSF ("load sharing facility")
- AWS Batch

Examples of workflow management systems are:

- Cromwell
- Nextflow

Job Scripts

A job script is a text file that has job setup information for the workload manager followed by commands to execute. A job script must start with a shebang (**`#!/bin/bash`** is commonly used but you can use others). A job script may be as simple as a few lines of shell script but is often more complicated.

Most jobs submitted to OpCenter include a job script, but it is possible to submit a job without a job script as long as the container image has an "entrypoint" (in Docker terminology).

Getting Started

Summary

To deploy MMCloud, complete the following two steps (the order is not significant).

- Obtain the license that enables you to use OpCenter.

  To request a license, log in to the MemVerge account server (register first if you don't have an account set up). Select the type of license and submit the form that is displayed.
- Create and start an OpCenter instance.

  Each CSP has a procedure for provisioning a collection of resources as a single group. MemVerge provides a template that describes the resources you need. Use this template with the CSP-specific service that assembles and instantiates these resources. With AWS, for example, use the AWS CloudFormation service.

  Once the OpCenter is running, the software can be upgraded without needing to re-provision CSP resources.

Activate the OpCenter license as follows.

- Open a browser and go to the IP address associated with your OpCenter. Use the public (private) IP address if you are connecting from outside (inside) the VPC.
- Log in to the OpCenter web interface with the username admin and default password memverge. You can change the admin password after you have logged in.
- On the top, right-hand side of any screen (except the *Submit Job* screen), click on the license icon. Follow the instructions in the pop-up screen to activate your license.

All interactions with the OpCenter use the MMCloud CLI or the OpCenter web interface.

There are three ways to access the CLI:

- Use the CLI shell provided by the OpCenter web interface.
- Download the CLI binary and run it in a terminal window on your local machine.
- Log in to the OpCenter server and run the CLI from a secure terminal session (not recommended).

Open the OpCenter web interface by using a browser to go to the IP address associated with your OpCenter. Use the public (private) IP address if you are connecting from outside (inside) the VPC.

The remainder of this document provides detailed instructions for each of these steps. The tasks have steps that are CSP-specific. Look for the section that applies to your CSP.

Obtaining an OpCenter License

This procedure applies to all Cloud Service Providers.

Step 1 Open a browser and go to the MemVerge account server; enter your credentials at the sign-in screen.

> **Note:** If you are not registered, click on the *Sign Up* link and follow the instructions.

Step 2 Select your preferred MMCloud license plan — the choices are *Essentials* (free), *Pro*, or *Enterprise*.

> **Note:** If you are not ready to choose a license, scroll down the page to locate the link to the live demo environment where you can try out a limited set of MMCloud features. Your credentials are provided under the link.

Step 3 Fill in the pop-up form and click on *Confirm*.

Step 4 Follow the instructions to confirm your license selection.

Deploying OpCenter in AWS

Before You Begin

- Contact your AWS administrator to obtain an AWS account with root privileges.
- You need to select a VPC subnet and enter information on CIDR blocks. If you don't have this information, contact your AWS administrator.

Step 1 Log in to your *AWS Management console*

Step 2 Create a key pair

Step 2.1 From the navigation bar, click on *Services* and select *EC2*.

Step 2.2 On the right-hand side of the navigation bar of the *EC2 Dashboard*, check what region you are in.

If you need to change regions, use the drop-down menu to select a new region.

Step 2.3 From the left-hand panel of the *EC2 Dashboard*, go to *Network & Security*, and click on *Key Pairs*.

The *Key Pairs* panel shows the key pairs that are available. If you need to create a key pair, click on the *Create key pair* button on the right-hand side and follow the instructions. A copy of the key pair file is automatically downloaded to your local machine.

> **Note:** The key pair is unique to a region. If you change regions, you must create a key pair for the new region.

Step 3 Identify the Availability Zone in which each VPC subnet is located

Step 3.1 Using the search tool in the navigation bar, find the *VPC dashboard*.

Step 3.2 On the left-hand panel of the *VPC dashboard*, click on *Subnets*.

The *Subnets* console displays a table that shows each VPC subnet, the VPC the subnet is part of, and the Availability Zone in which the subnet resides.

Step 3.3 Keep this browser tab open.

You need to retrieve information from this table later in the deployment procedure.

Step 4 Locate Memory Machine Cloud in the AWS Marketplace

Step 4.1 Log in to the *AWS Management console*.

Step 4.2 Go the AWS Marketplace and select *Discover products* from the left-hand panel.

Step 4.3 Search for Memory Machine Cloud.

Step 4.4 Click on the result and then proceed to the next step.

Step 5 Subscribe to Memory Machine Cloud

Step 5.1 Click on *Continue to Subscribe*.

Step 5.2 If you agree to the terms and conditions, click on *Accept Terms*, then click on *Continue to Configuration*.

Step 5.3 At the *Configure this software* screen, use the pull-down menu to select MMCE Topology as the fulfillment option. From the Region pull-down menu, select the region in which to deploy OpCenter.

Step 5.4 Click on *Continue to Launch*.

Step 5.5 Under *Choose Action*, use the pull-down menu to select *Launch CloudFormation*.

Step 5.6 Click on *Launch*.

You are now at the *CloudFormation* > *Stacks* > *Create stack* screen. The left-hand panel shows the four-step procedure for creating a new stack. You are at the first step.

Step 5.7 Click on *Next*.

Step 6 Specify the *CloudFormation* stack details

Step 6.1 Enter a unique stack name, using the allowed characters.

Step 6.2 Fill in the required parameters as follows (seek guidance from your AWS administrator if needed).

- *0mvOpCenterType*: Accept the default or use the pull-down menu to change the size of the VM to run OpCenter.
- *1mvFloatKeyName*: Use the pull-down menu to select key pair name you created in step 2 *(on page 18)*.
- *2mvFloatVPCID*: Select a VPC from the pull-down menu. (Usually, there is only one VPC; if there are multiple, consult your AWS administrator for guidance.)
- *3mvFloatSubnetID*: Select a VPC subnet from the pull-down menu. (Usually, VPC subnets are equivalent; consult your AWS administrator for guidance.)
- *4mvAvailabilityZone*: Go to the browser tab you opened in step 3 *(on page 18)* to determine the Availability Zone in which your VPC subnet resides. Then return here to use the pull-down menu to select the appropriate Availability Zone.
- *5mvFloatPublicCIDR*: Provide the range of IP addresses allowed to access server. To allow access from any address, enter `0.0.0.0/0`

- *6mvFloatSSHCIDR*: Provide the range of IP addresses allowed to access server using **`ssh`**. To allow **`ssh`** access from any address, enter **`0.0.0.0/0`**
- *7mvFloatInternalCIDR*: Provide the range of IP addresses for internal communication among OpCenter instances. To allow communication from any address, enter **`0.0.0.0/0`**

Step 6.3 Click on *Next*.

Step 7 Configure stack options

Keep the default options and click on *Next*.

Step 8 Review

Step 8.1 Check the box at the bottom of the page to acknowledge that you are aware *CloudFormation* may create IAM resources.

Step 8.2 Click on *Create stack* to proceed.

Wait until the process completes successfully.

Step 9 Check your stack

Step 9.1 Go to the *CloudFormation* > *Stacks* screen and select your stack.

Step 9.2 From the *CloudFormation* > *Stacks* > *"your_stack_name"* screen, select the *Resources* tab.

Step 9.3 Click on *Physical ID* associated with *mvOpCenter*.

This takes you to the *Instances* screen of the EC2 console.

Step 9.4 Select the OpCenter instance.

This screen displays OpCenter status as well as private and public IP addresses associated with the OpCenter.

Step 9.5 If you are outside (inside) your organization's virtual private cloud, open a browser and go to the OpCenter public (private) IP address, respectively.

If the OpCenter is running, the OpCenter web interface displays the version of OpCenter software installed.

Using the OpCenter Web Interface

OpCenter provides a web interface to activate the license and to submit and manage jobs.

Accessing the OpCenter Web Interface

- Open a browser and go to the IP address (public if you are outside the VPC, private if you are inside the VPC) associated with your OpCenter. The landing page allows you to do the following:
    - See the version of software installed
    - Log in to OpCenter
    - Download the CLI tool for your local operating system (Windows, Linux or macOS)
- Log in to the OpCenter by entering your username and password, and then clicking on the *Log In* button.

  The first time you log in, enter the username admin and default password memverge. After you log in, you can change the admin password.

Navigating the OpCenter Web Interface

The left-hand panel displays the available screens. The following screens are available.

- Submit Job
- Cost Summary (only visible if you log in as *admin*)
- Jobs
- App Library

At the top, right-hand side are a series of icons that allow you to perform the following actions.

- Open the web CLI shell
- Manage the OpCenter license
- Access documentation at docs.memverge.com
- Log out of the OpCenter

Applying the OpCenter License

- From any screen (except the *Submit Job* screen), click on the license icon (top, right-hand side).
- On the pop-up form, enter your login credentials for *account.memverge.com* and click on *Confirm*. If the license is successfully applied, the pop-up display shows the license ID and issue date.

Note: You can log in to a different account at account.memverge.com by clicking on the license icon and then clicking on the *Change* button in the pop-up screen.

Submit Job Screen

On the left-hand side of the *Submit Job* screen, you can fill in the fields in a form and then submit a job. The right-hand side shows the equivalent CLI commands. After completing the form, submit the job by clicking on the *Submit* button at the bottom, on the right.

Cost Summary Screen

The *Cost Summary* screen summarizes your cloud spending in a single dashboard. View cloud costs and savings by week, month, year, application, or user.

When you instantiate a VM, the CSP begins charging (in one second increments) for the compute instance and any storage devices you configure. You must configure disk space for the root volume (where the operating system is stored) and normally you configure additional disk space for your data (the data volume). VMs instantiated by OpCenter include two additional disk volumes: the image volume (to store the container image) and the snapshot volume (to store snapshots).

The default size for the root volume is 40 GB. If this volume is used for an entire month, the charge would be about USD 4.

Cloud cost for a single job is calculated as follows.

Cloud cost = Instance cost + data volume cost + image volume cost + snapshot volume cost

where the Instance can be an On-demand Instance or a Spot Instance. The root volume cost is *not* included.

Net savings is calculated by comparing what the cost would be if the job were run on an On-demand Instance, not using OpCenter, with the *Cloud Cost*. That is,

Net savings = On-demand cost - Actual instance cost - image volume cost - snapshot volume cost

Cloud Cost and *Net Savings* for each job are aggregated to arrive at the numbers reported per week, per month, etc.

Jobs Screen

Using the *Jobs* screen, you can view and manage current jobs, and you can view archived jobs. You can search for jobs and filter the displayed results. Click on a job to display detailed information about the job, including logs and graphs of resource utilization over time.

To migrate, modify or cancel a currently running job, identify the job by ID or by name. In the *Actions* column are three icons corresponding to migrate, modify, and cancel, respectively. Clicking on an icon brings up a dialog box. Fill in the dialog box to complete the action.

To view the logs associated with a current or archived job, click on the job ID. In the table that is displayed, click on *Attachments* to list the logs associated with the job. Click on the *Preview* icon to view the contents of a log.

To view a graphical display of resource utilization associated with a current or archived job, click on the job ID and then click on the *WaveWatcher* tab in the displayed table.

App Library Screen

The *App Library* screen shows the container images that are currently loaded in to the OpCenter image library. The display shows images that are automatically loaded (*Built-in*) as well as images that you add (*Private*). To add an image, go to the *Private* tab and click on *Add Image*.

Images with the status listed as *Ready* are currently loaded in to the OpCenter's image cache.

Using the MMCloud CLI

Step 1 Select your method for invoking MMCloud CLI commands.

Choose one of the following:

- Log in to the OpCenter web interface and open the *Web CLI* shell.

  When you log in for the first time, enter username *admin* and default password *memverge.* From any screen (except the *Submit Job* screen), click on the terminal icon on the top, right-hand side to open the *Web CLI* shell.

  A subset of CLI commands is available via the *Web CLI* shell. Type `-h` to see what commands are supported. Skip to step 3 *(on page 24)*.
- From the OpCenter web interface landing page, go to *Download command line tool for* displayed at the bottom of the page on the right-hand side. Click on the link for your operating system (Windows, Linux, or macOS)

  Use **`chmod`** to make the downloaded **`float`** binary executable and add path to the **`float`** binary to your PATH variable.

  Note: From macOS 10.15 (Catalina) onwards, zsh is the default shell. In zsh, float is a reserved word. To use the **`float`** CLI with macOS, either change the shell to bash or alias the word *float* to the **`float`** binary, as follows:

  ```
  alias float = /path/to/float_binary/float
  ```

  where `/path/to/float_binary/` is the path to where you placed the **`float`** binary.
- Establish a secure terminal session (using **`ssh`**) on the OpCenter server. The **`float`** CLI is automatically installed on this server.

Note: The *web CLI* shell is a web interface that provides a **`float`** prompt, that is, every entry you type automatically has **`float`** pre-pended. The other two methods provide a terminal with a shell prompt — you must type **`float`** followed by subcommand and options.

Step 2 Log in to the OpCenter as the *admin* user by typing the following:

```
float login -u admin -p memverge -a <OpCenter_ip_address>
```

where `<OpCenter_ip_address>` is the OpCenter's private IP address if you are within your organization's virtual private cloud (VPC), or public IP address if you are outside the VPC.

Step 3 Add a user by entering the following command:

```
float user add <user_name> --passwd <password> --group <group_name> --create
```

where `<user_name>`, `<password>`, and `<group_name>` are the user name, password, and group, respectively. Repeat the command with different parameters to add other users.

Step 4 Change the default password for the *admin* user by typing the following command:

```
float user passwd admin --passwd <admin_password>
```

where `<admin_password>` is the new password for the *admin* user.

Step 5 From a terminal session (not the *web CLI* shell), log in to the OpCenter as the *admin* user by typing the following:

```
float login -u admin -p <admin_password> -a <OpCenter_ip_address>
```

where `<OpCenter_ip_address>` is the OpCenter's private IP address if you are within your organization's virtual private cloud (VPC), or public IP address if you are outside the VPC.

Step 6 Upgrade to the latest OpCenter version.

Step 6.1 To see what releases are available, type the following command:

```
float release ls
```

Step 6.2 If there is a later release, upgrade by typing the following command:

```
float release upgrade -r <release_name>
```

where `<release_name>` is the latest release shown in the previous step. After the upgrade completes, log back in.

Step 6.3 Synchronize the `float` version with the OpCenter version by typing the following command:

```
float release sync
```

Submitting a Job Using the Web Interface

The web interface provides a form that you complete to submit a job.

Step 1 Open the OpCenter web interface and click on *Submit Job*.

Step 2 In the form that pops up, fill in the fields as follows.

- *Name* (optional): Enter a name to identify your job, for example, HelloWorld. If left blank, the OpCenter creates a name for you.
- *Image* (required): Use the drop-down menu to select a container image to run your application, for example, cactus.
- *Job Script*: Enter a location to reach your job script, such as an S3 bucket or a URL, or click on *Enter Content* to upload a file or paste commands into the window. Here is an example job script:

```
#!/bin/bash
echo "Congratulations! You have submitted your first job"
for(( c=1; c<3; c++))
do
        if [[ $(($c % 3)) == 1 ]]; then
                echo "Hello World!"
        else
                echo "Your next job will be more interesting"
        fi
                sleep 20s
done
echo "Job complete"
```

Note: Some containers have a built-in ENTRYPOINT, that is, a command that runs automatically when the container starts. These containers do not require a job script.

- Under *Instance Preferences*, select *CPU & Memory*, then choose, for example, 2 for *vCPU (min)* (required) and 4 for *Memory (min GiB)* (required).
- Under *Storage Volumes*, select *New volume*, for example, 10 for the *Size* and /data for the *Mount Point*.
- Ignore the *Advanced* settings.

Note: The generated CLI command string is shown on the right-hand side.

Step 3 Click on the *Submit* button at the bottom of the right-hand side.

Step 4 To view the status of the job you just submitted, click on *Jobs* in the left-hand panel.

Step 5 Click on the *ID* associated with your job and then go to the *Attachments* tab.

Step 6 Click on the *Preview* icon next to the log named *stdout.autosave* to view the output of the example job.

Submitting a Job Using the CLI

Use the MMCloud CLI to submit a job.

Step 1 On the computer where you installed the MMCloud CLI, create a file called helloworld.sh and insert the following contents.

```
#!/bin/bash
echo "Congratulations! You have submitted your first job"
for(( c=1; c<3; c++))
do
        if [[ $(($c % 3)) == 1 ]]; then
                echo "Hello World!"
        else
                echo "Your next job will be more interesting"
        fi
                sleep 20s
done
echo "Job complete"
```

Step 2 Save and then close the file.

Step 3 Open a terminal on this computer and go to the directory where you created helloworld.sh.

Step 4 Log in to the OpCenter by entering:

```
float login -a <op_center_ip_address> -u admin -p <password>
```

where `<op_center_ip_address>` is the public (if you are outside the VPC) or private (if you are inside the VPC) IP address associated with the OpCenter. Use the default value (memverge) for `<password>` if you have not changed it.

Step 5 Submit a job by entering:

```
float submit -i cactus -j helloworld.sh -c 2 -m 4 --dataVolume [size=10]:/data
```

Step 6 Query the job status by entering:

```
float squeue
```

The entry in the first column (labeled ID) is the identifier for your job.

Step 7 View the output of your example job by entering:

```
float log cat stdout.autosave -j <job_id>
```

where `<job_id>` is the identifier associated with your job.

Working with OpCenter

Introduction

The OpCenter is the control point for MMCloud. All interactions with the OpCenter use the web interface, the Web CLI shell, or the **`float`** CLI running in a terminal session. Although most features can be invoked using any of the three methods, the CLI running in a terminal session offers the most functionality. Most users only run a subset of the **`float`** CLI commands in their everyday use of OpCenter for running workloads. OpCenter also includes support for upgrading the software without affecting its virtual machine instance, for migrating running jobs to other virtual machine types, and for testing application compatibility.

Although similar functionality is accessible in the web interface as with the CLI, the web interface also provides a dashboard and graphical displays of resource utilization.

A detailed **`float`** command reference *(on page 90)* is available; this section provides summary descriptions for common use cases.

Running a Batch Job

The Memory Machine Cloud web interface and CLI have a rich set of options that allow you to customize the container runtime environment.

Step 1 Log in to the OpCenter.

Choose one of the following:

- If you are using the web interface, enter your credentials on the landing page.
- If you are using the *web CLI* shell, you are already logged in.
- If you are using a remote terminal or a terminal session on the OpCenter server, enter the following command:

```
float login -u <username> -p <password> -a <OpCenter_ip_address>
```

where **`<username>`** and **`<password>`** are login credentials, and **`<OpCenter_ip_address>`** is the OpCenter's private IP address if you are within your organization's virtual private cloud (VPC), or public IP address if you are outside the VPC.

Note: The OpCenter IP address is cached, so you can omit it when you log back in. After the cache expires, the IP address defaults to local host. If you get a "connection refused" error, retry with **`-a <OpCenter_ip_address>`** option included.

Step 2 Display the images that are available in the OpCenter library.

Choose one of the following:

- CLI: Enter the following command:

```
float image list
```

- Web interface: Select *App Library* from the side panel.

Step 3 If the required image is not available in the OpCenter library, upload the image.

Choose one of the following:

- CLI: If the image is available from a public or private repository, enter the following command:

```
float image add <image_name> <image_URI> --user <repo_access_user> --token
 <repo_access_token>
```

where **`<repo_access_user>`** and **`<repo_access_token>`** are the credentials for accessing the repository if it is private (obtain these from the repository owner). For example:

```
float image add python docker.io/bitnami/python
```

If you are using the CLI in a terminal session, you can upload an image from a local directory by entering the following command:

```
float image upload <image_name> --path </path/to/image>
```

where `</path/to/image>` is the path to the directory where the image tar file is located on your local machine.

- Web interface: If the image is available from a public or private repository, on the *App Library* screen, select *Private* > *Add Image*. Fill in the pop-up form and click on *Add*.

Step 4 Prepare the job script for the workload.

> **Note:** The job script can be a local file, but there are other access options, such as from an S3 bucket or from a web server.

Step 5 Submit a job to the OpCenter.

Choose one of the following:

- CLI: Enter the following command:

```
float sbatch -i <image_name> -j <job_script> --cpu <num_cpu> --mem <mem_size>
 --dataVolume [size=<vol_size>]:/<mnt_point>
```

where:

  - `<image_name>` is the docker image to run the job
  - `<job_script>` is the job script to execute (include the complete path to the job script file)
  - `<num_cpu>` is the minimum number of virtual CPUs to use (can also specify as a range in form `min:max`)
  - `<mem_size>` is minimum memory capacity to use in GB (can also specify as a range in form `min:max`)
  - `<vol_size>` is the capacity of the data directory (in GB)
  - `<mnt_point>` is the mount point for the data directory.

> **Note:** If you are using the CLI, the image does not have to be in the App Library. Instead of `-i <image_name>`, use `-i <image_uri>` where `<image_uri>` is the URI to pull the image.

Example:

```
float sbatch -i /bitnami/python -j python_job_script.sh --cpu 4 --mem 8
 --dataVolume [size=10]:/data
id: EOgf83Ru3U5jf9u2JK6Gp
name: image-xorwav-c5d.large
user: admin
imageID: docker.io/bitnami/python:latest
status: Initializing
...(edited)
```

Alternatively, you can use a definition file in conjunction with the **`float sbatch`** command. For example, to submit the same job, enter:

```
float sbatch -d ./def1.yaml
```

where `def1.yaml` is a file with the following contents:

```
image: "bitnami/python"
job: python_job_script.sh
cpu: 4
mem: 8
dataVolume:
 -  "[size=10]:/data"
```

◦ Web interface: Click on *Submit Job* (left-hand panel), fill in the fields in the pop-up form, and then click on *Submit*.

  As you fill in the fields, a CLI command string is generated (see the right-hand panel), which helps in understanding what effect the information in the fields has.

Step 6 Check job status.

Choose one of the following:

◦ CLI: Enter the following command:

```
float squeue
```

  The first column shows the unique job identifier associated with each job.

◦ Web interface: Click on *Jobs* (left-hand panel).

Step 7 Display detailed information on the job you submitted.

Choose one of the following:

◦ CLI: Enter the following command:

```
float show -j <job_id>
```

  where `<job_id>` is the unique job identifier shown in the previous step.

◦ Web interface: Click on *Jobs* (left-hand panel) and then click on the *ID* associated with your job (identify your job by ID or name). A screen entitled *Job Details - <job_name>* is displayed.

Step 8 Display the logs associated with a job.

Choose one of the following:

◦ CLI: Enter the following command:

```
float log ls <job_id>
```

  where `<job_id>` is the job identifier.

◦ Web interface: On the *Job Details - <job_name>* screen, click on the tab entitled *Attachments*.

Step 9 View log file contents.

Choose one of the following:

◦ CLI: Enter the following command:

```
float log tail --follow <log_file_name> -j <job_id>
```

  where `<log_file_name>` is the name of the log file you specified in the job script and `<job_id>` is the job identifier.

> **Note:** As the job runs, logs are written to a directory mounted from the OpCenter server. When the job ends, the logs, for example, **stderr** and **stdout**, are automatically saved by the OpCenter as **stderr.autosave** and **stdout.autosave**, respectively.

- Web interface: On the *Job Details* - <*job_name*> screen, click on the tab entitled *Attachments* and then click on the Preview icon next to the log you want to view. Click on the *Refresh* button to update the display.

Step 10 Modify parameters (security groups, migration policy, VM creation policy, or periodic snapshots) associated with a running a job.

Choose one of the following:

- CLI: Enter the following command.

```
float modify -j <job_id> --<option> <option_string>
```

  where `<option>` is one of the following:

  - `addSecurityGroup` (`<option_string>` is the identifier of security group to add)
  - `rmSecurityGroup` (`<option_string>` is the identifier of the security group to remove)
  - `migratePolicy` (`<option_string>` is the new migration policy to apply)
  - `snapshotInterval` (`<option_string>` is the new periodic snapshot interval to apply)
  - `vmPolicy` (`<option_string>` is the new VM creation policy to apply)

- Web interface: On the *Jobs* screen, identify your job by ID or name. Under the *Actions* column, click on the *Modify Job* icon associated with your job. In the dialog box, fill in the new parameter values to apply to the job and then click on *Modify*.

