Building and Provisioning Bioinformatics Environments on Public and Private Clouds

Enis Afgan[1,2,4], Konstantinos Krampis[3], Nuwan Goonasekera[4], Karolj Skala[1] and James Taylor[2]

[1]Center for Informatics and Computing (CIR)
Ruđer Bošković Institute (RBI)
Zagreb, Croatia

[2]Department of Biology
Johns Hopkins University
Baltimore, MD, USA

[3]Dept. of Biological Sciences, Hunter College
City University of New York,
New York, NY, USA

[4]Victorian Life Sciences Computation Center
University of Melbourne
Melbourne, Australia

Abstract - Unlike newly developed web applications that can be designed from the ground up to utilize cloud APIs and run natively within cloud infrastructure, most complex bioinformatics pipelines that are in advanced states of development can only be encapsulated within VMs along with all their software and data dependencies. To take advantage of the scalability offered by the cloud, additional frameworks are required to create virtualized compute clusters and emulate the most common infrastructure found on institutional resources where most of the existing bioinformatics pipelines are generally run. In this paper we describe one such framework, its compatibility with multiple Clouds and present an automated process for deploying the entire system so it can be made easily available on any Cloud.

I. Introduction

The adoption of cloud computing technologies has provided an opportunity to effectively democratize the field of computational biology, meaning that individual researchers and labs can now have access to the resources that were previously only available to the large sequencing and bioinformatics centers [1]. However, this presents challenges of its own, such as the need to configure and utilize bioinformatics software for next-generation sequencing, where significant expertise is required on UNIX based operating systems, programming languages, software compilation from source code, file systems, and large-scale data management [2]. For smaller laboratories this can become a significant obstacle, because in addition to coming up with the funds for building an informatics infrastructure with the capacity to handle large computations, they also need to hire trained bioinformaticians competent to install, configure and run sequence analysis software tools and manage the data. This can present a higher expense than that of acquiring the hardware.

The cloud computing model has showcased its ability to transform how access to compute resources is realized and has delivered on the notion of Infrastructure-as-Code [3]. However, this model presents the user with low-level resources that still need to be procured, configured, and managed. In the context of bioinformatics, this means installing required tools and reference data and setting up a virtualized cluster environment. As data volumes increase, this also requires utilizing compute and storage resources in a scalable manner. To address this, we have been developing CloudMan [4] as a versatile solution for enabling and managing compute clusters in cloud environments via a simple GUI web interface or a programmable API. As a full-fledged deployment, CloudMan integrates with the Galaxy framework [5], [6] and provides a comprehensive workbench for biomedical data analysis on the Cloud [7]; it enables one to seamlessly acquire cloud resources, assemble those into a compute cluster, upload data, perform data analysis by chaining output of one tool as the input of another, visualize the data, and finally share it. This is the end-user view. Internally, CloudMan manages cloud resources, operating system services, and application-level services.

As part of this article, we describe how we have extended CloudMan into an interoperable solution for multiple Clouds and built a framework for allowing anyone to deploy their own version. We describe the process for building the required cloud resources, how to launch an instance of the deployed components and how to configure CloudMan to manage a launched instance and correctly orchestrate any required runtime components. Once deployed, users will have a functional bioinformatics workbench and a dynamically scalable cluster for data analysis on a given cloud.

II. Background

In recent years, cloud based bioinformatics data analysis systems such as Galaxy [7], CloVR [8], Cloud BioLinux [9] and bioKepler [10] were released, allowing smaller laboratories and institutions to perform large-scale data analysis with genomic datasets. All these platforms are available on the Amazon Elastic Compute (EC2) cloud, which provides data centers across the world, allowing any researcher worldwide to connect to the cloud end-point closest to their geographic boundaries.

