To increase memory in a Linux system, you typically work with swap space, which acts as an overflow for your physical RAM. Here’s a detailed guide on how to check, create, and manage swap space in Linux.

## Step 1: Check Current Swap Space

Before making changes, check your current swap space using the following command:

bash

swapon --show

This command will display the swap partitions and files currently in use along with their sizes.You can also check overall memory usage with:

bash

free -h

## Step 2: Disable Existing Swap (if necessary)

If you need to resize an existing swap file, you must first disable it:

bash

sudo swapoff /path/to/swapfile

Replace /path/to/swapfile with the actual path of your swap file.

## Step 3: Create a New Swap File

1. Create a New Swap File: Use the dd command to create a new larger swap file. For example, to create a 2 GB swap file:

bash

sudo dd if=/dev/zero of=/swapfile bs=1G count=2

This command creates a file named swapfile in the root directory with a size of 2 GB.

1. Set Permissions: Change the permissions of the swap file so that only the root user can read and write to it:

bash

sudo chmod 600 /swapfile

1. Set Up the Swap File: Format the file to be used as swap space:

bash

sudo mkswap /swapfile

1. Enable the New Swap File: Activate the new swap file:

bash

sudo swapon /swapfile

## Step 4: Verify New Swap Space

After enabling the new swap file, verify that it is active and check its size:

bash

swapon --show  
free -h

You should see the new swap space listed and available for use.

## Step 5: Make Changes Permanent

To ensure that your new swap file is used after a reboot, you need to add it to the /etc/fstab file:

1. Open /etc/fstab in a text editor:

bash

sudo nano /etc/fstab

1. Add the following line at the end of the file:

text

/swapfile none swap sw 0 0

1. Save and exit (in nano, press CTRL + X, then Y, and hit Enter).

## Additional Considerations

* Swap Size Recommendations: It's generally recommended that your swap space should be equal to or double your physical RAM if you plan on hibernating your system or if you have less than 2 GB of RAM. For systems with more RAM, having a smaller ratio is typically sufficient.
* Monitoring Swap Usage: Regularly monitor your system's memory and swap usage using commands like free -h, htop, or vmstat.

By following these steps, you can effectively increase memory availability in your Linux system through managing swap space, which can help improve performance during memory-intensive operations.

Install Chocolatey and by using it install vagrant and oracle virtualbox.

To install Chocolatey, Vagrant, and Oracle VirtualBox on a Windows system, follow these steps:

## Install Chocolatey

1. Open an Administrative Shell: You can use either Command Prompt or PowerShell. Make sure to run it as an administrator.
2. Run the Installation Command: Copy and paste the following command into your shell and press Enter:For PowerShell:

powershell

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]::Tls12; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

For Command Prompt:

text

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

1. Verify Installation: After installation, verify that Chocolatey is installed by typing choco in your shell. You should see the version information if it was successful

[1](https://www.liquidweb.com/blog/how-to-install-chocolatey-on-windows/)

[2](https://docs.chocolatey.org/en-us/choco/setup/)

[3](https://chocolatey.org/install)

.

## Install Vagrant

1. Install Vagrant using Chocolatey: Once Chocolatey is installed, you can easily install Vagrant by running the following command in your shell:

text

choco install vagrant

1. Wait for Installation to Complete: Chocolatey will download and install Vagrant automatically.

## Install Oracle VirtualBox

1. Install VirtualBox using Chocolatey: Similarly, to install Oracle VirtualBox, execute the following command:

text

choco install virtualbox

1. Complete the Installation: Allow Chocolatey to handle the installation process.

## Summary of Commands

Here’s a quick summary of the commands you'll need:

text

# Install Chocolatey  
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]::Tls12; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))  
  
# Install Vagrant  
choco install vagrant  
  
# Install VirtualBox  
choco install virtualbox

After completing these steps, both Vagrant and Oracle VirtualBox should be installed on your system and ready for use.

Configure vagrant for virtualbox

To configure Vagrant for VirtualBox to run two instances of PostgreSQL, follow these detailed steps. This setup will allow you to have two separate PostgreSQL databases running on different ports, ideal for development and testing environments.