Step 11 View contents of submitted job file (recent or archived jobs).

Choose one of the following:

- CLI: Enter the following command:

```
float show -c -j <job_id>
```

- Web interface: Not supported.

Step 12 Cancel a running a job.

Choose one of the following:

- CLI: Enter the following command:

```
float scancel -j <job_id>
```

- Web interface: On the *Jobs* screen, identify your job by ID or name. Under the *Actions* column, click on the *Cancel* icon associated with your job.

What To Do Next

After the job runs to completion, retrieve your results from the location you specified in the job script.

SpotSurfer

SpotSurfer removes the risk of running stateful applications on Spot Instances while reducing costs significantly.

Feature Description

Cloud Service (CSPs) sell Spot Instances at a deep discount compared to On-demand Instances. The drawback is that the CSP can reclaim any Spot Instance with only nominal warning (usually, two minutes or less). This is a problem for stateful applications where all progress is lost if the job does not run to completion.

SpotSurfer is a feature that allows stateful applications to always run to completion by "surfing" to a new VM instance if the underlying Spot Instance is reclaimed by the CSP. Customers get the cost savings of Spot Instances without the risk of a job restarting from the beginning if the Spot Instance is reclaimed.

The technology that makes SpotSurfer possible is AppCapsule, MMCloud's checkpoint/restore (C/R) capability. The AppCapsule is a moment-in-time snapshot of the application instance, including in-memory state and relevant files, which allows the workload to move to a new virtual machine and resume running. AppCapsule underlies both SpotSurfer and WaveRider — two independent mobility features that can be enabled simultaneously for a given job.

SpotSurfer is automatically enabled (disabled) if the job runs an a Spot Instance (On-demand Instance). AppCapsule creation is triggered automatically when the CSP signals that it is reclaiming the Spot Instance. Job execution pauses and then resumes on a new Spot Instance.

Operation

To use SpotSurfer, you don't have to do anything. SpotSurfer's default behavior is to intercept the Spot Instance reclaim signal, create the AppCapsule, instantiate a new Spot Instance of the same type as the instance that is being reclaimed, and recover the job in the new instance.

Using the OpCenter or the CLI, it is possible to modify the VM creation policy while a job is running. If the VM creation policy is modified, the new instance is of the type specified by the new policy. In most cases, the new policy specifies a Spot Instance but it could specify an On-demand Instance.

SpotSurfer for Memory-optimized Applications

In its default mode, SpotSurfer captures a snapshot of the current state of a running job only once — immediately prior to migrating the job to a new virtual machine instance. If a Spot Instance reclaim event causes the job to migrate, the short time available to save the snapshot prevents SpotSurfer from working with applications that use a lot of memory, for example, larger than 128 GB.

If periodic snapshots are enabled (set `snapshotInterval` to at least 10m when submitting a job), snapshots are taken at fixed time intervals — intervals long enough to allow a snapshot to complete, that is, an AppCapsule created and written to persistent storage. If a snapshot cannot complete in the Spot Instance reclaim time window, the most recent complete snapshot is used to recover the job in the new virtual machine. This means that all job progress since the last snapshot is lost.

When OpCenter runs on AWS, additional measures are available to handle memory-optimized applications. If AWS determines that a Spot Instance has an increased likelihood of being reclaimed, AWS may, at their discretion, send an AWS Rebalance Recommendation signal. The OpCenter intercepts the signal and uses a rules-based approach to decide whether to keep the current Spot Instance or to pro-actively capture a snapshot and move to a new Spot Instance. If the decision is to move, there is more time than in the normal reclaim window in which to complete the snapshot. This allows an application with a memory footprint larger than 128 GB to run without interruption on a Spot Instance.

Use the configuration option (`--ignoreRebalance`) with `float submit` to always ignore the AWS Rebalance Recommendation signal for a particular job.

WaveRider

When enabled, WaveRider allows OpCenter to manage cloud resources by moving jobs to optimally-sized virtual machines.

Feature Description

Resource utilization (CPU, memory and storage) is not constant while a job runs. Users face a difficult choice: over-provision and overpay, or under-provision and risk out-of-memory conditions. With WaveRider, users can start with a small virtual machine (VM) and dynamically migrate to bigger VMs (and back down to smaller VMs) as the application demands change during a complete job run.

Job migration is initiated in three ways:

- Manual: migration initiated using CLI commands or the OpCenter web interface.
- Policy-driven: migration triggered by rules-based policy configured per job (policy based on CPU and memory utilization thresholds).
- Programmatic: migration executed at specific program breakpoints defined by job script.

Manual Job Migration

Running jobs can be moved manually at any time using the CLI or the OpCenter web interface, for example, to change the compute platform or the pay type.

Step 1 Log in to the OpCenter.

Choose one of the following:

- If you are using the web interface, enter your credentials on the landing page.
- If you are using the *web CLI* shell, you are already logged in.
- If you are using a remote terminal or a terminal session on the OpCenter server, enter the following command:

  ```
  float login -u <username> -p <password> -a <OpCenter_ip_address>
  ```

  where `<username>` and `<password>` are login credentials, and `<OpCenter_ip_address>` is the OpCenter's private IP address if you are within your organization's virtual private cloud (VPC), or public IP address if you are outside the VPC.

  Note: The OpCenter IP address is cached, so you can omit it when you log back in. After the cache expires, the IP address defaults to local host. If you get a "connection refused" error, retry with `-a <OpCenter_ip_address>` option included.

Step 2 Migrate a running job to a new virtual machine instance.

Note: The vmPolicy option associated with a running job can be modified. If the new policy is incompatible with the old policy (not considering `priceLimit`), then job migration is automatically triggered, so, for example, a job running on a Spot Instance migrates immediately to an On-demand Instance.

Choose one of the following:

- CLI: To migrate a job to a specific VM type, enter the following command:

  ```
  float migrate -t <instance_type> -j <job_id>
  ```

  where `<instance_type>` is a virtual machine type (for example, c4.xlarge in AWS) and `<job_id>` is the job identifier.

  To migrate to an instance whose capacities you define, enter the following command:

  ```
  float migrate --cpu <minCPU>:<maxCPU> --mem <minMem>:<maxMem> -j <job_id>
  ```

  where `<minCPU>:<maxCPU>` is the allowable range for the number of virtual CPUs, `<minMem>:<maxMem>` is the allowable range for the memory size in GB, and `<job_id>` is the job identifier. The upper bounds for number of virtual CPUs and memory size can be omitted, in which case only the lower bounds apply.
- Web interface: Go to the *Jobs* screen and locate your job by *ID* or *Name*. Under the *Actions* column, click on the *Migrate Job* icon. Fill out the fields in the pop-up screen and then click on *Migrate*.

Step 3 Check whether the job migrates successfully.

Choose one of the following:

- CLI: Keep entering the `float squeue` command until the job state changes from "Floating" to "Executing."
- Web interface: On the *Jobs* screen, click on the *Refresh* button until the job state changes from "Floating" to "Executing."

Step 4 View a record of any migration events

Choose one of the following:

- CLI: Use the `float log cat job.events -j <job_id>` command.
- Web interface: On the *Jobs* screen, click on your job, and then go to the *Instances* tab. If more than one instance is shown, the job migrated at least once. Go to the *Attachments* tab and click on the *Preview* icon next to the *job.events* log to see details.

Example of manual migration:

```
float migrate -f -t t3a.large -j Xx0r4CYE7X6MRmivjoITf
float log cat job.events -j Xx0r4CYE7X6MRmivjoITf
2023-04-19T14:42:13.334: Ready to migrate with spec job:
 lWpXdspWZcWESBMMy9Nbm, instType: t3a.large, CPU: 0:0, Memory: 0:0,
 zone: , payType:
2023-04-19T14:42:13.334: Attempt to find instance
 type for spec InstType:t3a.large,CPU:2 ~ 0,Memory:4 ~
 0,Zone:us-east-1b,CPUVendor:AuthenticAMD,priceLimit:0,priceLimitPerc:0
2023-04-19T14:42:13.389: Determined instance params:
 Zone:us-east-1b,InstType:t3a.large,CPU:2,Memory:8
2023-04-19T14:42:13.389: Ready to migrate with instance type: t3a.large,
 cpu: 2, memory: 8, zone: us-east-1b, last instance type: t3a.medium(Spot)
2023-04-19T14:42:13.389: Ready to checkpoint host i-0d7877d1978a4db4c
```

```
2023-04-19T14:42:14.759: Checkpointed host
 i-0d7877d1978a4db4c, result: [container:
 c63784ae91f26cc4b2f8d1980f84ce14bc17ae5b8478953c4a2cb06c61e3a8b3,
 checkpoint file: ], duration 1.369989165s
2023-04-19T14:42:27.401: Detached volume vol-0fc557b252c9a497c from host
 i-0d7877d1978a4db4c
2023-04-19T14:42:33.819: Detached volume vol-0ea186ddf37c64396 from host
 i-0d7877d1978a4db4c
2023-04-19T14:42:40.212: Detached volume vol-05bf0027cef28aa4f from host
 i-0d7877d1978a4db4c
2023-04-19T14:42:40.212: Ready to create new host to recover
2023-04-19T14:42:46.999: Created instance i-06107ed1970156e8d at
 us-east-1b, waiting for it to initialize
2023-04-19T14:45:02.014: Mounted vol-0fc557b252c9a497c:/mnt/float-data to
 i-06107ed1970156e8d
2023-04-19T14:45:02.014: Mounted vol-0ea186ddf37c64396:/mnt/float-image to
 i-06107ed1970156e8d
2023-04-19T14:45:02.014: Mounted vol-05bf0027cef28aa4f:/data to
 i-06107ed1970156e8d
2023-04-19T14:45:02.015: Created new host: i-06107ed1970156e8d(Spot)
2023-04-19T14:45:02.248: Got 1 containers on host i-06107ed1970156e8d
2023-04-19T14:45:02.248: Ready to recover
 {ID:c63784ae91f2,Checkpointed:true,Running:false} on host
 i-06107ed1970156e8d
2023-04-19T14:45:02.264: Job floated to instance i-06107ed1970156e8d (2
 CPU/8 GB) (Spot)
2023-04-19T14:45:02.893: Migrated to new VM: i-06107ed1970156e8d
```

Policy-driven Job Migration

When enabled, WaveRider uses a rules-based policy to determine when to migrate a job to a different virtual machine

The **`--migratePolicy`** option can be used with the **`float sbatch`** command to define the rules that determine when a job is migrated to a new virtual machine. The policy can be added to a running job or an existing policy associated with a running job can be changed by using the **`float modify`** command. Equivalent actions are available using the web interface.

The policy works as follows. If the upper threshold for CPU or memory utilization is crossed and utilization remains elevated for a specified interval, the job is migrated to a virtual machine that has more virtual CPUs or more memory. The increase in size is measured as a percentage defined by the **`step`** parameter. Similar behavior occurs if the lower threshold is crossed and utilization remains low for a specified interval: the job is moved to a smaller virtual machine.

If *stepAuto* is set to true, the manual *step* settings are ignored and the step size is calculated each time a threshold is crossed. The calculation depends on the specifications for the current virtual machine.

Step 1 Turn policy-driven migration on (default is off).

Choose one of the following:

- CLI: To enable policy-driven migration, use **`--migratePolicy [enable=true]`** or **`--migratePolicy [disable=false]`**.
- Web interface: On the *Submit Job* screen, go to *Advanced* section and toggle the *Auto-Migration Policy* setting from Off to On.

Step 2 If needed, override default values for policy rules.

Choose one of the following:

- CLI: Attach parameters to **`--migratePolicy`** as a string enclosed in square brackets (only include parameters that have values different from the default).

  The following parameters can be included in the string (default values listed in parentheses). If a unit is not shown, the value is a percentage of the maximum possible.

  - **`cpu.upperBoundRatio`** (90): upper threshold for utilization per virtual CPU (percentage)
  - **`cpu.lowerBoundRatio`** (5): lower threshold for utilization per virtual CPU (percentage)
  - **`cpu.upperBoundDuration`** (30s): time that utilization per virtual CPU must remain above the upper threshold before migration is triggered
  - **`cpu.lowerBoundDuration`** (5m0s): time that utilization per virtual CPU must remain below the lower threshold before migration is triggered
  - **`cpu.step`** (50): The percentage increase (or decrease) in the number of virtual CPUs in the new virtual machine versus the original virtual machine
  - **`cpu.limit`** (use 0 for no limit): The maximum number of vCPUs allowed. If a job migrates to a VM with this number of vCPUs, then migration to a VM with more vCPUs is not permitted. Migration to a VM with fewer vCPUs is permitted.
  - **`cpu.lowerLimit`** (use 0 for no limit): The minimum number of vCPUs allowed. If a job migrates to a VM with this number of vCPUs, then migration to a VM with fewer vCPUs is not permitted. Migration to a VM with more vCPUs is permitted.
  - **`mem.upperBoundRatio`** (90): upper threshold for memory utilization (percentage)
  - **`mem.lowerBoundRatio`** (5): lower threshold for memory utilization (percentage)
  - **`mem.upperBoundDuration`** (30s): time that memory utilization must remain above the upper threshold before migration is triggered
  - **`mem.lowerBoundDuration`** (5m0s): time that memory utilization must remain below the lower threshold before migration is triggered
  - **`mem.step`** (50): The percentage increase (or decrease) in memory capacity of the new virtual machine versus the original virtual machine
  - **`mem.limit`** (use 0 for no limit): The maximum memory capacity (in GB) allowed. If a job migrates to a VM with this memory capacity, then migration to a VM with more memory is not permitted. Migration to a VM with less memory is permitted.
  - **`mem.limit`** (use 0 for no limit): The minimum memory capacity (in GB) allowed. If a job migrates to a VM with this memory capacity, then migration to a VM with less memory is not permitted. Migration to a VM with more memory is permitted.

  - **`stepAuto`** (false): If **`stepAuto`** is set to true, then values for **`cpu.step`** and **`mem.step`** are calculated dynamically before each migration. Overrides values set with **`cpu.step`** and **`mem.step`**.
  - **`evade.OOM`** (true): If set to true, then the values set for **`cpu.limit`** and **`mem.limit`** are overridden. If an application touches swap space, migration continues until the largest VM offered by the CSP is reached.
- Web interface: In the *Submit Job* screen, go to the *Advanced* section and change values in fields pre-populated with default values. You can view the changes in the generated command line on the right-hand side.

Step 3 Submit job with auto-migration policy included.

Choose one of the following:

- CLI: Use the **`float submit`** command with the **`--migratePolicy`** option.

  Example:

```
float submit -i python -j ./python_job_script.sh --dataVolume [size=10]:/data -c 4
 -m 8 --migratePolicy [enable=true,mem.upperBoundRatio=60]
id: 8dF5j3yTnuzbHFKUIJTYt
name: python-
user: tester
imageID: docker.io/bitnami/python:latest
status: Submitted
submitTime: "2023-01-22T23:25:47Z"
duration: 7s
inputArgs: ' -j ./python_job_script.sh  -i python  --migratePolicy
 [enable=true,mem.upperBoundRatio=60]  -m 8  -c 4  --dataVolume
 [size=10]:/data '
vmPolicy:
    policy: spotFirst
    retryLimit: 3
    retryInterval: 10m0s
migratePolicy:
    cpu:
        upperBoundRatio: 90
        lowerBoundRatio: 5
        upperBoundDuration: 30s
        lowerBoundDuration: 5m0s
        step: 50
    mem:
        upperBoundRatio: 60
        lowerBoundRatio: 5
        upperBoundDuration: 30s
        lowerBoundDuration: 5m0s
        step: 50
```

- Web interface: In the *Submit Job* screen, fill in the fields (including those in the *Advanced* section) and then click on *Submit*.

Step 4 Modify auto-migration policy associated with running job.

Choose one of the following:

- CLI: To modify the auto-migration policy associated with a running job or to turn auto-migration on, use **`float modify --migratePolicy <policy-string> -j <job_id>`**

  Example (turn auto-migration on and change one parameter from its default value):

```
float modify --migratePolicy [enable=true,mem.upperBoundRatio=70] -j
 eFehjVpxNFet426BzCFgz
Warning: Are you sure you want to modify eFehjVpxNFet426BzCFgz?
New migratePolicy may impact auto-migration behavior.(yes/No): yes
Successfully modified eFehjVpxNFet426BzCFgz:  --migratePolicy
 [enable=true,mem.upperBoundRatio=70]
```

- Web interface: Go to the *Jobs* screen and locate your job by *ID* or *Name*. Under the *Actions* column, click on the *Modify Job*icon. Fill out the fields in the pop-up screen and then click on *Modify*.

Step 5 View a record of any migration events.

Choose one of the following:

- CLI: Use the `float log cat job.events -j <job_id>` command.
- Web interface: On the *Jobs* screen, click on your job, and then go to the *Instances* tab. If more than one instance is shown, the job migrated at least once. Go to the *Attachments* tab. Click on the *Preview* icon next to the *job.events* log to see details.

Programmatic Job Migration

Initiate job migration by inserting the **`float migrate`** command into the job script.

Overview

Although not strictly required, most jobs are submitted with a job script that sets up the execution environment and tells the job scheduler what to do. Some jobs proceed in a series of stages, for example, an initial stage to read in a large dataset followed by a computationally-intensive stage. These two stages may repeat several times until the computation is complete. Then the results are written out to disk.

The resource requirements (CPU, memory, and storage) for each of these stages are different. A **`float migrate`** command can be inserted at each point where the job moves to a new stage with different resource requirements. In this manner, the job runs on a virtual machine that matches the requirements for the given stage.

Example

The example shown here uses a simple shell program to demonstrate how programmatic migration can be implemented. In the example job script, the job is migrated at an arbitrary point. In a real application, the job would migrate at the point where resources were expected to change.

The contents of the job script are as follows:

```
cat progmigrate.sh
#!/usr/bin/env bash
LOG_PATH=$1
LOG_FILE=$LOG_PATH/output
touch $LOG_FILE
exec >$LOG_FILE 2>&1
echo "Starting to execute shell script"
echo "Job migrates when count reaches 50"
for(( c=1; c<300; c++))
do
        if [[ $(($c % 3)) == 0 ]]; then
```

```
                echo "$c is a multiple of three"
        else
                echo "$c is NOT a multiple of three" >&2
        fi
        if [[ $c == 50 ]]; then
                /opt/memverge/bin/float migrate -f --instType c5.2xlarge
                echo "Job migration initiated, wait for a while"
                sleep 20s
        fi
        sleep 1s
done
echo "Job is complete"
```

The job runs on a centos9 image.

```
float submit -i centos9 -j ./progmigrate.sh --instType c5.large --dataVolume [size=10]:/data
```

The output file shows where the job migration occurs.

```
float log tail --follow output -j 4BZXIxWu6L5ioCSqEMD63
Starting to execute shell script
Job migrates when count reaches 50
1 is NOT a multiple of three
----[edited]
50 is NOT a multiple of three
4BZXIxWu6L5ioCSqEMD63 on i-0027c7d62eaabafa6 is now migrating, please use squeue/show
 to monitor migration progress. Check logs for details
Job migration initiated, wait for a while
51 is a multiple of three
----[edited]
299 is NOT a multiple of three
Job is complete
```

The log file shows the events occurring when migrating the job from a c5.large instance to a c5.2xlarge instance.

```
float log cat job.events -j 4BZXIxWu6L5ioCSqEMD63
----[edited]
2023-01-24T20:33:24.997727759Z: Ready to migrate with instance type: c5.2xlarge, cpu:
 8, memory: 16, zone: us-east-1b, last instance type: c5.large(Spot)
2023-01-24T20:33:24.997929752Z: Ready to checkpoint host i-0027c7d62eaabafa6
2023-01-24T20:33:26.29553621Z: Checkpointed host i-0027c7d62eaabafa6, result:
 &{map[5ee5df78f592:]}, duration 1.297557077s
2023-01-24T20:33:26.515069329Z: Ready to reclaim host i-0027c7d62eaabafa6
2023-01-24T20:34:15.524359376Z: Ready to create new host to recover
2023-01-24T20:34:22.200051613Z: Reclaimed host i-0027c7d62eaabafa6
2023-01-24T20:35:55.597906896Z: Created new host: i-03fb670fbb577639e(Spot)
2023-01-24T20:35:55.694463545Z: Got 1 containers on host i-03fb670fbb577639e
2023-01-24T20:35:55.694528662Z: Ready to recover &{5ee5df78f592 false true} on host
 i-03fb670fbb577639e
2023-01-24T20:35:55.700293372Z: Job floated
2023-01-24T20:35:56.362953748Z: Migrated to new VM: i-03fb670fbb577639e
----[edited]
```

Controlling Virtual Machine Types

Allowed virtual machine types can be restricted to a user-defined list.

Feature Description

When the OpCenter selects a virtual machine, for example, to start a new job or to migrate an existing job to a new virtual machine, the virtual machine must meet the requirements specified by the user. The requirements are defined either by the number of vCPUs and memory capacity, or by the machine type (for example, t2.medium in AWS terminology).

The user can restrict the selection further by stipulating that the virtual machine must be one that appears on an *allow list* of virtual machine types.

Configuration

The allow list can be specified in two ways:

- Global configuration (applies to all jobs)
- Per-job configuration using **`float submit`** with the **`--allowList`** option.

The per-job configuration overrides the global configuration for that job.

The format for specifying an allow list is to concatenate one or more of the leading characters in the virtual machine type with * to indicate that the remaining characters can be anything. For example, to specify any virtual machine in the "t" family of virtual machine types, use t*. To specify any virtual machine in the "t2" family, use t2*, and so on. To specify a particular virtual machine, use the complete virtual machine name, for example, t2.xlarge. To specify any virtual machine type, use *.

To set the allow list in the global configuration, complete the following steps.

- Display the current setting by entering:

```
float config get provider.allowList
[*] <--- this is the default setting, allows all virtual machine types
```

- Set the allow list by entering, for example:

```
float config set provider.allowList "t*"
AllowList is set to [t*]
```

- Multiple entries can be inserted by separating the entries by "," or " ". For example:

```
float config set provider.allowList "t*,c*"
AllowList is set to [t* c*]
float config set provider.allowList "t* r5.8xlarge"
AllowList is set to [t* r5.8xlarge]
```

To set the allow list for a particular job, use the following procedure:

- Submit the job with the **`--allowList`** option. For example, to select from the "t" family of virtual machines:

```
float submit -i python -j python_job_script.sh --dataVolume [size=10]:/data -c 1 -m 4
 --allowList "t*"
float sinfo
+---------------------+--------+------------------------+---------+---
|       ENTITY        | STATUS |      INSTANCETYPE      | PAYTYPE |
+---------------------+--------+------------------------+---------+---
| i-0d9db5e9ec67984ad | normal | t3.medium(2 vCPU, 4 GB) | Spot    |
+---------------------+--------+------------------------+---------+---
(edited)
```

  For example, to select from the "c5" family of virtual machines:

```
float submit -i python -j python_job_script.sh --dataVolume [size=10]:/data -c 1 -m 4
 --allowList "c5*"
float sinfo
+---------------------+--------+------------------------+---------+---
|       ENTITY        | STATUS |      INSTANCETYPE      | PAYTYPE |
+---------------------+--------+------------------------+---------+---
| i-033dbf78595f31b5a | normal | c5.large(2 vCPU, 4 GB) | Spot    |
+---------------------+--------+------------------------+---------+---
(edited)
```

  For example, to select from the "t" or "c5" family of virtual machines:

```
float submit -i python -j python_job_script.sh --dataVolume [size=10]:/data -c 1 -m 4
 --allowList "t*" --allowList "c5*"
float sinfo
+---------------------+--------+------------------------+---------+---
|       ENTITY        | STATUS |      INSTANCETYPE      | PAYTYPE |
+---------------------+--------+------------------------+---------+---
| i-0fbfb17d32799f0e2 | normal | t3.medium(2 vCPU, 4 GB) | Spot    |
+---------------------+--------+------------------------+---------+---
(edited)
```

  To allow any virtual machine type, use **`--allowList "*"`**

Operation

How the allow list is applied depends on how the virtual machine instance is specified when the job is submitted.

