info@shapeblue.com +44(0) 20 3603 0540

Monitoring **Apache CloudStack** & Hypervisors

Summary

This whitepaper explores the intricacies of monitoring Apache CloudStack, focusing on the use of Prometheus exporters and Zabbix as effective monitoring solutions. The document provides an overview of Apache CloudStack, hypervisors, Prometheus and Zabbix, followed by a comparison of the two monitoring tools, and a step-by-step guide to integrate them with CloudStack and hypervisors for efficient performance monitoring.

Table of Contents

1. Introduction

1.1 Apache CloudStack

Apache CloudStack is an open-source, multi-tenant cloud management platform that enables the deployment and management of scalable and reliable cloud infrastructure. It offers a comprehensive solution for managing diverse virtualization environments, compute, storage, and networking resources, and is compatible with a variety of hypervisors, including KVM, VMware, Citrix Hypervisor and XCP-ng. CloudStack provides an easy-to-use web-based user interface and a robust API for managing and orchestrating virtual resources.

1.2 Hypervisors

Hypervisors are virtualization software that allows multiple virtual machines (VMs) to run on a single physical host. The hypervisor provides an abstraction layer between the underlying hardware and the VMs, managing the allocation of resources, such as CPU, memory, and storage. Some common hypervisors include KVM, VMware ESXi, Citrix Hypervisor, and XCP-ng.

1.3 Monitoring in Apache CloudStack

Monitoring is a crucial aspect of managing cloud infrastructure and virtualization environments. It helps administrators track the performance and health of resources, identify issues before they escalate, and optimize resource utilization. Monitoring solutions must provide visibility into Apache CloudStack components, such as management servers, hosts, and Instances, as well as hypervisor-specific metrics.

Here's a list of key components to monitor in the majority of cases:

- The management server OS
- CloudStack's Java process & JVM stats
- CloudStack's UI and API
- The DB server
- The hypervisors
- The network connecting it all
- The Instances - depending on circumstances, SLA and contractual agreements

1.4 The Importance of Monitoring

Effective monitoring of Apache CloudStack ensures that cloud infrastructure operates optimally, minimizing downtime and maximizing performance. By proactively identifying and resolving issues, administrators can maintain high service quality and prevent potential failures. Monitoring also helps in capacity planning, resource allocation, and workload balancing, ensuring efficient resource utilization.

2. Monitoring Tools Overview

2.2 Monitoring Platforms

In addition to Prometheus and Zabbix, there are other monitoring platforms available that could be used for monitoring aspects of Apache CloudStack, some examples:

1. **Nagios:** Nagios is a widely used, open-source monitoring solution known for its robustness and extensibility. With a vast ecosystem of plugins and integrations, Nagios can monitor various aspects of Apache CloudStack. However, its configuration and management can be more complex compared to Prometheus and Zabbix.
2. **Checkmk:** Checkmk is an open-source monitoring solution built on top of Nagios core, focusing on simplicity and ease of use. Its lightweight agent and powerful rule-based configuration system make it well-suited for monitoring Apache CloudStack. Checkmk also provides a user-friendly web interface, making it easier to manage and configure compared to some other monitoring tools.
3. **Datadog:** Datadog is a popular, commercial monitoring platform that offers comprehensive monitoring capabilities, including infrastructure, application, and log monitoring. It supports a wide range of integrations, including Apache CloudStack and various hypervisors. Its user-friendly interface and advanced analytics make it a strong choice, although it may come at a higher cost compared to open-source solutions.
4. **Icinga:** Icinga is an open-source monitoring tool that originated as a fork of Nagios. It offers a more modern interface and improved performance compared to its predecessor. Icinga's modular architecture and extensive plugin library make it suitable for monitoring Apache CloudStack, but it may require more manual configuration compared to Prometheus and Zabbix.
5. **SolarWinds:** SolarWinds is a commercial monitoring platform offering a suite of tools designed for various monitoring needs, including network, server,

application, and cloud monitoring. Its broad range of capabilities and integrations make it suitable for monitoring Apache CloudStack, but the cost and complexity of managing multiple tools may be a drawback for some users.

6. **Graphite:** Graphite is an open-source monitoring tool that focuses primarily on metrics collection, storage, and visualization. It provides real-time insights into system performance using its time-series database, Carbon; Grafana can be used as a web-based visualization tool. While Graphite excels at handling time-series data, it may not provide the same level of alerting and comprehensive monitoring capabilities as Prometheus or Zabbix.
7. **InfluxDB:** InfluxDB is a time-series database designed for storing and processing high volumes of timestamped data, making it well-suited for monitoring purposes. Grafana can be used for graphing.

Ultimately, the choice of monitoring platform depends on factors such as budget, technical expertise, scalability requirements, and specific monitoring needs. Each platform has its strengths and weaknesses, so it's essential to evaluate them carefully and choose the one that best aligns with your organization's objectives and resources.

Why Prometheus and Zabbix Were Chosen

Prometheus and Zabbix were selected as the primary focus of this whitepaper due to their combination of features, scalability, community support, and cost-effectiveness. Key factors include:

1. Open-source and cost-effective compared to commercial alternatives.
2. Scalability and performance for large-scale deployments.
3. Extensive community support and ecosystem of integrations, plugins, and exporters.
4. Ease of use, configuration, and accessibility to users with varying technical expertise.
5. Integration with popular tools, such as Grafana for visualization and alerting.
6. Flexibility and customization for tailored monitoring setups.

These factors make Prometheus and Zabbix well-suited for monitoring Apache CloudStack effectively, while providing a robust, scalable, and cost-effective solution.

2.2 Prometheus

Prometheus is an open-source monitoring and alerting toolkit designed for reliability and scalability. It supports multi-dimensional data collection and querying, allowing users to monitor various aspects of their infrastructure.

2.2.1 Architecture

Prometheus follows a pull-based model, which means it actively scrapes metrics from the target systems or applications at specified intervals. The collected metrics are stored in a time-series database, enabling users to analyse and visualize the data using various tools, such as Grafana. The Prometheus ecosystem also includes Alertmanager, a component responsible for handling alerts and notifications.

2.2.2 Key Components

Prometheus Server: Responsible for collecting and storing metrics in a time-series database.

Exporters: Applications that expose metrics in a format consumable by Prometheus.

Alertmanager: Manages alerts and notifications based on user-defined rules.

Query Language (PromQL): A flexible query language to access and analyze the collected metrics.

2.3 Zabbix

Zabbix is an open-source, enterprise-class monitoring solution for networks, servers, applications, and services. It provides a comprehensive set of features, including data collection, visualization, alerting, and distributed monitoring, to manage complex and large-scale IT environments.

2.3.1 Architecture

Zabbix uses a combination of push and pull mechanisms for data collection. Zabbix agents are installed on the target systems and can either actively send data to the Zabbix server (push) or wait for the server to request data (pull). The collected data is stored in a relational database management system (RDBMS), such as MySQL or PostgreSQL, and can be visualized using the built-in web interface or external tools.

2.3.2 Key Components

- **Zabbix Server**
  The central component responsible for data processing, alerting, and visualization.

- **Zabbix Agents**
  Lightweight processes installed on target systems to collect and transmit data.
- **Zabbix Proxy**
  An optional component that can offload data collection and processing tasks from the Zabbix server, improving scalability.
- **Zabbix Web Interface**
  A web-based user interface for managing and visualizing monitoring data.

3. Comparison of Prometheus and Zabbix

While both Prometheus and Zabbix are powerful monitoring solutions, they have different strengths and weaknesses. Prometheus is known for its scalability, performance, and ease of integration with container-based environments, such as Kubernetes. On the other hand, Zabbix offers a more comprehensive feature set and out-of-the-box support for a wide range of systems and applications. The choice between the two depends on the specific requirements and the existing infrastructure of the organization.

4. Integrating Prometheus with Apache CloudStack

4.1 Installation and Configuration of Prometheus

To start monitoring Apache CloudStack using Prometheus, follow these steps:

- Install Prometheus on a dedicated monitoring server.
- Configure Prometheus by editing the prometheus.yml configuration file. Define the scrape_configs section, specifying the targets for Apache CloudStack.
- Install and configure appropriate exporters for CloudStack and hypervisors. Some commonly used exporters include node_exporter for host-level metrics and cloudstack_exporter for CloudStack-specific metrics.
- Restart Prometheus to apply the configuration changes.

4.2 Configuring Apache CloudStack Metrics

- Install the cloudstack_exporter on the CloudStack management server or a separate monitoring server.
- Configure the cloudstack_exporter with the appropriate API endpoint, API key, and secret key.
- Verify the exporter is functioning correctly by accessing its metrics endpoint.
- Add the cloudstack_exporter as a target in the Prometheus configuration file.

4.3 Configuring Hypervisor Metrics

- Identify the hypervisor-specific exporters for your environment, such as libvirt_exporter for KVM or vmware_exporter for Vmware.
- Install and configure the selected exporter on the hypervisor host or a separate monitoring server.
- Verify the exporter is functioning correctly by accessing its metrics endpoint.
- Add the hypervisor-specific exporter as a target in the Prometheus configuration file.