## Step 1: Install Prerequisites

Ensure that you have the following installed:

* VirtualBox
* Vagrant

If not installed, you can use Chocolatey:

text

choco install virtualbox  
choco install vagrant

## Step 2: Create a Vagrant Project Directory

1. Create a Directory: Open your command line interface and create a new directory for your Vagrant project.

text

mkdir vagrant\_postgres\_instances  
cd vagrant\_postgres\_instances

1. Initialize Vagrant: Run the following command to create a Vagrantfile.

text

vagrant init ubuntu/bionic64

## Step 3: Configure the Vagrantfile

Open the Vagrantfile in a text editor and modify it to set up two PostgreSQL instances:

ruby

Vagrant.configure("2") do |config|  
 config.vm.box = "ubuntu/bionic64"  
  
 *# Forwarded ports for two PostgreSQL instances*  
 config.vm.network "forwarded\_port", guest: 5432, host: 5432 *# Instance 1*  
 config.vm.network "forwarded\_port", guest: 5433, host: 5433 *# Instance 2*  
  
 *# Provisioning script to install PostgreSQL instances*  
 config.vm.provision "shell", inline: <<-SHELL  
 apt-get update  
 apt-get install -y postgresql postgresql-contrib  
  
 # Setup first instance  
 sudo -u postgres pg\_createcluster 12 main --start  
 sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'password1';"  
  
 # Setup second instance (with different port)  
 sudo -u postgres pg\_createcluster 12 secondary --start --port 5433  
 sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'password2';"  
  
 # Allow remote connections for both instances (optional)  
 echo "listen\_addresses = '\*'" >> /etc/postgresql/12/main/postgresql.conf  
 echo "listen\_addresses = '\*'" >> /etc/postgresql/12/secondary/postgresql.conf  
  
 echo "host all all 0.0.0.0/0 md5" >> /etc/postgresql/12/main/pg\_hba.conf  
 echo "host all all 0.0.0.0/0 md5" >> /etc/postgresql/12/secondary/pg\_hba.conf  
  
 systemctl restart postgresql  
 SHELL  
end

## Key Configuration Details:

* Port Forwarding: The first instance listens on port 5432, while the second listens on port 5433.
* Provisioning Script: This script installs PostgreSQL and sets up two clusters (instances) with different configurations.

## Step 4: Start the Vagrant VM

Run the following command to start your Vagrant VM and provision it:

text

vagrant up

This command will create the VM and execute the provisioning script to install and configure both PostgreSQL instances.

## Step 5: Accessing PostgreSQL Instances

1. SSH into the VM:

text

vagrant ssh

1. Accessing First Instance:

bash

sudo -u postgres psql -h localhost -p 5432 -U postgres -W *# Use 'password1'*

1. Accessing Second Instance:

bash

sudo -u postgres psql -h localhost -p 5433 -U postgres -W *# Use 'password2'*

## Step 6: Verify Installation

You can verify that both instances are running by listing databases in each instance:

## For Instance 1:

bash

psql -h localhost -p 5432 -U postgres -W -c '\l'

## For Instance 2:

bash

psql -h localhost -p 5433 -U postgres -W -c '\l'

## Important Notes

* To stop or suspend the VM without losing data, use:
  + vagrant halt to shut down.
  + vagrant suspend to pause.
* To completely delete the VM and all data, use:

text

vagrant destroy -f

This setup allows you to run multiple PostgreSQL instances within a single Vagrant-managed VirtualBox VM, providing a flexible environment for development and testing purposes.

How to use OWASP ZAP and OWASP ASIDE with sonarqube?

To integrate OWASP ZAP and OWASP ASIDE with SonarQube, you can follow these steps for setup and configuration. This will allow you to leverage ZAP's security scanning capabilities and ASIDE's application security information for enhanced vulnerability detection within your SonarQube environment.

## Using OWASP ZAP with SonarQube

## Step 1: Install the ZAP SonarQube Plugin

1. Download the Plugin: You can use either the [pdsoftplan/sonar-zap](https://github.com/pdsoftplan/sonar-zap) or [OtherDevOpsGene/zap-sonar-plugin](https://github.com/OtherDevOpsGene/zap-sonar-plugin). Download the latest JAR file of the plugin.
2. Install the Plugin:
   * Copy the downloaded JAR file to your SonarQube installation directory:

text

[your-sonarqube-installation]/extensions/plugins

* + Restart SonarQube to enable the plugin.

