**Azure.sh**

```
#API_KEY="be150cafe27545caae89ad6f2d1d8172"
#ENDPOINT="https://genai-openai-gimsc.openai.azure.com"
#DEPLOYMENT_NAME="IMSC-GenAI-gpt-35-turbo-16k"
#API_VERSION="2023-03-15-preview"

source ./genaiconf.txt

if [ $genai = "true" ]
then
# Input prompt for OpenAI
PROMPT="What is Authorozation object s_tcode mean"
echo "$1"

# Create JSON payload
PAYLOAD=$(cat <<EOF
{
  "messages": [
    {"role": "system", "content": "You are a SAP Expert."},
    {"role": "user", "content": "$1"}
  ],
  "max_tokens": 500
}
EOF
)

# Make the API request
RESPONSE=$(curl -s -X POST "$ENDPOINT/openai/deployments/$DEPLOYMENT_NAME/chat/completions?api-
version=$API_VERSION" \
  -H "Content-Type: application/json" \
  -H "api-key: $API_KEY" \
  -d "$PAYLOAD")

# Extract and print the response
echo "Response from OpenAI:"
echo $RESPONSE
echo $RESPONSE >> output.txt
fi
```

Dates.txt

```
#Uncomment the date format that is being used by the logs.
#DateFormat="%Y-%m-%d"                          #2024-05-23
#DateFormat="%d-%m-%Y"                          #23-05-2024
#DateFormat="%m-%d-%Y"                          #05-23-2024
#DateFormat="%d/%m/%Y"                          #23/05/2024
#DateFormat="%m/%d/%Y"                          #05/23/2024
#DateFormat="%d.%m.%Y"                          #23.05.2024
#DateFormat="%Y-%m-%dT%H:%M:%S"                 #2024-05-23T14:30:00
#DateFormat="%Y-%m-%d %H:%M:%S"                 #2024-05-23 14:30:00
#DateFormat="%d-%m-%Y %H:%M:%S"                 #23-05-2024 14:30:00
#DateFormat="%m/%d/%Y %I:%M:%S %p"              #05/23/2024 02:30:00 PM
#DateFormat="%b %d %H:%M:%S"                    #May 23 14:30:00
#DateFormat="%d/%b/%Y:%H:%M:%S"                 #23/May/2024:14:30:00
#DateFormat="%d-%b-%Y %H:%M:%S"                 #23-May-2024 14:30:00
#DateFormat="%Y/%m/%d %H:%M:%S"                 #2024/05/23 14:30:00
#DateFormat="%a %b %d %H:%M:%S"                 #Thu May 23 14:30:00
#DateFormat="%a %b %e %H:%M:%S"                  #Thu May 2 14:30:00
DateFormat="%a %b %d"                          #Fri May 31
```

EmailConfig.txt

```
SendMail=true
To="test@mail.com, test2@mail.com"
Body="HTML report for the logs"
Subject="HTML Report"
```

fileinfo.txt

```
dev_w*
dev_disp*
messages
```

genaiconf.txt

```
genai="true"
API_KEY="be150cafe27545caae89ad6f2d1d8172"
ENDPOINT="https://genai-openai-gimsc.openai.azure.com"
DEPLOYMENT_NAME="IMSC-GenAI-gpt-35-turbo-16k"
API_VERSION="2023-03-15-preview"
```

hostconf.txt

```
SPLABSAPAPP01:10.68.19.198=SHD-D00
```

hosttemp.txt

```
SPLABSAPAPP01:10.68.19.198=SHD-D00
```

monconfnew.txt

```
PATINFO=/sapmnt/SID/moninfo/moninfowouserinput2.0/patinfo.txt
TRCOUT=/usr/sap/SID/Instance/crdir/
LOGOUT=/usr/sap/SID/Instance/crdir/
FILEINFO=/usr/sap/SID/Instance/crdir/fileinfo.txt
#FILETEMPINFO=/usr/sap/SID/Instance/moninfo/fileinfo.txt
FILETEMPINFO=/sapmnt/SID/moninfo/moninfowouserinput2.0/fileinfo.txt
#FILELOC=/usr/sap/SID/Instance/work/
FILELOC=/var/log/
FILESTORE=/usr/sap/SID/Instance/crdir/filestore.txt
LISTFILE=/usr/sap/SID/Instance/crdir/listfile.txt
HOSTINFO=/sapmnt/SID/moninfo/moninfowouserinput2.0/hostconf.txt
```