4.4 Alerting with Prometheus

- Install and configure the Alertmanager component.
- Define alerting rules in the Prometheus configuration file, using PromQL expressions to specify the conditions for triggering alerts.
- Configure alerting channels, such as email, Slack, or PagerDuty, in the Alertmanager configuration file.
- Test the alerting functionality by simulating a failure scenario.

5. Integrating Zabbix with Apache CloudStack

5.1 Installation and Configuration of Zabbix

To start monitoring Apache CloudStack using Zabbix, follow these steps:

- Install Zabbix server, web interface, and database on a dedicated monitoring server.
- Install Zabbix agents on the CloudStack management server and hypervisor hosts.
- Access the Zabbix web interface and complete the initial setup wizard.
- Configure the Zabbix server by editing the zabbix_server.conf configuration file, defining the parameters for the database and other settings.

5.2 Configuring Apache CloudStack Metrics

- Add the CloudStack management server as a host in the Zabbix web interface and link it to the imported monitoring template.
- Configure the Zabbix agent on the CloudStack management server to collect the necessary metrics, such as API response times, System VM status, and resource usage.
- Verify the data collection by checking the latest data and graphs for the CloudStack management server in the Zabbix web interface.

5.3 Configuring Hypervisor Metrics

- Import the hypervisor-specific monitoring template for Zabbix – if any, which includes pre-defined items, triggers, and graphs. These templates can be found in the Zabbix community or created from scratch based on specific requirements.
- Add the hypervisor hosts as hosts in the Zabbix web interface and link them to the imported monitoring template.
- If possible, configure the Zabbix agents on the hypervisor hosts to collect the necessary metrics, such as CPU usage, memory usage, and Instance status.
- Verify the data collection by checking the latest data and graphs for the hypervisor hosts in the Zabbix web interface.

5.4 Alerting with Zabbix

- Configure the desired alerting channels, such as email, SMS, or third-party integrations, in the Zabbix web interface.

- Create actions and triggers based on the predefined items in the monitoring templates. These triggers will define the conditions for generating alerts.
- Test the alerting functionality by simulating a failure scenario and verifying that alerts are generated, and notifications are sent.

6. Best Practices for Monitoring Apache CloudStack

- Define clear monitoring objectives and identify the key performance indicators (KPIs) relevant to your environment.
- Regularly review and update the monitoring configuration to adapt to changes in the infrastructure and evolving requirements.
- Implement a multi-tiered monitoring approach, combining Host-level, Instance-level, and application-level monitoring.
- Establish proactive alerting and notification mechanisms to minimize the impact of issues on service quality.
- Monitor the monitoring infrastructure itself to ensure its reliability and performance.
- Leverage visualization tools, such as Grafana or Zabbix built-in web interface, to create dashboards and reports for easy analysis and sharing of monitoring data.

7. Monitoring Considerations

7.1. Different Hypervisors

While the general principles of monitoring Apache CloudStack remain the same, there may be hypervisor-specific considerations to take into account. Here are some tips for monitoring different hypervisors:

- **KVM**

  Leverage the libvirt integration available in both Prometheus and Zabbix to collect detailed performance metrics and monitor the KVM hypervisor and its virtual machines.

  Tools like virt-top, virt-manager, and collectd are used for additional monitoring and management capabilities.

- **VMware ESXi**

  Utilize the vSphere API to collect metrics and monitor the health of VMware ESXi hypervisor, its hosts, and virtual machines.

  VMware vSphere Management SDK or PowerCLI are used for additional monitoring, automation, and management tasks.

- **Citrix Hypervisor and XCP-ng**

  Employ the XenServer API to gather metrics and monitor the health of Citrix Hypervisor/XCP-ng hypervisor, its Hosts, and Instances.

  Tools like Xen Orchestra, XCP-ng Center, and XenCenter are used for additional monitoring and management capabilities.

7.2 Monitoring Network Performance in Apache CloudStack

Network performance is a critical aspect of cloud infrastructure monitoring. To effectively monitor the network performance of Apache CloudStack, consider the following:

- Collect network performance metrics, such as bandwidth usage, latency, packet loss, and error rates, at the host, Instances, and application levels.
- Use tools like ntopng, Iperf, or SmokePing for specialized network performance monitoring and troubleshooting.

- Set up network performance baselines and thresholds to detect and alert on network anomalies and potential bottlenecks.
- Implement network segmentation and monitoring of inter-Instance traffic for enhanced security and visibility.

7.3 Monitoring Storage Performance in Apache CloudStack

Storage performance is another crucial aspect of cloud infrastructure monitoring. To effectively monitor storage performance in Apache CloudStack, consider the following:

- Collect storage performance metrics, such as IOPS, latency, throughput, and capacity utilization, at the Host, Instance, and application levels.
- Tools like iostat, blktrace, or fio are used for specialized storage performance monitoring and benchmarking.
- Set up storage performance baselines and thresholds to detect and alert on storage anomalies and potential bottlenecks.
- Monitor the health and performance of different storage backends, such as local storage, network-attached storage (NAS), or storage area networks (SAN).

7.4 Monitoring and Tuning Apache CloudStack Database Performance

The performance of the Apache CloudStack database is critical to the overall performance and stability of the cloud infrastructure. To effectively monitor and tune the database performance, consider the following:

- Collect database performance metrics, such as query response times, resource utilization, and transaction rates.
- Use tools like MySQLTuner or Percona Toolkit for specialized database performance monitoring and optimization.
- Set up database performance baselines and thresholds to detect and alert on database anomalies and potential bottlenecks.
- Implement database performance best practices, such as indexing, query optimization, and resource allocation, to ensure optimal performance and scalability.

7.5 Ensuring High Availability and Disaster Recovery for Monitoring Infrastructure

To ensure the high availability and disaster recovery of your monitoring infrastructure, consider implementing the following practices:

- Deploy redundant monitoring servers or clusters, either in an active-active or active-passive configuration, to minimize the impact of a single point of failure.
- Use tools like Pacemaker, Corosync, or Keepalived for monitoring server clustering and failover management.
- Regularly back up monitoring data, configurations, and metadata to a separate storage system or off-site location.
- Test the monitoring infrastructure recovery process periodically to validate its effectiveness and identify potential issues.
- Implement monitoring solutions that support horizontal scaling and load balancing to accommodate growth and distribute the workload across multiple instances.

7.6. Customizing Monitoring Solutions for Apache CloudStack

Both Prometheus and Zabbix offer a high degree of customizability, allowing you to tailor your monitoring solution to the specific needs of your Apache CloudStack and hypervisor environment. Consider the following customization options:

- Create custom monitoring templates, items, and triggers in Zabbix, or custom exporters and alerting rules in Prometheus, to capture unique metrics or events relevant to your environment.
- Develop custom scripts or integrations to collect data from third-party tools, APIs, or log files, and incorporate them into your monitoring solution.
- Use tools like Grafana or the Zabbix web interface to create custom dashboards, visualizations, and reports that provide meaningful insights and cater to the needs of different stakeholders.
- Leverage community resources, such as forums, blogs, and repositories, to find and share best practices, templates, exporters, and integrations related to Apache CloudStack and hypervisor monitoring.

7.7 Monitoring Apache CloudStack in Multi-Cloud and Hybrid Cloud Environments

In multi-cloud and hybrid cloud environments, where Apache CloudStack may be deployed across multiple on-premises and public cloud platforms, effective monitoring requires additional considerations:

- Implement a unified monitoring solution that can integrate with different cloud platforms, such as AWS, Azure, and GCP, as well as on-premises data centres, to provide a holistic view of the infrastructure.
- Use tools like Prometheus Federation or Zabbix Proxies to aggregate and centralize monitoring data from disparate sources and regions.
- Set up monitoring policies, alerting rules, and visualization templates that can accommodate the unique characteristics and requirements of different cloud platforms.
- Monitor and optimize cloud resource usage, cost, and compliance across multiple platforms using tools like CloudHealth, Cloudability, or CloudCheckr.

7.8 Automating Monitoring Configuration Management for Apache CloudStack

As cloud infrastructures grow and evolve, managing the monitoring configuration can become increasingly complex. Automating monitoring configuration management can help streamline the process and minimize human errors:

- Use configuration management tools like Ansible, Puppet, or Chef to automate the installation, configuration, and updates of monitoring components, such as Prometheus, Zabbix, and their respective agents or exporters.
- Store monitoring configuration files, templates, and scripts in a version control system like Git to track changes and collaborate more effectively.
- Implement continuous integration and continuous deployment (CI/CD) pipelines for monitoring configuration updates to ensure consistency and reliability across the infrastructure.
- Use tools like Prometheus' Alertmanager or Zabbix' auto-registration feature to automatically discover and configure new Hosts, Instances, or containers for monitoring as they are deployed in the environment.