## Step 2: Configure the Plugin

1. Report Path Configuration: You need to specify the path to the ZAP report in your project’s sonar-project.properties file or in your Maven pom.xml file:

text

sonar.zap.reportPath=target/zap-reports/zap-report.html

Adjust the path according to where your ZAP reports are generated.

1. Quality Gate Setup: You can set up quality gates in SonarQube based on the number of alerts reported by ZAP. This allows you to enforce security standards in your projects.

## Step 3: Generate ZAP Reports

* Run a ZAP scan against your application and generate an HTML or XML report. This can be done using ZAP's GUI or its command-line interface.

## Step 4: Execute SonarQube Analysis

* Run SonarQube analysis on your project. The ZAP plugin will parse the report and include security metrics in the analysis results.

## Using OWASP ASIDE with SonarQube

OWASP ASIDE (Application Security Integrated Development Environment) is designed to provide security insights during development. While there is no direct plugin for integrating ASIDE with SonarQube, you can follow these general steps:

## Step 1: Set Up ASIDE

1. Install ASIDE: Follow the instructions on the [OWASP ASIDE GitHub page](https://github.com/OWASP/ASIDE) to install it in your development environment.
2. Configure ASIDE: Set up ASIDE according to your project requirements, ensuring it scans for vulnerabilities during development.

## Step 2: Generate Reports from ASIDE

* After running scans with ASIDE, generate reports that detail identified vulnerabilities and security issues.

## Step 3: Integrate ASIDE Reports into SonarQube

1. Convert Reports: If necessary, convert ASIDE reports into a format compatible with SonarQube (e.g., Generic Issue format).
2. Upload Reports: Use the Generic Issue Import feature in SonarQube to include ASIDE findings in your project analysis:

text

sonar.genericIssues.reportPath=path/to/aside-report.json

1. Run Analysis: Execute a new analysis in SonarQube, which will now include findings from both OWASP ZAP and OWASP ASIDE.

## Conclusion

By following these steps, you can effectively integrate OWASP ZAP and OWASP ASIDE with SonarQube, enhancing your application's security posture through automated vulnerability detection and reporting. This integration supports continuous security practices within your CI/CD pipeline, ensuring that both code quality and security are maintained throughout development.

**Hardening** - Means Patch Installation

To do replication between two VMs PostgreSQL, we need to first empty data folder from secondary PostgreSQL and then configure .conf file of primary database VMs and in pg\_hba.conf [host-based authentication].

To synchronize PostgreSQL databases between different servers, you can follow these detailed steps, utilizing various methods such as replication or data synchronization tools. Here’s a structured approach to achieve effective database sync:

## Step 1: Define Your Synchronization Requirements

* Identify Objectives: Determine whether you need real-time synchronization, periodic updates, or one-time data migration.
* Assess Data Volume: Understand the amount of data to be synchronized and the frequency of changes.

## Step 2: Choose a Synchronization Method

You can choose between several methods based on your requirements:

## 2.1. PostgreSQL Replication

* Physical Replication: This involves creating a replica server that continuously receives data from the primary (master) server.
* Logical Replication: Allows selective replication of specific tables or schemas and can be set up for more flexible use cases.

## 2.2. Data Synchronization Tools

* pgsync: A command-line tool designed for fast synchronization of PostgreSQL databases. It allows syncing specific tables and rows, handles schema differences, and supports parallel transfers

[1](https://github.com/ankane/pgsync)

.

* DBSync or DBConvert: These tools provide GUI-based solutions for syncing databases with options for insert, update, and delete operations

[3](https://dbconvert.com/blog/what-is-database-synchronization/)

.

