Science Gateways
Community Institute

Trusted CI - SGCI Galaxy Engagement Final Report

July 1, 2020 - Dec 30, 2020

December 9th, 2020

v1.1

Distribution: Release to Public after February 2, 2020

Trusted CI - Science Gateways Community Institute Engagement Team:

Andrew Adams, Mark Krenz[1], Shane Filus, Anurag Shankar, Kelli Shute

Galaxy Project Engagement Team:

Enis Afgan, Dannon Baker, Nate Coraor, Nuwan Goonasekera, Marius van den Beek

[1]Engagement Lead

About Trusted CI

The mission of Trusted CI is to provide the NSF community with a coherent understanding of cybersecurity, its importance to computational science, and what is needed to achieve and maintain an appropriate cybersecurity program.

About the Science Gateways Community Institute

The Science Gateways Community Institute was created with a mission of providing resources, expertise, community support, and education to the creators of gateways serving science and engineering research and education.

Trusted CI - SGCI Partnership

The work on this engagement was made possible through a collaborative partnership between Trusted CI and SGCI. Throughout this document where the Trusted CI team is mentioned, it is referring to three of the team members involved with this partnership and also two additional members of Trusted CI.

Acknowledgments

Trusted CI's and SGCI's engagements are inherently collaborative; this report would not be possible without the collaborative effort that the Galaxy Project team provided, including: Enis Afgan, Dannon Baker, Nate Coraor, Nuwan Goonasekera, and Marius van den Beek.

This document is a product of the NSF Cybersecurity Center of Excellence (Trusted CI) and the Science Gateways Community Institute (SGCI). Trusted CI is supported by the National Science Foundation under grant ACI-1920430. For more information about Trusted CI please visit: http://trustedci.org/. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The Science Gateways Community Institute is funded by the National Science Foundation under award number ACI-1547611. For more information about SGCI please visit: https://sciencegateways.org. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The Galaxy Project is supported in part by NSF, NHGRI, The Huck Institutes of the Life Sciences, The Institute for CyberScience at Penn State, and Johns Hopkins University.

Table of Contents

Executive Summary

The Trusted CI team had a 6-month engagement with the Galaxy Project (galaxyproject.org) team to evaluate the security of Galaxy's Kubernetes Deployment model. Using a NIST SP 800-53 control-based System Security Plan (SSP)[4] process, the teams reviewed the architecture and data flows of the Galaxy's containerized deployment system, while discerning trust relationships between its components. The Trusted CI team evaluated this architecture, flow and trust relationships and identified areas where the security posture could be, and in some cases, *should* be improved, i.e., '3 Recommendations', below. Additionally, throughout the engagement, Trusted CI made observations about the project and software management process of the Galaxy Platform. Section 5 contains a list of ideas for potential future engagements between Trusted CI and Science Gateways Community Institute.

Using This Document

Throughout the report we refer to some external documents and resources through the use of footnotes. There is also a list of documents that we refer to multiple times using numbers in square brackets (ie. [2]). The list of these documents is in section 6 "References and Artifacts" of this report.

1. Engagement Background & Process

Galaxy is an open-source, scientific workflow system developed by the Galaxy Project (GP) community. Galaxy provides a means to build multi-step computational analyses using a graphical web user interface that allows a user to specify the type of data to operate on, what steps to take, and in what order. Galaxy is also a data integration platform for biological data. It supports data uploads from a user endpoint, via a URL-accessible web data source, and directly from many well-known, online resources (such as the UCSC Genome Browser, BioMart, and InterMine). It allows researchers to carry out analyses without having to do any programming.

Trusted CI engaged with the Galaxy Project team for the purpose of reviewing the security of a Galaxy software distribution being developed as a containerized package (referred to the Galaxy Kubernetes Deployment Model; Galaxy hereafter for simplicity), with an eye toward its use with sensitive information such as protected health information (PHI).

At the beginning of the engagement, Trusted CI and Galaxy decided upon and agreed to the engagement's primary goals, namely to review Galaxy's architecture, document and analyze its security controls, and recommend strategies to protect data managed through Galaxy. The detailed objectives of the engagement were as follows:

a. Review Galaxy to understand in detail its architecture and data flows.
b. Analyze the Galaxy architecture security.
c. Recommend changes/enhancements to Galaxy based on gaps identified.
d. Compare the existing Galaxy security safeguards with HIPAA Security Rule requirements and identify gaps.
e. Create a NIST 800-53 System Security Plan (SSP)[4] for Galaxy, with an eye toward the GP team offering this SSP to the community to help them comply with security regulations.

Trusted CI met with the Galaxy Project team weekly throughout the engagement period to supplement our understanding of the Galaxy architecture, data flows, existing security safeguards, and software development practices. Using this information, Trusted CI conducted an analysis of Galaxy security against industry best practices. Finally, we developed a set of recommendations based on the findings of this analysis and our experiences in this area. This report itemizes those recommendations, including

areas that Trusted CI believes could benefit from strengthening, based on the understanding of Galaxy and describes key findings.

This report is organized as follows:

- Section 1 explains the purpose, scope, and process of the engagement.
- Section 2 provides an overview of Galaxy.
- Section 3 details findings and recommendations for improving Galaxy.
- Section 4 identifies potential areas of focus for future Trusted CI / Galaxy engagements.
- The References section lists all of the artifacts reviewed and referenced throughout this report.
- The Appendix includes a boilerplate template that could be included as part of the distribution to address its use for sensitive data. It also contains screenshot copies of all the data flow diagrams created by the Galaxy Project team as part of the engagement.

2. Factual Summary

2.1 Galaxy Project management

The Galaxy Project[2] is a multi-institutional team distributed around the world with the goal of creating the Galaxy software, services, and ecosystem used in Galaxy deployments. These institutions include Johns Hopkins University (core), Penn State University (core), Oregon Health & Science Institute (core), Cleveland Clinic (core), Broad Institute (partner on project), University of Melbourne (community member), University of Freiburg (community member), and many other institutions that are part of the community of interest. Each year, the Galaxy Project organizes a community conference, called Galaxy Community Conference (GCC) meeting, in which they meet with community members.

The project's core team distributes the tasks of maintaining and developing the software between team members. They have traditionally relied on Johns Hopkins University and Penn State University for providing cybersecurity services and protection.

[2] https://galaxyproject.org/

The project currently has a few security related policies mostly focused around how to submit security vulnerabilities[3] and using two factor authentication (2FA) for code repository access.

2.2 The Galaxy Kubernetes deployment model

The Galaxy Kubernetes Deployment model is a newer implementation architecture of the Galaxy system intended to make production deployment easier and more consistent. It uses Kubernetes and their own custom deployment software, CloudMan, to deploy a complete Galaxy setup at IaaS cloud providers such as Amazon AWS, Jetstream, and others.

An instance of the Galaxy Kubernetes Deployment model makes use of several components, many of which run within their own Docker node as part of a deployed cluster. The components that Trusted CI reviewed during this engagement included Celery[4], CloudMan[5], CVMFS[6], Docker[7], Grafana[8], Gunicorn[9], Helm[10], InfluxDB[11], Keycloak[12], Kubernetes[13], NFS, nginx[14], PostgreSQL, Prometheus[15], Pulsar[16], RabbitMQ, Rancher[17], Slurm[18], Squid[19], and Swarm[20]. Trusted CI made recommendations below for a subset of these components. The communication between some of these nodes and services is shown in Appendix A.

While the Galaxy Project team develops the template for the Kubernetes Deployment model, individual deployments are installed and managed by administrators at their own institutions. These Kubernetes Deployments continue to use some of the Galaxy Project's components during and after installation. For instance, new tools for the

[3] https://github.com/galaxyproject/galaxy/blob/dev/SECURITY_POLICY.md
[4] http://celeryproject.org
[5] https://galaxyproject.org/cloudman/
[6] https://cernvm.cern.ch/portal/filesystem
[7] https://www.docker.com/
[8] https://grafana.com/
[9] https://gunicorn.org/
[10] https://helm.sh/
[11] https://www.influxdata.com/
[12] https://www.keycloak.org/
[13] https://kubernetes.io/
[14] https://www.nginx.com/
[15] https://prometheus.io/
[16] https://galaxyproject.org/cloudman/services/pulsar/
[17] https://rancher.com/
[18] https://slurm.schedmd.com/overview.html
[19] http://www.squid-cache.org/
[20] http://www.swarm.org

Galaxy deployment can be added via the ToolShed App store. Galaxy components may also connect to the Galaxy Project's CVMFS data store to retrieve additional, application-specific data.

2.3 The Galaxy Architecture and Data Flows

The Galaxy Kubernetes Deployment model has a number of components that communicate with each other. As part of this engagement with Trusted CI, the Galaxy Project team has documented and diagramed 15 important architectures and processes. The original document is located on *diagrams.net* and a copy of these diagrams are included in section 1.7.1 of the SSP document and also in Appendix A of this document.

2.4 Galaxy's external components

The functionality of a Galaxy instance depends on or makes use of components that a user does not run on their own instance. These include the ToolShed App store and the CVMFS data store. Independent Galaxy installations may make use of these external services during their lifetime.

2.5 Usegalaxy.org

This is a shared instance of Galaxy that is run by the Galaxy Project for the purpose of allowing those interested in trying out or using Galaxy with an already installed and configured system. It is not intended for hosting sensitive data.

3. Recommendations

The following is a set of recommendations that Trusted CI thinks the Galaxy Project team should implement in the near future.

3.1. Governance

3.1.1. Information Security Plan

A cybersecurity program is a documented set of an organization's information security policies, procedures, and processes. It is critical to proactively protect data while maintaining compliance with best practices and regulatory requirements, as well as customer standards. Galaxy should create a comprehensive security plan, one example

is the System Security Plan [4]. Alternatively, Galaxy could utilize a less rigorous approach such as the Trusted CI Framework[21].