Researchers or institutions with access to computing resources at their home institution have the option to implement their own private Cloud platform and run Virtual Machine (VM) servers, using the open-source Eucalyptus [11] or OpenStack cloud middleware. OpenStack is the official Cloud platform bundled with the Ubuntu Linux operating system [1] and can be readily installed on clusters running other Linux distributions[2],

[1] http://www.zdnet.com/article/ubuntu-syncs-up-with-
[2] http://docs.openstack.org/

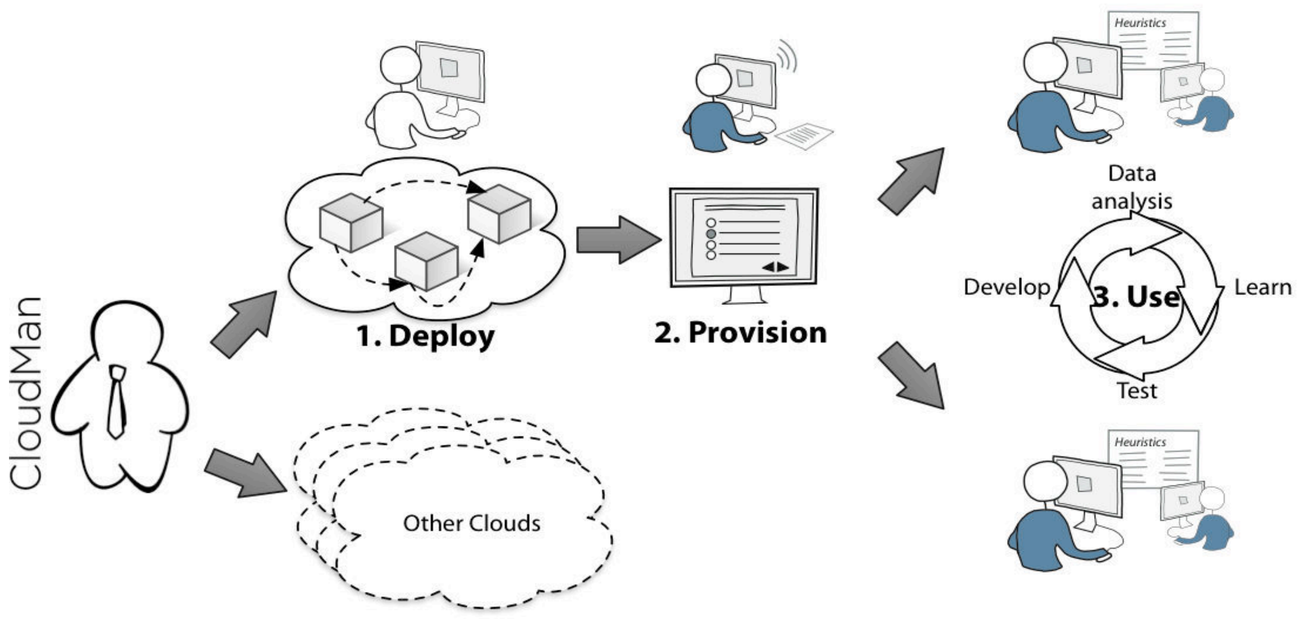

Figure 1. The effort described in this paper highlights the features of CloudMan as a platform for building and provisioning bioinformatics workbenches across multiple Clouds. With the multi-cloud compatibility of CloudMan (Section 3.A), **(1)** a deployer would use the provided automation methods to build and link the required cloud resources, tailored to the needs of the given research community (Section 3.B). Then, **(2)** a group leader, system administrator or even a domain researcher uses a launcher app to provision any number of instances of the workbench and possibly further customize those with custom data or tools (Section 3.C). Finally, **(3)** domain researchers use the workbench in their daily routine.

and Eucalyptus is similar. Both of these Cloud platforms offer Application Programming Interfaces (APIs) compatible with Amazon's EC2 service, and applications developed either on Eucalyptus or OpenStack work on the Amazon Cloud and vice-versa. Furthermore, users have the option to copy and execute VM server snapshots across installations of these Clouds as has been demonstrated with Cloud BioLinux.