7.9 Monitoring Apache CloudStack for Compliance and Security

Monitoring for compliance and security is an essential aspect of maintaining a secure and well-governed cloud infrastructure. To effectively monitor an Apache CloudStack deployment for compliance and security, consider some of the following:

- Implement security monitoring tools, such as intrusion detection systems (IDS), log analysers, and vulnerability scanners, to complement your performance and availability monitoring setup.
- Create custom alerting rules in Prometheus or Zabbix to detect and notify you of potential security incidents, such as unauthorized access, data breaches, or configuration drift.
- Monitor and enforce compliance with industry standards, such as GDPR, HIPAA, or PCI DSS, using tools like AWS Config, Azure Policy, or GCP Cloud Security Command Center.
- Regularly review and update monitoring configurations, policies, and alerting rules to adapt to changes in the threat landscape and regulatory environment.

7.10 Troubleshooting and Debugging Apache CloudStack Monitoring Issues

When monitoring issues arise in your Apache CloudStack and hypervisor environment, it's crucial to have a structured approach to troubleshooting and debugging. Consider the following steps:

- Review the monitoring data, logs, and alerts to identify patterns and pinpoint the root cause of the issue.
- Utilize specialized tools, such as system performance analysers, network analysers, or database profilers, to collect additional data and insights related to the issue.
- Consult the official documentation, community resources, and support channels for guidance on resolving the specific issue.
- Implement corrective actions and preventative measures, such as configuration updates, software patches, or infrastructure changes, to address the issue and prevent it from recurring.

7.11 Ensuring Monitoring Scalability for Apache CloudStack

As your cloud infrastructure grows, it's important to ensure that your monitoring solution can scale accordingly. To ensure monitoring scalability for Apache CloudStack, consider the following best practices:

- Plan for monitoring capacity and resource requirements based on your infrastructure size, growth projections, and monitoring needs.
- Optimize monitoring data retention, storage, and querying performance by leveraging techniques such as data sharing, partitioning, and indexing.
- Implement horizontal scaling and load balancing for your monitoring servers to distribute the workload across multiple Instances and accommodate growth.
- Use tools like Prometheus Federation or Zabbix Proxies to aggregate and centralize monitoring data from multiple sources, regions, or clusters.

7.12 Training and Knowledge Sharing for Apache CloudStack Monitoring

Ensuring that your team has the necessary skills and knowledge to effectively monitor your Apache CloudStack and hypervisor environment is crucial. To promote training and knowledge sharing for monitoring:

- Provide regular training sessions, workshops, and documentation to help team members stay up to date on the latest monitoring tools, techniques, and best practices.
- Encourage collaboration and knowledge sharing among team members through forums, chat platforms, or knowledge management systems.
- Participate in industry conferences, webinars, and community events to learn from peers and experts in the field of cloud infrastructure monitoring.
- Establish a culture of continuous learning and improvement, where team members are encouraged to explore new monitoring tools and techniques and share their findings with the rest of the team.

8. Conclusion and the Future

Monitoring Apache CloudStack is essential for maintaining the performance, reliability, and security of cloud infrastructure. Both Prometheus and Zabbix are powerful and flexible monitoring solutions that can be effectively integrated with CloudStack and hypervisors. The choice between the two depends on the specific needs and preferences of the organization. By following the best practices outlined in this whitepaper and adapting them to your environment, you can achieve efficient and proactive monitoring of your cloud infrastructure.

Things can always be improved, and I hope in the future we will be able to make CloudStack monitoring even better by introducing more Prometheus exports in CloudStack as well as improving Zabbix, such as adding specialised templates and auto-discovery rules.

9. Frequently Asked Questions (FAQ)

Q: Can I use both, Prometheus and Zabbix, to monitor my Apache CloudStack?

A: Yes, you can use both monitoring solutions simultaneously to take advantage of their respective strengths. You may find that certain features or integrations are better suited to one tool over the other, or you may want to use both for redundancy and to cross-verify the monitoring data.

Q: How often should I review and update my monitoring configuration?

A: The frequency of reviewing and updating the monitoring configuration depends on the specific needs and dynamics of your environment. Generally, it's a good practice to review the monitoring setup periodically (e.g., quarterly or semi-annually) and whenever there are significant changes in the infrastructure or business requirements.

Q: What is the recommended approach to monitor containerized workloads in Apache CloudStack?

A: To monitor containerized workloads, you can use a combination of Host-level, Instance-level, and container-level monitoring. For container-level monitoring, you can leverage tools like Prometheus with the cAdvisor exporter or Zabbix with the Docker monitoring module. This will enable you to collect metrics related to container performance, resource usage, and health.

Q: How can I monitor the security of my Apache CloudStack?

A: Monitoring the security of your cloud infrastructure involves collecting and analysing various security-related metrics and events. You can use tools like intrusion detection systems (IDS), log analysers, and vulnerability scanners to complement your monitoring setup. Additionally, you can create custom alerting rules in Prometheus or Zabbix to detect and notify you of potential security incidents.

Q: Are there any limitations of using Prometheus or Zabbix for monitoring Apache CloudStack?

A: Both Prometheus and Zabbix have their own limitations and trade-offs. For example, Prometheus relies on a pull-based model, which might not be ideal for monitoring systems with unreliable network connectivity. Zabbix, on the other hand, has a steeper learning curve due to its comprehensive feature set and can be more resource intensive. It is important to evaluate the specific requirements of your environment and choose the monitoring solution that best meets your needs.

10. Glossary

Apache CloudStack: An open-source cloud computing platform for creating, managing, and deploying Infrastructure as a Service (IaaS) cloud environments.

Hypervisor: A software component that enables the creation and management of virtual machines by abstracting and sharing the underlying hardware resources.

Instance: In the context of Apache CloudStack, Instance refers to a Virtual Machine (VM), which is an emulation of a computer system that provides the functionality of a physical computer.

Prometheus: An open-source monitoring and alerting toolkit designed for reliability and scalability, primarily used for monitoring cloud-native environments and containerized workloads.

Zabbix: A mature, open-source monitoring solution that provides comprehensive monitoring capabilities for networks, servers, applications, and cloud environments.

Exporter: In the context of Prometheus, an exporter is a software component that collects metrics from a specific source and exposes them in a format that Prometheus can scrape and ingest.

Metrics: Quantitative measurements that provide insights into the performance, utilization, and health of systems, applications, and infrastructure components.

Alerts: Notifications triggered by specific conditions or thresholds in the monitoring data, indicating potential issues, anomalies, or bottlenecks that may require attention or action.

Grafana: An open-source, feature-rich data visualization and dashboarding tool that supports multiple data sources, including Prometheus and Zabbix.

High Availability (HA): A design approach that aims to minimize downtime and service disruptions by providing redundancy, failover, and fault tolerance in the infrastructure.

Disaster Recovery (DR): The process of restoring normal operations after a disaster or major incident that has caused significant disruption or loss of data, systems, or infrastructure.

11. Appendix – practical howto on monitoring Apache CloudStack with Zabbix and Prometheus exporters

In this tutorial, we'll set up monitoring Apache CloudStack with a focus on Zabbix and Prometheus exporters.

Monitoring CloudStack per se involves checking for the following:

- **Java processes** running on the management servers as well as the agents, if applicable
- **Java internals** such as HEAP memory etc via the Java Management Extensions (JMX)
- The **DB service**
- **TCP ports** availability:
  - 8080/8443 – UI/API
  - 9090 – CloudStack Management Cluster Interface
  - 8250 – System VM to Management communications

Some or all of the following features can be leveraged to enable a reliable monitoring solution:

- the **Usage** service
- **Graphite** support
- the **Events Framework**
- the **Administrator Email Alerts**
- **Prometheus Exporter**
- 3rd party agents, such as Zabbix or Nagios agents

You should enable at least the email alerts which can be easily done from the Global Settings.

Going forward let's have a look at monitoring a typical CloudStack setup:

1. The management server OS
2. The SQL server
3. CloudStack's JVM stats
4. CloudStack's UI and API
5. CloudStack's Prometheus exporter
6. The hypervisors
7. The Instances

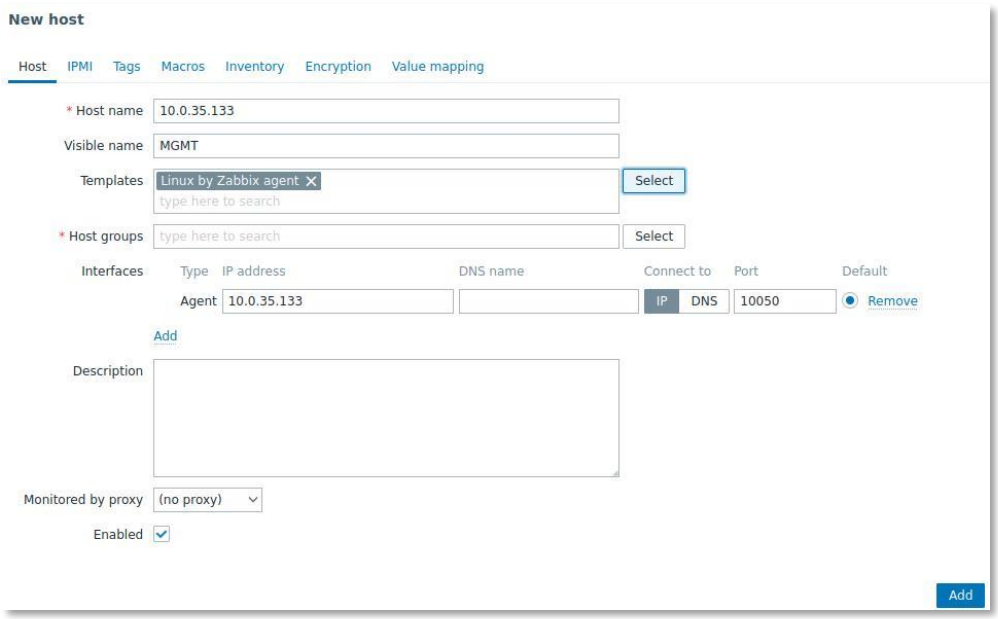
11.1 Monitoring the CloudStack Management Server OS.