3.1.2. Information Security Lead

Due to the complexity and breadth of cybersecurity issues[22] and the need for coordinated decision making, organizations require an individual role to lead cybersecurity. This position, often referred to as the Chief Information Security Officer (CISO), ensures the program educates and advises decision makers on cybersecurity matters including risk identification and mitigation and policy development. The position also provides leadership for services like incident response coordination and cybersecurity control selection and monitoring. Galaxy should designate an information security lead for the Galaxy Project.

3.1.3. Information System Inventory

Information assets are valuable, sensitive, and/or mission-critical information[23] and information systems.[24] Information asset documentation is the collection of artifacts describing the cybersecurity relevant details of information assets presented in a form that is useful to cybersecurity professionals and decision makers. Galaxy should create a comprehensive inventory of services managed by the Galaxy Project team that a site implementing the Galaxy software would be expected to use.

3.1.4. System Architecture

Although not a required component of an organization's cybersecurity program (see Section 3.1.1.), thorough documentation explaining the system's and/or services' architecture can greatly improve the comprehensivity of the cybersecurity program. Since Galaxy's architecture is complex, their cybersecurity program would be strengthened by providing security architecture documentation.

---

[21] https://www.trustedci.org/framework

[22] *See* NIST SP 800-181 NICE Cybersecurity Workforce Framework for the knowledge areas expected of cybersecurity professionals, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181.pdf.

[23] **Information** is any communication or representation of knowledge such as facts, data, or opinions in any medium or form, including textual, numerical, graphic, cartographic, narrative, or audiovisual. Organizational information may be stored and used within the organization's information systems, as well as flow out to third party systems. *See*, National Information Assurance (IA) Glossary, CNSS Instruction No. 4009, Apr. 2010.

[24] An **information system** is a discrete set of information and related resources (such as people, equipment, and information technology) organized for the collection, processing, maintenance, use, sharing, dissemination, and/or disposition of information. These include, but are not limited to, mobile devices, routers, servers (the usual commodity IT equipment), as well as industrial control systems (ICS) / supervisory control and data acquisition (SCADA) systems, physical security systems, heating, ventilation, and air conditioning (HVAC) systems, and any other connected devices. *See*, 44 U.S.C. 3502.

3.1.5. Risk Management Strategy

Programmatic evaluations are how the organization determines whether the cybersecurity program (see Section 3.1.1.) is achieving its purpose. Refinements are any changes designed to improve the program's efficiency or effectiveness. Evaluation and refinement of a cybersecurity program can take many forms depending on the formality and scope of the assessment and the type of evaluation (e.g., planned, comprehensive program evaluations; internal self-evaluations following an incident). Galaxy should conduct regular, documented risk analysis and mitigation.

3.1.6. Roles and Responsibilities

Personnel resources are commitments made by an organization to assign human effort to particular activities on behalf of the organization. Personnel resources allocated to cybersecurity include both full-time cybersecurity employees and employees with partial cybersecurity responsibilities. Personnel resources allocated to cybersecurity may be assigned to carry out a number of organizational activities, including security operations, governance, management, architecture, and incident response. In addition to the cybersecurity lead (see Section 3.1.2.), Galaxy should explicitly define and document team member roles and responsibilities in regards to the Galaxy Project's cybersecurity program (see Section 3.1.1.).

3.2. Source Code Management

The Galaxy Project has custom source code that they develop and manage using GitHub[25]. The Galaxy Project has also adopted a policy for handling security issues in their code that they follow and make the policy available to the public.[26] As a general strategy we recommend that the Galaxy Project team also review the recommended practices outlined in Trusted CI's Software Engineering Guide[27] to further enhance their existing development processes.

3.3. Tool-Specific Recommendations

The following is a list of recommendations for the tools that are used within a Galaxy cluster. Trusted CI has ordered the recommendations using their estimate of the amount of impact in reducing cybersecurity risk implementing a recommendation would cause and a general idea of what implementation would cost in terms of effort and fiscal

[25] https://github.com/galaxyproject/galaxy/
[26] https://github.com/galaxyproject/galaxy/blob/dev/SECURITY_POLICY.md
[27] https://sweguide.trustedci.org/

cost. During discussions with the Galaxy Project team, Trusted CI discussed the use of the CIA Triad (Confidentiality, Integrity, Availability) and how understanding how a project prioritizes these attributes can help it work towards its goals by prioritizing controls that focus on each of these attributes. The Galaxy Project team determined that their order of priority to help determine focus is on Integrity, then Availability, then Confidentiality. Thus, the recommendations below have also been ordered with these goals in mind.

3.3.1. Use principle of least privilege on services and components

When at all possible, a service should be running as its own less privileged user and should not be performing it's normal operations as the root user or equivalent. An example of this separation of privileges is in a webserver where it starts as root, but then sheds its root privileges and switches to a special user to process the request. This helps prevent a vulnerability in code becoming root level vulnerability.

Status: The CloudMan server has admin access to Kubernetes cluster

Recommendations: While understanding that the purpose of CloudMan is to create, start, stop and configure Kubernetes nodes, the exposure of this ability represents a significant risk to the overall system. We recommend working towards monitoring and limiting the actions of how the CloudMan system interacts with Kubernetes where possible. We also recommend treating access to CloudMan with the same level of control as you would a superuser account. For instance, required strong passwords and multi factor authentication.

3.3.2. HTTP Strict Transport Security should be enabled

From Wikipedia: HTTP Strict Transport Security (HSTS) is a web security policy mechanism that helps to protect websites against man-in-the-middle attacks such as protocol downgrade attacks and cookie hijacking.

Status: HTTP Strict Transport Security is enabled for the *locations* CloudMan, /cloudman, /grafana & /auth

Recommendations: Enable HTTP Strict Transport Security for *all* locations.

3.3.3. Remove unused/unneeded listening ports

A server should not be listening on ports other than those bound to the services the server is expected/documented to support. 'Backdoor' listening ports can be missed during maintenance and thus, lead to a security risk.

Status: The server appears to (additionally) be listening on port 8181

Recommendations: Determine if port 8181 is necessary and/or if it is exposed to the Internet. Document the needs to the port in both the configuration and in user documentation.

3.3.4 Restrict access to non-public network services

A server should only expose to the Internet those ports which it needs to. Exposing extra services to the public Internet increases the attack surface and puts the service at unnecessary risk.

Status: A scan of a Galaxy host revealed it was listening publicly on TCP ports 22, 80, 443, 2379, 2380, 6443, 10250, 10256, 30459, 30715, and 31330.

Recommendation: Determine the absolute minimum required ports that must be exposed to the Internet in order for Galaxy to function. Use a host firewall such as iptables or other methods of restricting access to only allow access to the ports that are necessary and block access to the other ports. The following Linux command, running on an external unrelated Linux host, was used to discover listening ports on a test/dev host provided by the Galaxy team:

- sudo nmap -sS -pT:1-65535 -T2 <target galaxy hostname>

3.3.5. Patch management

Software vulnerabilities are a leading cause of compromised systems and data. Proper patch management of software on systems can help reduce security risks.

Status: Galaxy does not regularly or automatically upgrade software on templates and deployed hosts.

Recommendation: Regularly update software on template systems so that users are deploying the latest available software. Consider automatically updating software as

recommended on deployed systems or notifying users when their system is out of date and requires critical patches to be applied.

3.3.6. Mitigate brute force authentication attacks

Malicious attackers are constantly trying to crack user passwords through brute forcing techniques. A common mitigation against this attack is to block an attacker after they reach a threshold for unsuccessful authentication attempts.

Status: The Galaxy implementation of Keycloak is not configured to rate limit authentication failures. SSH brute force attacks are also not prevented.

Recommendation: Enable by default Keycloak rate limiting settings such that client hosts are blocked after a set number of unsuccessful authentication attempts. Either use Keycloak's default recommended value for the threshold or choose a reasonable value, such as failed 10 attempts. As an additional measure, authentication on other services such as SSH and FTP can be rate limited using the fail2ban service.

3.3.7. Run a CIS scan on cluster

The CIS Kubernetes Benchmark[28] is a set of best practices that can be used to establish a secure configuration baseline for a cluster. The 'kube-bench' application can be used to perform a security scan against the CIS Benchmark and provide a report with remediation steps for failed tests.

Status: Unknown.

Recommendations: Run kube-bench[29] to check deployment for security issues against the CIS Kubernetes Benchmark.[30] This can be done within Rancher.[31]

3.3.8. Implement container scanning

This is a re-statement from the "SysCall7 Galaxy/Cloudman Security Assessment" report:

"Docker images used in production systems should be scanned for potential vulnerabilities."

---

[28] https://www.cisecurity.org/benchmark/kubernetes/
[29] https://github.com/aquasecurity/kube-bench
[30] https://www.cisecurity.org/benchmark/kubernetes/
[31] https://rancher.com/docs/rancher/v2.x/en/cis-scans/

Images unscanned for vulnerabilities could expose those vulnerabilities to the underlying host and-or cluster if not identified and patched.

Status: Container scanning is not being performed.

Recommendations: From the SysCall7 report: "Snyk.io is a useful tool for this process but only is available for certain systems (e.g. DockerHub). For images that are not supported by snyk.io, there are other scanning tools that can be used to discover vulnerabilities. An example is https://github.com/anchore/anchore-engine which can scan arbitrary Docker images and even supports running in a client environment."

3.3.9. Images should not be run as root

If a Docker image is run as root, then any process within that image can be run with super-user privileges. This behavior is easy to exploit by malicious users and-or applications.

Status: All Galaxy tool containers run as root. Unknown if other services do.

Recommendations: Create appropriate user to run as and execute the image as that user.