For users who do not own local compute resources, nor have funds available to lease computing time from a public Cloud provider, there is the possibility of using government-funded Cloud infrastructures, such as the US-based Open Science Data Cloud[3], the Australian NeCTAR Research Cloud[4], or Helix-Nebula in the EU[5]. In addition, a number of academic computing centers in both the US and other countries have similar clusters with OpenStack installed[6], to which scientists can request access.

III. Deploying and Provisioning Virtual Environments

Despite the general availability of cloud computing resources around the world, due to the complexity of setting up a functional bioinformatics platform for data analysis, we have (1) expanded the capabilities of CloudMan to be interoperable with multiple cloud providers; (2) devised an automated method for deploying a complete system on compatible clouds; and (3) developed a launcher application allowing end-users to provision instances of the platform when and where needed. The overall concept is depicted in Figure 1 and each of these contributions is described in the following sections.

A. A Multi-cloud Cluster Manager

Architecturally, CloudMan is a standalone application written in Python that orchestrates cloud components (virtualized compute resources and software programs) to deliver persistent and functional user-level services (see [12] for the implementation details). To operate, CloudMan uses a number of cloud-level features and assembles those into a functional platform for performing data analysis [4]. Internally, CloudMan abstracts away the interaction with multiple cloud providers via a common cloud interface (`cloud_interface`). All the internal methods utilize the given interface providing uniformity throughout the code while supporting multiple cloud technologies. Compatibility with multiple clouds is achieved via a custom implementation of the cloud interface for each of the supported clouds (AWS, OpenStack and Eucalyptus). Adding support for future clouds is a matter of implementing the methods defined in the `cloud_interface`, with the option to inherit from the default AWS implementation and, again, identifying and implementing only the differences.

Because a running instance of the CloudMan platform represents a composition of multiple cloud services, in order to deploy the platform, it is necessary for the cloud middleware to provide the required features, as follows:

- *Customizable machine image*: this is the VM itself containing the operating system (Ubuntu) and required system level packages for applications such as Nginx, Slurm, RabbitMQ.
- *Instance user data*: this enables runtime contextualization [13], which is what makes it

[3] https://www.opensciencedatacloud.org/
[4] http://nectar.org.au/research-cloud
[5] http://www.helix-nebula.eu/
[6] https://www.tacc.utexas.edu/systems/rodeo

possible to have different and independent instantiations of the same platform. Sample data included as part of the instance data includes the API keys that CloudMan should use to connect to the cloud provider.

- *Security groups*: a virtual firewall offered by the cloud provider that permits selective traffic to one's compute instances based on user-defined access rules. CloudMan uses HTTP, SSH, FTP and several custom ports to enable access to the running services.
- *Persistent POSIX data storage*: this is where the user-uploaded and analyzed data is stored, as well as installed tools, databases, and tool configurations; it is leveraged to provide longevity beyond the uptime of an instance. Ideally, this storage is provided in the format of block storage devices of arbitrary size (i.e., data volumes) but other storage devices can be used instead, including mountable external file systems, such as NFS or Gluster, or the ephemeral storage on individual instances.
- *Block storage snapshots*: given that a significant amount of bioinformatics data is read-only reference genome data, the CloudMan platform maximizes data reuse by utilizing a single source of this type of data for multiple deployed instances of the platform. Technically, this is delivered via point-in-time block storage snapshots, which are converted into data volumes at runtime. To realize the reuse aspect of this data, the snapshots or archives need to be sharable. Alternatively, it is possible to use "tarballs" of file systems, which are archived in an object store and extracted at runtime onto an instance's ephemeral storage.
- *Object store*: persistent configuration data describing a given instance of the platform is stored in a globally accessible object store and it is used to recompose the CloudMan platform after it had been shut down. Information about running platform services (i.e., applications), utilized resources (e.g., volume, snapshot, machine image), and the CloudMan runtime code are stored here.
- *Per object ACL controls in the object store*: in order to enable instance sharing [14] with individual users, it should be possible to set Access Control List (ACL) permissions on each individual object inside the object store. Note that this is a requirement for the instance sharing functionality only.
- *Resource metadata*: popularly known as tags, assigning metadata to each of the utilized cloud resources helps identify all the components making up the complete platform. It also provides a fallback in case the platform needs to be recreated by hand following an unexpected corruption.