- Job is submitted with the **`--type`** option

  The **`--type`** option specifies a particular virtual machine type (for example, r5.8xlarge). In this case, the allow list is *not* applied — the OpCenter attempts to start a virtual machine of the type specified. As the job executes, a job migration event may be triggered (manually, by policy, programmatically, or by a Spot Instance reclaim). If the job must be moved to a new virtual machine, then the allow list is applied — either the global configuration if **`--allowList`** is not used or the range of machines specified using the **`--allowList`** option if used.
- Job is submitted with the **`--cpu`** and **`--mem`** options

If the job is submitted with these options, the OpCenter searches for a virtual machine type that meets the requirements *and* also satisfies the constraints of the allow list (either the global configuration if **`--allowList`** not used or the types specified by the **`--allowList`** if used). If a migration event is triggered, then the allow list is applied to the selection of a virtual machine type for the new virtual machine instance.

Authentication

Authentication methods can be arranged hierarchically.

Methods

OpCenter maintains its own database of usernames and passwords for authenticating logins (users are added with the **`float user add`** command). This is the "built-in" method. There are additional authentication methods:

- Local Linux /etc/passwd file
- LDAP

If both methods are set to true, then authentication proceeds in this order: LDAP first, then local Linux passwd file (if username not found in LDAP directory), and then the "built-in" method (if username not found in /etc/passwd).

Configure Local Linux Authentication

You can configure local linux /etc/passwd authentication by using **`float`** commands or by editing the opcenter.yml configuration file.

To configure local Linux authentication using **`float`**, enter the following (does not require a restart of the OpCenter):

```
float config set security.enableLocal true
float config set security.adminGroup wheel
```

To configure local Linux authentication by editing the opcenter.yml file, complete the following steps.

- Log in to the OpCenter server.
- Open the file called `/etc/memverge/opcenter.yml` with a text editor.
- In the security section, insert the following lines.

  ```
  security:
    ...
    enableLocal: true
    adminGroup: wheel
  ```

- Save and close file.
- Restart the OpCenter by entering the following.

  ```
  float restart
  ```

Configure LDAP Authentication

LDAP authentication uses the Lightweight Directory Access Protocol to query a directory of usernames and passwords. You must configure and start your LDAP server before enabling LDAP authentication. The default port for LDAP is 636.

To enable LDAP authentication, complete the following steps.

- Log in to the OpCenter server.
- Open the file called `/etc/memverge/opcenter.yml` with a text editor.
- In the security section, insert the following lines.

```
security:
...
  enableLdap: true
  ldap:
    network: tcp
    addr: <ldap_server_ip_address>:636
    useTLS: true
    anonymous: true
    base: dc=memverge,dc=com
    adminGroup: wheel
    peopleOU: People
    groupOU: Group
  ...
```

  where `<ldap_server_ip_address>` is the IP address of the LDAP server.
- Save and close file.
- Update the OpCenter configuration by entering the following:

```
float config set security.enableLdap true
```

- Restart the OpCenter by entering the following.

```
float restart
```

Gateway Service

The gateway service provides a reverse proxy so that a job can migrate from one host to another without interrupting client connections.

Feature Description

In most cases, the OpCenter is used to schedule batch jobs, that is, jobs that run without an attached terminal — results are written to a file (or files) and retrieved when the job completes. OpCenter can also deploy containers that support interactive browser-based sessions, such as is required for RStudio or Jupyter Notebook. Each container can run on a Spot Instance in which case the OpCenter seamlessly migrates the job to a new virtual machine if the Spot Instance is reclaimed. The new virtual machine instance has a different IP address.

The gateway service provides a reverse proxy that ensures that each interactive session, for example, each RStudio session, appears to clients as an IP address that never changes even when the RStudio session moves to a different virtual machine. The gateway uses TCP port numbers to distinguish between different services running on virtual machines on the server side of the gateway.

Application

In the current release, RStudio and Jupyter Notebook/Lab servers can be connected to a gateway. For more detail on how to use RStudio with the gateway service, see Using RStudio with MMCloud *(on page 56)*.

Usage

In the current release, gateways can only be managed using the CLI. Enter:

```
float gateway -h
```

for usage and options. Gateways can be created (**`float gateway create`**), destroyed (**`float gateway destroy`**), or modified (**`float gateway modify`**). Use the **`float gateway modify`** command to add or remove a security group that applies to a particular gateway.

Each gateway has an associated ID. Each server, RStudio or Jupyter Notebook/Lab, has an associated job ID. Servers are connected to gateways by using the **`gateway connect`** command:

```
float gateway connect -g <gw_id> -j <job_id> --targetPort <server_port_1> --targetPort
 <server_port_2>...
```

where **`<gw_id>`** is the gateway ID, **`<job_id>`** is the job ID associated with the server, and **`<server_port_1>`**, **`<server_port_2>`** ... are the ports the gateway and server use to connect. In most cases, the gateway and server need only connect one port.

You can disconnect a server from one gateway (use the **`float gateway disconnect`** command) and connect it to another gateway (use the **`float gateway connect`** command).

To display information about all running gateways, enter:

```
float gateway list
```

To display detailed information (including connected clients) about one running gateway, enter:

```
float gateway info -g <gw_id>
```

Web Widget

When a browser connects to the gateway, the gateway places a web widget at the top, right-hand side of the window (the user can move the widget). The widget provides an interface to the OpCenter *Job Details* screen. After logging in (use your credentials for the OpCenter), the user can view status and perform actions such as migrate or cancel (for this job only). When the application server (RStudio, for example) is in the "Floating" state, the widget alerts the user that the screen is unresponsive until the state returns to "Executing."

Service Templates

Service templates simplify the procedure for submitting jobs that include many options.

Feature Description

The service template feature allows users to start jobs from preconfigured templates, reducing the number of input parameters required. Each release of OpCenter contains a library of templates that are available for use.

Usage

All template-related actions begin with the **`float template`** command followed by a subcommand and any options that apply. To see the available subcommands, enter **`float template -h`**.

To submit a job using a template, use the **`float template deploy -T <template_name>`** command followed by other options (for example, **`--cpu`**) where **`<template_name>`** is the name of the template, for example, matlab.

Upgrading OpCenter

If a later version of OpCenter software is available, an OpCenter instance can be upgraded as long as no jobs are currently running.

Step 1 Log in to OpCenter by entering the following command from a terminal:

```
float login -u admin -p <admin_passwd> -a <OpCenter_ip_address>
```

where `<admin_passwd>` is the admin password and `<OpCenter_ip_address>` is the OpCenter's private IP address if you are within your organization's virtual private cloud (VPC), or public IP address if you are outside the VPC.

Note: Only the *admin* user can upgrade software. Upgrades cannot be executed from the *web CLI* shell.

Step 2 To see what version of OpCenter software is currently running, enter the following command:

```
float version
```

Step 3 To see what versions of OpCenter software are available, enter the following command:

```
float release ls
```

Step 4 If a later version of the software is available, upgrade the OpCenter instance by entering the following command:

```
float release upgrade --release <version>
```

Note: If `--release` is omitted, the latest version of the software is selected. If a job is running, the upgrade does not proceed.

Step 5 After executing a software upgrade, check that the upgrade succeeded by entering the following command:

```
float version
```

Step 6 If the OpCenter upgrade is successful, upgrade **`float`** to the same version by entering the following command:

```
float release sync
downloading ...
the float binary is synced up with opcenter
```

Uninstalling OpCenter

Uninstalling requires deleting the CloudFormation stack running OpCenter.

Step 1 Log in to the OpCenter

Step 2 Use **`float squeue`** to determine if any jobs are running.

Step 3 Use **`float scancel -j <job_id>`** to cancel any running jobs.

Step 4 Wait until all virtual machine instances have been reclaimed.

Step 5 Log in to your *AWS Management console*.

Check what region you are in. If you need to change regions, use the drop-down menu to select the correct region.

Step 6 Use the search tool in the navigation bar to find the *CloudFormation* console.

Step 7 On the left-hand panel of the *CloudFormation* console, click on *Stacks*.

A list of all your CloudFormation stacks is displayed.

Step 8 Select the OpCenter instance to delete.

Step 9 Click on the *Delete* button.

Support and Troubleshooting

Support

Support for MMCloud software is available 24 x 7 x 365. Contact MemVerge support by sending email to support@memverge.com.

Logs

OpCenter compiles logs of events related to its operation as well as logs that are specific to a particular job or a particular host (worker node).

To see the logs that pertain to the operation of the OpCenter, enter the following:

```
float log ls
+---------------------+---------+----------------------+
|      LOG NAME       |  SIZE   |   LAST UPDATE TIME   |
+---------------------+---------+----------------------+
| etcd.log            |  239504 | 2023-02-02T01:46:04Z |
| opcenter.access_log | 2811941 | 2023-02-02T02:31:43Z |
| opcenter.log        |  756620 | 2023-02-02T02:30:25Z |
| upgrade.log         |    1974 | 2023-01-26T15:40:39Z |
+---------------------+---------+----------------------+
```

The most useful log for troubleshooting is the `opcenter.log`.

To see the logs that pertain to a particular job, enter the following:

```
float log ls -j <job_id>
```

Example:

```
float log ls -j TQ9PUJhoY0XLtL58KcU1M
+------------------------+-------+----------------------+
|         LOG NAME       | SIZE  |   LAST UPDATE TIME   |
+------------------------+-------+----------------------+
| environments           |   330 | 2023-02-02T02:21:04Z |
| job.events             |  3009 | 2023-02-02T02:27:02Z |
| metrics-a1e1970a868a.txt | 12232 | 2023-02-02T02:46:35Z |
| output                 | 25963 | 2023-02-02T02:46:40Z |
| stderr.autosave        |     0 | 2023-02-02T02:21:27Z |
| stdout.autosave        |     0 | 2023-02-02T02:21:27Z |
+------------------------+-------+----------------------+
```

The job script used to submit this job redirects **stderr** and **stdout** to a file called `output` (that is why `stderr.autosave` and `stdout.autosave` have zero size). The log called `job.events` is useful for troubleshooting.

To see the logs that pertain to a particular host, enter the following:

```
float log ls -i <host_id>
```

Example:

```
float log ls -i i-0e40db7d105cb6793
+-------------------+--------+----------------------+
|     LOG NAME      |  SIZE  |   LAST UPDATE TIME   |
+-------------------+--------+----------------------+
| fagent.access_log |   8494 | 2023-02-02T17:10:10Z |
| fagent.log        |  11735 | 2023-02-02T17:08:00Z |
| fagent_init.log   |    639 | 2023-02-02T17:06:34Z |
| internal_output   |   1116 | 2023-02-02T17:07:59Z |
| messages          | 154889 | 2023-02-02T17:10:01Z |
+-------------------+--------+----------------------+
```

The log called **`messages`** is useful for troubleshooting. The log called **`fagent.log`** records interactions between the worker node and the OpCenter.

Troubleshooting

The following table shows commonly encountered errors and how to fix them.

Table 2. Troubleshooting OpCenter Issues

Error	Cause	Remedy
float login returns 'Get "https://127.0.0.1/api/v1/login": dial tcp 127.0.0.1:443: connect: connection refused'	Incorrect IP address for OpCenter or the OpCenter IP address has aged out of the local float cache.	Check OpCenter IP address and try again with float login -a <ip_address>
float command returns "Error: Session timeout (code: 2001)"	Current OpCenter session timed out	Log in to OpCenter
float image add or float submit returns "Error: Authentication failed, incorrect username or password (code: 2003)"	Attempt to access private repository with incorrect or missing credentials	Rerun command with valid credentials to access repository
float submit returns "Error: Unsupported argument, No instance types meet combined --cpu and --mem constraints. (code: 1027)"	No VM instance found that meets all requirements including price limit for Spot Instance	Resubmit with higher price limit if it is Spot Instance. Else, resubmit with different memory and vCPU ranges.
float squeue shows status as "WaitingForLicense"	OpCenter cannot retrieve valid license	Check license status by typing **`float license info`** or by clicking on the license icon on the web interface or by logging in to the MemVerge Account Server. Obtain new license or apply an existing license.
float log cat <logfile> returns "Error: Invalid argument, No such log"	Delay in writing to log file after the container starts running	Wait and then retry command
float ps returns "Error: Job is not executing"	Job is either initializing or it has completed	Use float squeue to determine status of job

Table 2. Troubleshooting OpCenter Issues (continued)

Error	Cause	Remedy
float squeue returns "No jobs" although jobs have been submitted and completed	float squeue queries job history in a fixed time interval (default: last hour)	Use float squeue -A to include all jobs

Working with Applications

Introduction

Some applications, for example, RStudio, require specific configurations to work with MMCloud. This section describes how to configure select applications to work with MMCloud.

Using RStudio with MMCloud

Summary

MMCloud is a software product from MemVerge that is used to deploy containerized applications in public clouds. Usually, the containers are used to run batch jobs, that is, jobs that run without an attached terminal — results are written to a file (or files) and retrieved when the job completes. RStudio is an integrated development environment (IDE) for the R programming language, which means that an RStudio user engages in an interactive session. This document describes how to use MMCloud to deploy a container that supports an interactive browser-based RStudio session. The container can run on a Spot Instance in which case MMCloud seamlessly migrates the job to a new virtual machine if the Spot Instance is reclaimed.

MMCloud's support for RStudio sessions includes a gateway, which is a reverse proxy that ensures that the RStudio session appears to clients as an IP address that never changes even when the RStudio session moves to a different virtual machine.

Architecture

As shown in the figure, the gateway has two sides: the client side and the server side. The gateway represents all back-end servers as a single IP address on the client side. Multiple servers, hosting the same or different services, can connect to the same gateway. For example, RStudio servers and Jupyter Notebook servers can run simultaneously in a server farm. On the client side, each service, for example, an RStudio session, is represented as an IP address:Port combination.

In the case of a Spot reclaim event, the RStudio session moves to a new virtual machine instance (either Spot or On-demand depending on your policy). The new virtual machine has a different IP address from the reclaimed Spot instance, but to the client this change is transparent — the client session continues without breaking the connection.

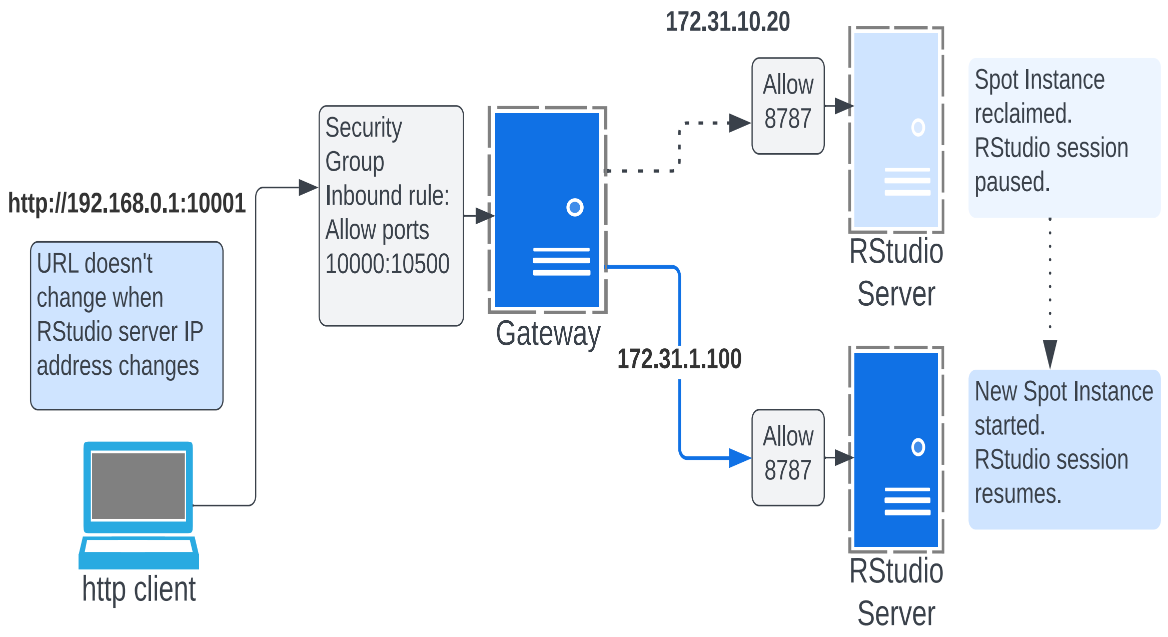

172.31.10.20
Allow 8787
RStudio Server
Spot Instance reclaimed. RStudio session paused.
http://192.168.0.1:10001
Security Group Inbound rule: Allow ports 10000:10500
Gateway
URL doesn't change when RStudio server IP address changes
172.31.1.100
Allow 8787
RStudio Server
New Spot Instance started. RStudio session resumes.
http client

Figure 3. RStudio Session Connected to Gateway

Operation

To the OpCenter, the RStudio session is a job, identified by a Job ID, that continues running until it is manually canceled. This job can run on a Spot Instance or an On-demand Instance. The gateway is a specialized server, started by the OpCenter, that runs on an On-demand Instance and continues running until it is manually canceled. The gateway is identified by a gateway ID.

The gateway and the RStudio server connect via a TCP port (the default port number for RStudio is 8787). The gateway selects the first available port from the range of ports configured on the client side of the gateway to identify the RStudio server to clients. In the example shown, the port 10001 is used by the gateway to identify the RStudio server to clients. When the client opens an http session on the client side of the gateway, the client uses port 10001 to indicate which RStudio server to connect to.

Prerequisites

To use RStudio, and the gateway, with MMCloud, you need the following:

- MMCloud Carmel 2.0 release or later
- Running instance of OpCenter with valid license

Create Inbound Firewall Rules

To access RStudio running on a container, a port on the container is *published*, that is, a port on the container host is mapped to a port on the container. This enables a browser to establish a TCP connection with the container. To open a port on the container host, create an inbound firewall rule by following these steps.

- Log in to your Cloud Service Provider (CSP)'s *Management Console* and follow the steps to create an inbound rule. The steps are different for each CSP. The steps shown here are for AWS (firewall rules are called *security groups* in AWS).
    - Open the *EC2 Dashboard* and click on *Security Groups*.
    - On the upper right, click on *Create security group*. The *Create security group* screen opens.
    - Enter a name and a description for the security group.
    - In the *Inbound rules* box, click on *Add rule*.
    - In the *Port range* box, enter 8787 (this is the default port number for using http to connect to RStudio).
    - In the *Source* box, enter 0.0.0.0/0 (this allows access from any host).
    - Scroll to the bottom of the page and click on *Create security group*.
    - After the security group is created, it appears in the table of *Security Groups*. Note the entry in the column titled *Security group ID*. It has the format *sg-xxxx*.
- Repeat these steps to create an inbound rule to allow access to a range of ports on the gateway. For example, if the gateway assigns ports in the range from 10000 to 10500, create an inbound rule to allow access to all ports in the range 10000 to 10500. Note the *Security group ID* for this rule.

Start the Gateway

- Log in to OpCenter.
- Enter **`float gateway create -h`** to show the options that are available when starting a new gateway. For most deployments, the default options are sufficient except for **`portRange`** and **`securityGroup`**, which must be provided. The port range must be between 10000 and 65535 (inclusive).
- Create a gateway by entering the following command:

  `float gateway create --portRange <minport>:<maxport> --securityGroup <sg-yyyy>`

  where `<minport>` is the lowest port number in the range, `<maxport>` is the highest port number in the range, and `<sg-yyyy>` is the ID for the Security Group that allows access to all the ports in the range. For example:

  ```
  float gateway create --portRange 10000:10500 --securityGroup sg-0fbb6a83983183364
  ```
- Show all running gateways by entering `float gateway list`. For example:

  ```
  float gateway list
  +-------+------+----------+--------------+------------+--------+-----+-------+
  |   ID  |STATUS| CONFIG   |   PUBLICIP   |  PORTRANGE | START  | JOBS|  COST |
  +-------+------+----------+--------------+------------+--------+-----+-------+
  | g-Qeh | Ready| 2Cores4GB| 35.175.116.18| 10000-10500|16:10:09| 1   | 14.20 |
  +-------+------+----------+--------------+------------+--------+-----+-------+
  (edited for clarity)
  ```
- To delete a gateway, enter `float gateway destroy -g <gw_id>` where `<gw_id>` is the ID of the gateway to delete.

Download the MMCloud RStudio Script

MMCloud includes a fully-functional CLI, but using a wrapper script simplifies starting and maintaining RStudio sessions.

- Use a browser to download the script here.
- Make the script executable by entering:

  ```
  chmod +x mmce_rstudio.sh
  ```

- Add the path to **mmce_rstudio.sh** to your PATH environment variable.
- To see what options are available to use with **mmce_rstudio.sh**, enter the following (default values are also displayed where applicable):

  ```
  mmce_rstudio.sh -h
  ```

  Regardless of the options used, the format must always be the following:

  ```
  mmce_rstudio.sh <option_1> <option_2>...<option_n> <action>
  ```

  where `<action>` is one of `start|list|stop|migrate`.

Start an RStudio Session and Connect to Gateway

- Log in to OpCenter.
- Check that the RStudio image is in the OpCenter repository. Enter the following:

  ```
  float image ls
  +------------+---------------------------+--------+-------------+
  |    NAME    |            URI            |  TAGS  | ACCESS USER |
  +------------+---------------------------+--------+-------------+
  | rstudio    | docker.io/memverge/rstudio | latest | opcenter    |
  +------------+---------------------------+--------+-------------+
  [edited for clarity]
  ```

  If the URI `docker.io/memverge/rstudio` is missing, upgrade the OpCenter software.
- Start an RStudio job using the default values and connect to a gateway (using port 8787) by entering:

  ```
  mmce_rstudio.sh -A <OpCenter_ip_address> -s <sg-xxxx> -g <gw_id> start
  ```

  where `<OpCenter_ip_address>` is the IP address of the OpCenter, `<sg-xxxx>` is the ID of the security group to allow access to port 8787, and `<gw_id>` is the ID of the gateway to connect to.

  The RStudio environment can be customized by overriding the default values. For example, use the `-t` option to change the RStudio login password.

  If the job is accepted, information about the job is displayed. Note the entries labeled *id:*, *publicIP:*, and *gatewayPort:*. For example:

  ```
  mmce_rstudiogw.sh -A 18.206.202.17 -s sg-0b23d4d7482ddb825 -g g-QehqwJldgogOFzeBKGG9P start
  float submit -i rstudio --cpu 2 --mem 4 -e RSTUDIO_USER=rstudio -e
   RSTUDIO_PASS=Welcome123! --securityGroup sg-0b23d4d7482ddb825  --publish
  ```

```
 8787:8787 --gateway g-QehqwJldgogOFzeBKGG9P --targetPort 8787  --extraOptions
 "--irmap-scan-path /home/rstudio/" -f
id: Rl6MmFMienMtyIwhbpQoR
name: rstudio-
user: admin
imageID: docker.io/memverge/rstudio:latest
status: Submitted
submitTime: "2023-03-20T21:52:39Z"
duration: 14s
cost: 0.0000 USD
inputArgs: -i rstudio --force -m 4 -c 2 --extraOptions
 '--irmap-scan-path /home/rstudio/' --gateway g-QehqwJldgogOFzeBKGG9P
 --targetPort 8787 --securityGroup sg-0b23d4d7482ddb825 --env
 RSTUDIO_USER=rstudio --env RSTUDIO_PASS=Welcome123! --publish 8787:8787
extraOptions: --irmap-scan-path /home/rstudio/
vmPolicy:
    policy: spotFirst
    retryLimit: 3
    retryInterval: 10m0s
envs:
    - RSTUDIO_USER=rstudio
    - RSTUDIO_PASS=Welcome123!
publishes:
    - 8787:8787
securityGroups:
    - sg-0b23d4d7482ddb825
gateway:
    gatewayID: g-QehqwJldgogOFzeBKGG9P
    publicIP: 35.175.116.18
    gatewayPort: "10001"
    targetPort: "8787"
```

- Check the status of the submitted job by entering the following (all RStudio jobs are shown):

  ```
  mmce_rstudio.sh -A <OpCenter_ip_address> list
  ```

  The first column (labeled *ID*) shows the IDs associated with RStudio jobs. Look for the ID that matches your job. When the *STATUS* changes from *Initializing* to *Executing*, you can connect to the RStudio session.