By simply installing **zabbix-agent** on the management server and enabling the "Linux by Zabbix agent" template we can out of the box have a generous overview of resources and performance. The agent can report back to Zabbix server a lot of data, such as free and used memory, free and used disk space, hostname and password changes, boot times, CPU and network utilisation and even check for strings in log files if you want to monitor Java exceptions.

The installation per se is trivial, here's an example for EL8:

```
yum install zabbix-agent
sed -i 's/^Server=127.0.0.1/Server=Your Server IP/g' /etc/zabbix/zabbix_agentd.conf
systemctl restart zabbix-agent
```

In the Zabbix UI you can now add this server:



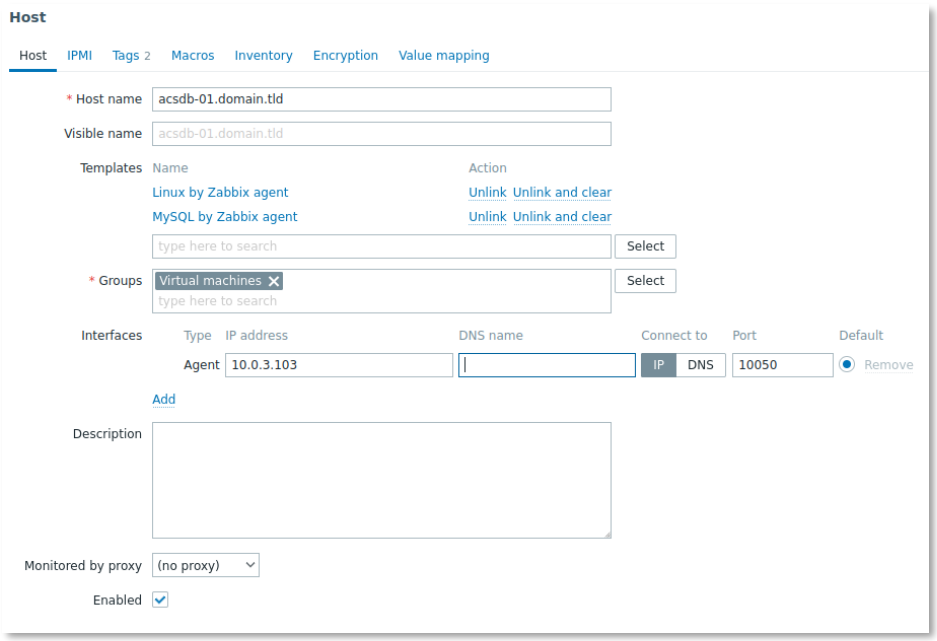
This can be leveraged to also monitor KVM and Citrix Hypervisor/XCP-Ng hypervisors since they come with a typical Linux userland, while for ESXi there is separate functionality available by using the VMWare template.

The following screenshot shows a selection of items from the Linux Zabbix Agent template:

Available memory	Triggers 1	vm.memory.size[available]
Available memory in %		vm.memory.size[pavailable]
Checksum of /etc/passwd	Triggers 1	vfs.file.cksum[/etc/passwd,sha256]
Context switches per second		system.cpu.switches
CPU guest nice time		system.cpu.util[,guest_nice]
CPU guest time		system.cpu.util[,guest]
CPU idle time		system.cpu.util[,idle]
CPU interrupt time		system.cpu.util[,interrupt]
CPU iowait time		system.cpu.util[,iowait]
CPU nice time		system.cpu.util[,nice]
CPU softirq time		system.cpu.util[,softirq]
CPU steal time		system.cpu.util[,steal]
CPU system time		system.cpu.util[,system]
CPU user time		system.cpu.util[,user]
CPU idle time: CPU utilization	Triggers 1	system.cpu.util
Free swap space		system.swap.size[,free]
Free swap space in %	Triggers 1	system.swap.size[,pfree]
Host name of Zabbix agent running		agent.hostname
Interrupts per second		system.cpu.intr
Load average (1m avg)	Triggers 1	system.cpu.load[all,avg1]
Load average (5m avg)	Triggers 1	system.cpu.load[all,avg5]
Load average (15m avg)	Triggers 1	system.cpu.load[all,avg15]
Maximum number of open file descriptors	Triggers 1	kernel.maxfiles
Maximum number of processes	Triggers 2	kernel.maxproc
Available memory in %: Memory utilization	Triggers 1	vm.memory.utilization
Number of CPUs	Triggers 1	system.cpu.num
Number of logged in users		system.users.num
Number of processes	Triggers 1	proc.num

11.2 Monitoring the SQL server

In addition to the standard coverage provided by the Zabbix agent functionality, it is easy to set up MySQL/MariaDB monitoring by employing the "MySQL by Zabbix Agent" template as per the screenshot below:



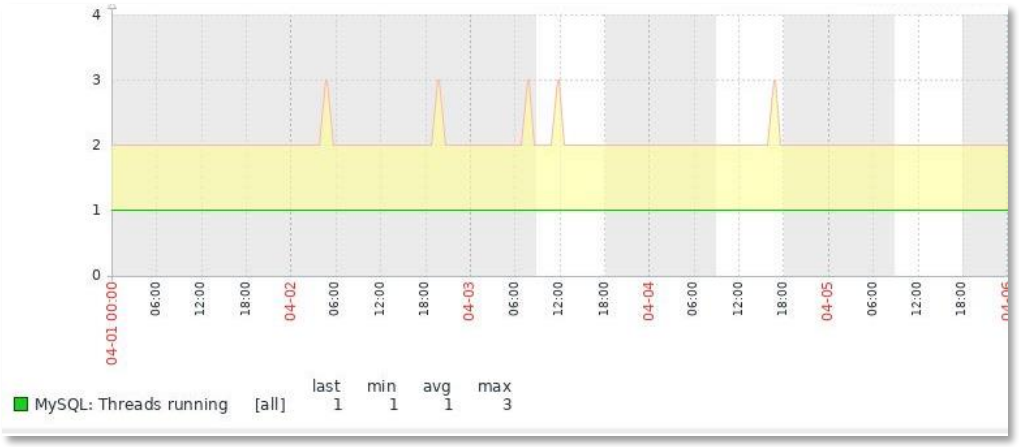
On the Zabbix Agent it will be required to add some extra configuration for this, for example you should create a file */etc/zabbix/zabbix_agentd.d/template_db_mysql.conf* with the following contents:

```
#template_db_mysql.conf created by Zabbix for "Template DB MySQL" and Zabbix 4.2
#For OS Linux: You need create .my.cnf in zabbix-agent home directory (/var/lib/zabbix by default)
#For OS Windows: You need add PATH to mysql and mysqladmin and create my.cnf in %WINDIR%\my.cnf,C:\my.cnf,BASEDIR\my.cnf
https://dev.mysql.com/doc/refman/5.7/en/option-files.html
#The file must have three strings:
#[client]
#user='zbx_monitor'
#password='<password>'
#
UserParameter=mysql.ping[*], mysqladmin -h"$1" -P"$2" ping
UserParameter=mysql.get_status_variables[*], mysql -h"$1" -P"$2" -sNX -e "show global status"
UserParameter=mysql.version[*], mysqladmin -s -h"$1" -P"$2" version
UserParameter=mysql.db.discovery[*], mysql -h"$1" -P"$2" -sN -e "show databases"
UserParameter=mysql.dbsize[*], mysql -h"$1" -P"$2" -sN -e "SELECT COALESCE(SUM(DATA_LENGTH + INDEX_LENGTH),0) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='$3'"
UserParameter=mysql.replication.discovery[*], mysql -h"$1" -P"$2" -sNX -e "show slave status"
UserParameter=mysql.slave_status[*], mysql -h"$1" -P"$2" -sNX -e "show slave status"
```

As per the instructions, don't forget to create a file */var/lib/zabbix/.my.cnf* with the appropriate user and password for MySQL. The zbx_monitor user in this case will need to be granted quite a few options, for example:

```
GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON
*.* TO 'zbx_monitor'@'%';
```

After completing the correct configuration, you should end up with very good MySQL monitoring, including MySQL replication which is often used in CloudStack deployments.