## Step 3: Set Up PostgreSQL Replication (If Chosen)

## 3.1. Configure the Primary Server

* Edit the postgresql.conf file to enable replication:

text

wal\_level = replica  
max\_wal\_senders = 3

* Configure pg\_hba.conf to allow replication connections:

text

host replication all <replica\_ip>/32 md5

## 3.2. Set Up the Replica Server

* Install PostgreSQL and configure it to follow the primary server.
* Create a base backup from the primary:

bash

pg\_basebackup -h <primary\_ip> -D /var/lib/postgresql/data -U <replication\_user> -P --wal-method=stream

## 3.3. Start the Replica Server

* Create a recovery.conf file in the data directory with:

text

standby\_mode = 'on'  
primary\_conninfo = 'host=<primary\_ip> port=5432 user=<replication\_user> password=<password>'  
trigger\_file = '/tmp/postgresql.trigger'

## Step 4: Using pgsync for Data Synchronization

If you prefer using a tool like pgsync, follow these steps:

## 4.1. Install pgsync

* Install pgsync using RubyGems:

bash

gem install pgsync

## 4.2. Configure pgsync

* Create a .pgsync.yml configuration file to define source and target databases:

text

# .pgsync.yml  
from:   
 database: source\_db  
to:   
 database: target\_db

## 4.3. Sync Data

* Run pgsync commands to sync specific tables:

bash

pgsync table\_name *# Syncs the specified table from source to target.*

* Use options like --preserve or --truncate based on your needs.

## Step 5: Monitor Synchronization Process

* Regularly check logs and status of replication or synchronization processes.
* Use monitoring tools like pgAdmin or custom scripts to ensure data integrity.

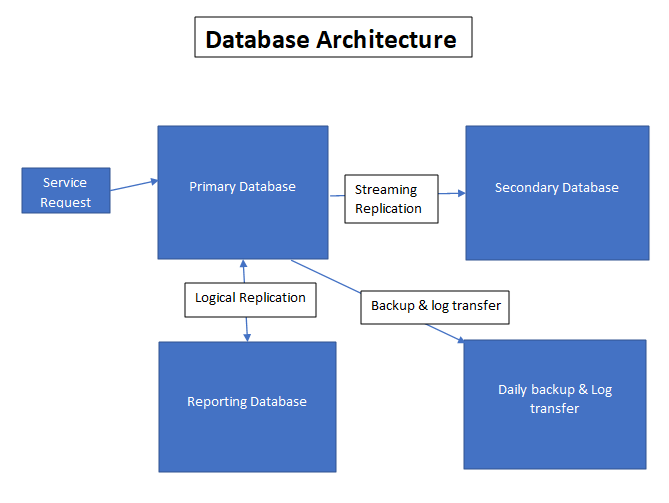
## Step 6: Automate Synchronization (If Applicable)

* For ongoing synchronization, consider setting up cron jobs or using built-in scheduling features in your chosen tool (like ThoughtSpot DataFlow)

## Step 7: Validate Data Consistency

* After synchronization, run queries to compare data between source and target databases to ensure consistency.

By following these steps, you can effectively synchronize PostgreSQL databases between servers, ensuring that your data remains consistent and up-to-date across different environments.



**Database architecture**

1.**Primary Database**: It consist of 500 database of organization. DML and DDL operations are performed here.

Using streaming replication its sends or create copy in Secondary Database.

In this, database can't interact with each other, so by using logical replication its sends copy to Reporting Database.

2.**Secondary Database**: It's also known as Standby Database. It consists copy of Primary Database. Streaming replication is being used to create copy in this Database.

3.**Reporting Database**: It is basically used for reporting purpose. Logical replication is being used to create copy in this Database. Only DML operations is transferred here from Primary database.

DDL operations needs to be carefully performed here as its affect Primary Database.

4.**Daily Backup &Log transfer**: Here daily backup and log transfer are performed through Primary Database.

1. How to setup streaming replication between primary database and secondary [standby] database.

2. How to setup logical replication between primary database and reporting database.

3. How to run script in query tool [received from client via email]

4. backup and log transfer script [via cronjob]

5. streaming replication [async replication, DML and DDL] vs logical replication [sync replication, only DML].

1. postgresql plugins [publication and subscription plugin]

2. pgAdmin plugins used.

3. Where plugin links visible in pgAdmin

4. How they are maintaining sequence between different databases. Which approach they follow.

5. wal file in postgresql.

6. How to re-run replication again.

7. standby pg\_promote utility.

8. logbased shipping replication -- become obsolete

9. streaming replication --- using archieval mechanism [gone obsolete], without using achieve [latest mechanism].