Given the feature-set of today's clouds, the above list is considered quite minimal and we strive towards minimizing it further. In the meantime, the required resources are available from a variety of cloud providers/middleware as bare-bones resources that behave consistently across those, making it possible to create the cross-cloud platform being described in this paper. This approach is in contrast to using custom (or high-level) services offered by a specific cloud provider and thus becoming locked-in to the given cloud or cloud features.

B. *Automating Component Deployment*

The previous section describes the multi-cloud support provided by CloudMan as a reusable framework and outlines the set of requirements a target Cloud must posses. In this section, we present a method for deploying the complete CloudMan platform. By deploying, we mean building the necessary components and making them available on the target cloud. These components will be joined at runtime (see next section) to deliver a functional, user-facing platform.

The process of deploying the required components has been automated using the Ansible automation framework. Ansible is a configuration management and provisioning tool; it uses files in YAML format to allow users to define tasks (e.g., `apt-get update`) and it then executes those tasks on target resources in an idempotent fashion. It uses SSH to communicate with target resources, hence imposing minimal requirements on setup. Conceptually, Ansible allows one to define reusable roles, which are composed of multiple tasks. It further allows multiple roles to be assembled into more comprehensive playbooks that can mix and match functionality offered by individual roles.

For the purposes of deploying a bioinformatics workbench backed by the CloudMan platform, we have developed a set of Ansible roles and assembled those into a playbook[7]. As stated above and described in detail in [12], the overall platform requires multiple components: the base machine image, an instance of the tools file system, and an instance of a file system containing indexed reference genomes. The playbook offers the capacity to build these components in an automated fashion. To start, it is necessary to build the machine image. This step leverages Packer[8], which is an additional layer of automation on top of Ansible. It handles provisioning of a VM on the target cloud, invoking the set of Ansible roles for building the machine image, creating the machine image, and cleaning up. Because Packer can run in parallel on multiple targets, it is possible to build the machine image with a single command (e.g., `packer build cloudman.json`) on multiple clouds at once. The built machine image will contain the required system packages and libraries, FTP server, Nginx proxy server, Slurm cluster manager and required configurations (e.g., users, environment settings).

Next, it is necessary to build the core file system, called `galaxyFS`. This file system holds all the tools, databases, the Galaxy application, and configuration

[7] https://github.com/galaxyproject/galaxy-cloudman-playbook
[8] http://packer.io

TABLE I. A SAMPLE FROM *SNAPS.YAML* CONFIGURATION FILE. THE FILE STRUCTURE ALLOWS MULTIPLE CLOUDS, MULTIPLE REGIONS, AND MULTIPLE DEPLOYMENTS WITHIN A REGION TO BE SPECIFIED THAT CAN BE READ BY THE LAUNCHER APPS AND CLOUDMAN TO PROPERLY ASSEMBLE AN INSTANCE OF THE PLATFORM.

```
version: 1
clouds:
 - name: amazon
   regions:
   - deployments:
     - name: GalaxyCloud
       filesystems:
       - name: galaxy
         roles: galaxyTools,galaxyData
         snap_id: snap-adad90fc
       - name: galaxyIndices
         roles: galaxyIndices
         snap_id: snap-5b030634
       default_mi: ami-118bfc78
       bucket: cloudman
     name: us-east-1
- name: nectar
<include details about other clouds/regions>
```

options that will be available via the workbench. Building *galaxyFS* is realized by launching an instance of the machine image built in the previous step, adding a file system of desired size (e.g., 10GB) via CloudMan's Admin page (note that CloudMan will be available because we are now using the image built in the previous step), and running the available Ansible roles. The roles will configure and initialize a PostgreSQL database as well as download and configure the Galaxy application with production settings. Installing the desired tools is realized via the ToolShed [15]. This step has been automated via a custom Python script that leverages the BioBlend API [16], [17] and installs more than a hundred tools at once. The list of installed tools is provided in a configuration file and can hence be easily modified. Once the process has been completed, it is necessary to create a snapshot of the resulting file system, which can be achieved via CloudMan's Admin page.