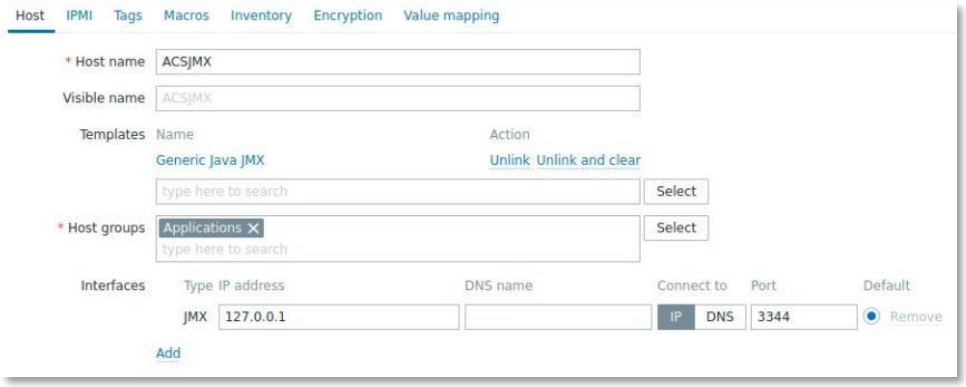


11.3 Monitoring the Java JVM stats

In order to monitor CloudStack's Java internal stats we need to make the Java Management Extensions available. This is done by editing **/etc/default/cloudstack-management** and adding the following parameters:

```
Dcom.sun.management.jmxremote=true
Dcom.sun.management.jmxremote.port=3344
Dcom.sun.management.jmxremote.ssl=false
Dcom.sun.management.jmxremote.authenticate=false
```

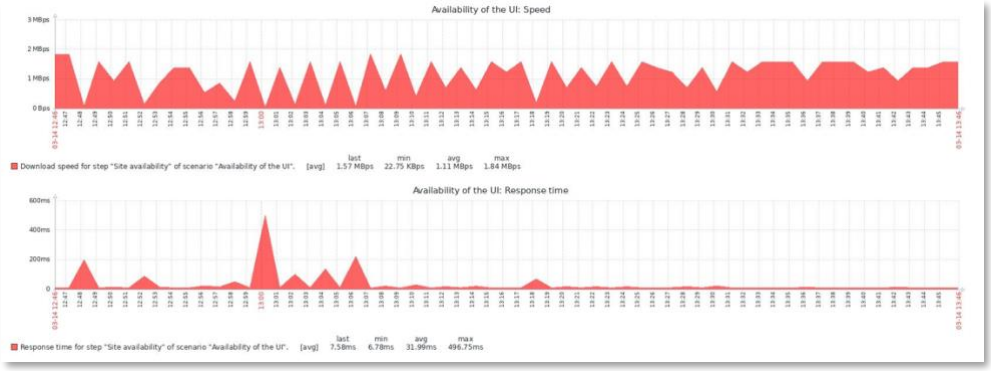
In a production environment, you should opt to use encryption and authentication. Once that is set up you can then add it into Zabbix as per the following example, employing the **Generic Java JMX** template. Replace the JMX IP address with your own.



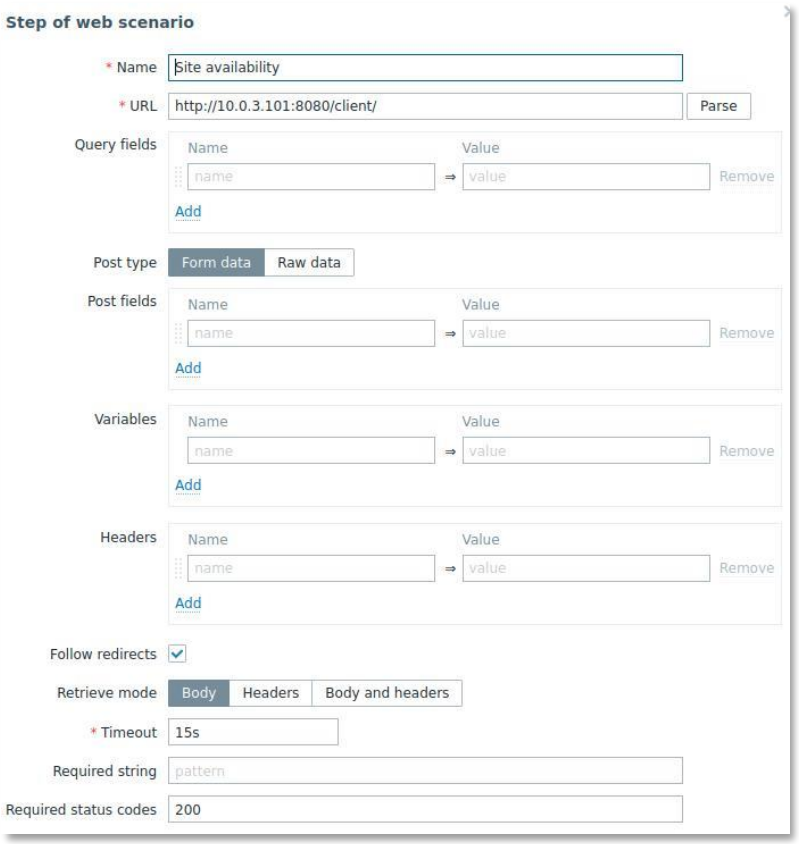
With everything in place, you are now able to check JMX stats such as "Heap memory committed", "Heap memory used" and so on. Data collection may take a while, so give it some time.

11.4 Monitoring the UI & API

The CloudStack UI can be easily monitored using the Web Scenario feature of Zabbix. It will check for response time as well as speed and you can also create a trigger based on this to raise alarms. Here's an example of the graphs.

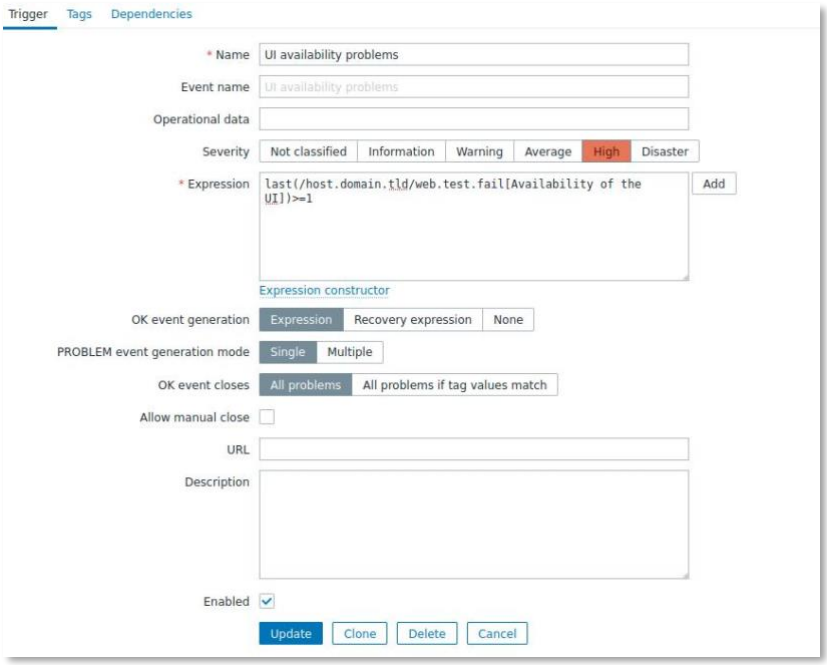


To create such a Web Scenario you need to make sure that you have at least one step in it which will query the right URL and expect a "200 OK" HTTP response:



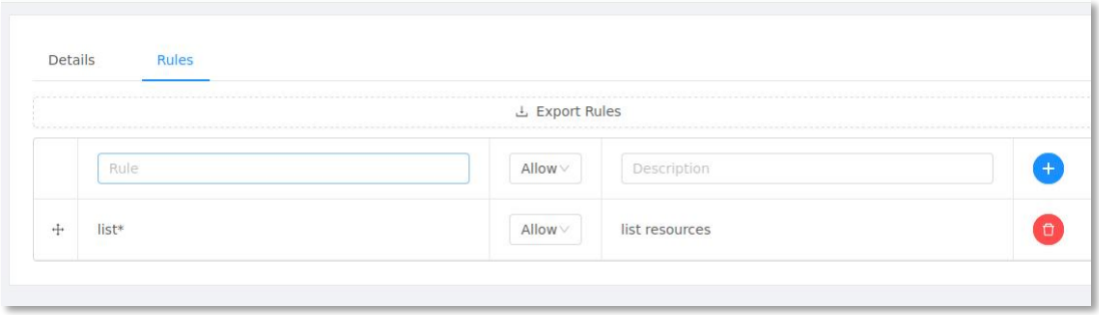
On its own, the Web Scenario will only do periodic checks and graph the values but will not generate any events. To do so you need to create a trigger with the following expression:

last(/host.domain.tld/web.test.fail[Availability of the UI])>=1



Monitoring the CloudStack API can be done in a similar manner, albeit with a few more steps involved because we need to authenticate.