- Use a browser to open an http session on the RStudio host by entering:

  ```
  http://<publicIP>:<gatewayPort>
  ```

  where `<publicIP>` is the client-side IP address of the gateway and `<gatewayPort>` is the port that the gateway uses to identify the RStudio server. These are the values noted previously. You can confirm what `gatewayPort` your job is connected to by entering:

  `float gateway info -g <gw_id>` and matching it with the information displayed by `float squeue`.

  Loading the RStudio image can take about ten minutes (the uncompressed image is over a GB). If the RStudio host refuses the http connection, wait a few minutes and retry.

- At the RStudio prompt, log in with the default username `rstudio` and default password `Welcome123!`. If you overrode the defaults, use the username and password that you specified when submitting the RStudio job.

Migrate an RStudio Job

There is no difference between running an RStudio job on an On-demand instance and running on a Spot Instance. For either instance type, the RStudio job can be manually migrated to a new instance of any type.

If the RStudio job runs on a Spot Instance, the Spot Instance can be reclaimed by the CSP. If this happens, OpCenter automatically migrates the RStudio job to a new instance and resumes execution — without losing any work in progress. Effectively, the RStudio session continues uninterrupted.

To manually migrate an RStudio job, enter the following:

```
mmce_rstudio.sh -A <OpCenter_ip_address> -j <job_id> migrate
```

where `<job_id>` is the ID associated with the RStudio job to migrate. You can also specify the number of virtual CPUs and memory required for the new host.

Regardless of whether the RStudio job migrated manually or automatically, the IP address of the new instance is different. The ID associated with the RStudio job **does not** change, neither does the client-side gateway IP address and port number. The RStudio job remains connected to the gateway. Your browser session is not interrupted and you do not have to log in again.

Stop an RStudio Job

The host running the RStudio job continues to run indefinitely. To stop the RStudio job and remove the host, enter the following:

```
mmce_rstudio.sh -A <OpCenter_ip_address> -j <job_id> stop
```

where `<job_id>` is the ID of the RStudio job.

Managing Gateways

Gateways can be created (`float gateway create`), destroyed (`float gateway destroy`), or modified (`float gateway modify`). Use the `float gateway modify` command to add or remove a security group that applies to a particular gateway.

An Rstudio job can be disconnected from one gateway (use the `float gateway disconnect` command) and connected to another gateway (use the `float gateway connect` command).

Enter any command with the `-h` option to display the information that is required for each command.

Using Nextflow with MMCloud

Summary

Nextflow is a workflow manager used to run and manage computational pipelines such as those found in bioinformatics. Workflow managers make it easier to run complex analyses where there are a series of — sometimes interconnected — steps, each of which may involve different software and dependencies.

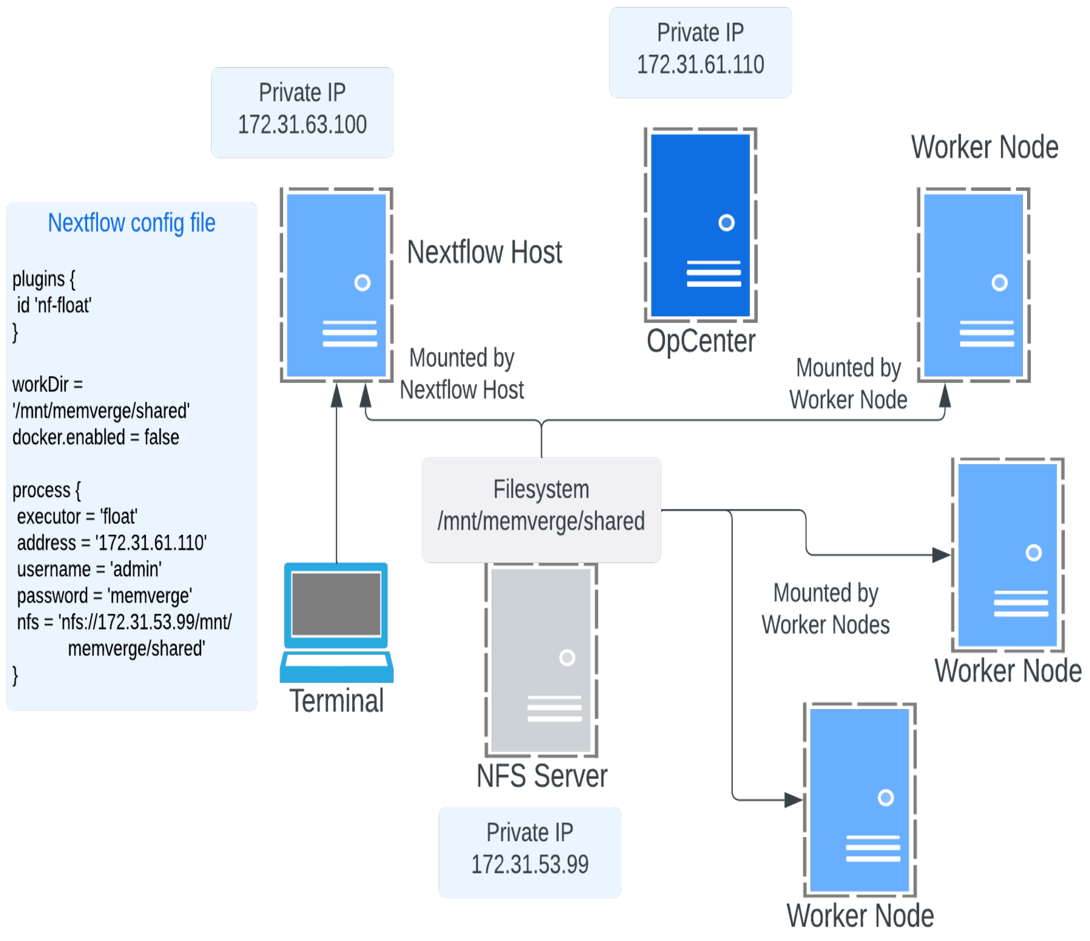
Nextflow provides a framework for describing how a workflow must be executed and includes a CLI for issuing **`nextflow`** commands. The execution environment for each step is described using the Nextflow DSL (Domain-Specific Language). In Nextflow terminology, each step is assigned to an "executor," which is a complete environment for running that step in the analysis.

By attaching a MemVerge-provided plugin to a workflow, Nextflow can use MMCloud as an "executor." From an MMCloud point of view, the execution step that is assigned to it is an independent job that it runs just like any other batch job. The Nextflow user enjoys the benefits that accrue because of using MMCloud, such as reduced costs and shorter execution times.

This document describes how to use Nextflow with MMCloud so that Nextflow can schedule one or more (or all) of the tasks in a workflow to run on MMCloud. Examples are used to demonstrate the principles; you can adapt and modify as needed to fit your workflow.

Configuration

A Nextflow workflow requires a working directory where temporary files are stored and where a process can access the output of an earlier step. When MMCloud is engaged as an executor, the OpCenter instantiates a Worker Node (a container running in its own virtual machine) for each step in the process pipeline. Every Worker Node and the Nextflow Host (the host where the nextflow binary is installed) must have access to the same working directory. In the figure below, a dedicated NFS Server provides shared access to the working directory and also acts as repository for input data and the final output.



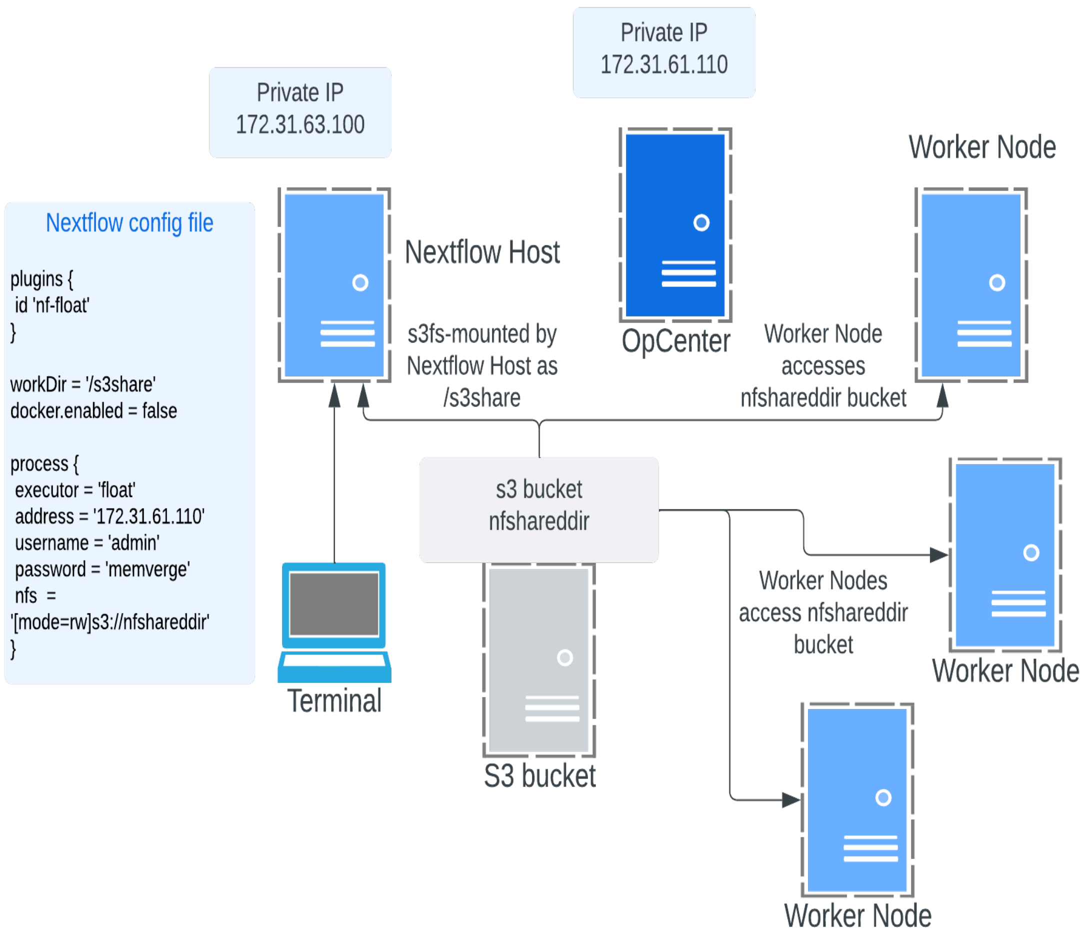
```
plugins {
 id 'nf-float'
}

workDir =
'/mnt/memverge/shared'
docker.enabled = false

process {
 executor = 'float'
 address = '172.31.61.110'
 username = 'admin'
 password = 'memverge'
 nfs = 'nfs://172.31.53.99/mnt/
          memverge/shared'
}
```

Figure 4. Nextflow Configuration using NFS Server

The Nextflow configuration file describes the environment for each executor. To use MMCloud as an executor, the configuration file must contain definitions for:

- Nextflow plugin
- Working directory mount point (where the shared directory is mounted on the Nextflow Host and all the Worker Nodes)
- IP address of the OpCenter
- Login credentials for the OpCenter
- Location of the shared directory (can be a file system on an NFS server or an S3 bucket in AWS)

The operation of Nextflow using an S3 bucket as the working directory is shown in the following figure.

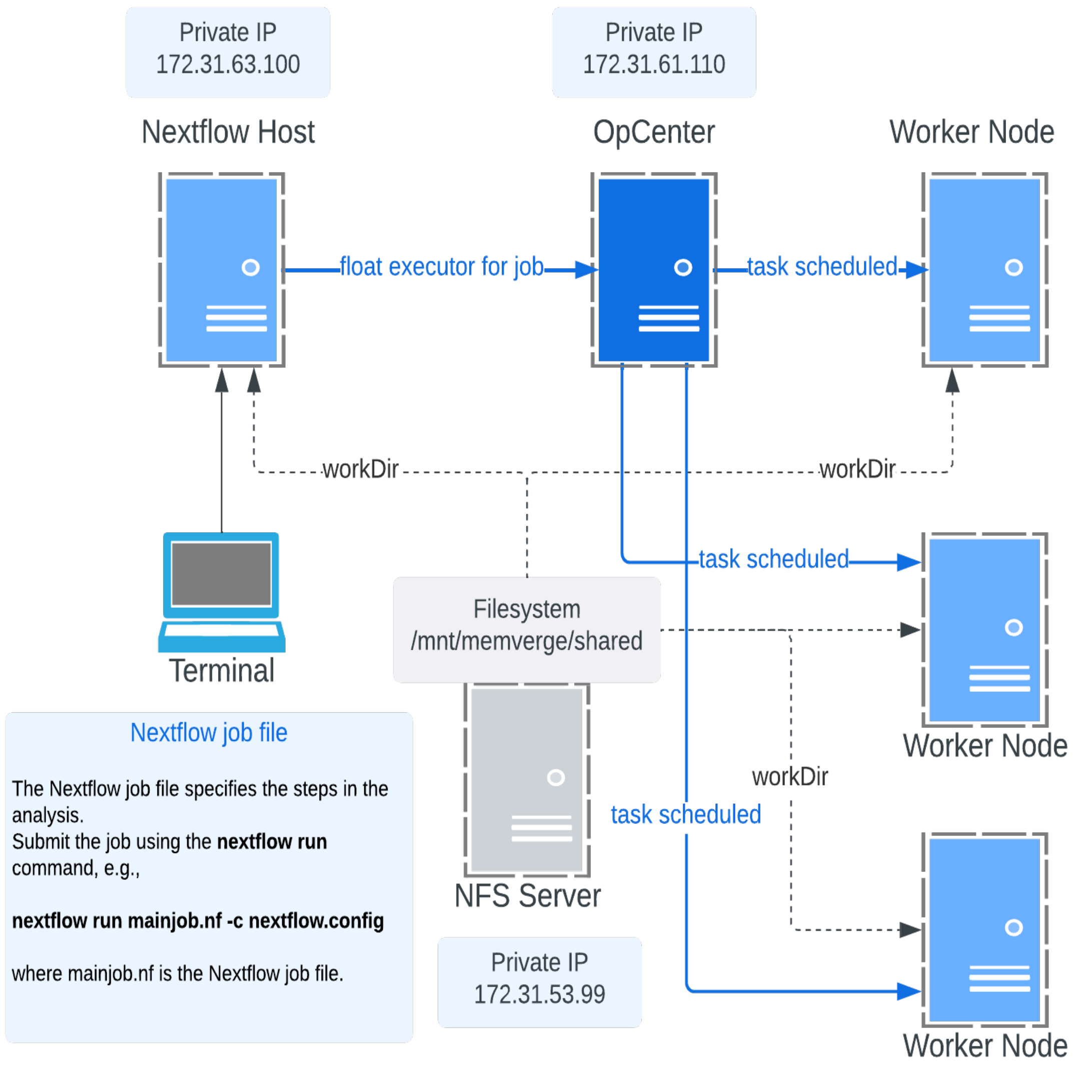


```
plugins {
 id 'nf-float'
}

workDir = '/s3share'
docker.enabled = false

process {
 executor = 'float'
 address = '172.31.61.110'
 username = 'admin'
 password = 'memverge'
 nfs =
'[mode=rw]s3://nfshareddir'
}
```

Figure 5. Nextflow Configuration using S3 Bucket

Operation

The Nextflow job file (a file with extension .nf) describes the workflow and specifies the executor for each process. When the user submits a job using the **`nextflow run`** command (as shown in the figure), any process with executor defined as "float" is scheduled for the OpCenter. Combining information from the configuration file and the job file, the Nextflow plugin formulates a **`float sbatch`** command and submits the job to the OpCenter. This procedure is repeated for every task in the workflow that has "float" as the executor. Every Worker Node mounts the same working directory so that the Nextflow Host and all the Worker Nodes read from, and write to, the same shared directory.



The Nextflow job file specifies the steps in the analysis.
Submit the job using the **nextflow run** command, e.g.,

nextflow run mainjob.nf -c nextflow.config

where mainjob.nf is the Nextflow job file.

Figure 6. Nextflow Operation with MMCloud

Requirements

To use Nextflow with MMCloud, you need the following:

- MMCloud Carmel 2.0 release or later
- Running instance of OpCenter with valid license
- Linux virtual machine running in same VPC as OpCenter (call this the Nextflow Host)
- On the Nextflow Host:
    - Java 11 or later release (the latest Long Term Support release is Java 17)
    - MMCloud CLI binary. You can download it from the OpCenter.
    - Nextflow

  - Nextflow configuration file
  - Nextflow job file
- (Optional) NFS Server to provide shared working directory. There are other possibilities; for example, the shared working directory can be hosted on the Nextflow Host or the shared working directory can be mounted directly from AWS S3.

Prepare the Nextflow Host

The Nextflow Host is a Linux virtual machine running in the same VPC as the OpCenter. If the Nextflow host is in a different VPC subnet, ensure that the Nextflow host can reach the OpCenter and that it can mount the file system from the NFS Server (if used).

All network communication among the OpCenter, the Nextflow Host, NFS Server (if used), and Worker Nodes **must** use private IP addresses. If the Nextflow Host uses an NFS-mounted file system as the working directory, ensure that any firewall rules allow access to port 2049 (the port used by NFS).

- Check the version of java installed on the Nextflow Host by entering:

```
java -version
openjdk version "17.0.6-ea" 2023-01-17 LTS
OpenJDK Runtime Environment (Red_Hat-17.0.6.0.9-0.4.ea.el9) (build
 17.0.6-ea+9-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-17.0.6.0.9-0.4.ea.el9) (build 17.0.6-ea+9-LTS,
 mixed mode, sharing)
```

- If needed, install Java 11 or later. Commercial users of Oracle Java need a subscription. Alternatively, you can install OpenJDK under an open-source license by entering (on a Red Hat-based Linux system):

```
sudo dnf install java-17-openjdk
```

- Install nextflow by entering:

```
sudo curl -s https://get.nextflow.io | bash
```

  This installs nextflow in the current directory. The installation described here assumes that you install nextflow in your home directory. You can also create a directory for your nextflow installation, for example, `sudo mkdir ~/nextflow`

- Check your nextflow installation by entering:

```
./nextflow run hello
N E X T F L O W  ~  version 22.10.7
Pulling nextflow-io/hello ...
 downloaded from https://github.com/nextflow-io/hello.git
Launching `https://github.com/nextflow-io/hello` [suspicious_ampere] DSL2 -
 revision: 1d71f857bb [master]
executor >  local (4)
[f9/d4d239] process > sayHello (3) [100%] 4 of 4 #
Bonjour world!

Ciao world!

Hola world!
```

```
Hello world!
```

If this job does not run, check the log called **.nextflow.log**.

- Download the OpCenter CLI binary for Linux hosts from the following URL:

  `https://<op_center_ip_address>/float`

  where `<op_center_ip_address>` is the public (if you are outside the VPC) or private (if you are inside the VPC) IP address for the OpCenter. You can click on the link to download the CLI binary (called float) or you can enter:

  `wget https://<op_center_ip_address>/float --no-check-certificate`

  If you download the CLI binary to your local machine, move the file to the Nextflow Host.
- Make the CLI binary file executable and add the path to the CLI binary file to your PATH variable.

(Optional) Prepare the Working Directory Host

The Nextflow Host and the Worker Nodes must have access to a shared working directory. There are several ways to achieve this. In the example shown here, a separate Linux virtual machine (the NFS Server) is started in the same VPC as the OpCenter.

Alternatively, you can use *s3fs-fuse* to mount an S3 bucket as a filesystem. Instructions on how to do this are in the next section *(on page 68)*.

You can obtain instructions on turning a generic CentOS-based server into an NFS server from this link. NFS uses port 2049 for connections, so ensure that any firewall rules allow access to port 2049. If the Working Directory Host is in a different VPC subnet, ensure that it can reach the Nextflow host and Worker Nodes. Set the subnet mask in **/etc/exports** to allow the Nextflow Host and Worker Nodes to mount file systems from the Working Directory Host.

For example:

```
cat /etc/exports
/mnt/memverge/shared 172.31.0.0/16(rw,sync,no_root_squash)
```

- Log in to the NFS Server and create the shared working directory.

  ```
  sudo mkdir /mnt/memverge/shared
  sudo chmod ugo+r+w+x /mnt/memverge/shared
  ```

- Log in to the Nextflow Server and mount the shared working directory (use the NFS Server's private IP address). Use **`df`** to check that the volume mounted successfully.

  ```
  sudo mkdir /mnt/memverge/shared
  sudo mount -t nfs <nfs_server_ip_address>:/mnt/memverge/shared /mnt/memverge/shared
  df
  ```

(Optional) Mount S3 Bucket as Filesystem

Some workflows initiate hundreds or even thousands of tasks simultaneously. If all these tasks access the NFS server at the same time, a bottleneck can occur. For these workflows, it can help to use an S3 bucket as the working directory.

With a small change to the configuration file, the Worker Node automatically mounts the S3 bucket as a linux file system. For the Nextflow Host, you must first install *s3fs-fuse* and manually mount the S3 bucket as a linux filesystem. Complete the following steps.

- Log in to your AWS Management Console.
  - Open the *Amazon S3* console.
  - From the left-hand panel, select *Buckets*.
  - On the right-hand side, click on *Create bucket* and follow the instructions.

    You must choose a bucket name (for example, nfshareddir) that is unique across all regions except China and the US government. Buckets are accessible across regions.
  - On the navigation bar, all the way to the right, click on your *username* and go to *Security credentials*.
  - Scroll down the page to the section called *Access keys* and click on *Create access key*.
  - Download the csv file.

    The csv file has two entries, one called *Access key ID* and one called *Secret access key*. You enter these in the next step.
- Log in to the Nextflow Host.
  - Enter `sudo yum install awscli`
  - Enter `yum install epel-release`
  - Enter `yum install s3fs-fuse`
  - Enter `aws configure` and enter the information you are prompted for.

    ```
    aws configure
    AWS Access Key ID [None]: AK....YB
    AWS Secret Access Key [None]: +H....f+
    Default region name [None]:
    Default output format [None]:
    ```
  - Create a directory to be the mount point for the S3 bucket.

    For example, enter `sudo mkdir /s3share` followed by `sudo chmod ugo+r+w+x /s3share`.
  - Mount the S3 bucket by entering (for example):

    `sudo s3fs nfshareddir /s3share -o allow_other`
  - Create a directory for writing results to by entering `sudo mkdir /s3share/results` followed by `sudo chmod ugo+r+w+x /s3share/results`.

Prepare the Configuration File

Nextflow configuration files can be extensive — they can include profiles for many executors. Create a simple configuration for using MMCloud as the sole executor by following these steps.

In the directory where you installed nextflow, create a file called **nextflow.config**. When a parameter requires an IP address, use a private IP address. The following configuration file uses the NFS server as the shared working directory.

```
cat nextflow.config
plugins {
  id 'nf-float'
}
workDir = '/mnt/memverge/shared'
docker.enabled = false
process {
  executor = 'float'
  address = 'OpCenter_ip_address'
  username = 'admin'
  password = 'memverge'
  cpus = '4'
  memory = '8 GB'
  nfs = 'nfs://nfs_server_ip_address/mnt/memverge/shared'
  commonExtra = '--migratePolicy [enable=true]'
}
```

The values for **cpus** and **memory** are used as the defaults if values are not specified in the job file.

The string following **commonExtra** is appended to the **float** command that is submitted to the OpCenter. The string shown here is an example: you can use any **float** command option.