3.3.10. Use CIS Benchmark and Self-Assessment to evaluate cluster controls

The benchmark self-assessment helps you evaluate the level of security of the hardened cluster.

Status: Self-assessment was not completed.

Recommendations: Use the self assessment guide on the Rancher documentation site to audit CIS controls.[32]

3.3.11. Encrypted communications should be used

[32] https://rancher.com/docs/rancher/v2.x/en/security/#cis-benchmark-rancher-self-assessment

Network traffic should have resilience to attack; TLS/SSL provides authenticated checksums and other sequence protections to ensure traffic has not been tampered with.

Status: The reverse-proxy is configured to redirect (i.e., 'rewrite') all HTTP requests to HTTPS, thus ensuring that all communications are encrypted. Moreover, Server-side Cipher-suites ('ssl-prefer-server-ciphers') is enforced, the number of TLS handshakes ('keepalive-timeouts') is enabled, and the session is kept alive for 10 mins ('ssl_session_cache' and 'ssl_session_timeout').

Recommendations:

1. It may behoove Galaxy to verify that TLS 1.0 has been deprecated, for there are known exploits for TLS 1.0.[33]

3.3.12. Implement multi factor authentication for Galaxy users

Multi factor authentication most commonly implemented as two factor authentication helps prevent user credential compromise from completely providing access to an account. Many security conscious users wish to further control access to their account. Furthermore, projects and institutions may have policy requirements that call for the need to require the use of two factor authentication.

Status: Galaxy's implementation of Keycloak does not allow multi factor authentication nor enforce it.

Recommendation: Enable or implement the option within Keycloak for two-factor authentication and allow sites to enforce the use of two-factor authentication.

3.3.13. Harden nodes

Malicious actors can potentially abuse access to kublet and the Kubernetes API to run unauthorized applications.

Status: TCP port 10250 is exposed to the Internet.

Recommendations: Restrict network access to kublet (ports 10250, 10255, and 10256) and require authentication and authorization to access the Kubernetes API.

[33] https://www.comodo.com/e-commerce/ssl-certificates/tls-1-deprecation.php

3.3.14. Keep Rancher (and other components) patched and updated

Vulnerabilities in services that are exposed to users need to be patched ASAP in order to prevent malicious actors from exploiting said vulnerabilities.

Status: Rancher is currently patched ad-hoc. Moreover, there is no service admin whose responsibilities include monitoring for security patches.

Recommendation: Write policy that states that Rancher must be updated expediently and assign the responsibility for patch management to a specific admin.

3.3.15. Implement Rancher Hardening Guide

The Rancher Hardening Guide provides prescriptive instruction for hardening a production instance of Rancher. This is to be used in conjunction with a Kubernetes CIS scan and Rancher Benchmark Self-Assessment.

Status: Unknown

Recommendations: Review, and implement policy and configuration from the Rancher Hardening guide specific to the versions of Rancher/Kubernetes installed.[34]

3.3.16. Use Kubeapps

Kubeapps is a web-based UI for deploying and managing applications in Kubernetes clusters. It provides a way to deploy Helm charts, administer Helm-based applications, and allows for fine grained access control and authorization using RBAC.

Status: Unknown.

Recommendations: Install and configure Kubeapps within Helm. [35]

3.3.17zoom. Restrict cloud provider credentials to privileged pods

This is a re-statement from the "SysCall7 Galaxy/Cloudman Security Assessment" report:

[34] https://rancher.com/docs/rancher/v2.x/en/security/#rancher-hardening-guide
[35] https://github.com/kubeapps/kubeapps/blob/master/docs/user/getting-started.md

“Using EC2-based IAM roles allows any pod within the cluster to have authorized access to the cloud provider API. Users able to run commands within pods can then interact with those resources in unintended ways. This applies specifically to the ‘cm2-kube-role’ role being used to grant access to the infrastructure services like nginx-ingress. This was demonstrated to be exploitable using a service like TerminalMan which can be configured to run arbitrary aws commands.”

Status: Roles are not restricted within the pod.

Recommendations: See the remediations listed in the ‘Restrict cloud provider credentials to privileged pods’ section within the SysCall7 report.

3.3.18. Provide users with the ability to limit access to specific IPs or netblocks

Services that don't require access from the internet as a whole as needlessly exposing these services to potential attacks.

Status: The Galaxy Kubernetes Deployment doesn't offer this option other than logging in and configuring the host firewall or Kubernetes host access restrictions.

Recommendation: Considering both host firewall access controls and Kubernetes based controls, offer users the ability to limit access to the cluster by IP or netblock at install time.

3.3.19. Check user input for command shell sequences

Passing user submitted data into functions that make use of command shells to execute programs can be vulnerable to command injection[36] (also called shell injection) attacks. Allowing characters such as semicolon, backticks, dollar signs and parenthesis.

Status: The Galaxy job submission system uses commands such as 'sed'.

Recommendation: Ensure that all user input that makes its way into a process involving command execution has been checked for the shell syntax characters as mentioned and that they are properly escaped or sanitized.

[36] https://owasp.org/www-community/attacks/Command_Injection

3.3.20. Enable Audit Logging for API calls

Audit logging provides information necessary for forensic analysis after a potential malicious event. Note, this is different from Kubernetes 'audit logging'.

Status: Unknown.

Recommendations: Audit Logging should be enabled within Rancher via the instructions posted on Rancher's online documentation.[37]

3.3.21. Provide a centralized logging option

Service and program log files are more vulnerable to compromise if they are kept only on the host on which they are generated. They can be better protected, accessed, and analyzed by sending them to a centralized log host. Some projects and institutions also prefer or have policies to centralize log files.

Status: Galaxy does not offer the option to send log files to a central log host.

Recommendation: Provide administrators with an option to send service log files to a central log host over encrypted transport. This is most commonly accomplished on Linux based systems using the syslog facility and custom configuration. This is especially important for the Keycloak, CloudMan, Galaxy, Kubernetes, and host services.

3.3.22. Implement resource limits

Enabling resource limits protects against resource starvation which can lead to Denial of Service (DoS) attacks.

Status: Resource limits are defined for the Galaxy resources. Status of other services is unknown.

Recommendations: Ensure that 'resource limits' are enabled by following the instructions posted on Rancher's online documentation:[38]

[37] https://rancher.com/docs/rancher/v2.x/en/installation/resources/advanced/api-audit-log/
[38] https://rancher.com/docs/rancher/v2.x/en/project-admin/resource-quotas/

3.3.23. Use unique namespaces in Kubernetes

Namespaces allow for segregation and segmentation of clusters; this allows different users, groups, or organizations to deploy multiple clusters without fear of exposure of data. Additionally, this makes resource management and access control more tractable.

Status: Galaxy is using namespaces

Recommendations: Galaxy should continue to use namespaces.

3.3.24. Run clusterwide Pod Security Policy (PSP)

Pod Security Policies are clusterwide resources that control what capabilities and configuration Pods must have in order to run within a cluster. These can be used to prevent a malicious actor from performing privilege escalation, breaking out of container isolation, and accessing other pods or services.

Status: Unknown

Recommendations: Enable Pod Security Policies in Kubernetes (through Rancher)[39] [40]

3.3.25 Check TLS security Best Practices for nginx

The Nginx server is exposed to the Internet, and due to this exposure, the implementation requires detailed examination. One component of this examination is a comparison of Galaxy's Nginx (current) configuration to the 'best practices' suggested.

Status: Unknown

Recommendation: Check published best practices for using TLS with nginx.[41]

[39] https://rancher.com/blog/2020/pod-security-policies-part-1/
[40] https://rancher.com/blog/2020/pod-security-policies-part-2
[41] https://www.linode.com/docs/web-servers/nginx/tls-deployment-best-practices-for-nginx/

3.3.26 Check Security Best Practices for RabbitMQ

RabbitMQ represents a common point of access and data transfer between nodes in the Galaxy cluster, because of this it is crucial to ensure that it is following security best practices.

Status: Unknown

Recommendations: Trusted CI recommends looking at the best practices prescribed by the RabbitMQ project to ensure that the Galaxy installation is in alignment. These are available at https://www.rabbitmq.com/best-practices.html

3.3.27. Online Certificate Status Protocol Stapling should be enabled

The Online Certificate Status Protocol (OCSP) is designed to provide a mechanism for obtaining the revocation status of X.509 certificates (replacing Certificate Revocation Lists). NGINX can cache browser requests to OCSP server,

Status: It does not appear that the configuration has OCSP Stapling enabled.

Recommendations: Enable OCSP Stapling.[42]

3.3.28. Trust between components

The openly communicating components behind the firewall should use a robust mutual trust mechanism such as digital certificates.

Status: Some Galaxy deployments are not authenticating themselves between components.

Recommendations: Require that services communicating within the cluster are authorizing each other.

3.3.29. HTTProxy should be mitigated against

[42] https://www.linode.com/docs/web-servers/nginx/tls-deployment-best-practices-for-nginx/

HTTProxy is a vulnerability that attempts to 'proxy' traffic to a site under the control of a malicious actor. Many servers were vulnerable to it, but NGINX has features in place to thwart it.

Status: The server is configured to mitigate against HTTProxy.

Recommendations: Continue to mitigate against this vulnerability.

3.4 General System Administration

3.4.1. Hardening Hosts

Systems & hosts should be resistant to malicious attacks.

Status: Unknown

Recommendation(s): There are many techniques that can be used to harden hosts. At a minimum, the following should be employed:

- Enable Multi-factor Authentication (MFA) or utilize a bastion host for all administrator privileged access
- Enable centralized logging in order for forensic analysis in incident response; note, this was also cited in Cloudman security review
- Employ managed (e.g., scheduled) patch maintenance in order to address security updates for all software components
- Periodically update all 'shared' SSH keys in order to mitigate against stolen/compromised private keys; additionally, private keys should be password protected, and beholden to the same password policies used for administrative user accounts.