Similar to building the *galaxyFS*, it is necessary to build *indicesFS* - a file system containing desired reference genomes. The process is similar in that a new file system of adequate size needs to be added. Adding the reference genomes is performed via Galaxy Data Managers [18], which integrate with Galaxy and allow one to simply choose which genomes to include. Once the process has been completed, it is also necessary to create a snapshot of the resulting file system. Finally, it may be desirable to make the created snapshots public (or share them with select users) so others can launch instances of the deployed components (see next section). Also note that instead of creating volume snapshots, it is possible to create downloadable archives or shared file systems but, for brevity, we focus on volume snapshots alone.

C. Provisioning

Once the required components have been built, they need to be assembled into a provisioned instance of the platform. Instances can be provisioned by projects or labs to offer an 'always-on' ready-to-use workbench or by individual users on an as-needed basis (see Section IV). To make the process accessible, we have further automated the provisioning process via launcher web applications.

The build process described in the previous section produces a machine image and two file system snapshots. The provisioning process launches a VM instance based on the image, which starts CloudMan, and CloudMan then creates volumes based on the previously created snapshots to make functional file systems (along with setting up a virtual cluster and starting several service processes). Before an instance can be provisioned, it is necessary to compose the instance user data that contains access details for the given Cloud (e.g., region endpoints, ports) and information about the cluster (e.g., cluster name, access credentials). Once an instance of the image is launched, CloudMan will automatically start and continue the workbench contextualization process. To obtain the snapshot ID's, CloudMan refers to the *default_bucket* that is passed in with user data. This is an accessible object store container where a file (by default *snaps.yaml*) is stored and contains the details about the snapshots and corresponding file systems (see Table I). CloudMan will read the contents of the given file and proceed with setting up the system.

The above process can be somewhat detail-oriented and technical in nature. To simplify it, we have introduced a notion of a launcher web application that will handle the provisioning process automatically. The launcher app offers users a web form asking for cloud access credentials, a name for the instance, and a choice of VM type. The app then composes and formats the required information, instantiates an instance of the base machine image and informs the user when the launch process has completed. It also allows a previously created cluster to be readily recreated. There are two launcher applications available: one within the Galaxy application itself (CloudLaunch) and a standalone one (BioCloudCentral). CloudLaunch is automatically available on any Galaxy instance (at */cloudlaunch* URI) but requires a local instance of Galaxy (a public one is available at *https://usegalaxy.org/cloudlaunch*). Alternatively,

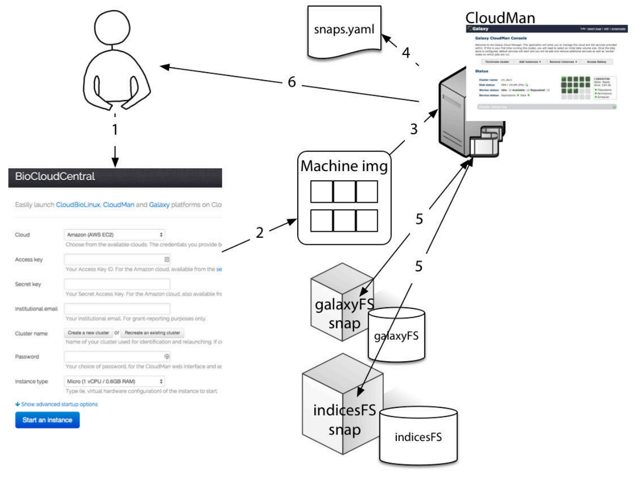

Figure 2. The process of deploying an instance of the platform, also showing all the required components.