The MemVerge plugin called **nf-float** is included in the Nextflow Plugins index. This means that the reference to **nf-float** resolves to "nf-float version: latest" and Nextflow automatically downloads the latest version of the **nf-float** plugin.

The following configuration file uses the S3 bucket as the shared working directory.

```
cat nextflow.config
plugins {
  id 'nf-float'
}
workDir = '/s3share'
docker.enabled = false
process {
  executor = 'float'
  address = 'OpCenter_ip_address'
  username = 'admin'
  password = 'memverge'
  cpus = '4'
  memory = '8 GB'
  nfs = '[mode=rw]s3://nfshareddir'
  commonExtra = '--migratePolicy [enable=true]'
}
```

Prepare the Nextflow Job File

The nextflow job file describes the workflow and how the workflow must be executed. To demonstrate how a simple workflow executes, follow these steps (example is adapted from nextflow.io.)

- Create a sample fasta file called **sample.fa** and place it in the shared working directory, for example, in **/mnt/memverge/shared** if you are using the NFS server, or in **/s3share** if you are using the S3 bucket.

```
cat /mnt/memverge/shared/sample.fa
>seq0
FQTWEEFSRAAEKLYLADPMKVRVVLKYRHVDGNLCIKVTDDLVCLVYRTDQAQDVKKIEKF
>seq1
KYRTWEEFTRAAEKLYQADPMKVRVVLKYRHCDGNLCIKVTDDVVCLLYRTDQAQDVKKIEKFHSQLMRLME
 LKVTDNKECLKFKTDQAQEAKKMEKLNNIFFTLM
>seq2
EEYQTWEEFARAAEKLYLTDPMKVRVVLKYRHCDGNLCMKVTDDAVCLQYKTDQAQDVKKVEKLHGK
>seq3
MYQVWEEFSRAVEKLYLTDPMKVRVVLKYRHCDGNLCIKVTDNSVCLQYKTDQAQDVK
>seq4
EEFSRAVEKLYLTDPMKVRVVLKYRHCDGNLCIKVTDNSVVSYEMRLFGVQKDNFALEHSLL
>seq5
SWEEFAKAAEVLYLEDPMKCRMCTKYRHVDHKLVVKLTDNHTVLKYVTDMAQDVKKIEKLTTLLMR
>seq6
FTNWEEFAKAAERLHSANPEKCRFVTKYNHTKGELVLKLTDDVVCLQYSTNQLQDVKKLEKLSSTLLRSI
>seq7
SWEEFVERSVQLFRGDPNATRYVMKYRHCEGKLVLKVTDDRECLKFKTDQAQDAKKMEKLNNIFF
>seq8
SWDEFVDRSVQLFRADPESTRYVMKYRHCDGKLVLKVTDNKECLKFKTDQAQEAKKMEKLNNIFFTLM
>seq9
KNWEDFEIAAENMYMANPQNCRYTMKYVHSKGHILLKMSDNVKCVQYRAENMPDLKK
>seq10
FDSWDEFVSKSVELFRNHPDTTRYVVKYRHCEGKLVLKVTDNHECLKFKTDQAQDAKKMEK
```

- Build a workflow that splits the fasta sequences into separate files and reverses the order by creating a nextflow job file called **splitfa.nf**.

  If you are using the S3 bucket, replace **params.in = "/mnt/memverge/shared/sample.fa"** with **params.in = "/s3share/sample.fa"**.

```
cat splitfa.nf
#!/usr/bin/env nextflow

params.in = "/mnt/memverge/shared/sample.fa"

/*
 * Split a fasta file into multiple files
 */
process splitSequences {
    executor = 'float'
    container = 'cactus'
    cpus = '4'
    memory = '8 GB'
    extra = '--vmPolicy [spotOnly=true]'

    input:
    path 'input.fa'
    output:
    path 'seq_*'

    """
    awk '/^>/{f="seq_"++d} {print > f}' < input.fa
    """
}

/*
 * Reverse the sequences
 */
process reverse {
    executor = 'float'
    container = 'cactus'
    cpus = '4'
    memory = '8 GB'

    input:
    path x

    output:
    stdout

    """
    cat $x | rev
    """
}

/*
 * Define the workflow
 */
workflow {
    splitSequences(params.in) \
      | reverse \
      | view
}
```

The string following **extra** is combined with the string following **commonExtra** in the config file and appended to the **float** command submitted to the OpCenter. The string value shown here after **extra** is an example: use any **float** command option. The **extra** setting overrides the **commonExtra** setting if they are in conflict.

Run a Workflow using Nextflow

Run a simple workflow by entering:

```
./nextflow run splitfa.nf -c nextflow.config -cache false
N E X T F L O W  ~  version 22.10.7
Launching `splitfa.nf` [zen_wing] DSL2 - revision: 641157438f
executor >  float (2)
[1a/7260c8] process > splitSequences [100%] 1 of 1 #
[c0/51267d] process > reverse        [100%] 1 of 1 #
0qes>
FKEIKKVDQAQDTRYVLCVLDDTVKICLNGDVHRYKLVVRVKMPDALYLKEAARSFEEWTQF
9qes>
KKLDPMNEARYQVCKVNDSMKLLIHGKSHVYKMTYRCNQPNAMYMNEAAIEFDEWNK
```

```
01qes>
KEMKKADQAQDTKFKLCEHNDTVKLVLKGECHRYKVVYRTTDPHNRFLEVSKSVFEDWSDF
1qes>
MLTFFINNLKEMKKAEQAQDTKFKLCEKNDTVKL
 EMLRMLQSHFKEIKKVDQAQDTRYLLCVVDDTVKICLNGDCHRYKLVVRVKMPDAQYLKEAARTFEEWTRYK
2qes>
KGHLKEVKKVDQAQDTKYQLCVADDTVKMCLNGDCHRYKLVVRVKMPDTLYLKEAARAFEEWTQYEE
3qes>
KVDQAQDTKYQLCVSNDTVKICLNGDCHRYKLVVRVKMPDTLYLKEVARSFEEWVQYM
4qes>
LLSHELAFNDKQVGFLRMEYSVVSNDTVKICLNGDCHRYKLVVRVKMPDTLYLKEVARSFEE
5qes>
RMLLTTLKEIKKVDQAMDTVYKLVTHNDTLKVVLKHDVHRYKTCMRCKMPDELYLVEAAKAFEEWS
6qes>
ISRLLTSSLKELKKVDQLQNTSYQLCVVDDTLKLVLEGKTHNYKTVFRCKEPNASHLREAAKAFEEWNTF
7qes>
FFINNLKEMKKADQAQDTKFKLCERDDTVKLVLKGECHRYKMVYRTANPDGRFLQVSREVFEEWS
8qes>
MLTFFINNLKEMKKAEQAQDTKFKLCEKNDTVKLVLKGDCHRYKMVYRTSEPDARFLQVSRDVFEDWS

Completed at: 04-Apr-2023 19:27:02
Duration    : 6m 56s
CPU hours   : 0.1
Succeeded   : 2
```

This nextflow job file defines two processes that use MMCloud as the executor. Using the **`float squeue`** command or the OpCenter GUI, you can view the two processes executed by OpCenter.

```
float squeue -f image='cactus' -f status='Completed'
+--------+----------+-------+-----------+------  -----+----------+------------+
|  ID    | NAME     | USER  |  STATUS   |SUBMI TTIME | DURATION |    COST    |
+--------+----------+-------+-----------+-----  ------+----------+------------+
|Tf89Q...| VTYRPN-1 | admin | Completed | 19:20:09Z  | 3m39s    | 0.0066 USD |
|vI7hV...| VTYRPN-2 | admin | Completed | 19:23:43Z  | 2m51s    | 0.0042 USD |
+--------+----------+-------+-----------+------------+----------+------------+
(edited)
```

Run an RNA Sequencing Workflow

This example (adapted from nextflow.io) uses publicly available data. Get the data here. Place this data in the shared working directory in a folder called **/mnt/memverge/shared/nextflowtest/data/ggal** if you are following the NFS example or in **/s3share/ggal** if you are following the S3 bucket example.

- Build a nextflow job file called **rnaseq.nf** with the following content if you are following the NFS server example. Replace **/mnt/memverge/shared** with **/s3share** (and **/mnt/memverge/shared/nextflowtest/data** with **/s3share**) if you are following the S3 bucket example.

  The rnaseq-nf image is not a "built-in" image in the OpCenter App Library. Specifying the URI for the rnaseq-nf image in the job file causes the OpCenter to pull the latest version of the image from Docker Hub.

  ```
  cat rnaseq.nf
  #!/usr/bin/env nextflow

  params.reads = "/mnt/memverge/shared/nextflowtest/data/ggal/ggal_gut_{1,2}.fq"
  ```

```
params.transcriptome =
 "/mnt/memverge/shared/nextflowtest/data/ggal/ggal_1_48850000_49020000.Ggal71.50
0bpflank.fa"
params.outdir = "/mnt/memverge/shared/results"

workflow {
    read_pairs_ch = channel.fromFilePairs( params.reads, checkIfExists: true )

    INDEX(params.transcriptome)
    FASTQC(read_pairs_ch)
    QUANT(INDEX.out, read_pairs_ch)
}

process INDEX {
    executor = 'float'
    container = 'nextflow/rnaseq-nf'
    cpus = '4'
    memory = '16 GB'
    tag "$transcriptome.simpleName"

    input:
    path transcriptome

    output:
    path 'index'

    script:
    """
    salmon index --threads $task.cpus -t $transcriptome -i index
    """
}

process FASTQC {
    executor = 'float'
    container = 'nextflow/rnaseq-nf'
    cpus = '4'
    memory = '16 GB'
    tag "FASTQC on $sample_id"
    publishDir params.outdir

    input:
    tuple val(sample_id), path(reads)

    output:
    path "fastqc_${sample_id}_logs"

    script:
    """
    mkdir fastqc_${sample_id}_logs
    fastqc -o fastqc_${sample_id}_logs -f fastq -q ${reads}
    """

}

process QUANT {
    executor = 'float'
    container = 'nextflow/rnaseq-nf'
    cpus = '4'
    memory = '16 GB'
    tag "$pair_id"
    publishDir params.outdir

    input:
    path index
    tuple val(pair_id), path(reads)

    output:
    path pair_id

    script:
    """
    salmon quant --threads $task.cpus --libType=U -i $index -1 ${reads[0]} -2
 ${reads[1]} -o $pair_id
    """
}
```

• Run the workflow by entering

```
./nextflow run rnaseq.nf -c nextflow.config -cache false
N E X T F L O W  ~  version 22.10.7
Launching `rnaseq.nf` [jolly_poincare] DSL2 - revision: d6def86b4e
executor >  float (3)
[8d/2a46e1] process > INDEX (ggal_1_48850000_49020000) [100%] 1 of 1 #
[57/23cd34] process > FASTQC (FASTQC on ggal_gut)      [100%] 1 of 1 #
[e0/a61466] process > QUANT (ggal_gut)                 [100%] 1 of 1 #
Completed at: 04-Apr-2023 20:06:54
Duration    : 13m 56s
CPU hours   : 0.3
Succeeded   : 3
```

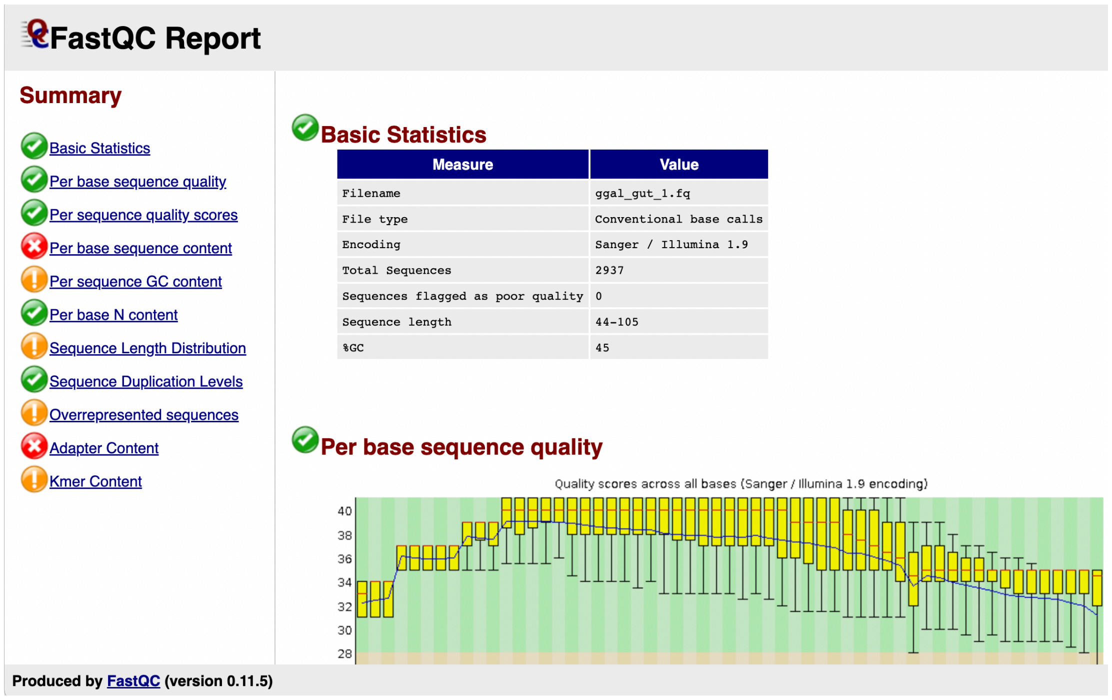
This nextflow job file defines three processes that use MMCloud as the executor. You can confirm that these processes execute on MMCloud.

```
float squeue -f name='D8G' -f status='Completed'
+-----------+-----------+-------+-----------+-----------+---------+----------+
|  ID       | NAME      | USER  |  STATUS   |SUBMIT TIME| DURATION|   COST   |
+-----------+--------  --+-------+-----------+-----------+---------+----------+
| IvwWWY... | D8GER6-1  | admin | Completed |21:52:33Z  | 2m38s   |0.0051 USD|
| lakXQg... | D8GER6-2  | admin | Completed |21:52:35Z  | 2m39s   |0.0052 USD|
| SyVSmL... | D8GER6-3  | admin | Completed |21:55:08Z  | 2m28s   |0.0048 USD|
+-----------+-----------+-------+-----------+-----------+---------+----------+
(edited)
```

• View the output at **/mnt/memverge/shared/results** (or **/s3share/results**).

```
ls
fastqc_ggal_gut_logs  ggal_gut
```

Some of the output is in html format, for example:



FastQC Report

Summary

- ✓ Basic Statistics
- ✓ Per base sequence quality
- ✓ Per sequence quality scores
- ✗ Per base sequence content
- ! Per sequence GC content
- ✓ Per base N content
- ! Sequence Length Distribution
- ✓ Sequence Duplication Levels
- ! Overrepresented sequences
- ✗ Adapter Content
- ! Kmer Content

✓ Basic Statistics

Measure	Value
Filename	ggal_gut_1.fq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	2937
Sequences flagged as poor quality	0
Sequence length	44-105
%GC	45

✓ Per base sequence quality

Quality scores across all bases (Sanger / Illumina 1.9 encoding)
40
38
36
34
32
30
28

Produced by FastQC (version 0.11.5)

Figure 7. Example Output from RNA Sequencing Workflow

Integration with Nextflow Tower

Nextflow Tower is a product from Seqera Labs that is used to launch, monitor, and manage computational pipelines from a web interface. You can also launch a Nextflow pipeline using the CLI on the Nextflow Host and monitor the progress in a cloud-hosted Nextflow Tower instance provided by Seqera Labs. Instructions are available here.

- Sign in to Nextflow Tower. If you do not have an account, follow the instructions to register.
- Create an access token using the procedure described here. Copy the access token to your clipboard.
- From a terminal on the Nextflow Host, enter:

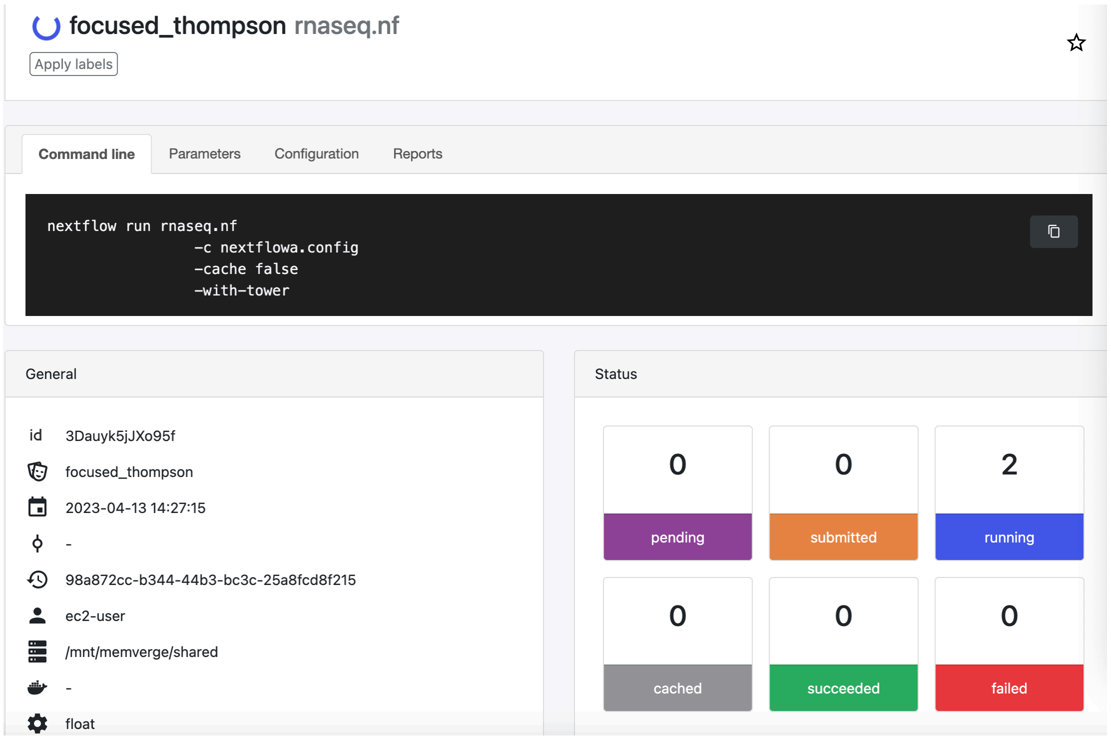
```
export TOWER_ACCESS_TOKEN=eyxxxxxxxxxxxxxxxQ1ZTE=
```

  where `eyxxxxxxxxxxxxxxxQ1ZTE=` is the access token you copied to the clipboard.
- Launch your Nextflow pipeline with the `with-tower` flag. For example:

```
nextflow run rnaseq.nf -c nextflowa.config -cache false -with-tower
        N E X T F L O W  ~  version 22.10.7
Launching `rnaseq.nf` [hungry_mestorf] DSL2 - revision: a379bdaa10
Monitor the execution with Nextflow Tower using this URL:
 https://tower.nf/user/<your_name>/watch/1S5pI3FwIaqKPg
executor >  float (3)
[b8/b4f470] process > INDEX (ggal_1_48850000_49020000) [100%] 1 of 1 #
[5a/0fce3a] process > FASTQC (FASTQC on ggal_gut)      [100%] 1 of 1 #
[d9/67eb6f] process > QUANT (ggal_gut)                 [100%] 1 of 1 #
Completed at: 13-Apr-2023 15:51:20
Duration    : 13m 36s
```

```
CPU hours    : (a few seconds)
Succeeded    : 3
```

• Open a browser and go to the URL.



```
nextflow run rnaseq.nf
                -c nextflowa.config
                -cache false
                -with-tower
```

id 3Dauyk5jJXo95f
focused_thompson
2023-04-13 14:27:15
-
98a872cc-b344-44b3-bc3c-25a8fcd8f215
ec2-user
/mnt/memverge/shared
-
float

pending	submitted	running	cached	succeeded	failed
0	0	2	0	0	0

Troubleshooting

As the nextflow job runs, log messages are written to a log file called "**.nextflow.log**", created in the directory where the nextflow job is running.

Using Cromwell with MMCloud

Summary

Workflows are computational pipelines, such as those found in bioinformatics, where there are a series of — sometimes interconnected — steps, each of which may involve different software and dependencies. Cromwell is an execution engine that allows users to run workflows written in the Workflow Description Language (WDL, pronounced widdle). WDL is a domain-specific language (DSL), that is, WDL is a language that has features customized for particular applications, in this case, for genomic analyses.

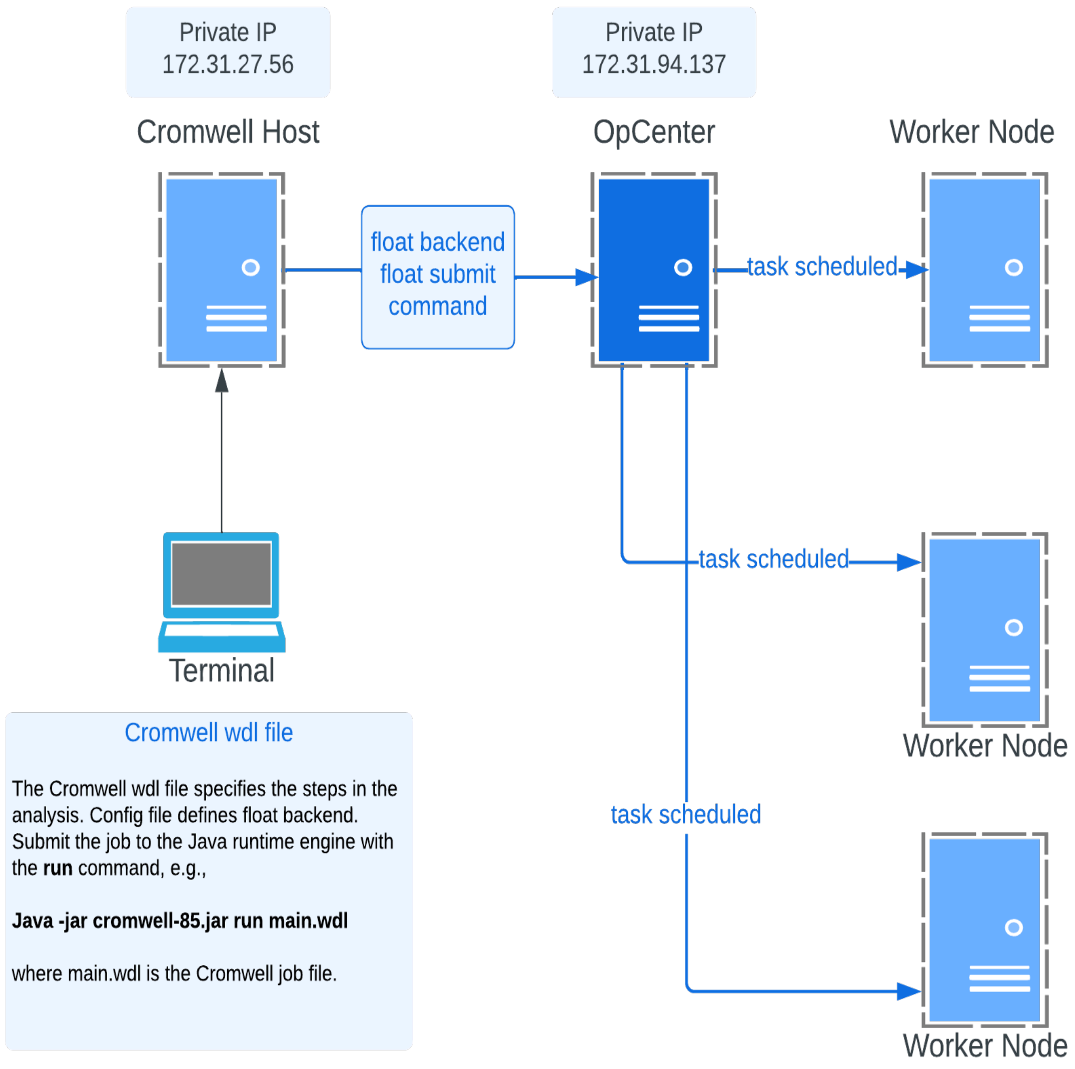
Cromwell is distributed as a Java ARchive (JAR) file, so running a workflow defined in a .wdl file requires a Java runtime engine to execute the Cromwell Java package. In a manner similar to Nextflow, the wdl file describes the steps in the workflow and how each step must be executed. The execution environment for each step in the analysis is described in Cromwell terminology as a "backend." The analogous concept in Nextflow is "executor."