The first step would be to create a separate restricted role and user for monitoring purposes. An example of such a role could be one that only allows "listing" resources. It the UI it would look something like this:

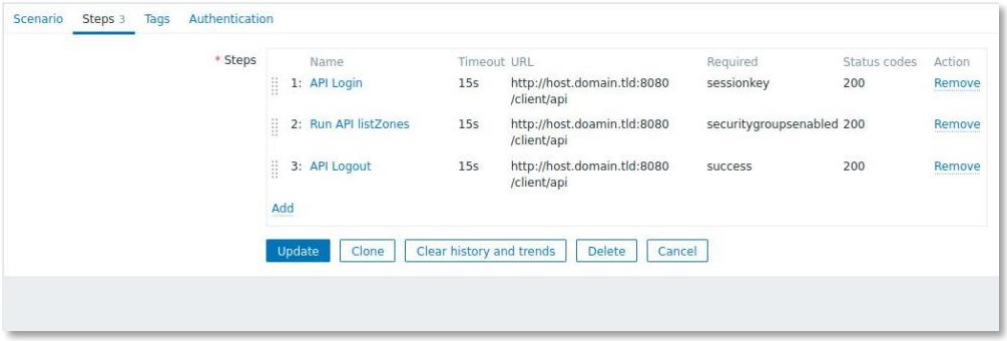


This ensures that the monitoring platform has "read-only" access, so you can be certain any unexpected CloudStack operations had nothing to do with it.

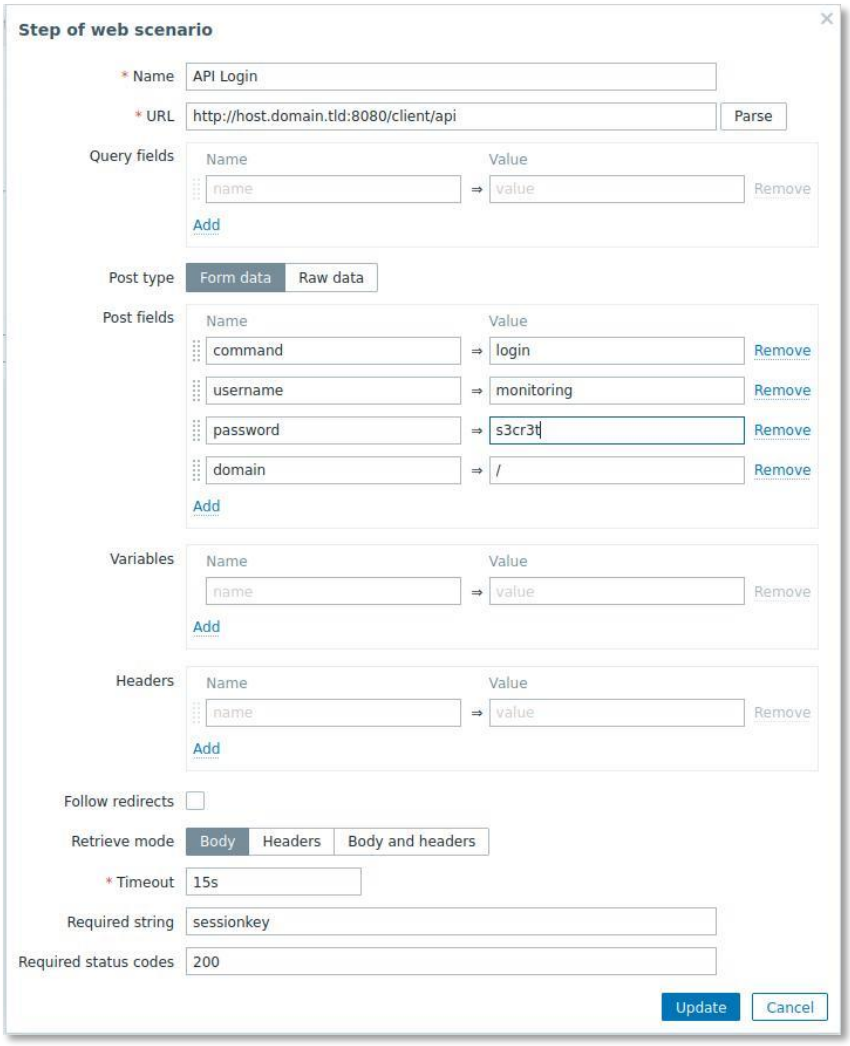
Once a limited role and user have been created, we can proceed with creating an appropriate web scenario.
The procedure is similar to the one above where we monitor the UI, however we need more steps; these are the following:

- Log in to the API

- Run the desired query, in this example we'll be using "listZones"
- Log out so we open new sessions in CloudStack and leave them like that:

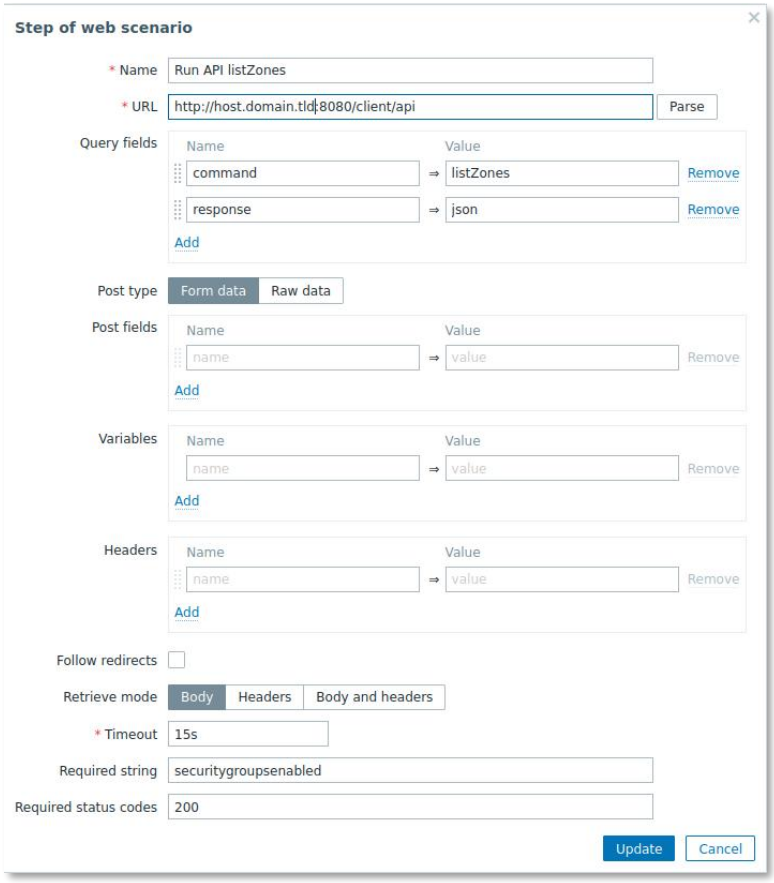


Logging into CloudStack from a Zabbix web scenario is pretty simple, we just need to send a POST request with the required details - the command, the user, the password and the domain. Check the screenshot below for an example. You will notice we expect a 200 "OK" HTTP code and also the string "sessionkey" to match, otherwise, the step is considered failed. There is also a timeout of 15 seconds, should this be exceeded the step will again be considered failed.

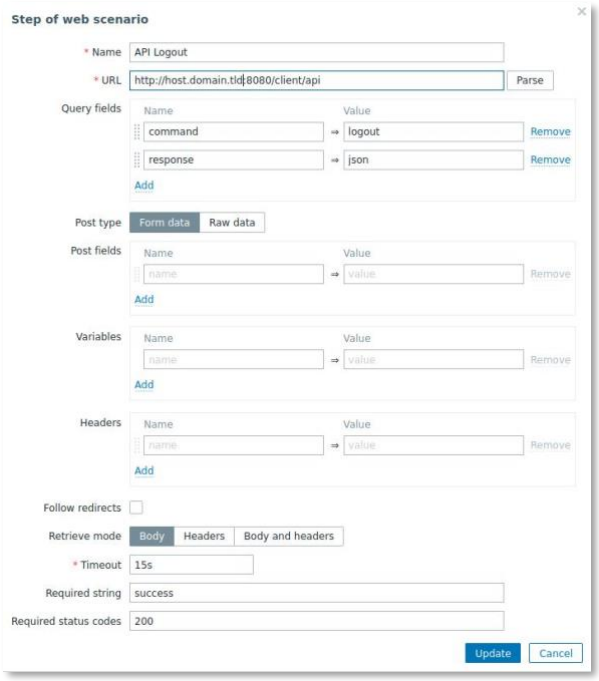


As with the previous step, we must receive a 200 "OK" HTTP code here to consider it valid, also we require the string "securitygroupsenabled" to be present and this all must happen within 15 seconds, else it will be marked as failed. "securitygroupsenabled" is part of a correct response, which is why we request it, even when the zone in question does not use the feature.

The command that we send at this step, as per the screenshot below, is "listZones". It doesn't have to be this specific one, but it's one of those calls which is likely to work in most scenarios.