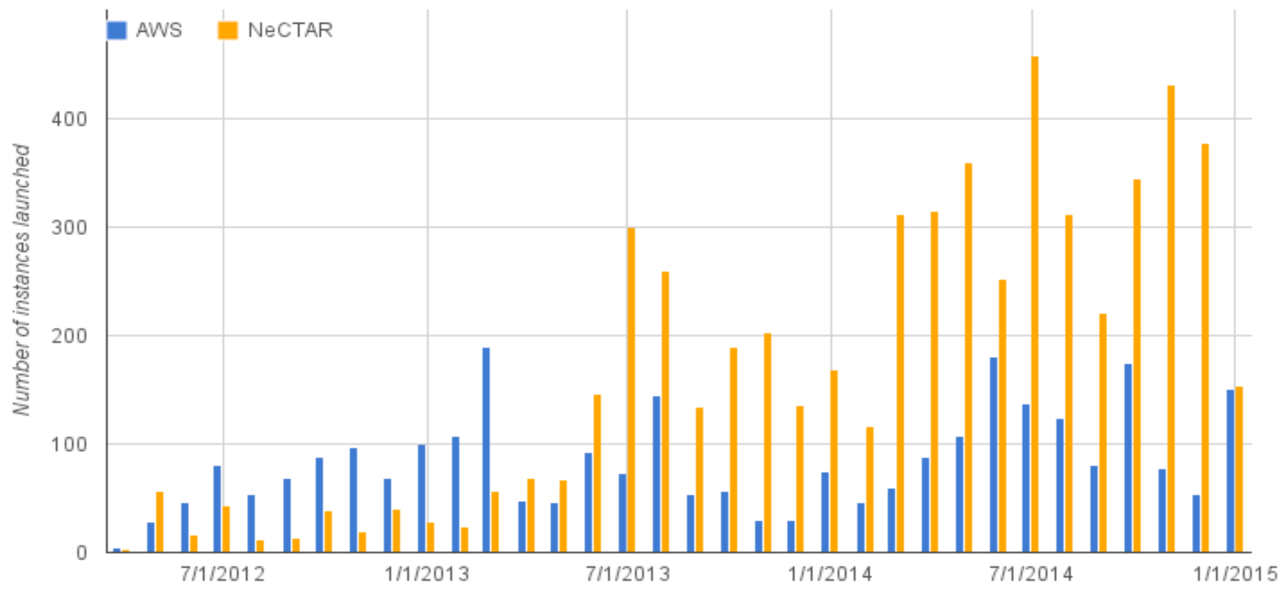

Figure 3. The process of launches per month of the CloudMan platform on two clouds, AWS and NeCTAR.

BioCloudCentral can be installed on a dedicated host and used independently of Galaxy (a public instance of this application is available at *https://biocloudcentral.org/*).

Figure 2 summarizes the described effort: after the required set of components has been built and registered in `snaps.yaml`, the launcher app is used to provision an instance of the platform. After the contextualization steps have completed, the user can start using the platform.

IV. Demonstration

The effort described in this paper allows anyone to build their own deployment of the CloudMan platform and the associated bioinformatics workbench. It further allows users to launch their own instances of the platform on the clouds on which it has been deployed. Currently, in addition to the long-standing deployment on AWS, the platform has been deployed on the Australian research cloud, NeCTAR; these represent two flagship deployments of the platform that are supported by the authors. Other deployments also exist but have been done by independent projects at their home institutions. Figure 3 captures the number of times the platform has been provisioned on the AWS and NeCTAR clouds over the past three years. The figure indicates that the NeCTAR cloud, despite the fact that it is available only to Australians, has been seeing approximately two to three times the number of instantiations as the world-wide deployment on AWS. We attribute this to the fact that the NeCTAR cloud is free of cost and that the Australian community has largely moved 'into' the cloud, especially for training and teaching purposes.