3.4.2. Process / Image / Network segmentation

All processes, images and networks should be segmented in order to ameliorate a malicious adversaries ability to move laterally through your system.

Status: Unknown

Recommendations: Based on the asset, there are many techniques to implement segmentation, since Galaxy predominantly relies on containers and networks, we

suggest the following: containers should only have access to what they need; and networks should be segmented using k8s pod security.

3.4.3. Repo best practices

External repositories, e.g., GitHub, provides a high level of convenience, but by their nature of being external, precautions must be taken to avoid exposing confidential information and-or granting unauthorized access to code that should remain protected. Moreover, since the repo usually houses source code and its dependencies, this provides an opportunity to monitor for security patch management.

Status: Unknown

Recommendations: The following actions/utilities should be enabled on eternal repositories:

- In Github, under Security, enable Security Advisories - a service that looks for secrets/credentials in code, as well as known vulnerabilities in used in dependent modules/packages.
- Additionally (or alternative to Secbot) enabled automated check for vulnerabilities with Clair [43], Snyk.io [44], or similar.
- Finally, subscribe to vulnerability mailing lists of all dependent, external software components, e.g., docker, Kubernetes, Rancher, Helm, etc.

3.4.4. Audit failures

By monitoring system logs periodically, they can be used as a method (behavioral analysis) to detect not only failures in systems, but also malicious attacks.

Status: Unknown

Recommendation: With centralized logging implemented (see 4.2.1. Harden Hosts), additional utility can be gleaned through mechanisms like a Security Information and Event Manager (SIEM) that parse logs for failures in critical systems, as well as look for abnormalities (a typical sign of an attack). It is also advisable to configure an event logger (e.g., auditd[45]) to monitor any logs produced by critical services. Also, the system in place should be reviewed to ensure logs are not missed.

[43] https://github.com/quay/clair
[44] https://snyk.io/
[45] https://linux.die.net/man/8/auditd

3.5 Review of usegalaxy.org

A high-level, non-privileged audit was performed against the usegalaxy.org site, both through its web interface, and also on the virtual machines that support these services. No extensive service scanning was performed to limit the chance of affecting the production system, as such, no commands were run with privileged access, which limits the information available in the audit process.

Positive takeaways

- 2FA is required for SSH access - TACC credentials and TACC Token
- RSA authentication is required for sudo/privilege escalation
- limited number of users in docker group
- remote syslogging is enabled on VMs
- non-root user used to run processes
- docker not exposed to network - local, file sockets
- minimal ports exposed to internet (22,80,443)

usegalaxy.org web service

The biggest attack surface is user input when working with tools; both the vetting of the tools as well as validating and sanitizing user input, which are being done by the Galaxy Project team.

Recommendations:

- Tighten up OpenSSL configuration - Remove support for TLS 1.0 and 1.1
    - “B” rating on Qualys OpenSSL server test [46]
- Run a web vulnerability scanning and fuzzing tool against the web services
    - nikto [47]
    - w3af [48]
    - Arachni [49]

usegalaxy.org VMs

The VMs have restrictive access control which greatly reduces the attack surface, so the biggest risk becomes the services that are exposed to the network. Keeping those

[46] https://www.ssllabs.com/ssltest/analyze.html?d=usegalaxy.org&s=129.114.60.56&latest
[47] https://github.com/sullo/nikto
[48] https://w3af.org/download
[49] https://www.arachni-scanner.com/download/

services patched and upgraded is crucial, as well as protecting and auditing those services.

Recommendations:

- Upgrade to CentOS 7.9 (recently released) - Running an old OS, package upgrades available (httpd/docker/mariadb/cvmfs).
- Regularly audit access to docker group members - `g2test,g2main`
- Install and configure fail2ban to protect exposed services that require authentication i.e. SSH
- Ensure service logs are being sent to remote syslog server
    - i.e. apache by default only logs to local files
- Run security auditing tools (as root user) to assist in identifying vulnerabilities, misconfigurations, etc
    - lynis [50]
    - docker-bench-security [51]
    - kube-bench [52] (if deployed with Kubernetes)

4. Primary Use Cases

4.1 User Operation

The primary use-case scenario is that a user connects to the Galaxy service that has been previously instantiated through CloudLaunch (see Appendix A CloudMan Deployment):

1. The user initiates a connection to the Galaxy instance. The user's HTTPS Galaxy connection (HTTP is redirected to HTTPS) terminates on a Nginx reverse-proxy for the Galaxy Kubernetes service, before being relayed from Nginx to the Galaxy service.
    a. The *internal* communication between Nginx and Galaxy is not encrypted.
    b. The Galaxy certificate (used as an endpoint on the Nginx server) is loaded into the Nginx container by Kubernetes.
2. If the user is not authenticated, the Galaxy server redirects the user's browser to the Keycloak service, where the user authenticates.

[50] https://github.com/CISOfy/Lynis
[51] https://github.com/docker/docker-bench-security
[52] https://github.com/aquasecurity/kube-bench

3. With valid authN credentials, the user accesses Galaxy and starts up a job (a job includes a variety of operations including data upload or execution of any tool installed in Galaxy). The job is propagated to the Kubernetes cluster for execution.
    a. A Docker image corresponding to the tool specified by the job is retrieved from a Docker image repository (hub.docker.com or quay.io most likely).
    b. The job can read/write to the network file system where input data is available from the job working directory. The job can also read data available on the CVMFS.
    c. Job outputs are stored in the job working directory on the network file system.
    d. Galaxy monitors the status of the job every several seconds. Once the job completes, Galaxy copies the outputs from the job working directory into a persistent data directory configured for Galaxy.

Alternate operation(s)

1. An admin user connects to CloudMan to monitor and manage the Kubernetes cluster. The user must be authenticated, using the same mechanism as with Galaxy except that CloudMan handles the authentication redirect to Keycloak. Once authenticated, the user may perform the following actions:
    a. Change Galaxy configurations settings. Once a change is made, CloudMan will apply the changes by updating the Galaxy configuration settings and restarting the Galaxy processes. These actions are performed using Helm.
    b. Change the size of the compute cluster by adding or removing cluster nodes. Once a change is initiated, CloudMan will (1) drain and cordon a node through the Kubernetes APIbefore deleting it; or (2) create a new virtual machine, configure it for inclusion in the cluster, and inform Kubernetes of the node's availability for inclusion in the cluster. In addition to changing the size of the cluster manually, the user may toggle use of auto-scaling for the cluster. The user selects the scaling limits and size of virtual machine of use before enabling the auto-scaling service.
    c. Monitor the status of the cluster through a built-in Grafana dashboard where information about the CPU, memory, and network usage is available.
2. While the Galaxy deployment is running, Grafana collects stats from the Galaxy database (which is populated from Kubernetes stats collection) and the Kubernetes cluster. The user can query Grafana to see the stats. As above,

nginx terminates the TLS connection and relays the communication to Grafana. If the user is not yet authenticated, Grafana redirects the user to Keycloak. There the user can obtain stats regarding the job or cluster status.

3. An admin user may add or remove system users. Via Keycloak, the admin may add new users, update their roles and security credentials. The admin may also disable access for users. Keycloak can also be used to add additional authentication, external, providers that support OAuth2 authentication protocol.

4.2. HIPAA Security Rule Compliance

Status: Trusted CI's gap analysis shows that full HIPAA Security Rule compliance for Galaxy will primarily depend on how a site implements the software, but that there are a number of gaps in the Galaxy architecture and project that should be addressed.

Recommendations: Address existing gaps to make the Galaxy architecture and project more capable of handling protected health information itself, and for making it easier for implementers to comply with HIPAA. A list of gaps and recommendations to fill each is provided in the accompanying HIPAA Security Rule Gap Analysis report [3].

5. Future Engagement Opportunities

The scope of Trusted CI's work in this engagement was limited to reviewing the Galaxy intrinsic architecture and data flows. In part due to both the engagement's scope and upon developing recommendations, a few areas emerged which could be the basis for future work. The topics follow:

5.1 Galaxy Code assessment

This engagement focused more on the components that make up a Galaxy installation and didn't provide any type of source code review or static analysis. Trusted CI thinks it would be highly beneficial to the Galaxy project to engage in the future in performing a code assessment of the Galaxy core code. This could take the form of a Software Assurance focused engagement offered through Trusted CI.

5.2. Review of an NSF site implementing Galaxy

This engagement focused mostly on a single deployment model of Galaxy, which is the Kubernetes Deployment model. Existing sites using Galaxy for important work may not

immediately see the benefits of this engagement. Thus, it would be helpful to NSF science to work with a project that is already using Galaxy to help them align with the latest best practices.

5.3 Future alignment process

Aligning new features for inclusion into the Galaxy framework may be non-trivial and require guidance. This provides another potential opportunity for Galaxy to engage with Trusted CI.

5.4 ToolShed review

ToolShed is an external service provided to Galaxy installations for the purpose of installing additional tools into a Galaxy . This model is commonly referred to as an App Store. App stores have known security issues and considerations must be made to ensure that the software offered is of the highest integrity. Reviewing the Galaxy ToolShed system would be a worthwhile engagement with Trusted CI or the Science Gateways Community Institute[53]

5.5 Galaxy CVMFS system review

Galaxy installations also make use of the CERN VM filesystem for storing common data that can be used by the community. This is another potential ingress point for malicious data and could benefit from an engagement either through Trusted CI or the Science Gateways Community Institute.