By including a MemVerge-provided configuration file with the Java runtime, Cromwell can use Memory Machine Cloud (MMCloud) as a "backend." From a MMCloud point of view, the execution step that is assigned to it is an independent job that it runs just like any other batch job, which means that all the MMCloud features, such as SpotSurfer and WaveRider, are available. The benefits to the Cromwell user include cost savings and cloud resource rightsizing.

This document describes how to use Cromwell with MMCloud so that Cromwell can schedule one or more (or all) of the tasks in a workflow to run on MMCloud. Examples are used to demonstrate the principles; you can adapt and modify as needed to fit your workflow.

Configuration

The Cromwell Host is the host where you install Java and load the Cromwell JAR file. To use MMCloud as a backend, you must include a MemVerge-provided configuration file when you run the Cromwell JAR file. The configuration file contains the logic that translates the Cromwell task commands into a job file that is submitted (using the `float` CLI) to the MMCloud OpCenter. The OpCenter instantiates a Worker Node (a container running in its own virtual machine) for each task in the process pipeline that uses MMCloud as a backend.



The Cromwell wdl file specifies the steps in the analysis. Config file defines float backend. Submit the job to the Java runtime engine with the **run** command, e.g.,

```
Java -jar cromwell-85.jar run main.wdl
```

where main.wdl is the Cromwell job file.

Figure 8. Cromwell Configuration with MMCloud

The Cromwell configuration file describes the environment for each backend. To use MMCloud as a backend, the configuration file must contain definitions for:

- IP address of the OpCenter
- Login credentials for the OpCenter
- Default number of vCPUs for the Worker Node (value can be left blank)
- Default memory capacity for the Worker Node (value can be left blank)
- Default container image (value can be left blank)

If a value is left blank, it must be provided in the *runtime* section of the wdl file.

Operation

The Cromwell job file (a file with extension .wdl) describes the workflow and specifies the backend for each task (the default backend is the local host). When the user submits a job using the **`java run`** command, any process with the backend defined as "float" is scheduled for the OpCenter. Combining information from the configuration file and the job file results in a **`float submit`** command that is sent to the OpCenter. This procedure is repeated for every task in the workflow that has "float" as the backend.

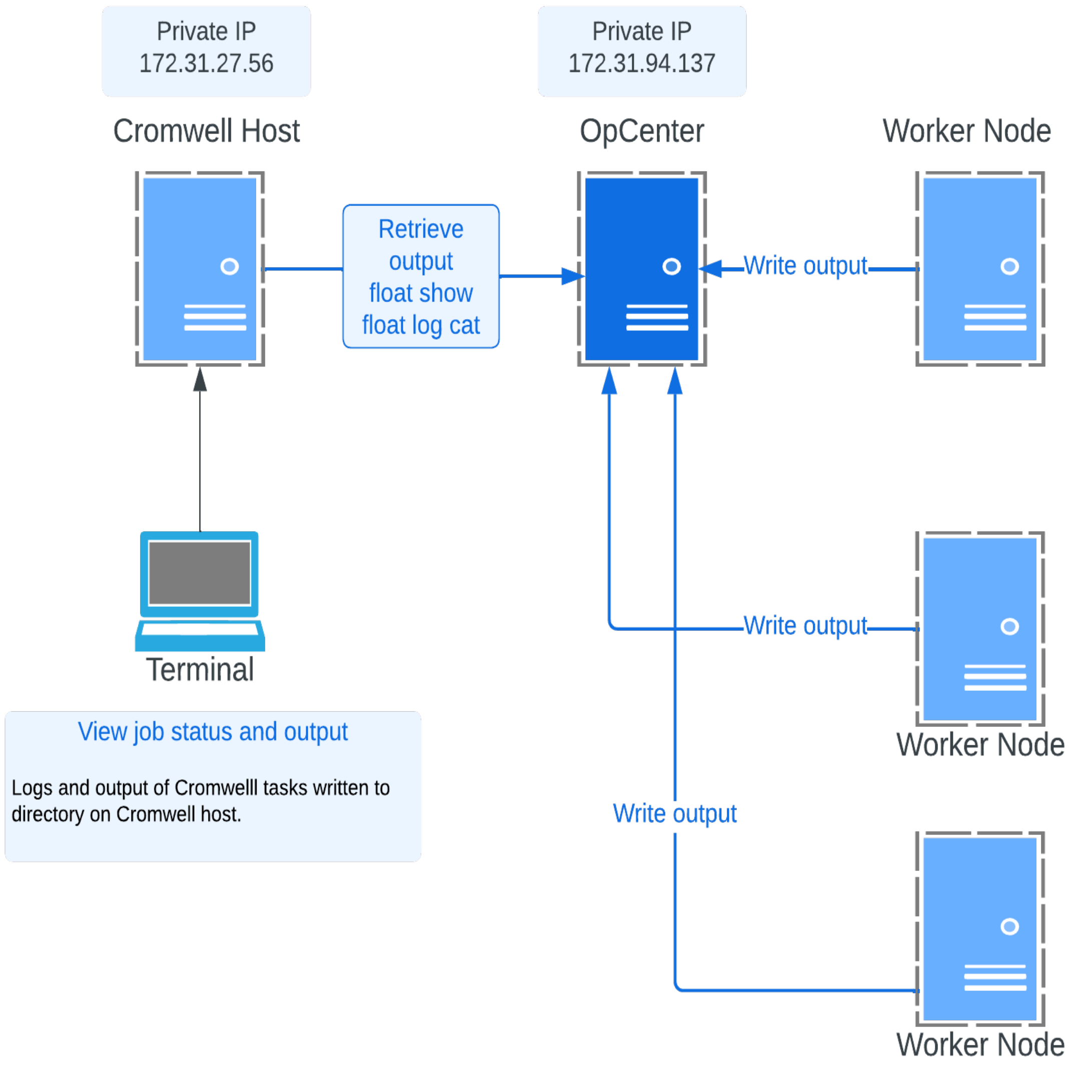


Figure 9. Cromwell Operation with MMCloud

Requirements

To use Cromwell with MMCloud, you need the following:

- MMCloud Carmel 2.0 release or later
- OpCenter instance with valid license
- Cromwell Host (can be your local computer or a Linux virtual machine running in same VPC as the OpCenter)
- Cromwell Host with the following:
    - Java 11
    - Cromwell jar file
    - MemVerge's Cromwell configuration file
    - Job file in wdl format
    - Input file(s) in json format
    - Options file in json format (optional)
    - MMCloud CLI binary. You can download it from the OpCenter.

Prepare the Cromwell Host

The Cromwell Host can be any computer that has access to the MMCloud. For complicated workflows, it is likely that the wdl file references objects in S3 buckets. For this reason and to comply with the same security policies that apply to the OpCenter, the instructions described here assume that the Cromwell Host is a Linux virtual machine running in the same VPC as the OpCenter. You can view instructions on how to create an AWS EC2 instance here. If the Cromwell Host is in a different VPC subnet, check that the Cromwell Host can reach the OpCenter. Ensure that any firewall rules allow access to ports 22 (the port used by ssh), 80, 443, and 8000.

- Check the version of java installed on the Cromwell Host by entering:

```
java -version
openjopenjdk version "11.0.18" 2023-01-17 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.18.0.10-3.el9) (build 11.0.18+10-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.18.0.10-3.el9) (build 11.0.18+10-LTS,
 mixed mode, sharing)
```

- If needed, install Java 11. Commercial users of Oracle Java need a subscription. Alternatively, you can install OpenJDK under an open-source licence by entering (on a Red Hat-based Linux system):

  `sudo dnf install java-11-openjdk`
- Create a directory called *cromwell* and `cd` to it
- Download the Cromwell jar file (85 is a recent version) from here and place it in the cromwell directory
- Check the Cromwell jar file by entering:

```
java -jar cromwell-85.jar --help
cromwell 85
Usage: java -jar /path/to/cromwell.jar [server|run|submit] [options] args>...

  --help                   Cromwell - Workflow Execution Engine
  --version
Command: server
```

```
Starts a web server on port 8000.  See the web server documentation for more
 details about the API endpoints.
Command: run [options] workflow-source
Run the workflow and print out the outputs in JSON format.
  workflow-source          Workflow source file or workflow url.
  --workflow-root <value>  Workflow root.
  -i, --inputs <value>     Workflow inputs file.
  -o, --options <value>    Workflow options file.
  -t, --type <value>       Workflow type.
  -v, --type-version <value>
                           Workflow type version.
  -l, --labels <value>     Workflow labels file.
  -p, --imports <value>    A zip file to search for workflow imports.
  -m, --metadata-output <value>
                           An optional JSON file path to output metadata.
Command: submit [options] workflow-source
Submit the workflow to a Cromwell server.
  workflow-source          Workflow source file or workflow url.
  --workflow-root <value>  Workflow root.
  -i, --inputs <value>     Workflow inputs file.
  -o, --options <value>    Workflow options file.
  -t, --type <value>       Workflow type.
  -v, --type-version <value>
                           Workflow type version.
  -l, --labels <value>     Workflow labels file.
  -p, --imports <value>    A zip file to search for workflow imports.
  -h, --host <value>       Cromwell server URL.
```

- Download the OpCenter CLI binary for Linux hosts from the following URL:

  `https://<op_center_ip_address>/float`

  where **`<op_center_ip_address>`** is the public (if you are outside the VPC) or private (if you are inside the VPC) IP address for the OpCenter. If you download the CLI binary (called float) to your local machine, move the file to the Cromwell Host.
- Make the CLI binary file executable and add the path to the CLI binary file to your PATH variable.
- Open a file called cromwell-float.conf and insert the following contents.

```
include required(classpath("application"))
# This is an example of how you can use Cromwell to interact with float.

backend {
  default = float

  providers {
    float {
      actor-factory =
 "cromwell.backend.impl.sfs.config.ConfigBackendLifecycleActorFactory"
      config {
        runtime-attributes="""
                String user = "admin"
                String pass = "memverge"
                String addr = "op_center_ip_address"
                String f_cpu = "minvCPUs"
                String f_memory = "minMemory"
                String f_docker = ""
                String f_extra = ""
        """

        # If an 'exit-code-timeout-seconds' value is specified:
        # - check-alive will be run at this interval for every job
        # - if a job is found to be not alive, and no RC file appears after this
interval
        # - Then it will be marked as Failed.
        # Warning: If set, Cromwell will run 'check-alive' for every job at this
interval
        exit-code-timeout-seconds = 30

        submit = """
            mkdir -p ${cwd}/execution
            tail -n +22 ${script} > ${cwd}/execution/no-header.sh
            head -n $(($(wc -l  < ${cwd}/execution/no-header.sh) - 14))
${cwd}/execution/no-header.sh > ${cwd}/execution/float-script.sh
            float sbatch -a ${addr} -u ${user} -p ${pass} -i ${f_docker} -j
${cwd}/execution/float-script.sh --cpu ${f_cpu} --mem ${f_memory} ${f_extra} >
${cwd}/execution/sbatch.out 2>&1
            cat ${cwd}/execution/sbatch.out | sed -n 's/id: \(.*\)/\1/p' >
${cwd}/execution/job_id.txt
            echo "receive float job id: "
            cat ${cwd}/execution/job_id.txt

            mkdir -p float-instance-$(cat ${cwd}/execution/job_id.txt)
            cd float-instance-$(cat ${cwd}/execution/job_id.txt)

            echo "cd ${cwd}/execution" > float-check-alive.sh
            echo "float -a ${addr} -u ${user} -p ${pass} log cat -j \$1
stdout.autosave > stdout" >> float-check-alive.sh
            echo "float show -a ${addr} -u ${user} -p ${pass} -j \$1
--runningOnly" >> float-check-alive.sh

            echo "cd ${cwd}/execution" > float-kill.sh
            echo "float scancel -a ${addr} -u ${user} -p ${pass} -j \$1" >>
float-kill.sh
            cat ${cwd}/execution/sbatch.out
        """

        kill = """
            source float-instance-${job_id}/float-kill.sh ${job_id}
        """

        check-alive = """
            source float-instance-${job_id}/float-check-alive.sh ${job_id}
        """

        job-id-regex = "id: (\\w+)\\n"
      }
    }
  }
}
```

where `op_center_ip_address` is the IP address of the OpCenter (public or private if the Cromwell Host is outside or inside the VPC, respectively), `minvCPUs` is the minimum number of vCPUs to use as the default for a Worker Node, and `minMemory` is the minimum memory capacity (in GB) to use as the default for a Worker Node.

- The string following `f_extra` is combined with the string following `f_extra` in the wdl file and appended to the `float submit` command sent to the OpCenter. The `f_extra` string in the wdl file overrides the `f_extra` string in the configuration file if they are in conflict.

Run a Simple Cromwell Workflow

The default backend for Cromwell is the local host. Run a simple "hello world" workflow on the local host by completing the following steps.

- Create a file called "helloworld.wdl" and insert:

```
# Example workflow
workflow myWorkflow {
    call myTask
}

task myTask {
    command {
        echo "hello world"
    }
    output {
        String out = read_string(stdout())
    }
}
```

- Run the "hello world" job by entering:

```
java -jar cromwell-85.jar run helloworld.wdl
```

The output from Cromwell is verbose. To determine that the job runs successfully and to view the output, look for the following section:

```
[2023-05-08 21:47:01,69] [info] BackgroundConfigAsyncJobExecutionActor
 [cb7c6fd7myWorkflow.myTask:NA:1]: Status change from - to Done
[2023-05-08 21:47:03,27] [info]
 WorkflowExecutionActor-cb7c6fd7-9020-41cf-841f-c95f43ce86da [cb7c6fd7]:
 Workflow myWorkflow complete. Final Outputs:
{
  "myWorkflow.myTask.out": "hello world"
}
[2023-05-08 21:47:06,67] [info] WorkflowManagerActor: Workflow actor for
 cb7c6fd7-9020-41cf-841f-c95f43ce86da completed with status 'Succeeded'. The
 workflow will be removed from the workflow store.
[2023-05-08 21:47:09,04] [info] SingleWorkflowRunnerActor workflow finished with
 status 'Succeeded'.
{
  "outputs": {
    "myWorkflow.myTask.out": "hello world"
  },
  "id": "cb7c6fd7-9020-41cf-841f-c95f43ce86da"
}
```

Run a Workflow using OpCenter as a Backend

A Cromwell workflow file can call many tasks. If the tasks are independent, Cromwell can scatter the tasks into parallel "shards." Once the shards are complete, the results can be gathered into a single output. The following example demonstrates different aspects of Cromwell.

- Task executed using Local as backend
- Task executed by OpCenter as backend
- Parallel sharding
- JSON files to supply input data and parameters

To run this example, complete the following steps:

- Create a file called 3step.wdl and insert the following content.

```
workflow scatterGather {
    Array[String] names

    call intro
    scatter (name in names) {
        call addList { input: name=name }
    }
    call compileList { input: items=addList.out }
}

task intro {
    command {
        echo "Starting to compile the grocery list"
    }
    output {
        String out = read_string(stdout())
    }
    runtime {
        backend: "Local"
    }
}


task addList {
    String name

    command {
        printf "[cromwell-addList] Add ${name} to list\n"
        sleep 30
    }
    output {
        String out = read_string(stdout())
    }
    runtime {
        f_docker: "cactus"
        f_cpu: "2"
        f_memory: "4"
        f_extra: "--name addList"
    }
}

task compileList {
    Array[String] items

    command {
        printf "[cromwell-compileList] These items are on the grocery list:\n" >
 my_file.txt
        sleep 1
        echo ${sep=". " items} >> my_file.txt
        cat my_file.txt
        sleep 30
    }
    output {
        String out = read_string(stdout())
    }
    runtime {
        f_docker: "cactus"
        f_extra: "--name compileList"
    }
}
```

- Create an input file called 3step.json and insert the following content.

```
{
   "scatterGather.names": ["Apples", "Bananas", "Milk", "Bread"]
}
```

- Create an options file called options.json and insert the following content.

```
{
    "write_to_cache": false,
    "read_from_cache": false
}
```

- Run the worklow by entering the following command.

```
java -Dconfig.file=cromwell-float.conf -jar cromwell-85.jar run 3step.wdl --inputs
 3step.json --options options.json
```

The task called "intro" runs locally. The task called addList creates four parallel shards that run on OpCenter. Finally, the output from the four shards are gathered by the task called compileList that runs on OpCenter.

The Cromwell output is voluminous. Check for key events during the run.

- Tasks assigned to backends.

```
[2023-05-09 20:45:17,93] [info] MaterializeWorkflowDescriptorActor
 [0b810f55]: Call-to-Backend assignments: scatterGather.addList -> float,
 scatterGather.intro -> Local, scatterGather.compileList -> float
```

- Parallel shards created.

```
[2023-05-09 20:45:22,46] [info]
 WorkflowExecutionActor-0b810f55-0d90-4c00-bf41-066e486e45b6 [0b810f55]:
 Starting scatterGather.addList (4 shards)
```

```
[2023-05-09 20:45:26,26] [info] Assigned new job execution tokens to the
 following groups: 0b810f55: 5
[2023-05-09 20:45:26,52] [info] BackgroundConfigAsyncJobExecutionActor
 [0b810f55scatterGather.intro:NA:1]: echo "Starting to compile the grocery list"
[2023-05-09 20:45:26,53] [info] DispatchedConfigAsyncJobExecutionActor
 [0b810f55scatterGather.addList:2:1]: printf "[cromwell-addList] Add Milk to
 list\n"
sleep 30
[2023-05-09 20:45:26,53] [info] DispatchedConfigAsyncJobExecutionActor
 [0b810f55scatterGather.addList:3:1]: printf "[cromwell-addList] Add Bread to
 list\n"
sleep 30
[2023-05-09 20:45:26,54] [info] DispatchedConfigAsyncJobExecutionActor
 [0b810f55scatterGather.addList:0:1]: printf "[cromwell-addList] Add Apples to
 list\n"
sleep 30
[2023-05-09 20:45:26,54] [info] DispatchedConfigAsyncJobExecutionActor
 [0b810f55scatterGather.addList:1:1]: printf "[cromwell-addList] Add Bananas to
 list\n"
```

- Results gathered.

```
[2023-05-09 20:51:14,26] [info]
 WorkflowExecutionActor-0b810f55-0d90-4c00-bf41-066e486e45b6 [0b810f55]:
 Starting scatterGather.compileList
```

- Job concluded successfully.

```
[2023-05-09 20:54:38,68] [info] SingleWorkflowRunnerActor workflow finished with
 status 'Succeeded'.
{
  "outputs": {
    "scatterGather.compileList.out": "[cromwell-compileList] These items are on
 the grocery list:\n[cromwell-addList] Add Apples to list. [cromwell-addList]
 Add Bananas to list. [cromwell-addList] Add Milk to list. [cromwell-addList]
 Add Bread to list",
    "scatterGather.intro.out": "Starting to compile the grocery list",
    "scatterGather.addList.out": ["[cromwell-addList] Add Apples to list",
 "[cromwell-addList] Add Bananas to list", "[cromwell-addList] Add Milk to
 list", "[cromwell-addList] Add Bread to list"]
  },
  "id": "0b810f55-0d90-4c00-bf41-066e486e45b6"
}
```

OpCenter shows the five tasks (four "scatter" tasks and one "gather" task).

```
float squeue -A -f name=List -d

+-----------------------+-------------+-------+-----------+----------------------+----
------+------------+
|          ID           |    NAME     | USER  |  STATUS   |     SUBMIT TIME      |
 DURATION |    COST    |
+-----------------------+-------------+-------+-----------+----------------------+----
------+------------+
| fihe9H73Mw7stcdCN1JPD | compileList | admin | Completed | 2023-05-09T20:51:17Z |
 2m37s    | 0.0089 USD |
| JO2HXLD0WJLYVPW2npISI | addList     | admin | Completed | 2023-05-09T20:45:29Z |
 4m3s     | 0.0025 USD |
| N8JDsSywHuKa3c7s8FZrl | addList     | admin | Completed | 2023-05-09T20:45:29Z |
 3m59s    | 0.0024 USD |
| 4t9qV52z8B7423YtfLbve | addList     | admin | Completed | 2023-05-09T20:45:29Z |
 3m59s    | 0.0024 USD |
| HsBFzGvPvhfekjFPSJ97A | addList     | admin | Completed | 2023-05-09T20:45:29Z |
 4m5s     | 0.0025 USD |
(edited)
```

Run Cromwell in Server Mode

When used in run mode, Cromwell launches a single workflow from the command line. Run mode is typically used for development. In server mode mode, Cromwell starts a web server that supports a feature-rich REST API. By default, the web server is started on the local host (0.0.0.0) and port 8000 (these can be changed in the configuration file).

To start a Cromwell server on a local Cromwell Host, perform the following steps.

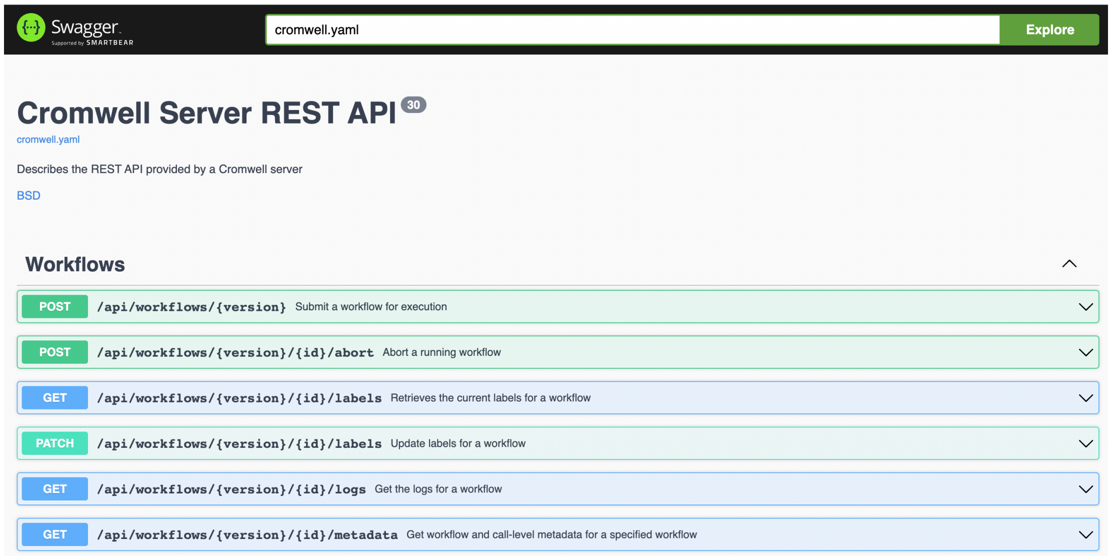
- On the Cromwell Host, enter the following command.

```
java -Dconfig.file=cromwell-float.conf -jar cromwell-85.jar server
```

- Check that the Cromwell Host's inbound filtering rules allow access to port 8000
- Open a browser and go to **`http://<cromwell_host_ip>:8000`** where **`<cromwell_host_ip>`**is the public (private) IP address of the Cromwell Host if you are outside (inside) the VPC.

To submit a job using the Cromwell web interface, complete the following steps.