In addition to the count of provisioned platforms, the authors have been maintaining an always-on instance of the CloudMan platform on the NeCTAR cloud as part of the Genomics Virtual Laboratory (GVL) project[9]. This instance of the platform offers an instance of the Galaxy application that has been tailored for bioinformaticians training. During the course of the GVL project, a number of tutorials has been developed that demonstrate bioinformatics data analysis activities, such as variant calling or RNA-Seq analyses[10]. The instance, called *galaxy-tut* (available at *http://galaxy-tut.genome.edu.au/*), is where the tutorial data and tools are hosted, representing a fertile playground for training sessions. Besides being used during live training sessions, the instance is always available for anyone to follow the tutorials at his or her own pace. To accommodate these scenarios, the compute infrastructure behind the *galaxy-tut* user interface needs to scale with demand: as a lab full of students initiates a mapping job, the resource requirements rapidly grow. At the end of the compute-intensive portion of a tutorial, or during non-teaching periods, the resource requirements are much lower. The cloud model fits this scenario perfectly and the CloudMan platform is entrusted with keeping up with the cycle of requesting, configuring, managing, and releasing the necessary resources.

The *galaxy-tut* instance was launched as a default CloudMan instance on the NeCTAR research cloud and then manually customized to include all the data and Galaxy histories required for the tutorials. The master instance of the CloudMan cluster is running on an extra-large instance type (8 virtualized CPU cores, 16GB RAM). During training sessions, running of the jobs on the master instance is disabled via CloudMan's Admin UI and additional cluster nodes are explicitly launched. In addition, CloudMan's auto-scaling is left *on* to accommodate possible spikes in usage. The instance has nearly 500 registered users, stores 1TB of user data, and runs approximately 2000 jobs per month.

We have also deployed and run CloudMan and Galaxy on an experimental Eucalyptus cloud installation at the J. Craig Venter Institute[11]. The installation comprised of four

[9] https://genome.edu.au/

[10] https://genome.edu.au/wiki/Learn

[11] http://www.jcvi.org/

identical physical compute servers (16 core, 32GB memory, 1TB disk each). One of these servers was used as the cloud's head node to run Cloud Controller (CLC) service of Eucalyptus, and the remaining nodes running the passive Node Controllers (NCs) for worker VMs. The CloudMan framework has enabled us to bootstrap on-demand virtualized compute clusters with an easily accessible Galaxy interface on our private cloud. By simply starting with a pre-configured VM dropped on the CLC node, and configuring the Galaxy cluster endpoint to be handled by CloudMan, we were able to instantiate multiple transient VMs for parallel job execution across the Eucalyptus cloud nodes with CloudMan performing the initialization, management and termination of the VMs.

V. Conclusions

Dealing with the infrastructure and system level operations, such as acquiring cloud instances or managing operating system level services, requires a significant investment in time and expertise. Yet, for modern day bioinformatics it is an essential requirement. In our previous work, we have developed a system that provides a ready-to-use bioinformatics workbench with dozens of tools and gigabytes of reference genome data on the AWS cloud. This paper describes the effort of making the overall system underlying the workbench compatible with multiple Clouds and an automated process for deploying it. This makes it possible for anyone to deploy their own instance of the platform on a private cloud or under their own account on a public account. Each deployed instance can have a custom set of tools and reference genome data available, which are tailored to the specific needs of a group or an institution. Once deployed, others can launch the given instance of the workbench via provided launcher web applications.

Acknowledgments

The efforts of the Galaxy Team (Dan Blankenberg, Dave Bouvier, Martin Cech, Dave Clements, Carl Eberhard, Jeremy Goecks, Nitesh Turaga, Sam Guerler, Jennifer Jackson and Anton Nekrutenko) were instrumental in making this work happen. This project was supported through grant number HG005542 from the National Human Genome Research Institute, National Institutes of Health as well as grants HG005133, HG004909 and HG006620 and NSF grant DBI 0543285. Additional funding is provided by Huck Institutes for the Life Sciences at Penn State and, in part, under a grant with the Pennsylvania Department of Health using Tobacco Settlement Funds. The Department specifically disclaims responsibility for any analyses, interpretations or conclusions. Additional funding has been provided by the EU FP7-PEOPLE programme, under grant 277144. Further funding has been provided by the Australian NeCTAR project, under the Genomics Virtual Laboratory.