5.6 More in depth review of usegalaxy.org

The usegalaxy.org website is a community science gateway that is a specific instance of the Galaxy software. Although we performed a brief review of this website. It could be beneficial to conduct a more serious in depth review. An alternative to a Trusted CI engagement would be a lighter weight cybersecurity engagement through the Science Gateways Community Institute

[53] https://sciencegateways.org/

6. References and Artifacts

The following are references to other documents and artifacts that were either provided by the Galaxy Project team, developed with the engagement, or provided by Trusted CI as a delivery of the engagement.

- [1] SysCall7 document provided by Galaxy Project team
- [2] Galaxy Kubernetes Data Flow Diagrams active document
- [3] HIPAA Security Rule Gap Analysis of Galaxy Architecture/Project
- [4] System Security Plan template

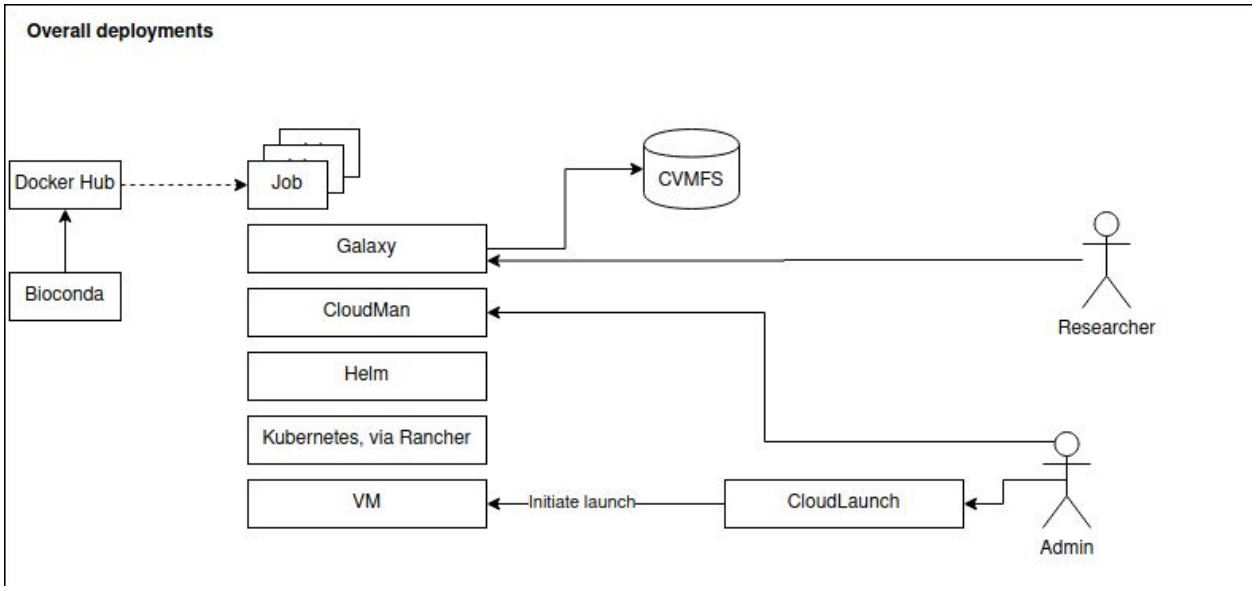
Conclusion

On behalf of Trusted CI and the Science Gateways Community Institute, the engagement team would like to thank the Galaxy Project team for its openness to the engagement process and hard work during the last 6 months. We are proud of the progress the Galaxy Project team has taken towards documenting their processes and open attitude with which they approached each issue. We hope that the Galaxy Project team finds these recommendations and accompanying documents useful and that it helps to improve the overall security of the Galaxy community. We look forward to further collaborations in years to come.

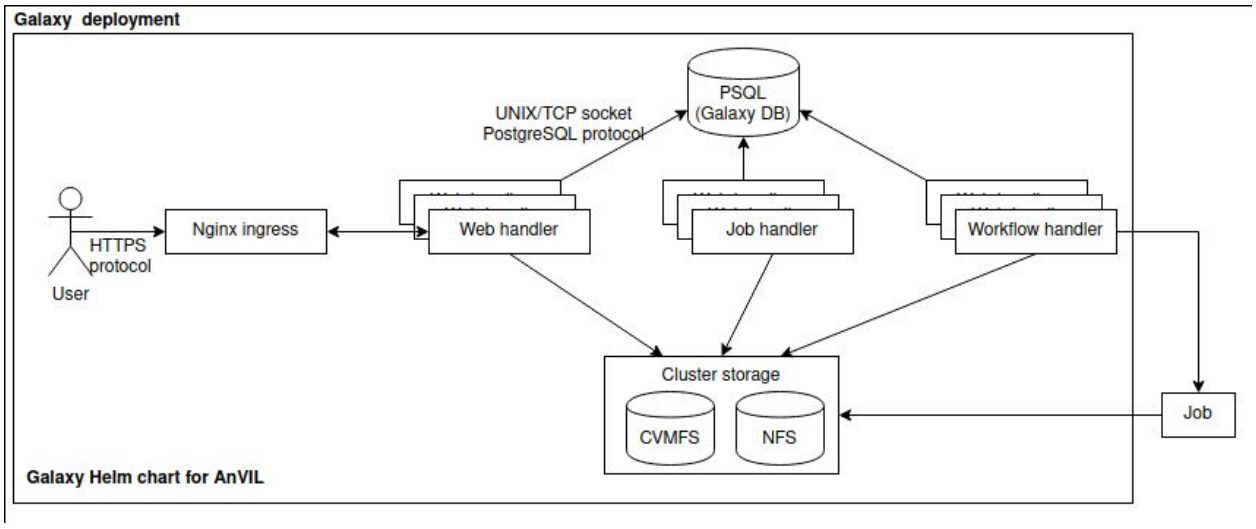
Appendix A. Galaxy Kubernetes Data Flow Diagrams

The following diagrams detailing actions and process flows of a Galaxy Kubernetes Deployment were created by the Galaxy Project team as part of the engagement with Trusted CI.