- Expand the *Workflows* section and then click on *Submit a workflow for execution*



Cromwell Server REST API 30

cromwell.yaml

Describes the REST API provided by a Cromwell server

BSD

Workflows

POST /api/workflows/{version} Submit a workflow for execution

POST /api/workflows/{version}/{id}/abort Abort a running workflow

GET /api/workflows/{version}/{id}/labels Retrieves the current labels for a workflow

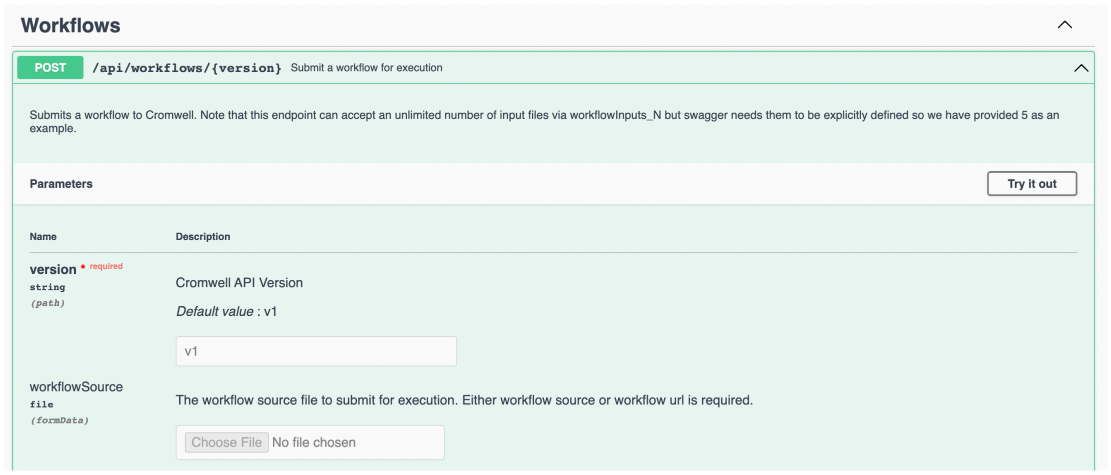
PATCH /api/workflows/{version}/{id}/labels Update labels for a workflow

GET /api/workflows/{version}/{id}/logs Get the logs for a workflow

GET /api/workflows/{version}/{id}/metadata Get workflow and call-level metadata for a specified workflow

Figure 10. Cromwell Server Interface

- Click on *Try it out* (on the right-hand side)



Workflows

POST /api/workflows/{version} Submit a workflow for execution

Submits a workflow to Cromwell. Note that this endpoint can accept an unlimited number of input files via workflowInputs_N but swagger needs them to be explicitly defined so we have provided 5 as an example.

Parameters

Try it out

Name	Description
version * required string *(path)*	Cromwell API Version *Default value* : v1 v1
workflowSource file *(formData)*	The workflow source file to submit for execution. Either workflow source or workflow url is required. Choose File No file chosen

Figure 11. Cromwell Server Interface

- Browse your local computer for a `workflowSource` wdl file
- Browse your local computer for a `workflowInputs` json file
- Browse your local computer for a `workflowOptions` json file
- Click on `Execute` (at the bottom of the section). If the job is accepted, the server returns a code 201.
- Copy the workflow ID
- Click on `Get the outputs of a workflow`
- Click on `Try it out` and then paste the workflow ID into the `id` box
- Click on `Execute`

Troubleshooting

As the Cromwell workflow runs, detailed log messages are written to the terminal.

MMCloud CLI

Introduction

MMCloud includes a CLI for interacting with the OpCenter. The OpCenter web interface provides a link to download the CLI binary for a Linux, macOS, or Windows clients. The CLI can also be used from a web console launched from the OpCenter web interface.

This part of the MMCloud User Guide provides a detailed command reference for the CLI.

CLI Command Reference

Use the **`float`** CLI commands to interact with the OpCenter; for example, to submit and manage jobs.

Version

The MMCloud CLI commands described here are consistent with the OpCenter release shown in the table.

MMCloud CLI	OpCenter Release	Date
v2.2.1-4bfffdf-ElNido	v2.2.1-4bfffdf-ElNido	2023-06-08T01:20:43Z

Global Flags

Use global flags with any **`float`** command or subcommand.

Flag	Usage	Definition
-a, --address **`ip_address`**	Connect to OpCenter server	IP address of OpCenter (default: localhost or last IP address used)
-F, --format **`json|yaml|table`**	Specify format for output	Output format (default: yaml)
-h, --help	Display help	Help for MMCloud CLI
--logLevel **`log_level`**	Specify log level	Log level (default: info)
-p, --password **`password`**	Log in to OpCenter	Login password
-u, --username **`user_name`**	Log in to OpCenter	Login username
-v, --verbose **`on|off`**	Turn verbose mode on or off	Verbose mode setting (default: off)

float cancel

The **`float cancel`** command is the same as **`float scancel`**.

float completion

Use the **`float completion`** command to generate auto-completion script for use in current shell or in every shell started subsequently. Using a subcommand with the **`-h`** flag displays help on how to use the auto-completion script.

Subcommands	Usage	Option	Option Definition
`bash`	Generate auto-completion script for bash	--no-descriptions	Disable completion descriptions

fish	Generate auto-completion script for fish	--no-descriptions	Disable completion descriptions
powershell	Generate auto-completion script for powershell	--no-descriptions	Disable completion descriptions
zsh	Generate auto-completion script for zsh	--no-descriptions	Disable completion descriptions

float config

Use the **`float config`** command to view or change the OpCenter configuration. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
get	Show runtime value of a single configuration parameter	`config_parameter`	Configuration parameter to display, e.g., sessionTimeout
list, **ls**	List configuration parameters for OpCenter (shows if a parameter can be changed and if a change requires a system restart)	-s, --scope `filter_string`	String to filter the list of configuration parameters (default: list all parameters)
set	Set runtime value of a single configuration parameter	`config_parameter parameter_value`	Configuration parameter and the value to set it to. If the word "default" is used for `parameter_value`, the configuration parameter is reset to its default value.

Examples

-
  ```
  float config get log.level
  info
  ```

-
  ```
  float config set sessionTimeout 1h
  sessionTimeout is set to 1h0m0s
  float config set sessionTimeout default
  sessionTimeout is set to 30m0s
  ```

float df

Use the **`float df`** command to display the file systems mounted by an executing job (includes used and available disk space). The linux command **`df`** must be installed in the container. This command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Display mounted file systems and associated disk space.	--args `df_args`	Arguments to pass to **`df`** command
		-j, --jobId `job_id`	Job to query

Example

```
float df --args "-h" -j WIY92p0jWyaCMP0CNQYjC
Filesystem                                                     Size  Used Avail Use%
 Mounted on
overlay                                                        6.0G  1.1G  5.0G
 17% /
/dev/nvme3n1                                                    10G  105M  9.9G
 2% /data
tmpfs                                                           64M     0   64M
 0% /dev
172.31.81.17:/mnt/memverge/slurm/work/nzG6oM5DoLCysXAkoZFCA/app  50G   20G   31G
 39% /mmce
/dev/nvme2n1                                                   6.0G  1.1G  5.0G
 17% /etc/hosts
/dev/nvme0n1p2                                                  40G  7.5G   33G
 19% /opt/aws
shm                                                             63M     0   63M
 0% /dev/shm
devtmpfs                                                       1.7G     0  1.7G
 0% /proc/key
```

float gateway

Use the **`float gateway`** command to create and manage a reverse proxy server. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
`connect`	Connect a running job to the gateway (reverse proxy)	-g, --gateway `gw_id`	ID of gateway to connect job to (format is g- followed by a fixed-length character string)
		-j,--job `job_id`	ID of job to connect to gateway to
		--targetPort `port_number`	Port used to connect to job, for example, 8787 for RStu-

			dio. Include --targetPort multiple times to connect multiple ports from a single server to the gateway
`create`	Create a gateway	--bandwidth | `bw`	Minimum gateway bandwidth in Mbps (default: 25). Only use with AliCloud.
		-c,--cpu `min_cpu`	Minimum number of virtual CPUs for gateway (default: 2)
		-t,--instType `instance_type`	VM instance type for gateway (e.g., c4.xlarge for AWS or ecs.g7.6xlarge for AliCloud). Do not combine with --cpu or --mem options.
		-m,--mem `min_mem`	Minimum memory capacity in GB for gateway (default: 4)
		--noPublicIP	Create gateway with private IP address only. Ensure that gateway is reachable inside VPC.
		--portRange `min:max`	Range of client-side ports opened on gateway (allowed range is between 10000 and 65535).
		--securityGroup `sec_group`	Security group applied to gateway. Use format sg-xxxxx. Include option multiple times to apply multiple security groups.
		-z, --zone `availability_zone`	Availability zone in which to create gateway
`destroy`	Destroy a gateway	-g, --gateway `gw_id`	ID of gateway to destroy (format is g- followed by a fixed-length character string)
		-f, --force	Automatically answer "yes" at confirmation prompt
`disconnect`	Disconnect a job from a gateway	-g, --gateway `gw_id`	ID of gateway to query (format is g- followed by a fixed-length character string)
		-j, --job `job_id`	ID of job to disconnect from gateway

		--port `port_num`	Server-side port to disconnect from gateway. If gateway connects to multiple ports on server, include --port for each port.
`info`	Display information about a gateway (including IP address and connected jobs)	-g, --gateway `gw_id`	ID of gateway to query (format is g- followed by a fixed-length character string)
`list`	List all running gateways (optionally include stopped gateways)	-A, --showAll	If option included, list stopped as well as running gateways
`modify`	Modify attributes of a gateway	--addSecurityGroup `sec_group`	Security group added to gateway. Use format sg-xxxxx. Include option multiple times for multiple security groups.
		-g, --gateway `gw_id`	ID of gateway to modify
		--rmSecurityGroup `sec_group`	Security group removed from gateway. Use format sg-xxxxx. Include option multiple times for multiple security groups.

Example

```
float gateway create --portRange 11000:11500
id: g-41X0xaJyeCaSQC7pcTKln
status: Creating
configuration: ""
publicIP: ""
portRange: 11000-11500
startTime: ""
numberOfJobs: 0
cost: ""
clientJobs: {}
```

float group

Use the **`float group`** command to manage OpCenter user groups. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
`add`, **`create`**	Create a new group	`new_group`	Name of new group

		--admin **user1**, **user2**...	Username(s) of admins of new group
		--user **user1**, **user2**...	Username(s) added to new group
delete, **remove**, **rm**	Delete a group	**group_name** | **group_id**	Name or id of group to delete
info, **show**	Display information about group including members	**group_name** | **group_id**	Name or id of group to query
list, **ls**	List groups that current user belongs to		
update	Update attributes of a group	**group_name** | **group_id**	Name or id of group to update
		--add **user1**,**user2**...	Username(s) added to group
		--admin **user1**,**user2**...	Username(s) given admin role in group
		--name **group_name**	New name for group
		--remove **user1**,**user2**...	Username(s) removed from group

Example

```
float group add team --user tester
name: team
gid: 3
admins: ""
users: tester
```

float hosts

The **`float hosts`** command is the same as **`float sinfo`**.

float image

Use the **`float image`** command to manage container images on OpCenter. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
add	Add image pull information to OpCenter	**image_name image_uri**	Image name and URI to pull image from repository.

			Do not use `image_uri` if --link specified.
		--import-all	Import all tags for this image from repository
		--link `file_link`	Link to access AWS S3 or AliCloud OSS file, e.g., `s3://<bucket_name>/file_path` Do not use if `image_uri` specified.
		--token `repo_access_token`	Token to pull image from private repository
		--user `repo_access_user`	Username to pull image from private repository
`cache`	Add (delete) image to (from) cache configured for OpCenter (NFS-mounted directory or S3 bucket)	`image_name`	Container image to cache or delete
		-d, --delete	Delete image from cache
		-f, --force	Overwrite existing cached version of image
		--tag `tag_string`	Tag to select specific container image (default: "latest")
`certify`	Validate that workload runs using OpCenter	`image_name`	Container image to validate
		--allowList `instance_type`	Allowed instance type. Use format like `"c5*"` which allows any VM type in the c5 family. Include --allowList multiple times to add other VM instance types to the allow list.
		--bandwidth `bandwidth`	Bandwidth (in Mbps) required for workload (only used in AliCloud)
		-A, --cmdArgs `command_string>`	Commands that are executed immediately after container starts (can be

		used with or without a job script)
	-c, --cpu `min_cpu`:`max_cpu`	Range of number of virtual CPUs to select from to run job (can omit `max_cpu`).
	--cpuVendor `cpu_vendor_name`	CPU vendor of allowed VM instances. Allowed values are amd, intel, or "". Default is "" which allows any vendor. Use with AWS only.
	--customTag `tagName:tagValue`	Tag customized by each MMCloud subscriber, for example, to generate customized usage reports. Multiple custom tags can be applied by using --customTag multiple times in a single command.
	--dataVolume `[size=vol_size,throughput=rate]:/data_dir` or `vol_id:/container_mnt_pt/path/to/dir` or `nfs://ip_address/server_export_dir:` `/container_mnt_pt/path/to/dir` or `[accesskey=x,secret=y,endpoint=z,mode=rw]` `s3://bucketname/path:` `/container_mnt_pt//path/to/dir`	Data volume mounted by container. `vol_size` in GB, `rate` in Mbps (can omit), `data_dir` is data directory created, or `vol_id` specifies existing EBS volume, `/container_mnt_pt/path/to/dir` is path to where EBS volume is mounted on container, or `ip_address` is IP address of NFS server, `/server_export_dir` is directory exported by NFS server, and `/container_mnt_pt/path/to/dir` is path to where NFS directory is mounted on container, or `bucketname` is S3 bucket (access credentials optional if action allowed by IAM

	role). For oss, replace `s3` with `oss`.
-d, --def `definition_file`	Path to file if using definition file (yaml or json format) to provide input parameters to `certify` command.
-e, --env `env_key=env_value`	Environment variable setting for the job. Include option multiple times if needed.
--extraOptions `extra_opt`	Container-specific string (enclose in quotes) passed to checkpoint process (needed for some containers)
-f, --force	Automatically answer "yes" at confirmation prompt
--gateway `gateway_id`	ID of gateway to connect job to. Replacing `gateway_id` with the word "auto" directs the OpCenter to select a gateway automatically.
--ignoreRebalance	Ignore rebalance recommendation signal (only applies to AWS)
--imageVolSize `image_vol_size`	Image volume size in GB to load image (default: 6)
-t, --instType `instance_type`	VM instance type (e.g., c4.xlarge for AWS or ecs.g7.6xlarge for AliCloud). Do not combine with --cpu or --mem options.
--interval `job_interrupt_interval`	Interval between job interruptions in format *hh***h***mm***m***ss***s** where **h**, **m**, and **s** are hours, minutes, and seconds, respectively (default: 10m)

	-j, --job `job_script`	Job script to run workload. Use format: path to local file, s3 file, OSS file or https | http URL.
	--mem `min_mem`:`max_mem`	Range of memory size (in GB) to select from to run job (can omit `max_mem`).
	--metricsInterval `metrics_int`	Time in seconds between queries to obtain container metrics (default: 10s)
	--migratePolicy `[migrate_policy]`	Policy to determine auto-migration behavior. Format is `[option1=value1, option2=value2...]`. The available options with their default values are (if no units are attached, the value is a percentage): • enable=false • disable=true (use enable or disable, not both) • evadeOOM=true • stepAuto=false • cpu.upperBoundRatio=90 • cpu.lowerBoundRatio=5 • cpu.upperBoundDuration=30s • cpu.lowerBoundDuration=5m0s • cpu.step=50 • cpu.limit=0 (no limit) • cpu.lowerLimit=0 (no limit) • mem.upperBoundRatio=90 • mem.lowerBoundRatio=5

	• mem.upperBoundDuration=30s • mem.lowerBoundDuration=5m0s • mem.step=50 • mem.limit=0 (no limit) • mem.lowerLimit=0 (no limit)
-n, --name `job_name`	Name to associate with job
--noPublicIP	No public IP address assigned to container host. Ensure host reachable inside VPC. AliCloud users must include `--endpoint` option in their job scripts.
-P, --publish `host_port:container_port`	Rule for publishing host port to container port, for example, `8080:80`. Include option multiple times to publish multiple ports.
--rootVolSize `root_vol_size`	Root volume size in GB to load base OS (default: 40)
--securityGroup `sec_group`	Security group added to VM instance for this job. Use format sg-xxxxx. Include option multiple times to apply multiple security groups.
--shmSize `shm_size`	Size of `/dev/shm` in format *n***u** where *n* is a number and **u** is b, k, m, or g for bytes, KiB, MiB or GiB, respectively (default: 64m)
-I, --snapshotInterval `snap_int`	Time between periodic snapshots in format *hh***h***mm***m***ss***s** where **h**, **m**, and **s** are hours, minutes, and seconds, respectively. Default is 0 (not enabled).

		--tag `tag_name`	Tag to select image version. Default is "latest" or only tag.
		--targetPort `port_num`	Server-side gateway port to connect job to. Use option multiple times to connect multiple ports.
		-T, --template `template_name`	Template in format `name:tag` to use for submitting this job. Default tag is only tag or "latest".
		-l, --timeLimit `max_time`	Maximum time that a job is allowed to run. Use format *hh***h***mm***m***ss***s** where **h**, **m**, and **s** are hours, minutes, and seconds, respectively. Default is unlimited (use 0).
		--vmPolicy `[vm_policy]`	VM creation policy. Format is `[key1=value1,key2=value2...]-`. For example, • [spotOnly=true,retryLimit=10, retryInterval=30s] • [spotFirst=true,retryLimit=3,optimize=true] • [onDemand=true,optimize=true] • [spotOnly=true,priceLimit=0.1]
		--withRoot	Run job with root privileges
		-z, --zone `availability_zone`	Availability zone in which to execute job
`delete`, **`rm`**, **`remove`**	Delete container image(s) and one or more tags	`image1 image2...` or	Container image(s) to delete. If no tags are spec-

		image1:tag1 image2:tag2...	ified, then all tags are removed.
		-f, --force	Automatically answer "yes" at confirmation prompt
		--tag tag_name	Tag associated with image1 image2... to remove. If this is the only tag associated with images, images are removed.
list, **ls**	Show all images available on OpCenter		
tags, **tag**	Display tags associated with image and image status	image_name	Container image name
update	Update information associated with image	image_name	Container image name
		--addtag image_tag1, image_tag2...	Additional tags to associate with image
		--name image_name	New name to identify image
		--token repo_access_token	New token required to access private repository
		--user repo_access_user	New username associated with token required to access private repository
upload	Load image from local server	image_name	Container image name
		--path /path/to/image	Path to where image is located.

Examples

-
  ```
  float image list
  +------------+------------------------------+--------+-------------+
  |    NAME    |             URI              | TAGS   | ACCESS USER |
  +------------+------------------------------+--------+-------------+
  | python     | docker.io/bitnami/python     | latest |             |
  | r-base     | docker.io/rocker/r-base      | latest |             |
  .....(edited)
  +------------+------------------------------+--------+-------------+
  ```

-
  ```
  float image delete r-base
  Deleting r-base
  ```

-
  ```
  float image add r-base docker.io/rocker/r-base
  name: r-base
  uri: docker.io/rocker/r-base
  owner: admin
  tags:
      latest:
          status: Available
          locked: false
          lastUpdated: 2023-06-15T16:12:41.739947963Z
          size: Unknown
  ```

-
  ```
  float image upload test2 --path /tmp/hello-world-latest.tar
  Start uploading image test2 (localhost/hello-world:latest)
   from /tmp/hello-world-latest.tar
  Progress: |>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>| 100.00%
   Complete (ETA. 0s)
  Uploaded image /tmp/hello-world-latest.tar, time spent: 0s

  name: test2
  uri: localhost/hello-world
  owner: admin
  tags:
      latest:
          status: Ready
          uri: file:///mnt/memverge/images/test2-latest.tar
          locked: false
          lastUpdated: 2023-04-17T19:26:38.584557866Z
          lastPushed: 2023-04-17T19:26:38.584557866Z
          size: Unknown
  ```

float license

Use the **`float license`** command to manage OpCenterlicenses. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
`acquire`	Acquire license from MemVerge account server	--A, --account `acct_name`	Username (email address) to access MemVerge account server
		-P, --password `passwd`	Password to access MemVerge account server
`info`	Display license information and status		
`upgrade`	Upgrade license from MMCloud		

	Essentials to MM-Cloud Pro		

float list

The **`float list`** command is the same as **`float squeue`**.

float log

Use the **`float log`** command to view and manage log files. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	**Usage**	**Option**	**Option Definition**
`cat`	Write log file contents to standard output	`log_file`	Log file whose contents are displayed
		-i, --hid `host_id`	Host whose logs are displayed (default: OpCenter server)
		-j, --job `job_id`	Job whose logs are displayed (default: OpCenter)
`list`, **`ls`**	List all logs associated with target	-i, --hid `host_id`	Host whose log files are listed (default: OpCenter server)
		-j, --job `job_id`	Job whose log files are listed (default: OpCenter)
`rm`	Remove all logs associated with target	-i, --hid `host_id`	Host whose log files are removed (default: OpCenter server)
		-j, --job `job_id`	Job whose log files are removed (default: OpCenter)
`tail`	Write last *n* lines of log file to standard output	`log_file`	Log file to display
		-f, --follow	Display new lines as they are appended to log file
		-i, --hid `host_id`	Host whose log file lines are displayed (default: OpCenter server)
		-j, --job `job_id`	Job whose log file lines are displayed (default: OpCenter)
		-n, --num `n`	Number of lines to display (default: 100)

Examples

-
  ```
  float log tail --follow output -j XGiUDRto7kwofWBNPkiW5
  Ready to prepare source data
  Ready to download pbmc_1k_v3_fastqs from s3
  Ready to download refdata-gex-GRCh38-2020-A from s3
  Ready to run test
  ...[output edited]
  ```

-
  ```
  float log ls
  ```
  ```

  +---------------------------------------+----------+----------------------+
  |               LOG NAME                |   SIZE   |   LAST UPDATE TIME   |
  +---------------------------------------+----------+----------------------+
  | etcd.log                              |   647667 | 2022-11-02T15:39:08Z |
  | opcenter.access_log                   |   912282 | 2022-11-02T15:40:55Z |
  | opcenter.log                          |  5678863 | 2022-11-02T15:51:24Z |
  | upgrade.log                           |      500 | 2022-10-03T16:03:24Z |
  +---------------------------------------+----------+----------------------+
  ```

float login

Use the **`float login`** command, with valid username and password, to log in to OpCenter. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Log in to OpCenter	--info	Display login status

Examples

```
float login -u admin -p memverge -a 192.168.0.1
```

```
float login --info
address: 192.168.0.1
username: admin
role: admin
```

```
float login
Username: admin
Password:
Login Succeeded!
```

float logout

Use the **`float logout`** command to log the current user out of the OpCenter and invalidate the authorization token. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

float migrate

Use the **`float migrate`** command to move a job from one VM instance to another VM instance of the same type or of a different type. The command has no subcommands. Using the command with the **`-h`** flag displays the options.