output.txt

```
{"choices":[{"finish_reason":"stop","index":0,"message":{"content":"Based on the log entries provided, it
seems that there is a repeated error occurring in the SAP system related to saving buffer assignment. The
specific error message indicates that the name or password is incorrect.\n\nTo resolve this issue, you can
try the following steps:\n\n1. Verify the username and password: Ensure that the username and password used
for the buffer assignment are correct. Check for any typos or changes in the credentials.\n\n2. Check the
user authorization: Confirm that the user has the necessary authorizations to perform buffer assignments.
Review the user's role and profile to ensure they have the required access.\n\n3. Reset the password: If you
suspect that the password may be incorrect, reset the password for the affected user. Follow the standard
password reset procedures in your organization.\n\n4. Review security settings: Check if there have been any
recent changes to the security settings in the SAP system. Ensure that the appropriate security measures are
in place to protect against unauthorized access.\n\n5. Analyze system logs: Look for any additional related
error messages or log entries that provide more details about the issue. Analyzing these logs can help
identify the root cause of the problem.\n\n6. Contact SAP support: If the issue persists or if you are unable
to determine the cause, it is recommended to reach out to SAP support for further assistance. Provide them
with the log files and any other relevant information to help with the investigation and resolution of the
problem.","role":"assistant"}}],"created":1722000486,"id":"chatcmpl-
9pFHKreByGCA6RPXkEs47294S7oPw","model":"gpt-35-turbo-
16k","object":"chat.completion","system_fingerprint":null,"usage":{"completion_tokens":286,"prompt_tokens":22
04,"total_tokens":2490}}
```