References

[1] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, and G. P. Nolan, "Computational solutions to large-scale data management and analysis.," *Nature Reviews Genetics*, vol. 11, no. 9, pp. 647–57, Sep. 2010.

[2] M. C. Schatz and B. Langmead, "The DNA data deluge," *IEEE Spectrum*, vol. 50, pp. 28–33, 2013.

[3] M. C. Schatz, B. Langmead, and S. L. Salzberg, "Cloud computing and the DNA data race.," *Nature Biotechnology*, vol. 28, pp. 691–693, 2010.

[4] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko, and J. Taylor, "Galaxy CloudMan: delivering cloud compute clusters," *BMC bioinformatics*, vol. 11 Suppl 1, p. S4, 2010.

[5] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biology*, vol. 11, no. 8, p. R86, Jan. 2010.

[6] E. Afgan, J. Goecks, D. Baker, N. Coraor, A. Nekrutenko, and J. Taylor, "Galaxy - a Gateway to Tools in e-Science," in *Guide to e-Science*, X. Yang, L. Wang, and W. Jie, Eds. Springer, 2011, pp. 145–177.

[7] E. Afgan, D. Baker, N. Coraor, H. Goto, I. M. Paul, K. D. Makova, A. Nekrutenko, and J. Taylor, "Harnessing cloud computing with Galaxy Cloud," *Nature Biotechnology*, vol. 29, no. 11, pp. 972–974, Nov. 2011.

[8] S. V Angiuoli, M. Matalka, A. Gussman, K. Galens, M. Vangala, D. R. Riley, C. Arze, J. R. White, O. White, and W. F. Fricke, "CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing," *BMC Bioinformatics*, vol. 12. p. 356, 2011.

[9] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bicak, D. Field, and K. Nelson, "Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community," *BMC Bioinformatics*, vol. 13. p. 42, 2012.

[10] I. Altintas, "Distributed Workflow-Driven Analysis of Large-Scale Biological Data Using bioKepler," in *2nd International Workshop on Petascal Data Analytics: Challenges and Opportunities (PDAC)*, 2011, pp. 41–42.

[11] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 124–131.

[12] E. Afgan, D. Baker, A. Nekrutenko, and J. Taylor, "A reference model for deploying applications in virtualized environments," *Concurrency Computation Practice and Experience*, vol. 24, pp. 1349–1361, 2012.

[13] K. Keahey and T. Freeman, "Contextualization: Providing one-click virtual clusters," in *4th IEEE International Conference on eScience*, 2008, pp. 301–308.

[14] E. Afgan, B. Chapman, and J. Taylor, "CloudMan as a platform for tool, data, and analysis distribution," *BMC Bioinformatics*, vol. 13, p. 315, 2012.

[15] D. Blankenberg, G. Von Kuster, E. Bouvier, D. Baker, E. Afgan, N. Stoler, J. Taylor, and A. Nekrutenko, "Dissemination of scientific software with Galaxy ToolShed," *Genome biology*, vol. 15, no. 2, p. 403, Jan. 2014.

[16] C. Sloggett, N. Goonasekera, and E. Afgan, "BioBlend: automating pipeline analyses within Galaxy and CloudMan," *Bioinformatics*, vol. 29, no. 13, pp. 1685–6, Jul. 2013.

[17] S. Leo, L. Pireddu, G. Cuccuru, L. Lianas, N. Soranzo, E. Afgan, and G. Zanetti, "BioBlend.objects: metacomputing with Galaxy," *Bioinformatics*, vol. 30, no. 19, pp. 2816–7, Oct. 2014.

[18] D. Blankenberg, J. E. Johnson, J. Taylor, and A. Nekrutenko, "Wrangling Galaxy's reference data," *Bioinformatics*, vol. 30, no. 13, pp. 1917–9, Jul. 2014.