Subcommands	Usage	Option	Option Definition
	Migrate job to a new VM instance. If no options used, migrate to identical instance in same availability zone.	--cpu `min_cpu`:`max_cpu`	Range of number of virtual CPUs to select from for new VM instance (can omit `max_cpu`).
		-f, --force	Automatically answer "yes" at confirmation prompt
		-t, --instType `instance_type`	VM instance type to migrate to (e.g., c4.xlarge in AWS or ecs.g7.6xlarge in AliCloud). Do not combine with --cpu or --mem options.
		-j, --job `job_id`	Job to migrate
		--mem `min_mem`:`max_mem`	Range of memory size (in GB) to select from for new VM instance (can omit `max_mem`).
		-P, --payType `spot | ondemand`	Pricing tier for VM instance (Spot or On-demand)
		--sync	Block all terminal input until job has migrated
		-z, --zone `availability_zone`	Availability zone in which to execute job

Examples

-
  ```
  float migrate -t c5.xlarge -j NDt428IsJtJZsB9WNUhGH
  ```

-
  ```
  float migrate -j lL07E84pQQpYqCQ88xeIQ
  entity:
    id: i-0cfbd3f1f82087dd5
    type: host
    name: 192.168.0.2
  status: normal
  instanceType: c5.xlarge
  ```

```
startTime: "2022-09-23T15:09:09Z"
downTime: ""
```

-
  ```
  float migrate -f --sync --payType ondemand -j tqrGc4Z6g18nkphzTeaxM
  tqrGc4Z6g18nkphzTeaxM is now migrating...
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

  tqrGc4Z6g18nkphzTeaxM has been migrated to 44.212.92.162 (4Core16GB/OnDemand).
  ```

float modify

Use the **`float modify`** command to change a subset of attributes associated with a running job. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Modify certain attributes associated with running job	--addSecurityGroup **`sec_group`**	Security group added to VM instance for this job. Use format sg-xxxxx. Include option multiple times to add multiple security groups.
		--force	Automatically answer "yes" at confirmation prompt
		-j, --job **`job_id`**	Job to apply changes to
		-M, --migratePolicy **`migrate_policy`**	New migrate policy to apply. See **`float sbatch`** for format.
		--rmSecurityGroup **`sec_group`**	Security group to remove from VM instance for this job. Use format sg-xxxxx. Include option multiple times to remove multiple security groups.
		--snapshotInterval **`snapshot_interval`**	New periodic snapshot interval for this job. Use "-1" or "disable" to turn off periodic snapshots.
		-V, --vmPolicy **`vm_policy`**	New VM creation policy to apply. See **`float sbatch`** for format.

Example

```
float modify --vmPolicy [SpotOnly=true] -j PF0bgCvlpdJkog0RCBZPg
Warning: Are you sure you want to modify PF0bgCvlpdJkog0RCBZPg?
New vmPolicy may trigger migration.(yes/No): yes
Successfully modified PF0bgCvlpdJkog0RCBZPg:  --vmPolicy [SpotOnly=true]
```

float ps

Use the **`float ps`** command to show the complete process tree of a running job (the linux command **`ps`** must be installed in the container). The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Show the complete process tree of job	--args `podman_args`	Arguments passed to **`podman`**
		-j, --jobId `job_id`	Job to query

float release

Use the **`float release`** command to manage the OpCenter software. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
`info`	Display information about OpCenter release	-r, --release `version`	The release to display information about
`list`, `ls`	List available OpCenter releases		
`sync`	Sync CLI version with OpCenter version		
`upgrade`	Upgrade OpCenter software	--force	Automatically answer "yes" at confirmation prompt
		-r, --release `version`	The release to upgrade to (default: "latest"). Only the *admin* user can upgrade software. Cannot upgrade using web CLI console.
		--sync	Wait for upgrade to complete and then sync the CLI to the new release

Examples

-
  ```
  float release ls
  +----------+-----------------------------------+-----------------------+--------
  ---+
  | VERSION  |              RELEASE              |     RELEASE TIME      |   SIZE
    |
  ```

```
+----------+----------------------------------+----------------------+--------
---+
| * v2.2.1 | FLOAT_v2.2.1-4bfffdf-ElNido.bin | 2023-06-08T01:20:43Z | 153.55 MB
 |
| v2.2.0   | FLOAT_v2.2.0-1363a1b-ElNido.bin | 2023-05-23T14:10:47Z | 153.55 MB
 |
+----------+----------------------------------+----------------------+--------
---+
```

-
  ```
  float release upgrade -r FLOAT_v2.2.1-4bfffdf-ElNido.bin --force
  ```

-
  ```
  float release sync
  downloading ...
  Progress: |>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>|
   100.00% Complete (ETA. 0s)
  The float binary is synced up with opcenter
  ```

float report

Use the **`float report`** command to download reports from OpCenter. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
`download`	Download OpCenter usage history report	--path `/path/to/dir`	Path to save report on local computer
`get`	Generate and display usage report with filter(s) applied	`report_name`	Name of report, e.g., usage_report_by_job
		-A, --all	Include all jobs (normal plus archived) in the report. Default: normal jobs only. Jobs archived after one hour of inactivity.
		-d, --date `date_string`	Date used to filter reports (default: current report)
		--filter `filter1 filter2`...	Filters to apply to usage report, for example: status=executing timeRange=2010-10-22~
		-l, --limit `num_reports`	Maximum number of reports to get (default: unlimited)
		-o, --orderBy `field1`,`field2`	Fields to order by. Only 'cost' and 'jobs' (default) are supported.

		-r, --refresh	Force the refresh of report
ls, **list**	List all available usage reports		

Examples:

```
float report get usage_report_by_app -A -f status=Completed
+----------------+-----------+-----------+-------------+---------------+------------
--------+-------------------+-----------------+---------------+-----------+-------
------+
|   IMAGE NAME   | JOB COUNT | WALL TIME | COMPUTE TIME | SPOT INSTANCES | ONDEMAND
 INSTANCES |      SAVINGS       | BASELINE COMPUTE | ACTUAL COMPUTE |  STORAGE   | APP
 CAPSULE |
+----------------+-----------+-----------+-------------+---------------+------------
--------+-------------------+-----------------+---------------+-----------+-------
------+
| image-yittpr   |        39 | 2h30m22s  | 2h29m40s     |             39 |
      0 | 0.3445 USD(58.76%) | 0.5862 USD       | 0.2418 USD     | 0.0014 USD | 0.0014
 USD  |
| cactus         |        23 | 1h25m30s  | 1h25m4s      |             23 |
      0 | 0.0515 USD(43.72%) | 0.1179 USD       | 0.0663 USD     | 0.0009 USD | 0.0009
 USD  |
| rnaseq-nf      |         6 | 50m24s    | 50m17s       |              6 |
      0 | 0.0045 USD(12.97%) | 0.0349 USD       | 0.0303 USD     | 0.0007 USD | 0.0007
 USD  |
  ...
(edited)
```

```
float report download --path ./temp.gzip
Downloaded to ./temp.gzip
file temp.gzip
temp.gzip: gzip compressed data, original size modulo 2^32 14336
```

float rerun (requeue, resubmit)

Use the **float rerun** (or **requeue** or **resubmit**) command to re-submit a completed job or a job that failed to complete. The command has no subcommands. Using the command with the **-h** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Re-submit completed or failed job	-j, --job **job_id**	Job to re-submit

float restart (reboot)

Use the **float restart** (or **reboot**) command to restart OpCenter (terminates all login sessions). The command has no subcommands. Using the command with the **-h** flag lists the options.

Subcommands	Usage	Option	Option Definition

	Restart OpCenter	-f, --force	Automatically answer "yes" at confirmation prompt

float sbatch (submit)

Use the **`float sbatch`** (or **`float submit)`**command to submit a job. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Submit job for execution	--allowList **`instance_type`**	Allowed instance type. Use format like **`"c5*"`** which allows any VM type in the c5 family. Include --allowList multiple times to add other VM instance types to the allow list.
		--bandwidth **`bandwidth`**	Bandwidth (in Mbps) required for workload (only used in AliCloud)
		-A, --cmdArgs **`command_string>`**	Commands that are executed immediately after container starts (can be used with or without a job script)
		-c, --cpu **`min_cpu`**:**`max_cpu`**	Range of number of virtual CPUs to select from to run job (can omit **`max_cpu`**)
		--cpuVendor **`cpu_vendor_name`**	CPU vendor of allowed VM instances. Allowed values are amd, intel, or "". Default is "" which allows any vendor. Use with AWS only.
		--customTag **`tagName:tagValue`**	Tag customized by each MMCloud subscriber, for example, to generate customized usage reports. Multiple custom tags can be applied by using --customTag multiple times in a single command.
		--dataVolume **`[size=vol_size,throughput=rate]:/data_dir`** or	Data volume mounted by container.

`vol_id:/container_mnt_pt/path/to/dir` or `nfs://ip_address/server_export_dir`: `/container_mnt_pt/path/to/dir` or `[accesskey=x,secret=y,endpoint=z,mode=rw]` `s3://bucketname/path`: `/container_mnt_pt/path/to/dir`	`vol_size` in GB, `rate` in Mbps (can omit), `data_dir` is data directory created, or `vol_id` specifies existing EBS volume, `/container_mnt_pt/path/to/dir` is path to where EBS volume is mounted on container, or `ip_address` is IP address of NFS server, `/server_export_dir` is directory exported by NFS server, and `/container_mnt_pt/path/to/dir` is path to where NFS directory is mounted on container, or `bucketname` is S3 bucket (access credentials optional if action allowed by IAM role). For oss replace `s3` with `oss`.
-d, --def `definition_file`	Path to file if using definition file (yaml or json format) to provide input parameters to `sbatch` command
-e, --env `env_key=env_value`	Environment variable setting for the job. Include --env multiple times if needed.
-f, --force	Automatically answer "yes" at confirmation prompt
--gateway `gateway_id`	ID of gateway to connect job to. Replacing `gateway_id` with the word "auto" directs the OpCenter to select a gateway automatically.
--ignoreRebalance	Ignore rebalance recommendation signal (only applies to AWS)
-i, --image `image_name`	Image name or image URI to create container for job
--imageVolSize `image_vol_size`	Image volume size in GB to load image (default: 6)

	-t, --instType `instance_type`	VM instance type (e.g., c4.xlarge for AWS or ecs.g7.6xlarge for AliCloud). Do not combine with --cpu or --mem options.
	-j, --job `job_script`	Job script to run workload. Use format: path to local file, s3 file, OSS file or https | http file.
	--mem `min_mem`:`max_mem`	Range of memory size (in GB) to select from to run job (can omit `max_mem`).
	--metricsInterval `metrics_int`	Time in seconds between queries to obtain container metrics (default: 10s)
	--migratePolicy `[migrate_policy]`	Policy to determine auto-migration behavior. Format is `[option1=value1, option2=value2...]`. The available options with their default values are (if no units are attached, the value is a percentage): • enable=false • disable=true (use enable or disable, not both) • evadeOOM=true • stepAuto=false • cpu.upperBoundRatio=90 • cpu.lowerBoundRatio=5 • cpu.upperBoundDuration=30s • cpu.lowerBoundDuration=5m0s • cpu.step=50 • cpu.limit=0 (no limit) • cpu.lowerLimit=0 (no limit) • mem.upperBoundRatio=90 • mem.lowerBoundRatio=5

	• mem.upperBoundDuration=30s • mem.lowerBoundDuration=5m0s • mem.step=50 • mem.limit=0 (no limit) • mem.lowerLimit=0 (no limit)
-n, --name `job_name`	Name to associate with job
--noPublicIP	No public IP address assigned to container host. Ensure that host is reachable inside the VPC. AliCloud users must include `--endpoint` option in their job scripts.
-P, --publish `host_port:container_port`	Rule for publishing container port to container host port, for example, `8080:80`. Include option multiple times to publish multiple ports.
--rootVolSize `root_vol_size`	Root volume size in GB to load base OS (default: 40)
--securityGroup `sec_group`	Security group added to VM instance for this job. Use format sg-xxxxx. Use option multiple times to apply multiple security groups.
--shmSize `shm_size`	Size of `/dev/shm` in format *n***u** where *n* is a number and **u** is b, k, m, or g for bytes, KiB, MiB or GiB, respectively (default: 64m)
-I, --snapshotInterval `snap_int`	Time between periodic snapshots in format *hh***h***mm***m***ss***s** where **h**, **m**, and **s** are hours, minutes, and seconds, respectively
--tag `tag_name`	Tag to select image version for job. Default is "latest" or only tag.

	--targetPort `port_num`	Port to connect job to on gateway. Use option multiple times to connect multiple ports.
	-T, --template `template_name`	Template in format `name:tag` to use for submitting this job. Default tag is only tag or "latest".
	-l, --timeLimit `max_time`	Maximum time that a job is allowed to run. Use format *hh***h***mm***m***ss***s** where **h**, **m**, and **s** are hours, minutes, and seconds, respectively. Default is unlimited (use 0).
	--vmPolicy `[vm_policy]`	VM creation policy. Format is `[key1=value1,key2=value2...]`. For example, • [spotOnly=true,retryLimit=10, retryInterval=30s] • [spotFirst=true,retryLimit=3,optimize=true] • [onDemand=true,optimize=true] • [spotOnly=true,priceLimit=0.1]
	--withRoot	Run job with root privileges
	-z, --zone `availability_zone`	Availability zone in which to execute job

Example

```
float sbatch -i tidyverse -j run_genericr.sh --cpu 4 --mem 8 --dataVolume [size=10]:/data
id: NDt428IsJtJZsB9WNUhGH
name: tidyverse-(worker)
workingHost: ""
usr: worker
status: submitted
submitTime: "2022-10-13T00:01:23Z"
```

float scancel (cancel)

Use the **`float scancel`** (or **`cancel`**) command to cancel a job. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Cancel job	--filter **filter1 filter2**...	Filter(s) to select jobs, for example: status=Executing timeRange=2010-10-22~
		-f, --force	Automatically answer "yes" at confirmation prompt
		-j, --job **job_id**	Job to cancel

Example

```
float scancel -j ctZLDo7OFG4BuJ8ytiTem
Warning: Are you sure you want to cancel this job?

ID: ctZLDo7OFG4BuJ8ytiTem
Name: python-c5d.large

All the related resources will be released. (yes/No): y
Request to cancel ctZLDo7OFG4BuJ8ytiTem has been submitted
```

float show

Use the **float show** command to display the status of a job or of a VM instance. The command has no subcommands. Using the command with the **-h** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Show current status of job or VM instance, or show contents of job script used for job	-c, --content	Display contents of job script
		-i, --hid **host_id**	VM instance to query
		-j, --job **job_id**	Job to query

Example

```
float show -j FOIW5Y16KZgJ6Tsd02QuS
id: ctZLDo7OFG4BuJ8ytiTem
name: python-c5d.large
workingHost: 52.7.123.178 (2Core4GB/Spot)
user: admin
imageID: docker.io/bitnami/python:latest
imageDigest: sha256:24c1d45bf41c396184bd9808b307c67267a809754cd176ac8d91cceb47d0f3ef
output: |-
    Getting image source signatures
    Copying blob
 sha256:1acb894a7ceb1ba5362fb85123b0248a064ed3195aaa24af74e8cb710ca1c5a4
```

```
    Copying config
 sha256:23bacce690702dac91557ef74ab312cb3db5a2b4bb54ada968d1352e7d9a110a
    Writing manifest to image destination
    Storing signatures
    Loaded image: docker.io/bitnami/python:latest
    First submit job ctZLDo7OFG4BuJ8ytiTem, call podman directly
    No cmd args provided, launch job directly
    4eda6b0d471fb31eecfd06b59e1d8c527310861009fa4baf967d834397934bac
status: Executing
.....[output edited]

float show -c -j S0JPnWd3a2hQmVmVWCMWc
#!/usr/bin/bash
LOG_PATH=$1
LOG_FILE=$LOG_PATH/output
touch $LOG_FILE
exec >$LOG_FILE 2>&1
echo "Congratulations! You have submitted your first job"
for(( c=1; c<3; c++))
do
        if [[ $(($c % 3)) == 1 ]] then
                echo "Hello World!"
        else
                echo "Your next job will be more interesting" >&2
        fi
                sleep 20s
done
echo "Job complete"
```

float sinfo (hosts)

Use the **float sinfo** (or **hosts**) command to show worker node details and current status. The command has no subcommands. Using the command with the **-h** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Show current worker node details including status	-A, --all	Show all jobs (normal plus archived). Default: normal only. Other records archived after one hour of inactivity.
		-f, --filter **filter1 filter2**...	Filter(s) to apply to job status. Example: status=executing timeRange=2010-10-22~

float snapshot

Use the **float snapshot** command to display information about snapshots. Using subcommand with the **-h** flag lists the options.

Subcommands	Usage	Option	Option Definition

`list`	List available snapshots	-A, --all	Show all snapshots associated with job
		-f, --filter **`filter1 filter2`**...	Filter(s) to apply to snapshots, for example: status=normal
		-j, --jobID **`job_id`**	Job to query
`show`, **`info`**	Display information about snapshot	-s, --snapID **`snapshot_id`**	Snapshot to query

Example:

```
float snapshot show -s jobSnap-Kp89d7qzJjiuhNeaOUfsA
id: jobSnap-Kp89d7qzJjiuhNeaOUfsA
jobID: h3xg4jpC8ydcgkVWBKj4S
volumeSnapshots:
    - zone: us-east-1b
      volumeId: vol-02786458cebdeca43
      volumeSize: 6
      status: completed
      snapshotId: snap-08e5b6ed8b7834b96
      createTime: "2023-04-18T15:51:17Z"
      mountPoint: /mnt/float-data
      cost: 0.0000 USD
    - zone: us-east-1b
      volumeId: vol-06dd42bf6b091466a
      volumeSize: 6
      status: completed
      snapshotId: snap-0f2833cdb639fa4b5
      createTime: "2023-04-18T15:51:17Z"
      mountPoint: /mnt/float-image
      cost: 0.0000 USD
    - zone: us-east-1b
      volumeId: vol-023c2e91e34f37335
      volumeSize: 10
      status: completed
      snapshotId: snap-025a733db664e9054
      createTime: "2023-04-18T15:51:17Z"
      mountPoint: /data
      cost: 0.0000 USD
status: normal
createTime: 2023-04-18T15:51:17.434161661Z
```

float squeue (list)

Use the **`float squeue`** (or **`list`**) command to show a filtered list of queued jobs. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
	Show filtered list of queued jobs (default:	-A, --all	Show all jobs (normal plus archived). Default: normal only.

	list jobs from oldest to newest)		Other records archived after one hour of inactivity.
		-d, --desc	List jobs in descending order using the attribute specified by **--orderBy**
		-f, --filter **filter1 filter2**...	Filter(s) to apply to list of queued jobs. Example: status=executing timeRange=2010-10-22~
		-o, --orderBy **attribute**	Attribute used to order job listing. Supported values are (default: start): • name • start • owner • image • status

Example

```
float squeue -A --filter status=Completed
```

```
+-----+------+------------+--------+---------+-----------+---------+------+
| ID  | NAME |WORKING HOST|  USER  |  STATUS |SUBMIT TIME|DURATION | COST |
+-----+------+------------+--------+---------+-----------+---------+------+
|FO...|cer...|            | worker |Completed| 2022-...  |2022...  |  0.. |
...[edited for clarity]
```

```
float squeue -o name
```

```
+--------+----------------------+------------------+-------+-----------+-----------
---+----------+-----------+
| ID     |        NAME          |    WORKING HOST  | USER  |  STATUS   | SUBMIT TIME
 | DURATION |    COST    |
+-------------------------------+------------------+-------+-----------+-----------
---+----------+-----------+
| SZd... | apple                | 54.163.154.116...| admin | Executing | ...19:11:47Z
 | 1h35m1s  | 0.1143 USD |
| btC... | banana               | 34.234.64.157... | admin | Executing | ...19:12:19Z
 | 1h34m30s | 0.0926 USD |
| hnP... | camel                |                  | admin | Completed | ...20:43:01Z
 | 3m28s    | 0.0031 USD |
| 5d1... | cherry               | 54.89.203.124... | admin | Executing | ...19:12:27Z
 | 1h34m22s | 0.0924 USD |
| grd... | python-t3a.xlarge    | 34.229.17.82...   | admin | Completed | ...20:43:48Z
 | 2m51s    | 0.0026 USD |
| nVD... | tidyverse-t3a.xlarge | 3.89.224.246...   | admin | Executing | ...19:14:39Z
 | 1h32m10s | 0.0903 USD |
+--------+----------------------+-----------------+-------+----------+-----------
---+----------+------------+
...[edited for clarity]
```

```
float squeue -f image=python -f status=Executing
```

```

+--------+------------------+-----------------------------------+-------+-----------+-
------------+----------+------------+
|   ID   |       NAME       |           WORKING HOST            | USER  |  STATUS   |
 SUBMIT TIME | DURATION |    COST    |
+--------+------------------+-----------------------------------+-------+-----------+-
------------+----------+------------+
| Xai... | python-c5.4xlarge | 34.228.10.73 (16Core32GB/Spot)   | admin | Executing
 |...20:38:32Z | 11m26s   | 0.0563 USD |
| bBF... | spinachpython     | 34.228.165.173 (16Core32GB/Spot) | admin | Executing
 |...20:43:26Z | 6m32s    | 0.0315 USD |
+--------+------------------+-----------------------------------+-------+-----------+-
------------+----------+------------+

...[edited for clarity]
```

float status

Use the **`float status`** command to show current status of the OpCenter. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Example

```
float status
id: bf11512f-7ada-403e-b749-525a990a2916
Server Status: normal
API Request Status: normal
License Status: valid
Init Time: "2023-05-26T15:37:58Z"
Up Time: "2023-06-12T20:00:57Z"
Current Time: "2023-06-15T19:15:46Z"
```

float submit

The **`float submit`** command is the same as **`float sbatch`**.

float template

Use the **`float template`** command to use a template o rto display information about templates. Using subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition

`deploy`	Submit a job using a template	-c, --cpu `min_cpu:max_cpu`	Range of number of virtual CPUs to select from to run job (can omit `max_cpu`)
		--gateway `gw_id`	ID of gateway to connect job to
		-t, --instType `instance_-type`	VM instance type (e.g., c4.xlarge for AWS). Do not combine with --cpu or --mem options.
		--mem `min_mem`:`max_mem`	Range of memory size (in GB) to select from to run job (can omit `max_mem`).
		--noPublicIP	No public IP address assigned to container host. Make sure host is reachable from within the VPC.
		--securityGroup `sec_group`	Security group added to VM instance for this job. Use format sg-xxxxx
		--targetPort `port_number`	Port used to connect to job, for example, 8787 for RStudio. Include --targetPort multiple times to connect multiple ports from a single server to the gateway
		-T, --template `template_-name`	Template in format `name:tag` to use for submitting this job. Default tag is only tag or "latest".
`info`, **`show`**	Display information about a template	-T, --template `template_-name`	Template in format `name:tag` to use to query. Default tag is only tag or "latest".
`list`, **`ls`**	Display available templates		

float top

Use the **`float top`** command to show a sorted list of information (including CPU and memory utilization) about current jobs. The display is updated at regular intervals. Enter `q` to stop the display. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Subcommands	**Usage**	**Option**	**Option Definition**

	Show utilization and other information about running jobs. Display updates continually. Enter `q` to exit.	--interval `top_int`	Time in seconds (3 or greater) between queries to obtain container metrics (default: 3s). Format is *n*s where *n* is integer.
		-j, --job_Id `job_id`	Job to query

float user

Use the **`float user`** command to manage OpCenter users. Using a subcommand with the **`-h`** flag lists the options.

Subcommands	Usage	Option	Option Definition
`add`, **`create`**	Add a new user	`new_username`	Username for new user
		--create	Create group if the group does not exist (default: false)
		--email `email_address`	Email address for new user
		-g, --group `group_name`	Group associated with new user (default: "default")
		--passwd `password`	Password for new user (default: "memverge")
`delete`, **`remove`**, **`rm`**	Delete a user	`user_name` | `user_id`	Username or id of user to delete
		-f, --force	Remove all active tokens belonging to user and delete user immediately
`info`, **`show`**	Display information about user	`user_name` | `user_id`	Username or id of user to query
`list`, **`ls`**	List groups that current user belongs to		
`passwd`	Reset user's password	`user_name` | `user_id`	Username or id of user to update
		--passwd `new_password`	New password for user
`update`	Update information associated with a user	`user_name` | `user_id`	Username or id of user to update

	--email `email_address`	New email address for user
	-g, --group `group_name`	New group associated with user
	--name `user_name`	New username for user
	--passwd `password`	New password for user

Example

```
float user add tester --passwd secret123
username: tester
uid: 5
role: normal
group: default
email: ""
ownGroup: ""
```

```
float user info tester
username: tester
uid: 5
role: normal
group: default
email: ""
ownGroup: ""
```

float version

Use the **`float version`** command to display the version of the `float` CLI client and the version of the OpCenter it is connected to. The command has no subcommands. Using the command with the **`-h`** flag lists the options.

Example

```
float version
float version: v2.2.1-4bfffdf-ElNido
OpCenter version: v2.2.1-4bfffdf-ElNido
```