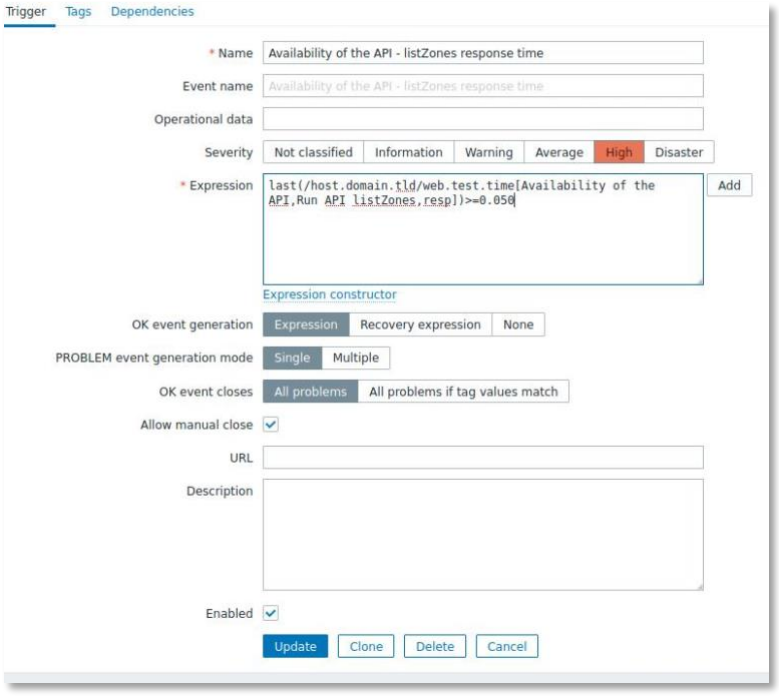


The next step is to log out because it's not ideal to leave so many open sessions in CloudStack. This is easily done by sending the "logout" command. This step also requires the command to complete within 15 seconds, return a 200 "OK" HTTP code and the response needs to contain the string "success".



This web scenario will produce some great graphs, allowing us to monitor and trend the API availability. See the screenshots below.
In order to get actual alerts, we need to create a trigger on this host.
The expression that we could use here is the following, it will trigger should the listZones API request take 50 milliseconds or more, modify to suit your needs, your expected time may be lower or higher.

last(/host.domain.tld/web.test.time[Availability of the API,Run API listZones,resp])>=0.050

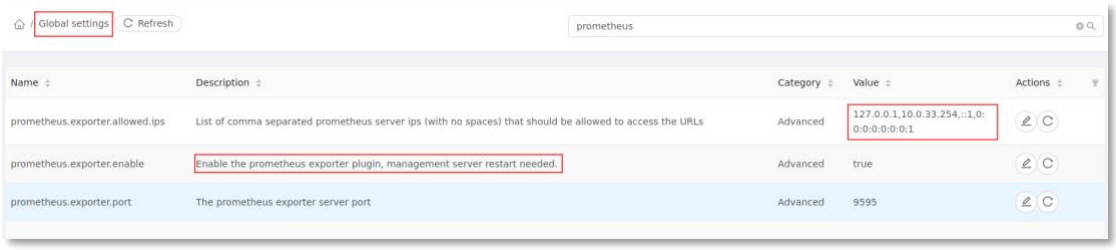


11.5 CloudStack's Prometheus exporter

Enabling CloudStack's Prometheus exporter is an easy job.

You need to go to Global Settings, find the "prometheus.exporter.enable" setting, turn it on and restart the CloudStack management server daemon. If you will connect from an address other than 127.0.0.1 then you should alter the "prometheus.exporter.allowed.ips" accordingly.

Check the following screenshot for a graphical example:

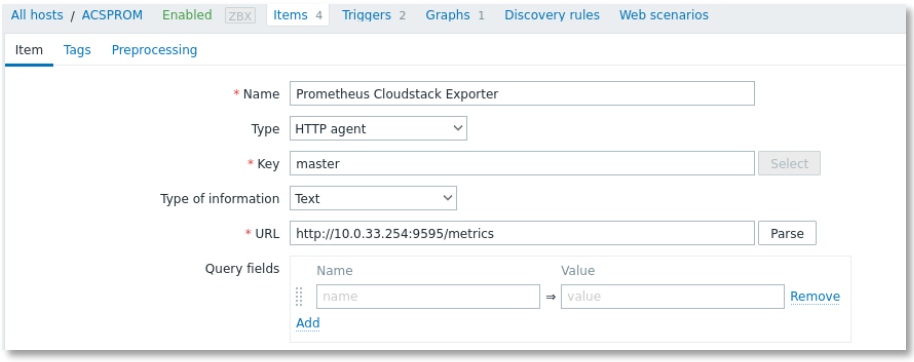


Once this is done the exported data will be available on the CloudStack management server on port 9595, so make sure the firewall allows traffic on this port.

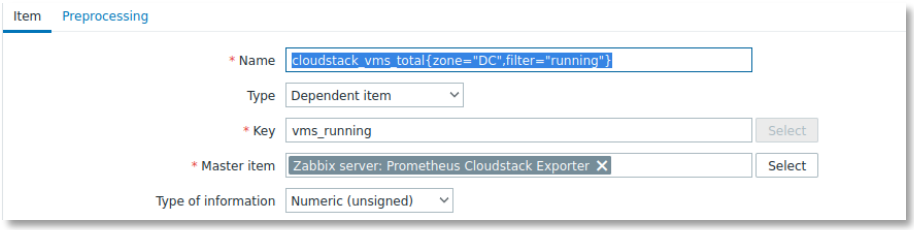
The CloudStack exporter on http://ACS-IP:9595/merics will export the following data. Some of these will be listed multiple times, for multiple items.

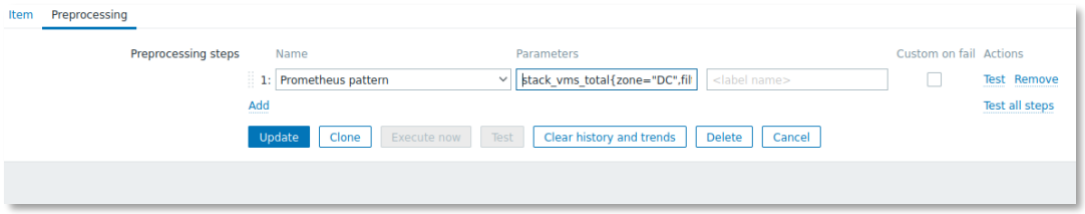
cloudstack_volumes_total
cloudstack_vms_total
cloudstack_vlans_total
cloudstack_storage_pool_gibs_total
cloudstack_shared_network_ips_total
cloudstack_public_ips_total
cloudstack_private_ips_total
cloudstack_host_vms_total
cloudstack_host_vms_cores_total
cloudstack_hosts_total
cloudstack_host_memory_usage_mibs_total
cloudstack_host_is_dedicated
cloudstack_host_cpu_usage_mhz_total
cloudstack_domain_resource_count
cloudstack_domain_limit_memory_mibs_total
cloudstack_domain_limit_cpu_cores_total

To be able to parse all this data in Zabbix we need to make sure there is an agent connection registered for the host, and then add a master item entry. Please see the following screenshot for an example:



The next step is to add a dependent item that will parse exported data. The following screenshots show how to do this for "cloudstack_vms_total", only taking into consideration "running" Instances.





Rinse and repeat for any other items of interest. A Zabbix discovery rule or template could make monitoring all entries much easier.

11.6 Monitoring CloudStack's Hypervisors

- **KVM**

  Monitoring KVM is easily done by installing the Zabbix-agent Linux package available in your distribution of choice. The procedure is similar to monitoring the CloudStack management server since we're essentially dealing with Linux servers, so please refer to point 11.1.

The agent will not return any information about virtualisation; however it will report back on many technical aspects, including CPU usage, network usage, storage information and so on.

To get information on the Instances, we need to rely again on Prometheus. CloudStack-agent does not have an exporter at this time, however, there are 3rd party programs out there that can be used instead. An example would be the libvirt_exporter from here: https://github.com/kumina/libvirt_exporter

- **Citrix Hypervisor and XCP-NG**

  Monitoring an Xen-based hypervisor is very similar to doing it for KVM since we are dealing with a Linux-based userland which uses RPMs.