patcapnoinputtestver2.0.sh

```
#!/bin/bash
#set -vx
input_conf=$1

if [ -z "$input_conf" ]
then
        echo 'You missed passing config file details'
        exit
fi


######################################################

######################################################
# Creating directory and populating variables required for the
# script execution
#####################################################

hostinfodet(){
thisdate=`date +"%d%m%H%M%S"`
hostip=`hostname -i`
dirinfo=sapconslog
#input_host="/sapmnt/$SAPSYSTEMNAME/moninfo/hosttemp.txt"
input_host="hosttemp.txt"
#`grep -i $hostip "/sapmnt/$SAPSYSTEMNAME/moninfo/hostconf.txt" > $input_host`
`grep -i $hostip "hostconf.txt" > $input_host`
{ while read myline; do
#echo $myline
SID=`echo $myline | awk -F '=' '{print$2}' | cut -d- -f1`
#echo $SID
Instance=`echo $myline | awk -F ':' '{print$2}' | cut -d- -f2`
#echo $Instance
crdir=$dirinfo$SID$Instance$thisdate
`mkdir /usr/sap/$SID/$Instance/$crdir/`
pat_input $SID $Instance $crdir
done } < $input_host
}

patinter(){
#pat_input $SID $Instance $crdir
#echo "Iam into patinter function"
pat_maintcreate
pat_date
```

```
 if [ $yourch == 1 ]
  then
    pat_latest
 fi

 if [ $yourch == 2 ]
  then
    pat_last48hrs
 fi

 if [ $yourch == 3 ]
   then
   mdays=0
   echo "Enter number of days  you want to check "; read mdays < /dev/tty
    pat_ndays $mdays
 fi
}



###############################################################################################
# maintaining the pattern file attributes
###############################################################################################
pat_maintcreate(){
pctr=1
patinfo=""
pat_create
pat_arrcreate
}
###################################################################
#Pattern array  creation
#
##################################################################

pat_create(){
ctr=0
{ while read -r  patline; do
  ((ctr=$ctr+1))
var[ctr]=$patline
done } < $patpth
}

##############################
#Pattern String creation
############################

pat_arrcreate(){
rctr=1
storeawk=""
storeawk="grep -B 3 -A 3"
for x in "${var[@]}"

 do
   #echo "${var[rctr]}"
uppercase=`echo ${var[rctr]}|tr '[:upper:]' '[:lower:]'`
lowercase=`echo ${var[rctr]}|tr '[:lower:]' '[:upper:]'`
storeawk="$storeawk -e $uppercase -e $lowercase"
  ((rctr=$rctr+1))
   #echo $storeawk
  done

}
#######################################################creating date string##################

dateformatsap(){
mval=`date +"%Oe"`
mmonth=`date +"%b"`
mday=`date +"%a"`
now=`date  +"%H"`
#prev2hrs=$((now-2))
#prev1hrs=$((now-1))
prev2hrs=`date  +"%H" -d '-2 hours'`
prev1hrs=`date  +"%H" -d '-1 hours'`
 if [ $mval -le 9 ]
  then
      m2hrsdate=`echo $mday $mmonth" "$mval $prev2hrs`
      m1hrsdate=`echo $mday $mmonth" "$mval $prev1hrs`
      mdate=`echo $mday $mmonth" "$mval $now`
 else
    m2hrsdate=`echo $mday $mmonth $mval $prev2hrs`
    m1hrsdate=`echo $mday $mmonth $mval $prev1hrs`
    mdate=`echo $mday $mmonth $mval $now`
 fi
}
```

```
##########################################################Accepting Date format#################
pat_date(){
dateformatsap
}
###################################################
sendmail(){
        source EmailConfig.txt
        if $SendMail ; then
        echo "Body" | mailx -s "$Subject" -a  "$To"
        echo "Mail Sent"
        fi
}
##################################################

#############################find latest logs and list it last 2 hours ##################
pat_latest(){
`cp $FILETEMPINFO $FILEINFO`
if echo "$DateFormat" | grep "%H:%M:%S"; then
  DateFormat=$(echo "$DateFormat" | sed 's/\:%M:%S//g')
elif echo "$DateFormat" | grep ":%M:%S"; then
  DateFormat=$(echo "$DateFormat" | sed 's/\:%M:%S//g')
fi
mdate="$(date +"$DateFormat")"
m1hrsdate="$(date +"$DateFormat" "-d -1 hours")"
m2hrsdate="$(date +"$DateFormat" "-d -2 hours")"
if [ $dtopch == 02 ]
then
 { while read -r  patline; do
   `find "$FILELOC" -name "$patline" -type f -mmin -120 -exec basename {} \; >> $listfile`
  done } < $FILEINFO

 { while read -r  listline; do