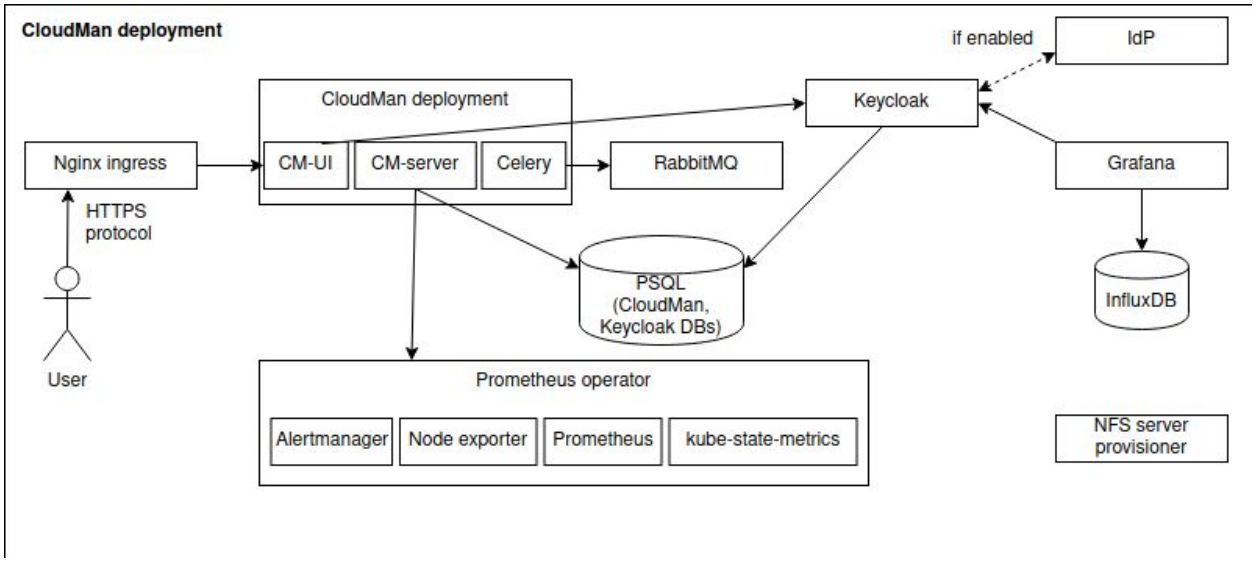
Overall deployments



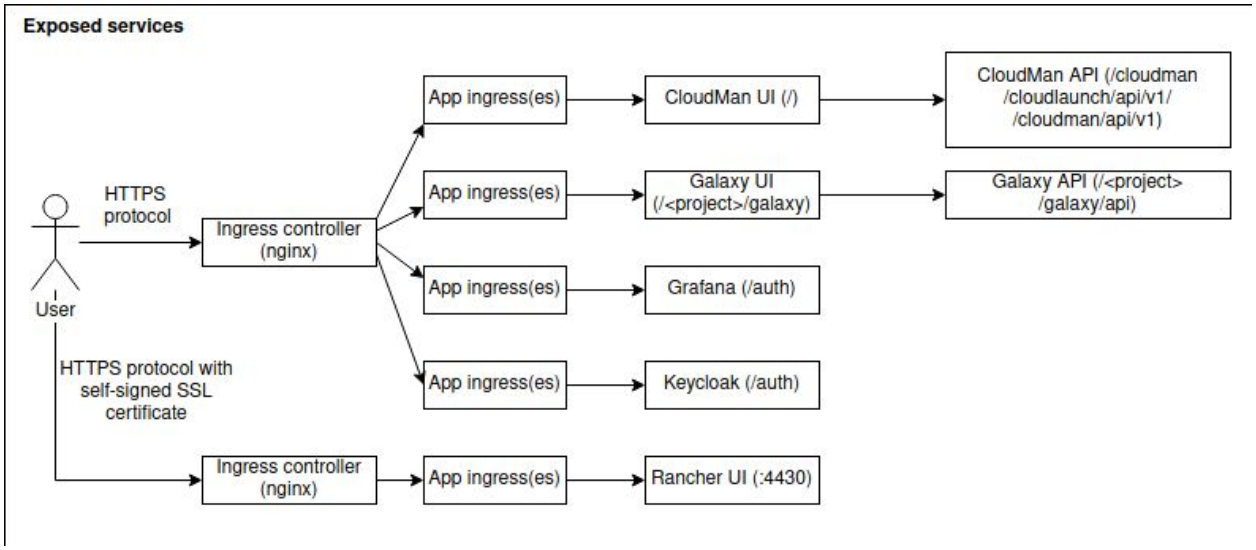
Galaxy deployment



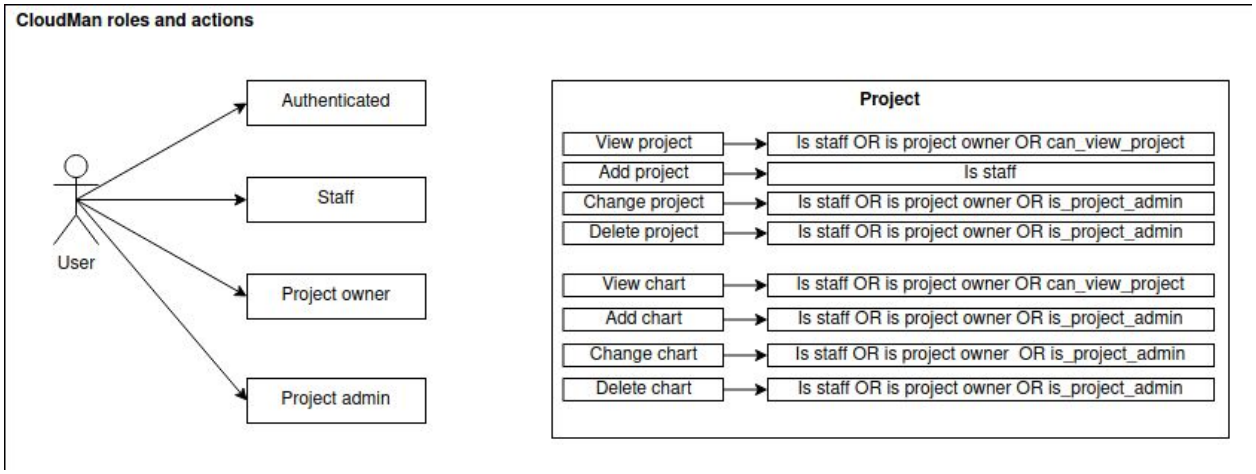
CloudMan deployment



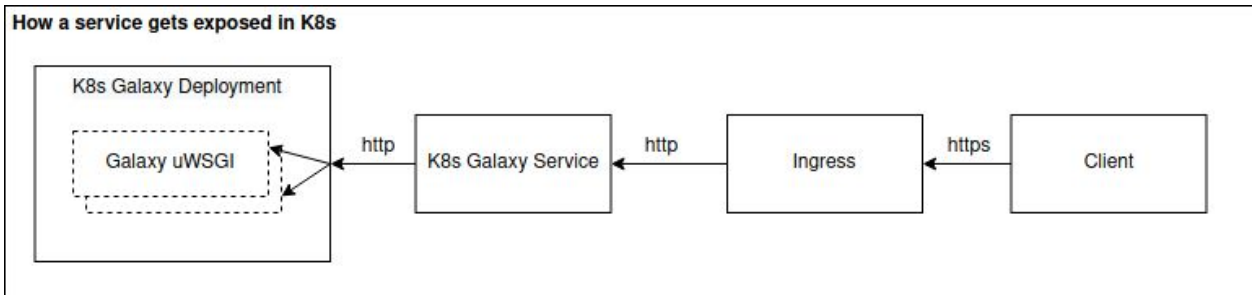
Exposed services



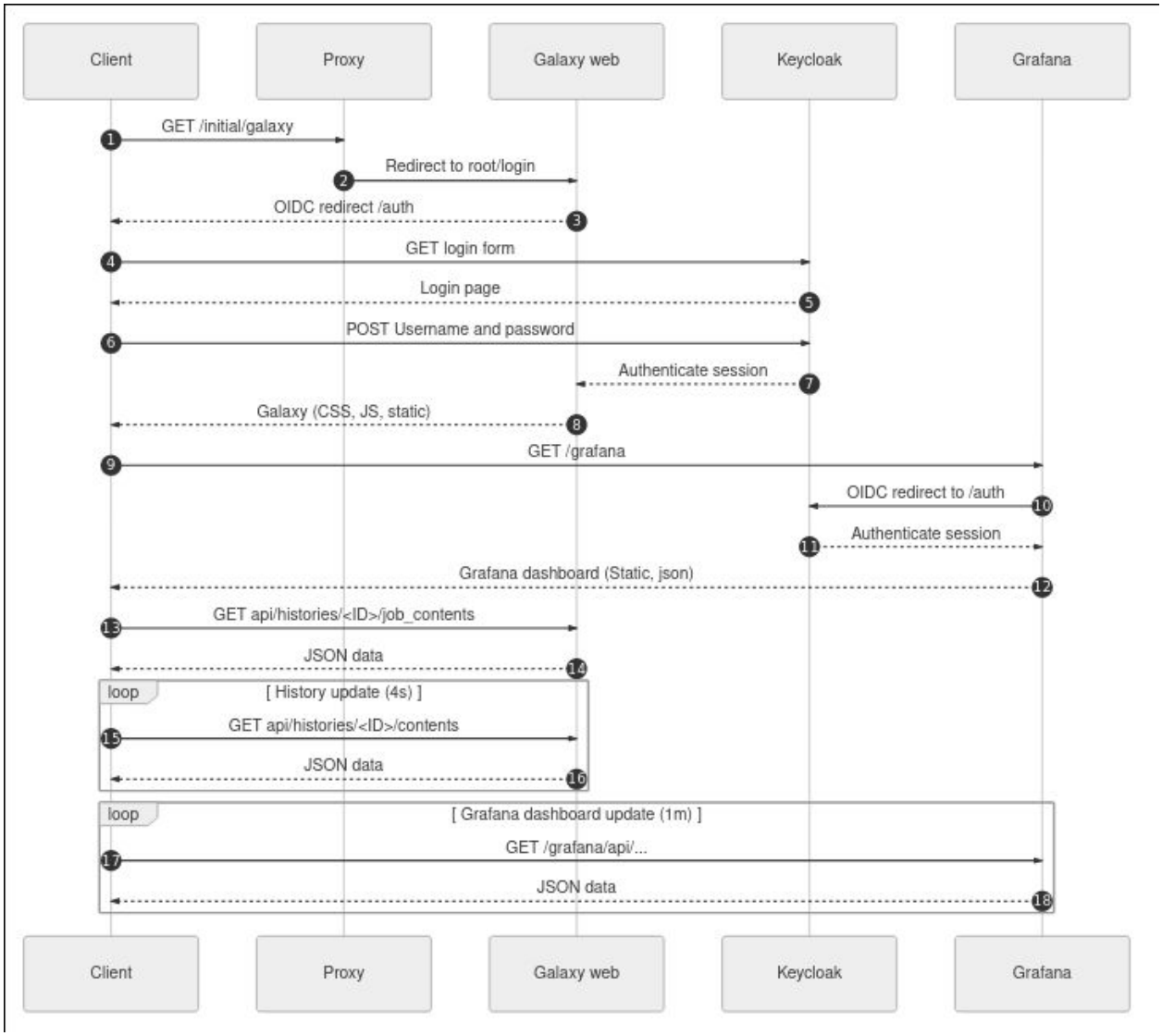