  As is the case with KVM, we will not be getting any virtualisation-related monitoring out of the box, however, a Prometheus exporter is available, such as:

  https://github.com/janeprather/xapi_exporter or
  https://github.com/MikeDombo/xen-exporter

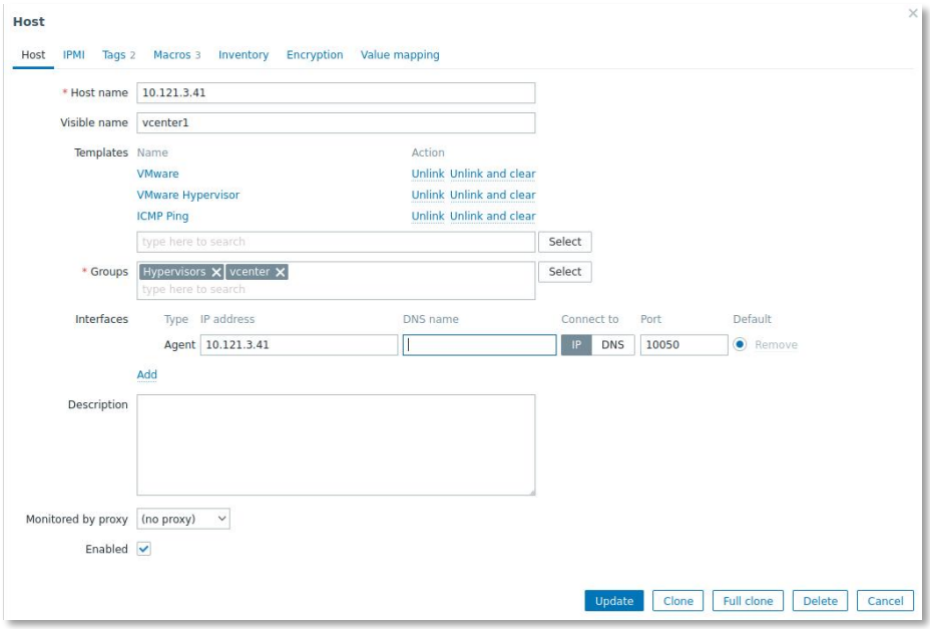
- **VMware**

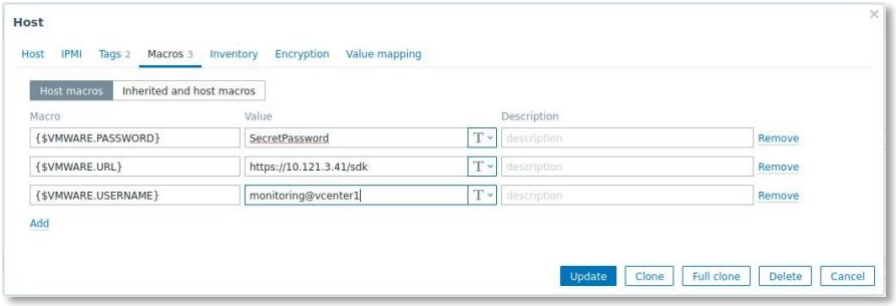
  VMware ESXi does not have a Linux userland we can leverage or the ability to install an agent, however, Zabbix has excellent support for it and is able to monitor it very neatly through the APIs, either each hypervisor separately or via vCenter.

  We need to register the vCenter host as a host and link it to various templates such as "VMware", "VMware Hypervisor" and "ICMP Ping". These templates should give a very reasonable amount of monitoring items and triggers.

  In addition to linking to the right templates, Zabbix will also require we define a few macros containing login information, such as "{$VMWARE.PASSWORD}", "{$VMWARE.URL}" and "{$VMWARE.USERNAME}".

  Refer to the screenshots below for examples:





With this configuration applied, Zabbix will auto-discover pretty much everything and leave you with a reasonably complete monitoring solution.

You will be getting stats and alerts based on data stores, networks, hypervisors and even Instances. See below a few screenshots showcasing some of the items covered for hypervisors and Instances:

Name ▲	Triggers	Key
VMware: Ballooned memory		vmware.hv.memory.size.ballooned[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Bios UUID		vmware.hv.hw.uuid[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Cluster name		vmware.hv.cluster.name[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU cores		vmware.hv.hw.cpu.num[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU frequency		vmware.hv.hw.cpu.freq[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU model		vmware.hv.hw.cpu.model[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU threads		vmware.hv.hw.cpu.threads[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU usage		vmware.hv.cpu.usage[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU usage in percents		vmware.hv.cpu.usage.perf[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: CPU utilization		vmware.hv.cpu.utilization[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Datacenter name		vmware.hv.datacenter.name[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Full name		vmware.hv.fullname[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Get sensors		vmware.hv.sensors.get[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Hypervisor ping	Triggers 1	icmpping[]
VMware: Model		vmware.hv.hw.model[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Number of bytes received		vmware.hv.network.in[{$VMWARE.URL},{$VMWARE.HV.UUID},bps]
VMware: Number of bytes transmitted		vmware.hv.network.out[{$VMWARE.URL},{$VMWARE.HV.UUID},bps]
VMware: Number of guest VMs		vmware.hv.vm.num[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Overall status	Triggers 2	vmware.hv.status[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Power usage		vmware.hv.power[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Power usage maximum allowed		vmware.hv.power[{$VMWARE.URL},{$VMWARE.HV.UUID},max]
VMware: Total memory		vmware.hv.hw.memory[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Uptime	Triggers 1	vmware.hv.uptime[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Used memory		vmware.hv.memory.used[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Vendor		vmware.hv.hw.vendor[{$VMWARE.URL},{$VMWARE.HV.UUID}]
VMware: Version		vmware.hv.version[{$VMWARE.URL},{$VMWARE.HV.UUID}]

Name ▲	Triggers	Key
VMware: Ballooned memory		vmware.vm.memory.size.ballooned[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Cluster name		vmware.vm.cluster.name[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Committed storage space		vmware.vm.storage.committed[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Compressed memory		vmware.vm.memory.size.compressed[{$VMWARE.URL},{$VMWARE.VM.UUID
VMware: CPU latency in percents		vmware.vm.cpu.latency[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: CPU readiness latency in percents		vmware.vm.cpu.readiness[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: CPU ready		vmware.vm.cpu.ready[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: CPU swap-in latency in percents		vmware.vm.cpu.swapwait[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: CPU usage		vmware.vm.cpu.usage[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: CPU usage in percents		vmware.vm.cpu.usage.perf[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Datacenter name		vmware.vm.datacenter.name[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Guest memory swapped		vmware.vm.guest.memory.size.swapped[{$VMWARE.URL},{$VMWARE.VM.U
VMware: Guest memory usage		vmware.vm.memory.size.usage.guest[{$VMWARE.URL},{$VMWARE.VM.UUID
VMware: Host memory consumed		vmware.vm.memory.size.consumed[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Host memory usage		vmware.vm.memory.size.usage.host[{$VMWARE.URL},{$VMWARE.VM.UUID}
VMware: Host memory usage in percents		vmware.vm.memory.usage[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Hypervisor name		vmware.vm.hv.name[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Memory size		vmware.vm.memory.size[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Number of virtual CPUs		vmware.vm.cpu.num[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Power state		vmware.vm.powerstate[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Private memory		vmware.vm.memory.size.private[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Shared memory		vmware.vm.memory.size.shared[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Swapped memory		vmware.vm.memory.size.swapped[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Uncommitted storage space		vmware.vm.storage.uncommitted[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Unshared storage space		vmware.vm.storage.unshared[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Uptime		vmware.vm.uptime[{$VMWARE.URL},{$VMWARE.VM.UUID}]
VMware: Uptime of guest OS	Triggers 1	vmware.vm.guest.osuptime[{$VMWARE.URL},{$VMWARE.VM.UUID}]

11.7 Monitoring CloudStack's Instances

As we have seen above, the VMWare Zabbix templates will automatically discover and monitor certain aspects of an Instance, but what can we do if we want to monitor an Instance more in-depth or if we want to monitor a KVM or Xen Instances which are not monitored out of the box at all?

Well, the answer is that we could always monitor each of these Instances individually. Depending on what is the interest, you could do remote checks - say you want to check for ICMP Ping or HTTP availability - which do not require an agent installation, or you could install, configure, and add the Zabbix agent as part of your Instance deployment procedure, either baked in the template or via user-data scripts. Don't forget, Zabbix agent can be installed in all the popular operating systems, such as Linux, Windows, BSDs etc.

One tricky situation you will encounter however is that in a regular CloudStack Isolated Network the VMs will be unreachable by the Zabbix server, since they are behind a Virtual Router. There are a few ways out of this:

- For each Instance you want to monitor, create a port-forwarding for Zabbix agent's 10050 TCP port in the Isolated Network Firewall. Now, in the Zabbix UI you will need to modify the port from 10050 to the one you have chosen.

- You can install the Zabbix agent in the VM and ask it to run in active mode by specifying ServerActive=ZabbixIP in its configuration. This way the agent will connect to the server rather than the other way around, solving the NAT/VR problem.
- Another solution would be to install a Zabbix active proxy in the same Isolated Network. This proxy would then be able to monitor all the Instances in that network and actively report to the Zabbix server.

The Zabbix proxy could be installed on the VR itself - not recommended due to its transient and complex nature - or in a dedicated Instance.

Depending on requirements and context, one or more of the above solutions could be leveraged to monitor the running Instances in your CloudStack installation.

Need help with **Apache CloudStack?**

Get in Touch

cloudstack.apache.org/

About **Apache CloudStack**

Apache CloudStack is the leading open source cloud orchestration platform, in use by many of the world's largest public and private clouds. It is a multi-hypervisor, multi-tenant, high-availability Infrastructure as a Service cloud management platform. CloudStack is software that provides a cloud orchestration layer, giving automation of the creation, provisioning and configuration of IaaS components.

CloudStack turns an existing virtual infrastructure into a cloud-based infrastructure as a Service (IaaS) platform. The fact CloudStack leverages existing infrastructure means that the cost and time for an organisation to build a multi-tenant IaaS platform is greatly reduced.

info@shapeblue.com

www.shapeblue.com

About **ShapeBlue**

ShapeBlue is the largest independent integrator of CloudStack technologies globally and are specialists in the design and implementation of IaaS cloud infrastructures for both private and public cloud implementations. We combine 100's of person-years of experience in designing and building complex network, storage and compute infrastructures with globally leading skills in Apache CloudStack.