   echo "===========$FILELOC$listline=============" >> $TRCOUT/finout"$mday"latest.txt
   `grep -e "$m2hrsdate" -e "$m1hrsdate" -e "$mdate" "$FILELOC$listline" -C 6 | $storeawk >>
$TRCOUT/finout"$mday"latest.txt`
   echo "<h1>$FILELOC$listline</h1>" >> $TRCOUT/finout"$mday"latest.html
   echo "<pre>" >> $TRCOUT/finout"$mday"latest.html
   `grep -e "$m2hrsdate" -e "$m1hrsdate" -e "$mdate" "$FILELOC$listline" -C 6 | $storeawk >>
$TRCOUT/finout"$mday"latest.html`
   echo "</pre" >> $TRCOUT/finout"$mday"latest.html
 done } < $listfile
fi
source EmailConfig.txt
        if $SendMail ; then
        echo "Body" | mailx -s "$Subject" -a $TRCOUT/finout"$mday"latest.html "$To"
        echo "Mail Sent"
        fi
while read -r pattern; do
       prom="$prom $(grep -i $pattern "$TRCOUT/finout"$mday"latest.txt" | sort -u | sed 's/^[A-
Z]\s*\**\s*//; s/^\s*//; s/"/\\"/g' | sed 's/\\/\\\\/g' | sed 's/"/\\"/g' | sed ':a;N;$!ba;s/\n/\\n/g' | sed
's/\r/\\r/g' | sed 's/\t/\\t/g' | sed "s/'[^']*'//g" )"
done < $patpth
promptOP=$(./azure.sh "$prom")
echo "$promptOP" | awk -v RS='","' '/"content"/ {print $0}' | sed 's/^.*"content":"//' | sed 's/\\n/\n/g' >
$TRCOUT/finout"$mday"ai.txt
./snow.sh ""$TRCOUT"finout"$mday"latest.txt" ""$TRCOUT"finout"$mday"ai.txt" > /dev/null 2>&1
}
#################################################Find last 48 hours logs###############
pat_last48hrs(){
`cp $FILETEMPINFO $FILEINFO`
echo $dtopch

if echo "$DateFormat" | grep "%H:%M:%S"; then
  DateFormat=$(echo "$DateFormat" | sed 's/\%H:%M:%S//g')
elif echo "$DateFormat" | grep "%I:%M:%S %p"; then
  DateFormat=$(echo "$DateFormat" | sed 's/\%I:%M:%S %p//g')
fi
mdate="$(date +"$DateFormat")"
m48date="$(date +"$DateFormat" "-d -1 days")"
if [ $dtopch == 02 ]
then
  { while read -r  patline; do
      `find "$FILELOC" -name "$patline" -type f -mtime -2 -exec basename {} \; >> $listfile`
       done } < $FILEINFO
  { while read -r  listline; do
      echo "===========$FILELOC$listline=============" >> $TRCOUT/finout"$mday"last48hrs.txt
      `grep -e "$m48date" -e "$mdate" "$FILELOC$listline" -C 6 | $storeawk >>
$TRCOUT/finout"$mday"last48hrs.txt`
       echo "<h1>$FILELOC$listline</h1>" >> $TRCOUT/finout"$mday"last48hrs.html
       echo "<pre>" >> $TRCOUT/finout"$mday"last48hrs.html
      `grep -e "$m48date" -e "$mdate" "$FILELOC$listline" -C 6 | $storeawk >>
$TRCOUT/finout"$mday"last48hrs.html`
       echo "</pre>" >> $TRCOUT/finout"$mday"last48hrs.html
       done } < $listfile
fi
source EmailConfig.txt
        if $SendMail ; then
        echo "Body" | mailx -s "$Subject" -a $TRCOUT/finout"$mday"last48hrs.html "$To"
        echo "Mail Sent"
        fi
while read -r pattern; do
        prom="$prom $(grep -i $pattern "$TRCOUT/finout"$mday"last48hrs.txt" | sort -u | sed 's/^[A-
Z]\s*\**\s*//; s/^\s*//; s/"/\\"/g' | sed 's/\\/\\\\/g' | sed 's/"/\\"/g' | sed ':a;N;$!ba;s/\n/\\n/g' | sed
's/\r/\\r/g' | sed 's/\t/\\t/g' | sed "s/'[^']*'//g" )"
done < $patpth
promptOP=$(./azure.sh "$prom")
echo "$promptOP" | awk -v RS='","' '/"content"/ {print $0}' | sed 's/^.*"content":"//' | sed 's/\\n/\n/g' >
$TRCOUT/finout"$mday"ai.txt
./snow.sh ""$TRCOUT"finout"$mday"last48hrs.txt" ""$TRCOUT"finout"$mday"ai.txt" > /dev/null 2>&1
}
########################################number of days option ###########################################

pat_ndays(){
ndays=$1
`cp $FILETEMPINFO $FILEINFO`
#echo "${DateFormat}"
if echo "$DateFormat" | grep "%H:%M:%S"; then
#      echo "in the loop"
  DateFormat=$(echo "$DateFormat" | sed 's/\%H:%M:%S//g')
elif echo "$DateFormat" | grep "%I:%M:%S %p"; then
  DateFormat=$(echo "$DateFormat" | sed 's/\%I:%M:%S %p//g')
fi
echo "$DateFormat"
if [ $dtopch == 02 ]
then
  { while read -r  patline; do
      `find "$FILELOC" -name "$patline" -type f -mtime -"$ndays" -exec basename {} \; >> $listfile`
       done } < $FILEINFO
   { while read -r  listline; do
        for (( i=0; i<$ndays; i++)); do
        ndate=$(date +"$DateFormat" "-d -$i days")
 echo "===========$FILELOC$listline=============" >> $TRCOUT/finout"$mday"last"$ndays"days.txt
       `grep -e "$ndate" "$FILELOC$listline" -C 6 | $storeawk >> $TRCOUT/finout"$mday"last"$ndays"days.txt`