CloudMan roles and actions



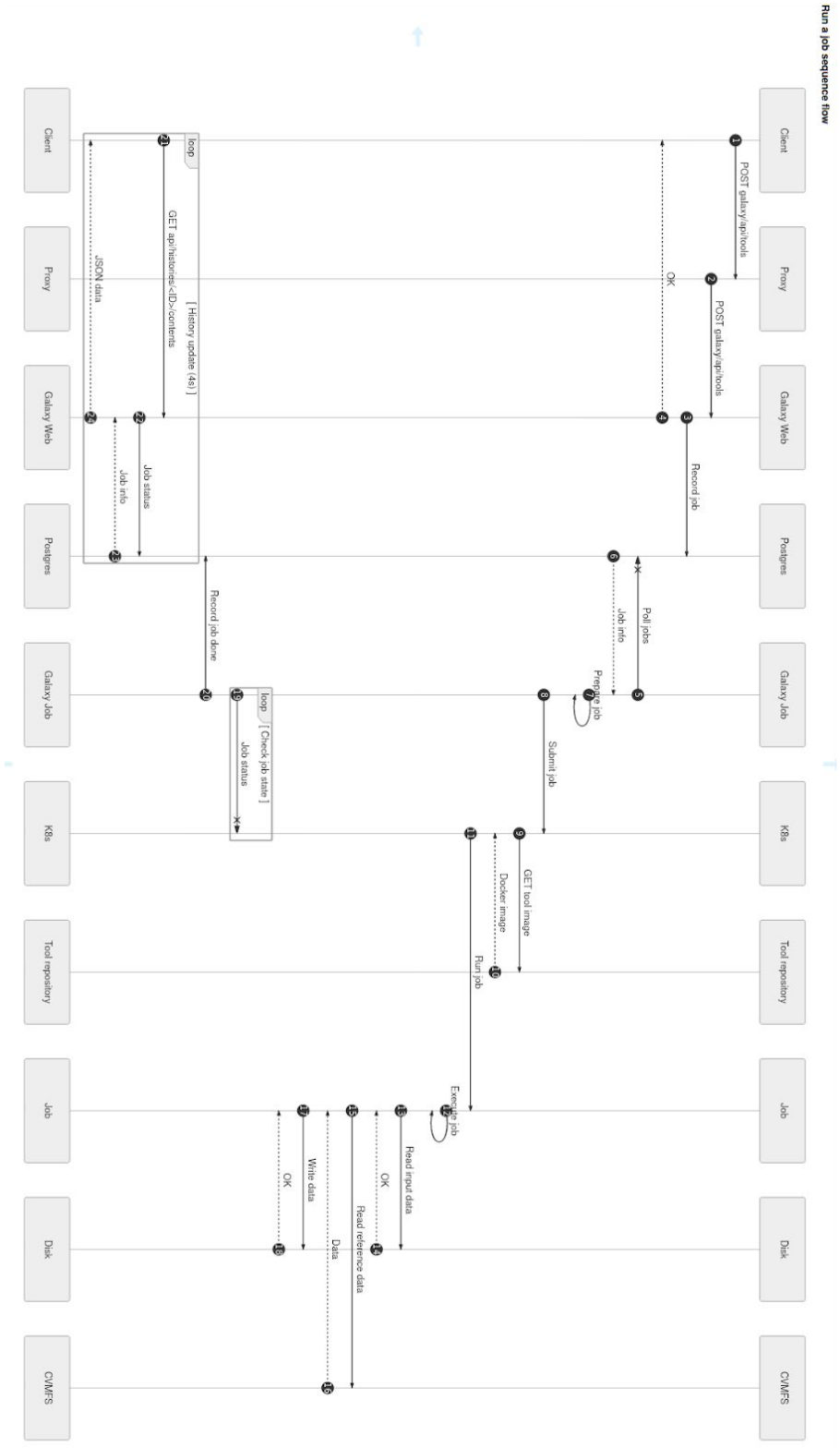
How a service gets exposed in K8s



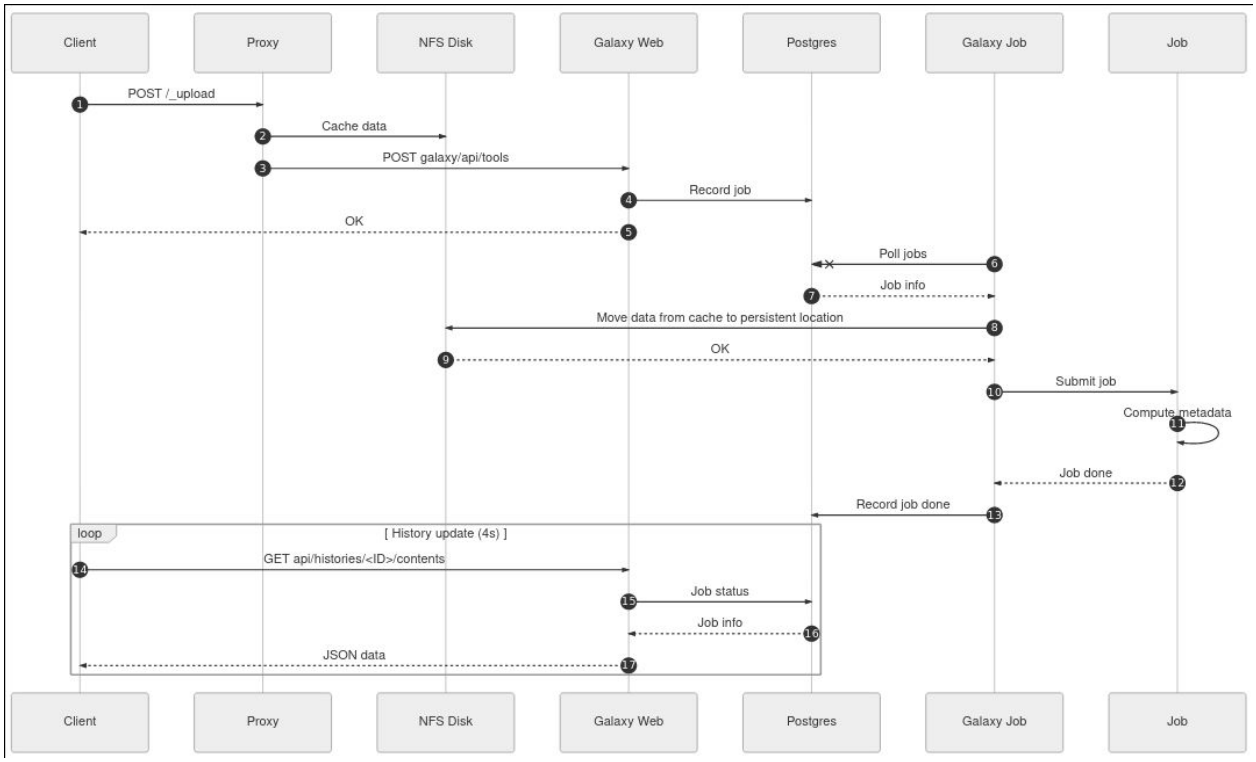
Authentication sequence flow



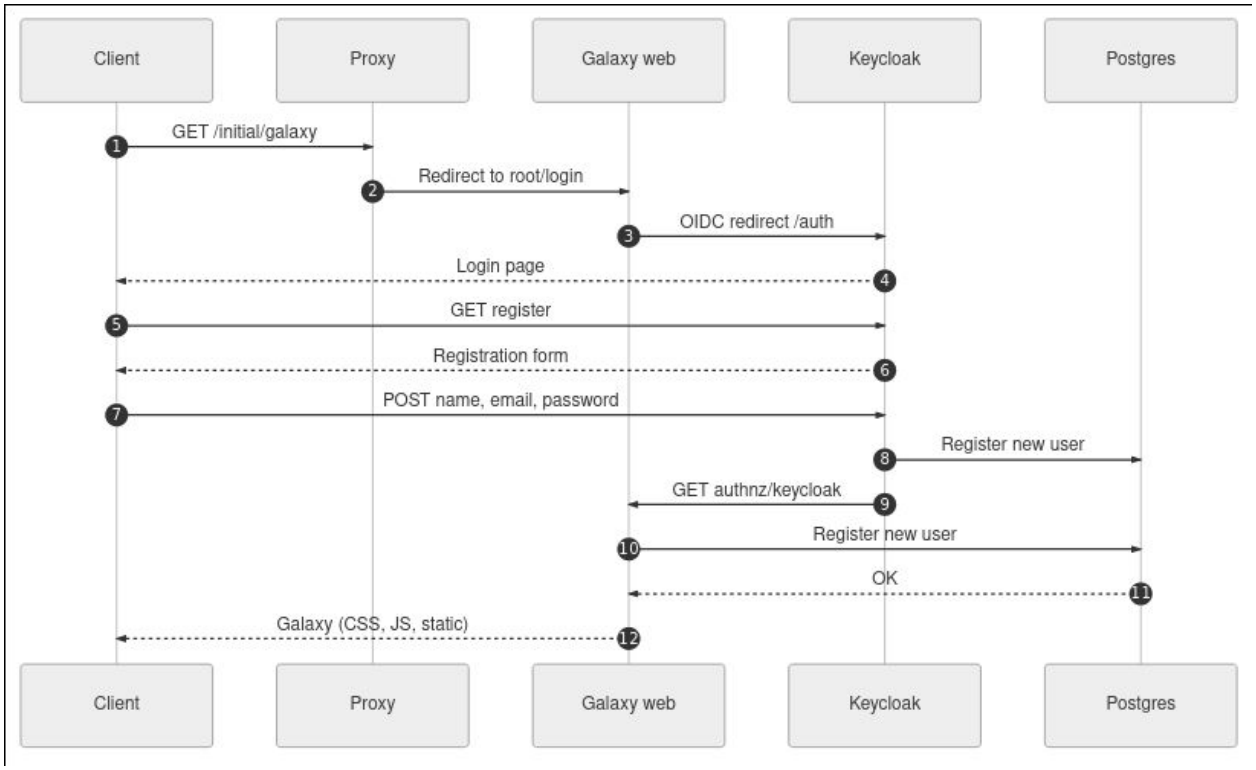
Run a job sequence flow



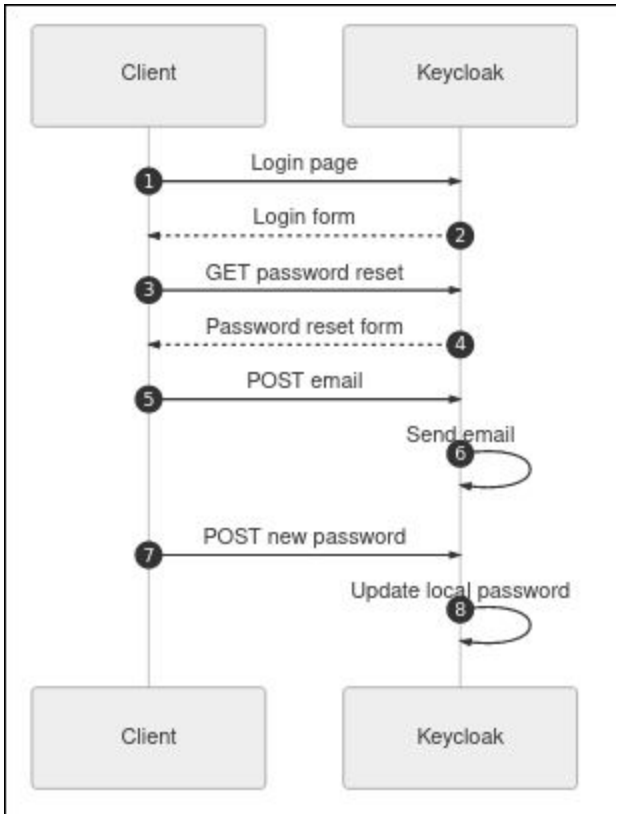
Upload Data Flow



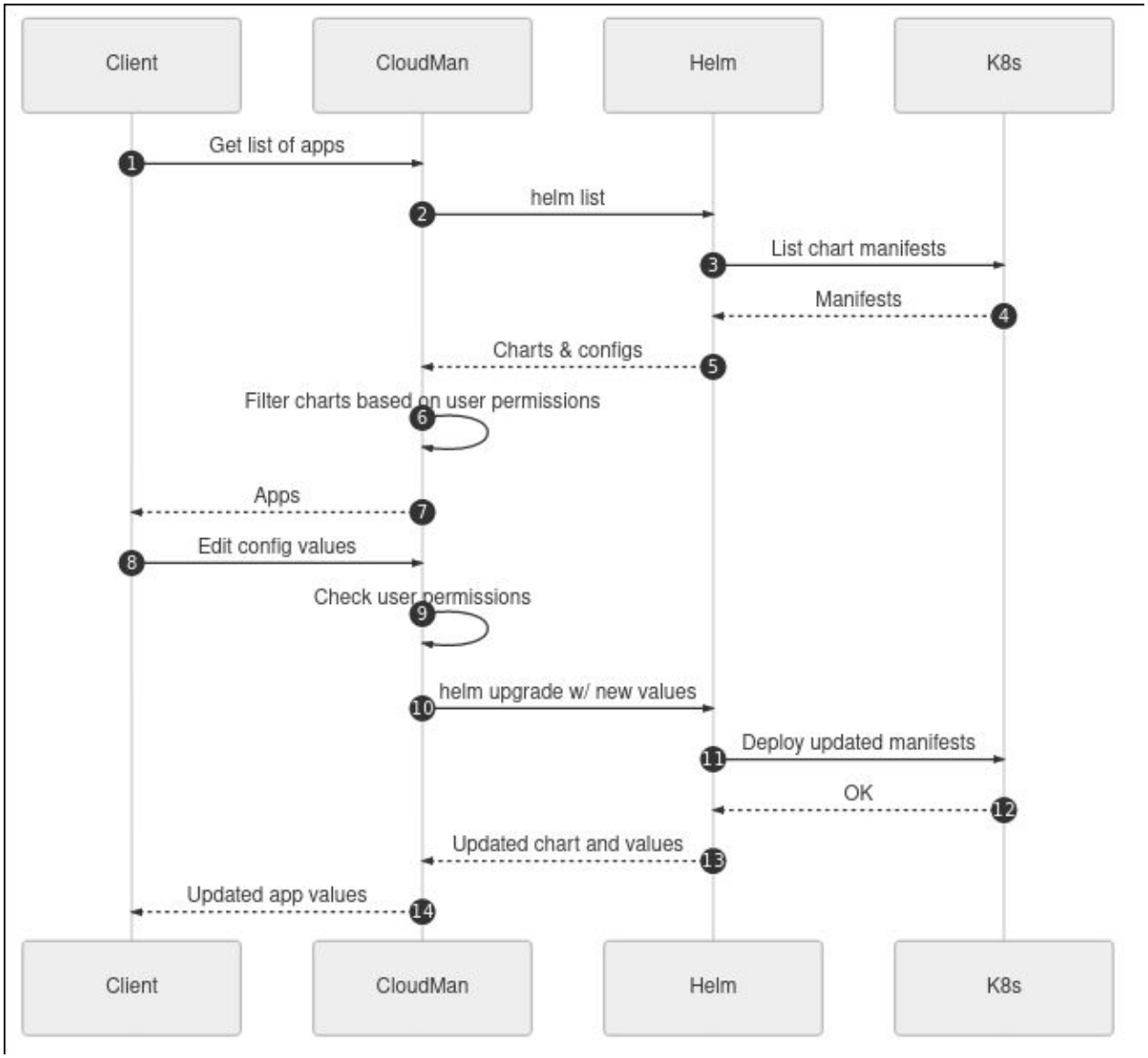
New User Registration



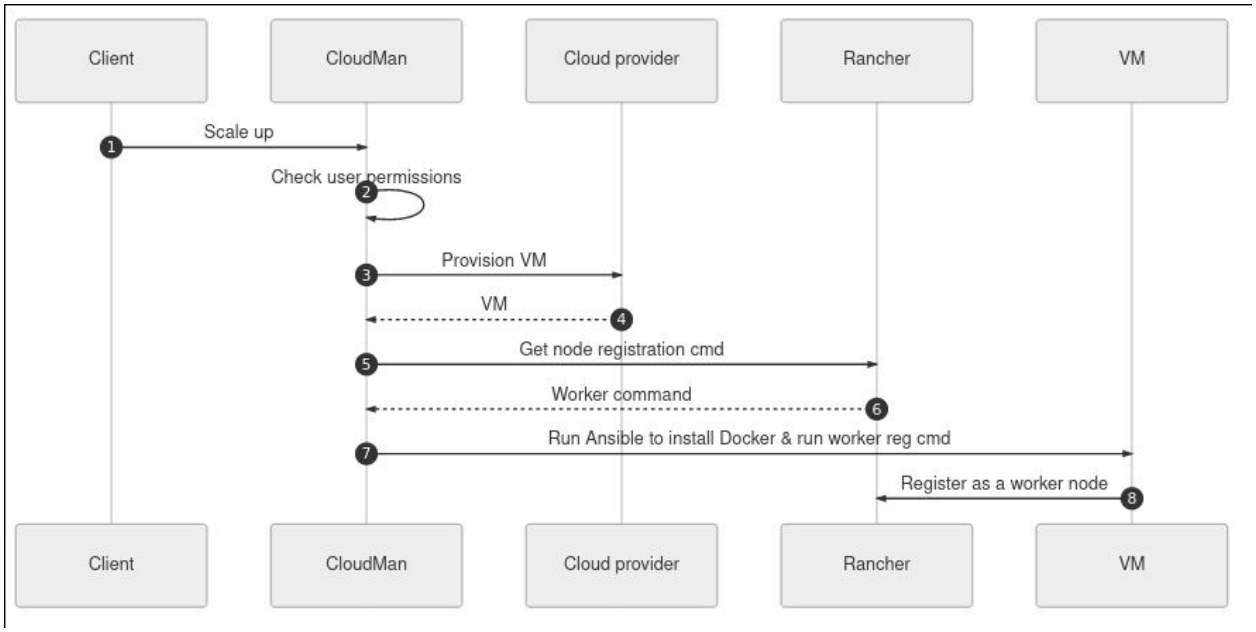
User password reset



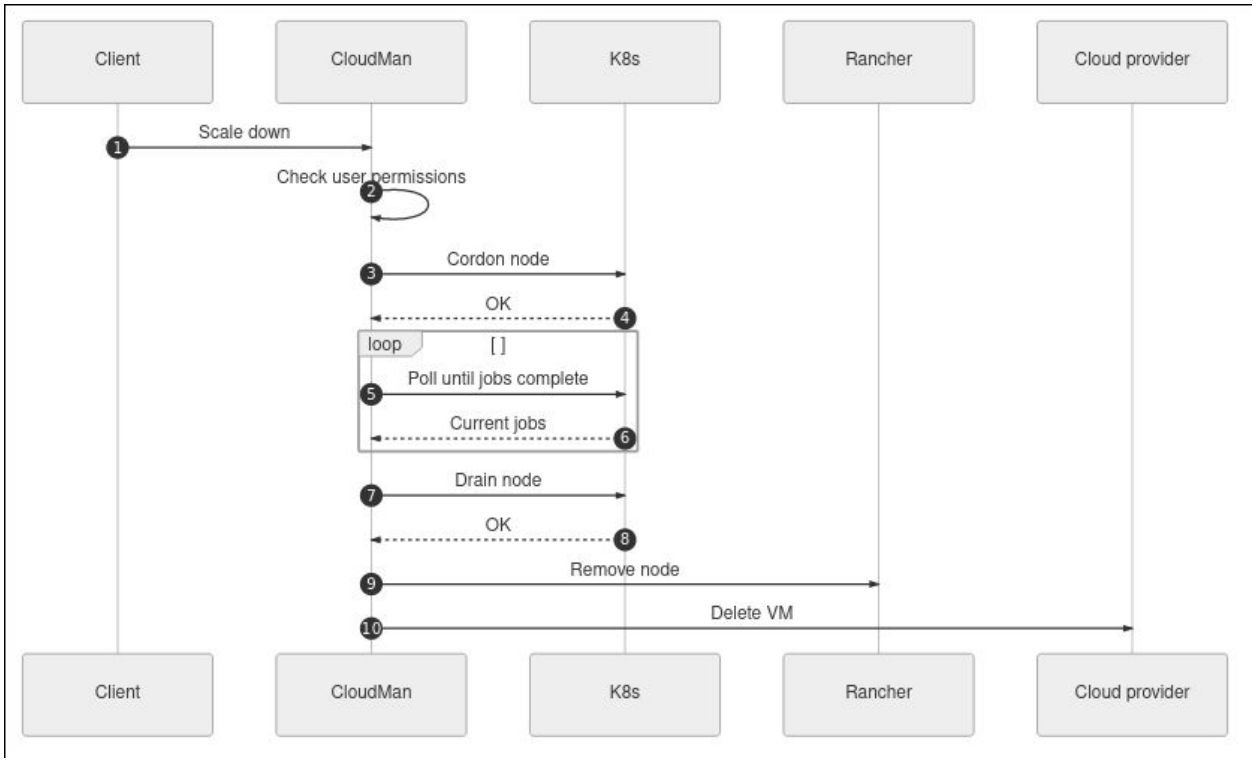
Update Galaxy Configs via CloudMan



Scale up cluster via CloudMan



Scale down cluster via CloudMan



Auto-scaling