       echo "<h1>$FILELOC$listline</h1>" >> $TRCOUT/finout"$mday"last"$ndays"days.html
echo "<pre>" >> $TRCOUT/finout"$mday"last"$ndays"days.html
       `grep -e "$ndate" "$FILELOC$listline" -C 6 | $storeawk >> $TRCOUT/finout"$mday"last"$ndays"days.html`
       echo "</pre>" >> $TRCOUT/finout"$mday"last"$ndays"days.html
       done
       done } < $listfile
fi
source EmailConfig.txt
        if $SendMail ; then
        echo "Body" | mailx -s "$Subject" -a $TRCOUT/finout"$mday"last"$ndays"days.html "$To"
        echo "Mail Sent"
        fi
#while read -r pattern; do
#        prom="$prom $(grep -i $pattern "$TRCOUT/finout"$mday"last"$ndays"days.txt" | sort -u | sed 's/^[A-
Z]\s*\**\s*//; s/^\s*//; s/"/\\"/g' | sed 's/\\/\\\\/g' | sed 's/"/\\"/g' | sed ':a;N;$!ba;s/\n/\\n/g' | sed
's/\r/\\r/g' | sed 's/\t/\\t/g' | sed "s/'[^']*'//g" )"
#done < $patpth
#promptOP=$(./azure.sh "$prom")
#echo "$promptOP" | awk -v RS='","' '/"content"/ {print $0}' | sed 's/^.*"content":"//' | sed 's/\\n/\n/g' >
$TRCOUT/finout"$mday"ai.txt
#./snow.sh ""$TRCOUT"finout"$mday"last"$ndays"days.txt" ""$TRCOUT"finout"$mday"ai.txt" > /dev/null 2>&1
}
################## main function#########################################################
pat_main(){
source ./Dates.txt
dtopch=02
while :
 do
 clear
 echo "---------------------------------------------"
 echo " * * * * * * * Main Menu * * * * * * * * * * "
 echo "---------------------------------------------"
 echo "[1] Execute Pattern Selection for latest Application logs"
 echo "[2] Execute Pattern Selection for 48 hours logs ie., (Application/OS) logs"
 echo "[3] Execute Pattern Selection for time frame Application  logs"
 echo "[4] Exit/stop"
 echo "---------------------------------------------"
 echo -n "Enter your menu choice [1-4]:"
 read yourch
 case $yourch in
 1) echo "Pattern Latest:" ; hostinfodet;; #pat_latest;;
 2) echo "Pattern Last 2 days:" ; hostinfodet;; #pat_last48hrs;;
 3) echo "Pattern Last "N" days:" ; hostinfodet;; # pat_ndays;;
 4) exit 0 ;;
 *) echo "Opps!!! Please select choice [1-4]";
 echo "Press a key. . ." ; read ;;
esac
done

}
#####################

###################Populating Variable from monconf file####################
pat_input(){
#echo "Reading input configuration file $input_conf"
SID=$1
Instance=$2
crdir=$3
{ while read myline; do
object=`echo $myline | cut -d= -f1`
case $object in
PATINFO)
patpth=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g" | sed -e
"s/crdir/$crdir/g"`
#echo $patpth
;;
TRCOUT)
TRCOUT=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g" | sed -e
"s/crdir/$crdir/g"`
;;
LOGOUT)
LOGOUT=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g" | sed -e
"s/crdir/$crdir/g"`
;;
FILEINFO)
FILEINFO=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g" | sed -e
"s/crdir/$crdir/g"`
;;
FILETEMPINFO)
FILETEMPINFO=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g"`
;;
FILESTORE)
FILESTORE=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g" | sed -e
"s/crdir/$crdir/g"`
;;
FILELOC)
FILELOC=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g"`
;;
LISTFILE)
listfile=`echo $myline | cut -d= -f2 | sed -e "s/SID/$SID/g" |  sed -e "s/Instance/$Instance/g" | sed -e
"s/crdir/$crdir/g"`
;;
esac
done } < $input_conf

patinter
}
######################################
#calling functions
pat_main
###################################
```

patinfo.txt

```
ERROR
INFO
```

snow.sh

```
#!/bin/bash
#set -vx
response=$(curl -q "https://dev191809.service-now.com/api/now/table/incident" \
--request POST \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{\"short_description\":\"This is demo\",\"priority\":\"3\"}" \
--user 'akash':'Lenovo@0727')

sys_id=`echo "$response" | sed -n 's/.*"sys_id":"\([^"]*\)".*/\1/p'`
send_attachment(){
curl -q "https://dev191809.service-
now.com/api/now/attachment/file?table_name=incident&table_sys_id="$sys_id"&file_name="$2"" \
        --request POST \
        --header "Content-Type: application/json" \
        --user "akash":"Lenovo@0727" \
        --data-binary @$1
}
send_attachment $1 "Raw"
send_attachment $2 "GenAI"
```